affine␣function
linear␣coefficient
constant␣coefficient

Chapter 9

Affine Functions: Local Analysis

Affine functions are specified by a global input-output rule of the form

$$x \xrightarrow{AFFINE} AFFINE(x) = \underbrace{a\,x^{+1} \oplus b\,x^{0}}_{\text{output-specifying code}}$$

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 9.1. The affine function $NINA$ specified by the linear coefficient -31.39 and the constant coefficient $+5.34$ is the function specified by the global input-output rule

$$x \xrightarrow{NINA} NINA(x) = \underbrace{-31.39}_{\text{linear coefficient}}\ x\ +\ \underbrace{5.34}_{\text{constant coefficient}}$$

It is worth noting that

Note 9.1. The terms in the global input output rule *need not* be written in order of *descending* exponent. This is just a habit we have.

term
linear term
constant term
linear␣part
constant␣part

EXAMPLE 9.2. 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

$$x \xrightarrow{NINA} NINA(x) = +5.34 - 31.39x$$

LANGUAGE 9.1. *Affine* functions are still called *linear* functions in PRECALCULUS textbooks on the grounds that, as we will see in ??, their global graph is a straight *line*. But the term *affine* came about nearly a century ago when mathematicians and physicists agreed to give the word *linear* a completely different meaning[1]. (See https://en.wikipedia.org/wiki/Linearity.)

We now develop some language which will enable us to describe very precisely what we we will be doing.

i. 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** of $AFFINE$.
- b which is called the **constant term** of $AFFINE$,

EXAMPLE 9.3. 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:

$$= \underbrace{-31.39x}_{\text{linear term}} \underbrace{+5.34}_{\text{constant term}}$$

ii. Corresponding to each term in the output-specifying code of

$$x \xrightarrow{AFFINE} AFFINE(x) = \underbrace{ax}_{\text{linear term}} + \underbrace{b}_{\text{constant term}}$$

there is a monomial function:

- The monomial function $x \to ax$ called the **linear part** of $AFFINE$
- The monomial function $x \to b$ called the **constant part** of $AFFINE$

[1] But of course Educologists have no trouble saying that linear functions aren't ... linear.

EXAMPLE 9.4. Corresponding to each term in the output-specifying code of

$$x \xrightarrow{NINA} NINA(x) = \underbrace{-31.39x}_{\text{linear term}} \underbrace{+5.34}_{\text{constant term}}$$

there is a monomial function:

- The linear function $x \to -31.39x$ is the *linear part* of $NINA$,
- The constant function $x \to +5.34$ is the *constant part* of $NINA$

LANGUAGE 9.2. 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.

9.1 Output *at* x_0

9.1 HOW TO Get the output at x_0 of the *affine* function specified by the global input-output rule $x \xrightarrow{AFFINE} AFFINE(x) = ax + b$

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

$$(x_0, ax_0 + b)$$

DEMO 9.1 To get the output at -3 of the function specified by the global input-output rule

$$x \xrightarrow{ALDA} ALDA(x) = -32.67x + 71.07$$

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*

$$(-3, +169.08)$$

As discussed in **??** though, and as was already the case with monomial functions, instead of getting the output of a function *at* an input, we will usually get the output of the function *near* that input.

9.2 Output *near* ∞

In the case of ∞, as pointed out in **??** on **??**, we just *cannot* get the output of a function *at* ∞ and can *only* get the output *near* ∞.

1. In order to input a neighborhood of ∞, we need to *declare* that $x \leftarrow \pm large$ that is that x is to be replaced by $\pm large$. However, inasmuch as, with affine functions and all functions after them, the output-specifying code will involves more than one term, it would become more and more cumbersome to compute with $\pm large$ and, unfortunately, there is no symbol universally agreed upon to stand for $\pm large$.
So, in conformity with universal practice, we will *declare* that "x is *large*" which is short for "Size $x = large$", in other words short for "$x = \pm large$". But, in the computations after that, we will just use x.
This, though, is dangerous and makes it absolutely imperative to keep in mind throughout the computations that everything that follows may be TRUE *only* because x has been declared to be $\pm large$.

2. We will then *execute* the output-specifying code, namely $ax+b$, into a

jet near ∞, that is with the terms in **descending *order of sizes***, which, since x is *large*, means that the powers of x must be in *descending* order of exponents. Once the dust has settled, we will have the **local input-output rule near** ∞:

jet near ∞

local input-output rule near ∞

$$x\,large \xrightarrow{AFFINE} AFFINE(x) = \underbrace{\text{Powers of } x \text{ in } descending \text{ order of exponents}}_{\text{jet near } \infty}$$

EXAMPLE 9.5. Given the function specified by the global input-output rule

$$x \xrightarrow{BIBA} BIBA(x) = -61.03 - 82.47x$$

Near ∞ we already have the powers of x and all we need is to get the *order of sizes*. First, -61.03 is *bounded*. Second, since -82.47 is *bounded* and x is *large* and *bounded* $\cdot$ *large* $=$ *large*, then $-82.47 \cdot x$ is *large*. So, in the jet near ∞, $-82.47 \cdot x$ comes first and -61.03 comes next and we get the local input-output rule near ∞:

$$x\,large \xrightarrow{AFFINE} AFFINE(x) = \underbrace{-82.47x - 61.03}_{\text{jet near } \infty}$$

3. Altogether, then:

9.2 HOW TO Get the output near ∞ of the *affine* function specified by the global input-output rule $x \xrightarrow{AFFINE} AFFINE(x) = ax + b$

i. *Declare* that x is to be replaced by *large*

$$x\Big|_{x\leftarrow large} \xrightarrow{AFFINE} AFFINE(x)\Big|_{x\leftarrow large} = ax + b\Big|_{x\leftarrow large}$$

which gives:

$$x\,large \xrightarrow{AFFINE} AFFINE(x) = \underbrace{ax + b}_{\text{output-specifying code}}$$

ii. *Execute* the output-specifying code into a *jet* near ∞

$$= \underbrace{[a]\,x \oplus [b]}_{\text{jet near } \infty}$$

which gives the *local input-output rule* near ∞:

$$x\,large \xrightarrow{AFFINE} AFFINE(x) = \underbrace{[a]\,x \oplus [b]}_{\text{jet near } \infty}$$

(Which *here* 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 9.2 To get the output near ∞ the function specified by the global input-output rule

$$x \xrightarrow{NINA} NINA(x) = -61.03 - 82.47x$$

i. We declare that x is to be replaced by $large$

$$x\Big|_{x \leftarrow large} \xrightarrow{NINA} NINA(x)\Big|_{x \leftarrow large} = -61.03 - 82.47x\Big|_{x \leftarrow large}$$

which gives:

$$x\,large \xrightarrow{NINA} NINA(x) = \underbrace{-61.03 - 82.47x}_{\text{output-specifying code}}$$

ii. We *execute* the output-specifying code into a *jet* near ∞:

$$= \Big[-82.47\Big]\, x \oplus \Big[-61.03\Big]$$

which gives the *local input-output rule* near ∞:

$$x\,large \xrightarrow{NINA} NINA(x) = \underbrace{\Big[-82.47\Big]\, x \oplus \Big[-61.03\Big]}_{\text{jet near } \infty}$$

(Observe that in *this* demo the *local* input-output rule near ∞ does *not* look the same as the *global* input-output rule because the terms in the global input-output rule that specifies $NINA$ happened *not* to be in descending order of exponents.)

9.3 Output *near* x_0

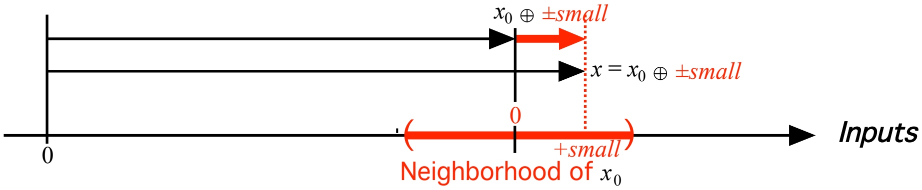
Contrary to what happened with monomial functions where we were not concerned with the neighborhood of any bounded input other than 0, with all other functions we *will* very often be interested in the neighborhood of some bounded input(s) *other* than 0.

In fact, while in the case of *regular monomial functions* 0 played just as important a role as ∞(reciprocity), this will not at all be the case with any other kind of function where 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.)

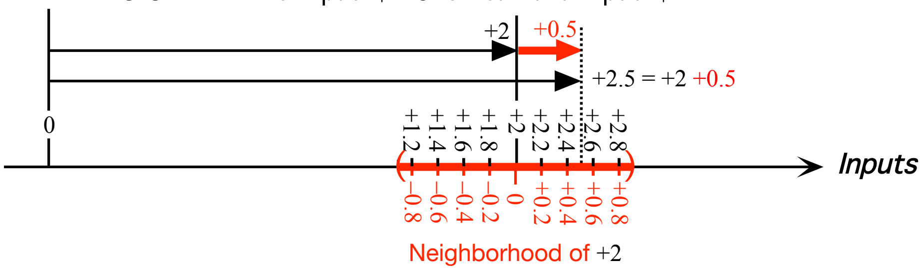
h
jet near x_0
local input-output rule near x_0

However, "thickening the plot" near a bounded inputx_0 will still involve the use of $\pm small$.

1. In order to input a neighborhood of x_0 we will have to *declare* that $x \leftarrow x_0 \pm small$ that is that x is to be replaced by $x_0 \oplus \pm small$.



Example 9.6. The input $+2.3$ is *near* the input $+2$:



And, just as with $\pm large$ above, inasmuch as, with affine functions and all functions after them, the output-specifying code will involves more than one term, it would become more and more cumbersome to compute with $\pm small$.

Fortunately, though, the letter h is universally accepted as standing for $\pm small$. In other words, as opposed to *small*, h 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 will then *execute* the input-output specifying phrase into a **jet near** x_0 that is with the terms in **descending *order of sizes*** which, since h is *small*, means that the powers of h must be in *ascending* order of exponents. Once the dust has settled—which is a bit more complicated than near ∞, we will have the **local input-output rule near** x_0:

$$x_0 + h \xrightarrow{AFFINE} AFFINE(x_0 + h) = \underbrace{\text{Powers of } h \text{ in } \textit{ascending} \text{ order of exponents}}_{\text{jet near } x_0}$$

Example 9.7. Given the function specified by the global input-output rule

$$x \xrightarrow{BIBA} BIBA(x) = -82.47x - 61.03$$

Near $+2$ we do *not* already have the powers of h and we must begin by getting them.

$$\begin{aligned} +2+h \xrightarrow{BIBA} BIBA(+2+h) &= -82.47(+2+h) - 61.03 \\ &= -82.47(+2) - 82.47h - 61.03 \\ &= -164.94 - 82.47h - 61.03 \\ &= -82.47h - 225.97 \end{aligned}$$

Now we need to get the powers of h in *descending order of sizes*: Since -82.47 is *bounded* and h is *small* then by **??** on **??**, $-82.47 \cdot h$ is *small* while -225.97 is *bounded* so that -225.97 comes first and we get the local input-output rule near $+2$:

$$+2+h \xrightarrow{AFFINE} AFFINE(+2+h) = \underbrace{-225.97 - 82.47h}_{\text{jet near } +2}$$

3. Altogether, then:

9.3 How To Get the output near x_0 of the *affine* function specified by the global input-output rule $x \xrightarrow{AFFINE} AFFINE(x) = ax + b$

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:

$$\begin{aligned} &= ax_0 + ah + b \\ &= \underbrace{\left[ax_0 + b\right] \oplus \left[a\right] h}_{\text{jet near } x_0} \end{aligned}$$

which gives the *local input-output rule* near x_0:

$$x_0 + h \xrightarrow{AFFINE} AFFINE(x_0+h) = \underbrace{\left[ax_0 + b\right] \oplus \left[a\right] h}_{\text{jet near } x_0}$$

jet near x_0

DEMO 9.3 To get the output near -3 of the function specified by the global input-output rule

$$x \xrightarrow{ALDA} ALDA(x) = -32.67x + 71.07$$

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:

$$\begin{aligned}
&= -32.67(-3) - 32.67h + 71.07\\
&= +98.01 - 32.67h + 71.07\\
&= +98.01 + 71.07 - 32.67h\\
&= \underbrace{\Big[+169.08\Big] \oplus \Big[-32.67\Big]h}_{\text{jet near } -3}
\end{aligned}$$

which gives the *local input-output rule* near -3:

$$-3+h \xrightarrow{ALDA} ALDA(-3+h) = \underbrace{\Big[+169.08\Big] \oplus \Big[-32.67\Big]h}_{\text{jet near } -3}$$

9.4 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* an input, be it *bounded* or *infinity*, from the *jet near* that input.

9.4 HOW TO Get the local graph near ∞ of the *affine* function specified by the global input-output rule $x \xrightarrow{AFFINE} AFFINE(x) = ax + b$

i. Get the jet near ∞

$$x\, large \xrightarrow{AFFINE} AFFINE(x) = \Big[a\Big]x + \Big[b\Big]$$

using How To 9.2 on page 149

ii. Get the graph of the linear term near ∞ by graphing near ∞ the

monomial function $x \to ax$ using **??** on **??**.
iii. Get the graph of the constant term near ∞ by graphing near ∞ the monomial function $x \to b$ using **??** on **??**.
iv. Get the local graph near ∞ of $NINA$ by adding-on the constant term near ∞ to the linear term near ∞ using **??** on **??**.

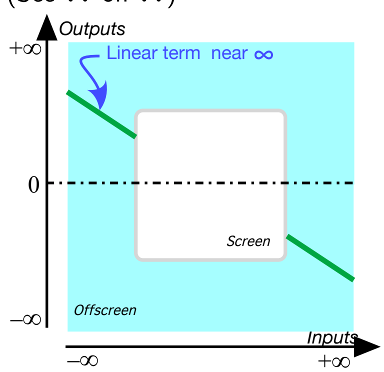
Demo 9.4 To get the local graph near ∞ of the function specified by the global input-output rule

$$x \xrightarrow{NINA} NINA(x) = -61.03 - 82.47x$$

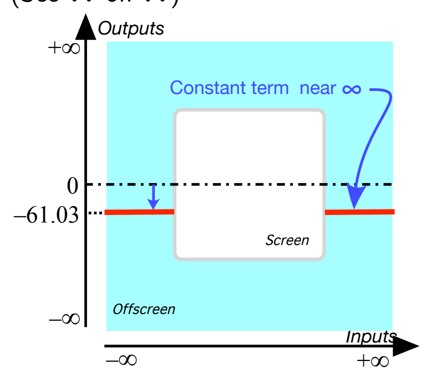
i. We get the jet near ∞ of $NINA$: (See Demo 9.2 on page 150)

$$x \xrightarrow{NINA} NINA(x) = \Big[-82.47\Big]x + \Big[-61.03\Big]$$

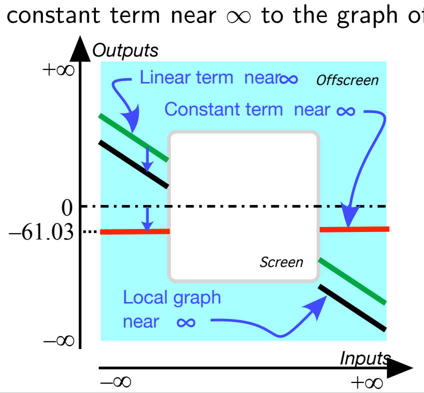
ii. We get the graph of the linear term near ∞ by graphing near ∞ the monomial function $x \to \Big[-82.47\Big]x$ (See **??** on **??**)



iii. We get the graph of the constant term near ∞ by graphing near ∞ the monomial function $x \to \Big[-61.03\Big]$ (See **??** on **??**)



iv. We get the local graph near ∞ of $NINA$ by adding-on the graph of the constant term near ∞ to the graph of the linear term near ∞. (See **??** on **??**)



9.5 HOW TO Get the local graph near x_0 of the *affine* function specified by the global input-output rule $x \xrightarrow{AFFINE} AFFINE(x) = ax + b$

i. Get the jet near x_0 of $AFFINE$ using How To 9.3 on page 152
ii. Get the graph of the constant term in the jet near x_0 namely of $\left[ax_0 + b \right]$
iii. Add-on the graph of the linear term in the jet near x_0 namely of $\left[a \right] h$

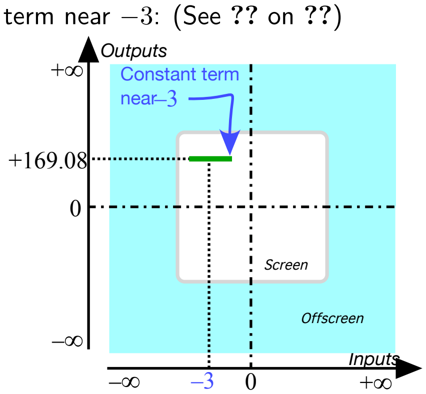
DEMO 9.5 To get the local graph near -3 of the function specified by the global input-output rule

$$x \xrightarrow{ALDA} ALDA(x) = -32.67x + 71.07$$

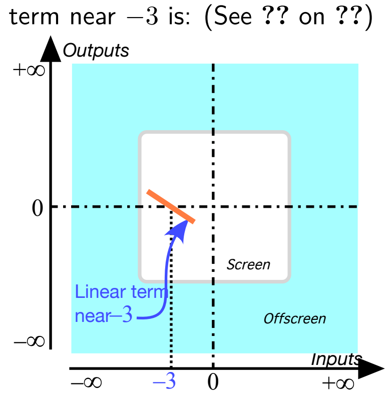
i. We get the jet near -3 of $ALDA$ by evaluating $ALDA$ near -3: (See Demo 9.3 on page 153)

$$-3 + h \xrightarrow{ALDA} ALDA(-3 + h) = \underbrace{\left[+169.08 \right] \oplus \left[-32.67 \right] h}_{\text{jet near } -3}$$

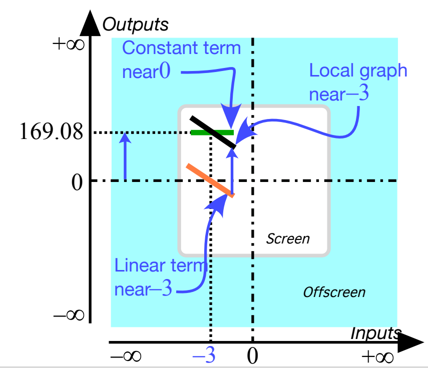
ii. We get the graph of the constant term near -3: (See ?? on ??)



iii. We get the graph of the linear term near -3 is: (See ?? on ??)



iv. We add-on the graph of the linear term near -3 to the graph of the linear term near -3. (See ?? on ??)



9.5 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 *jet* from the global input-output rule using How To 9.2 on page 149 when the given input is ∞ or How To 9.3 on page 149 when the given input is x_0.

ii. Get the *local graph* from the jet using How To 9.4 on page 153 when the given input is ∞ or How To 9.5 on page 155 when the given input is x_0.

iii. Get the *feature-sign* from the *local graph*.

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. In the case of *infinity* things are quite straightforward:

9.6 How To Get the feature-signs near ∞ of the *affine* function specified by the global input-output rule $x \xrightarrow{AFFINE} AFFINE(x) = ax + b$

i. Get the local input-output rule near ∞:

$$x\ large \xrightarrow{AFFINE} AFFINE(x) = ax + b$$
$$= \underbrace{[a]x \oplus [b]}_{\text{jet near } \infty}$$

ii. Then, in the *jet* near ∞:

- Get the *Height-sign* from the *linear term* $\big[a\big]x$ because the next term $\big[b\big]$ is *too small to matter*. So get Height-sign $AFFINE$ near ∞ from the Height-sign of the monomial function $x \to ax$ near ∞.
- Get the *Slope-sign* from the *linear term* $\big[a\big]x$ because the next term $\big[b\big]$ is *too small to matter*. So get Slope-sign $AFFINE$ near ∞ from the Slope-sign of the monomial function $x \to ax$ near ∞.
- Since both the *linear term* and the *constant term* have no concavity, $AFFINE$ has no *Concavity-sign* near ∞.

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

TEMO 9.1 L et $JULIE$ be the function specified by the global input-output rule

$$x \xrightarrow{JULIE} JULIE(x) = -2x - 6$$

Get the feature-signs near ∞.

TEMO 9.2 L et $PETER$ be the function specified by the global input-output rule

$$x \xrightarrow{PETER} PETER(x) = +3x + 6$$

Get the feature-signs near ∞.

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

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.

9.7 How To Get the feature-signs near x_0 of the *affine* function specified by the global input-output rule $x \xrightarrow{AFFINE} AFFINE(x) = ax + b$

i. Get the local input-output rule near x_0:

$$\begin{aligned} x_0 + h \xrightarrow{AFFINE} AFFINE(x_0 + h) &= a(x_0 + h) + b \\ &= ax_0 + ah + b \\ &= ax_0 + b + ah \\ &= \underbrace{\big[ax_0 + b\big] \oplus \big[a\big]h}_{\text{jet near } x_0} \end{aligned}$$

ii. Then, in the *jet* near x_0:

- Get the *Height-sign* from the *constant term* $\big[ax_0 + b\big]$ because the next term $\big[a\big]h$ is *too small to matter*. So get Height-sign $AFFINE$ near x_0 from the Height-sign of the monomial function $h \to ax_0 + b$ near 0.
  If the *constant term* is 0, then the next term, namely the *linear term* $\big[a\big]h$, does matter even though it is *small*. So, get Height-sign $AFFINE$ near x_0 from the 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*, So, get Slope-sign $AFFINE$ near x_0 from the 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 9.3 L et $JULIE$ be the function specified by the global input-output rule

$$x \xrightarrow{JULIE} JULIE(x) = -2x - 6$$

Get the feature-signs near $+2$.

i. We get the local input-output rule near $+2$:

$$\begin{aligned} +2 + h \xrightarrow{JULIE} JULIE(+2+h) &= -2(+2+h) - 6 \\ &= -2(+2) - 2h - 6 \\ &= -4 - 2h - 6 \\ &= -4 - 6 - 2h \\ &= \underbrace{\Big[-10\Big] \oplus \Big[-2\Big]h}_{\text{jet near } +2} \end{aligned}$$

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 $\langle \searrow, \searrow \rangle$, we get that Slope-sign $JULIE$ near $+2 = \langle \searrow, \searrow \rangle$.
- 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 9.4 L et $PETER$ be the function specified by the global input-output rule

$$x \xrightarrow{PETER} PETER(x) = +3x + 6$$

Get the feature-signs near -2.

i. We get the local input-output rule near -2:

$$\begin{aligned} -2 + h \xrightarrow{PETER} PETER(-2+h) &= +3(-2+h) + 6 \\ &= +3(-2) + 3h + 6 \\ &= -6 + 3h + 6 \\ &= -6 + 6 + 3h \\ &= \underbrace{\Big[0\Big] \oplus \Big[+3\Big]h}_{\text{jet near } -2} \end{aligned}$$

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.

=========THIS IS THE END OF THE CHAPTER=========

SOME OF THE STUFF COMMENTED BELOW GOES TO GLOBAL

For *affine* functions, the linear coefficient in the jet near ∞ is equal to the linear coefficient in the global input-output rule.

Index