

The numodel package*

Paul Zuurbier
mail@paulzuurbier.nl

May 19, 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	5
4	Public API	5
4.1	Variables and rules	5
4.2	Expression reference	6
4.3	Render commands	8
4.4	Series accessors	9
4.5	Namespace management	9
5	Variable types	10
6	Generated accessors	10
7	Multiple models in one document	10
8	Requirements	11
9	Implementation	11
9.1	Translation files	51
9.1.1	<code>numodel-EN.def</code> — English (XMILE)	51
9.1.2	<code>numodel-NL.def</code> — Dutch (CoachTaal)	52

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` v0.4.0, dated May 19, 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	$v = v + g \cdot dt$	$t = 0 \text{ s}$
2	$y = y + v \cdot dt$	$dt = 0.10 \text{ s}$
3	$t = t + dt$	$v = 0 \text{ m s}^{-1}$
4	IF $y \leq 0$ THEN STOP ENDIF	$y = 100 \text{ m}$ $g = -9.81 \text{ m s}^{-2}$

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}
```

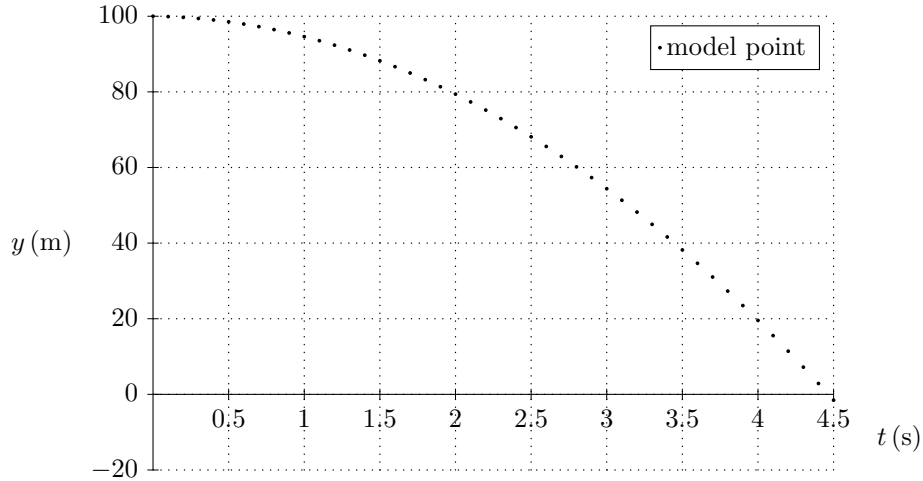


Figure 1: $y(t)$ -diagram

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.

tblrenv Which `tabularray` environment wraps the `\textmodel` table. Three values: **longtblr** (default, page-breakable); **tblr** (inline, no page breaks); **talltblr** (inline, no page breaks, supports `\caption/notes`). Pick **tblr** or **talltblr** when `\textmodel` sits inside an enclosing environment that suppresses page breaks (`subfigure`, `minipage`, ...); the default **longtblr** would otherwise emit spurious “(Continued)” / *Continued on next page* markers in that setting. May also be supplied per render as `\textmodel[tblrenv=...]`; the global setting is restored afterwards.

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`).

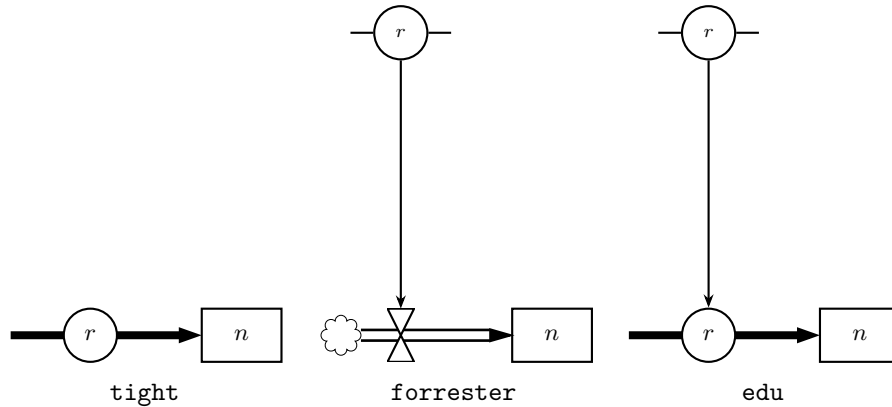
point use a full stop (`siunitx output-decimal-marker={.}`, `pgfplots/pgf/number format/use period`).

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}{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 `\<prefix><Name>`), $\langle text \rangle$ 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. `\mper\s\squared`), $\langle sig \rangle$ is the number of significant figures used by `\<prefix><Name>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. $\langle expr \rangle$ may use the full `\fp_eval` expression grammar. See Table 2 (Section 4.2) for a complete overview of supported operators and functions with their XMILE and CoachTaal equivalents.

\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 $\langle expr \rangle$ evaluates true. Exactly one `\mstop` per model prefix is required before `\computemodel`. Without one, `\computemodel` issues a warning.

4.2 Expression reference

Table 2 lists all expression constructs for `\mrule` bodies. The **XMILE** column shows XMILE v1.0 syntax, **l3fp** shows the `\fp_eval`-compatible form used inside `\mrule`, and **CoachTaal** shows the Coach 7 equivalent (function names from the standard CoachTaal math reference; note that argument separators in CoachTaal are semicolons). **Orange** entries are not yet fully supported by `numodel` (the expression computes correctly, but the rendered keyword in the rule table is not yet translated). *Italic* cells contain a derived equivalent rather than a native keyword.

Table 2: Expression reference: XMILE, `\fp_eval`, and CoachTaal

XMILE	l3fp (<code>\fp_eval</code>)	CoachTaal
<i>Arithmetic operators</i>		
+	+	+
-	-	-
*	*	*
/	/	/
^	^	^
MOD(x,y)	$x - \text{trunc}(x/y)*y$	$x - \text{Entier}(x/y)*y$
<i>Comparison operators</i>		
<	<	<
<=	<=	<=
>	>	>
>=	>=	>=
=	=	=
<>	!=	<>
<i>Boolean operators</i>		
AND	&&	EN

Continued on next page

Table 2: Expression reference: XMILE, \fp_eval, and CoachTaal (Continued)

XMILE	l3fp (\fp_eval)	CoachTaal
OR		OF
NOT	!	NIET
<i>Control flow</i>		
IF c THEN a ELSE b	c ? a : b	Als c Dan a Anders b EindAls
<i>General math functions</i>		
ABS(x)	abs(x)	Abs(x)
SIGN(x)	sign(x)	Teken(x)
SQRT(x)	sqrt(x)	Sqrt(x)
INT(x)	trunc(x)	$Entier(Abs(x)) * Teken(x)$
$INT(x+0.5)$	round(x)	Round(x)
$INT(x)$	floor(x)	Entier(x)
$-INT(-x)$	ceil(x)	$-Entier(-x)$
MIN(x,y)	min(x,y)	Min(x;y)
MAX(x,y)	max(x,y)	Max(x;y)
—	fact(x)	Fac(x)
$INT(LOG10(ABS(x)))$	logb(x)	$Entier(Log(Abs(x)))$
<i>Exponential and logarithmic</i>		
EXP(x)	exp(x)	Exp(x)
LN(x)	ln(x)	Ln(x)
LOG10(x)	$ln(x)/ln(10)$	Log(x)
<i>Trigonometry (radians)</i>		
SIN(x)	sin(x)	Sin(x)
COS(x)	cos(x)	Cos(x)
TAN(x)	tan(x)	Tan(x)
ARCSIN(x)	asin(x)	Arcsin(x)
ARCCOS(x)	acos(x)	Arccos(x)
ARCTAN(x)	atan(x)	Arctan(x)
ARCTAN2(y,x)	atan(y,x)	$Arctan(y/x)$
$1/TAN(x)$	cot(x)	$1/Tan(x)$
$1/SIN(x)$	csc(x)	$1/Sin(x)$
$1/COS(x)$	sec(x)	$1/Cos(x)$
$ARCSIN(1/x)$	acsc(x)	$Arcsin(1/x)$
$ARCCOS(1/x)$	asec(x)	$Arccos(1/x)$
$ARCTAN(1/x)$	acot(x)	$Arctan(1/x)$
$ARCTAN2(x,y)$	acot(y,x)	$Arctan(x/y)$
<i>Trigonometry (degrees)</i>		
$SIN(PI/180*x)$	sind(x)	$Sin(Pi/180*x)$
$COS(PI/180*x)$	cosd(x)	$Cos(Pi/180*x)$
$TAN(PI/180*x)$	tand(x)	$Tan(Pi/180*x)$
$180/PI*ARCSIN(x)$	asind(x)	$180/Pi*Arcsin(x)$
$180/PI*ARCCOS(x)$	acosd(x)	$180/Pi*Arccos(x)$
$180/PI*ARCTAN(x)$	atand(x)	$180/Pi*Arctan(x)$
$180/PI*ARCTAN2(y,x)$	atand(y,x)	$180/Pi*Arctan(y/x)$
$1/TAN(PI/180*x)$	cotd(x)	$1/Tan(Pi/180*x)$
$1/SIN(PI/180*x)$	cscd(x)	$1/Sin(Pi/180*x)$
$1/COS(PI/180*x)$	secd(x)	$1/Cos(Pi/180*x)$
$180/PI*ARCSIN(1/x)$	acscd(x)	$180/Pi*Arcsin(1/x)$
$180/PI*ARCCOS(1/x)$	asecd(x)	$180/Pi*Arccos(1/x)$

Continued on next page

Table 2: Expression reference: XMILE, \fp_eval, and CoachTaal (Continued)

XMILE	l3fp (\fp_eval)	CoachTaal
$180/PI*ARCTAN(1/x)$	<code>acotd(x)</code>	$180/Pi*Arctan(1/x)$
$180/PI*ARCTAN2(x,y)$	<code>acotd(y,x)</code>	$180/Pi*Arctan(x/y)$
<i>Constants</i>		
PI	<code>pi</code>	Pi
e	<code>exp(1)</code>	<i>Exp(1)</i>
INF	<code>inf</code>	—
NAN	<code>nan</code>	—
$PI/180$	<code>deg</code>	$Pi/180$
1	<code>true</code>	Aan
0	<code>false</code>	Uit
<i>Simulation-specific functions (XMILE only)</i>		
TIME	<i>user variable (\mvar)</i>	<i>user variable</i>
DT	<i>user variable (\mvar)</i>	<i>user variable</i>
STEP(h,t ₀)	$T \geq t_0 \text{ ? } h : 0$	<i>Als T >= t0 Dan h Anders 0 EindAls</i>
RAMP(s,t ₀)	$T > t_0 \text{ ? } s*(T-t_0) : 0$	<i>Als T > t0 Dan s*(T-t0) Anders 0 EindAls</i>
DELAY(x,dt)	<i>not supported</i>	<i>not supported</i>
SMOOTH(x,t)	<i>not supported</i>	<i>not supported</i>
INIT(x)	<i>not supported</i>	<i>not supported</i>
PREVIOUS(x)	<i>not supported</i>	<i>not supported</i>
RANDOM(0,1)	<code>rand()</code>	Rand
RANDOM(lo,hi)	$lo + (hi-lo)*rand()$	$lo + (hi-lo)*Rand$
<i>not supported</i>	<code>randint(n)</code>	<i>not supported</i>
<i>not supported</i>	<code>randint(m,n)</code>	<i>not supported</i>

4.3 Render commands

<code>\textmodel</code>	<p>Renders the rule-and-startvalue table. Optional [<code><keys></code>] accepts <code>prefix=<name></code> (render a non-current model), <code>units=true</code> or <code>units=false</code>, and <code>tblrenv=tblrlongtblrtalltblr</code>. Each per-call key overrides the global <code>\numodelsetup</code> setting for this single render only; the global state is restored afterwards.</p> <p><code>tblrenv</code> selects which <code>tabularray</code> environment wraps the table: <code>longtblr</code> (default) breaks across pages, <code>tblr</code> renders inline without page breaks, <code>talltblr</code> renders inline but with caption and note support. Use <code>tblr</code> or <code>talltblr</code> when <code>\textmodel</code> sits inside an enclosing environment that suppresses page breaks (<code>subfigure</code>, <code>minipage</code>, ...); the default <code>longtblr</code> would otherwise emit spurious continuation markers there.</p> <p>Row spacing. <code>numodel</code> loads <code>tabularray</code> and sets <code>\SetTblrInner{rowsep=0pt}</code> globally so the rule listing renders compactly. This applies to <i>every</i> <code>tabularray</code> table in the document; if you want the default spacing back in your own tables, issue <code>\SetTblrInner{rowsep=2pt}</code> (or whatever value you prefer) somewhere in your document.</p>
<code>\graphicmodel</code>	<p>Renders the Forrester stock-and-flow diagram. Variables of type <code>stock</code> become rectangles, <code>constant</code> become circles, <code>aux</code> 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.</p> <p>Optional [<code><keys></code>] accepts <code>prefix=<name></code> (render a non-current model) and <code>diagram-style=tightforrestered</code> which overrides the global <code>\numodelsetup</code> setting for this single render only. The global state is restored afterwards, so multiple <code>\graphicmodel</code> calls can each pick their own style without re-issuing <code>\numodelsetup</code>.</p>
<code>\computemodel</code>	<p>Runs the Euler simulation in Lua. Records every variable's value at every step, plus running min and max. After it returns, the accessors <code>\<prefix><Name>min</code> / <code>\<prefix><Name>max</code> hold the extrema, and <code>\<prefix><Name></code> holds the final-step value. The time-series can be retrieved with <code>\mcoords</code> / <code>\mstep</code>.</p>
<code>\diagrammodel</code>	<p>Convenience wrapper that produces a complete figure with caption and label. Signature:</p> $\backslash\text{diagrammodel}[\text{<keys>}]\{\text{<xvar>}\}\{\text{<yvar>}\}[\text{<extra>}]\{\text{<label>}\}$ <p>Reads min/max and display text from <code>\<prefix><xvar></code> etc., delegates to <code>\drawplot</code> from <code>numodel-</code></p>

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`.

4.4 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 endpoints}
}{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:

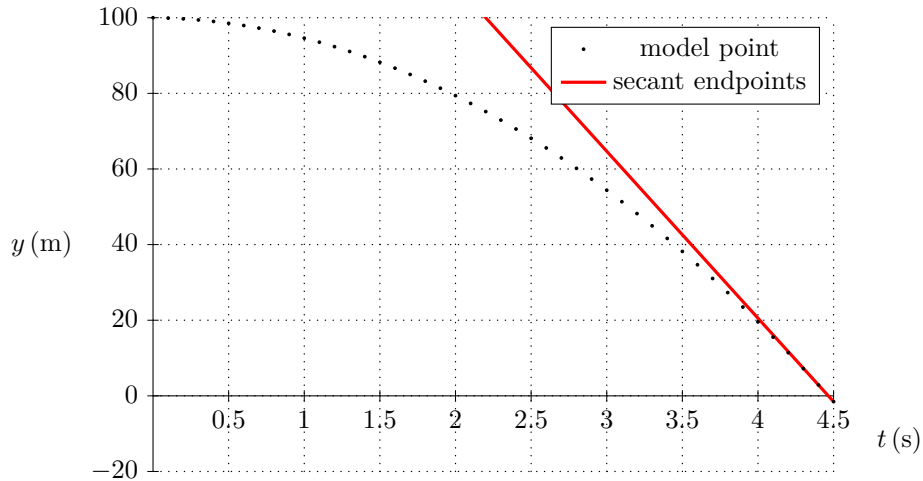


Figure 2: $y(t)$ -diagram

4.5 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 $\text{stock} \leftarrow \text{stock} + \dots$ 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}[2026/05/19 v0.4.0]
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{tabularray}    % \textmodel uses longtblr (page-breakable,
64                               % works inside floats/subfigure)
65 % Define a numodel theme for longtblr in \textmodel. The default
66 % 'normal' middlehead/lasthead prefixes the continuation marker with
67 % 'Table N:' from \tablename/\thetable; we don't issue \caption, so we
68 % want only the conthead text (which is itself localised via
69 % \tblrcontheadname, updated by \__numodel_refresh_kw:). Likewise for
70 % the foot. Activated via 'theme = {numodel}' in the longtblr options.
71 \DefTblrTemplate {firsthead}      {numodel} {}
72 \DefTblrTemplate {middlehead, lasthead} {numodel} { \UseTblrTemplate {conthead} {default} }
73 \DefTblrTemplate {firstfoot, middlefoot} {numodel} { \UseTblrTemplate {contfoot} {default} }
74 \DefTblrTemplate {lastfoot}      {numodel} {}
75 \NewTblrTheme {numodel}
76 {
77   \SetTblrTemplate {firsthead}      {numodel}
78   \SetTblrTemplate {middlehead, lasthead} {numodel}
79   \SetTblrTemplate {firstfoot, middlefoot} {numodel}
80   \SetTblrTemplate {lastfoot}      {numodel}
81 }
82 % Tighten the default row separation of all tabularray tables: the
83 % \textmodel rule listing is denser and more compact than the
84 % tabularray default. Users who want the default spacing back in
85 % their own tables can simply issue \SetTblrInner{rowsep=<value>} in
86 % their document.
87 \SetTblrInner{rowsep=0pt}
88 \RequirePackage{numodel-plot}
89
90 % =====
91 % Non-expl3 helpers: '\nm@def@num@cs' / '\nm@def@qty@cs' / '\nm@def@pre@cs'
92 % define '\<fullname><suffix>' macros whose body contains literal
93 % siunitx options. Defined OUTSIDE \ExplSyntaxOn so the option-string
94 % colons ('-3:n', '-0:0') and other punctuation keep normal catcodes.
95 % Storing those bodies inside an expl3-context would tokenise ':' as
96 % letter, which then breaks siunitx's option parser at expansion time.
97 % =====
98 \newcommand{\NumodelDefNumCs}[3]{% #1 = fullname, #2 = start expr, #3 = sigfigs
99   \expandafter\gdef\csname #1num\endcsname{%
100     \num[evaluate-expression=true,
101           round-mode=figures, round-precision=#3,
102           exponent-mode=threshold, exponent-thresholds=-3:#3]
103     {#2}\relax
104 }%

```

```

105 }
106 \newcommand{\NumodelDefQtyCs}[4]{% #1 = fullname, #2 = expr, #3 = sigfigs, #4 = unit
107   \expandafter\gdef\csname #1qty\endcsname{%
108     \qty[evaluate-expression=true,
109       round-mode=figures, round-precision=#3,
110       exponent-mode=threshold, exponent-thresholds=-3:#3]
111     {#2}{#4}\relax
112   }%
113 }
114 \newcommand{\NumodelDefPreCs}[4]{% same parameters as above
115   \expandafter\gdef\csname #1pre\endcsname{%
116     \qty[evaluate-expression=true,
117       round-mode=figures, round-precision=#3,
118       prefix-mode=combine-exponent,
119       exponent-mode=engineering,
120       exponent-thresholds=-0:0]
121     {#2}{#4}\relax
122   }%
123 }
124
125 \ExplSyntaxOn
126
127 % =====
128 % Lua module for data storage (O(1) append, O(N) total)
129 % =====
130 % All variable values are stored per step in Lua tables. This
131 % replaces the O(N^2) \xdef accumulation for coordinates and makes
132 % min/max tracking efficient. After \computemodel the results are
133 % pushed back to TeX macros.
134
135 \begin{luacode*}
136 local f = kpse.find_file("numodel.lua", "tex")
137 if f and f ~= "" then
138   dofile(f)
139 else
140   tex.error("numodel: cannot find companion Lua module numodel.lua."
141     .. " Install it in a directory searched by kpse"
142     .. " (e.g. TEXMFHOME/tex/lualatex/numodel/).")
143 end
144 \end{luacode*}
145
146 % =====
147 % Internal data structures
148 % =====
149
150 \seq_new:N \g_mvar_names_seq      % all model variable names
151 \seq_new:N \g_mvar_start_seq      % names with an initial value (for the table)
152 \seq_new:N \g_mrul_seq            % model rules (display strings)
153 \seq_new:N \g_mrul_type_seq       % type per display row: rule | cont
154 \seq_new:N \g_mrul_calc_seq       % model rules for execution: {name}{expr}
155 \int_new:N \g_mrul_counter_int    % rule counter (display numbering)
156 \tl_new:N \g_numodel_stop_expr_tl % stop-condition expression for execution
157 \int_new:N \g_numodel_steps_int    % number of executed steps
158 \int_new:N \g_numodel_maxiter_int  % safety limit (init via \numodelsetup)
159 \int_new:N \g_numodel_gridmaxx_int % \graphicmodel wrap threshold (0 = off)
160
161 % --- Graphic-model infrastructure (Phase 1) ---
162 \tl_new:N \l_numodel_tmp_tl       % temporary helper
163 \tl_new:N \l_numodel_scratch_tl   % scratch for type/text checks
164 \tl_new:N \l_numodel_scratch_y_tl % scratch y-coord (svg lookup)
165
166 % Diagram style for \graphicmodel (see \numodelsetup):
167 %   tight      -- valve takes the label of the direct inflow variable;
168 %               the aux/const is moved to the valve position and not
169 %               drawn as a separate node (compact, default).
170 %   forrester -- valve has no label; the aux/const stays at its own

```

```

171 %          gridy position; causal arrow from aux/const to the
172 %          valve.
173 %   edu      -- combination: valve takes the label *and* a separate
174 %          aux/const node remains with a causal arrow to it.
175 \tl_new:N \g__numodel_diagram_style_tl
176
177 % Sub-keys that fine-tune the appearance of the Forrester diagram
178 % (see \numodelsetup). Empty value = "follow diagram-style default".
179 %   flowarrow-style      : hollow | filled
180 %          forrester -> hollow, tight|edu -> filled
181 %   valve-style          : valve | circle | edu
182 %          forrester -> valve, tight|edu -> edu
183 %   flowarrow-cloud-tip: true | false
184 %          forrester -> true, tight|edu -> false
185 % Per-variable override for flowarrow-cloud-tip is stored in
186 % \<name>flowcloud (set via the [flowarrow-cloud-tip=...] key on
187 % \mvar; empty means "follow global key").
188 \tl_new:N \g__numodel_flowarrow_style_tl
189 \tl_new:N \g__numodel_valve_style_tl
190 \tl_new:N \g__numodel_flowcloud_tl
191
192 % Units in the \textmodel initial-values column (see \numodelsetup):
193 %   true  (default) -- initial value rendered via \<name>qty
194 %          (number + unit)
195 %   false          -- initial value rendered via \<name>num
196 %          (number only)
197 % Per-table override via \textmodel[units=true|false].
198 \bool_new:N \g__numodel_units_bool
199
200 % tabularray environment used by \textmodel (see \numodelsetup{tblrenv}):
201 %   longtblr -- default; page-breakable; suitable for body text.
202 %   tblr      -- basic; no page breaks; choose this when \textmodel sits
203 %          inside an outer environment that already suppresses page
204 %          breaks (subfigure, minipage, ...).
205 %   talltblr -- like tblr (no page breaks) but supports captions/notes.
206 % Per-table override via \textmodel[tblrenv=...].
207 \tl_new:N \g__numodel_tblrenv_tl
208
209 % Decimal separator for \textmodel initial values and \diagrammodel
210 % tick values (see \numodelsetup{decimal-separator}):
211 %   point -- '.' (siunitx output-decimal-marker={.})
212 %   comma -- ',' (siunitx output-decimal-marker={,})
213 % The default follows syntax: english -> point, coachtaal -> comma.
214 % An explicit decimal-separator key sets
215 % \g__numodel_dsep_explicit_bool so that a later syntax change no
216 % longer overrules the choice.
217 \tl_new:N \g__numodel_dsep_tl
218 \bool_new:N \g__numodel_dsep_explicit_bool
219
220 % =====
221 % Syntax lookup
222 % =====
223 % Determines the language of the text-rendered model. Affects display
224 % only (\textmodel, \mruletext, \mstop); \computemodel always uses
225 % \fpeval internally and is language-agnostic. Each value of
226 % \g__numodel_syntax_tl is a tag that selects both the backing
227 % \__numodel_kw_<tag>_<key>: macros and the file numodel-<tag>.def
228 % that defines them; the package ships EN (English/XMILE) and NL
229 % (Dutch/CoachTaal).
230 %
231 % The CTAN default is EN. The initial value is set below via
232 % \numodelsetup.
233
234 \tl_new:N \g__numodel_syntax_tl
235
236 % Lookup (fully expandable): \__numodel_kw:n {<key>} returns the

```

```

237 % translation in the current syntax language. The key also controls
238 % the surrounding spaces (see keys 'then' vs 'then_nl' etc.). The
239 % backing macros \_numodel_kw_<LANG>_<key>: are provided by the
240 % language file numodel-<LANG>.def loaded by \_numodel_load_syntax_def:n
241 % (see below).
242 \cs_new:Npn \_numodel_kw:n #1
243 { \use:c { \_numodel_kw_ \g__numodel_syntax_tl _ #1 : } }
244
245 % Pre-computed translations in tl variables so they are usable via
246 % \u{} in l3regex replacement text. (l3regex treats {...} in the
247 % replacement as brace groups, so a bare \_numodel_kw:n{key} call
248 % fails there.)
249 \tl_new:N \g__numodel_kw_if_tl
250 \tl_new:N \g__numodel_kw_then_tl
251 \tl_new:N \g__numodel_kw_then_nl_tl
252 \tl_new:N \g__numodel_kw_else_tl
253 \tl_new:N \g__numodel_kw_else_nl_tl
254 \tl_new:N \g__numodel_kw_endif_tl
255 \tl_new:N \g__numodel_kw_endif_nl_tl
256 \tl_new:N \g__numodel_kw_and_tl
257 \tl_new:N \g__numodel_kw_or_tl
258 \tl_new:N \g__numodel_kw_not_tl
259 \tl_new:N \g__numodel_kw_sign_tl
260 \tl_new:N \g__numodel_kw_abs_tl
261 \tl_new:N \g__numodel_kw_sqrt_tl
262 \tl_new:N \g__numodel_kw_exp_tl
263 \tl_new:N \g__numodel_kw_ln_tl
264 \tl_new:N \g__numodel_kw_sin_tl
265 \tl_new:N \g__numodel_kw_cos_tl
266 \tl_new:N \g__numodel_kw_tan_tl
267 \tl_new:N \g__numodel_kw_asin_tl
268 \tl_new:N \g__numodel_kw_acos_tl
269 \tl_new:N \g__numodel_kw_stop_tl
270
271 % Recomputes all tl caches from \g__numodel_syntax_tl. Must be
272 % called after any change to the syntax language.
273 \cs_new_protected:Npn \_numodel_refresh_kw:
274 {
275   \tl_gset:Ne \g__numodel_kw_if_tl { \_numodel_kw:n {if} }
276   \tl_gset:Ne \g__numodel_kw_then_tl { \_numodel_kw:n {then} }
277   \tl_gset:Ne \g__numodel_kw_then_nl_tl { \_numodel_kw:n {then_nl} }
278   \tl_gset:Ne \g__numodel_kw_else_tl { \_numodel_kw:n {else} }
279   \tl_gset:Ne \g__numodel_kw_else_nl_tl { \_numodel_kw:n {else_nl} }
280   \tl_gset:Ne \g__numodel_kw_endif_tl { \_numodel_kw:n {endif} }
281   \tl_gset:Ne \g__numodel_kw_endif_nl_tl { \_numodel_kw:n {endif_nl} }
282   \tl_gset:Ne \g__numodel_kw_and_tl { \_numodel_kw:n {and} }
283   \tl_gset:Ne \g__numodel_kw_or_tl { \_numodel_kw:n {or} }
284   \tl_gset:Ne \g__numodel_kw_not_tl { \_numodel_kw:n {not} }
285   \tl_gset:Ne \g__numodel_kw_sign_tl { \_numodel_kw:n {sign} }
286   \tl_gset:Ne \g__numodel_kw_abs_tl { \_numodel_kw:n {abs} }
287   \tl_gset:Ne \g__numodel_kw_sqrt_tl { \_numodel_kw:n {sqrt} }
288   \tl_gset:Ne \g__numodel_kw_exp_tl { \_numodel_kw:n {exp} }
289   \tl_gset:Ne \g__numodel_kw_ln_tl { \_numodel_kw:n {ln} }
290   \tl_gset:Ne \g__numodel_kw_sin_tl { \_numodel_kw:n {sin} }
291   \tl_gset:Ne \g__numodel_kw_cos_tl { \_numodel_kw:n {cos} }
292   \tl_gset:Ne \g__numodel_kw_tan_tl { \_numodel_kw:n {tan} }
293   \tl_gset:Ne \g__numodel_kw_asin_tl { \_numodel_kw:n {asin} }
294   \tl_gset:Ne \g__numodel_kw_acos_tl { \_numodel_kw:n {acos} }
295   \tl_gset:Ne \g__numodel_kw_stop_tl { \_numodel_kw:n {stop} }
296   % tabulararray continuation markers (used by longtblr in \textmodel)
297   \tl_gset:Ne \tblrcontfootname { \_numodel_kw:n {contfoot} }
298   \tl_gset:Ne \tblrcontheadname { \_numodel_kw:n {conthead} }
299 }
300
301 % Helper: \_numodel_kwt:n {<key>} expands to \text{<kw-value>}.
302 % Meant for use inside \tl_set:Ne / \seq_gput_right:Ne -- the kw is

```

```

303 % inserted during e-expansion while \text remains protected.
304 \cs_new:Npn \__numodel_kwt:n #1 { \text { \__numodel_kw:n {#1} } }
305
306 % Language-file loader. Each tag <LANG> is backed by a file
307 % numodel-<LANG>.def installed in a kpse-searched directory; the
308 % package ships numodel-EN.def and numodel-NL.def, users can drop
309 % additional files in TEXMFHOME/tex/latex/numodel/ without rebuilding
310 % the package.
311 %
312 % A small alias property maps the historical long names to the
313 % canonical two-letter tag used in the file name and the internal
314 % macro names. Users who add their own file just pass the file's tag
315 % to syntax=<tag> directly; no alias is required.
316 \prop_new:N \g__numodel_syntax_aliases_prop
317 \prop_gput:Nnn \g__numodel_syntax_aliases_prop { english } { EN }
318 \prop_gput:Nnn \g__numodel_syntax_aliases_prop { coactaal } { NL }
319 \prop_gput:Nnn \g__numodel_syntax_aliases_prop { dutch } { NL }
320
321 % Tags whose .def file has already been input during this run, so the
322 % lookup only triggers \InputIfFileExists the first time.
323 \seq_new:N \g__numodel_loaded_syntax_seq
324
325 \msg_new:nnn { numodel } { unknown-syntax }
326 {
327   Cannot-load-syntax~'#1':~file~'numodel-#1.def'~not~found~by~
328   kpse.~The~package~ships~with~EN~(English)~and~NL~(Dutch~
329   CoachTaal);~drop~your~own~numodel-<LANG>.def~in~
330   TEXMFHOME/tex/latex/numodel/~to~add~more~languages.
331 }
332
333 % \__numodel_load_syntax_def:n {<tag>}
334 % Inputs numodel-<tag>.def the first time it is requested. Raises a
335 % LaTeX error if the file is not found by kpse.
336 \cs_new_protected:Npn \__numodel_load_syntax_def:n #1
337 {
338   \seq_if_in:NnF \g__numodel_loaded_syntax_seq {#1}
339   {
340     \InputIfFileExists { numodel-#1.def }
341     { \seq_gput_right:Nn \g__numodel_loaded_syntax_seq {#1} }
342     { \msg_error:nnn { numodel } { unknown-syntax } {#1} }
343   }
344 }
345
346 \tl_new:N \l__numodel_syntax_tag_tl
347
348 % \__numodel_set_syntax:n {<value>}
349 % Public-facing setter behind the syntax key. Resolves aliases,
350 % loads the matching .def file (once), publishes the canonical tag
351 % in \g__numodel_syntax_tl, refreshes the keyword cache, and -- when
352 % the user has not pinned a decimal separator explicitly -- adopts the
353 % language's preferred decimal mark via the optional
354 % \__numodel_kw_<LANG>_dsep_default: hook.
355 \cs_new_protected:Npn \__numodel_set_syntax:n #1
356 {
357   \prop_get:NnNF \g__numodel_syntax_aliases_prop {#1}
358   \l__numodel_syntax_tag_tl
359   { \tl_set:Nn \l__numodel_syntax_tag_tl {#1} }
360   \exp_args:NV \__numodel_load_syntax_def:n \l__numodel_syntax_tag_tl
361   \tl_gset_eq:NN \g__numodel_syntax_tl \l__numodel_syntax_tag_tl
362   \__numodel_refresh_kw:
363   \bool_if:NF \g__numodel_dsep_explicit_bool
364   {
365     \cs_if_exist:cTF
366     { __numodel_kw_ \g__numodel_syntax_tl _dsep_default: }
367     { \tl_gset:Nn \g__numodel_dsep_tl { \__numodel_kw:n {dsep_default} } }
368     { \tl_gset:Nn \g__numodel_dsep_tl { point } }

```



```

369     }
370 }
371
372 % Applies the decimal separator (only for the duration of the
373 % surrounding TeX group, so a document-wide \sisetup is left
374 % untouched). Calls both siunitx (for \num/\qty in \textmodel
375 % initial values) and pgfplots' tick styles (for \diagrammodel tick
376 % labels) so the choice is consistent everywhere.
377 \cs_new_protected:Npn \__numodel_apply_dsep:
378 {
379     \str_if_eq:VnTF \g__numodel_dsep_tl { comma }
380     {
381         \sisetup{ output-decimal-marker = {,} }
382         % Append behind the existing numodel/axis style so our
383         % xticklabel-style replaces what numodel-plot hard-codes.
384         % The \pgfplotsset mutation is \def-based and hence
385         % group-local.
386         \pgfplotsset
387         {
388             numodel/axis/.append-style=
389             {
390                 xticklabel-style=
391                 { /pgf/number-format/.cd, use-comma, /tikz/.cd } ,
392                 yticklabel-style=
393                 { /pgf/number-format/.cd, use-comma, /tikz/.cd } ,
394             }
395         }
396     }
397     {
398         \sisetup{ output-decimal-marker = {.} }
399         \pgfplotsset
400         {
401             numodel/axis/.append-style=
402             {
403                 xticklabel-style=
404                 { /pgf/number-format/.cd, use-period, /tikz/.cd } ,
405                 yticklabel-style=
406                 { /pgf/number-format/.cd, use-period, /tikz/.cd } ,
407             }
408         }
409     }
410 }
411
412 % =====
413 % Configuration command \numodelsetup
414 % =====
415 % Runtime API for settings. Keys:
416 % syntax      -- language tag (file numodel-<tag>.def must be on kpse
417 %               path). Built-in: EN, NL. Legacy aliases: english,
418 %               coactaal, dutch.
419 % maxiter      -- safety limit for \computemodel (default 20000)
420 % graphscalex  -- horizontal grid spacing \dgridx for Forrester
421 %               diagrams (default 2)
422 % graphscaley  -- vertical grid spacing \dgridy for Forrester
423 %               diagrams (default 2)
424 % stockwidth   -- half-width of the stock rectangle (default 0.375)
425
426 % Helper for choice-with-empty-reset: validates #4 against #3 (a
427 % comma-separated whitelist) and writes it into #1 (a tl). An empty
428 % value clears the tl (which means "follow the diagram-style
429 % default" for the flowarrow-style / valve-style /
430 % flowarrow-cloud-tip keys). Any other value triggers a warning.
431 \msg_new:nnn { numodel } { bad-choice }
432 { Key~'#1'~accepts-only~#2,~or~empty~to~reset.~Got:~'#3'. }
433
434 \cs_new_protected:Npn \__numodel_setup_choice:Nnnn #1 #2 #3 #4

```

```

435 {
436   \tl_if_blank:nTF {#4}
437   { \tl_gclear:N #1 }
438   {
439     \clist_if_in:nnTF {#3} {#4}
440     { \tl_gset:Nn #1 {#4} }
441     { \msg_warning:nnnnn { numodel } { bad-choice } {#2} {#3} {#4} }
442   }
443 }
444
445 % Local-tl variant: same validation, local set/clear. Used by the
446 % \graphicmodel one-render override and the per-\mvar override.
447 \cs_new_protected:Npn \__numodel_local_choice:Nnnn #1 #2 #3 #4
448 {
449   \tl_if_blank:nTF {#4}
450   { \tl_clear:N #1 }
451   {
452     \clist_if_in:nnTF {#3} {#4}
453     { \tl_set:Nn #1 {#4} }
454     { \msg_warning:nnnnn { numodel } { bad-choice } {#2} {#3} {#4} }
455   }
456 }
457
458 \keys_define:nn { numodel / setup }
459 {
460   syntax .code:n =
461   { \__numodel_set_syntax:n {#1} },
462   maxiter .int_gset:N = \g_numodel_maxiter_int,
463   graphscalex .code:n = { \tl_gset:Nn \dgridx {#1} },
464   graphscaley .code:n = { \tl_gset:Nn \dgridy {#1} },
465   stockwidth .code:n = { \tl_gset:Nn \halfstockwidth {#1} },
466   gridmaxx .int_gset:N = \g_numodel_gridmaxx_int,
467   diagram-style .choice:,
468   diagram-style / tight .code:n =
469   { \tl_gset:Nn \g_numodel_diagram_style_tl { tight } },
470   diagram-style / forrester .code:n =
471   { \tl_gset:Nn \g_numodel_diagram_style_tl { forrester } },
472   diagram-style / edu .code:n =
473   { \tl_gset:Nn \g_numodel_diagram_style_tl { edu } },
474   flowarrow-style .code:n =
475   { \__numodel_setup_choice:Nnnn \g_numodel_flowarrow_style_tl
476     { flowarrow-style } { hollow , filled } {#1} },
477   valve-style .code:n =
478   { \__numodel_setup_choice:Nnnn \g_numodel_valve_style_tl
479     { valve-style } { valve , circle , edu } {#1} },
480   flowarrow-cloud-tip .code:n =
481   { \__numodel_setup_choice:Nnnn \g_numodel_flowcloud_tl
482     { flowarrow-cloud-tip } { true , false } {#1} },
483   units .choice:,
484   units / true .code:n =
485   { \bool_gset_true:N \g_numodel_units_bool },
486   units / false .code:n =
487   { \bool_gset_false:N \g_numodel_units_bool },
488   tblrenv .choice:,
489   tblrenv / tblr .code:n =
490   { \tl_gset:Nn \g_numodel_tblrenv_tl { tblr } },
491   tblrenv / longtblr .code:n =
492   { \tl_gset:Nn \g_numodel_tblrenv_tl { longtblr } },
493   tblrenv / talltblr .code:n =
494   { \tl_gset:Nn \g_numodel_tblrenv_tl { talltblr } },
495   decimal-separator .choice:,
496   decimal-separator / comma .code:n =
497   {
498     \tl_gset:Nn \g_numodel_dsep_tl { comma }
499     \bool_gset_true:N \g_numodel_dsep_explicit_bool
500   },

```

```

501     decimal-separator / point .code:n =
502     {
503         \tl_gset:Nn \g__numodel_dsep_tl { point }
504         \bool_gset_true:N \g__numodel_dsep_explicit_bool
505     },
506 }
507
508 \NewDocumentCommand{\numodelsetup}{m}
509 { \keys_set:nn { numodel / setup } {#1} }
510
511 % Defaults (this call also initialises \g__numodel_syntax_tl,
512 % \g__numodel_maxiter_int, \dgridx, \dgridy, \halfstockwidth and
513 % \g__numodel_diagram_style_tl).
514 \numodelsetup
515 {
516     syntax          = EN,
517     maxiter         = 20000,
518     graphscalex     = 2,
519     graphscaley     = 2,
520     stockwidth      = 0.375,
521     diagram-style   = tight,
522     units           = true,
523     tblrenv         = longtblr,
524 }
525
526 % Package-time options. \usepackage[syntax=NL]{numodel} delegates
527 % to the same key infrastructure as \numodelsetup.
528 \keys_define:nn { numodel / pkg }
529 {
530     syntax          .meta:nn = { numodel / setup }{ syntax          = #1 },
531     maxiter         .meta:nn = { numodel / setup }{ maxiter         = #1 },
532     graphscalex     .meta:nn = { numodel / setup }{ graphscalex     = #1 },
533     graphscaley     .meta:nn = { numodel / setup }{ graphscaley     = #1 },
534     stockwidth      .meta:nn = { numodel / setup }{ stockwidth      = #1 },
535     gridmaxx        .meta:nn = { numodel / setup }{ gridmaxx        = #1 },
536     diagram-style   .meta:nn = { numodel / setup }{ diagram-style   = #1 },
537     flowarrow-style .meta:nn = { numodel / setup }{ flowarrow-style = #1 },
538     valve-style     .meta:nn = { numodel / setup }{ valve-style     = #1 },
539     flowarrow-cloud-tip .meta:nn = { numodel / setup }{ flowarrow-cloud-tip = #1 },
540     units           .meta:nn = { numodel / setup }{ units           = #1 },
541     tblrenv         .meta:nn = { numodel / setup }{ tblrenv         = #1 },
542     decimal-separator .meta:nn = { numodel / setup }{ decimal-separator = #1 },
543 }
544 \ProcessKeyOptions [ numodel / pkg ]
545
546 % =====
547 % Prefix system
548 % =====
549 % Each model lives under a prefix (a short string). The "live state"
550 % sits in the \g_mvar_*, \g_mrul_*, \g_numodel_* variables above;
551 % \newmodelprefix and \switchmodelprefix swap that state to/from
552 % per-prefix backup storage.
553 %
554 % - \g_numodel_current_prefix_tl: current prefix (empty at package load)
555 % - \g_numodel_prefixes_seq: list of all registered prefixes
556 % - \l_numodel_cmd_prefix_tl: prefix passed via [prefix=...] key
557 % - \l_numodel_eff_prefix_tl: effective prefix for the current command
558
559 \tl_new:N \g_numodel_current_prefix_tl
560 \seq_new:N \g_numodel_prefixes_seq
561 \tl_new:N \l_numodel_cmd_prefix_tl
562 \tl_new:N \l_numodel_eff_prefix_tl
563 \tl_new:N \l_numodel_fullname_tl
564
565 % Per-prefix storage: on swap the current state is copied to
566 % \g__numodel_<P>_<slot>_{seq,tl,int,prop}. Load goes the other way.

```

```

567 \cs_new_protected:Npn \__numodel_save_state:n #1
568 {
569   \seq_gset_eq:cN { g__numodel_ #1 _vars_seq      } \g_mvar_names_seq
570   \seq_gset_eq:cN { g__numodel_ #1 _starts_seq    } \g_mvar_start_seq
571   \seq_gset_eq:cN { g__numodel_ #1 _rules_seq     } \g_mrul_seq
572   \seq_gset_eq:cN { g__numodel_ #1 _ruletypes_seq } \g_mrul_type_seq
573   \seq_gset_eq:cN { g__numodel_ #1 _rulecalc_seq  } \g_mrul_calc_seq
574   \int_gset:cn { g__numodel_ #1 _rulecounter_int }
575   { \int_use:N \g_mrul_counter_int }
576   \tl_gset_eq:cN { g__numodel_ #1 _stopexpr_tl   } \g_numodel_stop_expr_tl
577   \int_gset:cn { g__numodel_ #1 _steps_int      }
578   { \int_use:N \g_numodel_steps_int }
579 }
580
581 \cs_new_protected:Npn \__numodel_load_state:n #1
582 {
583   \seq_gset_eq:Nc \g_mvar_names_seq { g__numodel_ #1 _vars_seq      }
584   \seq_gset_eq:Nc \g_mvar_start_seq { g__numodel_ #1 _starts_seq    }
585   \seq_gset_eq:Nc \g_mrul_seq      { g__numodel_ #1 _rules_seq     }
586   \seq_gset_eq:Nc \g_mrul_type_seq { g__numodel_ #1 _ruletypes_seq }
587   \seq_gset_eq:Nc \g_mrul_calc_seq { g__numodel_ #1 _rulecalc_seq  }
588   \int_gset:Nn \g_mrul_counter_int
589   { \int_use:c { g__numodel_ #1 _rulecounter_int } }
590   \tl_gset_eq:Nc \g_numodel_stop_expr_tl { g__numodel_ #1 _stopexpr_tl }
591   \int_gset:Nn \g_numodel_steps_int
592   { \int_use:c { g__numodel_ #1 _steps_int } }
593 }
594
595 \cs_new_protected:Npn \__numodel_clear_state:
596 {
597   \seq_gclear:N \g_mvar_names_seq
598   \seq_gclear:N \g_mvar_start_seq
599   \seq_gclear:N \g_mrul_seq
600   \seq_gclear:N \g_mrul_type_seq
601   \seq_gclear:N \g_mrul_calc_seq
602   \int_gzero:N \g_mrul_counter_int
603   \tl_gclear:N \g_numodel_stop_expr_tl
604   \int_gzero:N \g_numodel_steps_int
605 }
606
607 % Initialise per-prefix backup variables (once, at \newmodelprefix).
608 \cs_new_protected:Npn \__numodel_init_backup:n #1
609 {
610   \seq_new:c { g__numodel_ #1 _vars_seq      }
611   \seq_new:c { g__numodel_ #1 _starts_seq    }
612   \seq_new:c { g__numodel_ #1 _rules_seq     }
613   \seq_new:c { g__numodel_ #1 _ruletypes_seq }
614   \seq_new:c { g__numodel_ #1 _rulecalc_seq  }
615   \int_new:c { g__numodel_ #1 _rulecounter_int }
616   \tl_new:c { g__numodel_ #1 _stopexpr_tl   }
617   \int_new:c { g__numodel_ #1 _steps_int      }
618 }
619
620 % Public commands for prefix management
621 \NewDocumentCommand{\newmodelprefix}{m}
622 {
623   \typeout{NEWPREFIX~start:~#1}
624   \seq_if_in:NnTF \g_numodel_prefixes_seq {#1}
625   { \msg_error:nnn { numodel } { prefix-exists } {#1} }
626   {
627     \typeout{NEWPREFIX~save~current:~\g_numodel_current_prefix_tl}
628     % Save current state under the previous prefix (if any)
629     \tl_if_empty:NF \g_numodel_current_prefix_tl
630     { \exp_args:NV \__numodel_save_state:n \g_numodel_current_prefix_tl }
631     \typeout{NEWPREFIX~register}
632     % Register new prefix + init backup vars + Lua state

```

```

633     \seq_gput_right:Nn \g_numodel_prefixes_seq {#1}
634     \typeout{NEWPREFIX~init_backup}
635     \__numodel_init_backup:n {#1}
636     \typeout{NEWPREFIX~lua_init}
637     \directlua{ numodel.init_prefix("#1") }
638     \typeout{NEWPREFIX~clear_state}
639     % Clear current state and set the prefix
640     \__numodel_clear_state:
641     \typeout{NEWPREFIX~set_current}
642     \tl_gset:Nn \g_numodel_current_prefix_tl {#1}
643     \typeout{NEWPREFIX~done:~#1}
644   }
645 }
646
647 \NewDocumentCommand{\switchmodelprefix}{ m }
648 {
649   \seq_if_in:NnTF \g_numodel_prefixes_seq {#1}
650   {
651     % Save current state (if any), load the new one
652     \tl_if_empty:NF \g_numodel_current_prefix_tl
653     { \exp_args:NV \__numodel_save_state:n \g_numodel_current_prefix_tl }
654     \__numodel_load_state:n {#1}
655     \tl_gset:Nn \g_numodel_current_prefix_tl {#1}
656   }
657   { \msg_error:nnn { numodel } { prefix-unknown } {#1} }
658 }
659
660 % Iterate over *all* variables from *all* prefixes.
661 % #1 = code that uses ##1 = full variable name (prefix + short).
662 % Public registration API for external tools such as worksheet.tex.
663 \NewDocumentCommand{\NumodelForEachVar}{ +m }
664 {
665   \seq_map_inline:Nn \g_numodel_prefixes_seq
666   {
667     \seq_map_inline:cn { g__numodel_ ##1 _vars_seq } {#1}
668   }
669 }
670
671 % Resolve the effective prefix for a command call. Takes
672 % [prefix=<p>] from the keyval argument; otherwise falls back to the
673 % current prefix. Stores the result in \l_numodel_eff_prefix_tl.
674 \keys_define:nn { numodel / cmd }
675 {
676   prefix .tl_set:N = \l__numodel_cmd_prefix_tl,
677   % \graphicmodel-only: temporary override of the global
678   % diagram-style. Cleared after every \graphicmodel call.
679   diagram-style .choice:,
680   diagram-style / tight .code:n =
681   { \tl_set:Nn \l__numodel_cmd_diagstyle_tl { tight } },
682   diagram-style / forrester .code:n =
683   { \tl_set:Nn \l__numodel_cmd_diagstyle_tl { forrester } },
684   diagram-style / edu .code:n =
685   { \tl_set:Nn \l__numodel_cmd_diagstyle_tl { edu } },
686   % \graphicmodel-only: temporary overrides of the corresponding
687   % global flowarrow/valve/cloud keys. Empty after every call.
688   flowarrow-style .code:n =
689   { \__numodel_local_choice:Nnnn \l__numodel_cmd_flowarrow_tl
690     { flowarrow-style } { hollow , filled } {#1} },
691   valve-style .code:n =
692   { \__numodel_local_choice:Nnnn \l__numodel_cmd_valve_tl
693     { valve-style } { valve , circle , edu } {#1} },
694   flowarrow-cloud-tip .code:n =
695   { \__numodel_local_choice:Nnnn \l__numodel_cmd_flowcloud_tl
696     { flowarrow-cloud-tip } { true , false } {#1} },
697   % \textmodel-only: temporary override of the global units bool.
698   % Cleared after every \textmodel call.

```

```

699     units                                .choice:,
700     units / true                         .code:n =
701     { \tl_set:Nn \l__numodel_cmd_units_tl { true } },
702     units / false                        .code:n =
703     { \tl_set:Nn \l__numodel_cmd_units_tl { false } },
704     % \textmodel-only: temporary override of the global tblrenv tl.
705     % Cleared after every \textmodel call.
706     tblrenv                              .choice:,
707     tblrenv / tblr                       .code:n =
708     { \tl_set:Nn \l__numodel_cmd_tblrenv_tl { tblr } },
709     tblrenv / longtblr                   .code:n =
710     { \tl_set:Nn \l__numodel_cmd_tblrenv_tl { longtblr } },
711     tblrenv / talltblr                   .code:n =
712     { \tl_set:Nn \l__numodel_cmd_tblrenv_tl { talltblr } },
713 }
714 \tl_new:N \l__numodel_cmd_diagstyle_tl
715 \tl_new:N \l__numodel_cmd_flowarrow_tl
716 \tl_new:N \l__numodel_cmd_valve_tl
717 \tl_new:N \l__numodel_cmd_flowcloud_tl
718 \tl_new:N \l__numodel_cmd_units_tl
719 \tl_new:N \l__numodel_cmd_tblrenv_tl
720
721 \cs_new_protected:Npn \__numodel_resolve_prefix:n #1
722 {
723     \tl_clear:N \l__numodel_cmd_prefix_tl
724     \keys_set:nn { numodel / cmd } {#1}
725     \tl_if_empty:NTF \l__numodel_cmd_prefix_tl
726     { \tl_set_eq:NN \l__numodel_eff_prefix_tl \g_numodel_current_prefix_tl }
727     { \tl_set_eq:NN \l__numodel_eff_prefix_tl \l__numodel_cmd_prefix_tl }
728 }
729
730 % Execute code under a temporarily different prefix (via state swap).
731 % #1 = target prefix (tl, in variable)
732 % #2 = code
733 \cs_new_protected:Npn \__numodel_with_prefix:Nn #1 #2
734 {
735     \tl_if_eq:NNTF #1 \g_numodel_current_prefix_tl
736     { #2 } % already the current prefix; no swap needed
737     {
738         \tl_set_eq:NN \l__numodel_saved_prefix_tl \g_numodel_current_prefix_tl
739         \exp_args:NV \switchmodelprefix #1
740         #2
741         \exp_args:NV \switchmodelprefix \l__numodel_saved_prefix_tl
742     }
743 }
744 \tl_new:N \l__numodel_saved_prefix_tl
745
746 % Build the full name = <prefix><shortname>. Stored in
747 % \l__numodel_fullname_tl.
748 \cs_new_protected:Npn \__numodel_set_fullname:n #1
749 {
750     \tl_set:Nn \l__numodel_fullname_tl { \l__numodel_eff_prefix_tl #1 }
751 }
752
753 % Keys for the optional argument of \mvar (grid position +
754 % initial-value aliases + prefix).
755 \tl_new:N \l__numodel_alias_tl
756 \tl_new:N \l__numodel_aliasleft_tl
757 \tl_new:N \l__numodel_aliasright_tl
758
759 \keys_define:nn { numodel / mvar }
760 {
761     prefix      .tl_set:N = \l__numodel_cmd_prefix_tl ,
762     gridx       .tl_set:N = \l__numodel_gridx_tl ,
763     gridy       .tl_set:N = \l__numodel_gridy_tl ,
764     alias       .tl_set:N = \l__numodel_alias_tl ,

```

```

765 aliasleft .tl_set:N = \l__numodel_aliasleft_tl ,
766 aliasright .tl_set:N = \l__numodel_aliasright_tl ,
767 flowarrow-cloud-tip .code:n =
768 { \__numodel_local_choice:Nnnn \l__numodel_mvar_flowcloud_tl
769 { flowarrow-cloud-tip } { true , false } {#1} },
770 }
771 \tl_new:N \l__numodel_mvar_flowcloud_tl
772
773 % =====
774 % \mvar[keys]{name}{display}{startvalue}{unit}{sig}{type}
775 % =====
776 % Declare a model variable. Generates the following macros:
777 %
778 % \<name> -- numeric value (\edef + \fpeval), or warning if empty
779 % \<name>text -- display name for the model table (e.g. F_{res})
780 % \<name>unit -- SI unit (e.g. kN)
781 % \<name>unitraw - raw SI unit (e.g. \kilo\N)
782 % \<name>sign -- number of significant figures
783 % \<name>type -- variable type: stock, constant, aux, system
784 % (Dutch synonyms voorraad/constante/hulp/systeem
785 % are normalised to the canonical English form)
786 % \<name>coord -- coordinate list (empty; filled by \computemodel)
787 % \<name>gridx -- x position in the graphic model (-1 = auto)
788 % \<name>gridy -- y position in the graphic model (-1 = auto)
789 % \<name>num -- number with significance (via \num)
790 % \<name>qty -- number + unit (via \qty)
791 % \<name>pre -- engineering-prefix notation (via \qty)
792 % \<name>alias -- replaces the whole initial-value cell
793 % (empty = default)
794 % \<name>aliasleft -- replaces the left symbol in the initial value
795 % (empty = default)
796 % \<name>aliasright -- replaces the right number in the initial value
797 % (empty = default)
798 %
799 % Keys accepted in [#1]:
800 % gridx, gridy -- position in the graphic model
801 % alias -- replace the entire initial-value cell
802 % (in math mode)
803 % aliasleft -- replace just the left symbol
804 % aliasright -- replace just the right value
805 %
806 % The base value is registered in \g_defqty_names_seq so that
807 % \includeimage expands it automatically.
808 %
809 % With an empty start value (#3 blank), the base value is not
810 % numerically defined; using it issues a warning. Useful for
811 % auxiliary variables computed by \mrule.
812 %
813 % Example:
814 % \mvar{modM}{m}{80}{\kg}{2}{constant}
815 % \mvar{modFres}{F_{res}}{}{\N}{2}{aux}
816 % \mvar[aliasright=\cdots]{modX}{x}{0}{\m}{3}{stock}
817 % \mvar[alias={x \text{?}}]{modX}{x}{0}{\m}{3}{stock}
818 %
819 % Normalise the variable type (sixth \mvar argument) to its
820 % canonical English form. Accepts the canonical names
821 % {stock, constant, aux, system} and the Dutch synonyms
822 % {voorraad, constante, hulp, systeem}. An unknown value is left
823 % as-is and a warning is issued. The result is returned through
824 % \l__numodel_type_tl.
825 \tl_new:N \l__numodel_type_tl
826 \cs_new_protected:Npn \__numodel_normalize_type:n #1
827 {
828 \str_case:nnF {#1}
829 {
830 { stock } { \tl_set:Nn \l__numodel_type_tl { stock } } }

```

```

831     { constant } { \tl_set:Nn \l__numodel_type_tl { constant } }
832     { aux      } { \tl_set:Nn \l__numodel_type_tl { aux      } }
833     { system   } { \tl_set:Nn \l__numodel_type_tl { system   } }
834     { voorraad } { \tl_set:Nn \l__numodel_type_tl { stock    } }
835     { constante } { \tl_set:Nn \l__numodel_type_tl { constant } }
836     { hulp     } { \tl_set:Nn \l__numodel_type_tl { aux      } }
837     { systeem  } { \tl_set:Nn \l__numodel_type_tl { system   } }
838   }
839   {
840     \msg_warning:nne { numodel } { unknown-type } {#1}
841     \tl_set:Nn \l__numodel_type_tl {#1}
842   }
843 }
844
845 \makeatletter
846 \NewDocumentCommand{\mvar}{ O{} m m m m m m }{%
847   \typeout{MVAR~start:~#2}
848   % Parse optional keys (prefix, gridx, gridy, alias, aliasleft, aliasright,
849   % flowarrow-cloud-tip)
850   \tl_set:Nn \l__numodel_gridx_tl { -1 }
851   \tl_set:Nn \l__numodel_gridy_tl { -1 }
852   \tl_clear:N \l__numodel_alias_tl
853   \tl_clear:N \l__numodel_aliasleft_tl
854   \tl_clear:N \l__numodel_aliasright_tl
855   \tl_clear:N \l__numodel_cmd_prefix_tl
856   \tl_clear:N \l__numodel_mvar_flowcloud_tl
857   \typeout{MVAR~before-keys-set}
858   \keys_set:nn { numodel / mvar } {#1}
859   \typeout{MVAR~after-keys-set}
860   \__numodel_resolve_eff_prefix:
861   \typeout{MVAR~eff:~\l__numodel_eff_prefix_tl}
862   \__numodel_with_prefix:Nn \l__numodel_eff_prefix_tl
863   { \__numodel_mvar_body:nnnnnn {#2}{#3}{#4}{#5}{#6}{#7} }%
864   \typeout{MVAR~done:~#2}
865 }
866 \makeatother
867
868 % Resolve the effective prefix from \l__numodel_cmd_prefix_tl.
869 \cs_new_protected:Npn \__numodel_resolve_eff_prefix:
870 {
871   \tl_if_empty:NTF \l__numodel_cmd_prefix_tl
872     { \tl_set_eq:NN \l__numodel_eff_prefix_tl \g_numodel_current_prefix_tl }
873     { \tl_set_eq:NN \l__numodel_eff_prefix_tl \l__numodel_cmd_prefix_tl }
874 }
875
876 % The \mvar body runs in the context of the right prefix (live state
877 % has already been swapped).
878 \cs_new_protected:Npn \__numodel_mvar_body:nnnnnn #1 #2 #3 #4 #5 #6
879 {
880   % Guard: current prefix must not be empty (\newmodelprefix required)
881   \tl_if_empty:NT \g_numodel_current_prefix_tl
882     { \msg_error:nn { numodel } { no-prefix } }
883   % Full name = current prefix + short name (#1)
884   \tl_set:Nn \l__numodel_fullname_tl { \g_numodel_current_prefix_tl #1 }
885   \typeout{MVAR~body~fullname:~\l__numodel_fullname_tl}
886   % Register in the per-prefix seq. The defqty registration is
887   % optional: it only fires when the user has loaded the project-
888   % specific 'defqty' system (used for worksheet expansion).
889   % Standalone, the package works without that seq.
890   \seq_gput_right:NV \g_mvar_names_seq \l__numodel_fullname_tl
891   \cs_if_exist:NT \g_defqty_names_seq
892     { \seq_gput_right:NV \g_defqty_names_seq \l__numodel_fullname_tl }
893   \directlua{ numodel.register(
894     "\g_numodel_current_prefix_tl",
895     "\l__numodel_fullname_tl") }
896   \cs_if_exist:cT { \l__numodel_fullname_tl }

```



```

897     { \msg_warning:nne { numodel } { redef } { \l__numodel_fullname_tl } }
898 \tl_if_blank:nTF {#3}
899     { \cs_gset:cpe { \l__numodel_fullname_tl } { \fp_eval:n { 0 } } }
900     {
901         \cs_gset:cpe { \l__numodel_fullname_tl } { \fp_eval:n {#3} }
902         \seq_gput_right:N \g_mvar_start_seq \l__numodel_fullname_tl
903     }
904 \cs_gset:cpn { \l__numodel_fullname_tl text } {#2}
905 \cs_gset:cpn { \l__numodel_fullname_tl unit } { \unit{#4} }
906 \cs_gset:cpn { \l__numodel_fullname_tl unitraw } {#4}
907 \cs_gset:cpn { \l__numodel_fullname_tl sign } {#5}
908 \__numodel_normalize_type:n {#6}
909 \cs_gset:cpe { \l__numodel_fullname_tl type }
910     { \tl_use:N \l__numodel_type_tl }
911 \cs_gset:cpn { \l__numodel_fullname_tl min } { inf }
912 \cs_gset:cpn { \l__numodel_fullname_tl max } { -inf }
913 \cs_gset:cpe { \l__numodel_fullname_tl gridx } { \tl_use:N \l__numodel_gridx_tl }
914 \cs_gset:cpe { \l__numodel_fullname_tl gridy } { \tl_use:N \l__numodel_gridy_tl }
915 % Preserve the original user input so that \__numodel_build_graphic:
916 % can reset auto-placed positions to -1 on a second invocation.
917 \cs_gset:cpe { \l__numodel_fullname_tl gridxinit } { \tl_use:N \l__numodel_gridx_tl }
918 \cs_gset:cpe { \l__numodel_fullname_tl gridyinit } { \tl_use:N \l__numodel_gridy_tl }
919 % Pillar A - Lua-side meta for compute_layout (additive in A1).
920 % Detokenize text so that \luaescapestring works on bare chars.
921 \tl_set:N \l__numodel_scratch_tl { \detokenize {#2} }
922 \directlua{ numodel.set_meta(
923     "\g_numodel_current_prefix_tl",
924     "\l__numodel_fullname_tl",
925     { type = "\tl_use:N \l__numodel_type_tl",
926       text = "\luaescapestring{\l__numodel_scratch_tl}",
927       gridx = \tl_use:N \l__numodel_gridx_tl,
928       gridy = \tl_use:N \l__numodel_gridy_tl } ) }
929 \cs_gset:cpe { \l__numodel_fullname_tl alias } { \exp_not:V \l__numodel_alias_tl }
930 \cs_gset:cpe { \l__numodel_fullname_tl aliasleft } { \exp_not:V \l__numodel_aliasleft_tl }
931 \cs_gset:cpe { \l__numodel_fullname_tl aliasright } { \exp_not:V \l__numodel_aliasright_tl }
932 \cs_gset:cpe { \l__numodel_fullname_tl flowcloud } { \exp_not:V \l__numodel_mvar_flowcloud_tl }
933 \tl_if_blank:nTF {#3}
934     {
935         \exp_args:N \NumodelDefNumCs \l__numodel_fullname_tl { \fp_eval:n {#3} } {#5}
936         \exp_args:N \NumodelDefQtyCs \l__numodel_fullname_tl { \fp_eval:n {#3} } {#5} {#4}
937         \exp_args:N \NumodelDefPreCs \l__numodel_fullname_tl { \fp_eval:n {#3} } {#5} {#4}
938     }
939 }
940
941 % =====
942 % Warning messages
943 % =====
944
945 \msg_new:nnn { numodel } { redef }
946 { Model-variable~'#1'~is-being-redefined. }
947 \msg_new:nnn { numodel } { empty-use }
948 { Model-variable~'#1'~has-no-start-value-and-is-used-before-assignment. }
949 \msg_new:nnn { numodel } { maxiter }
950 { computemodel~stopped-after~\int_use:N \g_numodel_maxiter_int ~iterations~
951   (safety~limit).~Check-stop-condition. }
952 \msg_new:nnn { numodel } { no-stop }
953 { computemodel:~no-stop-condition-defined.~Use~\token_to_str:N \mstop~space
954   before~\token_to_str:N \computemodel. }
955 \msg_new:nnn { numodel } { prefix-exists }
956 { Model-prefix~'#1'~is-already-registered.~
957   Use~\token_to_str:N \switchmodelprefix~space to~switch-to-an-existing-prefix. }
958 \msg_new:nnn { numodel } { prefix-unknown }
959 { Model-prefix~'#1'~is-not-registered.~
960   Use~\token_to_str:N \newmodelprefix~space to~create-it-first. }
961 \msg_new:nnn { numodel } { no-prefix }
962 { No-current-model-prefix.~

```

```

963 Call~\token_to_str:N \newmodelprefix{<name>}\space before~using~model~commands. }
964 \msg_new:nnn { numodel } { unknown-type }
965 { Unknown~variable~type~'#1'.~
966 Use~one~of~stock,~constant,~aux,~system~
967 (or~the~Dutch~aliases~voorraad,~constante,~hulp,~systeem). }
968
969 % =====
970 % Display helpers
971 % =====
972 % Translate computational expressions into syllabus-style display.
973 %
974 % Automatic translations:
975 % \modXxx -> display name (via \<name>text)
976 % * -> \cdot
977 % >= <= -> \geqslant \leqslant
978 % sign(...) -> SIGN(...) / Teken(...) in coachtaal
979 % abs(...) -> ABS(...) / Abs(...) in coachtaal
980 % sqrt(...) -> Sqrt(...) / Sqrt(...) in coachtaal
981 % exp(...) -> EXP(...) / Exp(...) in coachtaal
982 % ln(...) -> LN(...) / Ln(...) in coachtaal
983 % sin(...) -> SIN(...) / Sin(...) in coachtaal
984 % cos(...) -> COS(...) / Cos(...) in coachtaal
985 % tan(...) -> TAN(...) / Tan(...) in coachtaal
986 % asin(...) -> ARCSIN(...) / Arcsin(...) in coachtaal
987 % acos(...) -> ARCCOS(...) / Arccos(...) in coachtaal
988 % && -> AND / EN in coachtaal
989 % || -> OR / OF in coachtaal
990 % cond ? a : b -> IF cond THEN ... ELSE ... ENDIF (or coachtaal eq.)
991
992 \tl_new:N \l__numodel_display_tl
993 \tl_new:N \l__numodel_lhs_tl
994 \tl_new:N \l__numodel_rhs_tl
995 \tl_new:N \l__numodel_cond_tl
996 \tl_new:N \l__numodel_true_tl
997 \tl_new:N \l__numodel_false_tl
998
999 \cs_new_protected:Npn \__numodel_vars_to_display:N #1
1000 {
1001 \typeout{VTD-input:~\tl_to_str:N #1}
1002 \typeout{VTD-seq:~\seq_use:Nn \g_mvar_names_seq {||}}
1003 % Variable names -> display names
1004 \seq_map_inline:Nn \g_mvar_names_seq
1005 {
1006 \typeout{VTD-iter:~##1}
1007 \cs_if_exist:cT { ##1 text }
1008 {
1009 \tl_set:Nc \l_tmpa_tl { \use:c { ##1 text } }
1010 \typeout{VTD-replace~\c{##1}~with~\l_tmpa_tl}
1011 \regex_replace_all:nnN { \c{##1} } { \u{l_tmpa_tl} } #1
1012 }
1013 }
1014 % Arithmetic operators
1015 \regex_replace_all:nnN { \* } { \c{cdot} \x{20} } #1
1016 \regex_replace_all:nnN { >= } { \c{geqslant} \x{20} } #1
1017 \regex_replace_all:nnN { <= } { \c{leqslant} \x{20} } #1
1018 % Functions (syllabus notation). Order matters: asin/acos must
1019 % match before sin/cos, otherwise the inner sin/cos gets replaced
1020 % first and the outer arc- prefix is left dangling.
1021 \regex_replace_all:nnN { sign \ ( }
1022 { \c{text}\cB{\ \u{g__numodel_kw_sign_tl}\cE}\ ( } #1
1023 \regex_replace_all:nnN { abs \ ( }
1024 { \c{text}\cB{\ \u{g__numodel_kw_abs_tl}\cE}\ ( } #1
1025 \regex_replace_all:nnN { sqrt \ ( }
1026 { \c{text}\cB{\ \u{g__numodel_kw_sqrt_tl}\cE}\ ( } #1
1027 \regex_replace_all:nnN { exp \ ( }
1028 { \c{text}\cB{\ \u{g__numodel_kw_exp_tl}\cE}\ ( } #1

```

```

1029 \regex_replace_all:nnN { ln \ ( }
1030 { \c{text}\cB\{ \u{g__numodel_kw_ln_tl}\cE\}( } #1
1031 \regex_replace_all:nnN { asin \ ( }
1032 { \c{text}\cB\{ \u{g__numodel_kw_asin_tl}\cE\}( } #1
1033 \regex_replace_all:nnN { acos \ ( }
1034 { \c{text}\cB\{ \u{g__numodel_kw_acos_tl}\cE\}( } #1
1035 \regex_replace_all:nnN { sin \ ( }
1036 { \c{text}\cB\{ \u{g__numodel_kw_sin_tl}\cE\}( } #1
1037 \regex_replace_all:nnN { cos \ ( }
1038 { \c{text}\cB\{ \u{g__numodel_kw_cos_tl}\cE\}( } #1
1039 \regex_replace_all:nnN { \b tan \ ( }
1040 { \c{text}\cB\{ \u{g__numodel_kw_tan_tl}\cE\}( } #1
1041 % Logical operators (syllabus notation)
1042 \regex_replace_all:nnN { \&\& }
1043 { \c{text}\cB\{ \u{g__numodel_kw_and_tl}\cE\} } #1
1044 \regex_replace_all:nnN { \|\| }
1045 { \c{text}\cB\{ \u{g__numodel_kw_or_tl}\cE\} } #1
1046 }
1047
1048 % Ternary detection: cond ? true_expr : false_expr
1049 % Converted to: IF cond THEN lhs = true ELSE lhs = false ENDIF
1050 % (or the coactaal equivalent). Supports only flat (non-nested)
1051 % ternaries.
1052 \bool_new:N \l__numodel_is_ternary_bool
1053
1054 \cs_new_protected:Npn \__numodel_parse_ternary:nN #1 #2
1055 {
1056   \bool_set_false:N \l__numodel_is_ternary_bool
1057   \regex_match:nnT { (.+) \? (.+) \: (.+) } {#1}
1058   {
1059     \bool_set_true:N \l__numodel_is_ternary_bool
1060     \tl_set:Nn \l__numodel_cond_tl {#1}
1061     \regex_replace_once:nnN { \s*(.+) \s* \? .+ } { \1 } \l__numodel_cond_tl
1062     \tl_set:Nn \l__numodel_true_tl {#1}
1063     \regex_replace_once:nnN { .+ \? \? \s*(.+) \s* \: .+ } { \1 } \l__numodel_true_tl
1064     \tl_set:Nn \l__numodel_false_tl {#1}
1065     \regex_replace_once:nnN { .+ \: \s*(.+) \s* \Z } { \1 } \l__numodel_false_tl
1066     \__numodel_vars_to_display:N \l__numodel_cond_tl
1067     \__numodel_vars_to_display:N \l__numodel_true_tl
1068     \__numodel_vars_to_display:N \l__numodel_false_tl
1069   }
1070 }
1071
1072 % =====
1073 \mrule[keys]{varname}{calculation}
1074 % =====
1075 % Declare a model rule.
1076 %
1077 % #1 (star)      -- starred variant: multiline IF/THEN/ELSE/ENDIF
1078 %                for ternary expressions
1079 % #2 (optional)  -- keys: alias, aliasleft, aliasright
1080 %                alias      -- replaces the whole display row
1081 %                aliasleft  -- replaces the left-hand side
1082 %                (symbol)
1083 %                aliasright -- replaces the right-hand side
1084 %                (calculation)
1085 %                Use \cdots for omitted parts.
1086 % #3             -- name (string) of the left-hand variable
1087 % #4             -- calculation (with \modXxx macros and \fpeval
1088 %                syntax)
1089 %
1090 % With alias or aliasright the ternary logic is skipped (display
1091 % shows the alias content verbatim, not IF/THEN/ELSE). Execution
1092 % always uses the calculation from #4.
1093 %
1094 % The calculation accepts all \fpeval operators:

```

```

1095 % +, -, *, /, ^, sign(), abs(), sin(), cos(), sqrt(), round()
1096 % Ternary: cond ? expr_true : expr_false
1097 % Logical: && (AND), || (OR), comparisons (<, >, <=, >=)
1098 %
1099 % The rule is stored for both display (\textmodel) and execution
1100 % (\computemodel).
1101 %
1102 % Examples:
1103 % \mrule{modFres}{\modM * \modA}
1104 % \mrule{modFw}{sign(\modV) * \modK * \modV^2}
1105 % \mrule{modA}{(\modT < \modTq) || (\modT > \modTdq) ? \modA + \modDa : \modA - \modDa}
1106 % \mrule[aliasright=\cdots]{modT}{\modT + \modDt}
1107 % \mrule[aliasleft=a_x]{modAx}{\modFgx / \modM}
1108 % \mrule[alias={\text{(hidden)}}]{modAy}{\modFgy / \modM}
1109
1110 \NewDocumentCommand{\mrule}{s O{} m m}{%
1111 \typeout{MRULE~start:~#3}
1112 \bool_set_false:N \l__numodel_is_ternary_bool
1113 % Parse keys (prefix, alias, aliasleft, aliasright; gridx/gridy ignored)
1114 \tl_clear:N \l__numodel_alias_tl
1115 \tl_clear:N \l__numodel_aliasleft_tl
1116 \tl_clear:N \l__numodel_aliasright_tl
1117 \tl_clear:N \l__numodel_cmd_prefix_tl
1118 \keys_set:nn { numodel / mvar } {#2}
1119 \__numodel_resolve_eff_prefix:
1120 \__numodel_with_prefix:Nn \l__numodel_eff_prefix_tl
1121 { \__numodel_mrule_body:nnn {#1}{#3}{#4} }%
1122 \typeout{MRULE~done:~#3}
1123 }
1124
1125 \cs_new_protected:Npn \__numodel_mrule_body:nnn #1 #2 #3
1126 {
1127 \int_gincr:N \g_mrule_counter_int
1128 \tl_set:Ne \l__numodel_fullname_tl { \g_numodel_current_prefix_tl #2 }
1129 % Store execution data (target = full name)
1130 \seq_gput_right:Nx \g_mrule_calc_seq
1131 { { \l__numodel_fullname_tl } { \exp_not:n {#3} } }
1132 % Left-hand side: aliasleft or default display name
1133 \tl_if_blank:VTF \l__numodel_aliasleft_tl
1134 { \tl_set:Ne \l__numodel_lhs_tl { \use:c { \l__numodel_fullname_tl text } } }
1135 { \tl_set_eq:NN \l__numodel_lhs_tl \l__numodel_aliasleft_tl }
1136 % Generate display
1137 \tl_if_blank:VTF \l__numodel_alias_tl
1138 {
1139 \tl_if_blank:VTF \l__numodel_aliasright_tl
1140 {
1141 % Automatic display generation (ternary or normal)
1142 \__numodel_parse_ternary:nN {#3} \l__numodel_display_tl
1143 \bool_if:NTF \l__numodel_is_ternary_bool
1144 {
1145 \IfBooleanTF {#1}
1146 {
1147 % Starred ternary -> multiline IF/THEN/ELSE/ENDIF
1148 \tl_set:Ne \l__numodel_display_tl
1149 {
1150 \__numodel_kwt:n {if}
1151 \exp_not:V \l__numodel_cond_tl
1152 \__numodel_kwt:n {then_n1}
1153 }
1154 \seq_gput_right:NV \g_mrule_seq \l__numodel_display_tl
1155 \seq_gput_right:Nn \g_mrule_type_seq { rule }
1156 \tl_set:Ne \l__numodel_display_tl
1157 {
1158 \exp_not:n { \quad }
1159 \exp_not:V \l__numodel_lhs_tl \exp_not:n { \,=\, , }
1160 \exp_not:V \l__numodel_true_tl

```

```

1161     }
1162     \seq_gput_right:NV \g_mrule_seq \l__numodel_display_tl
1163     \seq_gput_right:Nn \g_mrule_type_seq { cont }
1164     \seq_gput_right:Ne \g_mrule_seq { \__numodel_kwt:n {else_n1} }
1165     \seq_gput_right:Nn \g_mrule_type_seq { cont }
1166     \tl_set:Ne \l__numodel_display_tl
1167     {
1168         \exp_not:n { \quad }
1169         \exp_not:V \l__numodel_lhs_tl \exp_not:n { \,=\, , }
1170         \exp_not:V \l__numodel_false_tl
1171     }
1172     \seq_gput_right:NV \g_mrule_seq \l__numodel_display_tl
1173     \seq_gput_right:Nn \g_mrule_type_seq { cont }
1174     \seq_gput_right:Ne \g_mrule_seq { \__numodel_kwt:n {endif_n1} }
1175     \seq_gput_right:Nn \g_mrule_type_seq { cont }
1176 }
1177 {
1178     % Unstarred ternary -> single-line
1179     \tl_set:Ne \l__numodel_display_tl
1180     {
1181         \__numodel_kwt:n {if}
1182         \exp_not:V \l__numodel_cond_tl
1183         \__numodel_kwt:n {then}
1184         \exp_not:V \l__numodel_lhs_tl \exp_not:n { \,=\, , }
1185         \exp_not:V \l__numodel_true_tl
1186         \__numodel_kwt:n {else}
1187         \exp_not:V \l__numodel_lhs_tl \exp_not:n { \,=\, , }
1188         \exp_not:V \l__numodel_false_tl
1189         \__numodel_kwt:n {endif}
1190     }
1191 }
1192 }
1193 {
1194     % Ordinary assignment: lhs = rhs
1195     \tl_set:Nn \l__numodel_rhs_tl {#3}
1196     \__numodel_vars_to_display:N \l__numodel_rhs_tl
1197     \typeout{MRULE~rhs~after:~\tl_to_str:N \l__numodel_rhs_tl}
1198     \tl_set:Ne \l__numodel_display_tl
1199     {
1200         \exp_not:V \l__numodel_lhs_tl \exp_not:n { \,=\, , }
1201         \exp_not:V \l__numodel_rhs_tl
1202     }
1203     \typeout{MRULE~display:~\tl_to_str:N \l__numodel_display_tl}
1204 }
1205 }
1206 {
1207     % aliasright: lhs = aliasright (no ternary processing)
1208     \tl_set:Ne \l__numodel_display_tl
1209     {
1210         \exp_not:V \l__numodel_lhs_tl \exp_not:n { \,=\, , }
1211         \exp_not:V \l__numodel_aliasright_tl
1212     }
1213 }
1214 }
1215 {
1216     % alias: replace the whole row
1217     \tl_set:Ne \l__numodel_display_tl { \exp_not:V \l__numodel_alias_tl }
1218 }
1219 % For starred ternary everything is already pushed above
1220 \bool_if:nF { \l__numodel_is_ternary_bool && #1 }
1221 {
1222     \seq_gput_right:NV \g_mrule_seq \l__numodel_display_tl
1223     \seq_gput_right:Nn \g_mrule_type_seq { rule }
1224 }
1225 % Pillar A - Lua-side rule registration. Detokenize the raw
1226 % expression so Lua sees the macro names, not the evaluated

```

```

1227 % fp values. Lua handles both the dependency extraction
1228 % (build_deps) and the flow detection (classify_flows) at
1229 % \graphicmodel time.
1230 \tl_set:N \l__numodel_scratch_tl { \detokenize {#3} }
1231 \directlua{ numodel.add_rule(
1232     "g_numodel_current_prefix_tl",
1233     "\l__numodel_fullname_tl",
1234     "\luaescapestring{\l__numodel_scratch_tl}",
1235     "\bool_if:NTF \l__numodel_is_ternary_bool { ternary } { calc }") }
1236 }
1237
1238 % =====
1239 % \mruletext{free text}
1240 % =====
1241 % Adds a free-text row in the model table. Useful for structure
1242 % that does not fit as \mrule, e.g.:
1243 % \mruletext{\text{IF } t < T/4 \text{ THEN}}
1244 % \mruletext{\quad a = a + da}
1245 % \mruletext{\text{ENDIF}}
1246 %
1247 % NOT executed by \computemodel (display only).
1248
1249 \NewDocumentCommand{\mruletext}{0}{ m }{%
1250 \tl_clear:N \l__numodel_cmd_prefix_tl
1251 \keys_set:nn { numodel / cmd } {#1}
1252 \__numodel_resolve_eff_prefix:
1253 \__numodel_with_prefix:Nn \l__numodel_eff_prefix_tl
1254 {
1255 \int_gincr:N \g_mruler_counter_int
1256 \seq_gput_right:Nn \g_mruler_seq {#2}
1257 \seq_gput_right:Nn \g_mruler_type_seq { rule }
1258 }
1259 }
1260
1261 % =====
1262 % \mstop[keys]{condition}
1263 % \mstop*[keys]{condition} (star reserved; currently identical)
1264 % =====
1265 % Declare the model's stop condition.
1266 % Display: "IF condition THEN STOP ENDIF"
1267 % Execution: the model stops when the condition is true (evaluates to 1).
1268 %
1269 % The condition uses \fpeval syntax with \mvar variables:
1270 % \mstop{\modT >= \modTmax}
1271 % \mstop{\modV <= 0}
1272 %
1273 % Supported keys:
1274 % alias={...} -- replaces the whole display row (execution
1275 % remains intact)
1276 % aliasleft/aliasright are not (yet) supported for \mstop.
1277
1278 \tl_new:N \l__numodel_stop_tl
1279 \NewDocumentCommand{\mstop}{s 0}{ m }{%
1280 \typeout{MSTOP-start}
1281 % Parse keys (only alias is honoured; prefix via key resolver)
1282 \tl_clear:N \l__numodel_alias_tl
1283 \tl_clear:N \l__numodel_aliasleft_tl
1284 \tl_clear:N \l__numodel_aliasright_tl
1285 \tl_clear:N \l__numodel_cmd_prefix_tl
1286 \keys_set:nn { numodel / mvar } {#2}
1287 \__numodel_resolve_eff_prefix:
1288 \__numodel_with_prefix:Nn \l__numodel_eff_prefix_tl
1289 { \__numodel_mstop_body:n {#3} }%
1290 \typeout{MSTOP-done}
1291 }
1292

```

```

1293 \cs_new_protected:Npn \__numodel_mstop_body:n #1
1294 {
1295   \int_gincr:N \g_mruler_counter_int
1296   % Store expression for execution (always the actual condition)
1297   \tl_gset:Nn \g_numodel_stop_expr_tl {#1}
1298   % Generate display
1299   \tl_if_blank:VTF \l__numodel_alias_tl
1300   {
1301     \tl_set:Nn \l__numodel_stop_tl {#1}
1302     \__numodel_vars_to_display:N \l__numodel_stop_tl
1303     \tl_set:Ne \l__numodel_display_tl
1304     {
1305       \__numodel_kwt:n {if}
1306       \exp_not:V \l__numodel_stop_tl
1307       \__numodel_kwt:n {then_n}
1308       \__numodel_kwt:n {stop}
1309       \__numodel_kwt:n {endif_n}
1310     }
1311   }
1312   {
1313     \tl_set:Ne \l__numodel_display_tl { \exp_not:V \l__numodel_alias_tl }
1314   }
1315   \seq_gput_right:N \g_mruler_seq \l__numodel_display_tl
1316   \seq_gput_right:Nn \g_mruler_type_seq { rule }
1317 }
1318
1319 % =====
1320 % \textmodel
1321 % =====
1322 % Builds a tabular with numbered model rules on the left and initial
1323 % values on the right. Can be placed anywhere, e.g. in a subfigure
1324 % next to a graphic model.
1325
1326 \tl_new:N \g__numodel_table_tl
1327 \int_new:N \l__numodel_row_int
1328 \int_new:N \l__numodel_dispnum_int
1329 \int_new:N \l__numodel_startrow_int
1330 \int_new:N \l__numodel_numrules_int
1331 \int_new:N \l__numodel_numstarts_int
1332
1333 % Emit one initial-value cell with alias-key support.
1334 % - \<name>alias not empty -> replaces whole cell (inside $...$)
1335 % - \<name>aliasleft not empty -> replaces left symbol, else
1336 % \<name>text
1337 % - \<name>aliasright not empty -> replaces right value, else
1338 % \<name>qty (units=true) or
1339 % \<name>num (units=false)
1340 \cs_new_protected:Npn \__numodel_emit_startcell:n #1
1341 {
1342   \tl_if_blank:eTF { \use:c { #1 alias } }
1343   {
1344     \tl_gput_right:Nn \g__numodel_table_tl { $ }
1345     \tl_if_blank:eTF { \use:c { #1 aliasleft } }
1346     {
1347       \tl_gput_right:Ne \g__numodel_table_tl
1348       { \exp_not:e { \use:c { #1 text } } }
1349     }
1350     {
1351       \tl_gput_right:Ne \g__numodel_table_tl
1352       { \exp_not:e { \use:c { #1 aliasleft } } }
1353     }
1354     \tl_gput_right:Nn \g__numodel_table_tl { = }
1355     \tl_if_blank:eTF { \use:c { #1 aliasright } }
1356     {
1357       % Emit \<name>qty or \<name>num as a token name without
1358       % expanding it; siunitx macros must only expand at

```

```

1359         % \tl_use:N time, in the correct tabular context.
1360         \bool_if:NTF \g__numodel_units_bool
1361         {
1362             \tl_gput_right:Nx \g__numodel_table_tl
1363             { \exp_not:c { #1 qty } }
1364         }
1365         {
1366             \tl_gput_right:Nx \g__numodel_table_tl
1367             { \exp_not:c { #1 num } }
1368         }
1369     }
1370     {
1371         \tl_gput_right:Ne \g__numodel_table_tl
1372         { \exp_not:e { \use:c { #1 aliasright } } }
1373     }
1374     \tl_gput_right:Nn \g__numodel_table_tl { $ }
1375 }
1376 {
1377     \tl_gput_right:Nn \g__numodel_table_tl { $ }
1378     \tl_gput_right:Ne \g__numodel_table_tl
1379     { \exp_not:e { \use:c { #1 alias } } }
1380     \tl_gput_right:Nn \g__numodel_table_tl { $ }
1381 }
1382 }
1383
1384 \cs_new_protected:Npn \__numodel_build_table:
1385 {
1386     \typeout{BUILD_TABLE~rules:~\seq_use:Nn \g_mrule_seq {||}}
1387     \typeout{BUILD_TABLE~types:~\seq_use:Nn \g_mrule_type_seq {||}}
1388     \int_set:Nn \l__numodel_numrules_int { \seq_count:N \g_mrule_seq }
1389     \int_set:Nn \l__numodel_numstarts_int { \seq_count:N \g_mvar_start_seq }
1390     \typeout{BUILD_TABLE~numrules:~\int_use:N \l__numodel_numrules_int}
1391     \typeout{BUILD_TABLE~numstarts:~\int_use:N \l__numodel_numstarts_int}
1392     \int_zero:N \l__numodel_row_int
1393     \int_zero:N \l__numodel_dispnum_int
1394     \int_zero:N \l__numodel_startrow_int
1395     \tl_gset:Nn \g__numodel_table_tl { \begin }
1396     \tl_gput_right:Ne \g__numodel_table_tl
1397     { { \tl_use:N \g__numodel_tblrenv_tl } }
1398     \tl_gput_right:Nn \g__numodel_table_tl
1399     { [theme={numodel}]{colspec={r|l|l}, rowhead=1} & \textbf }
1400     \tl_gput_right:Ne \g__numodel_table_tl
1401     { { \__numodel_kw:n {th_model} } }
1402     \tl_gput_right:Nn \g__numodel_table_tl
1403     { & \textbf }
1404     \tl_gput_right:Ne \g__numodel_table_tl
1405     { { \__numodel_kw:n {th_initvals} } }
1406     \tl_gput_right:Nn \g__numodel_table_tl
1407     { \\\hline }
1408     \typeout{BUILD_TABLE~before-step-table:~\tl_to_str:N \g__numodel_table_tl}
1409     \int_step_inline:nn { \l__numodel_numrules_int }
1410     {
1411         \typeout{BUILD_TABLE~iter-row:~\int_use:N \l__numodel_row_int}
1412         \int_incr:N \l__numodel_row_int
1413         \int_incr:N \l__numodel_startrow_int
1414         \typeout{BUILD_TABLE~row:~\int_use:N \l__numodel_row_int~type:~\seq_item:Nn \g_mrule_type_seq
1415         % Check type: rule -> show number, cont -> blank
1416         \str_if_eq:eeTF
1417         { \seq_item:Nn \g_mrule_type_seq { \l__numodel_row_int } }
1418         { cont }
1419         {
1420             % Continuation rows: no number
1421             \tl_gput_right:Ne \g__numodel_table_tl
1422             {
1423                 \exp_not:n { \rule{0pt}{2.6ex} }
1424                 \exp_not:n { & $ }

```



```

1425         \exp_not:e { \seq_item:Nn \g_mrule_seq { \l__numodel_row_int } }
1426         \exp_not:n { $ & }
1427     }
1428 }
1429 {
1430     % Numbered row
1431     \int_incr:N \l__numodel_dispnum_int
1432     \typeout{BUILD_TABLE~emit~rule~num:~\int_use:N \l__numodel_dispnum_int~content:~\seq_item:Nn \g_mrule_seq { \l__numodel_row_int } }
1433     \tl_gput_right:Ne \g__numodel_table_tl
1434     {
1435         \exp_not:n { \rule{0pt}{2.6ex} }
1436         \int_use:N \l__numodel_dispnum_int
1437         \exp_not:n { & $ }
1438         \exp_not:e { \seq_item:Nn \g_mrule_seq { \l__numodel_row_int } }
1439         \exp_not:n { $ & }
1440     }
1441     \typeout{BUILD_TABLE~after~emit~table~tail:~\tl_tail:N \g__numodel_table_tl}
1442 }
1443 % Initial-value column (independent of rule/cont)
1444 \int_compare:nNnTF { \l__numodel_startrow_int } > { \l__numodel_numstarts_int }
1445 {
1446     \tl_gput_right:Nn \g__numodel_table_tl { \\\ }
1447 }
1448 {
1449     \exp_args:Ne \__numodel_emit_startcell:n
1450     { \seq_item:Nn \g_mvar_start_seq { \l__numodel_startrow_int } }
1451     \tl_gput_right:Nn \g__numodel_table_tl { \\\ }
1452 }
1453 }
1454 \int_while_do:nNnn { \l__numodel_startrow_int } < { \l__numodel_numstarts_int }
1455 {
1456     \int_incr:N \l__numodel_startrow_int
1457     \tl_gput_right:Nn \g__numodel_table_tl { & & }
1458     \exp_args:Ne \__numodel_emit_startcell:n
1459     { \seq_item:Nn \g_mvar_start_seq { \l__numodel_startrow_int } }
1460     \tl_gput_right:Nn \g__numodel_table_tl { \\\ }
1461 }
1462 \tl_gput_right:Nn \g__numodel_table_tl { \end }
1463 \tl_gput_right:Ne \g__numodel_table_tl
1464 { { \tl_use:N \g__numodel_tblrenv_tl } }
1465 }
1466
1467 \NewDocumentCommand{\textmodel}{\textmodel}{0{}}{%
1468     \typeout{TEXTMODEL~start}
1469     \tl_clear:N \l__numodel_cmd_prefix_tl
1470     \tl_clear:N \l__numodel_cmd_units_tl
1471     \tl_clear:N \l__numodel_cmd_tblrenv_tl
1472     \keys_set:nn { numodel / cmd } {#1}
1473     \__numodel_resolve_eff_prefix:
1474     % Temporary units override: save the global value, apply the key
1475     % value before build, restore afterwards. This way
1476     % \textmodel[units=false] flips one render without touching the
1477     % global \numodelsetup state.
1478     \bool_set_eq:NN \l__numodel_saved_units_bool \g__numodel_units_bool
1479     \tl_if_empty:NF \l__numodel_cmd_units_tl
1480     {
1481         \str_if_eq:VnTF \l__numodel_cmd_units_tl { true }
1482         { \bool_gset_true:N \g__numodel_units_bool }
1483         { \bool_gset_false:N \g__numodel_units_bool }
1484     }
1485     % Same dance for tblrenv: save, apply per-call override, restore.
1486     \tl_set_eq:NN \l__numodel_saved_tblrenv_tl \g__numodel_tblrenv_tl
1487     \tl_if_empty:NF \l__numodel_cmd_tblrenv_tl
1488     { \tl_gset_eq:NN \g__numodel_tblrenv_tl \l__numodel_cmd_tblrenv_tl }
1489     \__numodel_with_prefix:Nn \l__numodel_eff_prefix_tl
1490     {

```

```

1491 \group_begin:
1492 \__numodel_apply_dsep:
1493 \__numodel_build_table:
1494 \typeout{TEXTMODEL~table:~\tl_to_str:N \g__numodel_table_tl}
1495 \tl_use:N \g__numodel_table_tl
1496 \group_end:
1497 }%
1498 \bool_gset_eq:NN \g__numodel_units_bool \l__numodel_saved_units_bool
1499 \tl_gset_eq:NN \g__numodel_tblrenv_tl \l__numodel_saved_tblrenv_tl
1500 \typeout{TEXTMODEL~done}
1501 }
1502 \bool_new:N \l__numodel_saved_units_bool
1503 \tl_new:N \l__numodel_saved_tblrenv_tl
1504
1505 % =====
1506 % \computemodel
1507 % =====
1508 % Run the model with the Euler method:
1509 % 1. Record all variable values in Lua
1510 % 2. Check stop condition (\mstop)
1511 % 3. Execute every \mrule rule (in declaration order)
1512 % 4. Repeat until stop or safety limit (default 20000 iterations)
1513 %
1514 % Requires: at least one \mstop and at least one \mrule.
1515 % Performance: ~440 steps in ~4s, ~20000 steps in ~60s.
1516
1517 \cs_new_protected:Npn \__numodel_exec_rule:nn #1 #2
1518 {
1519 \cs_gset:cpe {#1} { \fp_eval:n {#2} }
1520 }
1521
1522 % Record all current variable values in Lua (0(1) per variable).
1523 \cs_new_protected:Npn \__numodel_lua_record_all:
1524 {
1525 \seq_map_inline:Nn \g_mvar_names_seq
1526 {
1527 \directlua{ numodel.record(
1528 "\g_numodel_current_prefix_tl", "##1", \use:c{##1}) }
1529 }
1530 \directlua{ numodel.end_step("\g_numodel_current_prefix_tl") }
1531 }
1532
1533 % Set min/max TeX macros from Lua data after the simulation finishes.
1534 \cs_new_protected:Npn \__numodel_set_minmax_from_lua:
1535 {
1536 \seq_map_inline:Nn \g_mvar_names_seq
1537 {
1538 \cs_gset:cpe { ##1 min }
1539 { \directlua{ numodel.get_min("\g_numodel_current_prefix_tl", "##1") } }
1540 \cs_gset:cpe { ##1 max }
1541 { \directlua{ numodel.get_max("\g_numodel_current_prefix_tl", "##1") } }
1542 }
1543 }
1544
1545
1546 \NewDocumentCommand{\computemodel}{0}{}{}%
1547 \typeout{COMPUTE~start}
1548 \tl_clear:N \l__numodel_cmd_prefix_tl
1549 \keys_set:nn { numodel / cmd } {#1}
1550 \__numodel_resolve_eff_prefix:
1551 \__numodel_with_prefix:Nn \l__numodel_eff_prefix_tl
1552 { \__numodel_compute_body: }%
1553 \typeout{COMPUTE~done}
1554 }
1555
1556 \cs_new_protected:Npn \__numodel_compute_body:

```

```

1557 {
1558   \tl_if_empty:NT \g_numodel_stop_expr_tl
1559   { \msg_error:nn { numodel } { no-stop } }
1560   \int_gzero:N \g_numodel_steps_int
1561   % Initialise Lua storage for this prefix
1562   \directlua{ numodel.init("\g_numodel_current_prefix_tl") }
1563   % Main loop
1564   \bool_gset_false:N \g_tmpa_bool
1565   \bool_do_until:Nn \g_tmpa_bool
1566   {
1567     \__numodel_lua_record_all:
1568     \int_gincr:N \g_numodel_steps_int
1569     \int_compare:nNnT
1570     { \fp_eval:n { \g_numodel_stop_expr_tl } } = { 1 }
1571     { \bool_gset_true:N \g_tmpa_bool }
1572     \bool_if:NF \g_tmpa_bool
1573     {
1574       \seq_map_inline:Nn \g_mrule_calc_seq
1575       { \__numodel_exec_rule:nn ##1 }
1576     }
1577     \int_compare:nNnT
1578     { \g_numodel_steps_int } > { \g_numodel_maxiter_int }
1579     {
1580       \bool_gset_true:N \g_tmpa_bool
1581       \msg_warning:nn { numodel } { maxiter }
1582     }
1583   }
1584   \__numodel_set_minmax_from_lua:
1585 }
1586
1587 % On-demand coordinates from Lua (after \computemodel). Arguments
1588 % are SHORT names; the current prefix is prepended automatically.
1589 % See \mcoordsp for the form with an explicit prefix.
1590 %
1591 % Fully expandable (a plain \def around \directlua, no xparse
1592 % wrapper). Works directly inside \addplot coordinates
1593 % {\mcoords{T}{V}} and pgfplots' math-expression parser, without
1594 % pre-expansion via \edef.
1595 \cs_new:Npn \mcoords #1#2
1596 {
1597   \directlua{ numodel.get_coords(
1598     "\g_numodel_current_prefix_tl",
1599     "\g_numodel_current_prefix_tl" .. "#1",
1600     "\g_numodel_current_prefix_tl" .. "#2") }
1601 }
1602
1603 % Variant with explicit prefix as first argument (instead of an
1604 % optional argument, so the macro stays fully expandable).
1605 \cs_new:Npn \mcoordsp #1#2#3
1606 {
1607   \directlua{ numodel.get_coords(
1608     "#1", "#1" .. "#2", "#1" .. "#3") }
1609 }
1610
1611 % Value of variable #1 at step #2 (0-based). The variable name is
1612 % SHORT (current prefix is prepended automatically). See \mstepp for
1613 % the form with an explicit prefix.
1614 %
1615 % Example -- tangent line at step 0:
1616 % \addplot[domain=0:\modTmax]
1617 %   {\mstep{V}{0} + \mstep{A}{1} * (x - \mstep{T}{0})};
1618 \cs_new:Npn \mstep #1#2
1619 {
1620   \directlua{ numodel.get_step(
1621     "\g_numodel_current_prefix_tl",
1622     "\g_numodel_current_prefix_tl" .. "#1", #2) }

```

```

1623 }
1624
1625 \cs_new:Npn \mstepp #1#2#3
1626 {
1627   \directlua{ numodel.get_step("#1", "#1" .. "#2", #3) }
1628 }
1629
1630 % =====
1631 % \modelreset -- REMOVED
1632 % =====
1633 % \modelreset no longer exists. Use \newmodelprefix{<prefix>} to set
1634 % up a new model and \switchmodelprefix{<prefix>} to switch between
1635 % existing ones. Namespaces are additive -- variables from earlier
1636 % models remain available.
1637
1638 % =====
1639 % \graphicmodel -- Forrester diagram from \mvar/\mrule data
1640 % =====
1641 % Builds a tikzpicture with:
1642 %   - stock, valve, aux and const nodes based on type and gridx/gridy
1643 %   - flow arrows from valve to stock (via \flowarrow)
1644 %   - causal arrows from the dependency graph
1645 %
1646 % Positioning: automatic (auto-layout) or manual via
1647 %   \mvar[gridx=N, gridy=N]{...}. Mixed mode is supported.
1648 % Auto-layout places stocks+valves at gridy=0, aux at gridy=1, and
1649 % constants at gridy=2. Variables of type system are skipped.
1650 % Auxiliary variables that act as inflow are placed as a valve
1651 % (not as aux).
1652
1653 \tl_new:N \g__numodel_graphic_tl
1654 % Lua-populated cache (filled by numodel.tex_writeback at the start of
1655 % each \graphicmodel call, consumed by the emit helpers below).
1656 \prop_new:N \l__numodel_valve_for_prop      % aux/const -> stock (inflow)
1657 \prop_new:N \l__numodel_outvalve_for_prop   % aux/const -> stock (outflow)
1658 \prop_new:N \l__numodel_between_valve_prop  % aux/const -> source_stock (between-flow)
1659 \prop_new:N \l__numodel_between_target_prop % aux/const -> target_stock (between-flow)
1660 \prop_new:N \l__numodel_stock_valve_prop    % stock -> source stock (stock-as-flow)
1661 \prop_new:N \l__numodel_stock_phantom_valve_prop % stock -> source stock (phantom flow)
1662 % For diagram-style=forrester|edu: separate valve gridx position next
1663 % to the stock; the variable's own gridx/gridy keeps the natural
1664 % position at gridy=1 (aux) or gridy=2 (constant). vpos_y is always 0
1665 % so it lives implicitly in the emit helpers.
1666 \prop_new:N \l__numodel_vpos_x_prop          % varname -> valve gridx
1667 \prop_new:N \l__numodel_vpos_y_prop          % varname -> valve gridy (used when gridmaxx wrap shifted)
1668 \tl_new:N \l__numodel_flows_tl               % deferred flow arrows
1669 \tl_new:N \l__numodel_valves_tl              % deferred valve nodes (drawn on top)
1670 \tl_new:N \l__numodel_late_causals_tl        % causals pointing at valves
1671
1672 % --- Resolved render settings (set per-\graphicmodel call) ---
1673 % These hold the effective values (after resolving diagram-style
1674 % defaults and explicit global/local overrides) used by the emit
1675 % helpers and by the public \flowarrow / \flowoutarrow macros.
1676 \tl_new:N \l__numodel_flowarrow_eff_tl      % "hollow" or "filled"
1677 \tl_new:N \l__numodel_valve_eff_tl          % "valve" | "circle" | "edu"
1678 \tl_new:N \l__numodel_flowcloud_global_tl   % "true" or "false" (global default)
1679
1680 % Resolve flowarrow-style: explicit global key wins; otherwise
1681 % diagram-style picks the default (forrester -> hollow,
1682 % tight|edu -> filled).
1683 \cs_new_protected:Npn \__numodel_resolve_flowarrow:
1684 {
1685   \tl_if_empty:NTF \g__numodel_flowarrow_style_tl
1686   {
1687     \str_if_eq:VnTF \g__numodel_diagram_style_tl { forrester }
1688     { \tl_set:Nn \l__numodel_flowarrow_eff_tl { hollow } }

```

```

1689     { \tl_set:Nn \l__numodel_flowarrow_eff_tl { filled } }
1690   }
1691   { \tl_set_eq:NN \l__numodel_flowarrow_eff_tl
1692     \g__numodel_flowarrow_style_tl }
1693 }
1694
1695 % Resolve valve-style: explicit global key wins; otherwise
1696 % diagram-style picks the default (forrester -> valve,
1697 % tight|edu -> edu).
1698 \cs_new_protected:Npn \__numodel_resolve_valve:
1699 {
1700   \tl_if_empty:NTF \g__numodel_valve_style_tl
1701   {
1702     \str_if_eq:VnTF \g__numodel_diagram_style_tl { forrester }
1703     { \tl_set:Nn \l__numodel_valve_eff_tl { valve } }
1704     { \tl_set:Nn \l__numodel_valve_eff_tl { edu } }
1705   }
1706   { \tl_set_eq:NN \l__numodel_valve_eff_tl
1707     \g__numodel_valve_style_tl }
1708 }
1709
1710 % Resolve the global flowarrow-cloud-tip default (no per-stock
1711 % override yet; that's added on top in \__numodel_eff_flowcloud:n).
1712 \cs_new_protected:Npn \__numodel_resolve_flowcloud:
1713 {
1714   \tl_if_empty:NTF \g__numodel_flowcloud_tl
1715   {
1716     \str_if_eq:VnTF \g__numodel_diagram_style_tl { forrester }
1717     { \tl_set:Nn \l__numodel_flowcloud_global_tl { true } }
1718     { \tl_set:Nn \l__numodel_flowcloud_global_tl { false } }
1719   }
1720   { \tl_set_eq:NN \l__numodel_flowcloud_global_tl
1721     \g__numodel_flowcloud_tl }
1722 }
1723
1724 % Per-stock flowcloud lookup. #1 = stock varname. Sets
1725 % \l__numodel_flowcloud_eff_tl to "true" or "false". Order: per-mvar
1726 % override on the stock (via [flowarrow-cloud-tip=...]) wins over the
1727 % global default.
1728 \tl_new:N \l__numodel_flowcloud_eff_tl
1729 \cs_new_protected:Npn \__numodel_eff_flowcloud:n #1
1730 {
1731   \tl_set:Nn \l__numodel_flowcloud_eff_tl { \use:c { #1 flowcloud } }
1732   \tl_if_empty:NT \l__numodel_flowcloud_eff_tl
1733   { \tl_set_eq:NN \l__numodel_flowcloud_eff_tl
1734     \l__numodel_flowcloud_global_tl }
1735 }
1736
1737 % Sets the public \nmflowbody and \nmflowtip macros that the
1738 % \flowarrow / \flowoutarrow / \flowbetweenarrow macros expand to.
1739 % Reads the resolved flowarrow-style.
1740 \cs_new_protected:Npn \__numodel_apply_flowarrow_style:
1741 {
1742   \str_if_eq:VnTF \l__numodel_flowarrow_eff_tl { hollow }
1743   {
1744     \cs_gset:Npn \nmflowbody { flowpipe-hollow }
1745     \cs_gset:Npn \nmflowtip { flowpipe-hollow-tip }
1746     \cs_gset:Npn \nmflowtipcloudin { flowpipe-hollow-cloudin }
1747     \cs_gset:Npn \nmflowtipcloudout { flowpipe-hollow-cloudout }
1748   }
1749   {
1750     \cs_gset:Npn \nmflowbody { flowpipe-filled }
1751     \cs_gset:Npn \nmflowtip { flowpipe-filled-tip }
1752     \cs_gset:Npn \nmflowtipcloudin { flowpipe-filled-cloudin }
1753     \cs_gset:Npn \nmflowtipcloudout { flowpipe-filled-cloudout }
1754   }

```

```

1755 }
1756
1757 % Sets the tikz node-style name for the valve and the boolean that
1758 % controls whether the valve carries the variable's display label.
1759 % valve-style=valve -> [valve-forrester], no label
1760 % valve-style=circle -> [valve-circle], no label
1761 % valve-style=edu -> [valve-edu], with label
1762 \tl_new:N \l__numodel_valve_node_tl
1763 \bool_new:N \l__numodel_valve_label_bool
1764 \cs_new_protected:Npn \__numodel_apply_valve_style:
1765 {
1766     \str_case:VnF \l__numodel_valve_eff_tl
1767     {
1768         { valve }
1769         {
1770             \tl_set:Nn \l__numodel_valve_node_tl { valve-forrester }
1771             \bool_set_false:N \l__numodel_valve_label_bool
1772         }
1773         { circle }
1774         {
1775             \tl_set:Nn \l__numodel_valve_node_tl { valve-circle }
1776             \bool_set_false:N \l__numodel_valve_label_bool
1777         }
1778         { edu }
1779         {
1780             \tl_set:Nn \l__numodel_valve_node_tl { valve-edu }
1781             \bool_set_true:N \l__numodel_valve_label_bool
1782         }
1783     }
1784     {
1785         \tl_set:Nn \l__numodel_valve_node_tl { valve-edu }
1786         \bool_set_true:N \l__numodel_valve_label_bool
1787     }
1788 }
1789
1790 % --- Diagram-style mode flag ---
1791 % True at diagram-style=forrester|edu; consulted by \__numodel_place_node
1792 % to decide between single-emit and natural+phantom-valve double-emit.
1793 % Set in \__numodel_build_graphic from \g__numodel_diagram_style_tl.
1794 \bool_new:N \l__numodel_keep_natural_bool
1795
1796
1797 \cs_new_protected:Npn \__numodel_build_graphic:
1798 {
1799     \typeout{BUILD:~step-0~reset~auto-positions}
1800     % --- Reset gridx/gridy to the original user input ---
1801     % Auto-layout writes positions to var.gridx/gridy. On a second
1802     % \graphicmodel call those would, without reset, be treated as
1803     % "manually placed". The initial values are saved in
1804     % gridxinit/gridyinit by \mvar -- copy them back.
1805     \seq_map_inline:Nn \g_mvar_names_seq
1806     {
1807         \cs_gset:cpe { ##1 gridx } { \use:c { ##1 gridxinit } }
1808         \cs_gset:cpe { ##1 gridy } { \use:c { ##1 gridyinit } }
1809     }
1810     % Diagram-style mode: forrester|edu keep aux/const at their natural
1811     % gridy with a phantom valve next to the stock; tight collapses the
1812     % aux/const onto the valve position. Consumed by \__numodel_place_node
1813     % to dispatch between single-emit and natural+phantom emit.
1814     \bool_set_false:N \l__numodel_keep_natural_bool
1815     \str_if_eq:VnT \g__numodel_diagram_style_tl { forrester }
1816     { \bool_set_true:N \l__numodel_keep_natural_bool }
1817     \str_if_eq:VnT \g__numodel_diagram_style_tl { edu }
1818     { \bool_set_true:N \l__numodel_keep_natural_bool }
1819     \typeout{BUILD:~step-1~lua~layout~writeback}
1820     % Pillar A - Lua is the single source of truth for flow detection

```

```

1821 % and auto-layout. The writeback clears and fills \<var>gridx/gridy
1822 % and the flow-props that downstream emitters (place_node,
1823 % flow-builders, emit_natural_and_phantom, emit_stock_valve) read.
1824 \__numodel_lua_layout_writeback:
1825 \typeout{BUILD:~step-2~tikzpicture}
1826 % --- Build tikzpicture ---
1827 % Render order matters: each segment overlays the previous one,
1828 % so we draw flows first (one continuous arrow per flow), then
1829 % the valve nodes on top (with white fill so they cover the
1830 % section of the arrow that lies underneath), and finally the
1831 % causal arrows that point at those valves.
1832 \tl_gclear:N \g__numodel_graphic_tl
1833 \tl_clear:N \l__numodel_flows_tl
1834 \tl_clear:N \l__numodel_valves_tl
1835 \tl_clear:N \l__numodel_late_causals_tl
1836 \tl_gput_right:Nn \g__numodel_graphic_tl
1837 { \begin{tikzpicture}[gridscale] }
1838 % Nodes (stocks, aux, const, clouds; flows -> flows_tl,
1839 % valves -> valves_tl)
1840 \seq_map_inline:Nn \g_mvar_names_seq
1841 { \typeout{NODE:~##1} \__numodel_place_node:n {##1} }
1842 \typeout{BUILD:~step-5~stock-valve~nodes}
1843 % Stock-as-flow valve nodes
1844 \prop_map_inline:Nn \l__numodel_stock_valve_prop
1845 {
1846 % ##1 = stock (e.g. modH), ##2 = source stock (e.g. modV)
1847 % Skip entries that are not real stock-valves (e.g. __svgx keys)
1848 \tl_if_in:nnF {##1} { __sv }
1849 { \__numodel_emit_stock_valve:nn {##1} {##2} }
1850 }
1851 % Stock-as-rate phantom-valve nodes (source stock has no matching
1852 % outflow term, so the inflow gets a cloud at the open end)
1853 \prop_map_inline:Nn \l__numodel_stock_phantom_valve_prop
1854 {
1855 \tl_if_in:nnF {##1} { __sv }
1856 { \__numodel_emit_stock_phantom_valve:nn {##1} {##2} }
1857 }
1858 \typeout{BUILD:~step-6~flow~arrows}
1859 % Flow arrows (one segment each; drawn underneath the valves)
1860 \tl_gput_right:Nv \g__numodel_graphic_tl \l__numodel_flows_tl
1861 \typeout{BUILD:~step-6a~valve~nodes}
1862 % Valve nodes (white-filled, drawn ON TOP of the flow arrows)
1863 \tl_gput_right:Nv \g__numodel_graphic_tl \l__numodel_valves_tl
1864 \typeout{BUILD:~step-6b+7~causal~arrows~(Lua)}
1865 % A2: causal arrows via Lua - emit_causals demultiplexes on
1866 % tgt_is_valve and pushes to \g__numodel_graphic_tl or
1867 % \l__numodel_late_causals_tl respectively. Afterwards append
1868 % late-causals to graphic_tl once: that covers both the pushes
1869 % from the flow-builders (step 6/6a) and those from emit_causals.
1870 \__numodel_lua_emit_causals:
1871 \tl_gput_right:Nv \g__numodel_graphic_tl \l__numodel_late_causals_tl
1872 \typeout{BUILD:~step-8~done}
1873 \tl_gput_right:Nn \g__numodel_graphic_tl
1874 { \end{tikzpicture} }
1875 }
1876
1877 % --- Node placement ---
1878 \cs_new_protected:Npn \__numodel_place_node:n #1
1879 {
1880 \bool_set_true:N \l_tmpa_bool
1881 \tl_set:Nx \l__numodel_scratch_tl { \use:c { #1 type } }
1882 \exp_args:Nv \str_if_eq:nnT \l__numodel_scratch_tl { system }
1883 { \bool_set_false:N \l_tmpa_bool }
1884 \bool_if:NT \l_tmpa_bool
1885 {
1886 \int_compare:nNnT { \use:c { #1 gridx } } = { -1 }

```

```

1887     { \bool_set_false:N \l_tmpa_bool }
1888   }
1889   \bool_if:NT \l_tmpa_bool
1890   {
1891     \tl_set:Nx \l__numodel_tmp_tl { \use:c { #1 text } }
1892     % With forrester|edu, valve vars get a dual emission:
1893     % the natural aux/const node + a phantom valve next to the stock.
1894     \bool_set_false:N \l_tmpb_bool % is this a valve var?
1895     \prop_if_in:NnT \l__numodel_valve_for_prop {#1}
1896     { \bool_set_true:N \l_tmpb_bool }
1897     \prop_if_in:NnT \l__numodel_outvalve_for_prop {#1}
1898     { \bool_set_true:N \l_tmpb_bool }
1899     \prop_if_in:NnT \l__numodel_between_valve_prop {#1}
1900     { \bool_set_true:N \l_tmpb_bool }
1901     \bool_lazy_and:nnTF
1902     { \l__numodel_keep_natural_bool } { \l_tmpb_bool }
1903     { \__numodel_emit_natural_and_phantom:n {#1} }
1904     {
1905       \prop_if_in:NnTF \l__numodel_valve_for_prop {#1}
1906       { \__numodel_emit_valve:n {#1} }
1907       {
1908         \prop_if_in:NnTF \l__numodel_outvalve_for_prop {#1}
1909         { \__numodel_emit_outvalve:n {#1} }
1910         {
1911           \prop_if_in:NnTF \l__numodel_between_valve_prop {#1}
1912           { \__numodel_emit_between_valve:n {#1} }
1913           {
1914             \tl_set:Nx \l__numodel_scratch_tl { \use:c { #1 type } }
1915             \exp_args:NV \str_if_eq:nnT \l__numodel_scratch_tl { stock }
1916             { \__numodel_emit_stock:n {#1} }
1917             \exp_args:NV \str_if_eq:nnT \l__numodel_scratch_tl { aux }
1918             { \__numodel_emit_aux:n {#1} }
1919             \exp_args:NV \str_if_eq:nnT \l__numodel_scratch_tl { constant }
1920             { \__numodel_emit_const:n {#1} }
1921           }
1922         }
1923       }
1924     }
1925   }
1926 }
1927
1928 % --- Helper: emit a coordinate node at the valve position ---
1929 % The flow arrow starts/ends at this coordinate (or at an offset
1930 % relative to it, for the open-end without cloud). The visible
1931 % white-filled valve is added later (valves_tl) at the same
1932 % coordinates so it overlays the arrow.
1933 % #1 = coord id #2 = x #3 = y
1934 \cs_new_protected:Npn \__numodel_emit_valve_coord:nnn #1 #2 #3
1935 {
1936   \tl_gput_right:Nx \g__numodel_graphic_tl
1937   {
1938     \exp_not:N \coordinate ~
1939     (#1) ~ \exp_not:n{at} ~ (#2 , ~ #3) ;
1940   }
1941 }
1942
1943 % --- Helper: append a valve node to the deferred valves_tl ---
1944 % Renders \node[<valve-style>] (id) at (x,y) {<label>}; later in the
1945 % picture so the white fill overlays the flow arrow underneath.
1946 % #1 = node id #2 = x coord #3 = y coord #4 = label tl name
1947 \cs_new_protected:Npn \__numodel_defer_valve:nnnn #1 #2 #3 #4
1948 {
1949   \tl_put_right:Nx \l__numodel_valves_tl
1950   {
1951     \exp_not:N \node ~ [ \tl_use:N \l__numodel_valve_node_tl ] ~
1952     (#1) ~ \exp_not:n{at} ~ (#2 , ~ #3) ~

```



```

1953     {
1954         \bool_if:NTF \l__numodel_valve_label_bool
1955         { \exp_not:N $ \exp_not:V #4 \exp_not:N $ }
1956         { }
1957     } ;
1958 }
1959 }
1960
1961 % --- Forrester/edu-mode emission: natural node + phantom valve ---
1962 % The variable sits as an ordinary aux/const node at its own gridy.
1963 % The phantom valve (id #1__v) sits next to the stock at gridy=0; its
1964 % appearance follows the resolved valve-style (label included only
1965 % when valve-style=edu). A causal arrow from #1 -> #1__v is drawn
1966 % (deferred to late_causals_tl so it lands on top of the white-
1967 % filled valve).
1968 \tl_new:N \l__numodel_phantom_label_tl
1969 \cs_new_protected:Npn \__numodel_emit_natural_and_phantom:n #1
1970 {
1971     % --- 1. Natural node (aux or constant) ---
1972     \tl_set:Nc \l__numodel_scratch_tl { \use:c { #1 type } }
1973     \exp_args:NV \str_if_eq:nnT \l__numodel_scratch_tl { aux }
1974     { \__numodel_emit_aux:n {#1} }
1975     \exp_args:NV \str_if_eq:nnT \l__numodel_scratch_tl { constant }
1976     { \__numodel_emit_const:n {#1} }
1977     % --- 2. Locate the phantom valve x/y position ---
1978     \prop_get:NnNT \l__numodel_vpos_x_prop {#1} \l__numodel_phantom_x_tl
1979     {
1980         \prop_get:NnNF \l__numodel_vpos_y_prop {#1} \l__numodel_scratch_y_tl
1981         { \tl_set:Nc \l__numodel_scratch_y_tl { 0 } }
1982         \tl_set:Nc \l__numodel_phantom_label_tl { \use:c { #1 text } }
1983         % --- 3. Coordinate at the phantom valve position ---
1984         \__numodel_emit_valve_coord:nnn { #1 __vc }
1985         { \tl_use:N \l__numodel_phantom_x_tl }
1986         { \tl_use:N \l__numodel_scratch_y_tl }
1987         % --- 4. Flow arrow + (optional) cloud, one segment ---
1988         \prop_if_in:NnTF \l__numodel_valve_for_prop {#1}
1989         {
1990             \tl_set:Nc \l__numodel_tmp_tl
1991             { \prop_item:Nn \l__numodel_valve_for_prop {#1} }
1992             \__numodel_eff_flowcloud:V \l__numodel_tmp_tl
1993             \str_if_eq:VnTF \l__numodel_flowcloud_eff_tl { true }
1994             {
1995                 \tl_put_right:Nc \l__numodel_flows_tl
1996                 {
1997                     \exp_not:N \flowarrow [#1 __cl] {#1 __vc}
1998                     { \tl_use:N \l__numodel_tmp_tl }
1999                 }
2000             }
2001             {
2002                 \tl_put_right:Nc \l__numodel_flows_tl
2003                 {
2004                     \exp_not:N \flowarrow {#1 __vc}
2005                     { \tl_use:N \l__numodel_tmp_tl }
2006                 }
2007             }
2008         }
2009         {
2010             \prop_if_in:NnTF \l__numodel_outvalve_for_prop {#1}
2011             {
2012                 \tl_set:Nc \l__numodel_tmp_tl
2013                 { \prop_item:Nn \l__numodel_outvalve_for_prop {#1} }
2014                 \__numodel_eff_flowcloud:V \l__numodel_tmp_tl
2015                 \str_if_eq:VnTF \l__numodel_flowcloud_eff_tl { true }
2016                 {
2017                     \tl_put_right:Nc \l__numodel_flows_tl
2018                     {

```

```

2019             \exp_not:N \flowoutarrow [#1 __cl]
2020             { \tl_use:N \l__numodel_tmp_tl } {#1 __vc}
2021         }
2022     }
2023     {
2024         \tl_put_right:Ne \l__numodel_flows_tl
2025         {
2026             \exp_not:N \flowoutarrow
2027             { \tl_use:N \l__numodel_tmp_tl } {#1 __vc}
2028         }
2029     }
2030 }
2031 {
2032     % between (no cloud: both ends are stocks)
2033     \tl_set:Ne \l__numodel_tmp_tl
2034     { \prop_item:Nn \l__numodel_between_valve_prop {#1} }
2035     \tl_set:Ne \l__numodel_scratch_tl
2036     { \prop_item:Nn \l__numodel_between_target_prop {#1} }
2037     \tl_put_right:Ne \l__numodel_flows_tl
2038     {
2039         \exp_not:N \flowbetweenarrow
2040         { \tl_use:N \l__numodel_tmp_tl }
2041         {#1 __vc}
2042         { \tl_use:N \l__numodel_scratch_tl }
2043     }
2044 }
2045 }
2046 % --- 5. Defer the phantom valve drawing (overlays flow) ---
2047 \__numodel_defer_valve:nnnn { #1 __v }
2048 { \tl_use:N \l__numodel_phantom_x_tl }
2049 { \tl_use:N \l__numodel_scratch_y_tl }
2050 \l__numodel_phantom_label_tl
2051 % --- 6. Causal arrow natural -> phantom valve (deferred) ---
2052 \tl_put_right:Ne \l__numodel_late_causals_tl
2053 {
2054     \exp_not:N \draw ~ [\exp_not:n{causal}] ~
2055     (#1) ~ \exp_not:n{to} ~ (#1 __v) ;
2056 }
2057 }
2058 }
2059 \tl_new:N \l__numodel_phantom_x_tl
2060 \cs_generate_variant:Nn \__numodel_eff_flowcloud:n { V }
2061
2062 \cs_new_protected:Npn \__numodel_emit_stock:n #1
2063 {
2064     \tl_gput_right:Ne \g__numodel_graphic_tl
2065     {
2066         \exp_not:N \node ~ [\exp_not:n{stock}] ~
2067         (#1) ~ \exp_not:n{at} ~
2068         ( \use:c{#1 gridx} , ~ \use:c{#1 gridy} ) ~
2069         { \exp_not:N $ \exp_not:V \l__numodel_tmp_tl \exp_not:N $ } ;
2070     }
2071 }
2072
2073 % Tight-mode helpers: the valve sits at the variable's own gridx/
2074 % gridy. We emit a coordinate node (#1__vc) early so the flow
2075 % arrow has a positioning anchor, and defer the visible valve node
2076 % (#1) to valves_tl so it overlays the arrow with white fill.
2077 \cs_new_protected:Npn \__numodel_emit_tight_valve_setup:n #1
2078 {
2079     \__numodel_emit_valve_coord:nnn { #1 __vc }
2080     { \use:c{#1 gridx} } { \use:c{#1 gridy} }
2081     \__numodel_defer_valve:nnnn {#1}
2082     { \use:c{#1 gridx} } { \use:c{#1 gridy} }
2083     \l__numodel_tmp_tl
2084 }

```

```

2085
2086 \cs_new_protected:Npn \__numodel_emit_valve:n #1
2087 {
2088   \__numodel_emit_tight_valve_setup:n {#1}
2089   \tl_set:Nx \l__numodel_scratch_tl
2090     { \prop_item:Nn \l__numodel_valve_for_prop {#1} }
2091   \__numodel_eff_flowcloud:V \l__numodel_scratch_tl
2092   \str_if_eq:VnTF \l__numodel_flowcloud_eff_tl { true }
2093     {
2094       \tl_put_right:Nx \l__numodel_flows_tl
2095         {
2096           \exp_not:N \flowarrow [#1 __cl] {#1 __vc}
2097           { \tl_use:N \l__numodel_scratch_tl }
2098         }
2099     }
2100     {
2101       \tl_put_right:Nx \l__numodel_flows_tl
2102         {
2103           \exp_not:N \flowarrow {#1 __vc}
2104           { \tl_use:N \l__numodel_scratch_tl }
2105         }
2106     }
2107 }
2108
2109 \cs_new_protected:Npn \__numodel_emit_outvalve:n #1
2110 {
2111   \__numodel_emit_tight_valve_setup:n {#1}
2112   \tl_set:Nx \l__numodel_scratch_tl
2113     { \prop_item:Nn \l__numodel_outvalve_for_prop {#1} }
2114   \__numodel_eff_flowcloud:V \l__numodel_scratch_tl
2115   \str_if_eq:VnTF \l__numodel_flowcloud_eff_tl { true }
2116     {
2117       \tl_put_right:Nx \l__numodel_flows_tl
2118         {
2119           \exp_not:N \flowoutarrow [#1 __cl]
2120           { \tl_use:N \l__numodel_scratch_tl } {#1 __vc}
2121         }
2122     }
2123     {
2124       \tl_put_right:Nx \l__numodel_flows_tl
2125         {
2126           \exp_not:N \flowoutarrow
2127           { \tl_use:N \l__numodel_scratch_tl } {#1 __vc}
2128         }
2129     }
2130 }
2131
2132 \cs_new_protected:Npn \__numodel_emit_between_valve:n #1
2133 {
2134   \__numodel_emit_tight_valve_setup:n {#1}
2135   \tl_set:Nx \l__numodel_scratch_tl
2136     { \prop_item:Nn \l__numodel_between_valve_prop {#1} }
2137   \tl_set:Nx \l__numodel_tmp_tl
2138     { \prop_item:Nn \l__numodel_between_target_prop {#1} }
2139   \tl_put_right:Nx \l__numodel_flows_tl
2140     {
2141       \exp_not:N \flowbetweenarrow
2142       { \tl_use:N \l__numodel_scratch_tl }
2143       {#1 __vc}
2144       { \tl_use:N \l__numodel_tmp_tl }
2145     }
2146 }
2147
2148 \cs_new_protected:Npn \__numodel_emit_aux:n #1
2149 {
2150   \tl_gput_right:Nx \g__numodel_graphic_tl

```

```

2151     {
2152         \exp_not:N \node ~ [\exp_not:n{aux}] ~
2153         (#1) ~ \exp_not:n{at} ~
2154         ( \use:c{#1 gridx} , ~ \use:c{#1 gridy} ) ~
2155         { \exp_not:N $ \exp_not:V \l__numodel_tmp_tl \exp_not:N $ } ;
2156     }
2157 }
2158
2159 \cs_new_protected:Npn \__numodel_emit_const:n #1
2160 {
2161     \tl_gput_right:Ne \g__numodel_graphic_tl
2162     {
2163         \exp_not:N \constnode
2164         {#1}
2165         { \use:c{#1 gridx} , ~ \use:c{#1 gridy} }
2166         { \exp_not:N $ \exp_not:V \l__numodel_tmp_tl \exp_not:N $ }
2167     }
2168 }
2169
2170 % --- Stock-as-flow valve emission ---
2171 \cs_new_protected:Npn \__numodel_emit_stock_valve:nn #1 #2
2172 {
2173     % #1 = target stock (e.g. modH), #2 = source stock (e.g. modV)
2174     % Pull gridx from the saved property
2175     \prop_get:NnNTF \l__numodel_stock_valve_prop { #1 __svgx }
2176     \l__numodel_scratch_tl
2177     {
2178         % Look up the wrap-shifted y-slot; default to 0 (the row that
2179         % the auto layout used before any gridmaxx wrap was applied).
2180         \prop_get:NnNF \l__numodel_stock_valve_prop { #1 __svgy }
2181         \l__numodel_scratch_y_tl
2182         { \tl_set:Nn \l__numodel_scratch_y_tl { 0 } }
2183         % Coordinate at the valve position (used by the flow arrow)
2184         \__numodel_emit_valve_coord:nnn { #1 __svc }
2185         { \tl_use:N \l__numodel_scratch_tl }
2186         { \tl_use:N \l__numodel_scratch_y_tl }
2187         % Flow arrow from source-stock through valve into target-stock
2188         % (no cloud: both ends are stocks). The arrow runs in one
2189         % continuous segment; the visible valve will be drawn ON TOP
2190         % later via valves_tl.
2191         \tl_put_right:Ne \l__numodel_flows_tl
2192         {
2193             \exp_not:N \flowbetweenarrow {#2} {#1 __svc} {#1}
2194         }
2195         % Defer the visible valve node (white fill overlays arrow)
2196         \tl_set:Ne \l__numodel_tmp_tl { \use:c { #2 text } }
2197         \__numodel_defer_valve:nnnn { #1 __sv }
2198         { \tl_use:N \l__numodel_scratch_tl }
2199         { \tl_use:N \l__numodel_scratch_y_tl }
2200         \l__numodel_tmp_tl
2201         % Causal arrow source-stock -> valve. Bend it (curved) only
2202         % when the source stock and the valve sit on the same row -
2203         % then the straight causal would coincide with the flow pipe
2204         % and disappear behind it. When they are on different rows
2205         % (e.g. after a gridmaxx wrap) the causal is vertical-ish and
2206         % stays visible without a bend. Deferred so it lands on top
2207         % of the white-filled valve.
2208         \str_if_eq:eeTF { \tl_use:N \l__numodel_scratch_y_tl }
2209             { \use:c { #2 gridy } }
2210         {
2211             \tl_put_right:Ne \l__numodel_late_causals_tl
2212             {
2213                 \exp_not:N \draw ~ [\exp_not:n{causal}] ~
2214                 (#2) ~ \exp_not:n{to} ~ [\exp_not:n{bend~left=30}] ~ (#1__sv) ;
2215             }
2216         }

```

```

2217     {
2218         \tl_put_right:Ne \l__numodel_late_causals_tl
2219         {
2220             \exp_not:N \draw ~ [\exp_not:n{causal}] ~
2221             (#2) ~ \exp_not:n{to} ~ (#1__sv) ;
2222         }
2223     }
2224 }
2225 { } % no gridx found: skip
2226 }
2227
2228 % --- Stock phantom-valve emission (source stock as rate factor) ---
2229 % Source stock acts as a rate factor for the target stock without a
2230 % matching outflow term; render with a cloud-fed inflow valve next to
2231 % the target stock and link the source stock via a causal arrow.
2232 \cs_new_protected:Npn \__numodel_emit_stock_phantom_valve:nn #1 #2
2233 {
2234     % #1 = target stock (e.g. ballY), #2 = source stock (e.g. ballV)
2235     \prop_get:NnNTF \l__numodel_stock_phantom_valve_prop { #1 __svgx }
2236     \l__numodel_scratch_tl
2237     {
2238         % Look up the wrap-shifted y-slot; default to 0.
2239         \prop_get:NnNF \l__numodel_stock_phantom_valve_prop { #1 __svgy }
2240         \l__numodel_scratch_y_tl
2241         { \tl_set:Nn \l__numodel_scratch_y_tl { 0 } }
2242         % Coordinate at the valve position
2243         \__numodel_emit_valve_coord:nnn { #1 __svc }
2244         { \tl_use:N \l__numodel_scratch_tl }
2245         { \tl_use:N \l__numodel_scratch_y_tl }
2246         % Flow arrow: cloud (open end) -> valve -> target stock
2247         \__numodel_eff_flowcloud:n {#1}
2248         \str_if_eq:VnTF \l__numodel_flowcloud_eff_tl { true }
2249         {
2250             \tl_put_right:Ne \l__numodel_flows_tl
2251             {
2252                 \exp_not:N \flowarrow [#1 __svcl] {#1 __svc} {#1}
2253             }
2254         }
2255         {
2256             \tl_put_right:Ne \l__numodel_flows_tl
2257             {
2258                 \exp_not:N \flowarrow {#1 __svc} {#1}
2259             }
2260         }
2261         % Defer the visible valve node, labelled with the source stock
2262         \tl_set:Ne \l__numodel_tmp_tl { \use:c { #2 text } }
2263         \__numodel_defer_valve:nnnn { #1 __sv }
2264         { \tl_use:N \l__numodel_scratch_tl }
2265         { \tl_use:N \l__numodel_scratch_y_tl }
2266         \l__numodel_tmp_tl
2267         % Causal arrow source-stock -> valve. Bend (curved) only when
2268         % source and valve sit on the same row - see the equivalent
2269         % comment in \__numodel_emit_stock_valve:nn for the rationale.
2270         \str_if_eq:eeTF { \tl_use:N \l__numodel_scratch_y_tl }
2271         { \use:c { #2 gridy } }
2272         {
2273             \tl_put_right:Ne \l__numodel_late_causals_tl
2274             {
2275                 \exp_not:N \draw ~ [\exp_not:n{causal}] ~
2276                 (#2) ~ \exp_not:n{to} ~ [\exp_not:n{bend~left=30}] ~ (#1__sv) ;
2277             }
2278         }
2279         {
2280             \tl_put_right:Ne \l__numodel_late_causals_tl
2281             {
2282                 \exp_not:N \draw ~ [\exp_not:n{causal}] ~

```

```

2283         (#2) ~ \exp_not:n{to} ~ (#1__sv) ;
2284     }
2285 }
2286 }
2287 { } % no gridx found: skip
2288 }
2289
2290 % -----
2291 % Pillar A - Lua-side layout writeback. A single directlua call
2292 % that runs compute_layout (build_deps, classify_flows,
2293 % populate_occupied, auto_layout, causals) and writes the resulting
2294 % state back to the TeX-side props (\<var>gridx/gridy and the
2295 % flow-props) that the downstream emitters (place_node,
2296 % flow-builders, emit_natural_and_phantom, emit_stock_valve) read.
2297 % Lua is the single source of truth for the decision logic; TeX
2298 % only renders.
2299 % -----
2300 \cs_new_protected:Npn \__numodel_lua_layout_writeback:
2301 {
2302     % 1. Clear the props that Lua refills, so repeated \graphicmodel
2303     % calls don't produce duplicate entries.
2304     \prop_clear:N \l__numodel_valve_for_prop
2305     \prop_clear:N \l__numodel_outvalve_for_prop
2306     \prop_clear:N \l__numodel_between_valve_prop
2307     \prop_clear:N \l__numodel_between_target_prop
2308     \prop_clear:N \l__numodel_stock_valve_prop
2309     \prop_clear:N \l__numodel_stock_phantom_valve_prop
2310     \prop_clear:N \l__numodel_vpos_x_prop
2311     \prop_clear:N \l__numodel_vpos_y_prop
2312     % 2. Run the Lua pipeline.
2313     \directlua{ numodel.compute_layout(
2314         "\g_numodel_current_prefix_tl",
2315         "\g__numodel_diagram_style_tl",
2316         \int_use:N \g_numodel_gridmaxx_int) }
2317     % 3. Write gridx/gridy and all flow-props back.
2318     % tex_writeback emits a list of expl3 statements via tex.print.
2319     \directlua{ numodel.tex_writeback(
2320         "\g_numodel_current_prefix_tl") }
2321 }
2322
2323 \cs_new_protected:Npn \__numodel_lua_emit_causals:
2324 {
2325     \directlua{ numodel.emit_causals(
2326         "\g_numodel_current_prefix_tl") }
2327 }
2328
2329 \NewDocumentCommand{\graphicmodel}{0} {}{%
2330     \tl_clear:N \l__numodel_cmd_prefix_tl
2331     \tl_clear:N \l__numodel_cmd_diagstyle_tl
2332     \tl_clear:N \l__numodel_cmd_flowarrow_tl
2333     \tl_clear:N \l__numodel_cmd_valve_tl
2334     \tl_clear:N \l__numodel_cmd_flowcloud_tl
2335     \keys_set:nn { numodel / cmd } {#1}
2336     \__numodel_resolve_eff_prefix:
2337     % Temporary overrides: save the global state, apply the per-call
2338     % key values before build, restore afterwards. This way
2339     % \graphicmodel[diagram-style=forrester] flips one render without
2340     % touching the global \numodelsetup state.
2341     \tl_set_eq:NN \l__numodel_saved_diagstyle_tl \g__numodel_diagram_style_tl
2342     \tl_set_eq:NN \l__numodel_saved_flowarrow_tl \g__numodel_flowarrow_style_tl
2343     \tl_set_eq:NN \l__numodel_saved_valve_tl \g__numodel_valve_style_tl
2344     \tl_set_eq:NN \l__numodel_saved_flowcloud_tl \g__numodel_flowcloud_tl
2345     \tl_if_empty:NF \l__numodel_cmd_diagstyle_tl
2346     { \tl_gset_eq:NN \g__numodel_diagram_style_tl \l__numodel_cmd_diagstyle_tl }
2347     \tl_if_empty:NF \l__numodel_cmd_flowarrow_tl
2348     { \tl_gset_eq:NN \g__numodel_flowarrow_style_tl \l__numodel_cmd_flowarrow_tl }

```

```

2349 \tl_if_empty:NF \l__numodel_cmd_valve_tl
2350 { \tl_gset_eq:NN \g__numodel_valve_style_tl \l__numodel_cmd_valve_tl }
2351 \tl_if_empty:NF \l__numodel_cmd_flowcloud_tl
2352 { \tl_gset_eq:NN \g__numodel_flowcloud_tl \l__numodel_cmd_flowcloud_tl }
2353 % Resolve the effective values for this render and propagate them
2354 % to \nmflowbody / \nmflowtip used by the flow macros.
2355 \__numodel_resolve_flowarrow:
2356 \__numodel_resolve_valve:
2357 \__numodel_resolve_flowcloud:
2358 \__numodel_apply_flowarrow_style:
2359 \__numodel_apply_valve_style:
2360 \__numodel_with_prefix:Nn \l__numodel_eff_prefix_tl
2361 {
2362   \__numodel_build_graphic:
2363   \tl_use:N \g__numodel_graphic_tl
2364 }%
2365 \tl_gset_eq:NN \g__numodel_diagram_style_tl \l__numodel_saved_diagstyle_tl
2366 \tl_gset_eq:NN \g__numodel_flowarrow_style_tl \l__numodel_saved_flowarrow_tl
2367 \tl_gset_eq:NN \g__numodel_valve_style_tl \l__numodel_saved_valve_tl
2368 \tl_gset_eq:NN \g__numodel_flowcloud_tl \l__numodel_saved_flowcloud_tl
2369 }
2370 \tl_new:N \l__numodel_saved_diagstyle_tl
2371 \tl_new:N \l__numodel_saved_flowarrow_tl
2372 \tl_new:N \l__numodel_saved_valve_tl
2373 \tl_new:N \l__numodel_saved_flowcloud_tl
2374
2375 % =====
2376 % Plot machinery
2377 % =====
2378 % \diagrammodel calls the pgfplots/calcplothdims infrastructure
2379 % through internal wrappers \__numodel_calc_plot_dims: and
2380 % \__numodel_draw_plot:n. Those wrappers forward to \calcplothdims
2381 % and \drawplot from the sibling package numodel-plot.
2382 %
2383 % The wrappers are not \cs_new_eq:NN because numodel-plot can load
2384 % after this package; at load time \calcplothdims and \drawplot may
2385 % not yet exist. With a wrapper they are looked up at expansion
2386 % time.
2387
2388 \cs_new:Npn \__numodel_calc_plot_dims: { \calcplothdims }
2389 \cs_new:Npn \__numodel_draw_plot:n #1 { \drawplot {#1} }
2390
2391 % =====
2392 % \diagrammodel -- convenience command for model graphs
2393 % =====
2394 % Usage: \diagrammodel{xvar}{yvar}[extra drawplot code]{label}
2395 %
2396 % Builds a complete figure with model points. Sets axis dimensions
2397 % from min/max, axis labels from text/unit, and emits caption and
2398 % label automatically.
2399
2400 \tl_new:N \l__numodel_xname_tl
2401 \tl_new:N \l__numodel_yname_tl
2402
2403 \NewDocumentCommand{\diagrammodel}{0{} m m 0{} m }
2404 {
2405   \tl_clear:N \l__numodel_cmd_prefix_tl
2406   \keys_set:nn { numodel / cmd } {#1}
2407   \__numodel_resolve_eff_prefix:
2408   % Full names (prefix + short) for \use:c lookup on globals
2409   \tl_set:Nx \l__numodel_xname_tl { \l__numodel_eff_prefix_tl #2 }
2410   \tl_set:Nx \l__numodel_yname_tl { \l__numodel_eff_prefix_tl #3 }
2411   \edef\dm@coords{\mcoordsp{\l__numodel_eff_prefix_tl}{#2}{#3}}
2412   \def\xmin{\use:c{\l__numodel_xname_tl min}}
2413   \def\xmax{\use:c{\l__numodel_xname_tl max}}
2414   \def\xlabelqty{\use:c{\l__numodel_xname_tl text}}

```

```

2415 \def\xlabelunit{\use:c{\l__numodel_xname_tl unitraw}}
2416 \def\ymin{\use:c{\l__numodel_yname_tl min}}
2417 \def\ymax{\use:c{\l__numodel_yname_tl max}}
2418 \def\ylabelqty{\use:c{\l__numodel_yname_tl text}}
2419 \def\ylabelunit{\use:c{\l__numodel_yname_tl unitraw}}
2420 \begin{figure}[H]
2421 \centering
2422 \group_begin:
2423 \__numodel_apply_dsep:
2424 \__numodel_draw_plot:n
2425 {
2426 \addplot[only~marks,~mark=*,~mark~size=0.5pt]~coordinates~{\dm@coords};
2427 \addlegendentry{model~point}
2428 #4
2429 }
2430 \group_end:
2431 \caption{${\use:c{\l__numodel_yname_tl text}}${\use:c{\l__numodel_xname_tl text}}$\text{-
diagram}}
2432 % If the project-specific worksheet system is loaded, pick the
2433 % '-ws' label inside a worksheet; otherwise always the bare label.
2434 \cs_if_exist:NTF \ifinworksheet
2435 { \ifinworksheet{\label{fig:#5-ws}}{\label{fig:#5}} }
2436 { \label{fig:#5} }
2437 \end{figure}
2438 }
2439
2440 \ExplSyntaxOff
2441
2442 % =====
2443 % GRAPHIC MODEL - STYLES AND MACROS
2444 % =====
2445 %
2446 % Forrester-diagram conventions:
2447 %
2448 % Stock [stock] Rectangle v, s, T, x
2449 % Valve / inflow [valve] Circle on a thick a, dv, ds
2450 % flow pipe
2451 % Auxiliary [aux] Plain circle F_res, F_air
2452 % Constant \constnode Circle + dashes m, k, g
2453 %
2454 % Use \dgridx and \dgridy as the grid spacings for consistent positioning.
2455 % =====
2456
2457 % === Grid spacing -- defaults set via \numodelsetup {graphscalex, graphscaley, stockwidth} ===
2458
2459 % === Styles ===
2460 \tikzset{
2461 % Scale coordinates to the grid spacing
2462 gridscale/.style={x=\dgridx cm, y=\dgridy cm},
2463 % Stock (state variable): rectangle
2464 stock/.style={
2465 rectangle, draw, thick,
2466 minimum width=3 em,
2467 minimum height=2 em,
2468 font=\small
2469 },
2470 % Auxiliary (intermediate variable): plain circle
2471 aux/.style={
2472 circle, draw, thick,
2473 inner sep=2pt,
2474 minimum size=2 em,
2475 font=\scriptsize
2476 },
2477 % Constant: circle (dashes drawn by the \constnode macro)
2478 const/.style={
2479 circle, draw, thick,

```



```

2480 inner sep=2pt,
2481 minimum size=2 em,
2482 font=\scriptsize
2483 },
2484 % Valve (legacy alias): circle on the flow pipe. Kept for
2485 % backwards compatibility with user code that calls \flowarrow
2486 % manually. The package's emit helpers now pick one of the
2487 % style variants below based on \numodelsetup{valve-style=...}.
2488 valve/.style={
2489 circle, draw, thick, fill=white,
2490 minimum size=2 em,
2491 font=\small
2492 },
2493 % Valve variant: Forrester valve symbol -- two filled triangles
2494 % (white inside) whose tips meet at the centre. The symmetry
2495 % axis stands perpendicular to the (horizontal) flow. The
2496 % bounding box is tall and narrow so the apex angle at the
2497 % centre is roughly 60 degrees (sharper than the 90 degrees of
2498 % a square box), and the horizontal "ribs" along the top and
2499 % bottom edges close the two triangles.
2500 valve-forrester/.style={
2501 shape=rectangle,
2502 draw=none, fill=none,
2503 inner sep=0pt, outer sep=0pt,
2504 %minimum width=\dgridx*0.18 cm,
2505 minimum width=1 em,
2506 %minimum height=\dgridy*0.30 cm,
2507 minimum height=2 em,
2508 path picture={%
2509 \draw[thick, fill=white]
2510 (path picture bounding box.north west) --
2511 (path picture bounding box.north east) --
2512 (path picture bounding box.center) --
2513 cycle;
2514 \draw[thick, fill=white]
2515 (path picture bounding box.south west) --
2516 (path picture bounding box.south east) --
2517 (path picture bounding box.center) --
2518 cycle;
2519 }
2520 },
2521 % Valve variant: empty circle (no label).
2522 valve-circle/.style={
2523 circle, draw, thick, fill=white,
2524 minimum size=2 em,
2525 font=\small
2526 },
2527 % Valve variant: circle with the variable's display label inside.
2528 valve-edu/.style={
2529 circle, draw, thick, fill=white,
2530 minimum size=2 em,
2531 font=\small
2532 },
2533 % Flow pipe variants (body without arrow tip, used for the
2534 % segment from open-end / cloud to the valve):
2535 flowpipe/.style={line width=3pt}, % legacy alias
2536 flowpipe-filled/.style={line width=3pt},
2537 flowpipe-hollow/.style={
2538 line width=1pt,
2539 double=white, double distance=3pt
2540 },
2541 % Flow pipe variants WITH arrow tip (used for the segment from
2542 % valve to stock, or the open-end side of an outflow). Both
2543 % variants use the Stealth arrow tip. The hollow form
2544 % uses the double-line magic length 0pt-3-0 so that the arrow
2545 % tip merges cleanly with the gap between the two strokes.

```

```

2546 flowpipe-filled-tip/.style={line width=3pt, arrows={-Stealth[inset=0pt, angle=30:1em]}},
2547 flowpipe-hollow-tip/.style={
2548   line width=1pt,
2549   double=white, double distance=3pt,
2550   arrows={-Stealth[inset=0pt, angle=30:1em]}
2551 },
2552 % Flow pipe variants with the Cloud arrow tip on the open end:
2553 % inflow keeps the Stealth arrow head at the stock side.
2554 flowpipe-filled-cloudin/.style={
2555   line width=3pt,
2556   arrows={Cloud[fill=white, line width=0.8pt]-Stealth[inset=0pt, angle=30:1em]}
2557 },
2558 flowpipe-hollow-cloudin/.style={
2559   line width=1pt,
2560   double=white, double distance=3pt,
2561   arrows={Cloud[fill=white, line width=0.8pt]-Stealth[inset=0pt, angle=30:1em]}
2562 },
2563 % Outflow with cloud at the open end: first a Stealth tip at the
2564 % line end, then the cloud behind it (in arrows.meta spec, list
2565 % multiple tips separated by a dot so the shaft stops at the
2566 % Stealth tip and does not extend into the cloud; order = from
2567 % line outwards, hence Stealth.Cloud).
2568 flowpipe-filled-cloudout/.style={
2569   line width=3pt,
2570   arrows={-{Stealth[inset=0pt, angle=30:1em].Cloud[fill=white, line width=0.8pt]}}
2571 },
2572 flowpipe-hollow-cloudout/.style={
2573   line width=1pt,
2574   double=white, double distance=3pt,
2575   arrows={-{Stealth[inset=0pt, angle=30:1em].Cloud[fill=white, line width=0.8pt]}}
2576 },
2577 % Causal arrow (thin arrow for dependencies)
2578 causal/.style={thick, arrows={-Stealth[length=0.5em]}}
2579 }
2580
2581 % Default values for the body / tip macros so a manual call to
2582 % \flowarrow (outside \graphicmodel) keeps working. The
2583 % \graphicmodel pipeline overwrites these per render.
2584 \providecommand{\nmflowbody}{flowpipe-filled}
2585 \providecommand{\nmflowtip}{flowpipe-filled-tip}
2586 \providecommand{\nmflowtipcloudin}{flowpipe-filled-cloudin}
2587 \providecommand{\nmflowtipcloudout}{flowpipe-filled-cloudout}
2588
2589 % === Macro: constant node with dashes ===
2590 % Usage: \constnode{name}{(x,y)}{$label$}
2591 \newcommand{\constnode}[3]{%
2592   \node[const] (#1) at (#2) {#3};%
2593   \draw[thick] ([xshift=-3mm]#1.west) -- (#1.west);%
2594   \draw[thick] (#1.east) -- ([xshift=3mm]#1.east);%
2595 }
2596
2597 % === Macro: flow pipe with valve (inflow) ===
2598 % Draws ONE continuous flow arrow ending at the stock. The starting
2599 % position is the same regardless of cloud presence; passing any
2600 % non-blank optional argument switches to the cloud-tipped variant
2601 % so the arrow's tail becomes a white-filled cloud (drawn as a real
2602 % PGF arrow tip, not a separate node - moves with the line). The
2603 % valve overlays the middle of the arrow as a separate white-filled
2604 % node drawn later by the build pipeline.
2605 % Style keys: \nmflowtip / \nmflowtipcloudin.
2606 % Usage: \flowarrow[<any>]{<valve-coord>}{<stock-node>}
2607 \NewDocumentCommand{\flowarrow}{O{} m m }{%
2608   \IfBlankTF{#1}{%
2609     \draw[\nmflowtip] ([xshift=-3em]#2) -- ([xshift=0.5mm]#3.west);%
2610   }{%
2611     \draw[\nmflowtipcloudin] ([xshift=-3em]#2) -- ([xshift=0.5mm]#3.west);%

```

```

2612 }%
2613 }
2614
2615 % === Macro: flow pipe with valve (outflow) ===
2616 % Mirror of \flowarrow. The arrow always ends at the same offset
2617 % past the valve coordinate; passing a non-blank optional argument
2618 % switches to the cloud-tipped variant (cloud at the open end).
2619 % Usage: \flowoutarrow[<any>]{<stock-node>}{<valve-coord>}
2620 \NewDocumentCommand{\flowoutarrow}{0}{m m}{%
2621 \IfBlankTF{#1}{%
2622 \draw[\nmflowtip] (#2.east) -- ([xshift=4em]#3);%
2623 }{%
2624 \draw[\nmflowtipcloudout] (#2.east) -- ([xshift=5em]#3);%
2625 }%
2626 }
2627
2628 % === Macro: flow pipe between two stocks ===
2629 % One continuous arrow from source stock through valve coord into
2630 % target stock. No cloud option (no open end exists when both
2631 % sides are stocks).
2632 % Usage: \flowbetweenarrow[<source-stock>]{<valve-coord>}{<target-stock>}
2633 \newcommand{\flowbetweenarrow}[3]{%
2634 \draw[\nmflowtip] (#1.east) -- ([xshift=0.5mm]#3.west);%
2635 }
2636 \end{package}

```

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`.

```

2637 (*EN)
2638 \ProvidesExplFile{numodel-EN.def}{2026/05/19}{0.4.0}{
2639 {numodel-keyword~table~---English~(XMILE)}}
2640 \cs_new:Npn \__numodel_kw_EN_if: { IF~ }
2641 \cs_new:Npn \__numodel_kw_EN_then: { ~THEN~ }
2642 \cs_new:Npn \__numodel_kw_EN_then_nl: { ~THEN }
2643 \cs_new:Npn \__numodel_kw_EN_else: { ~ELSE~ }
2644 \cs_new:Npn \__numodel_kw_EN_else_nl: { ELSE }
2645 \cs_new:Npn \__numodel_kw_EN_endif: { ~ENDIF }
2646 \cs_new:Npn \__numodel_kw_EN_endif_nl: { ENDIF }
2647 \cs_new:Npn \__numodel_kw_EN_and: { ~AND~ }
2648 \cs_new:Npn \__numodel_kw_EN_or: { ~OR~ }
2649 \cs_new:Npn \__numodel_kw_EN_not: { NOT~ }
2650 \cs_new:Npn \__numodel_kw_EN_sign: { SIGN }
2651 \cs_new:Npn \__numodel_kw_EN_abs: { ABS }
2652 \cs_new:Npn \__numodel_kw_EN_sqrt: { SQRT }
2653 \cs_new:Npn \__numodel_kw_EN_exp: { EXP }
2654 \cs_new:Npn \__numodel_kw_EN_ln: { LN }
2655 \cs_new:Npn \__numodel_kw_EN_sin: { SIN }
2656 \cs_new:Npn \__numodel_kw_EN_cos: { COS }
2657 \cs_new:Npn \__numodel_kw_EN_tan: { TAN }
2658 \cs_new:Npn \__numodel_kw_EN_asin: { ARCSIN }
2659 \cs_new:Npn \__numodel_kw_EN_acos: { ARCCOS }
2660 \cs_new:Npn \__numodel_kw_EN_stop: { ~STOP~ }
2661 \cs_new:Npn \__numodel_kw_EN_th_model: { model }
2662 \cs_new:Npn \__numodel_kw_EN_th_initvals: { initial~values }
2663 \cs_new:Npn \__numodel_kw_EN_contfoot: { Continued~on~next~page }
2664 \cs_new:Npn \__numodel_kw_EN_conthead: { (Continued) }
2665 \cs_new:Npn \__numodel_kw_EN_dsep_default: { point }
2666 \end{EN}

```

9.1.2 numodel-NL.def – Dutch (CoachTaal)

CoachTaal convention. Sets the default decimal mark to comma.

```
2667 (*NL)
2668 \ProvidesExplFile{numodel-NL.def}{2026/05/19}{0.4.0}
2669 {numodel~keyword~table~---Dutch~(CoachTaal)}
2670 \cs_new:Npn \__numodel_kw_NL_if: { Als~ }
2671 \cs_new:Npn \__numodel_kw_NL_then: { ~Dan~ }
2672 \cs_new:Npn \__numodel_kw_NL_then_nl: { ~Dan }
2673 \cs_new:Npn \__numodel_kw_NL_else: { ~Anders~ }
2674 \cs_new:Npn \__numodel_kw_NL_else_nl: { Anders }
2675 \cs_new:Npn \__numodel_kw_NL_endif: { ~EindAls }
2676 \cs_new:Npn \__numodel_kw_NL_endif_nl: { EindAls }
2677 \cs_new:Npn \__numodel_kw_NL_and: { ~EN~ }
2678 \cs_new:Npn \__numodel_kw_NL_or: { ~OF~ }
2679 \cs_new:Npn \__numodel_kw_NL_not: { NIET~ }
2680 \cs_new:Npn \__numodel_kw_NL_sign: { Teken }
2681 \cs_new:Npn \__numodel_kw_NL_abs: { Abs }
2682 \cs_new:Npn \__numodel_kw_NL_sqrt: { Sqrt }
2683 \cs_new:Npn \__numodel_kw_NL_exp: { Exp }
2684 \cs_new:Npn \__numodel_kw_NL_ln: { Ln }
2685 \cs_new:Npn \__numodel_kw_NL_sin: { Sin }
2686 \cs_new:Npn \__numodel_kw_NL_cos: { Cos }
2687 \cs_new:Npn \__numodel_kw_NL_tan: { Tan }
2688 \cs_new:Npn \__numodel_kw_NL_asin: { Arcsin }
2689 \cs_new:Npn \__numodel_kw_NL_acos: { Arccos }
2690 \cs_new:Npn \__numodel_kw_NL_stop: { ~Stop~ }
2691 \cs_new:Npn \__numodel_kw_NL_th_model: { model }
2692 \cs_new:Npn \__numodel_kw_NL_th_initvals: { startwaarden }
2693 \cs_new:Npn \__numodel_kw_NL_contfoot: { Wordt~vervolgd~op~volgende~pagina }
2694 \cs_new:Npn \__numodel_kw_NL_conthead: { (Vervolg) }
2695 \cs_new:Npn \__numodel_kw_NL_dsep_default: { comma }
2696 (/NL)
```