

The numodel package*

Paul Zuurbier
mail@paulzuurbier.nl

May 17, 2026

Abstract

A LuaLaTeX package for writing and rendering numerical models (Euler-integrated dynamical systems) directly inside LaTeX documents, aimed at physics teaching material. It provides a text-model pipeline (`\mvar`, `\mrule`, `\computemodel`), Forrester stock-and-flow diagrams, and optional plots of the computed time series via the sibling package `numodel-plot`.

Contents

1	Introduction	1
2	First example: a free-falling ball	2
2.1	<code>\textmodel</code> — rule table	2
2.2	<code>\graphicmodel</code> — Forrester diagram	2
2.3	<code>\computemodel</code> + <code>\diagrammodel</code> — numerical plot	3
3	Configuration	3
3.1	Diagram styles in practice	4
4	Public API	5
4.1	Variables and rules	5
4.2	Render commands	6
4.3	Series accessors	6
4.4	Namespace management	7
5	Variable types	7
6	Generated accessors	8
7	Multiple models in one document	8
8	Requirements	8
9	Implementation	9
9.1	Translation files	47
9.1.1	<code>numodel-EN.def</code> — English (XMILE)	47
9.1.2	<code>numodel-NL.def</code> — Dutch (CoachTaal)	47

1 Introduction

`numodel` lets an author write a dynamical system (stocks, flows, helper variables, rules, and a stop condition) as a sequence of LaTeX macros and renders three complementary views of that model directly in the document:

- a *text model* — a typeset rule table with initial values (`\textmodel`);
- a *graphic model* — a Forrester stock-and-flow diagram with auto-layout (`\graphicmodel`);

*This document corresponds to `numodel` v0.3.0, dated May 17, 2026.

- a *diagram* — a numerical Euler simulation plus PGFPlot of any pair of variables (`\computemodel` followed by `\diagrammodel`; the plot is rendered through the sibling package `numodel-plot`).

All three views are produced from a single set of declarations so the textbook description, the conceptual stock-and-flow diagram, and the numerical result of the same model are guaranteed to stay in sync. Variables and rules live in namespaces (*prefixes*) so a document can contain multiple independent models; `\newmodelprefix{P}` starts a fresh one. The simulation engine runs in Lua (through `luacode`) for $\mathcal{O}(1)$ appends and cheap min/max tracking; the rendering layer is pure `expl3`.

2 First example: a free-falling ball

A ball dropped from $h_0 = 100$ m under constant gravitational acceleration. The complete model:

```
\usepackage[syntax=EN]{numodel}

\newmodelprefix{ball}
\mvar{T}{t}{0}{\s}{2}{system}
\mvar{Dt}{dt}{0.1}{\s}{2}{system}
\mvar{V}{v}{0}{\m\per\s}{2}{stock}
\mvar{Y}{y}{100}{\m}{3}{stock}
\mvar{G}{g}{-9.81}{\m\per\s\squared}{3}{aux}

\mrule{V}{\ballV + \ballG * \ballDt}
\mrule{Y}{\ballY + \ballV * \ballDt}
\mrule{T}{\ballT + \ballDt}
\mstop{\ballY <= 0}
```

After the declarations above, three render commands produce the three views shown below, each from the *same* model.

2.1 `\textmodel` — rule table

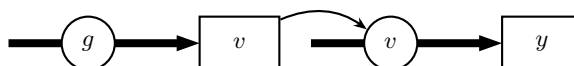
The verbatim source rendered by `\textmodel`:

	model	initial values
1	$v = v + g \cdot dt$	$t = 0$ s
2	$y = y + v \cdot dt$	$dt = 0.10$ s
3	$t = t + dt$	$v = 0$ m s ⁻¹
4	IF $y \leq 0$ THEN STOP ENDIF	$y = 100$ m $g = -9.81$ m s ⁻²

Each `\mvar` with a non-empty start value contributes a row in the *initial values* column; each `\mrule` contributes a row in the *model* column. Symbols come from the second `\mvar` argument (the display text), values are formatted through `siunitx`. `<=` is rendered as \leq .

2.2 `\graphicmodel` — Forrester diagram

`\graphicmodel` draws the same model as a stock-and-flow diagram. Stocks (type `stock`) are rectangles, constants (`constant`) are circles, helpers (`aux`) are unboxed identifiers; flows are inferred from rule structure (`\<stock> + ...` or `... + \<stock>` on the right-hand side):



Layout is automatic from the rule graph. Manual placement is available through `gridx/gridy` keys on the `\mvar` call (see Section 4). Wide diagrams can be capped to a maximum number of grid columns with `\numodelsetup{gridmaxx=N}`: when a row reaches N, the auto layout shifts the affected items up one row and continues filling.

2.3 `\computemodel` + `\diagrammodel` — numerical plot

`\computemodel` iterates the rules forward in time using Euler integration with step size `\ballDt`, stopping when `\mstop`'s condition becomes true (or when the `maxiter` safety limit is reached, see Section 3). `\diagrammodel{xvar}{yvar}{label}` then plots one variable against another:

```
\computemodel
\diagrammodel{T}{Y}{ball-fall}
```

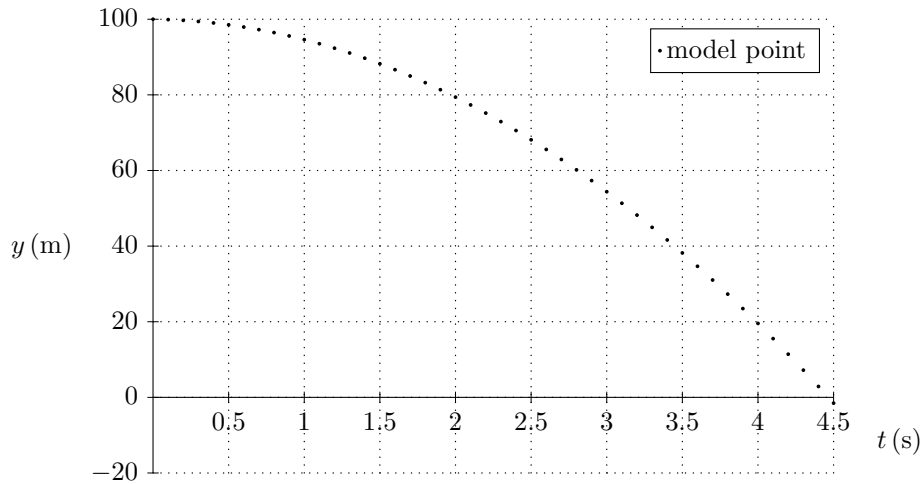


Figure 1: $y(t)$ -diagram

Axis ranges, tick lattice, and labels are computed automatically from the simulated min/max of each variable; the plot inherits the `numodel-plot` style (see that package's documentation for configuration).

3 Configuration

`\numodelsetup` Runtime configuration:

```
\numodelsetup{syntax=NL, maxiter=50000}
```

The same keys can also be passed as package options: `\usepackage[syntax=NL]{numodel}`. Recognised keys:

syntax Language tag for the rule-table rendering. Built-in values:

EN (default) XMLE-style ALL-CAPS keywords: IF/THEN/ELSE, AND, OR, ABS, SIGN, ...

NL Dutch CoachTaal keywords: Als/Dan/Anders, EN, OF, Abs, Teken, ...

The legacy names `english`, `coachtaal` and `dutch` are accepted as aliases for EN and NL. Each language tag `X` corresponds to a file `numodel-X.def` located via `kpse` when the package processes the key. The package ships with `numodel-EN.def` and `numodel-NL.def`; drop your own `numodel-FR.def` (or any other tag) in `TEXMFHOME/tex/latex/numodel/` and select it with `\usepackage[syntax=FR]{numodel}` – no package rebuild needed. The setting affects display only; the expression syntax in `\mrule` bodies is always `\fp_eval`-compatible.

maxiter Safety limit on the number of `\computemodel` iterations (default 20 000). When reached, the simulation aborts with a warning naming the unmet stop condition.

graphscalex Horizontal grid spacing in centimetres for `\graphicmodel`'s Forrester layout (default 2). Larger values spread the diagram out horizontally.

graphscaley Vertical grid spacing in centimetres for `\graphicmodel`'s Forrester layout (default 2). Larger values spread the diagram out vertically.

stockwidth Half-width of stock rectangles in `\graphicmodel` (default 0.375).

gridmaxx Maximum number of grid columns the auto layout may fill on any one row before wrapping (integer, default 0 = no limit). When the limit is reached, items already placed on the affected row (and everything above it for the stocks row, everything but stocks for the aux row, only the constants for the constants row) shift up by one row to free space, and placement continues from column 0. Manually positioned variables (`\mvar[gridx=...,gridy=...]`) are kept where they are. When wrapping is active the default centring of the aux row and the right-aligning of the stocks row are disabled, so the diagram fills left-to-right, bottom-to-top.

diagram-style Rendering style for the case where a helper or constant is the direct inflow/outflow of a stock. Three values:

tight (default) the valve takes the helper's/constant's label; the helper/constant itself is not drawn as a separate node. Compact and most LaTeX-native.

forrester Forrester/Sterman convention: the valve is drawn without a label and the helper/constant remains as a separate node connected to the valve by a causal arrow.

edu Didactic dual form: the valve carries the label *and* the helper/constant is drawn as a separate node with a causal arrow to the valve. Visually busy but pedagogically explicit.

flowarrow-style Visual style of the flow pipe. **hollow** renders the classic Forrester double-line pipe with an open arrow head; **filled** renders a thick solid arrow. The default tracks **diagram-style**: **forrester** picks **hollow**, the other styles pick **filled**. An explicit value overrides this coupling.

valve-style Visual style of the valve node. **valve** draws the bow-tie/butterfly icon (Forrester); **circle** draws an empty circle on the flow pipe; **edu** draws a labelled circle (the flow variable's display text inside). The default tracks **diagram-style**: **forrester** picks **valve**, the other styles pick **edu**.

flowarrow-cloud-tip Whether the open end of an inflow or outflow pipe is anchored to a cloud node, signalling the model boundary. Default tracks **diagram-style**: **forrester** picks **true**, the other styles pick **false**. May be set globally via `\numodelsetup`, per-render via `\graphicmodel`, or per-stock via `\mvar[flowarrow-cloud-tip=...]`. The most specific source wins.

units Whether the *initial values* cells in `\textmodel` display the SI unit alongside the value (`\qty`) or only the numeric value (`\num`). Boolean, default **true**. May also be supplied to `\textmodel[units=false]` as a per-table override; the global setting is restored after rendering.

decimal-separator Decimal mark used by every number that `numodel` renders: the *initial values* column of `\textmodel`, and the tick labels of `\diagrammodel`. Two values:

comma use a comma (siunitx output-decimal-marker={,}, pgfplots/pgf/number format/use comma).

point use a full stop (siunitx output-decimal-marker={.}, pgfplots/pgf/number format/use period).

The default tracks **syntax**: NL picks **comma**, EN picks **point**. Other language files can publish a default by defining `_numodel_kw_<LANG>dsep_default`: (expanding to **point** or **comma**); when the macro is absent the default is **point**. An explicit **decimal-separator** key locks that choice and overrides any future **syntax** change. The override is scoped: `numodel` applies it only inside its own renderers (siunitx's state is restored on group exit), so a document-wide `\sisetup` is not perturbed.

3.1 Diagram styles in practice

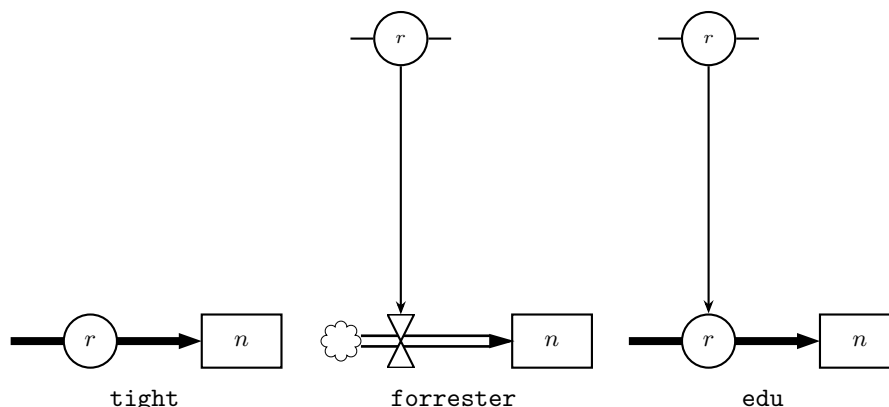
The same model rendered under each of the three **diagram-style** values. The model is the simplest case that distinguishes the styles: one stock *N* with a constant inflow *R*:

```
\newmodelprefix{flux}
\mvar{T}{t}{0}{\s}{2}{system}
\mvar{Dt}{dt}{1}{\s}{2}{system}
\mvar{N}{n}{0}{\s}{0}{stock}
\mvar{R}{r}{5}{\per\s}{2}{constant}
\mrule{N}{\fluxN + \fluxR * \fluxDt}
```

```

\mrule{T}{\fluxT + \fluxDt}
\mstop{\fluxT >= 5}
\graphicmodel[diagram-style=tight]
\graphicmodel[diagram-style=forrester]
\graphicmodel[diagram-style=edu]

```



- **tight** collapses the constant R into the valve label, producing the most compact diagram.
- **forrester** keeps the canonical System-Dynamics convention: unlabelled bow-tie valve, the constant remains a separate node, the link from R to the valve is a thin causal arrow.
- **edu** is a didactic dual: the valve carries the label *and* the constant remains as a separate node with a causal arrow. Less compact but pedagogically explicit – useful when first introducing the stock/flow vocabulary.

4 Public API

4.1 Variables and rules

\mvar Declares a model variable. Signature:

```
\mvar[<keys>]{<Name>}{<text>}{<start>}{<unit>}{<sig>}{<type>}
```

where $\langle Name \rangle$ is a short alphabetic identifier (the prefix-qualified accessor becomes $\langle prefix \rangle \langle Name \rangle$), $\langle text \rangle$ is the math-mode display symbol used in the rule table and diagram (e.g. F_{res}), $\langle start \rangle$ is the initial value (a number, or empty for helpers computed by a rule, or any `expl3` fp-evaluable expression involving previously defined model variables), $\langle unit \rangle$ is a bare `siunitx` unit macro sequence (e.g. $\text{m}\text{per}\text{s}\text{squared}$), $\langle sig \rangle$ is the number of significant figures used by $\langle prefix \rangle \langle Name \rangle \text{num/qty}$, and $\langle type \rangle$ is one of **stock**, **aux**, **constant**, or **system**. Each English type also accepts a Dutch alias (**voorraad**, **hulp**, **constante**, **systeem**) for backwards compatibility with existing teaching material. See Section 5.

Optional $\langle keys \rangle$:

prefix Override the current prefix for this single call.

gridx, **gridy** Manual placement in the `\graphicmodel` grid; integers, -1 leaves the slot to auto-layout (default).

alias Math-mode token list that replaces the entire *initial values* cell.

aliasleft, **aliasright** Replace just the left symbol or right value half of the cell.

\mrule Adds a rule of the form $\langle LHS \rangle \leftarrow \langle expr \rangle$. Signature:

```
\mrule*[<keys>]{<LHS>}{<expr>}
```

Both forms add the rule to *both* the rule table (`\textmodel`) and the simulation (`\computemodel`); execution is identical. The star only changes the typeset layout when $\langle expr \rangle$ is a ternary `cond ? a : b`. Without the star the ternary is rendered on a single table row,

```
IF cond THEN lhs = a ELSE lhs = b ENDIF
```

which is compact but wide. With the star (`\mrule*`) the same ternary is broken across rows,

```
IF cond THEN
    lhs = a
ELSE
    lhs = b
ENDIF
```

keeping the table column narrow so a `\graphicmodel` can sit alongside it, and making the source itself easier to read. For non-ternary expressions the star has no effect. *expr* may use the full `\fp_eval` expression grammar including `+` `-` `*` `/` `^`, `abs`, `sign`, ternary `cond ? a : b`, and Boolean operators `&&` (and) and `||` (or), and the comparison operators `<`, `<=`, `>`, `>=`, `=`, `!=`.

`\mruletext` Inserts a free-text row in the rule table without registering a rule with the simulator. Signature: `\mruletext[<keys>]{<text>}`. Useful for inserting comments or section dividers in long rule tables.

`\mstop` Sets the simulation stop condition. Signature: `\mstop[<keys>]{<expr>}`. The simulation halts at the first step where *expr* evaluates true. Exactly one `\mstop` per model prefix is required before `\computemodel`. Without one, `\computemodel` issues a warning.

4.2 Render commands

`\textmodel` Renders the rule-and-startvalue table. Optional [`<keys>`] accepts `prefix=<name>` (render a non-current model) and either `units=true` or `units=false`, which overrides the global `\numodelsetup` setting for this single render only. The global state is restored afterwards.

`\graphicmodel` Renders the Forrester stock-and-flow diagram. Variables of type `stock` become rectangles, `constant` become circles, `aux` become identifier nodes; flow arrows connect stocks to constant or helper sources/sinks based on which variables appear in the right-hand side of stock-updating rules.

Optional [`<keys>`] accepts `prefix=<name>` (render a non-current model) and `diagram-style=tightforrestered` which overrides the global `\numodelsetup` setting for this single render only. The global state is restored afterwards, so multiple `\graphicmodel` calls can each pick their own style without re-issuing `\numodelsetup`.

`\computemodel` Runs the Euler simulation in Lua. Records every variable's value at every step, plus running min and max. After it returns, the accessors `\<prefix><Name>min` / `\<prefix><Name>max` hold the extrema, and `\<prefix><Name>` holds the final-step value. The time-series can be retrieved with `\mcoords` / `\mstep`.

`\diagrammodel` Convenience wrapper that produces a complete figure with caption and label. Signature:

```
\diagrammodel[<keys>]{<xvar>}{<yvar>}[<extra>]{<label>}
```

Reads min/max and display text from `\<prefix><xvar>` etc., delegates to `\drawplot` from `numodel-plot`, and emits `\caption{$y(x)$-diagram}\label{fig:<label>}`. The optional *extra* argument is appended to the axis body, useful for additional `\addplot` lines (annotations, theoretical curves). Must be called after `\computemodel`.

4.3 Series accessors

`\mcoords` Returns a comma-separated PGFPlots coordinate list of the simulated series for two variables. Fully expandable. Signature: `\mcoords{<xvar>}{<yvar>}` (current prefix); `\mcoordsp{<prefix>}{<xvar>}{<yvar>}` (explicit prefix). Both forms are usable inside `\addplot coordinates{ ... }`.

`\mstep` Returns the value of one variable at a chosen iteration. Fully expandable. Signature: `\mstep{<Name>}{<i>}`; `\mstepp{<prefix>}{<Name>}{<i>}`. The current prefix is prepended automatically by `\mstep`. Step indexing is 0-based: step 0 is the initial-values row, step $N-1$ is the final recorded step after `\computemodel`. Negative indices count from the end (Python-style), so `\mstep{Y}{-1}` is the last recorded *y* and `\mstep{Y}{-2}` is the penultimate one. Returns nothing (silently) if the index is out of range.

Example – a red secant line through the last two simulated (*t*, *y*) points of the free-fall ball (the model from the first example), extrapolated across the whole *t*-domain and labelled in the legend. The slope is the rise-over-run of the last two steps, the intercept is the final *y*-value:

```
\diagrammodel{T}{Y}{%
    \addplot[red, very thick, domain=\ballTmin:\ballTmax]
        {\mstep{Y}{-1}}
```

```

+ (\mstep{Y}{-1} - \mstep{Y}{-2})
/ (\mstep{T}{-1} - \mstep{T}{-2})
* (x - \mstep{T}{-1}));
\addlegendentry{secant through last two steps}
]{ball-fall-secant}

```

Inside the optional [`<extra>`] argument `\mstep` expands as a literal number into PGFPlots' math parser, so each `\mstep` is evaluated only once (when the expression is constructed) rather than per sample. Applied to the `ball` model:

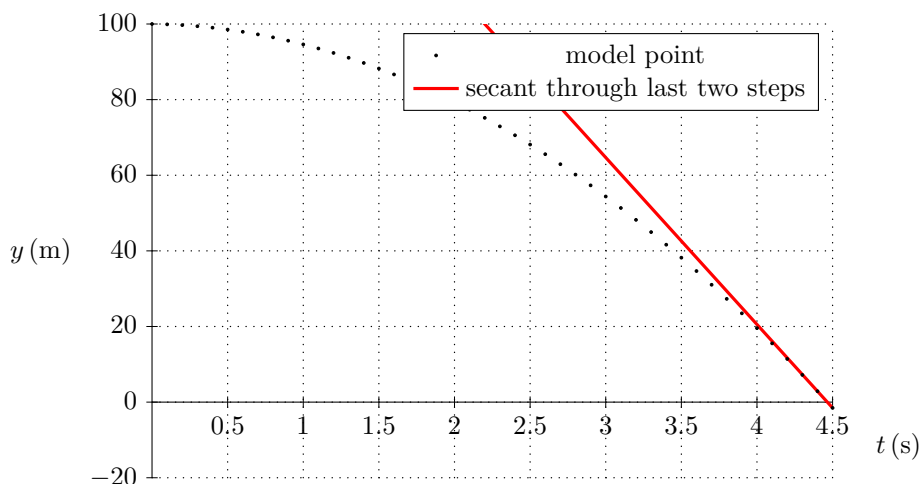


Figure 2: $y(t)$ -diagram

4.4 Namespace management

- `\newmodelprefix` Creates a new model namespace and switches to it. Signature: `\newmodelprefix{<name>}`. Subsequent `\mvar` / `\mrule` / `\mstop` calls bind to this prefix. All generated accessors are prefixed (so `\mvar{Y}{y}{...}{...}{...}{...}` under prefix `ball` produces `\ballY`, `\ballYtext`, etc.).
- `\switchmodelprefix` Switches to a previously created prefix. Useful when a document defines several models early and renders them later out of order. Signature: `\switchmodelprefix{<name>}`.
- `\NumodelForEachVar` Iterates a token list over every registered variable across every known prefix. Signature: `\NumodelForEachVar{<code with #1>}`; inside the body, `#1` is the full prefixed name (e.g. `ballY`). Used by external tools (e.g. a worksheet system) that need to expand every model accessor.

5 Variable types

The sixth `\mvar` argument (*<type>*) tags a variable with a role. The type drives both `\textmodel` layout (whether the row appears in *initial values*) and `\graphicmodel` node shape. Each type has a canonical English name and a Dutch alias; the two are interchangeable.

- stock** (alias *voorraad*) A stock that accumulates over time. Drawn as a rectangle in the Forrester diagram; its rule must be of the form `stock ← stock + ...` so that the integrator can detect inflows and outflows. Receives an *initial values* row.
- constant** (alias *constante*) A constant parameter (mass, gravitational acceleration, spring stiffness, ...). Drawn as a circle. Receives an *initial values* row.
- aux** (alias *hulp*) An auxiliary variable computed from other variables on each step. Drawn as a plain identifier node, no rectangle. No *initial values* row (start value is normally left empty).
- system** (alias *systeem*) System-level bookkeeping (time `T`, step size `Dt`, terminal time, ...). Drawn separately, no flow arrows. Receives an *initial values* row.

6 Generated accessors

Each `\mvar` call generates a family of accessor macros named `\<prefix><Name><suffix>`. The full set:

(no suffix) Current numeric value (post-rule-update if inside a step, post-final-step after `\computemodel`).

text The display symbol passed as the second argument of `\mvar`.

unit `\unit{...}` applied to the unit argument (siunitx-formatted).

unitraw The unit argument verbatim, without the `\unit{}` wrapper, suitable for use as a building block (e.g. `\xlabelunit` in `numodel-plot`).

num The current value formatted via siunitx's `\num{}` with the variable's significant figures. Used by `\textmodel` in the *initial values* column when `units=false`.

qty As `num`, but including the unit (`\qty{value}{unit}`). Used by `\textmodel` in the *initial values* column when `units=true` (the default).

pre As `qty`, but with engineering prefix mode (e.g. `1500 W` renders as `1{,}5\,kW`).

sign Significant-figure count (raw integer).

type Variable type (raw string).

min, max Extrema over the simulated series. Empty before `\computemodel`.

gridx, gridy Manual placement coordinates in the Forrester grid; `-1` if left to auto-layout.

alias, aliasleft, aliasright Override tokens for the *startwaarden* cell layout.

7 Multiple models in one document

Each `\newmodelprefix` starts a fresh namespace. This lets a document contain several unrelated models without name clashes:

```
\newmodelprefix{ball}
\mvar{Y}{y}{100}{\m}{3}{stock}
% ... ball model ...

\newmodelprefix{spring}
\mvar{X}{x}{0.1}{\m}{3}{stock}
% ... spring model ...

% Render the ball model:
\switchmodelprefix{ball}\textmodel\computemodel\diagrammodel{T}{Y}{fall}

% Render the spring model:
\switchmodelprefix{spring}\textmodel\computemodel\diagrammodel{T}{X}{spring}
```

Each prefix carries an independent rule list, stop condition, and recorded series.

8 Requirements

`numodel` requires LuaLaTeX (the engine, for the Lua runtime) and TeX Live 2022 or later. Mandatory dependencies: `expl3`, `xparse`, `l3keys2e`, `amsmath`, `amssymb`, `tikz`, `luacode`, `siunitx`, `float`, and the sibling package `numodel-plot` (which itself pulls in `pgfplots`). The companion Lua module `numodel.lua` must be installed alongside the `.sty` in a directory searched by `kpse`.

9 Implementation

```

1 (*package)
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{numodel}[2026/05/17 v0.3.0
4   Numerical physics models with Euler integration and Forrester diagrams]
5
6 \RequirePackage{expl3}
7 \RequirePackage{xparse}
8 \RequirePackage{l3keys2e}
9 \RequirePackage{amsmath}
10 \RequirePackage{amssymb}          % \leqslant / \geqslant in rendered <= / >=
11 \RequirePackage{tikz}
12 \usetikzlibrary{shapes.symbols}
13 \usetikzlibrary{arrows.meta}
14 % Custom Cloud arrow tip used as the open end of inflow / outflow
15 % pipes. The silhouette consists of nine concatenated elliptical
16 % arcs (\pgfpatharc): each arc starts at the current path point,
17 % which pgf treats as lying on an ellipse at angle <start>, and
18 % sweeps through to angle <end>. Varying the start and end angles
19 % per arc produces zigzagging half-circle puffs -- the
20 % characteristic cloud outline. A single factor \puffscale sets the
21 % rx and ry of every ellipse and thus the size of all puffs at once.
22 % The chain has a horizontal span of (4+2*sqrt(2))*rx ~ 6.83*rx;
23 % with puffscale=0.22 that is ~ 1.5*\pgfarrowlength. The path
24 % therefore starts at -0.5*\pgfarrowlength so the silhouette covers
25 % [-0.5L, L] in local arrow coordinates and the apex lands exactly
26 % at \pgfarrowlength -- where the built-in tips (Latex, Stealth,
27 % ...) also place their tip. The line width is set explicitly to
28 % 0.3pt, independent of the arrow's line width, so a thick arrow
29 % does not also get a thick cloud outline.
30 \pgfdeclarearrow{
31   name = Cloud,
32   parameters = { \the\pgfarrowlength \the\pgfarrowwidth },
33   setup code = {
34     \pgfarrowssettipend{\pgfarrowlength}
35     \pgfarrowssetbackend{-0.5\pgfarrowlength}
36     \pgfarrowssetlineend{-0.5\pgfarrowlength}
37     \pgfarrowssetvisualbackend{-0.5\pgfarrowlength}
38     \pgfarrowssetvisualtipend{\pgfarrowlength}
39   },
40   defaults = { length = 1em, width = 1em },
41   drawing code = {
42     \def\puffscale{0.22}
43     \edef\puffrx{\puffscale\pgfarrowlength}
44     \edef\puffry{\puffscale\pgfarrowwidth}
45     \pgfsetlinewidth{0.3pt}
46     \pgfpathmoveto{\pgfqpoint{-0.5\pgfarrowlength}{0pt}}
47     \pgfpatharc{180}{ 90}{\puffrx\space and \puffry}
48     \pgfpatharc{225}{ 45}{\puffrx\space and \puffry}
49     \pgfpatharc{180}{ 0}{\puffrx\space and \puffry}
50     \pgfpatharc{135}{-45}{\puffrx\space and \puffry}
51     \pgfpatharc{ 90}{-90}{\puffrx\space and \puffry}
52     \pgfpatharc{ 45}{-135}{\puffrx\space and \puffry}
53     \pgfpatharc{ 0}{-180}{\puffrx\space and \puffry}
54     \pgfpatharc{-45}{-225}{\puffrx\space and \puffry}
55     \pgfpatharc{-90}{-180}{\puffrx\space and \puffry}
56     \pgfpathclose
57     \pgfusepathqfillstroke
58   },
59 }
60 \RequirePackage{luacode}
61 \RequirePackage{siunitx}
62 \RequirePackage{float}          % \diagrammodel uses [H] placement
63 \RequirePackage{numodel-plot}
64

```

```

65 % =====
66 % Non-expl3 helpers: ` \nm@def@num@cs` / ` \nm@def@qty@cs` / ` \nm@def@pre@cs`
67 % define ` \<fullname><suffix>` macros whose body contains literal
68 % siunitx options. Defined OUTSIDE \ExplSyntaxOn so the option-string
69 % colons (`-3:n`, `-0:0`) and other punctuation keep normal catcodes.
70 % Storing those bodies inside an expl3-context would tokenise `` as
71 % letter, which then breaks siunitx's option parser at expansion time.
72 % =====
73 \newcommand{\NumodelDefNumCs}[3]{% #1 = fullname, #2 = start expr, #3 = sigfigs
74   \expandafter\gdef\csname #1num\endcsname{%
75     \num[evaluate-expression=true,
76       round-mode=figures, round-precision=#3,
77       exponent-mode=threshold, exponent-thresholds=-3:#3]
78     {#2}\relax
79   }%
80 }
81 \newcommand{\NumodelDefQtyCs}[4]{% #1 = fullname, #2 = expr, #3 = sigfigs, #4 = unit
82   \expandafter\gdef\csname #1qty\endcsname{%
83     \qty[evaluate-expression=true,
84       round-mode=figures, round-precision=#3,
85       exponent-mode=threshold, exponent-thresholds=-3:#3]
86     {#2}{#4}\relax
87   }%
88 }
89 \newcommand{\NumodelDefPreCs}[4]{% same parameters as above
90   \expandafter\gdef\csname #1pre\endcsname{%
91     \qty[evaluate-expression=true,
92       round-mode=figures, round-precision=#3,
93       prefix-mode=combine-exponent,
94       exponent-mode=engineering,
95       exponent-thresholds=-0:0]
96     {#2}{#4}\relax
97   }%
98 }
99
100 \ExplSyntaxOn
101
102 % =====
103 % Lua module for data storage (O(1) append, O(N) total)
104 % =====
105 % All variable values are stored per step in Lua tables. This
106 % replaces the O(N^2) \xdef accumulation for coordinates and makes
107 % min/max tracking efficient. After \computemodel the results are
108 % pushed back to TeX macros.
109
110 \begin{luacode*}
111 local f = kpse.find_file("numodel.lua", "tex")
112 if f and f ~= "" then
113   dofile(f)
114 else
115   tex.error("numodel: cannot find companion Lua module numodel.lua."
116     .. " Install it in a directory searched by kpse"
117     .. " (e.g. TEXMFHOME/tex/lualatex/numodel/).")
118 end
119 \end{luacode*}
120
121 % =====
122 % Internal data structures
123 % =====
124
125 \seq_new:N \g_mvar_names_seq      % all model variable names
126 \seq_new:N \g_mvar_start_seq      % names with an initial value (for the table)
127 \seq_new:N \g_mrulerule_seq      % model rules (display strings)
128 \seq_new:N \g_mrulerule_type_seq % type per display row: rule | cont
129 \seq_new:N \g_mrulerule_calc_seq % model rules for execution: {name}{expr}
130 \int_new:N \g_mrulerule_counter_int % rule counter (display numbering)

```

```

131 \tl_new:N \g_numodel_stop_expr_tl % stop-condition expression for execution
132 \int_new:N \g_numodel_steps_int % number of executed steps
133 \int_new:N \g_numodel_maxiter_int % safety limit (init via \numodelsetup)
134 \int_new:N \g_numodel_gridmaxx_int % \graphicmodel wrap threshold (0 = off)
135
136 % --- Graphic-model infrastructure (Phase 1) ---
137 \tl_new:N \l_numodel_tmp_tl % temporary helper
138 \tl_new:N \l_numodel_scratch_tl % scratch for type/text checks
139 \tl_new:N \l_numodel_scratch_y_tl % scratch y-coord (svgy lookup)
140
141 % Diagram style for \graphicmodel (see \numodelsetup):
142 % tight -- valve takes the label of the direct inflow variable;
143 % the aux/const is moved to the valve position and not
144 % drawn as a separate node (compact, default).
145 % forrester -- valve has no label; the aux/const stays at its own
146 % gridy position; causal arrow from aux/const to the
147 % valve.
148 % edu -- combination: valve takes the label *and* a separate
149 % aux/const node remains with a causal arrow to it.
150 \tl_new:N \g_numodel_diagram_style_tl
151
152 % Sub-keys that fine-tune the appearance of the Forrester diagram
153 % (see \numodelsetup). Empty value = "follow diagram-style default".
154 % flowarrow-style : hollow | filled
155 % forrester -> hollow, tight|edu -> filled
156 % valve-style : valve | circle | edu
157 % forrester -> valve, tight|edu -> edu
158 % flowarrow-cloud-tip: true | false
159 % forrester -> true, tight|edu -> false
160 % Per-variable override for flowarrow-cloud-tip is stored in
161 % \<name>flowcloud (set via the [flowarrow-cloud-tip=...] key on
162 % \mvar; empty means "follow global key").
163 \tl_new:N \g_numodel_flowarrow_style_tl
164 \tl_new:N \g_numodel_valve_style_tl
165 \tl_new:N \g_numodel_flowcloud_tl
166
167 % Units in the \textmodel initial-values column (see \numodelsetup):
168 % true (default) -- initial value rendered via \<name>qty
169 % (number + unit)
170 % false -- initial value rendered via \<name>num
171 % (number only)
172 % Per-table override via \textmodel[units=true|false].
173 \bool_new:N \g_numodel_units_bool
174
175 % Decimal separator for \textmodel initial values and \diagrammodel
176 % tick values (see \numodelsetup{decimal-separator}):
177 % point -- '.' (siunitx output-decimal-marker={.})
178 % comma -- ',' (siunitx output-decimal-marker={,})
179 % The default follows syntax: english -> point, coachtaal -> comma.
180 % An explicit decimal-separator key sets
181 % \g_numodel_dsep_explicit_bool so that a later syntax change no
182 % longer overrules the choice.
183 \tl_new:N \g_numodel_dsep_tl
184 \bool_new:N \g_numodel_dsep_explicit_bool
185
186 % =====
187 % Syntax lookup
188 % =====
189 % Determines the language of the text-rendered model. Affects display
190 % only (\textmodel, \mruletext, \mstop); \computemodel always uses
191 % \fpeval internally and is language-agnostic. Each value of
192 % \g_numodel_syntax_tl is a tag that selects both the backing
193 % \_numodel_kw_<tag>_<key>: macros and the file numodel-<tag>.def
194 % that defines them; the package ships EN (English/XMILE) and NL
195 % (Dutch/CoachTaal).
196 %

```

```

197 % The CTAN default is EN. The initial value is set below via
198 % \numodelsetup.
199
200 \tl_new:N \g__numodel_syntax_tl
201
202 % Lookup (fully expandable): \__numodel_kw:n {<key>} returns the
203 % translation in the current syntax language. The key also controls
204 % the surrounding spaces (see keys 'then' vs 'then_nl' etc.). The
205 % backing macros \__numodel_kw_<LANG>_<key>: are provided by the
206 % language file numodel-<LANG>.def loaded by \__numodel_load_syntax_def:n
207 % (see below).
208 \cs_new:Npn \__numodel_kw:n #1
209 { \use:c { __numodel_kw_ \g__numodel_syntax_tl _ #1 : } }
210
211 % Pre-computed translations in tl variables so they are usable via
212 % \u{ } in l3regex replacement text. (l3regex treats {...} in the
213 % replacement as brace groups, so a bare \__numodel_kw:n{key} call
214 % fails there.)
215 \tl_new:N \g__numodel_kw_if_tl
216 \tl_new:N \g__numodel_kw_then_tl
217 \tl_new:N \g__numodel_kw_then_nl_tl
218 \tl_new:N \g__numodel_kw_else_tl
219 \tl_new:N \g__numodel_kw_else_nl_tl
220 \tl_new:N \g__numodel_kw_endif_tl
221 \tl_new:N \g__numodel_kw_endif_nl_tl
222 \tl_new:N \g__numodel_kw_and_tl
223 \tl_new:N \g__numodel_kw_or_tl
224 \tl_new:N \g__numodel_kw_not_tl
225 \tl_new:N \g__numodel_kw_sign_tl
226 \tl_new:N \g__numodel_kw_abs_tl
227 \tl_new:N \g__numodel_kw_stop_tl
228
229 % Recomputes all tl caches from \g__numodel_syntax_tl. Must be
230 % called after any change to the syntax language.
231 \cs_new_protected:Npn \__numodel_refresh_kw:
232 {
233   \tl_gset:Ne \g__numodel_kw_if_tl { \__numodel_kw:n {if} }
234   \tl_gset:Ne \g__numodel_kw_then_tl { \__numodel_kw:n {then} }
235   \tl_gset:Ne \g__numodel_kw_then_nl_tl { \__numodel_kw:n {then_nl} }
236   \tl_gset:Ne \g__numodel_kw_else_tl { \__numodel_kw:n {else} }
237   \tl_gset:Ne \g__numodel_kw_else_nl_tl { \__numodel_kw:n {else_nl} }
238   \tl_gset:Ne \g__numodel_kw_endif_tl { \__numodel_kw:n {endif} }
239   \tl_gset:Ne \g__numodel_kw_endif_nl_tl { \__numodel_kw:n {endif_nl} }
240   \tl_gset:Ne \g__numodel_kw_and_tl { \__numodel_kw:n {and} }
241   \tl_gset:Ne \g__numodel_kw_or_tl { \__numodel_kw:n {or} }
242   \tl_gset:Ne \g__numodel_kw_not_tl { \__numodel_kw:n {not} }
243   \tl_gset:Ne \g__numodel_kw_sign_tl { \__numodel_kw:n {sign} }
244   \tl_gset:Ne \g__numodel_kw_abs_tl { \__numodel_kw:n {abs} }
245   \tl_gset:Ne \g__numodel_kw_stop_tl { \__numodel_kw:n {stop} }
246 }
247
248 % Helper: \__numodel_kwt:n {<key>} expands to \text{<kw-value>}.
249 % Meant for use inside \tl_set:Ne / \seq_gput_right:Ne -- the kw is
250 % inserted during e-expansion while \text remains protected.
251 \cs_new:Npn \__numodel_kwt:n #1 { \text { \__numodel_kw:n {#1} } }
252
253 % Language-file loader. Each tag <LANG> is backed by a file
254 % numodel-<LANG>.def installed in a kpse-searched directory; the
255 % package ships numodel-EN.def and numodel-NL.def, users can drop
256 % additional files in TEXMFHOME/tex/latex/numodel/ without rebuilding
257 % the package.
258 %
259 % A small alias property maps the historical long names to the
260 % canonical two-letter tag used in the file name and the internal
261 % macro names. Users who add their own file just pass the file's tag
262 % to syntax=<tag> directly; no alias is required.

```

```

263 \prop_new:N \g__numodel_syntax_aliases_prop
264 \prop_gput:Nnn \g__numodel_syntax_aliases_prop { english } { EN }
265 \prop_gput:Nnn \g__numodel_syntax_aliases_prop { coachtaal } { NL }
266 \prop_gput:Nnn \g__numodel_syntax_aliases_prop { dutch } { NL }
267
268 % Tags whose .def file has already been input during this run, so the
269 % lookup only triggers \InputIfFileExists the first time.
270 \seq_new:N \g__numodel_loaded_syntax_seq
271
272 \msg_new:nnn { numodel } { unknown-syntax }
273 {
274   Cannot-load-syntax~'#1':~file~'numodel-#1.def'~not~found~by~
275   kpse.~The~package~ships~with~EN~(English)~and~NL~(Dutch~
276   CoachTaal);~drop~your~own~numodel-<LANG>.def~in~
277   TEXMFHOME/tex/latex/numodel/~to~add~more~languages.
278 }
279
280 % \__numodel_load_syntax_def:n {<tag>}
281 % Inputs numodel-<tag>.def the first time it is requested. Raises a
282 % LaTeX error if the file is not found by kpse.
283 \cs_new_protected:Npn \__numodel_load_syntax_def:n #1
284 {
285   \seq_if_in:NnF \g__numodel_loaded_syntax_seq {#1}
286   {
287     \InputIfFileExists { numodel-#1.def }
288     { \seq_gput_right:Nn \g__numodel_loaded_syntax_seq {#1} }
289     { \msg_error:nnn { numodel } { unknown-syntax } {#1} }
290   }
291 }
292
293 \tl_new:N \l__numodel_syntax_tag_tl
294
295 % \__numodel_set_syntax:n {<value>}
296 % Public-facing setter behind the syntax key. Resolves aliases,
297 % loads the matching .def file (once), publishes the canonical tag
298 % in \g__numodel_syntax_tl, refreshes the keyword cache, and -- when
299 % the user has not pinned a decimal separator explicitly -- adopts the
300 % language's preferred decimal mark via the optional
301 % \__numodel_kw_<LANG>_dsep_default: hook.
302 \cs_new_protected:Npn \__numodel_set_syntax:n #1
303 {
304   \prop_get:NnNF \g__numodel_syntax_aliases_prop {#1}
305   \l__numodel_syntax_tag_tl
306   { \tl_set:Nn \l__numodel_syntax_tag_tl {#1} }
307   \exp_args:NV \__numodel_load_syntax_def:n \l__numodel_syntax_tag_tl
308   \tl_gset_eq:NN \g__numodel_syntax_tl \l__numodel_syntax_tag_tl
309   \__numodel_refresh_kw:
310   \bool_if:NF \g__numodel_dsep_explicit_bool
311   {
312     \cs_if_exist:cTF
313     { __numodel_kw_ \g__numodel_syntax_tl _dsep_default: }
314     { \tl_gset:Ne \g__numodel_dsep_tl { \__numodel_kw:n {dsep_default} } }
315     { \tl_gset:Nn \g__numodel_dsep_tl { point } }
316   }
317 }
318
319 % Applies the decimal separator (only for the duration of the
320 % surrounding TeX group, so a document-wide \sisetup is left
321 % untouched). Calls both siunitx (for \num/\qty in \textmodel
322 % initial values) and pgfplots' tick styles (for \diagrammodel tick
323 % labels) so the choice is consistent everywhere.
324 \cs_new_protected:Npn \__numodel_apply_dsep:
325 {
326   \str_if_eq:VnTF \g__numodel_dsep_tl { comma }
327   {
328     \sisetup{ output-decimal-marker = {,} }

```

```

329 % Append behind the existing numodel/axis style so our
330 % xticklabel-style replaces what numodel-plot hard-codes.
331 % The \pgfplotsset mutation is \def-based and hence
332 % group-local.
333 \pgfplotsset
334 {
335     numodel/axis/.append-style=
336     {
337         xticklabel-style=
338         { /pgf/number-format/.cd, use-comma, /tikz/.cd } ,
339         yticklabel-style=
340         { /pgf/number-format/.cd, use-comma, /tikz/.cd } ,
341     }
342 }
343 }
344 {
345     \sisetup{ output-decimal-marker = {.} }
346     \pgfplotsset
347     {
348         numodel/axis/.append-style=
349         {
350             xticklabel-style=
351             { /pgf/number-format/.cd, use-period, /tikz/.cd } ,
352             yticklabel-style=
353             { /pgf/number-format/.cd, use-period, /tikz/.cd } ,
354         }
355     }
356 }
357 }
358
359 % =====
360 % Configuration command \numodelsetup
361 % =====
362 % Runtime API for settings. Keys:
363 % syntax      -- language tag (file numodel-<tag>.def must be on kpse
364 %              path). Built-in: EN, NL. Legacy aliases: english,
365 %              coachtaal, dutch.
366 % maxiter     -- safety limit for \computemodel (default 20000)
367 % graphscalex -- horizontal grid spacing \dgridx for Forrester
368 %              diagrams (default 2)
369 % graphscaley -- vertical grid spacing \dgridy for Forrester
370 %              diagrams (default 2)
371 % stockwidth  -- half-width of the stock rectangle (default 0.375)
372
373 % Helper for choice-with-empty-reset: validates #4 against #3 (a
374 % comma-separated whitelist) and writes it into #1 (a tl). An empty
375 % value clears the tl (which means "follow the diagram-style
376 % default" for the flowarrow-style / valve-style /
377 % flowarrow-cloud-tip keys). Any other value triggers a warning.
378 \msg_new:nnn { numodel } { bad-choice }
379 { Key~'#1'~accepts-only~#2,~or~empty~to~reset.~Got:~'#3'. }
380
381 \cs_new_protected:Npn \__numodel_setup_choice:Nnnn #1 #2 #3 #4
382 {
383     \tl_if_blank:nTF {#4}
384     { \tl_gclear:N #1 }
385     {
386         \clist_if_in:nnTF {#3} {#4}
387         { \tl_gset:Nn #1 {#4} }
388         { \msg_warning:nnnnn { numodel } { bad-choice } {#2} {#3} {#4} }
389     }
390 }
391
392 % Local-tl variant: same validation, local set/clear. Used by the
393 % \graphicmodel one-render override and the per-\mvar override.
394 \cs_new_protected:Npn \__numodel_local_choice:Nnnn #1 #2 #3 #4

```

```

395 {
396   \tl_if_blank:nTF {#4}
397   { \tl_clear:N #1 }
398   {
399     \clist_if_in:nnTF {#3} {#4}
400     { \tl_set:Nn #1 {#4} }
401     { \msg_warning:nnnnn { numodel } { bad-choice } {#2} {#3} {#4} }
402   }
403 }
404
405 \keys_define:nn { numodel / setup }
406 {
407   syntax .code:n =
408   { \__numodel_set_syntax:n {#1} },
409   maxiter .int_gset:N = \g_numodel_maxiter_int,
410   graphscalex .code:n = { \tl_gset:Nn \dgridx {#1} },
411   graphscaley .code:n = { \tl_gset:Nn \dgridy {#1} },
412   stockwidth .code:n = { \tl_gset:Nn \halfstockwidth {#1} },
413   gridmaxx .int_gset:N = \g_numodel_gridmaxx_int,
414   diagram-style .choice:,
415   diagram-style / tight .code:n =
416   { \tl_gset:Nn \g_numodel_diagram_style_tl { tight } },
417   diagram-style / forrester .code:n =
418   { \tl_gset:Nn \g_numodel_diagram_style_tl { forrester } },
419   diagram-style / edu .code:n =
420   { \tl_gset:Nn \g_numodel_diagram_style_tl { edu } },
421   flowarrow-style .code:n =
422   { \__numodel_setup_choice:Nnnn \g_numodel_flowarrow_style_tl
423     { flowarrow-style } { hollow , filled } {#1} },
424   valve-style .code:n =
425   { \__numodel_setup_choice:Nnnn \g_numodel_valve_style_tl
426     { valve-style } { valve , circle , edu } {#1} },
427   flowarrow-cloud-tip .code:n =
428   { \__numodel_setup_choice:Nnnn \g_numodel_flowcloud_tl
429     { flowarrow-cloud-tip } { true , false } {#1} },
430   units .choice:,
431   units / true .code:n =
432   { \bool_gset_true:N \g_numodel_units_bool },
433   units / false .code:n =
434   { \bool_gset_false:N \g_numodel_units_bool },
435   decimal-separator .choice:,
436   decimal-separator / comma .code:n =
437   {
438     \tl_gset:Nn \g_numodel_dsep_tl { comma }
439     \bool_gset_true:N \g_numodel_dsep_explicit_bool
440   },
441   decimal-separator / point .code:n =
442   {
443     \tl_gset:Nn \g_numodel_dsep_tl { point }
444     \bool_gset_true:N \g_numodel_dsep_explicit_bool
445   },
446 }
447
448 \NewDocumentCommand{\numodelsetup}{m}
449 { \keys_set:nn { numodel / setup } {#1} }
450
451 % Defaults (this call also initialises \g_numodel_syntax_tl,
452 % \g_numodel_maxiter_int, \dgridx, \dgridy, \halfstockwidth and
453 % \g_numodel_diagram_style_tl).
454 \numodelsetup
455 {
456   syntax = EN,
457   maxiter = 20000,
458   graphscalex = 2,
459   graphscaley = 2,
460   stockwidth = 0.375,

```

```

461     diagram-style = tight,
462     units          = true,
463 }
464
465 % Package-time options. \usepackage[syntax=NL]{numodel} delegates
466 % to the same key infrastructure as \numodelsetup.
467 \keys_define:nn { numodel / pkg }
468 {
469     syntax          .meta:nn = { numodel / setup }{ syntax          = #1 },
470     maxiter         .meta:nn = { numodel / setup }{ maxiter         = #1 },
471     graphscalex     .meta:nn = { numodel / setup }{ graphscalex     = #1 },
472     graphscaley     .meta:nn = { numodel / setup }{ graphscaley     = #1 },
473     stockwidth      .meta:nn = { numodel / setup }{ stockwidth      = #1 },
474     gridmaxx        .meta:nn = { numodel / setup }{ gridmaxx        = #1 },
475     diagram-style   .meta:nn = { numodel / setup }{ diagram-style   = #1 },
476     flowarrow-style .meta:nn = { numodel / setup }{ flowarrow-style = #1 },
477     valve-style     .meta:nn = { numodel / setup }{ valve-style     = #1 },
478     flowarrow-cloud-tip .meta:nn = { numodel / setup }{ flowarrow-cloud-tip = #1 },
479     units           .meta:nn = { numodel / setup }{ units           = #1 },
480     decimal-separator .meta:nn = { numodel / setup }{ decimal-separator = #1 },
481 }
482 \ProcessKeyOptions [ numodel / pkg ]
483
484 % =====
485 % Prefix system
486 % =====
487 % Each model lives under a prefix (a short string). The "live state"
488 % sits in the \g_mvar_*, \g_mrul_*, \g_numodel_* variables above;
489 % \newmodelprefix and \switchmodelprefix swap that state to/from
490 % per-prefix backup storage.
491 %
492 % - \g_numodel_current_prefix_tl: current prefix (empty at package load)
493 % - \g_numodel_prefixes_seq: list of all registered prefixes
494 % - \l__numodel_cmd_prefix_tl: prefix passed via [prefix=...] key
495 % - \l__numodel_eff_prefix_tl: effective prefix for the current command
496
497 \tl_new:N \g_numodel_current_prefix_tl
498 \seq_new:N \g_numodel_prefixes_seq
499 \tl_new:N \l__numodel_cmd_prefix_tl
500 \tl_new:N \l__numodel_eff_prefix_tl
501 \tl_new:N \l__numodel_fullname_tl
502
503 % Per-prefix storage: on swap the current state is copied to
504 % \g__numodel_<P>_<slot>_{seq,tl,int,prop}. Load goes the other way.
505 \cs_new_protected:Npn \__numodel_save_state:n #1
506 {
507     \seq_gset_eq:cN { g__numodel_ #1 _vars_seq      } \g_mvar_names_seq
508     \seq_gset_eq:cN { g__numodel_ #1 _starts_seq     } \g_mvar_start_seq
509     \seq_gset_eq:cN { g__numodel_ #1 _rules_seq      } \g_mrul_seq
510     \seq_gset_eq:cN { g__numodel_ #1 _ruletypes_seq  } \g_mrul_type_seq
511     \seq_gset_eq:cN { g__numodel_ #1 _rulecalc_seq   } \g_mrul_calc_seq
512     \int_gset:cn { g__numodel_ #1 _rulecounter_int }
513     { \int_use:N \g_mrul_counter_int }
514     \tl_gset_eq:cN { g__numodel_ #1 _stopexpr_tl    } \g_numodel_stop_expr_tl
515     \int_gset:cn { g__numodel_ #1 _steps_int        }
516     { \int_use:N \g_numodel_steps_int }
517 }
518
519 \cs_new_protected:Npn \__numodel_load_state:n #1
520 {
521     \seq_gset_eq:Nc \g_mvar_names_seq { g__numodel_ #1 _vars_seq      }
522     \seq_gset_eq:Nc \g_mvar_start_seq { g__numodel_ #1 _starts_seq     }
523     \seq_gset_eq:Nc \g_mrul_seq       { g__numodel_ #1 _rules_seq      }
524     \seq_gset_eq:Nc \g_mrul_type_seq  { g__numodel_ #1 _ruletypes_seq  }
525     \seq_gset_eq:Nc \g_mrul_calc_seq  { g__numodel_ #1 _rulecalc_seq   }
526     \int_gset:Nn \g_mrul_counter_int

```



```

527     { \int_use:c { g__numodel_ #1 _rulecounter_int } }
528     \tl_gset_eq:Nc \g_numodel_stop_expr_tl { g__numodel_ #1 _stopexpr_tl }
529     \int_gset:Nn \g_numodel_steps_int
530     { \int_use:c { g__numodel_ #1 _steps_int } }
531 }
532
533 \cs_new_protected:Npn \__numodel_clear_state:
534 {
535     \seq_gclear:N \g_mvar_names_seq
536     \seq_gclear:N \g_mvar_start_seq
537     \seq_gclear:N \g_mrulseq
538     \seq_gclear:N \g_mrulseq_type_seq
539     \seq_gclear:N \g_mrulseq_calc_seq
540     \int_gzero:N \g_mrulseq_counter_int
541     \tl_gclear:N \g_numodel_stop_expr_tl
542     \int_gzero:N \g_numodel_steps_int
543 }
544
545 % Initialise per-prefix backup variables (once, at \newmodelprefix).
546 \cs_new_protected:Npn \__numodel_init_backup:n #1
547 {
548     \seq_new:c { g__numodel_ #1 _vars_seq }
549     \seq_new:c { g__numodel_ #1 _starts_seq }
550     \seq_new:c { g__numodel_ #1 _rules_seq }
551     \seq_new:c { g__numodel_ #1 _ruletypes_seq }
552     \seq_new:c { g__numodel_ #1 _rulecalc_seq }
553     \int_new:c { g__numodel_ #1 _rulecounter_int }
554     \tl_new:c { g__numodel_ #1 _stopexpr_tl }
555     \int_new:c { g__numodel_ #1 _steps_int }
556 }
557
558 % Public commands for prefix management
559 \NewDocumentCommand{\newmodelprefix}{ m }
560 {
561     \typeout{NEWPREFIX~start:~#1}
562     \seq_if_in:NnTF \g_numodel_prefixes_seq {#1}
563     { \msg_error:nnn { numodel } { prefix-exists } {#1} }
564     {
565         \typeout{NEWPREFIX~save~current:~\g_numodel_current_prefix_tl}
566         % Save current state under the previous prefix (if any)
567         \tl_if_empty:NF \g_numodel_current_prefix_tl
568         { \exp_args:NV \__numodel_save_state:n \g_numodel_current_prefix_tl }
569         \typeout{NEWPREFIX~register}
570         % Register new prefix + init backup vars + Lua state
571         \seq_gput_right:Nn \g_numodel_prefixes_seq {#1}
572         \typeout{NEWPREFIX~init~backup}
573         \__numodel_init_backup:n {#1}
574         \typeout{NEWPREFIX~lua~init}
575         \directlua{ numodel.init_prefix("#1") }
576         \typeout{NEWPREFIX~clear~state}
577         % Clear current state and set the prefix
578         \__numodel_clear_state:
579         \typeout{NEWPREFIX~set~current}
580         \tl_gset:Nn \g_numodel_current_prefix_tl {#1}
581         \typeout{NEWPREFIX~done:~#1}
582     }
583 }
584
585 \NewDocumentCommand{\switchmodelprefix}{ m }
586 {
587     \seq_if_in:NnTF \g_numodel_prefixes_seq {#1}
588     {
589         % Save current state (if any), load the new one
590         \tl_if_empty:NF \g_numodel_current_prefix_tl
591         { \exp_args:NV \__numodel_save_state:n \g_numodel_current_prefix_tl }
592         \__numodel_load_state:n {#1}

```

```

593     \tl_gset:Nn \g_numodel_current_prefix_tl {#1}
594   }
595   { \msg_error:nnn { numodel } { prefix-unknown } {#1} }
596 }
597
598 % Iterate over all variables from all prefixes.
599 % #1 = code that uses ##1 = full variable name (prefix + short).
600 % Public registration API for external tools such as worksheet.tex.
601 \NewDocumentCommand{\NumodelForEachVar}{+m }
602 {
603   \seq_map_inline:Nn \g_numodel_prefixes_seq
604   {
605     \seq_map_inline:cn { g__numodel_ ##1 _vars_seq } {#1}
606   }
607 }
608
609 % Resolve the effective prefix for a command call. Takes
610 % [prefix=<p>] from the keyval argument; otherwise falls back to the
611 % current prefix. Stores the result in \l__numodel_eff_prefix_tl.
612 \keys_define:nn { numodel / cmd }
613 {
614   prefix .tl_set:N = \l__numodel_cmd_prefix_tl,
615   % \graphicmodel-only: temporary override of the global
616   % diagram-style. Cleared after every \graphicmodel call.
617   diagram-style .choice:,
618   diagram-style / tight .code:n =
619     { \tl_set:Nn \l__numodel_cmd_diagstyle_tl { tight } },
620   diagram-style / forrester .code:n =
621     { \tl_set:Nn \l__numodel_cmd_diagstyle_tl { forrester } },
622   diagram-style / edu .code:n =
623     { \tl_set:Nn \l__numodel_cmd_diagstyle_tl { edu } },
624   % \graphicmodel-only: temporary overrides of the corresponding
625   % global flowarrow/valve/cloud keys. Empty after every call.
626   flowarrow-style .code:n =
627     { \__numodel_local_choice:Nnnn \l__numodel_cmd_flowarrow_tl
628       { flowarrow-style } { hollow , filled } {#1} },
629   valve-style .code:n =
630     { \__numodel_local_choice:Nnnn \l__numodel_cmd_valve_tl
631       { valve-style } { valve , circle , edu } {#1} },
632   flowarrow-cloud-tip .code:n =
633     { \__numodel_local_choice:Nnnn \l__numodel_cmd_flowcloud_tl
634       { flowarrow-cloud-tip } { true , false } {#1} },
635   % \textmodel-only: temporary override of the global units bool.
636   % Cleared after every \textmodel call.
637   units .choice:,
638   units / true .code:n =
639     { \tl_set:Nn \l__numodel_cmd_units_tl { true } },
640   units / false .code:n =
641     { \tl_set:Nn \l__numodel_cmd_units_tl { false } },
642 }
643 \tl_new:N \l__numodel_cmd_diagstyle_tl
644 \tl_new:N \l__numodel_cmd_flowarrow_tl
645 \tl_new:N \l__numodel_cmd_valve_tl
646 \tl_new:N \l__numodel_cmd_flowcloud_tl
647 \tl_new:N \l__numodel_cmd_units_tl
648
649 \cs_new_protected:Npn \__numodel_resolve_prefix:n #1
650 {
651   \tl_clear:N \l__numodel_cmd_prefix_tl
652   \keys_set:nn { numodel / cmd } {#1}
653   \tl_if_empty:NTF \l__numodel_cmd_prefix_tl
654     { \tl_set_eq:NN \l__numodel_eff_prefix_tl \g_numodel_current_prefix_tl }
655     { \tl_set_eq:NN \l__numodel_eff_prefix_tl \l__numodel_cmd_prefix_tl }
656 }
657
658 % Execute code under a temporarily different prefix (via state swap).

```

```

659 % #1 = target prefix (tl, in variable)
660 % #2 = code
661 \cs_new_protected:Npn \__numodel_with_prefix:Nn #1 #2
662 {
663   \tl_if_eq:NNTF #1 \g_numodel_current_prefix_tl
664   { #2 } % already the current prefix; no swap needed
665   {
666     \tl_set_eq:NN \l__numodel_saved_prefix_tl \g_numodel_current_prefix_tl
667     \exp_args:NV \switchmodelprefix #1
668     #2
669     \exp_args:NV \switchmodelprefix \l__numodel_saved_prefix_tl
670   }
671 }
672 \tl_new:N \l__numodel_saved_prefix_tl
673
674 % Build the full name = <prefix><shortname>. Stored in
675 % \l__numodel_fullname_tl.
676 \cs_new_protected:Npn \__numodel_set_fullname:n #1
677 {
678   \tl_set:Ne \l__numodel_fullname_tl { \l__numodel_eff_prefix_tl #1 }
679 }
680
681 % Keys for the optional argument of \mvar (grid position +
682 % initial-value aliases + prefix).
683 \tl_new:N \l__numodel_alias_tl
684 \tl_new:N \l__numodel_aliasleft_tl
685 \tl_new:N \l__numodel_aliasright_tl
686
687 \keys_define:nn { numodel / mvar }
688 {
689   prefix      .tl_set:N = \l__numodel_cmd_prefix_tl ,
690   gridx       .tl_set:N = \l__numodel_gridx_tl ,
691   gridy       .tl_set:N = \l__numodel_gridy_tl ,
692   alias       .tl_set:N = \l__numodel_alias_tl ,
693   aliasleft   .tl_set:N = \l__numodel_aliasleft_tl ,
694   aliasright  .tl_set:N = \l__numodel_aliasright_tl ,
695   flowarrow-cloud-tip .code:n =
696   { \__numodel_local_choice:Nnnn \l__numodel_mvar_flowcloud_tl
697     { flowarrow-cloud-tip } { true , false } {#1} },
698 }
699 \tl_new:N \l__numodel_mvar_flowcloud_tl
700
701 % =====
702 % \mvar[<keys>]{<name>}{<display>}{<startvalue>}{<unit>}{<sig>}{<type>}
703 % =====
704 % Declare a model variable. Generates the following macros:
705 %
706 % \<name>      -- numeric value (\edef + \fpeval), or warning if empty
707 % \<name>text  -- display name for the model table (e.g. F_{res})
708 % \<name>unit  -- SI unit (e.g. kN)
709 % \<name>unitraw - raw SI unit (e.g. \kilo\N)
710 % \<name>sign  -- number of significant figures
711 % \<name>type  -- variable type: stock, constant, aux, system
712 %                (Dutch synonyms voorraad/constante/hulp/systeem
713 %                are normalised to the canonical English form)
714 % \<name>coord -- coordinate list (empty; filled by \computemodel)
715 % \<name>gridx -- x position in the graphic model (-1 = auto)
716 % \<name>gridy -- y position in the graphic model (-1 = auto)
717 % \<name>num   -- number with significance (via \num)
718 % \<name>qty   -- number + unit (via \qty)
719 % \<name>pre   -- engineering-prefix notation (via \qty)
720 % \<name>alias -- replaces the whole initial-value cell
721 %                (empty = default)
722 % \<name>aliasleft -- replaces the left symbol in the initial value
723 %                (empty = default)
724 % \<name>aliasright -- replaces the right number in the initial value

```

```

725 %                                (empty = default)
726 %
727 % Keys accepted in [#1]:
728 %   gridx, gridy                -- position in the graphic model
729 %   alias                      -- replace the entire initial-value cell
730 %                               (in math mode)
731 %   aliasleft                  -- replace just the left symbol
732 %   aliasright                 -- replace just the right value
733 %
734 % The base value is registered in \g_defqty_names_seq so that
735 % \includeimage expands it automatically.
736 %
737 % With an empty start value (#3 blank), the base value is not
738 % numerically defined; using it issues a warning. Useful for
739 % auxiliary variables computed by \mrule.
740 %
741 % Example:
742 %   \mvar{modM}{m}{80}{\kg}{2}{constant}
743 %   \mvar{modFres}{F_{res}}{}{\N}{2}{aux}
744 %   \mvar[aliasright=\cdots]{modX}{x}{0}{\m}{3}{stock}
745 %   \mvar[alias={x \text{?}}]{modX}{x}{0}{\m}{3}{stock}
746 %
747 % Normalise the variable type (sixth \mvar argument) to its
748 % canonical English form. Accepts the canonical names
749 % {stock, constant, aux, system} and the Dutch synonyms
750 % {voorraad, constante, hulp, systeem}. An unknown value is left
751 % as-is and a warning is issued. The result is returned through
752 % \l__numodel_type_tl.
753 \tl_new:N \l__numodel_type_tl
754 \cs_new_protected:Npn \__numodel_normalize_type:n #1
755 {
756   \str_case:nnF {#1}
757   {
758     { stock      } { \tl_set:Nn \l__numodel_type_tl { stock      } }
759     { constant   } { \tl_set:Nn \l__numodel_type_tl { constant   } }
760     { aux        } { \tl_set:Nn \l__numodel_type_tl { aux        } }
761     { system     } { \tl_set:Nn \l__numodel_type_tl { system     } }
762     { voorraad   } { \tl_set:Nn \l__numodel_type_tl { stock      } }
763     { constante  } { \tl_set:Nn \l__numodel_type_tl { constant   } }
764     { hulp       } { \tl_set:Nn \l__numodel_type_tl { aux        } }
765     { systeem    } { \tl_set:Nn \l__numodel_type_tl { system     } }
766   }
767   {
768     \msg_warning:nne { numodel } { unknown-type } {#1}
769     \tl_set:Nn \l__numodel_type_tl {#1}
770   }
771 }
772
773 \makeatletter
774 \NewDocumentCommand{\mvar}{ O{} m m m m m m }{%
775   \typeout{MVAR~start:~#2}
776   % Parse optional keys (prefix, gridx, gridy, alias, aliasleft, aliasright,
777   % flowarrow-cloud-tip)
778   \tl_set:Nn \l__numodel_gridx_tl { -1 }
779   \tl_set:Nn \l__numodel_gridy_tl { -1 }
780   \tl_clear:N \l__numodel_alias_tl
781   \tl_clear:N \l__numodel_aliasleft_tl
782   \tl_clear:N \l__numodel_aliasright_tl
783   \tl_clear:N \l__numodel_cmd_prefix_tl
784   \tl_clear:N \l__numodel_mvar_flowcloud_tl
785   \typeout{MVAR~before-keys-set}
786   \keys_set:nn { numodel / mvar } {#1}
787   \typeout{MVAR~after-keys-set}
788   \__numodel_resolve_eff_prefix:
789   \typeout{MVAR~eff:~\l__numodel_eff_prefix_tl}
790   \__numodel_with_prefix:Nn \l__numodel_eff_prefix_tl

```

```

791 { \_numodel_mvar_body:nnnnnn {#2}{#3}{#4}{#5}{#6}{#7} }%
792 \typeout{MVAR-done:~#2}
793 }
794 \makeatother
795
796 % Resolve the effective prefix from \l__numodel_cmd_prefix_tl.
797 \cs_new_protected:Npn \_numodel_resolve_eff_prefix:
798 {
799   \tl_if_empty:NTF \l__numodel_cmd_prefix_tl
800     { \tl_set_eq:NN \l__numodel_eff_prefix_tl \g_numodel_current_prefix_tl }
801     { \tl_set_eq:NN \l__numodel_eff_prefix_tl \l__numodel_cmd_prefix_tl }
802 }
803
804 % The \mvar body runs in the context of the right prefix (live state
805 % has already been swapped).
806 \cs_new_protected:Npn \_numodel_mvar_body:nnnnnn #1 #2 #3 #4 #5 #6
807 {
808   % Guard: current prefix must not be empty (\newmodelprefix required)
809   \tl_if_empty:NT \g_numodel_current_prefix_tl
810     { \msg_error:nn { numodel } { no-prefix } }
811   % Full name = current prefix + short name (#1)
812   \tl_set:Nx \l__numodel_fullname_tl { \g_numodel_current_prefix_tl #1 }
813   \typeout{MVAR~body~fullname:~\l__numodel_fullname_tl}
814   % Register in the per-prefix seq. The defqty registration is
815   % optional: it only fires when the user has loaded the project-
816   % specific 'defqty' system (used for worksheet expansion).
817   % Standalone, the package works without that seq.
818   \seq_gput_right:NV \g_mvar_names_seq \l__numodel_fullname_tl
819   \cs_if_exist:NT \g_defqty_names_seq
820     { \seq_gput_right:NV \g_defqty_names_seq \l__numodel_fullname_tl }
821   \directlua{ numodel.register(
822     "\g_numodel_current_prefix_tl",
823     "\l__numodel_fullname_tl" ) }
824   \cs_if_exist:cT { \l__numodel_fullname_tl }
825     { \msg_warning:nne { numodel } { redef } { \l__numodel_fullname_tl } }
826   \tl_if_blank:nTF {#3}
827     { \cs_gset:cpe { \l__numodel_fullname_tl } { \fp_eval:n { 0 } } }
828     {
829       \cs_gset:cpe { \l__numodel_fullname_tl } { \fp_eval:n {#3} }
830       \seq_gput_right:NV \g_mvar_start_seq \l__numodel_fullname_tl
831     }
832   \cs_gset:cpn { \l__numodel_fullname_tl text } {#2}
833   \cs_gset:cpn { \l__numodel_fullname_tl unit } { \unit{#4} }
834   \cs_gset:cpn { \l__numodel_fullname_tl unitraw } {#4}
835   \cs_gset:cpn { \l__numodel_fullname_tl sign } {#5}
836   \_numodel_normalize_type:n {#6}
837   \cs_gset:cpe { \l__numodel_fullname_tl type }
838     { \tl_use:N \l__numodel_type_tl }
839   \cs_gset:cpn { \l__numodel_fullname_tl min } { inf }
840   \cs_gset:cpn { \l__numodel_fullname_tl max } { -inf }
841   \cs_gset:cpe { \l__numodel_fullname_tl gridx } { \tl_use:N \l__numodel_gridx_tl }
842   \cs_gset:cpe { \l__numodel_fullname_tl gridy } { \tl_use:N \l__numodel_gridy_tl }
843   % Preserve the original user input so that \_numodel_build_graphic:
844   % can reset auto-placed positions to -1 on a second invocation.
845   \cs_gset:cpe { \l__numodel_fullname_tl gridxinit } { \tl_use:N \l__numodel_gridx_tl }
846   \cs_gset:cpe { \l__numodel_fullname_tl gridyinit } { \tl_use:N \l__numodel_gridy_tl }
847   % Pillar A - Lua-side meta for compute_layout (additive in A1).
848   % Detokenize text so that \luaescapestring works on bare chars.
849   \tl_set:Nx \l__numodel_scratch_tl { \detokenize {#2} }
850   \directlua{ numodel.set_meta(
851     "\g_numodel_current_prefix_tl",
852     "\l__numodel_fullname_tl",
853     { type = "\tl_use:N \l__numodel_type_tl",
854       text = "\luaescapestring{\l__numodel_scratch_tl}",
855       gridx = \tl_use:N \l__numodel_gridx_tl,
856       gridy = \tl_use:N \l__numodel_gridy_tl } ) }

```

```

857 \cs_gset:cpe { \l__numodel_fullname_tl alias } { \exp_not:V \l__numodel_alias_tl }
858 \cs_gset:cpe { \l__numodel_fullname_tl aliasleft } { \exp_not:V \l__numodel_aliasleft_tl }
859 \cs_gset:cpe { \l__numodel_fullname_tl aliasright } { \exp_not:V \l__numodel_aliasright_tl }
860 \cs_gset:cpe { \l__numodel_fullname_tl flowcloud } { \exp_not:V \l__numodel_mvar_flowcloud_tl }
861 \tl_if_blank:nF {#3}
862 {
863 \exp_args:NV \NumodelDefNumCs \l__numodel_fullname_tl {\fp_eval:n {#3}} {#5}
864 \exp_args:NV \NumodelDefQtyCs \l__numodel_fullname_tl {\fp_eval:n {#3}} {#5} {#4}
865 \exp_args:NV \NumodelDefPreCs \l__numodel_fullname_tl {\fp_eval:n {#3}} {#5} {#4}
866 }
867 }
868
869 % =====
870 % Warning messages
871 % =====
872
873 \msg_new:nnn { numodel } { redef }
874 { Model~variable~'#1'~is~being~redefined. }
875 \msg_new:nnn { numodel } { empty-use }
876 { Model~variable~'#1'~has~no~start~value~and~is~used~before~assignment. }
877 \msg_new:nnn { numodel } { maxiter }
878 { computemodel~stopped~after~\int_use:N \g_numodel_maxiter_int ~iterations~
879 (safety~limit).~Check~stop~condition. }
880 \msg_new:nnn { numodel } { no-stop }
881 { computemodel:~no~stop~condition~defined.~Use~\token_to_str:N \mstop\space
882 before~\token_to_str:N \computemodel. }
883 \msg_new:nnn { numodel } { prefix-exists }
884 { Model~prefix~'#1'~is~already~registered.~
885 Use~\token_to_str:N \switchmodelprefix\space to~switch~to~an~existing~prefix. }
886 \msg_new:nnn { numodel } { prefix-unknown }
887 { Model~prefix~'#1'~is~not~registered.~
888 Use~\token_to_str:N \newmodelprefix\space to~create~it~first. }
889 \msg_new:nnn { numodel } { no-prefix }
890 { No~current~model~prefix.~
891 Call~\token_to_str:N \newmodelprefix{<name>}\space before~using~model~commands. }
892 \msg_new:nnn { numodel } { unknown-type }
893 { Unknown~variable~type~'#1'.~
894 Use~one~of~stock,~constant,~aux,~system~
895 (or~the~Dutch~aliases~voorraad,~constante,~hulp,~systeem). }
896
897 % =====
898 % Display helpers
899 % =====
900 % Translate computational expressions into syllabus-style display.
901 %
902 % Automatic translations:
903 % \modXxx -> display name (via \<name>text)
904 % * -> \cdot
905 % >= <= -> \geqslant \leqslant
906 % sign(...) -> SIGN(...) / Teken(...) in coachtaal
907 % abs(...) -> ABS(...) / Abs(...) in coachtaal
908 % && -> AND / EN in coachtaal
909 % || -> OR / OF in coachtaal
910 % cond ? a : b -> IF cond THEN ... ELSE ... ENDIF (or coachtaal eq.)
911
912 \tl_new:N \l__numodel_display_tl
913 \tl_new:N \l__numodel_lhs_tl
914 \tl_new:N \l__numodel_rhs_tl
915 \tl_new:N \l__numodel_cond_tl
916 \tl_new:N \l__numodel_true_tl
917 \tl_new:N \l__numodel_false_tl
918
919 \cs_new_protected:Npn \__numodel_vars_to_display:N #1
920 {
921 \typeout{VTD-input:~\tl_to_str:N #1}
922 \typeout{VTD-seq:~\seq_use:Nn \g_mvar_names_seq {||}}

```

```

923 % Variable names -> display names
924 \seq_map_inline:Nn \g_mvar_names_seq
925 {
926     \typeout{VTD-iter:~##1}
927     \cs_if_exist:cT { ##1 text }
928     {
929         \tl_set:Nc \l_tmpa_tl { \use:c { ##1 text } }
930         \typeout{VTD-replace~\c{##1}~with~\l_tmpa_tl}
931         \regex_replace_all:nnN { \c{##1} } { \u{l_tmpa_tl} } #1
932     }
933 }
934 % Arithmetic operators
935 \regex_replace_all:nnN { \* } { \c{cdot} \x{20} } #1
936 \regex_replace_all:nnN { >= } { \c{geqslant} \x{20} } #1
937 \regex_replace_all:nnN { <= } { \c{leqslant} \x{20} } #1
938 % Functions (syllabus notation)
939 \regex_replace_all:nnN { sign \< }
940 { \c{text}\cB{\u{g_numodel_kw_sign_tl}\cE}\< } #1
941 \regex_replace_all:nnN { abs \< }
942 { \c{text}\cB{\u{g_numodel_kw_abs_tl}\cE}\< } #1
943 % Logical operators (syllabus notation)
944 \regex_replace_all:nnN { \&\& }
945 { \c{text}\cB{\u{g_numodel_kw_and_tl}\cE}\&\& } #1
946 \regex_replace_all:nnN { \|\| }
947 { \c{text}\cB{\u{g_numodel_kw_or_tl}\cE}\|\| } #1
948 }
949
950 % Ternary detection: cond ? true_expr : false_expr
951 % Converted to: IF cond THEN lhs = true ELSE lhs = false ENDIF
952 % (or the coachtaal equivalent). Supports only flat (non-nested)
953 % ternaries.
954 \bool_new:N \l__numodel_is_ternary_bool
955
956 \cs_new_protected:Npn \__numodel_parse_ternary:nN #1 #2
957 {
958     \bool_set_false:N \l__numodel_is_ternary_bool
959     \regex_match:nnT { (.+) \? (.+) \: (.+) } {#1}
960     {
961         \bool_set_true:N \l__numodel_is_ternary_bool
962         \tl_set:Nn \l__numodel_cond_tl {#1}
963         \regex_replace_once:nnN { \s*(.+?) \s* \? .+ } { \1 } \l__numodel_cond_tl
964         \tl_set:Nn \l__numodel_true_tl {#1}
965         \regex_replace_once:nnN { .+? \? \s* (.+?) \s* \: .+ } { \1 } \l__numodel_true_tl
966         \tl_set:Nn \l__numodel_false_tl {#1}
967         \regex_replace_once:nnN { .+ \: \s* (.+?) \s* \Z } { \1 } \l__numodel_false_tl
968         \__numodel_vars_to_display:N \l__numodel_cond_tl
969         \__numodel_vars_to_display:N \l__numodel_true_tl
970         \__numodel_vars_to_display:N \l__numodel_false_tl
971     }
972 }
973
974 % =====
975 % \mrule[keys]{varname}{calculation}
976 % =====
977 % Declare a model rule.
978 %
979 % #1 (star)      -- starred variant: multiline IF/THEN/ELSE/ENDIF
980 %                for ternary expressions
981 % #2 (optional)  -- keys: alias, aliasleft, aliasright
982 %                alias      -- replaces the whole display row
983 %                aliasleft  -- replaces the left-hand side
984 %                (symbol)
985 %                aliasright -- replaces the right-hand side
986 %                (calculation)
987 %                Use \cdots for omitted parts.
988 % #3            -- name (string) of the left-hand variable

```

```

989 % #4          -- calculation (with \modXxx macros and \fpeval
990 %              syntax)
991 %
992 % With alias or aliasright the ternary logic is skipped (display
993 % shows the alias content verbatim, not IF/THEN/ELSE). Execution
994 % always uses the calculation from #4.
995 %
996 % The calculation accepts all \fpeval operators:
997 % +, -, *, /, ^, sign(), abs(), sin(), cos(), sqrt(), round()
998 % Ternary: cond ? expr_true : expr_false
999 % Logical: && (AND), || (OR), comparisons (<, >, <=, >=)
1000 %
1001 % The rule is stored for both display (\textmodel) and execution
1002 % (\computemodel).
1003 %
1004 % Examples:
1005 % \mrule{modFres}{\modM * \modA}
1006 % \mrule{modFw}{sign(\modV) * \modK * \modV^2}
1007 % \mrule{modA}{(\modT < \modTq) || (\modT > \modTdq) ? \modA + \modDa : \modA - \modDa}
1008 % \mrule{aliasright=\cdots}{\modT}{\modT + \modDt}
1009 % \mrule{aliasleft=a_x}{\modAx}{\modFgx / \modM}
1010 % \mrule{alias={\text{(hidden)}}}{\modAy}{\modFgy / \modM}
1011
1012 \NewDocumentCommand{\mrule}{s O{} m m }{%
1013   \typeout{MRULE~start:~#3}
1014   \bool_set_false:N \l__numodel_is_ternary_bool
1015   % Parse keys (prefix, alias, aliasleft, aliasright; gridx/gridy ignored)
1016   \tl_clear:N \l__numodel_alias_tl
1017   \tl_clear:N \l__numodel_aliasleft_tl
1018   \tl_clear:N \l__numodel_aliasright_tl
1019   \tl_clear:N \l__numodel_cmd_prefix_tl
1020   \keys_set:nn { numodel / mvar } {#2}
1021   \__numodel_resolve_eff_prefix:
1022   \__numodel_with_prefix:Nn \l__numodel_eff_prefix_tl
1023   { \__numodel_mrule_body:nnn {#1}{#3}{#4} }%
1024   \typeout{MRULE~done:~#3}
1025 }
1026
1027 \cs_new_protected:Npn \__numodel_mrule_body:nnn #1 #2 #3
1028 {
1029   \int_gincr:N \g_mrule_counter_int
1030   \tl_set:Nx \l__numodel_fullname_tl { \g_numodel_current_prefix_tl #2 }
1031   % Store execution data (target = full name)
1032   \seq_gput_right:Nx \g_mrule_calc_seq
1033   { { \l__numodel_fullname_tl } { \exp_not:n {#3} } }
1034   % Left-hand side: aliasleft or default display name
1035   \tl_if_blank:VTF \l__numodel_aliasleft_tl
1036   { \tl_set:Nx \l__numodel_lhs_tl { \use:c { \l__numodel_fullname_tl text } } }
1037   { \tl_set_eq:NN \l__numodel_lhs_tl \l__numodel_aliasleft_tl }
1038   % Generate display
1039   \tl_if_blank:VTF \l__numodel_alias_tl
1040   {
1041     \tl_if_blank:VTF \l__numodel_aliasright_tl
1042     {
1043       % Automatic display generation (ternary or normal)
1044       \__numodel_parse_ternary:nN {#3} \l__numodel_display_tl
1045       \bool_if:NTF \l__numodel_is_ternary_bool
1046       {
1047         \IfBooleanTF {#1}
1048         {
1049           % Starred ternary -> multiline IF/THEN/ELSE/ENDIF
1050           \tl_set:Nx \l__numodel_display_tl
1051           {
1052             \__numodel_kwt:n {if}
1053             \exp_not:V \l__numodel_cond_tl
1054             \__numodel_kwt:n {then_n}

```



```

1055     }
1056     \seq_gput_right:NV \g_mrule_seq \l__numodel_display_tl
1057     \seq_gput_right:Nn \g_mrule_type_seq { rule }
1058     \tl_set:Ne \l__numodel_display_tl
1059     {
1060         \exp_not:n { \quad }
1061         \exp_not:V \l__numodel_lhs_tl \exp_not:n { \,=\, , }
1062         \exp_not:V \l__numodel_true_tl
1063     }
1064     \seq_gput_right:NV \g_mrule_seq \l__numodel_display_tl
1065     \seq_gput_right:Nn \g_mrule_type_seq { cont }
1066     \seq_gput_right:Ne \g_mrule_seq { \__numodel_kwt:n {else_n1} }
1067     \seq_gput_right:Nn \g_mrule_type_seq { cont }
1068     \tl_set:Ne \l__numodel_display_tl
1069     {
1070         \exp_not:n { \quad }
1071         \exp_not:V \l__numodel_lhs_tl \exp_not:n { \,=\, , }
1072         \exp_not:V \l__numodel_false_tl
1073     }
1074     \seq_gput_right:NV \g_mrule_seq \l__numodel_display_tl
1075     \seq_gput_right:Nn \g_mrule_type_seq { cont }
1076     \seq_gput_right:Ne \g_mrule_seq { \__numodel_kwt:n {endif_n1} }
1077     \seq_gput_right:Nn \g_mrule_type_seq { cont }
1078 }
1079 {
1080     % Unstarred ternary -> single-line
1081     \tl_set:Ne \l__numodel_display_tl
1082     {
1083         \__numodel_kwt:n {if}
1084         \exp_not:V \l__numodel_cond_tl
1085         \__numodel_kwt:n {then}
1086         \exp_not:V \l__numodel_lhs_tl \exp_not:n { \,=\, , }
1087         \exp_not:V \l__numodel_true_tl
1088         \__numodel_kwt:n {else}
1089         \exp_not:V \l__numodel_lhs_tl \exp_not:n { \,=\, , }
1090         \exp_not:V \l__numodel_false_tl
1091         \__numodel_kwt:n {endif}
1092     }
1093 }
1094 }
1095 {
1096     % Ordinary assignment: lhs = rhs
1097     \tl_set:Nn \l__numodel_rhs_tl {#3}
1098     \__numodel_vars_to_display:N \l__numodel_rhs_tl
1099     \typeout{MRULE~rhs~after:~\tl_to_str:N \l__numodel_rhs_tl}
1100     \tl_set:Ne \l__numodel_display_tl
1101     {
1102         \exp_not:V \l__numodel_lhs_tl \exp_not:n { \,=\, , }
1103         \exp_not:V \l__numodel_rhs_tl
1104     }
1105     \typeout{MRULE~display:~\tl_to_str:N \l__numodel_display_tl}
1106 }
1107 }
1108 {
1109     % aliasright: lhs = aliasright (no ternary processing)
1110     \tl_set:Ne \l__numodel_display_tl
1111     {
1112         \exp_not:V \l__numodel_lhs_tl \exp_not:n { \,=\, , }
1113         \exp_not:V \l__numodel_aliasright_tl
1114     }
1115 }
1116 }
1117 {
1118     % alias: replace the whole row
1119     \tl_set:Ne \l__numodel_display_tl { \exp_not:V \l__numodel_alias_tl }
1120 }

```

```

1121 % For starred ternary everything is already pushed above
1122 \bool_if:nF { \l__numodel_is_ternary_bool && #1 }
1123 {
1124     \seq_gput_right:NV \g_mrulseq \l__numodel_display_tl
1125     \seq_gput_right:Nn \g_mrulseq { rule }
1126 }
1127 % Pillar A - Lua-side rule registration. Detokenize the raw
1128 % expression so Lua sees the macro names, not the evaluated
1129 % fp values. Lua handles both the dependency extraction
1130 % (build_deps) and the flow detection (classify_flows) at
1131 % \graphicmodel time.
1132 \tl_set:Nx \l__numodel_scratch_tl { \detokenize {#3} }
1133 \directlua{ numodel.add_rule(
1134     "\g_numodel_current_prefix_tl",
1135     "\l__numodel_fullname_tl",
1136     "\luaescapestring{\l__numodel_scratch_tl}",
1137     "\bool_if:NTF \l__numodel_is_ternary_bool { ternary } { calc }") }
1138 }
1139
1140 % =====
1141 % \mrulertext{free text}
1142 % =====
1143 % Adds a free-text row in the model table. Useful for structure
1144 % that does not fit as \mrulseq, e.g.:
1145 % \mrulertext{\text{IF } t < T/4 \text{ THEN}}
1146 % \mrulertext{\quad a = a + da}
1147 % \mrulertext{\text{ENDIF}}
1148 %
1149 % NOT executed by \computemodel (display only).
1150
1151 \NewDocumentCommand{\mrulertext}{0}{ m }{%
1152     \tl_clear:N \l__numodel_cmd_prefix_tl
1153     \keys_set:nn { numodel / cmd } {#1}
1154     \__numodel_resolve_eff_prefix:
1155     \__numodel_with_prefix:Nn \l__numodel_eff_prefix_tl
1156     {
1157         \int_gincr:N \g_mrulseq_counter_int
1158         \seq_gput_right:Nn \g_mrulseq {#2}
1159         \seq_gput_right:Nn \g_mrulseq { rule }
1160     }
1161 }
1162
1163 % =====
1164 % \mstop[keys]{condition}
1165 % \mstop*[keys]{condition} (star reserved; currently identical)
1166 % =====
1167 % Declare the model's stop condition.
1168 % Display: "IF condition THEN STOP ENDIF"
1169 % Execution: the model stops when the condition is true (evaluates to 1).
1170 %
1171 % The condition uses \fpeval syntax with \mvar variables:
1172 % \mstop{\modT >= \modTmax}
1173 % \mstop{\modV <= 0}
1174 %
1175 % Supported keys:
1176 % alias={...} -- replaces the whole display row (execution
1177 % remains intact)
1178 % aliasleft/aliasright are not (yet) supported for \mstop.
1179
1180 \tl_new:N \l__numodel_stop_tl
1181 \NewDocumentCommand{\mstop}{s 0}{ m }{%
1182     \typeout{MSTOP~start}
1183     % Parse via keys (only alias is honoured; prefix via key resolver)
1184     \tl_clear:N \l__numodel_alias_tl
1185     \tl_clear:N \l__numodel_aliasleft_tl
1186     \tl_clear:N \l__numodel_aliasright_tl

```

```

1187 \tl_clear:N \l__numodel_cmd_prefix_tl
1188 \keys_set:nn { numodel / mvar } {#2}
1189 \__numodel_resolve_eff_prefix:
1190 \__numodel_with_prefix:Nn \l__numodel_eff_prefix_tl
1191 { \__numodel_mstop_body:n {#3} }%
1192 \typeout{MSTOP~done}
1193 }
1194
1195 \cs_new_protected:Npn \__numodel_mstop_body:n #1
1196 {
1197   \int_gincr:N \g_mruler_counter_int
1198   % Store expression for execution (always the actual condition)
1199   \tl_gset:Nn \g_numodel_stop_expr_tl {#1}
1200   % Generate display
1201   \tl_if_blank:VTF \l__numodel_alias_tl
1202   {
1203     \tl_set:Nn \l__numodel_stop_tl {#1}
1204     \__numodel_vars_to_display:N \l__numodel_stop_tl
1205     \tl_set:Ne \l__numodel_display_tl
1206     {
1207       \__numodel_kwt:n {if}
1208       \exp_not:V \l__numodel_stop_tl
1209       \__numodel_kwt:n {then_n}
1210       \__numodel_kwt:n {stop}
1211       \__numodel_kwt:n {endif_n}
1212     }
1213   }
1214   {
1215     \tl_set:Ne \l__numodel_display_tl { \exp_not:V \l__numodel_alias_tl }
1216   }
1217   \seq_gput_right:NV \g_mruler_seq \l__numodel_display_tl
1218   \seq_gput_right:Nn \g_mruler_type_seq { rule }
1219 }
1220
1221 % =====
1222 % \textmodel
1223 % =====
1224 % Builds a tabular with numbered model rules on the left and initial
1225 % values on the right. Can be placed anywhere, e.g. in a subfigure
1226 % next to a graphic model.
1227
1228 \tl_new:N \g__numodel_table_tl
1229 \int_new:N \l__numodel_row_int
1230 \int_new:N \l__numodel_dispnum_int
1231 \int_new:N \l__numodel_startrow_int
1232 \int_new:N \l__numodel_numrules_int
1233 \int_new:N \l__numodel_numstarts_int
1234
1235 % Emit one initial-value cell with alias-key support.
1236 % - \<name>alias not empty -> replaces whole cell (inside $...$)
1237 % - \<name>aliasleft not empty -> replaces left symbol, else
1238 % \<name>text
1239 % - \<name>aliasright not empty -> replaces right value, else
1240 % \<name>qty (units=true) or
1241 % \<name>num (units=false)
1242 \cs_new_protected:Npn \__numodel_emit_startcell:n #1
1243 {
1244   \tl_if_blank:eTF { \use:c { #1 alias } }
1245   {
1246     \tl_gput_right:Nn \g__numodel_table_tl { $ }
1247     \tl_if_blank:eTF { \use:c { #1 aliasleft } }
1248     {
1249       \tl_gput_right:Ne \g__numodel_table_tl
1250       { \exp_not:e { \use:c { #1 text } } }
1251     }
1252   }

```

```

1253         \tl_gput_right:Ne \g__numodel_table_tl
1254         { \exp_not:e { \use:c { #1 aliasleft } } }
1255     }
1256     \tl_gput_right:Nn \g__numodel_table_tl { = }
1257     \tl_if_blank:eTF { \use:c { #1 aliasright } }
1258     {
1259         % Emit \<name>qty or \<name>num as a token name without
1260         % expanding it; siunitx macros must only expand at
1261         % \tl_use:N time, in the correct tabular context.
1262         \bool_if:NTF \g__numodel_units_bool
1263         {
1264             \tl_gput_right:Nx \g__numodel_table_tl
1265             { \exp_not:c { #1 qty } }
1266         }
1267         {
1268             \tl_gput_right:Nx \g__numodel_table_tl
1269             { \exp_not:c { #1 num } }
1270         }
1271     }
1272     {
1273         \tl_gput_right:Ne \g__numodel_table_tl
1274         { \exp_not:e { \use:c { #1 aliasright } } }
1275     }
1276     \tl_gput_right:Nn \g__numodel_table_tl { $ }
1277 }
1278 {
1279     \tl_gput_right:Nn \g__numodel_table_tl { $ }
1280     \tl_gput_right:Ne \g__numodel_table_tl
1281     { \exp_not:e { \use:c { #1 alias } } }
1282     \tl_gput_right:Nn \g__numodel_table_tl { $ }
1283 }
1284 }
1285
1286 \cs_new_protected:Npn \__numodel_build_table:
1287 {
1288     \typeout{BUILD_TABLE~rules:~\seq_use:Nn \g_mrule_seq {||}}
1289     \typeout{BUILD_TABLE~types:~\seq_use:Nn \g_mrule_type_seq {||}}
1290     \int_set:Nn \l__numodel_numrules_int { \seq_count:N \g_mrule_seq }
1291     \int_set:Nn \l__numodel_numstarts_int { \seq_count:N \g_mvar_start_seq }
1292     \typeout{BUILD_TABLE~numrules:~\int_use:N \l__numodel_numrules_int}
1293     \typeout{BUILD_TABLE~numstarts:~\int_use:N \l__numodel_numstarts_int}
1294     \int_zero:N \l__numodel_row_int
1295     \int_zero:N \l__numodel_dispnum_int
1296     \int_zero:N \l__numodel_startrow_int
1297     \tl_gset:Nn \g__numodel_table_tl
1298     { \begin{tabular}{r|l|l} & \textbf{ }
1299     \tl_gput_right:Ne \g__numodel_table_tl
1300     { { \__numodel_kw:n {th_model} } }
1301     \tl_gput_right:Nn \g__numodel_table_tl
1302     { & \textbf{ }
1303     \tl_gput_right:Ne \g__numodel_table_tl
1304     { { \__numodel_kw:n {th_initvals} } }
1305     \tl_gput_right:Nn \g__numodel_table_tl
1306     { \\\hline }
1307     \typeout{BUILD_TABLE~before-step-table:~\tl_to_str:N \g__numodel_table_tl}
1308     \int_step_inline:nn { \l__numodel_numrules_int }
1309     {
1310         \typeout{BUILD_TABLE~iter-row:~\int_use:N \l__numodel_row_int}
1311         \int_incr:N \l__numodel_row_int
1312         \int_incr:N \l__numodel_startrow_int
1313         \typeout{BUILD_TABLE~row:~\int_use:N \l__numodel_row_int~type:~\seq_item:Nn \g_mrule_type_seq
1314         % Check type: rule -> show number, cont -> blank
1315         \str_if_eq:eeTF
1316         { \seq_item:Nn \g_mrule_type_seq { \l__numodel_row_int } }
1317         { cont }
1318         {

```

```

1319 % Continuation rows: no number
1320 \tl_gput_right:Ne \g__numodel_table_tl
1321 {
1322     \exp_not:n { \rule{0pt}{2.6ex} }
1323     \exp_not:n { & $ }
1324     \exp_not:e { \seq_item:Nn \g_mrul_seq { \l__numodel_row_int } }
1325     \exp_not:n { $ & }
1326 }
1327 }
1328 {
1329 % Numbered row
1330 \int_incr:N \l__numodel_dispnum_int
1331 \typeout{BUILD_TABLE~emit~rule~num:~\int_use:N \l__numodel_dispnum_int~content:~\seq_item:Nn \g_mrul_seq { \l__numodel_row_int } }
1332 \tl_gput_right:Ne \g__numodel_table_tl
1333 {
1334     \exp_not:n { \rule{0pt}{2.6ex} }
1335     \int_use:N \l__numodel_dispnum_int
1336     \exp_not:n { & $ }
1337     \exp_not:e { \seq_item:Nn \g_mrul_seq { \l__numodel_row_int } }
1338     \exp_not:n { $ & }
1339 }
1340 \typeout{BUILD_TABLE~after~emit~table~tail:~\tl_tail:N \g__numodel_table_tl}
1341 }
1342 % Initial-value column (independent of rule/cont)
1343 \int_compare:nNnTF { \l__numodel_startrow_int } > { \l__numodel_numstarts_int }
1344 {
1345     \tl_gput_right:Nn \g__numodel_table_tl { \\\ }
1346 }
1347 {
1348     \exp_args:Ne \__numodel_emit_startcell:n
1349     { \seq_item:Nn \g_mvar_start_seq { \l__numodel_startrow_int } }
1350     \tl_gput_right:Nn \g__numodel_table_tl { \\\ }
1351 }
1352 }
1353 \int_while_do:nNnn { \l__numodel_startrow_int } < { \l__numodel_numstarts_int }
1354 {
1355     \int_incr:N \l__numodel_startrow_int
1356     \tl_gput_right:Nn \g__numodel_table_tl { & & }
1357     \exp_args:Ne \__numodel_emit_startcell:n
1358     { \seq_item:Nn \g_mvar_start_seq { \l__numodel_startrow_int } }
1359     \tl_gput_right:Nn \g__numodel_table_tl { \\\ }
1360 }
1361 \tl_gput_right:Nn \g__numodel_table_tl { \end{tabular} }
1362 }
1363
1364 \NewDocumentCommand{\textmodel}{\textmodel}{0{}} {%
1365     \typeout{TEXTMODEL~start}
1366     \tl_clear:N \l__numodel_cmd_prefix_tl
1367     \tl_clear:N \l__numodel_cmd_units_tl
1368     \keys_set:nn { numodel / cmd } {#1}
1369     \__numodel_resolve_eff_prefix:
1370     % Temporary units override: save the global value, apply the key
1371     % value before build, restore afterwards. This way
1372     % \textmodel[units=false] flips one render without touching the
1373     % global \numodelsetup state.
1374     \bool_set_eq:NN \l__numodel_saved_units_bool \g__numodel_units_bool
1375     \tl_if_empty:NF \l__numodel_cmd_units_tl
1376     {
1377         \str_if_eq:VnTF \l__numodel_cmd_units_tl { true }
1378         { \bool_gset_true:N \g__numodel_units_bool }
1379         { \bool_gset_false:N \g__numodel_units_bool }
1380     }
1381     \__numodel_with_prefix:Nn \l__numodel_eff_prefix_tl
1382     {
1383         \group_begin:
1384         \__numodel_apply_dsep:

```

```

1385     \__numodel_build_table:
1386     \typeout{TEXTMODEL~table:~\tl_to_str:N \g__numodel_table_tl}
1387     \tl_use:N \g__numodel_table_tl
1388   \group_end:
1389 }%
1390 \bool_gset_eq:NN \g__numodel_units_bool \l__numodel_saved_units_bool
1391 \typeout{TEXTMODEL~done}
1392 }
1393 \bool_new:N \l__numodel_saved_units_bool
1394
1395 % =====
1396 % \computemodel
1397 % =====
1398 % Run the model with the Euler method:
1399 %   1. Record all variable values in Lua
1400 %   2. Check stop condition (\mstop)
1401 %   3. Execute every \mrule rule (in declaration order)
1402 %   4. Repeat until stop or safety limit (default 20000 iterations)
1403 %
1404 % Requires: at least one \mstop and at least one \mrule.
1405 % Performance: ~440 steps in ~4s, ~20000 steps in ~60s.
1406
1407 \cs_new_protected:Npn \__numodel_exec_rule:nn #1 #2
1408 {
1409   \cs_gset:cpe {#1} { \fp_eval:n {#2} }
1410 }
1411
1412 % Record all current variable values in Lua (O(1) per variable).
1413 \cs_new_protected:Npn \__numodel_lua_record_all:
1414 {
1415   \seq_map_inline:Nn \g_mvar_names_seq
1416   {
1417     \directlua{ numodel.record(
1418       "\g_numodel_current_prefix_tl", "##1", \use:c{##1}) }
1419   }
1420   \directlua{ numodel.end_step("\g_numodel_current_prefix_tl") }
1421 }
1422
1423 % Set min/max TeX macros from Lua data after the simulation finishes.
1424 \cs_new_protected:Npn \__numodel_set_minmax_from_lua:
1425 {
1426   \seq_map_inline:Nn \g_mvar_names_seq
1427   {
1428     \cs_gset:cpe { ##1 min }
1429     { \directlua{ numodel.get_min("\g_numodel_current_prefix_tl", "##1") } }
1430     \cs_gset:cpe { ##1 max }
1431     { \directlua{ numodel.get_max("\g_numodel_current_prefix_tl", "##1") } }
1432   }
1433 }
1434
1435
1436 \NewDocumentCommand{\computemodel}{ O{} }{%
1437   \typeout{COMPUTE~start}
1438   \tl_clear:N \l__numodel_cmd_prefix_tl
1439   \keys_set:nn { numodel / cmd } {#1}
1440   \__numodel_resolve_eff_prefix:
1441   \__numodel_with_prefix:Nn \l__numodel_eff_prefix_tl
1442   { \__numodel_compute_body: }%
1443   \typeout{COMPUTE~done}
1444 }
1445
1446 \cs_new_protected:Npn \__numodel_compute_body:
1447 {
1448   \tl_if_empty:NT \g_numodel_stop_expr_tl
1449   { \msg_error:nn { numodel } { no-stop } }
1450   \int_gzero:N \g_numodel_steps_int

```

```

1451 % Initialise Lua storage for this prefix
1452 \directlua{ numodel.init("\g_numodel_current_prefix_tl") }
1453 % Main loop
1454 \bool_gset_false:N \g_tmpa_bool
1455 \bool_do_until:Nn \g_tmpa_bool
1456 {
1457   \__numodel_lua_record_all:
1458   \int_gincr:N \g_numodel_steps_int
1459   \int_compare:nNnT
1460     { \fp_eval:n { \g_numodel_stop_expr_tl } } = { 1 }
1461     { \bool_gset_true:N \g_tmpa_bool }
1462   \bool_if:NF \g_tmpa_bool
1463   {
1464     \seq_map_inline:Nn \g_mrule_calc_seq
1465       { \__numodel_exec_rule:nn ##1 }
1466   }
1467   \int_compare:nNnT
1468     { \g_numodel_steps_int } > { \g_numodel_maxiter_int }
1469   {
1470     \bool_gset_true:N \g_tmpa_bool
1471     \msg_warning:nn { numodel } { maxiter }
1472   }
1473 }
1474 \__numodel_set_minmax_from_lua:
1475 }
1476
1477 % On-demand coordinates from Lua (after \computemodel). Arguments
1478 % are SHORT names; the current prefix is prepended automatically.
1479 % See \mcoordsp for the form with an explicit prefix.
1480 %
1481 % Fully expandable (a plain \def around \directlua, no xparse
1482 % wrapper). Works directly inside \addplot coordinates
1483 % {\mcoords{T}{V}} and pgfplots' math-expression parser, without
1484 % pre-expansion via \edef.
1485 \cs_new:Npn \mcoords #1#2
1486 {
1487   \directlua{ numodel.get_coords(
1488     "\g_numodel_current_prefix_tl",
1489     "\g_numodel_current_prefix_tl" .. "#1",
1490     "\g_numodel_current_prefix_tl" .. "#2") }
1491 }
1492
1493 % Variant with explicit prefix as first argument (instead of an
1494 % optional argument, so the macro stays fully expandable).
1495 \cs_new:Npn \mcoordsp #1#2#3
1496 {
1497   \directlua{ numodel.get_coords(
1498     "#1", "#1" .. "#2", "#1" .. "#3") }
1499 }
1500
1501 % Value of variable #1 at step #2 (0-based). The variable name is
1502 % SHORT (current prefix is prepended automatically). See \mstepp for
1503 % the form with an explicit prefix.
1504 %
1505 % Example -- tangent line at step 0:
1506 % \addplot[domain=0:\modTmax]
1507 %   {\mstep{V}{0} + \mstep{A}{1} * (x - \mstep{T}{0})};
1508 \cs_new:Npn \mstep #1#2
1509 {
1510   \directlua{ numodel.get_step(
1511     "\g_numodel_current_prefix_tl",
1512     "\g_numodel_current_prefix_tl" .. "#1", #2) }
1513 }
1514
1515 \cs_new:Npn \mstepp #1#2#3
1516 {

```

```

1517 \directlua{ numodel.get_step("#1", "#1" .. "#2", #3) }
1518 }
1519
1520 % =====
1521 % \modelreset -- REMOVED
1522 % =====
1523 % \modelreset no longer exists. Use \newmodelprefix{<prefix>} to set
1524 % up a new model and \switchmodelprefix{<prefix>} to switch between
1525 % existing ones. Namespaces are additive -- variables from earlier
1526 % models remain available.
1527
1528 % =====
1529 % \graphicmodel -- Forrester diagram from \mvar/\mrule data
1530 % =====
1531 % Builds a tikzpicture with:
1532 % - stock, valve, aux and const nodes based on type and gridx/gridy
1533 % - flow arrows from valve to stock (via \flowarrow)
1534 % - causal arrows from the dependency graph
1535 %
1536 % Positioning: automatic (auto-layout) or manual via
1537 % \mvar[gridx=N, gridy=N]{...}. Mixed mode is supported.
1538 % Auto-layout places stocks+valves at gridy=0, aux at gridy=1, and
1539 % constants at gridy=2. Variables of type system are skipped.
1540 % Auxiliary variables that act as inflow are placed as a valve
1541 % (not as aux).
1542
1543 \tl_new:N \g__numodel_graphic_tl
1544 % Lua-populated cache (filled by numodel.tex_writeback at the start of
1545 % each \graphicmodel call, consumed by the emit helpers below).
1546 \prop_new:N \l__numodel_valve_for_prop % aux/const -> stock (inflow)
1547 \prop_new:N \l__numodel_outvalve_for_prop % aux/const -> stock (outflow)
1548 \prop_new:N \l__numodel_between_valve_prop % aux/const -> source_stock (between-flow)
1549 \prop_new:N \l__numodel_between_target_prop % aux/const -> target_stock (between-flow)
1550 \prop_new:N \l__numodel_stock_valve_prop % stock -> source stock (stock-as-flow)
1551 \prop_new:N \l__numodel_stock_phantom_valve_prop % stock -> source stock (phantom flow)
1552 % For diagram-style=forrester|edu: separate valve gridx position next
1553 % to the stock; the variable's own gridx/gridy keeps the natural
1554 % position at gridy=1 (aux) or gridy=2 (constant). vpos_y is always 0
1555 % so it lives implicitly in the emit helpers.
1556 \prop_new:N \l__numodel_vpos_x_prop % varname -> valve gridx
1557 \prop_new:N \l__numodel_vpos_y_prop % varname -> valve gridy (used when gridmaxx wrap shifted)
1558 \tl_new:N \l__numodel_flows_tl % deferred flow arrows
1559 \tl_new:N \l__numodel_valves_tl % deferred valve nodes (drawn on top)
1560 \tl_new:N \l__numodel_late_causals_tl % causals pointing at valves
1561
1562 % --- Resolved render settings (set per-\graphicmodel call) ---
1563 % These hold the effective values (after resolving diagram-style
1564 % defaults and explicit global/local overrides) used by the emit
1565 % helpers and by the public \flowarrow / \flowoutarrow macros.
1566 \tl_new:N \l__numodel_flowarrow_eff_tl % "hollow" or "filled"
1567 \tl_new:N \l__numodel_valve_eff_tl % "valve" | "circle" | "edu"
1568 \tl_new:N \l__numodel_flowcloud_global_tl % "true" or "false" (global default)
1569
1570 % Resolve flowarrow-style: explicit global key wins; otherwise
1571 % diagram-style picks the default (forrester -> hollow,
1572 % tight|edu -> filled).
1573 \cs_new_protected:Npn \__numodel_resolve_flowarrow:
1574 {
1575 \tl_if_empty:NTF \g__numodel_flowarrow_style_tl
1576 {
1577 \str_if_eq:VnTF \g__numodel_diagram_style_tl { forrester }
1578 { \tl_set:Nn \l__numodel_flowarrow_eff_tl { hollow } }
1579 { \tl_set:Nn \l__numodel_flowarrow_eff_tl { filled } }
1580 }
1581 { \tl_set_eq:NN \l__numodel_flowarrow_eff_tl
1582 \g__numodel_flowarrow_style_tl }

```



```

1583 }
1584
1585 % Resolve valve-style: explicit global key wins; otherwise
1586 % diagram-style picks the default (forrester -> valve,
1587 % tight|edu -> edu).
1588 \cs_new_protected:Npn \__numodel_resolve_valve:
1589 {
1590   \tl_if_empty:NTF \g__numodel_valve_style_tl
1591   {
1592     \str_if_eq:VnTF \g__numodel_diagram_style_tl { forrester }
1593     { \tl_set:Nn \l__numodel_valve_eff_tl { valve } }
1594     { \tl_set:Nn \l__numodel_valve_eff_tl { edu } }
1595   }
1596   { \tl_set_eq:NN \l__numodel_valve_eff_tl
1597     \g__numodel_valve_style_tl }
1598 }
1599
1600 % Resolve the global flowarrow-cloud-tip default (no per-stock
1601 % override yet; that's added on top in \__numodel_eff_flowcloud:n).
1602 \cs_new_protected:Npn \__numodel_resolve_flowcloud:
1603 {
1604   \tl_if_empty:NTF \g__numodel_flowcloud_tl
1605   {
1606     \str_if_eq:VnTF \g__numodel_diagram_style_tl { forrester }
1607     { \tl_set:Nn \l__numodel_flowcloud_global_tl { true } }
1608     { \tl_set:Nn \l__numodel_flowcloud_global_tl { false } }
1609   }
1610   { \tl_set_eq:NN \l__numodel_flowcloud_global_tl
1611     \g__numodel_flowcloud_tl }
1612 }
1613
1614 % Per-stock flowcloud lookup. #1 = stock varname. Sets
1615 % \l__numodel_flowcloud_eff_tl to "true" or "false". Order: per-mvar
1616 % override on the stock (via [flowarrow-cloud-tip=...]) wins over the
1617 % global default.
1618 \tl_new:N \l__numodel_flowcloud_eff_tl
1619 \cs_new_protected:Npn \__numodel_eff_flowcloud:n #1
1620 {
1621   \tl_set:Ne \l__numodel_flowcloud_eff_tl { \use:c { #1 flowcloud } }
1622   \tl_if_empty:NT \l__numodel_flowcloud_eff_tl
1623   { \tl_set_eq:NN \l__numodel_flowcloud_eff_tl
1624     \l__numodel_flowcloud_global_tl }
1625 }
1626
1627 % Sets the public \nmflowbody and \nmflowtip macros that the
1628 % \flowarrow / \flowoutarrow / \flowbetweenarrow macros expand to.
1629 % Reads the resolved flowarrow-style.
1630 \cs_new_protected:Npn \__numodel_apply_flowarrow_style:
1631 {
1632   \str_if_eq:VnTF \l__numodel_flowarrow_eff_tl { hollow }
1633   {
1634     \cs_gset:Npn \nmflowbody { flowpipe-hollow }
1635     \cs_gset:Npn \nmflowtip { flowpipe-hollow-tip }
1636     \cs_gset:Npn \nmflowtipcloudin { flowpipe-hollow-cloudin }
1637     \cs_gset:Npn \nmflowtipcloudout { flowpipe-hollow-cloudout }
1638   }
1639   {
1640     \cs_gset:Npn \nmflowbody { flowpipe-filled }
1641     \cs_gset:Npn \nmflowtip { flowpipe-filled-tip }
1642     \cs_gset:Npn \nmflowtipcloudin { flowpipe-filled-cloudin }
1643     \cs_gset:Npn \nmflowtipcloudout { flowpipe-filled-cloudout }
1644   }
1645 }
1646
1647 % Sets the tikz node-style name for the valve and the boolean that
1648 % controls whether the valve carries the variable's display label.

```

```

1649 % valve-style=valve -> [valve-forrester], no label
1650 % valve-style=circle -> [valve-circle], no label
1651 % valve-style=edu -> [valve-edu], with label
1652 \tl_new:N \l__numodel_valve_node_tl
1653 \bool_new:N \l__numodel_valve_label_bool
1654 \cs_new_protected:Npn \__numodel_apply_valve_style:
1655 {
1656   \str_case:VnF \l__numodel_valve_eff_tl
1657   {
1658     { valve }
1659     {
1660       \tl_set:Nn \l__numodel_valve_node_tl { valve-forrester }
1661       \bool_set_false:N \l__numodel_valve_label_bool
1662     }
1663     { circle }
1664     {
1665       \tl_set:Nn \l__numodel_valve_node_tl { valve-circle }
1666       \bool_set_false:N \l__numodel_valve_label_bool
1667     }
1668     { edu }
1669     {
1670       \tl_set:Nn \l__numodel_valve_node_tl { valve-edu }
1671       \bool_set_true:N \l__numodel_valve_label_bool
1672     }
1673   }
1674   {
1675     \tl_set:Nn \l__numodel_valve_node_tl { valve-edu }
1676     \bool_set_true:N \l__numodel_valve_label_bool
1677   }
1678 }
1679
1680 % --- Diagram-style mode flag ---
1681 % True at diagram-style=forrester|edu; consulted by \__numodel_place_node
1682 % to decide between single-emit and natural+phantom-valve double-emit.
1683 % Set in \__numodel_build_graphic from \g__numodel_diagram_style_tl.
1684 \bool_new:N \l__numodel_keep_natural_bool
1685
1686
1687 \cs_new_protected:Npn \__numodel_build_graphic:
1688 {
1689   \typeout{BUILD:~step-0~reset~auto~positions}
1690   % --- Reset gridx/gridy to the original user input ---
1691   % Auto-layout writes positions to var.gridx/gridy. On a second
1692   % \graphicmodel call those would, without reset, be treated as
1693   % "manually placed". The initial values are saved in
1694   % gridxinit/gridyinit by \mvar -- copy them back.
1695   \seq_map_inline:Nn \g_mvar_names_seq
1696   {
1697     \cs_gset:cpe { ##1 gridx } { \use:c { ##1 gridxinit } }
1698     \cs_gset:cpe { ##1 gridy } { \use:c { ##1 gridyinit } }
1699   }
1700   % Diagram-style mode: forrester|edu keep aux/const at their natural
1701   % gridy with a phantom valve next to the stock; tight collapses the
1702   % aux/const onto the valve position. Consumed by \__numodel_place_node
1703   % to dispatch between single-emit and natural+phantom emit.
1704   \bool_set_false:N \l__numodel_keep_natural_bool
1705   \str_if_eq:VnT \g__numodel_diagram_style_tl { forrester }
1706   { \bool_set_true:N \l__numodel_keep_natural_bool }
1707   \str_if_eq:VnT \g__numodel_diagram_style_tl { edu }
1708   { \bool_set_true:N \l__numodel_keep_natural_bool }
1709   \typeout{BUILD:~step-1~lua~layout~writeback}
1710   % Pillar A - Lua is the single source of truth for flow detection
1711   % and auto-layout. The writeback clears and fills \<var>gridx/gridy
1712   % and the flow-props that downstream emitters (place_node,
1713   % flow-builders, emit_natural_and_phantom, emit_stock_valve) read.
1714   \__numodel_lua_layout_writeback:

```

```

1715 \typeout{BUILD:~step-2~tikzpicture}
1716 % --- Build tikzpicture ---
1717 % Render order matters: each segment overlays the previous one,
1718 % so we draw flows first (one continuous arrow per flow), then
1719 % the valve nodes on top (with white fill so they cover the
1720 % section of the arrow that lies underneath), and finally the
1721 % causal arrows that point at those valves.
1722 \tl_gclear:N \g__numodel_graphic_tl
1723 \tl_clear:N \l__numodel_flows_tl
1724 \tl_clear:N \l__numodel_valves_tl
1725 \tl_clear:N \l__numodel_late_causals_tl
1726 \tl_gput_right:Nn \g__numodel_graphic_tl
1727 { \begin{tikzpicture}[gridscale] }
1728 % Nodes (stocks, aux, const, clouds; flows -> flows_tl,
1729 % valves -> valves_tl)
1730 \seq_map_inline:Nn \g_mvar_names_seq
1731 { \typeout{NODE:~##1} \__numodel_place_node:n {##1} }
1732 \typeout{BUILD:~step-5~stock-valve~nodes}
1733 % Stock-as-flow valve nodes
1734 \prop_map_inline:Nn \l__numodel_stock_valve_prop
1735 {
1736   % ##1 = stock (e.g. modH), ##2 = source stock (e.g. modV)
1737   % Skip entries that are not real stock-valves (e.g. __svgx keys)
1738   \tl_if_in:nnF {##1} { __sv }
1739   { \__numodel_emit_stock_valve:nn {##1} {##2} }
1740 }
1741 % Stock-as-rate phantom-valve nodes (source stock has no matching
1742 % outflow term, so the inflow gets a cloud at the open end)
1743 \prop_map_inline:Nn \l__numodel_stock_phantom_valve_prop
1744 {
1745   \tl_if_in:nnF {##1} { __sv }
1746   { \__numodel_emit_stock_phantom_valve:nn {##1} {##2} }
1747 }
1748 \typeout{BUILD:~step-6~flow~arrows}
1749 % Flow arrows (one segment each; drawn underneath the valves)
1750 \tl_gput_right:NV \g__numodel_graphic_tl \l__numodel_flows_tl
1751 \typeout{BUILD:~step-6a~valve~nodes}
1752 % Valve nodes (white-filled, drawn ON TOP of the flow arrows)
1753 \tl_gput_right:NV \g__numodel_graphic_tl \l__numodel_valves_tl
1754 \typeout{BUILD:~step-6b+7~causal~arrows~(Lua)}
1755 % A2: causal arrows via Lua - emit_causals demultiplexes on
1756 % tgt_is_valve and pushes to \g__numodel_graphic_tl or
1757 % \l__numodel_late_causals_tl respectively. Afterwards append
1758 % late-causals to graphic_tl once: that covers both the pushes
1759 % from the flow-builders (step 6/6a) and those from emit_causals.
1760 \__numodel_lua_emit_causals:
1761 \tl_gput_right:NV \g__numodel_graphic_tl \l__numodel_late_causals_tl
1762 \typeout{BUILD:~step-8~done}
1763 \tl_gput_right:Nn \g__numodel_graphic_tl
1764 { \end{tikzpicture} }
1765 }
1766
1767 % --- Node placement ---
1768 \cs_new_protected:Npn \__numodel_place_node:n #1
1769 {
1770   \bool_set_true:N \l_tmpa_bool
1771   \tl_set:Nc \l__numodel_scratch_tl { \use:c { #1 type } }
1772   \exp_args:NV \str_if_eq:nnT \l__numodel_scratch_tl { system }
1773   { \bool_set_false:N \l_tmpa_bool }
1774   \bool_if:NT \l_tmpa_bool
1775   {
1776     \int_compare:nNnT { \use:c { #1 gridx } } = { -1 }
1777     { \bool_set_false:N \l_tmpa_bool }
1778   }
1779   \bool_if:NT \l_tmpa_bool
1780   {

```

```

1781 \tl_set:Nc \l__numodel_tmp_tl { \use:c { #1 text } }
1782 % With forrester|edu, valve vars get a dual emission:
1783 % the natural aux/const node + a phantom valve next to the stock.
1784 \bool_set_false:N \l_tmpb_bool % is this a valve var?
1785 \prop_if_in:NnT \l__numodel_valve_for_prop {#1}
1786 { \bool_set_true:N \l_tmpb_bool }
1787 \prop_if_in:NnT \l__numodel_outvalve_for_prop {#1}
1788 { \bool_set_true:N \l_tmpb_bool }
1789 \prop_if_in:NnT \l__numodel_between_valve_prop {#1}
1790 { \bool_set_true:N \l_tmpb_bool }
1791 \bool_lazy_and:nnTF
1792 { \l__numodel_keep_natural_bool } { \l_tmpb_bool }
1793 { \__numodel_emit_natural_and_phantom:n {#1} }
1794 {
1795 \prop_if_in:NnTF \l__numodel_valve_for_prop {#1}
1796 { \__numodel_emit_valve:n {#1} }
1797 {
1798 \prop_if_in:NnTF \l__numodel_outvalve_for_prop {#1}
1799 { \__numodel_emit_outvalve:n {#1} }
1800 {
1801 \prop_if_in:NnTF \l__numodel_between_valve_prop {#1}
1802 { \__numodel_emit_between_valve:n {#1} }
1803 {
1804 \tl_set:Nc \l__numodel_scratch_tl { \use:c { #1 type } }
1805 \exp_args:NV \str_if_eq:nnT \l__numodel_scratch_tl { stock }
1806 { \__numodel_emit_stock:n {#1} }
1807 \exp_args:NV \str_if_eq:nnT \l__numodel_scratch_tl { aux }
1808 { \__numodel_emit_aux:n {#1} }
1809 \exp_args:NV \str_if_eq:nnT \l__numodel_scratch_tl { constant }
1810 { \__numodel_emit_const:n {#1} }
1811 }
1812 }
1813 }
1814 }
1815 }
1816 }
1817
1818 % --- Helper: emit a coordinate node at the valve position ---
1819 % The flow arrow starts/ends at this coordinate (or at an offset
1820 % relative to it, for the open-end without cloud). The visible
1821 % white-filled valve is added later (valves_tl) at the same
1822 % coordinates so it overlays the arrow.
1823 % #1 = coord id #2 = x #3 = y
1824 \cs_new_protected:Npn \__numodel_emit_valve_coord:nnn #1 #2 #3
1825 {
1826 \tl_gput_right:Nc \g__numodel_graphic_tl
1827 {
1828 \exp_not:N \coordinate ~
1829 (#1) ~ \exp_not:n{at} ~ (#2 , ~ #3) ;
1830 }
1831 }
1832
1833 % --- Helper: append a valve node to the deferred valves_tl ---
1834 % Renders \node[<valve-style>] (id) at (x,y) {<label>}; later in the
1835 % picture so the white fill overlays the flow arrow underneath.
1836 % #1 = node id #2 = x coord #3 = y coord #4 = label tl name
1837 \cs_new_protected:Npn \__numodel_defer_valve:nnnn #1 #2 #3 #4
1838 {
1839 \tl_put_right:Nc \l__numodel_valves_tl
1840 {
1841 \exp_not:N \node ~ [ \tl_use:N \l__numodel_valve_node_tl ] ~
1842 (#1) ~ \exp_not:n{at} ~ (#2 , ~ #3) ~
1843 {
1844 \bool_if:NNTF \l__numodel_valve_label_bool
1845 { \exp_not:N $ \exp_not:V #4 \exp_not:N $ }
1846 { }

```

```

1847     } ;
1848   }
1849 }
1850
1851 % --- Forrester/edu-mode emission: natural node + phantom valve ---
1852 % The variable sits as an ordinary aux/const node at its own gridy.
1853 % The phantom valve (id #1_v) sits next to the stock at gridy=0; its
1854 % appearance follows the resolved valve-style (label included only
1855 % when valve-style=edu). A causal arrow from #1 -> #1_v is drawn
1856 % (deferred to late_causals_tl so it lands on top of the white-
1857 % filled valve).
1858 \tl_new:N \l__numodel_phantom_label_tl
1859 \cs_new_protected:Npn \__numodel_emit_natural_and_phantom:n #1
1860 {
1861   % --- 1. Natural node (aux or constant) ---
1862   \tl_set:Nx \l__numodel_scratch_tl { \use:c { #1 type } }
1863   \exp_args:NV \str_if_eq:nnT \l__numodel_scratch_tl { aux }
1864   { \__numodel_emit_aux:n {#1} }
1865   \exp_args:NV \str_if_eq:nnT \l__numodel_scratch_tl { constant }
1866   { \__numodel_emit_const:n {#1} }
1867   % --- 2. Locate the phantom valve x/y position ---
1868   \prop_get:NnNT \l__numodel_vpos_x_prop {#1} \l__numodel_phantom_x_tl
1869   {
1870     \prop_get:NnNF \l__numodel_vpos_y_prop {#1} \l__numodel_scratch_y_tl
1871     { \tl_set:Nn \l__numodel_scratch_y_tl { 0 } }
1872     \tl_set:Nx \l__numodel_phantom_label_tl { \use:c { #1 text } }
1873     % --- 3. Coordinate at the phantom valve position ---
1874     \__numodel_emit_valve_coord:nnn { #1 __vc }
1875     { \tl_use:N \l__numodel_phantom_x_tl }
1876     { \tl_use:N \l__numodel_scratch_y_tl }
1877     % --- 4. Flow arrow + (optional) cloud, one segment ---
1878     \prop_if_in:NnTF \l__numodel_valve_for_prop {#1}
1879     {
1880       \tl_set:Nx \l__numodel_tmp_tl
1881       { \prop_item:Nn \l__numodel_valve_for_prop {#1} }
1882       \__numodel_eff_flowcloud:V \l__numodel_tmp_tl
1883       \str_if_eq:VnTF \l__numodel_flowcloud_eff_tl { true }
1884       {
1885         \tl_put_right:Nx \l__numodel_flows_tl
1886         {
1887           \exp_not:N \flowarrow [#1 __cl] {#1 __vc}
1888           { \tl_use:N \l__numodel_tmp_tl }
1889         }
1890       }
1891     }
1892     \tl_put_right:Nx \l__numodel_flows_tl
1893     {
1894       \exp_not:N \flowarrow {#1 __vc}
1895       { \tl_use:N \l__numodel_tmp_tl }
1896     }
1897   }
1898 }
1899 {
1900   \prop_if_in:NnTF \l__numodel_outvalve_for_prop {#1}
1901   {
1902     \tl_set:Nx \l__numodel_tmp_tl
1903     { \prop_item:Nn \l__numodel_outvalve_for_prop {#1} }
1904     \__numodel_eff_flowcloud:V \l__numodel_tmp_tl
1905     \str_if_eq:VnTF \l__numodel_flowcloud_eff_tl { true }
1906     {
1907       \tl_put_right:Nx \l__numodel_flows_tl
1908       {
1909         \exp_not:N \flowoutarrow [#1 __cl]
1910         { \tl_use:N \l__numodel_tmp_tl } {#1 __vc}
1911       }
1912     }
1913   }
1914 }

```

```

1913         {
1914             \tl_put_right:Ne \l__numodel_flows_tl
1915             {
1916                 \exp_not:N \flowoutarrow
1917                 { \tl_use:N \l__numodel_tmp_tl } {#1 __vc}
1918             }
1919         }
1920     }
1921     {
1922         % between (no cloud: both ends are stocks)
1923         \tl_set:Ne \l__numodel_tmp_tl
1924         { \prop_item:Nn \l__numodel_between_valve_prop {#1} }
1925         \tl_set:Ne \l__numodel_scratch_tl
1926         { \prop_item:Nn \l__numodel_between_target_prop {#1} }
1927         \tl_put_right:Ne \l__numodel_flows_tl
1928         {
1929             \exp_not:N \flowbetweenarrow
1930             { \tl_use:N \l__numodel_tmp_tl }
1931             {#1 __vc}
1932             { \tl_use:N \l__numodel_scratch_tl }
1933         }
1934     }
1935 }
1936 % --- 5. Defer the phantom valve drawing (overlays flow) ---
1937 \__numodel_defer_valve:nnnn { #1 __v }
1938 { \tl_use:N \l__numodel_phantom_x_tl }
1939 { \tl_use:N \l__numodel_scratch_y_tl }
1940 \l__numodel_phantom_label_tl
1941 % --- 6. Causal arrow natural -> phantom valve (deferred) ---
1942 \tl_put_right:Ne \l__numodel_late_causals_tl
1943 {
1944     \exp_not:N \draw ~ [\exp_not:n{causal}] ~
1945     (#1) ~ \exp_not:n{to} ~ (#1 __v) ;
1946 }
1947 }
1948 }
1949 \tl_new:N \l__numodel_phantom_x_tl
1950 \cs_generate_variant:Nn \__numodel_eff_flowcloud:n { V }
1951
1952 \cs_new_protected:Npn \__numodel_emit_stock:n #1
1953 {
1954     \tl_gput_right:Ne \g__numodel_graphic_tl
1955     {
1956         \exp_not:N \node ~ [\exp_not:n{stock}] ~
1957         (#1) ~ \exp_not:n{at} ~
1958         ( \use:c{#1 gridx} , ~ \use:c{#1 gridy} ) ~
1959         { \exp_not:N $ \exp_not:V \l__numodel_tmp_tl \exp_not:N $ } ;
1960     }
1961 }
1962
1963 % Tight-mode helpers: the valve sits at the variable's own gridx/
1964 % gridy. We emit a coordinate node (#1__vc) early so the flow
1965 % arrow has a positioning anchor, and defer the visible valve node
1966 % (#1) to valves_tl so it overlays the arrow with white fill.
1967 \cs_new_protected:Npn \__numodel_emit_tight_valve_setup:n #1
1968 {
1969     \__numodel_emit_valve_coord:nnn { #1 __vc }
1970     { \use:c{#1 gridx} } { \use:c{#1 gridy} }
1971     \__numodel_defer_valve:nnnn {#1}
1972     { \use:c{#1 gridx} } { \use:c{#1 gridy} }
1973     \l__numodel_tmp_tl
1974 }
1975
1976 \cs_new_protected:Npn \__numodel_emit_valve:n #1
1977 {
1978     \__numodel_emit_tight_valve_setup:n {#1}

```

```

1979 \tl_set:Ne \l__numodel_scratch_tl
1980 { \prop_item:Nn \l__numodel_valve_for_prop {#1} }
1981 \__numodel_eff_flowcloud:V \l__numodel_scratch_tl
1982 \str_if_eq:VnTF \l__numodel_flowcloud_eff_tl { true }
1983 {
1984     \tl_put_right:Ne \l__numodel_flows_tl
1985     {
1986         \exp_not:N \flowarrow [#1 __cl] {#1 __vc}
1987         { \tl_use:N \l__numodel_scratch_tl }
1988     }
1989 }
1990 {
1991     \tl_put_right:Ne \l__numodel_flows_tl
1992     {
1993         \exp_not:N \flowarrow {#1 __vc}
1994         { \tl_use:N \l__numodel_scratch_tl }
1995     }
1996 }
1997 }
1998
1999 \cs_new_protected:Npn \__numodel_emit_outvalve:n #1
2000 {
2001     \__numodel_emit_tight_valve_setup:n {#1}
2002     \tl_set:Ne \l__numodel_scratch_tl
2003     { \prop_item:Nn \l__numodel_outvalve_for_prop {#1} }
2004     \__numodel_eff_flowcloud:V \l__numodel_scratch_tl
2005     \str_if_eq:VnTF \l__numodel_flowcloud_eff_tl { true }
2006     {
2007         \tl_put_right:Ne \l__numodel_flows_tl
2008         {
2009             \exp_not:N \flowoutarrow [#1 __cl]
2010             { \tl_use:N \l__numodel_scratch_tl } {#1 __vc}
2011         }
2012     }
2013     {
2014         \tl_put_right:Ne \l__numodel_flows_tl
2015         {
2016             \exp_not:N \flowoutarrow
2017             { \tl_use:N \l__numodel_scratch_tl } {#1 __vc}
2018         }
2019     }
2020 }
2021
2022 \cs_new_protected:Npn \__numodel_emit_between_valve:n #1
2023 {
2024     \__numodel_emit_tight_valve_setup:n {#1}
2025     \tl_set:Ne \l__numodel_scratch_tl
2026     { \prop_item:Nn \l__numodel_between_valve_prop {#1} }
2027     \tl_set:Ne \l__numodel_tmp_tl
2028     { \prop_item:Nn \l__numodel_between_target_prop {#1} }
2029     \tl_put_right:Ne \l__numodel_flows_tl
2030     {
2031         \exp_not:N \flowbetweenarrow
2032         { \tl_use:N \l__numodel_scratch_tl }
2033         {#1 __vc}
2034         { \tl_use:N \l__numodel_tmp_tl }
2035     }
2036 }
2037
2038 \cs_new_protected:Npn \__numodel_emit_aux:n #1
2039 {
2040     \tl_gput_right:Ne \g__numodel_graphic_tl
2041     {
2042         \exp_not:N \node ~ [\exp_not:n{aux}] ~
2043         (#1) ~ \exp_not:n{at} ~
2044         ( \use:c{#1 gridx} , ~ \use:c{#1 gridy} ) ~

```

```

2045      { \exp_not:N $ \exp_not:V \l__numodel_tmp_tl \exp_not:N $ } ;
2046    }
2047  }
2048
2049 \cs_new_protected:Npn \__numodel_emit_const:n #1
2050 {
2051   \tl_gput_right:Ne \g__numodel_graphic_tl
2052   {
2053     \exp_not:N \constnode
2054     {#1}
2055     { \use:c{#1 gridx} , ~ \use:c{#1 gridy} }
2056     { \exp_not:N $ \exp_not:V \l__numodel_tmp_tl \exp_not:N $ }
2057   }
2058 }
2059
2060 % --- Stock-as-flow valve emission ---
2061 \cs_new_protected:Npn \__numodel_emit_stock_valve:nn #1 #2
2062 {
2063   % #1 = target stock (e.g. modH), #2 = source stock (e.g. modV)
2064   % Pull gridx from the saved property
2065   \prop_get:NnNTF \l__numodel_stock_valve_prop { #1 __svgx }
2066   \l__numodel_scratch_tl
2067   {
2068     % Look up the wrap-shifted y-slot; default to 0 (the row that
2069     % the auto layout used before any gridmaxx wrap was applied).
2070     \prop_get:NnNF \l__numodel_stock_valve_prop { #1 __svgy }
2071     \l__numodel_scratch_y_tl
2072     { \tl_set:Nn \l__numodel_scratch_y_tl { 0 } }
2073     % Coordinate at the valve position (used by the flow arrow)
2074     \__numodel_emit_valve_coord:nnn { #1 __svc }
2075     { \tl_use:N \l__numodel_scratch_tl }
2076     { \tl_use:N \l__numodel_scratch_y_tl }
2077     % Flow arrow from source-stock through valve into target-stock
2078     % (no cloud: both ends are stocks). The arrow runs in one
2079     % continuous segment; the visible valve will be drawn ON TOP
2080     % later via valves_tl.
2081     \tl_put_right:Ne \l__numodel_flows_tl
2082     {
2083       \exp_not:N \flowbetweenarrow {#2} {#1 __svc} {#1}
2084     }
2085     % Defer the visible valve node (white fill overlays arrow)
2086     \tl_set:Ne \l__numodel_tmp_tl { \use:c { #2 text } }
2087     \__numodel_defer_valve:nnnn { #1 __sv }
2088     { \tl_use:N \l__numodel_scratch_tl }
2089     { \tl_use:N \l__numodel_scratch_y_tl }
2090     \l__numodel_tmp_tl
2091     % Causal arrow source-stock -> valve. Bend it (curved) only
2092     % when the source stock and the valve sit on the same row -
2093     % then the straight causal would coincide with the flow pipe
2094     % and disappear behind it. When they are on different rows
2095     % (e.g. after a gridmaxx wrap) the causal is vertical-ish and
2096     % stays visible without a bend. Deferred so it lands on top
2097     % of the white-filled valve.
2098     \str_if_eq:eeTF { \tl_use:N \l__numodel_scratch_y_tl }
2099       { \use:c { #2 gridy } }
2100     {
2101       \tl_put_right:Ne \l__numodel_late_causals_tl
2102       {
2103         \exp_not:N \draw ~ [\exp_not:n{causal}] ~
2104         (#2) ~ \exp_not:n{to} ~ [\exp_not:n{bend~left=30}] ~ (#1__sv) ;
2105       }
2106     }
2107     {
2108       \tl_put_right:Ne \l__numodel_late_causals_tl
2109       {
2110         \exp_not:N \draw ~ [\exp_not:n{causal}] ~

```



```

2111         (#2) ~ \exp_not:n{to} ~ (#1__sv) ;
2112     }
2113 }
2114 }
2115 { } % no gridx found: skip
2116 }
2117
2118 % --- Stock phantom-valve emission (source stock as rate factor) ---
2119 % Source stock acts as a rate factor for the target stock without a
2120 % matching outflow term; render with a cloud-fed inflow valve next to
2121 % the target stock and link the source stock via a causal arrow.
2122 \cs_new_protected:Npn \__numodel_emit_stock_phantom_valve:nn #1 #2
2123 {
2124     % #1 = target stock (e.g. ballV), #2 = source stock (e.g. ballV)
2125     \prop_get:NnNTF \l__numodel_stock_phantom_valve_prop { #1 __svgx }
2126     \l__numodel_scratch_tl
2127     {
2128         % Look up the wrap-shifted y-slot; default to 0.
2129         \prop_get:NnNF \l__numodel_stock_phantom_valve_prop { #1 __svgy }
2130         \l__numodel_scratch_y_tl
2131         { \tl_set:Nn \l__numodel_scratch_y_tl { 0 } }
2132         % Coordinate at the valve position
2133         \__numodel_emit_valve_coord:nnn { #1 __svc }
2134         { \tl_use:N \l__numodel_scratch_tl }
2135         { \tl_use:N \l__numodel_scratch_y_tl }
2136         % Flow arrow: cloud (open end) -> valve -> target stock
2137         \__numodel_eff_flowcloud:n {#1}
2138         \str_if_eq:VnTF \l__numodel_flowcloud_eff_tl { true }
2139         {
2140             \tl_put_right:Ne \l__numodel_flows_tl
2141             {
2142                 \exp_not:N \flowarrow [#1 __svcl] {#1 __svc} {#1}
2143             }
2144         }
2145         {
2146             \tl_put_right:Ne \l__numodel_flows_tl
2147             {
2148                 \exp_not:N \flowarrow {#1 __svc} {#1}
2149             }
2150         }
2151         % Defer the visible valve node, labelled with the source stock
2152         \tl_set:Ne \l__numodel_tmp_tl { \use:c { #2 text } }
2153         \__numodel_defer_valve:nnnn { #1 __sv }
2154         { \tl_use:N \l__numodel_scratch_tl }
2155         { \tl_use:N \l__numodel_scratch_y_tl }
2156         \l__numodel_tmp_tl
2157         % Causal arrow source-stock -> valve. Bend (curved) only when
2158         % source and valve sit on the same row - see the equivalent
2159         % comment in \__numodel_emit_stock_valve:nn for the rationale.
2160         \str_if_eq:eeTF { \tl_use:N \l__numodel_scratch_y_tl }
2161         { \use:c { #2 gridy } }
2162         {
2163             \tl_put_right:Ne \l__numodel_late_causals_tl
2164             {
2165                 \exp_not:N \draw ~ [\exp_not:n{causal}] ~
2166                 (#2) ~ \exp_not:n{to} ~ [\exp_not:n{bend~left=30}] ~ (#1__sv) ;
2167             }
2168         }
2169         {
2170             \tl_put_right:Ne \l__numodel_late_causals_tl
2171             {
2172                 \exp_not:N \draw ~ [\exp_not:n{causal}] ~
2173                 (#2) ~ \exp_not:n{to} ~ (#1__sv) ;
2174             }
2175         }
2176     }

```

```

2177     { } % no gridx found: skip
2178 }
2179
2180 % -----
2181 % Pillar A - Lua-side layout writeback. A single directlua call
2182 % that runs compute_layout (build_deps, classify_flows,
2183 % populate_occupied, auto_layout, causals) and writes the resulting
2184 % state back to the TeX-side props (\<var>gridx/gridy and the
2185 % flow-props) that the downstream emitters (place_node,
2186 % flow-builders, emit_natural_and_phantom, emit_stock_valve) read.
2187 % Lua is the single source of truth for the decision logic; TeX
2188 % only renders.
2189 % -----
2190 \cs_new_protected:Npn \__numodel_lua_layout_writeback:
2191 {
2192     % 1. Clear the props that Lua refills, so repeated \graphicmodel
2193     % calls don't produce duplicate entries.
2194     \prop_clear:N \l__numodel_valve_for_prop
2195     \prop_clear:N \l__numodel_outvalve_for_prop
2196     \prop_clear:N \l__numodel_between_valve_prop
2197     \prop_clear:N \l__numodel_between_target_prop
2198     \prop_clear:N \l__numodel_stock_valve_prop
2199     \prop_clear:N \l__numodel_stock_phantom_valve_prop
2200     \prop_clear:N \l__numodel_vpos_x_prop
2201     \prop_clear:N \l__numodel_vpos_y_prop
2202     % 2. Run the Lua pipeline.
2203     \directlua{ numodel.compute_layout(
2204         "g_numodel_current_prefix_tl",
2205         "g__numodel_diagram_style_tl",
2206         \int_use:N \g_numodel_gridmaxx_int) }
2207     % 3. Write gridx/gridy and all flow-props back.
2208     % tex_writeback emits a list of expl3 statements via tex.print.
2209     \directlua{ numodel.tex_writeback(
2210         "g_numodel_current_prefix_tl") }
2211 }
2212
2213 \cs_new_protected:Npn \__numodel_lua_emit_causals:
2214 {
2215     \directlua{ numodel.emit_causals(
2216         "g_numodel_current_prefix_tl") }
2217 }
2218
2219 \NewDocumentCommand{\graphicmodel}{0} {} {%
2220     \tl_clear:N \l__numodel_cmd_prefix_tl
2221     \tl_clear:N \l__numodel_cmd_diagstyle_tl
2222     \tl_clear:N \l__numodel_cmd_flowarrow_tl
2223     \tl_clear:N \l__numodel_cmd_valve_tl
2224     \tl_clear:N \l__numodel_cmd_flowcloud_tl
2225     \keys_set:nn { numodel / cmd } {#1}
2226     \__numodel_resolve_eff_prefix:
2227     % Temporary overrides: save the global state, apply the per-call
2228     % key values before build, restore afterwards. This way
2229     % \graphicmodel[diagram-style=forrester] flips one render without
2230     % touching the global \numodelsetup state.
2231     \tl_set_eq:NN \l__numodel_saved_diagstyle_tl \g__numodel_diagram_style_tl
2232     \tl_set_eq:NN \l__numodel_saved_flowarrow_tl \g__numodel_flowarrow_style_tl
2233     \tl_set_eq:NN \l__numodel_saved_valve_tl \g__numodel_valve_style_tl
2234     \tl_set_eq:NN \l__numodel_saved_flowcloud_tl \g__numodel_flowcloud_tl
2235     \tl_if_empty:NF \l__numodel_cmd_diagstyle_tl
2236     { \tl_gset_eq:NN \g__numodel_diagram_style_tl \l__numodel_cmd_diagstyle_tl }
2237     \tl_if_empty:NF \l__numodel_cmd_flowarrow_tl
2238     { \tl_gset_eq:NN \g__numodel_flowarrow_style_tl \l__numodel_cmd_flowarrow_tl }
2239     \tl_if_empty:NF \l__numodel_cmd_valve_tl
2240     { \tl_gset_eq:NN \g__numodel_valve_style_tl \l__numodel_cmd_valve_tl }
2241     \tl_if_empty:NF \l__numodel_cmd_flowcloud_tl
2242     { \tl_gset_eq:NN \g__numodel_flowcloud_tl \l__numodel_cmd_flowcloud_tl }

```

```

2243 % Resolve the effective values for this render and propagate them
2244 % to \nmflowbody / \nmflowtip used by the flow macros.
2245 \__numodel_resolve_flowarrow:
2246 \__numodel_resolve_valve:
2247 \__numodel_resolve_flowcloud:
2248 \__numodel_apply_flowarrow_style:
2249 \__numodel_apply_valve_style:
2250 \__numodel_with_prefix:Nn \l__numodel_eff_prefix_tl
2251 {
2252   \__numodel_build_graphic:
2253   \tl_use:N \g__numodel_graphic_tl
2254 }%
2255 \tl_gset_eq:NN \g__numodel_diagram_style_tl \l__numodel_saved_diagstyle_tl
2256 \tl_gset_eq:NN \g__numodel_flowarrow_style_tl \l__numodel_saved_flowarrow_tl
2257 \tl_gset_eq:NN \g__numodel_valve_style_tl \l__numodel_saved_valve_tl
2258 \tl_gset_eq:NN \g__numodel_flowcloud_tl \l__numodel_saved_flowcloud_tl
2259 }
2260 \tl_new:N \l__numodel_saved_diagstyle_tl
2261 \tl_new:N \l__numodel_saved_flowarrow_tl
2262 \tl_new:N \l__numodel_saved_valve_tl
2263 \tl_new:N \l__numodel_saved_flowcloud_tl
2264
2265 % =====
2266 % Plot machinery
2267 % =====
2268 % \diagrammodel calls the pgfplots/calcplothdims infrastructure
2269 % through internal wrappers \__numodel_calc_plot_dims: and
2270 % \__numodel_draw_plot:n. Those wrappers forward to \calcplothdims
2271 % and \drawplot from the sibling package numodel-plot.
2272 %
2273 % The wrappers are not \cs_new_eq:NN because numodel-plot can load
2274 % after this package; at load time \calcplothdims and \drawplot may
2275 % not yet exist. With a wrapper they are looked up at expansion
2276 % time.
2277
2278 \cs_new:Npn \__numodel_calc_plot_dims: { \calcplothdims }
2279 \cs_new:Npn \__numodel_draw_plot:n #1 { \drawplot {#1} }
2280
2281 % =====
2282 % \diagrammodel -- convenience command for model graphs
2283 % =====
2284 % Usage: \diagrammodel{xvar}{yvar}[extra drawplot code]{label}
2285 %
2286 % Builds a complete figure with model points. Sets axis dimensions
2287 % from min/max, axis labels from text/unit, and emits caption and
2288 % label automatically.
2289
2290 \tl_new:N \l__numodel_xname_tl
2291 \tl_new:N \l__numodel_yname_tl
2292
2293 \NewDocumentCommand{\diagrammodel}{O{} m m O{} m }
2294 {
2295   \tl_clear:N \l__numodel_cmd_prefix_tl
2296   \keys_set:nn { numodel / cmd } {#1}
2297   \__numodel_resolve_eff_prefix:
2298   % Full names (prefix + short) for \use:c lookup on globals
2299   \tl_set:N \l__numodel_xname_tl { \l__numodel_eff_prefix_tl #2 }
2300   \tl_set:N \l__numodel_yname_tl { \l__numodel_eff_prefix_tl #3 }
2301   \edef\dm@coords{\mcoordsp{\l__numodel_eff_prefix_tl}{#2}{#3}}
2302   \def\xmin{\use:c{\l__numodel_xname_tl min}}
2303   \def\xmax{\use:c{\l__numodel_xname_tl max}}
2304   \def\xlabelqty{\use:c{\l__numodel_xname_tl text}}
2305   \def\xlabelunit{\use:c{\l__numodel_xname_tl unitraw}}
2306   \def\ymin{\use:c{\l__numodel_yname_tl min}}
2307   \def\ymax{\use:c{\l__numodel_yname_tl max}}
2308   \def\ylabelqty{\use:c{\l__numodel_yname_tl text}}

```

```

2309 \def\ylabelunit{\use:c{\l__numodel_ynname_tl unitraw}}
2310 \begin{figure}[H]
2311 \centering
2312 \group_begin:
2313 \__numodel_apply_dsep:
2314 \__numodel_draw_plot:n
2315 {
2316 \addplot[only~marks,~mark=*,~mark~size=0.5pt]~coordinates~{\dm@coords};
2317 \addlegendentry{model~point}
2318 #4
2319 }
2320 \group_end:
2321 \caption{$\use:c{\l__numodel_ynname_tl text}$($\use:c{\l__numodel_xname_tl text}$)\text{-
diagram}}
2322 % If the project-specific worksheet system is loaded, pick the
2323 % '-ws' label inside a worksheet; otherwise always the bare label.
2324 \cs_if_exist:NTF \ifinworksheet
2325 { \label{fig:#5-ws}}{\label{fig:#5}} }
2326 { \label{fig:#5}} }
2327 \end{figure}
2328 }
2329
2330 \ExplSyntaxOff
2331
2332 % =====
2333 % GRAPHIC MODEL - STYLES AND MACROS
2334 % =====
2335 %
2336 % Forrester-diagram conventions:
2337 %
2338 % Stock [stock] Rectangle v, s, T, x
2339 % Valve / inflow [valve] Circle on a thick a, dv, ds
2340 % flow pipe
2341 % Auxiliary [aux] Plain circle F_res, F_air
2342 % Constant \constnode Circle + dashes m, k, g
2343 %
2344 % Use \dgridx and \dgridy as the grid spacings for consistent positioning.
2345 % =====
2346
2347 % === Grid spacing -- defaults set via \numodelsetup {graphscalex, graphscaley, stockwidth} ===
2348
2349 % === Styles ===
2350 \tikzset{
2351 % Scale coordinates to the grid spacing
2352 gridscale/.style={x=\dgridx cm, y=\dgridy cm},
2353 % Stock (state variable): rectangle
2354 stock/.style={
2355 rectangle, draw, thick,
2356 minimum width=3 em,
2357 minimum height=2 em,
2358 font=\small
2359 },
2360 % Auxiliary (intermediate variable): plain circle
2361 aux/.style={
2362 circle, draw, thick,
2363 inner sep=2pt,
2364 minimum size=2 em,
2365 font=\scriptsize
2366 },
2367 % Constant: circle (dashes drawn by the \constnode macro)
2368 const/.style={
2369 circle, draw, thick,
2370 inner sep=2pt,
2371 minimum size=2 em,
2372 font=\scriptsize
2373 },

```

```

2374 % Valve (legacy alias): circle on the flow pipe. Kept for
2375 % backwards compatibility with user code that calls \flowarrow
2376 % manually. The package's emit helpers now pick one of the
2377 % style variants below based on \numodelsetup{valve-style=...}.
2378 valve/.style={
2379   circle, draw, thick, fill=white,
2380   minimum size=2 em,
2381   font=\small
2382 },
2383 % Valve variant: Forrester valve symbol -- two filled triangles
2384 % (white inside) whose tips meet at the centre. The symmetry
2385 % axis stands perpendicular to the (horizontal) flow. The
2386 % bounding box is tall and narrow so the apex angle at the
2387 % centre is roughly 60 degrees (sharper than the 90 degrees of
2388 % a square box), and the horizontal "ribs" along the top and
2389 % bottom edges close the two triangles.
2390 valve-forrester/.style={
2391   shape=rectangle,
2392   draw=none, fill=none,
2393   inner sep=0pt, outer sep=0pt,
2394   %minimum width=\dgridx*0.18 cm,
2395   minimum width=1 em,
2396   %minimum height=\dgridy*0.30 cm,
2397   minimum height=2 em,
2398   path picture={%
2399     \draw[thick, fill=white]
2400       (path picture bounding box.north west) --
2401       (path picture bounding box.north east) --
2402       (path picture bounding box.center) --
2403       cycle;
2404     \draw[thick, fill=white]
2405       (path picture bounding box.south west) --
2406       (path picture bounding box.south east) --
2407       (path picture bounding box.center) --
2408       cycle;
2409   }
2410 },
2411 % Valve variant: empty circle (no label).
2412 valve-circle/.style={
2413   circle, draw, thick, fill=white,
2414   minimum size=2 em,
2415   font=\small
2416 },
2417 % Valve variant: circle with the variable's display label inside.
2418 valve-edu/.style={
2419   circle, draw, thick, fill=white,
2420   minimum size=2 em,
2421   font=\small
2422 },
2423 % Flow pipe variants (body without arrow tip, used for the
2424 % segment from open-end / cloud to the valve):
2425 flowpipe/.style={line width=3pt}, % legacy alias
2426 flowpipe-filled/.style={line width=3pt},
2427 flowpipe-hollow/.style={
2428   line width=1pt,
2429   double=white, double distance=3pt
2430 },
2431 % Flow pipe variants WITH arrow tip (used for the segment from
2432 % valve to stock, or the open-end side of an outflow). Both
2433 % variants use the Stealth arrow tip. The hollow form
2434 % uses the double-line magic length 0pt-3-0 so that the arrow
2435 % tip merges cleanly with the gap between the two strokes.
2436 flowpipe-filled-tip/.style={line width=3pt, arrows={-Stealth[inset=0pt, angle=30:1em]}},
2437 flowpipe-hollow-tip/.style={
2438   line width=1pt,
2439   double=white, double distance=3pt,

```

```

2440 arrows={-Stealth[inset=0pt, angle=30:1em]}
2441 },
2442 % Flow pipe variants with the Cloud arrow tip on the open end:
2443 % inflow keeps the Stealth arrow head at the stock side.
2444 flowpipe-filled-cloudin/.style={
2445   line width=3pt,
2446   arrows={Cloud[fill=white, line width=0.8pt]-Stealth[inset=0pt, angle=30:1em]}
2447 },
2448 flowpipe-hollow-cloudin/.style={
2449   line width=1pt,
2450   double=white, double distance=3pt,
2451   arrows={Cloud[fill=white, line width=0.8pt]-Stealth[inset=0pt, angle=30:1em]}
2452 },
2453 % Outflow with cloud at the open end: first a Stealth tip at the
2454 % line end, then the cloud behind it (in arrows.meta spec, list
2455 % multiple tips separated by a dot so the shaft stops at the
2456 % Stealth tip and does not extend into the cloud; order = from
2457 % line outwards, hence Stealth.Cloud).
2458 flowpipe-filled-cloudout/.style={
2459   line width=3pt,
2460   arrows={-[-Stealth[inset=0pt, angle=30:1em].Cloud[fill=white, line width=0.8pt]]}
2461 },
2462 flowpipe-hollow-cloudout/.style={
2463   line width=1pt,
2464   double=white, double distance=3pt,
2465   arrows={-[-Stealth[inset=0pt, angle=30:1em].Cloud[fill=white, line width=0.8pt]]}
2466 },
2467 % Causal arrow (thin arrow for dependencies)
2468 causal/.style={thick, arrows={-Stealth[length=0.5em]}}
2469 }
2470
2471 % Default values for the body / tip macros so a manual call to
2472 % \flowarrow (outside \graphicmodel) keeps working. The
2473 % \graphicmodel pipeline overwrites these per render.
2474 \providecommand{\nmflowbody}{flowpipe-filled}
2475 \providecommand{\nmflowtip}{flowpipe-filled-tip}
2476 \providecommand{\nmflowtipcloudin}{flowpipe-filled-cloudin}
2477 \providecommand{\nmflowtipcloudout}{flowpipe-filled-cloudout}
2478
2479 % === Macro: constant node with dashes ===
2480 % Usage: \constnode{name}{(x,y)}{<label>}
2481 \newcommand{\constnode}[3]{%
2482   \node[const] (#1) at (#2) {#3};%
2483   \draw[thick] ([xshift=-3mm]#1.west) -- (#1.west);%
2484   \draw[thick] (#1.east) -- ([xshift=3mm]#1.east);%
2485 }
2486
2487 % === Macro: flow pipe with valve (inflow) ===
2488 % Draws ONE continuous flow arrow ending at the stock. The starting
2489 % position is the same regardless of cloud presence; passing any
2490 % non-blank optional argument switches to the cloud-tipped variant
2491 % so the arrow's tail becomes a white-filled cloud (drawn as a real
2492 % PGF arrow tip, not a separate node - moves with the line). The
2493 % valve overlays the middle of the arrow as a separate white-filled
2494 % node drawn later by the build pipeline.
2495 % Style keys: \nmflowtip / \nmflowtipcloudin.
2496 % Usage: \flowarrow[<any>]{<valve-coord>}{<stock-node>}
2497 \NewDocumentCommand{\flowarrow}{ O{} m m }{%
2498   \IfBlankTF{#1}{%
2499     \draw[\nmflowtip] ([xshift=-3em]#2) -- ([xshift=0.5mm]#3.west);%
2500   }{%
2501     \draw[\nmflowtipcloudin] ([xshift=-3em]#2) -- ([xshift=0.5mm]#3.west);%
2502   }%
2503 }
2504
2505 % === Macro: flow pipe with valve (outflow) ===

```

```

2506 % Mirror of \flowarrow. The arrow always ends at the same offset
2507 % past the valve coordinate; passing a non-blank optional argument
2508 % switches to the cloud-tipped variant (cloud at the open end).
2509 % Usage: \flowoutarrow[<any>]{<stock-node>}{<valve-coord>}
2510 \NewDocumentCommand{\flowoutarrow}{0}{m m}{%
2511   \IfBlankTF{#1}{%
2512     \draw[\nmflowtip] (#2.east) -- ([xshift=4em]#3);%
2513   }{%
2514     \draw[\nmflowtipcloudout] (#2.east) -- ([xshift=5em]#3);%
2515   }%
2516 }
2517
2518 % === Macro: flow pipe between two stocks ===
2519 % One continuous arrow from source stock through valve coord into
2520 % target stock. No cloud option (no open end exists when both
2521 % sides are stocks).
2522 % Usage: \flowbetweenarrow{<source-stock>}{<valve-coord>}{<target-stock>}
2523 \newcommand{\flowbetweenarrow}[3]{%
2524   \draw[\nmflowtip] (#1.east) -- ([xshift=0.5mm]#3.west);%
2525 }
2526 \endpackage

```

9.1 Translation files

Each `numodel-<LANG>.def` file populates the lookup table consulted by `__numodel_kw:n`. The package loads exactly the file matching the current syntax key, via `\InputIfFileExists` (and hence `kpse`), so additional languages can be supplied through `TEXMFHOME/tex/latex/numodel/numodel-<LANG>.def` with no package rebuild. The two shipped files follow.

9.1.1 numodel-EN.def – English (XMILE)

XMILE v1.0 OASIS style: ALL-CAPS, case sensitive. Sets the default decimal mark to point.

```

2527 (*EN)
2528 \ProvidesExplFile{numodel-EN.def}{2026/05/17}{0.3.0}
2529 {numodel-keyword-table----English~(XMILE)}
2530 \cs_new:Npn \__numodel_kw_EN_if: { IF~ }
2531 \cs_new:Npn \__numodel_kw_EN_then: { ~THEN~ }
2532 \cs_new:Npn \__numodel_kw_EN_then_nl: { ~THEN }
2533 \cs_new:Npn \__numodel_kw_EN_else: { ~ELSE~ }
2534 \cs_new:Npn \__numodel_kw_EN_else_nl: { ELSE }
2535 \cs_new:Npn \__numodel_kw_EN_endif: { ~ENDIF }
2536 \cs_new:Npn \__numodel_kw_EN_endif_nl: { ENDIF }
2537 \cs_new:Npn \__numodel_kw_EN_and: { ~AND~ }
2538 \cs_new:Npn \__numodel_kw_EN_or: { ~OR~ }
2539 \cs_new:Npn \__numodel_kw_EN_not: { NOT~ }
2540 \cs_new:Npn \__numodel_kw_EN_sign: { SIGN }
2541 \cs_new:Npn \__numodel_kw_EN_abs: { ABS }
2542 \cs_new:Npn \__numodel_kw_EN_stop: { ~STOP~ }
2543 \cs_new:Npn \__numodel_kw_EN_th_model: { model }
2544 \cs_new:Npn \__numodel_kw_EN_th_initvals: { initial-values }
2545 \cs_new:Npn \__numodel_kw_EN_dsep_default: { point }
2546 \end{EN}

```

9.1.2 numodel-NL.def – Dutch (CoachTaal)

CoachTaal convention. Sets the default decimal mark to comma.

```

2547 (*NL)
2548 \ProvidesExplFile{numodel-NL.def}{2026/05/17}{0.3.0}
2549 {numodel-keyword-table----Dutch~(CoachTaal)}
2550 \cs_new:Npn \__numodel_kw_NL_if: { Als~ }
2551 \cs_new:Npn \__numodel_kw_NL_then: { ~Dan~ }
2552 \cs_new:Npn \__numodel_kw_NL_then_nl: { ~Dan }
2553 \cs_new:Npn \__numodel_kw_NL_else: { ~Anders~ }
2554 \cs_new:Npn \__numodel_kw_NL_else_nl: { Anders }
2555 \cs_new:Npn \__numodel_kw_NL_endif: { ~EindAls }
2556 \cs_new:Npn \__numodel_kw_NL_endif_nl: { EindAls }

```

```

2557 \cs_new:Npn \__numodel_kw_NL_and:      { ~EN~ }
2558 \cs_new:Npn \__numodel_kw_NL_or:        { ~OF~ }
2559 \cs_new:Npn \__numodel_kw_NL_not:        { NIET~ }
2560 \cs_new:Npn \__numodel_kw_NL_sign:       { Teken }
2561 \cs_new:Npn \__numodel_kw_NL_abs:        { Abs }
2562 \cs_new:Npn \__numodel_kw_NL_stop:       { ~Stop~ }
2563 \cs_new:Npn \__numodel_kw_NL_th_model:   { model }
2564 \cs_new:Npn \__numodel_kw_NL_th_initvals: { startwaarden }
2565 \cs_new:Npn \__numodel_kw_NL_dsep_default: { comma }
2566 </NL>

```