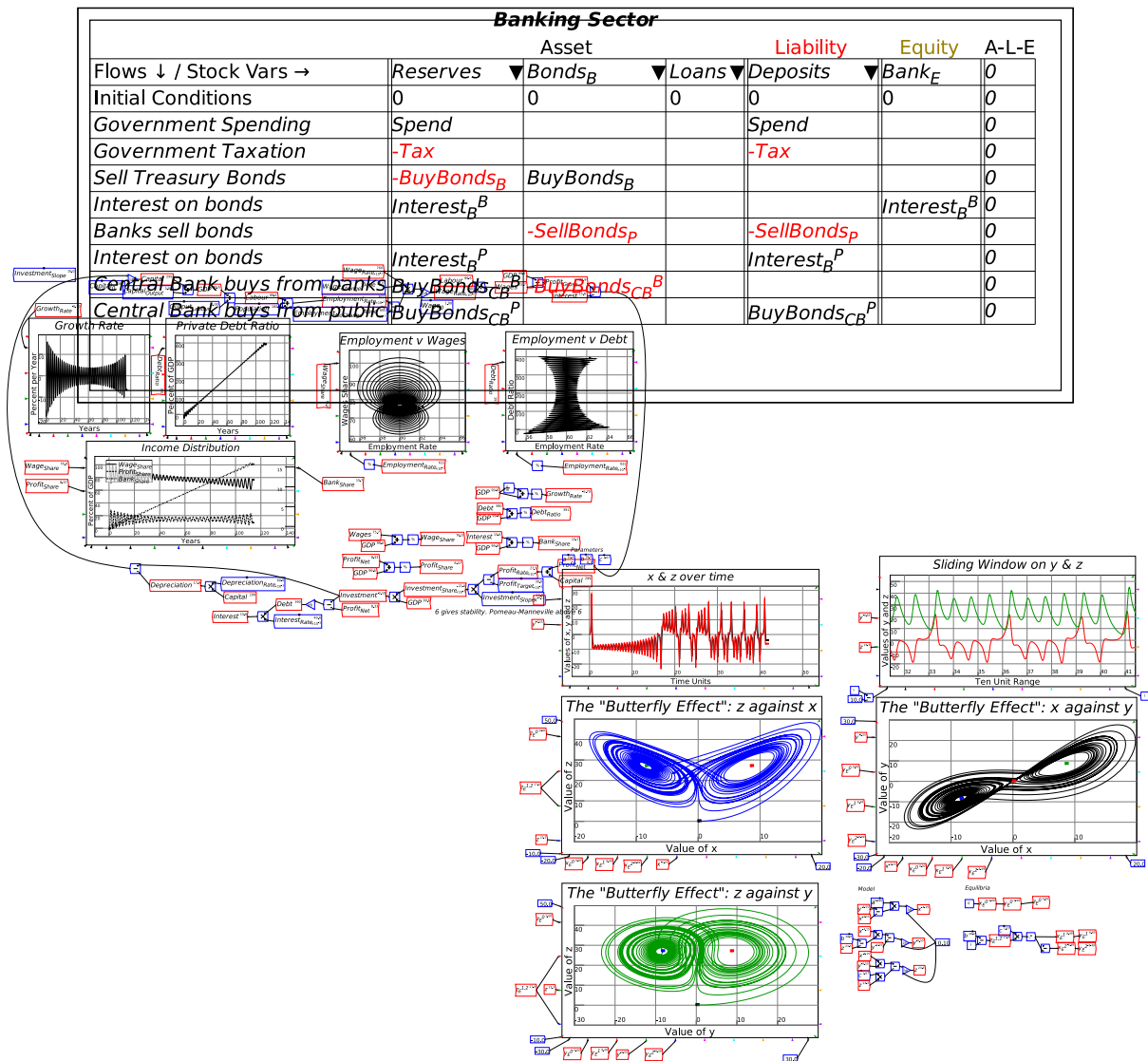
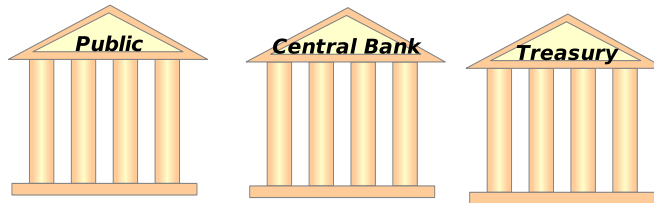


A Quick Introduction to *Minsky*

September 2022

© Steve Keen



Contents

1	What is <i>Minsky</i> ?	4
2	The Interface	8
2.1	Control menu	8
2.1.1	File : saving models, loading models, exporting them to different formats (PDF, SVG, etc.)	8
2.1.2	Edit : copying, cutting, pasting model components, controlling dimensions and units, modifying model layout automatically	8
2.1.3	Bookmarks : creating and navigating regions of Minsky's design canvas	8
2.1.4	Insert : inserting model components	8
2.1.5	Options : system-wide settings	8
2.1.6	Simulation : numerical features of Minsky	8
2.1.7	Help : Minsky's online documentation	8
2.2	Run bar	8
2.2.1	Recalculating a model	9
2.2.2	Recording a model's construction	9
2.2.3	Replaying its construction	9
2.2.4	Running it in reverse,	9
2.2.5	Running, pausing, stopping and stepping a simulation	9
2.2.6	The speed of simulation	9
2.2.7	The scale of display of the canvas	9
2.3	Tabs	9
2.3.1	Designing a model ("Wiring")	9
2.3.2	Seeing the model's equations	9
2.3.3	Listing its parameters	9
2.3.4	Listing variables that are not defined on the Wiring canvas	9
2.3.5	Showing selected plots	9
2.3.6	Showing Godley Tables	9
2.3.7	Accessing the command line (Terminal)	9
2.4	Widgets	9
2.4.1	Import multidimensional data (available in Ravel only)	10
2.4.2	Attach data to a Ravel (available in Ravel only)	10
2.4.3	Insert a plot	10
2.4.4	Inserting a spreadsheet	10
2.4.5	Placing a constant, parameter and variable on the canvas and access the Browser window	10

2.4.6	Insert a lock (used in Ravel rather than Minsky)	10
2.4.7	Insert a note.....	10
2.4.8	Insert the model time	10
2.4.9	Mathematical operators combining two or more entities	10
2.4.10	Mathematical operators taking a single input.....	10
2.4.11	Fundamental constants ($e, \pi, 0, 1, \infty$) and the percent operator	10
2.4.12	Reduction operators (infimum, etc.).....	10
2.4.13	Aggregation operators (Sum, Product, Difference).....	10
2.4.14	Tensor operators (inner product, outer product, etc.)	10
2.4.15	Logical switch operator	10
2.4.16	User defined functions	10
2.4.17	Godley Tables	10
2.4.18	Integral block	10
2.4.19	Differential operator	10
2.5	Canvases.....	10
3	Widgets and wires	13
4	Mathematical operators.....	15
5	The Browser Window	16
6	Godley Tables.....	17
7	Plots and sliders	22
8	Integrated general models of financial flows.....	25
9	Flowchart modelling	29
10	Organizing a model: Bookmarks	34
11	Organizing a model: Groups.....	36
12	Getting Help	38
13	Helping Minsky	40
13.1	Future Developments.....	40

1 What is *Minsky*?

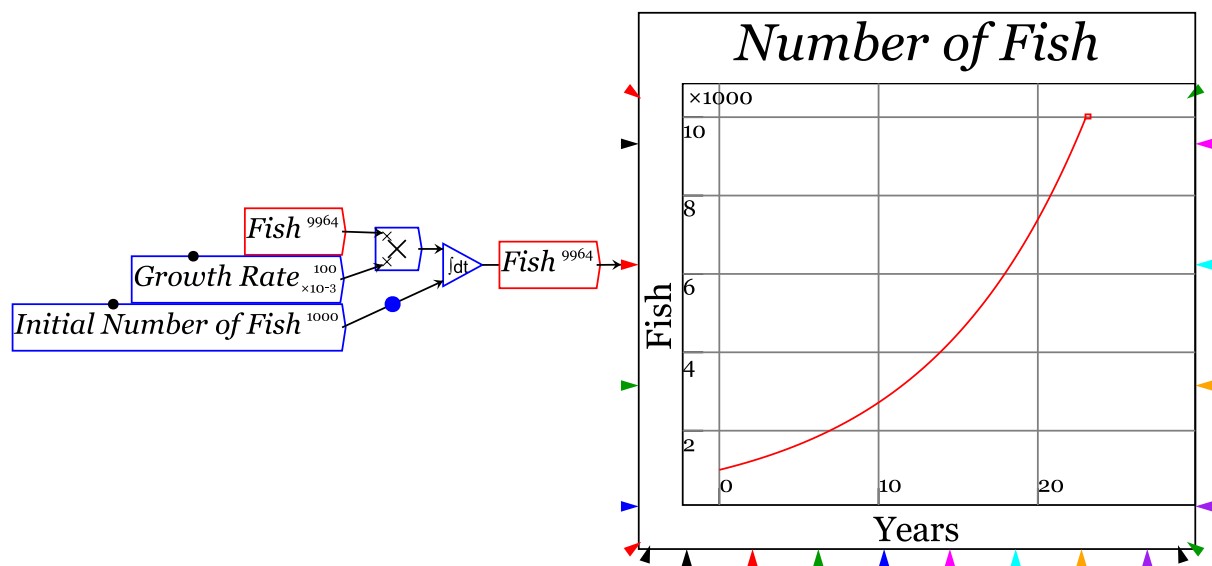
Minsky is an Open-Source [system-dynamics](https://sourceforge.net/projects/minsky/) program.

Open-Source means that its code is readily available, and that it is free. To use it, download it from <https://sourceforge.net/projects/minsky/>, and install it on your computer.

System-dynamics means that it can model two or more factors which cause changes in each other over time.

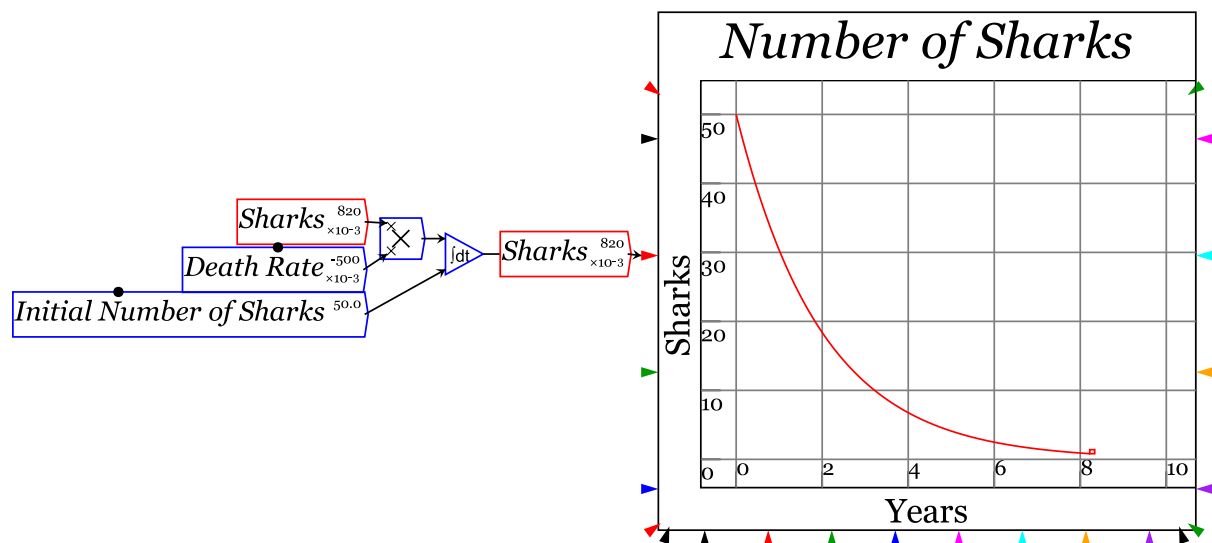
A simple model of population growth with just one species—Fish, for example—is not a system. It generates the familiar exponential curve of fish numbers growing forever—see Figure 1.

Figure 1: A model of population growth



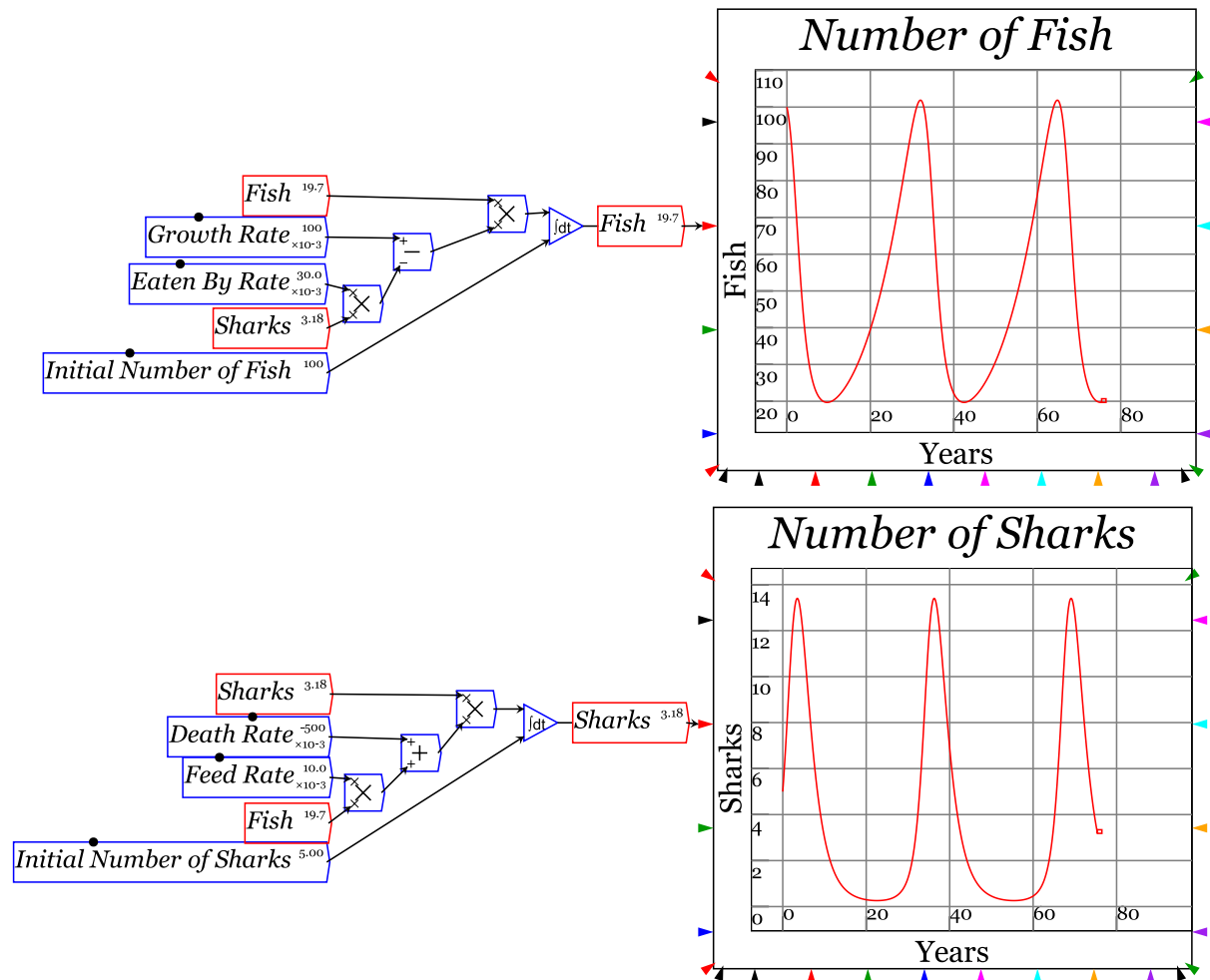
Nor is a model of population decay with just one species—say, Sharks with no Fish to eat—a system. It generates the opposite effect of exponential decay—see Figure 2 (presumably, the sharks are eating each other as their numbers fall).

Figure 2: A model of population decline



But a model with both Fish and Sharks is a system, and it has a surprising outcome: it never settles down to an equilibrium. Instead, as Figure 3 illustrates, the number of fish and sharks rise and fall in cycles that go on forever.

Figure 3: A system dynamics model of Fish and Sharks



The capacity that system dynamics programs have to model *feedback effects* between variables frees the modeler from the tyranny of “[ceteris paribus](#)”—the bedrock assumption of a conventional first-year course in economics that “all other things remain equal”. With a system dynamics program, you no longer have to make that manifestly assumption, and a much richer and more informative model of the world emerges—for instance, the “[predator-prey](#)” model indicates that not all years are the same when you take the risk of sharing the ocean with sharks.

System dynamics programs were first invented in the 1950s by the brilliant MIT Professor of Management [Jay Forrester](#). There are [many different programs](#) today—[Stella](#), [Vensim](#), [Simulink](#), [System Modeler](#), to mention but a few. *Minsky*’s unique feature in this crowded marketplace is its capacity to model financial dynamics: systems involving the exchange, creation and destruction of money.

Minsky does this using what we call *Godley Tables*.¹ System dynamics programs were arguably the first-ever computer programs to employ a GUI (“[Graphical User Interface](#)”). *Godley Tables* employ a GUI that long predates computers: the accountant’s device of “[Double-entry Bookkeeping](#)”.

Double-entry Bookkeeping (*DEB* for short) was invented so that merchants could be sure that their financial transactions were accurately recorded. Though it takes a degree in accounting (or at least a course in bookkeeping) to fully understand it, DEB boils down to two simple principles:

1. All financial claims are classified as either Assets or Liabilities; and
2. All financial transactions are recorded twice.

Assets are financial claims on another person or entity; *Liabilities* are a financial claim that another person or entity has on you. For any person or entity (a business, charity, etc.), the gap between its *Assets* and its *Liabilities* is its net worth, which is called its *Equity*. The recording of all transactions twice enables a check on the accuracy of an accountant’s books via the formula that:

- $Assets - Liabilities = Equity$, or
- $Assets - Liabilities - Equity = 0$

In accounting, if a given transaction is properly recorded, then this equation will hold. If there’s a mistake, this equation is violated. This simple set of rules enables accountants to show the state of a business.

Minsky uses this “fundamental accounting equation” to enable you to build a model of financial dynamics easily and rapidly, without the messy wiring of the standard system dynamics “flowchart” paradigm. Instead, we use *Godley Tables*, where all claims are recorded as either Assets (A) or Liabilities (L), the Equity (E) of each entity in a model is tracked, and any error is checked by the simple rule $A - L - E = 0$. Figure 4 show a blank *Godley Table*.

Figure 4: A blank *Godley Table*

	Asset	Liability	Equity	A-L-E
Flows ↓ / Stock Vars →	▼	▼	▼	0
Initial Conditions				0

This simple GUI makes it easy to answer questions that are still contentious in economics and finance today:

- Do banks lend out deposits, or do they create money “from nothing”?
- Do banks lend out reserves, or do loans create deposits with no need for reserves?

Economics textbooks argue that banks need deposits in order to be able lend, and that reserves—the cash that banks have in their vaults, plus the funds that private banks have in their own deposit accounts at the Central Bank—play an essential role in lending and money creation. In 2014, the Bank of England said that the textbooks were wrong:

¹ *Minsky* was named to honour the great non-mainstream economist [Hyman Minsky](#). *Godley Tables* are named after his equally great non-mainstream colleague [Wynne Godley](#).

“The reality of how money is created today differs from the description found in some economics textbooks:

- Rather than banks receiving deposits when households save and then lending them out, bank lending creates deposits.
- In normal times, the central bank does not fix the amount of money in circulation, nor is central bank money ‘multiplied up’ into more loans and deposits.” {McLeay, 2014 #5066, p. 41}

Using a *Godley Table* in *Minsky*, you can quickly see for yourself that the Bank of England is correct—“bank lending creates deposits”. There is also something fishy about the two main stories that economics textbooks tell about lending: that banks are just intermediaries who enable savers to lend to borrowers; and that banks “multiply up” reserves in order to create money.

Minsky’s simple interface makes it easy to understand an issue that is still contentious today. The first row in Figure 5 shows the real-world situation: a bank creates money by putting money into a borrower’s bank account, and simultaneously recording the same amount as a debt the borrower owes to the bank.

Figure 5: A Godley Table showing three forms of lending

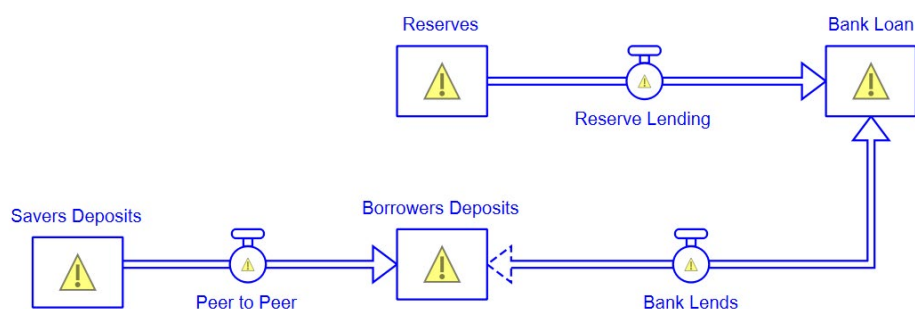
Flows ↓ / Stock Vars →	Asset		Liability		Equity	A-L-E
	Reserves	Bank Loans	Savers Deposits	Borrowers Deposits	Bank Equity	
Initial Conditions	100	200	200	50	50	0
Bank lends to Borrowers		Bank Lends		Bank Lends		0
Bank lends from Reserves	-Reserve Lending	Reserve Lending				0
Savers lend to Borrowers			-Peer to Peer	Peer to Peer		0

The second row shows banks “lending from reserves”. It works—Reserves go down and Bank Loans go up— but there’s a problem: how does the borrower get the money?

The third row shows the model of “loanable funds”, where savers lend to borrowers through the intermediary of a bank. Here the borrower gets the money, but to do so, the deposit accounts of savers must fall. But this doesn’t happen with ordinary “at call” bank deposits—if it did, savers would find their accounts going up and down as banks made loans and borrowers paid them back. That just isn’t the case in the real world.

There’s much more to using *Godley Tables* to create a model, but the tables themselves make it easy to read a model of financial flows. In contrast, the flowchart paradigm—which Minsky also provides—just isn’t as informative when it comes to financial flows: see Figure 6, which shows the same model as in Figure 5, rendered in the market-leading system dynamics program [Stella](#).

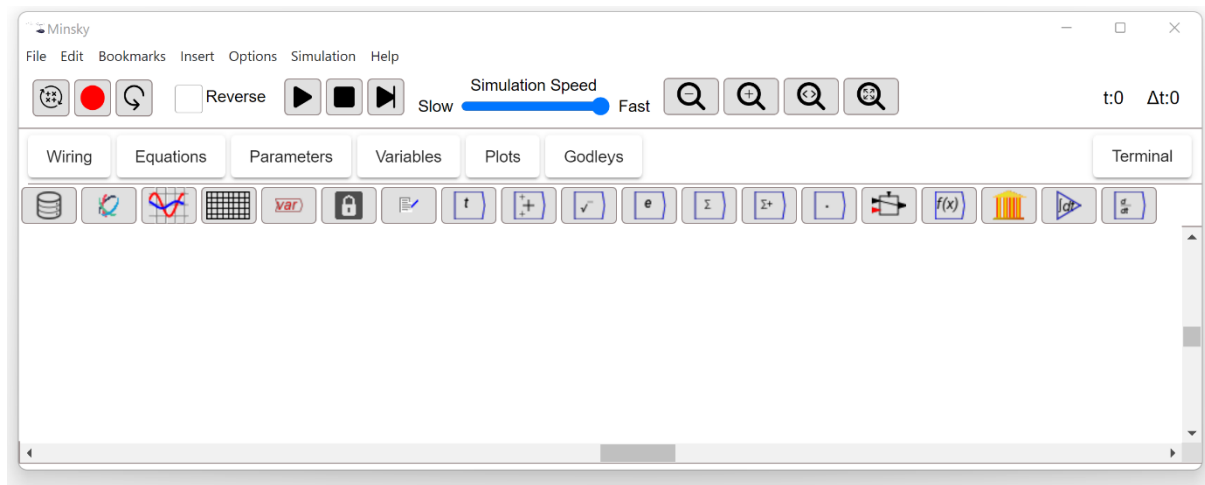
Figure 6: The same model as in the Godley Table in Figure 5



2 The Interface

Figure 7 shows Minsky's interface.

Figure 7: Minsky's Interface



There are 5 main elements to the interface:

2.1 Control menu

- 2.1.1 **File:** saving models, loading models, exporting them to different formats (PDF, SVG, etc.)
- 2.1.2 **Edit:** copying, cutting, pasting model components, controlling dimensions and units, modifying model layout automatically
- 2.1.3 **Bookmarks:** creating and navigating regions of Minsky's design canvas
- 2.1.4 **Insert:** inserting model components
 - 2.1.4.1 *variables, operators, etc*
- 2.1.5 **Options:** system-wide settings
 - 2.1.5.1 *whether Godley Tables display numeric values, the color of the canvas*
- 2.1.6 **Simulation:** numerical features of Minsky
 - 2.1.6.1 *time units—seconds, days, years, etc.—and technical details of the numerical solver)*
- 2.1.7 **Help:** Minsky's online documentation

2.2 Run bar

Icons here control:

- 2.2.1 Recalculating a model
- 2.2.2 Recording a model's construction
- 2.2.3 Replaying its construction
- 2.2.4 Running it in reverse,
- 2.2.5 Running, pausing, stopping and stepping a simulation
- 2.2.6 The speed of simulation
- 2.2.7 The scale of display of the canvas

2.2.7.1 zooming out, in, to original scale, and filling the window;

2.3 Tabs

There are tabs for

- 2.3.1 Designing a model ("Wiring")
- 2.3.2 Seeing the model's equations
- 2.3.3 Listing its parameters
- 2.3.4 Listing variables that are not defined on the Wiring canvas
- 2.3.5 Showing selected plots
- 2.3.6 Showing Godley Tables
- 2.3.7 Accessing the command line (Terminal)

2.4 Widgets

These icons place Minsky components on the canvas. They are, reading left to right:

- 2.4.1 Import multidimensional data (available in Ravel only)
- 2.4.2 Attach data to a Ravel (available in Ravel only)
- 2.4.3 Insert a plot
- 2.4.4 Inserting a spreadsheet
- 2.4.5 Placing a constant, parameter and variable on the canvas and access the Browser window
- 2.4.6 Insert a lock (used in Ravel rather than Minsky)
- 2.4.7 Insert a note
- 2.4.8 Insert the model time
- 2.4.9 Mathematical operators combining two or more entities
 - 2.4.9.1 (*add, subtract, multiply, etc.*);
 - 2.4.10 Mathematical operators taking a single input
 - 2.4.10.1 (*sin, cosine, modulus, etc.*);
 - 2.4.11 Fundamental constants ($e, \pi, 0, 1, \infty$) and the percent operator
 - 2.4.12 Reduction operators (infimum, etc.)
 - 2.4.13 Aggregation operators (Sum, Product, Difference)
 - 2.4.14 Tensor operators (inner product, outer product, etc.)
 - 2.4.15 Logical switch operator
 - 2.4.16 User defined functions
 - 2.4.17 Godley Tables
 - 2.4.18 Integral block
 - 2.4.19 Differential operator

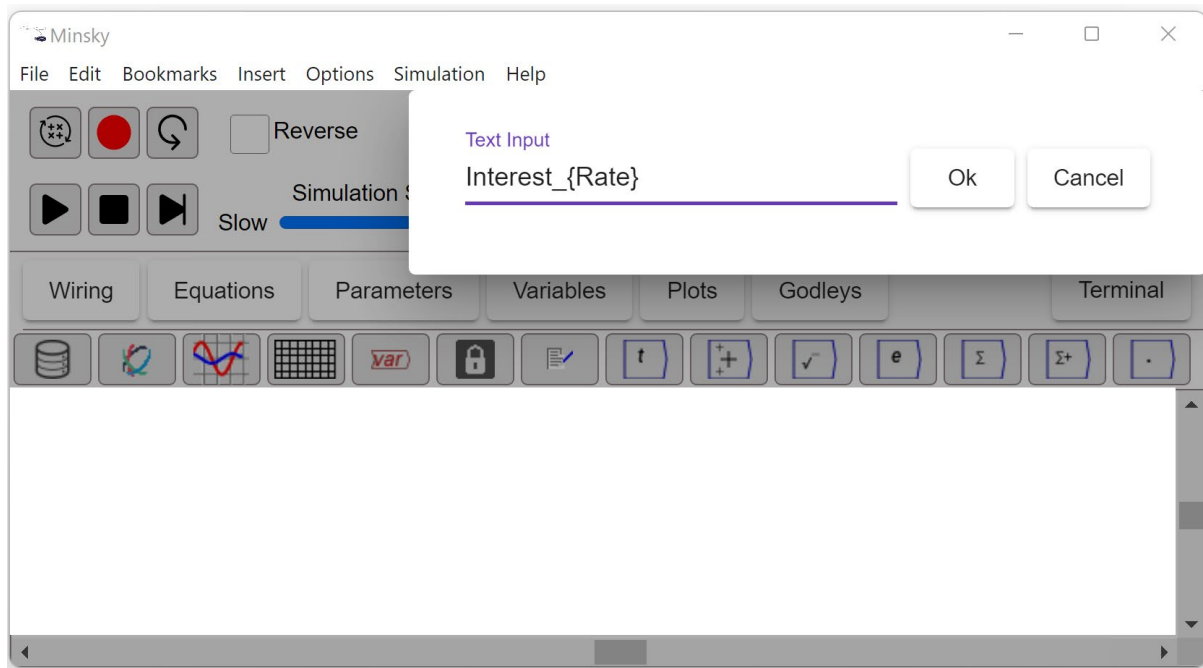
2.5 Canvases

This is where you design, inspect, and run a model. There is a different canvas for each Tab, with several for display purposes only—the equation, parameters and variables tabs—but others where you have some functionality: plots and Godleys where you can move elements around on screen as a prelude to exporting them for documentation purposes. The functionality of these other tabs will rise as we extend *Minsky*, but for now the main active canvas is Wiring.

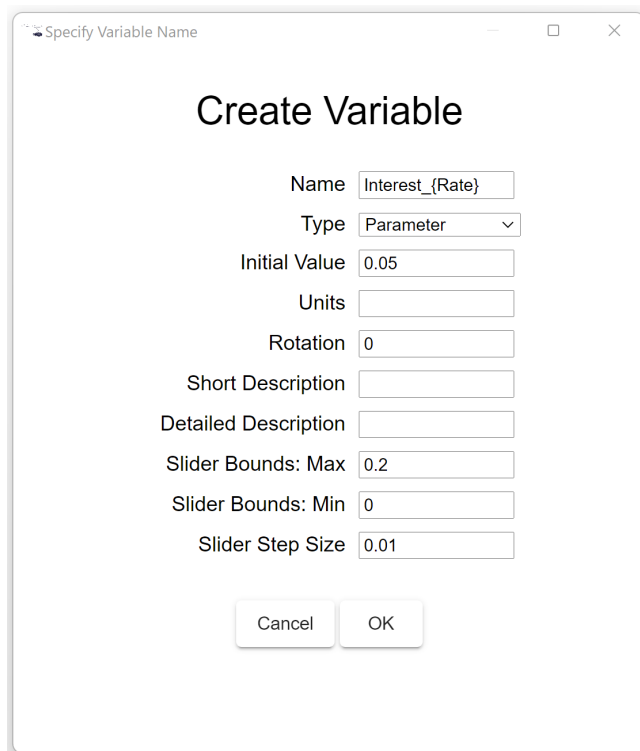
You can enter objects onto the Wiring canvas in two ways:

1. By clicking on the relevant object in the widget bar, and then clicking where you want to place the object placed on the canvas; or
2. By simply typing on the canvas where you'd like the object to appear.

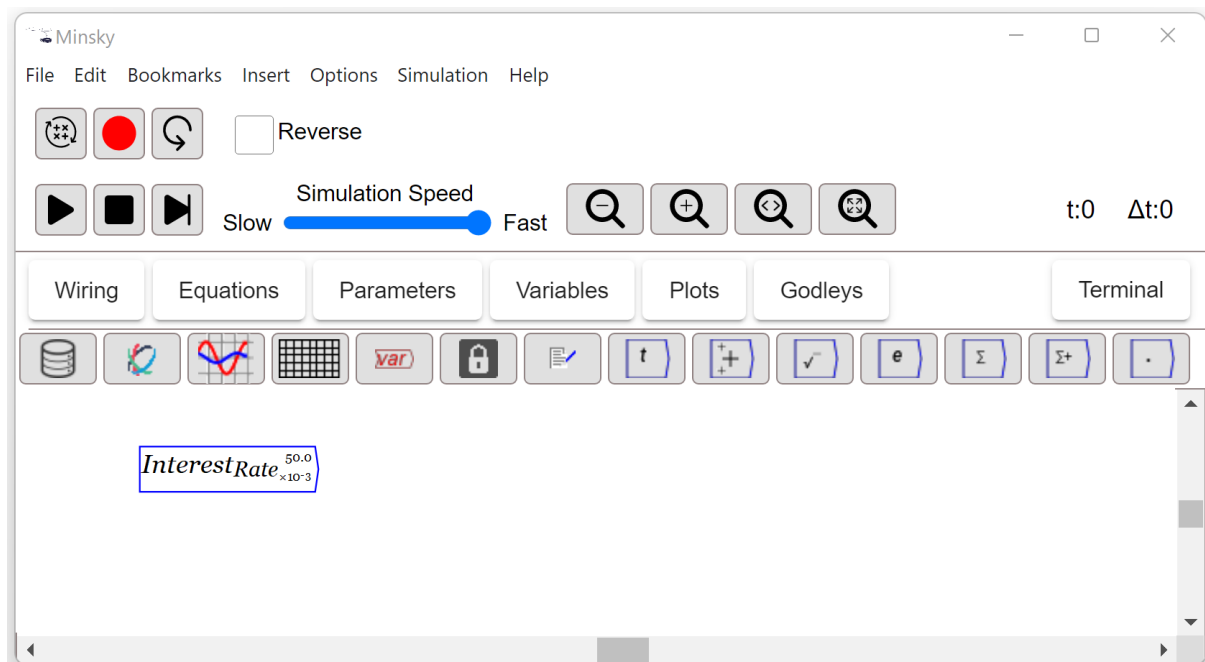
If you start typing text—for example, the string `Interest_{Rate}`—then a text entry window will pop up, as shown in Figure 8.

Figure 8: Creating the parameter InterestRate by typing directly onto the canvas

When you click on OK or press Enter, the “create. Variable” dialog box will pop up, where you can add properties to the variable—including change its type from a Variable to a Parameter (see Figure 9).

Figure 9: The Create Variable dialog box

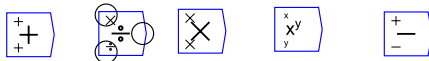
Once you click OK or press Enter on this form, the entity you’ve created will be attached to your mouse cursor. Move to where you wish to place it on the canvas and click, and it will appear there—see Figure 10.

Figure 10: $Interest_{Rate}$ now placed on the canvas

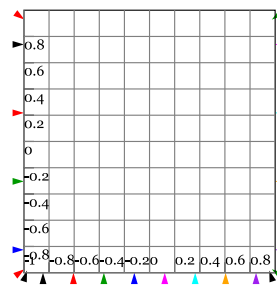
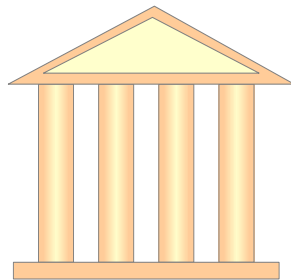
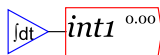
Minsky also allows you to enter common mathematical operators on the canvas directly by simply typing their key. Figure 11 summarizes the main direct entry operators.

Figure 11: The main direct entry operators

Entered by obvious keyboard key



Entered by special keys: & = and @ respectively



Entered by name then Enter:

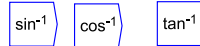
sin cos tan



sinh cosh tanh



asin acos atan

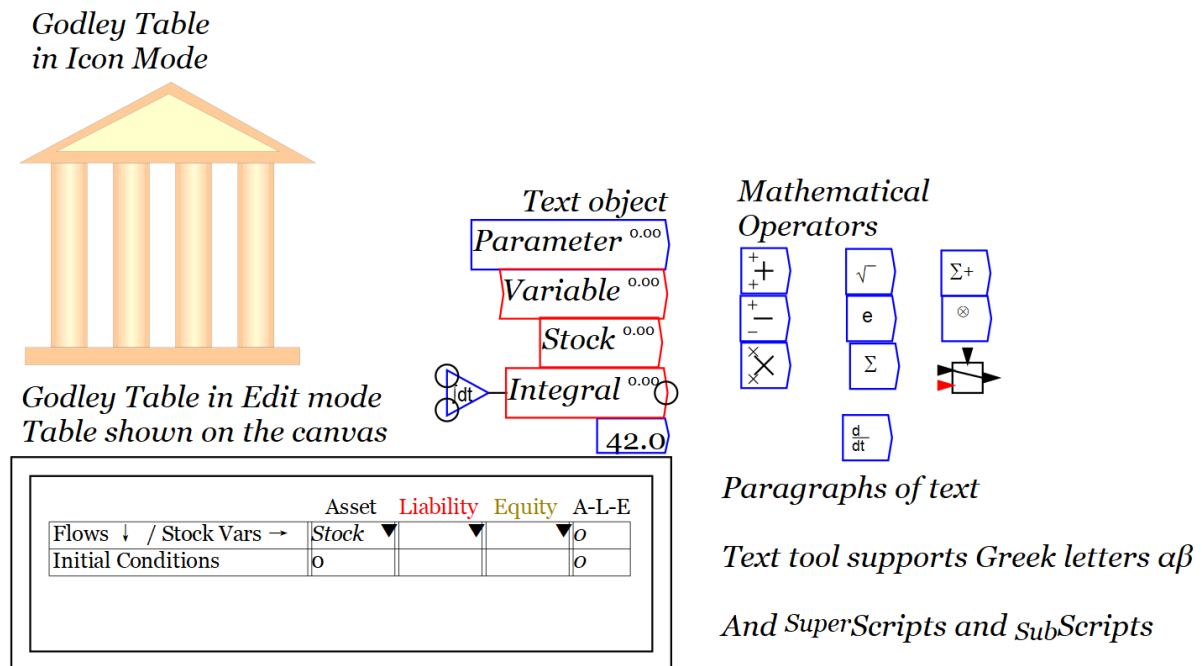


The only two with some complexity are the minus key and the percentage key. Since some users may want to name a variable with a preceding minus sign (or percentage sign), if you type $-$, the variable entry dialog box comes up. If you just want the minus operator, press Enter (or click on OK) and it will appear. If the minus is actually part of a name, keep typing and that variable will be created.

3 Widgets and wires

Minsky's major components, including Godley Tables (in two formats, widget and tabular view) parameters, variables, stocks and integrals are shown in Figure 12. Parameters, constants and mathematical operators are shown in blue, while all other user-defined components (variables, stocks, integrals) are shown in red.

Figure 12: Minsky's basic components



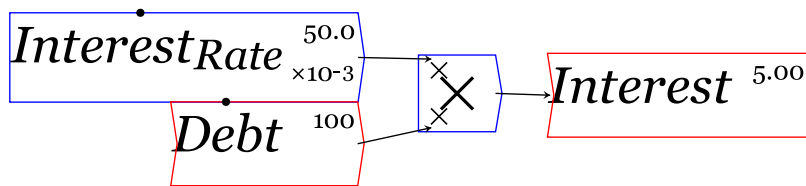
To make a model using Minsky's components, you have to wire them together.

Components have input and output ports, which are shown as circles when your mouse hovers over a component. The number of such ports depends on the component. Text boxes have none; parameters have an output port, but no input port; stocks are defined in and given numerical values by a Godley Table, and therefore also have only an output port when placed on the canvas; variables have an input and an output port; operators like the plus widget have two input ports and one output port. In Figure 12, the mouse was hovering over the *Integral* block, and therefore its output port and two input ports—one for the flow into the integral block, the other for its initial conditions—are visible.

A Minsky model is constructed by wiring together components, using the output and input ports. To wire elements together, you click in an output port, which causes a wire with an arrowhead to appear. Hold the mouse button down and drag the arrowhead towards the input port you wish to attach it to. When you are close, release the mouse button and the arrowhead will snap to the nearest input port.

In Figure 13, Interest has been defined as the $Interest_{Rate}$ times Debt.

Figure 13: Multiplying a parameter and a variable together to produce another variable.



Click on the Equations Tab and you will see Equation (1):

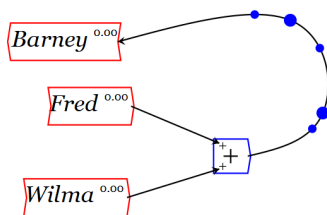
$$\begin{aligned} \text{Debt} &= 100 \\ \text{Interest}_{\text{Rate}} &= 0.05 \\ \text{Interest} &= \text{Interest}_{\text{Rate}} \times \text{Debt} \end{aligned} \quad (1)$$

Minsky continues to document your model in mathematical notation as you build it. The equations can be exported for use in a word processing program using the LaTeX option on the *Export Canvas* as item on the *File* menu.

Wires are straight by default, but can easily be curved. When your mouse is hovering over a wire, a blue dot will appear on the wire. Click on the blue dot (or anywhere else on the wire) and drag and you will change the shape of the wire into a curve.

This can get quite elaborate, and we allow for this getting out of control by the right-click menu item “Straighten” when you are hovering over a wire.

Figure 14: Curving a connection wire

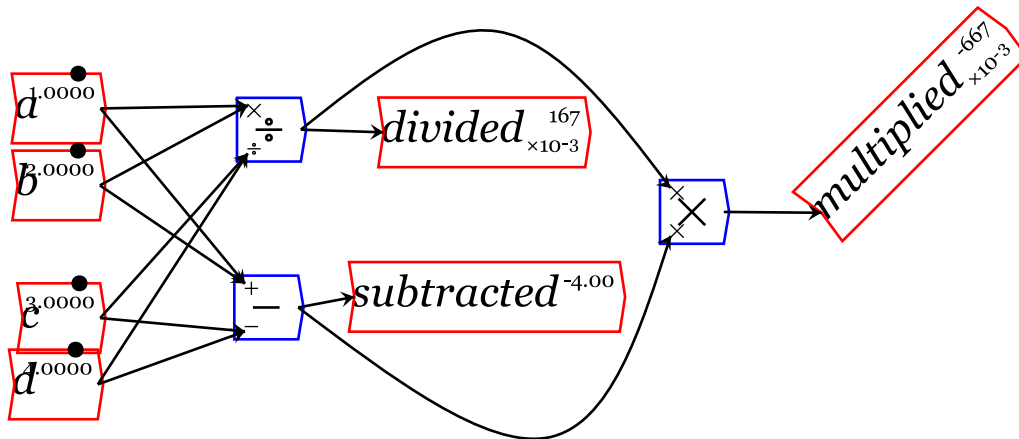


You will note that the Barney points in the opposite direction to Fred and Wilma in Figure 15; this is executed by the “Flip” command on the right menu for entities like variables, parameters, mathematical operators etc. Arbitrary angles are supported via the right-click Edit menu—see Figure 16.

4 Mathematical operators

Minsky supports many mathematical operators, from the familiar add (+), subtract (−), multiply (×) and divide (/), through trigonometric relations (\sin , \cos , \cosh) to operators for tensor mathematics (dot products, inner products, etc.). Many operators can be overloaded: an operator like + can take many inputs, not just two. Figure 16 illustrates overloading, with multiple inputs can be wired to one object, and multiple outputs as well.

Figure 15: Operator overloading, multiple outputs from output ports, and entity rotation



These are the equations generated by the functions in Figure 16:

$$a=1$$

$$b=2$$

$$c=3$$

$$d=4$$

$$\text{divided} = \frac{a \times b}{c \times d}$$

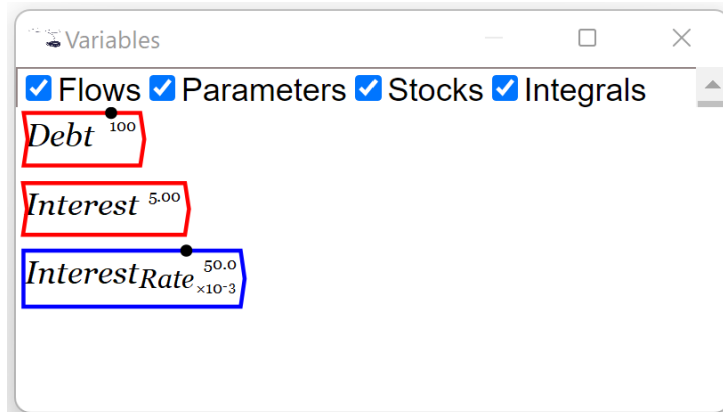
$$\text{subtracted} = a + b - (c + d)$$

$$\text{multiplied} = \frac{a \times b}{c \times d} \times (a + b - (c + d))$$

5 The Browser Window


As you build a model in Minsky, it generates a catalog of the entities in that model, which you can assess directly by the Browser window. To bring it up, click on the **var** widget and then choose *Browser*. Figure 16 shows the three variables from the equation in Figure 13 in the browser. You can copy these and place them elsewhere on the canvas by clicking once on an entity in the Browser, and then clicking again on the canvas.

Figure 16: The three entities in the model, shown on the Browser menu



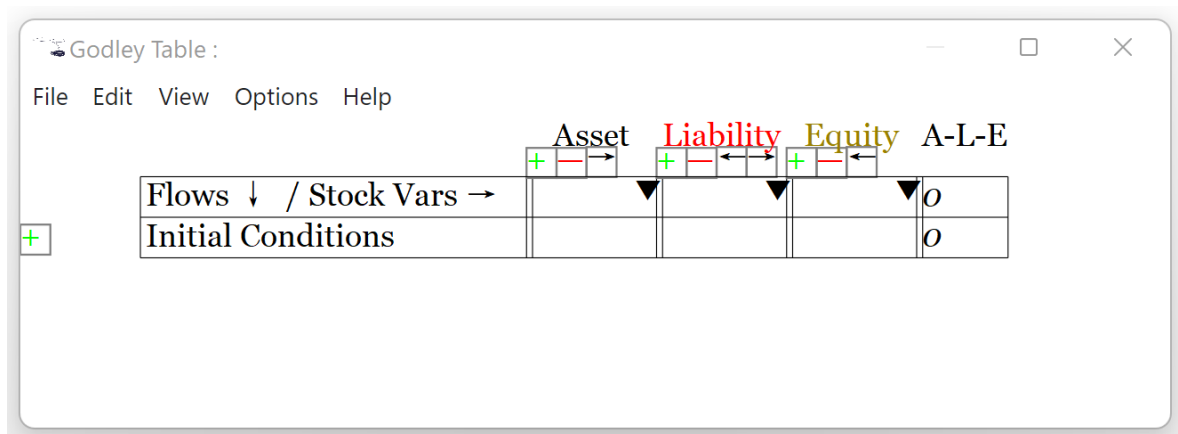
6 Godley Tables

Godley Tables were primarily designed to enable the easy modelling of monetary dynamics. The examples in this chapter focus on the macroeconomics of money, but Godley Tables can be used to model the internal accounting of a corporation, its financial interactions with suppliers, etc. They can be used to model *any process in which the entities being modelled are in one exclusive state or another*—such as a pandemic in which people either have the disease or not, have recovered or not, died or not, etc., components passing through stages of a manufacturing process, and so on.

To insert a Godley Table into a model, click on the Godley Table Icon  on the Widget bar. The icon will then attach to your mouse. Move to where you want to place it on the canvas and click the right mouse button again and the icon will be placed there.

To edit the Godley Table, either double click on the icon, or click on the right mouse button and choose “Open Godley Table” from the drop-down menu. When you do, you will see Figure 17:

Figure 17: Godley Table



The menu bar at the top controls exporting a Godley Table as CSV or LaTeX file for documentation purposes, editing (cut/copy/paste etc.), zooming in and out, useful options (whether to show the values of stocks and flows during a simulation, whether to use +/- or the accountants CR/DR, and whether to have a single or multiple Equity columns), and Help on using a Godley Table.

To use a Godley Table, you need to enter names for the Assets, Liabilities and Equity columns in it, enter initial conditions, and then record financial transactions on its rows. The default table has room for one Asset, Liability and Equity column.

Additional columns can be added by clicking on the relevant + button below the Asset, Liability and Equity headings; columns are deleted using the – button, and moved left or right using the ← and → buttons.

Rows for financial transactions are added by clicking on the + button next to “Initial Conditions”. Figure 18 shows a simple financial model with a single Asset (Loans), a single Liability (Deposits), and the Banking sector’s Equity (the subscripted text is created by preceding the word with an underscore _ and enclosing the text in curly brackets {} (Banks_{Equity})).

Figure 18: A Godley Table with Stocks created, initial conditions recorded, and a first row for a financial transaction added

Godley Table :

File Edit View Options Help

	Asset	Liability	Equity	A-L-E
Flows ↓ / Stock Vars →	Loans ▼	Deposits ▼	BanksEquity ▼	O
Initial Conditions	900	800	100	0
New Bank Loans	Credit	Credit		0

Flows are entered as words rather than numbers: numerical values for flows are defined on the canvas itself. Two more operations are needed for a complete model, and these are shown in Figure 19 (the Title **Banking Sector** was added using the Title command from the Edit menu, and this image was exported as an SVG file from the *Godleys Tab*).

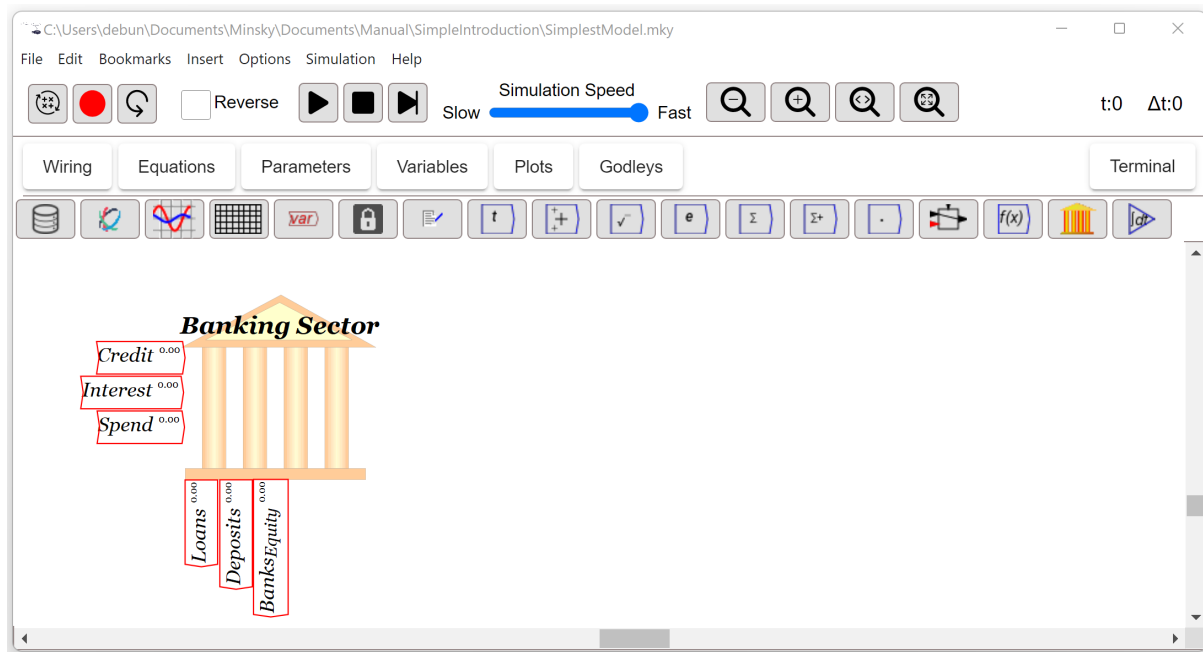
Figure 19: The simplest possible model of a monetary economy

Banking Sector

	Asset	Liability	Equity	A-L-E
Flows ↓ / Stock Vars →	Loans ▼	Deposits ▼	BanksEquity ▼	O
Initial Conditions	900	800	100	0
New Bank Loans	Credit	Credit		0
Pay Interest		-Interest	Interest	0
Banking sector spending		Spend	-Spend	0

To complete this model, the flows *Credit*, *Interest* and *Spend* have to be defined. The pre-requisite to doing this is to place these variables on the canvas. After the Godley Table in Figure 19 has been defined, the canvas will look like Figure 20.

Figure 20: The canvas after the Godley Table in Figure 19 has been defined



The stocks and flows in a Godley Table can be placed on the canvas in three ways:

- All at once, by clicking on the right mouse button while hovering over the Godley icon and choosing “Copy Stock Variables” and “Copy Flow Variables”. This will attach all Stocks and all Flows respectively to the mouse cursor; click anywhere on the canvas and they will be placed there;
- One at a time, by hovering over a flow (shown on the left-hand side of the icon) or stock (shown below the icon) and choosing “Copy”; and
- By clicking on the desired stock or flow in the browser window (see Figure 21) and then clicking anywhere on the canvas.

Figure 21: The canvas with the browser window beside it

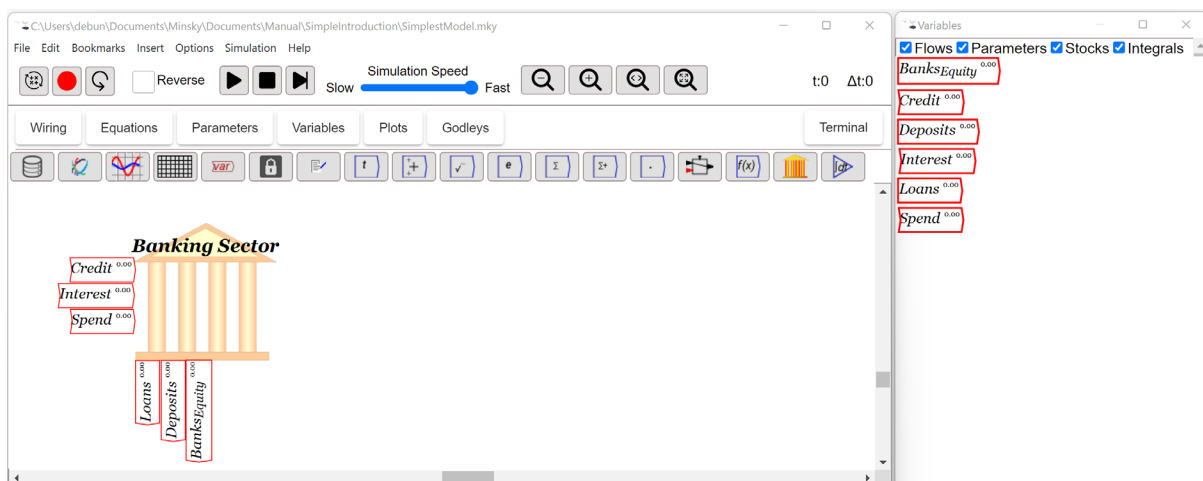
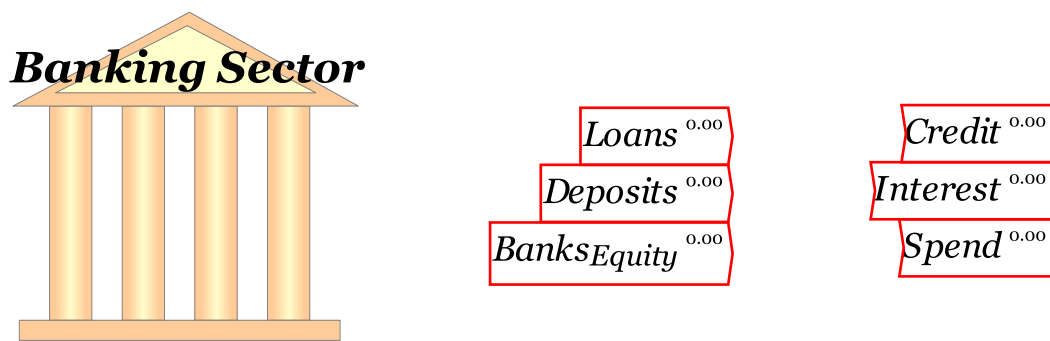


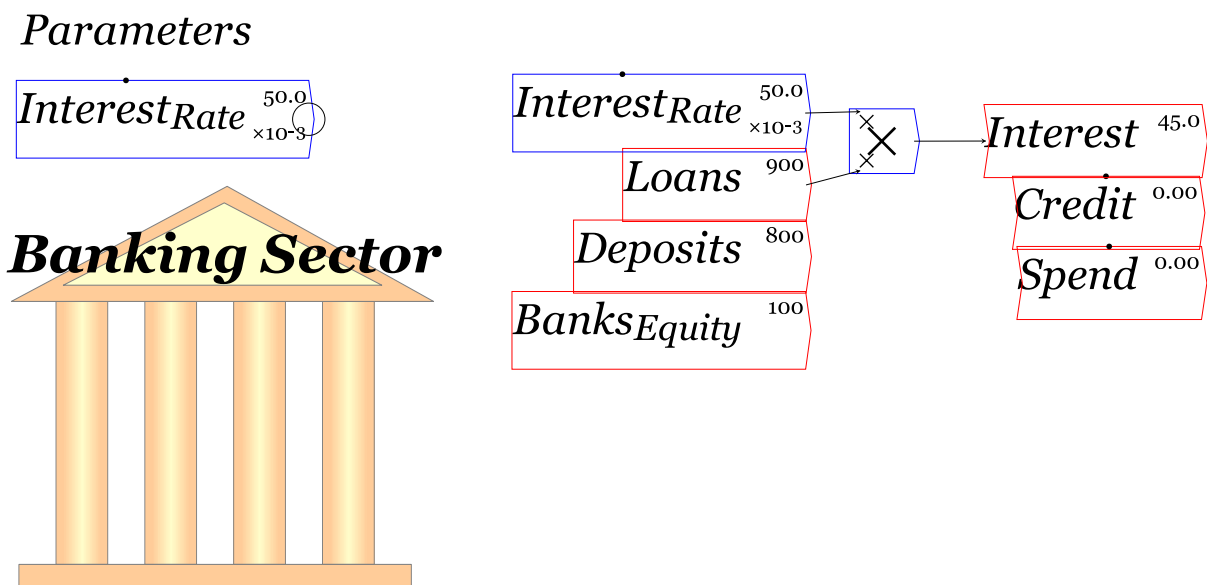
Figure 22 shows the stocks and flows on the canvas, where you can give them definitions.

Figure 22: Stocks and Flows placed on the canvas



The most obvious is the definition of the flow *Interest*, which will be equal to the rate of interest on loans times *Loans*. Therefore, you need to define the rate of interest *InterestRate*. In a simple model, this will be a *parameter*: something set by the modeler rather than determined by interactions with other entities in the model as are Stocks and Flows.

Click where you'd like to place *InterestRate* on the canvas—above the stock *Loans* makes sense. Then type a multiply key (the * above 8 on the numeric keypad) and a multiply block will appear on the canvas. Wire *Loans* and *InterestRate* to the input ports of the multiply block, and wire the output to *Interest*, and you've defined the first equation in this model.

Figure 23: The model with the parameter *InterestRate* created and wired to calculate *Interest*

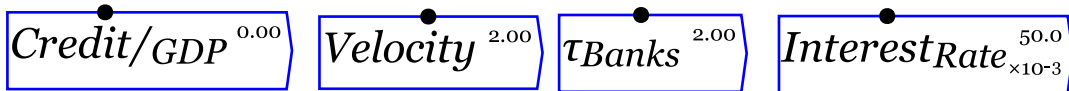
Defining the other elements of the model requires a little more thought. The easiest way to create a basic economic model here is to define another variable *Money*, as the sum of *Deposits* and *BanksEquity*,² and then introduce a parameter for how rapidly it turns over (Velocity). *Velocity* is a parameter in this simplest of models, and *Velocity* times *Money* equals *GDP*. Then I define *Credit* as a percentage of *GDP*. Finally, *Spend* is related to the amount of money in the Banking Sector's (short-term) Equity (*BanksEquity*), using what engineers call a "[time constant](#)"—signified here by τ_{Banks} (which is entered as `\tau_{Banks}` in *Minsky*). This is a measure, expressed in years, of how long banks could take to spend their equity down to zero if they maintained a constant rate of spending

² This is short-term equity rather than long-term. Minsky can enable multiple Equity columns to differentiate short-term from long-term equity.

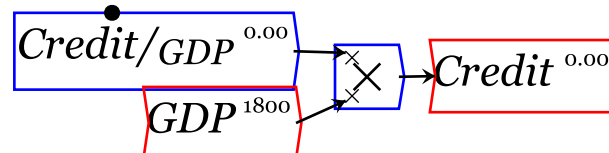
and there were no further inflows. The completed model, though without any plots to show its dynamics, is shown in Figure 24.

Figure 24: Completed final model, without plots

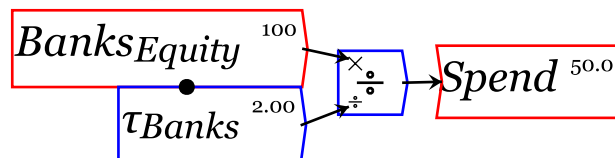
Control parameters



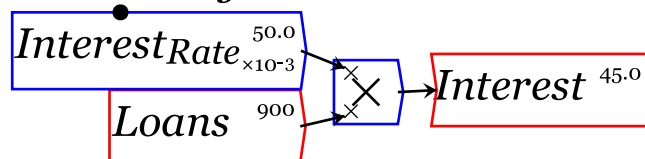
Credit money creation



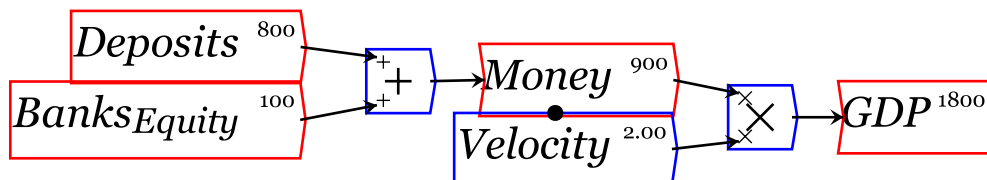
Bank Spending



Interest Payments



Defining money and GDP

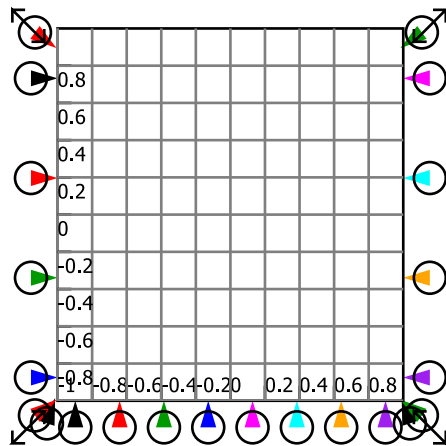


7 Plots and sliders

Unlike the majority of system dynamics programs, Minsky enables the simulation itself to be dynamic: graphs are updated dynamically, and the model's parameters can be varied as a simulation runs to see what happens. This is the reason that I placed Figure 24's parameters at the top of the diagram: once plots are added, we will vary the parameters during a simulation.

Plots are added to the canvas using either the plot widget on the widget bar, or by pressing the @ key while on the Wiring canvas. This will create a default plot as shown in Figure 27.

Figure 25: The default Plot, with its stretch arrows highlighted

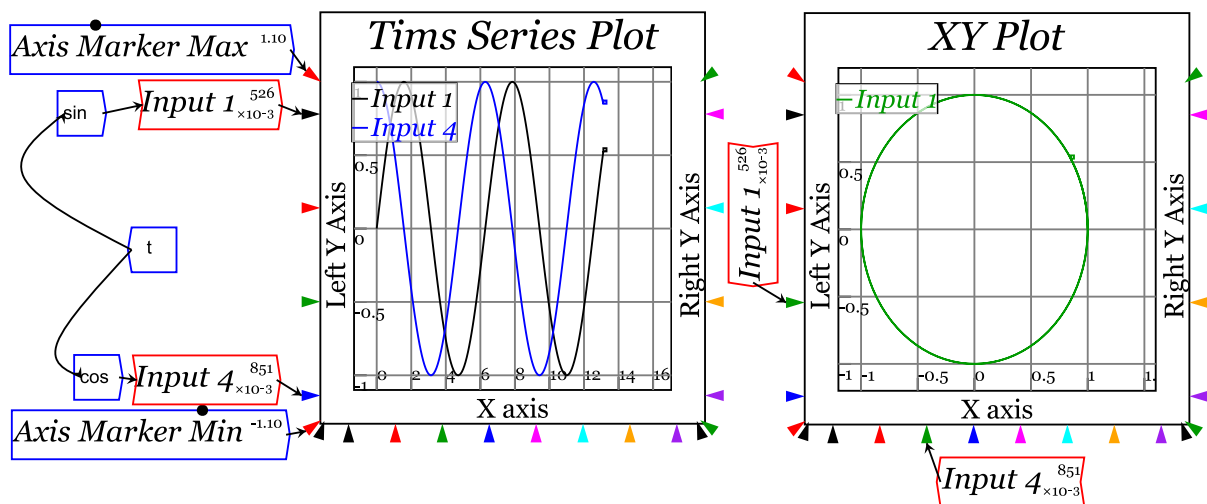


The key features of a plot are:

- The input ports: 4 on each Y axis and 8 on the X-axis;
 - The Y-axis ports support up to 4 inputs on each Y-axis;
 - The 8 input ports on the X-axis enable XY plots—up to 8 per plot. With nothing wired to the X-axis, the default input is simulation time, so that the plot is a time series graph. With an input on a given color on a Y-axis and the same on the X-axis, an XY-plot is created;
- The axis max/min entries—shown as slightly tilted inputs with the same color on each axis (Red on the Y1 axis, Black on the X, Green on the Y2 axis).

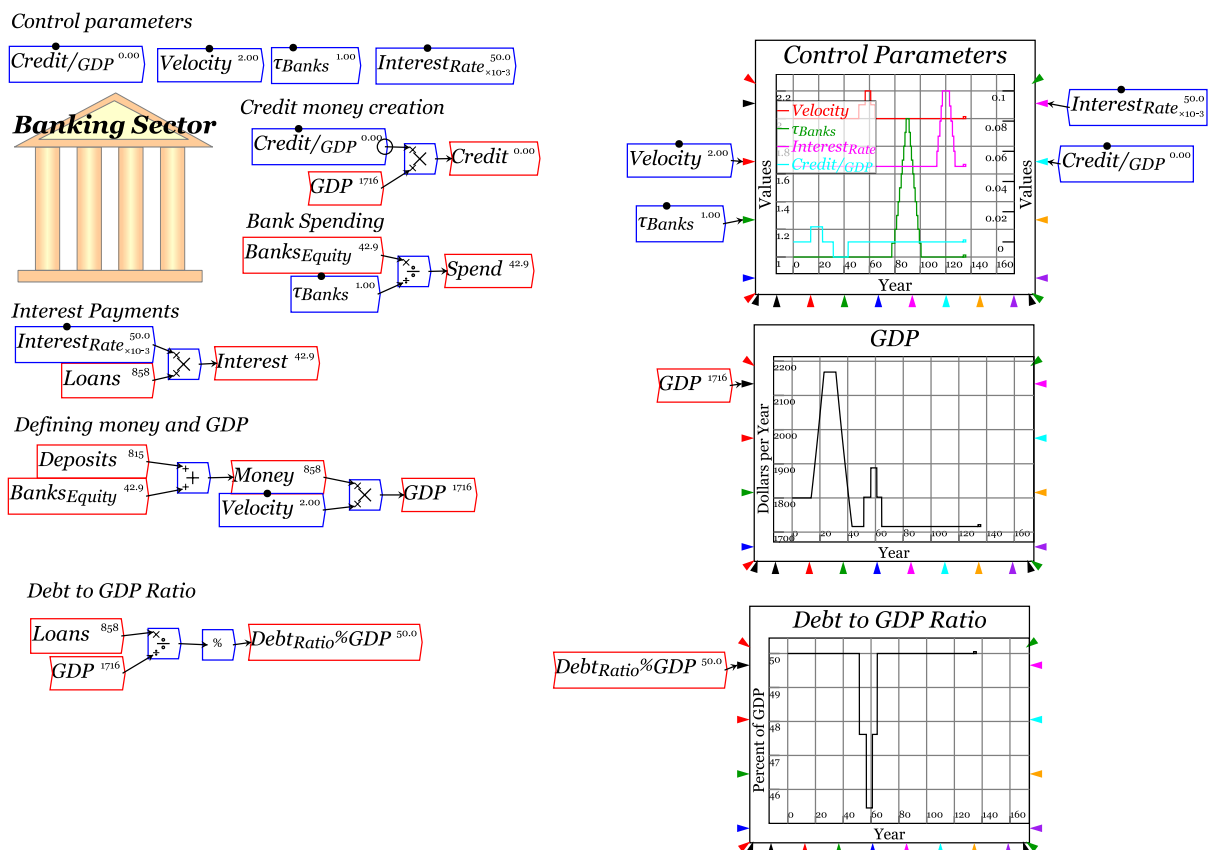
Figure 26 shows how to use these features (there will be considerable improvements to the appearance and controls for plots in future releases).

Figure 26: Illustrating the elements of a plot



Variable parameters and dynamically updated plots enable a “what if?” approach to modelling. Figure 27 illustrates this by varying all 4 parameters at different times, and seeing what the impact was. Briefly, increasing credit as a percentage of GDP caused the money supply to grow and the economy to expand; increasing velocity caused the debt to GDP ratio to fall; the other two controls had largely no impact.

Figure 27: Simple model with plots and varying parameters



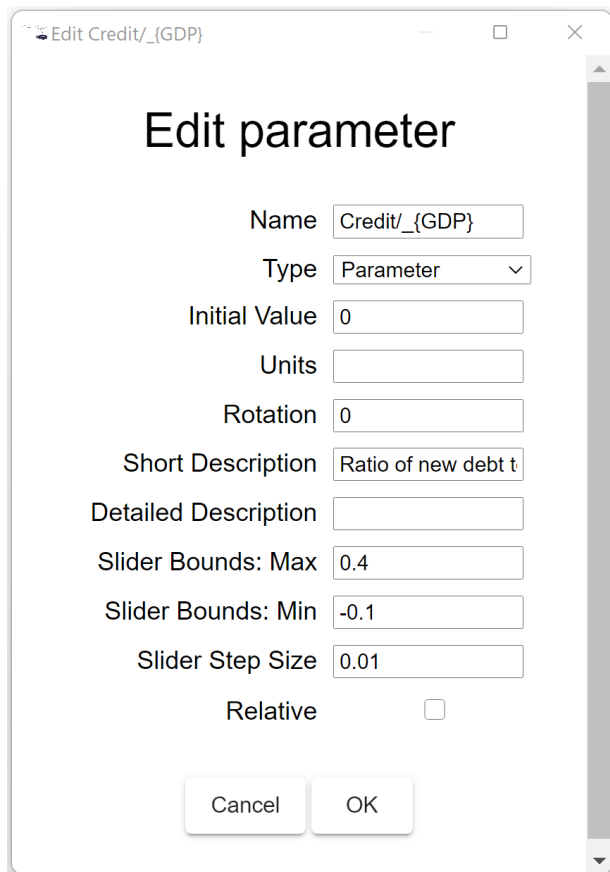
Parameter values can be altered in two ways:

1. By clicking on the black dot on the top of a parameter and dragging it right (to increase its value) or left (to decrease it); and

2. Pressing the up-arrow or right-arrow (to increase its value) or down-arrow or left-arrow (to decrease its value) when the mouse is hovering over the parameter.

The range of movement of the parameter's value is set on the parameter definition popup—see Figure 28. *Slider Bounds: Max* and *Slider Bounds: Min* determine the parameter's upper and lower bounds, while *Slider Step Size* determines how much the parameter's value varies for each mouse movement or key press.

Figure 28: The Edit parameter popup



The screenshot shows a window titled "Edit Credit/_{GDP}" with a close button. The main heading is "Edit parameter". The form contains the following fields and controls:

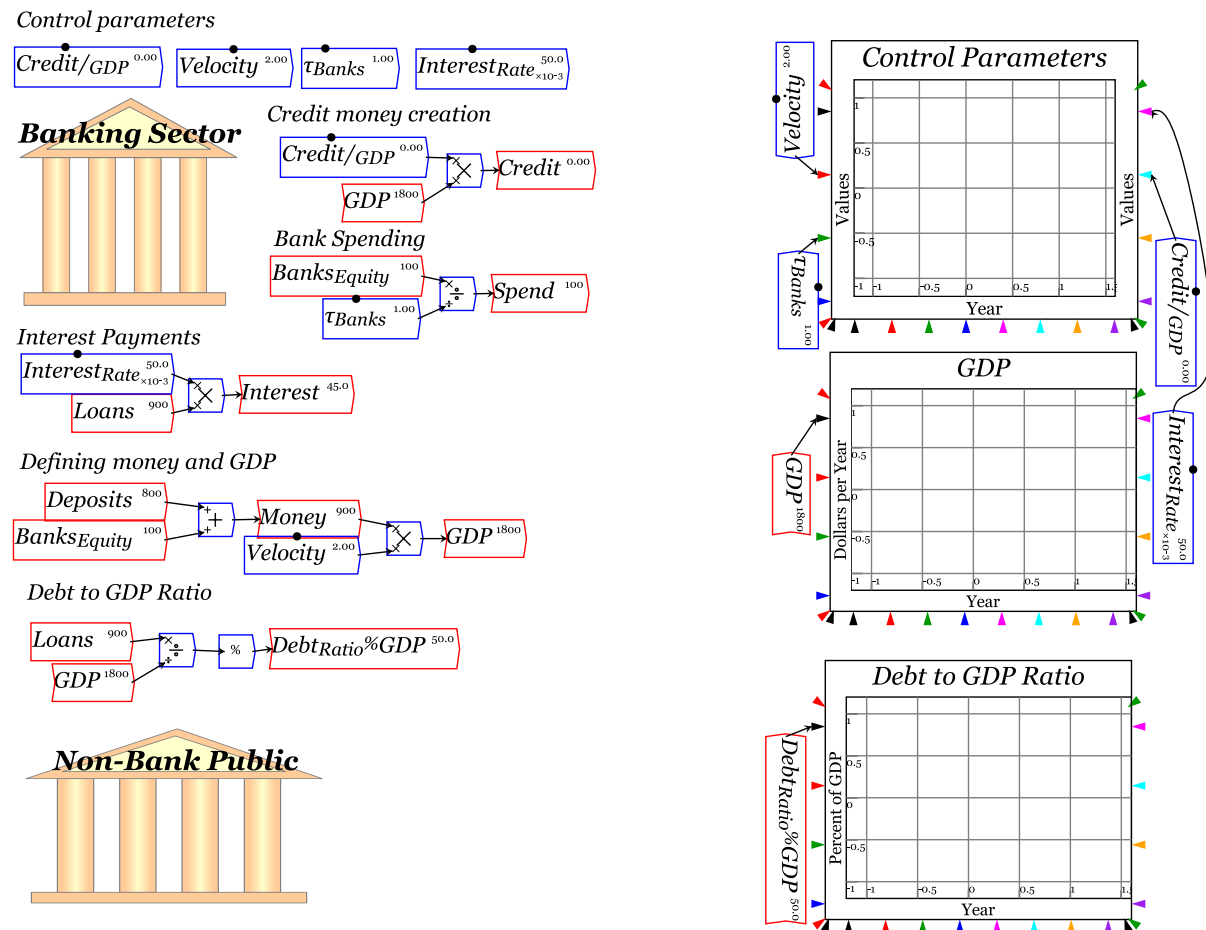
- Name:
- Type:
- Initial Value:
- Units:
- Rotation:
- Short Description:
- Detailed Description:
- Slider Bounds: Max:
- Slider Bounds: Min:
- Slider Step Size:
- Relative: ☐

At the bottom are two buttons: "Cancel" and "OK".

8 Integrated general models of financial flows

The model in Figure 27 is not quite complete, because it shows the system from the banking sector's perspective, but not their customers perspective—the “Non-Bank Public”. This is easily corrected by adding another Godley Table—see Figure 29—and then using Minsky's capacity to keep track of the rule that “one entity's Asset is another's Liability”.³

Figure 29: Simplest model with a second Godley Table added

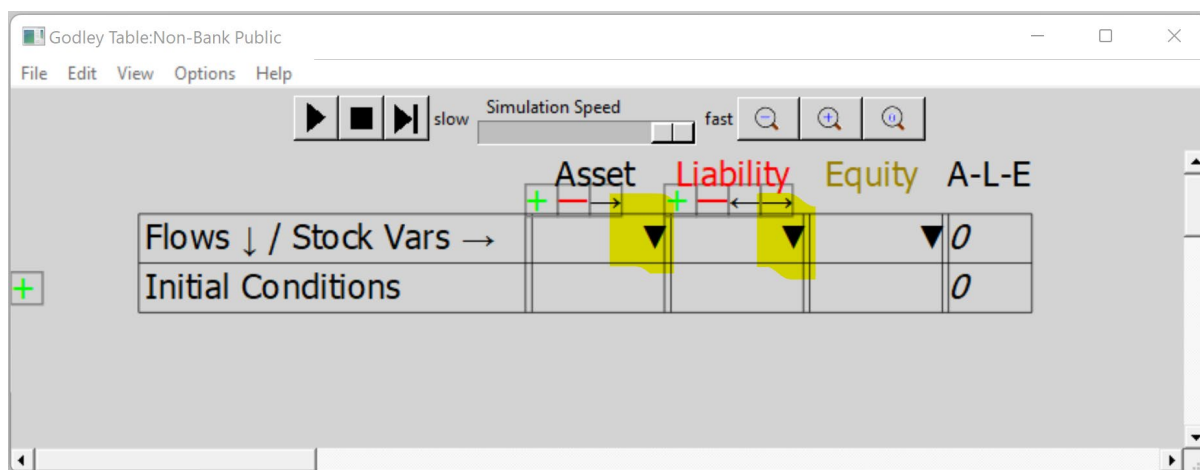


This is where the wedge you may have noticed in previous Godley Tables comes in—see Figure 30 where they are highlighted. If you click on the one below Asset, it will show you all Liabilities that have not yet been allocated as another entity's Liability; the one below Liability does likewise about Assets.⁴

³ This is for financial assets, which are claims on another entity. There are also nonfinancial assets, which are an Asset to their owners and a Liability to no-one. Minsky handles this via the device of multiple Equity columns, which enables nonfinancial assets to be shown as an Asset and an Equity of the owner.

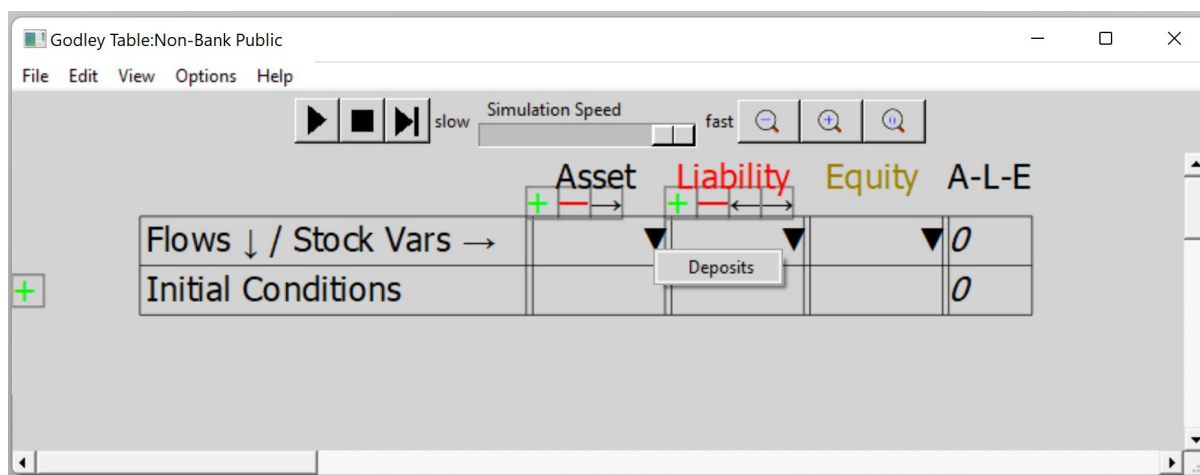
⁴ When multiple Equity columns are enabled, the Equity dropdown will also show Assets of the same entity that have not yet been allocated as Liabilities of another entity—this enables the modelling of nonfinancial assets.

Figure 30: The matching Asset-Liability wedges in a Godley Table



From the Banking Sector's perspective, there is only one Asset—*Loans*—and only one Liability—*Deposits*. When you click on the wedge for Assets, the Banking Sector's Liability of *Deposits* will turn up as a potential Asset for the Non-Bank Public (*NBP* in what follows)—See Figure 31.

Figure 31: The wedge for Assets clicked and the Banking Sector's Liability of Deposits showing



Click on *Deposits*, and all the operations that affect *Deposits* on the Bank's Godley Table are replicated here. Do the same for the Liabilities column, and you will see Figure 32.

Figure 32: The Non-Bank Public's Godley Table, with some details not yet completed

	Asset	Liability	Equity	A-L-E
Flows ↓ / Stock Vars →	$\text{Deposits} = 800$	$\text{Loans} = 900$		0
Initial Conditions	800	900		-100
New Bank Loans	Credit = 0.00	Credit = 0.00		0
Pay Interest	-Interest = -45.0			-Interest
Banking sector spending	Spend = 100			Spend

There are several very important aspects to this table, which will turn up in any more elaborate model that you create:

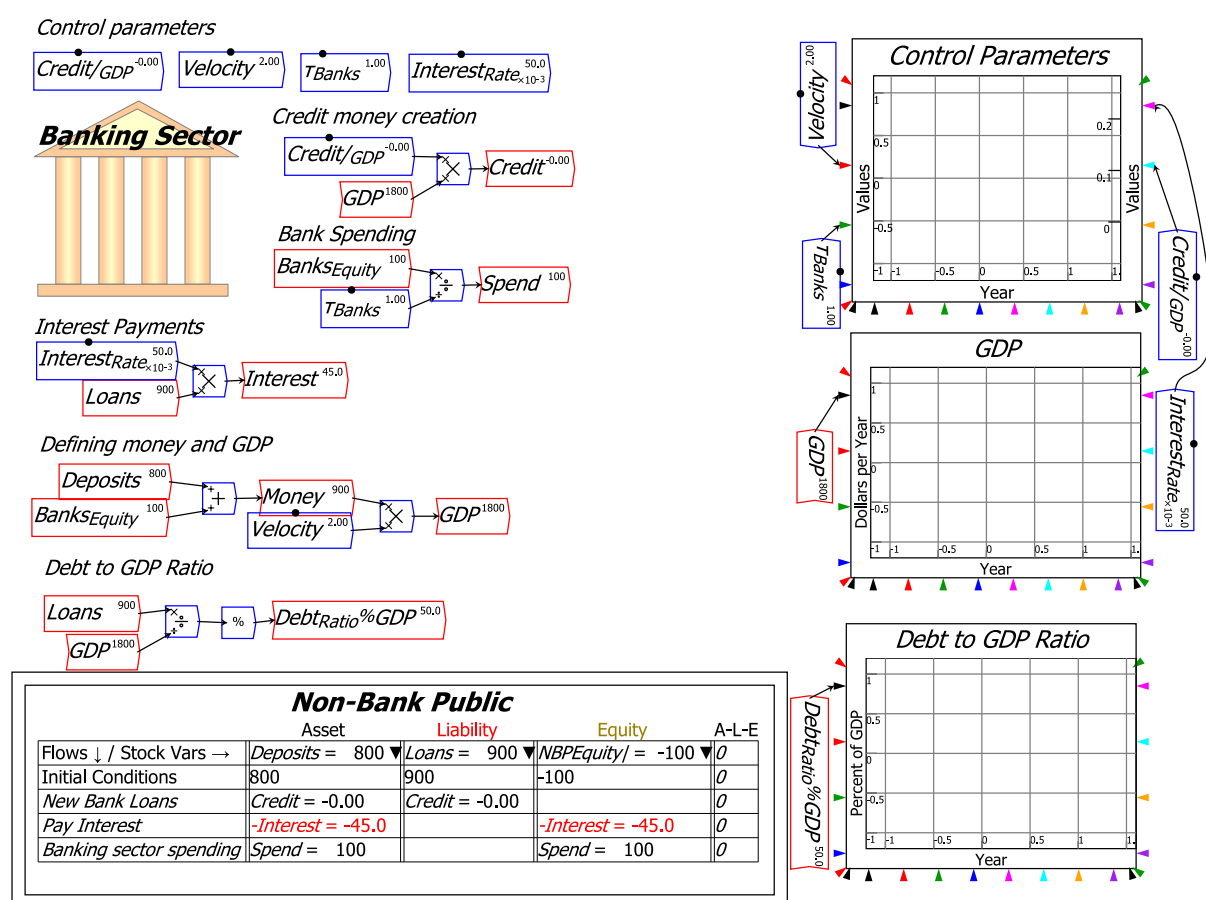
1. The value of Deposits (800) is less than the value of Loans (900), so that the *NBP* is in negative financial equity—and this is precisely the same magnitude as the positive equity of

the Banking Sector. *This is a necessity.* Given that a *financial* Asset is a claim on another entity, the sum of all financial assets is zero;⁵ and

- Operations which increased the Equity of the other entity—the Banking Sector in this case—are incomplete on this table and therefore show up in the $A - L - E$ checksum. We don't automatically allocate them to the Equity column of this entity because of the possibility of multiple Equity columns (we may alter this in a later release).

The model is complete once you have entered those operations yourself in the NBP's Equity column: see Figure 33, where I have also chosen "Editor Mode" for the display of the NBP's Godley Table, and enabled the display of the numerical values of its flows and stocks as the model is simulated.

Figure 33: The completed model with two Godley Tables



Notice that despite its negative equity position, the Non-Bank Public's initial position is that its income exceeds its expenditure: interest payments of 45 per year are less than the Banking Sector's Spend of 100 per year. This, however, necessarily means that the Banking Sector is losing equity at the same rate—this is not an equilibrium position (in more complex models, non-equilibrium positions can continue indefinitely, because nonlinear feedbacks make the equilibrium unstable, while also ensuring that the model itself doesn't "break down" by generating unrealistic values).

Much more elaborate models than this can be built with numerous entities, and Minsky will keep track of the flows of money between them, no matter how complicated the model becomes. This is Minsky's primary advantage over all other system dynamics programs for modelling financial

⁵ Nonfinancial assets enable aggregate positive equity.

dynamics, and any process in which there are exclusive categories—such as process control in a factory, or a pandemic, or modelling the diffusion of a new product in product marketing.

There are also processes which don't have exclusive categories, and for these the classic system-dynamics “flowchart” paradigm is appropriate.

9 Flowchart modelling

The standard system dynamics paradigm is a flowchart: flow inputs are aggregated into a stock, with wires connecting the flows and the stocks. *Minsky* supports this paradigm as well, but with a few differences to the industry standard:

- Our flowchart elements are modelled on the relevant mathematical operations—integration and differentiation—rather than being shown as “taps and bathtubs”, as in several leading programs;
- All our equations are created graphically, and are visible on the canvas (unless they are in a group).
 - Standard programs enter equations as lines of text, and they are stored inside dialog boxes which sit behind the graphical representation of variables on the canvas; and
- Parameters and variables can be “passed by name” as well as “passed by wire”, resulting in much less of a “spaghetti diagram” issue with *Minsky*.
 - Standard system dynamics programs require every entity that is used in defining another entity to be wired to it, and they allow only one instance of each entity on the canvas. This results in a lot of wires!
 - *Minsky* cuts this down dramatically by enabling multiple copies of an entity on a diagram, and it can be wired to only the entity it is used to define at that point. Consequently, Minsky diagrams look more like lots of small cobwebs, rather than one enormous bowl of spaghetti.

This gives Minsky a more compact and modular feeling than standard system dynamics programs. But it also requires adjustments for experienced system dynamics modellers. Figure 34 shows the conventional “inflows→Stock→Outflows approach:

Figure 34: Standard system dynamics display of flows and stocks (<https://ccmodelingsystems.com/how-to-read-a-stella-model-diagram/>)

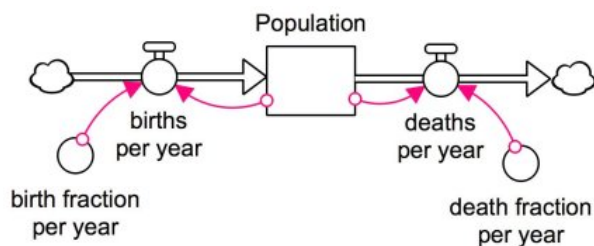
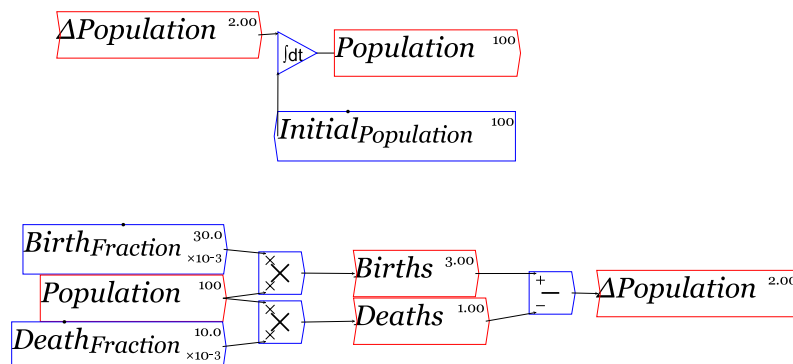


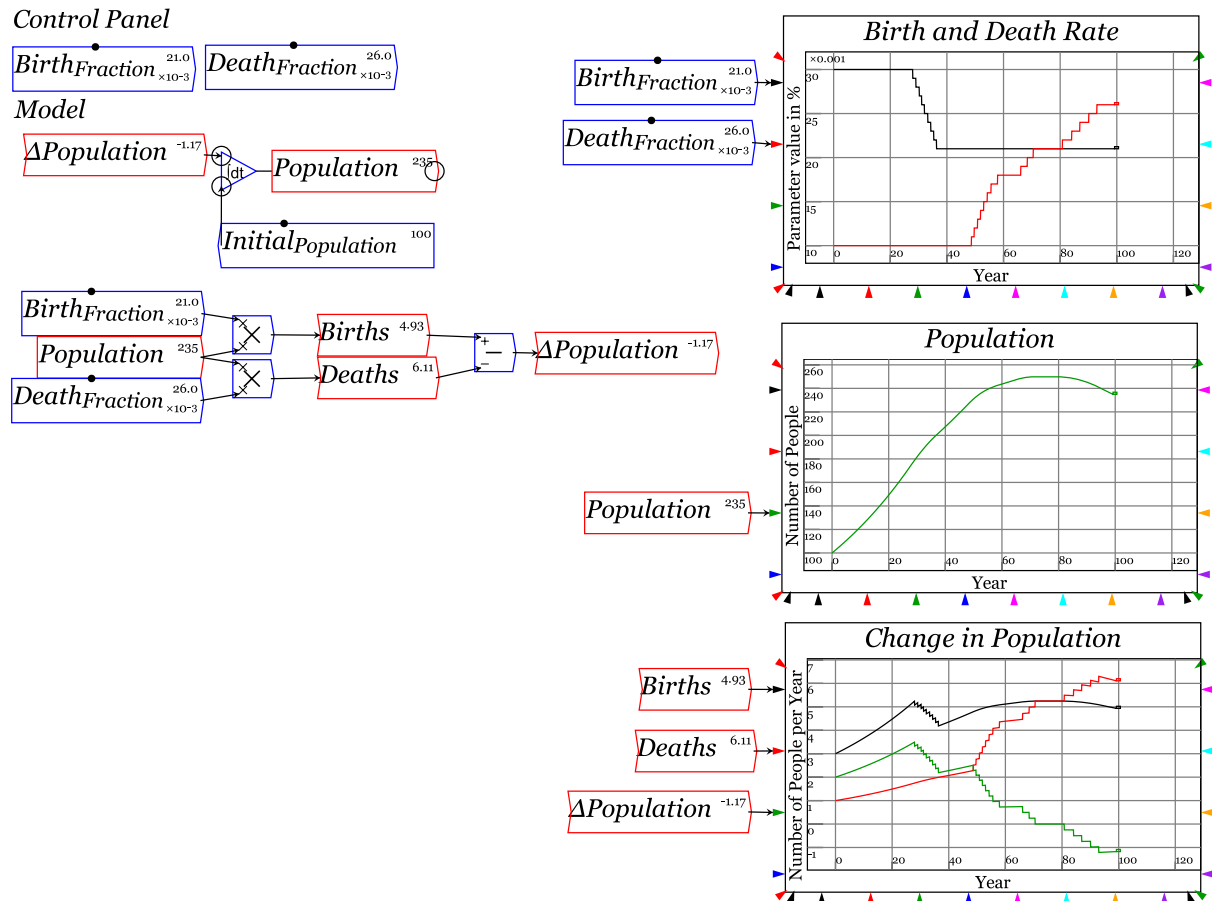
Figure 35 shows the same model in *Minsky* (our symbol for Stocks is the integral sign as used by mathematicians).

Figure 35: Minsky's approach to Stocks and Flows



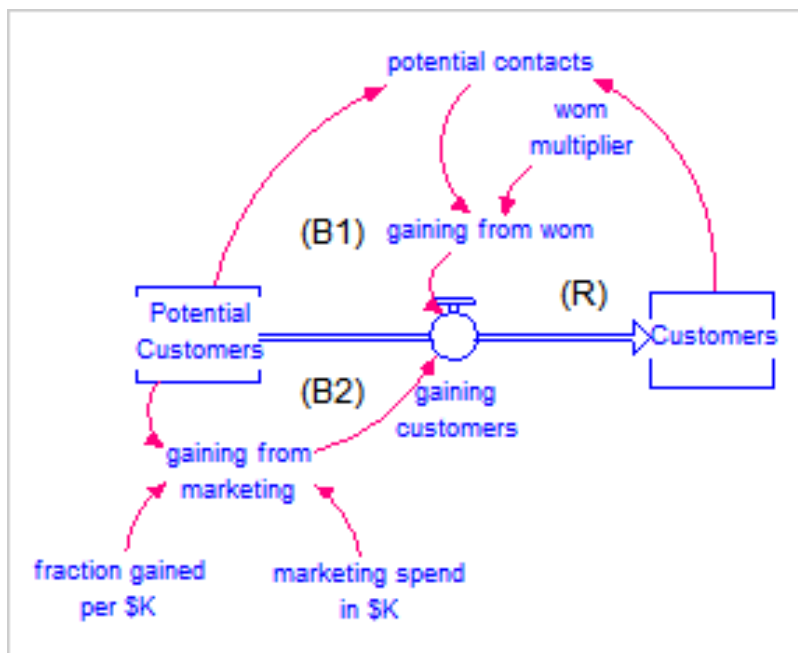
There are strengths and weaknesses to both approaches, but we prefer ours because (a) the equations for births and deaths per year are visible on the canvas (though you can conceal them in a group if you wish), and (b) the simulation values are shown dynamically as the model runs—see Figure 36.

Figure 36: Varying population outcome by varying birth & death parameters during a simulation



The stylized [dynamics of the introduction of a new product to the marketplace](#) is a classic system dynamics model. Figure 37 shows a slight generalization of the original Bass model of product diffusion (named after its developer Frank Bass).

Figure 37: A Stella model of a product diffusion model



The model divides the population into two types: innovators and imitators. Innovators try out new things—and hence are the potential first buyers of a new product. Imitators do what other people are doing, so they don't adopt a new product until they see it being used by others.

Innovators have an innate tendency to try something new, and the equation describing their movement from non-adopters to adopters is the percentage of current non-adopters that are innovators, times the number of non-adopters:

$$\text{Innovators} = \text{Innovator}_{\text{Fraction}} \times \text{PotentialAdopters} \quad (2)$$

Imitators don't have an innate tendency to try something new, but the *probability* of them trying something depends on the number of people that they see using it, times their tendency to imitate, times the fraction of the population that are still non-adopters:

$$\text{Imitators} = \text{Fraction}_{\text{Non-adopters}} \times \text{Adopters} \times \text{Imitation}_{\text{Tendency}} \quad (3)$$

The fraction of potential adopters is the potential adopters divided by the population:

$$\text{Fraction}_{\text{Non-adopters}} = \frac{\text{PotentialAdopters}}{\text{Population}} \quad (4)$$

The total number of adopters at any point in time is the sum of Innovators plus Imitators:

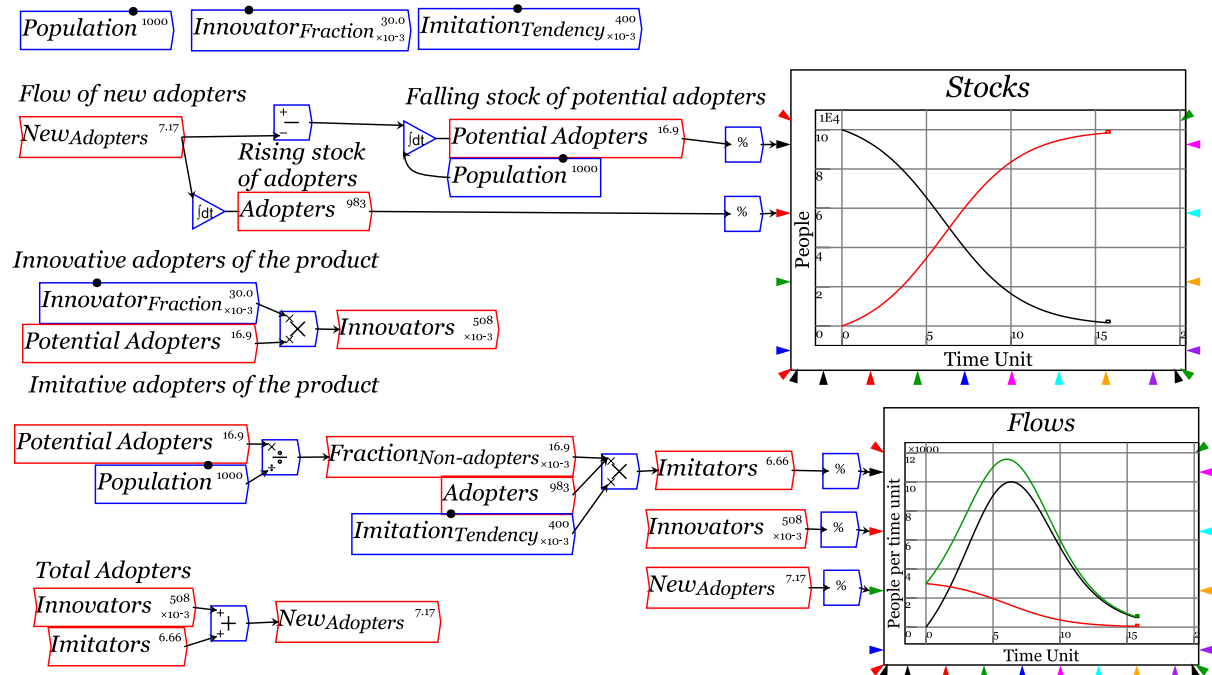
$$\text{New}_{\text{Adopters}} = \text{Innovators} + \text{Imitators} \quad (5)$$

The two stocks are the Potential Adopters and the Adopters, and the increase of the former is identical to the fall in the latter:

$$\frac{d\text{Adopters}}{dt} = \text{New}_{\text{Adopters}}$$

$$\frac{d\text{PotentialAdopters}}{dt} = -\text{New}_{\text{Adopters}} \quad (6)$$

Figure 38: The basic Bass model of product diffusion in Minsky using its flowchart technology



The model shows the importance of what we today call “Influencers”—as well as the importance of their imitators. Without influencers, nothing happens (set *InnovatorFraction* to zero in the above and run the model). With no imitators and Influencers being just 3% of the population (as in the simulation in Figure 38), it takes over a century for the product to diffuse through the entire population (set *ImitationTendency* to zero). With imitators, it takes just 5 years (given the parameters in Figure 38).

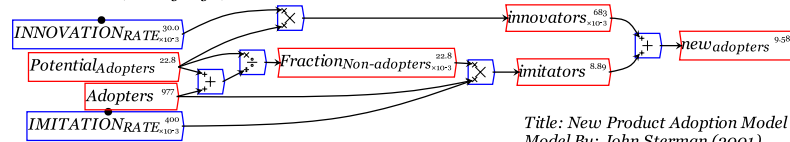
This model can also be constructed using a Godley Table. even though the flows are of people rather than money, because the two states—Potential Adopters and Adopters—are mutually exclusive categories. Figure 39 shows the model done this way. The advantages are (a) that it’s easier to read, since there are no differential equations shown (the Godley Table generates them automatically); and (b) that it’s easier to construct, especially as more factors are added, since the Godley Table automatically makes sure that the flows out of one category are balanced by flows into the other.

Figure 39: New product diffusion model built using a Godley Table

STOCK & FLOW (Godley Table)

	Asset		Liability	Equity	A-L-E
Flows ↓ / Stock Vars →	PotentialAdopters = 22.8	Adopters = 977		Balance = 1000	0
Initial Conditions	1000	0		1000	0
new adopters	newadopters = -9.58	newadopters = 9.58			0

FLOWCHART (Minsky Style)



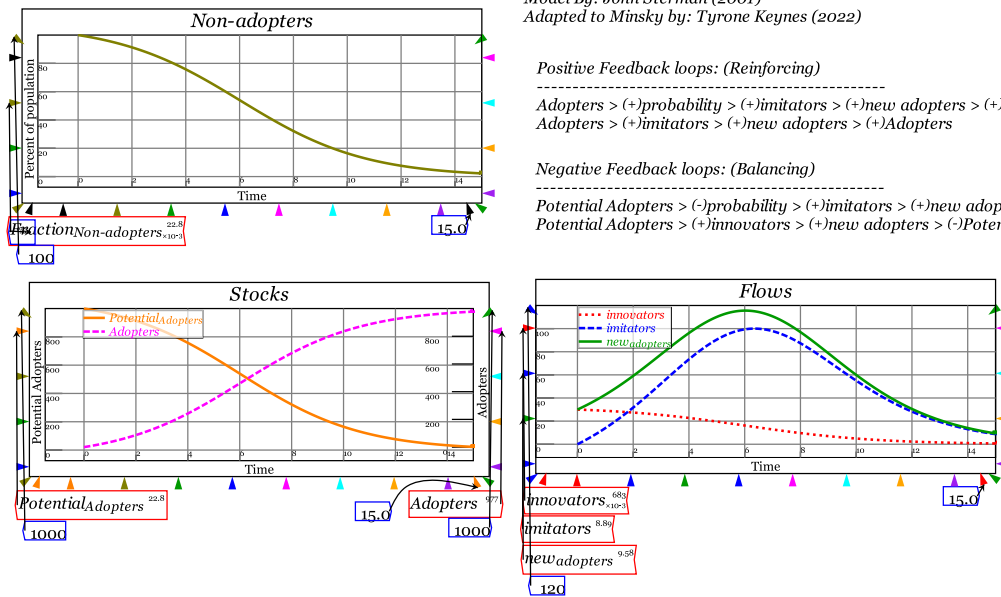
Title: New Product Adoption Model
Model By: John Sterman (2001)
Adapted to Minsky by: Tyrone Keynes (2022)

Positive Feedback loops: (Reinforcing)

Adopters > (+)probability > (+)imitators > (+)new adopters > (+)Adopters
Adopters > (+)imitators > (+)new adopters > (+)Adopters

Negative Feedback loops: (Balancing)

Potential Adopters > (-)probability > (+)imitators > (+)new adopters > (-)Potential Adopters
Potential Adopters > (+)innovators > (+)new adopters > (-)Potential Adopters



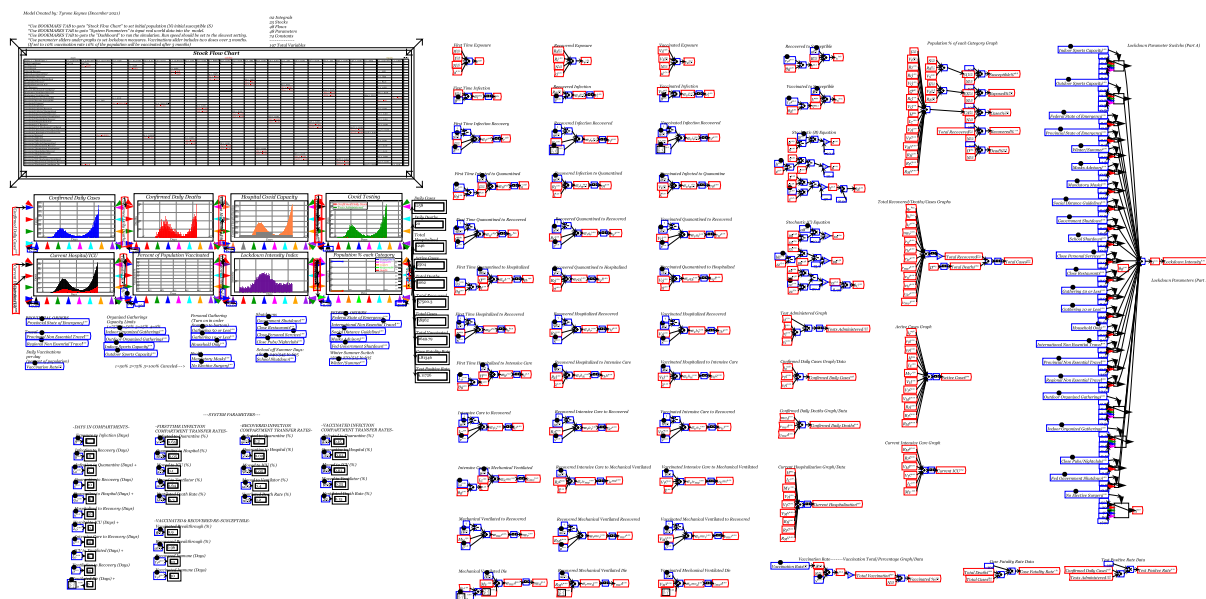
Given the built-in stock-flow accounting performed by the Godley Table, it's worth considering whether you can use a Godley Table in your model before you start to construct it. The obvious rule is that where the flow out of one system state is the same as the flow into another, a Godley Table version is possible. One case where this does not apply is the "Predator-Prey" model, shown in Figure 3 on page 5: the negative for Fish (the fall in their number from predation by Sharks) is not the same as the positive for Sharks.

10 Organizing a model: Bookmarks

The models shown here to date are quite small, and easy to navigate on a single PC screen—especially with judicious use of the Zoom “magnifying glasses” on the control bar. But Minsky can support enormous models which are far too large to take in on just one screen.

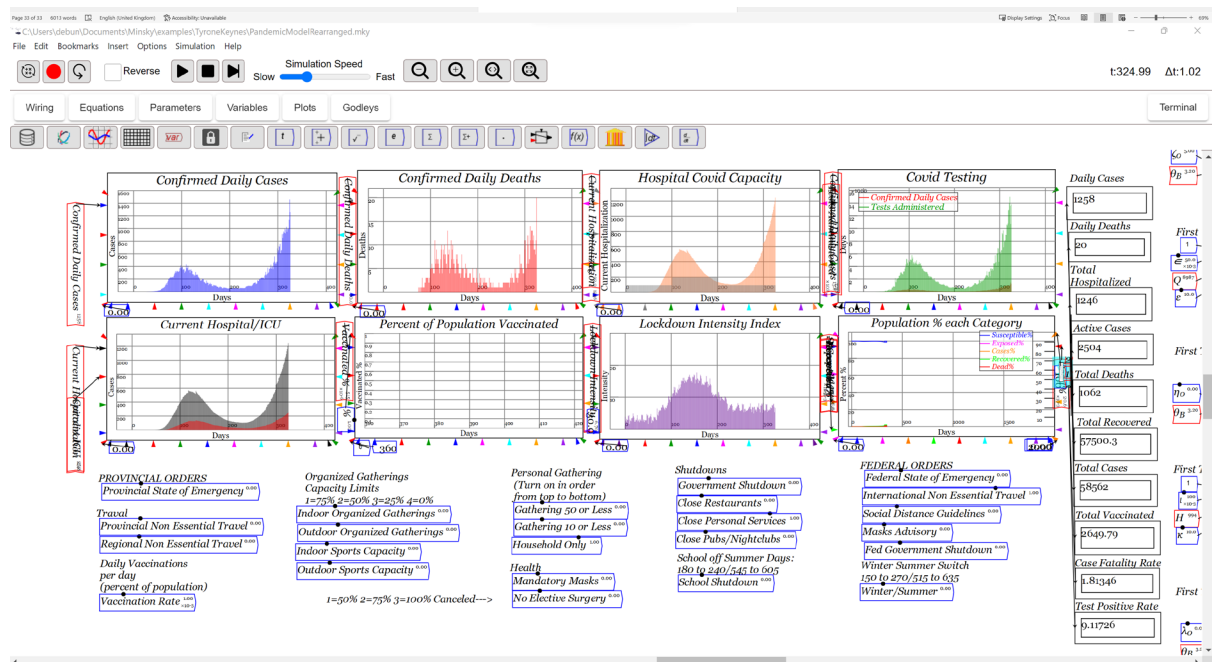
One such example is the model of the pandemic, developed by Minsky evangelist Tyrone Keynes (do subscribe to his YouTube Channel [Modelling with Minsky](#)). This is a “SEIRD” model, where the acronym stands for “Susceptible→Exposed→Infected→Recovered or Dead”. Most such models of a pandemic have just those five system states, or at most ten, because it is impossible to keep up with interactions between so many system states using the flowchart paradigm. Minsky’s Godley Tables makes it straightforward, but the problem then becomes navigating such a large model.

Figure 40: Tyrone Keynes's 25-state pandemic model



Bookmarks—which store both a location and the zoom level for the canvas—are our solution. Clicking on a Bookmark changes the view of the model so that the bookmark is at the top left-hand corner of the screen, and the zoom level is set so that the desired subset of the model fits within the screen itself. Figure 41 shows the Dashboard bookmark for this model; it is designed to enable policymakers to try out various policies in an artificial society before trying them on the real one. At this scale, you can easily alter some settings—banning international travel for example—and see its impact on the spread of the pandemic.

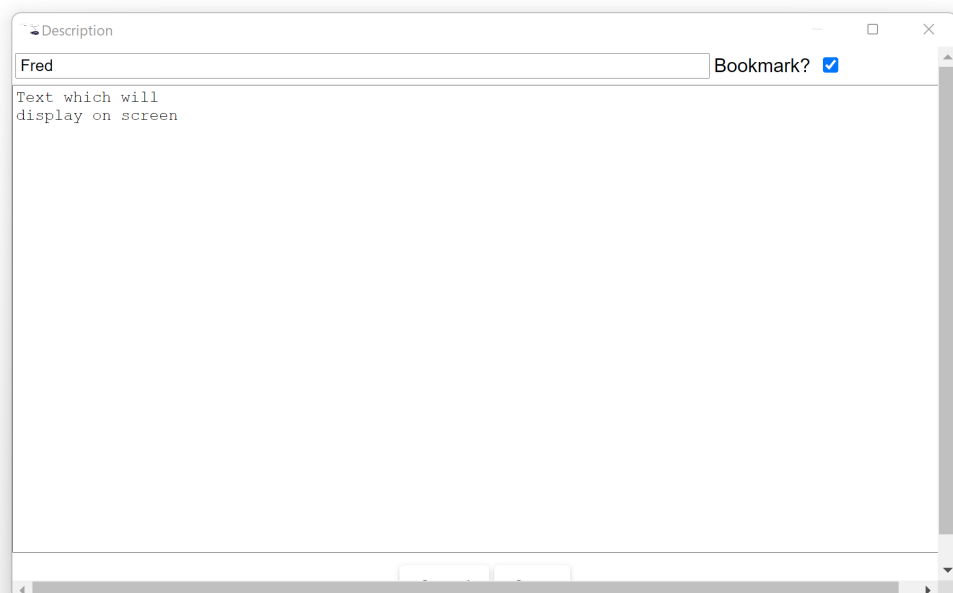
Figure 41: The Dashboard Bookmark for Tyrone Keynes's Pandemic Model



Bookmarks can be created in two ways:

1. By scrolling until the top left-hand corner shows the part of the model you wish to display, and the Zoom level is set so that the bottom right-hand corner encapsulates the region you wish to display, and then choosing "Bookmark this position" from the Bookmarks menu; and
2. By creating a Note, and checking the "Bookmark?" Checkbox. A Bookmark will be created using the Short Description text of the Note, and the text typed into the text box will display on screen.

Text which will
display on screen



11 Organizing a model: Groups

Another way to organize a model is to place subsets of the model into Groups. *Minsky* supports grouping, but there are limitations in our current implementation of it: firstly, grouped variables are local to the group; secondly, creating inputs to and outputs from a group creates numbered input and output ports which clutter the model.

Figure 42 shows the strengths of grouping in *Minsky*:

- One function can be used more than once in a model. In the top group of the figure, the parameters suit a nonlinear “Phillips Curve”, where the inputs are the employment rate that leads to no change to the wage level, the slope of the function at that point, and the minimum annual change in wages; in the bottom, the inputs are the profit rate that leads to investment just equally profits, the slope of the function at that point, and the minimum level of investment.
- A plot can be made the interface to the group, and like all plots in *Minsky*, this is dynamic;
- It is also possible to shrink a group to save space—see the bottom group in Figure 43—and to magnify it sufficiently that its contents can be seen and even edited directly on the canvas—something that is not possible in any other system dynamics program.

Figure 42: Using the same variables for two different behavioural functions

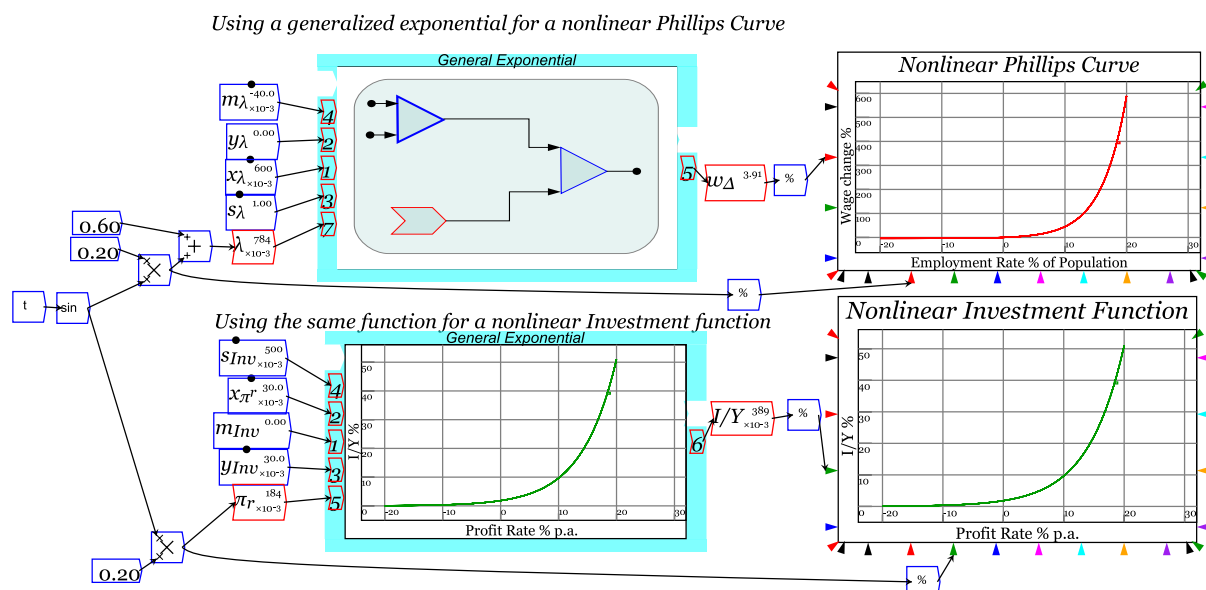
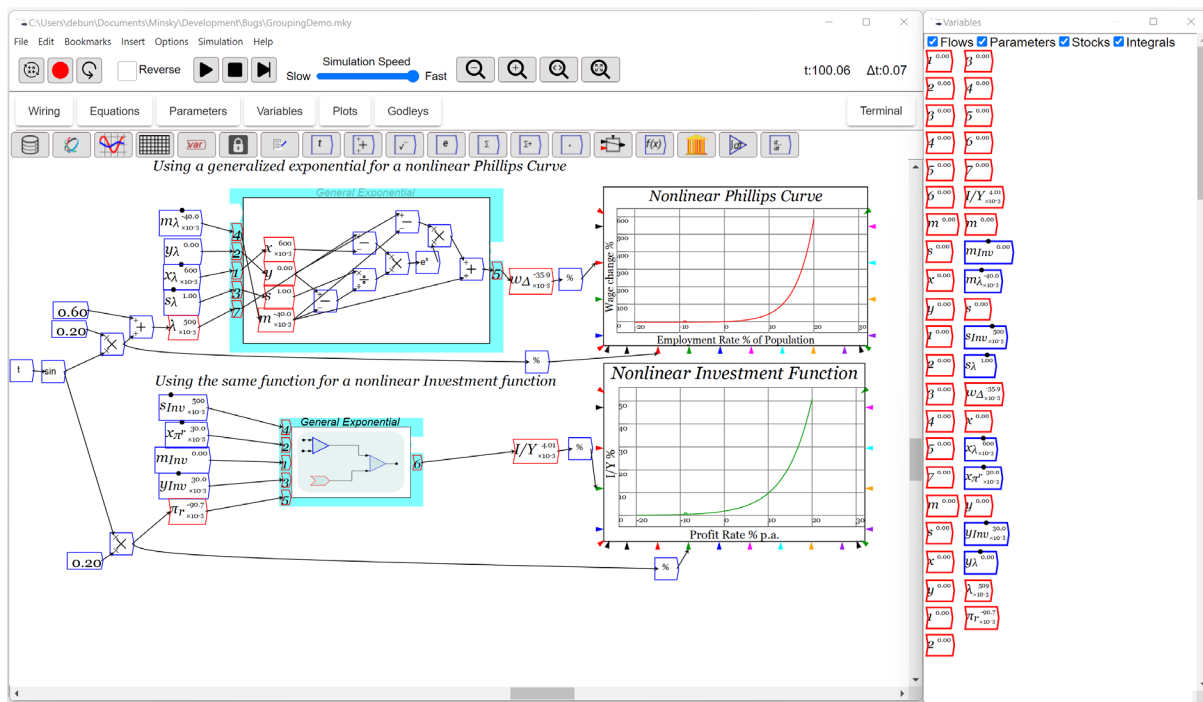


Figure 43 shows one of the weaknesses: numbered inputs are added to the model as an interface between the top-level canvas and the groups, and these turn up in the browser window. This clutters the model from the designer’s point of view, even though the group reduces clutter for the end user.

Figure 43: Current drawbacks of grouping: extraneous constants created



We hope to release an improved implementation of grouping in Minsky 3.1, which will:

- Do away with the intermediate numeric variables; and
- Support global variables within groups, as well as local variables.

In the meantime, we'd advise using groups where they are useful, as in Figure 42, where the same function is used twice via groups, but otherwise use Bookmarks to organise a large model.

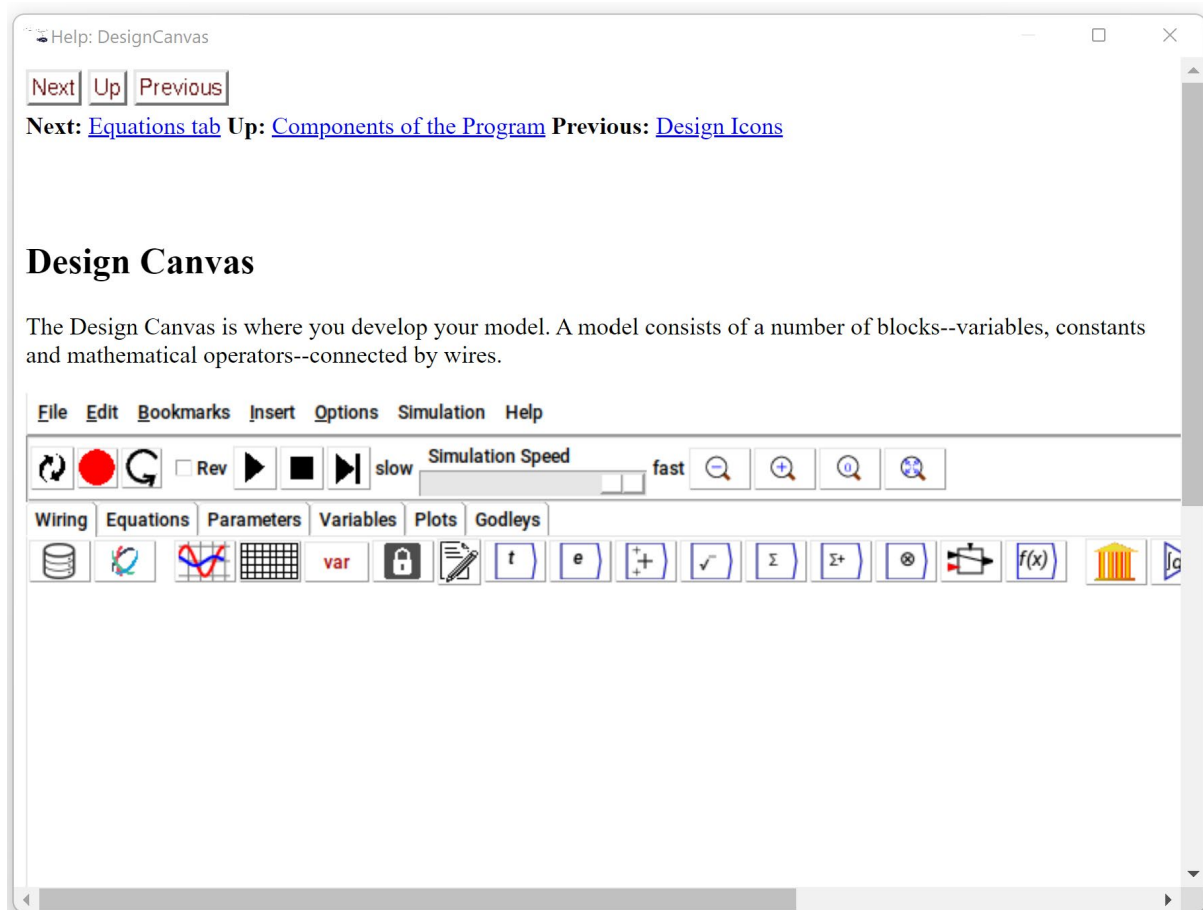
12 Getting Help

There are five main sources of help on using Minsky:

1. This manual, downloadable from <http://www.profstevekeen.com/Minsky>;
2. A lengthier half-manual, half-book on non-mainstream economics entitled *Modelling with Minsky*, also downloadable from <http://www.profstevekeen.com/Minsky>;
3. Tyrone Keynes's YouTube Channel *Modelling with Minsky* (<https://www.youtube.com/c/ModellingwithMinsky>); and
4. The online reference manual at <https://minsky.sourceforge.io/manual/minsky.html>; and
5. Context-sensitive help within the program.

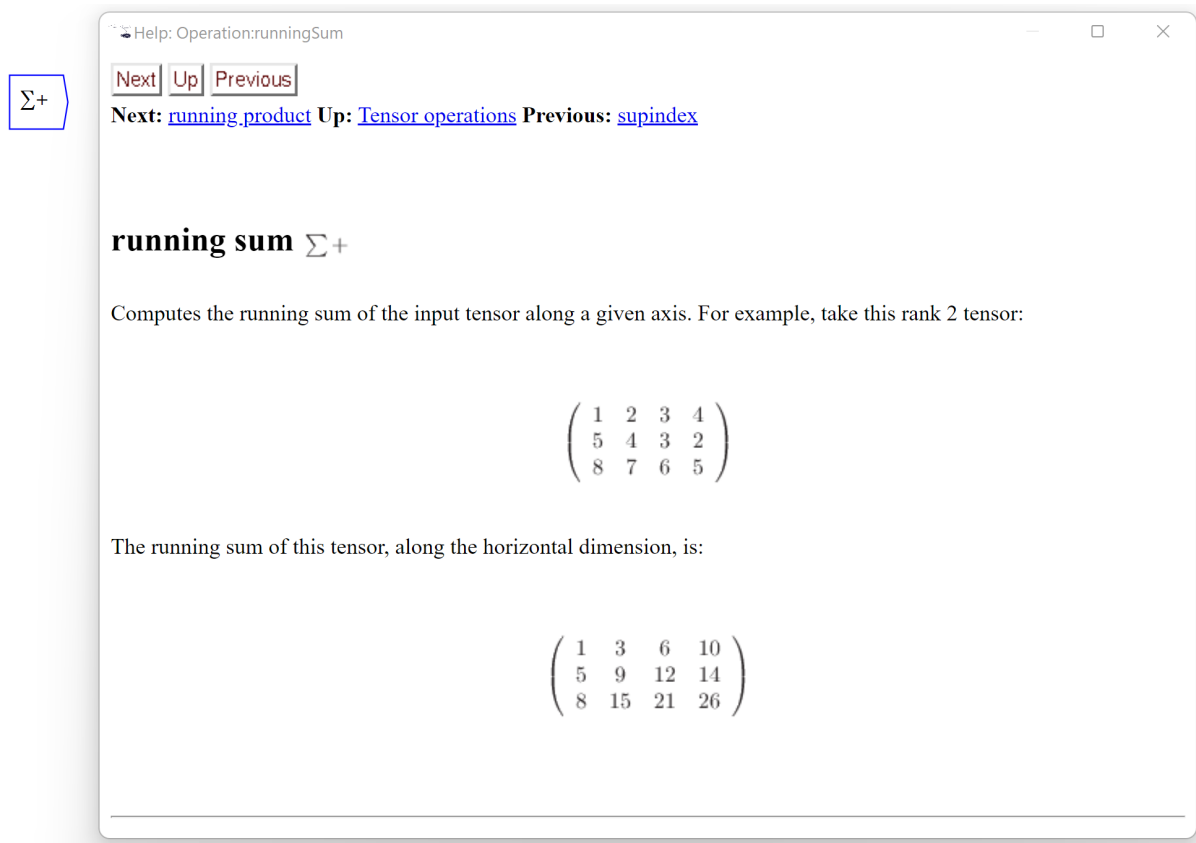
Context-sensitive help is available for all operators in Minsky via the right-click menu. Pressing the right-mouse-button while on the main canvas brings up the Canvas Help—see Figure 44.

Figure 44: The Help item for the Design Canvas



Similar Help menus apply for every component of Minsky. For example, Figure 45 shows the help entry for the running sum operator.

Figure 45: The Help entry for the running sum function



13 Helping Minsky

Minsky relies upon a community of users who help identify problems with the program, and suggest ways in which it can be improved. We encourage any Minsky user to join at least one of two *Minsky* groups via SourceForge:

1. Minsky-users: <https://sourceforge.net/projects/minsky/lists/minsky-users>; and
2. Minsky-beta-testers: <https://sourceforge.net/projects/minsky/lists/minsky-betatesters>.

Minsky's initial development was financed by [a grant from the Institute for New Economic Thinking in 2011](#), and a subsequent grant from [Friends Provident Foundation](#) enabled it to be developed to its current standard.

Many successful Open-Source programs are funded by companies that use these programs in their daily work. We hope that this will one day apply to Minsky. In the meantime, we have established a crowdfunding page for Minsky: <https://patreon.com/hpcoder>. If you find *Minsky* useful and would like to see it develop further, please sign up (for as little as \$1 a month) and help *Minsky* develop further.

13.1 Future Developments

Minsky's development is managed by the ticket system at <https://sourceforge.net/projects/minsky/>. Two major development goals for Minsky 3.0 are:

1. Producing a web browser version of *Minsky* that can be run without needing to install a program on your PC; and
2. Dramatic improvements in the quality of *Minsky*'s plots.

Many other improvements are planned as well: like any piece of computer software, *Minsky* will never be finished, but will instead always be evolving. We would appreciate your help in guiding that evolution.