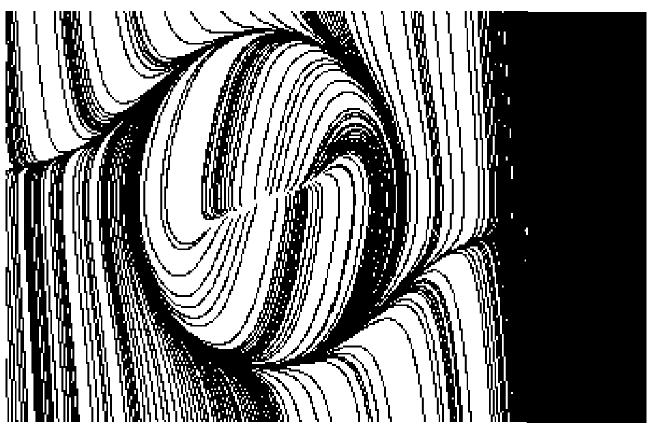
Calculus According to the Real World

Alain Schremmer

Calculus According to the Real World

For People Who Think Textbooks Ought To Make Sense.

Volume 1 Polynomial And Rational Functions



FreeMathTexts.org

Version 1.0 — Tuesday 25th August, 2020

To Françoise.

A mathematician according to mathematicians.

Contents

The pure mathematician's view is, It's not what you know, it's what you can prove. The difficulty with this view is that it is very hard to prove something before you know what you need to prove.

rigorous

Jacob Rubinstein[1]

Preface

This is for the prospective *reader* because, before anything else, they should be made aware of what it is that they may be about to get into. Indeed, in many deep ways, this text is *truly* very different from the usual CALCULUS textbooks.

The usual preface is for convincing teachers *that the book is what* they *want for that class* they *are going to teach next semester.*

1 Language.

2 Rigor.

3 Exercises.

4 Proof/Belief.

The first thing is that this is *not* a **rigorous** text. One reason is that the CALCULUS is extraordinarily difficult to present rigorously (https://en.wikipedia.org/wiki/Calculus#Foundations). For instance, it was only in 1950 that "Delta functions" (https://en.wikipedia.org/wiki/Dirac_delta_function) were made rigorous by Laurent Schwartz—for which he was awarded the Fields Medal[2]

In fact, rigorous presentations go under the name of ADVANCED CALCULUS or REAL ANALYSIS and, for what it's worth, most CALCULUS textbooks just skip the many long, hard parts of ADVANCED CALCULUS.

[1]Bulletin of the American Mathematical Society, Volume 55, Number 1, January 2018, Pages 123-129 (http://dx.doi.org/10.1090/bull/1581)

[2]One of the very highest honors for a *mathematician*. (https://en.wikipedia.org/wiki/Fields_Medal)

Never mind that those who have to pay for the book have no say in choosing ~~the book.~~

free

different

informal

that downloading the pdf is free and that they can print it freely.

=======OK SO FAR=========

This text is for "*just plain folks*" who want to *learn* CALCULUS and it's **free**.

—So, what's this "Not a Preface" all about?

—Mostly a bit of advice on how to use this text.

—Ok, let's have it.

—One way this book is **different** is because it was designed to be read *onscreen* rather than on paper.

—What's the point?

—When we read a scientific text, to really make *sense* of what we read we need to remember the *exact* meaning of each and every word. Which is impossible. Which is why all scientific texts have an *index* to let you find *where* each word, in bold black, was explained. But in *this* text, *onscreen*, just *clicking* on any word in red-black will also get you there.

—I guess. So what's the advice?

—Start at the beginning. Don't skip. Don't go ahead until things *make sense*. Don't even try to remember what a word means, just *click*.

—You said "one way this text is different", what about the other ways?

—

—Because, in contrast with most textbooks which present the CALCULUS from the *mathematician*'s point of view, *this* text aims at the **informal** CALCULUS that *physicists*, *chemists*, *biologists* (https://en.wikipedia.org/wiki/Hard_and_soft_science), and *engineers* have been using for a very long time—and are still using.

In particular, but most importantly, "infinitesimals" were routinely used informally from 1684 on by *physicists*—as well as by *mathematicians*—even though it was realized almost from the start that "infinitesimals" were not rigorous (https://en.wikipedia.org/wiki/Non-standard_analysis). And when, some two centuries later, "limits" were finally made rigorous and most *mathematicians* stopped using "infinitesimals" in favor of "limits", *physicists*, and for a long time even *differential geometers*, kept using "infinitesimals" because they are being closer to the real world (https://en.wikipedia.org/wiki/Calculus#Limits_and_infinitesimals).

In any case, in 1961, Abraham Robinson, three years over the age limit for the Fields Medal, made "infinitesimals" rigorous (https://en.wikipedia.org/wiki/Abraham_Robinson). In spite of which, most textbooks still avoid "infinitesimals" like the plague!

Surprise, surprise!

Yet, as Vladimir Arnold (https://en.wikipedia.org/wiki/Vladimir_Arnold) wrote in 1990: "*Nowadays, when teaching analysis, it is not very* *popular to talk about infinitesimal quantities. Consequently present-day students are not fully in command of this language. Nevertheless, it is still necessary to have command of it.*" (https://en.wikipedia.org/wiki/Infinitesimal)

All this to say that, if this text doesn't follow current fashions, it is nevertheless *rooted* in rigorous mathematics.

—Whew! All this to say just that! That was dense. Any other reason why I should buy *your* book?

—Remember, you can download this text for free and print it if you want. So, just keep on reading and make up *your* mind yourself

—That it?

—No. Another way this text is free is that it is *open source*. So, after you got something you first had trouble with, after you got it your way, you can put *that* way on http://freemathtexts.org/ to help *others*. Another way it's different.

What is important is the **real world**, that is physics, but it can be explained only in mathematical terms.

Dennis Serre[1]

real world
number
set
given
information
describe
specify

Chapter 1

Numbers

The point of this first chapter is to discuss aspects of **numbers** usually not given much attention in ARITHMETIC textbooks but which are at the heart of their relationship to the real world and therefore most relevant to the CALCULUS ACCORDING TO THE REAL WORLD.

In other words, nowhere near the obligatory "Review of things you oughtn't to have forgotten" in standard textbooks.

1 Numbers for what?

There are many different **sets** of numbers, each used for many different purposes, but "the rest of us" **give** numbers as **information** to **describe** what we *have* or to **specify** what we *want*. More precisely, in the real world, depending on:

[0]Bulletin of the AMS, Vol 47 Number 1 Pages 139-144

way
magnitude
count
plain whole number
counting number
natural number
positive integer

A. What kind of real world **object** we want to describe or specify namely:

- A **collection** of **items** that we can deal with *one at a time*,

or

- An **amount** of **stuff** that we can deal with *only in bulk*,

and

B. What kind of information we want to give about the object namely:

- The **magnitude** of the object,

or

- The magnitude of the object *together with* the **way** (as in "two-way street") the object goes.

we only use numbers from the following four sets of numbers:

	Collections of items	Amounts of stuff
Magnitude	Plain *whole* numbers	Plain *decimal* numbers
Magnitude *and* Way	Signed *whole* numbers	Signed *decimal* numbers

Scientists other than physicists just say "quantity".

LANGUAGE 1.1 Amount is what *physicists* call "physical quantity".

LANGUAGE 1.2 Way is *not* a very good word but neither is "direction" unless we are willing to say "one of *two* opposite directions".

2 Plain Whole Numbers

Because we can deal with items *one at a time*, both describing and specifying *how many* items there are in a collection are easy: we just **count** the items. Then, *how many* items are in the collection will be given by a **plain whole number**.

.

EXAMPLE 1.1. Apples are *items*. (We can eat apples one at a time.) To say how many  are in the collection  we *count* them that is we point successively at each  while singsonging "one, two, three".

At least, "counting" reminds us of how we get them but "natural"?

LANGUAGE 1.3 Plain whole numbers are also called **counting numbers** and **natural numbers** (https://en.wikipedia.org/wiki/Natural_number).

LANGUAGE 1.4 Plain whole numbers are also called **positive integers** (https://en.wikipedia.org/wiki/Integer)

unit
plain decimal number
"Positive integer" *makes sense but only to* ...teachers *since only* they *already know what* integers *are.*

The CALCULUS, though, is *not* concerned with collections of items and only with amounts of stuff and so *we* will use *whole* numbers only occasionally, mostly as a backdrop for *decimal* numbers.

3 Plain Decimal Numbers

Because we can only deal with stuff *in bulk*, both describing and specifying *how much* stuff there is in an amount are quite a bit more complicated than for collections of items. There are two complications.

1. Units. The first complication is that before we can describe or specify an amount of stuff we must decide on a **unit** amount of that stuff. Indeed, "*The Weights and Measures Division promotes uniformity in U.S. weights and measures laws, regulations, and standards to achieve equity between buyers and sellers in the marketplace.*" (https://www.usa.gov/federal-agencies/weights-and-measures-division)

Then, *how much* stuff is in an amount will be given by a **plain decimal number** of units of that stuff.

EXAMPLE 1.2. Milk is *stuff* that we drink and before we can *describe* or *specify* how much milk we must decide on a *unit* of milk, for instance liters of milk. Then, for instance, *how much* milk could be 6.4 liters of milk.

To help remind ourselves that we are talking about plain *decimal* numbers rather than plain *whole* numbers,

AGREEMENT 1.1 The decimal point will *never* go without saying in this text.

EXAMPLE 1.3. We will always distinguish the plain *decimal* number $27.$ which we would give to describe of specify an amount of *stuff* from the plain *whole* number 27 which we would give to describe or specify a collection of *items*.

measure
uncertainty

2. To err is human. The second complication is a lot harder to deal with because for an amount of stuff, specifying how much we *want* and describing how much we *have* involve different issues.

a. To *describe* an amount of stuff, the complication is that we have to **measure** this amount of stuff so that there will always be some **uncertainty** about the measured plain decimal number because of things such as the *quality* of the equipment used to measure the amount, the *ability* of the person doing the measurement, etc.

EXAMPLE 1.4. We cannot really say "we *have* 2.3 quarts of milk" because what we really have depends on the care with which the milk was measured. The *uncertainty* may of course be too small to matter . . . but then may not.

As Timothy Gowers, Fields Medal 1998, put it (6^{th} paragraph of https://www.dpmms.cam.ac.uk/~wtg10/continuity.html.), "*a measurement of a physical quantity will not be an exactly accurate infinite decimal. Rather, it will usually be given in the form of a finite decimal together with some error estimate:* $x = 3.14 \pm 0.02$ *or something like that.*"[2] [Where 3.14 ± 0.02 is to be read as $3.14\pm$ some error smaller than 0.02]

b. To *specify* an amount of stuff, the complication then is that while, in the case of a collection of items

- the plain whole number *given* to *specify* how many items we *want*,

will *never* **differ** from

- the plain whole number *counted* to *describe* how many items we *get*

in the case of an amount of stuff,

- the plain decimal number *given* to *specify* how much stuff we *want*.

will *always* differ by some **plain error** from

- the plain decimal number *measured* to *describe* how much stuff we *get*

In other words:

NOTE 1.1 A plain decimal number *by itself* can never specify an amount of stuff.

[2] At *this* time most of Gowers' paper will be much too hard to read but even a cursory glance will show that our concern with the real world is quite the same as his.

significant
small relative
tolerance
specification

EXAMPLE 1.5. We cannot say "6.4 quarts of milk" without also saying how big a plain error *we* are willing to put up with. A spoonful? A quart?

c. However, not all differences are **significant**, that is carry information that is relevant to the real world situation.

EXAMPLE 1.6. The difference between \$3. and \$8. is the same as the difference between \$1 000 000 003. and \$1 000 000 008., namely \$5.. However, while the difference between \$3. and \$8. is *significant* because \$5. is in the same range as \$3. and \$8., the difference between \$1 000 000 003. and \$1 000 000 008. is ... *insignificant* because \$5. is *much smaller* than both \$1 000 000 003. and \$1 000 000 008..

Now, while we cannot avoid errors, we sure want to avoid *significant* errors, that is we want the error to remain **small relative** to the plain decimal number that specifies what we want. So. along with the plain decimal number that specifies what we want, we must also specify a **tolerance**, that is the largest plain error we *can* put up with (`https://en.wikipedia.org/wiki/Engineering_tolerance`).
And, in the spirit of Gowers' "*measurement* of a physical quantity", we set

> **DEFINITION 1.1 A specification** for an amount of stuff consists of *two* plain decimal numbers:
> - a plain decimal number to specify the amount we *want*,
> - a plain decimal number to specify the errors we can *tolerate*.
>
> which we will *write*
>
> given plain decimal number $\pm$ given tolerance
>
> but which, as with Gower's "error estimate", we will *read*
>
> given plain decimal number $\pm$ plain decimal number smaller than given tolerance

EXAMPLE 1.7. While we cannot specify an amount of 6.4 quarts of milk (Example 1.5, page 5.), we *can* specify an amount of 6.4 ± 0.02 quarts of milk where ± 0.02 quarts of milk is the *tolerance*: what *can* be poured will then be $6.4 \pm$ a plain decimal number *smaller than* 0.02 quarts of milk.

We can then restate **??** in a more constructive manner:

0
signed-number

> **NOTE 16.1 (Restated)** A plain decimal number *without a tolerance* can never specify an amount of stuff.

3. What about zero? As we will see again and again, **0** is a very special number and indeed already "*the ancient Greeks seemed unsure about the status of zero as a number.*" (https://en.wikipedia.org/wiki/0)

With plain decimal number other than 0, even though we cannot *have* the exact amount of stuff specified by the given plain decimal number of unit of stuff that we *want*, that exact amount of stuff does *exist*.

But 0 is special because when we specify 0 unit of some stuff, there is no such thing as 0 unit of that stuff in the real world and all we get is the error!

EXAMPLE 1.8. There is no such thing as a perfect vacuum. (https://en.wikipedia.org/wiki/Vacuum).
There is no such thing as an absolute zero temperature. (https://en.wikipedia.org/wiki/Absolute_zero)

So,

> **NOTE 1.2 0 is special** because:
> **i.** 0 specifies nothing.

4 Signed Numbers

Most of the time, we need not only to describe or specify *how many* items there are in a collection or *how much* stuff there is in an amount, but also the *way* the collection of items or the amount of stuff is going.

EXAMPLE 1.9. How many people are *going into* a building as opposed to how many are *coming out* of the building usually depends on the time of the day.
How much money is *coming into* or *going out* of our bank account usually depends on the day of the month.

> **LANGUAGE 1.5 Signed whole numbers** are usually called integers.

1. Size and sign. So, both **signed *whole* numbers** and **signed *decimal* numbers** carry *two* kinds of information:

signed *whole* number
signed *decimal* number
size
quantitative
$|\ \ |$
absolute value
sign
qualitative
$+$
$-$
positive
negative

- The **size** of a signed-number (whole or decimal) is the **quantitative** information which is given by the plain-number that describes or specifies *how many* items in the collection or *how much* stuff in the amount. The standard symbol for size is $|\ \ |$

EXAMPLE 1.10. Instead of "size $-3 = 3$" we can write "$|-3| = 3$".

> **LANGUAGE 1.6 Absolute value** is often used in textbooks instead of size but *we* will stick with size because that's what's used in the real world. Instead of the word "size", textbooks mostly use "absolute value" but, sometimes, "numerical value" or "modulus" or "norm". None of these words will be used in this text.

- The **sign** of a signed-number (whole or decimal) is the **qualitative** information which is given by $+$ or $-$, the symbols that describe or specify which *way* the collection or the amount is going.
  Positive numbers (whole or decimal) are the signed-numbers whose sign is $+$,
  Negative numbers (whole or decimal) are the signed-numbers whose sign is $-$.

EXAMPLE 1.11. $+17.43$ **Dollars** specifies a real world money transaction in which:

- The *size* of $+17.43$ is 17.43 which *specifies how much* money was transacted,
- The *sign* of $+17.43$ is $+$ which specifies *which way* the money went.

> **AGREEMENT 1.2 The *sign*** $+$ will *never* go without saying In this text.

EXAMPLE 1.12. We will always distinguish, for instance,

- $+51.7$ which is a *signed* number from 51.7 which is the *plain* number that is the *size* of $+51.7$. (As well as the *size* of -51.7
- $+643$ which is a *signed* number from 643 which is the *plain* number that is the *size* of $+643$. (As well as the *size* of -643)

opposite
signed error

A positive number and a negative number with the same size are said to be **opposite**.

EXAMPLE 1.13. Opposite$+32.048 = -32.048$

2. Zero has no sign. 0 is neither positive nor negative. So,

> **NOTE ?? (Restated) ??** because:
> **i.** 0 specifies nothing.
> **ii.** 0 has no sign

Nevertheless, we will want to consider 0 as a signed decimal number because, in spite of not having a sign, 0 does come up in many computations with signed decimal numbers.

EXAMPLE 1.14. A number and its opposite add up to 0. Conversely, if two numbers add up to 0 then they are opposite.

3. Signed error. While scientists can never know what the plain error in a measurement *is*, scientists often know if the *measured* plain decimal number is larger or smaller than the *given* plain decimal number. So scientists use **signed errors** whose size is the plain error and whose sign is:

$+$ when the *measured* plain decimal number is larger than the *given* plain decimal number

$-$ when the *measured* plain decimal number is smaller than the *given* plain decimal number

However, even with *signed* errors,

> **NOTE 1.3** **The tolerance is a *plain* decimal number** because the tolerance is the largest *size* of signed error we can put up with.

=======Begin MISPLACED=======

EXAMPLE 1.15. It makes no sense to specify $-6.4 \pm$ a *plain* error smaller than 0.02. What we can specify is $-6.4 \oplus$ a *signed* error whose *size* is smaller than 0.02 where 0.02 is the given *tolerance* (*plain* number).

=======End MISPLACED ========

4. Numbers to go. As already mentioned in **??**. we will mostly use *signed* decimal numbers—except of course when dealing with the *size* of *signed* decimal numbers. So, to make our life a little easier, we will use:

number
given number
actual number
plain number
error
general
generic
x_0
x_1
x_2
x_3

AGREEMENT 1.3 Number is short for signed decimal number including 0. In particular:

- **Given number** is short for *given* signed decimal number including 0.
- **Measured number** is short for *measured* signed decimal number including 0,
- **Error** is short for *signed* error.

And, in order to discuss the *size* of numbers,

- **Plain number** is short for plain decimal number including 0.

5. Generic given numbers. In order to make **general** statements, we will use **generic** symbols.

EXAMPLE 1.16. In ARITHMETIC, we may check that $2+3=3+2$ and then that $4+7=7+4$. Then, maybe after further experimentation or maybe just as a wild guess, we may want to make the general statement that the order in which we add two plain whole numbers does not change the result. To make that statement, we would use two generic symbols for plain whole numbers, say, a and b, and then we would state that $a+b=b+a$

In other words, a generic symbol stands for something whose identity is to remain undisclosed for the time being. In particular, a *generic* given number is a given number whose "identity" remains undisclosed so that *any* given number can later be substituted for the generic given number.

EXAMPLE 1.17. In Example 1.20, after we have stated that $a+b=b+a$, we can state without further ado that, say, $152\,695+4\,082=4\,082+152\,695$ just by replacing a by $152\,695$ and b by $4\,082$

We will use the following:

DEFINITION 1.2 $\boldsymbol{x_0}$, $\boldsymbol{x_1}$, $\boldsymbol{x_2}$, $\boldsymbol{x_3}$, etc are symbols for generic given numbers including 0.

=========OK SO FAR=========

$\oplus$
$\ominus$

=======Begin WORK ZONE=======
=======End WORK ZONE========

5 Computing With Given Numbers

We assume the reader knows how to perform the four operations with given numbers but there are nevertheless a few issues worth discussing, if only for the sake of clarity.

1. Addition and subtraction. The symbols $+$ and $-$ are vastly overused because not only are we using the symbols $+$ and $-$ for both

i. *addition* and *subtraction* of plain *whole* numbers

and

ii. *addition* and *subtraction* of plain *decimal* numbers

which already are two very different sets of numbers with very different procedures for addition and subtraction, but we are also using the symbols $+$ and $-$ to

iii. distinguish positive numbers from negative numbers

which has little to do with *addition* or *subtraction*.

So, it would really be asking for trouble for us to use, on top of all that, the symbols $+$ and $-$ for *addition* and *subtraction* of *signed* decimal numbers and this is where we draw the line:

> **DEFINITION 1.3** **$\oplus$ and $\ominus$,** pronounced "oplus" and "ominus", will be the symbols for *addition* and *subtraction* of *signed* decimal numbers.

In other words, the $\bigcirc$ around the operation symbol will remind us to take care of the signs but, as an added benefit, $\oplus$ and $\ominus$ will also let us avoid using lots of parentheses.[3]

EXAMPLE 1.18. Instead of:

$-23.87 + (-3.03)$, $\quad -44,29 - (+22.78)$, $\quad +12.04 - (-41.38)$

we will write:

$-23.87 \oplus -3.03$, $\quad -44,29 \ominus +22.78$, $\quad +12.04 \ominus -41.38$

[3] Which is presumably why, say $+13.73$ and -78.02 used to be written as ${}^{+}13.73$ and ${}^{-}78.02$ since ${}^{+}13.73 - {}^{-}78.02$ has the same advantage as $+13.73 \ominus -78.02$.

2. Multiplication and division.

$\odot$
$\overline{}$
reciprocal

i. We will use

- The operation symbol $\odot$ (instead of $\otimes$) for multiplication of signed decimal numbers,
- The operation symbol $\overline{}$ (fraction bar instead of $\oplus$) for division of signed decimal numbers.

ii. For future reference, we recall

THEOREM 1.1 Multiplication and Division of Signs

	+	−
+	+	−
−	−	+

EXAMPLE 1.19.
$+2 \odot +5 = +10$, $+2 \odot -5 = -10$, $-2 \odot +5 = -10$, $-2 \odot -5 = +10$
$\frac{+12}{+3} = +4$, $\frac{+12}{-3} = -4$, , $\frac{-12}{+3} = -4$, $\frac{-12}{-3} = +4$,

3. Reciprocal. The **reciprocal** of a number is $+1.$ divided by that number. (https://www.mathsisfun.com/reciprocal.html)

So:

i. Reciprocal $+1. = +1.$ and Reciprocal $-1. = -1.$

ii. The reciprocal of 1 followed or preceded by 0s is easy to get: read the number you want the reciprocal of and insert/remove "th" accordingly,

iii. The reciprocal of most other numbers needs to be computed and we may as well use a calculator.

EXAMPLE 1.20.

$$\text{Reciprocal } +1\,000. = +1 \text{ thousand th} = +0.001$$

$$\text{Reciprocal } -0.000\,001 = -1 \text{ million}\cancel{\text{th}} = -1\,000\,000.$$

$$\text{Reciprocal } +4.00 = \frac{+1.00}{+4.00} = +0.25 \text{ (Hopefully by hand.)}$$

$$\text{Reciprocal } -0.89 = \frac{+1.00}{-0.89} = -1.13 \text{ (Use a calculator.)}$$

$$\text{Reciprocal } -2.374 = \frac{+1.00}{-2.374} = -0.421 \text{ (Use a calculator.)}$$

EXAMPLE 1.21. In ALGEBRA, to prove that:

- When we oplus a number and its opposite, the result is 0, we compute $x_0 \oplus$ Opposite x_0 to show that the result is 0.
- When we oplus two numbers, the *order* does not matter, we compute $x_1 \oplus x_2$ and $x_2 \oplus x_1$ to show that the results are the same.
- When we oplus three numbers, the *grouping* does not matter, we compute $[x_1 \oplus x_2] \oplus x_3$ and $x_1 \oplus [x_2 \oplus x_3]$ to show that the results are the same.

$x_0 \otimes oppx_0$ is negative

4. Computing with Zero? As far as $\oplus$ and $\ominus$ are concerned, 0 is not at all special since oplussing 0 and ominussing 0 do not do anything and so, do not cause any difficulty.

On the other hand, inasmuch as

- Multiplying *any* number by 0 always gives 0 as a result,
- Dividing *any* number by 0 is impossible. (https://en.wikipedia.org/wiki/Division_by_zero)

this is yet another way

NOTE ?? (Restated) ??:
i. 0 specifies nothing.
ii. 0 has no sign
iii. Multiplying *any* number by 0 always gives 0 as a result,
iv. Dividing *any* number by 0 is impossible.

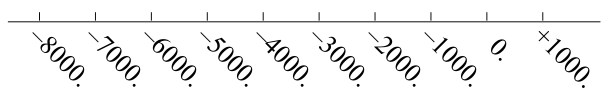
6 Picturing Given Numbers

picture
ruler
tickmark
origin
number line
symmetrical
side

1. Quantitative rulers. To **picture** given numbers, we will use quantitative **rulers** which are essentially just what goes by the name of "ruler" in the real world.

> **AGREEMENT 1.4 Origin** The **tickmarks** on a quantitative ruler must include an **origin**, that is a tickmark labeled 0.

EXAMPLE 1.22. The following :



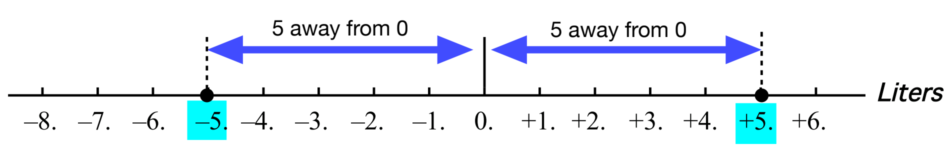
is a *quantitative* ruler.

> **LANGUAGE 1.7 Number line** is the name often used instead of quantitative ruler but in this text we will stick to quantitative ruler.

2. Graphic meanings. From the *graphic* viewpoint:

- The size of a given number specifies *how far* from 0 the given number is on a quantitative ruler. So opposite numbers are **symmetrical** relative to the origin.

EXAMPLE 1.23. The numbers -5.0 and $+5.0$ have the same *size*, namely 5.0, so they are equally far from 0:



- The sign of a given number specifies which **side** of the origin the given number is—as seen when facing 0:

> **AGREEMENT 1.5 Sides of the origin**
> ▸ Positive numbers ($+$ sign) will be to the *right* of the origin 0,

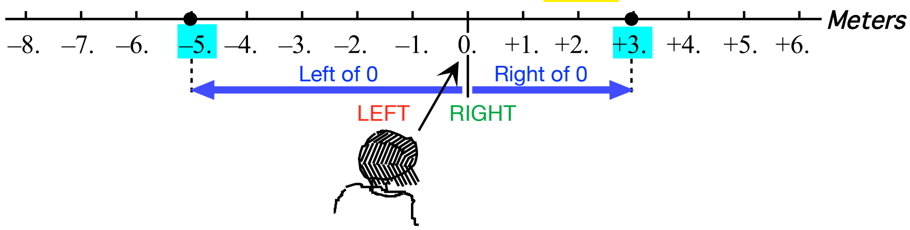
$x_0 \oplus h$

▸ Negative numbers ($-$ sign) will be to the *left* of the origin 0.

EXAMPLE 1.24. On a quantitative ruler,

Since Sign $-5 = -$, the number -5 is *left* of 0.

Since Sign $+3 = +$, the number $+3$ is *right* of 0.



==========THIS IS WHERE THINGS GET HARD ==========

7 Nearby Numbers

We already saw several instances where by itself a number does not provide much information if at all. and neighborhood of given number

1. However, see **??**, a plain decimal number *unaccompanied by a tolerance* can never specify an amount of stuff.

So now we can say that a *measured* signed decimal number is the *given* signed decimal number $\oplus$ a *signed* error whose size is smaller than the given tolerance.

To code a generic *nearby* number for the real world number x_0, we will use

DEFINITION 1.4 $x_0 \oplus h$
Given x_0, $x_0 \oplus h$ is code for a generic *nearby* number.

but a frequent mistake is to forget that (**??**, **??**)

2. Neighborhood

3. Real World Numbers (Section 10, page 26) was easy because we knew exactly where were the numbers we wanted to picture. But in the case of *measured* numbers, all we know is that the numbers we want to picture are somewhere within the given tolerance of the real world number.

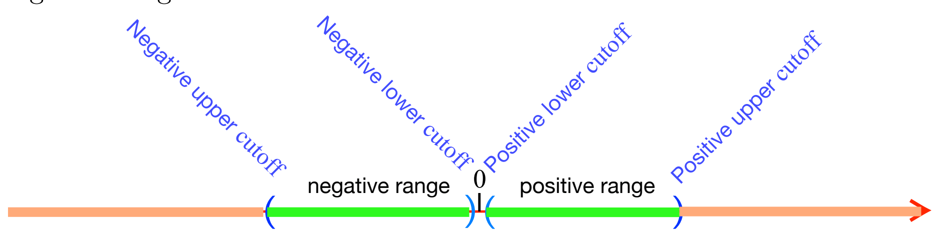
But here again qualitative rulers are not up to the task because of its scale and, here again, we must aim a magnifier at x_0 to see a neighborhood of x_0.

single points usually do not carry enough information for the purposes of the CALCULUS. So, what we will do is to **thicken** the point we want to look at, that is we will look at the point as **center** of a **neighborhood**, that is we will look at the point together with **nearby** *numbers* that is numbers within a given **radius** of the center. (http://mathworld.wolfram.com/Neighborhood.html.)

As useful as *quantitative* rulers are, and they are used a lot in *engineering* and the *sciences* to help picture **data**, that is *lots* of real world numbers, they do not lend themselves to picturing neighborhoods and to picture neighborhoods we will use **qualitative ruler**, that include just:

- A tickmark for the origin 0
- An **arrowhead** to indicate the way *up*. In this text, according to Agreement 1.4 (Page 13), the arrowhead will always point to our *right*.
- **Parentheses** to mark the cutoffs of a generic positive range and a generic negative range

thicken
center
neighborhood
nearby
radius
data
qualitative ruler
arrowhead
Parentheses

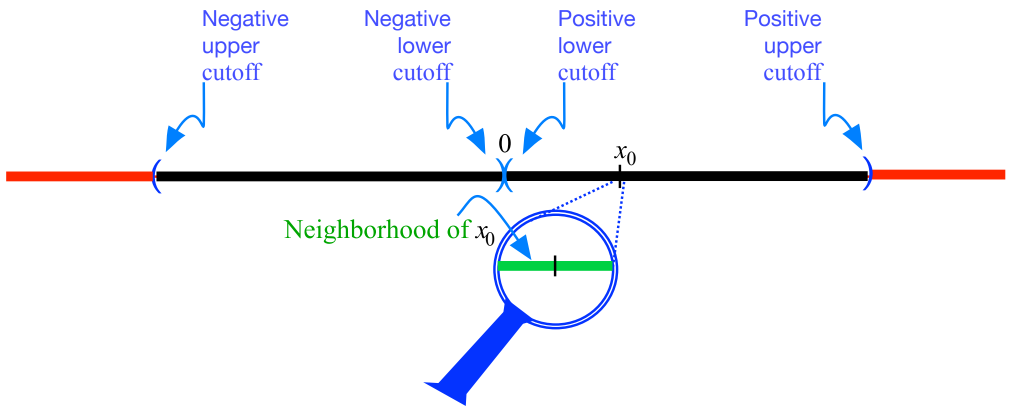


4. Since ∞ and 0 are diametrically opposed on a Magellan circle, it is of course tempting to think of ∞ and 0 as being reciprocal. Unfortunately we can't divide by 0 and ∞ is not even a number so that's that. Yet there *has* to be something to it and we will get to it later.

5. Picturing x_0+h. For the real world number x_0, the nearby numbers are in a neighborhood of x_0 with the radius of the neighborhood being the tolerance . (See https://en.wikipedia.org/wiki/Neighbourhood_(mathematics).)

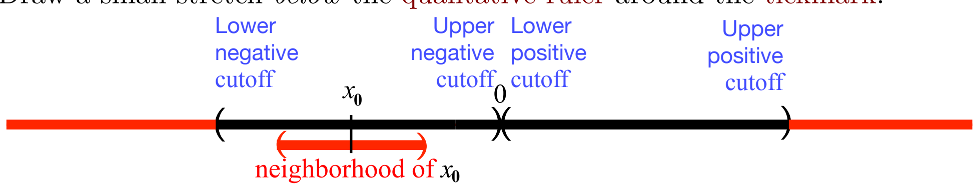
So, in order to picture $x_0 + h$ we aim a magnifier at x_0 to see a neighborhood of x_0.

x_0^+

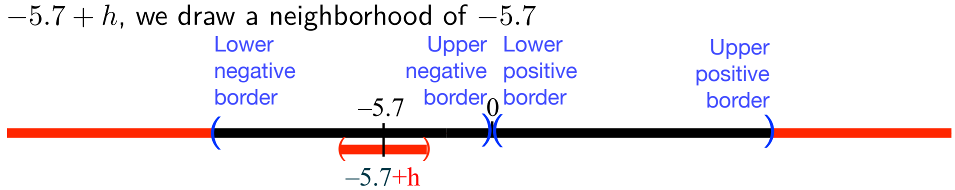


The tolerance is the radius of the neighborhood but, of course, since in this text the tolerance will remain undisclosed, we will just:

- Draw a tickmark for the real world number,
- Draw a small stretch *below* the qualitative ruler around the tickmark.



EXAMPLE 1.25. Given the number -5.7, to picture the actual number $-5.7 + h$, we draw a neighborhood of -5.7



In other words, given a number x_0, the actual number $x_0 + h$ will be a nearby number.

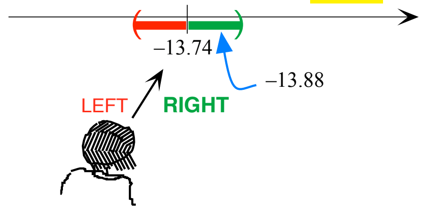
6. Sides of a neighborhood of x_0. In order to deal *separately* with each side of a neighborhood of x_0, we will use

- ▸ $\boldsymbol{x_0^+}$ (namely x_0 with a little $+$ up and to the right) which is standard code for nearby numbers right of x_0, that is for nearby numbers *larger than* x_0. (They are indeed to *our* right when *we* are facing x_0, the center of the neighborhood.)

  Another way to code nearby numbers right of x_0 is: $x_0 + h$ with $h > 0$

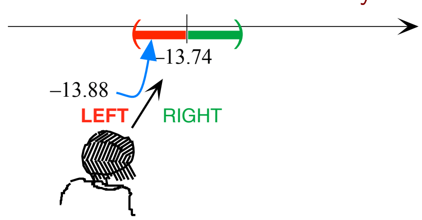
x_0^-
compare
sign-size compare
smaller

EXAMPLE 1.26. $-13.74^{\boxed{+}}$ refers to nearby numbers right of -13.74 (such as for instance -13.88):



▸ $\boldsymbol{x_0^-}$ (namely x_0 with a little $\boxed{-}$ up and to the right) which is standard code for nearby numbers left of x_0, that is for nearby numbers *smaller than* x_0. (They are indeed to *our* right when *we* are facing x_0, the center of the neighborhood.)
Another way to code nearby numbers left of x_0 is: $x_0 + h$ with $h < 0$

EXAMPLE 1.27. $-13.74^{\boxed{-}}$ refers to nearby numbers left of -13.74 such as -13.88:



8 Comparing Given Numbers

We assume that the reader knows how to **compare** *plain*-numbers but it is probably worthwhile reminding the reader that

> **NOTE 1.4 Comparing numbers without units makes no sense at all.**
> .

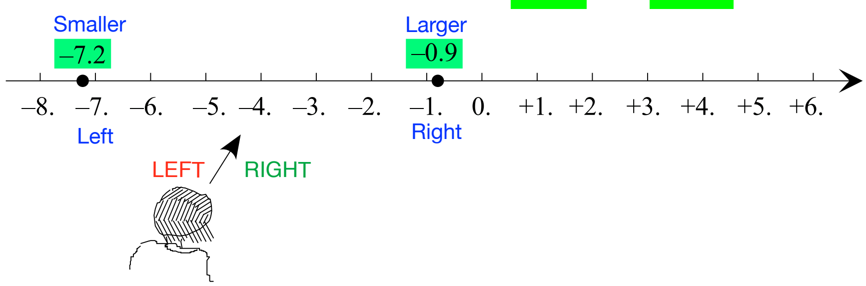
In the case of *signed*-number, though, things are more complicated because there are *two* very different ways to compare *signed*-number depending on whether or not we take the *signs* into account or only the *sizes*.

1. Sign-size comparison. To **sign-size compare** signed-numbers, that is to take both signs and sizes into account, the easiest way is to picture the two numbers on a quantitative ruler and then, because of Agreement 1.4 (Page 13), the number to *our left* will be **smaller** than the number to *our* larger

right and the number to *our right* will be **larger** than the number to *our left*.

> **NOTE 1.5** **sign-size *goes without saying* when we say larger and/or smaller.**

EXAMPLE 1.28. Given the numbers -7.2 and $-0.9.$ we have

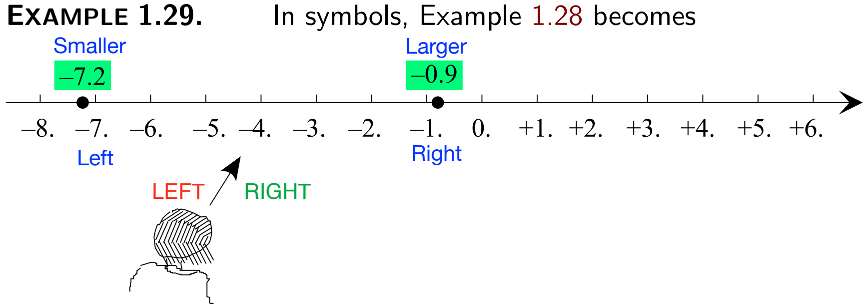


so -7.2 is smaller than -0.9 and -0.9 is larger than -7.2

The *standard* symbols for sign-size-comparisons of *all four kinds* of numbers are:

Sign-size-comparisons	Symbols
equal to	$=$
not equal to	$\neq$
smaller than	$<$
smaller than or equal to	$\leqq$
larger than	$>$
larger than or equal to	$\geqq$

EXAMPLE 1.29. In symbols, Example 1.28 becomes



so $-7.2 < -0.9$ as well as $-0.9 > -7.2$

2. Size-comparison. To **size-compare** *signed*-number is to compare them *only* in terms of their *sizes* and to *ignore* their signs. size-compare

EXAMPLE 1.30. On a ticket for speeding on a two-way road, only the *size* of the speed is mentioned, not which *way* we were going.

In fact,

NOTE 1.6 In common English, "higher" and "lower" do *not* correspond to the mathematical larger and smaller but to larger-in-size and smaller-in-size.

EXAMPLE 1.31. In common English, we say that a \$700 *expense* is *higher* than a \$300 *expense* even though -700 is *smaller* than -300 This is because -700 is *larger-in-size* than -300.

The trouble is that "size-comparing" is almost always confused with "comparing sizes". But the difference is *what* we are comparing in each case and *that* is important.

EXAMPLE 1.32. Suppose that:
i. Jack is 41 year old
ii. Jack's daughter is 15 year old
iii. Jill is 39 years old
iv. Jill'son is 17 years old
Now:
a. If we compare Jack and Jill in terms of their *own* age, we get that

Jack *is-older* than Jill,

.

But since Jack's *child* is-younger than Jill's *child*

b. If we compare Jack and Jill in terms of *their child's* age, we get from **b.** that

Jack has-a-*younger*-child than Jill

Similarly:
$+2.7$ is larger than -17.4
but since Size $+2.7$ is smaller that Size -17.4
$+2.7$ is smaller-in-size than -17.4

closer to

PROCEDURE 1.1 To size-compare two signed decimal numbers

i. Get the plain decimal numbers that are the size of each of the two signed decimal numbers,
ii. Compare the plain decimal numbers that are the sizes of the signed decimal numbers,
iii. And then

- If the *size* of the first signed decimal number is smaller than the *size* of the second signed decimal number, then the first signed decimal number is itself **smaller-in-size** than the second signed decimal number
- If the *size* of the first signed decimal number is larger than the *size* of the second signed decimal number, then the first signed decimal number is itself **larger-in-size** than the second signed decimal number

TEMO 1.1 To *size-compare* the numbers -7.5 and $+3.2$

i. We compare their *sizes*: since the size of -7.5 is the plain-number 7.5 and the size of $+3.2$ is the plain-number 3.2 and since

$$7.5 > 3.2$$

we can conclude that

size -7.5 *is-larger-than* size $+3.2$

or, in symbols, that

$$|-7.5| > |+3.2|$$

ii. On the basis of which we conclude that

-7.5 *is-larger-in-size-than* $+3.2$

And the trouble in most textbooks is that the first step is the only one that is explicited while the second step is supposed to "go without saying", perhaps because, unfortunately,

NOTE 1.7 There are *no symbols* for size-comparisons of signed-numbers.

so that *we* will have to say it in so many words.

Graphically:

- The signed-number that is smaller-in-size than the other signed-number is **closer to** 0 than the other signed-number
- The signed-number that is larger-in-size than the other signed-number is

farther from 0 than the other signed-number. farther from

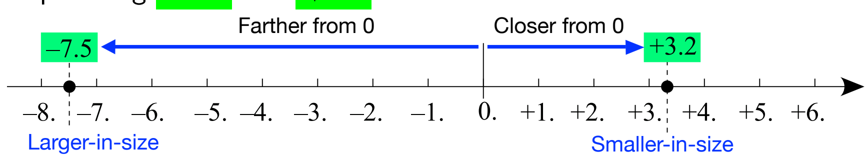
EXAMPLE 1.33. Given the numbers -7.5 and $+3.2$, we saw in ?? that

- -7.5 is *larger-in-size-than* $+3.2$,

and therefore that

- $+3.2$ is *smaller-in-size-than* -7.5,

After picturing -7.5 and $+3.2$



we see that

- -7.5 is *farther from* 0 than $+3.2$,
- $+3.2$ is *closer to* 0 than -7.5,

9 Qualitative Sizes

We can of course give *any* number and any tolerance we want and, indeed, *mathematicians* treat *all* the numbers in a set of numbers in exactly the same manner, regardless of their size.

EXAMPLE 1.34. $+36.42$ and -105.71 are added, subtracted, multiplied and divided by the same rules as $-41\,008\,333\,836\,092.017$ and -0.000001607.

In the real world however numbers come in vastly different sizes.

EXAMPLE 1.35. The numbers that astrophysicists (https://en.wikipedia.org/wiki/Astrophysics) use are entirely different from the numbers that nanophysicists (https://en.wikipedia.org/wiki/nanophysicist) use.

Well worth looking up in this regard are

- The 9 minutes 1977 classic video *at the bottom* of http://www.eamesoffice.com/the-work/powers-of-ten/,
- Terence Tao, Fields Medal 2006, http://terrytao.files.wordpress.com/2010/10/cosmic-distance-ladder.pdf

1. Out of this world. The first two limitations have to do with the fact that numbers can be incredibly large-in-size as well as incredibly small-in-size:

- We all went through a stage as children when we would count, say, "*one, two, three, twelve, seven, fourteen, ...*" but, not too long after that we were able to count properly and then, soon after that, we discovered that there was no end to whole: we could *always* count one more. But that was only the tip of the iceberg.

EXAMPLE 1.36. Start with, say, -73.8 and insert **0**s *left* of the decimal point

$$\begin{aligned} -73\mathbf{0}.8 \\ -7\,30\mathbf{0}.8 \\ -73\,00\mathbf{0}.8 \\ \dots \\ -73\,000\,000\,000\,000\,000\,000\,00\mathbf{0}.8 \end{aligned}$$

This last number is probably already a lot larger-in-size-than any number you are likely to have ever encountered but, if not, just keep inserting **0**s until you get one!

Also, see https://en.wikipedia.org/wiki/Large_numbers#Large_numbers_in_the_everyday_world)

- On the other hand, as children knowing only whole numbers, we thought there was a number smaller than all others, namely 1. With decimal numbers, though, there is no number smallest-in-size.

EXAMPLE 1.37. Start with $+0.8$ and insert **0**s *right* of the decimal point

$$\begin{aligned} &+0.\mathbf{0}8 \\ &+0.\mathbf{0}08 \\ &+0.\mathbf{0}00\,8 \\ &\quad\dots \\ &+0.\mathbf{0}00\,000\,000\,000\,000\,000\,000\,000\,000\,08 \end{aligned}$$

This last number is probably already a lot smaller-in-size-than any number you are likely to have ever encountered but, if not, just keep inserting **0**s until you get one!

2. Qualitative sizes. In view of the above, we have to face the fact that, in any real world situation, most numbers will be either too large-in-size or too small-in-size.

cutoff
upper cutoff
lower cutoff

EXAMPLE 1.38. In Example 1.50 (Page 27), how likely is a number with a million 0s *left* of the decimal point to specify anything in the real world? How about with a billion 0s ? A trillion 0s?

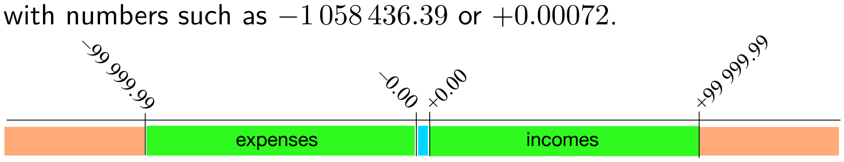
EXAMPLE 1.39. In Example 1.37 (Page 22), how is a tolerance with a million 0s *right* of the decimal point likely to work in the real world? How about with a billion 0s? A trillion 0s?

More precisely, in any real world situation, there will always be two **cutoff** *sizes*:

- An **upper cutoff** *size* above which numbers will be too large-in-size to be relevant to the situation,
- A **lower cutoff** *size* below which numbers will be too small-in-size to be relevant to the situation.

> **NOTE 1.8** **Upper and lower** refer only to the *size* of the *cutoff* numbers.

EXAMPLE 1.40. A mom and pop business could use $99\,999.99$ and 0.01 as cutoff *sizes* for their acwhole system as it probably would never have to deal with numbers such as $-1\,058\,436.39$ or $+0.00072$.



Of course, the upper cutoff size and the lower cutoff size will depend on the situation.

EXAMPLE 1.41. In contrast with the mom and pop business of Example 1.44, the acwhole system for a multination corporation would certainly use much larger cutoff sizes.

At least to an extent, the limitation on size of the specifying number and the limitation on the size of the tolerance are linked.

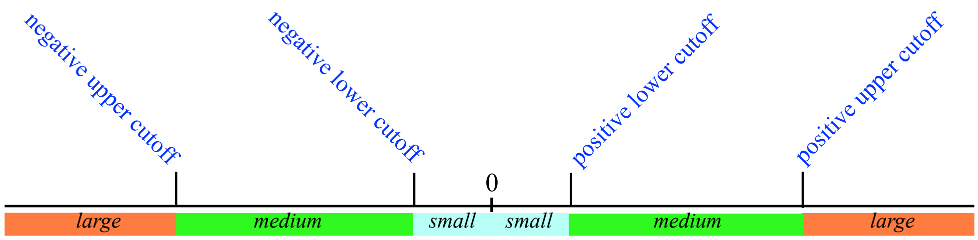
EXAMPLE 1.42. We cannot specify a distance in light years with a tolerance in inches.

qualitative size
real world number
large number
small number
finite
infinite
infinitesimal

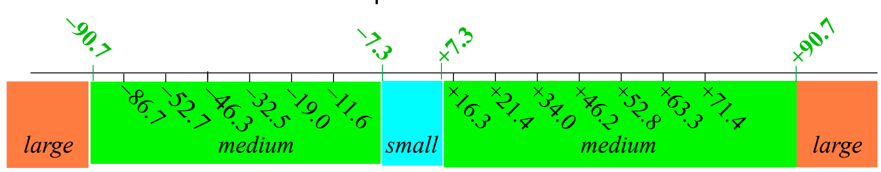
As far as we will be concerned, just knowing that, *any* real world situation, there always *are* cutoff sizes will be enough for us to use the following:

DEFINITION 1.5 Qualitative Sizes. Given a real world situation,

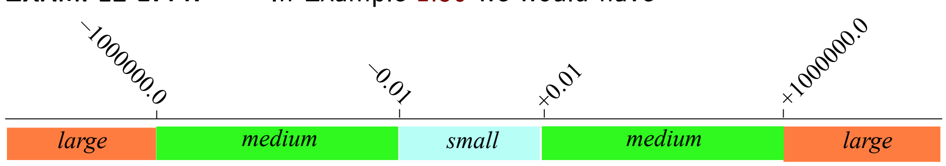
- ***Large numbers*** will be numbers whose *size* is *above* the *upper* cutoff *size*
- ***Smalll numbers*** will be numbers whose *size* is *below* the *lower* cutoff *size*



EXAMPLE 1.43. In Example 1.33 we would have



EXAMPLE 1.44. In Example 1.50 we would have



LANGUAGE 1.8 Standard Words For Qualitative Sizes.

- The standard' word for *large* is ***infinite***. (https://www.merriam-webster.com/dictionary/infinite.)
- The standard word for *small* is ***infinitesimal***. (https://en.wikipedia.org/wiki/Infinitesimal.

We will stick to the words large and small because using the words "infinite" and "infinitesimal" informally, the way *scientists* do, really annoys *mathematicians*—which *we* can't afford.

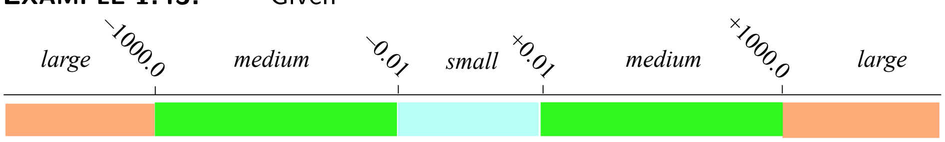
3. Zero has no qualitative size since 0 is excluded from *small* numbers. (Definition 1.8 Real World Numbers, page 27.)

NOTE 1.3 (Restated) The tolerance is a *plain* decimal number because:
i. 0 specifies nothing.
ii. 0 has no sign
iii. Multiplying *any* number by 0 always gives 0 as a result,
iv. Dividing *any* number by 0 is impossible. See https://en.wikipedia.org/wiki/Division_by_zero
v. 0 has no qualitative size.

4. Reciprocals. It is very tempting to think that the reciprocal of a *small* number is a *large* number and, the other way round, that the reciprocal of a *large* number is a *small* number.

But this is *not necessarily* the case because qualitative sizes depend on the *cutoffs* which *we* set

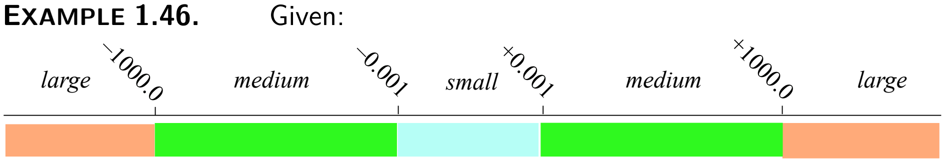
EXAMPLE 1.45. Given



i. $+0.009$ is below the positive lower cutoff $(+0.009 < +0.010 = 0.01)$ and is therefore a *small* number,
ii. The reciprocal of $+0.009$ is $+111.1$ (Use a calculator.)
iii. $+111.1$ is below the positive upper cutoff and is therefore *not* a *large* number.

But indeed, *if* we were to let the lower cutoffs and the upper cutoffs be reciprocals of each other, then it *would* be the case that the reciprocal of a *small* number would necessarily be a *large* number and the other way round.

EXAMPLE 1.46. Given:



L
h
digit
figure

where the lower cutoffs and the upper cutoffs are reciprocal of each other,
i. $+0.000\,9$ is below the positive lower cutoff ($+0.0009 < +0.0010 = 0.001$) and is therefore a *small* number,
ii. The reciprocal of $+0.0009$ is $+1\,111.1$ (Use a calculator.)
iii. $+1\,111.1$ is above the positive upper cutoff and is therefore a *large* number.

So, why didn't we also say in Agreement 1.5 that the lower cutoffs and the upper cutoffs would always have to be reciprocals of each other? Because that is not always the case in the real world.

EXAMPLE 1.47.

- In business, one penny is probably the size of the smallest amount a business can earn or lose but the reciprocal of $+0.01$ is $+100.0$ and even the tiniest business will, at least occasionally, earn or lose more than the reciprocal, \$100.0
- In astronomy, a million*th* of a mile would be unmanageably *small* but the reciprocal, a million miles, would be quite *medium*.
- In biology, a thousand inches would be unmanageably *large* but the reciprocal, a thousand*th* of an inch, would be quite *medium*.

5. Generic names.
i. There is *no standard* symbol for generic large number and *we* will use

> **DEFINITION 1.6** L is a generic *large* number

ii. There is a *standard* symbol for generic *small* number namely:

> **DEFINITION 1.7** h is a generic *small* number.

10 Real World Numbers

1. Significant digits. Both whole numbers and decimal numbers are made up of **digits**.

EXAMPLE 1.48. Both the whole number $516\,026\,618$ and the decimal number $516.026\,618$ are made of the digits $0, 1, 2, 5, 6, 8$

Language 1.9 Figure is the name often used instead of digit but In this text we will stick to digit.

a. However, not all the digits in a number are significant.

Example 1.49. To say that "*the estimated population of the US was* "$328\,285\,992$ *as of January 12, 2019*" (https://en.wikipedia.org/wiki/Demography_of_the_United_States on 2019/02/06) is not reasonable because at least the rightmost digit, 2, is certainly *not* significant: on that day, some people died and some babies were born so the population could just as well been given as, say, $328\,285\,991$ or $328\,285\,994$.
Note that further along in the Wikipedia article, the population is given more reasonably: "*from about 76 million in 1900 to 281 million in 2000*".

But as always, what is significant depends on the situation.

Example 1.50. The numbers given in https://en.wikipedia.org/wiki/Iron_and_steel_industry_in_the_United_States are much more reasonable: '*In 2014, the United States [...] produced' 29 million metric tons of pig iron and 88 million tons of steel.*" Similarly, "*Employment as of 2014 was 149,000 people employed in iron and steel mills, and 69,000 in foundries. The value of iron and steel produced in 2014 was 113 billion.*"

Identifying significant digits, however, is not quite a simple matter (https://en.wikipedia.org/wiki/Significant_figures#Identifying_significant_figures) and neither is determining in the result of a computation which digits will be significant (https://en.wikipedia.org/wiki/Significant_figures#Arithmetic).

b. The third limitation has to do with the fact that, just like small numbers, *any* number can have incredibly many decimal digits and of course only so many of these digits will be significant.

Example 1.51. What could $\$312.374333840$ possibly correspond to in the real world?

Definition 1.8 Real World Numbers. Given a real world situation, ***real world numbers*** will be numbers:
- whose size is *between the upper and lower* cutoff sizes,

finite
infinite
infinitesimal
Magellan circle

and

- whose digits are all significant

Language 1.10 Standard Words For Qualitative Sizes.

- The standard word for real world number is ***finite***. (https://en.wikipedia.org/wiki/Finite_number.)

=========OK SO FAR=========

11 Picturing small numbers

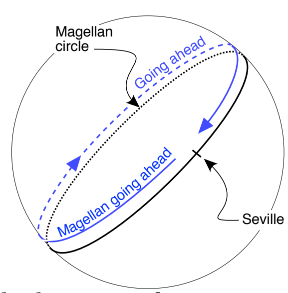
12 Picturing large numbers

13 Infinity

Rulers are "anthropocentric" inasmuch as we tend to think of ourselves as being "somewhere on the ruler". And, indeed, the idea that the earth is flat goes only so far and, similarly, so does the idea of picturing numbers with straight rulers.

1. The earth is round. If, starting from the origin, we go straight ahead on a ruler, in either direction, farther and farther, we have the feeling that the longer we go, the farther away from the origin we will get and that there is nothing that can keep us from getting as far away as we want from the origin.

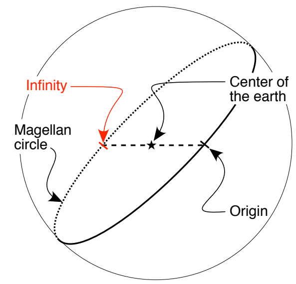
But this is not the case in the real world: even though Magellan died in 1521 while trying to go as far away from Seville as he could, his ships kept on going west and one of them eventually reached ...home, bearing witness that there was no going around the fact that the earth is round. (https://en.wikipedia.org/wiki/Ferdinand_Magellan)



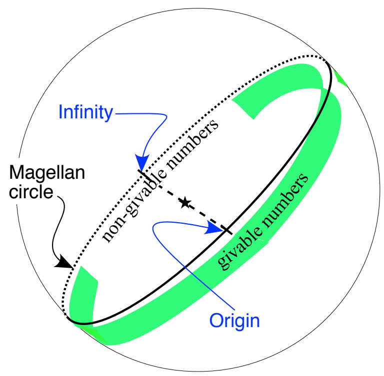
Thus, in the real world, what looks to us like a *straight line* is in fact just a piece of a **Magellan circle**.

infinity
∞
Magellan view

2. Down under. On a Magellan circle, the point **diametrically opposed** to the origin is the point farthest away from the origin. This point is called **infinity** and the symbol for infinity is ∞.



3. Magellan view. When we use a Magellan circle instead of a ruler, which is what we will call the **Magellan view**, the view is not "anthropocentric" anymore because now ∞ is in the middle of non-real world numbers the way 0 is in the middle of the real world numbers:



4. ∞ is not a number. Indeed, we will have to be careful and keep in mind that, while we can *always* compute with x_0 and *part of the time* with 0,

> **NOTE 1.9** ∞ **is *not* a number** and we can *never* compute with ∞ the way we compute with x_0 or even 0.

=======Begin WORK ZONE=======

However:

- $-\infty < x_0$
- $+\infty > x_0$
- $x_0 \oplus +\infty = +\infty$
- $x_0 \oplus -\infty = -\infty$
-

point

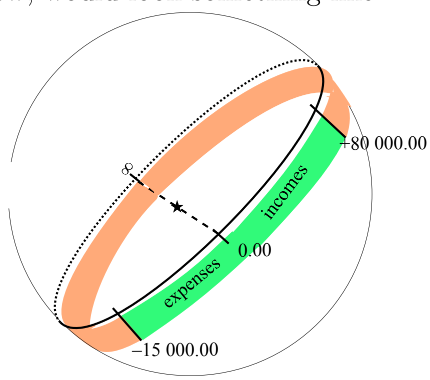
-
-

5. Points. Nevertheless, as we will see, it will be extremely convenient to use the word **point** to stand for "x_0, 0, or ∞". But we won't use a synbol for *points* because computing with such a symbol would be much too dangerous as we can *always* compute with x_0 (Definition 1.2, page 9.), only *sometimes* with 0 (**??**, **??**.) and *never* with ∞ (**??**, **??**.).

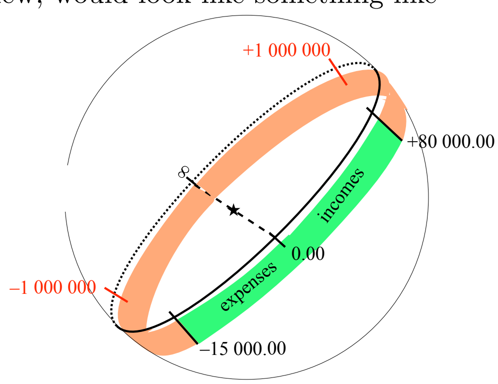
=======End WORK ZONE========

=======Begin LOOK UP=======

which, in a Magellan view, would look something like



which, in a Magellan view, would look like something like

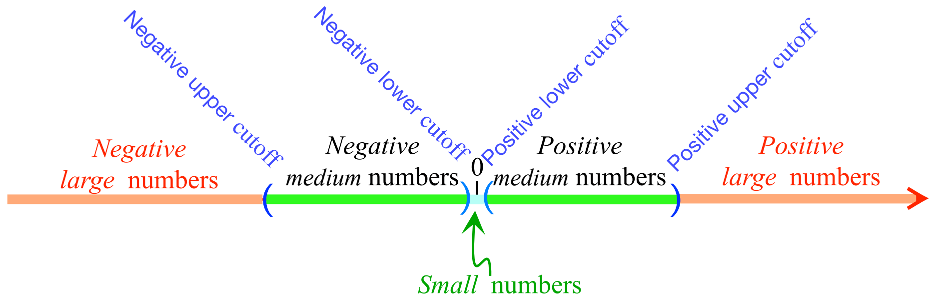


=======End LOOK UP =======

=======End WORK ZONE========

14 Picturing Small Numbers

1. Qualitative rulers give the wrong impression by making it look like there are a lot more *large* numbers than there are *small* numbers: magnifier



But since where the cutoffs are depends on what the real world numbers are, and therefore on what the particular situation being dealt with is, it is not possible to make a *general* argument as to why indeed this impression might be wrong.

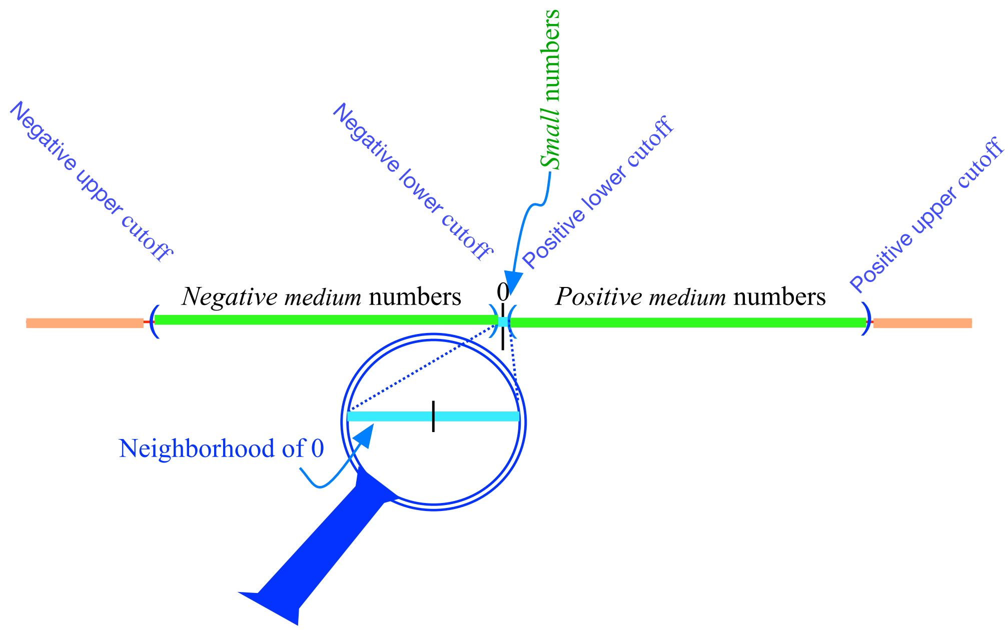
On the other hand, the two upper cutoffs, and therefore *large* numbers, are usually *beyond* -1.0 and $+1.0$ so that their reciprocals are between -1.0 and $+1.0$ and therefore have a good chance of falling between the negative lower cutoff and the positive lower cutoff and therefore to be *small*. So, it is fairly likely that each *large* number is matched with a *small* number, namely its reciprocal.

Which means, though, that *small* numbers must be packed more tightly than *large* numbers.

Example 1.52. While $+7\,000$ and $+8\,000$ differ by $+1\,000$, their reciprocals, which are $+0.000\,143$ and $+0.000\,125$ differ only by $-0.000\,018$

2. Since *small* numbers are packed so tightly, to picture *small* numbers we aim a **magnifier** at 0 to see a neighborhood of 0:

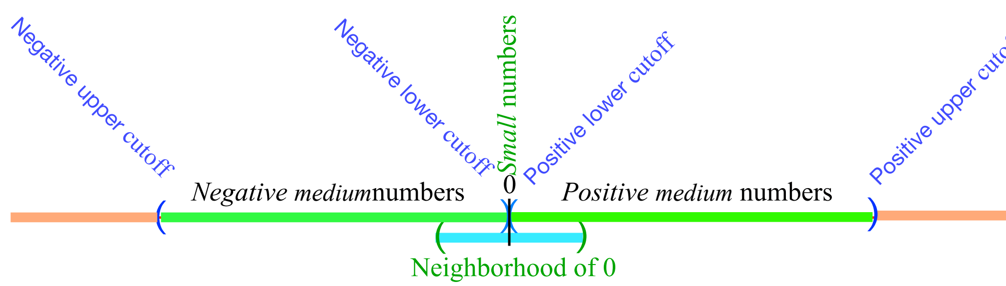
scale



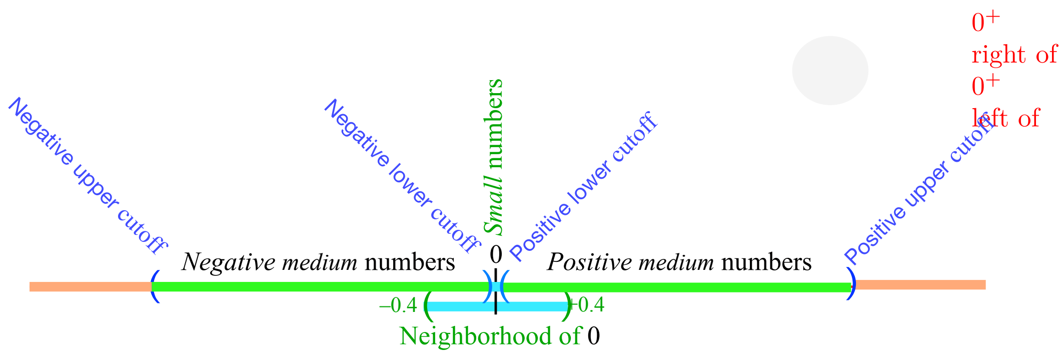
The radius of the neighborhood is the tolerance but since in this text the tolerance will remain undisclosed so will the radius of the neighborhood and we will just:

- Draw a tickmark for 0,
- Draw a small stretch *below* the qualitative ruler around the tickmark.

Since the **scale** of the neighborhood is larger than the scale of the ruler (`https://en.wikipedia.org/wiki/Scale_(map)#Large_scale,_medium_scale,_small_scale`), though, drawing the neighborhood on top of the ruler, as is often done, can be misleading and, for the picture to be perfectly clear, *we* will draw the neighborhood of 0 just *under* the qualitative ruler:



Example 1.53. With a given radius of 0.4 the neighborhood of 0 would extend from -0.4 to $+0.4$:



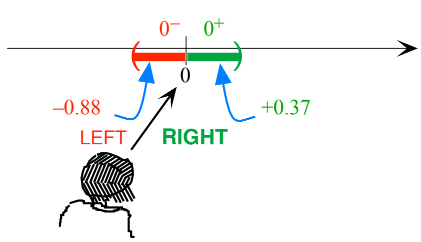
So, a

> **DEFINITION 1.9 Neighborhood of 0** consists of the *small* numbers.

3. Sides of a neighborhood of 0. In order to deal *separately* with each side of a neighborhood of 0, we will use

- **0^+** (namely 0 with a little $+$ up and to the right) which is *standard* code for nearby numbers **right of** 0, that is for positive *small* numbers. (They are indeed to *our* right when *we* are facing 0, the center of the neighborhood.) .
- **0^-** (namely 0 with a little $-$ up and to the right) which is *standard* code for nearby numbers **left of** 0, that is for negative *small* numbers. (They are indeed to *our* left when *we* are facing 0, the center of the neighborhood.) .

EXAMPLE 1.54. 0^{+} refers to nearby numbers right of 0 (such as for instance $+0.37$) and 0^{-} refers to nearby numbers left of 0 (such as for instance -0.88):



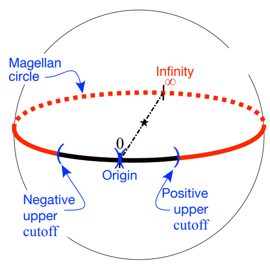
So, never forget that

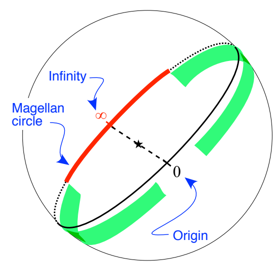
neighborhood of ∞
neighborhood
center
Mercator view

NOTE 1.10 A small $^+$ or $^-$, *alone* and up to the right, is *not* an "exponent".

15 Picturing Large Numbers

1. In the Magellan view, we see that the two stretches beyond the ranges make up in fact a single stretch of the Magellan circle whose center is ∞:



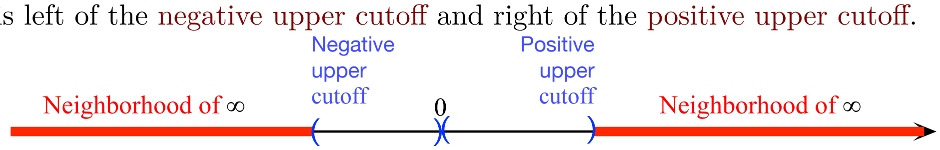


So,

DEFINITION 1.10 A neighborhood of ∞ is the part of the Magellan circle that is beyond the upper cutoffs and whose **center** is ∞.

In other words, large numbers are near ∞ and to thicken ∞ will mean to look instead at ∞ together with *large* numbers.

2. But in the **Mercator view** (`https://en.wikipedia.org/wiki/Mercator_projection`), which is when we look just at the qualitative ruler, we will say that a neighborhood of ∞ is the part of the qualitative ruler that is left of the negative upper cutoff and right of the positive upper cutoff.



In other words:

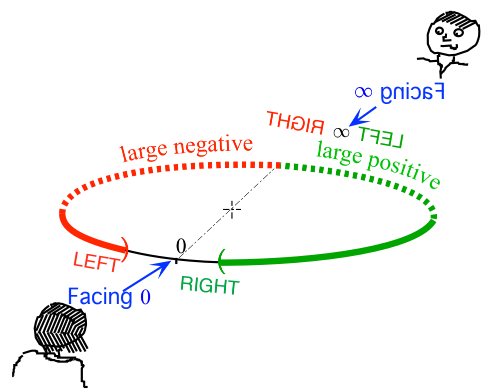
DEFINITION 1.9 (Restated) Neighborhood of 0 consists of the *large* numbers.

near
side
right of ∞
-large

3. Nearness. Instead of saying that a number is in a neighborhood of something, it is standard to say that the number is **near** that something. We then have a couple more ways to think of *large*:

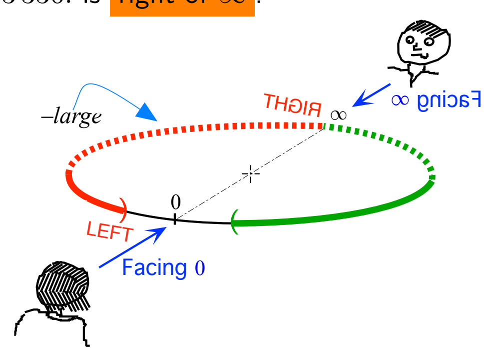
DEFINITION 1.7 (Restated) ***h***
- *Large* numbers are the numbers that are in a neighborhood of ∞.
- *Large* numbers are the numbers that are near ∞.

4. Sides of a neighborhood of ∞. In order to refer separately to each **side** of a neighborhood of ∞, we need to imagine that we are facing the center of the neighborhood in the Magellan view, that is that we are "facing ∞":



We will then say that:
- Numbers **right of** ∞ refers to *large negative* numbers because if we could be facing ∞ *large negative* numbers would then be right of ∞. We will use $-$***large*** as code for numbers right of ∞.

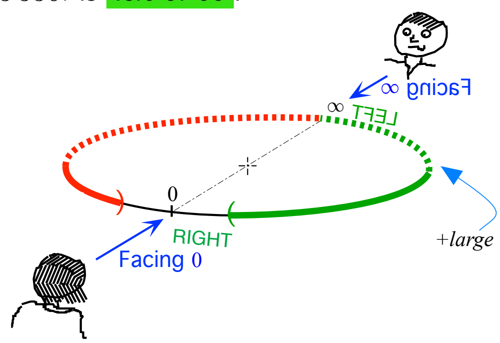
EXAMPLE 1.55. $-724\,873\,336.$ is right of ∞:



left of ∞
+large
$-\infty$
$+\infty$

- Numbers **left of** ∞ refers to *large positive* numbers because if we could be facing ∞ *large positive* numbers would then be left of ∞. We will use $\boldsymbol{+large}$ as code for numbers left of ∞.

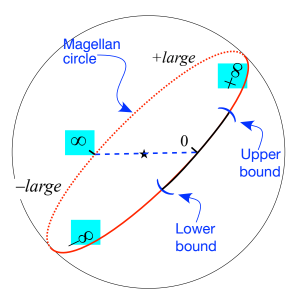
EXAMPLE 1.56. $+724\,873\,336.$ is left of ∞:



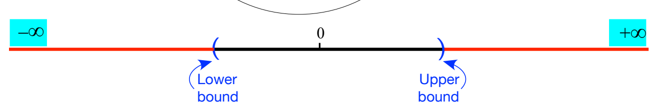
5. Sides of ∞ in Mercator view. But we will also use the Mercator view and while it is fairly easy to remember which side is left of ∞ and which side is right of ∞ in the Magellan view, it;s easy to forget in the Mercator view.

So, we will also use the sign of the *large* numbers to refer to the sides of ∞ and we will label the extremities of qualitative rulers with $-\infty$ and $+\infty$ for which, however, there are no tickmarks because they do not label numbers but only "the end of the line".

- Magellan view:



- Mercator view:



But keep in mind that the

NOTE 1.11 Sign in front of ∞ *cannot* be "*the sign of* ∞" because (??) in the first place. (Again, the sign in front of ∞ designates a *side* of ∞.) undetermined

16 Computing With Qualitative Sizes

For *computational* purposes, qualitative sizes make up a rather crude system because:

- *medium* carries no information about where the lower cutoffs and the upper cutoffs are,
- *large* carries no information about where the upper cutoffs are, that is "how large" *large* is
- *small* carries no information about where the lower cutoffs are, that is "how small" *small* is

On the other hand, as we will see, qualitative sizes will carry plenty enough information for *our* investigations in *this* text. [4]

1. A good rule of thumb for experimenting with numbers of qualitative sizes will be

- *medium*: try ± 1,
- *large*: try ± 10.0 or ± 100.0 or ± 1000.0 etc
- *small*: try ± 0.1 or ± 0.01 or ± 0.001 etc

And of course, if a number is:

- *large*, then all numbers that are *larger-in-size* will themselves be *large*,
- *small*, then all numbers that are *smaller-in-size* will themselves be *small*.

We will now see to what extent we can *compute* with *large*, *small* and *medium*.

2. Addition and subtraction. *large* $-$ *large* is **undetermined** because the result could be *large*, *small* or *medium* depending on "how large" each one of the two *large* is.

[4] Moreover, *qualitative sizes* lead quite naturally to Bachmann-Landau's o's and O's (See https://en.wikipedia.org/wiki/Big_O_notation) and in turn to asymptotic expansions (See https://en.wikipedia.org/wiki/Asymptotic_expansion) which is what *physicists*, *chemists*, *biologists*, and *engineers* use *all the time*.

Example 1.57. Here are two instances of $large - large$ that are different in *qualitative sizes*:
$+1\,000\,000\,000\,000.7 \;-\; +1\,000\,000\,000.4 \;=\; +999\,000\,000\,000.3$,
but
$+1\,000\,000\,000\,000.5 \;-\; +1\,000\,000\,000\,000.2 \;=\; +0.3$.

3. Reciprocals and we have:

Theorem 1.2 Reciprocal of qualitative sizes

- The reciprocal of $\pm large$ is $\pm small$: $\frac{+1.0}{\pm large} = \pm small$,
- The reciprocal of $\pm medium$ is $\pm medium$: $\frac{+1.0}{\pm medium} = \pm medium$,
- The reciprocal of $\pm small$ is $\pm large$: $\frac{+1.0}{\pm small} = \pm large$.

Keeping in mind that generic codes always *include* the sign, we have

Theorem 1.2 (Restated) Reciprocal of qualitative sizes

$$\text{Reciprocal of } h = \frac{+1.0}{h} = L$$
$$\text{Reciprocal of } L = \frac{+1.0}{L} = h$$
$$\text{Reciprocal of } x_0 = \frac{+1.0}{x_0} = y_0$$

The fact that numbers that are near ∞ are *far* from 0 and therefore the change of viewpoint from 0 to ∞ makes it of course tempting to say that 0 and ∞ are reciprocal of each other and the more so that, on a Magellan circle, 0 and ∞ are diametrically opposed to each other. However,

- Since a neighborhood of ∞ looks a lot bigger than a neighborhood of 0 the situation is not really *that* symmetrical.
- Since we cannot divide *any* number by 0 (See **??**, **??**), we certainly cannot divide $+1.0$ by 0 and so 0 has no reciprocal.
- Since ∞ is not a number to begin with (See **??**, **??**) ∞ has no reciprocal.

The last two, though, is where thickening with nearby numbers comes to the rescue:

- We can thicken 0 with *small* numbers
- We can thicken ∞ with *large* numbers

and then, using Theorem 1.2 (Page 38.) we get:

THEOREM 1.2 (Restated) Reciprocal of qualitative sizes

- The reciprocal of being near ∞ is being near 0,
- The reciprocal of being near 0 is being near ∞,
- The reciprocal of being near a *medium* number is being near a *medium* number.

4. Multiplication. While *multiplying* by 0 always gives 0—see **??** (**??**), and while we cannot *multiply* by ∞ at all—See **??** (**??**), we can often *multiply* by *small* and by *large*:

THEOREM 1.3 Multiplication of qualitative sizes

$\cdot$	*large*	*medium*	*small*
large	*large*	*large*	?
medium	*large*	*medium*	*small*
small	?	*small*	*small*

i. The non-highlighted entries are pretty much as we would expect.

EXAMPLE 1.58. $10\,000 \cdot 1\,000 = 10\,000\,000$ and $0.01 \cdot 0.001 = 0.00001$

ii. The two highlighted entries, that is *large* $\cdot$ *small* and *small* $\cdot$ *large*, are undetermined because the result could be any of *large*, *small* or *medium* depending on how small is *small* compared to how large *large* is.

EXAMPLE 1.59. Here are three instances of *large* $\cdot$ *small* that are different in *qualitative sizes*:

$1\,000 \cdot 0.1 = 100$, $1\,000 \cdot 0.001 = 1$, $1\,000 \cdot 0.00\,001 = 0.01$

5. Division. While we cannot *divide* by 0—see **??** (**??**), we can often *divide* by *small* and more generally we have::

THEOREM 1.4 Division of qualitative sizes

$\div$	*large*	*medium*	*small*
large	?	*large*	*large*
medium	*small*	*medium*	*large*
small	*small*	*small*	?

real number

i. The non-highlighted entries follow from Theorem 1.3 and the fact that Reciprocal of $large = 1 \div large = small$ and Reciprocal of $small = 1 \div small = large$ (Theorem 1.2).

EXAMPLE 1.60. $1\,000 \div 0.01 = 1\,000 \cdot$ Reciprocal $0.01 = 1\,000 \cdot 100 = 100\,000$

ii. The two highlighted entries, namely $\boxed{large \div large}$ and $\boxed{small \div small}$, are undetermined because the result could be any one of *large*, *small* or *medium* depending on how large each one of the two *large* is and how small each one of the two *small* are.

EXAMPLE 1.61. Here are three instances of $\boxed{large \div large}$ that are different in *qualitative size*:
$1\,000 \div 10 = \boxed{100}$, $1\,000 \div 1\,000 = \boxed{1}$, $100 \div 1\,000 = \boxed{0.1}$

EXAMPLE 1.62. Here are three instances of $\boxed{small \div small}$ that are different in *qualitative size*:
$0.001 \div 0.1 = \boxed{0.01}$, $0.001 \div 0.001 = \boxed{1}$, $0.01 \div 0.001 = \boxed{10}$

17 Real Numbers

As opposed to Numbers, most textbooks use so-called **real numbers** which are an entirely different kind of numbers. This text will not really *use* real numbers and the purpose of this section is only to give the reader an idea of what the difficulties with *really* using real numbers would be and thus to explain *why* we will mostly use signed decimal numbers and how *we* will occasionally use real numbers.

1. What are *real* numbers anyway? Even though most college mathematics textbooks claim to *use* real numbers the closest they ever come to *explaining* what real numbers *are* is something along the lines of "*a real number is a value of a continuous quantity that can represent a distance along a line.*" (`https://en.wikipedia.org/wiki/Real_number`.) Which, one has to admit, isn't particularly enlightening. [5]

But there is a very good reason for that: in contrast with signed decimal numbers, real numbers are *extremely* complicated to pin down.

[5]Moreover, this "definition" keeps changing with time! A sign of unease?

fraction
root

EXAMPLE 1.63. "*The real number system* $(\mathbb{R}; +; \cdot; <)$ *can be defined axiomatically* [...] *There are also many ways to construct "the" real number system, for example, starting from natural numbers,* (https://en.wikipedia.org/wiki/Natural_number) *then defining rational numbers algebraically* (https://en.wikipedia.org/wiki/Rational_number), *and finally defining real numbers as equivalence classes of their Cauchy sequences or as Dedekind cuts, which are certain subsets of rational numbers.*" (https://en.wikipedia.org/wiki/Real_number#Definition)
Which, unless you are a mathematician, is not exactly enlightening either.
Moreover, the above "construction" is, in fact, quite incomplete as one really should: **i.** go the Dedekind cuts route *and also* extend the *metric and* show that the quotient is metric-complete, *and* **ii.** go the Cauchy sequence route *and also* extend the *order and* show that the quotient is order-complete, *and* **iii.** show that the two quotients are both metric-isomorphic and order-isomorphic.
In *any* case, a *very* tall order.

2. Fractions and roots In fact, *at best*, that is when the given real number is a **fraction** or a **root**, a given real number is only like a Birth Certificate in that the given real number is just a *name* that says where the real number is coming from. But, by itself, certainly gives no indication of what its size is.

EXAMPLE 1.64.
- The *fraction* $\frac{4168}{703}$ is just a *name* for the solution of the equation $703\,x = 4168$ (Assuming there *is* a solution!)
- The *root* $\sqrt[3]{-17.3}$ is just a *name* for the solution of the equation $x^3 = -17.3$. (Assuming there *is* a solution!)

However, this *best* case is actually extremely rare and most given real numbers do not tell us by themselves where they are coming from which leaves us with no way to get even a rough idea of what the size of that given real number might be. You just have to find out from somewhere.

In textbooks it's of course the other way around,

EXAMPLE 1.65.
- π is just a *name* that does *not* say by itself that π is "*the ratio of a circle's circumference to its diameter*". (https://en.wikipedia.org/wiki/Pi)
- e is just a *name* that does *not* say by itself that e is "*a mathematical*

approximate

constant which appears in many different settings throughout mathematics". (https://en.wikipedia.org/wiki/E_(mathematical_constant))

3. Computing with real numbers can be done directly from the code *only* with the same two kinds of real numbers, that is when the real numbers are fractions or roots:

i. When the real numbers are fractions, there are rules to compare, add, subtract, multiply and divide directly with the codes. (https://en.wikipedia.org/wiki/Rational_number#Arithmetic)

EXAMPLE 1.66. To know which is the larger of $\frac{4168}{703}$ and $\frac{5167}{831}$ we can use a rule that involves computing the "common denominator".

ii. When the real numbers are roots , there are rules to multiply and divide directly with the codes but *not* to add or subtract. (https://en.wikipedia.org/wiki/Nth_root#Identities_and_properties)

iii. However, it is usually not possible to *compute* with both kinds of real numbers at the same time.

EXAMPLE 1.67. Add e and π or figure out which of the two is larger. (Hint: you can't do either from the code.)

And, even when the real numbers are fractions and roots, things can still be difficult.

EXAMPLE 1.68. Add $\sqrt[3]{64}$ and $\frac{876}{12}$ or figure out which of the two is larger. (Hint: you *can* do both but *not* with the only slightly different $\sqrt[3]{65}$ and $\frac{875}{12}$.)

iv. Of course, the examples in textbools use mostly fractions and/or roots even though it is at the cost of being immensely misleading if only because *most* real numbers are *neither* fractions nor roots. [6]

18 Decimal Approximations

The way *engineers* and *physicists*, *chemists*, *biologists*, compute with real numbers is by **approximating** the real numbers with signed decimal num-

[6]It is also at the expense of a unified view and therefore of forcing memorization of scattered recipes.

bers.

procedure
[...]
largest permissible error

1. To begin with, one way or the other, *all* real numbers, *including* fractions and roots, come with a **procedure** for computing approximations by signed decimal numbers. Of course, the more "exotic" the real number is, the more complicated the procedure for approximating is.

EXAMPLE 1.69.

- To approximate $\frac{4168}{703}$, we use the division procedure to *divide* 703 into 4168. Few divisions, though, end of themselves. But when a division does not, the more we push the division, the better the approximation.
- To approximate $\sqrt[3]{17.3}$, we essentially proceed by trials and errors:
  $2.0^3 = 8.0$, $3.0^3 = 27.0$, so, since 17.3 is between 8.0 and 27.0, $\sqrt[3]{17.3}$ *must* be somewhere between 2.0 and 3.0. (But how do we know that it *must*?)
  $2.7^3 = 19.683$ so, since 17.3 is less than 19.183, $\sqrt[3]{17.3}$ *must* be less than 2.7, etc. (But how do we know that it *must*?)
  Of course, the actual procedure is *systematic* but that's the idea.
- There are many ways to approximate π. Perhaps the simplest one is the Gregory-Leibniz series whose first few terms are:
  $\frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \frac{4}{13} \quad \ldots$
  However, even with "*500,000 terms, it produces only five correct decimal digits of π*" (https://en.wikipedia.org/wiki/Pi#Approximate_value)
- One of the very many ways to approximate e is:
  $1 + \frac{1}{1} + \frac{1}{1\cdot 2} + \frac{1}{1\cdot 2\cdot 3} + \frac{1}{1\cdot 2\cdot 3\cdot 4} \quad \ldots$
  (https://en.wikipedia.org/wiki/E_(mathematical_constant)#Asymptotics)

2. Since a given real number is usually *not* equal to the signed decimal number that we will use to approximates it, in order to write *equalities* we will have to use:

> **DEFINITION 1.11** **[...]** will be code for "some *small* number, positive or negative, whose size is too *small* to matter here".

In other words, [...] is a *signed* number about which the only thing we know is that the size of [...] is *less* than the **largest permissible error** whichi is the equivalent here of a tolerance.

EXAMPLE 1.70.

- $\frac{4168}{703} = 5.929 + [...]$ where $[...]$ is less than 0.001 which is the largest permissible error. (Else the procedure would have generated 5.928 or 5.930 instead of 5.929.)
- $\sqrt[3]{17.3} = 2.586\,318\,666\,944\,673 + [...]$ where $[...]$ is less than $0.000\,000\,000\,000\,001$ which is the largest permissible error. (Else the procedure would have generated $2.586\,318\,666\,944\,672$ or $2.586\,318\,666\,944\,674$ instead of $2.586\,318\,666\,944\,673$.)
- $\pi = 3.141\,5 + [...]$ where $[...]$ is less than $0.000\,01$ which is the largest permissible error. (Else the procedure would have generated $3.141\,4$ or $3.141\,6$ instead of $3.141\,5$.)
- $e = 2.718\,281\,82 + [...]$ where $[...]$ is less than $0.000\,000\,01$ which is the largest permissible error. (Else the procedure would have generated $2.718\,281\,81$ or $2.718\,281\,83$ instead of $2.718\,281\,82$.)

3. So we have come full circle back to signed decimal numbers and the question then is why should people who want to *learn* CALCULUS have to use real numbers that they will then have to *approximate* with signed decimal numbers rather than use signed decimal numbers directly from the start?

Engineers, *physicists*, *chemists*, *biologists*, etc all use signed *decimal* numbers. After all, and to quote Gowers again, "*physical measurements are not real numbers. That is, a measurement of a physical quantity will not be an exactly accurate infinite decimal. Rather, it will usually be given in the form of a finite decimal together with some error estimate:* $x = 3.14 \pm 0.02$ *or something like that.*" [7]

And, certainly not least, "*most calculators do* not *operate on real numbers. Instead, they work with finite-precision approximations.*" See "In computation" at https://gowthamweb.wordpress.com/2016/05/01/real-numbers/

The answer to the above question then is: no reason at all. As *engineers* are fond of saying, the *real* real numbers are the *decimal* numbers.

Except possibly if you want to become a *mathematician*. And even then, having worked with signed decimal numbers can help you learn about real numbers. (See Gowers' https://www.dpmms.cam.ac.uk/~wtg10/decimals.html)

So, in this text, like for *engineers*, *scientists*, and *calculators*, number

[7] https://www.dpmms.cam.ac.uk/~wtg10/continuity.html

(Agreement 1.3, page 9).

We are now, finally, ready to start on the CALCULUS!

=======Begin WORK ZONE=======

In **??** **??** of the Introduction of **??** Numbers on **??**
In section 3 Plain Decimal Numbers on page 3
In subsection 3.1 Units on page 3
3
3.1

=======End WORK ZONE========

Dans $x_0 + h$, $|h| < 1$ pour que $|h| > |h^2| > |h^3| \ldots$

Quand x_0 n'est pas trop grand, cela correspond à quelque chose de réel. Par exemple, si x_0 est un nombre que l'on veut réaliser, h est l'erreur que l'on commettra et $x_0 + h$ sera ce qu'on obtiendra.

Mais je ne vois pas à quoi de réel h correspond quand x_0 est grand.

.

EXAMPLE 1.71.

Lorsque x_0 est, disons 73 un h de 0.1 correspond, par exemple, à une incertitude de mesure.
Lorsque x_0 est, disons $73\,000\,000$, à quoi correspond un h de 0.1?
Bien sûr, avec des unitées, on peut remplacer $73\,000\,000$ mètres par 73 mégamètres et le h devient 0.1 mégamètres. Mais je ne crois pas que ça réponde vraiment à la question.

Functions of various kinds are "the central objects of investigation" in most fields of modern mathematics.

relation

Michael Spivak[1]

Chapter 2

Functions

As we will see, the CALCULUS is about *calculating* with "functions" which are entities that "*are widely used in science, and in most fields of mathematics.*" (https://en.wikipedia.org/wiki/Function_(mathematics).)

1 Relations

That a "single point usually does not carry enough information" (**?? ??**, **??**.) is in fact an instance of a general principle, namely that there isn't a thing in the real world that stands alone, all by itself: every single thing in the real world is **related** to many other things.

EXAMPLE 2.1.
- Everything sits on something: people sit on chairs that sit on floors that sit on joists that sit on walls that sit on ...
- Human beings can only live in a society.

[0]Calculus, 4th edition

output
input
relation
pair
table

1. In fact, a thing is known only by the things that are related to it. The thing we want to know about will be the **output** and the thing that will give us the information about the output is the **input**.

EXAMPLE 2.2. The following are variants found in many cultures of the same thought:

You tell me: (input)	then	I'll tell you: (output)	
The company you keep		what you are	(Dutch)
Who's your friend		who you are	(Russian)
What you are eager to buy		what you are	(Mexican)
With whom you go		what you do	(English)
Who your father is		who you are	(Philippine)
What you eat		what you are	(French)

(https://answers.yahoo.com/question/index?qid=20090403194549AAYzSEr)

2. More precisely, a **relation** is specified by whatever process, device, procedure, agency, converter, exchanger, translator, etc, that **pairs** each input to the related output(s). See https://en.wikipedia.org/wiki/Binary_relation

For instance, in disciplines like *psychology*, *sociology*, *business*, *accounting*, etc but also in the *experimental part* of *physics*, *chemistry*, *biology*, *engineering*, etc relations are often specified by **tables**.

EXAMPLE 2.3. The table

People: (input)	Thing(s) people like to do: (output(s))		
Andy	walking	playingi music	
Beth			
Cathy	reading	walking	learning calculus

specifies a relation in which Andy is paired (among others) to playing music, Beth is paired to nothing, and Cathy is paired (among others) to learning calculus.

3. Given a relation, there will be

call for

> **DEFINITION 2.1 Two kinds of problems :**
> - **Direct problems** where an *input* is given and we have to find *all* the outputs that the given input is paired to.
> - **Reverse problems** where an *output* is given and we have to find all the inputs that are paired to the given output.

EXAMPLE 2.4. Given the relation in Example 2.3 (Page 50)

- A *direct problem* might be: What are all the things Andy likes doing?
  Answer: walking, playing music
- A *reverse problem* might be: What are all the people who like walking?
  Answer: Andy, Cathy

2 Functions

To see if something is changing qualitatively we *must* look at it in relation to something else.

EXAMPLE 2.5. The only way to realize we are moving when we are in an airplane is to look out the window. Which is why, similarly, it took a long time for people to realize the earth is moving around the sun. Which is why to realize the entire galaxy we are in is moving is even harder.

This is even more the case for quantitative information.

EXAMPLE 2.6. We might say that someone's income tax was $2 270 but, by itself, that would not really be much information.
For instance, $2 270 was a lot less money in, for instance, Year 2013 than it was a century earlier, in Year 1913—the year income tax was first established.
Similarly, $2 270 would not be much money for a billionaire but would be a lot of money for a working stiff.
So, for saying that someone's income tax is $2 270 to be real information, there would have to be some table pairing Years or Incomes with Income Tax.

function

1. However, the fact that there is nothing to prevent a relation from pairing *one* input to *many* outputs can make seeing *changes* quite difficult.

EXAMPLE 2.7. That a *slot machine* can pair a number of coins with just about any number of coins makes the gambler's life quite hard.
That a *parking meter* pairs a number of coins with only one parking time makes life a lot easier.

2. So we will restrict ourselves to **functions**, that is relations that satisfy the

DEFINITION 2.2 Functional requirement

No input can be paired to *more than one* output.

or, to put it as mathematicians would,

An input can be paired to *at most one* output.

EXAMPLE 2.8. In Example 2.7 (Page 52)
The *slot machine* does not satisfy the functional requirement because even when two persons input the same amount of money the slot machine can output different amounts of money.
The *parking meter* does satisfy the functional requirement because whenever two persons input the same amount of money the parking meter will always output the same amount of parking time.

EXAMPLE 2.9. The relation specified by the table

People (Input)	Things people like to do (Output)
Dave	skating
Eddy	driving
Fran	singing

satisfies the functional requirement

EXAMPLE 2.10. The relation specified by income tax tables is a function. return domain

3. According to definition 2.2, given an input, a function *may* **return** *one* output but

> **NOTE 2.1 A function *may* return *no* output.**[2]

EXAMPLE 2.11. The relation specified by the table

People (Input)	Things people like to do (Output)
Guy Hazel Izzy	skiing

satisfies the functional requirement.

EXAMPLE 2.12. The relation specified by income tax tables is a function even though incomes below the minimum owe no income tax. (On the other hand, one might argue that the tax they owe is $\$0.00$ so this is perhaps not really quite a good example.)

4. On the other hand, it is quite possible for a function to pair *many* inputs to *one* same output . In other words, the very same output may be returned by a function for *many* inputs.

EXAMPLE 2.13. A business may be looked upon as the function specified by the *input-output table* of its profits/losses over the years:

[2]Actually, functions should not be allowed to return *no* output because that causes a *theoretical* difficulty and one should introduce the notion of **domain**. But since this *theoretical* difficulty is not about to come up any time soon, here we need not complicate things unnecessarily.

for
at
direct problem
reverse problem

Fiscal Year	Profit/Loss
1998	$+5\,000$
1999	$-2\,000$
2000	
2001	$+5\,000$
2002	$-2\,000$
2003	$-1\,000$
2004	
2005	$+5\,000$

In 1998, 2001, and 2005 the business returned the same profit/loss namely +5 000

AGREEMENT 2.1 "at" versus "for" We will often say "the output **at** the given input" as a shorthand for "the output returned by the function **for** the given input". .

5. In the case of a *function*, the two kinds of problems (Definition 2.1, page 51) become

DEFINITION 2.1 (Restated) Two kinds of problems:

- **Direct problems** where an *input* is given and we have to find the *single* output (if any) that the function returns for the given input,
- **Reverse problems** where an *output* is given and we have to find *all* the (possibly several) inputs for which the function will return the given output.

EXAMPLE 2.14. Given the business in Example 2.13 (Page 53),

- A *direct* problem might be: What was the profit/loss in 1999?
  Answer: -2000
- A *reverse* problem might be: In what year(s) (if any) did the business return $+5\,000$?
  Answer: 1998, 2001, 2005.

input-output pair
input ruler
output ruler
link

We will see that *direct* problems are usually easy to solve but, as might be expected, it is solving *reverse* problems, which is what solving "equations" is all about, that matters most in the real world.

EXAMPLE 2.15. Solving the *direct* problem of how much parking time three quarters will buy you is easy: just put three quarters in the parking meter and see how much parking time you get!
But in the real world, what we need to solve is the *reverse* problem of, when we want, say, two hours parking time, figuring how many quarters we need to put in the parking meter.

6. Given a function, an **input - output pair** is an input together with *the* (there can be at most one) output that the function returns for the input. It is standard to write input-output pairs within parentheses with a comma to separate the input from the output: (input, output).

EXAMPLE 2.16. Given the business in Example 2.13 (Page 53),

- (1998, $+5000$) and (2002, -2000) are input-output pairs,
- (1999, +3000) is *not* an input-output pair because the table does *not* pair 1999 with $+3\,000$,
- There is no input-output pair involving 2000
- There is no input-output pair involving $+3\,000$

3 Picturing Input-Output Pairs

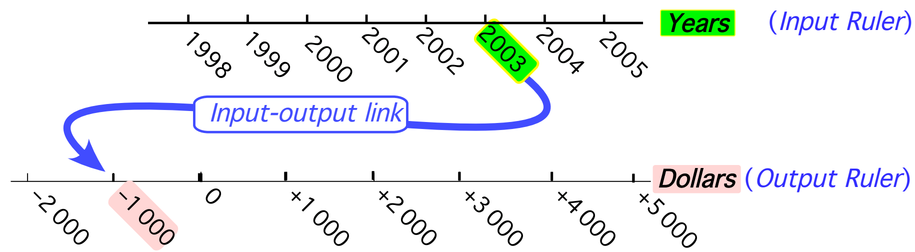
Given a function, we will often want to picture input-output pairs.

1. A simple-minded way to picture an input-output pair would be to:

- Tickmark the input on a *quantitative* **input ruler** as in section 10, (Page 26),
- Tickmark the output on a *quantitative* **output ruler** as in section 10, (Page 26),
- Draw an input-output **link** from the input on the input ruler to the output on the output ruler.

plot
Cartesian setup
screen

EXAMPLE 2.17. The *input-output* pair (Year 2003 , $-1 000) in Example 2.13 (Page 53) could thus be pictured as follows:



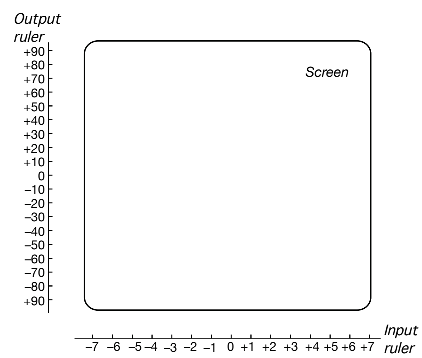
Obviously, though, picturing input-output pairs that way is not going to work very well with more than a very few input-output pairs.

2. So, in order to **plot** input-output pairs, we will use:

A. A *quantitative* **Cartesian setup**, that is:

- A rectangular area which we will call **screen**.
- A *quantitative* input ruler placed horizontally below the screen
- A *quantitative* output ruler placed vertically left of the screen

EXAMPLE 2.18.



B. The following

axis
histogram
bar graph

PROCEDURE 2.1 To get the plot point for an input-output pair

i. Tickmark the input on the input ruler,
ii. Draw an input **level line**, that is a *vertical* line through the input,
iii. Tickmark the output on the output ruler,
iv. Draw an output level line, that is a *horizontal* line through the output,
v. Then use:

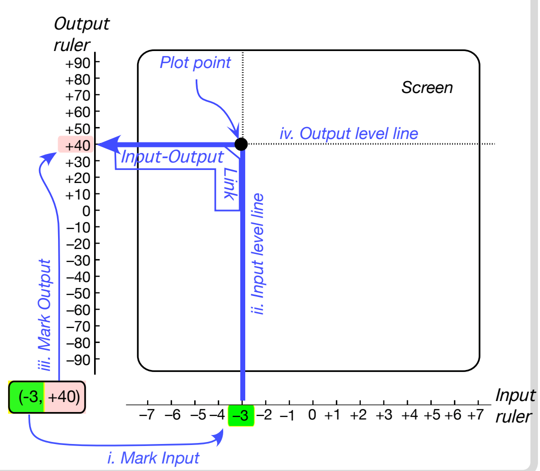
- A **solid dot** to indicates that the intersection of the input level line and the output level line *is* a **plot point**, (The input-output link then goes from the marked input to the plot point to the marked output.)

or, as we wll need occasionally,

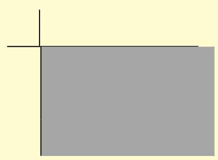
- A **hollow dot** to indicates that the intersection of the input level line and the output level line *is not* a plot point.

TEMO 2.1 Plot the input-output pair (-3, $+40$),

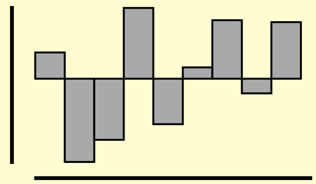
i. We tickmark the *input* -3 on the *input ruler*,
ii. We draw the *input level line* through -3,
iii. We tickmark the *output* $+40$ on the *output ruler*,
iv. We draw the *output level line* through $+40$,
v. We plot the intersection of the input level line with the output level line.
(The plot point is the elbow of the *input-output link*)



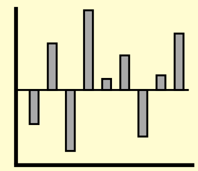
LANGUAGE 2.1 This setup is *not* the one used in most *textbooks* but in the real world it is *standard practice* to keep the rulers out of the way:



Table



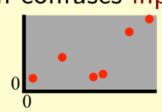
Histogram



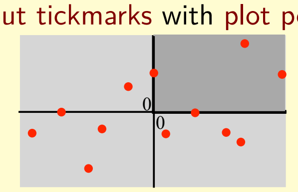
Bar graph

What happened is that Descartes, the inventor of ANALYTIC GEOMETRY, did not believe in *negative numbers* so when he used the 0-level lines as his rulers *they* were (almost) out of the way as usual.

But when mathematicians accepted *negative numbers*, they kept the 0-level lines as rulers even though *these* were now in the middle of the picture which confuses input and output tickmarks with plot points:



Descartes plot



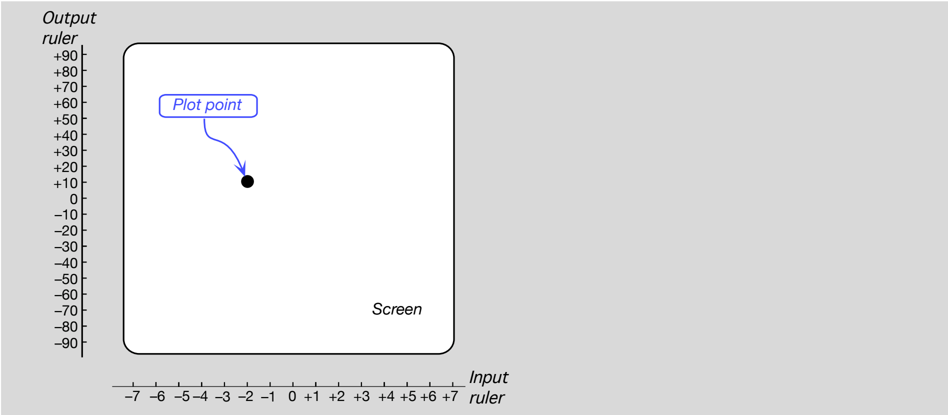
Modern plot

4. From a plot point, we can get back the input-output pair using:

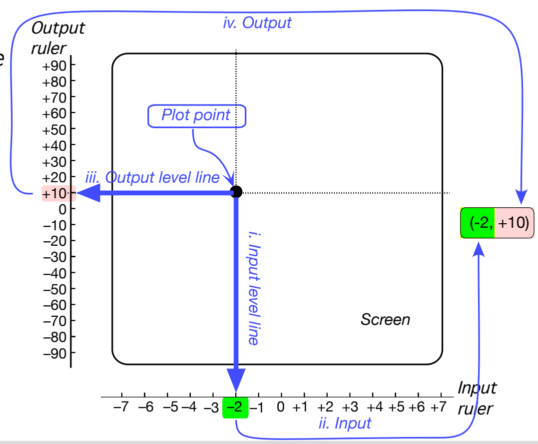
PROCEDURE 2.2 To get the input-output pair from a plot point

i. Draw an input level line through the plot point,
ii. The input is where the level line intersects the input ruler,
iii. Draw an output level line through the plot point,
iv. The output is where the level line intersects the output ruler.

TEMO 2.2 Get the input-output pair from the plot-point curve



i. We draw the *input level line* through the *plot point*,
ii. The input is where the input level line intersects the input ruler: -2
iii. We draw the *output level line* through the given plot point,
iv. The output is where the output level line intersects the output ruler: $+10$



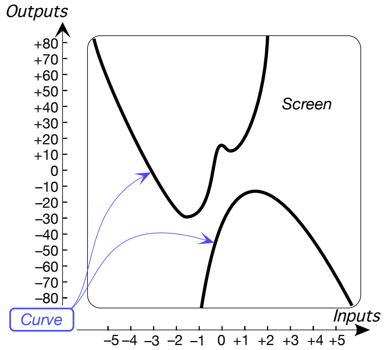
4 Functions Specified By A Global Graph

On the *experimental* side of *engineering* and the *sciences*, relations are also often specified by a **curve** drawn across a screen by some instrument.

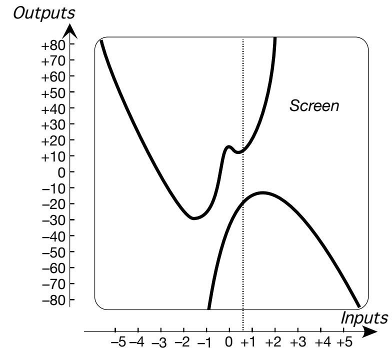
1. These relation though are *not* necessarily functions because there might very well be input level lines with more than one intersection with the curve.

graph

Example 2.19. Given the curve



the input-level lines for inputs between -1 and $+2$ intersect the curve in more than one point:



So the curve does *not* specify a *function*

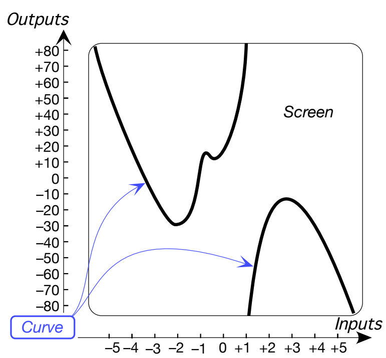
2. But if it so happens that the curve meets the

Definition 2.2 (Restated) Functional requirement

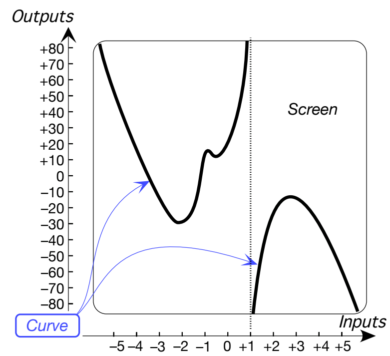
No input level line intersects the curve in more than *one* point.

then the curve *will* specify a function and we will say that the curve is the **graph** of that function.

Example 2.20. Given the curve



no input-level line intersects the curve in more than one point:



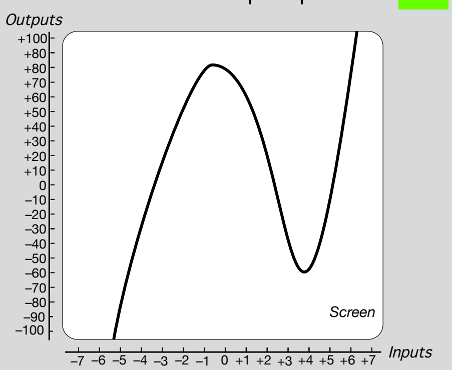
So the curve *specifies* a *function*

3. When a function is specified by a graph, we get the plot point at a given input using:

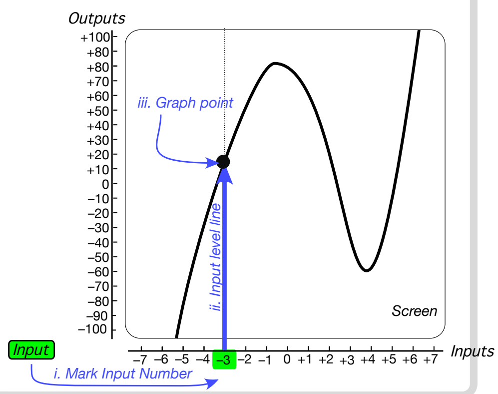
PROCEDURE 2.3 To get the plot point at x_0 for a function specified by a *global graph*

i. Tickmark the given input on the input ruler,
ii. Draw the input level line through the given input,
iii. The plot point is the intersection of the input level line with the graph,

TEMO 2.3 Get the plot point *at* -3 for the function specified by



i. We tickmark the input -3 on the *input ruler*,
ii. We draw the *input level line* through -3
iii. The *plot point* is at the intersection of the *input level line* with the *graph*,

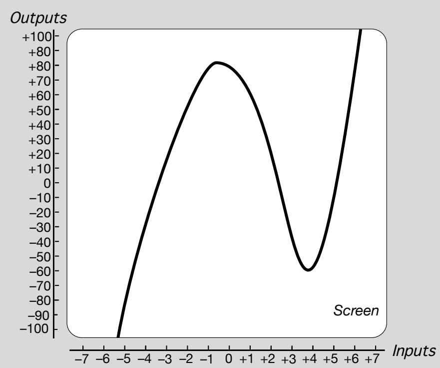


4. When a function is specified by a global graph, then, for a given input, we get the output using:

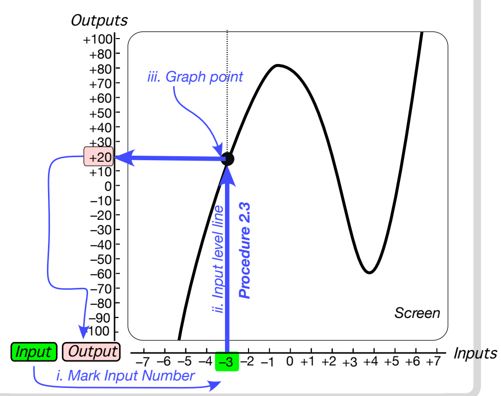
PROCEDURE 2.4 To get the output at x_0 for a function specified by a global graph.

i. Tickmark the giveninput on the input ruler,
ii. Get the plot point with **??** (**??**)
iii. Draw the output level line through the plot point,
iv. The output is at the intersection of the output level line with the output ruler.

TEMO 2.4 Get the output *at* -3 for the function specified by



i. We tickmark the input -3 on the *input ruler*,
ii. We use **??** to get the *plot point*,
iii. We draw the *output level line* through the *plot point*,
iv. The *output* is at the intersection of the *output level line* with the *output ruler*: $+20$



5 Functions Specified By A Global I-O Rule

Nevertheless, both in *engineering* and the *sciences*, the name of the game is functions specified “mathematically” rather than by tables or curves.

1. Functional symbols. The following is completely standard:

x
unspecified input
$f(x)$
unspecified output
$\xrightarrow{f}$
arrow notation
Reverse Polish Notation
explicit function
output-specifying code
global input-output rule

a. We will use $\boldsymbol{f}$ as name of a generic function.
b. We will use:

- $\boldsymbol{x}$ as an **unspecified input** which is like an empty box waiting for *us* to *specify* the input,

and therefore

- $\boldsymbol{f(x)}$, to be read f of x, as the **unspecified output** which is like an empty box waiting for the *function* to *return* the output.

EXAMPLE 2.21. Say JOE is the name of our favorite parking meter. Then x represents the slot waiting for us to put the coins and $JOE(x)$ represents the display where the parking time that JOE will give us in return for our coins will appear.

c. We will then use $\xrightarrow{f}$ to write the so-called **arrow notation**

$$x \xrightarrow{f} f(x)$$ [3]

EXAMPLE 2.22. In Example 3.11 we can write the arrow notation

$$x \xrightarrow{JOE} JOE(x)$$

2. The first of the two "mathematical" ways *engineers* and *scientists* use to specify a function is the way used for the functions to be investigated in *this* volume, namely:

> **DEFINITION 2.3 Explicit Functions** are functions specified by a **global input-output rule** in which $f(x)$ is specified in terms of x by some **output-specifying code**:
>
> $$\underbrace{x}_{\text{Unspecified input}} \xrightarrow{f} \underbrace{f(x)}_{\text{Unspecified output}} = \underbrace{\text{Code that specifies } f(x) \text{ in terms of } x}_{\text{Output-specifying code}}$$

[3]Thus, $x\,f$, known as the **Reverse Polish Notation** for the *output*, would be much better *code* than $f(x)$ because:

i. In the arrow notation, x would be ahead of f in *both* places: $x \xrightarrow{f} x\,f$.

ii. Not to mention that $x\,f$ requires *no* parentheses.

(https://en.wikipedia.org/wiki/Reverse_Polish_notation) Unfortunately, even Hewlett Packard was eventually forced to abandon the Reverse Polish Notation.

declare

(The reason we have to say "global" is that later we will have to distinguish *global* input-output rules from "local" input-output rules.)

EXAMPLE 2.23. In the *global input-output rule*

$$x \xrightarrow{JILL} JILL(x) = \frac{-2.71(x+54.23)}{-5.68x^3+217.43},$$

the output-specifying code is $\frac{-2.71(x+54.23)}{-5.68x^3+217.43}$

From now on,

> **AGREEMENT 2.2** **Function** will be short for explicit function but only in *this* volume.

6 Declaring Inputs

Since explicit functions involve output-specifying *code*, our first step in inputting something will always be, as programmers do and no matter what the something, to **declare** the something by writing the declaration

$$\Big|_{x \leftarrow \text{something}}$$

to the right of anything in the arrow notation involving x.

EXAMPLE 2.24. Say JOE is the parking meter in Example 3.14 so that the arrow notation is as in Example 4.15:

$$x \xrightarrow{JOE} JOE(x),$$

To declare that we put 3 Quarters in the *slot*, we write the declaration

$$\Big|_{x \leftarrow \text{3 Quarters}}$$

to the right of anything in the arrow notation that involves x:

$$x\Big|_{x \leftarrow \text{3 Quarters}} \xrightarrow{JOE} JOE(x)\Big|_{x \leftarrow \text{3 Quarters}}$$

Then, after the replacement is done, we have

$$\text{3 Quarters} \xrightarrow{JOE} JOE(\text{3 Quarters})$$

where $JOE(\text{3 Quarters})$ stands for what shows on the *display*.

The something can be a givable number x_0 but most of the time, the something will be a *neighborhood* of a point.

=========OK SO FAR=========

1. Given a function f, in order to input a givable number x_0, we declare that the unspecified input x is to be replaced by the givable number x_0 which we do by writing the declaration

$$\Big|_{x \leftarrow x_0}$$

to the right of anything in the arrow notation that involves x:

$$x\Big|_{x \leftarrow x_0} \xrightarrow{\quad f \quad} f(x)\Big|_{x \leftarrow x_0}$$

Then, after the replacement has been done, we have

$$x_0 \xrightarrow{\quad f \quad} f(x_0)$$

EXAMPLE 2.25. Given the function $FRAN$, in order to declare the givable number -31.76 we write

$$x\Big|_{x \leftarrow -31.76} \xrightarrow{\quad FRAN \quad} FRAN(x)\Big|_{x \leftarrow -31.76}$$

Then, after the replacement has been done, we have

$$-31.76 \xrightarrow{\quad FRAN \quad} FRAN(-31.76)$$

2. Given a function f, in order to input a *neighborhood* of a givable number x_0, we will declare the *generic* nearby number $x_0 \oplus h$ which we do by writing the declaration

$$\Big|_{x \leftarrow x_0 \oplus h}$$

to the right of anything in the arrow notation that involves the unspecified input x:

$$x\Big|_{x \leftarrow x_0 \oplus h} \xrightarrow{\quad f \quad} f(x)\Big|_{x \leftarrow x_0 \oplus h}$$

so that, after the replacement has been done, we have

$$x_0 \oplus h \xrightarrow{\quad f \quad} f(x_0 \oplus h)$$

result
y_0

EXAMPLE 2.26. Given the function $MIKE$, to thicken the *given* input -31.76, we declare the *actual* input $-31.76 \oplus h$

$$x\Big|_{x \leftarrow -31.76 \oplus h} \xrightarrow{MIKE} MIKE(x)\Big|_{x \leftarrow -31.76 \oplus h}$$

Then, after the replacement has been done, we have

$$-31.76 \oplus h \xrightarrow{MIKE} MIKE(-31.76 \oplus h)$$

3.

4.

7 Returned Outputs

Given a function f, after we have declared whatever thing we want to input, the output-specifying code will usually return something. (But *not* necessarily, see Note 2.1, page 53.)

1.

qualitative Cartesian setup

Given a function f, after we have declared the given number x_0, the output-specifying code will usually return a **resulting** number. (But *not* necessarily, see Note 2.1, page 53.)

Using $\boldsymbol{y_0}$ as generic code for a resulting number, we can write:

$$x_0 \xrightarrow{f} f(x_0) = y_0$$

or just

$$x_0 \xrightarrow{f} y_0$$

which, inasmuch as it involves the name of the function, is a more precise way to write the input-output pair

$$\big(x_0, y_0\big)$$

one which we will use especially when there is more than one function involved in a situation.

EXAMPLE 2.27. Say JOE is the function in Example 3.14 so that the arrow notation is as in Example 4.15

$$x \xrightarrow{JOE} JOE(x),$$

and that we put 3 Quarters in the *slot* as in Example 2.25 so that we have:

$$\text{3 Quarters} \xrightarrow{JOE} JOE(\text{3 Quarters})$$

Now say JOE showed 45 Minutes on the display. (The internal mechanism of the parking meter is the real world equivalent of the *output-specifying code*.) Then we would write

$$\text{3 Quarters} \xrightarrow{JOE} JOE(\text{3 Quarters}) = \text{45 Minutes}$$

or just

$$\text{3 Quarters} \xrightarrow{JOE} \text{45 Minutes}$$

which, inasmuch as it involves the name of the parking meter, is a more precise way to write the input-output pair

$$\Big(\text{3 Quarters}, \text{45 Minutes}\Big)$$

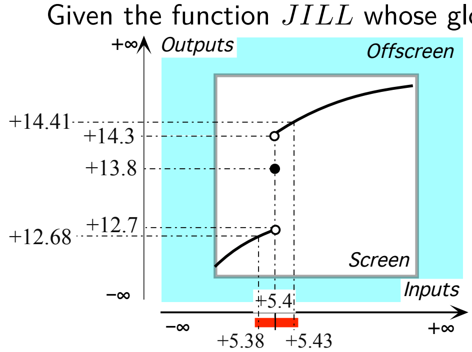
one which we would use when, say, comparing different parking meters.

2. Given a function f, after we have thickened the given number x_0, the output can turn out to be quite complicated. In fact, dealing with $f(x_0 \oplus h)$ is going to be a major part of our investigations. This is because $f(x_0 \oplus h)$ is going to depend on h and *what* the output-specifying code is going to do with h is going to depend very much on the kind of function f is.

is sometimes used as a short for outputs for nearby inputs but doing so would risk being extremely *misleading* because

NOTE 2.2 Nearby outputs are *not* necessarily near the output for the given input.

EXAMPLE 2.28. Given the function $JILL$ whose global graph is,



graph
onscreen graph
offscreen graph

- For nearby inputs *left* of $+5.4$ $JILL$ returns outputs near $+12.7$. For instance, $JILL(+5.38) = +12.68$
- For nearby inputs *right* of $+5.4$ $JILL$ returns outputs near $+14.3$. For instance, $JILL(+5.43) = +14.41$

and neither $+12.68$ nor $+14.41$ are near $JILL(+5.4) = +13.8$

So:

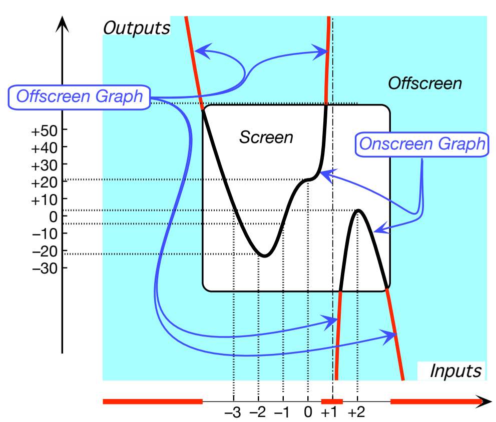
AGREEMENT 2.3 **Nearby output** will *never* be used in this text.

=======Begin WORK ZONE======= **graph** point will be used with qualitative Cartesian setup and global graph for qualitative graphs that include large inputs and small inputs

3. Of course, even though *we* can only draw on the *screen*, there are inputs too *large* to be drawn on the screen so that the global graph of a function really consists of two parts:

- The **onscreen graph** which is the part of the global graph that ... shows on the screen,
- An **offscreen graph** which is the part of the global graph that ... does not show on the screen and for which we reserved the offscreen space.

EXAMPLE 2.29.



8 Onscreen Graph

The onscreen graph involves the medium inputs for which the function returns medium outputs

=======End WORK ZONE========

9 Functioning With Infinity

Obviously, because The tolerance is a *plain* decimal number (Page 8),

NOTE 2.3 ∞ can neither:
i. be *declared* as an input in output-specifying code,

nor

ii. result as an output from output-specifying code.

However this is *not* going to cause us any trouble because we will be able to thicken ∞ to *large* with

THEOREM 2.1 *large*
i. x can be declared to be *large*
ii. *large* can result from output-specifying code.

EXAMPLE 2.30. Let REC be the function specified by the global input-output rule:

$$x \xrightarrow{REC} REC(x) = Reciprocal\,of\,x$$

Then we get from Theorem 1.2 Reciprocal of qualitative sizes (Page 38):

input	output
large	*small*
small	*large*

.

10 Computing Input-Output Pairs

1. We can now show how we get input-output pairs using a global input-output rule:

execute
decode
perform
format

Procedure 2.5 To get the output at the given input x_0

i. Declare the unspecified input x to be the given number x_0

$$x\Big|_{x\leftarrow x_0} \xrightarrow{f} f(x)\Big|_{x\leftarrow x_0} = \underbrace{\text{Code specifying } f(x) \text{ in terms of } x}_{\text{Output-specifying code}}\Bigg|_{x\leftarrow x_0}$$

so that, after the replacement is done, we have

$$x_0 \xrightarrow{f} f(x_0) = \underbrace{\text{Code specifying } f(x_0) \text{ in terms of } x_0}_{\text{Output-specifying code}}$$

ii. Execute the output-specifying code that is:

a. Decode the output-specifying code which means write out the computations specified by the output-specifying code.

b. Perform the computations specified by the output-specifying code and thus get the number y_0 which $f(x_0)$ is.

iii. Format the *input-output pair* according to the purpose:

- For *computational* purposes, use the equality
$$f(x_0) = y_0$$
- For *graphic* purposes, use the pair:
$$(x_0, y_0)$$
- For *conceptual* purposes, use the arrow notation:
$$x_0 \xrightarrow{f} y_0$$

Temo 2.5 Get the output *at* -3 for the function specified by $x \xrightarrow{JACK} JACK(x) = -4x \oplus +7$

i. We *declare* that x is to be replaced by -3:

$$x\Big|_{x\leftarrow -3} \xrightarrow{JACK} JACK(x)\Big|_{x\leftarrow -3} = -4x+7\Big|_{x\leftarrow -3}$$

so that, after the replacement is done, we have

$$-3 \xrightarrow{JACK} JACK(\text{-}3) = \underbrace{-4(-3)\oplus +7}_{\text{output-specifying code}}$$

ii. We *execute* the output-specifying code that is:

a. We *decode* the output-specifying code which says to multiply the number -4 by a copy of the input -3 and then to $\oplus$ the number $+7$

$$= -4(-3) \oplus +7$$

b. We *perform* the computations:

$$= +12 \oplus +7$$
$$= +19$$

iii. Format the input-output pair:

- For computational purposes: $JACK(-3) = +19$
- For *graphic* purposes: $(-3, +19)$
- For *conceptual* purposes: $-3 \xrightarrow{JACK} +19$

2. Graphically, we can then use

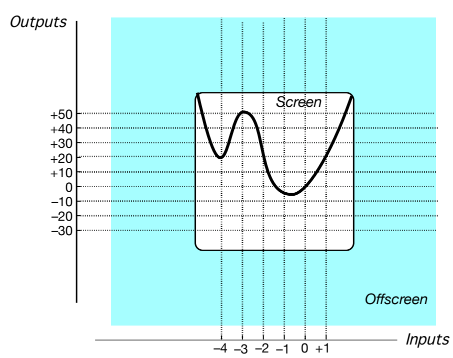
PROCEDURE 2.6 To get the plot point at a given input for an algebraic function

1. Get the output returned for the given input by the function with **??** (**??**),
2. Get the *plot point* for the input-output pair with **??** (**??**)

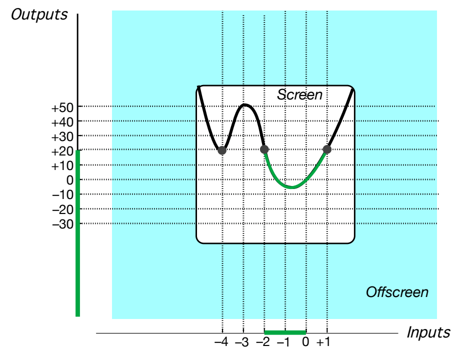
11 Fundamental Problem

Our overall goal in this text will be, roughly speaking, the investigation of various, very different, ways that functions can return outputs. But it is often useful to *see* on a global graph those inputs for which the function will return outputs that meet some requirement we are interested in.

EXAMPLE 2.31. Given the function $MILT$ specified by the global graph



find the inputs whose *output* is less than $+20$. From the onscreen graph, join



we see that the answer is "All *inputs* between -2 and $+1$"

So, in fact, we will devote quite a bit of time and energy to the

> **Definition 2.4 Fundamental Problem** *To get the global graph of a function specified by a global input-output rule.*

12 Joining Plot Points

Indeed, solving the **Fundamental Problem** is almost never a simple matter because declaring given inputs can almost never get us a global graph any more than given numbers can specify an amount of stuff. Yet, chances are you were once told that to get the global graph of a function specified by a global input-output rule, you "just" had to:

i. Declare a few inputs and compute the outputs returned by the function for these inputs.

ii. Plot these input-output pairs,

iii. Join the plot points.

However, this so-called "procedure" is in fact *total garbage* which we therefore have to "dispose of properly":

1. Narrow mindedness To begin with, this so-called "procedure" cannot possibly get us the offscreen graph since the only input-output pairs we can plot are those for medium inputs as medium inputs are the only inputs we can declare in a global input-output rule. Which is already regrettable since just because something is offscreen doesn't mean it is not interesting.

EXAMPLE 2.32. "*Many ancient civilizations collected astronomical information in a systematic manner through observation.*" See https://en.wikipedia.org/wiki/History_of_science

But what is *most* regrettable is that much of what happens onscreen is *caused* by what happens offscreen.

EXAMPLE 2.33. Even though what happens *on earth* is what we are immediately concerned with, much of what happens on earth depends on what happens *very far away*: tides are due to the pull of the *moon* and all the energy we use originates, one way or the other, from the *sun* and life on earth would cease instantly if the sun were to black out.

So:

Question i. *How do we know* what's onscreen is all there is to see?[4]

This is in fact a complicated question which we will address in Chapter 4 Features Near ∞.

2. Incomprehensibility But this so-called "procedure" is not likely to get us the onscreen graph either because of three additional questions[5]:

Question ii. *How do we know* which medium inputs we are to declare in the global input-output rule?

Question iii. *How do we know* which way to join the plot points?

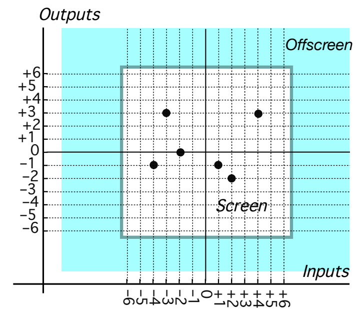
Question iv. *How do we know*, after we have somehow joined whatever plot points we somehow got, if the curve we get *is* the onscreen graph?

EXAMPLE 2.34. Given a function specified by some *global input-output rule*, suppose we somehow got the following input-output pairs and therefore the *plot*:

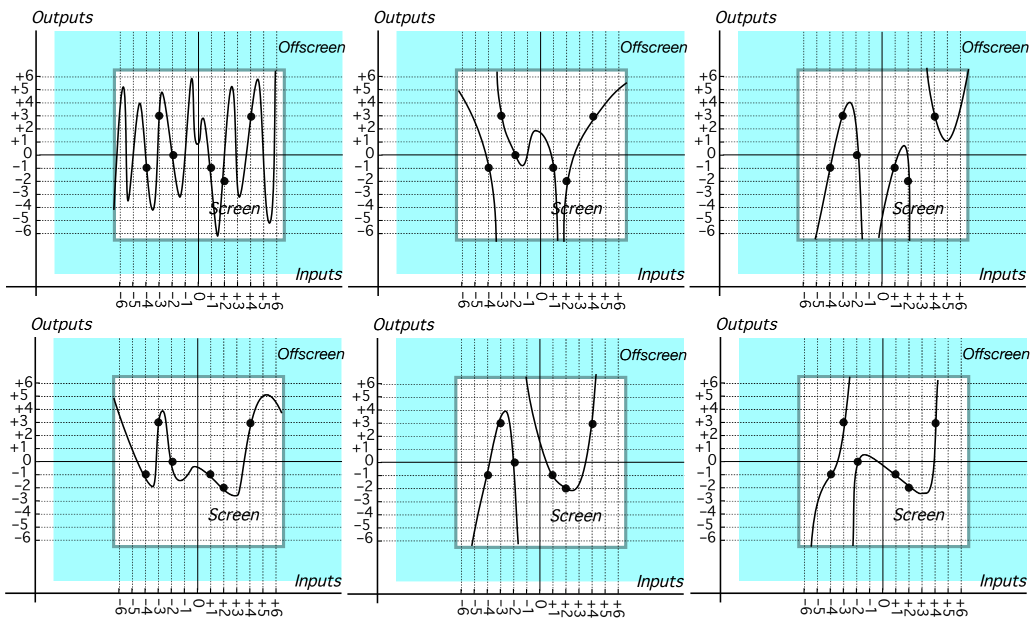
[4]Which Educologists do not seem to wonder about or even be aware of.

[5]Educologists have much to answer for never even raising these questions.

Inputs	-4	-3	-2	$+1$	$+2$	$+4$
Outputs	-1	$+3$	0	-1	-2	$+3$



Which of the following would you then say is the *onscreen graph* of the function:



As Example 2.34 demonstrates the answers to the above questions are:

Question ii. *How do we know* which medium inputs we are to declare in the global input-output rule?

Answer: At the very least, *which* inputs we declare will *have* to depend on the nature of the particular function we are trying to graph,

Question iii. *How do we know* which way to join the plot points?

Answer: Other than very exceptionally, there cannot possibly be a set way to join smoothly a plot,

Question iv. *How do we know*, after we have somehow joined whatever plot points we somehow got, if the curve we get *is* the onscreen graph?

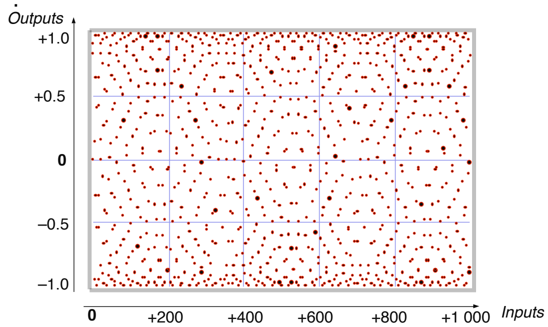
Answer: On the basis of only a number of plot points, there is no way we can decide what the global graph is going to look like.

As always in the real world, recreating an *analog entity* (a global graph) from a *discrete sampling* (a plot) is nowhere near simple.

EXAMPLE 2.35. Ask a sound engineer: how do you recreate from, say, a CD (*discrete sampling*) a music performance (*analog signal*)?

3. At this point, we were usually told "just get more plot points" but too many plot points can in fact make it impossible to join smoothly.

EXAMPLE 2.36. The function $SINE$ belongs to the next volume, TRANSCENDENTAL FUNCTIONS, but the point here is **Strang's Famous Computer Plot** of $SINE$ [6]:



How are we to "join smoothly"?

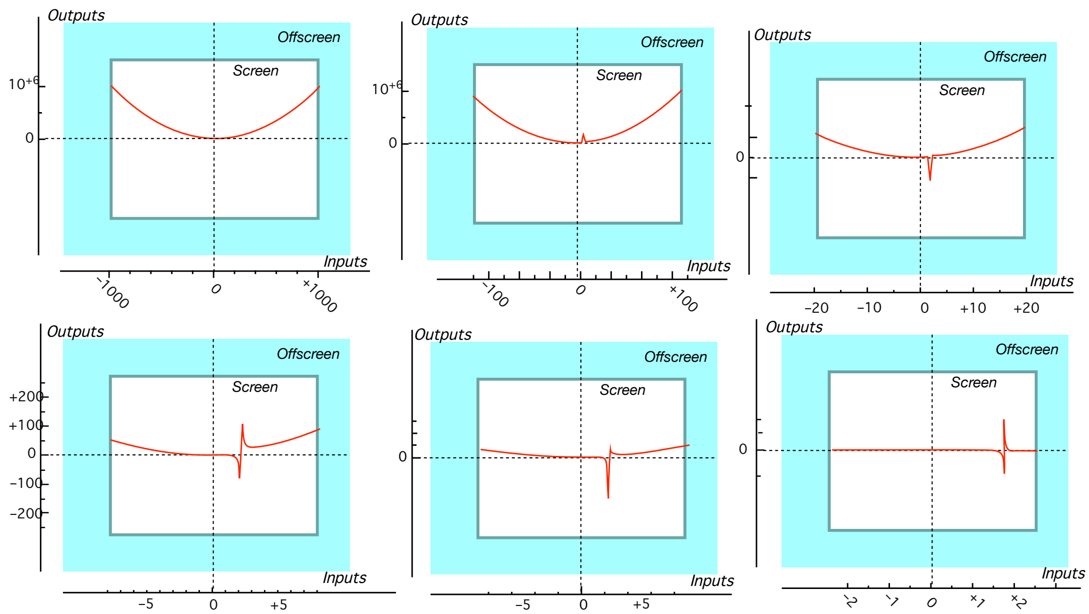
And even computer generated graphs cannot always be taken at face value.

EXAMPLE 2.37. Given the function specified by the global input-output rule

$$x \xrightarrow{CAT} CAT(x) = \frac{x^3 - 1}{x - 2}$$

which of the following *computer generated* graphs is the right one?

[6] The plot appears on the back cover of Strang's *Calculus*, 1991, Wellesley-Cambridge Press, where it is discussed in Section 1.6 A Thousand Points of Light, pages 34-36.



4. Since we cannot rely on declaring Inputs (Page 64) to get the global graph of a function specified by a global input-output rule, we will have to develop:

i. What to use instead of plot points to get an onscreen graph which we will do in Chapter 3 Features Near x_0

ii. How to get the offscreen graph, which we will do in Chapter 4 Features Near ∞,

iii. How to put all this together to get a global graph which we will do in Chapter 5 Global Analysis

In those three chapters, though, our goal will only be:

- To *introduce* and *discuss* graphically the necessary *concepts* and
- To provide the reader with the means for *picturing* the "why" and the "how" of the *computations* we will need to do later when we investigate given algebraic functions.

Then, with **?? ??**, we will finally start on our systematic investigation of increasingly complicated algebraic functions in which, of course, we will get their global graph.

In a crime novel, the victim is not the story. The story is *around* the victim.

Anonymous crime writer

input level band

Chapter 3

Features Near x_0

You may recall that:

i. We saw in **?? ??**that when we specify in the real world a given amount of stuff x_0, the number we get when we measure the actual amount of stuff will always differ from x_0 by an error h so that the actual amount of stuff is described by $x_0 + h$

ii. We saw in section 10 that just getting $f(x_0)$ can almost never get us any information about $f(x_{0+h})$.

So, given a function f and given an input x_0, we now have two reasons for wanting to thicken the given input x_0 into a neighborhood of x_0, that is into $x_0 + h$ and then investigating $f(x_0 + h)$.

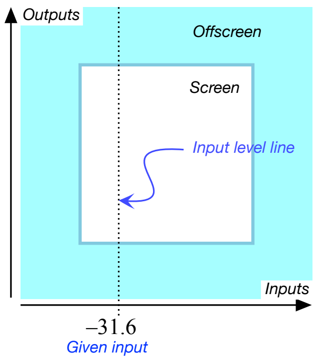
1 Local Place

Thickening input numbers into input neighborhoods implies that we first need to do a thick equivalent of picturing Input-Output Pairs (Section 3, page 55.)

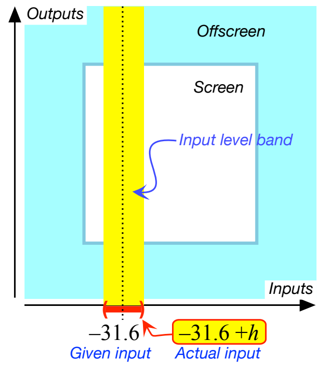
1. We will thicken input level lines into **input level bands**, that is into vertical bands through the input neighborhoods.

output neighborhood
output level band

EXAMPLE 3.1. We will thicken the input level line



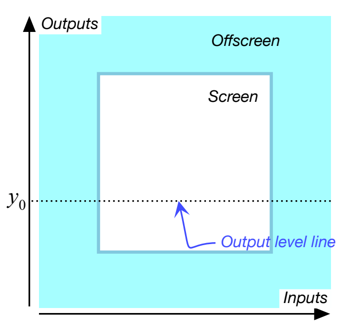
into the input level band



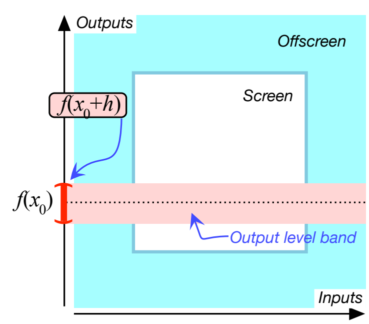
2. On the other hand, we won't always be able to thicken an output into an **output neighborhood** because it is the *function* which returns the nearby outputs and $f(x_0 + h)$ is not necessarily going to be near $f(x_0)$ (nearby outputs are not necessarily going to be near the resulting output) which, in fact, may not even exist. See note 2.1 We will discuss this in Section 8 Smoothness near ∞. (Page 131)

But should we somehow know that $f(x_0 + h)$ is near $f(x_0)$ (that is that nearby outputs are near the resulting output), then we will thicken the output level line into an **output level band** that is a horizontal band through the neighborhood of $f(x_0)$.

EXAMPLE 3.2. We will thicken the output level line



into the output level band



3. Since a plot point is at the intersection of an input level line and an output level line, we will thicken a plot point into a **local graph place**, that is into the rectangle at the intersection of an input level band near x_0 and an output level band near $f(x_0)$.

local graph place
sided local graph place

=======Begin WORK ZONE=======

However, inasmuch as we will usually deal separately with each one of the two

??

(**??**),

we will usually know which side of the input is linked to which side of the output

=======End WORK ZONE========

and the **sided local graph place** will then consist of two smaller rectangles, one on each side of the input level line. To get a sided local graph place then,

we just thicken

PROCEDURE 3.1 To get the sided local graph place for an input-output pair knowing which side of the input neighborhood is paired with which side of the output neighborhood.

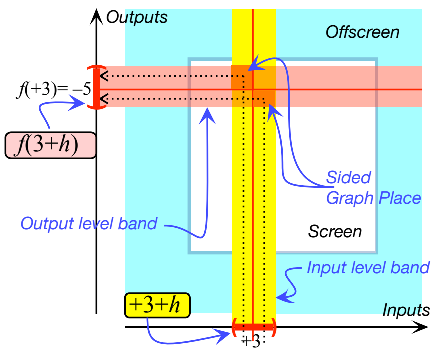
i. Mark a *neighborhood* of the input on the input ruler,
ii. Draw the *input level band*,
iii. Mark a *neighborhood* of the output on the output ruler,
iv. Draw the *output level band*,
v. Mark which side of the input neighborhood is linked to which side of the output neighborhood,
vi. The place for the given input-output pair is at the intersection of the corresponding *sides* of the level bands.

TEMO 3.1 Get the sided place for ($+3$, -5) given that:

- $+3^{-} \longrightarrow -5^{+}$
- $+3^{+} \longrightarrow -5^{-}$

local graph near x_0

i. We mark a *neighborhood of* $+3$ on the *input ruler*,
ii. We draw the *input level band* through the *neighborhood* of $+3$,
iii. We mark a *neighborhood of* -5 on the *output ruler*,
iv. We draw the *output level band* through the *neighborhood* of -5,
v. Mark: • left of $+3$ $\to$ above -5
• right of $+3$ $\to$ below -5



vi. The *sided graph place* for ($+3$, -5) is at the intersection of the corresponding *sides* of the level bands.

2 Local graph

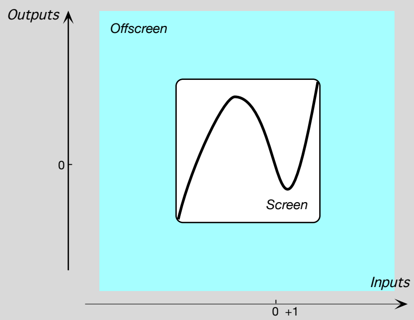
In some cases , depending on the kind of information we want, we will be able to get this information from the local graph *place* but in most cases we will need the **local graph** near x_0, that is the part of the global graph in the local graph place. To get the local graph near a bounded input then, we just thicken **??** (**??**):

PROCEDURE 3.2 To get the local graph near x_0 of a function specified by a global graph

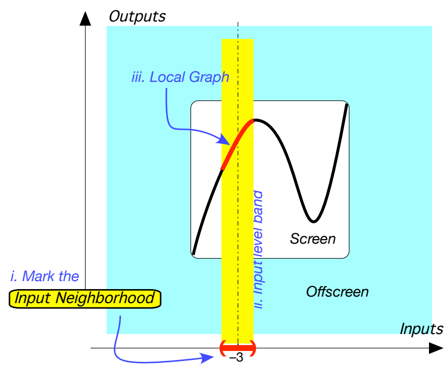
i. Mark a neighborhood of x_0 on the input ruler,
ii. Draw the input level band through the neighborhood of x_0,
iii. The local graph near x_0 is the intersection of the input level band with the global graph.

TEMO 3.2 Get the local graph near -3 of the function whose global graph is

⟨
⟩
local code
angles
,



i. We mark a *neighborhood of* -3 on the *input ruler*,
ii. We draw the *input level* band through the *neighborhood* of -3,
iii. The *local graph* near -3 is the intersection of the input level *band* with the global graph,



3 Local code

In order to describe *separately* what happens on each side of the given input, we will need:

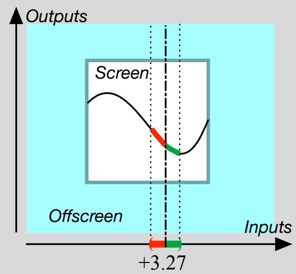
Definition 3.1 Local Code near x_0
i. We will use a pair of **angles**, ⟨ ⟩, to stand for the input neighborhood with a comma **,** between ⟨ and ⟩ to stand for x_0.
ii. We will have to *face* x_0 when coding local features.
iii. Then, the code that will record the local feature for

nearby inputs that are	nearby inputs that are
left of x_0	*right* of x_0
will go	will go
left of **,**	*right* of **,**

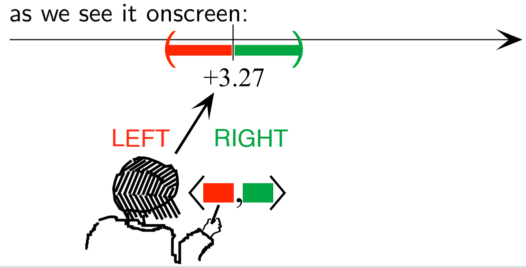
$\langle \, \blacksquare \, , \, \blacksquare \, \rangle$

height
Height-sign

TEMO 3.3 Set up for the local code to record the local behavior near $+3.27$



Since the local graph is near a *bounded* input, we are facing it and we will *code* the local feature as we see it onscreen:

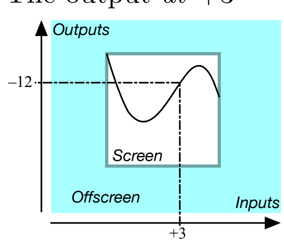


4 Local Height

Given a function f and a given input x_0, we will thicken the output *at* x_0 into the **height** *near* x_0. As the use of the word "near" indicates, the height is a *local* feature and we will occasionally remind the reader of that by saying "local height" instead of just "height".

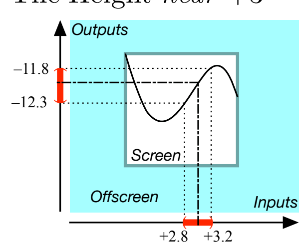
EXAMPLE 3.3.

The output *at* $+3$



is -12

The Height *near* $+3$



is $-12 \pm$ *small*

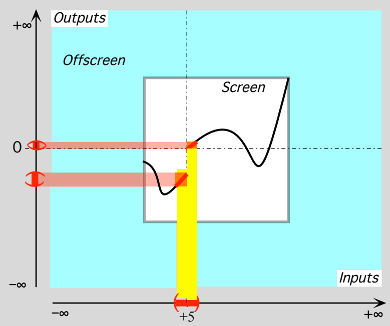
1. The **Height-sign** of f near x_0 is the sign, $+$ or $-$, of the outputs for nearby inputs on each side of the given input.

height-size

PROCEDURE 3.3 To get the Height-sign near a given input of a function from its global graph,

i. Get from the local graph the sign, $+$ or $-$, of the *outputs* for nearby inputs on each side of the given input,
ii. Code Height-sign f according to Definition 3.1 (Page 81)

TEMO 3.4 Get Height-sign near $+5$ for the function IAN from the local graph near $+5$



i. We get from the local graph the sign of the *outputs* for nearby inputs on each side of $+5$:

- The sign of the outputs *left* of $+5$ is $-$
- The sign of the outputs *right* of $+5$ is $+$

ii. We code the Height-sign:

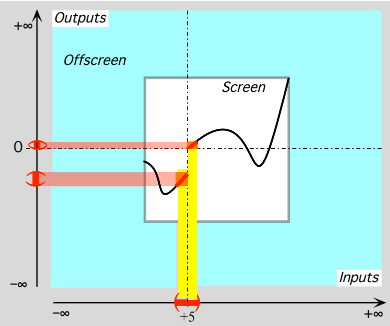
Height-sign IAN near $+5 = \langle -, + \rangle$

2. The **height-size** of f near a given input is the qualitative size, *large*, *bounded* or *small*, of the *outputs* for nearby inputs on each side of the given input.

PROCEDURE 3.4 To get the Height-size near a given input of a function from its global graph,

i. Get from the local graph the qualitative size, *large*, *bounded* or *small*, of the *outputs* for nearby inputs on each side of the given input,
ii. Code Height-size f according to Definition 3.1 (Page 81)

TEMO 3.5 Get Height-size near $+5$ for the function IAN from the local graph near $+5$



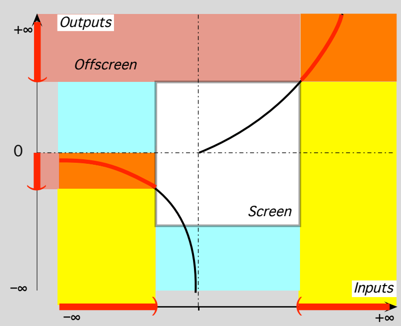
i. We get from the local graph the qualitative size, $large$, $bounded$ or $small$, of the *outputs* for nearby inputs on each side of $+5$:

- The size of the outputs *left* of $+5$ is $large$
- The size of the outputs *right* of $+5$ is $small$

ii. We code the Height-size:

Height-size IAN near $+5 = \langle large, small\rangle$

TEMO 3.6 Get Height-size near ∞ for the function IAN from the local graph near ∞



i. We get from the local graph the qualitative size, $large$, $bounded$ or $small$, of the *outputs* for nearby inputs on each side of ∞:

- The size of the height *left* of ∞ is $large$
- The size of the height *right* of ∞ is $small$

ii. We code the Height-size:

Height-size IAN near $\infty = \langle large, small\rangle$

5 Local extreme

We will often compare the *output at* a given *bounded* input with the *height near* the given *bounded* input.

1. A **local maximum-height input** is a *bounded* input whose output is *larger* than the height near the bounded input. In other words, the output *at* a local maximum-height input is *larger* than the outputs for all nearby inputs.

local maximum-height input
$x_{\text{maxi-height}}$
local minimum-height input

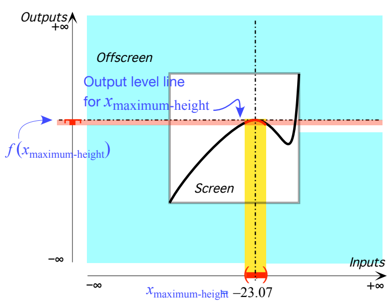
$$\boxed{x_0 \text{ is al local maximum-height input whenever } f(x_0) > f(x_0 + h)}$$

We will use $\boldsymbol{x}_{\textbf{max-height}}$ as a name for a local maximum-height input.

> **LANGUAGE 3.1** $\boldsymbol{x}_{\textbf{max}}$ is the usual name for a local maximum-height input but x_{max} tends to suggest that it is the input x that is *maximum* while it is the *output*, $f(x_{\text{max}})$, which is "maximum".

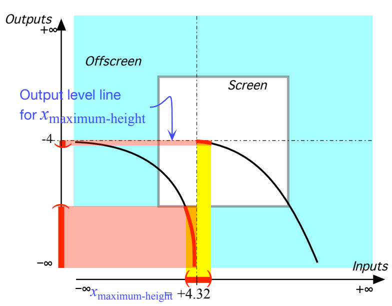
Graphically, the local graph near $x_{\text{max-height}}$ is *below* the output-level line for $x_{\text{max-height}}$.

EXAMPLE 3.4. The function



has a local *maximum* at -23.07 because the output *at* -23.07 is *larger* than the outputs for *nearby inputs*

EXAMPLE 3.5. The function



has a local *maximum* at $+4.32$ because the output *at* $+4.32$ is *larger* than the outputs for *nearby inputs*

2. A **local minimum-height input** is a *bounded* input whose output is *smaller* than the height near the given input. In other words, the

$x_{\text{min-height}}$
local extreme-height input

output *at* a local minimum-height input is *smaller* than the outputs for all nearby inputs.

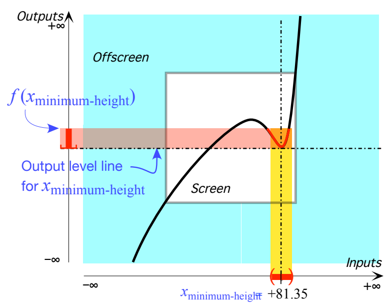
$$\boxed{x_0 \text{ is al local minimum-height input whenever } f(x_0) < f(x_0+h)}$$

We will use $\boldsymbol{x_{\text{min-height}}}$ as name for a local minimum-height input.

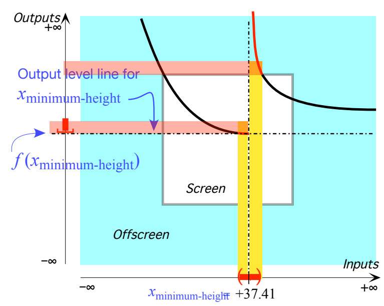
LANGUAGE 3.2 $\boldsymbol{x_{\text{min}}}$ is the usual name for a local minimum-height input but x_{min} tends to suggest that it is the input x that is *minimum* while it is its *output*, $f(x_{\text{min}})$, which is "minimum".

Graphically, the *local graph* near $x_{\text{min-height}}$ is *above* the output-level line for $x_{\text{min-height}}$.

EXAMPLE 3.6. The function has a local *minimum* at $+81.35$ because the output *at* $+81.35$ is *smaller* than the outputs for *nearby inputs*.



EXAMPLE 3.7. The function has a local *minimum* at $+37.41$ because the output *at* $+37.41$ is *smaller* than the outputs for *nearby inputs*.



3. Local extreme-height input are *bounded* inputs which are either a local maximum-height input or a local minimum-height input.

NOTE 3.1 Local extreme-height inputs can only be *bounded* inputs.

4. Minimization problems and maximization problems (https://en.wikipedia.org/wiki/Mathematical_optimization) as well as min-max problems (https://en.wikipedia.org/wiki/Minimax) are of primary importance in *real life*. So, zero parity even zero odd zero

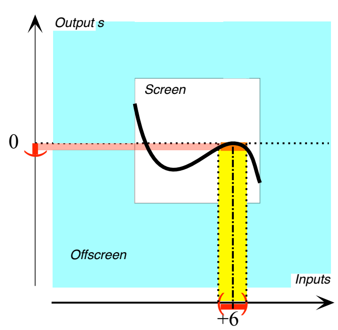
- It would be pointless to allow ∞ as a local extreme-height input since it cannot be reached in the *real world*,
- It would be meaningless to allow $+\infty$ as a locally largest output since $+\infty$ is *always* larger than any output or to allow $-\infty$ as a locally smallest output since $-\infty$ is *always* smaller than any output.

6 Zeros And Poles

1. Given a function f, a **zero** of f is a *bounded* input whose Height-size is $\langle small, small\rangle$. We will distinguish two kinds of zeros according to their **parity**:

- An **even zero** is a zero whose Height-sign is either $\langle +, +\rangle$ or $\langle -, -\rangle$.

EXAMPLE 3.8. For the function f specified by the global graph



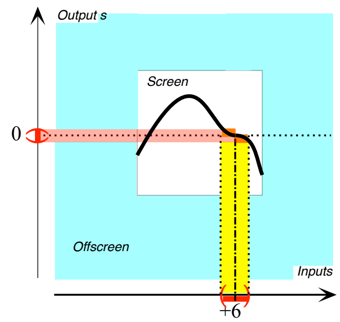
the bounded input $+6$ is an *even zero* because:

- the *outputs* for inputs *near* $+6$ are all $small$,
- Height-sign f near $+6 = \langle -, -\rangle$ (Same signs.)

- An **odd zero** is a zero whose Height-sign is either $\langle +, -\rangle$ or $\langle -, +\rangle$.

pole
parity
even pole
odd pole

Example 3.9. For the function f specified by the global graph



the bounded input $+6$ is an *odd zero* because:

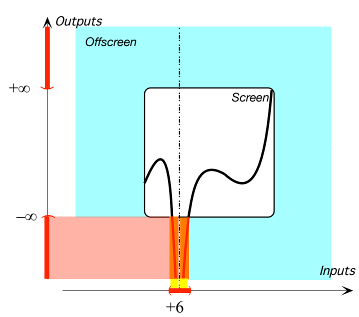
- the *outputs* for inputs *near* $+6$ are all $small$,
- Height-sign f near $+6 = \langle +, - \rangle$ (Opposite signs.)

2. Given a function f, a **pole** of f is a *bounded* input whose Height-size is $\langle large, large \rangle$. We will distinguish two kinds of poles according to their parity:

We will distinguish two kinds of poles according to their **parity**:

- An **even pole** is a pole whose Height-sign is either $\langle +, + \rangle$ or $\langle -, - \rangle$.

Example 3.10. For the function f specified by the global graph

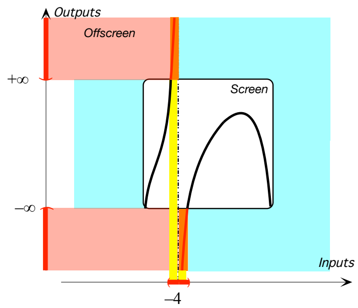


the bounded input $+6$ is an *even pole* because:

- the *outputs* for inputs *near* $+6$ are all $large$,
- Height-sign f near $+6 = \langle -, - \rangle$ (Same signs.)

- An **odd pole** is a pole whose Height-sign is either $\langle +, - \rangle$ or $\langle -, + \rangle$.

EXAMPLE 3.11. For the function f specified by the global graph



the bounded input $+-4$ is an *odd pole* because:

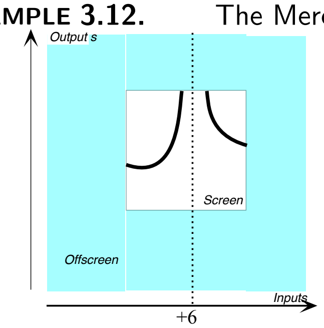
- the *outputs* for inputs *near* -4 are all $large$,
- Height-sign f near $-4 = \langle +, - \rangle$ (Opposite signs.)

slope
conclusive
inconclusive

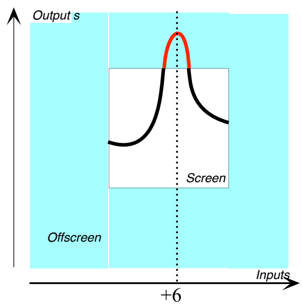
7 Conclusive information

Inasmuch as we can see the Magellan input and the Magellan output, a Magellan view is **conclusive** while a Mercator view may often be **inconclusive**.

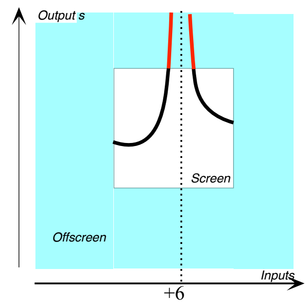
EXAMPLE 3.12. The Mercator view



is *inconclusive* regarding the *outputs* for *inputs near* $+6$ because when zooming out we could get something like continuation



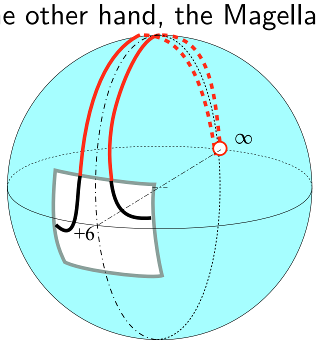
or like



But

- While the Mercator view on the left would be *conclusive* regarding the outputs for inputs near $+6$,
- The Mercator view on the right would still be *inconclusive*.

On the other hand, the Magellan view



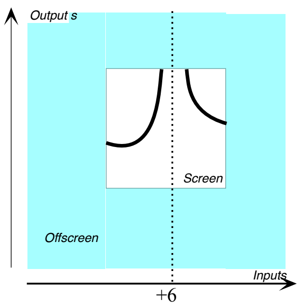
would be *conclusive* as we would see that the input $+6$ is a *pole*.

For the sake of simplicity, from now on

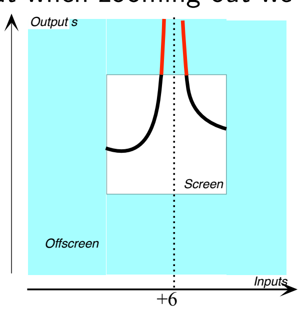
AGREEMENT 3.1 Mercator view The Mercator view will always be *assumed* to be conclusive.

In other words, the offscreen graph will always be assumed to be a **continuation** of the onscreen graph. Of course, this begs the question: What is a continuation? For the time being we will just give a couple of examples and leave the answer for when we have local features with which to describe things.

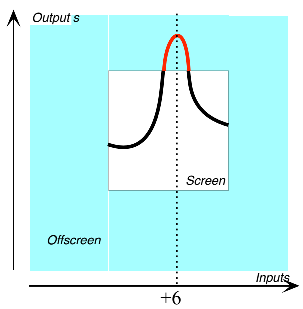
EXAMPLE 3.13. To assume that the *Mercator view*



is *conclusive* regarding the *outputs* returned for *inputs near* $+6$ is to assume that when zooming out we *would* get something like

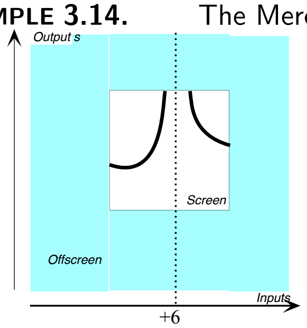


but *not* like



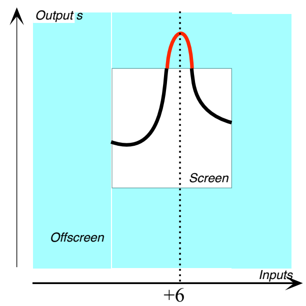
Inasmuch as we can see the Magellan input and the Magellan output, a Magellan view is .

EXAMPLE 3.14. The Mercator view

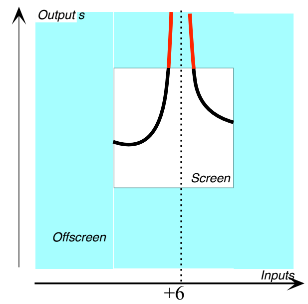


is *inconclusive* regarding the *outputs* for *inputs near* $+6$ because when zooming out we could get something like

slope
slope-sign



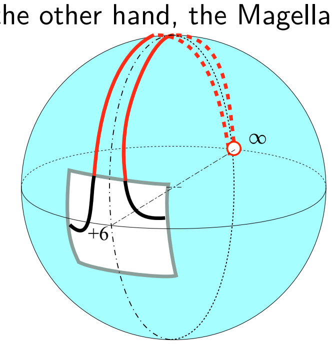
or like



But

- While the Mercator view on the left would be *conclusive* regarding the outputs for inputs near $+6$,
- The Mercator view on the right would still be *inconclusive*.

On the other hand, the Magellan view



would be *conclusive* as we would see that the input $+6$ is a *pole*.

8 Local Slope

1. Inasmuch as, in this text, we will only deal with *qualitative* information we will be mostly interested in the **slope-sign**: .

PROCEDURE 3.5 To get Slope-sign near a given input for a function specified by a global graph

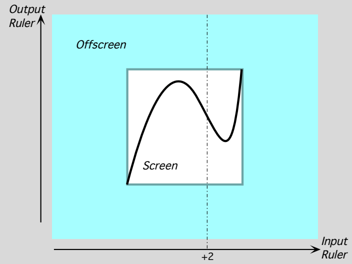
i. Mark the local graph near the given input
ii. Then the slope-sign is:
$\nearrow$ when the local graph looks like $\nearrow$ or $\nearrow$, that is when the *outputs* are **increasing** as the inputs are going the way of the input ruler,

$\searrow$ when the local graph looks like ⟍ or ⟍, that is when the *outputs* are **decreasing** as the inputs are going the way of the input ruler.

iii. Code Slope-sign f according to Definition 3.1 (Page 81)

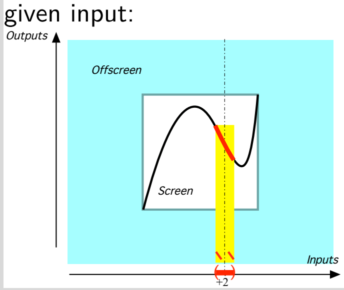
LANGUAGE 3.3 Slope-sign The usual symbols are $+$ Instead of $\nearrow$ and $-$ instead of $\searrow$ but, in this text, in order not to overuse $+$ and $-$, we will use $\nearrow$ and $\searrow$.[1]

TEMO 3.7 Let HIC be the function whose Mercator graph is



and let the given input be $+2$. Then to get Slope-sign HIC near $+2$

i. We get the local graph near the given input:



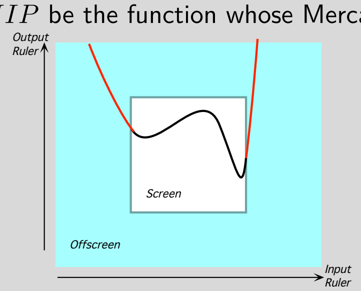
ii. We then get

- The slope sign *left* of $+2$ is $\searrow$
- The slope sign *right* of $+2$ is $\searrow$

which we code as:

Slope-sign HIC near $+2 = \langle \searrow, \searrow \rangle$

TEMO 3.8 Let HIP be the function whose Mercator graph is

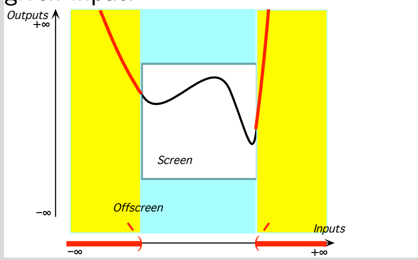


and let the given input be ∞. Then to get Slope sign HIP near ∞

[1] Educologists will surely appreciate "Sign-slope $f = \nearrow$ *iff* Sign-heigth $f' = +$".

slope-size
concavity
concavity-size
concavity-sign

i. We get the local graph near the given input:



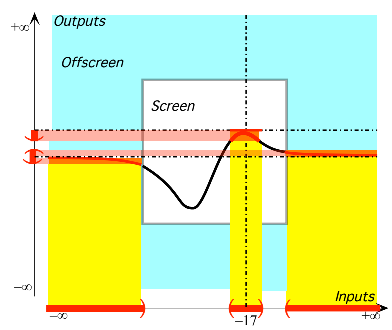
ii. We then get that:
· The slope sign *left* of ∞, that is near $+\infty$, is $\diagup$
· The slope sign *right* of ∞, that is near $-\infty$, is $\diagdown$

which we code as:

Slope-sign HIP near $\infty = \langle \diagup, \diagdown \rangle$

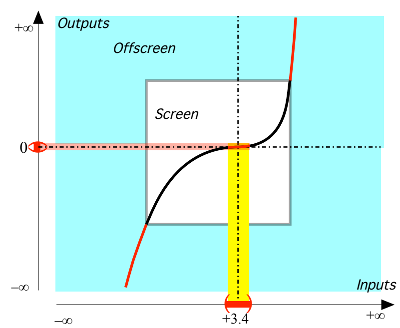
2. In this text, we will not deal with **slope-size** other than in the case of a 0**-slope input** that is an input, be it x_0 or ∞, near which slope-size is *small*. This is because 0-slope inputs will be extremely important in *global analysis* as finding 0-slope inputs comes up all the time in direct "applications" to the real world:

EXAMPLE 3.15. The function



has both -17 and ∞ as 0-slope inputs

EXAMPLE 3.16. The function



Only $+3.4$ is a 0-slope input.

9 Local Concavity

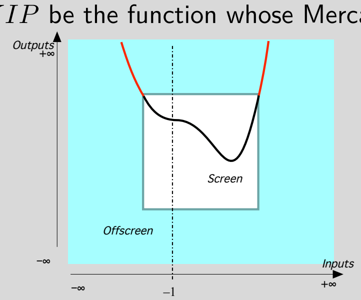
1. Inasmuch as, in this text, we will be only interested in *qualitative analysis* we will not deal with the **concavity-size** but only with the **concavity-sign**:

PROCEDURE 3.6 To get Concavity-sign near a given input for a function specified by a *global graph*

i. Mark the local graph near the given input
ii. Then the concavity-sign is:
$\cup$ when the local graph is *bending up* like ⟍ or ⟋,
$\cap$ when the local graph is *bending down* like ⟋ or ⟍.
iii. Code Slope-sign f according to Definition 3.1 (Page 81)

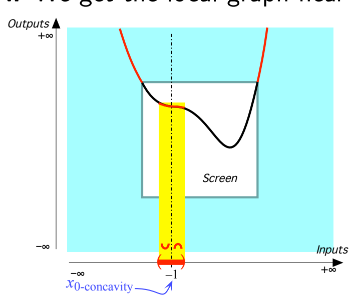
LANGUAGE 3.4 Concavity-sign The usual symbols are $+$ Instead of $\cup$ and $-$ instead of $\cap$ but, in this text, in order not to overuse $+$ and $-$, we will use $\cup$ and $\cap$.[2]

TEMO 3.9 Let KIP be the function whose Mercator graph is



and let the given input be -1. Then to get Concavity sign KIP near -1

i. We get the local graph near the given input:



ii. We then get that:
· The concavity sign *left* of -1, is $\cup$
· The concavity sign *right* of -1, is $\cap$

which we code as:

Concavity Sign KIP near $-1 = \langle \cup, \cap \rangle$

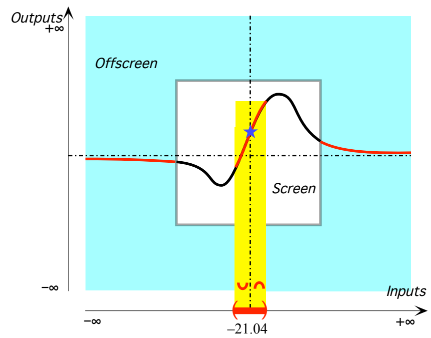
2. Given a function f, the inputs whose Concavity-size is 0 will be particularly important in *global analysis*:

A *bounded* input x_0 is a 0**-concavity input** if inputs that are near x_0 have *small concavity*. We will use $x_{0\text{-concavity}}$ to refer to 0-concavity inputs.

[2]Educologists will surely appreciate “Sign-concavitye $f = \cup$ *iff* Sign-heigth $f'' = +$”.

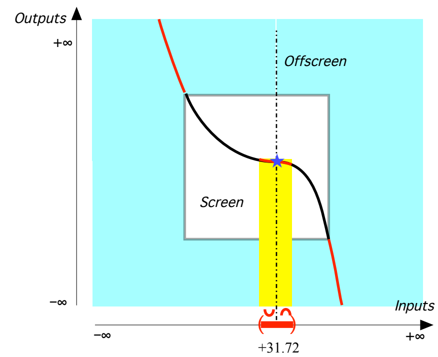
continuous at x_0
continuous

EXAMPLE 3.17. Given the function whose Mercator graph is



$x_{0-concavity} = -21.04$

EXAMPLE 3.18. Given the function whose Mercator graph is



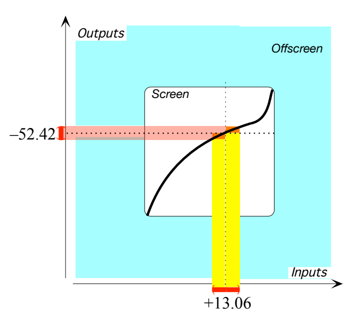
$x_{0-concavity} = +31.72$

10 Pointwise Continuity

The use of nearby inputs instead of the given input raises a most important question: To what extent are the nearby outputs (outputs for *nearby* inputs) *all* near the output *at* the given input? And, as it turns out, the question has no simple answer. So, as a backdrop to what will be the case with Algebraic Functions, we will just illustrate some of the many different possible answers.

1. Continuity at x_0. Given a bounded input x_0, a function f is **continuous at x_0** when *all* the outputs for nearby inputs are themselves near $f(x_0)$, the output at x_0.

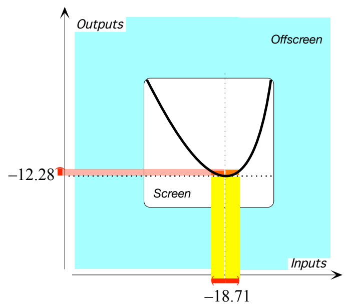
EXAMPLE 3.19. The function



is *continuous* at $+13.06$ because:

- the output *at* $+13.06$ is -52.42 and
- the outputs for *all* nearby Inputs, both *left* of $+13.06$ and *right* of $+13.06$, are themselves near -52.42.

EXAMPLE 3.20. The function



is *continuous* at -18.71 because

- the output *at* -18.71 is -12.28

and

- the outputs for *all* nearby Inputs, both *left* of -18.71 and *right* of -18.71, are themselves *near* -12.28.

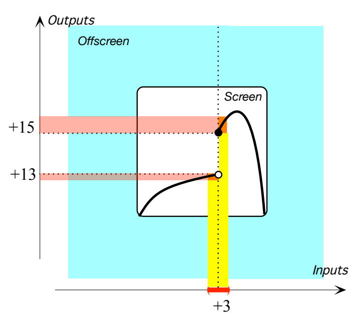
discontinuous
discontinuous at x_0
jump
hollow dot

2. Discontinuity at x_0. Given a bounded input x_0, a function is **discontinuous at x_0** when *not all* the outputs for nearby inputs are near $f(x_0)$, the output at x_0.

- A function can be discontinuous at x_0 because the function has a **jump** at x_0, that is because the outputs for nearby inputs on one side of x_0 are all near one bounded output while all the outputs for nearby inputs on the other side of x_0 are near a different bounded output.

  Since we use solid dots to picture input-output pairs, we will use **hollow dots** for points that *do not* picture input-output pairs.

EXAMPLE 3.21. The function



is *discontinuous* at $+3$ because the function has a *jump* at $+3$ that is:

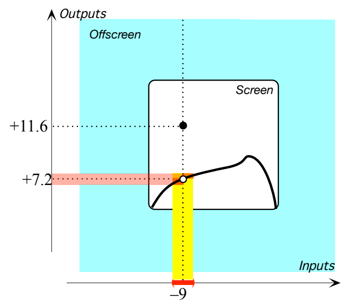
- the outputs for nearby inputs *right* of $+3$ are all near $+15$,

but

- the outputs for nearby Inputs *left* of $+3$ are all near $+13$.

gap
cut-off input
on-off function
transition function
transition input

EXAMPLE 3.22. The function

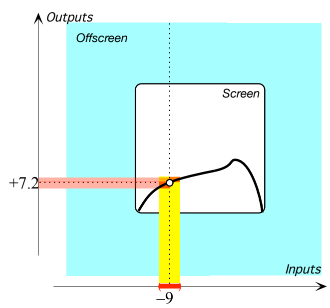


is *discontinuous* at -9 because the function has a *double jump* at -9 that is:

- even though the outputs for nearby inputs, both inputs *right* of -9 and inputs *left* of -9, are all near $+7.2$,
- the output for -9 itself is $+11.6$.

- A function can be discontinuous at x_0 because the function has a **gap** at x_0, that is because the function does not return a bounded output for x_0

EXAMPLE 3.23. The function

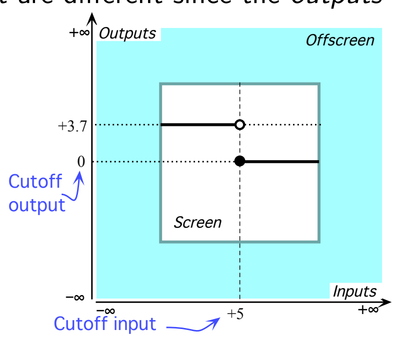


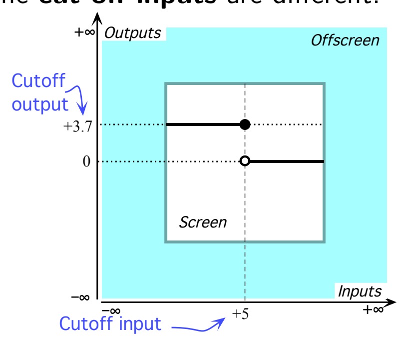
is *discontinuous* at -9 because the function has a *gap* at -9 that is:

- even though the outputs for nearby inputs, both inputs *right* of -9 and inputs *left* of -9, are all near $+7.2$,
- there is no output for -9 itself.

- Actually, discontinuous functions are quite common in Engineering.

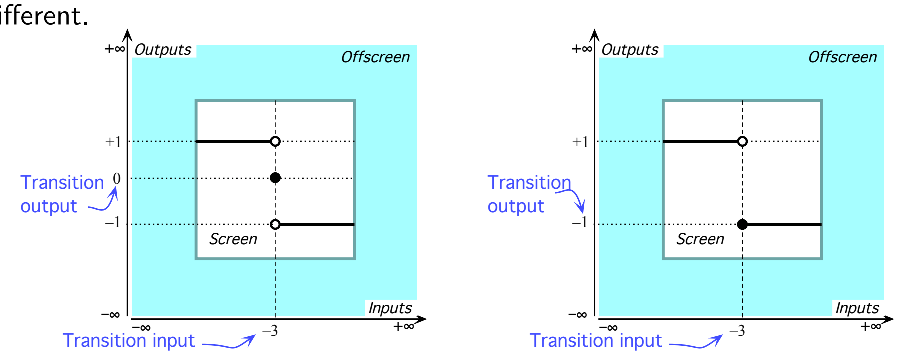
EXAMPLE 3.24. The following **on-off functions** are both *discontinuous* but are different since the *outputs* for the **cut-off inputs** are different.





quasi-continuous at
removable discontinuity at
remove
override
supplement

EXAMPLE 3.25. The following **transition functions** are both *discontinuous* but are different since the *outputs* for the **transition inputs** are different.



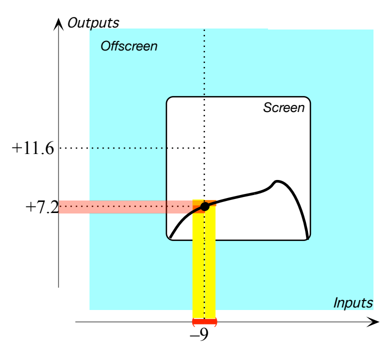
- And, finally, there are even functions that are discontinuous everywhere! See https://en.wikipedia.org/wiki/Nowhere_continuous_function

3. Quasi-continuity at x_0. A function is **quasi-continuous at** x_0 if the discontinuity could be **removed** by **overriding** or **supplementing** the global input-output rule with an input-output table.

> **LANGUAGE 3.5 Removable discontinuity at x_0** is the standard term but, for the sake of language consistency, rather than saying that a function *has* (or *does not have*) a removable discontinuity at x_0, *we* will prefer to say that a function *is* (or *is not*) quasi-continuous at x_0.

EXAMPLE 3.26. The function in Example 3.22 is *discontinuous* at -9 but the discontinuity could be *removed* by overriding the input-output pair $(-9, +11.6)$ with the input-output table

Input	Output
-9	$+7.2$



=========OK SO FAR=========
=======Begin WORK ZONE=======

smoothness
kink
smooth

11 Local Smoothness

For several reasons, **smoothness** is quite a bit more difficult to pin down than continuity.

1. Roughly, smoothness extends to slope and concavity the requirements that continuity made on the height namely that slope and concavity should not change abruptly. There is a big difference though:

- In the case of continuity, we need to look at what happens *at* the given input and then to what happens *near* the given input but only to see if there is a jump and not even when there is a gap at x_0.
- In the case of slope and concavity, on the other hand, even with local graphs, neither slope nor concavity makes sense *at* the given input and what matters is only what happens *near* the given input.

> **Note 3.2 Smothness *near* vs. smoothness *at*** Most unfortunately, the *usual* mathematical concept of smoothness implies continuity which is not the way we think of smoothness in the real world.

Example 3.27. A PVC sewer and drain pipe is usually perceived as being "smooth" regardless of whether or not it is solid or perforated and a smoothly bending copper pipe doesn't stop being so if and when it develops a pinhole.

So, in this text and in trying to *picture* smoothness, *we* will go by $f(x_0+h)$ and not pay any attention to $f(x_0)$.

2. The *first* degree of smoothness is for the slope to be continuous, that is, to borrow a word from plumbing, we don't want the curve to have any **kink**. More precisely, we don't want any input x_0 for which there is a "jump in slope" from one side of x_0 to the other side of x_0. In other words, we don't want any input x_0 for which the slope on one side differs from the slope on the other side by some bounded number.

3. The *second* degree of smoothness is for the concavity to be continuous but this is much harder to picture because it is hard to judge by just looking how much a curve is bending.

So, in this text, **smoothness** will refer to just the *first* degree of smoothness, that is for the curve to have no kink which, fortunately and as we will see, will make it easy to be "reasonable" about smoothness.

=======End WORK ZONE========

Shoes need feet to walk!
Everything needs something to function!.

Mehmet Murat ildan

compactification
Magellan input
input Magellan circle

Chapter 4

Features Near ∞

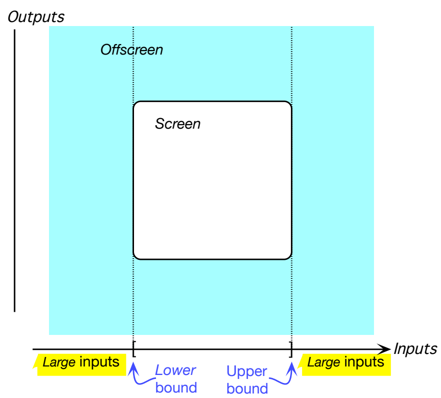
While what we will do in this chapter near ∞ will essentially be the same as what we did near x_0 in Chapter 3 Features Near x_0, the difficulty near ∞ will be "seeing" *large* numbers in the Mercator view as they really are.

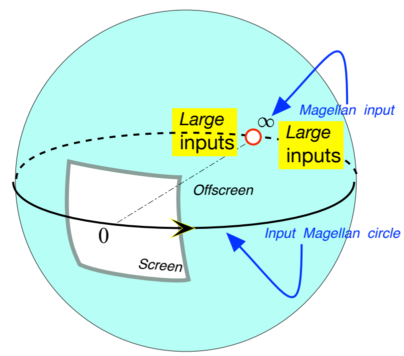
1 Compactification

This is where the Magellan view will be most helpful because, not only is the information provided by the Mercator view not always conclusive with regards to large numbers, but the Magellan view will often explain what happens offscreen and therefore also what happens onscreen.

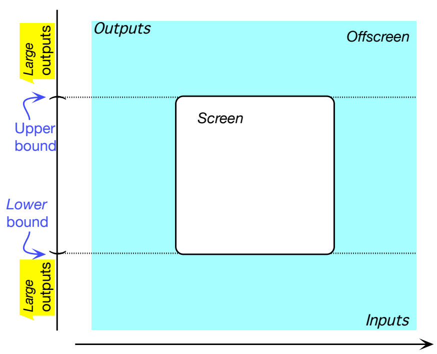
1. Magellan inputs So, the first thing we need is the equivalent of a Cartesian setup with an **input Magellan circle** in place of an input ruler

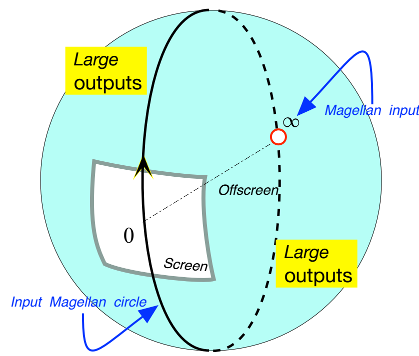
output Magellan circle
thicken
neighborhood of ∞
nearby input





and an **output Magellan circle** instead of an output ruler





=========OK SO FAR=========

2. We will **thicken** ∞ into a **neighborhood of ∞**. Then, by **nearby inputs**, with ∞ going without saying, we will mean *large* inputs

Since we will use the words near inputs both when the given input is x_0 and when the given input is ∞, we must clarify:

> **NOTE 5.1 (Restated) Location of essential inputs** will be short for outputs returned by the function f for nearby inputs that is:
> - When the given input is *bounded*, nearby inputs are *bounded* inputs near the given input,
> - When the given input is ∞, nearby inputs are *large bounded* inputs near the ∞,

=======Begin WORK ZONE=======

3. Magellan views are conclusive.

Any answer, though, will obviously depend on whether or not ∞ is allowed as Magellan input and Magellan output and the reader must be warned that the prevalent stand *in this country* is that ∞ does not exist and that one should use **limits**. (For what limits are, see https://en.wikipedia.org/wiki/Limit_(mathematics).) This for no apparent reason and certainly for none ever given.[1] limit input level band

As for us, we *will* allow ∞ as Magellan input and Magellan output, an old, tried and true approach. See https://math.stackexchange.com/questions/354319/can_a_function_be_considered_continuous_if_it_reaches_infinity_at_one_point and, more comprehensively, https://en.wikipedia.org/wiki/Extended_real_number_line.

=======End WORK ZONE========

=====TRANSFERRED FROM OLD 3======

Nor can we declare a Magellan input because $\boldsymbol{\infty}$ can neither: (Page 69).

However, in both cases we can, and will, declare nearby inputs and so, even though the computations will actually be different, the concept will be the same and so it will be convenient to agree that, from now on, a

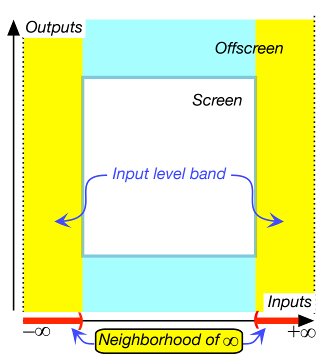
> **AGREEMENT 4.1 Given input** can be either a bounded input x_0 or the Magellan input ∞.

=========OK SO FAR=========

=========OK SO FAR=========

4.

5. We will thicken input level lines into **input level bands** that is vertical bands through the input neighborhoods.



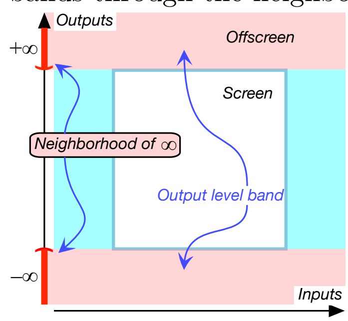
[1]The absolute silence maintained by Educologists in this regard is rather troubling.

output neighborhood
output level band
local behavior
local analysis
locate
global analysis
local graph place

==========OK SO FAR==========
==========OK SO FAR==========

6. On the other hand, we won't always be able to thicken an output into an **output neighborhood** because it is the *function* which returns the outputs for nearby inputs and the outputs for nearby inputs may *not* be near the output at the given input. (We will discuss this in the next section, Smoothness near ∞. (Page 131)

Nevertheless, we will often have to use **output level bands** that is horizontal bands through the neighborhoods of outputs



7. The **local behavior** of a function f at a given input will then be determined by the outputs returned by f for nearby inputs. The **local analysis** of f will be the investigation of the local behaviors of f.

In contrast, **locating** the input(s), if any, at which a function f has a required local behavior will be a problem in **global analysis** inasmuch as it will involve searching among *all possible* inputs.

==========OK SO FAR==========
==========OK SO FAR==========

2 Local graph place near ∞

========Begin WORK ZONE========

Since plot points are at the intersection of an input level line and an output level line, we will thicken plot points into **local graph places** at the intersection of an input level band and an output level band.

========End WORK ZONE=========

But given an input, be it ∞ or x_0, we will usually deal separately with each side of the input neighborhood. See **?? ??** (**??**) and **?? ??** (**??**.). We will thus know which side of the input is linked to which side of the output and the **sided local graph place** will then consist of two smaller local graph places, one on each side of the given input.

sided local graph place

1. We obtain the procedure to get a sided local graph place just by thickening **??** (Page 57):

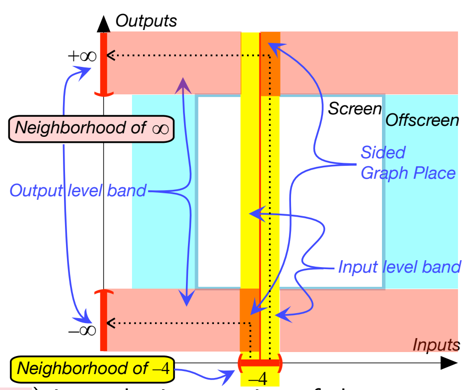
PROCEDURE 4.1 To get the sided local graph place for an input-output pair knowing which side of the input neighborhood is paired with which side of the output neighborhood.

i. Mark a *neighborhood* of the input on the input ruler,
ii. Draw the *input level band*,
iii. Mark a *neighborhood* of the output on the output ruler,
iv. Draw the *output level band*,
v. Mark which side of the input neighborhood is linked to which side of the output neighborhood,
vi. The place for the given input - output pair is at the intersection of the corresponding *sides* of the level bands.

TEMO 4.1 Get the sided place for (-4 , ∞) given that:

- $-4^- \longrightarrow -\infty$
- $-4^+ \longrightarrow +\infty$

i. We mark a *neighborhood of* -4 on the *input ruler*,
ii. We draw the *input level band* through the *neighborhood* of -4,
iii. We mark a *neighborhood of* ∞ on the *output ruler*,
iv. We draw the *output level band* through the *neighborhood* of ∞,
v. Mark: • left of -4 $\to$ near $-\infty$
• right of -4 $\to$ near $+\infty$

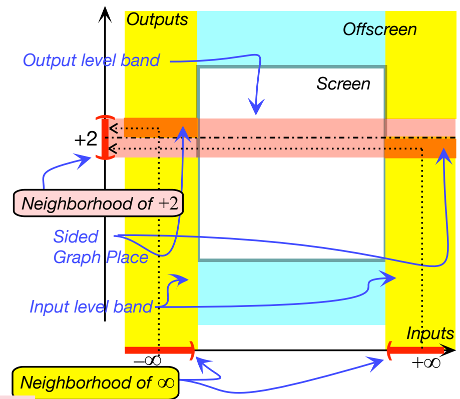


vi. The *sided graph place* for (-4 , ∞) is at the intersection of the corresponding *sides* of the level bands.

TEMO 4.2 Get the sided place for (∞ , $+2$) given that:

- $-\infty \longrightarrow +2^+$
- $+\infty \longrightarrow +2^-$

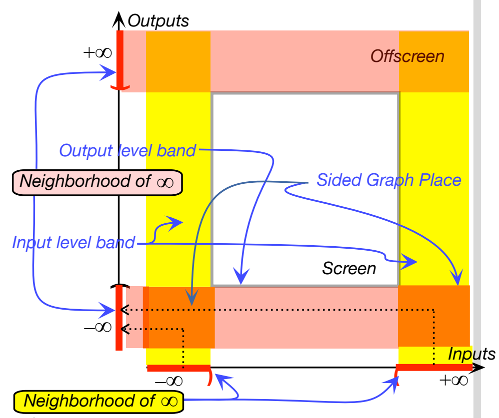
i. We mark a *neighborhood of* ∞ on the *input ruler*,
ii. We draw the *input level band* through the *neighborhood* of ∞,
iii. We mark a *neighborhood of* $+2$ on the *output ruler*,
iv. We draw the *output level band* through the *neighborhood* of $+2$,
v. Mark: • $-\infty \to +2^+$
• $+\infty \to +2^-$



vi. The *sided graph place* for (∞, $+2$) is at the intersection of the corresponding *sides* of the level bands.

TEMO 4.3 Get the sided place for (∞, ∞) given that:
• $-\infty \longrightarrow -\infty$
• $+\infty \longrightarrow -\infty$

i. We mark a *neighborhood of* ∞ on the *input ruler*,
ii. We draw the *input level band* through the *neighborhood* of ∞,
iii. We mark a *neighborhood of* ∞ on the *output ruler*,
iv. We draw the *output level band* through the *neighborhood* of ∞,
v. Mark: • $-\infty \to -\infty$
• $+\infty \to -\infty$



vi. The *sided graph place* for (∞, ∞) is at the intersection of the corresponding *sides* of the level bands.

=========OK SO FAR=========
=========OK SO FAR=========

3 Local graph near ∞

As we will see, the local graph place near ∞ will get us the information we want for some local feature but in most cases we will need the **local graph**

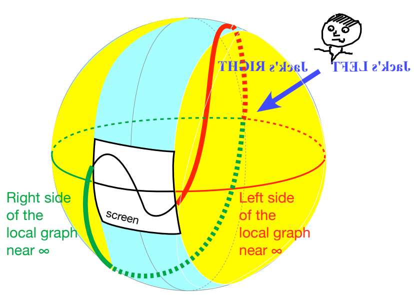
near ∞ near the given input, that is the part of the global graph which is in the local graph place. local graph extremity

Later, we will get local graphs from the global input-output rule but for the time being, and since in *this* chapter we only want to *name* and *describe* local features, the global input-output rule will go without saying and, as per **??** (**??**), we will get local graphs from the global graph of the function.

1. Local graph near ∞ When the given input is ∞, how we proceed depends on whether we have a Mercator view or a Magellan view of the global graph:

- With a Magellan view of the global graph, we proceed pretty much as in **??** and once we imagine facing ∞, we can *see* which side is which.

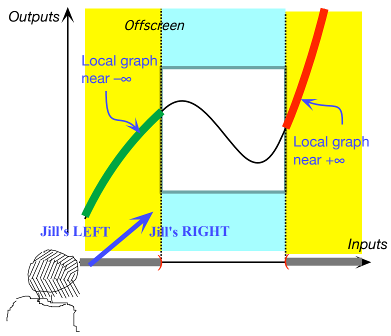
EXAMPLE 4.1.



Jack is facing ∞ so the local graph near $+\infty$ which is to *his left* is *left* of ∞ and the local graph near $-\infty$ which is *to* his right *is right* of ∞.

- With only a Mercator view of the global graph, there is of course no way we can get the whole local graph near ∞ and we will have to content ourselves with just the **extremities** of the local graph near ∞. However, since we cannot face ∞ and can only face the screen, we have to keep in mind **?? ??** (**??**) so that
  - ▸ The extremity of the local graph near $+\infty$ (*left of* ∞) is to *our right*,
  - ▸ The extremity of the local graph near $-\infty$ (*right of* ∞) is to *our left*.

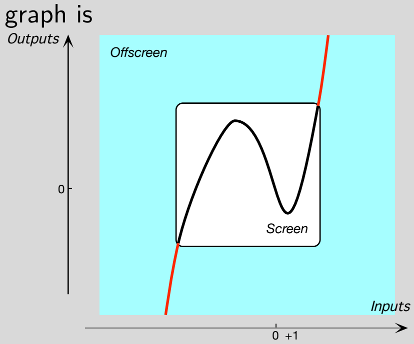
Example 4.2.



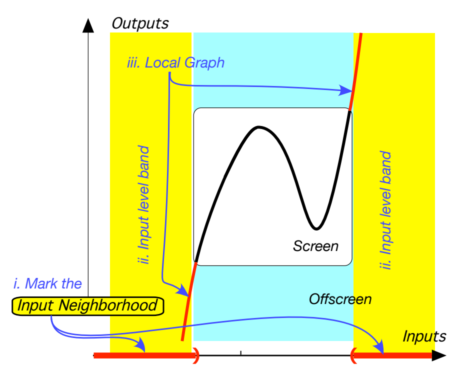
Jill is facing the *screen* so she can only see the *extremities* of the local graph near ∞ and she must keep in mind **??** **??** (**??**) so that the local graph near $+\infty$ (to *her right*) is *left* of ∞ and the local graph near $-\infty$ (to *her left*) is *right* of ∞.

But then we can still use **??** to get a local graph near ∞.

Temo 4.4 Get the local graph near ∞ of the function whose Mercator graph is



i. We mark a *neighborhood of* ∞ on the *input ruler* by marking the part of the input ruler beyond the bounds.,
ii. We draw the *input level band* through the *neighborhood* of ∞,
iii. The *local graph* near ∞ is the intersection of the input level *band* with the global graph,



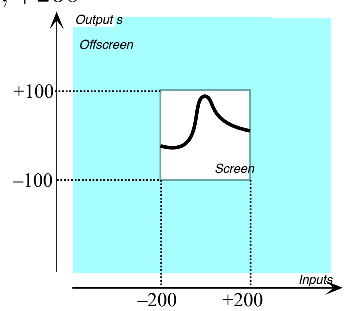
4 Offscreen graph

conclusive

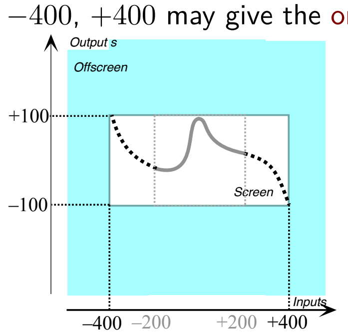
We come now to **??** which we raised in **????**. More precisely, an onscreen graph is very likely to be **inconclusive** in that the information given by an onscreen graph is most likely to depend on both:

- The input bounds.

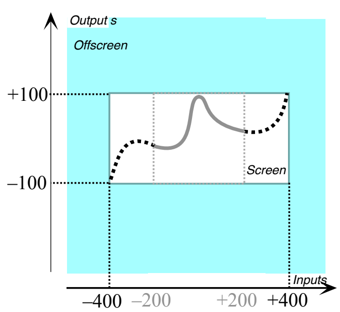
  EXAMPLE 4.3. The onscreen graph within the input bounds $-200, +200$

  is *not conclusive* because, increasing the input bounds from -200, $+200$ to -400, $+400$ may give the onscreen graph

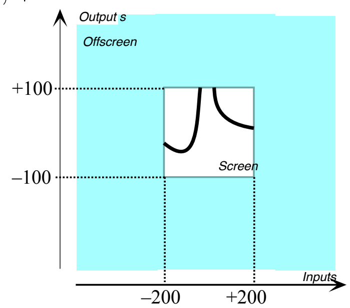
  just as well as the onscreen graph

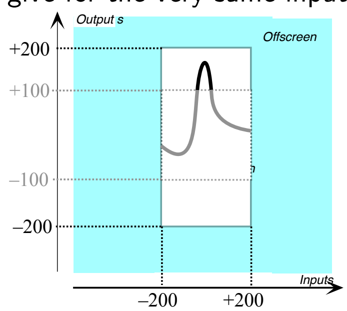
- The output bounds.

local graph near ∞

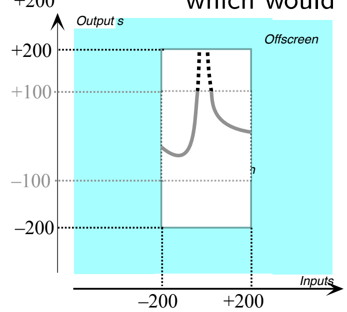
Example 4.4. The onscreen graph within the output bounds $-100, +100$



is *not conclusive* because, increasing the output bounds to -200, $+200$ may give for the very same inputs



which would be conclusive just as well as



which still would not be conclusive.

So, the offscreen graph can involve *two very different kinds* of inputs.

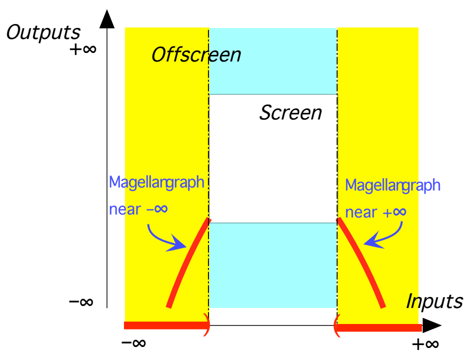
1. The offscreen graph *always* includes the **Local graph near ∞**, which is the part of the global graph, for large inputs left and right of the screen. Even though the local graph near ∞ is really in one single piece because large inputs are in a neighborhood of ∞, the local graph near ∞ appears to be in two pieces, one piece on each side of the screen:

▶ The part of the local graph near ∞ for inputs near $-\infty$, that is for inputs that are $-large$ and which is therefore left of the screen but right of ∞. polar graph

- The part of the local graph near ∞ for inputs near $+\infty$, that is for inputs that are $+large$ and which is therefore right of the screen but left of ∞.

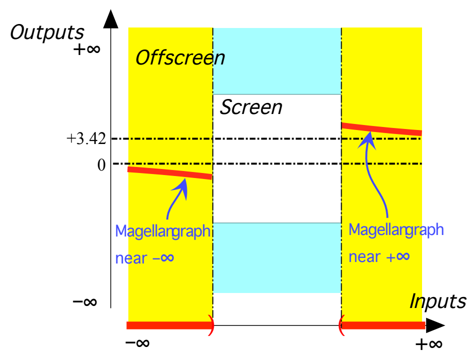
On the other hand, keep in mind that even for large inputs, a function may return outputs of any qualitative size, *bounded*, *large* or *small*.

EXAMPLE 4.5.



The *large* ***inputs*** both left and right of the screen have *large* outputs.

EXAMPLE 4.6.



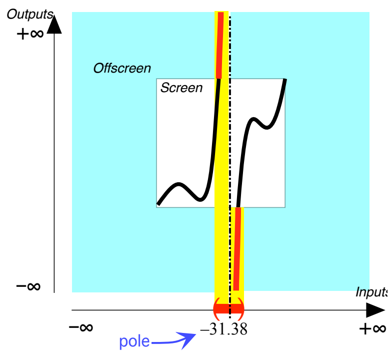
- The *large* inputs *left* of the screen have *small* ***outputs***,
- The *large* inputs *right* of the screen have *bounded* ***outputs***,

2. The offscreen graph *may* include parts, the **polar graphs**, which are for bounded inputs that are near poles, that is near bounded input(s) for whose nearby inputs the function returns *large* outputs. A polar graph is in two parts, one on each side of the pole

- The left part of the polar graph, that is the part of the polar graph which is for nearby inputs that are left of the pole,
- The right part of the polar graph, that is the part of the polar graph which is for nearby inputs that are right of the pole,

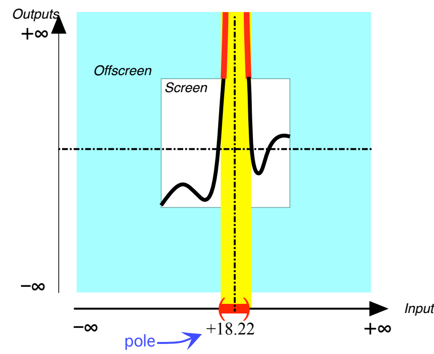
$64,000 Question

EXAMPLE 4.7.



- The nearby inputs *left* of -31.38 have $+large$ *outputs*,
- The nearby inputs *right* of -31.38 have $-large$ *outputs*,

EXAMPLE 4.8.



The nearby *inputs* both left and right of$+18.22$ have $+large$ outputs.

3. Kinds of offscreen graphs Because, as we will see, some algebraic functions *do not have* any pole while some algebraic functions *do have* pole(s), what the offscreen graph will be in each instance will depend on the answer to what will turn out to be the

DEFINITION 4.1 $64,000 Question
- Do *all bounded* inputs have *bounded* outputs?

or
- Is(Are) there *any* pole(s)?

So, the first step in our overall approach to the **??** will be:

PROCEDURE 4.2 To get the *offscreen* graph.

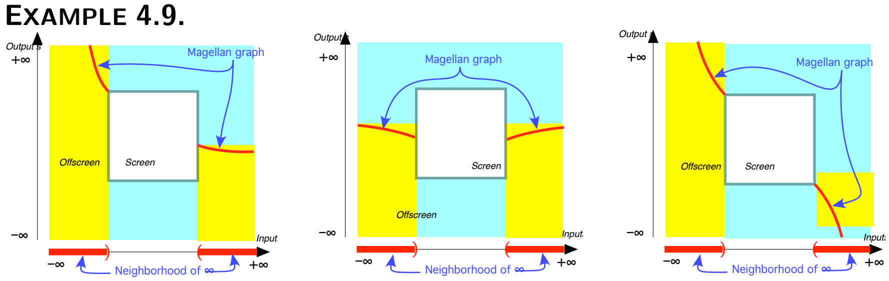
i. Get the local graph near ∞.
ii. Answer the **??**:
iii. Get the polar graph(s) if any.

In other words, depending on the answer to the , there will be two kinds of *offscreen* graphs:

- If the function *has no* pole, that is if the function returns *bounded* outputs for *all* bounded inputs, then the offscreen graph will include just the local graph near ∞.

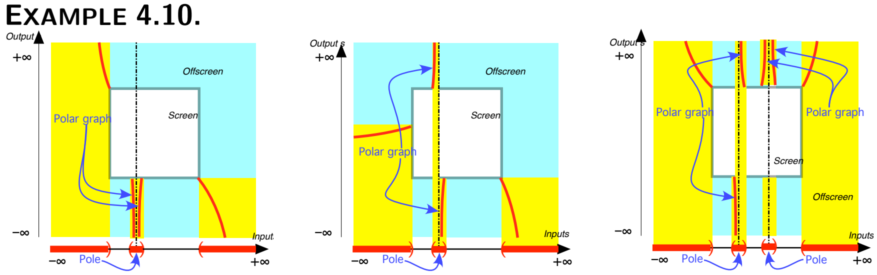
⟨
⟩
local code
angles

Example 4.9.



- If the functional requirement *has* pole(s), that is if there are bounded input(s) near which the function returns *large* outputs, then the offscreen graph will include polar graphs in addition to the local graph near ∞.

Example 4.10.



5 Local code near ∞

Since there is no reason to expect the local behavior of a function to be the same on both sides of the given input, be it x_0 or ∞, see **??** **??** (**??**) and **??** **??** (**??**), in order to describe *separately* the local behavior on each side of the given input, we need:

Definition 4.2 Local Code Given an input, be it x_0 or ∞,
i. We will use a pair of **angles** to stand for the input neighborhood with a comma in-between the angles to separate the sides: $\langle\ \ ,\ \ \rangle$.
ii. We must *face* the given input when coding local features.
iii. Then, the code that records the local feature for nearby inputs

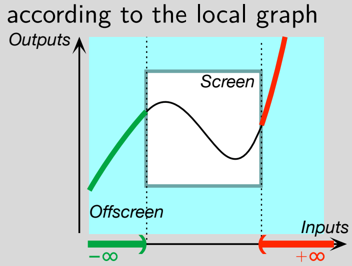
nearby inputs that are *left* of x_0 will go *left* of ,

nearby inputs that are *right* of x_0 will go *right* of ,

$\langle\ \blacksquare\ ,\ \blacksquare\ \rangle$

(Keep in mind that when the given input is ∞ we must *imagine* facing ∞ to know which is the *left* side and which is the *right* side of ∞.)

TEMO 4.5 Set up for the local code to record the local behavior near ∞ according to the local graph



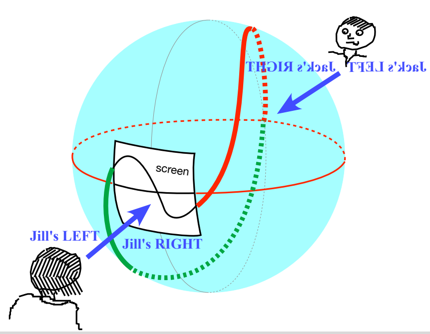
Since the local graph is near *infinity*, which we can only imagine facing, to encode the local behavior, the local feature

for inputs left of ∞ goes left of the comma.

for inputs right of ∞ goes right of the comma.

$\langle\ \blacksquare\ ,\ \blacksquare\ \rangle$

We must imagine facing ∞:



=========OK SO FAR=========

==

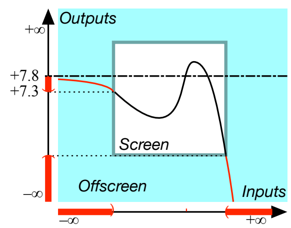
=========OK SO FAR=========

6 Height near ∞

height
Height-sign

We will just extend the concept of D for a bounded input x_0 to the concept of local height for ∞

The Height near ∞

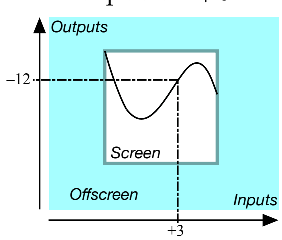


is $-large$ for inputs left of ∞ and $-small$ for inputs right of ∞

Given a function f, we will thicken the output *at* a given input, be it x_0 or ∞, into the **height** *near* the given input.

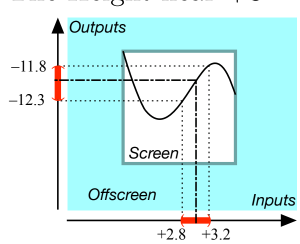
EXAMPLE 4.11.

The output at $+3$



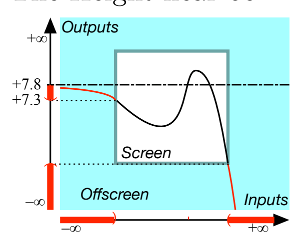
is -12

The Height near $+3$



is $-12 \pm small$

The Height near ∞



is $-large$ for inputs left of ∞ and $-small$ for inputs right of ∞

=========OK SO FAR=========

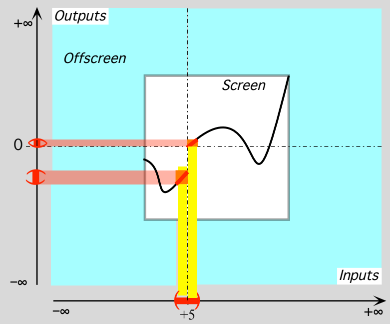
=========OK SO FAR=========

1. The **Height-sign** of f near a given input is the sign, $+$ or $-$, of the outputs for nearby inputs on each side of the given input.

PROCEDURE 4.3 To get the Height-sign near a given input of a function from its global graph,

i. Get from the local graph the sign, $+$ or $-$, of the *outputs* for nearby inputs on each side of the given input,
ii. Code Height-sign f according to Definition 3.1 (Page 81)

Temo 4.6 Get Height-sign near $+5$ for the function IAN from the local graph near $+5$



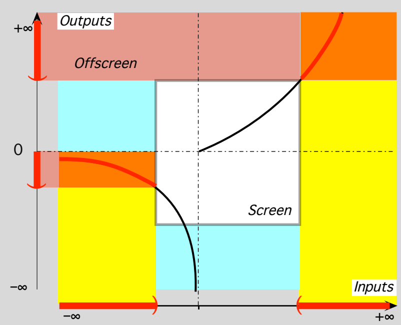
i. We get from the local graph the sign of the *outputs* for nearby inputs on each side of $+5$:

- The sign of the outputs *left* of $+5$ is $-$
- The sign of the outputs *right* of $+5$ is $+$

ii. We code the Height-sign:

Height-sign IAN near $+5 = \langle -, + \rangle$

Temo 4.7 Get Height-sign near ∞ for the function IAN from the local graph near ∞



i. We get from the local graph the sign of the outputs for nearby inputs on each side of ∞:

- The sign of the height *left* of ∞ is $+$
- The sign of the height *right* of ∞ is $-$

ii. We code the Height-sign:

Height-sign IAN near $\infty = \langle +, - \rangle$

=========OK SO FAR=========

=========OK SO FAR=========

=======Begin WORK ZONE=======

whose Height-sign is either $\langle +, + \rangle$ or $\langle -, - \rangle$. In other words, poles and zeros are *even*

whose Height-sign is either $\langle +, - \rangle$ or $\langle -, + \rangle$. In other words, poles and zeros are *odd*

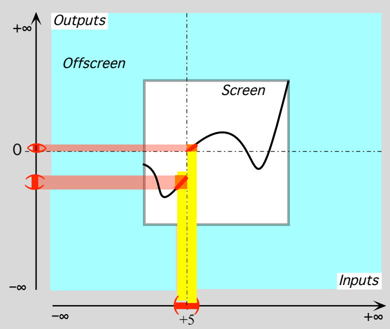
=======End WORK ZONE======== height-size

2. The **height-size** of f near a given input is the qualitative size, *large*, *bounded* or *small*, of the *outputs* for nearby inputs on each side of the given input.

PROCEDURE 4.4 To get the Height-size near a given input of a function from its global graph,

i. Get from the local graph the qualitative size, *large*, *bounded* or *small*, of the *outputs* for nearby inputs on each side of the given input,
ii. Code Height-size f according to Definition 3.1 (Page 81)

TEMO 4.8 Get Height-size near $+5$ for the function IAN from the local graph near $+5$



i. We get from the local graph the qualitative size, *large*, *bounded* or *small*, of the *outputs* for nearby inputs on each side of $+5$:

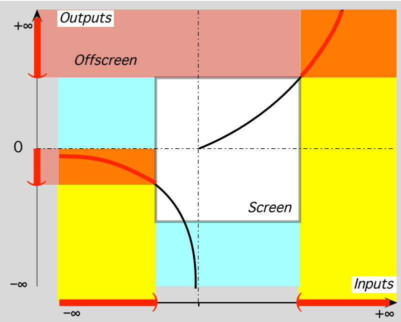
- The size of the outputs *left* of $+5$ is *large*
- The size of the outputs *right* of $+5$ is *small*

ii. We code the Height-size:

Height-size IAN near $+5 = \langle large, small\rangle$

TEMO 4.9 Get Height-size near ∞ for the function IAN from the local graph near ∞

x_{∞}-height
x_{0}-height



i. We get from the local graph the qualitative size, $large$, $bounded$ or $small$, of the *outputs* for nearby inputs on each side of ∞:

- The size of the height *left* of ∞ is $large$
- The size of the height *right* of ∞ is $small$

ii. We code the Height-size:

Height-size IAN near $\infty = \langle large, small \rangle$

3. The concept of Height provides us with conveniently systematic names:

- For a pole: $\boldsymbol{x}_{\infty\textbf{-height}}$
- For a zero: $\boldsymbol{x}_{0\textbf{-height}}$

==========OK SO FAR==========

==========OK SO FAR==========

=======Begin WORK ZONE=======

To do for the offscreen graph what we did in Chapter 3 for the on-screen graph requires that we first thicken infinity just the way we thickened bounded inputs in Chapter 3.

Obviously, the *means* in the case of ∞ will be quite different from the *means* we used in Chapter 3 for bounded inputs but, interestingly enough, the *ends* in both cases, that for infinity as well as that for bounded inputs, will be strikingly similar.

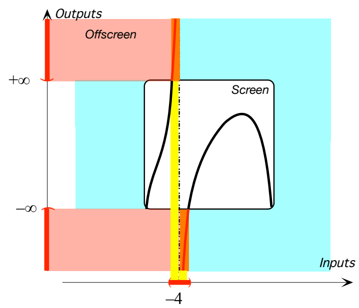
In fact, even the *means*, if not similar, will nevertheless remain in remarkably the same *spirit* and the reader should make every effort to identify and determine this spirit.

=======End WORK ZONE========

A function can be discontinuous at x_0 because the function has a pole at x_0.

conclusive
inconclusive

EXAMPLE 4.12. The function



is *discontinuous* at -4 because not only does the function have a gap at -4 but the function has a *pole* at -4 that is:

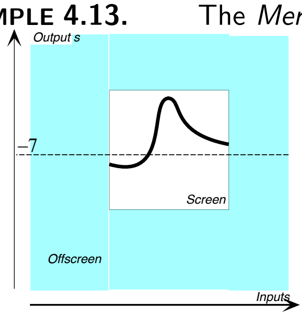
- the outputs for nearby inputs, both inputs *right* of -4 and inputs *left* of -4, are all $large$,

but

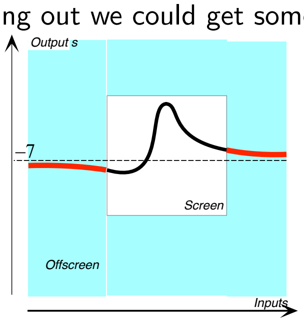
- -4 has no bounded output.

4. Conclusive information Inasmuch as we can see the Magellan input and the Magellan output, a Magellan view is **conclusive** while a Mercator view may often be **inconclusive**.

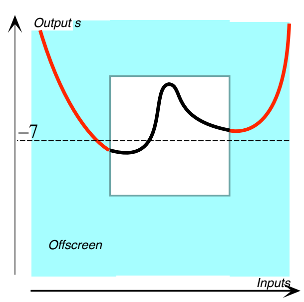
EXAMPLE 4.13. The *Mercator view*



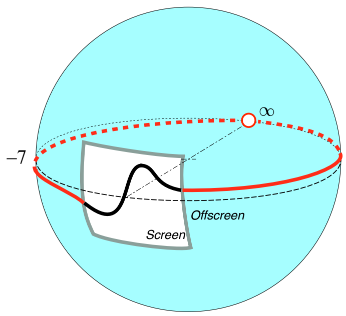
is *inconclusive* regarding the *outpust* returned for $large$ *inputs* because when zooming out we could get something like



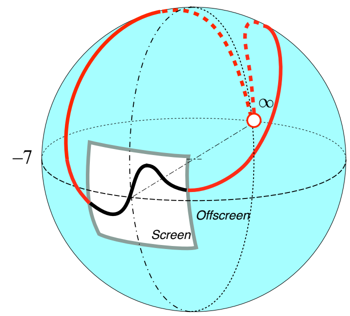
or like



which would both still be *inconclusive* regarding the *outputs* returned for $large$ *inputs*. On the other hand, either one of the *Magellan views*,



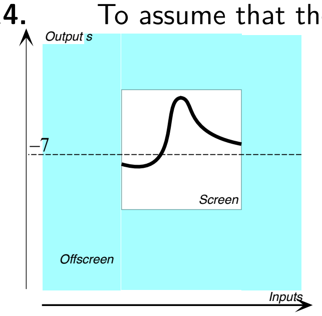
and



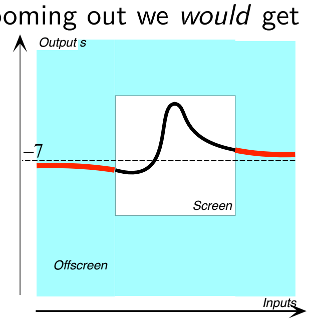
would be *conclusive* as we would see:

- from the *Magellan view* on the left that, for the *Magellan input* ∞, the function returns the *bounded output* -7,
- from the *Magellan view* on the right that, for the *Magellan input* ∞ is a *pole*.

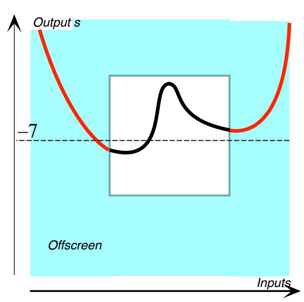
Example 4.14. To assume that the *Mercator view*



is *conclusive* regarding the *outputs* for $large$ *inputs* is to assume that when zooming out we *would* get something like

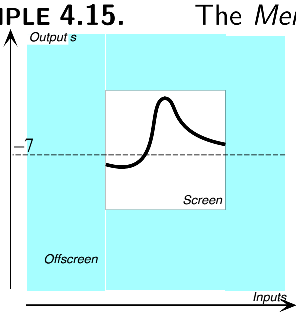


but *not* like

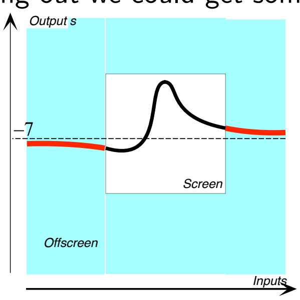


=======**Begin WORK ZONE**=======

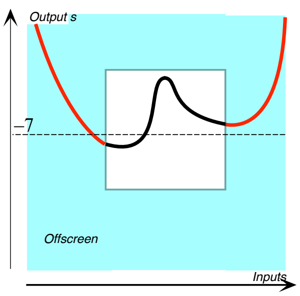
EXAMPLE 4.15. The *Mercator view*



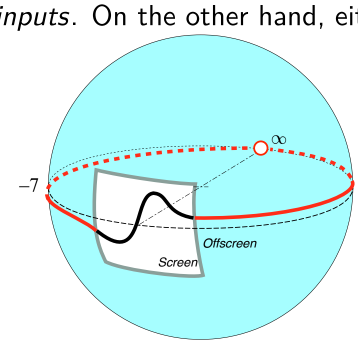
is *inconclusive* regarding the *outpust* returned for $large$ *inputs* because when zooming out we could get something like



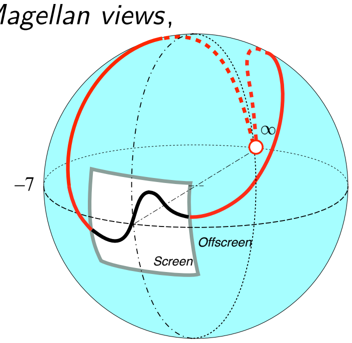
or like



which would both still be *inconclusive* regarding the *outputs* returned for $large$ *inputs*. On the other hand, either one of the *Magellan views*,



and



would be *conclusive* as we would see:

- from the *Magellan view* on the left that, for the *Magellan input* ∞, the function returns the *bounded output* -7,
- from the *Magellan view* on the right that, for the *Magellan input* ∞ is a *pole*.

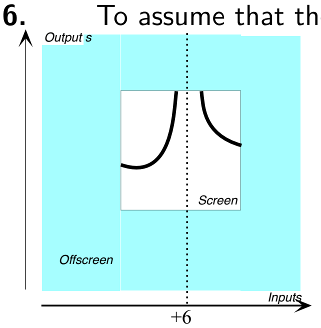
For the sake of simplicity, from now on

continuation

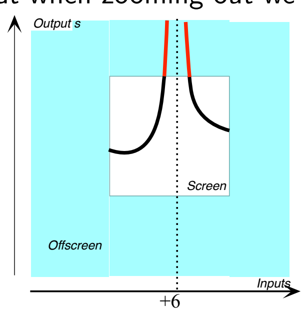
AGREEMENT 4.2 Mercator view The Mercator view will always be *assumed* to be conclusive.

In other words, the offscreen graph will always be assumed to be a **continuation** of the onscreen graph. Of course, this begs the question: What is a continuation? For the time being we will just give a couple of examples and leave the answer for when we have local features with which to describe things.

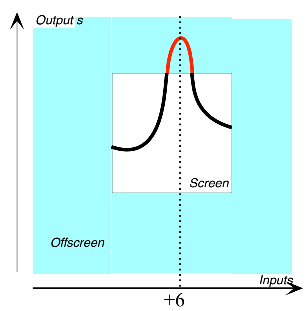
EXAMPLE 4.16. To assume that the *Mercator view*



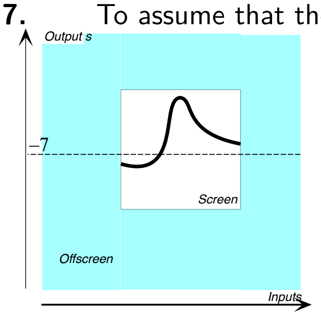
is *conclusive* regarding the *outputs* returned for *inputs near* $+6$ is to assume that when zooming out we *would* get something like



but *not* like

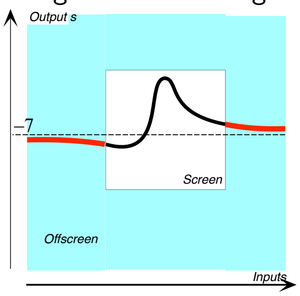


EXAMPLE 4.17. To assume that the *Mercator view*

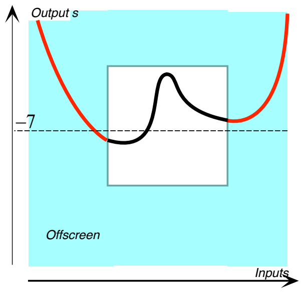


limit

is *conclusive* regarding the *outputs* for $large$ *inputs* is to assume that when zooming out we *would* get something like

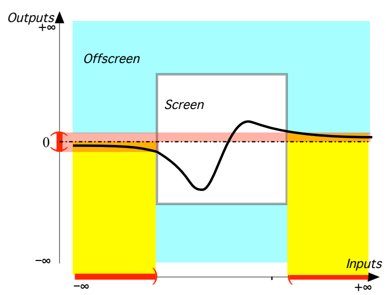


but *not* like



=======End WORK ZONE========

EXAMPLE 4.18. For the function



the Magellan input ∞ is a *zero* because:

the outputs for nearby inputs, both inputs *right* of ∞ and inputs *left* of ∞, are all $small$,

7 Continuity at ∞

The use of nearby inputs instead of the given input raises a crucial question: Are the outputs for *nearby* inputs *all* near the output *at* the given input?

Any answer, though, will obviously depend on whether or not ∞ is allowed as Magellan input and Magellan output and the reader must be warned that the prevalent stand *in this country* is that ∞ does not exist and that one should use **limits**. (For what limits are, see https://en.wikipedia.org/wiki/Limit_(mathematics).) This for no apparent reason and certainly for none ever given.[2]

As for us, we *will* allow ∞ as Magellan input and Magellan output, an old, tried and true approach. See https://math.stackexchange.com/questions/354319/can_a_function_be_considered_continuous_if_it_

[2]The absolute silence maintained by Educologists in this regard is rather troubling.

Magellan continuous at

reaches_infinity_at_one_point and, more comprehensively, https://en.wikipedia.org/wiki/Extended_real_number_line.

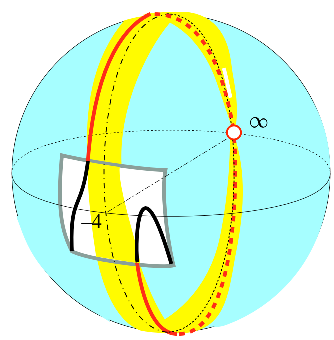
As a backdrop to what we will be doing with Algebraic Functions, we will now show some of the many different possible answers to the above question. For clarity, we will deal with bounded inputs and bounded outputs separately from ∞ as Magellan input and Magellan output.

Keep in mind that we use solid dots to picture input-output pairs as opposed to hollow dots which do *not* picture input-output pairs.

1. Magellan continuity at x_0. A function is **Magellan continuous at** x_0 when we could remove the discontinuity at x_0 by overriding or supplementing the global input-output rule with an input-output table involving ∞ as Magellan output.

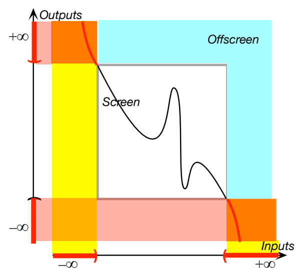
Example 4.19. The function in Example 4.12 is *discontinuous* at -4 because the function has a gap at -4 but *Magellan continuous* as we could *remove* the gap by *supplementing* the global input-output rule with the input-output table

Input	Output
-4	∞



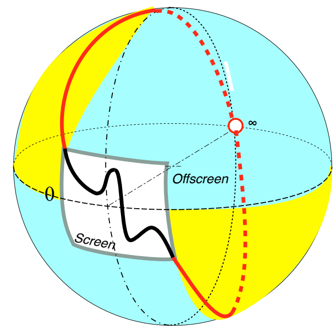
2. Magellan continuity at ∞. A function is **Magellan continuous at** ∞ when we could remove the discontinuity at ∞ by overriding or supplementing the global input-output rule with an input-output table involving ∞ as Magellan input and/or as Magellan output.

EXAMPLE 4.20. The function

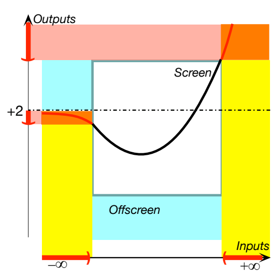


is *discontinuous* at ∞ but is *Magellan continuous* since we could remove the discontinuity with an input-output table involving ∞ as *Magellan input* and *Magellan output*,

Input	Output
$+\infty$	$-\infty$
$-\infty$	$+\infty$

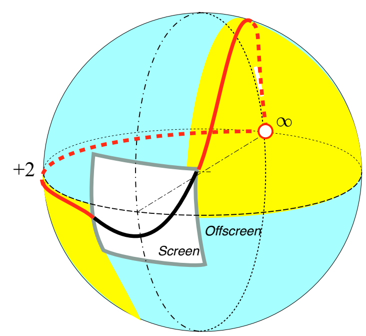


EXAMPLE 4.21. The function



is *discontinuous* at ∞ but is *Magellan continuous* since we could remove the discontinuity with an input-output table involving ∞ as Magellan input and Magellan output

Input	Output
$+\infty$	$+\infty$
$-\infty$	-2^{-}



3. Dealing with poles. The difficulty here stems only from whether or not it is "permisible" to use ∞ as a given input and/or as an output.

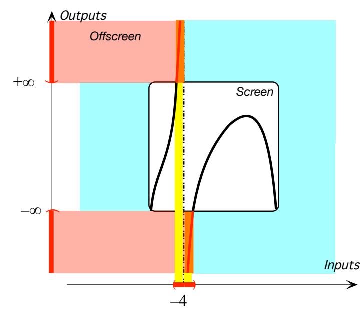
Even though, because There are *no symbols* for size-comparisons of signed-numbers (Page 69), $\boldsymbol{\infty}$ can neither;: *we do* use ∞ as a (Magellan) input and as a (Magellan) output because, as explained in **??** (**??**), we will *only* declare nearby inputs. (Which will shed much light on the local behavior of functions, in particular on the question of continuity.)

However, the reader ought to be aware that many mathematicians *in this country*, for reasons never stated, flatly refuse to use nearby inputs with their students.

Another reason *we do* is because Magellan views are a very nice basis on which to discuss the local behavior of functions for inputs near ∞ and when outputs are near ∞. In particular, we can see that discontinuiities caused by poles can be removed using ∞ as a Magellan output.

When ∞ as is not permissible as Magellan input and/or Magellan output, many functions are discontinuous

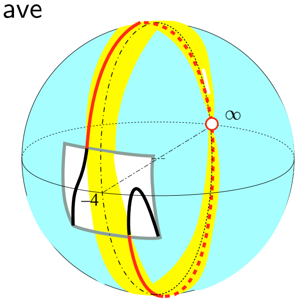
EXAMPLE 4.22. The discontinuity at -4 of the function in ?? whose Mercator graph is



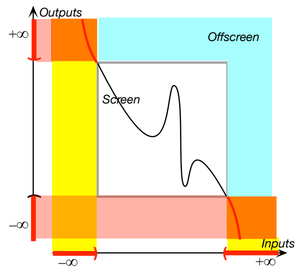
can be *removed* by *supplementing* the global input-output rule with the input-output table:

Input	Output
-4	∞

If we imagine the Mercator graph compactified into a Magellan graph, we have



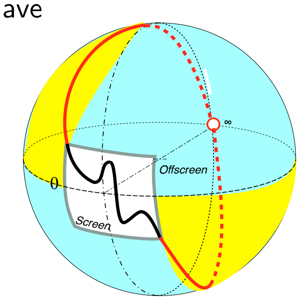
EXAMPLE 4.23. The discontinuity at ∞ of the function BIB in ?? whose Mercator graph is



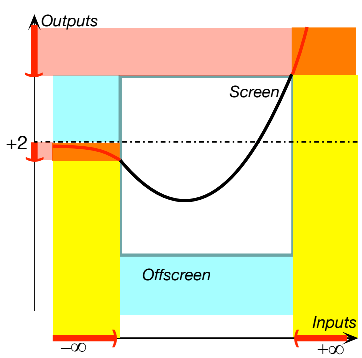
can be *removed* by *supplementing* the global input-output rule with the input-output table:

Input	Output
∞	∞

If we imagine the Mercator graph compactified into a Magellan graph, we have

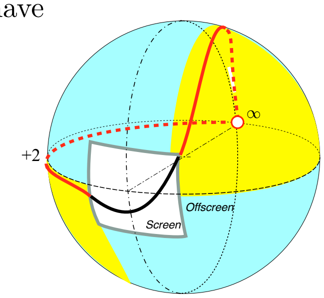


EXAMPLE 4.24. The function whose the global graph in *Mercator view* is



is *discontinuous* at ∞ not only because the global graph has a *gap* at ∞ since Local extreme-height inputs but also because the global graph has a *jump* at ∞.

If we imagine the Mercator view *compactified* into a Magellan view, we have



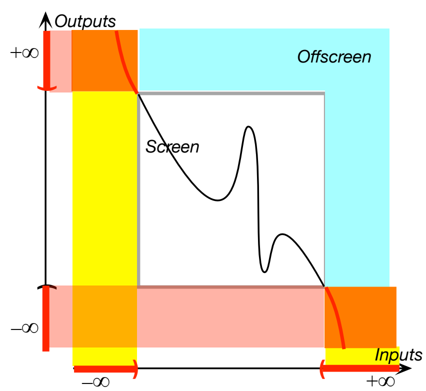
4. At ∞ The matter here revolves around whether or not ∞ should be allowed as a given input. *We* did but,

Also, in this section, for a reason which we will explain after we are done, we will have to deal separately with the case when the given input is x_0 and the case when the given input is ∞.

In accordance with **??**, we should say that all functions are discontinuous at ∞ since the outputs for inputs near ∞ cannot be near the output for ∞ for the very good reason that we cannot use ∞ as input to begin with.

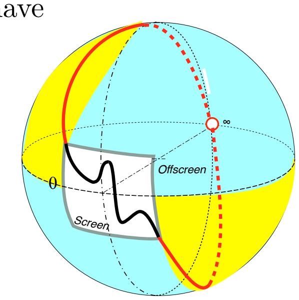
LANGUAGE 4.1 Continuity at ∞ At ∞, things are a bit sticky:
- With a Magellan view, we can *see* if a function is continuous at ∞ or not.
- Technically, though, to talk of continuity at ∞ requires being able to take computational precautions not worth taking here but many teachers feel uneasy dealing with continuity at ∞ without taking these precautions. So, we will not discuss continuity at ∞ in this text.

EXAMPLE 4.25. The function whose global graph in Mercator view is



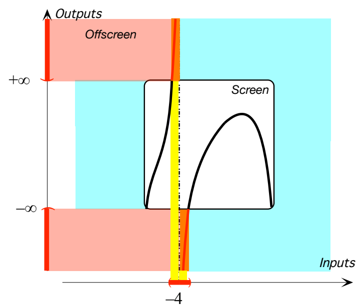
is *discontinuous* at ∞ because, even though the outputs of inputs near ∞ are all $large$,the global graph has a gap at ∞ since Local extreme-height inputs.

If we imagine the Mercator view *compactified* into a Magellan view, we have



Magellan continuous at

EXAMPLE 4.26. The function



is *discontinuous* at -4 because the global graph has a **pole** at -4:

- the outputs for nearby inputs, both inputs *right* of -4 and inputs *left* of -4, are all *large*,

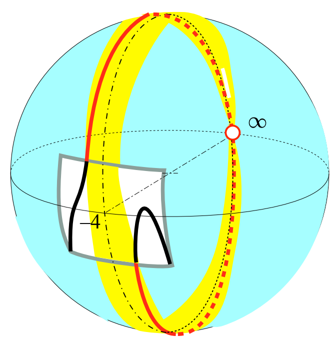
but, since Local extreme-height inputs,

- -4 itself has no output.

5. Magellan continuity at a pole x_0. A function is **Magellan continuous at** x_0 when we could remove the discontinuity at x_0 by overriding or supplementing the global input-output rule with an input-output table involving ∞ as Magellan output.

EXAMPLE 4.27. The function in Example 4.12 is *discontinuous* at -4 because the function has a gap at -4 but *Magellan continuous* as we could *remove* the gap by *supplementing* the global input-output rule with the input-output table

Input	Output
-4	∞



8 Smoothness near ∞

Think globally, act locally.[1]

Several Mathematicians[2]

global feature

Chapter 5

Global Analysis

that is the largest error that will not change the qualitative information we are looking for. The largest permissible error, which is the equivalent of a tolerance, will turn out to be easy to determine.

We can see from Chapter 3 that the reasoncould not possibly give us a global graph is that, if a plot point may tell us where the global graph "is at", a plot point certainly cannot tell us anything about where the global graph "goes from there". And, since the latter is precisely what local graphs do with slope and concavity, we are now in a position to:

i. *Describe* how to interpolate local graphs into a global graph. This corresponds to the second of the about

ii. *Describe* and *name* **global features** that a *function* may or may not have. As opposed to local features, which involved only inputs near a given input, global features will involve *all* inputs.

iii. Discuss questions about interpolating local graphs which correspond to the other two

i. How will we know near which inputs to get the local graphs?

ii. After we have interpolated the local graphs, how will we know if the curve we got *is* the global graph?

[1] https://en.wikipedia.org/wiki/Think_globally%2C_act_locally

[2] Educologists may want to look up https://math.stackexchange.com/questions/34053/list-of-local-to-global-principles

joining curve
interpolate
transition input
compatible
essential interpolation
essential
forced

Here again, to help focussing, the functions in the **EXAMPLES** in this chapter will always be presumed to have been defined by a global input-output rule but and the global graph of the function will be provided instead.

1 Interpolation

Interpolating will be for local graphs what joining cannot be for plot points, that is, interpolating local graphs will eventually provide us with global graphs.

1. Just to be part of the graph of a *function*, the **joining curve** we draw from one local graph to the other local graph will have to meet

i. The FUNDAMENTAL PROBLEM.

But, in order for a joining curve to be an **interpolation** of local graphs of *algebraic* functions which, as we will see, are continuous at all inputs as well as smooth near all inputs,

ii. The joining curve will itself have to be continuous at, as well as smooth near, all inputs,

iii. The joining curve will also have to be:

- Continuous at, as well as smooth near, the **transition inputs**, that is the inputs where the joining curve meets the local graphs,
- **Compatible** with the local graphs, that is the slope-sign and the concavity-sign will have to be the same on both sides of the transition inputs.

A particularly important kind of interpolation will be **essential interpolations** that is interpolations in which

iv. The joining curve is **essential**, that is has *only* local features that are **forced** by the local graphs being interpolated.

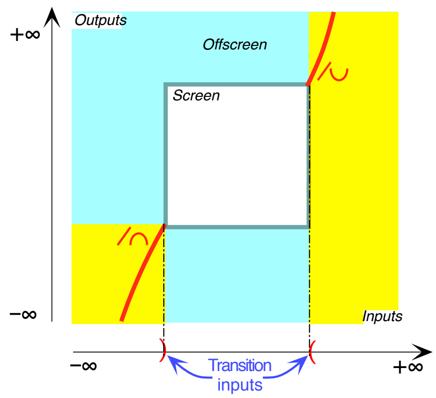
Which local graphs we will interpolate will depend on the answer in each case to Explicit Functions.

We will now illustrate these requirements with **EXAMPLES** that will show what makes a joining curve an interpolation of local graphs and what prevents a joining curve from being an interpolation of local graphs.

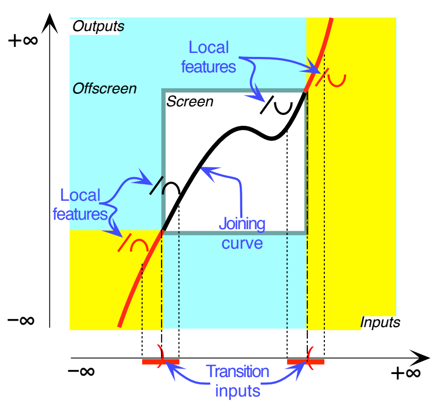
2. When the answer to the Explicit Functions is that the function *does not* have a pole, the offscreen graph consists of just the local graph near ∞ and therefore we interpolate with a joining curve from one end of the local graph near ∞ across the screen to the other end. The transition inputs are thus the lower bound and the upper bound.

In Example 5.1 to Example 5.5 we will examine whether or not the joining curve is an interpolation for the

DATA 5.1 Function with the offscreen graph



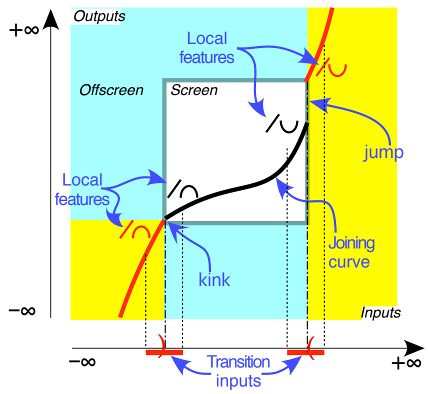
EXAMPLE 5.1. For functions whose offscreen graph is as in Data 5.1, the *joining curve*



i. is *continuous* at all inputs and *smooth* near all inputs,
ii. is *continuous at*, as well as *smooth near*, both transition inputs,
iii. is *compatible* with the offscreen graph near both transition inputs,
iv. is *not essential* because the local minimum and the local maximum are *not* forced by the offscreen graph.

So: This *joining curve is* an *interpolation* of the offscreen graph but *not* an *essential interpolation*.

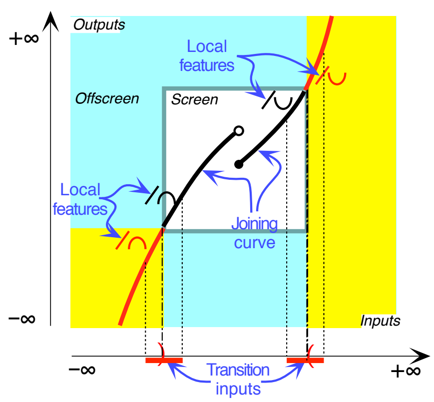
EXAMPLE 5.2. For functions whose offscreen graph is as in Data 5.1, the *joining curve*



i. is *continuous* at all inputs and *smooth* near all inputs,
ii. is *not smooth near* the left transition input and *not continuous at* the right transition input,
iii. is *compatible* with the offscreen graph near both transition inputs,
iv. is *essential*

So: This *joining curve* is *not* an *interpolation* of the offscreen graph and therefore *not* an *essential interpolation* either.

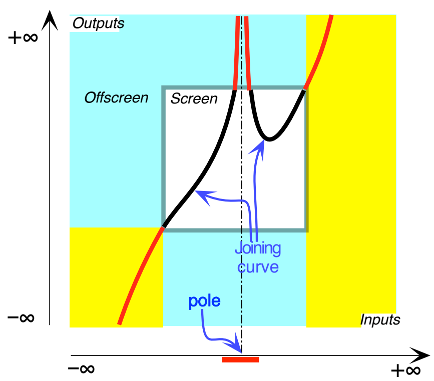
EXAMPLE 5.3. For functions whose offscreen graph is as in Data 5.1, the *joining curve*



i. is *not continuous* at all inputs but is *smooth* near all inputs,
ii. is *continuous at* the *transition inputs* as well as *smooth near* both *transition inputs*,
iii. is *compatible* with the offscreen graph near both transition inputs,
iv. is *not essential* because the jump is *not* forced by the offscreen graph.

So: This *joining curve* is *not* an *interpolation* of the offscreen graph and therefore *not* an *essential interpolation* either.

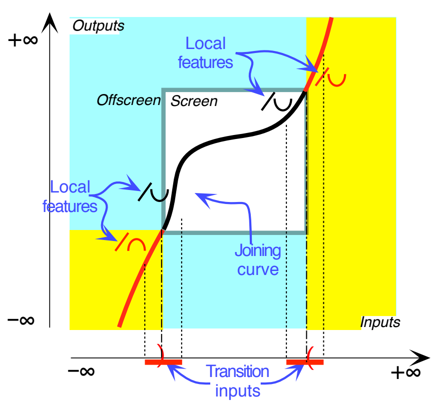
EXAMPLE 5.4. For functions whose offscreen graph is as in Data 5.1, the joining curve



introduces a *pole* in the *offscreen graph* so the function is *not* as in Data 5.1 anymore and nothing *onscreen* matters after that.

So: This *joining curve* is *not* an *interpolation* of the offscreen graph and therefore *not* an *essential interpolation* either.

EXAMPLE 5.5. For functions whose offscreen graph is as in Data 5.1, the *joining curve*



i. is *continuous* at all inputs and is *smooth* near all inputs,
ii. is both *continuous* at the *transition inputs* and *smooth* near the *transition inputs*,
iii. is *compatible* with the offscreen graph near the right transition input but is *not compatible* with the offscreen graph near the left transition input (Easy to miss.),
iv. is *not essential*.(Easy to miss.)

So: This *joining curve* is *not* an *interpolation* of the offscreen graph and therefore *not* an *essential interpolation* either.

3. When the answer to the Explicit Functions is that the function *does* have a pole $x_{\infty\text{-height}}$, the offscreen graph consists of the local graph near ∞ together with the local graph near $x_{\infty\text{-height}}$ and therefore we must interpolate with a joining curve in two pieces:

- one piece between:
  - ▸ the end of the right side of the local graph near ∞

  and
  - ▸ the end of the left side of the local graph near $x_{\infty\text{-height}}$,

and

- another piece between:

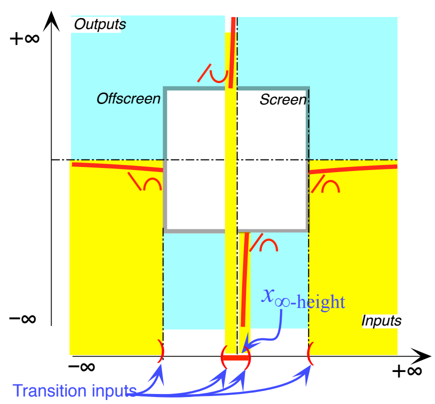
- ▸ the end of the right side of the local graph near $x_{\infty\text{-height}}$,
and
- ▸ the end of the left side of the local graph near ∞

The transition inputs are then:

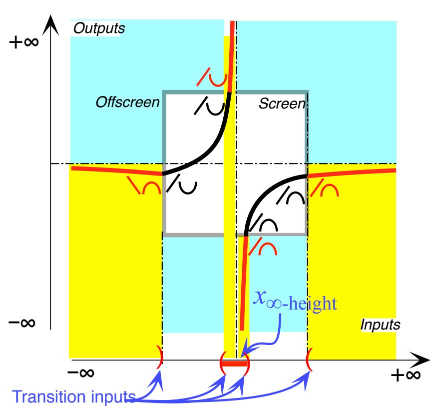
- the lower bound and the extremity of the left side of the neighborhood of $x_{\infty\text{-height}}$
- the extremity of the right side of the neighborhood of $x_{\infty\text{-height}}$ and the upper bound.

In Example 5.6 and Example 5.7 we will examine whether or not the joining curve is an interpolation for the

DATA 5.2 Function with the offscreen graph



EXAMPLE 5.6. For functions whose offscreen graph is as in Data 5.2, the *joining curve*

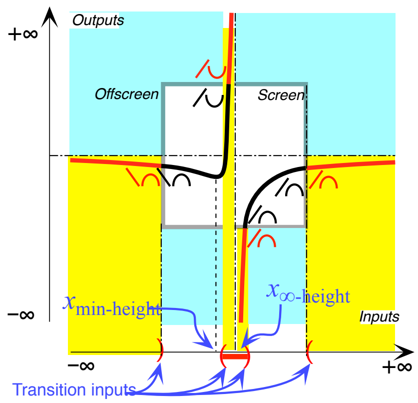


i. is *continuous* at all inputs and is *smooth* near all inputs,
ii. is *continuous* at all *transition inputs* and *smooth* near all *transition inputs* except the leftmost *transition input*,
iii. is *compatible* with the offscreen graph near all transition inputs except the leftmost *transition input*,
iv. is *essential*.

So: This *joining curve* is *not* an *interpolation* of the offscreen graph and therefore *not* an *essential interpolation* either.

fudge

EXAMPLE 5.7. For functions whose offscreen graph is as in Data 5.2, the *joining curve*



i. is *continuous* at all inputs and is *smooth* near all inputs,
ii. is *continuous* at all *transition inputs* and *smooth* near all *transition inputs*,
iii. is *compatible* with the offscreen graph near all transition inputs,
iv. is *essential* because $x_{\text{min-height}}$ is forced by the offscreen graph.

So: This *joining curve is* an *interpolation* of the offscreen graph and *is* an *essential interpolation*.

4. Occasionally, we will need to interpolate with an onscreeen local graph near a bounded input (As opposed to a local graph near a pole which is offscreen).

5. Occasionally, we will need to **fudge** the offscreen graph that is .

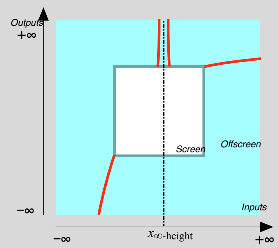
6. So, based on the preceding **EXAMPLES**, to draw an interpolation, we proceed as follows

PROCEDURE 5.1 Interpolate an offscreen graph.

i. Going from left to right, mark the features where the *offscreen graph* **enters the screen** and where the *offscreen graph* **exits the screen**
ii. Draw the joining curve(s) from the point(s) where the offscreen graph enters the screen to the point(s) where the offscreen graph exits the screen making sure that:

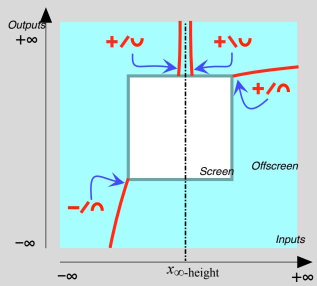
- Each *joining curve* is *smooth*,
- Each *transition* between a joining curve and the local graph is *smooth*
- The joining curves do *not* introduce any *infinite height* input.

TEMO 5.1 Let f be the function whose *offscreen* graph is

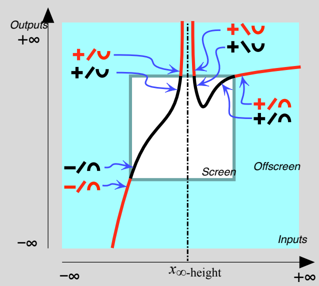


To interpolate,

i. Mark the features where the *offscreen graph* enters and exits the screen:



ii. Draw the joining curve(s) *smoothly*

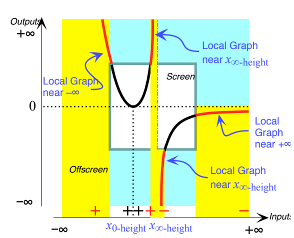


2 Feature Sign-Change Inputs

We will often need to find *bounded* inputs such that the outputs for nearby inputs left of x_0 and the outputs for nearby inputs right of x_0 have specified feature-signs.

1. An input is a **Height sign-change input**whenever Height sign $= \langle +, - \rangle$ or $\langle -, + \rangle$. We will use $x_{\textbf{Height sign-change}}$to refer to a *bounded* Height sign-change input.

Example 5.8.
Let f be the function specified by the global graph

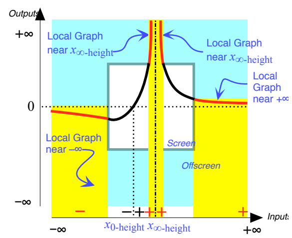


Then,

- $x_{0\text{-height}}$ *is not* a Height sign-change input,
- $x_{\infty\text{-height}}$ *is* a Height sign-change input.
- ∞ *is* a Height sign-change input.

Example 5.9.

Let f be the function specified by the global graph



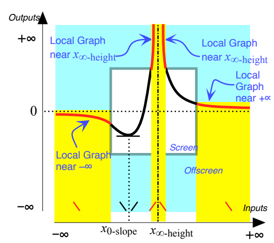
Then,

- $x_{0\text{-height}}$ *is* a Height sign-change input,
- $x_{\infty\text{-height}}$ *is not* a Height sign-change input,
- ∞ *is* a Height sign-change input.

2. An input is a **Slope sign-change input** whenever Slope sign $= \langle\diagup, \diagdown\rangle$ or $\langle\diagdown, \diagup\rangle$. We will use $x_{\textbf{Slope sign-change}}$ to refer to a Slope sign-change input.

Example 5.10.

Let f be the function specified by the global graph

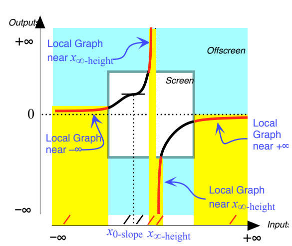


Then,

- $x_{0\text{-slope}}$ *is* a Slope sign-change input,
- $x_{\infty\text{-height}}$ *is* a Slope sign-change input,
- ∞ *is not* a Slope sign-change input.

Example 5.11.

Let f be the function specified by the global graph



Then,

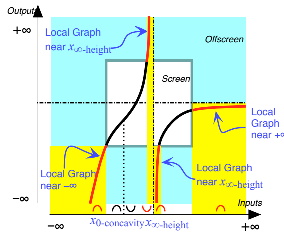
- $x_{0\text{-slope}}$ *is not* a Slope sign-change input,
- $x_{\infty\text{-slope}}$ *is not* a Slope sign-change input,
- ∞ *is not* a Slope sign-change input.

3. An input is a **Concavity sign-change input** whenever Concavity sign $= \langle\cup, \cap\rangle$ or $\langle\cap, \cup\rangle$. We will use $x_{\textbf{Concavity sign-change}}$ to refer to a Concavity sign-change input.

Example 5.12.

essential

Let f be the function specified by the global graph

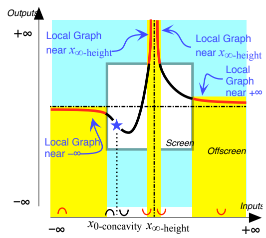


Then,

- $x_{0\text{-concavity}}$ *is* a Concavity sign-change input,
- $x_{\infty\text{-height}}$ *is* a Concavity sign-change input.
- ∞ *is not* a Concavity sign-change input.

EXAMPLE 5.13.

Let f be the function specified by the global graph



Then,

- $x_{0\text{-concavity}}$ *is* a Concavity sign-change input,
- $x_{\infty\text{-height}}$ *is not* a Concavity sign-change input,
- ∞ *is* a Concavity sign-change input.

3 Essential Feature Sign-Changes Inputs

1. A feature sign-change input is **essential** whenever its **existence**is **forced**by the offscreen graph. So, given the offscreen graph of a function, in order

PROCEDURE 5.2 Establish the existence of essential feature sign change inputs in a joining curve

i. For each piece of the joining curve, check the feature sign at both end of the piece.

- If the feature signs at the two ends of the piece are *opposite*, there *has* to be a feature sign change input for that piece.
- If the feature signs at the two ends of the piece are the *same*, there does *not* have to be a feature sign change input for that piece.

ii. For each ∞ height input, if any, check the feature sign on either side of the ∞ height input:

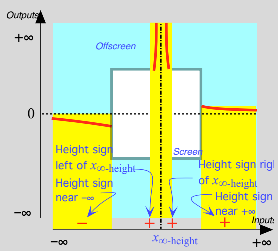
- If the feature signs on the two sides of the ∞ height input are *opposite*, the ∞ height input *is* a feature sign change input.
- If the feature signs on the two sides of the ∞ height input are the

same, the ∞ height input *is not* a feature sign change input..

iii. Check the feature sign on the two sides of ∞

- If the feature signs on the two sides of ∞ are *opposite*, ∞ *is* a feature sign change input.
- If the feature signs on the two sides of ∞ are the *same*, ∞ *is not* a feature sign change input..

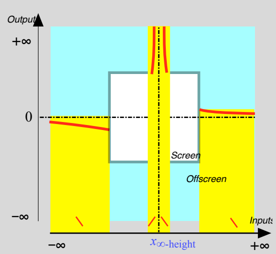
TEMO 5.2 Let f be the function whose *offscreen* graph is



To establish the existence of Height-sign change inputs

- Since the Height signs near $-\infty$ and left of $x_{\infty\text{-height}}$ are *opposite* there *is* an essential Height sign-change between $-\infty$ and $x_{\infty\text{-height}}$.
- Since the Height signs right of $x_{\infty\text{-height}}$ and near $+\infty$ are *the same* there *is no* essential Height sign-change between $x_{\infty\text{-height}}$ and $+\infty$.

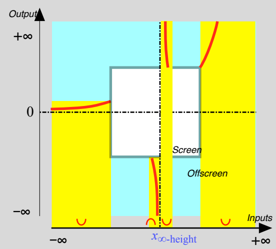
TEMO 5.3 Let f be the function whose *offscreen* graph is



To establish the existence of Slope-sign change inputs

- Since the Slope signs near $-\infty$ and left of $x_{\infty\text{-height}}$ are *opposite* there *is* an essential Slope sign-change between $-\infty$ and $x_{\infty\text{-height}}$.
- Since the Slope signs right of $x_{\infty\text{-height}}$ and near $+\infty$ are *the same* there *is no* essential Slope sign-change between $x_{\infty\text{-height}}$ and $+\infty$.

TEMO 5.4 Let f be the function whose *offscreen* graph is

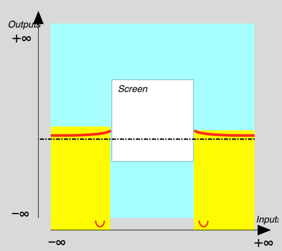


To establish the existence of Concavity-sign change inputs

- Since the Concavity signs near $-\infty$ and left of $x_{\infty\text{-height}}$ are *opposite* there *is* an essential Concavity sign-change between $-\infty$ and $x_{\infty\text{-height}}$.
- Since the Concavity signs right of $x_{\infty\text{-height}}$ and near $-\infty$ are *the same* there *is no* essential Concavity sign-change between $x_{\infty\text{-height}}$ and $+\infty$.

2. However, things can get a bit more complicated.

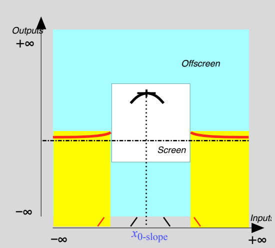
TEMO 5.5 Let f be the function whose *offscreen* graph is



To establish the existence of Concavity-sign change inputs

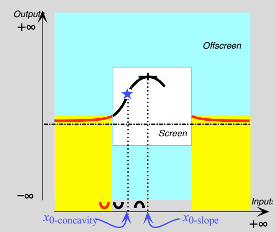
- Since the concavity-sign at the transitions from $-\infty$ is $\cup$ and the concavity-sign at the transition to $+\infty$ is also $\cup$, one might be tempted to say that there is no essential concavity sign-change input.
- However, attempting a smooth interpolation shows that things are a bit more complicated than would at first appear.

i. Since the slope-signs at the transition *from* $-\infty$ is $\diagup$ and the slope-sign at the transition *to* $+\infty$ is $\diagdown$ there has to be an essential Slope sign-change input near which Concavity sign $= \langle \cap, \cap \rangle$

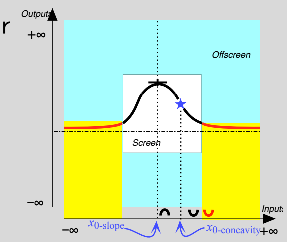


essential␣local␣extreme-height␣input

ii. Since the concavity-signs near $-\infty$ and *left* of $x_{0\text{-slope}}$ are *opposite*, there is an essential Concavity sign-change input between $-\infty$ and $x_{0\text{-slope}}$.

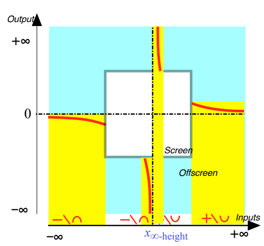


iii. Since the concavity-signs *right* of $x_{0\text{-slope}}$ and near $+\infty$ are *opposite*, there is an essential Concavity sign-change input between $x_{0\text{-slope}}$ and $+\infty$.

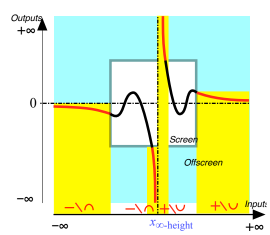


3. That there is no *essential* feature sign-change input does not mean that there could not actually be a *non-essential* feature sign-change input.

EXAMPLE 5.14.
Let f be the function whose offscreen graph is



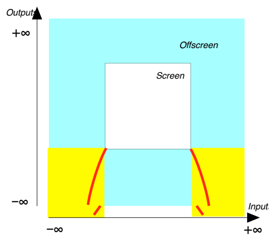
- There is no *essential* Height sign-change input, no *essential* Slope sign-change input, and no *essential* Concavity sign-change input.
- However, the actual bounded graph could very well be:



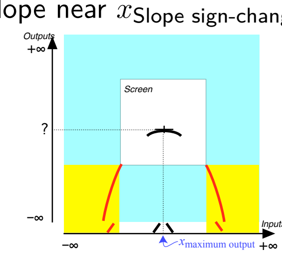
4 Essential Extreme-Height Inputs

An extreme-height input is an **essential local extreme-height input** if the existence of the local extreme-height input is forced by the offscreen graph in the sense that *any* smooth interpolation *must* have a local extreme-height input.

EXAMPLE 5.15.
Let f be a function whose offscreen graph is

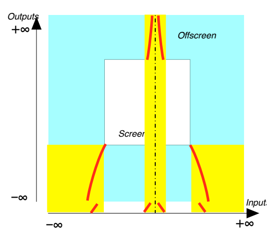


Then,
i. Since the Slope signs near $-\infty$ and $+\infty$ are *opposite* there *is* an essential Slope sign-change between $-\infty$ and $+\infty$.
ii. Since the Height of $x_{\text{Slope sign-change}}$ is not infinite, the slope near $x_{\text{Slope sign-change}}$ must be 0



iii. $x_{0\text{-slope}}$ is a local essential Maximum-Height input.

EXAMPLE 5.16.
Let f be a function whose offscreen graph is



Then,
i. Since the Slope signs near $-\infty$ and near $+\infty$ are *opposite* there *is* an essential Slope sign-change between $-\infty$ and $+\infty$.
ii. But since there is an ∞-height input, the Height near $x_{slopesign-change}$ is infinite and there is no essential local maximum height input.

5 Non-essential Features

While, as we have just seen, the *offscreen graph* may force the existence of certain feature-sign changes in the *onscreen graph*, there are still many other smooth interpolations of the *offscreen graph* that are not forced by the onscreen graph.

EXAMPLE 5.17. The moon has an influence on what happens on earth—for instance the tides—yet the phases of the moon do not seem to have an influence on the growth of lettuce (see http://www.almanac.com/content/

bump
wiggle

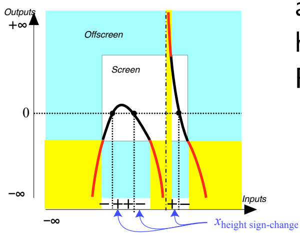
farming-moon) or even on the mood of the math instructor.

We will say that a global feature is **non-essential**if it is *not* forced by the offscreen graph.

1. As we saw above, feature sign-change inputs can be non-essential.

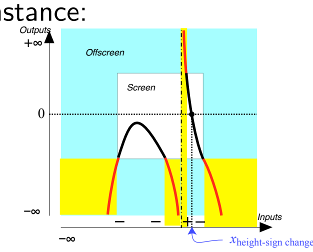
EXAMPLE 5.18.

Let f be a function whose graph is



Then,

i. The two Height sign-change inputs left of $x_{\infty\text{-height}}$ are non-essential because if the 0-output level line were higher, there would be no Height sign-change input. For instance:

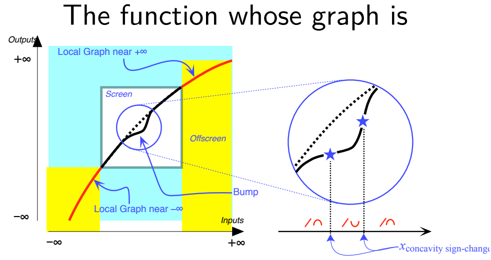


ii. The Height sign-change input right of $x_{\infty\text{-height}}$ is essential because, no matter where the 0-output level line might be, the joining curve has to cross it.

2. There other non-essential features:

- A *smooth* function can have a **bump** in which the slope does not change sign but the concavity changes sign twice.

EXAMPLE 5.19. The function whose graph is

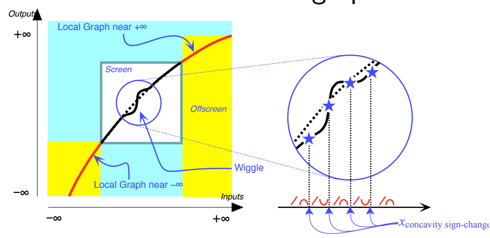


has a *bump*.

- A *smooth* function can also have a **wiggle**, that is a pair of bumps in opposite directions with the slope keeping the same sign throughout but with *three* inputs where the concavity changes sign.

max-min␣fluctuation
min-max␣fluctuation

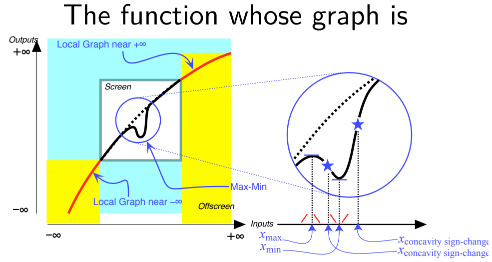
EXAMPLE 5.20. The function whose graph is



has a *wiggle*.

- A *smooth* function can also have a **max-min fluctuation** or a **min-max fluctuation** that is a sort of "extreme wiggle" which consists of a pair of *extremum-heights inputs* in opposite directions. In other words, a fluctuation involves:
  - *two* inputs where the *slope* changes sign
  - *two* inputs where the *concavity* changes sign

EXAMPLE 5.21. The function whose graph is



has a *max-min fluctuation*.

6 Essential Onscreen Graph

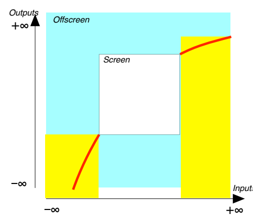
It should be realized that in each and everyone of the above **EXAMPLES** we were only able to determine *how many* **essential inputs**

> **NOTE 5.1 Location of essential inputs** *Locating* essential inputs is a totally different question from finding *how many* essential inputs there are. *Locating* essential inputs is usually a much more difficult question which, except in a very few cases, we will not deal with in this text.

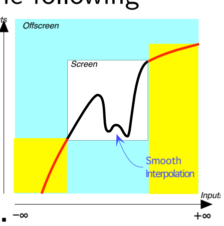
We will thus use the following

> **DEFINITION 5.1 An essential onscreen graph** is a **simplest** possible smooth interpolation of the offscreen graph, that is without any *nonessential* feature-sign change inputs and without any *nonessential* features.

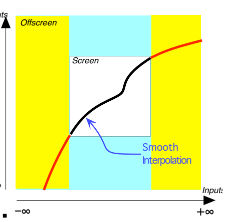
EXAMPLE 5.22. Given the offscreen graph,



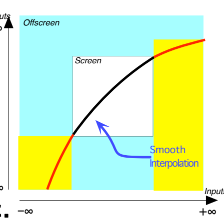
the following



a.



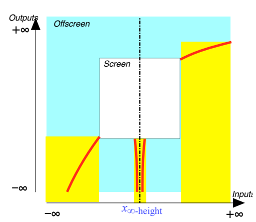
b.



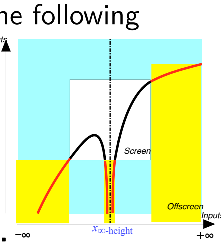
c.

are all smooth interpolations but only **c.** is an *essential* onscreen graph.

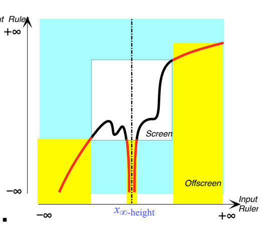
EXAMPLE 5.23. Given the offscreen graph,



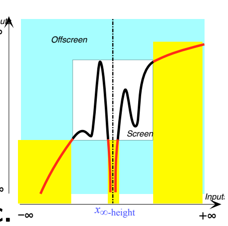
the following



a.



b.



c.

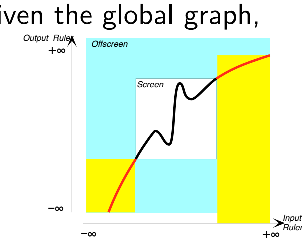
are all smooth interpolations but only **a.** is an *essential* onscreen graph.

1. The *essential onscreen graph* will be about the best we will be able to get with the technology in this text and, in order to detect, locate and investigate nonessential features such as *bumps*, *hiccups* and *fluctuations*, one needs the stronger technology of the Differential Calculus

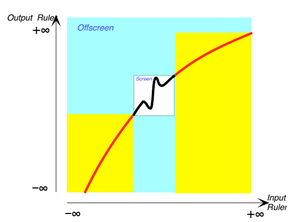
2. There are two ways *essential onscreen graphs* come up in the real world:

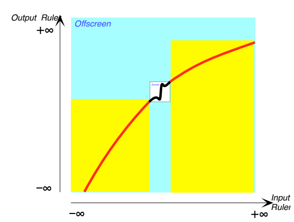
- The *essential onscreen graph* is how we see the actual graph from "faraway" inasmuch as nonessential features such as *bumps*, *hiccups* and *fluctuations* are too small to be seen from faraway.

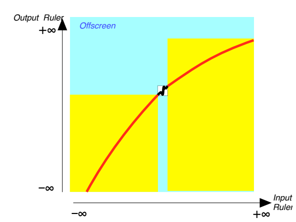
EXAMPLE 5.24. Given the global graph,

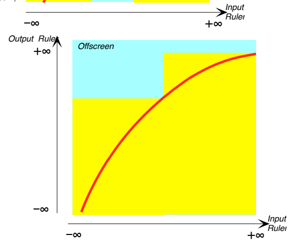


here is what we see from further and further away:



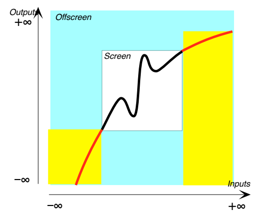




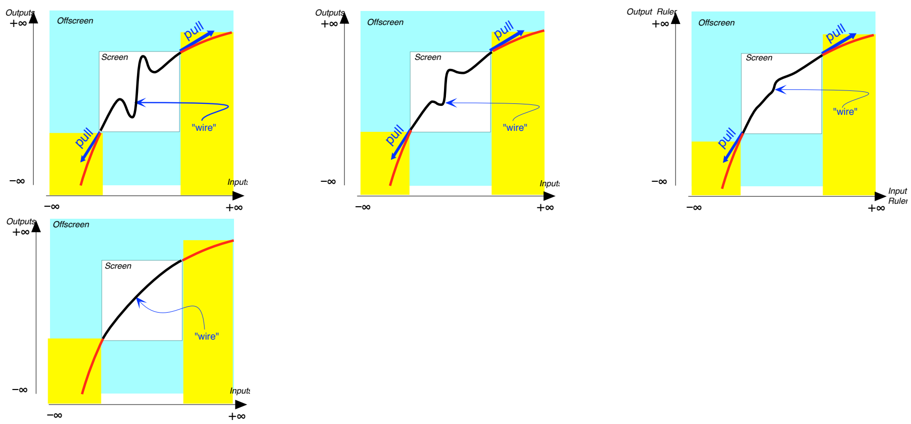


- The *essential onscreen graph* is what we would get if the onscreen graph were a wire being pulled out so as to straighten it.

EXAMPLE 5.25. Given the global graph,



we can imagine the *non-essential* onscreen graph as a "wire" being pulled by the offscreen graph so as to smooth it out into an *essential* bounded graph.



monomial function
coefficient
output-specifying code
exponent
power

Chapter 6

Regular Monomial Functions - Local Analysis

Monomial functions are functions that *multiply* or *divide* a given number, referred to as the **coefficient**, by a number of *copies of the input*.

1. More precisely,

DEFINITION 6.1 Monomial Functions are algebraic functions whose global input-output rule is of the form

$$\underbrace{x}_{\text{input}} \xrightarrow{f} \underbrace{f(x)}_{\text{output}} = \underbrace{\boxed{\text{coefficient}}\ \boxed{x^{\text{exponent}}}}_{\textbf{output-specifying code}}$$

where:

- ▶ The $\boxed{\text{coefficient}}$ can be any *bounded* number.
- ▶ The **exponent** in the **power** $\boxed{x^{\text{exponent}}}$ is a *signed counting* number that specifies what the function is to do to the *coefficient* with the copies of x:
  - The *size* of the exponent specifies *how many copies* of x are to be made. (If the exponent is 0, no copy is to be made and the coefficient is to be left alone.)
  - The *sign* of the exponent specifies whether the coefficient is to

power function
regular monomial function
exceptional monomial function

be *multiplied* or to be *divided* by the copies of x:
$+$ means the coefficient is to be *multiplied* by the copies of x,
$-$ means the coefficient is to be *divided* by the copies of x.

LANGUAGE 6.1 Power Functions is the name that is normally used for those monomial functions whose *coefficient* is $+1$ or -1. Unfortunately, the name power function is often used in place of monomial function and, even more unfortunately, this was the case in the previous editions of *this* text.

2. For reasons that will appear shortly we will distinguish:

- The **regular monomial functions**, to be discussed in this and the next chapter, which are those monomial functions whose *exponent* is any signed counting number *other* than 0 or $+1$.

from

- The **exceptional monomial functions**, to be discussed in chapter 8, which are those monomial functions whose *exponent* is *either* 0 or $+1$.

1 Output At x_0

Let f be the *regular* monomial function specified by the global input-output rule

$$\underbrace{x}_{\text{input}} \xrightarrow{f} \underbrace{f(x)}_{\text{output}} = \underbrace{ax^{\pm n}}_{\text{output-specifying code}}$$

where n is the number of copies used by f, and let x_0 be the *specified input*. To get the output of the function f at the specified input x_0, we use **??** on **??** which, for regular monomial functions, becomes:

PROCEDURE 6.1 To get the output *at* x_0 of a regular monomial function f .

i. *Declare* that x is to be replaced by x_0

$$x\Big|_{x\leftarrow x_0} \xrightarrow{f} f(x)\Big|_{x\leftarrow x_0} = ax^{\pm n}\Big|_{x\leftarrow x_0}$$

which, once carried out, gives:

$$x_0 \xrightarrow{f} f(x_0) = \underbrace{ax_0^{\pm n}}_{\text{output-specifying code}}$$

ii. *Execute* the output-specifying code that is:

a. *Decode* the output-specifying code, that is write out the computations to be performed according to the output-specifying code.

b. *Perform* the computations specified by the output-specifying code and thus get the output $f(x_0)$;

• For *positive* exponents, the code specifies that the output $f(x_0)$ is obtained by multiplying the coefficient a by n copies of the specified input x_0:

$$f(x_0) = a \cdot \underbrace{x_0 \cdot \ \dots \ \cdot x_0}_{n \text{ copies of } x_0}$$

• For *negative* exponents, the code specifies that the output $f(x_0)$ is obtained by dividing the coefficient a by the n copies of the specified input x_0:

$$f(x_0) = \frac{a}{\underbrace{x_0 \cdot \ \dots \ \cdot x_0}_{n \text{ copies of } x_0}}$$

Demo 6.1 Let $FLIP$ be the function specified by the global input-output rule

$$x \xrightarrow{FLIP} FLIP(x) = (+527.31)x^{+11}$$

To get the output of the function $FLIP$ at -3:

i. We *declare* that x is to be replaced by -3

$$x\Big|_{x \leftarrow -3} \xrightarrow{FLIP} FLIP(x)\Big|_{x \leftarrow -3} = (+527.31)x^{+11}\Big|_{x \leftarrow -3}$$

which, once the replacement has been carried out, gives:

$$-3 \xrightarrow{FLIP} FLIP(-3) = \underbrace{(+527.31) \cdot (-3)^{+11}}_{\text{output-specifying code}}$$

ii. We *execute* the output-specifying code that is:

a. We *decode* the output-specifying code: since we have a *positive* exponent, the code specifies that the output $FLIP(-3)$ is obtained by multiplying the coefficient $+527.31$ by 11 copies of the specified input -3:

$$FLIP(-3) = (+527.31) \cdot \underbrace{(-3) \cdot \ \dots \ \cdot (-3)}_{11 \text{ copies of } -3}$$

b. We *perform* the computations specified by the code. Dealing separately with the $signs$ and the $sizes$, we have

$$= (527.31) \cdot \underbrace{(-) \cdot \ \dots \ \cdot (-)}_{11 \text{ copies of } -} \cdot \underbrace{(3) \cdot \ \dots \ \cdot (3)}_{11 \text{ copies of } 3}$$

and since

- by theorem 19.2 on page 365, an *odd* number of copies of $-$ multiply to $-$ and we get

$$= (527.31) \cdot (-) \cdot (177\,147)$$
$$= -93\,411\,384.57$$

The input-output pair is $(-3, -93\,411\,384.57)$

Demo 6.2 Let $FLOP$ be the function specified by the global input-output rule

$$x \xrightarrow{FLOP} FLOP(x) = (+3\,522.38)x^{-6}$$

To get the output of the function $FLOP$ at -3:

i. We *declare* that x is to be replaced by -3

$$x \Big|_{x \leftarrow -3} \xrightarrow{FLOP} FLOP(x) \Big|_{x \leftarrow -3} = (+3\,522.38)x^{-6} \Big|_{x \leftarrow -3}$$

which, once carried out, gives:

$$-3 \xrightarrow{FLOP} FLOP(-3) = \underbrace{(+3\,522.38) \cdot (-3)^{-6}}_{\text{output-specifying code}}$$

ii. We *execute* the output-specifying code that is:

a. We *decode* the output-specifying code: since we have a $negative$ exponent, the code specifies that the output $FLOP(-3)$ is obtained by *dividing* the coefficient $+3\,522.38$ by 6 copies of the specified input -3:

$$FLOP(-3) = \frac{+3\,522.38}{\underbrace{(-3) \cdot \ \dots \ \cdot (-3)}_{6 \text{ copies of } -3}}$$

b. We *perform* the computations specified by the code. Dealing separately with the $signs$ and the $sizes$, we have

$$= \frac{+3\,522.38}{\underbrace{(-) \cdot \ \dots \ \cdot (-)}_{even \text{ number of copies of } -} \cdot \underbrace{(3) \cdot \ \dots \ \cdot (3)}_{6 \text{ copies of } 3}}$$

and since,

- by theorem 19.2 on page 365, an *even* number of copies of $-$ multiply to $+$ and we get

$$= \frac{+3\,522.38}{(+)\cdot(729)}$$

$$= +4.8317 + [...]$$

The input-output pair is $(-3, +4.8317 + [...])$

2 Plot Point

Let f be the *regular* monomial function specified by the global input-output rule

$$\underbrace{x}_{\text{input}} \xrightarrow{f} f(x) = \underbrace{ax^{\pm n}}_{\text{output-specifying code}}$$

where n is the number of copies used by f, and let x_0 be the *specified input*. To plot the input-output pair for the specified input x_0, we use **??** on **??** which, in the case of regular monomial functions, becomes

PROCEDURE 6.2 To get the *plot point* for a specified *bounded* input

1. *To get* the output at the specified input using **??** on **??** to get the input-output pair,
2. *Locate* the plot point with **??** on **??**.

DEMO 6.3 Let $FLIP$ be the function specified by the global input-output rule

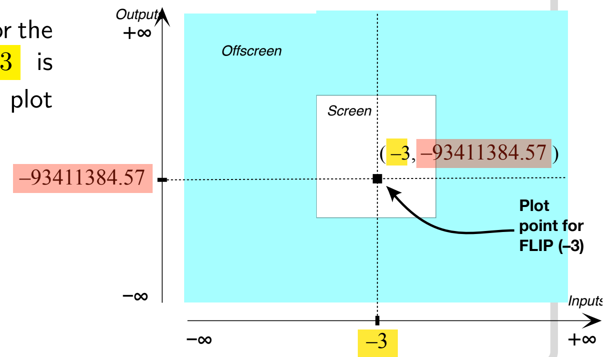
$$x \xrightarrow{FLIP} FLIP(x) = (+527.31)x^{+11}$$

To *plot* the input-output pair for the input -3:

1. We get the output of the function $FLIP$ at -3. We found in **EXAMPLE 5.1** above that $FLIP(-3) = -93\,411\,384.57$

features, of input-output rule
Coefficient Sign
Exponent Sign
Exponent Parity
even
odd

2. Thus, the *input-output pair* for the plot point of $FLIP$ at -3 is $(-3, -93\,411\,384.57)$ and the plot point is:



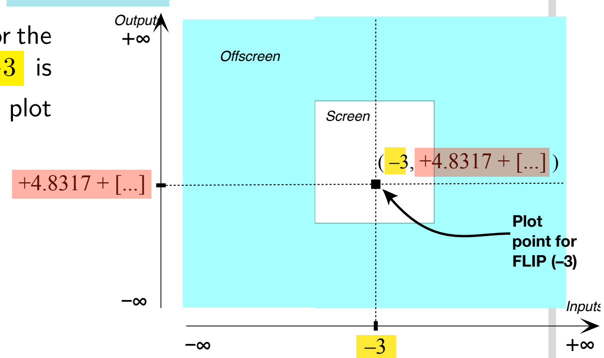
Demo 6.4 Let $FLOP$ be the function specified by the global input-output rule

$$x \xrightarrow{FLOP} FLOP(x) = (+3\,522.38)x^{-6}$$

To *plot* the input-output pair for the input -3:

1. We get the output of the function $FLOP$ at -3. We found in Demo 6.2 on page 156 that $FLOP(-3) = +4.8317 + [...]$

2. Thus, the *input-output pair* for the graph point of $FLOP$ at -3 is $(-3, +4.8317 + [...])$ and the plot point is:



3 Normalization

Since in this text we will take a *qualitative* viewpoint, all the **features** of the *global input-output rule* that specifies a regular monomial function will not be equally important for us.

As we will see, the three features that will be important for us are:

- **Coefficient Sign** which can be $+$ or $-$.
- **Exponent Sign** which can be $+$ or $-$,
- **Exponent Parity** which can be **even** or **odd** depending on whether the *size* of the exponent, that is the number of copies, is *even* or *odd*.

Coefficient Size
Exponent Size
normalize

DEMO 6.5 The function specified by the global input-output rule

$$x \xrightarrow{BLIP} BLIP(x) = (-160.42)x^{+7}$$

is a monomial function whose global input-output rule has the following *features*

- Coefficient Sign $BLIP = -$.
- Exponent Sign $BLIP = +$,
- Exponent Parity $BLIP = odd$,

But, because, in this text, we are only interested in *qualitative* analysis, we will not pay any attention to the following two features:

- **Coefficient Size** (other than the coefficient having to be *bounded*)
- **Exponent Size** (other than the size of the exponent being *even* or *odd*)

NOTE 6.1
A deeper analysis *would require* taking into account the actual number of copies but even then the size of the coefficient would still not matter much.

Accordingly, in order to focus on the important features of regular monomial functions, it will often be helpful to **normalize** the global input-output rule of a regular monomial function as follows:

PROCEDURE 6.3 Normalize the global I-O rule of a regular monomial function.

i. Replace the *Coefficient Size* by the word *bounded*,
ii. Replace the *Exponent Size* by the *Exponent Parity*

TEMO 6.1 Let $BLIP$ be the function specified by the global input-output rule

$$x \xrightarrow{BLIP} BLIP(x) = (-\ 160.42\ )x^{+\ 7}$$

To normalize $BLIP$.

i. We replace the *Coefficient Size*, namely 160.42, by the word *bounded*
ii. We replace the *Exponent Size*, namely 7, by the word *odd*

The *normalized* global input-output rule of $BLIP$ is thus

$$x \xrightarrow{BLIP} BLIP(x) = (-\ bounded\ ) \cdot x^{+\ odd}$$

TEMO 6.2 Let $BLOP$ be the function specified by the global input-output rule

$$x \xrightarrow{BLOP} BLOP(x) = (-\ 365.28\ )x^{-\ 6}$$

To normalize $BLOP$,

code
bi-level sign
$\pm$
$\mp$

i. We replace the *Coefficient Size*, namely 365.28, by the word *bounded*
ii. We replace the *Exponent Size*, namely 6, by the word *even*

The *normalized* global input-output rule of $BLOP$ is thus

$$x \xrightarrow{BLOP} BLOP(x) = (-\,\boxed{bounded}\,) \cdot x^{-\boxed{even}}$$

4 Thickening The Plot

As mentioned in on , instead of using *single inputs* to get *single plot points*, we will "thicken the plot" that is we will use *neighborhoods* of given inputs to get *graph places*. But to use *neighborhoods* with *global input-output rules*, we will first have to introduce **code** to be able to *declare* by what to replace x. And, since this at the very core of what we will be doing in the rest of this text, we want to proceed with the utmost caution.

Since we are dealing here with *regular monomial functions* we will only be interested in inputs *near* ∞ and/or inputs *near* 0 and so here all we will need is the *sign-size*.

In order to declare by what we want to replace x, we will use the following code:

Near	Side				Code
Infinity	Left	$0 \cdots\cdots \infty$	*positive*	$+\infty$	$+large$
	Right	$\infty \cdots\cdots 0$	*negative*	$-\infty$	$-large$
Zero	Left	$\infty \cdots\cdots 0$	*negative*	0^-	$-small$
	Right	$0 \cdots\cdots \infty$	*positive*	0^+	$+small$

For the input-output pairs on *one* side, we will basically use on but *declare* that x is to be replaced using the above code for the given input.

For the input-output pairs of *both* sides, we will use the **bi-level signs** $\pm$ and $\mp$ as follows:

Instead of	We can just write	and the I-O pair
$+ \longrightarrow +$ and $- \longrightarrow +$	$\pm \longrightarrow +$	$(\pm, +)$
$+ \longrightarrow -$ and $- \longrightarrow -$	$\pm \longrightarrow -$	$(\pm, -)$
$+ \longrightarrow +$ and $- \longrightarrow -$	$\pm \longrightarrow \pm$	$(\pm, \pm)$
$+ \longrightarrow -$ and $- \longrightarrow +$	$\pm \longrightarrow \mp$	$(\pm, \mp)$

5 Output Near ∞

1. When we want to thicken only one side of ∞, we proceed as follows:

PROCEDURE 6.4 To get the input-output pairs *on one side* of ∞.

1. *Normalize* the global input-input rule using **??** on **??**
2. *Declare* that x is to be replaced by $+large$ or $-large$
3. *Execute* the output-specifying code that is:
a. *Decode* the output-specifying code, that is write out the computations to be performed according to the output-specifying code.
b. *Perform* the computations specified by the code using theorem 19.2 on page 365 and theorem 1.2 on page 38 or theorem 1.3 on page 39

DEMO 6.6 Let $NADE$ be the function specified by the global input-output rule

$$x \xrightarrow{NADE} NADE(x) = (-83.91)x^{-5}$$

To get the input-output pairs *near* $+\infty$ for $NADE$:

i. We *normalize* $NADE$:

$$x \xrightarrow{NADE} NADE(x) = (-bounded)\; x^{-odd}$$

ii. We *declare* that x is to be replaced by $+large$

$$x\,\Big|_{x\leftarrow +large} \xrightarrow{NADE} NADE(x)\,\Big|_{x\leftarrow +large} = (-bounded)x^{-odd}\,\Big|_{x\leftarrow +large}$$

which, once carried out, gives:

$$+large \xrightarrow{NADE} NADE(+large) = \underbrace{(-bounded)(+large)^{-odd}}_{\text{output-specifying code}}$$

iii. We *execute* the output-specifying code that is:

a. We *decode* the output-specifying code: since the exponent is *negative*, we get the output $NADE(+large)$ by *dividing* the coefficient $-bounded$ by an odd number of copies of the specified input $+large$:

$$= \frac{-bounded}{\underbrace{(+large)\cdot\ \ldots\ \cdot(+large)}_{odd\text{ number of copies of }+large}}$$

b. We *perform* the computations specified by the code. Dealing separately with the $signs$ and the $sizes$, we have

$$= \frac{-bounded}{\underbrace{(+)\cdot\ \dots\ \cdot(+)}_{odd \text{ number of copies of } +} \cdot \underbrace{(large)\cdot\ \dots\ \cdot(large)}_{odd \text{ number of copies of } large}}$$

and since,

- by theorem 19.2 on page 365, *any* number of copies of $+$ multiply to $+$,
- by the **Definition** of $large$, *any* number of copies of $large$ multiply to $large$

$$= \frac{-bounded}{+\cdot large}$$

and by theorem 19.2 on page 365 and theorem 1.3 on page 39 we get

$$= -small$$

iv. The input-output pairs are $(+large, -small)$

DEMO 6.7 Let $RADE$ be the function specified by the global input-output rule

$$x \xrightarrow{\ RADE\ } RADE(x) = (+45.67)x^{-4}$$

To get the input-output pairs near $-\infty$ for $RADE$:

i. We *normalize* $RADE$:

$$x \xrightarrow{\ RADE\ } RADE(x) = (+bounded)\ x^{-even}$$

ii. We *declare* that x is to be replaced by $-large$

$$x\Big|_{x\leftarrow -large} \xrightarrow{\ RADE\ } RADE(x)\Big|_{x\leftarrow -large} = (+bounded)x^{-even}\Big|_{x\leftarrow -large}$$

which, once carried out, gives:

$$-large \xrightarrow{\ RADE\ } RADE(-large) = \underbrace{(+bounded)(-large)^{-even}}_{\text{output-specifying code}}$$

iii. We *execute* the output-specifying code that is:

a. We *decode* the output-specifying code: since the exponent is $negative$, we get the output $RADE(-large)$ by *dividing* the coefficient $+bounded$ by an $even$ number of copies of the specified input $-large$:

$$= \frac{+bounded}{\underbrace{(-large)\cdot\ \dots\ \cdot(-large)}_{even \text{ number of copies of } -large}}$$

b. We *perform* the computations specified by the code:

Dealing separately with the *signs* and the *sizes*, we have

$$= \frac{+bounded}{\underbrace{(-)\cdot \ \dots \ \cdot(-)}_{even \text{ number of copies of } -} \cdot \underbrace{(large)\cdot \ \dots \ \cdot(large)}_{even \text{ number of copies of } large}}$$

and since,

- by the **Sign Multiplication Rule**, any *even* number of copies of $-$ multiply to $+$
- by the **Definition** of $large$, *any* number of copies of $large$ multiply to $arge$

$$= \frac{+bounded}{+\cdot large}$$

and by the **Sign Division Rule** and the **Size Division Theorem**

$$= +small$$

iv. The input-output pairs are $(-large, +small)$

2. When we want to thicken both sides of ∞, we declare that x is to be replaced by $\pm large$ and keep track of the signs as we *perform the computations* specified by the output-specifying code.

Demo 6.8 Let $DADE$ be the function specified by the global input-output rule

$$x \xrightarrow{DADE} DADE(x) = (-83.91)x^{+5}$$

To get the input-output pairs *near* ∞ for $DADE$:

i. We *normalize* $DADE$:

$$x \xrightarrow{DADE} DADE(x) = (-bounded)\ x^{+odd}$$

ii. We *declare* that x is to be replaced by $\pm large$

$$x\Big|_{x\leftarrow\pm large} \xrightarrow{DADE} DADE(x)\Big|_{x\leftarrow\pm large} = (-bounded)x^{+odd}\Big|_{x\leftarrow\pm large}$$

which, once carried out, gives:

$$\pm large \xrightarrow{DADE} DADE(\pm large) = \underbrace{(-bounded)(\pm large)^{+odd}}_{\text{output-specifying code}}$$

iii. We *execute* the output-specifying code that is:

a. We *decode* the output-specifying code: since the exponent is $positive$, we get that the output $DADE(\pm large)$ is obtained by *multiplying* the coefficient $-bounded$ by an odd number of copies of the specified input $\pm large$:

$$= (-bounded)\cdot\underbrace{(\pm large)\cdot \ \dots \ \cdot(\pm large)}_{odd \text{ number of copies of } \pm large}$$

b. We *perform* the computations specified by the code. Dealing separately with the $signs$ and the $sizes$, we have

$$= (-bounded) \cdot \underbrace{(\pm) \cdot \ \ldots \ \cdot (\pm)}_{odd \text{ number of copies of } \pm} \cdot \underbrace{(large) \cdot \ \ldots \ \cdot (large)}_{odd \text{ number of copies of } large}$$

and since,

- by the **Sign Multiplication Rule**, an *odd* number of copies of $+$ multiply to $+$ and an *odd* number of copies of $-$ multiply to $-$
- by the **Definition** of $large$, *any* number of copies of $large$ multiply to $large$

$$= (-bounded) \cdot \pm \cdot large$$

and by the **Sign Multiplication Rule** and the **Size Multiplication Theorem**

$$= \mp large$$

iv. The input-output pairs are $(\pm large, \mp large)$

Demo 6.9 Let $PADE$ be the function specified by the global input-output rule

$$x \xrightarrow{PADE} PADE(x) = (-65.18)x^{+6}$$

To get the input-output pairs *near* ∞ for $PADE$

i. We *normalize* $PADE$.

$$x \xrightarrow{PADE} PADE(x) = (-bounded)\ x^{+even}$$

ii. We *declare* that x is to be replaced by $\pm large$

$$x\Big|_{x \leftarrow \pm large} \xrightarrow{PADE} PADE(x)\Big|_{x \leftarrow \pm large} = (-bounded)x^{+even}\Big|_{x \leftarrow \pm large}$$

which, once carried out, gives:

$$\pm large \xrightarrow{PADE} PADE(\pm large) = \underbrace{(-bounded)(\pm large)^{+even}}_{\text{output-specifying code}}$$

iii. We *execute* the output-specifying code that is:

a. We *decode* the output-specifying code: since the exponent is $positive$, we get the output $PADE(\pm large)$ by *multiplying* the coefficient $-bounded$ by an $even$ number of copies of the specified input $\pm large$:

$$= (-bounded) \cdot \underbrace{(\pm large) \cdot \ \ldots \ \cdot (\pm large)}_{even \text{ number of copies of } \pm large}$$

b. We *perform* the computations specified by the code. Dealing separately with the *signs* and the *sizes*, we have

$$= (-bounded) \cdot \underbrace{(\pm) \cdot \ \ldots \ \cdot (\pm)}_{even \text{ number of copies of } \pm} \cdot \underbrace{(large) \cdot \ \ldots \ \cdot (large)}_{even \text{ number of copies of } large}$$

and since,

- by the **Sign Multiplication Rule**, an *even* number of copies of $+$ multiply to $+$ and an *even* number of copies of $-$ multiply to $+$
- by the **Definition** of $large$, *any* number of copies of $large$ multiply to $large$

$$= (-bounded) \cdot + \cdot large$$

and by the **Sign Division Rule** and the **Size Division Theorem**

$$= -large$$

iv. The input-output pairs are $(\pm large, -large)$

6 Output Near 0

1. When we want to thicken only one side of 0, we proceed as follows:

PROCEDURE 6.5 To get the input-output pairs *on one side* of 0.

1. *Normalize* the global input-input rule using **??** on **??**
2. *Declare* that x is to be replaced by $+small$ or $-small$
3. *Execute* the output-specifying code that is:

a. *Decode* the output-specifying code, that is write out the computations to be performed according to the output-specifying code.

b. *Perform* the computations specified by the code using theorem 19.2 on page 365 and theorem 1.2 on page 38 or theorem 1.3 on page 39

DEMO 6.10 Let $MADE$ be the function specified by the global input-output rule

$$x \xrightarrow{MADE} MADE(x) = (+27.61)x^{+5}$$

To get the input-output pairs *near* 0^+ for $MADE$:

i. We *normalize* $MADE$:

$$x \xrightarrow{MADE} MADE(x) = (+bounded)\ x^{+odd}$$

ii. We *declare* that x is to be replaced by $+small$

$$x\Big|_{x\leftarrow +small} \xrightarrow{MADE} MADE(x)\Big|_{x\leftarrow +small} = (+bounded)x^{+odd}\Big|_{x\leftarrow +small}$$

which, once carried out, gives:

$$+small \xrightarrow{MADE} MADE(+small) = \underbrace{(-bounded)(+small)^{+odd}}_{\text{output-specifying code}}$$

iii. We *execute* the output-specifying code that is:

a. We *decode* the output-specifying code: since the exponent is $positive$, we get that the output $MADE(+small)$ is obtained by *multiplying* the coefficient $+bounded$ by an odd number of copies of the specified input $+small$:

$$= (+bounded) \cdot \underbrace{(+small) \cdot \ \dots \ \cdot (+small)}_{odd \text{ number of copies of } +small}$$

b. We *perform* the computations specified by the code. Dealing separately with the $signs$ and the $sizes$, we have

$$= (+bounded) \cdot \underbrace{(+) \cdot \ \dots \ \cdot (+)}_{odd \text{ number of copies of } +} \cdot \underbrace{(small) \cdot \ \dots \ \cdot (small)}_{odd \text{ number of copies of } small}$$

and since,

- by the **Sign Multiplication Rule**, *any* number of copies of $+$ multiply to $+$
- by the **Definition** of $small$, *any* number of copies of $small$ multiply to $small$

$$= (+bounded) \cdot + \cdot small$$

and by the **Sign Multiplication Rule** and the **Size Multiplication Theorem**

$$= +small$$

iv. The input-output pairs are $(+small, -small)$

DEMO 6.11 Let $WADE$ be the function specified by the global input-output rule

$$x \xrightarrow{WADE} WADE(x) = (-28.34)x^{-3}$$

To get the output of $WADE$ near 0^+

i. We *normalize* $WADE$:

$$x \xrightarrow{WADE} WADE(x) = (-bounded)\ x^{-even}$$

ii. We *declare* that x is to be replaced by $+small$

$$x\Big|_{x\leftarrow +small} \xrightarrow{WADE} WADE(x)\Big|_{x\leftarrow +small} = (-bounded)x^{-even}\Big|_{x\leftarrow +small}$$

which, once carried out, gives:

$$+small \xrightarrow{WADE} WADE(+small) = \underbrace{(-bounded)(+small)^{-even}}_{\text{output-specifying code}}$$

iii. We *execute* the output-specifying code that is:

a. We *decode* the output-specifying code: since the exponent is $negative$, we get the output $WADE(+small)$ by *dividing* the coefficient $-bounded$ by an $even$ number of copies of the specified input $+small$:

$$= \frac{-bounded}{\underbrace{(+small) \cdot \ \dots \ \cdot (+small)}_{even \text{ number of copies of } +small}}$$

iv. **b.** We *perform* the computations specified by the code. Dealing separately with the $signs$ and the $sizes$, we have

$$= \frac{-bounded}{\underbrace{(+) \cdot \ \dots \ \cdot (+)}_{even \text{ number of copies of } +} \cdot \underbrace{(small) \cdot \ \dots \ \cdot (small)}_{even \text{ number of copies of } small}}$$

and since,

- by the **Sign Multiplication Rule**, *any* number of copies of $+$ multiply to $+$
- by the **Definition** of $small$, *any* number of copies of $small$ multiply to $small$

$$= \frac{-bounded}{+ \cdot small}$$

and by the **Sign Division Rule** and the **Size Division Theorem**

$$= -large$$

iv. The input-output pairs are $(+small, -large)$

2. When we want to thicken both sides, we will declare that x is to be replaced by $\pm small$ and keep track of the signs as we *perform the computations* specified by the output-specifying code.

Demo 6.12 Let $JADE$ be the function specified by the global input-output rule

$$x \xrightarrow{JADE} JADE(x) = (-65.71)x^{-5}$$

To get the output of $JADE$ near 0,

i. We *normalize* $JADE$:

$$x \xrightarrow{JADE} JADE(x) = (-bounded)\ x^{-odd}$$

ii. We *declare* that x is to be replaced by $\pm small$

$$x\Big|_{x\leftarrow \pm small} \xrightarrow{JADE} JADE(x)\Big|_{x\leftarrow \pm small} = (-bounded)x^{-odd}\Big|_{x\leftarrow \pm small}$$

which, once carried out, gives:

$$\pm small \xrightarrow{JADE} JADE(\pm small) = \underbrace{(-bounded)(\pm small)^{-odd}}_{\text{output-specifying code}}$$

iii. We *execute* the output-specifying code that is:

a. We *decode* the output-specifying code: since the exponent is $negative$, we get the output $JADE(\pm small)$ by *dividing* the coefficient $-bounded$ by an odd number of copies of the specified input $\pm small$:

$$= \frac{-bounded}{\underbrace{(\pm small)\cdot \ \dots \ \cdot(\pm small)}_{odd \text{ number of copies of } \pm small}}$$

b. We *perform* the computations specified by the code. Dealing separately with the $signs$ and the $sizes$, we have

$$= \frac{-bounded}{\underbrace{(\pm)\cdot \ \dots \ \cdot(\pm)}_{odd \text{ number of copies of } \pm} \cdot \underbrace{(small)\cdot \ \dots \ \cdot(small)}_{odd \text{ number of copies of } small}}$$

and since,

- by the **Sign Multiplication Rule**, an *odd* number of copies of $+$ multiply to $+$ and an *odd* number of copies of $-$ multiply to $-$
- by the **Definition** of $small$, *any* number of copies of $small$ multiply to $small$

$$= \frac{-bounded}{\pm \cdot small}$$

and by the **Sign Division Rule** and the **Size Division Theorem**

$$= \mp large$$

iv. The input-output pairs are $(\pm small, \mp large)$

DEMO 6.13 Let $FADE$ be the function specified by the global input-output rule

$$x \xrightarrow{FADE} FADE(x) = (-65.18)x^{+6}$$

To get the input-output pairs *near* 0 for $FADE$:

i. We *normalize* $FADE$.

$$x \xrightarrow{FADE} FADE(x) = (-bounded)\ x^{+even}$$

ii. We *declare* that x is to be replaced by $\pm small$

$$x\Big|_{x\leftarrow \pm small} \xrightarrow{FADE} FADE(x)\Big|_{x\leftarrow \pm small} = (-bounded)x^{+even}\Big|_{x\leftarrow \pm small}$$

which, once carried out, gives:

$$\pm small \xrightarrow{FADE} FADE(\pm small) = \underbrace{(-bounded)(\pm small)^{+even}}_{\text{output-specifying code}}$$

iii. We *execute* the output-specifying code that is:

a. We *decode* the output-specifying code: since the exponent is $positive$, we get the output $FADE(-small)$ by *multiplying* the coefficient $-bounded$ by an $even$ number of copies of the specified input $\pm small$:

$$= (-bounded)\cdot\underbrace{(\pm small)\cdot\ \dots\ \cdot(\pm small)}_{even\text{ number of copies of }\pm small}$$

b. We *perform* the computations specified by the code. Dealing separately with the $signs$ and the $sizes$, we have

$$= (-bounded)\cdot\underbrace{(\pm)\cdot\ \dots\ \cdot(\pm)}_{even\text{ number of copies of }\pm}\cdot\underbrace{(small)\cdot\ \dots\ \cdot(small)}_{even\text{ number of copies of }small}$$

and since,

- by the **Sign Multiplication Rule**, an *even* number of copies of $+$ multiply to $+$ and an *even* number of copies of $-$ multiply to $+$
- by the **Definition** of $small$, *any* number of copies of $small$ multiply to $small$

$$= (-bounded)\cdot + \cdot small$$

and by the **Sign Multiplication Rule** and the **Size Multiplication Theorem**

$$= -small$$

iv. The input-output pairs are $(\pm small, -small)$

7 Graph Place Near ∞ and Near 0

Once we have the input-output pairs near ∞ and near 0, we get the graph places as in **?? ??** on **??**. Here again,

i. In the first four demos, Demo 6.14 on page 170, Demo 6.15 on page 170, Demo 6.16 on page 171, Demo 6.17 on page 171, we will deal with only one side or the other.

ii. In the next four demos, Demo 6.18 on page 172, Demo 6.19 on page 172, Demo 6.20 on page 173, Demo 6.21 on page 173, we will deal with both sides at the same time.

PROCEDURE 6.6 Locate the *graph place near* ∞ or 0

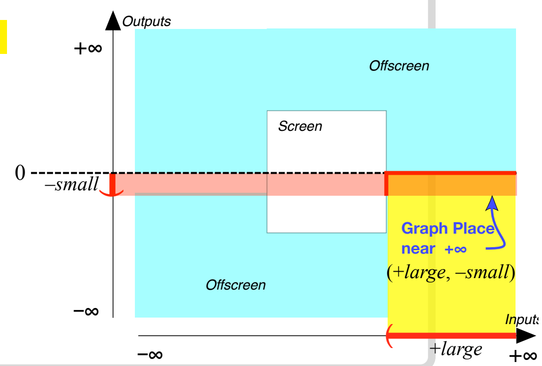
1. Get the input-output pairs using **?? ??** on **??** or **?? ??** on **??**.
2. *Locate* the *graph place* using **?? ??** on **??**.

DEMO 6.14 Let $NADE$ be the function specified by the global input-output rule

$$x \xrightarrow{NADE} NADE(x) = (-83.91)x^{-5}$$

To *locate* the *graph place* of $NADE$ near $+\infty$:

1. We get that the *input-output pairs* for $NADE$ near $+\infty$ are ($+large$,$-small$) (See Demo 6.6 on page 161)
2. The *graph place* of $NADE$ near $+\infty$ then is:



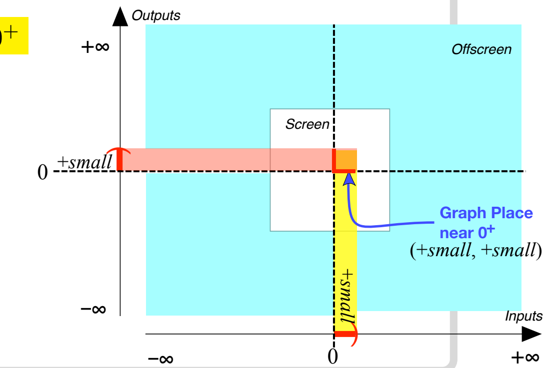
DEMO 6.15 Let $MADE$ be the function specified by the global input-output rule

$$x \xrightarrow{MADE} MADE(x) = (+27.61)x^{+5}$$

To *locate* the *graph place* of $MADE$ near 0^+:

1. We get that the *input-output pairs* for $MADE$ near 0^+ are [$+small$,$+small$](See Demo 6.10 on page 165)

2. The *graph place* of $MADE$ near 0^+ then is:



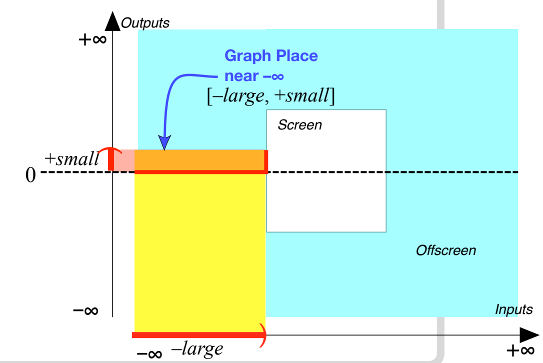
Demo 6.16 Let $RADE$ be the function specified by the global input-output rule

$$x \xrightarrow{RADE} RADE(x) = (+45.67)x^{-4}$$

To *locate* the *graph place* of $RADE$ near $-\infty$:

1. We get that the *input-output pairs* for $RADE$ near $-\infty$ are $[-large, +small]$(See Demo 6.7 on page 162)

2. The *graph place* near $-\infty$ then is:



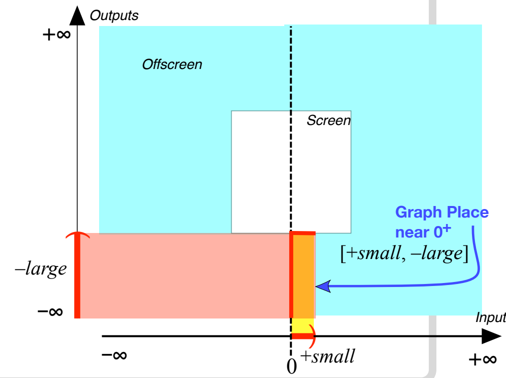
Demo 6.17 Let $WADE$ be the function specified by the global input-output rule

$$x \xrightarrow{WADE} WADE(x) = (-28.34)x^{-3}$$

To *locate* the *graph place* of $WADE$ near 0^+:

1. We get that the *input-output pairs* for $WADE$ near 0^+ are $[+small, -large]$(See Demo 6.17 on page 171)

2. The *graph place* near 0^+ then is:



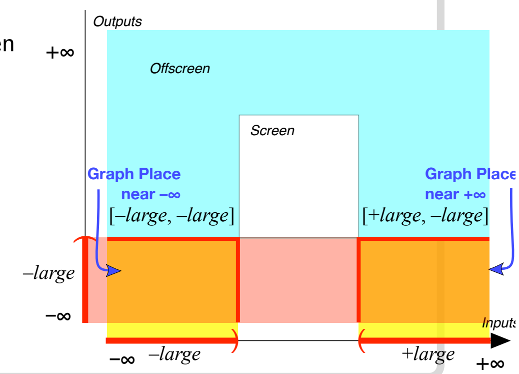
DEMO 6.18 Let $PADE$ be the function specified by the global input-output rule

$$x \xrightarrow{PADE} PADE(x) = (-65.18)x^{+6}$$

To *locate* the *graph place* of $PADE$ near ∞.

1. We get that the *input-output pairs* for $PADE$ near ∞ are $[\pm large, -large]$(See Demo 6.9 on page 164)

2. The *graph place* of $PADE$ near ∞ then is:



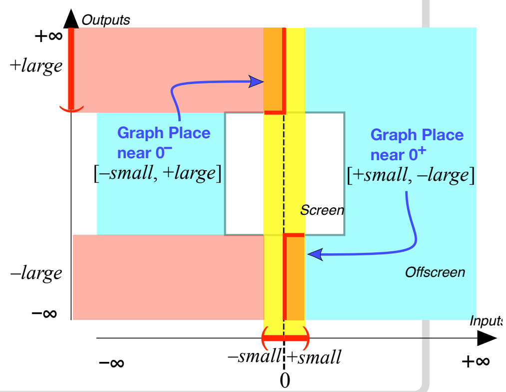
DEMO 6.19 Let $JADE$ be the function specified by the global input-output rule

$$x \xrightarrow{JADE} JADE(x) = (-65.71)x^{-5}$$

To *locate* the *graph place* of $JADE$ near 0:

1. We get that the *input-output pairs* for $JADE$ near 0 are $[\pm small, \mp large]$ (See Demo 6.12 on page 167)

2. The *graph place* of $JADE$ near $\boxed{0}$ then is:



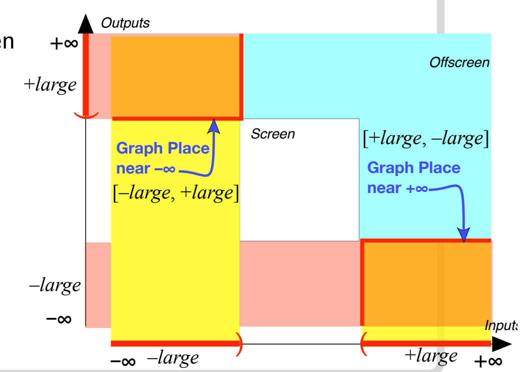
Demo 6.20 Let $DADE$ be the function specified by the global input-output rule

$$x \xrightarrow{DADE} DADE(x) = (-83.91)x^{+5}$$

To *locate* the *graph place* of $DADE$ near $\boxed{\infty}$:

1. We get that the *input-output pairs* for $DADE$ near $\boxed{\infty}$ are $[\pm large, \mp large]$ (See Demo 6.8 on page 163)

2. The *graph place* of $DADE$ near $\boxed{\infty}$ then is:



Demo 6.21 Let $FADE$ be the function specified by the global input-output rule

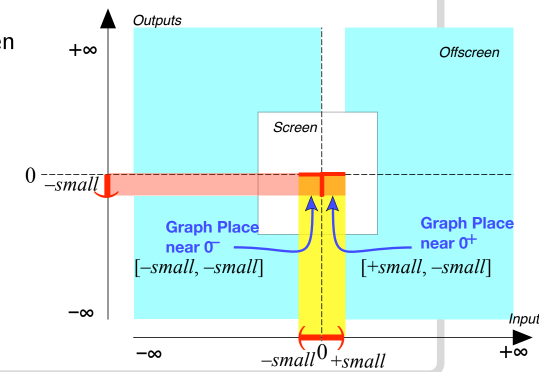
$$x \xrightarrow{FADE} FADE(x) = (-65.18)x^{+6}$$

To *locate* the *graph place* of $FADE$ near $\boxed{0^-}$.

1. We get that the *input-output pairs* for $FADE$ near $\boxed{0}$ are $[\pm small, -small]$ (See Demo 6.13 on page 168)

shape
forced

2. The *graph place* of $FADE$ near $\boxed{0}$ then is:



8 Local Graph Near ∞ and Near 0

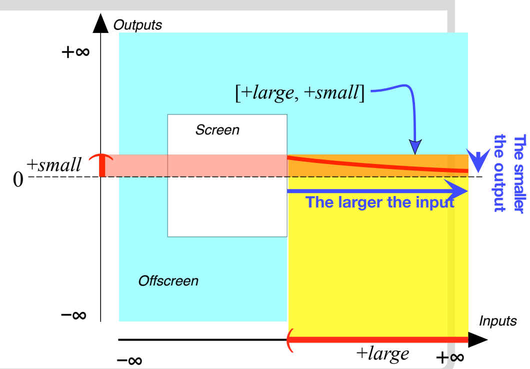
Regular *monomial functions* are very nice in that the **shapes** of the local graphs near ∞ and near 0 are **forced** by the *graph place*. In other words, once we know the *graph place*, there is only one way we can draw the *local graph* because:

i. The smaller or the larger the input is, the smaller or the larger the output will be,

ii. The local graph cannot escape from the place.

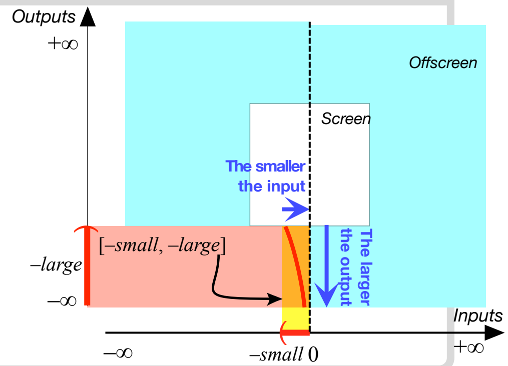
DEMO 6.22 Given a monomial function for which the place of a local graph is $[+large, +small]$, we get the *shape* of the local graph as follows

i. The *slope* is forced by the fact that the *larger* the input is, the *smaller* the output will be.
ii. The *concavity* is forced by the fact that the local graph cannot cross the 0-output level line.



Demo 6.23 Given a monomial function for which the place of a local graph is $[-small, -large]$, we get the *shape* of the local graph as follows

i. The *slope* is forced by the fact that the *smaller* the input is,
the *larger* the output will be.
ii. The *concavity* is forced by the fact that the local graph cannot cross the 0-input level line.



9 Local Features Near ∞ and Near 0

1. Given a regular monomial function being specified by a global input-output rule, to get the *Height sign* near ∞ or near 0, we need only compute the sign of the outputs for nearby inputs with the global input-output rule.

Demo 6.24 Let JOE be the function specified by the global input-output rule

$$x \xrightarrow{JOE} JOE(x) = (-65.18)x^{+6}$$

To get the *Height sign* of JOE near 0^+
We ignore the *size* and just look at the *sign*:

$$\begin{aligned} + \xrightarrow{JOE} JOE(+) &= (-)(+)^{+6} \\ &= (-)\cdot(+) \\ &= - \end{aligned}$$

and

$$\begin{aligned} - \xrightarrow{JOE} JOE(-) &= (-)(-)^{+6} \\ &= (-)\cdot(+) \\ &= - \end{aligned}$$

So, Height sign JOE near 0 is $\langle -, - \rangle$

2. Given a regular monomial function being specified by a global input-output rule, to get the *Slope sign* or the *Concavity sign* near ∞ or near 0, we need the *local graph* near ∞ or near 0.

DEMO 6.25 Let $JILL$ be the function specified by the global input-output rule

$$x \xrightarrow{JILL} JILL(\pm) = (+32.06)(\pm)^{+6}$$

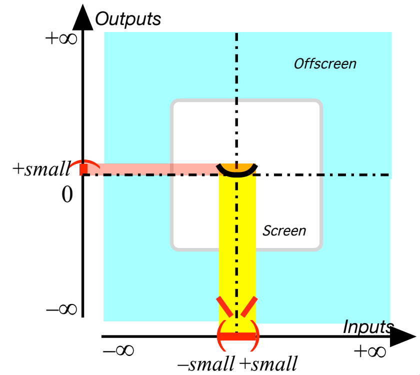
To *get* the *Slope sign* of $JILL$ near 0

We need the *local graph* of $JILL$ near 0.

i. We get the output for $JILL$ near 0

$$\begin{aligned}\pm small \xrightarrow{JILL} & JILL(\pm small)\\ &= (+bounded)(\pm small)^{+even}\\ &= (+bounded)(\pm)^{even}(small)^{+}\\ &= (+bounded)(+)\cdot(small)\\ &= +small\end{aligned}$$

ii. The local graph of $JILL$ near 0 is



iii. Slope sign $JILL$ near $0 = \langle \diagdown, \diagup \rangle$

DEMO 6.26 Let JIM be the function specified by the global input-output rule

$$x \xrightarrow{JIM} JIM(x) = (-72.49)x^{-5}$$

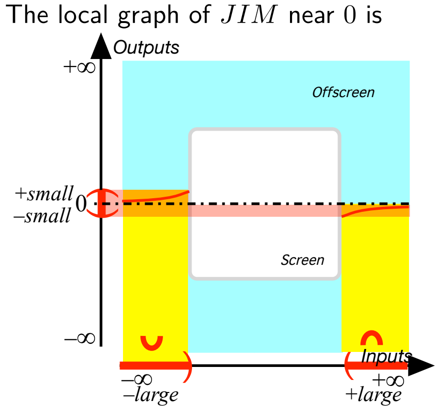
To *get* the *Concavity sign* of JIM near ∞

We need the *local graph* of JIM near ∞.

i. We get the output for JIM near ∞

$$\begin{aligned}\pm large \xrightarrow{JIM} & JIM(\pm large)\\ &= (-bounded)(\pm large)^{-odd}\\ &= \frac{-bounded}{\underbrace{(\pm large)...(\pm large)}_{\text{odd number of copies}}}\\ &= \frac{-bounded}{\pm large}\\ &= -bounded\cdot\pm small\\ &= \mp small\end{aligned}$$

ii. The local graph of JIM near 0 is



iii. Concavity sign JIM near $\infty = \langle \cap, \cup \rangle$

type

Chapter 7

Regular Monomial Functions - Global Analysis

The GLOBAL ANALYSIS of regular monomial functions is very *systematic* because the global input-output rule is very simple.

1 Types of Global Input-Output Rules

1. From the point of view of their *global input-output rule*, there are *eight* **types** of regular monomial functions:

Coefficient Sign	*Exponent* Sign	*Exponent* Parity	Output-specifying code
$+$	$+$	*even*	$(+bounded)\, x^{+\,even}$
		odd	$(+bounded)\, x^{+\,odd}$
	$-$	*even*	$(+bounded)\, x^{-\,even}$
		odd	$(+bounded)\, x^{-\,odd}$
$-$	$+$	*even*	$(-bounded)\, x^{+\,even}$
		odd	$(-bounded)\, x^{+\,odd}$
	$-$	*even*	$(-bounded)\, x^{-\,even}$
		odd	$(-bounded)\, x^{-\,odd}$

2. There are two kinds of regular monomial functions which come up so often that they have special names:

DEFINITION 7.1 Square Functions are monomial functions with *exponent* $+2$, that is functions specified by $x \xrightarrow{SQUARE} SQUARE(x) = ax^{+2}$. (Where Sign a can be either $+$ or $-$.)

EXAMPLE 7.1. The function specified by $x \xrightarrow{SQUARE} SQUARE(x) = -41.87x^{+2}$ is a *square function*.

DEFINITION 7.2 Cube Functions are monomial functions with *exponent* $+3$, that is functions specified by $x \xrightarrow{CUBE} CUBE(x) = ax^{+3}$. (Where Sign a can be either $+$ or $-$.)

EXAMPLE 7.2. The function specified by $x \xrightarrow{CUBE} CUBE(x) = +27.61x^{+3}$ is a *cube function*.

2 Output Sign

Since *Exponent Sign* specifies only whether the *coefficient* is to be multiplied or divided by the copies of the input and since theorem 19.2 on page 365 says that signs are multiplied and divided the same way, *Exponent Sign* cannot have any effect on *Output Sign*.

1. More precisely, since

$$\textit{output} = \textit{coefficient} \text{ multiplied/divided } \textit{power}$$

we have

$$\textit{Output Sign} = \textit{Coefficient Sign} \text{ multiplied/divided } \textit{Power Sign}$$

so that only *Coefficient Sign* and *Input Sign* can possibly have an effect on *Output Sign*. But then:

- If Exponent Parity $=$ *even*, then as a consequence of theorem 19.2 on page 365, *Power Sign* $= +$ both when Input Sign $= +$ and when Input Sign $= -$

and therefore, when Exponent Parity $=$ *even*

- ▷ *Output Sign* $=$ *Coefficient Sign* both when Input Sign $= +$ and when Input Sign $= -$.

- If Exponent Parity $=$ *odd*, then as a consequence of theorem 19.2 on page 365,
  - ▷ *Power Sign* $= +$ when Input Sign $= +$,
  - ▷ *Power Sign*$= -$ when Input Sign $= -$,

and therefore, when Exponent Parity $=$ *odd*

- ▷ *Output Sign* $=$ *Coefficient Sign* when Input Sign $= +$,
- ▷ *Output Sign* $=$ Opposite *Coefficient Sign* when Input Sign $= -$,

EXAMPLE 7.3. Given the function specified by the *global input-output rule*

$$x \xrightarrow{KIP} KIP(x) = (+82.33) \cdot x^{-4}$$

using theorem 19.2 on page 365, we have

$$+ \xrightarrow{KIP} KIP(+) = \frac{+}{\underbrace{+ \cdot \; \dots \; \cdot +}_{even \text{ number of copies of } +}}$$

$$= \frac{+}{+}$$

$$= +$$

EXAMPLE 7.4. Given the function specified by the *global input-output rule*

$$x \xrightarrow{KAP} KAP(x) = (-73.93) \cdot x^{+11}$$

using theorem 19.2 on page 365, we have

$$+ \xrightarrow{KAP} KAP(+) = - \cdot \underbrace{(+) \cdot \; \dots \; \cdot (+)}_{odd \text{ number of copies of } +}$$

$$= - \cdot +$$

$$= -$$

EXAMPLE 7.5. Given the function specified by the *global input-output rule*

$$x \xrightarrow{KAT} KAT(x) = (-25.25) \cdot x^{+7}$$

using theorem 19.2 on page 365, we have

$$-\xrightarrow{KAT} KAT(-) = - \cdot \underbrace{(-)\cdot\ \ldots\ \cdot(-)}_{odd\ \text{number of copies of}\ -}$$
$$= - \cdot -$$
$$= +$$

EXAMPLE 7.6. Given the function specified by the *global input-output rule*

$$x \xrightarrow{KIT} KIT(x) = (+44.06)\cdot x^{-4}$$

using theorem 19.2 on page 365, we have

$$-\xrightarrow{KIT} KIT(-) = \frac{+}{\underbrace{-\cdot\ \ldots\ \cdot -}_{even\ \text{number of copies of}\ -}}$$
$$= \frac{+}{+}$$
$$= +$$

We therefore have the following which summarizes the results of the above investigation.

> **THEOREM 7.1 Output Sign** (For Regular Monomial Functions.)
> - If Input Sign $= +$,
>   Output Sign $=$ Coefficient Sign.
> - If Input Sign $= -$,
>   Output Sign depends on Exponent Parity:
>   - ▷ If Exponent Parity $=$ *even*,
>   Output Sign $=$ Coefficient Sign,
>   - ▷ If Exponent Parity $=$ *odd*,
>   Output Sign $=$*Opposite* Coefficient Sign.

2. Then, for

> **PROCEDURE 7.1 To get the Output Sign *for a regular monomial function***
>
> Use theorem 7.1 on page 180.

horizontal flip

Demo 7.1 Given the function specified by the *global input-output rule*

$$x \xrightarrow{KIP} KIP(x) = (+82.33) \cdot x^{-4}$$

get the Output Sign

Using theorem 7.1 on page 180 we get immediately

$$+ \xrightarrow{KIP} KIP(+) = +$$
$$- \xrightarrow{KIP} KIP(-) = +$$

Demo 7.2 Given the function specified by the *global input-output rule*

$$x \xrightarrow{KAP} KAP(x) = (-73.93) \cdot x^{+11}$$

get the Output Sign

Using theorem 7.1 on page 180 we get immediately

$$+ \xrightarrow{KAP} KAP(+) = -$$
$$- \xrightarrow{KAP} KAP(-) = +$$

Demo 7.3 Given the function specified by the *global input-output rule*

$$x \xrightarrow{KAT} KAT(x) = (-25.25) \cdot x^{+7}$$

get the Output Sign

Using theorem 7.1 on page 180 we get immediately

$$+ \xrightarrow{KAT} KAT(+) = -$$
$$- \xrightarrow{KAT} KAT(-) = +$$

Demo 7.4 Given the function specified by the *global input-output rule*

$$x \xrightarrow{KIT} KIT(x) = (+44.06) \cdot x^{-4}$$

get the Output Sign

Using theorem 7.1 on page 180 we get immediately

$$+ \xrightarrow{KIT} KIT(+) = +$$
$$- \xrightarrow{KIT} KIT(-) = +$$

3. In order to graph monomial functions more efficiently, we need to invest a little bit on a couple of graphic maneuvers:

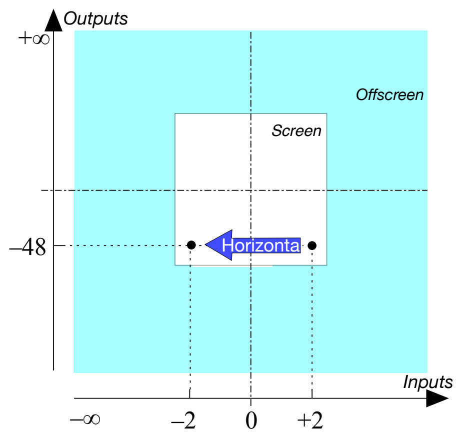
a. If we do a **horizontal flip** on a first *plot point* we get a second *plot point*

vertical flip
diagonal flip

and

- The *input* of the second plot point will be the *opposite* of the input of the first plot point
- The *output* of the second plot point will be the *same* as the output of the first plot point

EXAMPLE 7.7. If we do a *horizontal flip* on a the plot point $(+2, -48)$ we will get a second plot point and:



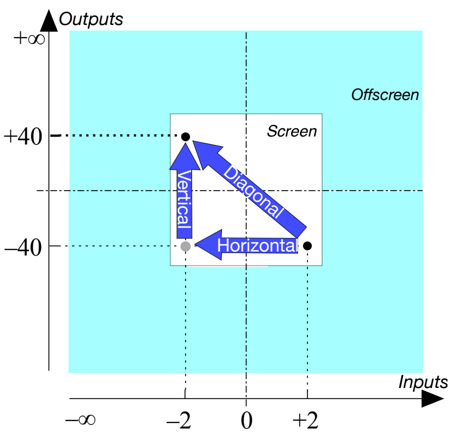
- the *input* of the second plot point will be -2
- the *output* of the second plot point will be -48

b. If we follow the horizontal flip on the *first plot point* by a **vertical flip** on the *second plot point*, we will get a *third plot point* and:

- the *input* of the third plot point will be the *same* as the *input* of the second plot point, that is the *opposite* of the input of the first plot point
- the *output* of the third plot point will be the *opposite* of the *output* of the second plot point, that is the *opposite* of the output of the first plot point

In other words, we can get the third plot point by a **diagonal flip** on the first plot point.

EXAMPLE 7.8. If we do a *horizontal flip* on the plot point $(+2, -48)$ we get a second plot point and if we follow by a vertical flip on the second plot point, we get a third plot point and:



opposite input

- the *input* of the second plot point will be -2
- the *output* of the second plot point will be -40

and then

- the *input* of the third plot point will be -2
- the *output* of the third plot point will be $+40$

In other words, both the *input* and the *output* of the third plot point are *opposite* of the input and output of the first plot point and so to get the third plot point directly from the first plot point we can just use a *diagonal flip* instead of a horizontal flip followed by a vertical flip.

4. So a consequence of theorem 7.1 on page 180 is that once we have the plot point for an input, we can get the plot point for the **opposite input**, that is for the input with the *same* size and *opposite* sign with just one flip:

THEOREM 7.2 Symmetry (For Regular Monomial Functions.) Given the plot point for an input, we get the plot point for the *opposite input* with:
- A *horizontal*-flip if Exponent Parity = *even*,
- A *diagonal*-flip if Exponent Parity = *odd*.

EXAMPLE 7.9. Given the function specified by the global input-ouput rule

$$x \xrightarrow{KAT} KAT(x) = (-3) \cdot x^{+4}$$

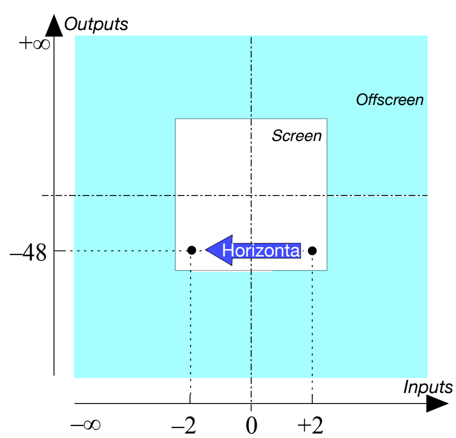
a. For instance

$$+2 \xrightarrow{KAT} KAT(+2) = -3 \bullet +2 \bullet +2 \bullet +2 \bullet +2$$
$$= -48$$

and

$$-2 \xrightarrow{KAT} KAT(-2) = -3 \bullet -2 \bullet -2 \bullet -2 \bullet -2$$
$$= -48$$

b. We see that we can get the plot point for input -2 by a *horizontal* flip of the plot point for input $+2$:



Example 7.10. Given the function specified by the global input-ouput rule

$$x \xrightarrow{KAT} KAT(x) = (+5) \cdot x^{+3}$$

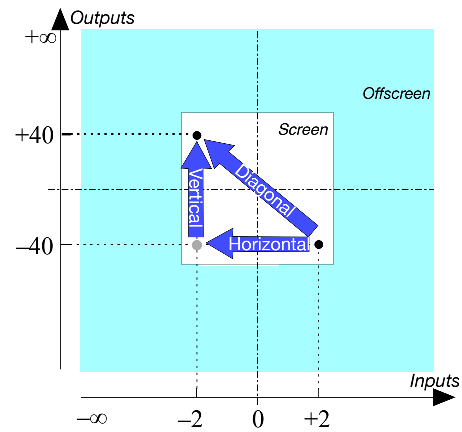
a. For instance

$$+2 \xrightarrow{KAT} KAT(+2) = +5 \bullet +2 \bullet +2 \bullet +2$$
$$= +40$$

and

$$-2 \xrightarrow{KAT} KAT(-2) = +5 \bullet -2 \bullet -2 \bullet -2$$
$$= -40$$

b. We see that we can get the plot point for input -2 by a *diagonal flip* of the plot point for input $+2$:



3 Output Qualitative Size

Language 7.1 Size. When it is clear from the context that we refer to *Qualitative* Size, as in this section, we will just say Size as in, for instance, "Input Size = $small$" instead of "Input Qualitative Size = $small$".

Since *Exponent Sign* specifies if the *coefficient* is to be multiplied or divided by the copies of the *input* and since, depending on *Exponent Sign*, we use either theorem 1.2 or theorem 1.3, *Output Size* will depend on *Exponent Sign*.

1. More precisely, *Output Size* has to depend on both *Input Size* and *Exponent Sign*:

- If Exponent Sign is +, the coefficient is to be *multiplied* by copies of the input then, as a consequence of theorem 1.2 on page 38:
  - ▷ If Input Size = $large$, Output Size = $bounded \times large = large$
  - ▷ If Input Size = $small$, Output Size = $bounded \times small = small$
- If Exponent Sign = −, then the coefficient is to be *divided* by copies of the input so that, as a consequence of theorem 1.3 on page 39:
  - ▷ If Input Size = $large$, Output Size = $bounded \div large = small$
  - ▷ If Input Size = $small$, Output Size = $bounded \div small = large$

EXAMPLE 7.11. Given the function specified by the *global input-output rule*

$$x \xrightarrow{KIP} KIP(x) = (+82.33) \cdot x^{-4}$$

Using theorem 1.3, we have

$$\begin{aligned} small \xrightarrow{KIP} KIP(small) &= \frac{bounded}{small \cdot \ \ldots \ \cdot small} \\ &= \frac{bounded}{small} \\ &= large \end{aligned}$$

EXAMPLE 7.12. Given the function specified by the *global input-output rule*

$$x \xrightarrow{KAP} KAP(x) = (-73.93) \cdot x^{+11}$$

Using theorem 1.2, we have

$$\begin{aligned} large \xrightarrow{KAP} KAP(large) &= bounded \cdot large \cdot \ \ldots \ \cdot large \\ &= bounded \cdot large \\ &= large \end{aligned}$$

EXAMPLE 7.13. Given the function specified by the *global input-output rule*

$$x \xrightarrow{KAT} KAT(x) = (-25.25) \cdot x^{+7}$$

Using theorem 1.2, we have

$$\begin{aligned} small \xrightarrow{KAT} KAT(small) &= bounded \cdot small \cdot \ \ldots \ \cdot small \\ &= bounded \cdot small \\ &= small \end{aligned}$$

EXAMPLE 7.14. Given the function specified by the *global input-output rule*

$$x \xrightarrow{KIT} KIT(x) = (+44.06) \cdot x^{-4}$$

Using theorem 1.3, we have

$$large \xrightarrow{KIT} KIT(large) = \frac{bounded}{large \cdot \ \ldots \ \cdot large}$$
$$= \frac{bounded}{large}$$
$$= small$$

We therefore have the following which summarizes the results of the above investigations

THEOREM 7.3 Output Size (For Regular Monomial Functions)

- If Exponent Sign $= +$,
  Output Size = Input Size.
- If Exponent Sign $= -$,
  Output Size = *Reciprocal* Input Size.

(For "*Reciprocal*" see theorem 15.12 on page 318.)

2. Then, for

PROCEDURE 7.2 To get the Output Size *for a regular monomial function*

Use theorem 7.3 on page 186.

DEMO 7.5 Given the function specified by the *global input-output rule*

$$x \xrightarrow{KIP} KIP(x) = (+82.33) \cdot x^{-4}$$

get the Output Size.

Using theorem 7.3 on page 186 we get

$$large \xrightarrow{KIP} small$$
$$small \xrightarrow{KIP} large$$

DEMO 7.6 Given the function specified by the *global input-output rule*

$$x \xrightarrow{KAP} KAP(x) = (-73.93) \cdot x^{+11}$$

get the Output Size.
Using theorem 7.3 on page 186 we get

$$large \xrightarrow{KAP} large$$
$$small \xrightarrow{KAP} small$$

DEMO 7.7 Given the function specified by the *global input-output rule*

$$x \xrightarrow{KAT} KAT(x) = (-25.25) \cdot x^{+7}$$

get the Output Size.
Using theorem 7.3 on page 186 we get

$$large \xrightarrow{KAT} large$$
$$small \xrightarrow{KAT} small$$

DEMO 7.8 Given the function specified by the *global input-output rule*

$$x \xrightarrow{KIT} KIT(x) = (+44.06) \cdot x^{-4}$$

get the Output Size.
Using theorem 7.3 on page 186 we get

$$large \xrightarrow{KIT} small$$
$$small \xrightarrow{KIT} large$$

4 Reciprocity

1. Another way to look at theorem 7.3 on 186 is to realize that, for a monomial function,

- If Output Size = Input Size, this can only be because Exponent Sign = $+$,
- If Output Size = *Reciprocal* Input Size, this can only be because Exponent Sign = $-$.

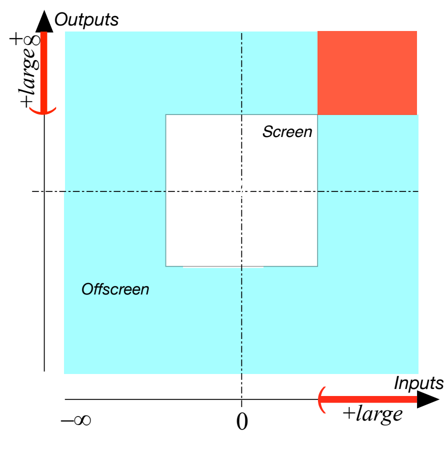
Which gives us the following which we will use to graph regular monomial functions efficiently:

Theorem 7.4 Reciprocity (For Regular Monomial Functions.)

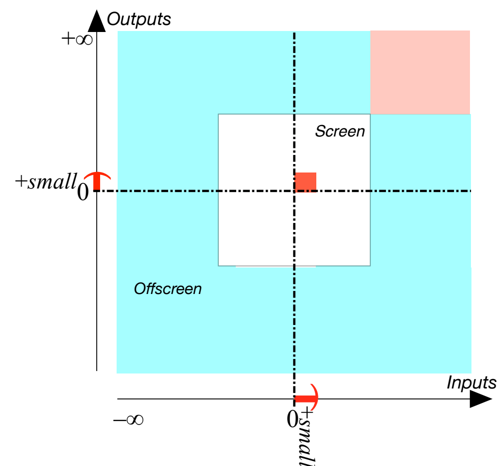
- If *large* $\to$ *large*, then *small* $\to$ *small* (And vice versa.)
- If *large* $\to$ *small*, then *small* $\to$ *large* (And vice versa.)

Example 7.15.

After we have found, for instance,

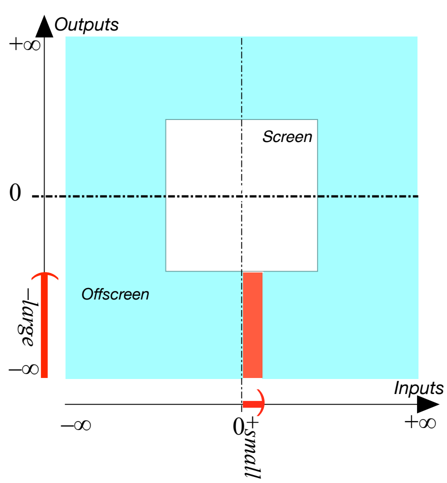


We get from theorem 7.4

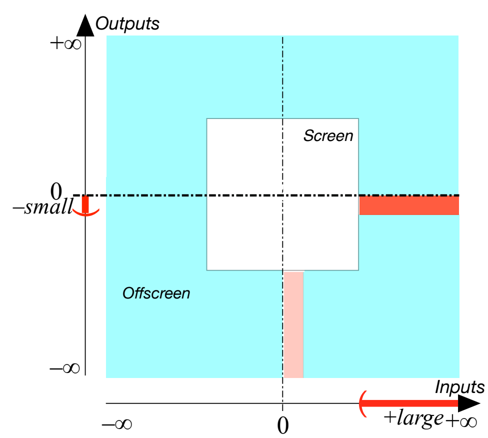


Example 7.16.

After we have found, for instance,

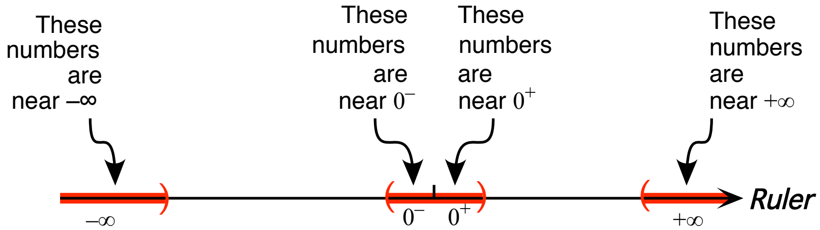


We get from theorem 7.4



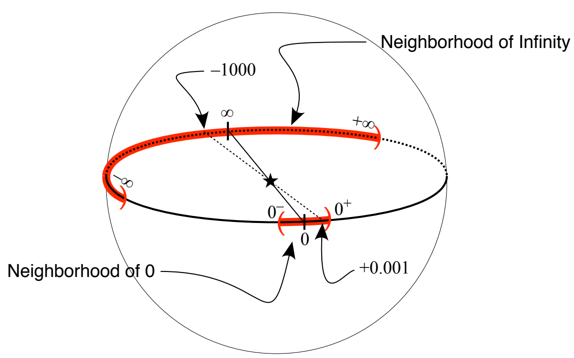
2. The relationship between ∞ and 0 is not only important but also fascinating.

a. Even though, as an input, 0 is usually not particularly important, there is an intriguing "symmetry" between ∞ and 0 namely:



More precisley, *small numbers* are some sort of inverted image of *large numbers* since the *reciprocal* of a *large number* is a *small number* and vice versa.

EXAMPLE 7.17.
The opposite of the reciprocal of -0.001 is $+1\,000$. In a Magellan view, we have

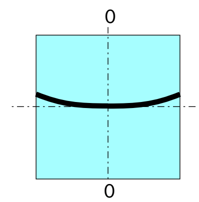


b. Here is yet another way to look at reciprocity. We start with the graph of a monomial function and we "turn" it so as to see it while facing ∞ and we then compare it with the graph near 0 of the reciprocal function.

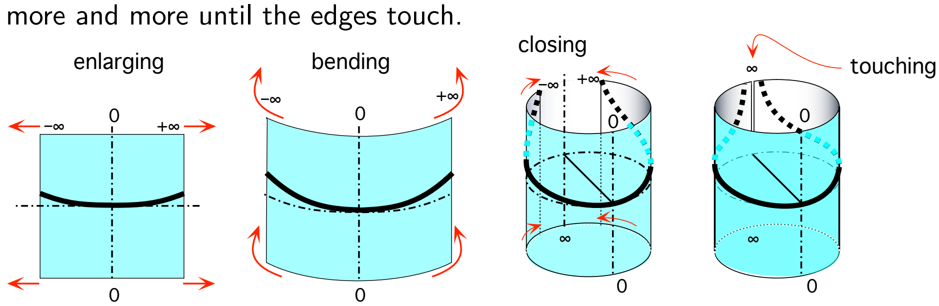
EXAMPLE 7.18.
Let the monomial function specified by the global input-output rule

$$x \xrightarrow{RAIN} RAIN(x) = (+1)x^{+4}$$

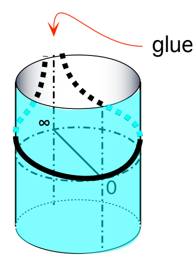
the local graph near 0 of $RAIN$ is:



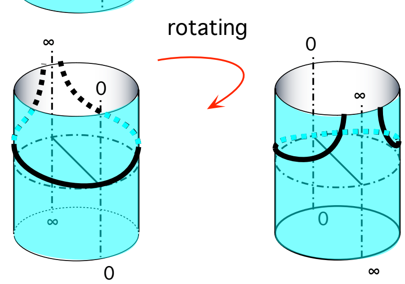
We enlarge the *extent* of the input ruler more and more while shrinking the *scale* by the edges more and more and, as we do so, we bend the screen backward more and more until the edges touch.



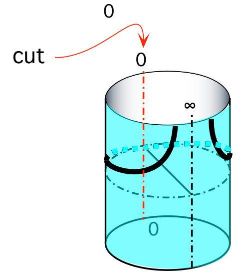
We then glue shut the edges of the screen at ∞ to get a cylinder.



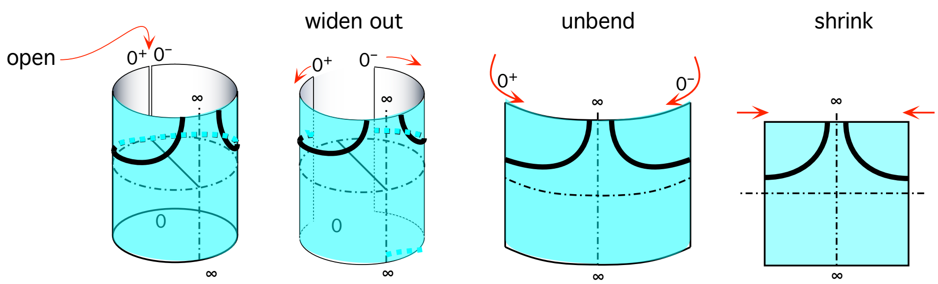
Then we turn the cylinder half a turn so that ∞ gets to be in front of us:



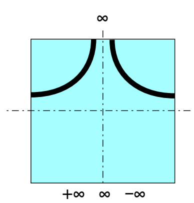
Now we cut open the cylinder along the input level line for 0



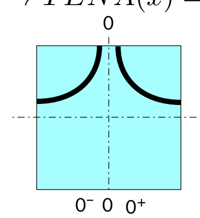
Finally we widen the cut and unbend the screen forward more and more until it becomes flat.



The local graph near ∞ that we got for $RAIN$ is:



It is the same as the local graph near 0 of the reciprocal function specified by the global input-output rule
$x \xrightarrow{TENA} TENA(x) = (+1)x^{-4}$

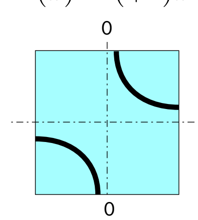


(Keep in mind that the left side of ∞ is the positive side of ∞ and the right side of ∞ is the negative side of ∞. So the graphs on the positive sides are the same and the local graphs on the negative sides are also the same.)

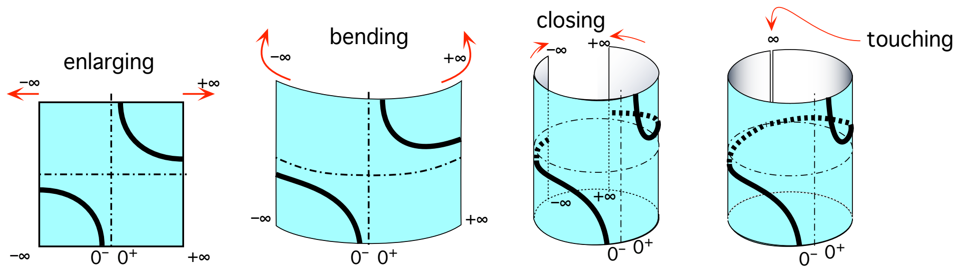
Example 7.19.
Given the monomial function specified by the global input-output rule

$$x \xrightarrow{MIKE} MIKE(x) = (+1)x^{-3}$$

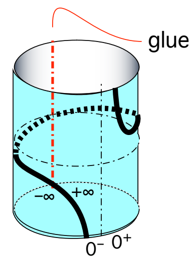
the local graph near 0 of $MIKE$ is:



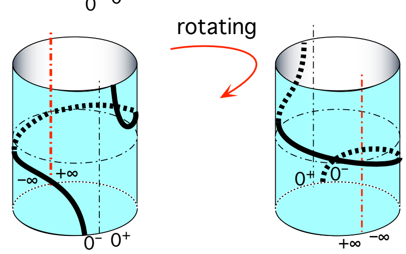
We enlarge the *extent* of the input ruler more and more while shrinking the *scale* by the edges more and more and, as we do so, we bend the screen backward more and more closing down the gap until the edges touch:



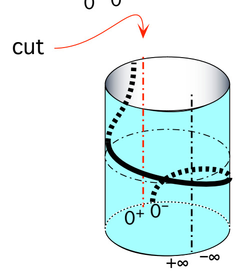
We then glue shut the edges of the screen at ∞ to get a cylinder.



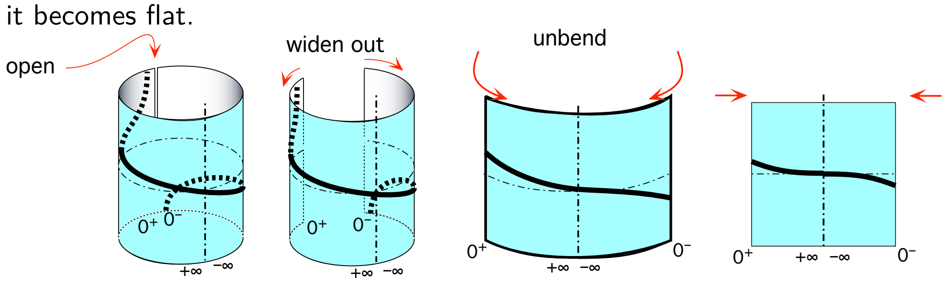
Then we rotate the cylinder half a turn so that ∞ gets to be in front of us:



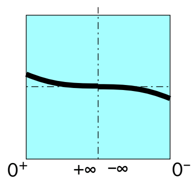
Now we cut open the cylinder along the input level line for 0



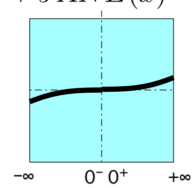
Finally we widen the cut and unbend the screen forward more and more until it becomes flat.



The local graph near ∞ that we just got for $MIKE$ is:



It is the same as the local graph near 0 of the reciprocal function specified by the global input-output rule $x \xrightarrow{JANE} JANE(x) = (+1)x^{+3}$



(Keep in mind that the left side of ∞ is the positive side of ∞ and the right side of ∞ is the negative side of ∞. So the graphs on the positive sides *are* the same and the local graphs on the negative sides also *are* the same.)

5 Global Graphing

We can of course get the global graph the way we will get the global graph of all the other functions in this text, that is as described in **??**, but, in the case of *regular monomial functions*, we will be taking advantage of the following THEOREMS which we must become *completely familiar* with—but which we certainly must not *memorize*:

- The first part of theorem 7.1 on page 180 namely:

> **THEOREM 7.5 Output Sign for *positive* inputs.** (For Regular Monomial Functions.)
>
> Output Sign for *positive* inputs = Coefficient Sign.

- Theorem 7.3 on page 186
- Theorem 7.4 on page 188
- Theorem 7.2 on page 183

Then, after a little bit of practice, we will be able to get the global graph *very rapidly*:

PROCEDURE 7.3 Graph a *regular* monomial function:

a. Locate the *graph place* for inputs near $+\infty$ as follows:

i. Determine if the graph place for inputs near $+\infty$ is *above* or

below the 0-output level line.
(Use theorem 7.5 on page 180)

ii. Determine if the graph place for inputs near $+\infty$ is near the 0-*output* level line or near the ∞-*output* level line,
(Use theorem 7.3 on page 186)

b. Locate the *graph place* for inputs near 0^+.
(Use theorem 7.4 on page 188).

c. Locate the *graph places* for inputs near $-\infty$ and inputs near 0^-.
(Use theorem 7.2 on page 183)

d. Draw the *global graph* through the *graph places*.

Demo 7.9 Get the *global graph* of the function specified by the *global input-output rule*

$$x \xrightarrow{KIR} KIR(x) = (+52.92) \cdot x^{-13}$$

1. We locate the graph place for inputs near $+\infty$:

i. Since Coefficient Sign $= +$,

$$+ \xrightarrow{KIR} +$$

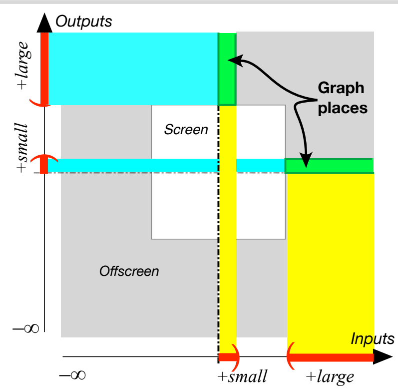
(Using theorem 7.5 on page 193.)

ii. Since Exponent Sign $= -$,

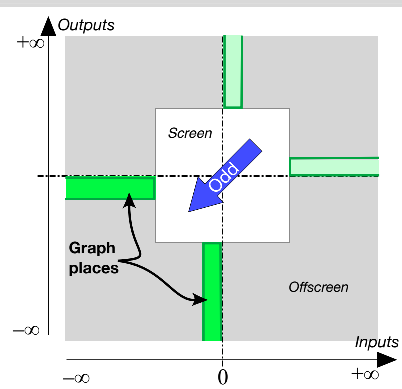
$$large \xrightarrow{KIR} small$$

(Using theorem 7.3 on page 186.)

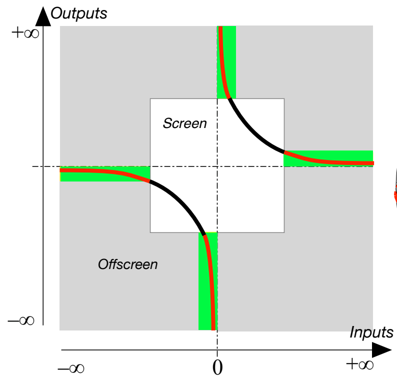
2. We locate the graph place for inputs near 0^+.
(Using theorem 7.4 on page 188.)

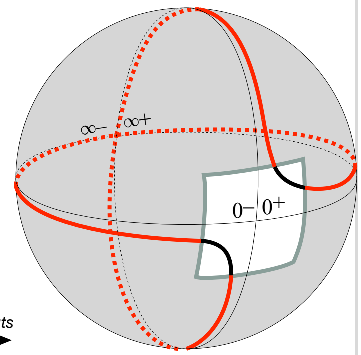


3. We locate the graph places for inputs near $-\infty$ and near 0^-.
(Using theorem 7.2 on page 183.)



4. We draw the *global graph* through the graph places. And, to the right is a Magellan view of the global graph.





Demo 7.10 Get the *global graph* of the function specified by the *global input-output rule*

$$x \xrightarrow{KIM} KIM(x) = (-40)x^{+6}$$

1. We locate the graph place for inputs near $+\infty$:

i. Since Coefficient Sign $= -$,

$$+ \xrightarrow{KIM} +$$

(Using theorem 7.5 on page 193.)

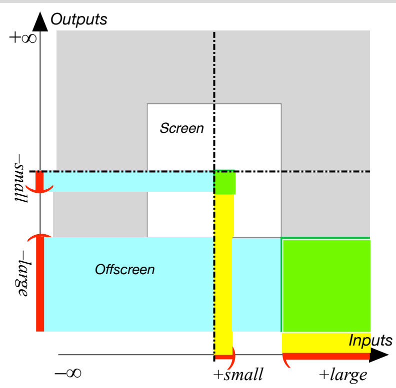
ii. Since Exponent Sign $= +$,

$$large \xrightarrow{KIM} large$$

(Using theorem 7.3 on page 186.)

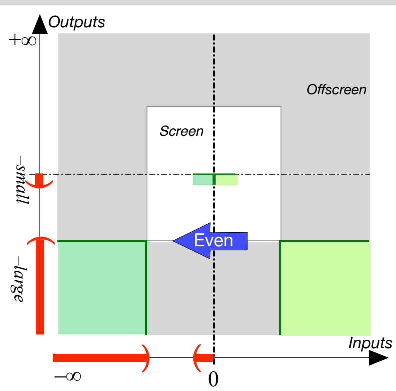
2. We locate the graph place for inputs near 0^+.

(Using theorem 7.4 on page 188.)

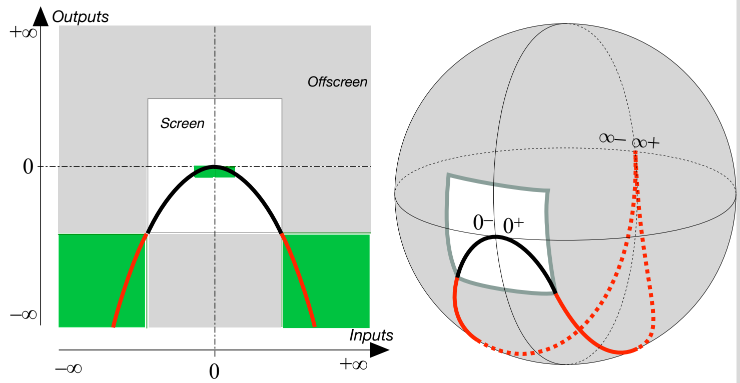


3. We locate the graph places for inputs near $-\infty$ and near 0^-.

(Using theorem 7.2 on page 183.)



4. We draw the *global graph* through the graph places. And, to the right is a Magellan view of the global graph.



DEMO 7.11 Get the *global graph* of the function specified by the *global input-output rule*

$$x \xrightarrow{KIN} KIN(x) = (-40.87)x^{+5}$$

1. We locate the graph place for inputs near $+\infty$:

i. Since Coefficient Sign $= -$,

$$+ \xrightarrow{KIN} -$$

(Using theorem 7.5 on page 193.)

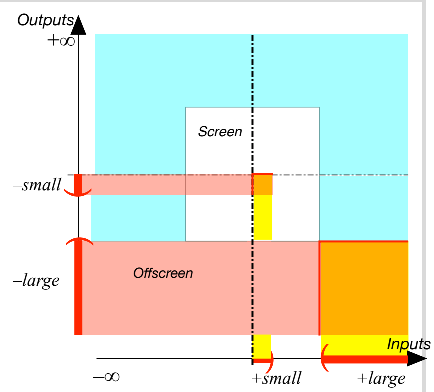
ii. Since Exponent Sign $= +$,

$$large \xrightarrow{KIN} large$$

(Using theorem 7.3 on page 186.)

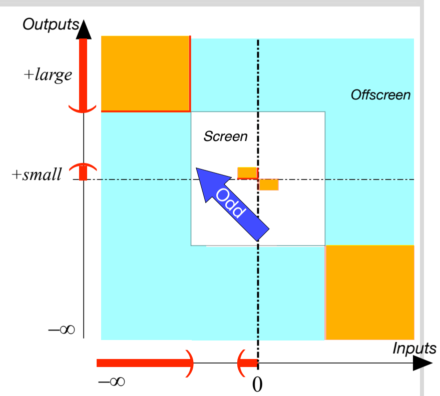
2. We locate the graph place for inputs near 0^+.

(Using theorem 7.4 on page 188.)

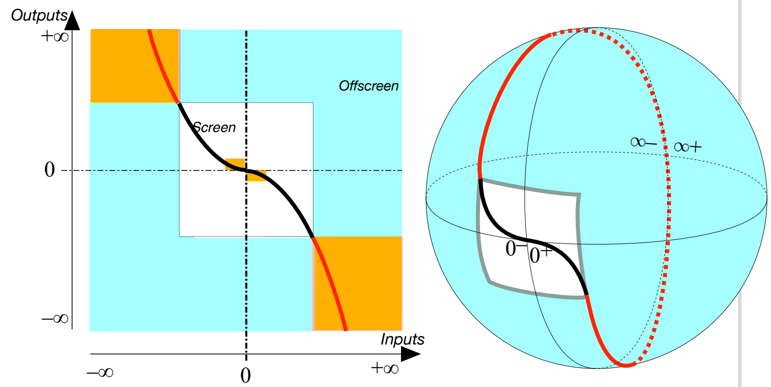


3. We locate the graph places for inputs near $-\infty$ and near 0^-.

(Using theorem 7.2 on page 183.)



4. We draw the *global graph* through the graph places. And, to the right is a Magellan view of the global graph.



DEMO 7.12 Get the *global graph* of the function specified by the *global input-output rule*

$$x \xrightarrow{KIB} KIB(x) = (+77.03) \cdot x^{-8}$$

1. We locate the graph place for inputs near $+\infty$:

i. Since Coefficient Sign $= +$,

$$+ \xrightarrow{KIB} +$$

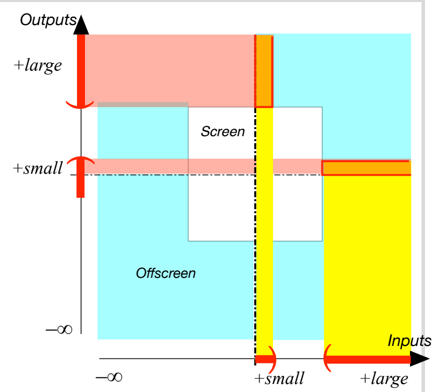
(Using theorem 7.5 on page 193.)

ii. Since Exponent Sign $= -$,

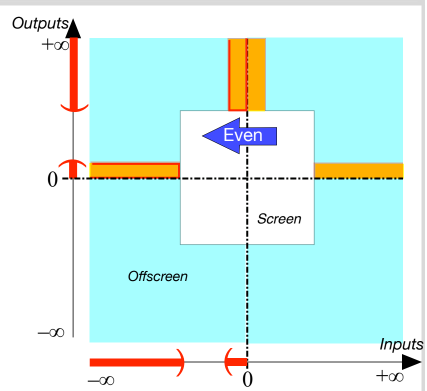
$$large \xrightarrow{KIB} small$$

(Using theorem 7.3 on page 186.)

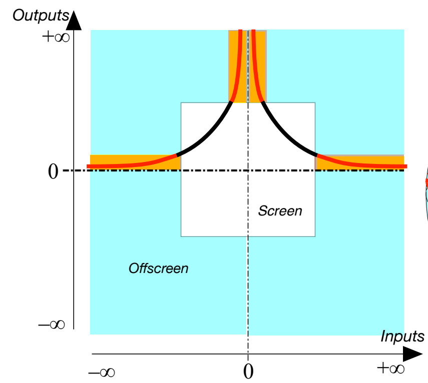
2. We locate the graph place for inputs near 0^+.
(Using theorem 7.4 on page 188.)

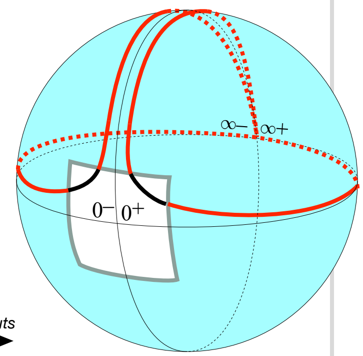


3. We locate the graph places for inputs near $-\infty$ and near 0^-.
(Using theorem 7.2 on page 183.)



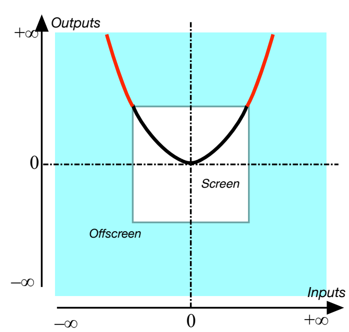
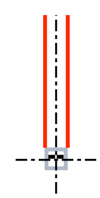
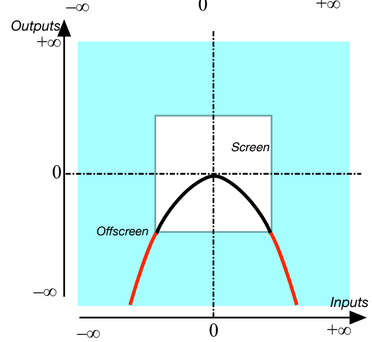
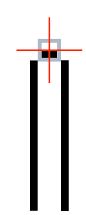
4. We draw the *global graph* through the graph places. And, to the right is a Magellan view of the global graph.



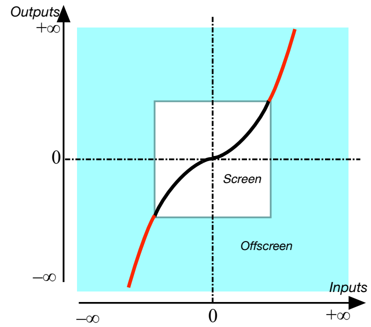
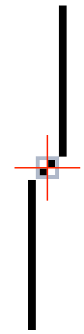
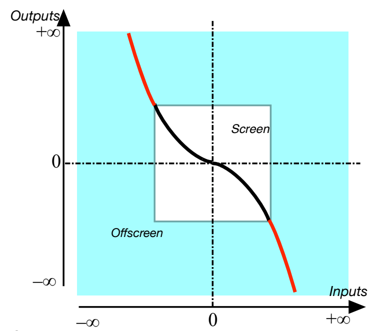
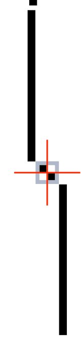
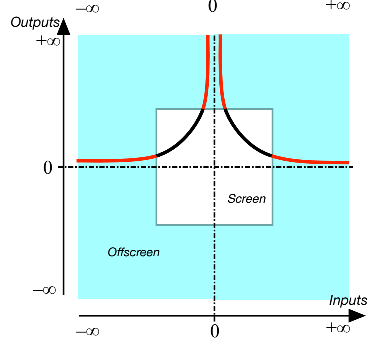
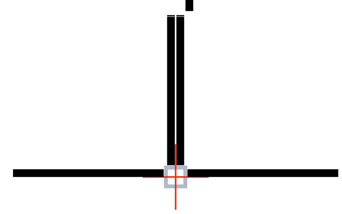
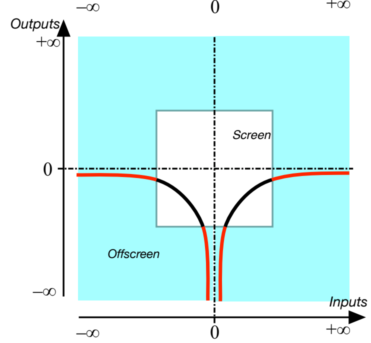
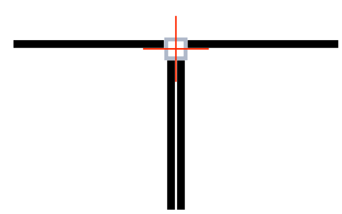


6 Types of Global Graphs

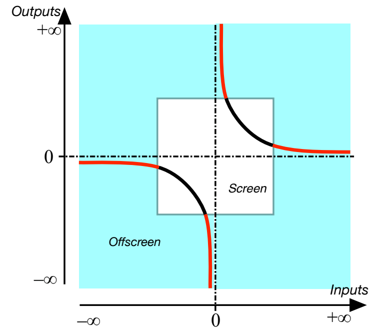
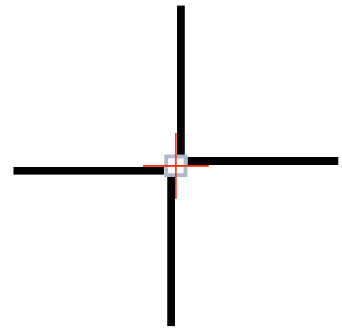
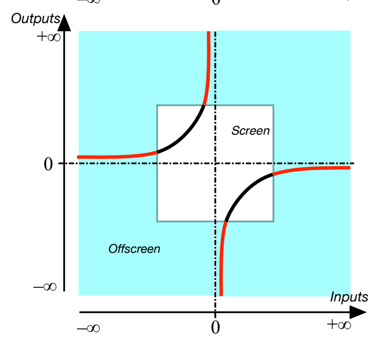
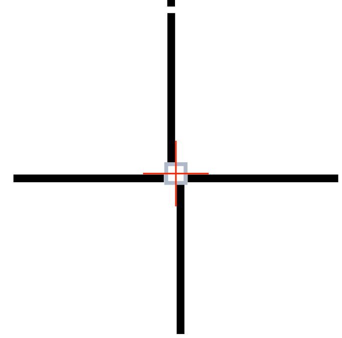
Each type of *global input-output rule* corresponds to a type of *global graph*. The global graphs are shown both from "close-up" to see the bounded graph and from "faraway" to see how the graphs flatten out.

Input-output rule	From "close-up"	From "faraway"
$x \xrightarrow{PEP} PEP(x) = (+1)x^{+even}$		
$x \xrightarrow{PEN} PEN(x) = (-1)x^{+even}$		

Continued on next page

Input-output rule	From "close-up"	From "faraway"
$x \xrightarrow{POP} POP(x) = (+1)x^{+odd}$		
$x \xrightarrow{PON} PON(x) = (-1)x^{+odd}$		
$x \xrightarrow{NEP} NEP(x) = +x^{-even}$		
$x \xrightarrow{NEN} NEN(x) = -x^{-even}$		

Continued on next page

Input-output rule	From "close-up"	From "faraway"
$x \xrightarrow{KIR} KIR(x) = +x^{-odd}$		
$x \xrightarrow{NON} NON(x) = -x^{-odd}$		

approximate
constant coefficient

Chapter 8

Exceptional Monomial Functions

We now investigate the *exceptional monomial functions*, that is the monomial functions with *exponent* $\boxed{0}$ and the monomial functions with *exponent* $\boxed{+1}$. Even though they are ... exceptionally simple, they are ... exceptionally important, the monomial functions with *exponent* $\boxed{0}$ because they are used to **approximate** other functions in INTEGRAL CALCULUS and the monomial functions with *exponent* $\boxed{+1}$ because they are at the basis of APPLIED MATHEMATICS.

FUNCTIONS

DEFINITION 8.1 Constant Functions are monomial functions with *exponent* $\boxed{0}$, that is functions specified by $x \xrightarrow{CONSTANT} CONSTANT(x) = ax^0$. (Where a, called the **constant coefficient**, is the *bounded* number that specifies the function $CONSTANT$.)

EXAMPLE 8.1. The *constant* function $FLAP$ specified by the constant

abuse of language
$UNIT_+$
$UNIT_-$

coefficient $+5\,273.1$ is the function specified by the global input-output rule

$$x \xrightarrow{FLAP} FLAP(x) = \underbrace{+5\,273.1}_{\text{constant coefficient}}\, x^0$$
$$= \underbrace{+5\,273.1}_{\text{constant coefficient}}$$

LANGUAGE 8.1 The name *constant functions* is an **abuse of language** because it is not the function *itself* which is constant but its *output* which is constant in the sense that, since the coefficient is neither multiplied nor divided by any copy of x and thus to be left alone, the output remains *constantly* equal to the *coefficient* no matter what the input is.

NOTE 8.1 $x^{\boxed{0}}$ must absolutely *not* be read "x multiplied by 0" because that would give the output 0 no matter what. (This is a very common error among beginners.)

Contrary to what we did with *regular monomial functions* we will *not* normalize constant functions. In fact, the constant functions with constant coefficients $+1$ and -1 have special names:

- The constant function with coefficient $+1$ is usually called $\boldsymbol{UNIT_+}$. In other words, $UNIT_+$ is the function specified by the *global input-output rule*
$$x \xrightarrow{UNIT_+} UNIT_+(x) = +1$$
- The constant function with coefficient -1 is usually called $\boldsymbol{UNIT_-}$. In other words, $UNIT_-$ is the function specified by the *global input-output rule*
$$x \xrightarrow{UNIT_-} UNIT_-(x) = -1$$

1 Outputs Of Constant Functions

1. In order to get the output *at* a given bounded input x_0 of a monomial function with exponent $\boxed{0}$, we still use **??** on **??** which, in the case of *constant* functions, boils down to nothing.

PROCEDURE 8.1 To get the output at x_0 of the *constant* function specified by the global input-output rule $x \xrightarrow{CONSTANT} CONSTANT(x) = a$

i. *Declare* that x is to be replaced by x_0

$$x\,\Big|_{x\leftarrow x_0} \xrightarrow{CONSTANT} CONSTANT(x)\,\Big|_{x\leftarrow x_0} = a\,\Big|_{x\leftarrow x_0}$$

which however, since there is nothing to replace with x_0, gives:

$$x_0 \xrightarrow{CONSTANT} CONSTANT(x_0) = \underbrace{a}_{\text{output-specifying code}}$$

ii. There is nothing to *execute* and the output *number* is:

$$= a$$

which gives the input-output pair

$$(x_0, a)$$

TEMO 8.1 To get the output at -3 of the function specified by the global input-output rule

$$x \xrightarrow{FLAP} FLAP(x) = \underbrace{+5\,273.1}_{\text{output specifying code}}$$

This is short for

$$= \underbrace{+5\,273.1x^0}_{\text{output specifying code}}$$

and since the exponent is 0 so that we do *not* multiply or divide the coefficient by any copy of the input there is no point *declaring* that the input is -3 and nothing to *execute* and the output of the function $FLAP$ at -3 is just the coefficient:

$$= +5\,273.1$$

In other words, ... just as stated by the output-specifying code to begin with!

2. Since the output of a constant function is the coefficient no matter what the input, the size of the output does not matter and the outputs, both for inputs *near* ∞ and for inputs near 0, are again going to be the *coefficient*.

2 Graphs Of Constant Functions

Constant functions are the first of the only three kinds of functions for which we can get the global graph *directly* because the global graph is a **straight**

straight line

line.

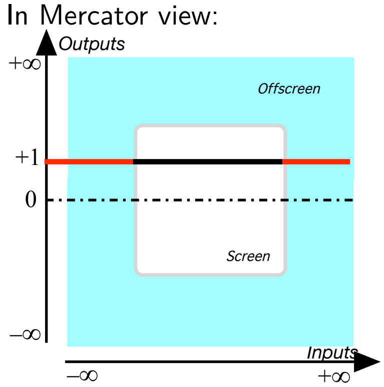
1. Since the output of a constant function is equal to the constant coefficient no matter what the input is, the quantitative global graph will be the output level line of the constant coefficient.

PROCEDURE 8.2 Graph the function specified by $x \xrightarrow{CONSTANT} CONSTANT(x) = a$

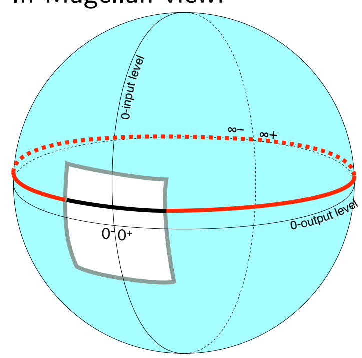
i. Mark the constant coefficient a on the output ruler
ii. Draw the output level line through the tickmark

EXAMPLE 8.2. Here is the global graph of the function $UNIT_+$:

In Mercator view:

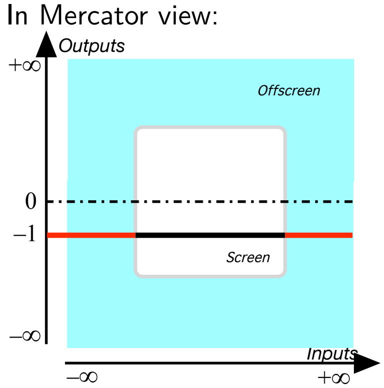


In Magellan view:

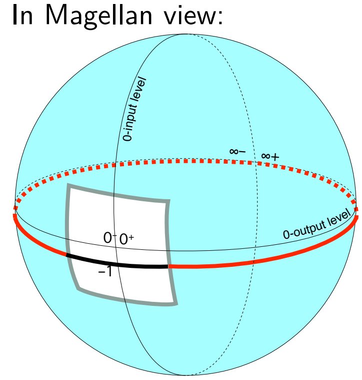


EXAMPLE 8.3. Here is the global graph of the function $UNIT_-$:

In Mercator view:



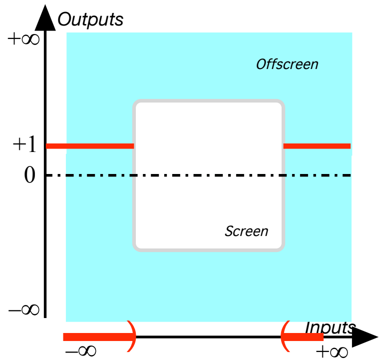
In Magellan view:



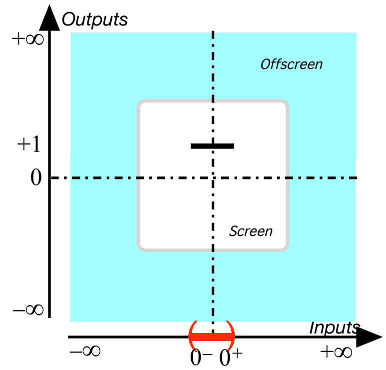
2. Since the global graph is so easy to get, we get the local graphs from the global using**??** on **??**. In fact, we will usually need only the local graph near ∞ and the local graph near 0 .

global height

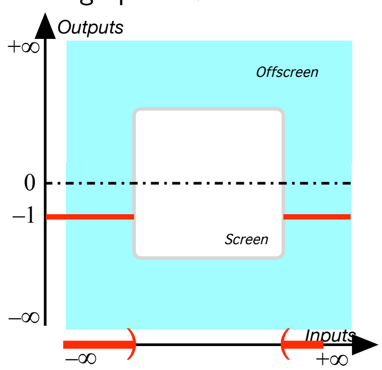
EXAMPLE 8.4.
Local graph of $UNIT_+$ near ∞:



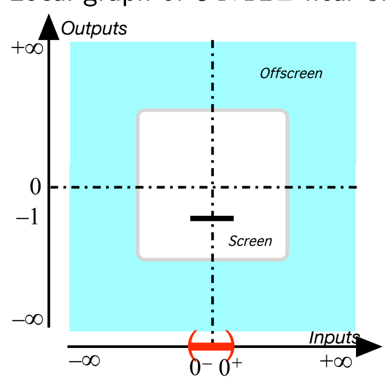
Local graph of $UNIT_+$ near 0:



EXAMPLE 8.5.
Local graph of $UNIT_-$ near ∞:



Local graph of $UNIT_-$ near 0:



3 Features Of Constant Functions

What makes constant functions *exceptional* among monomial functions is that they lack both *local slope* and *local concavity* and have only *local height*.

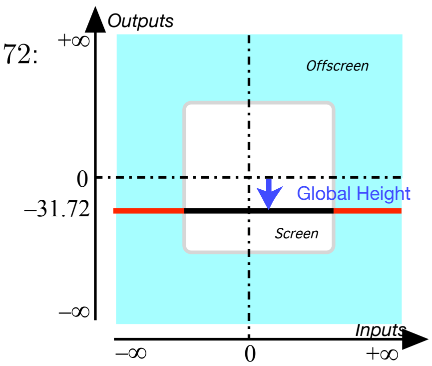
But then, since for a constant function the *local height* is the same everywhere, we can talk of the **global height** of a *constant function*.

EXAMPLE 8.6. Let f be the function specified by the *global input-output* linear coefficient

rule

$$x \xrightarrow{f} f(x) = (-31.72)x^0$$
$$= -31.72$$

the global height of f is -31.72:



LINEAR FUNCTIONS

DEFINITION 8.2 **Linear Functions** are monomial functions with *exponent* $+1$, that is functions specified by $x \xrightarrow{LINEAR} LINEAR(x) = ax^{+1}$. (Where a, called the **linear coefficient**, is the *bounded* number that specifies the function $LINEAR$.)

EXAMPLE 8.7. The *constant* function $FLOP$ specified by the constant coefficient $+5\,273.1$ is the function specified by the global input-output rule

$$x \xrightarrow{FLOP} FLOP(x) = \underbrace{+5\,273.1}_{\text{linear coefficient}} x^{+1}$$
$$= \underbrace{+5\,273.1}_{\text{linear coefficient}} x$$

LANGUAGE 8.2 The reason monomial functions with *exponent* $+1$ are called *linear* functions is that they are (the simplest instance of) a kind of functions with an extremely desirable but extremely rare feature, namely *linearity*. (See https://en.wikipedia.org/wiki/Linearity.) However, one should be careful because the name linear function is also used in PRECALCULUS textbooks for a different kind of functions, which we will investigate in chapter 9 and chapter 10 under the name of *affine functions*.

Contrary to what we did with *regular monomial functions*—and just like what we did with constant functions, we will *not* normalize linear functions. In fact, the linear functions with linear coefficients $+1$ and -1 have special names:

$IDENTITY$
$OPPOSITE$
opposite of x

- The linear function with coefficient $+1$ is usually called $\boldsymbol{IDENTITY}$ because the output is identical with the input. In other words, $IDENTITY$ is the function specified by the *global input-output rule*
$$x \xrightarrow{IDENTITY} IDENTITY(x) = x$$
- The linear function with coefficient -1 is usually called $\boldsymbol{OPPOSITE}$. In other words, $OPPOSITE$ is the function specified by the *global input-output rule*
$$x \xrightarrow{OPPOSITE} OPPOSITE(x) = -x \quad \text{(Where } -x \text{ is read } \textbf{opposite of } x\text{)}$$

4 Output Of Linear Functions *at* x_0

In order to get the output *at* a given bounded input x_0 of a linear function, we proceed exactly as with regular monomial functions, that is we still use **??** on **??** but, in the case of *linear* functions, the *execution* boils down to just one multiplication.

TEMO 8.2 To get the output at -3 of the function specified by the global input-output rule

$$x \xrightarrow{BINK} BINK(x) = \underbrace{(-26.18)x^{+1}}_{\text{output-specifying code}}$$

i. We *declare* that x is to be replaced by -3

$$x\Big|_{x\leftarrow -3} \xrightarrow{BINK} BINK(x)\Big|_{x\leftarrow -3} = (-26.18)x^{+1}\Big|_{x\leftarrow -3}$$

which gives:

$$-3 \xrightarrow{BINK} BINK(-3) = \underbrace{(-26.18)\cdot(-3)^{+1}}_{\text{output specifying code}}$$

ii. We *execute* the output-specifying code into an output *number*:

$$= -26.18 \cdot -3$$

and, using theorem 19.2 on page 365, we get:

$$= +78.54$$

which gives the *input-output pair*

$$(-3, +78.54)$$

5 Outputs Of Linear Functions *near* ∞ and 0

In order to get the output *near* ∞ or near 0 of a linear function, we proceed exactly as in the case of regular monomial functions but, in the case of *linear* functions, the *execution* boils down to just one multiplication.

TEMO 8.3 To get the output near ∞ of the function specified by the global input-output rule

$$x \xrightarrow{BINK} BINK(x) = \underbrace{(-26.18)x^{+1}}_{\text{output-specifying code}}$$

i. We *declare* that x is to be replaced by $\pm large$

$$x\Big|_{x \leftarrow \pm large} \xrightarrow{BINK} BINK(x)\Big|_{x \leftarrow \pm large} = (-26.18)x^{+1}\Big|_{x \leftarrow \pm large}$$

which gives:

$$\pm large \xrightarrow{BINK} BINK(-3) = \underbrace{(-26.18)\cdot(\pm large)^{+1}}_{\text{output specifying code}}$$

ii. We *execute* the output-specifying code

$$= -26.18 \cdot \pm large$$

Since 26.18 is *bounded*, theorem 1.2 on page 38 gives $26.18 \cdot large = large$ and, using theorem 19.2 on page 365, we get:

$$= \mp large$$

which gives the *input-output pair*

$$(\pm large, \mp large)$$

TEMO 8.4 To get the output near 0 of the function specified by the global input-output rule

$$x \xrightarrow{JINK} JINK(x) = \underbrace{(+45.57)x}_{\text{output-specifying code}}$$

i. We *declare* that x is to be replaced by $\pm small$

$$x\Big|_{x \leftarrow \pm small} \xrightarrow{JINK} JINK(x)\Big|_{x \leftarrow \pm small} = (+45.57)x^{+1}\Big|_{x \leftarrow \pm small}$$

which gives:

$$\pm small \xrightarrow{JINK} JINK(\pm small) = \underbrace{(+45.57)\cdot(\pm small)^{+1}}_{\text{output specifying code}}$$

ii. We *execute* the output-specifying code

$$= +45.57 \cdot \pm small$$

Since 45.57 is *bounded*, theorem 1.2 on page 38 gives $45.57 \cdot \textit{small} = \textit{small}$ and, using theorem 19.2 on page 365, we get:

$$= \pm \textit{small}$$

which gives the *input-output pair*

$$(\pm \textit{small}, \pm \textit{small})$$

6 Graphs Of Linear Functions

After the *constant functions*, the *linear functions* are the second of only three kinds of functions for which we can get the global graph *directly* because the global graph is a *straight line*.

With *linear functions*, though, it is not as easy to make the case that the global graph is a *straight line* as with *constant functions* because making the case requires having a geometric definition of what a straight line is. So, here we will take for granted that the global graph of a linear function is a *straight line*.

1. Given a linear function specified by a *global input-output rule*, the key to finding the *quantitative global graph* is another theorem from GEOMETRY, namely that a *straight line* is specified once we know *two* of its points. (Which, in the real world, corresponds to the fact that all we need to draw a straight line through two points is a straightedge.) As a consequence, the quantitative global graph of a *linear function* will be specified by *two* input-output pairs.

There is no restriction as to what bounded inputs to use but given the linear function specified by the global input-output rule

$$x \xrightarrow{\quad f \quad} f(x) = a \cdot x$$

there are two bounded inputs that make it very easy namely 0 and $+1$ because:

- Inputting 0 gives:

$$x \Big|_{x \leftarrow 0} \xrightarrow{\quad f \quad} f(x) \Big|_{x \leftarrow 0} = a \cdot x \Big|_{x \leftarrow 0}$$

$$0 \xrightarrow{\quad f \quad} f(0) = a \cdot 0$$

  and, because any number multiplied by 0 gives 0

$$= 0$$

  So, $(0, 0)$ is an input-output pair.

- Inputting $+1$ gives:

$$x\Big|_{x\leftarrow+1} \xrightarrow{f} f(x)\Big|_{x\leftarrow+1} = a\cdot x\Big|_{x\leftarrow+1}$$
$$+1 \xrightarrow{f} f(+1) = a\cdot(+1)$$

and, because any number multiplied by $+1$ gives that number

$$= a$$

So, $(+1, a)$ is an input-output pair.

2. Given a function specified by a global input-output rule, we will use:

PROCEDURE 8.3 Graph the function specified by $x \xrightarrow{LINEAR} LINEAR(x) = a\cdot x$

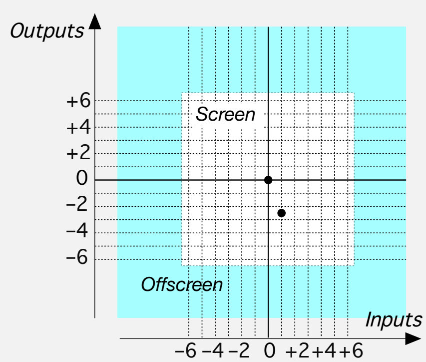
i. Plot the input-output pairs for two inputs, for instance 0 and 1
ii. Draw a straight line through the two plot points

EXAMPLE 8.8. Let f be the function specified by the *global input-output rule*

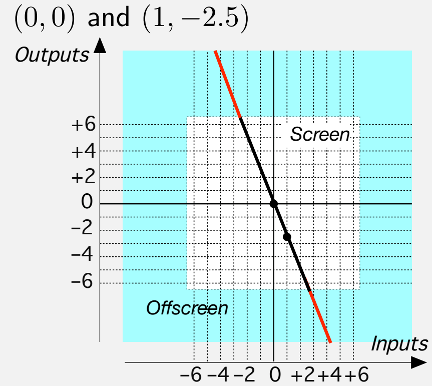
$$x \xrightarrow{f} f(x) = -2.5x$$

in order to get the quantitative global graph,

i. We plot the input-output pairs $(0,0)$ and $(1,-2.5)$

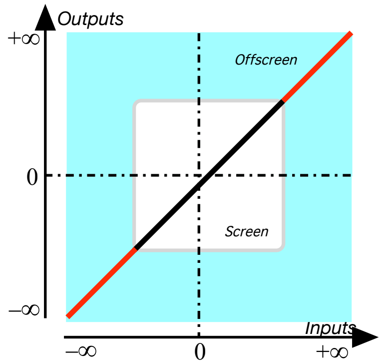


ii. We draw a straight line through the input-output pairs $(0,0)$ and $(1,-2.5)$

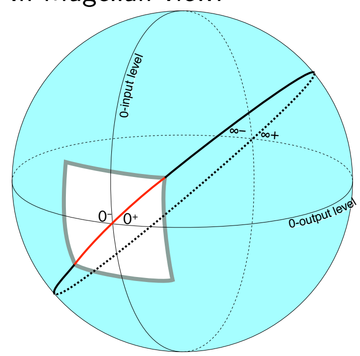


EXAMPLE 8.9. Here is the global graph of the function $IDENTITY$:

In Mercator view:

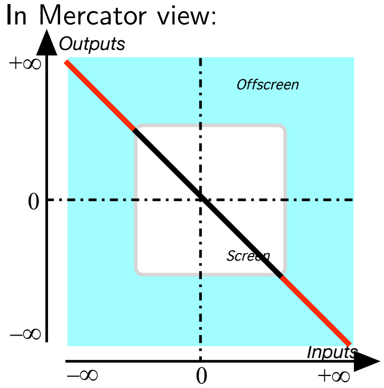


In Magellan view:

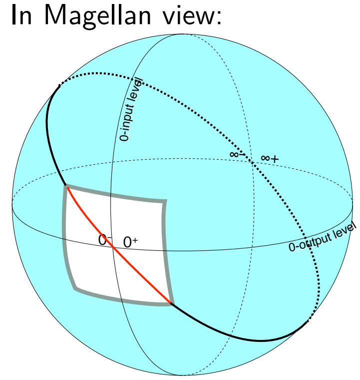


EXAMPLE 8.10. Here is the global graph of the function $OPPOSITE$:

In Mercator view:



In Magellan view:

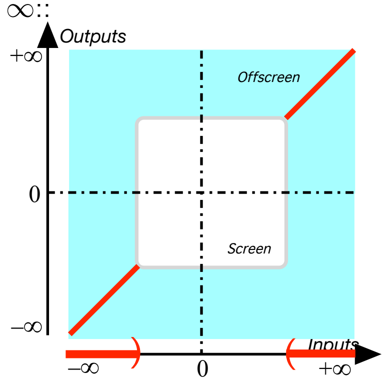


3. Since the global graph is so easy to get, we get the local graphs near 0 and near ∞ using **??** on **??**. In the rest of this text, though, given a linear function, we will usually need only the local graph near ∞ and the local graph near 0.

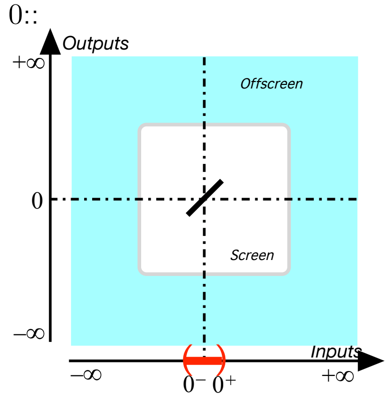
EXAMPLE 8.11.

global slope
run
rise

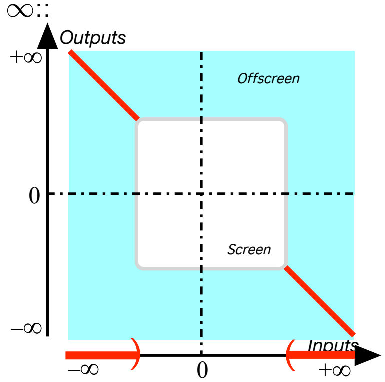
Local graph of $IDENTITY_{+}$ near ∞::



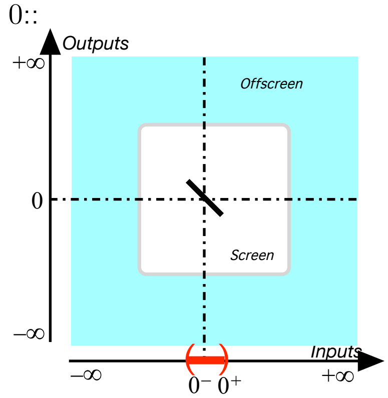
Local graph of $IDENTITY_{+}$ near 0::



Example 8.12.
Local graph of $IDENTITY_{-}$ near ∞::



Local graph of $IDENTITY_{-}$ near 0::



7 Features Of Linear Functions

What makes linear functions *exceptional* among monomial functions is that they lack *local concavity* and have only *local height* and *local slope*.

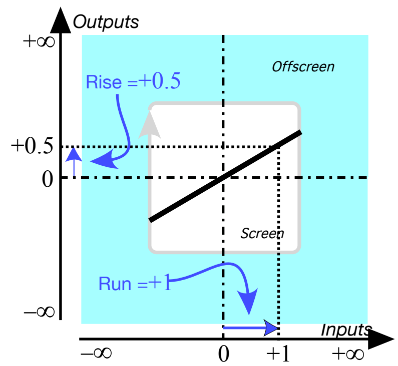
But then, since for a linear function the *local slope* is the same everywhere, the graph of a linear function has a **global slope**, that is the fraction $\frac{\text{Rise}}{\text{Run}}$ where, given two input-output pairs, the **run** is the difference from one input to the other and the **rise** is the corresponding difference from one output to the other.

In fact, the reason we like to use the inputs 0 and 1 is that they make it easy to see that the *global slope* of the *global graph* of a linear function is the *linear coefficient* of the *global input-output rule*. dilation function

EXAMPLE 8.13. Let f be the function specified by the *global input-output rule*

$$x \xrightarrow{f} f(x) = (+0.5)x^{+1}$$
$$= +0.5x$$

the global slope of f is $\frac{\text{Rise}}{\text{Run}} = \frac{+0.5}{+1} = +0.5$



LANGUAGE 8.3 Another name for *linear function* is **dilation function** because it is easy to prove that the distance between any two outputs is obtained by just "dilating" the distance between the two inputs by the coefficient. (See `https://en.wikipedia.org/wiki/Dilation_(metric_space)`.)

base function
add-on number
add-on function
sum function

Chapter 9

Prelude To Polynomial Functions

As already mentioned, monomial functions will be the building blocks from which all the functions we will be investigating in this text are built from. So we will always have to use more than a single monomial function at a time.

1 Adding Functions

1. Given a function, to which we will refer as **base function**, one often needs to add a number to each output that the base function returns. Whether or not this **add-on number** remains the same regardless of the input or differs depending on the input, we can look upon the add-on number as being itself the output returned for the same input by some other function to which we will refer as **add-on function**. (If the add-on number is the same regardless of the input this just means that the add-on function is a *constant function*.)
There is then going to be a third function, to be referred as **sum function**, which, for each input, will return the output returned by the base function plus the add-on number returned by the add-on function for that input.
In other words, given the two functions

bar graph
bar

$$x \xrightarrow{BASE} BASE(x)$$

and

$$x \xrightarrow{ADD\text{-}ON} ADD\text{-}ON(x)$$

there will be a third function specified as

$$x \xrightarrow{SUM} SUM(x) = BASE(x) + ADD\text{-}ON(x)$$

2. In sciences such as
textscBiology,
textscPsychology and
textscEconomics the three functions are often in *tabular* form.

EXAMPLE 9.1. When we shop online for, say for a textbook, we first see a *price list*—the *base function*. However, a *shipping charge*, which might or might not depend on the textbook, is usually added-on to the *list price* and is given by the *Shipping charge list*—the *add-on function*. The price we end-up having to pay is thus given by the *actual price list*—the *sum function*.

$x \xrightarrow{LIST} LIST(x)$	
English	140
History	80
Biology	130
Math	10
Poetry	70

$x \xrightarrow{SHIP} SHIP(x)$	
English	13.15
History	3.45
Biology	7.25
Math	3.75
Poetry	5.32

x	$\xrightarrow{PAY} PAY(x)$
English	$140+13.15 = 153.15$
History	$80+3.45 = 83.45$
Biology	$130+7.35 = 137.25$
Math	$10+3.75 = 13.75$
Poetry	$70+5.32 = 75.32$

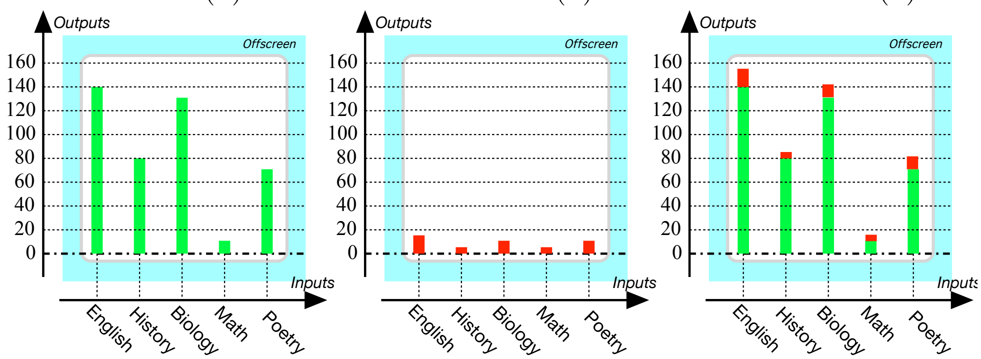
which says, for instance, that while the *list price* of the English textbook is \$140, a *shipping charge* of \$13.15 brings the price to be *paid* to \$140 + \$13.15 = \$153.15.

3. Instead of representing the functions by *tables*, one might want to represent them by *graphs*. Rather than to use *plots*, though, one often uses **bar graphs** in which the pieces of input level lines that are between the 0-*output level line* and the *plot point* are highlighted into **bars**.

binomial function

EXAMPLE 9.2. The situation in the above example would be represented by the following bar graphs.

$x \xrightarrow{LIST} LIST(x)$ $\qquad$ $x \xrightarrow{SHIP} SHIP(x)$ $\qquad$ $x \xrightarrow{PAY} PAY(x)$



2 Binomial Functions

1. Given a *base* function which is a monomial function, when we *add-on* a monomial function with the *same* exponent, the *sum* is a monomial function with the same exponent.

EXAMPLE 9.3. Given the base function $MINT$ specified by the *global input-output rule*

$$x \xrightarrow{MINT} MINT(x) = -12.82x^{+4}$$

and given the add-on function TEA specified by the *global input-output rule*

$$x \xrightarrow{TEA} TEA(x) = +49.28x^{+4}$$

then the sum function will be specified by the *global input-output rule*

$$\begin{aligned} x \xrightarrow{\quad SUM \quad} SUM(x) &= MINT(x) + TEA(x) \\ &= -12.82x^{+4} \oplus +49.28x^{+4} \\ &= [-12.82 \oplus +49.28]\, x^{+4} \\ &= +36.46x^{+4} \end{aligned}$$

2. However, when the exponent of the *add-on* function is different from the exponent of the *base* function, then the *sum* function is not a *monomial function* but what is called a **binomial function**.

EXAMPLE 9.4. Let $BASE$ be specified by the global input-ouput rule

$$x \xrightarrow{BASE} BASE(x) = (-3)x^{+2}$$

and let $ADD\text{-}ON$ be specified by the global input-ouput rule

$$\begin{aligned} x \xrightarrow{ADD\text{-}ON} ADD\text{-}ON(x) &= (+5)x^0 \\ &= +5 \end{aligned}$$

then the SUM function is specified by the global input-ouput rule

$$\begin{aligned} x \xrightarrow{SUM} SUM(x) &= (-3)x^{+2} \oplus (+5)x^0 \\ &= (-3)x^{+2} + 5 \end{aligned}$$

To see that SUM *cannot* be replaced by a single monomial function, we first evaluate all three functions at some input, for instance $+2$:

$$\begin{aligned} +2 \xrightarrow{BASE} BASE(+2) &= (-3)(+2)^{+2} \\ &= -12 \end{aligned}$$

and

$$\begin{aligned} +2 \xrightarrow{ADD\text{-}ON} ADD\text{-}ON(+2 &= (+5)(+2)^0 \\ &= +5 \end{aligned}$$

then

$$\begin{aligned} x \xrightarrow{SUM} SUM(x) &= (-3)(+2)^{+2} \oplus (+5)(+2)^0 \\ &= -12 \oplus +5 \\ &= -7 \end{aligned}$$

The question then is what *monomial function* could return the output -7 for the input $+2$.

Of course, we can easily find a monomial function that would return the output -7 for the input $+2$. For instance, the dilation function $x \xrightarrow{f} f(x) = -\frac{7}{2}x$ does return the output -7 for the input $+2$. But$f$ is *not* going to return the same output as SUM for other inputs, say, $+3$, $+4$, etc which it should. So, the *binomial function*

$$x \xrightarrow{SUM} SUM(x) = (-3)x^{+2} + 5$$

cannot be replaced by the single *monomial function*

$$x \xrightarrow{f} f(x) = -\frac{7}{2}x$$

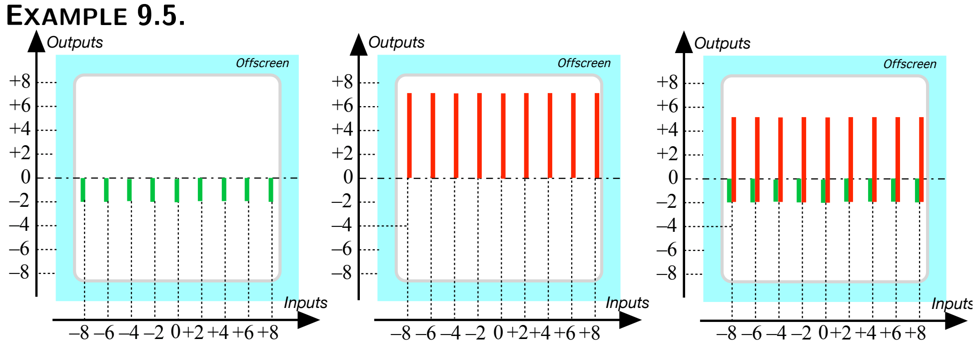
NOTE 9.1
We noted at the beginning of Chapter 5 that monomial functions were only rarely called *monomial functions* and that this was unfortunate: indeed, it would be nicer to say that a *binomial* function cannot be replaced by a single *monomial* function. (We cannot have two for the price of one.)

3 Graphs of Binomial Functions

1. When the exponent of the *add-on* function is the same as the exponent of the *base* function, the bar graphs show exactly why the *sum* function will have again the same exponent.

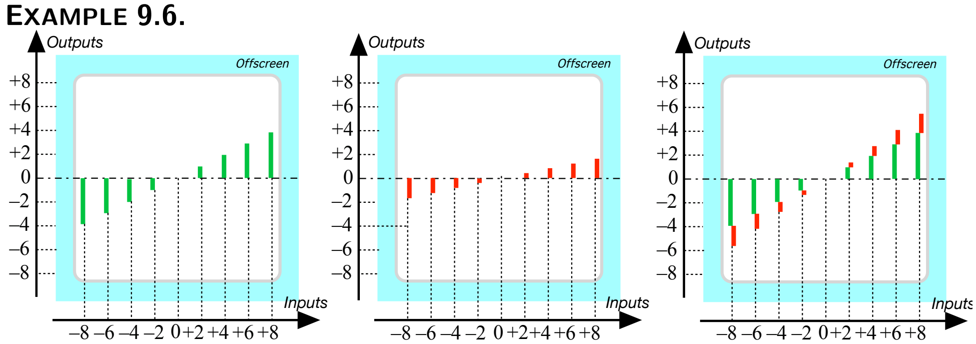
a. Given a constant base function, adding-on a constant function:

EXAMPLE 9.5.



b. Given a dilation base function, adding-on a dilation function:

EXAMPLE 9.6.

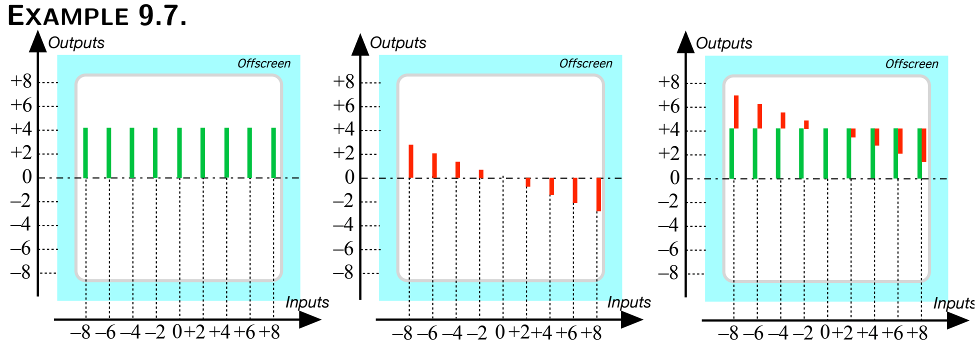


2. When the exponent of the add-on function is *not* the same as the exponent of the base function, the bar graphs show clearly why the *sum* function cannot be a monomial function.

a. Given a constant base function,

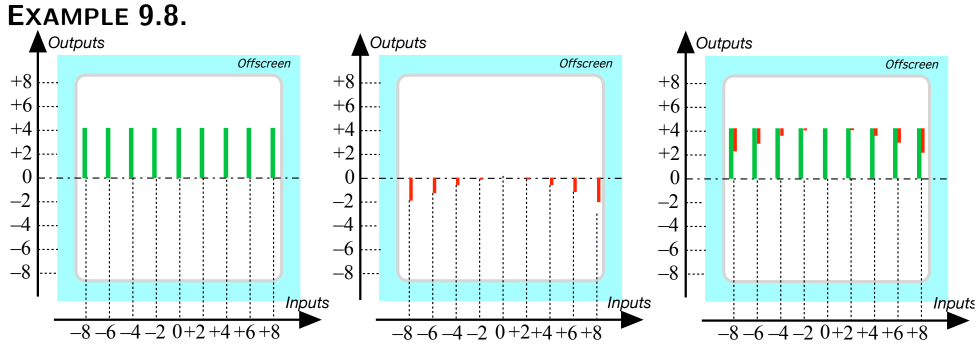
- Adding-on a dilation function:

EXAMPLE 9.7.



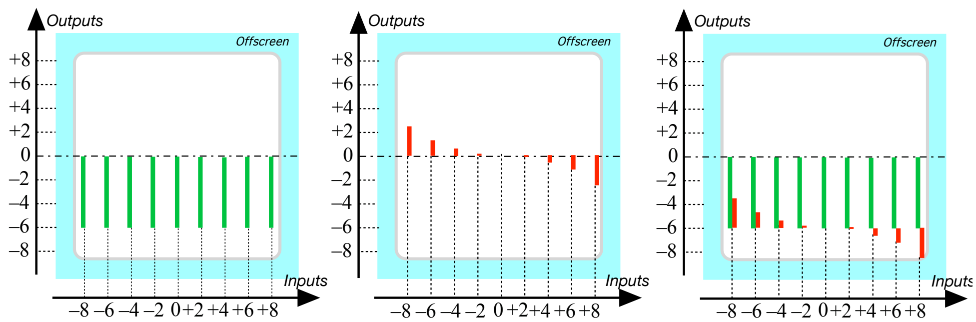
- Adding-on an even positive exponent monomial function:

EXAMPLE 9.8.



- Adding-on an odd positive exponent monomial function:

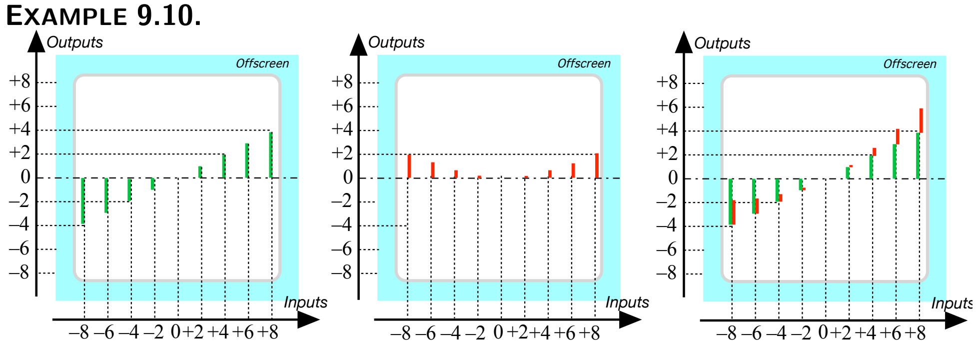
EXAMPLE 9.9.



b. Given a dilation base function,

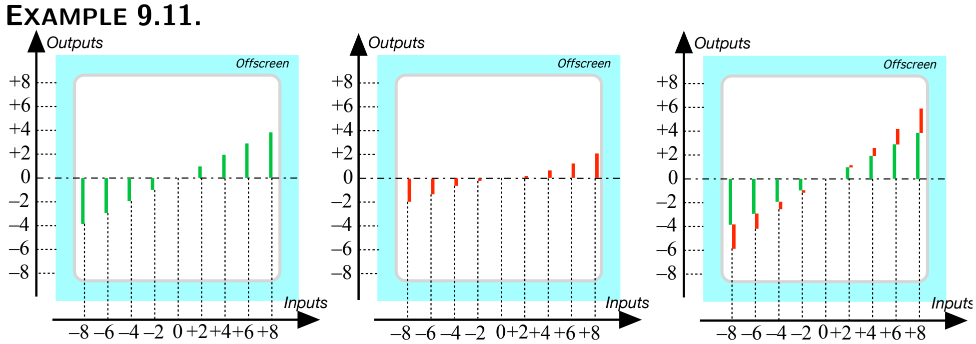
- Adding-on an even monomial function:

Example 9.10.



- Adding-on an odd monomial function:

Example 9.11.



trinomial function

4 Trinomial Functions

There is of course no reason why the base function could not itself be a binomial function. In fact, this can very well be the case and the sum function will then be called a **trinomial function**.

EXAMPLE 9.12. Let $BASE$ be specified by the global input-ouput rule

$$x \xrightarrow{BASE} BASE(x) = (-3)x^0 \oplus (+7)x^{+1}$$

and let $ADD\text{-}ON$ be specified by the global input-ouput rule

$$x \xrightarrow{ADD\text{-}ON} ADD\text{-}ON(x) = (+5)x^{+3}$$

then the SUM function is specified by the global input-ouput rule

$$\begin{aligned} x \xrightarrow{SUM} SUM(x) &= (-3)x^0 \oplus (+7)x^{+1} \oplus (+5)x^{+3} \\ &= -3 + 7x + 5x^{+3} \end{aligned}$$

EXAMPLE 9.13. Let $BASE$ be specified by the global input-ouput rule

$$x \xrightarrow{BASE} BASE(x) = (-3)x^{+1} \oplus (+7)x^0$$

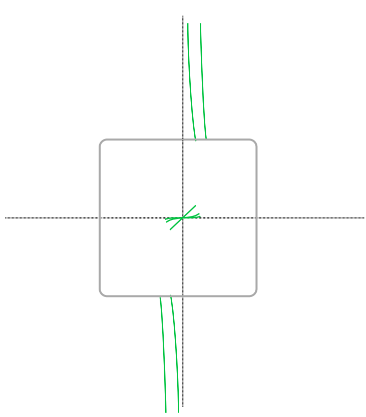
and let $ADD\text{-}ON$ be specified by the global input-ouput rule

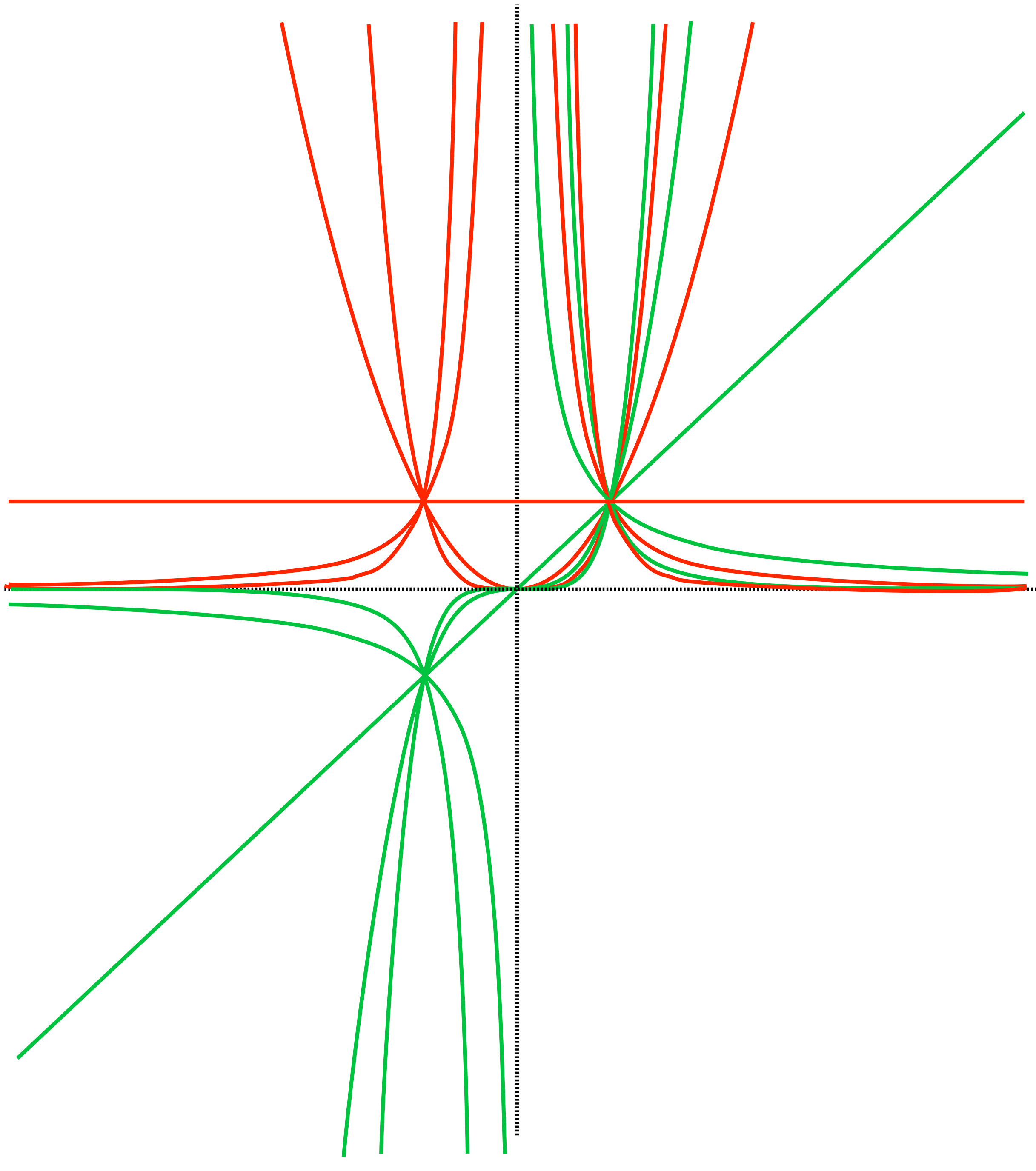
$$x \xrightarrow{ADD\text{-}ON} ADD\text{-}ON(x) = (+5)x^{-2}$$

then the SUM function is specified by the global input-ouput rule

$$\begin{aligned} x \xrightarrow{SUM} SUM(x) &= (-3)x^{+1} \oplus (+7)x^0 \oplus (+5)x^{-2} \\ &= -3x + 7 + 5x^{-2} \end{aligned}$$

5 Comparing Monomial Functions





affine␣function
generic global input-output rule
linear␣coefficient
constant␣coefficient

Chapter 10

Affine Functions: Local Analysis

Affine functions are specified by global input-output rules like the **generic global input-output rule**:

$$x \xrightarrow{AFFINE} AFFINE(x) = \underbrace{a\,x^{+1} \oplus b\,x^{0}}_{\text{output-specifying code}}$$

which we usually write

$$= \underbrace{ax + b}_{\text{output-specifying code}}$$

where a, called the **linear coefficient**, and b, called the **constant coefficient**, are the *bounded* numbers that specify the function $AFFINE$.

EXAMPLE 10.1. The affine function $NINA$ specified by the linear coefficient -31.39 and the constant coefficient $+5.34$ is the function specified by

$$x \xrightarrow{NINA} NINA(x) = \underbrace{-31.39}_{\text{linear coefficient}} \; x \; + \underbrace{5.34}_{\text{constant coefficient}}$$

It is worth noting that

term
linear term
constant term
x_0
generic given input

NOTE 10.1 The terms in the global input output rule *need not* be written in order of *descending* exponent. This is just a habit we have.

EXAMPLE 10.2. The function specified by the global input-output rule

$$x \xrightarrow{NINA} NINA(x) = -31.39x + 5.34$$

could equally well be specified by the global input-output rule

$$x \xrightarrow{NINA} NINA(x) = +5.34 - 31.39x$$

We now introduce some standard terminology to help us describe very precisely what we we will be doing.

The output-specifying code of the affine function specified by

$$x \xrightarrow{AFFINE} AFFINE(x) = \underbrace{ax + b}_{\text{output-specifying code}}$$

consists of two **terms**:

- ax which is called the **linear term**.
- b which is called the **constant term**.

EXAMPLE 10.3. The output-specifying code of the function specified by the global input-output rule

$$x \xrightarrow{NINA} NINA(x) = \underbrace{-31.39x + 5.34}_{\text{Output specifying formula}}$$

consists of two terms:

$$= \underbrace{-31.39x}_{\text{linear term}} \ \underbrace{+5.34}_{\text{constant term}}$$

LANGUAGE 10.1 Whether we look upon b as the constant *coefficient*, that is as the *coefficient* of x^0 in the constant *term* bx^0 or as the constant *term* bx^0 itself with the power x^0 "going without saying" will be clear from the context.

1 Output at x0

We will use $\boldsymbol{x_0}$ as a **generic given input**, that is x_0 is a *bounded* input that has been *given* but whose identity remains *undisclosed* for the time being.

PROCEDURE 10.1 To evaluate at x_0 the function specified by $x \xrightarrow{AFFINE} AFFINE(x) = ax + b$

i. *Declare* that x is to be replaced by x_0

$$x \Big|_{x \leftarrow x_0} \xrightarrow{AFFINE} AFFINE(x) \Big|_{x \leftarrow x_0} = ax + b \Big|_{x \leftarrow x_0}$$

which gives:

$$x_0 \xrightarrow{AFFINE} AFFINE(x_0) = \underbrace{ax_0 + b}_{\text{output-specifying code}}$$

ii. *Execute* the output-specifying code into an output *number*:

$$= ax_0 + b$$

which gives the input-output pair

$$(x_0, ax_0 + b)$$

TEMO 10.1 To evaluate at -3 the function specified by

$$x \xrightarrow{ALDA} ALDA(x) = -32.67x + 71.07$$

i. We *declare* that x is to be replaced by -3

$$x \Big|_{x \leftarrow -3} \xrightarrow{ALDA} ALDA(x) \Big|_{x \leftarrow -3} = -32.67x + 71.07 \Big|_{x \leftarrow -3}$$

which gives

$$-3 \xrightarrow{ALDA} ALDA(-3) = \underbrace{-32.67(-3) + 71.07}_{\text{output specifying code}}$$

ii. We *execute* the output-specifying code into an output *number*:

$$= +98.01 + 71.07$$

$$= +169.08$$

which gives the *input-output pair*

$$(-3, +169.08)$$

However, as already discussed in **?? ??** and as has already been the case with *monomial* functions, instead of getting the output of an affine function *at* a given input, be it ∞ or x_0, we will usually get the output of the affine function *near* that given input.

jet
local input-output rule near ∞

2 Output near ∞

In order to get the output *near* ∞, we could proceed as we did in section 5 Output Near ∞ with monomial functions, that is we could *declare* "x is $\pm large$" and replace x everywhere in the output-specifying code by $\pm large$. However, the output-specifying code of affine functions and all functions thereafter will involve more than just one term and using $\pm large$ would become more and more time consuming.

So, in conformity with universal practice, we will declare "x near ∞" but write just x after that. This, though, is extremely dangerous as it is easy to forget that what we write may be TRUE *only* because x has been declared to be near ∞.

1. We will *execute* the output-specifying code, here $ax + b$, into a **jet**, that is with the terms in *descending order of sizes*, which, because x is *large*, means that the powers of x must be in *descending order of exponents*. We will then have the **local input-output rule near ∞**:

$$x \text{ near } \infty \xrightarrow{AFFINE} AFFINE(x) = \underbrace{ax + b}_{\text{output jet near } \infty}$$

EXAMPLE 10.4. Given the function specified by

$$x \xrightarrow{BIBA} BIBA(x) = -61.03 - 82.47x$$

To get the jet near ∞, we first need to get the *order of sizes*.

i. -61.03 is *bounded*

ii. -82.47 is *bounded* and x is *large*. So, since $bounded \cdot large = large$, $-82.47 \cdot x$ is *large*

Then, in the jet near ∞, $-82.47x$ must come first and -61.03 comes second

So, we get the local input-output rule near ∞:

$$x \text{ near } \infty \xrightarrow{BIBA} BIBA(x) = \underbrace{-82.47x - 61.03}_{\text{output jet near } \infty}$$

2. Altogether, then:

PROCEDURE 10.2 To evaluate near ∞ the function specified by $x \xrightarrow{AFFINE} AFFINE(x) = ax + b$

i. *Declare* that x is near ∞

$$x\Big|_{x \text{ near } \infty} \xrightarrow{AFFINE} AFFINE(x)\Big|_{x \text{ near } \infty} = ax + b\Big|_{x \text{ near } \infty}$$

which gives:

$$\boxed{x \text{ near } \infty} \xrightarrow{AFFINE} \boxed{AFFINE(x)} = \underbrace{a\boxed{x} + b}_{\text{output-specifying code}}$$

ii. *Execute* the output-specifying code into a *jet* near ∞

$$\boxed{x \text{ near } \infty} \xrightarrow{AFFINE} \boxed{AFFINE(x)} = \underbrace{\left[\boxed{a}\right] \boxed{x} \oplus \left[\boxed{b}\right]}_{\text{output jet near } \infty}$$

where

- $\boxed{a}$ is the *linear* coefficient in the jet near ∞
- $\boxed{b}$ is the *constant* coefficient in the jet near ∞.

which gives the *local input-output rule* near ∞:

$$\boxed{x \text{ near } \infty} \xrightarrow{AFFINE} \boxed{AFFINE(x)} = \underbrace{\boxed{\left[\boxed{a}\right] \boxed{x} \oplus \left[\boxed{b}\right]}}_{\text{output jet near } \infty}$$

(*Here* the jet near ∞ looks the same as the given global input-output rule but that is only because the output-specifying code *happened* to be written in *descending* order of exponents.)

linear coefficient in the jet near ∞

constant coefficient in the jet near ∞

TEMO 10.2 To evaluate $\boxed{\text{near } \infty}$ the function specified by

$$x \xrightarrow{NINA} NINA(x) = -61.03 - 82.47x$$

i. We *declare* that x is near $\boxed{\infty}$

$$x \Big|_{x \text{ near } \infty} \xrightarrow{NINA} NINA(x) \Big|_{x \text{ near } \infty} = -61.03 - 82.47x \Big|_{x \text{ near } \infty}$$

which gives:

$$\boxed{x \text{ near } \infty} \xrightarrow{NINA} \boxed{NINA(x)} = \underbrace{-61.03 - 82.47\boxed{x}}_{\text{output-specifying code}}$$

ii. We *execute* the output-specifying code into a *jet* near ∞:

$$= \left[\boxed{-82.47}\right] \boxed{x} \oplus \left[\boxed{-61.03}\right]$$

which gives the *local input-output rule* near ∞:

$$\boxed{x \text{ near } \infty} \xrightarrow{NINA} \boxed{NINA(x)} = \underbrace{\left[\boxed{-82.47}\right] \boxed{x} \oplus \left[\boxed{-61.03}\right]}_{\text{output jet near } \infty}$$

where:

- $\boxed{-82.47}$ is the *linear* coefficient in the jet near ∞

approximate

- -61.03 is the *constant* coefficient in the jet near ∞.

(Here the jet near ∞ does *not* look the same as the *global* input-output rule because the output-specifying code happened *not* to be in descending order of exponents.)

3. The reason we use *jets* here is that the term *largest in size* is the *first* term so that to **approximate** the output we need only write the *first* term in the jet and just replace the remaining terms by [...] which stands for "something too small to matter here". In other words,

THEOREM 10.1 Approximate output near ∞. For *affine* functions, the term in the jet that contributes most to the output near ∞ is the *highest degree term* in the output jet near ∞:

$$x \text{ near } \infty \xrightarrow{AFFINE} AFFINE(x) = \left[a \right] x + [...]$$

EXAMPLE 10.5. Given the function specified by

$$x \xrightarrow{NINA} NINA(x) = -61.03 - 82.47x$$

$$x \text{ near } \infty \xrightarrow{NINA} NINA(x) = \left[-82.47 \right] x + \left[-61.03 \right]$$

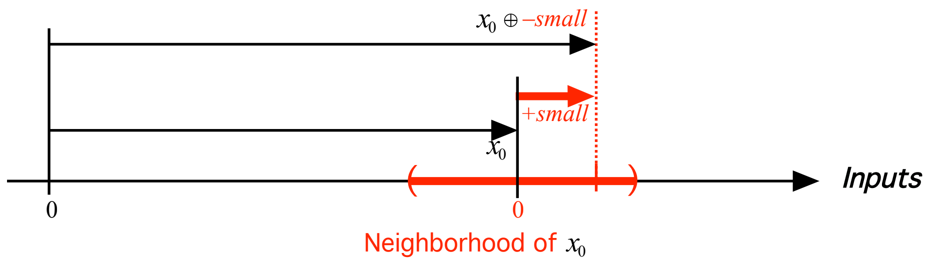
near ∞ we will often just use the *approximation*

$$x \text{ near } \infty \xrightarrow{NINA} NINA(x) = \left[-82.47 \right] x + [...]$$

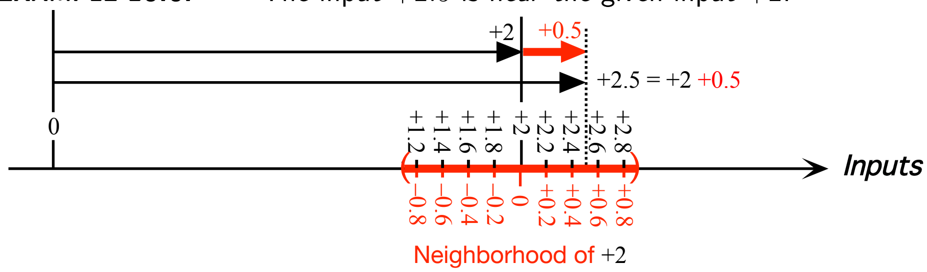
3 Output near x_0

While with monomial functions 0 played just as important a role as ∞ (Section 4 Reciprocity), this will not at all be the case with affine functions and all functions thereafter as we *will* very often be interested in the neighborhood of some *given* bounded input(s) *other* than 0. As a matter of fact, the input 0 will usually not be of much more interest than other bounded inputs. (But we will often be concerned with the *output* 0.)

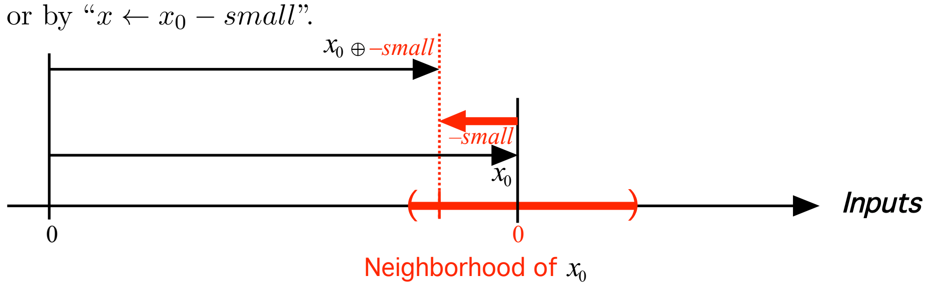
1. In order to "thicken the plot" near a given *bounded* input, we could proceed basically just as we did in section 6 Output Near 0 with monomial functions, that is *declare* "$x \leftarrow x_0 + small$" or "$x \leftarrow x_0 - small$"and replace x everywhere in the output-specifying code by "$x_0 \oplus +small$"



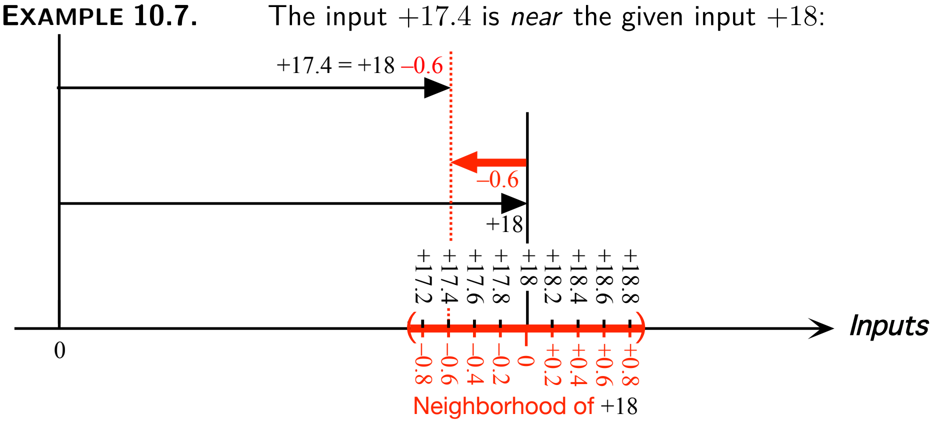
EXAMPLE 10.6. The input $+2.5$ is *near* the given input $+2$:



or by "$x \leftarrow x_0 - small$".



EXAMPLE 10.7. The input $+17.4$ is *near* the given input $+18$:



However, as already pointed out in **??** **??**, unlike monomial functions the

h
output jet near x_0

output-specifying code of affine functions and all functions thereafter will involves more than just one term. So, using "$x_0 \oplus +small$" or "$x_0 \oplus -small$" would become more and more time consuming and instead we will use "$x_0 + h$" where the letter $\boldsymbol{h}$ is universally accepted as standing for $+small$ or $-small$. In other words, h already includes the *sign*.
Of course, in order to input a neighborhood of 0, we will declare that $x \leftarrow h$, aka $x \leftarrow 0 + h$, in other words that x is to be replaced by h.

2. We can then *execute* the input-output specifying phrase into a *jet* that is with the terms in **descending *order of sizes*** which here, since h is *small*, means that the powers of h will have to be in *ascending* order of exponents. We will then have the local input-output rule near the given input:

$$x_0 \oplus h \xrightarrow{AFFINE} AFFINE(x_0 \oplus h) = \underbrace{\text{Powers of } h \text{ in } \textit{ascending order of exponents}}_{\text{output jet near } \infty}$$

3. We will therefore use:

PROCEDURE 10.3 To evaluate near x_0 the function specified by $x \xrightarrow{AFFINE} AFFINE(x) = ax + b$

i. *Declare* that x is to be replaced by $x_0 + h$

$$x\Big|_{x \leftarrow x_0+h} \xrightarrow{AFFINE} AFFINE(x)\Big|_{x \leftarrow x_0+h} = ax + b\Big|_{x \leftarrow x_0+h}$$

which gives:

$$x_0 + h \xrightarrow{AFFINE} AFFINE(x_0 + h) = \underbrace{a(x_0 + h) + b}_{\text{output-specifying code}}$$

ii. *Execute* the output-specifying code into a *jet* near x_0:

$$= ax_0 + ah + b$$

$$= \underbrace{\Big[ax_0 + b\Big] \oplus \Big[a\Big]h}_{\text{output jet near } x_0}$$

which gives the *local input-output rule* near x_0:

$$x_0 + h \xrightarrow{AFFINE} AFFINE(x_0 + h) = \underbrace{\Big[ax_0 + b\Big] \oplus \Big[a\Big]h}_{\text{output jet near } x_0}$$

TEMO 10.3 To evaluate near -3 the function specified by

$$x \xrightarrow{ALDA} ALDA(x) = -32.67x + 71.07$$

i. We declare that x is to be replaced by $-3+h$

$$x\,\Big|_{x\leftarrow -3+h} \xrightarrow{ALDA} ALDA(x)\,\Big|_{x\leftarrow -3+h} = -32.67x + 71.07\,\Big|_{x\leftarrow -3+h}$$

which gives

$$-3+h \xrightarrow{ALDA} ALDA(-3+h) = \underbrace{-32.67(-3+h) + 71.07}_{\text{output specifying code}}$$

ii. We *execute* the output-specifying code into a *jet* near -3:

$$\begin{aligned}
&= -32.67(-3) - 32.67h + 71.07\\
&= +98.01 - 32.67h + 71.07\\
&= +98.01 + 71.07 - 32.67h\\
&= \underbrace{\Big[+169.08\Big] \oplus \Big[-32.67\Big]h}_{\text{output jet near } -3}
\end{aligned}$$

which gives the *local input-output rule* near -3:

$$-3+h \xrightarrow{ALDA} ALDA(-3+h) = \underbrace{\Big[+169.08\Big] \oplus \Big[-32.67\Big]h}_{\text{output jet near } -3}$$

4. When all we want is a feature-sign, though, the above procedure is inefficient and we will then use the following procedure based directly on the fact that an *affine function* is the addition of:

- a *linear function*, (See **??** on **??**.)
- a *constant function*. (See **??** on **??**.)

that is:

$$x \xrightarrow{AFFINE} AFFINE(x) = \underbrace{cx}_{\text{linear}} \oplus \underbrace{d}_{\text{constant}}$$

We declare that x is near x_0 that is that x must be replaced by x_0+h:

$$x \xrightarrow{AFFINE} AFFINE(x) = \underbrace{c\,(x_0+h)}_{\text{linear}} \oplus \underbrace{d}_{\text{constant}}$$

The output of the local input-output rule near x_0 will have to be a *jet*:

$$x_0+h \xrightarrow{AFFINE} AFFINE(x_0+h) = \Big[\quad\Big] \oplus \Big[\quad\Big]h$$

and we want to be able to get any one of the coefficients of the output jet without having to compute any of the other coefficients. So, what we will do is to get the contribution of each monomial function to the term we want.

More precisely,

i. If we want the *coefficient* of h^0 in the output jet:

- The linear function contributes cx_0
- The constant function contributes d

so we have:

$$x_0 + h \xrightarrow{AFFINE} AFFINE(x_0+h) = \Big[cx_0 + d\Big] \oplus \Big[\quad\Big]h$$

ii. If we want the *coefficient* of h^1 in the output jet:

- The linear monomial function contributes c
- The constant monomial function contributes nothing

so we have:

$$x_0 + h \xrightarrow{AFFINE} AFFINE(x_0+h) = \Big[\quad\Big] \oplus \Big[c\Big]h$$

4 Local graphs

Just as we get a *plot point at* a *bounded* input from the *output at* that input, we get the *local graph near* any input, be it *bounded* or *infinity*, from the *jet near* that input.

PROCEDURE 10.4 To graph near ∞ the function specified by $x \xrightarrow{AFFINE} AFFINE(x) = ax + b$

1. Get the jet near ∞ using **?? ??** on **??**

$$x \text{ near } \infty \xrightarrow{AFFINE} AFFINE(x) = \Big[a\Big]x + \Big[b\Big]$$

2. Get the local graph near ∞ of each term:

a. Get the graph of the *linear term* near ∞ by graphing near ∞ the monomial function $x \to ax$ using **?? ??** on **??**.

b. Get the graph of the *constant term* near ∞ by graphing near ∞ the monomial function $x \to b$ using **?? ??** on **??**.

3. Get the local graph near ∞ of $AFFINE$ by adding-on the constant term to the linear term using chapter 9.

TEMO 10.4 To graph near ∞ the function specified by

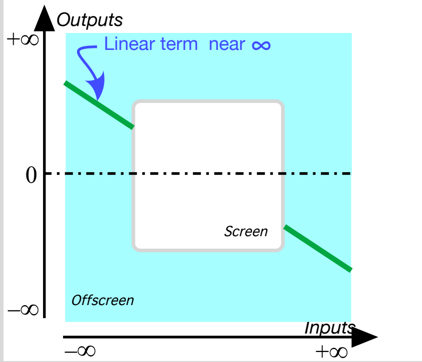
$$x \xrightarrow{NINA} NINA(x) = -61.03 - 82.47x$$

1. We get the jet near ∞: (See Demo 10.2 on page 229)

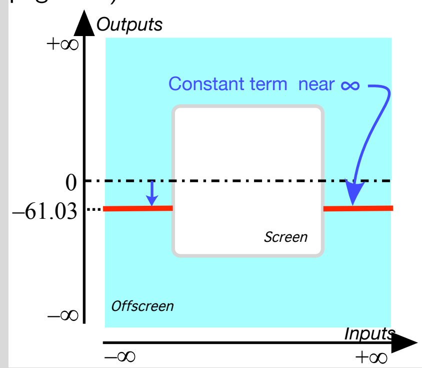
$$x \text{ near } \infty \xrightarrow{NINA} NINA(x) = \Big[-82.47\Big]x + \Big[-61.03\Big]$$

2. Get the local graph near ∞ of each term:

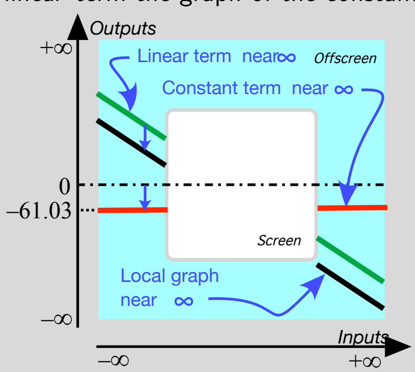
a. We get the graph of the linear term by graphing near ∞ the monomial function $x \to \Big[-82.47\Big]x$ (See Demo 6.24 on page 175)



b. We get the graph of the constant term near ∞ by graphing near ∞ the monomial function $x \to \Big[-61.03\Big]$ (See Demo 6.24 on page 175)



3. We get the local graph near ∞ of $NINA$ by adding-on to the graph of the *linear* term the graph of the *constant* term. (See Demo 6.24 on page 175)



PROCEDURE 10.5 To graph near x_0 the function specified by the generic global input-output rule $x \xrightarrow{AFFINE} AFFINE(x) = ax + b$

i. Get the jet near x_0 of $AFFINE$ using **??** **??** on **??**
ii. Get the graph of the constant term in the jet near x_0 namely of $\Big[ax_0 + b\Big]$
iii. Add-on the graph of the linear term in the jet near x_0 namely of $\Big[a\Big]h$

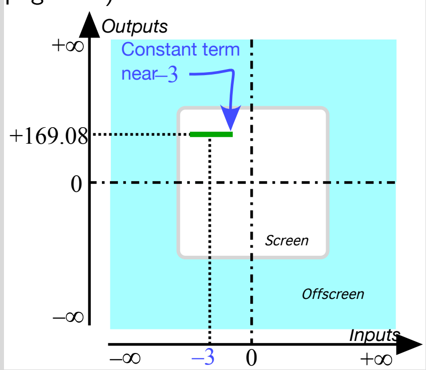
Temo 10.5 To graph near -3 the function specified by

$$x \xrightarrow{ALDA} ALDA(x) = -32.67x + 71.07$$

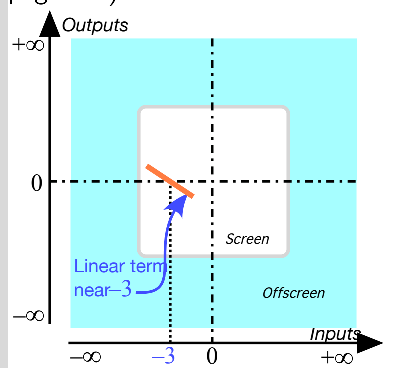
i. We get the jet near -3 of $ALDA$ by evaluating $ALDA$ near -3: (See Demo 10.3 on page 232)

$$-3+h \xrightarrow{ALDA} ALDA(-3+h) = \underbrace{\left[+169.08\right] \oplus \left[-32.67\right]h}_{\text{output jet near } -3}$$

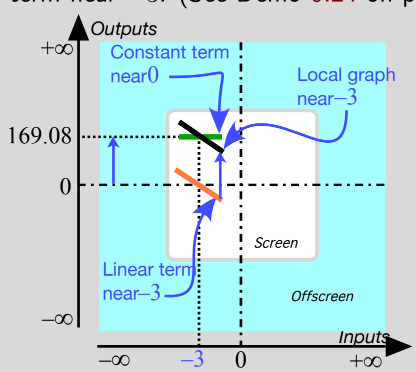
ii. We get the graph of the constant term near -3: (See Demo 6.24 on page 175)



iii. We get the graph of the linear term near -3 is: (See Demo 6.24 on page 175)



iv. We add-on the graph of the linear term near -3 to the graph of the linear term near -3. (See Demo 6.24 on page 175)



5 Local Feature-signs

As we saw in **?? ??**, a feature-sign near a given input, be it near ∞ or near x_0, can be read from the *local graph* and so all we need to do is:

i. Get the *output jet* from the global input-output rule. (See **??** on **??** when the given input is ∞ or **??** on **??** when the given input is x_0.)

ii. Get the *local graph* from the output jet. (See **??** on **??** when the given input is ∞ or **??** on **??** when the given input is x_0.)

iii. Get the *feature-sign* from the *local graph*(See **??**

However, with a little bit of reflection, it is faster and *much more useful* to read the feature-signs directly from the *jet* in the local input-output rule. But since, in order for the terms in the jet to be in *descending order of sizes*,

- In the case of *infinity*, the exponents of x have to be in *descending* order.
- In the case of a *bounded input*, the exponents of h have to be in *ascending* order.

we will deal with ∞ and with x_0 separately.

1. Near*infinity* things are quite straightforward:

PROCEDURE 10.6 To get the feature-signs near ∞ of the function specified by $x \xrightarrow{AFFINE} AFFINE(x) = ax + b$

i. Get the local input-output rule near ∞:

$$x \text{ near } \infty \xrightarrow{AFFINE} AFFINE(x) = ax + b$$
$$= \underbrace{\big[a\big]x \oplus \big[b\big]}_{\text{output jet near } \infty}$$

ii. Then, in the *jet* near ∞:

- Get both the *Height-sign* and the *Slope-sign* from the *linear term* $\big[a\big]x$ because the next term $\big[b\big]$ is *too small to matter*.
- Since both the *linear term* and the *constant term* have no concavity, $AFFINE$ has no *Concavity-sign* near ∞.

TEMO 10.6 Get the Height-sign near $\infty =$ of the function specified by

$$x \xrightarrow{JULIE} JULIE(x) = -2x + 6$$

i. We get the local input-output rule near ∞:

$$x \text{ near } \infty \xrightarrow{JULIE} JULIE(x) = -2x + 6$$
$$= \underbrace{\Big[-2\Big]x \oplus \Big[+6\Big]}_{\text{output jet near } \infty}$$

ii. We get *Height-sign* from the *linear* term $\Big[-2\Big]x$ because the *constant* term $\Big[+6\Big]$ is *too small to matter*.
Since the *linear coefficient* -2 is negative, we get that Height-sign $JULIE$ near $\infty = \langle -, + \rangle$. (Seen from ∞.)

TEMO 10.7 Get the Slope-signs near ∞ of the function specified by

$$x \xrightarrow{PETER} PETER(x) = +3x - 8$$

i. We get the local input-output rule near ∞:

$$\begin{aligned} x \text{ near } \infty \xrightarrow{PETER} PETER(x) &= +3x - 8 \\ &= \underbrace{\Big[+3\Big]x \oplus \Big[-8\Big]}_{\text{output jet near } \infty} \end{aligned}$$

ii. We get *Slope-sign* from the *linear* term $\Big[+3\Big]x$ because the *constant* term $\Big[-8\Big]$ is *too small to matter* (Not to mention that a *constant* term has *no* slope.)
Since the *linear coefficient* $+3$ is positive, we get that Slope-sign $PETER$ near $\infty = \langle \diagup, \diagup \rangle$. (Seen from ∞.)

2. In the case of a *bounded input*, things are a bit more complicated because the bounded input may turn out to be *ordinary* or *critical* for the *height*. But it will always be *ordinary* for the slope.

PROCEDURE 10.7 To get the feature-signs near x_0 of the function specified by $x \xrightarrow{AFFINE} AFFINE(x) = ax + b$

i. Get the local input-output rule near x_0:

$$\begin{aligned} x_0 + h \xrightarrow{AFFINE} AFFINE(x_0 + h) &= a(x_0 + h) + b \\ &= ax_0 + ah + b \\ &= ax_0 + b + ah \\ &= \underbrace{\big[ax_0 + b\big] \oplus \big[a\big]h}_{\text{output jet near } x_0} \end{aligned}$$

ii. Then, in the *jet* near x_0:

- If x_0 is *ordinary*, that is if $\big[ax_0 + b\big] \neq 0$, get the *Height-sign* from the *sign* of the constant term $\big[ax_0 + b\big]$ because the next term $\big[a\big]h$ is *too small to matter*. In other words, Height-sign $AFFINE$ near $x_0 =$ Height-sign of the monomial function $h \to ax_0 + b$ near 0.
  But if x_0 is *critical*, that is if $\big[ax_0 + b\big] = 0$, the next term, namely

the *linear term* $\big[a\big]h$, now does matter even though it is *small*. In other words, now Height-sign $AFFINE$ near x_0 = Height-sign of the monomial function $h \to ah$ near 0.

- Since the *constant term* has no slope, get the *Slope-sign* from the next smaller term in the jet, namely the *linear term*. In other words, Slope-sign $AFFINE$ near x_0 = Slope-sign of the monomial function $h \to ah$ near 0.
- Since both the *constant term* and the *linear term* have no concavity, $AFFINE$ has no *Concavity-sign* near x_0.

TEMO 10.8 Get the feature-signs near $+2$ of the function specified by

$$x \xrightarrow{JULIE} JULIE(x) = -2x - 6$$

i. We get the local input-output rule near $+2$:

$$\begin{aligned} +2+h \xrightarrow{JULIE} JULIE(+2+h) &= -2(+2+h) - 6 \\ &= -2(+2) - 2h - 6 \\ &= -4 - 2h - 6 \\ &= -4 - 6 - 2h \\ &= \underbrace{\Big[-10\Big] \oplus \Big[-2\Big]h}_{\text{output jet near } +2} \end{aligned}$$

ii. Then, from the *jet*:

- We get the Height-sign of $JULIE$ from the *constant term* $\Big[-10\Big]$ and since the Height-sign of the monomial function $h \to -10$ near 0 is $\langle -, - \rangle$, we get that Height-sign $JULIE$ near $+2 = \langle -, - \rangle$.
- Since the *constant term* $\Big[-10\Big]$ has no slope we get Slope-sign from the next term, namely the *linear term* $\Big[-2\Big]h$, and since the Slope-sign of the monomial function $h \to -2h$ near 0 is $\langle \diagdown, \diagdown \rangle$, we get that Slope-sign $JULIE$ near $+2 = \langle \diagdown, \diagdown \rangle$.
- Since the *constant term* $\Big[-10\Big]$ and the *linear term* $\Big[-2h\Big]$ both have no concavity, $JULIE$ has no Concavity-sign near $+2$.

TEMO 10.9 Get the feature-signs near -2 of the function specified by

$$x \xrightarrow{PETER} PETER(x) = +3x + 6$$

i. We get the local input-output rule near -2:

$$\begin{aligned}-2+h \xrightarrow{\ PETER\ } PETER(-2+h) &= +3(-2+h)+6\\ &= +3(-2)+3h+6\\ &= -6+3h+6\\ &= -6+6+3h\\ &= \underbrace{\Big[0\Big] \oplus \Big[+3\Big]h}_{\text{output jet near } -2}\end{aligned}$$

ii. Then, from the *jet*:

- Since the *constant term* is 0, we get Height-sign of $PETER$ from the next term, namely the *linear term* $\Big[+3\Big]h$ even though it is $small$. Since the Height-sign of the monomial function $h \to +3h$ near 0 is $\langle -, + \rangle$ we get that Height-sign $PETER$ near $-2 = \langle -, + \rangle$.
- Since the *constant term* $\Big[0\Big]$ has no slope we get Slope-sign from the next term, namely the *linear term* $\Big[+3\Big]h$, and since the Slope-sign of the monomial function $h \to +3h$ near 0 is $\langle \diagup, \diagup \rangle$ we get that Slope-sign $PETER$ near $-2 = \langle \diagup, \diagup \rangle$
- Since the *constant term* $\Big[0\Big]$ and the *linear term* $\Big[+3h\Big]$ both have no concavity, $PETER$ has no Concavity-sign near -2.

EVERYTHING IN THE SOURCE AFTER THIS BELONG ELSEWHERE AND IS COMMENTED OUT THREE TIMES HERE.

general local analysis

Chapter 11

Affine Functions: Global Analysis

In contrast with *local* analysis which involves only inputs that are near a given input, be it ∞ or x_0, *global* analysis involves, one way or the other, *all* inputs. We will see that, while the *local analysis* of all algebraic functions will turn out to remain essentially the same, the *global analysis* of each kind of algebraic functions will turn out to be vastly different.

In fact, with most functions, we will be able to solve only *some* global problems and mostly only *approximately* so. *Affine functions*, though, are truly *exceptional* in that we will be able to solve *all* global problems *exactly*.

Anyway, the first step in investigating the global behavior of a kind of algebraic function will always be to do the **general local analysis** of that kind of algebraic function, that is the local analysis of the *generic algebraic function* of that kind near ∞ and near a generic input x_0.

1 Smoothness

Given the function specified by the generic global input-output rule

$$x \xrightarrow{AFFINE} AFFINE(x) = ax + b$$

generic local input-output rule
first derivative

the **generic local input-output rule** is:

$$x_0 + h \xrightarrow{AFFINE} AFFINE(x_0 + h) = \underbrace{\left[\boxed{ax_0 + b}\right] \oplus \left[\boxed{a}\right] h}_{\text{jet near } x_0}$$

1. The constant term in the jet near x_0, namely $\left[ax_0 + b\right]$, is just the output *at* x_0. (See **??** on **??**). In other words:

> **THEOREM 11.1** The function which outputs *at* the given input the *constant coefficient in the jet* of a given affine function near a given *bounded* input is the given affine function itself.

EXAMPLE 11.1. Observe that in the local input-output rule in Demo 10.3 on page 232 the *constant coefficient* in the jet *near* -3, namely $\boxed{+169.08}$, is just the output *at*-3. (See Demo 12.1 on page 255)

2. Since the *linear term* in the jet of an affine function near x_0, namely $\left[a\right]h$, is small, we have:

> **THEOREM 11.2 Approximate output near x_0.** For *affine* functions, inputs *near* x_0 have outputs that are *near* the output *at* x_0.

which, with the language we introduced in **??**, we can rephrase as:

> **THEOREM ?? (Restated) ??** All affine functions are *continuous* at all inputs.

(In fact, we will see that this will also be the case for all the functions which we will be investigating *in this text*.)

3. The function which outputs the linear coefficient in the jet of a given affine function near a given input is called the **first derivative** of the given function.

2 The Essential Question

As always when we set out to investigate any kind of functions, the first thing we must do is to find out if the *offscreen graph* of an *affine function* consists of just the *local graph near* ∞ or if it also includes the *local graph near one or more* ∞*-height inputs.*

In other words, we need to ask the **Essential Question**:

- Do all *bounded inputs* have *bounded outputs*

or

- Are there *bounded inputs* that are ∞-height inputs, that is are there inputs whose nearby inputs have unbounded outputs?

Now, given a *bounded* input x, we have that:
- since a is bounded, ax is also bounded
- b is bounded

and so, altogether, we have that $ax + b$ is bounded and that the answer to the **Essential Question** is:

> **THEOREM 11.3 Approximate output near ∞.** Under an *affine function*, all bounded inputs return *bounded outputs*.

and therefore

> **THEOREM 11.4 Offscreen Graph.** The *offscreen graph* of an *affine function* consists of just the *local graph near* ∞.

EXISTENCE THEOREMS

The notable inputs are those
- whose existence is forced by the *offscreen graph* which, by the **Bounded Height Theorem** for affine functions, consists of only the *local graph near* ∞.
- whose number is limited by the interplay among the three features

Since polynomial functions have no *bounded* ∞-height input, the only way a feature can change sign is near an input where the feature is 0. Thus, with affine functions, the feature-change inputs will also be 0-feature inputs.

None of the theorems, though, will indicate *where* the notable inputs are. The **Location Theorems** will be dealt with in the last part of the chapter.

EXAMPLE 11.2. When somebody has been shot dead, we can say that there is a murderer somewhere but locating the murderer is another story.

3 Slope-sign

Given the affine function $AFFINE_{a,b}$, that is the function specified by the global input-output rule

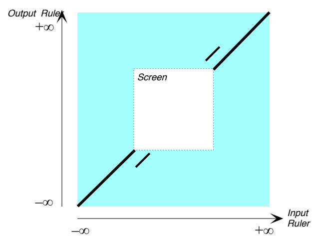
$$x \xrightarrow{AFFINE} AFFINE(x) = ax + b$$

recall that when x is near ∞ the **Slope-sign Near ∞ Theorem** says that:

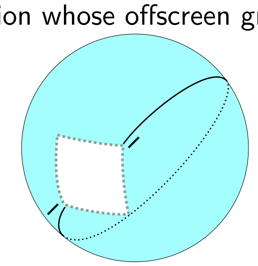
- When a is $+$, $\text{Slope-Sign}|_{x \text{ near } \infty} = (\diagup, \diagup)$
- When a is $-$, $\text{Slope-Sign}|_{x \text{ near } \infty} = (\diagdown, \diagdown)$

1. Since the slope does *not* changes sign as x goes through ∞ from the left side of ∞ to the right side of ∞, the slope need not change sign as x goes *across the screen* from the left side of ∞ to the right side of ∞ so there does not have to be a *bounded* Slope-sign change input:

EXAMPLE 11.3. Given an affine function whose offscreen graph is

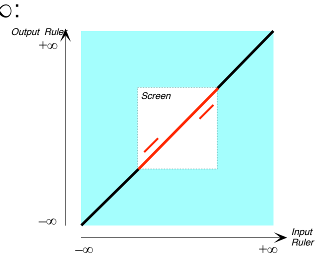


Mercator view

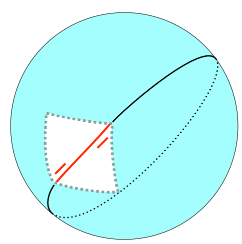


Magellan view

we don't need a bounded slope-sign change input to join smoothly the local graphs near ∞:



Mercator view



Magellan view

2. In fact, not only does there not have to be a bounded slope-sign change input, there *cannot* be a bounded slope-sign change input since the *local* linear coefficient is equal to the *global* linear coefficient a and the slope must therefore be the same everywhere:

> **THEOREM 11.5 Slope-Sign Change Non-Existence .** An affine function has no *bounded* Slope-Sign Change input.

3. Another consequence of the fact that the local slope does not depend on x_0, and is thus the same everywhere, is that it is a feature of the function $AFFINE_{a,b}$ itself and so that the function $AFFINE_{a,b}$ has a **global slope** specified by the global linear coefficient a. global slope

4. Moreover, the slope cannot be equal to 0 somewhere because the slope is equal to a everywhere. So, we also have:

THEOREM 11.6 0-Slope Input Non-Existence . An affine function has *no* bounded 0-*slope* input.

4 Extremum

From the *optimization* viewpoint, an affine function has no extremum input, that is no bounded input whose output would be larger (or smaller) than the output of nearby inputs.

THEOREM 11.7 Extremum Non-existence . An affine function has no bounded local extremum input.

5 Height-sign

Given the affine function $AFFINE_{a,b}$, that is the function specified by the global input-output rule

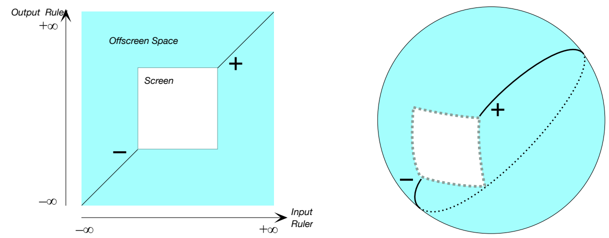
$$x \xrightarrow{AFFINE} AFFINE(x) = ax + b$$

recall that when x is near ∞ the **Height-sign Near ∞ Theorem** says that:

- When a is $+$, Height-Sign$|_{x \text{ near } \infty} = (+, -)$
- When a is $-$, Height-Sign$|_{x \text{ near } \infty} = (-, +)$

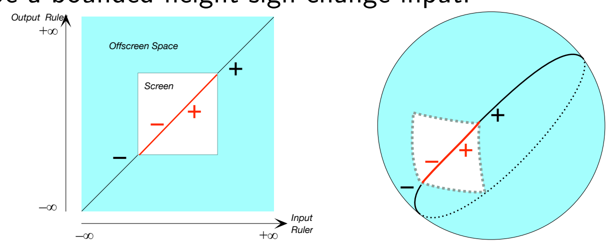
1. Since the height changes sign as x goes from the left side to the right side of ∞ across ∞, the height must also change sign as x goes from the left side to the right side of ∞ *across the screen* so there has to be at least one *bounded Height-sign change input*:

EXAMPLE 11.4. Given the affine function whose offscreen graph is straight



Mercator view Magellan view

there has to be a bounded height-sign change input:



Mercator view Magellan view

2. On the other hand, an affine function can have *at most one* 0-height input because, if it had more, it would have to have 0-slope inputs in-between the 0-height inputs which an affine function cannot have. So, we have:

THEOREM 11.8 0-Height Existence. An affine function has *exactly one* bounded 0-*height* input and it is a 0-height input: $x_{\text{Height-sign change}} = x_{\text{0-height}}$

6 Bounded Graph

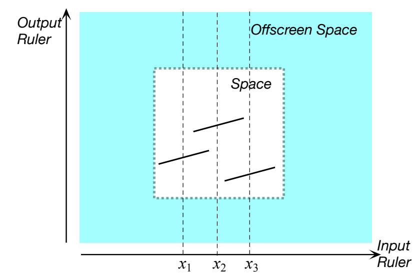
There are two ways to look at the shape of the bounded graph.

1. As a consequence of the **Bounded Height Theorem** for *affine* functions, the offscreen graph consists only of the local graph near ∞ and we can obtain the *forced bounded graph* by extrapolating smoothly the local graph near ∞.

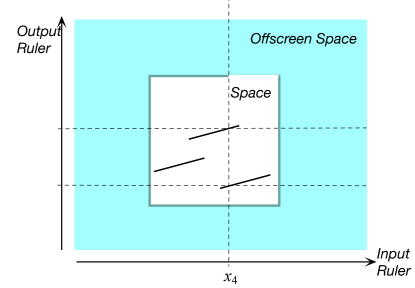
There remains however a question namely whether the extrapolated bounded graph is **straight** that is has no concavity. However, affine functions have no concavity and that settles the mater: the local graph near $-\infty$ and the local graph near $+\infty$ must be lined up and can therefore be joined smoothly with a straight line.

2. In the case of *affine functions*, it happens that we can also obtain the *bounded graph* by interpolating local graphs near bounded inputs:
We start from the local graphs near a number of bounded points as follows:

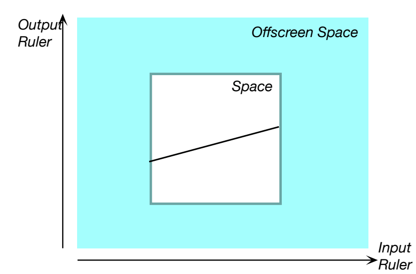
We construct local graphs near, say, three different *bounded* inputs, x_1, x_2, x_3. They would look something like this:



However, this is not possible because that would mean that inputs such as x_4 would have *two* outputs:



As a result, the *local* graphs near bounded inputs *must* all line up and so the *bounded* graph *must* be a straight line:



Of course, the bounded graph must line up with the local graph near ∞ as, otherwise, there would have to be a jump in the transition zone.

Location Theorems

Previously, we only established the *existence* of certain notable features of affine functions and this investigation was based on *graphic* considerations. Here we will investigate the *location* of the inputs where these notable features occur and this investigation will be based on *input-output rule* considerations.

7 0-Slope Location

We saw earlier that affine functions cannot have a 0-slope input. On the other hand, since the slope is the same everywhere, it is a global feature of the function itself and we have:

THEOREM 11.9 Global Slope-sign . Given the affine function $AFFINE_{a,b}$,
- When a is *positive*, Slope-sign $AFFINE = \diagup$.
- When a is *negative*, Slope-sign $AFFINE = \diagdown$

8 Locating Inputs Whose Output $= y_0$

The simplest global problem is, given a number y_0, to ask for the input numbers for which the function returns the output y_0.

PROCEDURE 11.1 Find the input(s), if any, whose output under the function specified by

$$x \xrightarrow{AFFINE} AFFINE(x) = ax + b$$

Solve the equation $ax + b = y_0$ (See **??** on **??**.)

9 Locating Inputs Whose Output $> y_0$ Or $< y_0$

Given the affine function $AFFINE_{a,b}$, we are now ready to deal with the global problem of finding all inputs whose output is smaller (or larger) than some given number y_0.

EXAMPLE 11.5. Given the inequation problem in which
- the *data set* consists of all numbers
- the *inequation* is

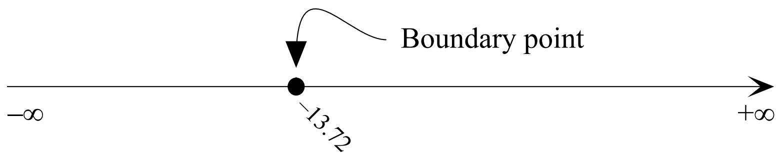
$$x \geqq -13.72$$

we locate separately.

i. The *boundary point* of the solution subset of the inequation problem is the solution of the *associated equation*:

$$x = -13.72$$

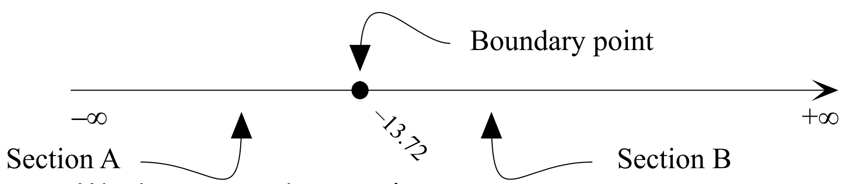
which, of course, is -13.72 and which we graph as follows since the boundary point is a *solution* of the inequation.



ii. The *interior* of the solution subset, that is the solution subset of the associated *strict inequation*

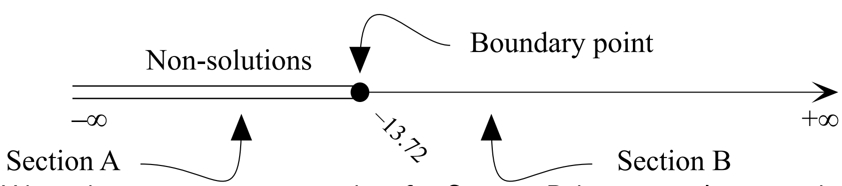
$$x > -13.72$$

i. The boundary point -13.72 separates the data set in two intervals, Section A and Section B:

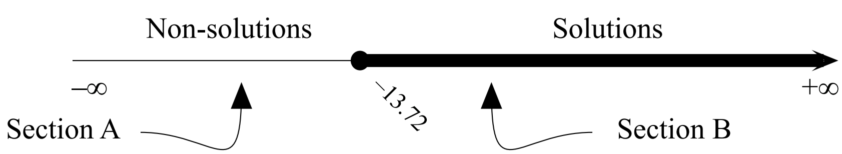


ii. We then test each interval:

- We pick $-1\,000$ as test number for Section A because, almost without a glance we know $-1\,000$ is going to be in Section A and because it is easy to check in the inequation: we find that $-1\,000$ is a *non-solution* so that, by **Pasch Theorem**, all numbers in Section A are *non-solutions*.



- We pick $+1\,000$ as test number for Section B because, almost without a glance we know $+1\,000$ is going to be in Section B and because it is easy to check in the inequation: we find that $+1\,000$ is a *solution* so that, by **Pasch Theorem**, all numbers in Section A are *solutions*.



10 Initial Value Problem

An **Initial Value Problem** asks the question:

What is the *input-output rule* of a function F given that:

- The function F is *affine*
- The *slope* of the function F is to be a given number a
- The *output* returned by the function F for a given input x_0 is to be a given number y_0.

EXAMPLE 11.6. Find the global input-output rule of the function $KATE$ given that it is affine, that its slope is -3 and that the output for the input $+2$ is $+5$.
We use all three given pieces of information:
i. Since we are given that $KATE$ is an affine function, we give temporary names for the dilation coefficient, say a, and for the constant term, say b, and we write the global input-output rule of $KATE$ in terms of these names:

$$x \xrightarrow{KATE_{a,b}} KATE_{a,b}(x) = ax + b$$

ii. By the **Local Slope Theorem**, the slope is equal to the dilation coefficient:

$$-3 = a$$

which give the equation $a = -3$
iii. Since the output for the input $+2$ is $+5$, we write

$$\begin{aligned} KATE_{a,b}(x)\big|_{x:=+2} &= +5 \\ ax + b\big|_{x:=+2} &= +5 \\ a(+2) + b &= +5 \end{aligned}$$

which give the equation $2a + b = +5$
iv. So we must solve the system of two equations for two unknowns a and b:
===========

$$\text{AND}\begin{cases} a = -3 \\ 2a + b = +5 \end{cases}$$

This kind of system is very simple to solve since we need only replace a by -3 in the second equation to get the equation:

$$2(-3) + b = +5$$

which we solve using the REDUCTION METHOD:

$$\begin{aligned} -6 + b &= +5 \\ -6 + b + 6 &= +5 + 6 \\ b &= +11 \end{aligned}$$

v. So, the global input-output rule for $KATE$ is

$$x \xrightarrow{KATE_{-3,+11}} KATE_{-3,+11}(x) = -3x + 11$$

11 Boundary Value Problem

A **Boundary Value Problem** asks the question:

What is the *input-output rule* of a function F, given that:

- The function F is *affine*
- The *output* returned by the function F for a given input x_1 is to be a given number y_1.
- The *output* returned by the function F for a given input x_2 is to be a given number y_2.

In other words, we want to find an affine function F such that:

$$\text{BOTH}\begin{cases} x_1 \xrightarrow{F} F(x_1) = y_1 \\ x_2 \xrightarrow{F} F(x_2) = y_2 \end{cases}$$

EXAMPLE 11.7. Find the global input-output rule of the function $DAVE$ given that it is affine, that the output for the input $+2$ is -1 and that the output for the input -4 is -19.

We use all three pieces of information that we are given:

i. Since we are given that $DAVE$ is an affine function, we give temporary names for the dilation coefficient, say a, and for the constant term, say b, and we write the global input-output rule of $DAVE$ in terms of these names:

$$x \xrightarrow{DAVE_{a,b}} DAVEa, b(x) = ax + b$$

ii. Since the output for the input $+2$ is -1 we write:

$$\begin{aligned} DAVE_{a,b}(x)\big|_{x:=+2} &= -1 \\ ax + b\big|_{x:=+2} &= -1 \\ a(+2) + b &= -1 \end{aligned}$$

which give the equation $+2a + b = -1$

iii. Since the output for the input -4 is -19 we write:

$$\begin{aligned} DAVE_{a,b}(x)\big|_{x:=-4} &= -19 \\ ax + b\big|_{x:=+2} &= -19 \\ a(-4) + b &= -19 \end{aligned}$$

which give the equation $-4a + b = -19$

iv. So we must solve the system of two equations for two unknowns a and b:

$$\begin{cases} +2a + b = -1 \\ -4a + b = -19 \end{cases}$$

This kind of system is a bit more complicated to solve but since b appears in both equations, we replace one of the two equations, say the second one, by "the first one minus the second one":

$$\begin{cases} +2a+b=-1 \\ [+2a+b]-[-4a+b]=[-1]-[-19] \end{cases}$$

This gives us:

$$\begin{cases} +2a+b=-1 \\ +2a+b+4a-b=-1+19 \end{cases}$$

that is

$$\begin{cases} +2a+b=-1 \\ +6a=+18 \end{cases}$$

that is

$$\begin{cases} +2a+b=-1 \\ \dfrac{+6a}{+6}=\dfrac{+18}{+6} \end{cases}$$

that is

$$\begin{cases} +2a+b=-1 \\ a=+3 \end{cases}$$

and now we replace in the first equation a by $+3$:

$$\begin{cases} +2a+b=-1\big|_{a:=+3} \\ a=+3 \end{cases}$$

that is

$$\begin{cases} +2(+3)+b=-1 \\ a=+3 \end{cases}$$

that is

$$\begin{cases} +6+b=-1 \\ a=+3 \end{cases}$$

and we reduce the first equation

$$\begin{cases} +6+b-6=-1-6 \\ a=+3 \end{cases}$$

which gives us, finally

$$\begin{cases} b=-7 \\ a=+3 \end{cases}$$

v. So the global input-output rule of $DAVE$ is

$$x \xrightarrow{DAVE_{+3,-7}} DAVE_{+3,-7}(x)=+3x-7$$

Quadratic␣function
quadratic␣coefficient
linear␣coefficient
constant␣coefficient

Chapter 12

Quadratic Functions: Local Analysis

Quadratic functions are specified by global input-output rules like the generic global input-output rule:

$$x \xrightarrow{QUADRATIC} QUADRATIC(x) = \underbrace{ax^{+2} \oplus bx^{+1} \oplus cx^{0}}_{\text{output-specifying code}}$$

which we usually write

$$= \underbrace{ax^2 + bx + c}_{\text{output-specifying code}}$$

where a, called the **quadratic coefficient**, b, called the **linear coefficient**, and c, called the **constant coefficient**, are the *bounded* numbers that specify the function $QUADRATIC$.

EXAMPLE 12.1. The quadratic function $RINA$ specified by the quadratic coefficient -23.04, the linear coefficient -17.39 and the constant coefficient $+5.84$ is the function specified by the global input-output rule

$$x \xrightarrow{RINA} RINA(x) = \underbrace{-23.04}_{\text{quadratic coeff.}} \; x^2 \underbrace{-17.39}_{\text{linear coeff.}} x \underbrace{+5.84}_{\text{constant coeff.}}$$

It is worth noting again that

term
quadratic term
linear term
constant term
affine␣part

NOTE 12.1 The terms in the global input output rule *need not* be written in order of *descending* exponent. This is just a habit we have.

EXAMPLE 12.2. The function specified by the global input-output rule

$$x \xrightarrow{BIBI} BIBI(x) = +21.03x^2 - 31.39x + 5.34$$

could equally well be specified by the global input-output rule

$$x \xrightarrow{BIBI} BIBI(x) = +5.34 + 21.03x^2 - 31.39x$$

or by the global input-output rule

$$x \xrightarrow{BIBI} BIBI(x) = -31.39x + 5.34 + 21.03x^2$$

We now introduce some standard terminology to help us describe very precisely what we will be doing. The output-specifying code of the affine function specified by

$$x \xrightarrow{AFFINE} QUADRATIC(x) = \underbrace{ax^2 + bx + c}_{\text{output-specifying code}}$$

consists of three **terms**:

- ax^2 which is called the **quadratic term**.
- bx which is called the **linear term**.
- c which is called the **constant term**,

and there is of course also

- $bx + c$ which is called the **affine part**

EXAMPLE 12.3. The output-specifying code of the function specified by the global input-output rule

$$x \xrightarrow{RINA} RINA(x) = \underbrace{-23.04}_{\text{quadratic coeff.}} x^2 \underbrace{-31.39}_{\text{linear coeff.}} x \underbrace{+5.84}_{\text{constant coeff.}}$$

consists of three terms:

$$= \underbrace{-23.04x^2}_{\text{quadratic term}} \underbrace{-31.39x}_{\text{linear term}} \underbrace{+5.34}_{\text{constant term}}$$

LANGUAGE 12.1 Whether we look upon c as the constant *coefficient*, that is as the *coefficient* of x^0 in the constant *term* cx^0 or as the constant *term* cx^0 itself with the power x^0 "going without saying" will be clear from the context.

1 Output *at* x_0

1. Remember from section 1 that x_0 is a *generic given input*, that is that x_0 is a *bounded* input that has been *given* but whose identity remains *undisclosed* for the time being.

2. We will use

PROCEDURE 12.1 To evaluate at x_0 the function specified by $x \xrightarrow{QUADRATIC} QUADRATIC(x) = ax^2 + bx + c$

i. *Declare* that x is to be replaced by x_0

$$x\Big|_{x \leftarrow x_0} \xrightarrow{QUADRATIC} QUADRATIC(x)\Big|_{x \leftarrow x_0} = ax^2 + bx + c\Big|_{x \leftarrow x_0}$$

which gives:

$$x_0 \xrightarrow{QUADRATIC} QUADRATIC(x_0) = \underbrace{a{x_0}^2 + bx_0 + c}_{\text{output-specifying code}}$$

ii. *Execute* the output-specifying code into an output *number*:

$$= ax_0^2 + bx_0 + c$$

which gives the input-output pair

$$\left(x_0, ax_0^2 + bx_0 + c\right)$$

DEMO 12.1 To evaluate at -3 the function specified by

$$x \xrightarrow{AVIA} AVIA(x) = +21.03x^2 - 32.67x + 71.07$$

i. We *declare* that x is to be replaced by -3

$$x\Big|_{x \leftarrow -3} \xrightarrow{AVIA} AVIA(x)\Big|_{x \leftarrow -3} = +21.03x^2 - 32.67x + 71.07\Big|_{x \leftarrow -3}$$

which gives

$$-3 \xrightarrow{AVIA} AVIA(-3) = \underbrace{+21.03(-3)^2 - 32.67(-3) + 71.07}_{\text{output specifying code}}$$

ii. We *execute* the output-specifying code into an output *number*:

$$= \underbrace{+189.26 \oplus +98.01 \oplus +71.07}_{\text{output number at } -3}$$

$$= \underbrace{+358.34}_{\text{output number at } -3}$$

which gives the input-output *pair*

$$(-3\ ,\ \underbrace{+358.34}_{\text{output number at } -3})$$

3. However, as already discussed in **?? ??** and as has already been the case with *monomial* functions and *affine* functions, instead of getting the output *number* returned by a quadratic function *at* a given input, we will usually want *all* the outputs returned by the quadratic function for inputs *near* that given input. So, instead of getting the single *input-output pair at* the given input, we will get the *local input-output rule* with which to get *all* the input-output pairs *near* the given input.

2 Output *near* ∞

As already discussed in **?? ??**, in order to input a neighborhood of ∞, we will *declare* that "x is near ∞" but write only x after that. This, again, is extremely dangerous as it is easy to forget that what we write may be TRUE *only* because x has been declared to be near ∞.

1. We will *execute* the output-specifying code, namely $ax^2 + bx + c$, into an *output jet*, that is with the terms in *descending* order of sizes, which, since here x is *large*, means that here the powers of x must be in *descending* order of exponents. We will then have the *local input-output rule near* ∞:

$$x \text{ near } \infty \xrightarrow{QUADRATIC} QUADRATIC(x) = \underbrace{\text{Powers of } x \text{ in } \textit{descending order of exponents}}_{\text{output jet near } \infty}$$

EXAMPLE 12.4. Given the function specified by the global input-output rule

$$x \xrightarrow{RIBA} RIBA(x) = -61.03 - 82.47x + 45.03x^2$$

To get the *output jet* near ∞, we first need to get the *order of sizes*.

i. -61.03 is $bounded$

ii. -82.47 is $bounded$ and x is $large$. So, since $bounded \cdot large = large$, $-82.47 \cdot x$ is $large$

iii. $+45.03$ is $bounded$ and x is $large$. So, since $bounded \cdot large = large$, $+45.03 \cdot x$ is $large$ too. But $large \cdot large$ is larger in size than $large$ so $+45.03 \cdot x^2$ is even larger than $-82.47 \cdot x$

So, in the output jet near ∞, $+45.03x^2$ must come first, $-82.47x$ comes second and -61.03 comes third

Then, we can write the local input-output rule near ∞:

$$x \text{ near } \infty \xrightarrow{RIBA} RIBA(x) = \underbrace{+45.03x^2 - 82.47x - 61.03}_{\text{output jet near } \infty}$$

2. So, we will use:

PROCEDURE 12.2 To evaluate near ∞ the function specified by $x \xrightarrow{QUADRATIC} QUADRATIC(x) = ax^2 + bx + c$

i. *Declare* that x is near ∞:

$$x\Big|_{x \text{ near } \infty} \xrightarrow{QUADRATIC} QUADRATIC(x)\Big|_{x \text{ near } \infty} = ax^2 + bx + c\Big|_{x \text{ near } \infty}$$

which gives:

$$x \text{ near } \infty \xrightarrow{QUADRATIC} QUADRATIC(x) = \underbrace{ax^2 + bx + c}_{\text{output-specifying code}}$$

ii. *Execute* the output-specifying code into an *output jet*:

$$= \underbrace{[a]\, x^2 \oplus [b]\, x \oplus [c]}_{\text{output jet near } \infty}$$

which gives the *local input-output rule* near ∞:

$$x \text{ near } \infty \xrightarrow{QUADRATIC} QUADRATIC(x) = \underbrace{[a]\, x^2 \oplus [b]\, x \oplus [c]}_{\text{output jet near } \infty}$$

(The *output jet* in the local input-output rule near ∞ looks the same as the *output-specifying code* in the given global input-output rule but that is only because *here* the output-specifying code *happened* to be written in *descending* order of exponents.)

DEMO 12.2 To evaluate near ∞ the function specified by

$$x \xrightarrow{KINA} KINA(x) = -61.03 + 51.32x^2 - 82.47x$$

i. We *declare* that x is near ∞:

$$x\Big|_{x \text{ near } \infty} \xrightarrow{KINA} KINA(x)\Big|_{x \text{ near } \infty} = -61.03 + 51.32x^2 - 82.47x\Big|_{x \text{ near } \infty}$$

which gives:

$$x \text{ near } \infty \xrightarrow{NINA} KINA(x) = \underbrace{-61.03 + 51.32\, x^2 - 82.47\, x}_{\text{output-specifying code}}$$

addition formula

ii. We *execute* the output-specifying code into an *output jet*:

$$= \left[+51.32\right] x^2 \oplus \left[-82.47\right] x \oplus \left[-61.03\right]$$

which gives the *local input-output rule* near ∞:

$$x \text{ near } \infty \xrightarrow{KINA} KINA(x) = \underbrace{\left[+51.32\right] x^2 \oplus \left[-82.47\right] x \oplus \left[-61.03\right]}_{\text{output jet near } \infty}$$

(The *output jet* in the local input-output rule near ∞ does *not* look the same as the *output-specifying code* in the *global* input-output rule because *here* the output-specifying code happened *not* to be in descending order of exponents.)

3. The reason we use *jets* here is that the term *largest in size* is the *first* term so that to *approximate* the output we need only write the *first* term in the jet and just replace the remaining terms by [...] which stands for "something too small to matter here". In other words,

Theorem 12.1 Approximate output near ∞. For *quadratic* functions, what contributes most to the output near ∞ is the *highest degree term* in the output jet near ∞:

$$x \text{ near } \infty \xrightarrow{QUADRATIC} QUADRATIC(x) = \left[a\right] x^2 + [...]$$

Example 12.5. Given the function specified by the global input-output rule

$$x \xrightarrow{KINA} KINA(x) = -61.03 + 51.32x^2 - 82.47x$$

near ∞ we will often just use the *approximation*

$$x \text{ near } \infty \xrightarrow{KINA} KINA(x) = \left[+51.32\right] x^2 \oplus [...]$$

3 Output *near* x_0

We now deal with the output of the neighborhood of some *given* bounded input x_0.

1. In order to input a neighborhood of a given input x_0 we will declare that $x \leftarrow x_0 \oplus h$ that is that x is to be replaced by $x_0 \oplus h$. As a result, we will have to compute $(x_0 \oplus h)^2$ for which we will have to use an **addition formula** from algebra, namely THEOREM ??? on page .

2. We can then *execute* the input-output specifying phrase into an *output jet* that is with the terms in *descending order of sizes* which here, since h

is *small*, means that the powers of h will have to be in *ascending* order of exponents. We will then have the *local input-output rule* near the given input: output jet near x_0

$$x_0 \oplus h \xrightarrow{QUADRATIC} QUADRATIC(x_0 \oplus h) = \underbrace{\text{Powers of } h \text{ in } \textit{ascending order of exponents}}_{\text{output jet near } \infty}$$

We will therefore use:

PROCEDURE 12.3 To evaluate near x_0 the function specified by $x \xrightarrow{QUADRATIC} QUADRATIC(x) = ax^2 + bx + c$

i. *Declare* that x is near x_0: (So x is to be replaced by $x_0 + h$.)

$$x\Big|_{x \leftarrow x_0+h} \xrightarrow{QUADRATIC} QUADRATIC(x)\Big|_{x \leftarrow x_0+h} = ax^2 + bx + c\Big|_{x \leftarrow x_0+h}$$

which gives:

$$x_0 + h \xrightarrow{QUADRATIC} QUADRATIC(x_0 + h) = \underbrace{a(x_0 + h)^2 + b(x_0 + h) + c}_{\text{output-specifying code}}$$

ii. *Execute* the output-specifying code into an *output jet*:

$$\begin{aligned}
&= a\left(x_0^2 + 2x_0 h + h^2\right) + b\,(x_0 + h) + c \\
&= ax_0^2 \oplus 2ax_0\, h \oplus a\, h^2 \\
&\oplus bx_0 \oplus b\, h \\
&\oplus c \\
&= \underbrace{\left[ax_0^2 + bx_0 + c\right] \oplus \left[2ax_0 + b\right] h \oplus \left[a\right] h^2}_{\text{output jet near } x_0}
\end{aligned}$$

which gives the *local input-output rule* near x_0:

$$x_0 + h \xrightarrow{QUADRATIC} QUADRATIC(x_0 + h) = \underbrace{\left[ax_0^2 + bx_0 + c\right] \oplus \left[2ax_0 + b\right] h \oplus \left[a\right] h^2}_{\text{output jet near } x_0}$$

DEMO 12.3 To evaluate near -3 the function specified by

$$x \xrightarrow{ARNA} ARNA(x) = -32.67x + 71.07 + 81.26x^2$$

i. We *declare* that x is near -3: (So x is to be replaced by $-3 + h$.)

$$x\Big|_{x \leftarrow -3+h} \xrightarrow{ARNA} ARNA(x)\Big|_{x \leftarrow -3+h} = -32.67x + 71.07 + 81.26x^2\Big|_{x \leftarrow -3+h}$$

which gives

$$-3+h \xrightarrow{ARNA} ARNA(-3+h) = \underbrace{-32.67(-3+h)+71.07+81.26(-3+h)^2}_{\text{output specifying code}}$$

ii. We *execute* the output-specifying code into an *output jet*:

$$
\begin{aligned}
&= -32.67(-3+h)+71.07+81.26\left((-3)^2+2(-3)h+h^2\right)\\
&= -32.67(-3)-32.67h\\
&\quad +71.07\\
&\quad +81.26(-3)^2+81.26(2)(-3)h+81.26h^2\\
&= +98.01 \oplus -32.67\,h\\
&\quad \oplus +71.07\\
&\quad \oplus +731.34 \oplus -487.56\,h \oplus +81.26\,h^2\\
&= \Big[+98.01+71.07+731.34\Big] \oplus \Big[-32.67-487.56\Big]h \oplus \Big[+81.26\Big]h^2\\
&= \underbrace{\Big[+900.42\Big] \oplus \Big[-519.63\Big]h \oplus \Big[+81.26\Big]h^2}_{\text{output jet near } -3}
\end{aligned}
$$

which gives the *local input-output rule* near -3:

$$-3+h \xrightarrow{ARNA} ARNA(-3+h) = \underbrace{\Big[+900.42\Big] \oplus \Big[-519.63\Big]h \oplus \Big[+81.26\Big]h^2}_{\text{output jet near } -3}$$

3. When all we want is a feature-sign, though, the above procedure is very inefficient and we will then use the following procedure based directly on the fact that a *quadratic function* is the addition of:

- a *square function*, (See **??** on **??**)
- a *linear function*, (See **??** on **??**.)
- a *constant function*. (See **??** on **??**.)

that is:

$$x \xrightarrow{QUADRATIC} QUADRATIC(x) = \underbrace{bx^2}_{\text{square}} \oplus \underbrace{cx}_{\text{linear}} \oplus \underbrace{d}_{\text{constant}}$$

We declare that x is near x_0 that is that x must be replaced by x_0+h:

$$x \xrightarrow{QUADRATIC} QUADRATIC(x) = \underbrace{b\,(x_0+h)^2}_{\text{square}} \oplus \underbrace{c\,(x_0+h)}_{\text{linear}} \oplus \underbrace{d}_{\text{constant}}$$

The output of the local input-output rule near x_0 will have to be a *jet*:

$$x_0+h \xrightarrow{QUADRATIC} QUADRATIC(x_0+h) = \Big[\quad\Big] \oplus \Big[\quad\Big]h \oplus \Big[\quad\Big]h^2$$

and we want to be able to get any one of the coefficients of the output jet without having to compute any of the other coefficients. So, what we will do is to get the contribution of each monomial function to the term we want. This requires us to have the *addition formula* at our finger tips:

a.

$$(x_0+h)^2 = x_0^2 + 2x_0h + h^2$$ (See **??** on page 403)

More precisely,

i. If we want the *coefficient* of h^0 in the output jet:

- The square function contributes bx_0^2
- The linear function contributes cx_0
- The constant function contributes d

so we have:

$$x_0+h \xrightarrow{QUADRATIC} QUADRATIC(x_0+h) = \Big[bx_0^2 + cx_0 + d\Big] \oplus \Big[\quad\Big]h \oplus \Big[\quad\Big]h^2$$

ii. If we want the *coefficient* of h^1 in the output jet:

- The square function contributes $2bx_0$
- The linear function contributes c
- The constant function contributes nothing

so we have:

$$x_0+h \xrightarrow{QUADRATIC} QUADRATIC(x_0+h) = \Big[\quad\Big] \oplus \Big[2bx_0 + c\Big]h \oplus \Big[\quad\Big]h^2$$

iii. If we want the *coefficient* of h^2 in the output jet:

- The square function contributes c
- The linear function contributes nothing
- The constant function contributes nothing

so we have:

$$x_0+h \xrightarrow{QUADRATIC} QUADRATIC(x_0+h) = \Big[\quad\Big] \oplus \Big[\quad\Big]h \oplus \Big[c\Big]h^2$$

4 Local graphs

Just the way we get the *plot point at* a given *bounded* input from the *output number at* that input, we get the *local graph near* any given input, be it *bounded* or *infinity*, from the *output jet near* that input.

PROCEDURE 12.4 To graph near ∞ the function specified by $x \xrightarrow{QUADRATIC} QUADRATIC(x) = ax^2 + bx + c$

1. Get the *local input-output rule* near ∞ using **??** on **??**:

$$x \text{ near } \infty \xrightarrow{QUADRATIC} QUADRATIC(x) = \underbrace{\left[a\right] x^2 \oplus \left[b\right] x \oplus \left[c\right]}_{\text{output jet near } \infty}$$

2. Get the *local graph* near ∞ of each term:

a. For the *quadratic term*, graph near ∞ the monomial function $x \to \left[a\right] x^2$ (See **??** on **??**.)

b. For the *linear term*, graph near ∞ the monomial function $x \to \left[b\right] x$ (See **??** on **??**.)

c. For the *constant term*, graph near ∞ the monomial function $x \to \left[c\right]$ (See **??** on **??**.)

3. Get the *local graph* near ∞ of *QUADRATIC* by adding to the local graph of the *quadratic term* the local graph of the *linear term* and the local graph of the *constant term*.(See chapter 9)

DEMO 12.4 To graph near ∞ the function specified by

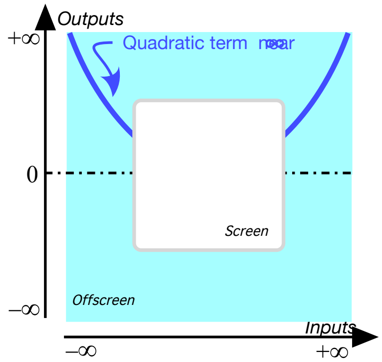
$$x \xrightarrow{KINA} KINA(x) = -61.03 + 51.32x^2 - 82.47x$$

1. We get the local *input-output rule* near ∞: (See Demo 12.2 on page 257)

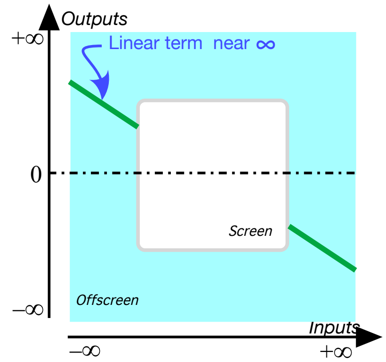
$$x \text{ near } \infty \xrightarrow{KINA} KINA(x) = \underbrace{\left[+51.32\right] x^2 + \left[-82.47\right] x + \left[-61.03\right]}_{\text{output jet near } \infty}$$

2. We get the *local graph* near ∞ of each term:

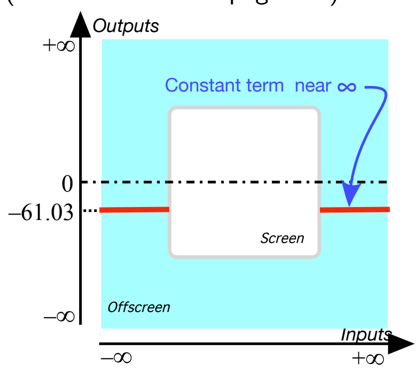
a. For the graph of the *quadratic* term, we graph the monomial function $x \to \left[\boxed{+51.32}\right] x^2$ near ∞
(See Demo 6.24 on page 175)



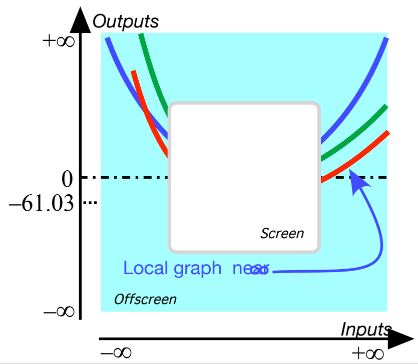
b. For the graph of the *linear* term, we graph the monomial function $x \to \left[\boxed{-82.47}\right] x$ near ∞
(See Demo 6.24 on page 175)



c. For the graph of the *constant* term, we graph the monomial function $x \to \left[\boxed{-61.03}\right]$ near ∞
(See Demo 6.24 on page 175)



3. We get the *local graph* near ∞ of $KINA$ by adding to the local graph of the *quadratic* term the local graph of the *linear* term and the local graph of the *constant* term. (See Demo 6.24 on page 175)



PROCEDURE 12.5 To graph near x_0 the function specified by $x \xrightarrow{QUADRATIC} QUADRATIC(x) = ax^2 + bx + c$

1. Get the *local input-output rule* near x_0 using **??** on **??**:

$$x_0 + h \xrightarrow{QUADRATIC} QUADRATIC(x_0 + h) = \underbrace{\left[ax_0^2 + bx_0 + c\right] \oplus \left[2ax_0 + b\right] h \oplus \left[a\right] h^2}_{\text{output jet near } x_0}$$

2. Get the *local graphs* near 0 of each term:

a. For the *constant term*, graph near 0 the monomial function $x \to \left[ax_0^2 + bx_0 + c\right]$. (See **??** on **??**.)

b. For the *linear term*, graph near 0 the monomial function $x \to \left[2ax_0 + b\right] x$. (See **??** on **??**.)

c. For the *quadratic term*, graph near 0 the monomial function $x \to \left[a\right] x^2$. (See **??** on **??**.)

3. Get the *local graph* near x_0 of $QUADRATIC$ by adding to the local graph of the *constant term* the local graph of the *linear term*, the local graph of the *quadratic term*.

DEMO 12.5 To graph near -3 the function specified by

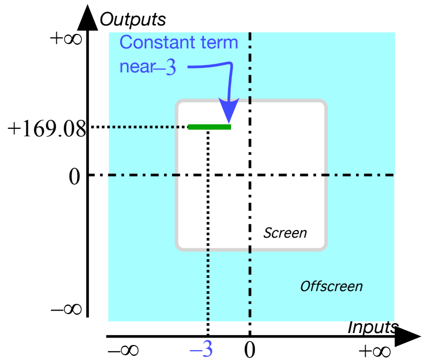
$$x \xrightarrow{ARNA} ARNA(x) = -32.67x + 71.07 + 81.26x^2$$

1. We get the local *input-output rule* near -3. (See Demo 12.3 on page 259):

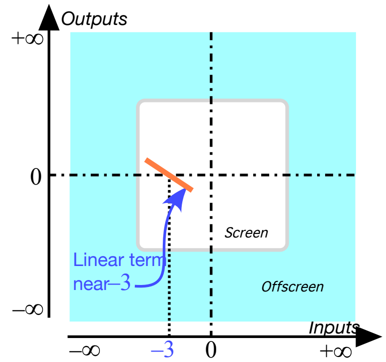
$$-3 + h \xrightarrow{ARNA} ARNA(-3 + h) = \underbrace{\left[+900.42\right] \oplus \left[-519.63\right] h \oplus \left[+81.26\right] h^2}_{\text{output jet near } -3}$$

2. We get the local graph near -3 of each term:

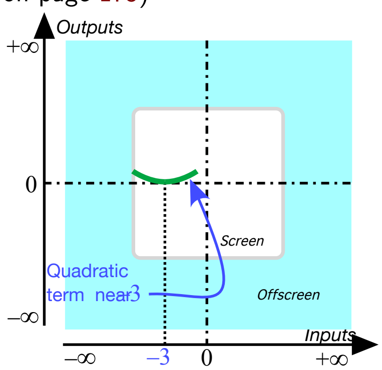
a. For the *constant* term, we graph near 0 the monomial function $x \to \left[\boxed{+900.428}\right]$. (See Demo 6.24 on page 175)



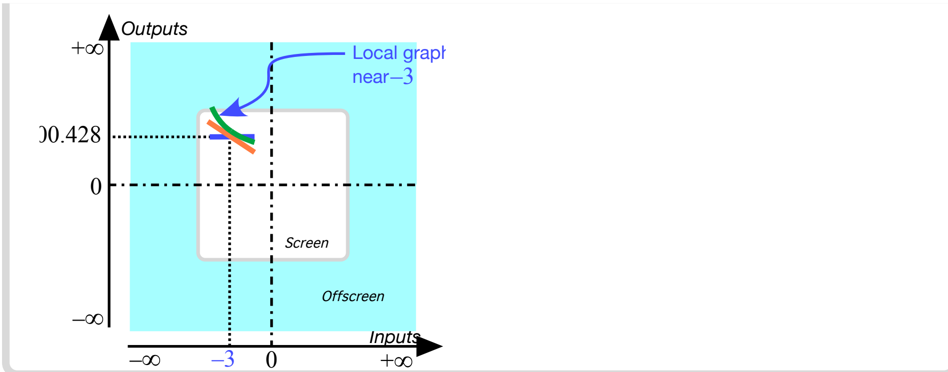
b. For the *linear* term, we graph near 0 the monomial function $x \to \left[\boxed{-519.63}\right]x$ (See Demo 6.24 on page 175)



c. For the *quadratic* term, we graph near 0 the monomial function $x \to \left[\boxed{+81.26}\right]x^2$: (See Demo 6.24 on page 175)



3. We get the local graph near -3 of $ARNA$ by adding to the local graph of the *constant term* the local graph of the *linear term* and the local graph of the *quadratic term*. (See Demo 6.24 on page 175)



5 Local Feature-signs

As we saw in **??** **??**, a feature-sign near a given input, be it near ∞ or near x_0, can be read from the *local graph* and so we already know how to proceed:

i. Get the *local input-output rule* near the given input (See **??** on **??** when the given input is ∞ or **??** on **??** when the given input is x_0.)

ii. Get the *local graph* from the local input-output rule (See **??** on **??**.)

iii. Get the *feature-sign* from the *local graph*. (See **??** **??**.)

However, things are in fact much simpler: Given an input, be it ∞ or a bounded input x_0, to get a required feature-sign near that given input, we look for the term in the output jet near that input that

i. Has the required feature.

and

ii. Is the largest-in-size of all those terms with the required feature.

So, as we will now see, we usually need to get only *one* term in the output jet rather than the whole output jet.

1. Near *infinity* things are quite straightforward because, for a quadratic function, the first term in the output jet near ∞ is both the *largest-in-size* and a *regular* monomial so that it has *all three features*:

PROCEDURE 12.6 To get the feature-signs near ∞ of the function specified by $x \xrightarrow{QUADRATIC} QUADRATIC(x) = ax^2 + bx + c$

i. Get the *approximate* local input-output rule near ∞:

$$x \text{ near } \infty \xrightarrow{QUADRATIC} QUADRATIC(x) = \underbrace{\big[a\big]x^2 \oplus \big[b\big]x \oplus \big[c\big]}_{\text{output jet near } \infty}$$
$$= \underbrace{\big[a\big]x^2 \oplus [...]}_{\text{approximate output jet near } \infty}$$

ii. Then, in the *approximate output jet* near ∞:

- Get the *Height-sign*, the *Slope-sign* and the *Concavity-sign* all from the *quadratic term* $\big[a\big]x^2$ because the next terms, $\big[b\big]x$ and $\big[c\big]$ are *too small to matter*. (Not to mention the fact that a linear term has no concavity and a constant term has neither concavity nor slope.)

TEMO 12.1 L et $CELIA$ be the function specified by

$$x \xrightarrow{CELIA} CELIA(x) = -2x^2 + 63x - 155$$

Get Height-sign near ∞.

i. We get the local input-output rule near ∞:

$$x \text{ near } \infty \xrightarrow{CELIA} CELIA(x) = -2x^2 + 63x - 155$$
$$= \underbrace{\Big[-2\Big]x^2 \oplus \Big[+63\Big]x \oplus \Big[-155\Big]}_{\text{output jet near } \infty}$$

ii. We get *Height-sign* from the *quadratic* term $\Big[-2\Big]x^2$ because the linear term $\Big[+63\Big]x$ and the *constant* term $\Big[-155\Big]$ are *too small to matter*.
iii. Since the *quadratic coefficient* $\Big[-2\Big]$ is *negative*, we get that Height-sign $CELIA$ near $\infty = \langle -, - \rangle$. (Seen from ∞.)

TEMO 12.2 L et $PETER$ be the function specified by the global input-output rule

$$x \xrightarrow{DIETER} DIETER(x) = +3.03x^2 - 81.67x + 46.92$$

Get Slope-signs near ∞.

critical for the Height
critical for the Slope

i. We get the local input-output rule near ∞:

$$x \text{ near } \infty \xrightarrow{DIETER} DIETER(x) = +3.03x^2 - 81.67x + 46.92$$
$$= \underbrace{\Big[+3.03\Big]x^2 \oplus \Big[-81.67\Big]x \oplus \Big[+46.92\Big]}_{\text{output jet near } \infty}$$

ii. We get *Slope-sign* from the *quadratic* term $\Big[+3.03\Big]x^2$ because the *linear* term $\Big[-81.67\Big]$ is *too small to matter* and the *constant* term has *no* slope. Since the *linear coefficient* $+3$ is positive, we get that Slope-sign $DIETER$ near $\infty = \langle \diagup, \diagup \rangle$. (Seen from ∞.)

2. Near a *bounded input* though, things are a bit more complicated:

i. The *first* term in the output jet is *usually* the *largest-in-size* so that it gives the Height-sign. However, the first term *usually* has neither Slope nor Concavity because the first term is *usually* a constant term.

ii. The *second* term in the output jet is *usually* too smalll-in-size to change the Height-sign as given by the first term but it is *usually* the *largest-in-size* term that can give the Slope-sign. However, the second term has no Concavity because the second term is *usually* a linear term.

iii. The third *term* in the output jet is *usually* too smalll-in-size to change the Height-sign given by the first term and the Slope-sign given by the second term but it is *usually* the *only term* that can give the Concavity-sign.

So we can *usually* read each feature-sign directly from the appropriate term in the output jet - keeping in mind that the exceptional monomial functions do not have all the features.

However, near a *bounded input*, the given bounded input may turn out to be *critical* for the local feature:

i. If the *constant term* in the output jet is 0, then the term which gives the Height-sign can be either the *linear term* or even the *quadratic term* if the *linear term* is 0. The bounded input is then said to be **critical for the Height**.

ii. If the *linear term* in the output jet is 0, then the term which gives the Slope-sign is the *quadratic term*. The bounded input is then said to be **critical for the Slope**.

So, we *usually* need to compute only one coefficient in the output jet. But if the given bounded input turns out to be *critical* for that feature, then we need to compute the next coefficient: So we use

PROCEDURE 12.7 To get the feature-signs near x_0 of the function specified by $x \xrightarrow{QUADRATIC} QUADRATIC(x) = ax^2 + bx + c$

i. Get the local input-output rule near x_0:

$$\begin{aligned} x_0 + h \xrightarrow{QUADRATIC} QUADRATIC(x_0+h) &= a(x_0+h)^2 + b(x_0+h) + c \\ &= a\left(x_0^2 + 2x_0h + h^2\right) + b\left(x_0+h\right) + c \\ &= \underbrace{\left[ax_0^2 + bx_0 + c\right] \oplus \left[2ax_0 + b\right]h \oplus \left[a\right]h^2}_{\text{output jet near } x_0} \end{aligned}$$

ii. Then, in the *output jet* near x_0:

- Get the *Height-sign* from the *constant term* $\left[ax_0^2 + bx_0 + c\right]$ (The *linear term* and the *quadratic term* are *too small to matter*.)
  If the *constant coefficient* is 0, get the *Height-sign* from the *linear term* $\left[2ax_0 + b\right]h$. (The quadratic term is *too small to matter*.)
  If the *linear coefficient* is 0, get the *Height-sign* from the *quadratic term* $\left[a\right]h^2$.
- Since the *constant term* has no slope, get the *Slope-sign* from the *linear term* $\left[2ax_0 + b\right]h$.
  If the *linear coefficient* is 0, get the *Slope-sign* from the *quadratic term* $\left[a\right]h^2$
- Since both the *constant term* and the *linear term* have no concavity, we get Concavity-sign from the *quadratic term*..

TEMO 12.3 L et $ARNA$ be the function specified by the global input-output rule

$$x \xrightarrow{ARNA} ARNA(x) = -32.67x + 71.07 + 81.26x^2$$

Get the feature-signs near -3.

i. We get the local input-output rule near -3 as in Demo 12.3 on page 259:

$$\begin{aligned} -3 + h \xrightarrow{ARNA} ARNA(-2+h) &= \underbrace{-32.67(-3+h) + 71.07 + 81.26(-3+h)^2}_{\text{output specifying code}} \\ &= \underbrace{\Big[+900.428\Big] \oplus \Big[-519.63\Big]h \oplus \Big[+81.26\Big]h^2}_{\text{output jet near } -3} \end{aligned}$$

ii. Then, from the *jet*:

- Since the *constant term* $\Big[+900.428\Big]$ is *positive*, we get that Height-sign $ARNA$ near $-3 = \langle +, + \rangle$.

- Since the *linear term* $\Big[-519.63\Big]h$ is *negative*. we get that Slope-sign $ARNA$ near $-3 = \langle\diagdown,\diagdown\rangle$
- Since the *quadratic term* $\Big[+81.26\Big]h^2$ is *positive*, we get that Concavity-sign $ARNA$ near $-3 = \langle\cup,\cup\rangle$

Chapter 13

Quadratic Functions: Global Analysis

=======Begin WORK ZONE=======

The "style" of this chapter is going to be very different from the "style" of the other chapters because we want to take the occasion to give the reader an idea of what happens when a research mathematician is facing a "new problem", that is a problem that no one else has solved before so that s/he cannot just look somewhere or ask someone "how to do it". So, in this chapter, instead of *showing* how to determine the global behavior of a quadratic function $x \xrightarrow{q} q(x) = ax^2 + bx + c$, we will pretend that this is a "research problem".

The first thing we do is to think about the problem itself: What do we mean by "global behavior"? Exactly *what* are we after? The idea is to see what a *precise* statement of the problem might suggest.

One answer might be that "we want to know everything there is to know about a quadratic function". But that is still much too vague to give us any hint as to what to do. Another answer might be "We want to see how the global graph of $x \xrightarrow{q} q(x) = ax^2 + bx + c$ looks?" This is already much better because it specifies the function we want to know about—even if the coefficients a, b, c remain to be specified later. But we really should say what we mean by "global graph", in particular what we want the global graph to show as opposed to what we don't expect the global graph to show.

On the other hand, we care about the global graph only inasmuch as it makes information "graphic" and it is really the information itself that we are after. So, what might this information be that we want? Exactly as with power functions, we will want to know about 0-feature inputs, namely:

- 0-height inputs,
- 0-slope inputs,
- 0-concavity inputs

and about feature-sign change inputs, namely

- height-sign change inputs,
- slope-sign change inputs,
- concavity-sign change inputs.

There still remains a question about what we want to know about these inputs. Do we want to know about:

- The *existence* or *non-existence* of these inputs,

or

- The *location* of these inputs—assuming they exist.

Let us say we want to know everything (But now, as opposed to before, we know exactly what "everything" covers.).

So, now that we know exactly what we want, what do we do to get it? First, though, let us review the equipment we have available:

-
-
-

=======End WORK ZONE=======

In the case of quadratic functions, we will still be able to solve *some* global problems *exactly* but since everything begins to be computationally more complicated, we will deal with only a few types of global problems.

1 The Essential Question

As usual, the first thing we do is to find out if the *offscreen graph* of a *quadratic function* consists of just the *local graph near* ∞ or if it also includes the *local graph near one or more* ∞*-height inputs*.

In other words, given the quadratic function $QUADRATIC_{a,b,c}$, that is the function specified by the global input-output rule

$$x \xrightarrow{QUADRATIC} QUADRATIC(x) = a^2x + bx + c$$

we ask the **Essential Question**:

- Do all *bounded inputs* have *bounded outputs*

or

- Are there *bounded inputs* that have ∞-height, that is are there inputs whose nearby inputs have *large* outputs?

Now, given a *bounded* input x, we have that:

- since a is bounded, ax^2 is also bounded
- since b is bounded, bx is also bounded
- c is bounded

and so, altogether, we have that $ax^2 + bx + c$ is bounded and that the answer to the **Essential Question** is:

THEOREM 13.1 Bounded Height Under a *quadratic* functions, all bounded inputs have *bounded outputs*.

and therefore that

THEOREM 13.2 Offscreen Graph The *offscreen graph* of a *quadratic* function consists of just the *local graph near* ∞.

EXISTENCE THEOREMS

The notable inputs are those

- whose existence is forced by the *offscreen graph* which, by the **Bounded Height Theorem** for quadratic functions, consists of only the *local graph near* ∞.
- whose number is limited by the interplay among the three features

Since polynomial functions have no *bounded* ∞-height input, the only way a feature can change sign is near an input where the feature is 0. Thus, with quadratic functions, the feature-change inputs will also be 0-feature inputs.

None of the theorems, though, will indicate *where* the notable inputs are. The **Location Theorems** will be dealt with in the last part of the chapter.

2 Concavity-sign

Given the quadratic function $QUADRATIC_{a,b,c}$, that is the function specified by the global input-output rule

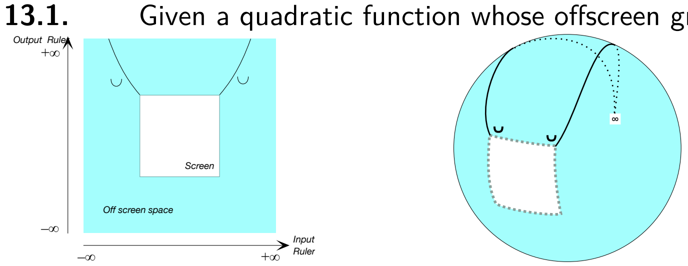
$$x \xrightarrow{QUADRATIC} QUADRATIC(x) = a^2x + bx + c$$

recall that when x is near ∞ the **Concavity-sign Near ∞ Theorem** for quadratic functions says that:

- When a is $+$, Concavity-Sign$|_{x \text{ near } \infty} = (\cup, \cup)$
- When a is $-$, Concavity-Sign$|_{x \text{ near } \infty} = (\cap, \cap)$

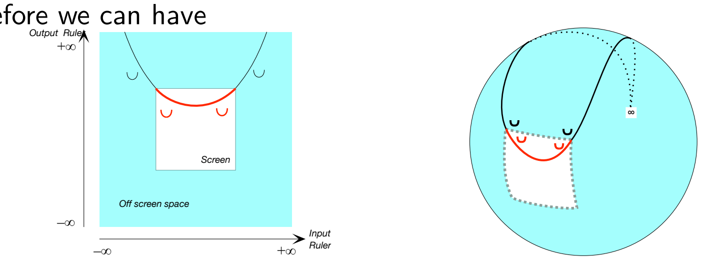
1. Since the concavity does *not* changes sign as x goes through ∞ from the left side of ∞ to the right side of ∞, the concavity does not have to change sign as x goes *across the screen* from the left side of ∞ to the right side of ∞ so there does not have to be a *bounded* Concavity-sign change input:

EXAMPLE 13.1. Given a quadratic function whose offscreen graph is



Mercator view Magellan view

there is no need for a bounded concavity-sign change input, $x_{\text{Concavity-sign change}}$ and therefore we can have



Mercator view Magellan view

2. In fact, not only does there not have to be a bounded concavity-sign change input, there *cannot* be a bounded concavity-sign change input since the *local* square coefficient is equal to the *global* square coefficient a and the concavity must therefore be the same everywhere:

global concavity

> **THEOREM 13.3 Concavity-sign Change Non-Existence** A quadratic function has no bounded *Concavity-sign change* input.

3. Another consequence of the fact that the local concavity does not depend on x_0, and is thus the same everywhere, is that it is a feature of the function $QUADRATIC_{a,b,c}$ itself and so that the function $QUADRATIC_{a,b,c}$ has a **global concavity** specified by the global square coefficient a.

4. Moreover, the concavity cannot be equal to 0 somewhere because the concavity is equal to a everywhere. So, we also have:

> **THEOREM 13.4 0-Concavity Input Non-Existence** A quadratic function has *no* bounded 0-*concavity* input.

3 Slope-sign

Given the quadratic function $QUADRATIC_{a,b,c}$, that is the function specified by the global input-output rule

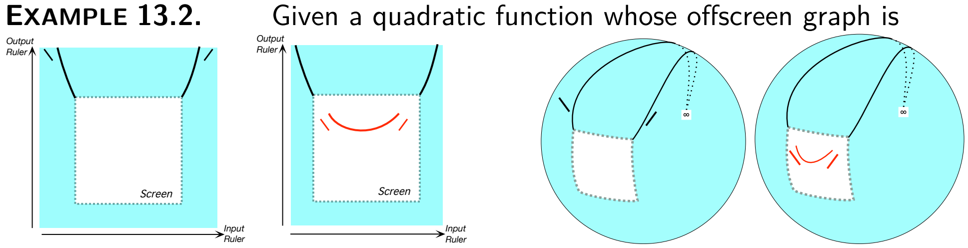
$$x \xrightarrow{QUADRATIC} QUADRATIC(x) = a^2x + bx + c$$

recall that when x is near ∞ the **Slope-sign Near ∞ Theorem** for quadratic functions says that:

- When a is $+$, Slope-Sign$|_{x \text{ near } \infty} = (\diagup, \diagdown)$
- When a is $-$, Slope-Sign$|_{x \text{ near } \infty} = (\diagdown, \diagup)$

1. Since the slope changes sign as x goes from the left side of ∞ to the right side of ∞ *across* ∞ , the slope has also to change sign as x goes from the left side of ∞ to the right side of ∞ *across the screen*. In other words, there has to be a *bounded* slope-sign change input.

EXAMPLE 13.2. Given a quadratic function whose offscreen graph is



Mercator view Magellan view

there has to be a *bounded* slope-sign change input to make up.

So we have

THEOREM 13.5 Slope-sign Change Existence A quadratic function must have at least one bounded Slope-sign change input.

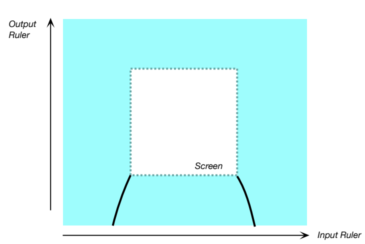
2. On the other hand, a quadratic function can have *at most one* 0-slope input because, if it had more, it would have to have 0-concavity inputs in-between the 0-slope inputs which a quadratic function cannot have. So we have

THEOREM 13.6 0-Slope Existence A quadratic function has exactly one slope-sign change input and it is a 0-slope input: $x_{\text{Slope-sign change}} = x_{\text{0-slope}}$

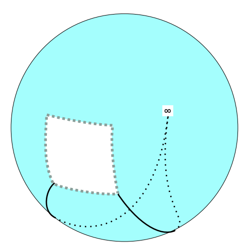
4 Extremum

From the *optimization* viewpoint, a quadratic function has an extreme input, that is an bounded input whose output is larger (or smaller) than the output of nearby inputs

EXAMPLE 13.3. Given a quadratic function whose offscreen graph is

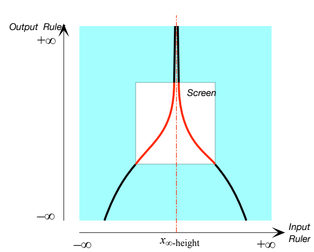


Mercator view

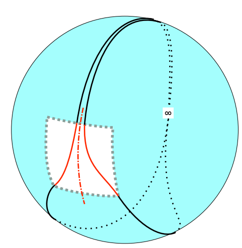


Magellan view

and since quadratic function cannot have an ∞-height input, we cannot have



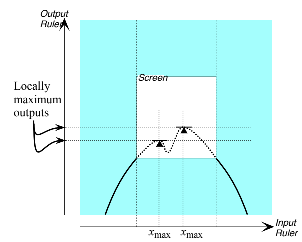
Mercator view



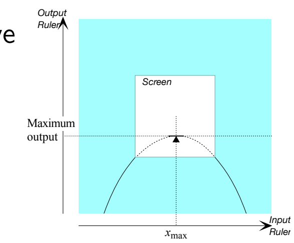
Magellan view

and therefore there has to be at least one input, x_{max}, whose output is *maximum*.

But since we cannot have a Concavity-sign change input as in



we must have



THEOREM 13.7 Extremum Existence A quadratic function has a single extremum input

5 0-Concavity Location

6 0-Slope Location

Given a quadratic function, the global problem of *locating* an input where the local slope is 0 is still fairly simple.

More precisely, given the quadratic function $QUADRATIC_{a,b,c}$, that is the function specified by the global input-output rule

$$x \xrightarrow{QUADRATIC} QUADRATIC(x) = ax^2 + bx + c$$

since the *slope* near x_0 is the local linear coefficient $2ax_0 + b$, in order to find the input(s) where the local slope is 0, we just need to solve the equation

$$2ax + b = 0$$

which is an affine equation that we solve by reducing it to a basic equation:

$$\begin{aligned} 2ax + b - b &= 0 \text{ -b} \\ 2ax &= -b \\ \frac{2ax}{2a} &= \frac{-b}{2a} \\ x &= \frac{-b}{2a} \end{aligned}$$

So, we have:

THEOREM 13.8 0-slope Location For any quadratic function $QUADRATIC_{a,b,c}$,

$$x_{0-slope} = \frac{-b}{2a}$$

In fact, we also have:

THEOREM 13.9 Global Slope-sign Given a quadratic function $QUADRATIC_{a,b,c}$,

- When a is positive,

$$\begin{aligned}
\text{Slope-sign } QUADRATIC|_{\text{Everywhere}<\frac{-b}{2a}} &= (\searrow, \searrow) \\
\text{Slope-sign } QUADRATIC|_{\frac{-b}{2a}} &= (\searrow, \nearrow) \\
\text{Slope-sign } QUADRATIC|_{\text{Everywhere}>\frac{-b}{2a}} &= (\nearrow, \nearrow)
\end{aligned}$$

- When a is negative,

$$\begin{aligned}
\text{Slope-sign } QUADRATIC|_{\text{Everywhere}<\frac{-b}{2a}} &= (\nearrow, \nearrow) \\
\text{Slope-sign } QUADRATIC|_{\frac{-b}{2a}} &= (\nearrow, \searrow) \\
\text{Slope-sign } QUADRATIC|_{\text{Everywhere}>\frac{-b}{2a}} &= (\searrow, \searrow)
\end{aligned}$$

The case is easily made by testing the corresponding inequations near ∞.

7 Extremum Location

From the **Extremum Existence Theorem**, we know that

$$x_{\text{extremum}} = x_{\text{0-slope}}$$

and so we have that

$$x_{\text{extremum}} = \frac{-b}{2a}$$

We now want to compute the extremum *output* which is the output for $x_{\text{0-slope}}$:

$$\begin{aligned}
QUADRATIC(x_{\text{0-slope}}) &= ax_{\text{0-slope}}^2 + bx_{\text{0-slope}} + c \\
&= a\left(\frac{-b}{2a}\right)^2 + b\left(\frac{-b}{2a}\right) + c \\
&= a\left(\frac{(-b)^2}{(2a)^2}\right) + b\left(\frac{-b}{2a}\right) + c
\end{aligned}$$

quadratic equation

$$
\begin{aligned}
&= a\left(\frac{b^2}{4a^2}\right) + b\left(\frac{-b}{2a}\right) + c \\
&= \frac{ab^2}{4a^2} + b\left(\frac{-b}{2a}\right) + c \\
&= \frac{\cancel{a}b^2}{4\cancel{a}a} + b\left(\frac{-b}{2a}\right) + c \\
&= \frac{b^2}{4a} + b\left(\frac{-b}{2a}\right) + c \\
&= \frac{b^2}{4a} + \frac{-2\cdot b^2}{2\cdot 2a} + c \\
&= \frac{b^2}{4a} + \frac{-2b^2}{4a} + c \\
&= \frac{-b^2}{4a} + c \\
&= \frac{-b^2}{4a} + \frac{4a\cdot c}{4a} \\
&= \frac{-b^2 + 4ac}{4a}
\end{aligned}
$$

8 0-Height Location

Given a quadratic function, the global problem of *locating* a given local height is the problem of locating the input(s), if any, whose output is equal to the given height.

More precisely, given the quadratic function $QUADRATIC_{a,b,c}$, that is the function specified by the global input-output rule

$$x \xrightarrow{QUADRATIC} QUADRATIC(x) = ax^2 + bx + c$$

and given the local height H_0, what we are looking for are the input(s), if any, whose output is equal to H_0, that is:

$$x \xrightarrow{QUADRATIC} QUADRATIC(x) = H_0$$

In other words, we must solve the equation

$$ax^2 + bx + c = H_0$$

This is called a **quadratic equation**. Since we are looking for the 0-height inputs, we let H_0 be 0 and we will want to solve the equation

$$ax^2 + bx + c = 0$$

1. Solving a quadratic equation is quite a bit more complicated than solving an affine equation because we cannot reduce a quadratic equation to a basic equation the way we reduce an affine equation to a basic equation.
The reason is that affine equations have *two* terms and the = sign has *two* sides so that we could *separate* the terms by having an x-term on the left side of the = sign and a constant term on the right side of the = sign which gave us a basic equation.
However, we cannot *separate* the terms in a quadratic equation because the output $QUADRATIC(x)$ has *three* terms while the = sign has only *two* sides.
This, though, may have something to do with the fact that inputs are counted from the 0 on the ruler which can be anywhere in relation to the *global graph* of the function, rather than from an input which is meaningful for the global graph of that function.
What we will do then is to try to use, instead of the inputs themselves, the *location* of the inputs relative to an input that is meaningful for the function at hand and the obvious thing is to try is $x_{\text{0-slope}}$ and so we will try to use:

$$u = x - x_{\text{0-slope}}$$

so that

$$x = x_{\text{0-slope}} + u$$

and therefore, instead of using the global input-output rule

$$x \xrightarrow{QUADRATIC} QUADRATIC(x) = ax^2 + bx + c$$

we will use the global input-ouput rule

$$x|_{x \leftarrow x_{\text{0-slope}}+u} \xrightarrow{QUADRATIC} QUADRATIC(x)\ |_{x \leftarrow x_{\text{0-slope}}+u} = \left. ax^2 + bx + c \right|_{x \leftarrow x_{\text{0-slope}}+u}$$

that is

$$u \xrightarrow{QUADRATIC_{(x_{\text{0-slope}})}} QUADRATIC(x_{\text{0-slope}} + u)$$
$$= \Big[a\Big]u^2 + \Big[2ax_{\text{0-slope}} + b\Big]u + \Big[ax^2_{\text{0-slope}} + bx_{\text{0-slope}} + c\Big]$$

By the way, note that we will continue to count the *outputs* from the 0 on the output ruler. (Some people don't and prefer to count the outputs from $QUADRATIC(x_{\text{0-slope}})$.)

2. But since $x_{\text{0-slope}} = \frac{-b}{2a}$, this reduces to

$$u \xrightarrow{QUADRATIC_{(x_{\text{0-slope}})}} QUADRATIC(x_{\text{0-slope}} + u)$$
$$= \Big[a\Big]u^2 + \Big[0\Big]u + \Big[ax^2_{\text{0-slope}} + bx_{\text{0-slope}} + c\Big]$$

that is to only two terms

$$= \Big[a\Big]u^2 + \Big[ax^2_{\text{0-slope}} + bx_{\text{0-slope}} + c\Big]$$

and the equation we want to solve, then, is

$$\Big[a\Big]u^2 + \Big[ax^2_{\text{0-slope}} + bx_{\text{0-slope}} + c\Big] = H_0$$

that is

$$\Big[a\Big]u^2 = H_0 - \Big[ax^2_{\text{0-slope}} + bx_{\text{0-slope}} + c\Big]$$

that is

$$u^2 = \frac{H_0 - \Big[ax^2_{\text{0-slope}} + bx_{\text{0-slope}} + c\Big]}{a}$$

in which everything on the right-hand side is known so that we have *separated* the known from the unknown.

3. Since we are trying to locate the 0-height inputs, we let $H_0 = 0$. In that case, the equation reduces to

$$\begin{aligned} u^2 &= \frac{-\Big[ax^2_{\text{0-slope}} + bx_{\text{0-slope}} + c\Big]}{a} \\ &= \frac{-QUADRATIC(x_{\text{extremum}})}{a} \end{aligned}$$

and, using the Extremum Location Theorem,

$$\begin{aligned} &= \frac{-\frac{-\text{Discriminant}_{QUADRATIC}}{4a}}{a} \\ &= \frac{\text{Discriminant}_{QUADRATIC}}{4a^2} \end{aligned}$$

Altogether then, instead of the original equation

$$ax^2 + bx + c = 0$$

we have the rather nice (nicer?) equation

$$\boxed{u^2 = \frac{\text{Discriminant}_{QUADRATIC}}{4a^2}}$$

4. Now, of course, whether or not we can solve depends on whether or not the right hand side is positive and since the denominator is a square, and therefore always positive, whether or not we can solve depends only on the sign of $\text{Disc}_{QUADRATIC}$ (hence the name "discriminant"):

- If $\text{Disc}_{QUADRATIC}$ is *negative*, the equation has *no* solution,
- If $\text{Disc}_{QUADRATIC}$ is 0, the equation has *one* solution, namely 0,
- If $\text{Disc}_{QUADRATIC}$ is *positive*, the equation has *two* solutions, namely
  - $u = -\frac{\sqrt{\text{Disc}_{QUADRATIC}}}{2a}$
  - $u = +\frac{\sqrt{\text{Disc}_{QUADRATIC}}}{2a}$

de-locate

This, of course, is hardly surprising inasmuch as the discriminant is intimately tied with the extremum output and thus this theorem fits very well with the 0**-height Existence Theorem**.

5. It remains only to **de-locate**, that is to return to the input x. For that, we need only use the fact that

$$u = x - x_{\text{0-slope}}$$

to get

- $x - x_{\text{0-slope}} = -\frac{\sqrt{\text{Disc}_{QUADRATIC}}}{2a}$
- $x - x_{\text{0-slope}} = +\frac{\sqrt{\text{Disc}_{QUADRATIC}}}{2a}$

that is

- $x = x_{\text{0-slope}} - \frac{\sqrt{\text{Disc}_{QUADRATIC}}}{2a}$
- $x = x_{\text{0-slope}} + \frac{\sqrt{\text{Disc}_{QUADRATIC}}}{2a}$

and thus the celebrated "quadratic formula":

- $x = x_{\text{0-slope}} - \frac{\sqrt{b^2-4ac}}{2a}$
- $x = x_{\text{0-slope}} + \frac{\sqrt{b^2-4ac}}{2a}$

which, by the way, shows that, when they exist, the two 0-height inputs are symmetrical with respect to $x_{\text{0-slope}}$

6. Altogether, then, we have

THEOREM 13.10 0-height Location For any quadratic function $QUADRATIC_{a,b,d}$,

- ▶ If $\text{Disc}_{QUADRATIC}$ is *negative*, $QUADRATIC$ has *no* 0-height input,
- ▶ If $\text{Disc}_{QUADRATIC}$ is 0, $QUADRATIC$ has *one* 0-height input, namely $\frac{-b}{2a}$,
- ▶ If $\text{Disc}_{QUADRATIC}$ is *positive*, $QUADRATIC$ has *two* solutions, namely
  - $\frac{-b}{2a} - \frac{\sqrt{b^2-4ac}}{2a}$
  - $\frac{-b}{2a} + \frac{\sqrt{b^2-4ac}}{2a}$

7. Finally, here are a couple of examples.

EXAMPLE 13.4. To find the 0-height inputs of the quadratic function specified by the global input-output rule

$$x \xrightarrow{Rick} Rick(x) = +4x^2 - 24x + 7$$

we can proceed as follows:

i. Either we remember that $x_{0-slope} = \frac{-b}{2a}$ so that we get $x_{0-slope} = \frac{+12}{2(+4)} = +3$, or, if worse comes to worst, we look for the 0-slope input by localizing at an undisclosed input x_0 and then setting the coefficient of u equal to 0 to get $x_{0-slope}$.

ii. Then, we get the u-equation by setting $x = x_{0-slope} + u$, that is, here, by setting $x = +3 + u$:

$$\begin{aligned} +3 + u \xrightarrow{Rick} Rick(x)|_{\text{when } x=+3+u} &= +4x^2 - 24x + 7\Big|_{\text{when } x=+3+u} \\ &= +4\,[+3+u]^2 - 24\,[+3+u] + 7 \\ &= +4\left[+9 + 6u + u^2\right] - 24\,[+3+u] + 7 \\ &= +36 + 24u + 4u^2 - 72 - 24u + 7 \\ &= -29 + 4u^2 \end{aligned}$$

iii. We now solve the u-equation

$$\begin{aligned} -29 + 4u^2 &= 0 \\ +4u^2 &= +29 \\ u^2 &= \frac{+29}{+4} \\ u^2 &= +7.25 \end{aligned}$$

and so we have:

- ▶ $u_{0-output} = +\sqrt{+7.25} = +2.69 + [...]$

  and

- ▶ $u_{0-output} = -\sqrt{+7.25} = -2.69 + [...]$

and therefore

- ▶ $x_{0-output} = +3 + 2.693 + [...] = +5.693 + [...]$

  and

- ▶ $x_{0-output} = +3 - 2.693 + [...] = +0.307 + [...]$

Alternatively, if we remember the **0-height Theorem**, then we can proceed by first computing the discriminant and

EXAMPLE 13.5. We look at the same equation but assume that we remember the **0-height Theorem**

$$x \xrightarrow{Rick} Rick(x) = +4x^2 - 24x + 7$$

that is:

$$\begin{aligned}\text{Discriminant } Rick &= (-24)^2 - 4(+4)(+7)\\ &= +576 - 112\\ &= +464\end{aligned}$$

And since the discriminant is positive, we have

$$\begin{aligned}x_{0-output} &= x_{0-slope} + \frac{\sqrt{\text{Discriminant}}}{2a}\\ &= \frac{+24}{2(+4)} + \frac{\sqrt{+464}}{2(+4)}\\ &= \frac{+24}{+8} + \frac{21.541 + [...]}{+8}\\ &= \frac{45.541 + [...]}{+8}\\ &= +5.693 + [...]\end{aligned}$$

and similarly

$$\begin{aligned}x_{0-output} &= x_{0-slope} - \frac{\sqrt{\text{Discriminant}}}{2a}\\ &= \frac{+24}{2(+4)} - \frac{\sqrt{+464}}{2(+4)}\\ &= \frac{+24}{+8} - \frac{21.541 + [...]}{+8}\\ &= \frac{2.460 + [...]}{+8}\\ &= +0.307 + [...]\end{aligned}$$

Either way, the reader should check that, indeed,

$$+5.693 \xrightarrow{Rick} 0 + [...]$$

and

$$+0.307 \xrightarrow{Rick} 0 + [...]$$

8. As a consequence of the 0**-height Location Theorem**, we have:

THEOREM 13.11 Global Height-sign For any quadratic function $QUADRATIC_{a,b,c}$, *Height-sign* $QUADRATIC = (Sign\ a, Sign\ a)$ everywhere except, when $Disc_{QUADRATIC}$ is *positive*, between the two $x_{0\text{-}height}$ inputs where *Height-sign* $QUADRATIC = (-Sign\ a, -Sign\ a)$

As a result, when looking for the inputs for which the output has a given sign, we have two approaches:
i. We can solve the associate equation, one way or the other, and then test each one of the sections determined by the 0-height input(s), if any.

EXAMPLE 13.6. To solve the *inequation* $-3x^2 + tx - 11 < 0$, we can begin by looking for its *boundary inputs* by solving the *associated equation* $-3x^2 + tx - 11 = 0$ and then test the resulting intervals.

ii. We can use the **Global Height-sign Theorem**.

=========OK SO FAR=========

(1) The difficulty is that there are two cases to deal with:

when a>0, concavity is u, the graph bottoms out and so there is a smallest bounded output when a<0, concavity is n, the graph culminates and so there is a largest bounded output

and that we want to cover them both in one single statement.

So, we use the term "extreme bounded output" to cover both cases and we can now say that the extreme bounded output is the output for $x_0 - slope$. (regardless of the sign of a.)

(2) Yesterday, using qualitative global graphs, we agreed that: if the sign of the extreme bounded output is the same as the height-sign near infinity, there can be no 0-height input. if the sign of the extreme bounded output is the opposite of the height-sign near infinity, there will be two 0-height inputs.

But the height-sign near infinity is the sign of the coefficient a so this becomes: if the sign of the extreme bounded output is the same as the sign of the coefficient a, there can be no 0-height input. if the sign of the extreme bounded output is the opposite of the sign of the coefficient a, there will be two 0-height inputs.

We also found that the extreme output is the output for $x_0 - slope =$ -b/2a and we computed that the extreme output is equal to $[-b^2 + 4ac]/4a$

As I recall, this is where we left off.

(3) Since the number $b^2 - 4ac$ is what is called the discriminant of the function, we have that the extreme output = -Discriminant/4a

And now we are ready for the kill. The weapon will be that: Two numbers have the same sign if they multiply to + Two numbers have opposite signs if they multiply to -

(4) So our agreement above can now be restated as: if the extreme bounded output and the coefficient a multiply to +, there can be no 0-height input. if the extreme bounded output and the coefficient a multiply to -, there will be two 0-height inputs.

that is: if Sign of -Discriminant/4a * a = +, there can be no 0-height input. if Sign of -Discriminant/4a * a = -, there will be two 0-height inputs.

that is, after canceling the coefficient a if Sign of -Discriminant = +, there can be no 0-height input. if Sign of -Discriminant = -, there will be two 0-height inputs.

that is, since Sign of -Discriminant is the opposite of Sign of Discriminant if Sign of Discriminant = -, there can be no 0-height input. if Sign of Discriminant = +, there will be two 0-height inputs.

Which is the qualitative part of the 0-height Theorem FOR QUADRATIC FUNCTIONS.

(The quantitative part of the 0-height Theorem FOR QUADRATIC FUNCTIONS is that the 0-height inputs—when they exist—are at a distance of Ã$Disc/2afromx_0 - slope.$)

Quadratic␣function
cubic␣coefficient
quadratic␣coefficient
linear␣coefficient
constant␣coefficient

Chapter 14

Cubic Functions: Local Analysis

Quadratic functions are specified by global input-output rules like the generic global input-output rule:

$$x \xrightarrow{CUBIC} CUBIC(x) = \underbrace{ax^{+3} \oplus bx^{+2} \oplus cx^{+1} \oplus dx^{0}}_{\text{output-specifying code}}$$

which we usually write

$$= \underbrace{ax^3 + bx^2 + cx + d}_{\text{output-specifying code}}$$

where a, called the **cubic coefficient**, b, called the **quadratic coefficient**, c, called the **linear coefficient**, and d, called the **constant coefficient**, are the *bounded* numbers that specify the function $CUBIC$.

EXAMPLE 14.1. The cubic function $TINA$ specified by the cubic coefficient $+72.55$, the quadratic coefficient -23.04, the linear coefficient -17.39 and the constant coefficient $+5.84$ is the function specified by the global input-output rule

$$x \xrightarrow{RINA} TINA(x) = \underbrace{-72.55}_{\text{cubic coeff.}}\, x^3 \quad \underbrace{-23.04}_{\text{quadratic coeff.}}\, x^2 \quad \underbrace{-17.39}_{\text{linear coeff.}}\, x \quad \underbrace{+5.84}_{\text{constant coeff.}}$$

It is worth noting again that

term
cubic term
quadratic term
linear term
constant term
quadratic␣part

NOTE 14.1 The terms in the global input output rule *need not* be written in order of *descending* exponent. This is just a habit we have.

EXAMPLE 14.2. The function specified by the global input-output rule

$$x \xrightarrow{DIDI} DIDI(x) = -12.06x^3 + 21.03x^2 - 31.39x + 5.34$$

could equally well be specified by the global input-output rule

$$x \xrightarrow{DIDI} DIDI(x) = +5.34 + 21.03x^2 - 31.39x - 12.06x^3$$

or by the global input-output rule

$$x \xrightarrow{DIDI} DIDI(x) = -31.39x + 5.34 - 12.06x^3 + 21.03x^2$$

We now introduce some standard terminology to help us describe very precisely what we will be doing. The output-specifying code of the affine function specified by

$$x \xrightarrow{AFFINE} CUBIC(x) = \underbrace{ax^3 + bx^2 + cx + d}_{\text{output-specifying code}}$$

consists of four **terms**:

- ax^3 which is called the **cubic term**.
- bx^2 which is called the **quadratic term**.
- cx which is called the **linear term**.
- d which is called the **constant term**,

and there is of course also

- $bx^2 + cx + d$ which is called the **quadratic part**

EXAMPLE 14.3. The output-specifying code of the function specified by the global input-output rule

$$x \xrightarrow{TINA} TINA(x) = \underbrace{-71.41}_{\text{cubic coeff.}} x^3 \quad \underbrace{-23.04}_{\text{quadratic coeff.}} x^2 \quad \underbrace{-31.39}_{\text{linear coeff.}} x \quad \underbrace{+5.84}_{\text{constant coeff.}}$$

consists of four terms:

$$= \underbrace{-71.41x^3}_{\text{cubic term}} \quad \underbrace{-23.04x^2}_{\text{quadratic term}} \quad \underbrace{-31.39x}_{\text{linear term}} \quad \underbrace{+5.34}_{\text{constant term}}$$

LANGUAGE 14.1 Whether we look upon d as the constant *coefficient*, that is as the *coefficient* of x^0 in the constant *term* dx^0 or as the constant *term* dx^0 itself with the power x^0 "going without saying" will be clear from the context.

1 Output *at* x_0

Remember from section 1 that x_0 is a *generic given input*, that is that x_0 is a *bounded* input that has been *given* but whose identity remains *undisclosed* for the time being.

PROCEDURE 14.1 To evaluate at x_0 the function specified by $x \xrightarrow{CUBIC} CUBIC(x) = ax^3 + bx^2 + cx + d$

i. *Declare* that x is to be replaced by x_0

$$x \Big|_{x \leftarrow x_0} \xrightarrow{CUBIC} CUBIC(x) \Big|_{x \leftarrow x_0} = ax^3 + bx^2 + cx + d \Big|_{x \leftarrow x_0}$$

which gives:

$$x_0 \xrightarrow{CUBIC} CUBIC(x_0) = \underbrace{a{x_0}^3 + b{x_0}^2 + c x_0 + d}_{\text{output-specifying code}}$$

ii. *Execute* the output-specifying code into an output *number*:

$$= ax_0^3 + bx_0^2 + cx_0 + d$$

which gives the input-output pair

$$\left(x_0, ax_0^3 + bx_0^2 + cx_0 + d\right)$$

DEMO 14.1 To evaluate at -3 the function specified by

$$x \xrightarrow{ARIA} ARIA(x) = +17.52x^3 + 21.03x^2 - 32.67x + 71.07$$

i. We *declare* that x is to be replaced by -3

$$x \Big|_{x \leftarrow -3} \xrightarrow{ARIA} ARIA(x) \Big|_{x \leftarrow -3} = +17.52x^3 + 21.03x^2 - 32.67x + 71.07 \Big|_{x \leftarrow -3}$$

which gives

$$-3 \xrightarrow{ARIA} ARIA(-3) = \underbrace{+17.52(-3)^3 + 21.03(-3)^2 - 32.67(-3) + 71.07}_{\text{output specifying code}}$$

ii. We *execute* the output-specifying code into an output *number*:

$$\begin{aligned} &= -473.04 \oplus +189.26 \oplus +98.01 \oplus +71.07 \\ &= -114.7 \end{aligned}$$

which gives the *input-output pair*

$$(-3, -114.7)$$

However, as already discussed in **?? ??** and as has already been the case with *monomial* functions, *affine* functions and *quadratic* functions, instead of getting the output *number* returned by a quadratic function *at* a given input, we will usually want *all* the outputs returned by the quadratic function for inputs *near* that given input. So, instead of getting the single *input-output pair at* the given input, we will get the *local input-output rule* with which to get *all* the input-output pairs *near* the given input.

2 Output *near* ∞

As already discussed in **?? ??** and in section 2 Output *near* ∞, in order to input a neighborhood of ∞, we will *declare* that "x is near ∞" but write only x after that. This, again, is extremely dangerous as it is easy to forget that what we write may be TRUE *only* because x has been declared to be near ∞.

1. We will *execute* the output-specifying code, namely ax^3+bx^2+cx+d, into an *output jet*, that is with the terms in *descending* order of sizes, which, since here x is *large*, means that here the powers of x must be in *descending* order of exponents. We will then have the *local input-output rule near* ∞:

$$x \text{ near } \infty \xrightarrow{CUBIC} CUBIC(x) = \underbrace{\text{Powers of } x \text{ in } \textit{descending order of exponents}}_{\text{output jet near } \infty}$$

EXAMPLE 14.4. Given the function specified by the global input-output rule

$$x \xrightarrow{TIBA} TIBA(x) = -61.03 + 37.81x^3 - 82.47x + 45.03x^2$$

To get the output jet near ∞, we first need to get the *order of sizes*.

i. -61.03 is *bounded*

ii. -82.47 is *bounded* and x is *large*. So, since *bounded* $\cdot$ *large* = *large*, $-82.47 \cdot x$ is *large*

iii. $+45.03$ is *bounded* and x is *large*. So, since *bounded* $\cdot$ *large* = *large*, $+45.03 \cdot x$ is *large* too. But *large* $\cdot$ *large* is larger in size than *large* so $+45.03 \cdot x^2$ is even larger than $-82.47 \cdot x$

iv. $+37.81$ is *bounded* and x is *large*. So, since *bounded* $\cdot$ *large* = *large*, $+37.81 \cdot x$ is *large* too. But *large* $\cdot$ *large* $\cdot$ *large* is larger in size than *large* $\cdot$ *large* so $+37.81 \cdot x^3$ is even larger than $+45.03 \cdot x^2$

So, in the output jet near ∞, $+37.81x^3$ must come first, $+45.03x^2$ must come second, $-82.47x$ comes third and -61.03 comes fourth

Then, we get the local input-output rule near ∞:

$$x \text{ near } \infty \xrightarrow{RIBA} TIBA(x) = \underbrace{+37.81x^3 + 45.03x^2 - 82.47x - 61.03}_{\text{output jet near } \infty}$$

2. Altogether, then:

PROCEDURE 14.2 To evaluate near ∞ the function specified by $x \xrightarrow{CUBIC} CUBIC(x) = ax^3 + bx^2 + cx + d$

i. *Declare* that x is near ∞

$$x\Big|_{x \text{ near } \infty} \xrightarrow{CUBIC} CUBIC(x)\Big|_{x \text{ near } \infty} = ax^3 + bx^2 + cx + d\Big|_{x \text{ near } \infty}$$

which gives:

$$x \text{ near } \infty \xrightarrow{CUBIC} CUBIC(x) = \underbrace{ax^3 + bx^2 + cx + d}_{\text{output-specifying code}}$$

ii. *Execute* the output-specifying code into a *jet* near ∞

$$= \underbrace{[a]\, x^3 \oplus [a]\, x^2 \oplus [b]\, x \oplus [c]}_{\text{output jet near } \infty}$$

which gives the *local input-output rule* near ∞:

$$x \text{ near } \infty \xrightarrow{CUBIC} CUBIC(x) = \underbrace{[a]\, x^3 \oplus [a]\, x^2 \oplus [b]\, x \oplus [c]}_{\text{output jet near } \infty}$$

(*Here* the jet near ∞ looks the same as the given global input-output rule but that is only because the output-specifying code *happened* to be written in *descending* order of exponents.)

DEMO 14.2 To evaluate near ∞ the function specified by the global input-output rule

$$x \xrightarrow{DINA} DINA(x) = -61.03 + 37.81x^3 + 51.32x^2 - 82.47x$$

i. We *declare* that x is near ∞

$$x\Big|_{x \text{ near } \infty} \xrightarrow{DINA} DINA(x)\Big|_{x \text{ near } \infty} = -61.03 + 37.81x^3 + 51.32x^2 - 82.47x\Big|_{x \text{ near } \infty}$$

which gives:

$$x \text{ near } \infty \xrightarrow{DINA} DINA(x) = \underbrace{-61.03 + 37.81\,x^3 + 51.32\,x^2 - 82.47\,x}_{\text{output-specifying code}}$$

ii. We *execute* the output-specifying code into a *jet* near ∞:

$$= \left[+37.81\right] x^3 \oplus \left[+51.32\right] x^2 \oplus \left[-82.47\right] x \oplus \left[-61.03\right]$$

which gives the *local input-output rule* near ∞:

$$x \text{ near } \infty \xrightarrow{DINA} DINA(x) = \underbrace{\left[+37.81\right] x^3 \oplus \left[+51.32\right] x^2 \oplus \left[-82.47\right] x \oplus \left[-61.03\right]}_{\text{output jet near } \infty}$$

(Here the jet near ∞ does *not* look the same as the *global* input-output rule because the output-specifying code happened *not* to be in descending order of exponents.)

3. The reason we use *jets* here is that the term *largest in size* is the *first* term so that to *approximate* the output we need only write the *first* term in the jet and just replace the remaining terms by [...] which stands for "something too small to matter here". In other words,

THEOREM 14.1 Approximate output near ∞. For *cubic* functions, the term in the jet that contributes most to the output near ∞ is the *highest degree term* in the output jet near ∞:

$$x \text{ near } \infty \xrightarrow{CUBIC} CUBIC(x) = \left[a\right] x^3 + [...]$$

EXAMPLE 14.5. Given the function specified by the global input-output rule

$$x \xrightarrow{DINA} DINA(x) = -61.03 + 37.81x^3 + 51.32x^2 - 82.47x$$

near ∞ we will often just use the *approximation*

$$x \text{ near } \infty \xrightarrow{KINA} KINA(x) = \left[+37.81\right] x^3 \oplus [...]$$

3 Output *near* x_0

We now deal with the output of the neighborhood of some *given* bounded input x_0.

1. In order to input a neighborhood of a given input x_0 we will declare that $x \leftarrow x_0 \oplus h$ that is that x is to be replaced by $x_0 \oplus h$. As a result, we will have to compute $(x_0 \oplus h)^2$ for which we will have to use an *addition formula* from
textscalgebra, namely **??** in **??** on page **??**.

2. We can then *execute* the input-output specifying phrase into a *jet* that is with the terms in *descending order of sizes* which here, since h is *small*, means that the powers of h will have to be in *ascending* order of exponents. We will then have the local input-output rule near the given input: jet near x_0

$$x_0 \oplus h \xrightarrow{CUBIC} CUBIC(x_0 \oplus h) = \underbrace{\text{Powers of } h \text{ in } \textit{ascending order of exponents}}_{\text{output jet near } \infty}$$

We will therefore use:

PROCEDURE 14.3 To evaluate near x_0 the function specified by $x \xrightarrow{CUBIC} CUBIC(x) = ax^3 + bx^2 + cx + d$

i. *Declare* that x is to be replaced by $x_0 + h$

$$x\,\Big|_{x \leftarrow x_0+h} \xrightarrow{CUBIC} CUBIC(x)\,\Big|_{x \leftarrow x_0+h} = ax^3 + bx^2 + cx + d\,\Big|_{x \leftarrow x_0+h}$$

which gives:

$$x_0 + h \xrightarrow{CUBIC} CUBIC(x_0 + h) = \underbrace{a(x_0+h)^3 + b(x_0+h)^2 + c(x_0+h) + d}_{\text{output-specifying code}}$$

ii. *Execute* the output-specifying code into a *jet* near x_0:

$$= a\left(x_0^3 + 3x_0^2h + 3x_0h^2 + h^3\right) + b\left(x_0^2 + 2x_0h + h^2\right) + c\,(x_0 + h) + d$$

$$= ax_0^3 \oplus 3ax_0^2\,h \oplus 3ax_0\,h^2 \oplus a\,h^3$$

$$\oplus\ bx_0^2 \oplus 2bx_0\,h \oplus b\,h^2$$

$$\oplus\ cx_0 \oplus c\,h$$

$$\oplus\ d$$

$$= \underbrace{\left[ax_0^3 + bx_0^2 + cx_0 + d\right] \oplus \left[3ax_0^2 + 2bx_0 + c\right]h \oplus \left[3ax_0 + b\right]h^2 \oplus \left[a\right]h^3}_{\text{output jet near } x_0}$$

which gives the *local input-output rule* near x_0:

$$x_0 + h \xrightarrow{CUBIC} CUBIC(x_0 + h)$$

$$= \underbrace{\left[ax_0^3 + bx_0^2 + cx_0 + d\right] \oplus \left[3ax_0^2 + 2bx_0 + c\right]h \oplus \left[3ax_0 + b\right]h^2 \oplus \left[a\right]h^3}_{\text{output jet near } x_0}$$

DEMO 14.3 To evaluate near -3 the function specified by

$$x \xrightarrow{ARBA} ARBA(x) = -32.67x - 31.18x^3 + 71.07 + 81.26x^2$$

i. We declare that x is to be replaced by $-3+h$

$$x \Big|_{x \leftarrow -3+h} \xrightarrow{ARBA} ARBA(x) \Big|_{x \leftarrow -3+h} = -32.67x - 31.18x^3 + 71.07 + 81.26x^2 \Big|_{x \leftarrow -3+h}$$

which gives

$$-3+h \xrightarrow{ARBA} ARBA(-3+h) = \underbrace{-32.67(-3+h) - 31.18(-3+h)^3 + 71.07 + 81.26(-3+h)^2}_{\text{output specifying code}}$$

ii. We *execute* the output-specifying code into a *jet* near -3:

$$= -32.67(-3+h) - 31.18\left((-3)^3 + 3(-3)^2h + 3(-3)h^2 + h^3\right) + 71.07 + 81.26\left((-3)^2 + 2(-3)h + h^2\right)$$

$$\begin{aligned} &= -32.67(-3) - 32.67h \\ &\quad - 31.18(-3)^3 - 31.18 \cdot 3(-3)^2h - 31.18 \cdot 3(-3)h^2 - 31.18h^3 \\ &\quad + 71.07 \\ &\quad + 81.26(-3)^2 + 81.26 \cdot 2(-3)h + 81.26h^2 \end{aligned}$$

$$\begin{aligned} &= +98.01 \oplus -32.67\,h \\ &\quad \oplus +841.86 \oplus -841.86\,h \oplus +280.62\,h^2 \oplus -31.18\,h^3 \\ &\quad \oplus +71.07 \\ &\quad \oplus +731.34 \oplus -487.56\,h \oplus +81.26\,h^2 \end{aligned}$$

$$\begin{aligned} &= \Big[+98.01 + 841.86 + 71.07 + 731.34\Big] \\ &\quad \oplus \Big[-32.67 - 841.86 - 487.56\Big]h \\ &\quad \oplus \Big[+280.62 + 81.26\Big]h^2 \\ &\quad \oplus \Big[-31.18\Big]h^3 \end{aligned}$$

$$= \underbrace{\Big[+1742.28\Big] \oplus \Big[-1362.09\Big]h \oplus \Big[+361.88\Big]h^2 \oplus \Big[+81.26\Big]h^3}_{\text{output jet near } -3}$$

which gives the *local input-output rule* near -3:

$$-3+h \xrightarrow{ARNA} ARBA(-3+h) = \underbrace{\Big[+1742.28\Big] \oplus \Big[-1362.09\Big]h \oplus \Big[+361.88\Big]h^2 \oplus \Big[+81.26\Big]h^3}_{\text{output jet near } -3}$$

3. When all we want is a feature-sign, though, the above procedure is very inefficient and we will then use the following procedure based directly on the fact that a *cubic function* is the addition of:

- a *cube function*, (See **??** on **??**)
- a *square function*, (See **??** on **??**)
- a *linear function*, (See **??** on **??**.)
- a *constant function*. (See **??** on **??**.)

that is:

$$x \xrightarrow{CUBIC} CUBIC(x) = \underbrace{ax^3}_{\text{cube}} \oplus \underbrace{bx^2}_{\text{square}} \oplus \underbrace{cx}_{\text{linear}} \oplus \underbrace{d}_{\text{constant}}$$

We declare that x is near x_0 that is that x must be replaced by $x_0 + h$:

$$x \xrightarrow{CUBIC} CUBIC(x) = \underbrace{a\,(x_0+h)^3}_{\text{cube}} \oplus \underbrace{b\,(x_0+h)^2}_{\text{square}} \oplus \underbrace{c\,(x_0+h)}_{\text{linear}} \oplus \underbrace{d}_{\text{constant}}$$

The output of the local input-output rule near x_0 will have to be a *jet*:

$$x_0 + h \xrightarrow{CUBIC} CUBIC(x_0+h) = \Big[\quad\Big] \oplus \Big[\quad\Big]h \oplus \Big[\quad\Big]h^2 \oplus \Big[\quad\Big]h^3$$

and we want to be able to get any one of the coefficients of the output jet without having to compute any of the other coefficients. So, what we will do is to get the contribution of each monomial function to the term we want. This requires us to have the *addition formulas* at our finger tips:

a.

$$(x_0+h)^2 = x_0^2 + 2x_0h + h^2 \text{(See ?? on page 403)}$$

b.

$$(x_0+h)^3 = x_0^3 + 3x_0^2h + 3x_0h^2 + h^3 \text{(See ?? on ??)}$$

More precisely,

i. If we want the *coefficient* of h^0 in the output jet:

- The cube function contributes ax_0^3
- The square function contributes bx_0^2
- The linear function contributes cx_0
- The constant function contributes d

so we have:

$$x_0 + h \xrightarrow{CUBIC} CUBIC(x_0+h) = \Big[ax_0^3 + bx_0^2 + cx_0 + d\Big] \oplus \Big[\quad\Big]h \oplus \Big[\quad\Big]h^2 \oplus \Big[\quad\Big]h^3$$

ii. If we want the *coefficient* of h^1 in the output jet:

- The cube function contributes $3bx_0^2$
- The square function contributes $2bx_0$

- The linear function contributes c
- The constant function contributes nothing

so we have:
$x_0 + h \xrightarrow{CUBIC} CUBIC(x_0 + h) = \Big[\quad\Big] \oplus \Big[3bx_0^2 + 2bx_0 + c\Big]h \oplus \Big[\quad\Big]h^2 \oplus \Big[\quad\Big]h^3$

iii. If we want the *coefficient* of h^2 in the output jet:

- The cube function contributes $3bx_0$
- The square function contributes c
- The linear function contributes nothing
- The constant function contributes nothing

so we have:
$x_0 + h \xrightarrow{CUBIC} CUBIC(x_0 + h) = \Big[\quad\Big] \oplus \Big[\quad\Big]h \oplus \Big[3bx_0 + c\Big]h^2 \oplus \Big[\quad\Big]h^3$

iv. If we want the *coefficient* of h^3 in the output jet:

- The cube function contributes a
- The square function contributes nothing
- The linear function contributes nothing
- The constant function contributes nothing

so we have:
$x_0 + h \xrightarrow{CUBIC} CUBIC(x_0 + h) = \Big[\quad\Big] \oplus \Big[\quad\Big]h \oplus \Big[\quad\Big]h^2 \oplus \Big[a\Big]h^3$

4 Local graphs

Just as we get a *plot point at* a *bounded* input from the *output at* that input, we get the *local graph near* any input, be it *bounded* or *infinity*, from the *jet near* that input.

Procedure 14.4 To graph near ∞ the function specified by $x \xrightarrow{CUBIC} CUBIC(x) = ax^3 + bx^2 + cx + d$

1. Get the *output jet* near ∞:
$x \text{ near } \infty \xrightarrow{CUBIC} CUBIC(x) = \underbrace{\Big[a\Big] x^3 \oplus \Big[b\Big] x^2 \oplus \Big[c\Big] x \oplus \Big[d\Big]}_{\text{output jet near } \infty}$

(See **??** on **??**.)

2. Get the *local graphs*:

a. Of the *cubic term* by graphing near ∞ the monomial function $x \to \Big[a\Big]x^3$ using **?? ??** on **??**.

b. Of the *quadratic term* by graphing near ∞ the monomial function $x \to \left[\boxed{a}\right] x^2$ using **??** **??** on **??**.

c. Of the *linear term* by graphing near ∞ the monomial function $x \to \left[\boxed{b}\right] x$ using **??** **??** on **??**.

d. Of the *constant term* by graphing near ∞ the monomial function $x \to \left[\boxed{c}\right]$ using **??** **??** on **??**.

3. Get the local graph near ∞ of $CUBIC$ using chapter 9 by adding-on to the local graph of the *cubic term* the local graph of the *quadratic term*, the local graph of the the local graph of, and the local graph of the *constant term*.

DEMO 14.4 To graph near ∞ the function specified by the global input-output rule

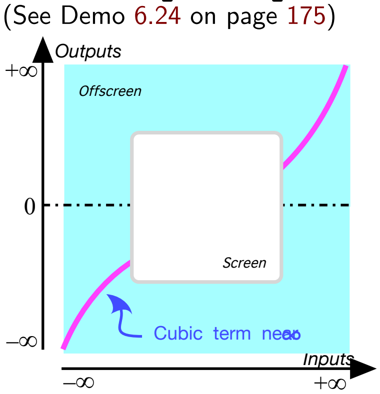
$$x \xrightarrow{DINA} DINA(x) = -61.03 + 37.81x^3 + 51.32x^2 - 82.47x$$

1. We get the output jet near ∞: (See Demo 14.2 on page 291)

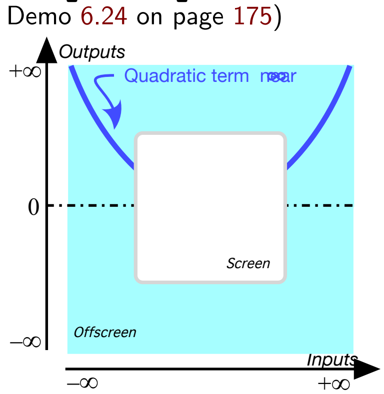
$$x \text{ near } \infty \xrightarrow{DINA} DINA(x) = \underbrace{\left[+37.81\right] x^3 \oplus \left[+51.32\right] x^2 \oplus \left[-82.47\right] x \oplus \left[-61.03\right]}_{\text{output jet near } \infty}$$

2. Get the local graph near ∞ of each term:

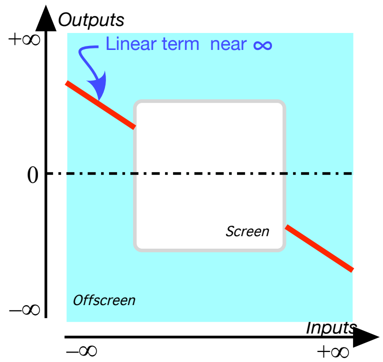
a. We get the graph of the *cubic* term by graphing the monomial function $x \to \left[+37.81\right] x^3$ near ∞ (See Demo 6.24 on page 175)



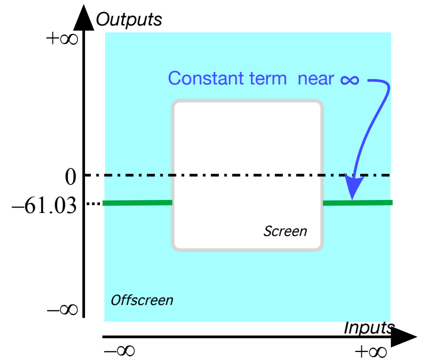
b. We get the graph of the *quadratic* term by graphing the monomial function $x \to \left[+51.32\right] x^2$ near ∞ (See Demo 6.24 on page 175)



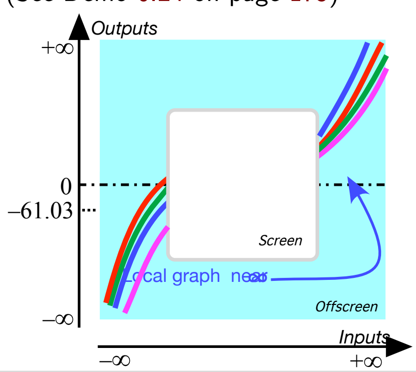
c. We get the graph of the *linear* term by graphing the monomial function $x \to \left[\boxed{-82.47}\right] x$ near ∞ (See Demo 6.24 on page 175)



d. We get the graph of the *constant* term by graphing the monomial function $x \to \left[\boxed{-61.03}\right]$ near ∞ (See Demo 6.24 on page 175)



3. We get the local graph near ∞ of $DINA$ by adding-on to the graph of the *quadratic* term the graph of the *linear* term and the graph of the *constant* term. (See Demo 6.24 on page 175)



PROCEDURE 14.5 To graph near x_0 the function specified by $x \xrightarrow{CUBIC} CUBIC(x) = ax^3 + bx^2 + cx + d$

1. Get the local *input-output rule* near x_0 of $CUBIC$ using **??** **??** on **??**

$$x_0 + h \xrightarrow{CUBIC} CUBIC(x_0 + h)$$
$$= \underbrace{\left[ax_0^3 + bx_0^2 + cx_0 + d\right] \oplus \left[3ax_0^2 + 2bx_0 + c\right]h \oplus \left[3ax_0 + b\right]h^2 \oplus \left[a\right]h^3}_{\text{output jet near } x_0}$$

2. Get the *local graphs*:

a. Of the *constant* term by graphing near 0 the monomial function $x \to \left[\boxed{ax_0^3 + bx_0^2 + cx_0 + d} \right]$

b. Of the *linear* term by graphing near 0 the monomial function $x \to \left[\boxed{3ax_0^2 + 2bx_0 + c} \right] x$

c. Of the *quadratic* term by graphing near 0 the monomial function $x \to \left[\boxed{3ax_0 + b} \right] x^2$

d. Of the *cubic* term by graphing near 0 the monomial function $x \to \left[\boxed{a} \right] x^3$

3. Get the *local graph* of $CUBIC$ near x_0 by adding to the local graph of the *constant term*, the local graph of the *linear term*, the local graph of the *quadratic term*, the local graph of the *cubic term*.

DEMO 14.5 To graph near -3 the function specified by

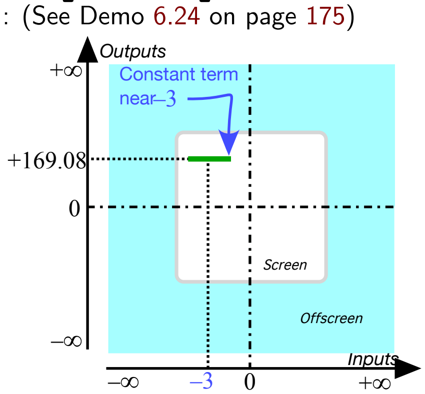
$$x \xrightarrow{ARBA} ARBA(x) = -32.67x - 31.18x^3 + 71.07 + 81.26x^2$$

1. We get the *local input-output rule* near -3 of $ARBA$ (See Demo 14.3 on page 294):

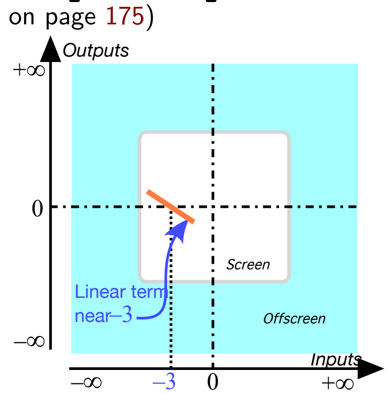
$$-3+h \xrightarrow{ARNA} ARBA(-3+h) = \underbrace{\left[\boxed{+1742.28} \right] \oplus \left[\boxed{-1362.09} \right] h \oplus \left[\boxed{+361.88} \right] h^2 \oplus \left[\boxed{+81.26} \right] h^3}_{\text{output jet near } -3}$$

2. We get the local graphs

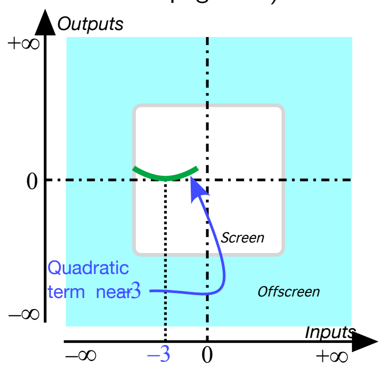
a. We get the graph of the *constant* term near -3 by graphing the monomial function $x \to \left[\boxed{+1742.28} \right]$.
: (See Demo 6.24 on page 175)



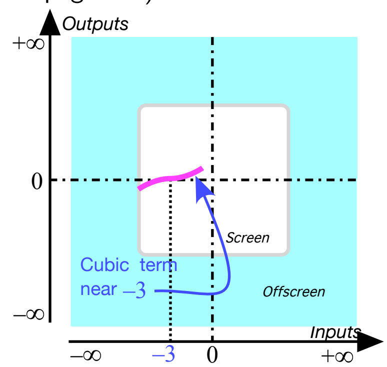
b. We get the graph of the *linear* term near -3 by graphing the monomial function $x \to \left[\boxed{-1362.09} \right] x$ (See Demo 6.24 on page 175)



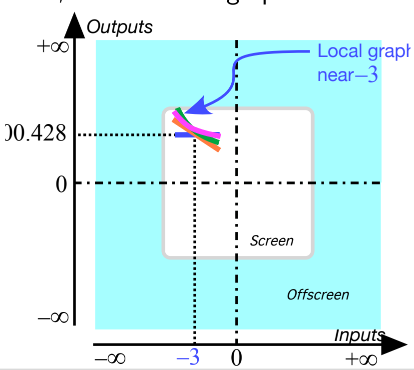
c. We get the graph of the *quadratic* term near -3 by graphing the monomial function $x \to \left[\boxed{+361.88}\right] x^2$: (See Demo 6.24 on page 175)



d. We get the graph of the *cubic* term near -3 by graphing the monomial function $x \to \left[\boxed{+81.26}\right] x^3$: (See Demo 6.24 on page 175)



3. We get the *local graph* near -3 of $ARBA$ by adding to the local graph of the *constant term* the local graph of the *linear term*, the local graph of the *quadratic term*, and the local graph of the *cubic term*. (See Demo 6.24 on page 175)



5 Local Feature-signs

As we saw in **??** **??**, a feature-sign near a given input, be it near ∞ or near x_0, can be read from the *local graph* and so we could proceed as follows:

i. Get the *local input-output rule* near the given input (See **??** on **??** when the given input is ∞ or **??** on **??** when the given input is x_0.)

ii. Get the *local graph* from the local input-output rule (See **??** on **??**.)

iii. Get the *feature-sign* from the *local graph*. (See **??** **??**.)

However, things are in fact much simpler: Given an input, be it ∞ or a bounded input x_0, to get a required feature-sign near that given input, we look for the term in the output jet near that input that

i. Has the required feature.

and

ii. Is the largest-in-size of all those terms with the required feature.

So, as we will now see, we usually need to get only *one* term in the output jet rather than the whole output jet.

1. Near *infinity* things are quite straightforward because, for a cubic function, the first term in the output jet near ∞ is both the *largest-in-size* and a *regular* monomial so that it has *all three features*:

PROCEDURE 14.6 To get the feature-signs near ∞ of the function specified by $x \xrightarrow{CUBIC} CUBIC(x) = ax^3 + bx^2 + cx + d$

i. Get the *approximate* local input-output rule near ∞:

$$x \text{ near } \infty \xrightarrow{CUBIC} CUBIC(x) = \underbrace{\left[a\right]x^3 \oplus \left[b\right]x^2 \oplus \left[c\right]x \oplus \left[d\right]}_{\text{output jet near } \infty}$$
$$= \underbrace{\left[a\right]x^3 \oplus [...]}_{\text{approximate output jet near } \infty}$$

ii. Then, in the *approximate output jet* near ∞:

- Get the *Height-sign*, the *Slope-sign* and the *Concavity-sign* all from the *cubic term* $\left[a\right]x^3$ because the next terms, $\left[b\right]x^2$, $\left[c\right]x$ and $\left[d\right]$ are *too small to matter*. (Not to mention the fact that a linear term has no concavity and a constant term has neither concavity nor slope.)

DEMO 14.6

To get the Height-sign near ∞ of the function specified by

$$x \xrightarrow{DELIA} DELIA(x) = +12x^3 - 2x^2 + 63x - 155$$

.

i. We get the local input-output rule near ∞:

$$x \text{ near } \infty \xrightarrow{DELIA} DELIA(x) = +12x^3 - 2x^2 + 63x - 155$$
$$= \underbrace{\Big[+12\Big]x^3 \oplus \Big[-2\Big]x^2 \oplus \Big[+63\Big]x \oplus \Big[-155\Big]}_{\text{output jet near } \infty}$$

ii. We get *Height-sign* from the *cubic* term $\Big[+12\Big]x^3$. (The *quadratic* term $\Big[-2\Big]x^2$, the linear term $\Big[+63\Big]x$ and the *constant* term $\Big[-155\Big]$ are *too small* *to matter*)

iii. Since the *cubic coefficient* $\Big[+12\Big]$ is *positive*, we get that Height-sign $DELIA$ near $\infty = \langle +, - \rangle$. (Seen from ∞.)

Demo 14.7

Get the *slope-sign* near ∞ of the function specified by the global input-output rule

$$x \xrightarrow{DETER} DETER(x) = -0.45x^3 + 3.03x^2 - 81.67x + 46.92$$

i. We get the local input-output rule near ∞:

$$x \text{ near } \infty \xrightarrow{DETER} DETER(x) = -045x^3 + 3.03x^2 - 81.67x + 46.92$$
$$= \underbrace{\Big[-0.45\Big]x^3 \oplus \Big[+3.03\Big]x^2 \oplus \Big[-81.67\Big]x \oplus \Big[+46.92\Big]}_{\text{output jet near } \infty}$$

ii. We get *Slope-sign* from the *cubic* term $\Big[-0.45\Big]x^3$. (The *quadratic term* $\Big[+3.03\Big]x^2$, the *linear term* $\Big[-81.67\Big]x$ and the *constant term* $\Big[+46.92\Big]$ are too small to matter.)

Since the *cubic coefficient* -0.45 is negative, we get that Slope-sign $DETER$ near $\infty = \langle \searrow, \searrow \rangle$. (Seen from ∞.)

2. Near a *bounded input* though, things are a bit more complicated:

i. The *first* term in the output jet is *usually* the *largest-in-size* so that it gives the Height-sign. However, the first term *usually* has neither Slope nor Concavity because the first term is *usually* a constant term.

ii. The *second* term in the output jet is *usually* too smalll-in-size to change the Height-sign as given by the first term but it is *usually* the *largest-in-size* term that can give the Slope-sign. However, the second term has no Concavity because the second term is *usually* a linear term.

iii. The third *term* in the output jet is *usually* too smalll-in-size to change the Height-sign given by the first term and the Slope-sign given by the second term but it is *usually* the *only term* that can give the Concavity-sign.

So we can *usually* read each feature-sign directly from the appropriate term in the output jet - keeping in mind that the exceptional monomial functions do not have all the features.

However, near a *bounded input*, the given bounded input may turn out to be *critical* for the local feature:

i. If the *constant term* in the output jet is 0, then the term which gives the Height-sign can be either the linear term or the quadratic term if the linear term is 0 or even the cubic term if the quadratic term turns out to be 0 too. The bounded input is then again said to be *critical for the Height*.

ii. If the *linear term* in the output jet is 0, then the term which gives the Slope-sign is the *quadratic term* or the *cubic term* is the *quadratic term* turns out to be 0 too. The bounded input is then said to be *critical for the Slope*. critical for the Concavity
iii. If the *quadratic term* in the output jet is 0, then the term which gives the Concavity-sign is the *cubic term*. The bounded input is then said to be **critical for the Concavity**.
So, we *usually* need to compute only one coefficient in the output jet. But if the given bounded input turns out to be *critical* for that feature, then we need to compute the next coefficient: So we use

PROCEDURE 14.7 To get the feature-signs near x_0 of the function specified by $x \xrightarrow{CUBIC} CUBIC(x) = ax^3 + bx^2 + cx + d$

i. Get the local input-output rule near x_0:

$$\begin{aligned} x_0 + h \xrightarrow{CUBIC} CUBIC(x_0+h) &= a(x_0+h)^3 + b(x_0+h)^2 + c(x_0+h) + d \\ &= a\left(x_0^3 + 3x_0^2h + 3x_0h^2 + h^3\right) + b\left(x_0^2 + 2x_0h + h^2\right) + c\,(x_0+h) + d \\ &= \underbrace{\left[ax_0^3 + bx_0^2 + cx_0 + d\right] \oplus \left[3ax_0^2 + 2bx_0 + c\right]h \oplus \left[3ax_0 + b\right]h^2 \oplus \left[a\right]h^3}_{\text{output jet near } x_0} \end{aligned}$$

ii. Then, in the *output jet* near x_0:

- Get the *Height-sign* from the *constant term* $\left[ax_0^3 + bx_0^2 + cx_0 + d\right]$. (The *linear term*, the *quadratic term* and the *cubic term* are *too small to matter*.)
  If the *constant coefficient* is 0, get the *Height-sign* from the *linear term* $\left[3ax_0^2 + 2bx_0 + c\right]h$. (The *quadratic term* and the *cubic term* are too small to matter.)
  If the *linear coefficient* is 0 too, get the *Height-sign* from the *quadratic term* $\left[3ax_0 + b\right]h^2$. (The *quadratic term* and the *cubic term* are too small to matter.)
  If the *quadratic coefficient* is 0 too, get the *Height-sign* from the *cubic term* $\left[a\right]h^3$. (The *quadratic term* and the *cubic term* are too small to matter.)
- Since the *constant term* has no slope, get the *Slope-sign* from the *linear term* $\left[3ax_0^2 + 2bx_0 + c\right]h$. (The *quadratic term* and the *cubic term* are too small to matter.)
  If the *linear coefficient* is 0, get the *Slope-sign* from the *quadratic term* $\left[3ax_0 + b\right]h^2$. (The *cubic term* is too small to matter.)
  If the *quadratic coefficient* is 0 too, get the *Slope-sign* from the *cubic term* $\left[a\right]h^3$.

- Since both the *constant term* and the *linear term* have no concavity, get *Concavity-sign* from the *quadratic term* $\big[3ax_0 + b\big]h^2$. (The *cubic term* is too small to matter.)
  If the *quadratic coefficient* is 0, get the *Slope-sign* from the *cubic term* $\big[a\big]h^3$.

DEMO 14.8

To get the feature signs near -3 of the function specified by the global input-output rule

$$x \xrightarrow{ARBA} ARBA(x) = -32.67x + 71.07 + 81.26x^2$$

i. We get the local input-output rule near -3 (See Demo 14.3 on page 294):

$$-3+h \xrightarrow{ARBA} ARBA(-2+h) = \underbrace{-32.67(-3+h) + 71.07 + 81.26(-3+h)^2}_{\text{output specifying code}}$$

$$= \underbrace{\Big[+900.428\Big] \oplus \Big[-519.63\Big]h \oplus \Big[+81.26\Big]h^2}_{\text{output jet near } -3}$$

ii. Then, from the *output jet*:

- Since the *constant coefficient* $\Big[+900.428\Big]$ is *positive*, we get that Height-sign $ARBA$ near $-3 = \langle +, + \rangle$.
- Since the *linear coefficient* $\Big[-519.63\Big]h$ is *negative*. we get that Slope-sign $ARBA$ near $-3 = \langle \diagdown, \diagdown \rangle$
- Since the *quadratic coefficient* $\Big[+81.26\Big]h^2$ is *positive*, we get that Concavity-sign $ARBA$ near $-3 = \langle \cup, \cup \rangle$

THEOREM 14.2 Approximation Near ∞

6 Local Graph Near Infinity

THEOREM 14.3 Height-sign Near ∞

THEOREM 14.4 Slope-sign Near ∞

THEOREM 14.5 Concavity-sign Near ∞

THEOREM 14.6 Local Input-Output Rule

THEOREM 14.7 Height-sign Near x_0

THEOREM 14.8 Slope-sign Near x_0

THEOREM 14.9 Concavity-sign Near x_0 Given the function $CUBICa, b, c, d$

- $When$ Local *square* coefficient of $CUBIC(x_0) = +$,

$$\textit{Concavity-sign}\, CUBIC\,|_{\text{near } x_0} = (\cup, \cup)$$

- $When$ Local *square* coefficient of $CUBIC(x_0) = -$,

$$\textit{Concavity-sign}\, CUBIC\,|_{\text{near } x_0} = (\cap, \cap)$$

- $When$ Local *square* coefficient of $CUBIC(x_0) = 0$,

$\textit{Concavity-sign}\, CUBIC|_{\text{near } x_0}$ depends on the sign of the local *cube* coefficient of $CUBIC(x_0)$

Chapter 15

Cubic Functions: Global Analysis

In the case of cubic functions, we will be able to solve *exactly* only a very few global problems because everything begins to be truly computationally complicated.

1 Global Graph

As always, we use

PROCEDURE 15.1 Essential graph of a function specified by $x \xrightarrow{CUBIC} CUBIC(x) = ax^3 + bx^2 + cx + d$

i. Graph the function near ∞, (See **??** on **??**.)
ii. Ask the ESSENTIAL QUESTION:

- Do all *bounded inputs* have *bounded outputs*

or

- Are there *bounded inputs* whose nearby inputs have unbounded outputs? (∞-height inputs.)

essential-feature input

iii. Use the local input-output rule near x_0 to get further information. (See **??** on **??**.)

But, given a *bounded* input x_0, we have that:

- a being bounded, ax_0^3 is also bounded
- b being bounded, bx_0^2 is also bounded
- c being bounded, cx_0 is also bounded
- and d being bounded

altogether, we have that $ax_0^3 + bx_0x^2 + cx_0 + d$ is bounded and that the answer to the ESSENTIAL QUESTION is:

THEOREM 15.1 Bounded Height Under a *cubic* functions, all bounded inputs have *bounded outputs*.

and therefore

THEOREM 15.2 Offscreen Graph The *offscreen graph* of a *cubic* function consists of just the *local graph near* ∞.

We now deal in detail with the third step.

EXISTENCE THEOREMS

Since cubic functions have no *bounded* ∞-height input, the only way a feature can change sign near a bounded input is when the feature is 0 near the bounded input. In particular, **essential 0-feature inputs** are bounded inputs

- with a 0 feature,
- whose existence is forced by the *offscreen graph*—which, in the case of cubic functions consists, by theorem 15.2, of only the *local graph near* ∞.

None of the following theorems, though, will indicate *where* the 0-feature inputs inputs are *located*. The **Location Theorems** will be dealt with in the last part of the chapter.

2 Concavity-sign

Given the function specified by the global input-output rule

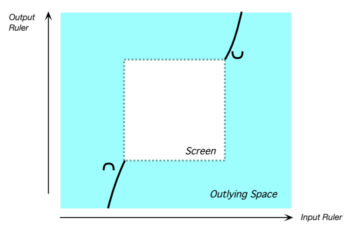
$$x \xrightarrow{\ CUBIC\ } CUBIC(x) = ax^3 + b^2x + cx + d$$

recall that when x is near ∞ the **Concavity-sign Near ∞ Theorem** for cubic functions says that:

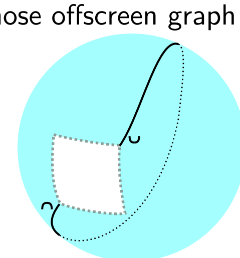
- When a is $+$, Concavity-Sign$|_{x \text{ near } \infty} = (\cup, \cap)$
- When a is $-$, Concavity-Sign$|_{x \text{ near } \infty} = (\cap, \cup)$

1. Since the concavity changes sign as x goes from the left side of ∞ to the right side of ∞ *across* ∞ , the concavity also has to change sign as x goes from the left side of ∞ to the right side of ∞ *across the screen*. In other words, there has to be a *bounded* concavity-sign change input.

EXAMPLE 15.1. Given a cubic function whose offscreen graph is

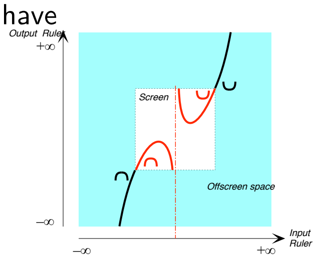


Mercator view

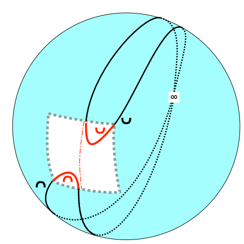


Magellan view

there has therefore to be a bounded concavity-sign change input, $x_{\text{concavity sign-change}}$. But since there cannot be a bounded ∞-height input, we cannot have

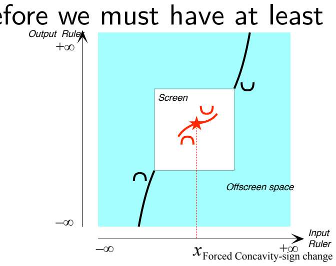


Mercator view

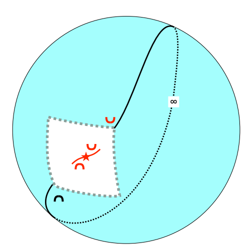


Magellan view

and therefore we must have at least



Mercator view



Magellan view

So, based on the off-screen graph, we have

THEOREM 15.3 Concavity sign-change A cubic function must have *at least one* bounded concavity sign-change input.

2. On the other hand, based on the off-screen graph, a cubic function could have any *odd* number of 0-concavity inputs. Based on the *general local input-output rule*, we will see that a cubic function can have *at most one* 0-concavity input. But, at this point, all we know for sure is

THEOREM 15.4 0-Concavity Existence A cubic functions must have *at least one* concavity-sign change input:

$$x_{\text{concavity sign-change}} = x_{\text{0-concavity}}$$

3 Slope-sign

Given the cubic function $CUBIC_{a,b,c,d}$, that is the function specified by the global input-output rule

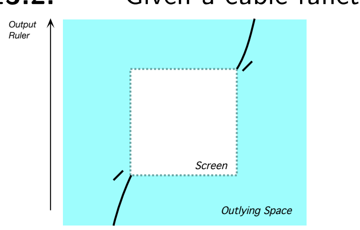
$$x \xrightarrow{CUBIC} CUBIC(x) = ax^3 + b^2x + cx + d$$

recall that when x is near ∞ the **Slope-sign Near ∞ Theorem** for cubic functions says that:

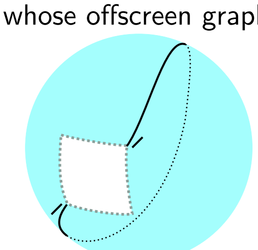
- When a is $+$, Slope-Sign$|_{x \text{ near } \infty} = (\diagup, \diagup)$
- When a is $-$, Slope-Sign$|_{x \text{ near } \infty} = (\diagdown, \diagdown)$

1. Since the slope does *not* changes sign as x goes through ∞ from the left side of ∞ to the right side of ∞, the slope does not have to change sign as x goes *across the screen* from the left side of ∞ to the right side of ∞ so there does not have to be a *bounded* slope-sign change input:

EXAMPLE 15.2. Given a cubic function whose offscreen graph is

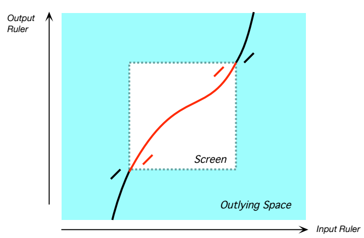


Mercator view

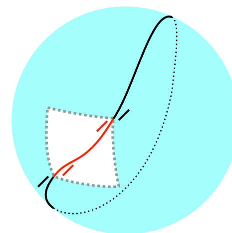


Magellan view

there is no need for a bounded slope-sign change input, $x_{\text{Slope-sign change}}$ and therefore we can have



Mercator view


Magellan view

2. On the other hand, based on just *graphic* considerations, a cubic function could have any number of 0-slope inputs. Based on *input-output rule* considerations, we will see that a cubic function can have only zero, one or two 0-slope inputs. But, at this point, all we know for sure is

> **THEOREM 15.5 Slope-Sign Change Existence** A cubic function need not have a *Slope-sign change* input.

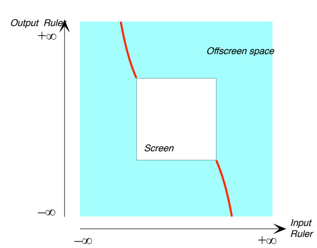
And thus also

> **THEOREM 15.6 0-Slope Existence** A cubic function need not have a 0-*Slope* input.

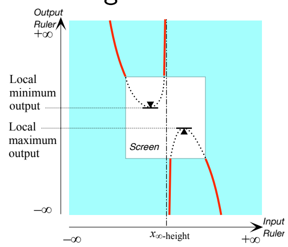
4 Extremum

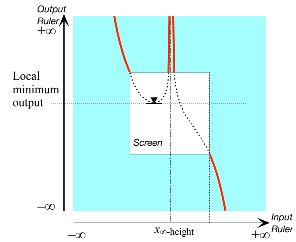
From the *optimization* viewpoint, the most immediately striking feature of an affine function is the absence of a forced extreme input, that is of a bounded input whose output is either larger than the output of nearby inputs or smaller than the output of nearby inputs. On the other hand, at this point we cannot prove that there is no extreme input.

EXAMPLE 15.3. Given a cubic function with the offscreen graph:

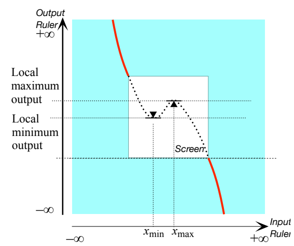


Since there can be no ∞-height input, we cannot have, for instance, either one of the following

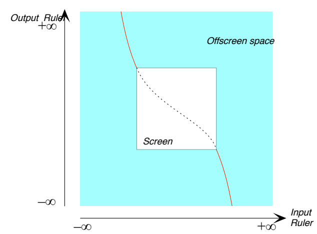




On the other hand, there is nothing to prevent a *fluctuation* such as:



But no extremum input is *forced*:



So, we have

THEOREM 15.7 Extremum Existence A cubic function has no *forced* extremum input

5 Height-sign

Given the cubic function $CUBIC_{a,b,c,d}$, that is the function specified by the global input-output rule

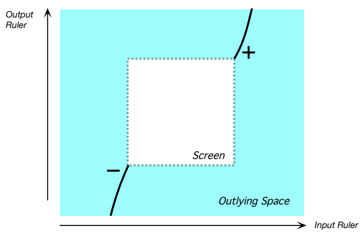
$$x \xrightarrow{CUBIC} CUBIC(x) = ax^3 + b^2x + cx + d$$

recall that when x is near ∞ the **Height-sign Near ∞ Theorem** for cubic functions says that:

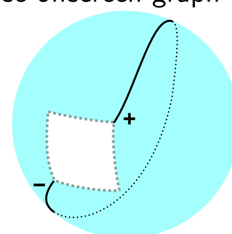
- When a is $+$, Height-Sign$|_{x \text{ near } \infty} = (+, -)$
- When a is $-$, Height-Sign$|_{x \text{ near } \infty} = (-, +)$

1. Since the height changes sign as x goes from the left side of ∞ to the right side of ∞ *across* ∞ , the height has also to change sign as x goes from the left side of ∞ to the right side of ∞ *across the screen.* In other words, there has to be a *bounded* height-sign change input.

EXAMPLE 15.4. Given a cubic function whose offscreen graph is

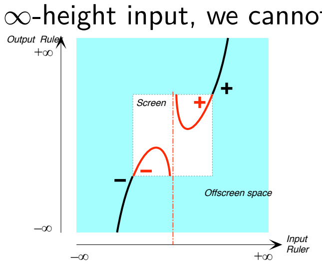


Mercator view

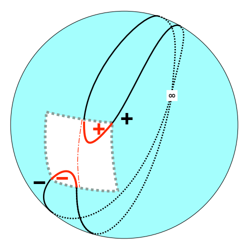


Magellan view

there has therefore to be a height-sign change input But since there cannot be a bounded ∞-height input, we cannot have

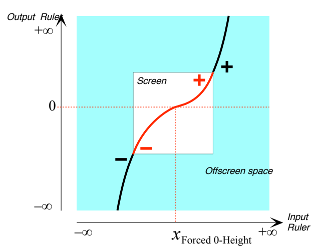


Mercator view

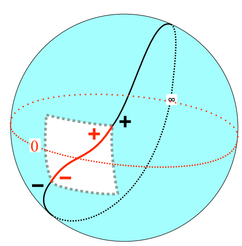


Magellan view

and therefore we must have



Mercator view



Magellan view

2. Moreover, because there is no *bounded ∞-height input* where the height could change sign, $x_{\text{height-sign change}}$ has to be a bounded input where the height is 0. As a result, we have that

THEOREM 15.8 Height-Sign Change Existence A cubic functions must have a *Height-sign change* input and

$$x_{\text{Height-sign change}} = x_{\text{0-height}}$$

LOCATION THEOREMS

Previously, we only established the *existence* of certain essential bounded inputs of cubic functions and this investigation was based on *graphic* considerations. Here we will investigate the *location* of the essential bounded inputs and this investigation will be based on the *generic local input-output rule*.

6 0-Concavity Location

Given a cubic function, the global problem of *locating* an input where the local concavity is 0 is still fairly simple.

More precisely, given a cubic function $CUBIC_{a,b,c,d}$, that is the cubic function specified by the global input-output rule

$$x \xrightarrow{CUBIC} CUBIC(x) = ax^3 + b^2x + cx + d$$

since the *concavity* near x_0 is the local square coefficient $3ax_0 + b$, in order to find the input(s) where the local concavity is 0, we need to solve the affine equation

$$3ax + b = 0$$

by reducing it to a basic equation:

$$\begin{aligned} 3ax + b - b &= 0 - b \\ 3ax &= -b \\ \frac{3ax}{3a} &= \frac{-b}{3a} \\ x &= \frac{-b}{3a} \end{aligned}$$

So, we have:

THEOREM 15.9 0-slope Location For any cubic function $CUBIC_{a,b,c,d}$,

$$x_{0-concavity} = \frac{-b}{3a}$$

In fact, we also have:

THEOREM 15.10 Global Concavity-sign Given a cubic function $CUBIC_{a,b,c,d}$,

- When a is positive,

$$\begin{aligned}
\text{Concavity-sign } CUBIC|_{\text{Everywhere} < \frac{-b}{3a}} &= (\cap, \cap) \\
\text{Concavity-sign } CUBIC|_{\frac{-b}{3a}} &= (\cap, \cup) \\
\text{Concavity-sign } CUBIC|_{\text{Everywhere} > \frac{-b}{3a}} &= (\cup, \cup)
\end{aligned}$$

- When a is negative,

$$\begin{aligned}
\text{Concavity-sign } CUBIC|_{\text{Everywhere} < \frac{-b}{3a}} &= (\cup, \cup) \\
\text{Concavity-sign } CUBIC|_{\frac{-b}{3a}} &= (\cup, \cap) \\
\text{Concavity-sign } CUBIC|_{\text{Everywhere} > \frac{-b}{3a}} &= (\cap, \cap)
\end{aligned}$$

The case is easily made by testing near ∞ the intervals for the corresponding inequations.

7 0-Slope Location

In the case of affine functions and of quadratic functions, we were able to prove that there was no shape difference with the principal term near ∞ by showing that there could be no *fluctuation*:

- In the case of *affine functions* we were able to prove that there was no shape difference with *dilation functions*
- In the case of *quadratic functions* we were able to prove that there was no shape difference with *square functions*.

More precisely, given the cubic function $CUBIC_{a,b,c,d}$, that is the function specified by the global input-output rule

$$x \xrightarrow{CUBIC} CUBIC(x) = ax^3 + b^2x + cx + d$$

since the slope near x_0 is the local linear coefficient $3ax^2 + 2bx + c$, in order to find the input(s) where the local slope is 0, we need to solve the *quadratic equation*

$$3ax^2 + 2bx + c$$

which we have seen we cannot solve by reduction to a basic equation and for which we will have to use the 0-**Height Theorem** for quadratic functions, keeping in mind, though, that

- For a as it appears in 0-**Height Theorem** for quadratic functions, we have to substitute the *squaring* coefficient of $3ax^2 + 2bx + c$, namely $3a$,
- For b as it appears in 0-**Height Theorem** for quadratic functions, we have to substitute the *linear coefficient* of $3ax^2 + 2bx + c$ namely $2b$,
- For c as it appears in 0-**Height Theorem** for quadratic functions, we have to substitute the *constant* coefficient of $3ax^2 + 2bx + c$ namely c.

1. It will be convenient, keeping in mind the above substitutions, first to compute

$$\begin{aligned} x_{0-slope \text{ for } [3ax^2+2bx+c]} &= -\frac{2b}{2 \cdot 3a} \\ &= -\frac{2b}{6a} \\ &= -\frac{b}{3a} \\ &= x_{0-concavity \text{ for } CUBIC} \end{aligned}$$

2. Then, still keeping in mind the above substitutions, we compute the discriminant of $3ax^2 + 2bx + c$:

$$\begin{aligned} \text{Discriminant}[3ax^2 + 2bx + c] &= (2b)^2 - 4(3a)(c) \\ &= 4b^2 - 12ac \end{aligned}$$

3. Then we have:

- When Discriminant $[3ax^2+2bx+c] = 4b^2-12ac < 0$, the local linear coefficient of $CUBIC$, $[3ax^2+2bx+c]$, has *no* 0-height input and therefore $CUBIC$ has *no* 0-slope input.
- When Discriminant $[3ax^2 + 2bx + c] = 4b^2 - 12ac = 0$, the local linear coefficient of $CUBIC$, $[3ax^2 + 2bx + c]$, has *one* 0-height input and therefore $CUBIC$ has *one* 0-slope input, namely
  - ► $x_{0-\text{slope for } CUBIC} = x_{0-height \text{ for } [3ax^2+2bx+c]} = -\frac{b}{3a}$,
- When Discriminant $[3ax^2 + 2bx + c] = 4b^2 - 12ac > 0$, the local linear coefficient of $CUBIC$, $[3ax^2 + 2bx + c]$, has *two* 0-height inputs and therefore $CUBIC$ has *two* 0-slope inputs., namely:
  - ► $x_{0-\text{slope for } CUBIC} = x_{0-height \text{ for } [3ax^2+2bx+c]} = -\frac{b}{3a}+\frac{\sqrt{4b^2-12ac}}{2a}$

  and
  - ► $x_{0-\text{slope for } CUBIC} = x_{0-height \text{ for } [3ax^2+2bx+c]} = -\frac{b}{3a}-\frac{\sqrt{4b^2-12ac}}{2a}$

In terms of the function $CUBIC$, this gives us:

THEOREM 15.11 0-slope Location Given the cubic function $CUBIC_{a,b,c,d}$, when
- •Disc. $[3ax^2 + 2bx + c] = 4b^2 - 12ac < 0$, $CUBIC$ has *no* 0-Slope input
- •Disc. $[3ax^2 + 2bx + c] = 4b^2 - 12ac = 0$, $CUBIC$ has *one* 0-Slope input
- •Disc. $[3ax^2 + 2bx + c] = 4b^2 - 12ac > 0, CUBIC$ has *two* 0-Slope inputs

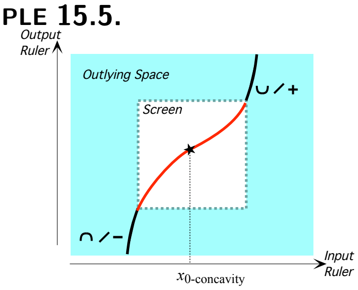
Shape type O
Shape type I

8 Extremum Location

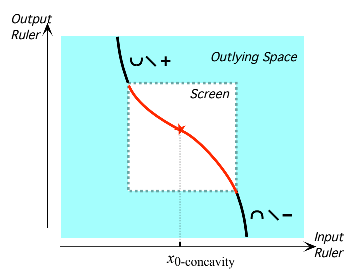
The 0-slope inputs are the only ones which can be extremum inputs. So, there will therefore be three types of cubic functions according to the number of 0-slopes inputs:

1. When Discriminant $[3ax^2+2bx+c] = 4b^2-12ac < 0$ so that $CUBIC$ has *no* 0-Slope input, there can be no extremum input and we will say that this type of cubic is of **Shape type 0**.

EXAMPLE 15.5.



Cube coefficient positive



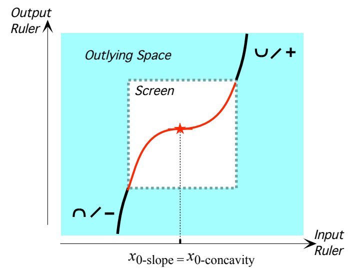
Cube coefficient negative

Since cubic function of Shape type O have no 0-Slope input, their shape is *not* like that of *cubing functions*.

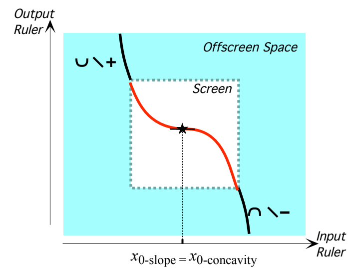
2. When Discriminant $[3ax^2+2bx+c] = 4b^2-12ac = 0$ so that $CUBIC$ has *one* 0-Slope input, there will still be no extremum input and we will say that this type of cubic is of **Shape type I**.

EXAMPLE 15.6.

Shape type II



Cube coefficient positive

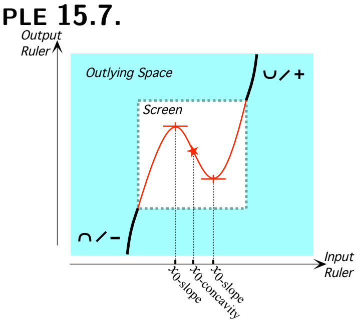


Cube coefficient negative

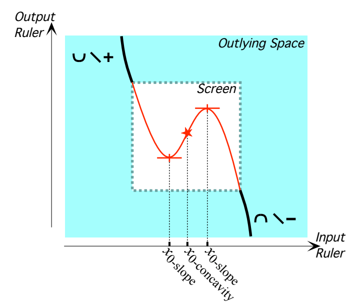
Since cubic function of *Shape type I* do have one 0-Slope input, their shape is very much like that of *cubing functions*.

3. When Discriminant $[3ax^2+2bx+c] = 4b^2 - 12ac > 0$ so that $CUBIC$ has *two* 0-Slope input, there will be one minimum input and one maximum input and we will say that this type of cubic is of **Shape type II**.

EXAMPLE 15.7.



Cube coefficient positive



Cube coefficient negative

We can thus state:

THEOREM 15.12 Extremum Location Given the cubic function $CUBIC_{a,b,c,d}$, when

- Discriminant $[3ax^2 + 2bx + c] = 4b^2 - 12ac < 0$, $CUBIC$ has no locally extremum input.
- Discriminant $[3ax^2 + 2bx + c] = 4b^2 - 12ac = 0$, $CUBIC$ has one locally minimum-maximum input or one locally maximum-minimum input.
- Discriminant $[3ax^2 + 2bx + c] = 4b^2 - 12ac > 0$, $CUBIC$ has both

- $x_{\textit{locally minimum-output}}$,
- $x_{\textit{locally maximum-output}}$,

9 0-Height Location

The location of 0-height inputs in the case of a cubic function is usually not easy.

1. So far, the situation has been as follows:

i. The number of 0-height inputs for *affine functions* is always *one*,

ii. The number of 0-height inputs for *quadratic functions* is already more complicated in that, depending on the sign of the extreme-output compared with the sign of the outputs for inputs near ∞, it can be *none*, *one* or *two*.

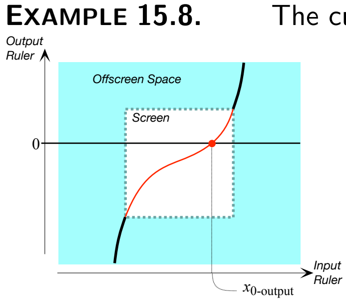
It follows from the **Extremum Location Theorem** that

iii. The number of 0-height inputs for *cubic functions* depends

a. On the Shape type of the cubic function,

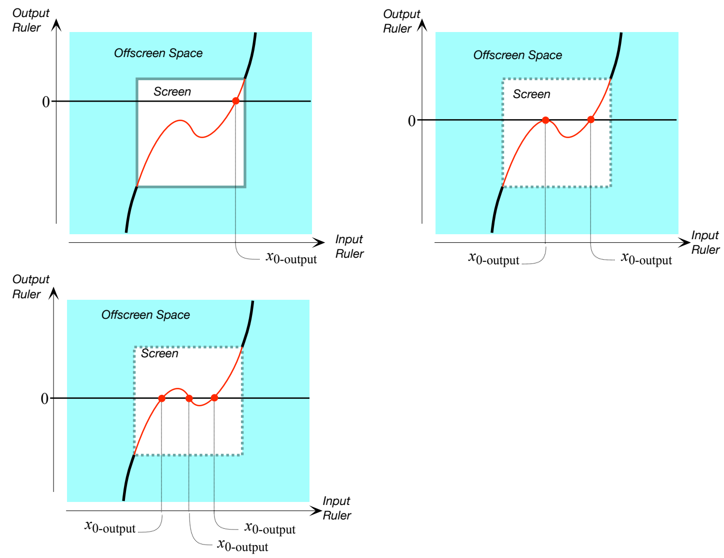
b. In the case of Shape type II, on the sign of the extremum outputs relative to the sign of the cubing coefficient

EXAMPLE 15.8. The cubic function specified by the global graph



is of Shape Type O (No 0-slope input) and always has a single 0-height input.

EXAMPLE 15.9. The cubic function specified by the global graphs are all of the same shape of Type II and the number of 0-height inputs depends on how high the graph is in relation to the 0-output level line.



2. The *obstruction* to computing the solutions that we encountered when trying to solve *quadratic equations*, namely that there was one more term than an equation has sides is even worse here since we have four terms and an equation still has only two sides. See **??** on **??**

Chapter 16

Rational Degree & Algebra Reviews

Rational functions are functions whose *global input-output rule* is of the form

$$x \xrightarrow{RAT} RAT(x) = \frac{POLY_{Num}(x)}{POLY_{Den}(x)}$$

where $POLY_{Num}(x)$ and $POLY_{Den}(x)$ stand for two *positive-exponent* polynomial expressions.

EXAMPLE 16.1. The function whose global input-output rule is

$$x \xrightarrow{TAB} TAB(x) = \frac{-3x^2 + 4x - 7}{-5x^4 - 8}$$

is a rational function in which:

- $POLY_{Num}(x)$ is $-3x^2 + 4x - 7$
- $POLY_{Den}(x)$ is $-5x^4 - 8$

1 Rational Degree

Because the *upper degree* of polynomial functions is what we used to sort polynomial functions into different *types*, we now try to extend the idea of

rational degree
regular rational function

upper degree to the case of rational functions in the hope that this will also help us sort rational functions into different *types*.

Given a rational function whose global input-output rule is

$$x \xrightarrow{RAT} RAT(x) = \frac{POLY_{Num}(x)}{POLY_{Den}(x)}$$

the **rational degree** of this rational function is the upper degree of $POLY_{Num}(x)$ minus the upper degree of $POLY_{Den}(x)$:

$$\text{Rat.Deg. of } \frac{POLY_{Num}(x)}{POLY_{Den}(x)} = \text{UppDeg. of } POLY_{Num}(x) - \text{UppDeg. of } POLY_{Den}(x)$$

Thus, the *rational degree* of a rational function can well be *negative*.

Note 16.1

The *rational degree* is to rational function very much what the *size* is to arithmetic fractions in "school arithmetic" which distinguishes fractions according to the *size* of the numerator compared to the *size* of the denominator even though, by now, the distinctions are only an inconsequential remnant of history..

What happened is that, historically, the earliest arithmetic fractions were "unit fractions" , that is reciprocals of whole numbers such as one half, one third, one quarter, etc. Later came "Egyptian fractions", that is combinations of (distinct) unit fractions, such as one third and one fifth and one eleventh, etc. A much later development were the "proper fractions", also called "vulgar fractions", such as two thirds, three fifths etc. Later still, came "improper fractions" such as five thirds, seven halves, etc. And finally "mixed numbers", such as three and two sevenths. Today, none of these distinctions matters inasmuch as we treat all fractions in the same manner.

However, while these "school arithmetic" distinctions are based on the *size* of the numerator versus the *size* of the denominator and make no real differences in the way we handle arithmetic fractions, in the case of rational functions, the above distinction based on the *upper degree* of the numerator versus the *upper degree* of the denominator will make a difference—even though no major one—in the way we will handle rational functions of different types.

In fact, by analogy with what we did with *power functions*, we will say that

- Rational functions whose rational degree is either > 1 or < 0, are **regular rational functions**,

- Rational functions whose rational degree is either $= 0$ or $= 1$, are **exceptional rational functions**.

exceptional rational function

EXAMPLE 16.2. Find the rational degree of the function $DOUGH$ whose global input-output rule is

$$x \xrightarrow{DOUGH} DOUGH(x) = \frac{+1x^4 - 6x^3 + 8x^2 + 6x - 9}{x^2 - 5x + 6}$$

Since the rational degree is given by

$$\text{Rat.Deg. of } \frac{POLY_{Num}(x)}{POLY_{Den}(x)} = \text{UppDeg. of } POLY_{Num}(x) - \text{UppDeg. of } POLY_{Den}(x)$$

and since, here,

- $POLY_{Num}(x) = +1x^4 - 6x^3 + 8x^2 + 6x - 9$
- $POLY_{Den}(x) = +1x^2 - 5x + 6$

we get from the definition of the upper degree of a polynomial that:

$$\begin{aligned} \text{UppDeg. of } +1x^4 - 6x^3 + 8x^2 + 6x - 9 &= \text{Exponent of Highest Term} \\ &= \text{Exponent of } +1x^4 \\ &= 4 \\ \text{UppDeg. of } +1x^2 - 5x + 6 &= \text{Exponent of Highest Term} \\ &= \text{Exponent of } +1x^2 \\ &= 2 \end{aligned}$$

so that the rational degree of the rational function $DOUGH$ is:

$$\begin{aligned} \text{Rat.Deg. of } \frac{+1x^4 - 6x^3 + 8x^2 + 6x - 9}{+1x^2 - 5x + 6} &= \text{Exponent of } +1x^4 - \text{Exponent of } +1x^2 \\ &= 4 - 2 \\ &= 2 \end{aligned}$$

so that $DOUGH$ is an example of a rational function of degree > 1 and therefore of a *regular* rational function.

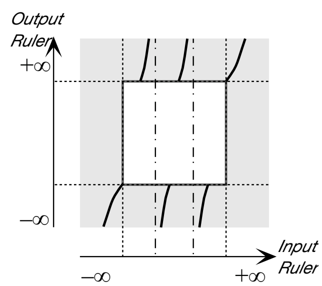
2 Graphic Difficulties

Finally, when there is one or more ∞-height bounded input(s), beginners often encounter difficulties when trying to interpolate smoothly the outlying graph of a rational function.

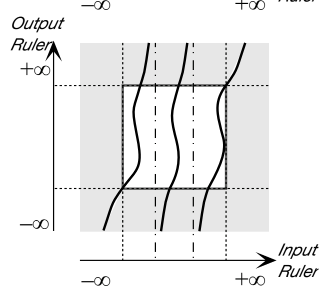
The difficulties are caused by the fact that, when we draw the local graph near ∞ and the local graphs near the ∞-height inputs from the local input-output rules, we are only concerned with drawing the local graphs themselves from the local input-output rules. In particular, when we draw the local graph near ∞ and the local graphs near the ∞-height inputs, we want to bend them enough to show the concavity but we often end up bending them *too much* to interpolate them.

But then, what often happens as a result is that, when we want to interpolate, the local graphs may not line up well enough for us to interpolate them (smoothly).

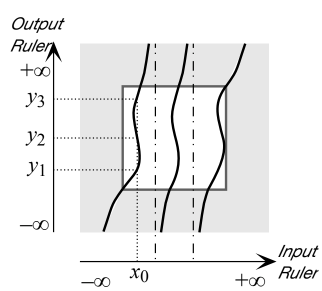
EXAMPLE 16.3.
Given the rational function whose offscreen graph was drawn so as to show the concavity.



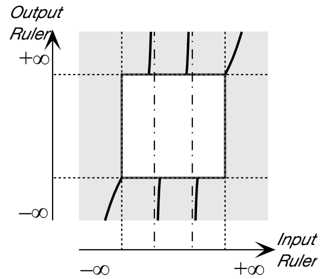
Here is what can happens when we attempt to interpolate



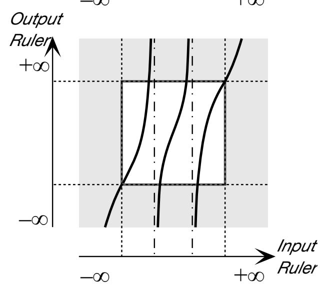
Of course, this is absolutely impossible since, according to this global graph, there would be inputs, such as x_0, with more than one output, y_1, y_2, . . . :



But if we unbend the local graphs just a bit as in

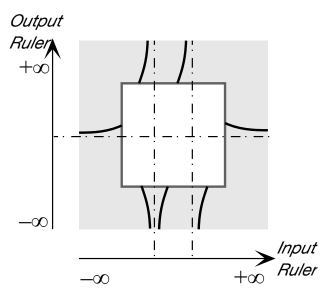


we have no trouble interpolating:

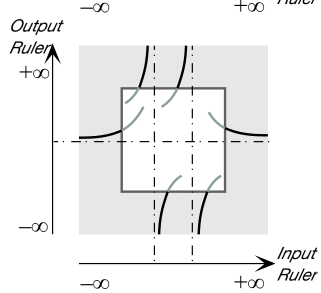


The way to avoid this difficulty is not to wait until we have to interpolate but to catch any problem as we draw the local graphs by mentally extending the local graphs slightly into the transitions.

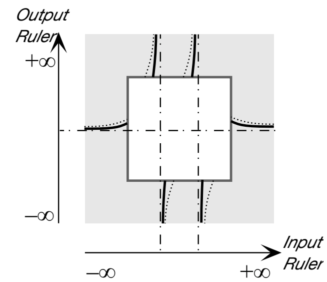
EXAMPLE 16.4.
Given the rational function whose offscreen graph was drawn do as to show the concavity



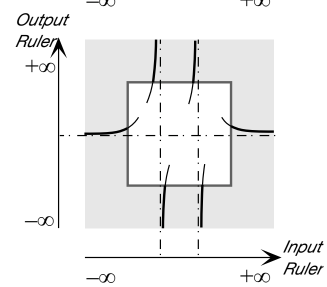
we can already see by extending the local graphs just a little bit into the transitions that this will cause a lot of trouble when we try to interpolate the local graph:



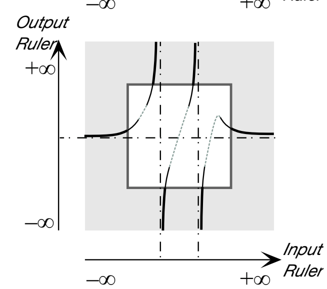
So, here, we bend the local graph near ∞ a little bit more and we unbend the local graphs near the ∞-height inputs a little bit:



We check again by extending the local graphs just a little bit into the transitions:



and indeed now we have no trouble interpolating:



extract

Chapter 17

Rational Functions: Local Analysis Near ∞

To do local analysis we work in a neighborhood of some given input and thus count inputs from the given input since it is the center of the neighborhood. When the given input is ∞, counting from ∞ means setting $x \leftarrow$ *large* and computing with powers of *large* in descending order of sizes.

Recall that the *principal term* near ∞ of a given polynomial function $POLY$ is simply its highest power term which is therefore easy to **extract** from the global input-output rule. The approximate input-output rule near ∞ of $POLY$ is then of the form

$$x|_{x \text{ near } \infty} \xrightarrow{POLY} POLY(x)|_{x \text{ near } \infty} = Highest\ Term\ POLY + [...]$$

However, the complication here is that to get the principal part near ∞ of a rational function we must approximate the two polynomial and divide—or the other way round—and the result need not be a polynomial but can also be a negative-exponent power function and the main issue will be whether to do the approximation before or after the division.

1 Local Input-Output Rule Near ∞

Given a rational function RAT, we look for the function whose input-output rule will be simpler than the input-output rule of RAT but whose local graph near ∞ will be qualitatively the same as the local graph near ∞ of RAT.

More precisely, given a rational function RAT specified by the global input-output rule

$$x \xrightarrow{RAT} RAT(x) = \frac{POLY_{Num}(x)}{POLY_{Den}(x)}$$

what we will want then is an *approximation* for the output of the local input-output rule near ∞

$$x|_{x \text{ near } \infty} \xrightarrow{RAT} RAT(x)|_{x \text{ near } \infty} = \left.\frac{POLY_{Num}(x)}{POLY_{Den}(x)}\right|_{x \text{ near } \infty}$$

from which to *extract* whatever controls the wanted feature.

1. Since the center of the neighborhood is ∞, we *localize* both

- $POLY_{Num}(x)$

and

- $POLY_{Den}(x)$

by writing them in *descending* order of exponents.

$$\frac{POLY_{Num}(x)}{POLY_{Den}(x)} \quad \begin{matrix} \xrightarrow{\text{Localize near } \infty} \\ \xrightarrow{\text{Localize near } \infty} \end{matrix} \quad \frac{POLY_{Num}(x)|_{x \text{ near } \infty}}{POLY_{Den}(x)|_{x \text{ near } \infty}}$$

2. Depending on the circumstances, we will take one of the following two routes to *extract* what controls the wanted feature:

■ The *short route* to $Princ.TERM\ RAT(x)\,|_{x \text{ near } \infty}$, that is:

i. We approximate both $POLY_{Num}(x)|_{x \text{ near } \infty}$ and $POLY_{Den}(x)|_{x \text{ near } \infty}$ to their *principal term*—that is to just their *highest size term*—which, since x is near ∞, is their *highest exponent term*:

$$\frac{POLY_{Num}(x)}{POLY_{Den}(x)} \quad \begin{matrix} \xrightarrow{\text{Localize near } \infty} \\ \xrightarrow{\text{Localize near } \infty} \end{matrix} \quad \frac{POLY_{Num}(x)|_{x \text{ near } \infty}}{POLY_{Den}(x)|_{x \text{ near } \infty}} \quad \begin{matrix} \xrightarrow{\text{i. Approximate}} \\ \xrightarrow{\text{i. Approximate}} \end{matrix} \quad \frac{Princ.TERM_{Num}(x)\,|_{x \text{ near } \infty} + [...]}{Princ.TERM_{Den}(x)|_{x \text{ near } \infty} + [...]}$$

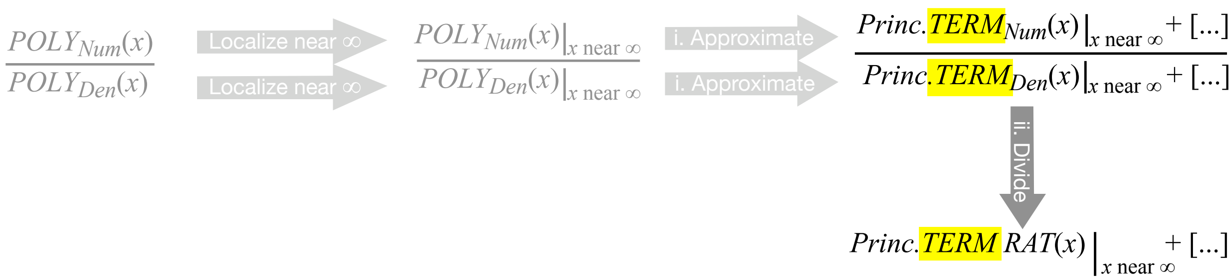
ii. In order to divide $Princ.TERM_{Num(x)}\big|_{x \text{ near } \infty}$, that is the principal term near ∞ of the *numerator* of RAT by $Princ.TERM_{Den(x)}\big|_{x \text{ near } \infty}$, that is the principal term near ∞ of the *denominator* of RAT we use monomial division

$\boxed{\dfrac{ax^{+m}}{bx^{+n}} = \dfrac{a}{b}x^{+m\ominus+n}}$ where $+m \ominus +n$ can turn out positive, negative or 0

$$\begin{aligned} Princ.TERM\ RAT(x)\Big|_{x \text{ near } \infty} &= \frac{Princ.TERM_{Num}(x)\Big|_{x \text{ near } \infty}}{Princ.TERM_{Den}(x)\Big|_{x \text{ near } \infty}} \\ &= \frac{\text{coef. } Princ.TERM_{Num}(x)\Big|_{x \text{ near } \infty}}{\text{coef. } Princ.TERM_{Den}(x)\Big|_{x \text{ near } \infty}} \cdot x^{\text{UppDeg.}POLY_{Num}(x) - \text{UppDeg.}POLY_{Den}(x)} \end{aligned}$$

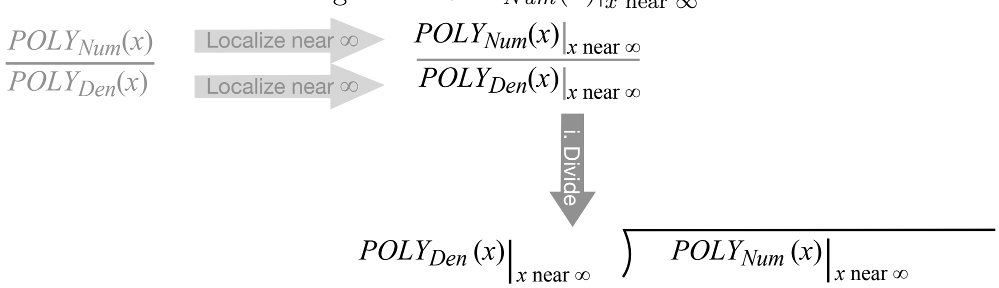
$$= \frac{\text{coef. } Princ.TERM_{Num}(x)\big|_{x \text{ near } \infty}}{\text{coef. } Princ.TERM_{Den}(x)\big|_{x \text{ near } \infty}} \cdot x^{\text{RatDeg.}RAT(x)}$$

The resulting monomial is $Princ.TERM\,RAT(x)\big|_{x \text{ near } \infty}$, that is the *principal term* of the rational function RAT near ∞:

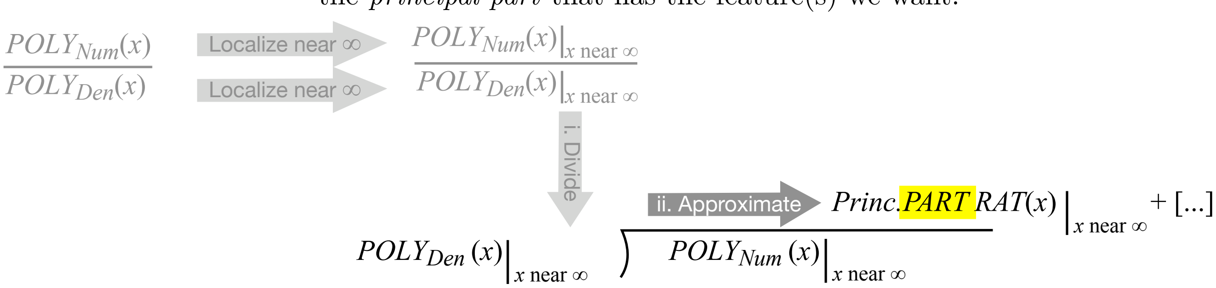


■ The *long route* to $Princ.PART\ RAT(x)\ |_{x \text{ near } \infty}$:

i. In order to divide $POLY_{Num}(x)|_{x \text{ near } \infty}$ *by* $POLY_{Den}(x)|_{x \text{ near } \infty}$, we set up the division as a *long division*, that is $POLY_{Den}(x)|_{x \text{ near } \infty}$ dividing *into* $POLY_{Num}(x)|_{x \text{ near } \infty}$:



ii. We approximate by stopping the long division as soon as we have the *principal part* that has the feature(s) we want:



3. Which route we will take in each particular case will depend both on the *wanted feature(s)* near ∞ and on the *rational degree* of RAT and so we will now look separately at how we get $Height$-sign$|_{x \text{ near } \infty}$, $Slope$-sign$|_{x \text{ near } \infty}$ and $Concavity$-sign$|_{x \text{ near } \infty}$

LOCAL ANALYSIS NEAR ∞

When the wanted features are to be found near ∞, the *rational degree* of the rational function tells us up front whether or not the *short route* will allow us to extract the term that controls the wanted feature.

2 Height-sign Near ∞

No matter what the *rational degree* of the given rational function RAT, $Princ.TERM\ RAT(x)\ |_{x \text{ near } \infty}$ will give us $Height$-sign$|_{x \text{ near } \infty}$ because, no matter what its exponent, *any* power function has $Height$-sign$|_{x \text{ near } \infty}$. So, no matter what the *rational degree* of RAT, to *extract* the term responsible for $Height$-sign$|_{x \text{ near } \infty}$ we can take the *short route* to $Princ.TERM\ RAT(x)\ |_{x \text{ near } \infty}$:

$$\left.\frac{POLY_{Num}(x)}{POLY_{Den}(x)}\right|_{x \text{ near } \infty} \xrightarrow{\text{Localize near } \infty} \frac{POLY_{Num}(x)|_{x \text{ near } \infty}}{POLY_{Den}(x)|_{x \text{ near } \infty}} \xrightarrow{\text{i. Approximate}} \frac{Princ.TERM_{Num}(x)|_{x \text{ near } \infty} + [...]}{Princ.TERM_{Den}(x)|_{x \text{ near } \infty} + [...]}$$

$$\xrightarrow{\text{ii. Divide}} Princ.TERM\ RAT(x)\Big|_{x \text{ near } \infty} + [...]$$

EXAMPLE 17.1. Given the rational function $DOUGH$ specified by the global input-output rule

$$x \xrightarrow{DOUGH} DOUGH(x) = \frac{+12x^5 - 6x^3 + 8x^2 + 6x - 9}{-3x^2 - 5x + 6}$$

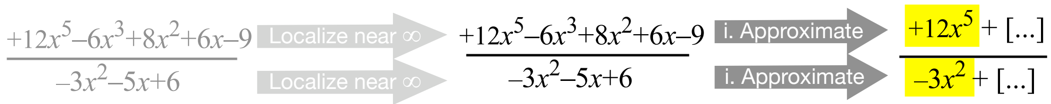
find $Height$-sign $DOUGH|_{x \text{ near } \infty}$.

a. We localize both the numerator and the denominator near ∞—which amounts only to making sure that the terms are in *descending order of exponents*.

$$\frac{+12x^5-6x^3+8x^2+6x-9}{-3x^2-5x+6} \xrightarrow{\text{Localize near } \infty} \frac{+12x^5-6x^3+8x^2+6x-9}{-3x^2-5x+6}$$

b. Inasmuch as $Princ.TERM\ DOUGH(x)\ |_{x \text{ near } \infty}$ has *Height* no matter what the degree, in order to *extract* the term that controls $Height$-sign$|_{x \text{ near } \infty}$ we take the short route to $Princ.TERM\ DOUGH(x)\ |_{x \text{ near } \infty}$:

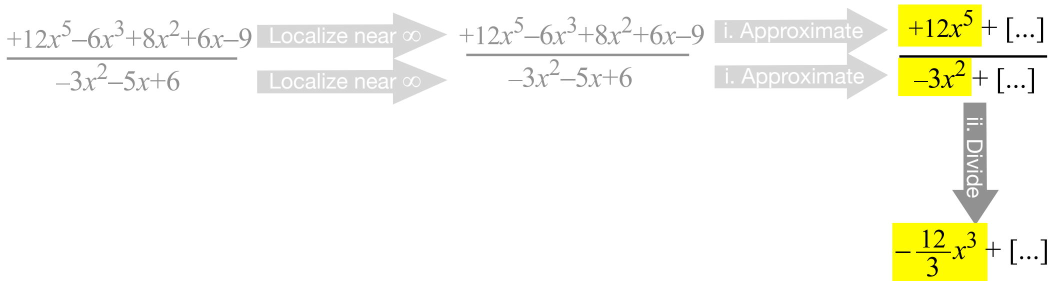
i. We *approximate*



that is we approximate

- the numerator $+12x^5 - 6x^3 + 8x^2 + 6x - 9$ to its *principal term*, $-12x^5$
- the denominator $-3x^2 - 5x + 6$ to its *principal term*, $-3x^2$

ii. And then we *divide*:



where

$$\begin{aligned}\frac{+12x^5}{-3x^2} &= \frac{+12 \cdot x \cdot x \cdot x \cdot x \cdot x}{-3 \cdot x \cdot x} \\ &= -\frac{12}{3}x^{5-2}\end{aligned}$$

The more usual way to write all this is something as follows:

$$\begin{aligned} x|_{x \text{ near } \infty} \xrightarrow{DOUGH} DOUGH(x)|_{x \text{ near } \infty} &= \left.\frac{+12x^5 - 6x^3 + 8x^2 + 6x - 9}{-3x^2 - 5x + 6}\right|_{x \text{ near } \infty} \\ &= \frac{+12x^5 - 6x^3 + 8x^2 + 6x - 9|_{x \text{ near } \infty}}{-3x^2 - 5x + 6|_{x \text{ near } \infty}} \\ &= \frac{+12x^5 - 6x^3 + 8x^2 + 6x - 9}{-3x^2 - 5x + 6} \\ &= \frac{+12x^5 + [...]}{-3x^2 + [...]} \\ &= -\frac{12}{3}x^{5-2} + [...] \end{aligned}$$

Whatever we write, the *principat term* of $DOUGH$ near ∞ is $-\frac{12}{3}x^3$ and it gives

$$Height\text{-sign } DOUGH|_{x \text{ near } \infty} = (-, +)$$

EXAMPLE 17.2. Given the function PAC specified by the global input-output rule

$$x \xrightarrow{PAC} PAC(x) = \frac{-12x^3 + 7x + 4}{+4x^5 - 6x^4 - 17x^2 - 2x + 10}$$

find Height-sign $PAC|_{x \text{ near } \infty}$.
Inasmuch as $Princ.TERM\ PAC(x)\ |_{x \text{ near } \infty}$ has *Height* no matter what the degree, in order to *extract* the term that controls $Height$-sign$|_{x \text{ near } \infty}$ we take the short route to $Princ.TERM\ DOUGH(x)\ |_{x \text{ near } \infty}$:

$$\begin{aligned} x|_{x \text{ near } \infty} \xrightarrow{PAC} PAC(x)|_{x \text{ near } \infty} &= \left.\frac{-12x^3 + 7x + 4}{+4x^5 - 6x^4 - 17x^2 - 2x + 10}\right|_{x \text{ near } \infty} \\ &= \frac{-12x^3 + 7x + 4|_{x \text{ near } \infty}}{+4x^5 - 6x^4 - 17x^2 - 2x + 10|_{x \text{ near } \infty}} \\ &= \frac{-12x^{+3} + [...]}{+4x^{+5} + [...]} \\ &= \frac{-12}{+4} x^{+3 \ominus +5} + [...] \\ &= -3x^{-2} + [...] \end{aligned}$$

and we get that

$$\text{Height-sign } PAC|_{x \text{ near } \infty} = (-, -)$$

3 Slope-sign Near ∞

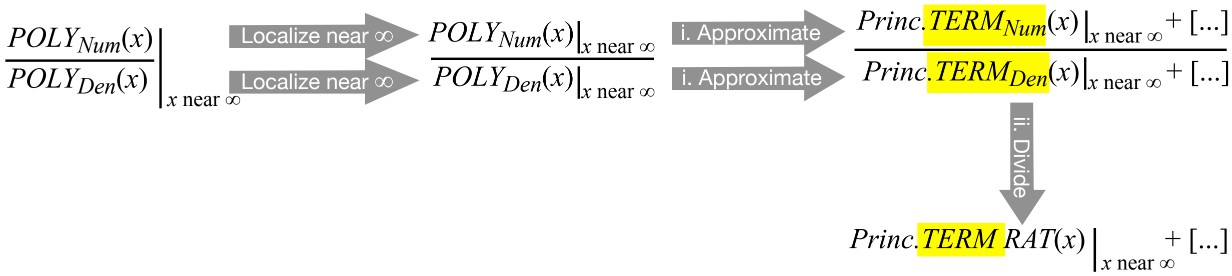
In the case of $Slope$-sign $RAT|_{x \text{ near } \infty}$, there are two cases depending on the *rational degree* of the given rational function:

- If the rational functionRAT is either:
  - A *regular* rational function, that is of rational degree > 1 or < 0

  or

  - An *exceptional* rational function of rational degree $= 1$,

  that is *not* an exceptional rational function of rational degree $= 0$, then $Princ.\ TERM\ RAT(x)\Big|_{x \text{ near } \infty}$ will be a *power function* that will have *Slope* near ∞ and so in order to *extract* the term that controls $Slope$-sign$|_{x \text{ near } \infty}$ we take the short route to $Princ.\ TERM\ RAT(x)\Big|_{x \text{ near } \infty}$:



EXAMPLE 17.3. Given the rational function $SOUTH$ specified by the global input-output rule

$$x \xrightarrow{SOUTH} SOUTH(x) = \frac{-3x^2 - 5x + 6}{+12x^4 - 6x^3 + 8x^2 + 6x - 9}$$

find Slope-sign of $SOUTH$ near ∞

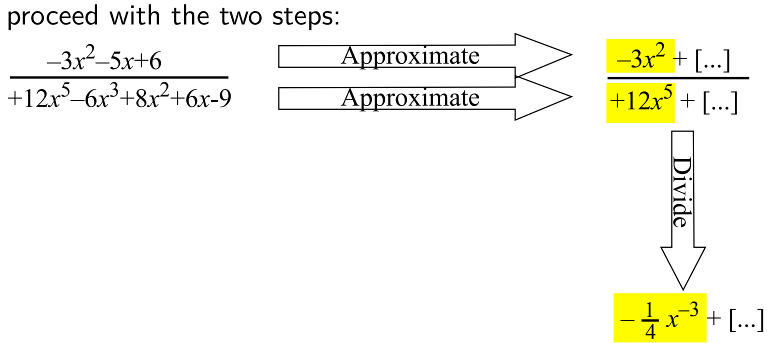
i. We get the local graph near ∞ of $SOUTH$

a. We have

$$x|_{x \text{ near } \infty} \xrightarrow{SOUTH} SOUTH(x)|_{x \text{ near } \infty} = \left.\frac{-3x^2 - 5x + 6}{+12x^5 - 6x^3 + 8x^2 + 6x - 9}\right|_{x \text{ near } \infty}$$

$$= \frac{-3x^2 - 5x + 6|_{x \text{ near } \infty}}{+12x^5 - 6x^3 + 8x^2 + 6x - 9|_{x \text{ near } \infty}}$$

We now proceed with the two steps:



b. The more usual presentation is:

$$x|_{x \text{ near } \infty} \xrightarrow{SOUTH} SOUTH(x)|_{x \text{ near } \infty} = \left.\frac{-3x^2 - 5x + 6}{+12x^5 - 6x^3 + 8x^2 + 6x - 9}\right|_{x \text{ near } \infty}$$

$$= \frac{-3x^2 - 5x + 6|_{x \text{ near } \infty}}{+12x^5 - 6x^3 + 8x^2 + 6x - 9|_{x \text{ near } \infty}}$$

We *approximate* $-3x^2 - 5x + 6\big|_{x \text{ near } \infty}$ and $+12x^5 - 6x^3 + 8x^2 + 6x - 9\big|_{x \text{ near } \infty}$

$$= \frac{-3x^2 + [\ldots]}{+12x^5 + [\ldots]}$$

and then we *divide*:

$$= \frac{-3}{+12}x^{2-5} + [\ldots]$$

$$= -\frac{1}{4}x^{-3} + [\ldots]$$

c. Since the degree of the power function

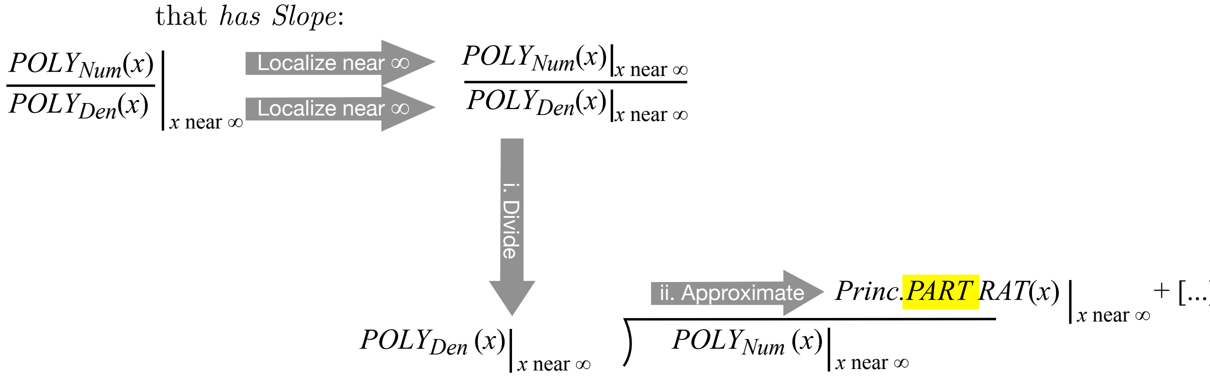
$$x \xrightarrow{POWER} POWER(x) = -\frac{1}{4}x^{-3}$$

which approximates $SOUTH$ near ∞ is < 0, the power function $POWER$ has all three features, *concavity*, *slope* and *height*. (This was of course to be expected from the fact that the *rational degree* of $SOUTH$ is < 0.)

ii. We get

$$\text{Slope-sign of } SOUTH \text{ near } \infty = (\diagup, \diagdown)$$

■ If the rational function RAT *is* an *exceptional rational function* whose rational degree $= 0$, then $Princ.TERM\ RAT(x)\ |_{x \text{ near } \infty}$ will be an *exceptional power function* with exponent $= 0$ and $Princ.TERM\ RAT(x)\ |_{x \text{ near } \infty}$ will *not* have *Slope* and so in order to *extract* the term that controls *Slope*-sign$|_{x \text{ near } \infty}$ we will have to take the long route to a $Princ.PART\ RAT(x)\ |_{x \text{ near } \infty}$ that *has Slope*:



4 Concavity-sign Near ∞

In the case of *Concavity*-sign $RAT|_{x \text{ near } \infty}$, there are *two* cases depending on the rational degree of the given rational function.

■ If the rational function RAT is a *regular* rational function, that is if the rational degree of RAT is either > 1 or < 0, then $Princ.\ TERM\ RAT(x)\ |_{x \text{ near } \infty}$ will be a *regular power function*, that is a power function whose exponent is either > 1 or < 0 and then, in either case, $Princ.\ TERM\ RAT(x)\ |_{x \text{ near } \infty}$ will have *Concavity* and so in order to *extract* the term that controls $Concavity\text{-sign}|_{x \text{ near } \infty}$ we take the short route to $Princ.TERM_{Den}(x)\ |_{x \text{ near } \infty}$:

$$\left.\frac{POLY_{Num}(x)}{POLY_{Den}(x)}\right|_{x \text{ near } \infty} \xrightarrow{\text{Localize near } \infty} \frac{POLY_{Num}(x)|_{x \text{ near } \infty}}{POLY_{Den}(x)|_{x \text{ near } \infty}} \xrightarrow{\text{i. Approximate}} \frac{Princ.TERM_{Num}(x)|_{x \text{ near } \infty} + [...]}{Princ.TERM_{Den}(x)|_{x \text{ near } \infty} + [...]}$$

$$\xrightarrow{\text{ii. Divide}} Princ.TERM\ RAT(x)\Big|_{x \text{ near } \infty} + [...]$$

EXAMPLE 17.4. Given the rational function $SOUTH$ specified by the global input-output rule

$$x \xrightarrow{SOUTH} SOUTH(x) = \frac{-3x^2 - 5x + 6}{+12x^4 - 6x^3 + 8x^2 + 6x - 9}$$

find Concavity-sign of $SOUTH$ near ∞

i. We get the local graph near ∞ of $SOUTH$

a. We have

$$x|_{x \text{ near } \infty} \xrightarrow{SOUTH} SOUTH(x)|_{x \text{ near } \infty} = \left.\frac{-3x^2 - 5x + 6}{+12x^5 - 6x^3 + 8x^2 + 6x - 9}\right|_{x \text{ near } \infty}$$

$$= \frac{-3x^2 - 5x + 6|_{x \text{ near } \infty}}{+12x^5 - 6x^3 + 8x^2 + 6x - 9|_{x \text{ near } \infty}}$$

We now proceed with the two steps:

$$\frac{-3x^2-5x+6}{+12x^5-6x^3+8x^2+6x-9} \xrightarrow{\text{Approximate}} \frac{-3x^2 + [...]}{+12x^5 + [...]} \xrightarrow{\text{Divide}} -\frac{1}{4}x^{-3} + [...]$$

b. The more usual presentation is:

$$x\big|_{x \text{ near } \infty} \xrightarrow{SOUTH} SOUTH(x)\big|_{x \text{ near } \infty} = \left.\frac{-3x^2 - 5x + 6}{+12x^5 - 6x^3 + 8x^2 + 6x - 9}\right|_{x \text{ near } \infty}$$
$$= \frac{-3x^2 - 5x + 6\big|_{x \text{ near } \infty}}{+12x^5 - 6x^3 + 8x^2 + 6x - 9\big|_{x \text{ near } \infty}}$$

We *approximate* $-3x^2 - 5x + 6\big|_{x \text{ near } \infty}$ and $+12x^5 - 6x^3 + 8x^2 + 6x - 9\big|_{x \text{ near } \infty}$

$$= \frac{-3x^2 + [...]}{+12x^5 + [...]}$$

and then we *divide*:

$$= \frac{-3}{+12}x^{2-5} + [...]$$
$$= -\frac{1}{4}x^{-3} + [...]$$

c. Since the degree of the power function

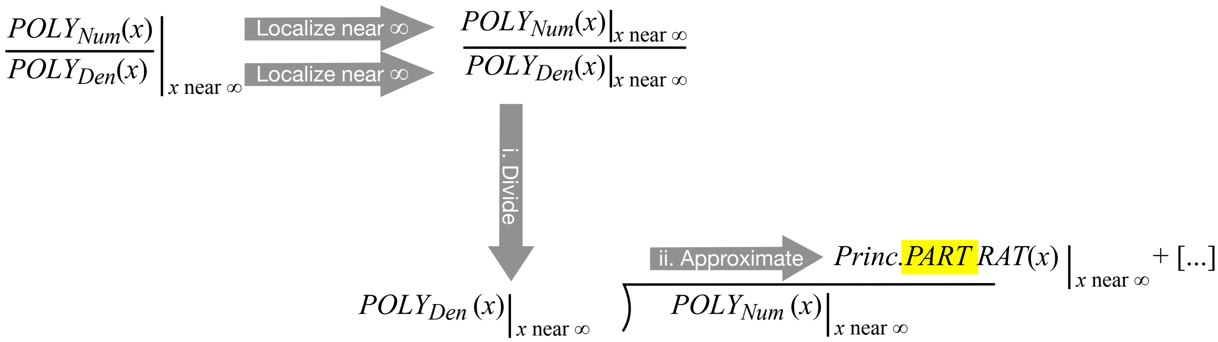
$$x \xrightarrow{POWER} POWER(x) = -\frac{1}{4}x^{-3}$$

which approximates $SOUTH$ near ∞ is < 0, the power function $POWER$ has all three features, *concavity*, *slope* and *height*. (This was of course to be expected from the fact that the *rational degree* of $SOUTH$ is < 0.)

ii. We get

$$\text{Concavity-sign of } SOUTH \text{ near } \infty = (\cap, \cap)$$

■ If the rational function RAT is an *exceptional* rational function that is if the rational degree of RAT is either $= 1$ or $= 0$ then $Princ.TERM\ RAT(x)\ |_{x \text{ near } \infty}$ will be an *exceptional power function* with exponent either $= 1$ or $= 0$ (**Chapter 7**) and in both cases $Princ.TERM\ RAT(x)\ |_{x \text{ near } \infty}$ will *not* have *Concavity* and in order *extract* the term that controls $Concavity$-sign$|_{x \text{ near } \infty}$ we will have to take the long route to a $Princ.PART\ RAT(x)\ |_{x \text{ near } \infty}$ that does have *Concavity*.

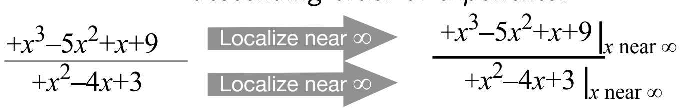


EXAMPLE 17.5. Given the rational function $BATH$ specified by the global input-output rule

$$x \xrightarrow{BATH} BATH(x) = \frac{+x^3 - 5x^2 + x + 6}{+x^2 - 4x + 3}$$

find $Concavity$-sign $BATH|_{x \text{ near } \infty}$.

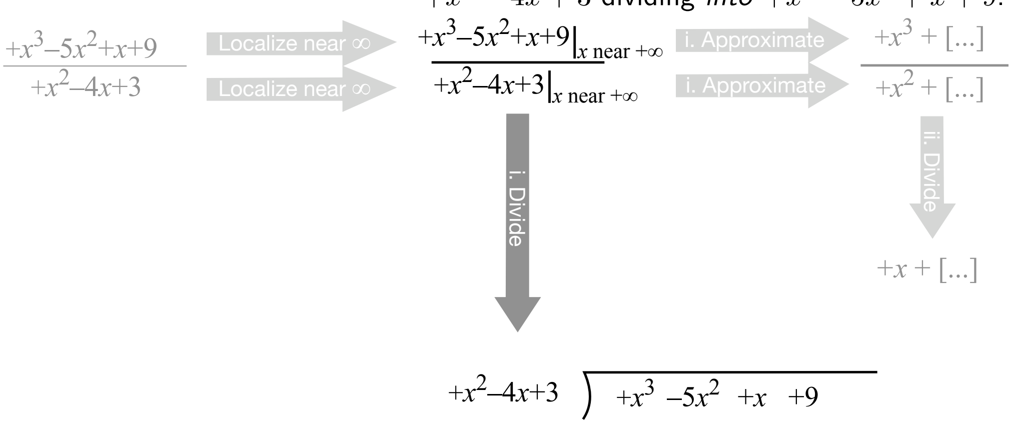
a. The *localization step* is to *localize* both the numerator and the denominator near ∞—which amounts only to making sure that the terms are in *descending order of exponents*.



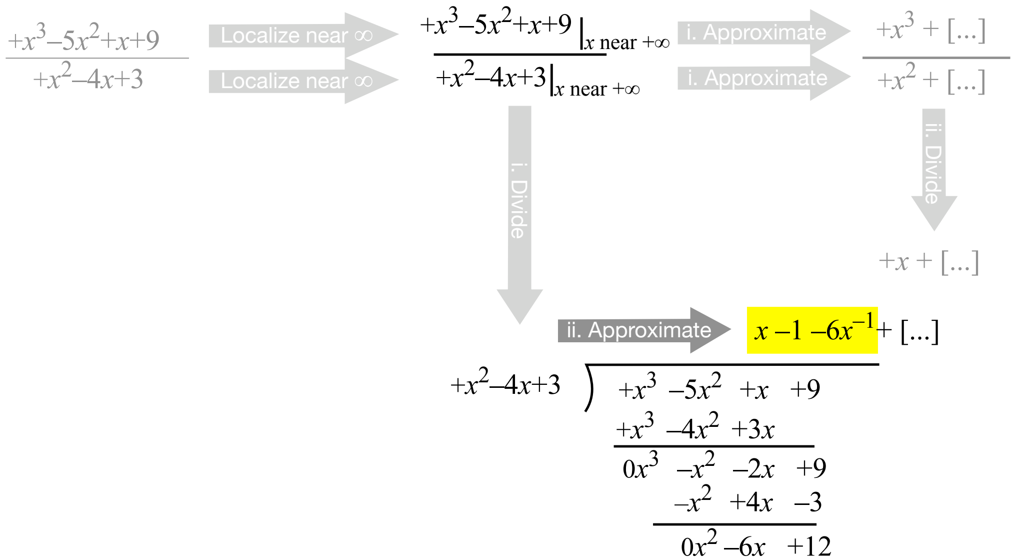
b. Since $Princ.\ TERM\ BATH(x)\ |_{x \text{ near } \infty}$ has no *Concavity*, the *extraction step* to get $Concavity$-sign $BATH|_{x \text{ near } \infty}$ must take the long route to a $Princ.\ PART\ BATH(x)\ |_{x \text{ near } \infty}$ that has *Concavity*:

i. We set up the division as a *long division*:

$+x^2 - 4x + 3$ dividing *into* $+x^3 - 5x^2 + x + 9$.



ii. We *approximate* by stopping the long division as soon as we have the *principal part* of the quotient that has *Concavity*:



that is we stop with $-6x^{-1}$ since it is the term responsible for *Concavity*. The more usual way to write all this is:

$$
\begin{aligned}
x|_{x \text{ near } \infty} \xrightarrow{BATH} BATH(x)|_{x \text{ near } \infty} &= \left.\frac{+x^3-5x^2+x+9}{+x^2-4x+3}\right|_{x \text{ near } \infty} \\
&= \frac{+x^3-5x^2+x+9|_{x \text{ near } \infty}}{+x^2-4x+3|_{x \text{ near } \infty}} \\
&= \frac{+x^3-5x^2+x+9}{+x^2-4x+3}
\end{aligned}
$$

and then we *divide* (in the *latin* manner):

$$
\begin{array}{r|llll}
 & +x & -1 & -6x^{-1} & +[...] \\
\hline
+x^2 \ -4x \ +3 \ ) & +x^3 & -5x^2 & +x & +9 \\
 & +x^3 & -4x^2 & +3x & \\
\cline{2-4}
 & 0x^3 & -x^2 & -2x & +9 \\
 & & -x^2 & +4x & -3 \\
\cline{3-5}
 & & 0x^2 & -6x & +12
\end{array}
$$

Whichever way we write it, $Princ.\,PART\ BATH(x)\,|_{x \text{ near } \infty} = +x-1-6x^{-1}$ and its third term, $-6x^{-1}$, gives

$$
Concavity\text{-sign } BATH|_{x \text{ near } \infty} = (\cap, \cup)
$$

5 Local Graph Near ∞

In order to get the local graph near ∞, we need a local input-output rule that gives us the *concavity*-signÑand therefore the *slope*-sign and the *height*-sign.

So, the route we must take in order to get the local graph near ∞ is the route that will get us the concavity-sign near ∞.

EXAMPLE 17.6. Given the rational function $SOUTH$ whose global input-output rule is

$$x \xrightarrow{SOUTH} SOUTH(x) = \frac{-3x^2 - 5x + 6}{+12x^4 - 6x^3 + 8x^2 + 6x - 9}$$

find its local graph near ∞.

i. We get the *local input-output rule* near ∞ as in EXAMPLE 1.

We have:

$$x|_{x \text{ near } \infty} \xrightarrow{SOUTH} SOUTH(x)|_{x \text{ near } \infty} = \left. \frac{-3x^2 - 5x + 6}{+12x^5 - 6x^3 + 8x^2 + 6x - 9} \right|_{x \text{ near } \infty}$$

$$= \frac{-3x^2 - 5x + 6|_{x \text{ near } \infty}}{+12x^5 - 6x^3 + 8x^2 + 6x - 9|_{x \text{ near } \infty}}$$

We *approximate* separately the *numerator* and the *denominator*:

$$= \frac{-3x^2 + [...]}{+12x^5 + [...]}$$

We *divide* the approximations:

$$= \frac{-3}{+12} x^{2-5} + [...]$$

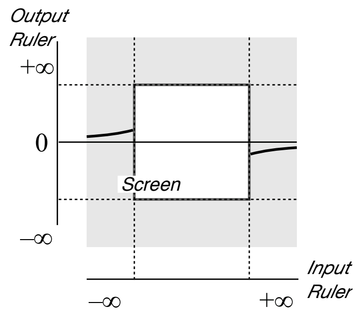
$$= -\frac{1}{4} x^{-3} + [...]$$

ii. Since the degree of the power function

$$x \xrightarrow{POWER} POWER(x) = -\frac{1}{4} x^{-3}$$

is < 0, the power function $POWER$ is *regular* and has both *concavity* and *slope*. So, the local graph of the power function $POWER$ near ∞ will be approximately the graph near ∞ of the rational function $SOUTH$.

The local graph near ∞ of the rational function $SOUTH$ is therefore:



EXAMPLE 17.7. Given the rational function $DOUGH$ whose global input-output rule is

$$x \xrightarrow{DOUGH} DOUGH(x) = \frac{+12x^4 - 6x^3 + 8x^2 + 6x - 9}{-3x^2 - 5x + 6}$$

find its local graph near ∞.

i. We get the *local input-output rule* near ∞.

We have:

$$x|_{x \text{ near } \infty} \xrightarrow{DOUGH} DOUGH(x)|_{x \text{ near } \infty} = \left.\frac{+12x^5 - 6x^3 + 8x^2 + 6x - 9}{-3x^2 - 5x + 6}\right|_{x \text{ near } \infty}$$

$$= \frac{+12x^5 - 6x^3 + 8x^2 + 6x - 9|_{x \text{ near } \infty}}{-3x^2 - 5x + 6|_{x \text{ near } \infty}}$$

We *approximate* separately the *numerator* and the *denominator*:

$$= \frac{+12x^5 + [...]}{-3x^4 + [...]}$$

We *divide* the approximations:

$$= -\frac{+12}{-3}x^{5-2} + [...]$$

$$= -4x^{+3} + [...]$$

ii. Since the degree of the power function

$$x \xrightarrow{POWER} POWER(x) = -4x^{+3}$$

is > 1, the power function $POWER$ is *regular* and has both *concavity* and *slope*. So, the local graph of the power function $POWER$ near ∞ will be approximately the graph near ∞ of the rational function $DOUGH$.

The local graph near ∞ of the rational function $DOUGH$ is therefore:

EXAMPLE 17.8. Given the rational function $BATH$ specified by the global input-output rule

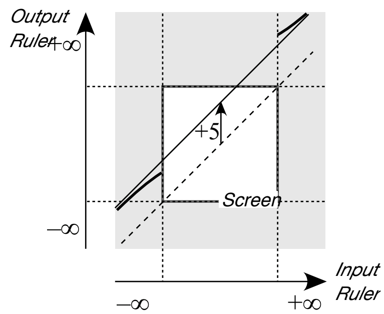
$$x \xrightarrow{BATH} BATH(x) = \frac{+x^3 + x^2 - 5x + 6}{+x^2 - 4x + +3}$$

as in EXAMPLE 1, find the local graph near ∞.

i. We get the *local input-output rule* near ∞ that gives all three features as we did in EXAMPLE 1:

$x|_{x \text{ near } \infty} \xrightarrow{BATH} BATH(x)|_{x \text{ near } \infty} = +x + 5 + 27x^{-1} + [...]$

ii. So the local graph near ∞ of the function $BATH$ is

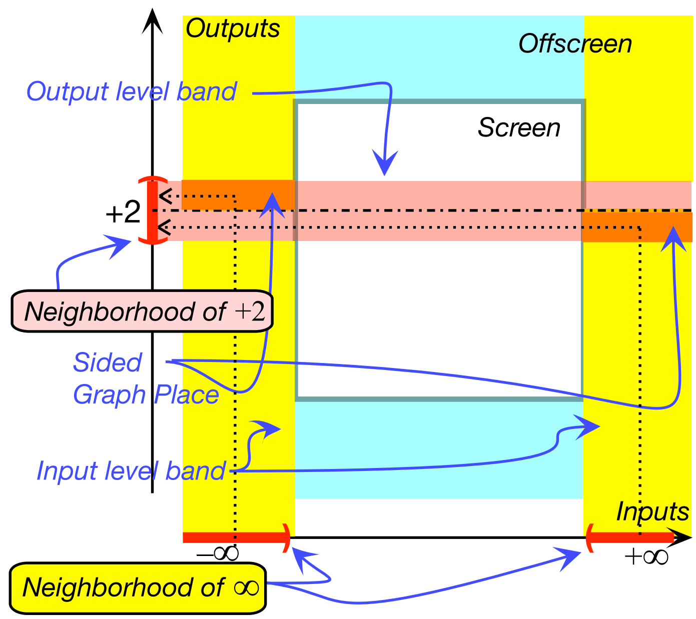


Chapter 18

Rational Functions: Local Analysis Near x_0

Doing local analysis means working in a neighborhood of some given input and thus counting inputs from the given input since it is the *center* of the neighborhood. When the given input is x_0, we *localize* at x_0, that is we set $x = x_0 + h$ where h is *small* and we compute with powers of h in descending order of sizes.

EXAMPLE 18.1. Given the input $+2$, then the location of the number $+2.3$ relative to $+2$ is $+0.3$:



Recall that the *principal part* near x_0 of a given polynomial function $POLY$ is the local quadratic part

$x|_{x \text{ near } x_0} \xrightarrow{POLY} POLY(x)|_{x \text{ near } x_0} = \Big[\quad\Big] + \Big[\quad\Big]h + \Big[\quad\Big]h^2 + [...]$

However, the complication here is that to get the principal part near x_0 of a rational function we must approximate the two polynomial and divide—or the other way round—and the result need not be a polynomial but can also be a negative-exponent power function and the main issue will be whether to do the approximation before or after the division.

1 Local Input-Output Rule Near x_0

Given a rational function RAT, we look for the function whose input-output rule will be simpler than the input-output rule of RAT but whose local graph near x_0 will be qualitatively the same as the local graph near x_0 of RAT.

More precisely, given a rational function RAT specified by the global input-output rule

$$x \xrightarrow{RAT} RAT(x) = \frac{POLY_{Num}(x)}{POLY_{Den}(x)}$$

what we will want then is an *approximation* for the output of the local input-output rule near x_0

$$x\big|_{x \text{ near } x_0} \xrightarrow{RAT} RAT(x)\big|_{x \text{ near } x_0} = \frac{POLY_{Num}(x)}{POLY_{Den}(x)}\bigg|_{x \text{ near } x_0}$$

from which to *extract* whatever controls the wanted feature.

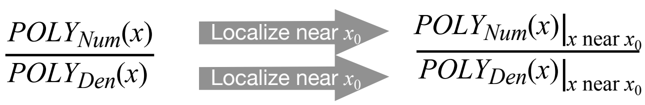
1. Since the center of the neighborhood is x_0, we *localize* both

- $POLY_{Num}(x)$

and

- $POLY_{Den}(x)$

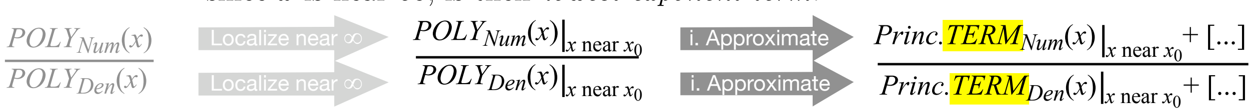
by letting $x \leftarrow x_0 + h$ and writing the terms in *ascending* order of exponents.



2. Depending on the circumstances, we will take one of the following two routes to *extract* what controls the wanted feature:

■ The *short route* to $Princ.TERM\ RAT(x)\ |_{x \text{ near } x_0}$, that is:

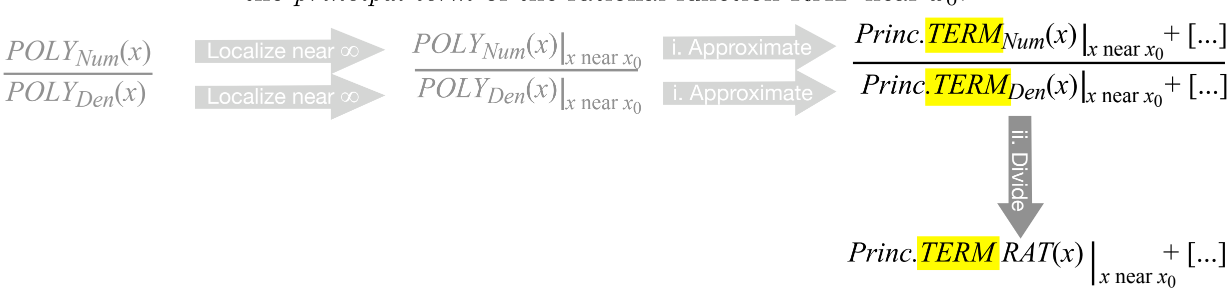
i. We approximate both $POLY_{Num}(x)|_{x \text{ near } x_0}$ and $POLY_{Den}(x)|_{x \text{ near } x_0}$ to their *principal term*—that is to just their *lowest size term*—which, since x is near ∞, is their *lowest exponent term*:



ii. In order to divide $Princ.TERM_{Num(x)}\Big|_{x \text{ near } x_0}$, that is the principal term near x_0 of the *numerator* of RAT by $Princ.TERM_{Den(x)}\Big|_{x \text{ near } x_0}$, that is the principal term near x_0 of the *denominator* of RAT we use monomial division

$\boxed{\dfrac{ah^{+m}}{bh^{+n}} = \dfrac{a}{b}h^{+m \ominus +n}}$ where $+m \ominus +n$ can turn out positive, negative or 0

The resulting monomial is $Princ.TERM\,RAT(x)\Big|_{x \text{ near } x_0}$, that is the *principal term* of the rational function RAT near x_0.



However, $Princ.TERM\,RAT(x)\Big|_{x \text{ near } x_0}$ is useful only in four cases:

- When it is a constant term *and* what we want is the Height-sign,
- When it is a linear term *and* what we want is the Height-sign or the Slope-sign,
- When it is a square term,
- When it is a negative-exponent term.

■ The *long route* to $Princ.PART\ RAT(x)\ |_{x \text{ near } x_0}$:

i. In order to divide $POLY_{Num}(x)|_{x \text{ near } x_0}$ *by* $POLY_{Den}(x)|_{x \text{ near } x_0}$, we set up the division as a *long division*, that is $POLY_{Den}(x)|_{x \text{ near } x_0}$ dividing *into* $POLY_{Num}(x)|_{x \text{ near } x_0}$ and since these are polynomials in h, in order to be in order of descending sizes, they must be in order of ascending exponents.

ii. We approximate by stopping the long division as soon as we have the *principal part* that has the feature(s) we want:

iii. The difficulty will be that we will have to approximate at two different stages:

- While we localize both the numerator and the denominator,
- When we divide the approximate localization of the numerator by the approximate localization of the denominator

So, we will have to make sure that the approximations in the localizations of the numerator and the denominator do not interfere with the approximation in the division, that is that, as we divide, we do not want to bump into a [...] coming from having approximated the numerator and the denominator too much, that is before we can extract from the division the term that controls the wanted feature.

3. Which route we will take in each particular case will depend both on the *wanted feature(s)* near x_0 and so we will now look separately at how we get *Height*-sign$|_{x \text{ near } \infty}$, *Slope*-sign$|_{x \text{ near } x_0}$ and *Concavity*-sign$|_{x \text{ near } x_0}$

LOCAL ANALYSIS NEAR x_0

When the wanted features are to be found near x_0, the *rational degree* of the rational function does not tell us which of the *short route* or the *long route* will allow us to extract the term that controls the wanted feature.

2 Height-sign Near x_0

If all we want is the Height-sign, then we can always go the short route.

Example 18.2. Let $SOUTH$ be the function specified by the global input-output rule

$$x \xrightarrow{SOUTH} SOUTH(x) = \frac{x^2+5x+6}{x^4-x^3-10x^2+x-15}$$

Find the height-sign of $SOUTH$ near $+2$

i. We localize both the numerator of $SOUTH$ and the denominator of $SOUTH$ near $+2$

$$\begin{aligned} h \xrightarrow{SOUTH_{+2}} SOUTH(+2+h) &= \left.\frac{x^2+5x+6}{x^4-x^3-10x^2+x-15}\right|_{x\leftarrow +2+h} \\ &= \frac{x^2+5x+6|_{x\leftarrow +2+h}}{x^4-x^3-10x^2+x-15|_{x\leftarrow +2+h}} \\ &= \frac{(+2+h)^2+5(+2+h)+6}{(+2+h)^4-(+2+h)^3-10(+2+h)^2+(+2+h)-15} \end{aligned}$$

ii. Since we want the local input-output rule that will give us the height-sign, we try to approximate *before* we divide:

$$\begin{aligned} &= \frac{\Big[(+2)^2+5\cdot(+2)+6\Big]+[...]}{\Big[(+2)^4-(+2)^3-10(+2)^2+2-15\Big]+[...]} \\ &= \frac{\Big[+4+10+6\Big]+[...]}{\Big[+16-8-40+2-15\Big]+[...]} \\ &= \frac{+20+[...]}{-45+[...]} \\ &= -\frac{20}{45}+[...] \end{aligned}$$

and since the approximate local input-output rule near $+2$ is

$$h \xrightarrow{SOUTH_{+2}} SOUTH(+2+h) = -\frac{20}{45}+[...]$$

and the local input-output rule includes the term that gives the Height-sign near $+2$

$$-\frac{20}{45}$$

we have:

$$\text{Height-sign } SOUTH \text{ near } +2 = (-.-)$$

EXAMPLE 18.3. Let $SOUTH$ be the function specified by the global input-output rule

$$x \xrightarrow{SOUTH} SOUTH(x) = \frac{x^2+5x+6}{x^4-x^3-10x^2+x-15}$$

Find the height-sign of $SOUTH$ near -3

i. We localize both the numerator of $SOUTH$ and the denominator of $SOUTH$ near -3

$$\begin{aligned} h \xrightarrow{SOUTH_{-3}} SOUTH(-3+h) &= \left.\frac{x^2+5x+6}{x^4-x^3-10x^2+x-15}\right|_{x\leftarrow -3+h} \\ &= \frac{x^2+5x+6|_{x\leftarrow -3+h}}{x^4-x^3-10x^2+x-15|_{x\leftarrow -3+h}} \\ &= \frac{(-3+h)^2+5(-3+h)+6}{(-3+h)^4-(-3+h)^3-10(-3+h)^2+(-3+h)-15} \end{aligned}$$

ii. Since we want the local input-output rule that will give us the height-sign, we try to approximate to the constant terms:

$$\begin{aligned} &= \frac{\Big[(-3)^2+5\cdot(-3)+6\Big]+[...]}{\Big[(-3)^4-(-3)^3-10(-3)^2-3-15\Big]+[...]} \\ &= \frac{\Big[+9-15+6\Big]+[...]}{\Big[+81+27-90-3-15\Big]+[...]} \\ &= \frac{\Big[0\Big]+[...]}{\Big[0\Big]+[...]} \end{aligned}$$

We cannot divide as we could get

$$= any\,size$$

iii. We therefore must approximate the localizations at least to h

$$\begin{aligned} &= \frac{\Big[0\Big]+\Big[2\cdot(-3)+5\Big]h+[...]}{\Big[0\Big]+\Big[+4(-3)^3-3(-3)^2-10\cdot 2(-3)+1\Big]h+[...]} \\ &= \frac{\Big[-6+5\Big]h+[...]}{\Big[-108-27+60+1\Big]h+[...]} \\ &= \frac{\Big[-1\Big]h+[...]}{\Big[-74\Big]h+[...]} \end{aligned}$$

$$= \frac{-h + [...]}{-74h + [...]}$$

We divide

$$= +\frac{1}{74} + [...]$$

and since the approximate local input-output rule near -3 is

$$h \xrightarrow{SOUTH_{-3}} SOUTH(-3+h) = +\frac{1}{74} + [...]$$

and the local input-output rule includes the term that gives the Height-sign near -3

$$+\frac{1}{74}$$

we have:

$$\text{Height-sign } SOUTH \text{ near } -3 = (+,+)$$

3 Slope-sign Near x_0

EXAMPLE 18.4. Let $SOUTH$ be the function specified by the global input-output rule

$$x \xrightarrow{SOUTH} SOUTH(x) = \frac{x^2+5x+6}{x^4-x^3-10x^2+x-15}$$

find the slope-sign of $SOUTH$ near $+2$

i. We localize both the numerator of $SOUTH$ and the denominator of $SOUTH$ near $+2$ and since we want the approximate local input-output rule for the slope-sign, we will approximate to h:

$$+2+h \xrightarrow{SOUTH} SOUTH(+2+h) = \left.\frac{x^2+5x+6}{x^4-x^3-10x^2+x-15}\right|_{x\leftarrow +2+h}$$

$$= \frac{x^2+5x+6\big|_{x\leftarrow +2+h}}{x^4-x^3-10x^2+x-15\big|_{x\leftarrow +2+h}}$$

$$= \frac{(+2+h)^2+5(+2+h)+6}{(+2+h)^4-(+2+h)^3-10(+2+h)^2+(+2+h)-15}$$

$$= \frac{\Big[(+2)^2+5\cdot(+2)+6\Big] + \Big[2(+2)+5\Big]h + [...]}{\Big[(+2)^4-(+2)^3-10\cdot(+2)^2+(+2)-15\Big] + \Big[4(+2)^3-3(+2)^2-10\cdot 2(+2)+1\Big]h + [...]}$$

$$= \frac{\Big[+20\Big] + \Big[+9\Big]h + [...]}{\Big[-45\Big] + \Big[-19\Big]h + [...]}$$

ii. We set up the division with

$[-45] + [-19]h + [...]$ dividing *into* $[+20] + [+9]h + [...]$

that is:

$$\begin{array}{r|llll}
 & -\frac{20}{45} & -\frac{[9\cdot 45]-[19\cdot 20]}{45^2}h & +[...] \\
\hline
-45 \; -19h \; +[...] & +20 & +9h & +[...] \\
 & +20 & +\frac{19\cdot 20}{45}h & +[...] \\
\hline
 & 0 & +\frac{[9\cdot 45]-[19\cdot 20]}{45}h & +[...]
\end{array}$$

And since $[9 \cdot 45] - [19 \cdot 20] = 405 - 380 = +25$, the approximate local input-output rule near $+2$ is:

$$h \xrightarrow{SOUTH_{+2}} SOUTH(+2+h) = -\frac{20}{45} - \frac{25}{45^2}h + [...]$$

and the term that gives the slope-sign near $+2$ is

$$-\frac{25}{45^2}h$$

so that

$$\text{Slope-sign } SOUTH \text{ near } +2 = (\diagdown.\diagdown)$$

4 Concavity-sign Near x_0

EXAMPLE 18.5. Let $SOUTH$ be the function specified by the global input-output rule

$$x \xrightarrow{SOUTH} SOUTH(x) = \frac{x^2+5x+6}{x^4-x^3-10x^2+x-15}$$

find the concavity-sign of $SOUTH$ near $+2$

i. We localize both the numerator of $SOUTH$ and the denominator of $SOUTH$ near $+2$ and since we want the approximate local input-output rule for the slope-sign, we will approximate to h^2:

$$+2+h \xrightarrow{SOUTH} SOUTH(+2+h) = \left.\frac{x^2+5x+6}{x^4-x^3-10x^2+x-15}\right|_{x\leftarrow +2+h}$$

$$= \frac{x^2+5x+6|_{x\leftarrow +2+h}}{x^4-x^3-10x^2+x-15|_{x\leftarrow +2+h}}$$

$$= \frac{(+2+h)^2 + 5(+2+h) + 6}{(+2+h)^4 - (+2+h)^3 - 10(+2+h)^2 + (+2+h) - 15}$$

$$= \frac{\Big[(+2)^2 + 5\cdot(+2) + 6\Big] + \Big[2(+2) + 5\Big]h + \Big[1\Big]h^2}{\Big[(+2)^4 - (+2)^3 - 10\cdot(+2)^2 + (+2) - 15\Big] + \Big[4(+2)^3 - 3(+2)^2 - 10\cdot 2(+2) + 1\Big]h + \Big[6(+2)^2 - 3(+2) - 10\Big]h^2 + [...]}$$

$$= \frac{\Big[+20\Big] + \Big[+9\Big]h + \Big[1\Big]h^2}{\Big[-45\Big] + \Big[-19\Big]h + \Big[8\Big]h^2 + [...]}$$

ii. We set up the division with

$$-45 + -19h + 8h^2 + [...] \quad \text{dividing } \textit{into} \quad +20 + 9h + h^2$$

but carry it out *latin style* (that is, we *write* the *result* of the multiplication as it comes out instead of the *opposite of the result*.)

$$\begin{array}{r|ccccc}
 & -\frac{20}{45} & -\frac{[9\cdot45]-[19\cdot20]}{45^2}h & -\left[\frac{+45[45-8\cdot20]-19[[9\cdot45]-[19\cdot20]]}{45^3}\right]h^2 & +[...] \\
\hline
-45\ -19h\ +8h^2\ +[...]\ ) & +20 & +9h & +h^2 & \\
 & +20 & +\frac{19\cdot20}{45}h & -\frac{8\cdot20}{45}h^2 & +[...] \\
\hline
 & 0 & +\frac{[9\cdot45]-[19\cdot20]}{45}h & +\frac{45-8\cdot20}{45}h^2 & +[...] \\
 & & +\frac{[9\cdot45]-[19\cdot20]}{45^2}h & +19\frac{[9\cdot45]-[19\cdot20]}{45^2}h^2 & +[...] \\
\hline
 & & +0h & +\left[\frac{+45[45-8\cdot20]-19[[9\cdot45]-[19\cdot20]]}{45^2}\right]h^2 & +[...]
\end{array}$$

And since $\frac{+45[45-8\cdot20]-19[[9\cdot45]-[19\cdot20]]}{45^2} = -\frac{2401}{45^2}$, the local input-output rule near $+2$ is:

$$h \xrightarrow{SOUTH_{+2}} SOUTH(+2+h) = -\frac{20}{45} - \frac{25}{45^2}h - \frac{2401}{45^2}h^2 + [...]$$

and the term that gives the concavity-sign near $+2$ is

$$-\frac{2401}{45^2}h^2$$

so that

$$\text{Concavity-sign } SOUTH \text{ near } +2 = (\cap, \cap)$$

5 Local Graph Near x_0

EXAMPLE 18.6. Let $SOUTH$ be the function specified by the global input-output rule

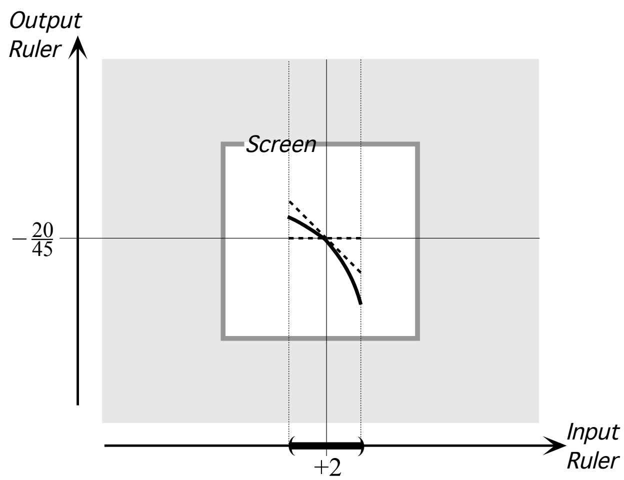
$$x \xrightarrow{SOUTH} SOUTH(x) = \frac{x^2 + 5x + 6}{x^4 - x^3 - 10x^2 + x - 15}$$

find the local graph of $SOUTH$ near $+2$

Since, in order to get the local graph near $+2$ we need all three features near $+2$, height-sign, slope-sign and concavity-sign, we need to get the approximate local input-output rule as we did in the previous example:

$$h \xrightarrow{SOUTH_{+2}} SOUTH(+2+h) = -\frac{20}{45} - \frac{25}{45^2}h - \frac{2401}{45^2}h^2 + [...]$$

from which we get:



Chapter 19

Rational Functions: Global Analysis

Contrary to what we were able to do with polynomial functions, with rational functions we will *not* be able to establish global theorems. Of course, we did not really establish global theorems for *all* polynomial functions either but only for polynomial functions *of a given degree*, 0, 1, 2 and 3. But, in the case of rational functions, even the *rational degree* will not separate rational functions into kinds that we can establish global theorems for inasmuch as even rational functions with a given rational degree can be very diverse.

So, what we will do here is to focus on how to get global information about *any given rational function*.

1 The Essential Question

Given a *rational function*, as with any function, the *offscreen graph* will consist:

- certainly of the local graph near ∞. This is because, as soon as the *input* is *large*, the graph point is going to be left or right of the screen and therefore *offscreen* regardless of the size of the *output*,

- possibly of the local graph(s) near certain *bounded input(s)*. This is because, in case the outputs for inputs near certain bounded inputs are *large*, the graph points will then be above or below the screen and therefore *offscreen* even though the inputs are *bounded*.

So, as always, we will need to ascertain whether

- There might be *bounded inputs* for which nearby inputs will have a *large* output ,

or, as was the case with all polynomial functions,

- The outputs for any *bounded input* are themselves necessarily *bounded*

In other words, in order to get the *offscreen graph*, we must begin by asking the **Essential Question**:

> - Do all *bounded inputs* have *bounded outputs*
>
> or
>
> - Is there one (or more) *bounded input* which is an ∞-height input, that is, a *bounded input* whose nearby inputs have *unbounded outputs*?

And, indeed, we will find that there are two kinds of rational functions:

- rational functions that *do* have ∞-height input(s)
- rational function that *do not* have any ∞-height input as was the case with power functions and polynomial functions.

2 Locating ∞-Height Inputs

However, given a rational function, not only will we need to know whether or not there *exists* ∞-height input(s), if there are any, we will also have to *locate* the ∞-height inputs, if any, because we will need to get the local graph near these ∞-height input(s). More precisely:

1. Given a rational function RAT specified by a global input-output rule

$$x \xrightarrow{RAT} RAT(x) = \frac{NUMERATOR_{RAT}(x)}{DENOMINATOR_{RAT}(x)}$$

we want to find whether or not there can be a *bounded input* x_0 such that the outputs for *nearby* inputs, $x_0 + h$, are *large*. In other words, we want to know if there can be x_0 such that

$$h \xrightarrow{RAT} RAT(x)|_{x \leftarrow x_0+h} = large$$

But we have

$$\begin{aligned} RAT(x)\big|_{x\leftarrow x_0+h} &= \left.\frac{NUMERATOR_{RAT}(x)}{DENOMINATOR_{RAT}(x)}\right|_{x\leftarrow x_0+h} \\ &= \frac{NUMERATOR_{RAT}(x)\big|_{x\leftarrow x_0+h}}{DENOMINATOR_{RAT}(x)\big|_{x\leftarrow x_0+h}} \\ &= \frac{NUMERATOR_{RAT}(x_0+h)}{DENOMINATOR_{RAT}(x_0+h)} \end{aligned}$$

So, what we want to know is if there can be an x_0 for which

$$\frac{NUMERATOR_{RAT}(x_0+h)}{DENOMINATOR_{RAT}(x_0+h)} = large$$

2. Since it is a *fraction* that we want to be *large*, we will use the **Division Size Theorem** from **Chapter 2**:

THEOREM 2 (**Division Size**)

$$\begin{array}{lll} \dfrac{large}{large} = any\ size & \dfrac{large}{medium} = \boldsymbol{large} & \dfrac{large}{small} = \boldsymbol{large} \\ \dfrac{medium}{large} = small & \dfrac{medium}{medium} = medium & \dfrac{medium}{small} = \boldsymbol{large} \\ \dfrac{small}{large} = small & \dfrac{small}{medium} = small & \dfrac{small}{small} = any\ size \end{array}$$

There are thus two ways that a fraction can be *large*:

- When the numerator is *large*
- When the denominator is *small*

In each case, though, we need to make sure of the other side of the fraction. So, rather than look at the size of both the numerator and the denominator at the same time, we will look separately at:

- The first *row*, that is when the *numerator* of the fraction is *large*

$$\begin{array}{lll} \dfrac{large}{large} = any\ size & \dfrac{large}{medium} = \boldsymbol{large} & \dfrac{large}{small} = \boldsymbol{large} \\ \dfrac{medium}{large} = small & \dfrac{medium}{medium} = medium & \dfrac{medium}{small} = \boldsymbol{large} \\ \dfrac{small}{large} = small & \dfrac{small}{medium} = small & \dfrac{small}{small} = any\ size \end{array}$$

  because in that case all we will then have to do is to make sure that the *denominator* is *not large* too.
- The last *column*, that is when the *denominator* of the fraction is *small*.

possible ∞-height input

$$\frac{large}{large} = any\ size \qquad \frac{large}{medium} = \boldsymbol{large} \qquad \frac{large}{\boxed{small}} = \boldsymbol{large}$$

$$\frac{medium}{large} = small \qquad \frac{medium}{medium} = medium \qquad \frac{medium}{\boxed{small}} = \boldsymbol{large}$$

$$\frac{small}{large} = small \qquad \frac{small}{medium} = small \qquad \frac{small}{\boxed{small}} = any\ size$$

because in that case all we will then have to do is to make sure that the numerator is *not small* too.

3. We now deal with $\frac{NUMERATOR_{RAT}(x_0+h)}{DENOMINATOR_{RAT}(x_0+h)}$, looking separately at the numerator and the denominator:

- Since the *numerator*, $NUMERATOR_{RAT}(x_0 + h)$, is the output of a *polynomial function*, namely

$$x \xrightarrow{NUMERATOR_{RAT}} NUMERATOR_{RAT}(x)$$

and since we have seen that *the only way* the outputs of a *polynomial function* can be *large* is when the inputs are themselves *large*, *there is no way* that $NUMERATOR_{RAT}(x_0+h))$ could be *large* for inputs that are *bounded*. So there is no way that the output of RAT could be large for *bounded* inputs that make the *numerator* large and we need not look any further.

- Since the *denominator*, $DENOMINATOR_{RAT}(x_0 + h)$, is the output of the *polynomial function*

$$x \xrightarrow{DENOMINATOR_{RAT}} DENOMINATOR_{RAT}(x)$$

and since we have seen that polynomial functions *can* have *small* outputs if they have 0-height inputs and the inputs are near the 0-height inputs, $DENOMINATOR_{RAT}(x_0+h)$ *can* be *small* for certain bounded inputs and thus so can $\frac{NUMERATOR_{RAT}(x_0+h)}{DENOMINATOR_{RAT}(x_0+h)}$. However, we will then have to make sure that $NUMERATOR_{RAT}(x_0 + h)$, is *not small* too near these bounded inputs, that is we will have to make sure that x_0 does *not* turn out to be a 0-height input for $NUMERATOR_{RAT}$ as well as for $DENOMINATOR_{RAT}$ so as not to be in the case:

$$\frac{small}{small} = any\ size$$

We will thus refer to a 0-height input for $DENOMINATOR_{RAT}$ as only a **possible ∞-height input** for RAT

Altogether, then, we have:

THEOREM 19.1 Possible ∞-height Input The 0-height inputs of the *denominator* of a rational function, if any, are the only *possible ∞-height inputs* for the rational function.

4. However, this happens to be one of these very rare situations in which there *is* "an easier way": After we have located the 0-height inputs for $DENOMINATOR_{RAT}$, instead of first making sure that they are not also 0-height inputs for $NUMERATOR_{RAT}$, we will gamble and just get the local input-output rule near each one of the 0-height inputs for $DENOMINATOR_{RAT}$. Then,

- If the local input-ouput rule turns out to start with a *negative-exponent power function*, then we will have determined that x_0 *is* an ∞-height input for RAT and the payoff will be that we will now get the local graph near x_0 for free.
- If the local input-ouput rule turns out *not* to start with a *negative-exponent power function*, then we will have determined that x_0 is *not* a ∞-height input for RAT after all and our loss will be that we will probably have no further use for the local input-output rule.

Overall, then, we will use the following two steps:

Step i. *Locate* the 0-height inputs for the *denominator*, $DENOMINATOR_{RAT}(x)$, by solving the equation

$$DENOMINATOR_{RAT}(x) = 0$$

Step ii. Compute the *local input-output rule* near each one of the 0-height inputs for the *denominator*, if any.

The advantage is that we need not even refer to the **Division Size Theorem**: once we have a possible ∞-height input, we just get the local input-output rule near that possible ∞-height input, "for the better or for the worse".

EXAMPLE 19.1. Let $COUGH$ be the function specified by the global input-output rule

$$x \xrightarrow{COUGH} COUGH(x) = \frac{x^4 - x^3 - 10x^2 + x - 15}{x^2 + 5x + 6}$$

locate the ∞-height input(s) of $COUGH$, if any.

Step i. The possible ∞-height input(s) of $COUGH$ are the 0-height input(s) of $DENOMINATOR_{COUGH}(x)$, that is the solution(s), if any, of the equation

$$x^2+5x+6=0$$

In general, solving an equation may or may not possible but in this case, the equation is a *quadratic* one and we have learned how to do this in **Chapter 12**. One way or the other, we find that there are two solutions:

$$-3, -2$$

which are the *possible* ∞*-height inputs* of the rational function $COUGH$.

Step ii. We compute the local input-output rules near -3 and near -2:

- Near -3:

$$\begin{aligned}
h \xrightarrow{COUGH\text{ near }-3} COUGH(-3+h) &= \left.\frac{x^4-x^3-10x^2+x-15}{x^2+5x+6}\right|_{x\leftarrow -3+h}\\
&= \frac{x^4-x^3-10x^2+x-15|_{x\leftarrow -3+h}}{x^2+5x+6|_{x\leftarrow -3+h}}\\
&= \frac{(-3+h)^4-(-3+h)^3-10(-3+h)^2+(-3+h)-15}{(-3+h)^2+5(-3+h)+6}
\end{aligned}$$

We try to approximate to the constant terms:

$$\begin{aligned}
&= \frac{(-3)^4+[...]-(-3)^3+[...]-10(-3)^2+[...]-3+[...]-15}{(-3)^2+[...]+5(-3)+[...]+6}\\
&= \frac{+81+27-90-3-15+[...]}{+9-15+6+[...]}\\
&= \frac{0+[...]}{0+[...]}\\
&= \frac{[...]}{[...]}\\
&= \textit{any size}
\end{aligned}$$

So we must go back and try to approximate to the linear terms, ignoring the constant terms since we just saw that they add up to 0 both in the numerator and the denominator:

$$\begin{aligned}
&= \frac{4(-3)^3h+[...]-3(-3)^2h+[...]-10\cdot 2(-3)h+[...]+h}{2\cdot(-3)h+[...]+5h}\\
&= \frac{-108h+[...]-27h+[...]+60h+[...]+h}{-6h+[...]+5h}\\
&= \frac{-74h+[...]}{-h+[...]}\\
&= +74+[...]
\end{aligned}$$

so that -3 is *not* an ∞-heigth input

- Near -2:

$$h \xrightarrow{COUGH_{\text{near } -2}} COUGH(-2+h) = \left.\frac{x^4 - x^3 - 10x^2 + x - 15}{x^2 + 5x + 6}\right|_{x \leftarrow -2+h}$$

$$= \frac{x^4 - x^3 - 10x^2 + x - 15|_{x \leftarrow -2+h}}{x^2 + 5x + 6|_{x \leftarrow -2+h}}$$

$$= \frac{(-2+h)^4 - (-2+h)^3 - 10(-2+h)^2 + (-2+h) - 15}{(-2+h)^2 + 5(-2+h) + 6}$$

We try to approximate to the constant terms:

$$= \frac{(-2)^4 + [...] - (-2)^3 + [...] - 10(-2)^2 + [...] - 2 + [...] - 15}{(-2)^2 + [...] + 5(-2) + [...] + 6}$$

$$= \frac{+16 + 8 - 40 - 2 - 15 + [...]}{+4 - 10 + 6 + [...]}$$

$$= \frac{-33 + [...]}{0 + [...]}$$

$$= \frac{-33}{[...]}$$

$$= large$$

So -2 is an ∞-height input for $COUGH$ and we need only find exactly how small [...] is to get the local input-output rule near -2

$$= \frac{-33 + [...]}{2 \cdot (-2)h + [...] + 5h}$$

$$= \frac{-33 + [...]}{h + [...]}$$

$$= -33h^{-1} + [...]$$

3 Offscreen Graph

Once the Essential Question has been answered, and if we do not already have the local input-output rule near each one of the ∞-height inputs, we need to get them and the corresponding local graphs so that we can then join them smoothly to get the offscreen graph.

Altogether, given a rational function RAT the procedure to obtain the *offscreen graph* is therefore:

i. Get the approximate input-output rule near ∞ and the local graph near ∞

ii. Answer the **Essential Question** and locate the ∞ input(s), if any,

iii. Find the local input-output rule and then the local graphs near each ∞-height inputs

EXAMPLE 19.2. Let $MARA$ be the function specified by the global input-output rule

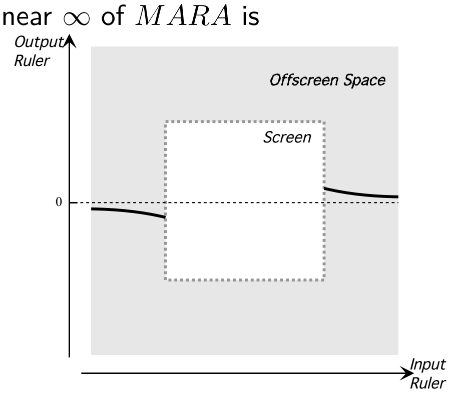
$$x \xrightarrow{MARA} MARA(x) = \frac{x-15}{x^2+5x+7}$$

Find the offscreen graph.

i. We get the local approximation near ∞:

$$\text{Near } \infty,\ x \xrightarrow{MARA} MARA(x) = \frac{x+[...]}{x^2+[...]}$$
$$= +x^{-1}+[...]$$

and the local graph near ∞ of $MARA$ is



ii. We locate the ∞-height inputs, if any. The possible ∞-height input(s) of $MARA$ are the 0-height input(s) of $DENOMINATOR_{MARA}(x)$, that is the solution(s), if any, of the equation

$$x^2+5x+7=0$$

In general, solving an equation may or may not possible but in this case, the equation is a *quadratic* one and we have learned how to do this in **Chapter 12**. One way or the other, we find that there are no solution. So, the function $MARA$ has no ∞-height input.

iii. The *offscreen graph* therefore consists of only the local graph near ∞.

4 Feature-sign Change Inputs

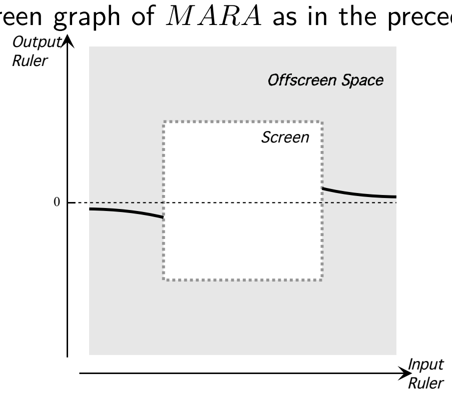
Given a rational function, in order to get the feature-sign change input(s), if any, we need only get the outlying graph and then we proceed as in **Chapter 3** so we need only give an example.

EXAMPLE 19.3. Let $MARA$ be the function specified by the global input-output rule

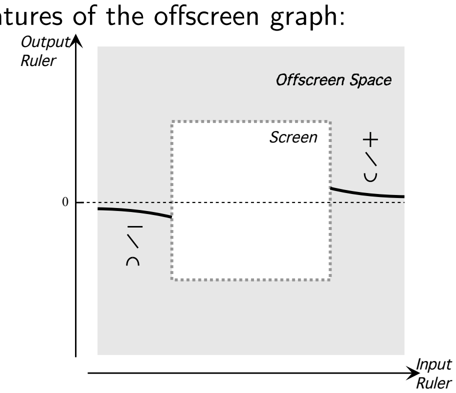
$$x \xrightarrow{MARA} MARA(x) = \frac{x - 15}{x^2 + 5x + 7}$$

Find the feature-sign change inputs of $MARA$, if any.

i. We find the offscreen graph of $MARA$ as in the preceding example:



ii. We mark the features of the offscreen graph:



iii. Therefore:

- there must be at least one height-sign change input,
- there does not have to be a slope-sign change input
- there must be at least one concavity-sign change input,

5 Global Graph

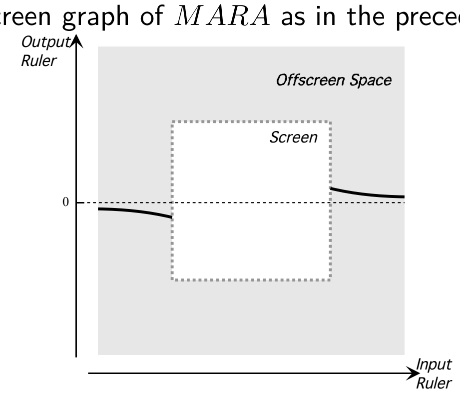
Given a rational function, in order to get the essential global graph, we need only get the outlying graph and then we join smoothly so we need only give an example.

EXAMPLE 19.4. Let $MARA$ be the function specified by the global input-output rule

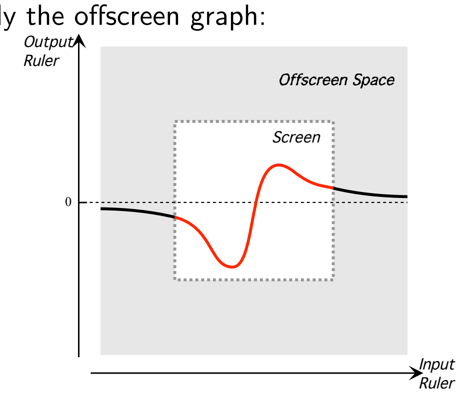
$$x \xrightarrow{MARA} MARA(x) = \frac{x-15}{x^2+5x+7}$$

Find the feature-sign change inputs of $MARA$, if any.

i. We find the offscreen graph of $MARA$ as in the preceding example:

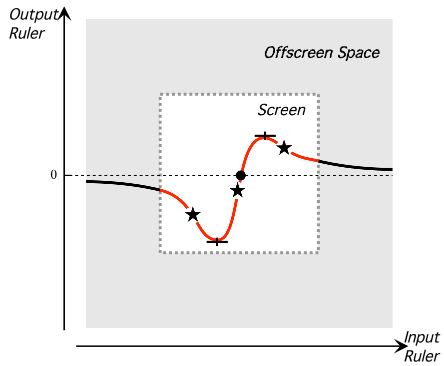


ii. We join smoothly the offscreen graph:



iii. Observe that, in fact,

- there must be at least one height-sign change input,
- there must be at least *two* slope-sign change inputs
- there must be at least *three* concavity-sign change input,



6 Locating 0-Height Inputs

Locating the 0-height inputs of a given rational function is pretty much the mirror image of what we did to locate its ∞-height inputs. More precisely:

1. Given a rational function RAT specified by a global input-output rule

$$x \xrightarrow{RAT} RAT(x) = \frac{NUMERATOR_{RAT}(x)}{DENOMINATOR_{RAT}(x)}$$

we want to find whether or not there can be a *bounded input* x_0 such that the outputs for *nearby* inputs, $x_0 + h$, are *small*. In other words, we want to know if there can be x_0 such that

$$h \xrightarrow{RAT} RAT(x)|_{x \leftarrow x_0+h} = small$$

But we have

$$\begin{aligned} RAT(x)|_{x \leftarrow x_0+h} &= \left.\frac{NUMERATOR_{RAT}(x)}{DENOMINATOR_{RAT}(x)}\right|_{x \leftarrow x_0+h} \\ &= \frac{NUMERATOR_{RAT}(x)|_{x \leftarrow x_0+h}}{DENOMINATOR_{RAT}(x)|_{x \leftarrow x_0+h}} \\ &= \frac{NUMERATOR_{RAT}(x_0+h)}{DENOMINATOR_{RAT}(x_0+h)} \end{aligned}$$

So, what we want to know is if there can be an x_0 for which

$$\frac{NUMERATOR_{RAT}(x_0+h)}{DENOMINATOR_{RAT}(x_0+h)} = small$$

2. Since it is a *fraction* that we want to be *small*, we will use the **Division Size Theorem** from **Chapter 2**:

THEOREM 2 (Division Size)

$$\frac{large}{large} = any\ size \qquad \frac{large}{medium} = large \qquad \frac{large}{small} = large$$
$$\frac{medium}{large} = \boldsymbol{small} \qquad \frac{medium}{medium} = medium \qquad \frac{medium}{small} = large$$
$$\frac{small}{large} = \boldsymbol{small} \qquad \frac{small}{medium} = \boldsymbol{small} \qquad \frac{small}{small} = any\ size$$

There are thus two ways that a fraction can be *small*:

- When the numerator is *small*
- When the denominator is *large*

In each case, though, we need to make sure of the other side of the fraction. So, rather than look at the size of both the numerator and the denominator at the same time, we will look separately at:

- The third *row*, that is when the *numerator* of the fraction is *small*

$$\frac{large}{large} = any\ size \qquad \frac{large}{medium} = large \qquad \frac{large}{small} = large$$
$$\frac{medium}{large} = small \qquad \frac{medium}{medium} = medium \qquad \frac{medium}{small} = large$$
$$\frac{small}{large} = \boldsymbol{small} \qquad \frac{small}{medium} = \boldsymbol{small} \qquad \frac{small}{small} = any\ size$$

  because in that case all we will then have to do is to make sure that the *denominator* is *not small* too.

- The first *column*, that is when the *denominator* of the fraction is *large*.

$$\frac{large}{large} = any\ size \qquad \frac{large}{medium} = large \qquad \frac{large}{small} = large$$
$$\frac{medium}{large} = \boldsymbol{small} \qquad \frac{medium}{medium} = medium \qquad \frac{medium}{small} = large$$
$$\frac{small}{large} = \boldsymbol{small} \qquad \frac{small}{medium} = \boldsymbol{small} \qquad \frac{small}{small} = any\ size$$

  because in that case all we will then have to do is to make sure that the numerator is *not large* too.

3. We now deal with $\frac{NUMERATOR_{RAT}(x_0+h)}{DENOMINATOR_{RAT}(x_0+h)}$, looking separately at the numerator and the denominator:

- Since the *numerator*, $NUMERATOR_{RAT}(x_0 + h)$, is the output of a *polynomial function*, namely

$$x \xrightarrow{NUMERATOR_{RAT}} NUMERATOR_{RAT}(x)$$

and since we have seen that polynomial functions *can* have *small* outputs if they have 0-height inputs and the inputs are near the 0-height inputs, $NUMERATOR_{RAT}(x_0+h)$ *can* be *small* for certain bounded inputs and thus so can $\frac{NUMERATOR_{RAT}(x_0+h)}{DENOMINATOR_{RAT}(x_0+h)}$. However, we will then have to make sure that $DENOMINATOR_{RAT}(x_0+h)$, is *not small* too near these bounded inputs, that is we will have to make sure that x_0 does *not* turn out to be a 0-height input for $DENOMINATOR_{RAT}$ as well as for $NUMERATOR_{RAT}$ so as not to be in the case: 0-height input

$$\frac{small}{small} = any\ size$$

We will thus refer to a 0-height input for $NUMERATOR_{RAT}$ as only a possible **0-height input** for RAT.

- Since the *denominator*, $DENOMINATOR_{RAT}(x_0+h)$, is the output of a *polynomial function*, namely

$$x \xrightarrow{DENOMINATOR_{RAT}} DENOMINATOR_{RAT}(x)$$

and since we have seen that *the only way* the outputs of a *polynomial function* can be *large* is when the inputs are themselves *large*, *there is no way* that $DENOMINATOR_{RAT}(x_0+h))$ could be *large* for inputs that are *bounded*. So there is no way that the output of RAT could be *small* for *bounded* inputs that make the *denominator* large and we need not look any further.

Altogether, then, we have:

THEOREM 19.2 Possible 0-height Input The 0-height inputs of the *numerator* of a rational function, if any, are the only *possible* 0-*height inputs* for the rational function.

4. However, this happens to be one of these very rare situations in which there *is* "an easier way": After we have located the 0-height inputs for $NUMERATOR_{RAT}$, instead of first making sure that they are not also 0-height inputs for $DENOMINATOR_{RAT}$, we will gamble and just get the local input-output rule near each one of the 0-height inputs for $NUMERATOR_{RAT}$. Then,

- If the local input-ouput rule turns out to start with a *positive-exponent power function*, then we will have determined that x_0 *is* a 0-height input for RAT and the payoff will be that we will now get the local graph near x_0 for free.
- If the local input-ouput rule turns out to start with a 0-*exponent power*

function or a *negative-exponent power function*, then we will have determined that x_0 is *not* a 0-height input for RAT after all and our loss will be that we will probably have no further use for the local input-output rule.

Overall, then, we will use the following two steps:

> **Step i.** *Locate* the 0-height inputs for the *numerator*, $NUMERATOR_{RAT}(x)$, by solving the equation
> $$NUMERATOR_{RAT}(x) = 0$$
> **Step ii.** Compute the *local input-output rule* near each one of the 0-height inputs for the *numerator*, if any.

The advantage is that we need not even refer to the **Division Size Theorem**: once we have a possible 0-height input, we just get the local input-output rule near that possible 0-height input, "for the better or for the worse".

EXAMPLE 19.5. Let $TARA$ be the function specified by the global input-output rule

$$x \xrightarrow{TARA} TARA(x) = \frac{x^3 - 8}{x^2 + 3x - 10}$$

locate the 0-height input(s) if any.

Step i. The possible 0-height input(s) of $TARA$ are the 0-height input(s) of $NUMERATOR_{TARA}(x)$, that is the solution(s), if any, of the equation

$$x^3 - 8 = 0$$

In general, solving an equation may or may not possible and in this case, the equation is a *cubic* one. Still, here it is a very incomplete one and we can see that the solution is $+2$ which is the possible 0-height input of the rational function $TARA$.

Step ii. We compute the local input-output rule near $+2$.

$$\begin{aligned} h \xrightarrow{TARA \text{ near } -3} TARA(+2+h) &= \left.\frac{x^3 - 8}{x^2 + 3x - 10}\right|_{x \leftarrow +2+h} \\ &= \frac{x^3 - 8\big|_{x \leftarrow +2+h}}{x^2 + 3x - 10\big|_{x \leftarrow +2+h}} \\ &= \frac{(+2+h)^3 - 8}{(+2+h)^2 + 3(+2+h) - 10} \end{aligned}$$

We try to approximate to the constant terms:

$$= \frac{(+2)^3 + [...] - 8}{(+2)^2 + [...] + 3(+2) + [...] - 10}$$

$$= \frac{+8 - 8 + [...]}{+4 + 6 - 10 + [...]}$$
$$= \frac{0 + [...]}{0 + [...]} = \frac{[...]}{[...]} = \text{ } any\ size$$

So we must go back and approximate to the linear terms, ignoring the constant terms since we just saw that they add up to 0 both in the numerator and the denominator:

$$= \frac{3(+2)^2 h + [...]}{2(+2)h + [...] + 3h}$$
$$= \frac{+12h + [...]}{+4h + [...] + 3h} = \frac{+12h + [...]}{+7h + [...]}$$
$$= +\frac{12}{7} + [...]$$

and, since $+\frac{12}{7} \neq 0$, $+2$ is *not* an 0-heigth input for $TARA$.

Epilogue

Where to from here on?

- Derived functions
- Functions defined equationally
- Matters of *size* e.g. the bigger the size of the exponent, the boxier the graph

Check that reciprocity has been moved correctly to **Chapter 7**

1 Looking Back

Until now, the global graph of each new kind of function was qualitatively very different as we moved from one kind of functions to the next.

1. In the case of the *power functions*, we found that the *qualitative features* of the global graphs of

i. regular positive-exponent power functions,
ii. negative-exponent power functions,
iii. exceptional power functions, that is
- 0-exponent power functions
- 1-exponent power functions

were very different but the differences among power functions of any particular type were not really that great in that, from the point of view of the *shape* of the global graph, there were really only four types of regular power functions (depending on the *sign* and the *parity* of the *exponent*) and only two types of exceptional power functions (depending on the *parity* of the *exponent*).

2. In the case of the *polynomial functions*, we found that the *qualitative features* of the global graphs changed a lot when we moved from one degree to the next:

i. The global graph of a *constant function* (Degree 0)
- has no *height*-sign change input, (same *height* everywhere)
- has no *slope*,
- has no *concavity*,

ii. The global graph of an *affine function* (Degree 1)
- always has exactly one *height*-sign change input,
- has no *slope*-sign change input, (same *slope* everywhere)
- has no *concavity*,

iii. The global graph of a *quadratic function* (Degree 2)
- may or may not have *height*-sign change input(s),
- always has exactly one *slope*-change input,
- has no *concavity*-sign change input, (same *concavity* everywhere)

iv. The global graph of a *cubic function* (Degree 3)
- has at least one *height*-sign change inputs,
- may or may not have *slope*-change input(s),
- has exactly one *concavity*-sign change input,

As for the qualitative differences among the global graphs of polynomial functions of a *same* degree, they are not great—but growing along with the *degree*.

i. The difference among *constant functions* is the *height* of the global graph.

ii. The differences among *affine functions* are the *height* and the slope of the global graph.

iii. The differences among *quadratic functions* are the *height*, the slope and the concavity of the global graph.

iv. The differences among *cubic functions* are not only the *height*, the slope and the *concavity* of the global graph but also whether or not there is a *bounded fluctuation*.

Thus, in terms of content organization, the *degree* of polynomial functions was a very powerful organizer if only because this allowed us introduce the features, *height*, *slope*, *concavity*, one at a time.

The emphasis throughout will be to convince ourselves of the need to proceed very systematically while keeping our eyes open so as to take advantage of whatever might make our life easier and not to do anything that we do not absolutely have to do.

2 Looking Ahead

We will now say a few words about the way rational functions will be dealt with in the rest of this text.

1. While, so far, we have had a very transparent content organization, in contrast, in the case of *rational functions*, the *rational degree* will *not* be such a powerful organizer because the four different types of rational functions will not be markedly different.

Still, in each one of the next four chapters, we will investigate a given type of rational function but this will be mostly in order not to upset the reader with too much variety from the get go. However, we will not be able to develop much of a theory for each type and we will mostly go about gathering experience investigating rational functions without paying too much attention to the type of rational function being dealt with, taking things as they come.

On the other hand, the differences among rational functions of any given type of rational degree, will be quite significant because of the possible ∞-height inputs.

Thus, the other side of the coin will be that, while, until now, once we had a theory of a kind of function, the investigation of this kind of functions quickly became a bit boring in that we knew what the overall global graph was going to look like, in the case of rational functions, there will be a much more interesting *diversity*.

2. Before anything else, it should be stressed that in the investigations of any given *rational function* we will follow essentially the exact same approaches that we used in the investigation of any given *power function* and of any given *polynomial function*: We will thus

i. get its local graph near ∞,
ii. get the answer to the ESSENTIAL QUESTION and find the ∞*-height input(s)*, if any. (This will involve solving an equation.)
iii. get the local graph near the ∞*-height inputs*, if any.
iv. get the global graph by interpolating the local graph near ∞ and the local graphs near the ∞*-height inputs*, if any.

3. As happened each time we investigated a new kind of function, finding the local rule near bounded inputs—and therefore near ∞*-height input(s)*—will require a new algebra tool.

4. As with any function, rational or otherwise, what we will actually do will depend of course on what information we need to find and there are reciprocal function

going to be two main kinds of questions:

a. *Local questions*, that is, for instance:

- Find the local concavity-sign near a given input,
- Find the local slope-sign near a given input,
- Find the local height-sign near a given input,
- Find the local graph near a given input,

The given input can of course be *any* input, that is ∞ or any given *bounded* input, for instance an ∞-height input, a concavity-sign change input, a slope-sign change input, a height-sign change input or any *ordinary* input whatsoever.

b. *Global questions*, that is, for instance

- Find the concavity-sign change input(s), if any
- Find the slope-sign change input(s), if any
- Find the height-sign change input(s), if any
- Find where the output has a given concavity-sign
- Find where the output has a given slope-sign
- Find where the output has a given height-sign
- Find the global graph

In the case of global questions, it will usually be better to start by getting the *bounded graph* and then to get the required information from the bounded graph. But then of course, since the bounded graph is really only the *essential* bounded graph, that is the graph that is interpolated from the *outlying graph*, the global information that we will get will only be about the *essential* features that is the features forced onto the bounded graph by the *outlying graph*.

The curious reader will obviously have at least three questions:

i. How do the various power functions compare among each other?

ii. What of polynomial functions of degree higher than 3?

iii. What of Laurent polynomial functions?

In the "overview", we will discuss the several manners in which *regular positive-power* functions, *negative-power* functions and *exceptional-power* functions all fit together. This will require discussing the *size* of slope.

3 Reciprocity Between 0 and ∞

We will now investigate the relationship between 0 and ∞

1. Reciprocal Function The **reciprocal function** is the power function with exponent -1 and coefficient $+1$, that is the function whose global input-output rule is reciprocal

$$x \xrightarrow{RECIPROCAL} RECIPROCAL(x) = (+1)x^{-1}$$
$$= +\frac{1}{x}$$

so that the output is the **reciprocal** of the input (hence the name).

1. The first thing about the reciprocal function is that it is typical of negative-exponent power functions in terms of what it does to the *size* of the output:

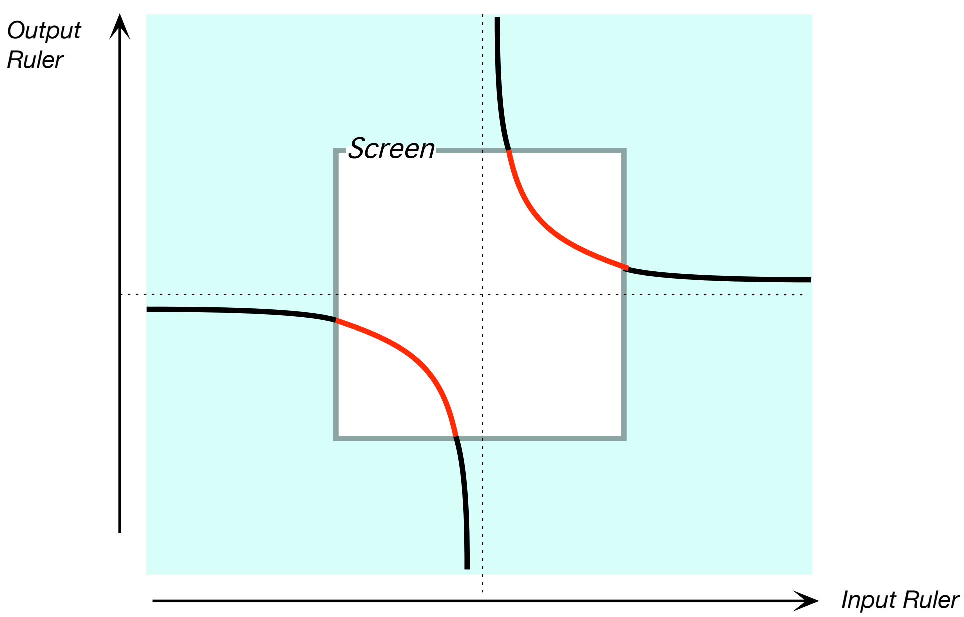
$$+large \xrightarrow{RECIPROCAL} RECIPROCAL(large) = +small$$
$$-large \xrightarrow{RECIPROCAL} RECIPROCAL(large) = -small$$

and

$$+small \xrightarrow{RECIPROCAL} RECIPROCAL(small) = +large$$
$$-small \xrightarrow{RECIPROCAL} RECIPROCAL(small) = -large$$

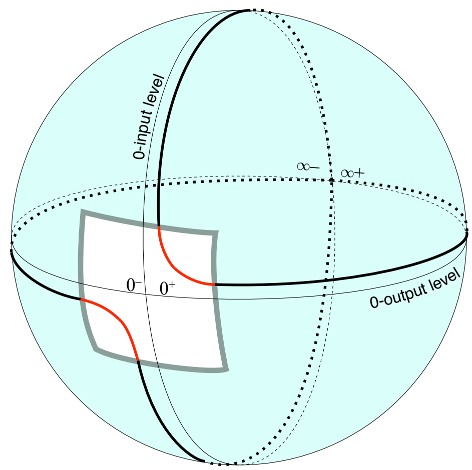
2. More generally, the global graph of the reciprocal function is:

- Mercator picture:



- Magellan picture:

family
prototypical



3. Although quite different from the *identity function*, the *reciprocal functions* does play a role in the **family** of all power functions that is quite similar in some respects to the role played by the *identity function*

For instance, because the size of the exponent in both cases is 1, they are both the "first" of their kind.

However, that is not very important because:

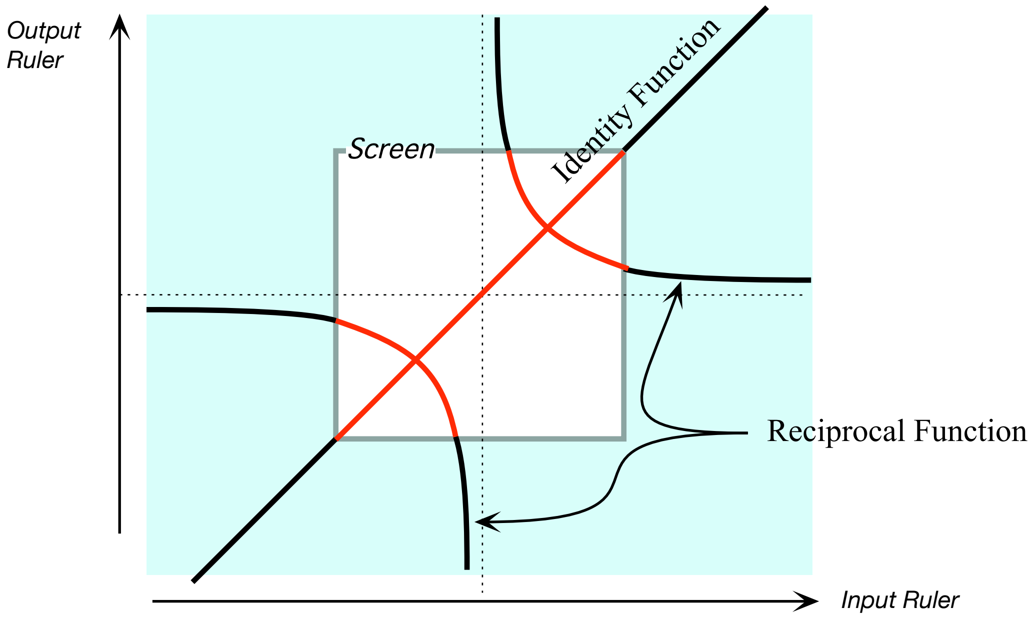
- The *identity function* is *not* **prototypical** of the other power functions because the identity function is a linear function and has no concavity.

EXAMPLE 19.6. The identity function lack concavity while all regular power function have concavity.

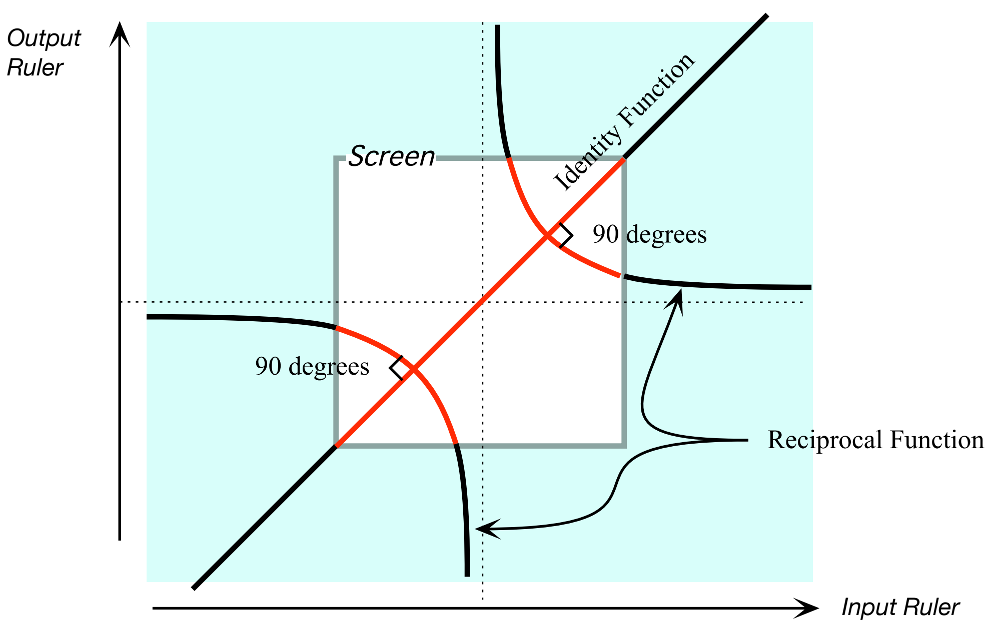
- The *reciprocal function* is *prototypical* of the other *negative power* functions in many ways.

EXAMPLE 19.7. The shape of the reciprocal function is essentially the same as the shape of all (negative-exponent) power functions of type NOP

One thing the identity function and the reciprocal function have in common, though and for what it's worth at this time, is that the reciprocal function is the mirror image of itself when the mirror is the identity function.

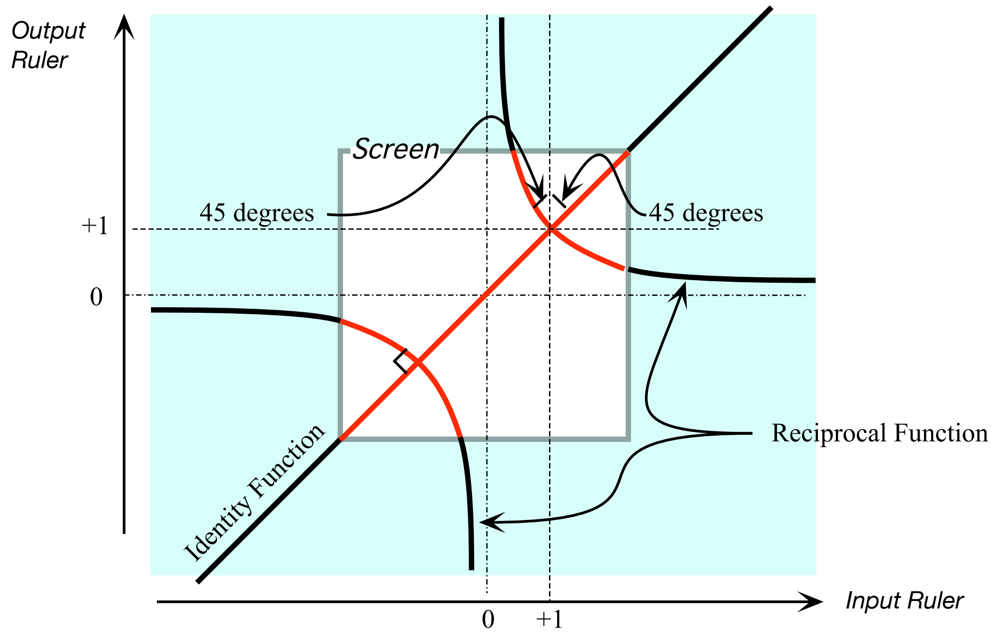


In particular, they intersect at a 90 degree angle.



Another way way to look at it is that the local graphs near +1 are locally mirror images of each other when the mirror is the input level line for +1:

far
reciprocal of each other



2. Reciprocity

1. It will be convenient to introduce two new terms:

- We introduced the word "*near*" almost from the begining and, with Magellan graphs in mind, we will now introduce the word "**far**". Thus,
  - When an input is *large*, it is *near* ∞ and therefore *far* from 0,
  - When an input is *small*, it is *near* 0 and therefore *far* from ∞.
- More generally, we will say that *two* power functions are **reciprocal of each other** when:
  - their coefficient are *the same*,
  - the *size* of their exponents are *the same*,
  - the *sign* of their exponents are *the opposite*.

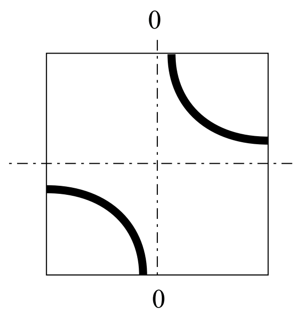
In other words, two power functions are *reciprocal of each other* whenever they differ only by the *sign* of their *exponents*.

Example 19.8. The identity function and the reciprocal function are reciprocal.

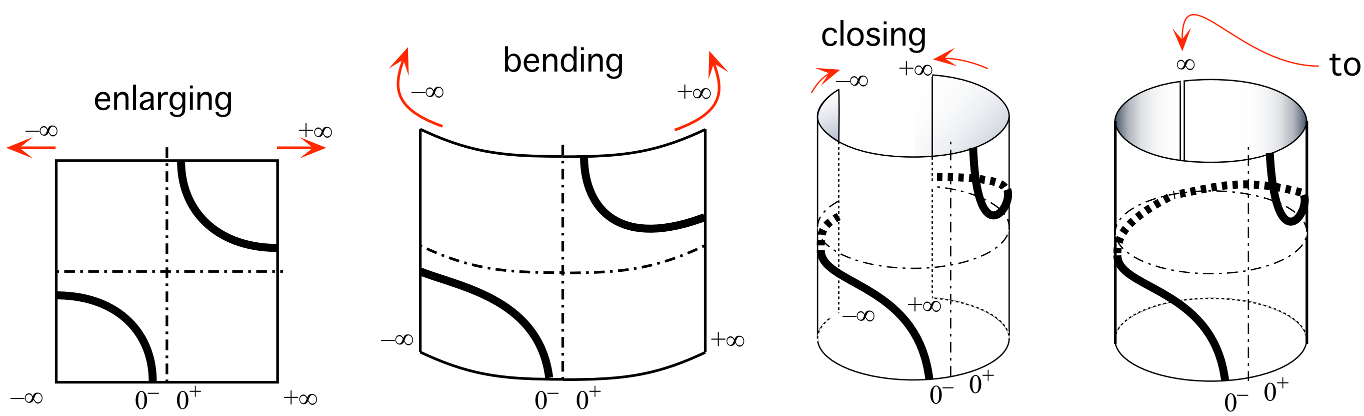
We will see that, when the mirror is the input level line for +1, the local graphs near +1 of two power functions that are *reciprocal of each other* are approximately mirror images of each other. But the angles will not be 45 degrees anymore.

2. The point of all this is that the local graph near ∞ of a regular power function is the same as the local graph near 0 of the power function that it is reciprocal of and, vice versa, the local graph near 0 of a regular power function is the same as the local graph near ∞ of the power function that it is reciprocal of.

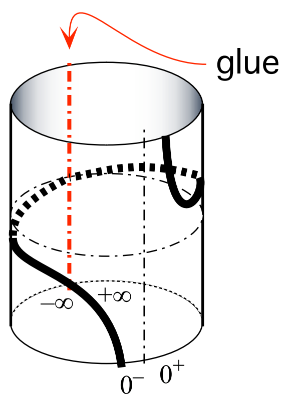
Example 19.9. Given the local graph near 0 of $JACK$, an odd *positive* power function with positive coefficient :



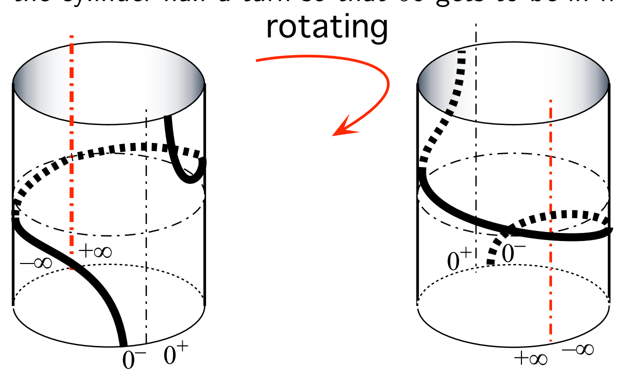
We enlarge the *extent* of the input ruler more and more while shrinking the *scale* by the edges more and more and, as we do so, we bend the screen backward more and more closing down the gap until the edges touch.



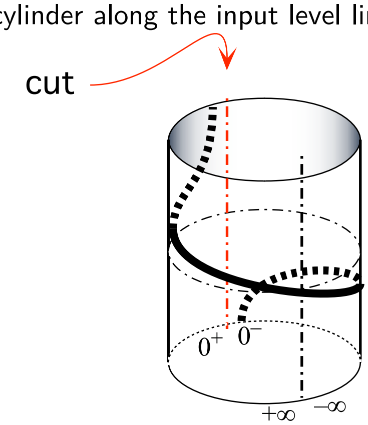
We then glue shut the edges of the screen at ∞ to get a cylinder.



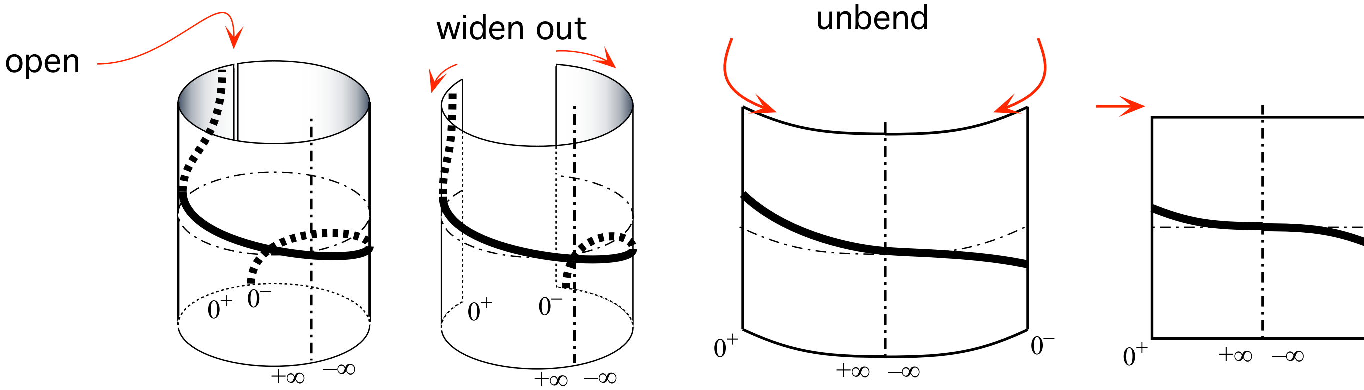
Then we turn the cylinder half a turn so that ∞ gets to be in front of us:



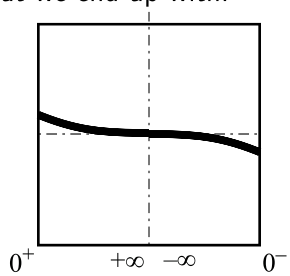
Now we cut open the cylinder along the input level line for 0



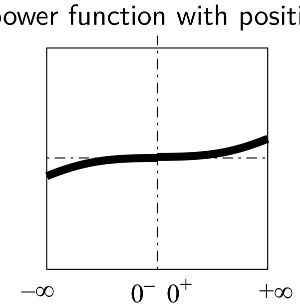
We widen out and unbend the screen forward more and more until it becomes flat.



The local graph near ∞ that we end up with:

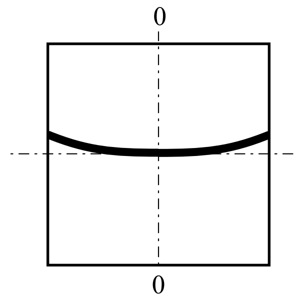


is exactly like the local graph near 0 of $JACK$'s *reciprocal* power function which is an odd *positive* power function with positive coefficient:

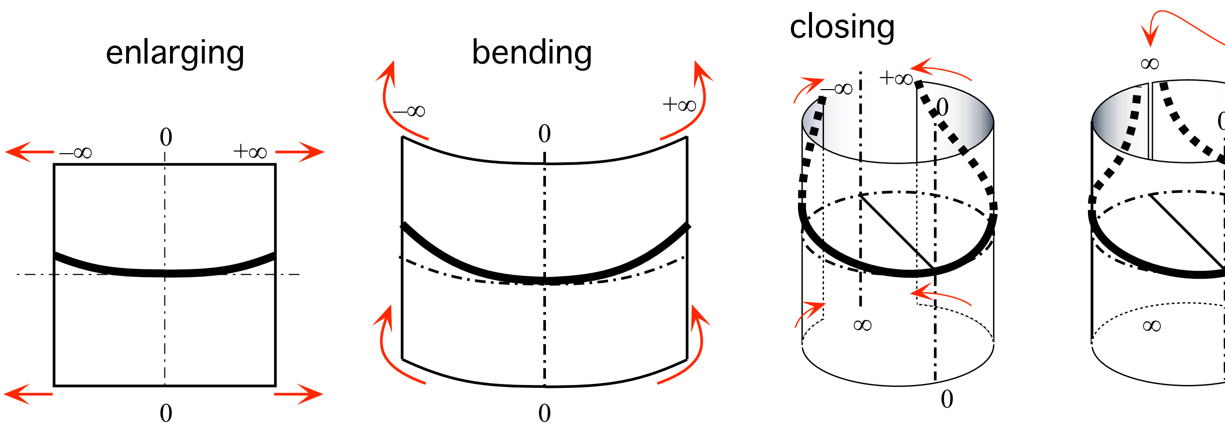


(On both graphs, outputs for negative inputs are negative and outputs for positive inputs are positive.)

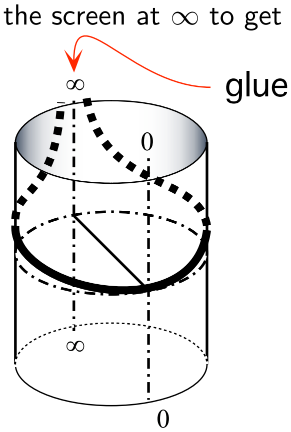
Example 19.10. Given the local graph near 0 of the even *positive* power function $JILL$:



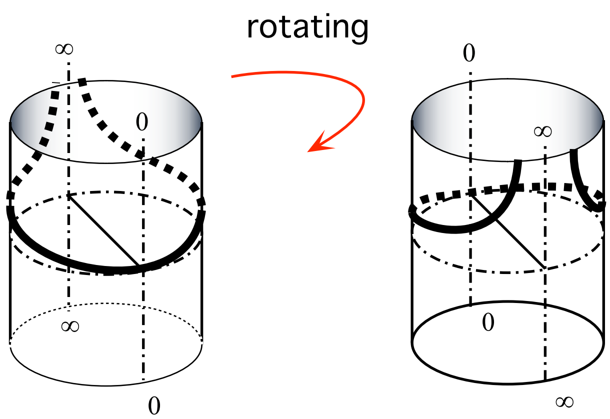
We enlarge the *extent* of the input ruler more and more while shrinking the *scale* by the edges more and more and, as we do so, we bend the screen backward more and more until the edges touch.



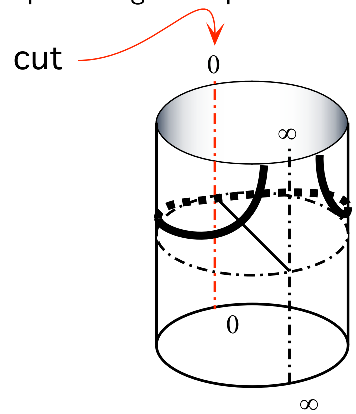
We then glue the edges of the screen at ∞ to get a cylinder.



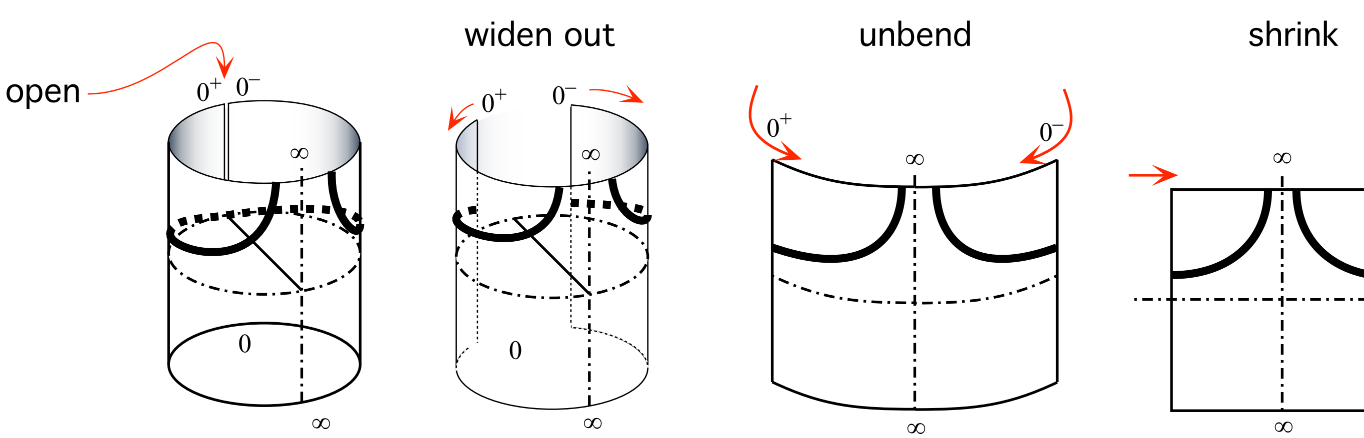
Then we turn the cylinder half a turn so that ∞ gets to be in front of us:



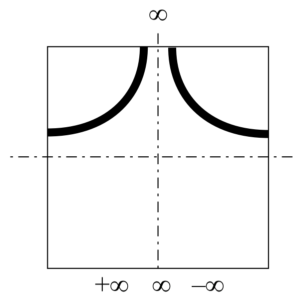
Now we cut the cylinder open along the input level line for 0



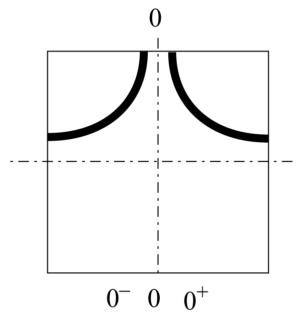
We unbend the screen forward more and more until it becomes flat.



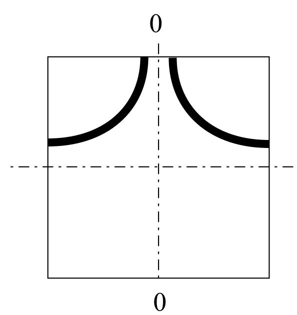
The local graph near ∞ that we get (Remember that the left side of ∞ is the positive side of ∞ and the right side of ∞ is the negative side of ∞):



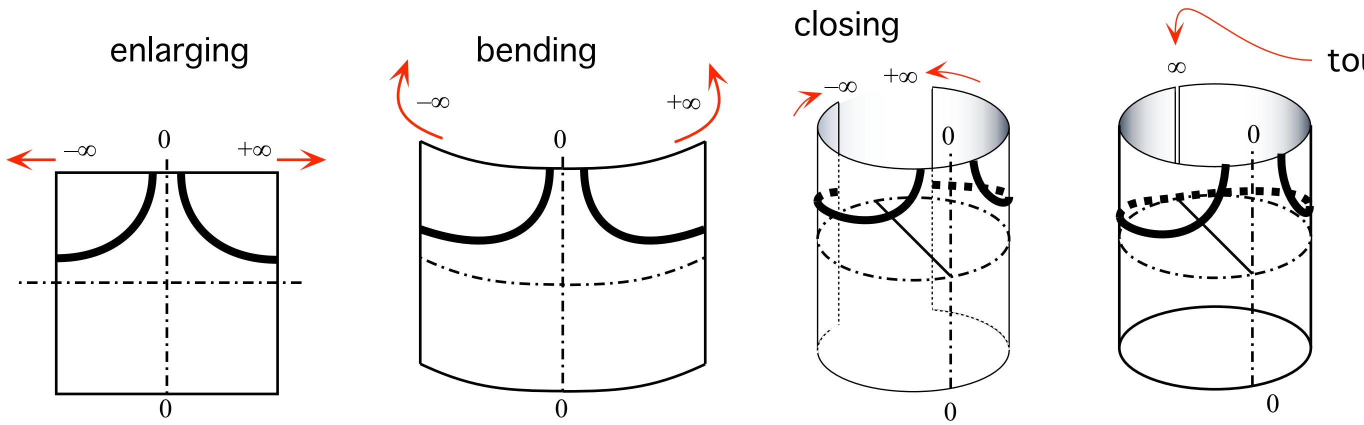
is just like the local graph near 0 of $JILL$'s *reciprocal* power function which is a *negative, even-exponent* power function:



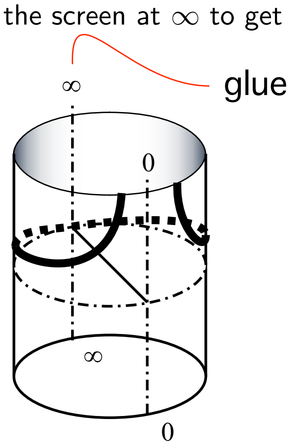
EXAMPLE 19.11. Given the local graph near 0 of the even *positive* power function $JACK$:



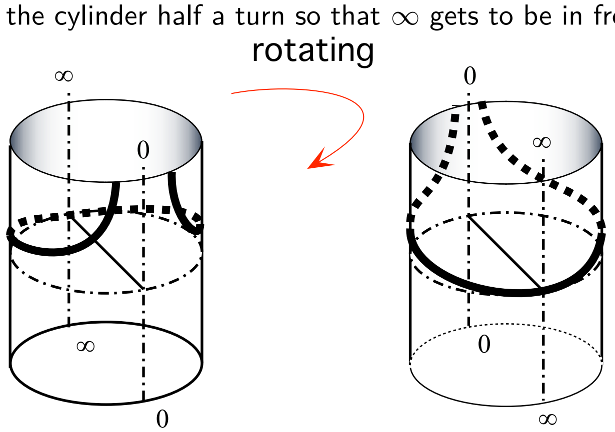
We enlarge the *extent* of the input ruler more and more while shrinking the *scale* by the edges more and more and, as we do so, we bend the screen backward more and more until the edges touch.



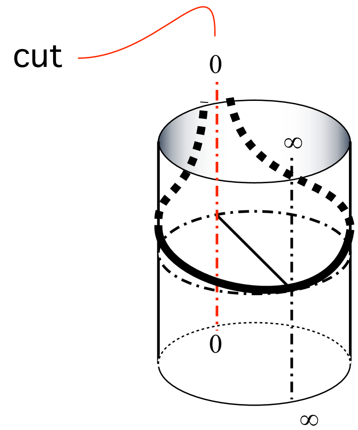
We then glue the edges of the screen at ∞ to get a cylinder.



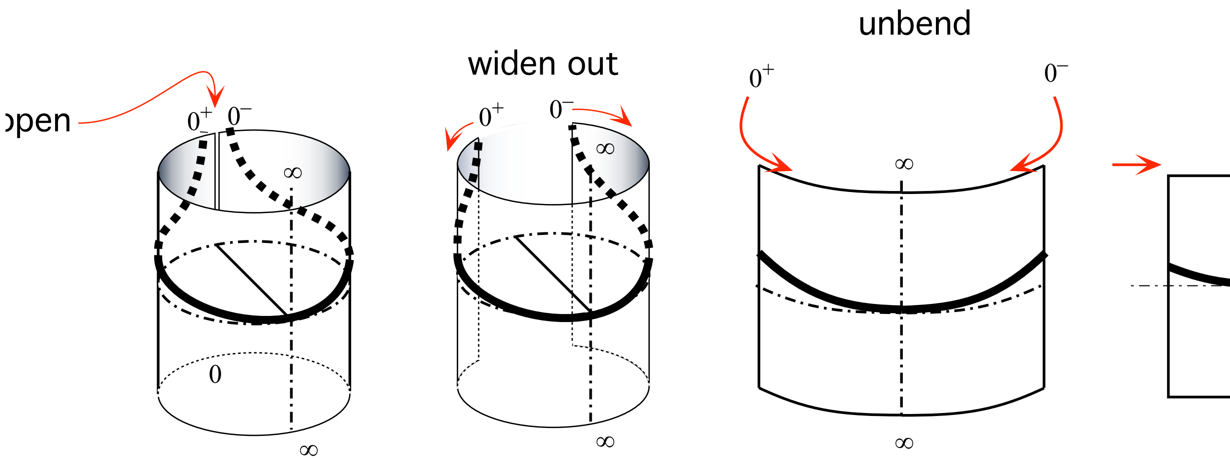
Then we turn the cylinder half a turn so that ∞ gets to be in front of us:



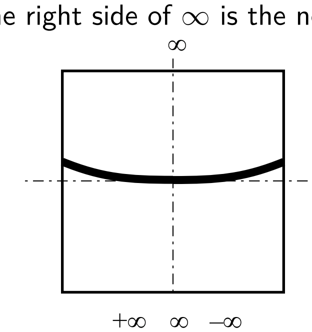
Now we cut the cylinder at 0



and we unbend the screen forward more and more until it becomes flat.

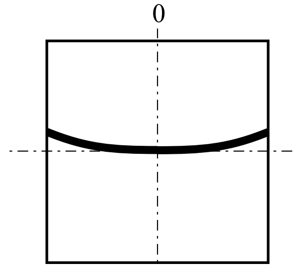


The local graph near ∞ that we get (Remember that the left side of ∞ is the positive side of ∞ and the right side of ∞ is the negative side of ∞):



is just like the local graph near 0 of $JACK$'s *reciprocal* power function which size-preserving

is an even *positive* power function:



4 The Family of Power Functions

The following is more of an informative nature at this stage than something that we will be building on in this text. The purpose here is mostly to give some coherence to all the power functions by showing various ways in which they fit together. It should help the reader organize her/his vision of power functions.

1. Types of Regular Functions This is just a recapitulation of stuff we saw in the preceding two chapters:

Sign exponent	Parity exponent	Sign coefficient	TYPE
+	*Even*	+	*PEP*
		−	*PEN*
	Odd	+	*POP*
		−	*PON*
−	*Even*	+	*NEP*
		−	*NEN*
	Odd	+	*NOP*
		−	*NON*

2. What Power Functions Do To Size We will say that a function is **size-preserving** when the size of the output is the same as the size of the input, that is "small gives small" and "large gives large".

size-inverting
fixed point

EXAMPLE 19.12. Regular positive-exponent power functions are *size-preserving*:

Correspondingly, we will say that a function is **size-inverting** when the size of the output is the *reciprocal* of the size of the input, that is "small gives large" and "large gives small".

EXAMPLE 19.13. Negative-exponent power functions are *size-inverting*:

By contrast, with *exponent-zero* power functions, the output for *small* inputs has size 1 and so is neither *small* nor *large* and so *exponent-zero* power functions are neither *size-preserving* nor *size-inverting*. You might say that they are "size-squashing".

Thus, in a way, constant functions separate *regular positive-exponent* power functions from *negative-exponent* power functions.

On the other hand, even though linear functions are exceptional, they are nevertheless *size-preserving*.

3. Fixed point A **fixed point** for a function is an input whose output is equal to the input.

EXAMPLE 19.14. Given the identity function, every input is a *fixed point*. In particular, both 0 and $+1$ are *fixed points*.

EXAMPLE 19.15. 0 is a fixed point for all regular power functions.

EXAMPLE 19.16. $+1$ is a fixed point for all regular power functions.

template

EXAMPLE 19.17. -1 is a fixed point for all regular even-exponent power functions.

5 The bigger the size of the exponent the boxier the graph

We will call **template** something that looks like it could be the graph of a regular power function except that it is not a function because the inputs -1 and $+1$ both have an unbounded number of outputs. Each type of regular power function has its own template.

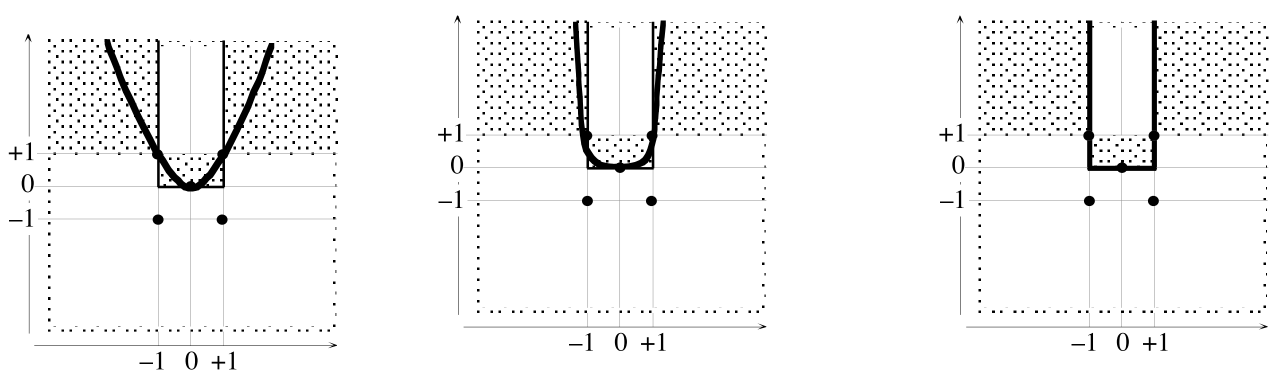
1. We begin by comparing power functions with their template two at a time.

EXAMPLE 19.18. The positive-even-exponent power function whose global input-output rule is

$$x \xrightarrow{POWER_{+4}} POWER_{+4}(x) = +x^{+4}$$

is much closer to its template than the positive-even-exponent power function whose global input-output rule is

$$x \xrightarrow{POWER_{+2}} POWER_{+2}(x) = +x^{+2}$$

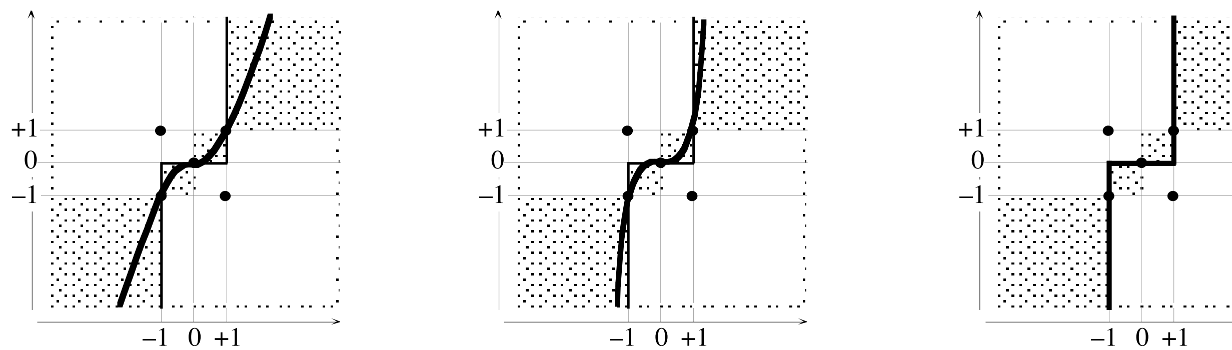


EXAMPLE 19.19. The positive-odd-exponent power function whose global input-output rule is

$$x \xrightarrow{POWER_{+5}} POWER_{+5}(x) = +x^{+5}$$

is much closer to its template than the positive-odd-exponent power function whose global input-output rule is

$$x \xrightarrow{POWER_{+3}} POWER_{+3}(x) = +x^{+3}$$

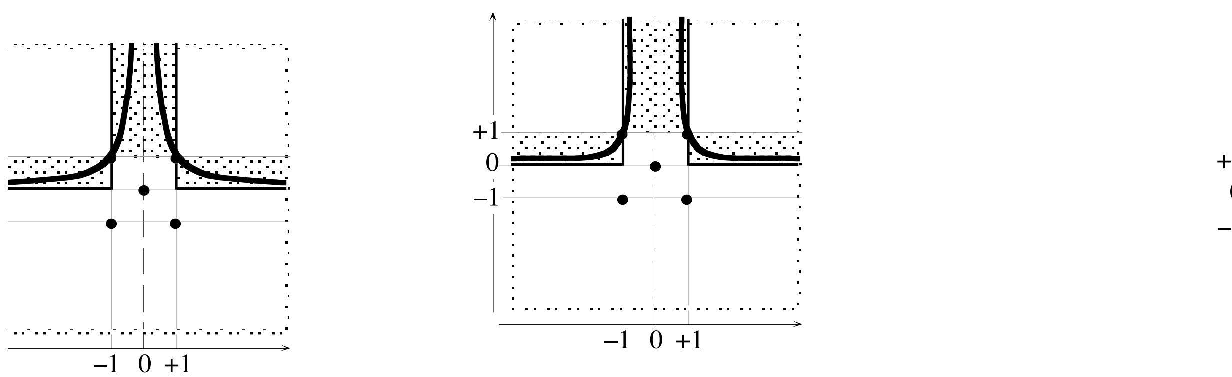


Example 19.20. The negative-even-exponent power function whose global input-output rule is

$$x \xrightarrow{POWER_{-4}} POWER_{+5}(x) = +x^{-4}$$

is much closer to its template than the negative-even-exponent power function whose global input-output rule is

$$x \xrightarrow{POWER_{-2}} POWER_{-2}(x) = +x^{-2}$$

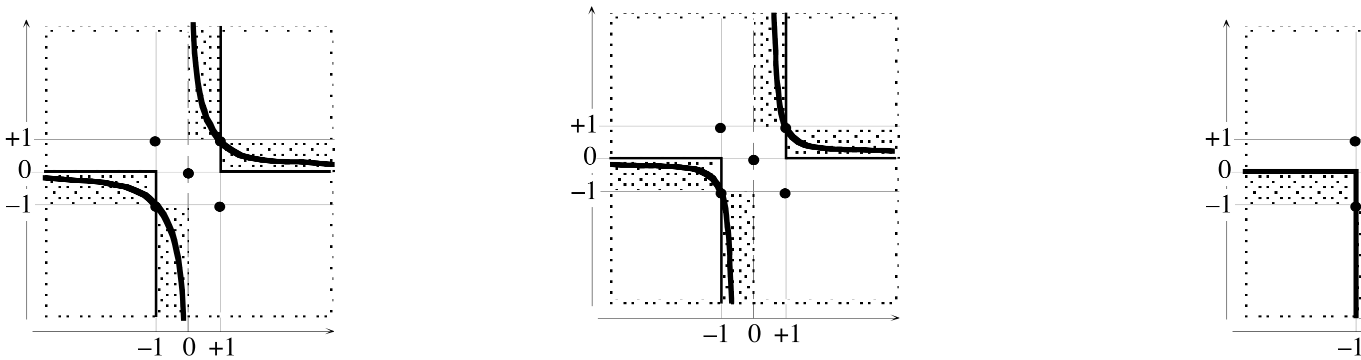


EXAMPLE 19.21. The negative-odd-exponent power function whose global input-output rule is

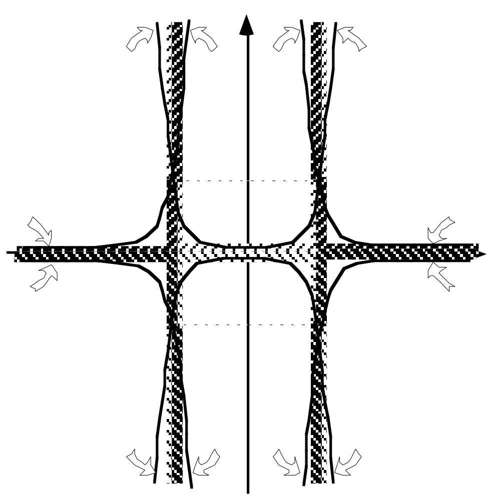
$$x \xrightarrow{POWER_{-3}} POWER_{-3}(x) = +x^{-3}$$

is much closer to its template than the negative-odd-exponent power function whose global input-output rule is

$$x \xrightarrow{POWER_{-1}} POWER_{-1}(x) = +x^{-1}$$

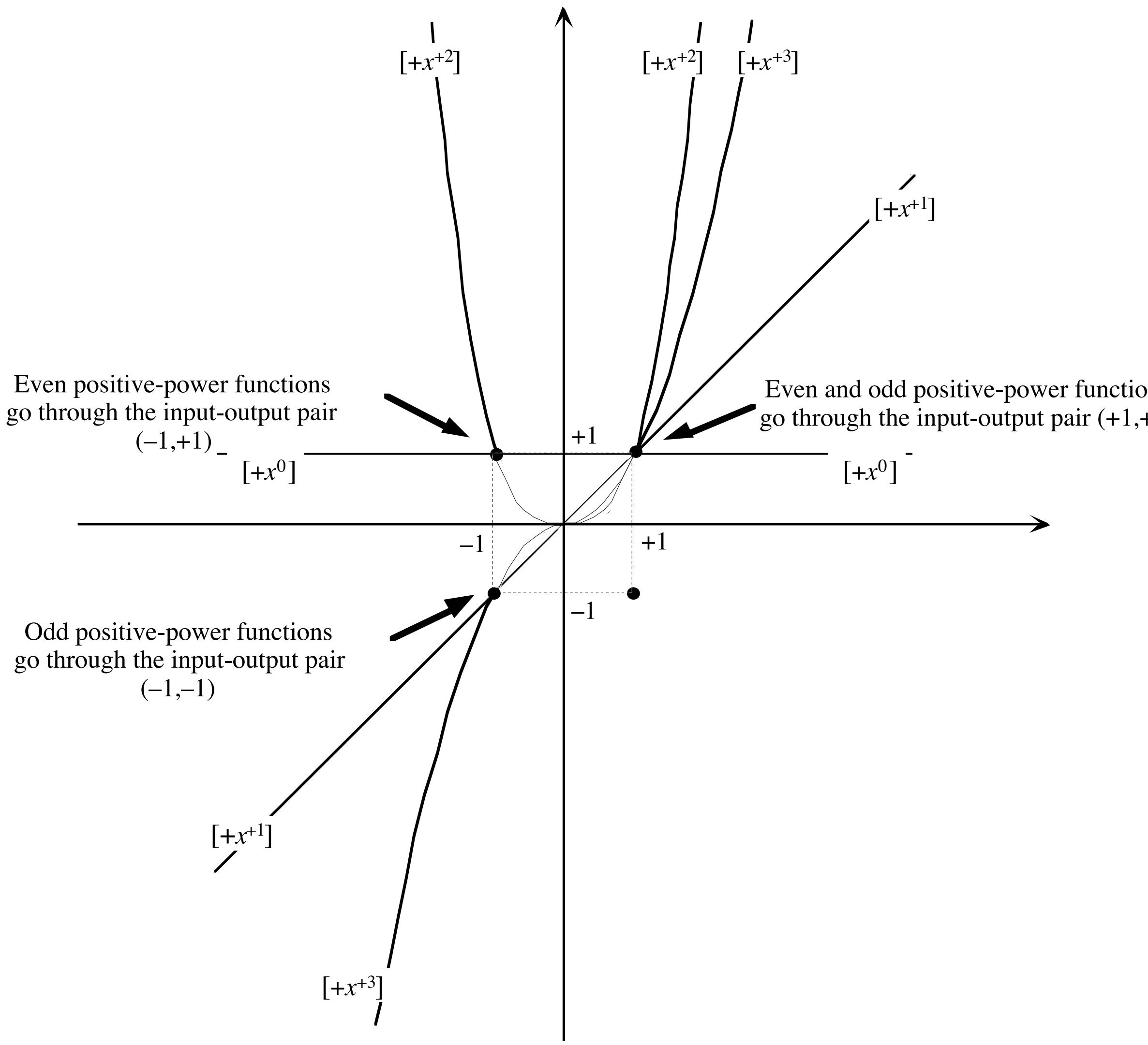


2. Together, power functions make an interesting pattern:

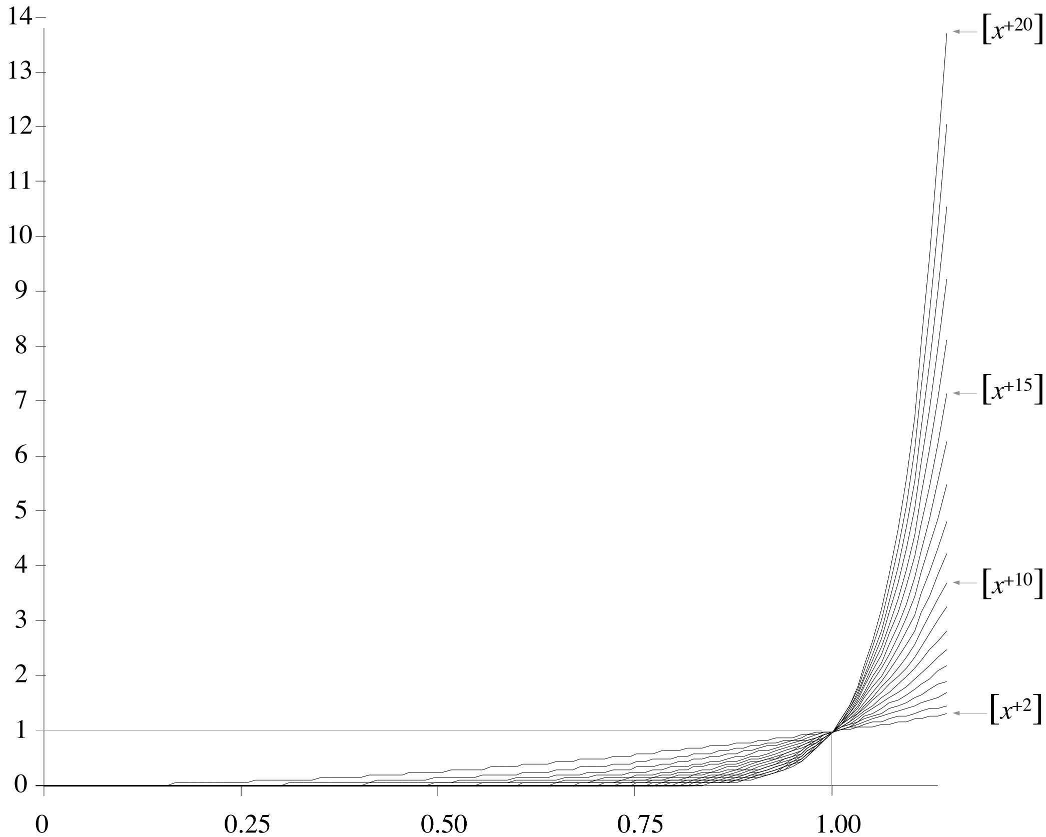


6 Local Quantitative Comparisons

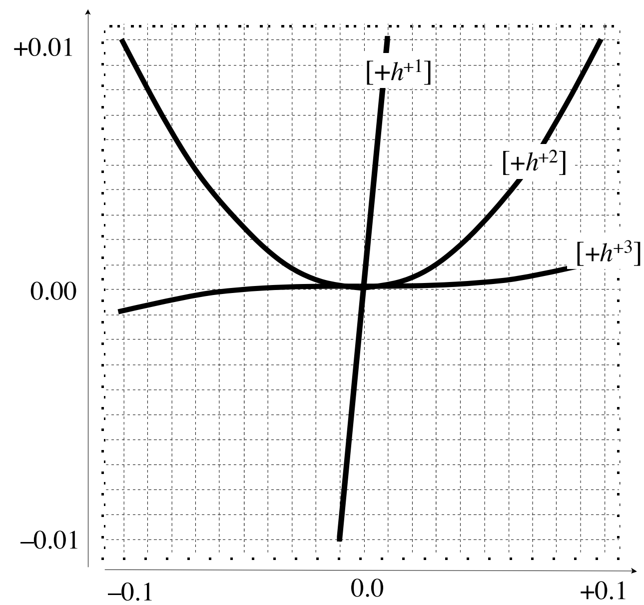
1. Local quantitative comparison near ∞



2. Local quantitative comparison near +1



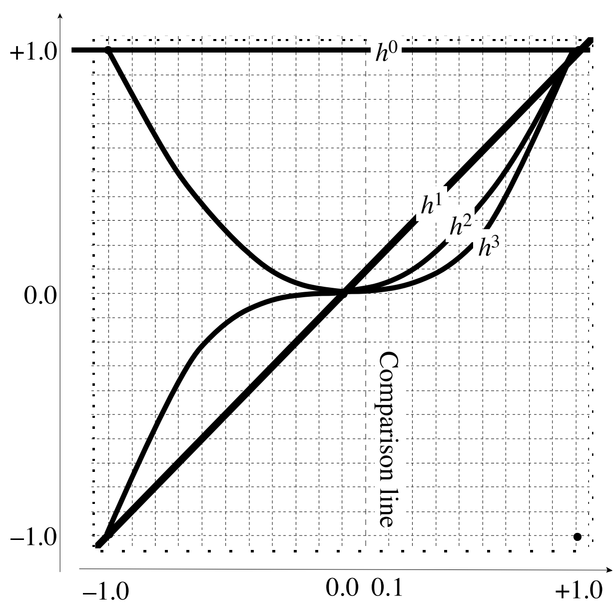
Local quantitative comparison near 0, between −0.1 and +0.1



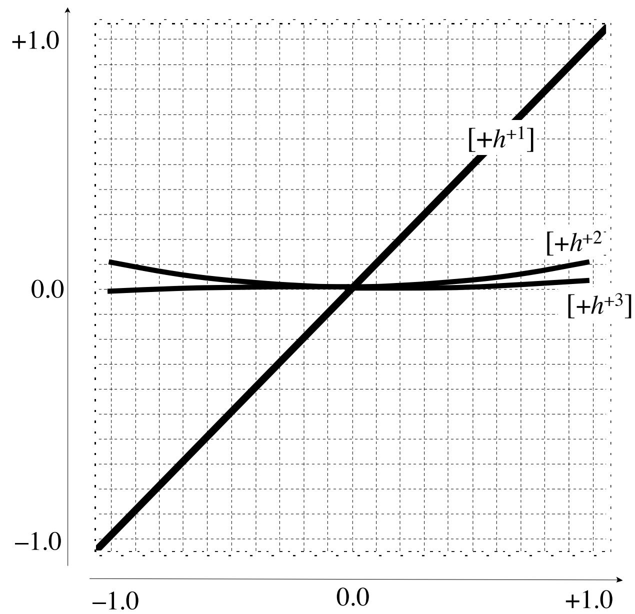
3.

7 Global Quantitative Comparisons

1. Global quantitative comparison between -1 and $+1$

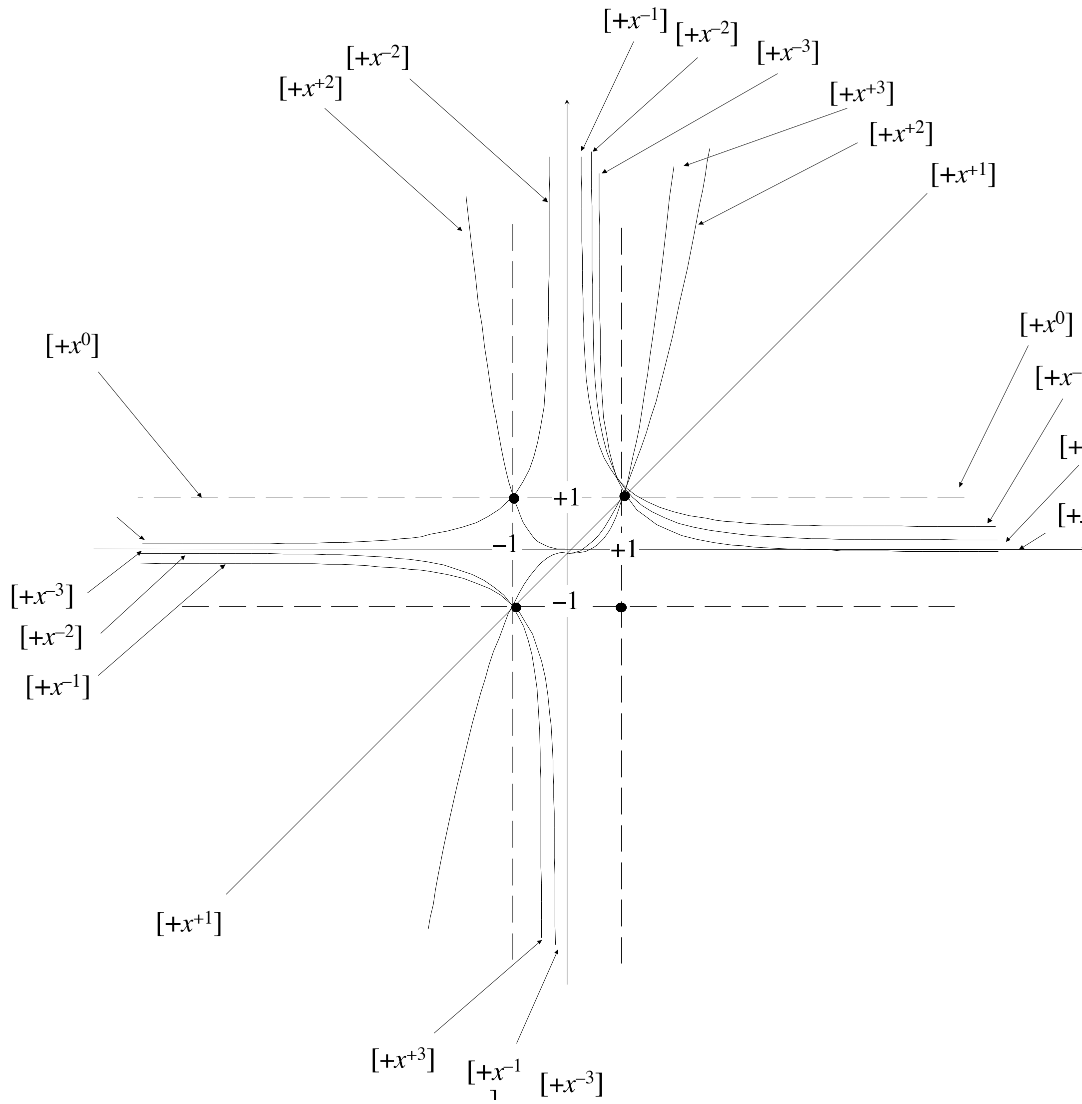


2. Global quantitative comparison between -1 and $+1$

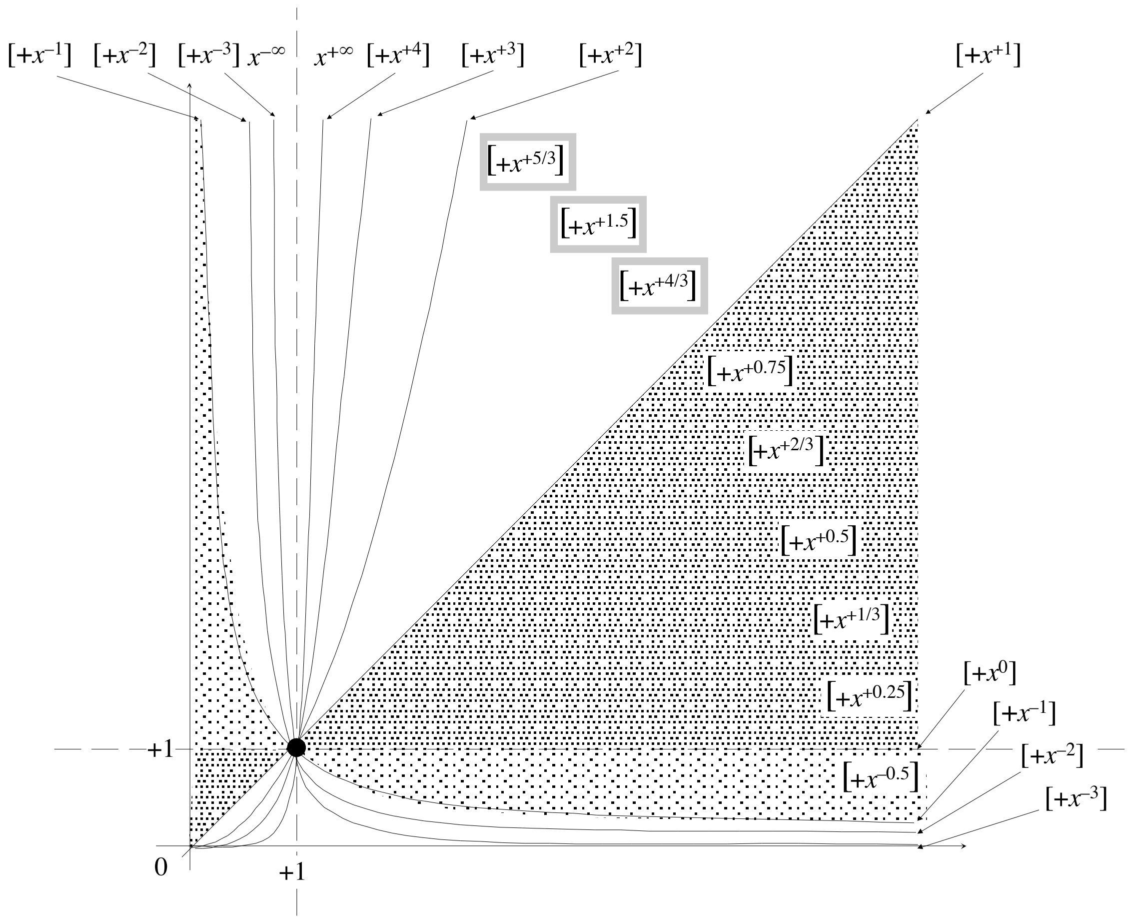


1. Symmetries Of Power Functions

3.



2. Coverage By Power Functions



Observe that there are graphs of power functions whose exponent is a fraction or a decimal number and that these graphs are exactly where we would expect them to be based on the way the fractional or decimal exponent fits with the whole number exponents. This, though, is a something that will be investigated in the next volume:
textscReasonable Transcendental Functions.

Appendices

relative

Appendix A

Localization

Inputs are counted from the origin that comes with the ruler. However, rather than counting inputs **relative** to the origin of the ruler, it is often desirable to use some other origin to count inputs from.

Reverse problem

Appendix B

Reverse Problems

Reverse problems are called that way because, in a reverse problem, what is GIVEN is the *feature* that the outputs are to have and what is WANTED are the inputs for which the function returns outputs with the *given feature* so that

Appendix C

Addition Formulas

1 Dimension $n = 2$: $(x_0 + h)^2$ (Squares)

In order to get

Appendix D

Polynomial Divisions

1 Division in Descending Exponents

Since *decimal numbers* are combinations of powers of TEN, it should not be surprising that the procedure for dividing decimal numbers should also work for *polynomials* which are combinations of powers of x.

Appendix E

List of Definitions

Appendix F

List of Theorems

Appendix G

List of Notes

Appendix H

List of Agreements

GNU Free Documentation License

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **"Document"**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **"you"**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **"Modified Version"** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **"Secondary Section"** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **"Invariant Sections"** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **"Cover Texts"** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **"Transparent"** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called **"Opaque"**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **"Title Page"** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section **"Entitled XYZ"** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **"Acknowledgements"**, **"Dedications"**, **"Endorsements"**, or **"History"**.) To **"Preserve the Title"** of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the

Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index