Perpetual Motion Machine #2 (PMM #2)

An Agent-Based Model
of a Most-Simple Complete Sustainable Economy
-
(A Model of the ModEco Application
Developed Using NetLogo 5.0.5
Described Using the ODD Protocol)

Author: Garvin H. Boyle
Date: 27 November 2014

Perpetual Motion Machine #2 (PMM #2)
An Agent-Based Model
# of a Most-Simple Complete Sustainable Economy

(A Model of the ModEco Application
Developed Using NetLogo 5.0.5
Described Using the ODD Protocol)

Table of Contents

List of Tables

List of Figures

Perpetual Motion Machine #2 (PMM #2)
-
An Agent-Based Model of a Most-Simple Complete Sustainable Economy
-
(A Model of the ModEco Application Developed Using NetLogo 5.0.5 Described Using the ODD Protocol)

Abstract
ModEco is a software application in which a user can design and run model economies. The ultimate goal of the ModEco project is to model most-simple sustainable economies, and ultimately, to identify the necessary and sufficient conditions for sustainability. The first sustainable ModEco-based economy, dubbed the Perpetual Motion Machine #1 (PMM #1) was written in C++. A second implementation which is similar, but not the same, was implemented in NetLogo 5.0.5. The ODD protocol (Overview, Design concepts, Details) is intended to be used to standardize the descriptions of agent-based models of socio-economic systems, so that such models may be effectively replicated, and the related experiments repeated and validated. The protocol was first published in 2006, and reviewed and updated in 2010. The sustainable economy PMM #2 is described herein in detail following the ODD protocol.

Keywords
Agent-Based Model, Biophysical Economics, Complex Adaptive System, Co-evolution, Ecological Economics, Model description, Model replication, ODD Protocol, Resilience, Sustainability

1 Introduction
This paper will describe a sustainable model economy dubbed the “Perpetual Motion Machine #2” (PMM #2). PMM #1 was implemented in C++ as found in MS Visual Studio 2010, an extremely flexible application development environment (ADE) which, unfortunately, is difficult to learn to use and, it seems, rapidly falling out of use. Few developers of agent-based models now use C++ as an ADE. The second implementation of the PMM was undertaken using NetLogo version 5.0.5, a substantially less sophisticated ADE having a less steep learning curve, but also having substantially less flexibility. This description of the model will cover the NetLogo implementation only, i.e. PMM #2. It will be described using the ODD protocol (Grimm et al., 2006; Grimm et al., 2010). This protocol is designed for the description of agent-based ecological or social models, with the express purpose of making such models easier to understand and easier to replicate effectively. A copy of the ModEco software (C++ ADE) with associated technical notes and ODD description can be downloaded from the website ‘OpenABM.com’. (Boyle, 2013 [04])

Note that the PMM #2 varies in some significant ways from PMM #1, as several innovations were needed to achieve the same form of ‘perpetual motion’ using the NetLogo ADE that was previously achieved using the C++ ADE. However, it seems that complete economic systems are quite unstable when modeled using either ADE, and the techniques that worked in C++ were not fully effective in NetLogo. As a result, while this ODD description of PMM #2 is largely the same as the description for PMM #1, it does vary in some important details. Therefore, this second description of the PMM was deemed necessary. Each ODD description is complete, and can stand on its own. I make no further apology for the fact that these two documents are very similar, and duplicate each other substantially.

Most modern economies are clearly unsustainable in the long term. Any regime based on the exploitation and consumption of finite stocks of non-renewable resources must, of necessity, eventually come to an end. Furthermore, any regime additionally based on the rapid and continued degradation of the means to replenish renewable resources merely brings about its own demise ever-more-quickly. As the many national economies around the world become ever-more-closely linked they are evolving to become one monolithic unsustainable global economy.

The lesson from history is that all economies eventually collapse. Those previous societies that collapsed caused irreversible damage to their immediate locale. What is particularly alarming, now, in our modern times, is the scale of irreversible damage caused by our emerging global economy. So, the question is not “Can we tweak our modern life style, technology and practices to make our society sustainable?” Recycling pop cans and kitchen waste is not the answer. Neither, as some believe, will it be sufficient to transition from fossil fuels to renewable energy sources. These and similar ‘tweaks’ are all important and urgent, but somehow they seem to misapprehend both the scale, in the first instance, and the nature, in the second instance, of the problem. We don’t really know what it means to be ‘sustainable’. Rather, the real questions are questions of system and social dynamics. Can humankind first learn what it means to live sustainably? What does a sustainable economy look like? What are its dynamics, its extent, its relation to the rest of the biosphere? Is collapse inevitable? How does economic collapse happen? What are its triggers? What are the dynamics of sustained economic action? What are the necessary and sufficient conditions under which a modern economy can avoid collapse and be sustainable?

A sustainable economy is one which, in the words of Paul Hawken, is conservative of non-renewable resources, and restorative of the means to replace renewable resources (Hawken, 1992). I found his book to be both refreshing and thought-provoking. But such a definition is, ultimately, insufficient. These two concepts (i.e. conservative and restorative) only address the concept of ecological sustainability. One must also address the notion of economic sustainability. And that is the concept that has motivated the ModEco project.

An ideal society would not only provide an environment of peace and prosperity to its current generation, but also be stable from generation to generation, delivering peace and prosperity to its people for many generations, and this requires both ecological and economic stability, as well as mechanisms to maintain social justice (Jackson, 2009). The masses of humanity form a complex adaptive system, the principal nature of which is biophysical. We are an immense eating machine, consuming food and producing waste on a scale never before imagined. Our global economy is also a complex adaptive system, the principal nature of which is, by contrast, social, rooted in and growing out of the human biophysical system, and compounding its physical effects. These two complex adaptive systems co-evolve and co-adapt, and, in some fashion, will share the same fate. In addition, they are themselves embedded in a global ecological system which provides a variety of natural or ecosystem services which are not valued by the human economic system and are, for the most part, not well understood. These include the rain forests which provide oxygen in immense quantities, the hydrological cycle which provides fresh water for food crops, the oceans with their ability to produce immense quantities of food and oxygen, and the soil itself with its ability to produce immense quantities of food. These services are used by all people, as we draw freely from common funds, but in recent years we find these funds are being depleted, exhausted and irreversibly destroyed. We must learn to manage these necessary commons.

Putting all of this together, I here define a sustainable global economy as follows:

> ***A sustainable global economy is one which is conservative of non-renewable resources, is restorative of renewable resources, and is biophysically, socially and economically stable and self-perpetuating over many generations of people. In addition, it is mindful of planetary limits, and is accountable for its use of commonly-needed shared natural resources and ecosystem services.***

Please note that social justice of any kind is not a necessary part of this definition of sustainability. The vision of sustainability that includes shared prosperity within a contemporaneous cohort (Jackson, 2009) adds an additional difficult dimension to the definition. While the ModEco-based economies are designed only to investigate biophysical and economic sustainability without reference to social justice, some of the output of both PMM #1 and PMM #2 shows that the two high visions (sustainability and social justice) are not totally compatible. (Boyle, 2013 [03])

The above definition tells us how we might identify a sustainable global economy if we had one to observe, but it does not tell us how to construct one. So, how do we discover the operational characteristics of a sustainable economy? The goal of the ModEco project is to discover some of the necessary and sufficient characteristics that make an economy sustainable. This draws from the tools and concepts of two emerging fields of study:

- **Steady-State Economics (SSE)** is an approach to the study of economics which is grounded in an understanding of the constraints of the laws of physics, and an understanding of ecological processes. The concepts of SSE are generally considered to have first appeared in the 1966 essay "*Analytical Economics*" by Nicholas Georgescu-Roegen (Georgescu-Roegen, 1986), and were further developed by his student Herman Daly. (Daly, 1991)
- **Agent-Based Modeling (ABM)** is an approach to the study of complex adaptive systems in which computerized simulations of mutually independent agents interact with one another in a logically constructed space (Tesfatsion, 2002). A study has shown ABMs to be particularly suited to the study of ecological, social and economic systems in which sustainability is of interest (Boulanger and Bréchet, 2005).

The approach is to build an agent-based model at two levels, the top level being a very simple and very highly abstracted agricultural economic system, and the lower level being an ecological/biophysical system in which the economy is embedded. The long-term vision is to establish a more complete understanding of the dynamics of sustainability in this highly abstracted and exceedingly simple system, and then to elaborate the model(s) step-by-step as the level of abstraction is reduced and the level of realism is increased. I am hopeful that others will see the value of this vision, improve or replace this model, and seriously address the questions around the dynamics of deep long-term sustainable economics.

I undertook this project while tutoring some gifted and enthusiastic high school students. Regrettably, they have moved on and I labour on alone.

The ModEco applications (using both the C++ and NetLogo ADEs) are each conceived as a laboratory in which a number of different model economies can be enabled through toggled features and altered parameters. A parameter is, in effect, a sliding scale which changes the behaviour of a system. A toggle, on the other hand, is a switch which can entirely remove the effect of a parameter or operational feature, and, possibly, replace it with another parameter or operational feature (e.g. using a switch or chooser). A well-constructed toggle allows a user to remove all of the logic associated with a parameter, or, replace the logic around one feature with the logic of another feature. For example, a chooser might be used to select one of several scenarios.

An economy is easy to model using agent-based techniques, and, using the C++ ADE of the ModEco application, the behaviour of a number of interesting such economies was explored. However, it was discovered that, even with an utterly simple economy in mind, a stable economy is rather difficult to construct. Rampant inflation or deflation eventually brings down most ModEco-based economies. To date, the only sustainable ModEco-based economies constructed have one very severe restriction: the monetary price paid for all goods and services must be precisely equal to the intrinsic value based on metabolic needs. A worker's wages must precisely equal the cost of the energy spent in performing the work. A farmer's income from selling food must precisely equal the sum of the costs of agricultural inputs. When a gap is allowed between monetary price and intrinsic value (i.e. when price negotiations are allowed) then the potential for profits and losses arises, inflation creeps in, and positive feedback mechanisms cause the rapid collapse of the economy within a few generations.

The first such severely restricted sustainable economy that we found has been dubbed the "Perpetual Motion Machine One" (PMM #1) because it has been run up to an amazing 20 million ticks (requiring about 3 weeks of processing time) before being turned off by the user. This was achieved by eliminating the 'friction' of profit. PMM #1 is a ModEco-based economy in which many of the available C++ ModEco functions and features are toggled off. ModEco (NetLogo ADE) is a much simpler application in which many of those C++ ModEco features that play no role in PMM #1 have been set aside, and only the features used in the successful PMM #1 were re-implemented using the NetLogo ADE. However, it seems that the reduced sophistication of the NetLogo ADE made it difficult to implement the PMM #1 model in such a way that it was stable. Additional features were explored to achieve the desired sustainability. The purpose of this paper is to describe PMM #2 using the techniques of the ODD protocol.

2 The ODD Protocol

2.1 History of the ODD protocol

ODD stands for Overview, Design concepts, Details. These are the three main divisions of the protocol.

Early agent-based models (ABMs) were described in an ad hoc fashion. The ODD protocol was recently proposed as a means to encourage organized, complete and replicable descriptions. (Grimm et al, 2006) The proposed protocol was well received, and a number of agent-based models were subsequently described using it. Four years after it was first published, a review of its usages was undertaken, and a revised version of the ODD protocol was released. (Grimm et al., 2010).

2.2 Use of the ODD protocol

This paper follows the revised description of 2010. The ODD protocol was designed specifically for the description of ABMs, and, the NetLogo ADE was designed specifically for the development of ABMs, so, unlike the C++ implementation of the PMM concept, as instantiated in PMM #1, PMM #2 was comparatively easier to describe using the protocol.

3 The Perpetual Motion Machine Number Two (PMM #2)

3.1 Overview

PMM #2 is a scenario in the ModEco application (NetLogo ADE). It is a highly abstracted, agent-based model (ABM) of an utterly simple but conservative and complete agricultural economy. It is in large part a replication of the only sustainable scenario that can be established using the ModEco application (C++ ADE). I.e. it is a replication of PMM #1. (Boyle, 2013, [02]) It contains four types of agents. *wrkr*s and *frmr*s exist in multiple copies scattered throughout a toroidal logical space called the *township*. In addition, two central immortal agents called the material manager (*MMgr*) and the estate manager (*EMgr*) run the affairs of the *township*. PMM #2 is sustainable, in the sense that it obeys the conservation laws of mass and energy found in nature, it recycles renewable resources, and it can apparently run forever without external intervention of any kind. It appears to follow an almost infinitely varied self-correcting trajectory through its phase space,

The journey of discovery that led to the implementation of the PMM started with some design goals, as described in some detail in section 3.2.1 below. But, achieving some sort of sustainable system proved to be very difficult. The design goals included a dynamic design process that can only be described as a journey. It was a messy journey, filled with many experiments, many dead ends, discouragement and some surprises – a journey too long and too incoherent to recount in detail. It involves the invention of immortal agents, the establishment of regulations on the size of stocks and flows of resources, the tweaking of economic policies on taxes and grants, and, ultimately, the imposition of a rigid pricing policy. The tidied-up version of that journey is recounted with some detail in section 3.2.3.

A run of a ModEco economy is what one of my students mockingly called a ‘Zero Player Game’. A run is started, then you watch it, hoping it will not collapse. Execution of a ‘run’ of a model in ModEco can proceed in a few steps in which basic features are exercised, and some optional features may be exercised:

- INITIALIZATION:
  - Basic – Select a scenario (PMM #2 or GROWTH), select a random seed, click on ‘Setup’.
  - Optional –
    - Set the ‘Halt at:’ value, the tick at which you want a run to automatically stop;
    - Turn plot activities off or on (default) as desired for real-time graphs and data collection;
    - Turn DPX, DPT and DPG data collection to CSV files on or off (default);
    - Set parametric constants to non-default values using sliders to explore parametric space, or return all to default values;
    - Construct some ‘behaviour space’ parameter settings, and prepare to run them;
    - Set debug reporting for a single step or ‘all’ steps, set output to file or command centre, and turn the debug facility on or off (default).
- EXECUTION
  - Basic – Click on ‘Go’. Note that, for ease of use, I have placed several ‘Go’ buttons about the user interface panels. I suggest only one be activated at a time.
  - Optional –
    - Stop the action, mid-run, and ‘export’ the contents of plots to CSV files;
    - Toggle plots off or on, clearing the plots if needed;
    - Toggle CSV data collection for DPX, DPT and DPG records on or off;
    - Toggle debug facility on or off;
    - Alter parameters mid-run (not tested, not recommended, but possibly interesting)
- TERMINATION

- Basic – A run stops when the economy collapses, or it runs forever (tested to 1 million ticks).
- Optional –
  - Stops automatically when the tick indicated in the 'Halt at' parameter is finished;
  - Stops when the active 'Go' button is deactivated;
  - Stops when the completion conditions for the 'behaviour space' set is completed.

3.1.1 Purpose

The purpose of PMM #2 is to replicate PMM #1 using the NetLogo ADE as closely as is practicable.

The ultimate purpose of both models (PMM #1 and PMM #2) is to use such complete ABMs to understand the dynamics of an economic sub-system coupled to a biophysical sub-system, and to construct a tool for the exploration of the necessary and sufficient conditions for sustainability of such a coupled dynamic system.

The purpose of this description of PMM #2 using the ODD protocol is to enable others to replicate the PMM using an ADE other than the C++ and NetLogo ADEs.

3.1.2 Entities, State Variables and Scales

The entities and scales identified in Table I, together with their associated parametric constants, derived variables, and state variables play a significant role in the PMM. A detailed description of each can be found in the indicated subsection.

Table 01 – Entities and Scales of ModEco.

Entities	**Sub-section**	**Representing**
• *ModEcoSys*	3.3.2.1	The dual ecological/economic system
• *Township*	3.3.2.2	The geographic area
• *Lot*s	3.3.2.3	A portion of the *Township*
o Residences	3.3.2.3.1	A portion of a residential lot
• The economic agents	3.3.2.4	Agents
o The mortal agents	3.3.2.4.1	Agents in the private sector
▪ *wrkr*s	3.3.2.4.1.1	Agents who are workers
▪ *frmr*s	3.3.2.4.1.2	Agents who own land
o The immortal agents	3.3.2.4.2	Agents in the public sector
▪ The material manager (*MMgr*)	3.3.2.4.2.1	Agent who buys and sells waste
▪ The estate manager (*EMgr*) assets	3.3.2.4.2.2	Agent who receives and distributes
Scales	**Sub-section**	**Representing**
• The spatial scale	3.3.2.5	Relative spatial meanings
• The temporal scale	3.3.2.6	Relative temporal meanings
Non-participating entities	**Sub-section**	**Representing**
• Real-time data display entities	3.3.2.7.1	Display data in real time on the screen
• Data capture entities	3.3.2.7.2	Capture data in CSV files

Before identifying the state variables of the PMM, I must first clarify the meaning of the word 'variable' in this context. In this document we have coming together two traditions in which the word variable, when used as a noun, has two different meanings: computer science and systems science.

In computer science, a constant is a number or string of characters which is embedded in the lines of code, and which is used each and every time those lines of code are run. Such numbers are said to be ‘hard-coded’. On the other hand, a variable is an address of a location in memory that holds a suitable value (number or string) which is imported into the lines of code each time they are executed. The contents of that location may change from time to time, and the result of executing those line of code may then be different each time they are executed, depending on the contents found at that address. Such a number is said to be ‘soft-coded’.

In systems science, a variable is a characteristic of the system that changes, and a constant is a characteristic that does not change. For example, if you view the human body as a system, height changes over time, whereas species does not.

When you use a computer system to emulate a physical system, these two uses of the noun ‘variable’ may not coincide. For example, the species of an agent may be stored in a computer variable but it still represents a constant of the model. In ModEco (NetLogo ADE) every effort has been made to put all needed numbers into computer variables. They are almost all soft-coded and available to the user as explicit parameters. This does not take away from the fact that such ‘computer variables’ have two different uses in the model economies. Some are used as model constants, and some are used as model variables. The model constants are often referred to as parameters of the model, since they are initialized before a model is run, and do not change during a run. The model variables divide into two types: the state variables and the derived variables. A set of state variables is a smallest set of dynamic variables which are independent and able to precisely describe the state of the system at all points in time. The derived variables are those that are neither state variables nor parameters.

In section 3.3 there is a complete description of all parameters, state variables and derived variables used in a ModEco model. In Table 02 is a brief list of the state variables, prefixed by the entities in which they are found.

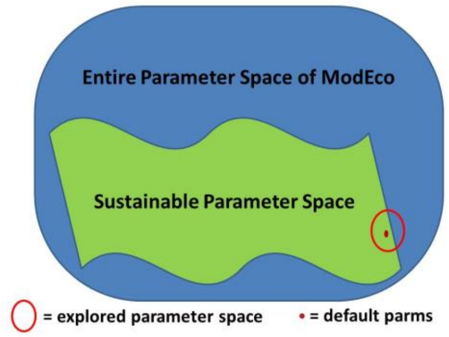
Table 02 – State Variables of the PMM

ModEcoSys:ticks	agent:ycor	agent:cash	agent:waste
agent:breed	agent:age	agent:energy	frmr:recycled
agent:xcor	agent:age-to-reproduce	agent:supplies	frmr:inventory
ModEcoSys:MMgr-cash	ModEcoSys:MMgr-mass	ModEcoSys:EMgr-cash	ModEcoSys:EMgr-mass
ModEcoSys:EMgr-energy			

Those that are prefixed with ‘agent:’ are state variables of both wrkrs and frmrs. Ticks is a built-in state variable. Breed, (xcor, ycor) and age-to-reproduce are set for each agent at birth, and are constant during the life of the agent. However, since agents live short lives, compared to the length of a run, they are state variables of the model. Wrkrs have five additional fully dynamic state variables; and frmrs have seven.

The entity ModEcoSys is characterized by these seventeen state variables held within itself and its included entities, and also by twenty-nine parameters.

Figure 01 – The parameter space


Entire Parameter Space of ModEco
Sustainable Parameter Space
= explored parameter space
= default parms

A Venn diagram showing PMM #2 as a set of models in explored and unexplored parameter space of ModEco.

(See Table 21.) One might ask how these state variables and parameters relate to each other. The parameters constrain and tune the behaviour of the biophysical sub-system, the economic sub-system, and the bridging actions between them. The state variables are constrained thereby.

The parameters span a logical space called the **'parameter space'** which includes all possible variations of settings – a practically uncountable number of possible settings. The default initialization settings are just one configuration from all those that are possible. I would hypothesize that the vast majority of settings for these parameters would result in economic collapse. Only a very small portion of parameter space has been explored, and a number of the parameters were added in the search for a configuration of parameters that did not lead to ultimate collapse. Having achieved 'sustainability', of a sort, with one configuration of parameters, a small portion of nearby configuration space has been explored, only to find that some nearby configurations cause collapse, and some do not. Figure 01 provides a two-dimensional representation of this twenty-nine-dimensional problem. The entire twenty-nine-dimensional parameter space is represented by the blue patch. That portion which leads to sustainable economies is represented by the green patch. That portion which has been explored, so far, is within the red circle. The default settings can be represented by the single point within the red circle. The unanswered question behind this figure is this: What portion of parameter space is sustainable, and what does this tell us about the necessary and sufficient conditions for sustainability in our modern economy?

The state variables span a logical space called the 'state space'. This is the set of all combinations of values for the state variables, or, the set of all states. Within a single time-limited run of the model that lasts 80,000 ticks (roughly 100 generations of agents), the probability is that exactly 80,000 of these states will be realized. There is a thirtieth parameter, not counted among the twenty-nine mentioned above, that determines the seed for the PRNG (Pseudo-Random Number Generator). I allow seeds to vary from 1 to 100. This means, for each configuration of parameters, there are 100 possible different runs, which can produce, at most, 8,000,000 different states.

You can view it this way. The twenty-nine explicit parameters allow you to explore the parameter space, by designing a model economy. The PRNG seed allows you to explore the state space of that chosen model economy. If a model economy is truly sustainable, it should be sustainable for all 100 seeds. PMM #2 is hereby defined as that subspace of parameter space for which a run lasts more than 80,000 ticks, for all possible seeds, and can therefor be considered truly 'sustainable'.

3.1.3 Process Overview and Scheduling

The details about the processes in the ModEco application are provided at section 3.3.3. There are three key processes, as follows:

- **startup** – This is a NetLogo standard routine executed automatically by the observer (the central application controller) when the ModEco application is first loaded. It ensures that the default parameter settings are set in all user interface controls, and then invokes the 'setup' routine.
- **setup** – This is a NetLogo standard routine that is invoked either automatically by **startup**, or under user control. It uses the contents of the explicit parameters (held in the user interface controls) to construct a model economy as it initializes a set of agents, computes all derived parameters, and prepares for a run.
- **go** – this is a NetLogo standard routine that is either called once, by a "One Tick" button, or called repeatedly in succession by a "Go" button. Each time the **go** routine is executed, one tick of model time elapses. The **go** routine calls a number of sub-routines, called steps, in succession. The steps, in order of execution, are: do-pre-tick, do-sell-waste-buy-recycled, do-hire-wrkrs, do-sell-inventory, do-eat-supplies, do-reproduce, do-death, do-post-tick.

3.2 Design Concepts

3.2.1 Basic Principles – The Design Goal

The design goal was to produce a sustainable economy that was:

- ***Highly abstract*** – The primary goal is to demonstrate the dynamics of an abstraction of a sustainable economy. Verification of a model is the business of removing bugs and ensuring it works as specified. Validation of a model is the business of comparing the model with a real-world system that it emulates or simulates. The meaning of this goal is that the model need not be validated in comparison to a real-world economy, in the technical sense. While it is expected and intended that lessons learned from such an abstract model can be applied in the study of real-world models, it is not expected or intended that the abstract model be able to be validated using real-world data. It can be verified against the design criteria that follow. Once resilience and sustainability are attained, the value is in the study of the dynamics of the resulting system.
- ***Most-simple*** – I hypothesize that the PMM #2 is the most simple model possible that is consistent with the characteristics described in the following paragraphs. While I agree that this model is complex, I claim that it is not needlessly complex. All operational functions and features in the PMM #2 were added through the process described in section 3.2.2, and are considered both necessary and sufficient. I challenge others to prove that hypothesis to be incorrect.
- **Conservative:**
  - ***Conservation of mass and energy*** – The conservation of mass and energy are fundamental characteristics of the real physical world, and were considered necessary characteristics of a model of sustainable economics. In ModEco, mass and energy are conserved at every step, and throughout every transaction. The model is a closed system, with respect to these conserved quantities, and the total mass and energy in the system is tallied and displayed at every step.
  - ***Conservation and ubiquity of cash –*** Cash is conserved in each transaction, and at the system level. All transfers of goods, and all deliveries of services, are mediated by the reciprocal transfer of cash. There is no bartering or payment in kind. There is no volunteerism.
  - ***Conservation of intrinsic value -*** Intrinsic economic value, based on human metabolic needs, is assigned to mass and energy. This arises from the concept that a day's earnings must pay for a day's metabolic needs. Intrinsic value is conserved implicitly in all commercial transactions. In the NetLogo implementation, due to the elimination of price negotiations and the closure of the monetary price/intrinsic value gap, the monetary value and the intrinsic value are always equal in every transaction.
- **Horizontally complete*:***
  - ***Complete biotic life cycle -*** Agents are born, live a life of limited length, and successfully reproduce (via fission, the most simple form of reproduction) when they are old enough and healthy enough, or they die of starvation, or old age. Death comes from metabolic insufficiencies.
  - ***Complete abiotic resource cycle –*** Mass and energy cycle through the system in a closed loop. I realize that, in reality, the quality of energy degrades on use and it cannot be recycled, but, in the spirit of 'most simple', I decided to put it into a loop. [*Any attempts to model an open energy economy have failed, so far, as such models have all quickly collapsed*.] I also realize that, in reality, mass cannot be recycled forever, and its quality and usefulness quickly degrades. I have not yet attempted to construct a model economy in which mass degrades in quality. Cash, the third explicitly conserved abiotic resource, also cycles in a closed loop.
- **Vertically complete*:***
  - ***Utterly simple biophysical sub-system*** - At the lower metabolic level, agents harvest food, eat it, and recycle the waste. This cycle could not be more simple and still be horizontally complete.

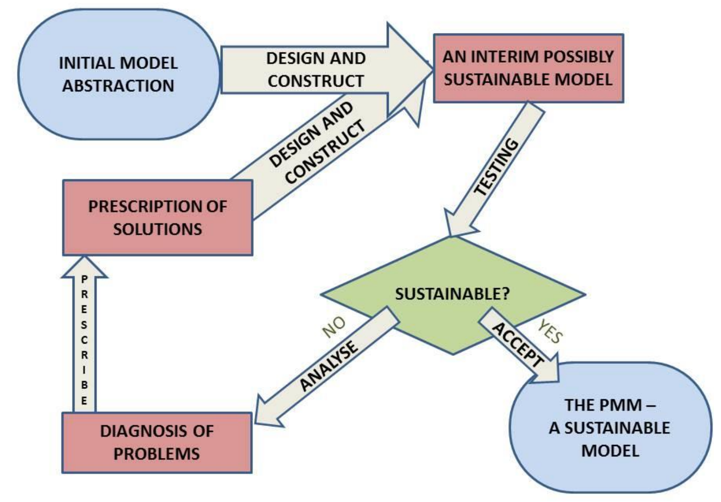
- ***Utterly simple economic sub-system -*** At the higher economic level, farmers hire workers to harvest the food and place it in inventory, then farmers sell food to consumers, consumers sell waste to a Material Manager, who, in turn, recycles it and resells it to farmers. With each and every exchange of goods and services, an equivalent value of cash changes hands. This economic activity could not be more simple and still be horizontally complete.

- **Limited economic knowledge, both spatial and temporal** – Mortal agents do not have global knowledge or perfect local knowledge of the market, but, rather, have limited knowledge of the existence of only those other agents that reside within a local commuting area and no memory. That is, their knowledge is limited in space and time (to the here and now). They do not pick the best local commercial opportunity, but are randomly presented local opportunities, and must simply decide whether or not to exercise each as it is presented. Agents have knowledge of, and their behaviour is constrained by many system-wide constants. These include business factors that control diversification of internal asset classes, transaction quotas that limit the sizes of resource flows between agents, and controls on metabolic processes and life functions. But, this is not knowledge that differentiates agents. Agents have no knowledge whatsoever of the state of other agents, mortal or otherwise, except in response to specific transactions. Knowledge available from such transactions is not collected or stored by agents over time.
- **Limited personal goals –** Mortal agents, if they can be said to have a goal, have only one goal: to survive biophysically until they are old enough and healthy enough to reproduce. This requires that they remain economically fully enfranchised by diversifying their assets.

3.2.2 The Design Process

Since the purpose in building the PMM #2 is to discover the necessary and sufficient characteristics of a truly sustainable modern economy, one needs to follow a process that strips away all complexity, both necessary and unnecessary, leaving only utter simplicity, and then adds some needed complexity back until sufficiency is achieved. The process used in the design of ModEco is:

Figure 02 – The Design Process


INITIAL MODEL ABSTRACTION
DESIGN AND CONSTRUCT
AN INTERIM POSSIBLY SUSTAINABLE MODEL
DESIGN AND CONSTRUCT
TESTING
PRESCRIPTION OF SOLUTIONS
SUSTAINABLE?
NO
ANALYSE
YES
ACCEPT
PRESCRIBE
DIAGNOSIS OF PROBLEMS
THE PMM – A SUSTAINABLE MODEL

(a) **Initial Abstraction** – Abstract away most complexity, leaving only the concepts of conservation, of vertical and horizontal completeness, of incomplete temporal and spatial knowledge, and of limited goals.
(b) **Model Construction** – Build a (better) most-simple complete conservative economic model.
(c) **Model Testing** – Test for sustainability. If the model economy does not collapse, you are done. Otherwise, go to step (d).
(d) **Diagnosis and Prescription** – Collect and analyse data from the faulty model. Hypothesize as to why the economic model collapses. Design necessary corrective features that are as simple as possible but improve the sustainability. Unfortunately, this almost always adds complexity to the model.
(e) **Rebuild** – Return to step (b), in a cycle, until sustainability is achieved.

3.2.3 The Design Journey

Like the C++ ModEco economies, the NetLogo implementation of ModEco has some built-in features that can be turned on and off, and so there are some configurations of features that are not, in fact, ‘perpetual’ or sustainable. Many so-called ‘debug’ features were added over the course of the development of the model with the purpose of understanding why the model economies were, in fact, not sustainable, but always ended in collapse. In spite of the fact that the NetLogo implementation was intended to be a simple and quick replication of the singular C++ ModEco feature set that was sustainable, a substantial amount of further experimentation had to be undertaken, much of which enhanced the ModEco (NetLogo ADE) program, until a sustainable model was finally achieved. The following paragraphs describe some of the thought processes that lead to the discovery of PMM #1 (in C++), and then after that, PMM #2 (in NetLogo).

3.2.3.1 PMM #1 and the need for immortal agents

A closed system must have a clear path through which conserved quantities can travel. Intrinsic value (carried by energy and mass) and cash always flow in opposite directions. These paths must return upon themselves forming an unbroken closed loop. Two problems arose.

Diagnosis: First, consumers held waste (mass, with its associated intrinsic value) and needed to sell it for cash to complete the mass, intrinsic value and cash cycles. Note that in ModEco garbage must retain value and must be sold to complete the cycles. While the possibility of having the consumers sell waste directly to the farmers was tried, it failed for two reasons. The conversion of waste mass to recycled mass by the frmrs seemed to be a bottleneck, resulting in early collapse. Also, the mass tended to become more and more centrally concentrated causing a spatial concentration of the scarce resource, resulting in eventual collapse. This latter effect was an emergent phenomenon which was not part of the design intent, and detrimental to the sustainability of the society. The analogy to garbage building in the core of ancient cities is interesting. The analogy to power (resources) concentrating in the hands of an elite is also interesting.

Prescription: To counter these perceived problems, the material manager (*MMgr*) was invented, having the ability to buy waste from any consumer at any time, and immediately sell it to any *frmr*, regardless of their location in the township. This opened up the bottleneck, and distributed the mass away from the city centre.

The role of the *MMgr* seems to be necessary, due to its ability to avoid spatial concentration of mass, while playing a role in providing an efficient closed loop for the four conserved quantities. Necessary, but insufficient.

Diagnosis: Second, when agents die without offspring, where do the assets go? If the economy is to be both sustainable and conservative, those assets need to be placed back into the economy.

Prescription: The assets were handed over to the estate manager (*EMgr*) to be distributed to poor but deserving agents. A “deserving” agent is defined as one that has the means to participate in a commercial opportunity as offered. This excludes the poorest of agents, but includes most agents most of the time. A “poor” agent is one which has sufficient resources to participate in a commercial opportunity, but not enough to maximize participation. Such a poor agent can apply to the *EMgr* for a grant of the resource in short supply, should it be available. Each such grant increases the wealth of the receiving agent since it is not a value-for-value commercial transaction, but a grant of wealth having intrinsic value associated with it.

The mere introduction of the central agents was not sufficient. The roles of the MMgr and EMgr became significant means to fine-tune the transfer of assets between agents. The net worth of every agent is conserved in every action with two exceptions: (a) when an agent reproduces (via fission) each daughter agent receives half of the net worth of its single parent agent, so the long-term trend would be asymptotically towards zero net worth of all agents; but (b) when an agent dies having failed to reproduce, its assets are transferred to the EMgr, and the net worth of the EMgr climbs exponentially towards the total value of the economy. It is the job of the EMgr to redistribute this wealth to the mortal agents as quickly as is reasonable, and those actions of redistribution of wealth cause the net worth of receiving agents to increase. When the long-term trend towards a decrease in net worth due to reproduction is balanced by a long-term trend towards increasing net worth due to redistribution of the assets of dead agents, then long-term stability and sustainability are achievable.

The decisions respecting to whom these assets should be given, and when, are economic policy decisions. There is a very wide range of options. Doing nothing leads to immediate collapse. Many policies lead to eventual collapse. The economic policy of the EMgr is a means to tune the system to find a sustainable dynamic.

The role of the EMgr seems to be necessary, due to its ability to provide a closed loop for the four conserved quantities. This role is necessary for sustainability, but insufficient.

3.2.3.2 PMM #1 and regulation

The design of a model economy is primarily the design of stocks and flows of conserved quantities. In a sustainable economy, the finite resources must be able to flow equitably to all agents, in some sense, and must not be allowed to pool (stop flowing) too much in the stocks of one agent. Disenfranchised agents die, and when too many die, the economy collapses. I attempted to introduce as little regulation as possible into the models. Nevertheless, it was found that a minimum amount of regulation was needed, both at the level of the individual agents, and at the level of the economy as a whole, in order for the model to operate sustainably.

Diagnosis: Individual agents had to ensure that their net worth was spread over all asset classes in order to be ready to participate in economic opportunities as they arose. Lack of cash, lack of energy, or lack of food (whether as inventory for sale, or as supplies for consumption) at any moment could reduce economic participation leading to death. The survival of individual agents in a competition for scarce resources requires internal regulation of activities.

Prescription: When economic opportunities arise for an agent, the relative size of key internal asset classes are compared against associated ‘business factors’ by the agent, and the response is regulated. Those fixed global business factors are used by an agent to dynamically determine target sizes of each key asset class, regulating the individual activities of agents. This is analogous to the bankers’ advice that investments should be diversified.

Diagnosis: But on a higher system level, the interaction of agents with the environment, with each other, and with the central immortal agents (who could be viewed as government agents) often caused collapse. A wealthy agent could, in one tick, buy up all of a resource and essentially remove it from the economy. This massive concentration of resources under the control of a few agents causes the economy to collapse.

Prescription: System-wide quotas are placed on all flows between agents (on a per transaction basis), even those involving the central immortal agents. This disallows massive transactions and seems to dramatically stabilize the flow of resources.

Regulation of the size of stocks of assets within agents and flows of assets between agents was necessary to achieve sustainability, but, in and of itself, insufficient.

3.2.3.3 PMM #1 and independent resilience of subsystems

Diagnosis: Sustainability in an ecosystem or an economy is closely associated with the concept of resilience, which is the ability of a system or subsystem to return to a normed state following a severe disturbance. A resilient system exists and moves within the basin of attraction of a set of attractor states within its state space. If an external or internal event causes it to move away from that attractor set, a resilient system will eventually return to orbit near the attractor set. Each subsystem within a stable economy must have its own state space, its own basins of attraction, and its own ability to be resilient. The interactions of the subsystems must not interfere with this local resilience.

For example, let's imagine an ABM of a simple ecological system. Suppose you have a finite toroidal automaton in which logical bugs eat energy-carrying algae found in a field. Each bug can hold a maximum amount of energy E_{max} and the system holds a fixed amount of energy E_{total}. In such a system energy is neither entering nor leaving the system, making it "closed". If a population of bugs becomes too large, the average energy per bug drops, the energy in the field becomes scarce, and metabolic requirements will cause many bugs to die of starvation, returning the energy to the algae in the field. But, when the population becomes small, then most of the energy is returned to algae in the field making food easily available and encouraging a return to a higher population number. The population of bugs then stays at or near the carrying capacity of the field. The number of bugs in such a population is resilient, and the population is sustainable.

Prescription: In PMM #1 we have chosen to make the system closed with respect to the four conserved quantities mentioned above, with a minor exception for one of the four. Energy, mass and intrinsic value are strictly conserved, and no agent is allowed to have negative amounts of these physical quantities. However, total cash is conserved, but a central agent is allowed to go into debt. That is, the *MMgr* (described above) can purchase waste mass using money it does not have. This ability to buy up surplus waste assets and inject money into the economy is analogous to the concept of "quantitative easing" by which "toxic assets" are purchased by a government. In ModEco, such a technique partially decouples the biophysical and economic sub-systems and allows the economic sub-system to approach its own economy-related steady state while the metabolic sub-system approaches a biology-related steady state, allowing each to be resilient somewhat independently of the other.

Decoupling the biophysical and economic sub-systems was necessary to achieve sustainability, but, insufficient, even when combined with the other techniques already discussed.

Summary of techniques:

- The MMgr was introduced to open up the bottleneck on the flow of mass from consumers to frmrs, and to undo the flow of mass from the edges of the community to the centre.
- The EMgr was introduced to collect the assets of agents who die without offspring, and to return it to the poor but deserving agents, ensuring that the resources of dead agents stay in circulation, and establishing economic welfare policy needed for sustainability.
- Business factors were introduced to ensure that agents internal stocks of resources remained sufficiently diversified to keep the agent fully economically enfranchised. Agents refrain from

participating in commercial opportunities that put the ratios of stock values too far from standards coded in the business factors.

- Transaction quotas were introduced to ensure that resources did not pool in the control of a small elite of exceptionally wealthy agents.
- The ability of the MMgr to assume debt was introduced to allow the two sub-systems to decouple to improve the resilience of each.

3.2.3.4 PMM #1 and the monetary price / intrinsic value gap

Diagnosis: With all of the above innovations and the associated added complexities, all model economies nevertheless collapsed within a few generations of agents, at best. After spending some time thinking about physics lessons in which students learn about friction-free mechanics, we (my students and I) sought an understanding of what the analogy of friction might be in an economy. We came to the conclusion that intrinsic value in our economic models plays a role similar to the role of energy in a physical model, and is, in fact, the bridge between the two sub-systems. Friction causes the dissipation of energy, causing a system to run down and stop. Similarly, we reasoned by analogy, the dissipation of intrinsic value, in some fashion, caused the economic system to run down and collapse. But, since intrinsic value is conserved, the real cause was the lack of maintenance of parity between monetary price and intrinsic value, and this price/value gap caused by price-negotiations allowed profits and losses, and led to inflation. Every transaction in which the value of cash transferred differs from the intrinsic value of the goods and services transferred reciprocally; every such transaction causes a modicum of inflation or deflation. This price/value gap eventually generates positive feedback, causing continued inflation or deflation and, ultimately, failure.

Prescription: So, in PMM #1 we removed the price/value gap. Every commercial transaction involves a precisely equal exchange of monetary value for intrinsic value. Unit price becomes an exogenous variable that does not change over time. There is no profit and no loss associated with any transaction. Buyer and seller always agree on the price, and it is always an accurate assessment of the intrinsic value to 19 decimal digits. Taking this concept to its logical conclusion, we realized that all agents would have unchanging wealth throughout their lives, but for the intervention of the EMgr.

While this extreme constraint is clearly unrealistic, it appears to be a necessary characteristic of a model economy that is conservative and complete and sustainable, as described above as our design intent. Necessary, and, together with all of the previous innovations sufficient, in the C++ implementation.

3.2.3.5 PMM #2 and additional individual regulation

For reasons not entirely understood, all of the above enabled us to achieve sustainable operation in PMM #1 (C++ ADE), but failed in PMM #2 (NetLogo ADE). It is reasonable to speculate that the somewhat less sophisticated ADE, and the necessarily less sophisticated code of the NetLogo implementation, introduces an instability. However, that is only speculation, however obvious it may seem. In the search for sustainability using the NetLogo ADE, two additional forms of individual regulation were introduced into the toolset of the mortal agents:

(a) **Diagnosis**: Widespread extreme poverty caused sudden collapse. In the process of diversifying the investments in the different asset classes, the agents attempted to maintain relative sizes of asset classes, even when very poor, but the poorest agents were therefor passing up economic opportunities too often while doing so. **Prescription**: Stocks of key asset classes held by agents were assigned a minimum absolute size, below which relative size became irrelevant to the mortal agents. The result is, the desperately poor agents had diversification restraints removed, and they were allowed to participate in every economic opportunity that was otherwise available to them, asking for help from the EMgr in the form of municipal grants, enabling a larger flow of assets from the EMgr to the poorer agents, and thereby prolonging life.

(b) **Diagnosis**: Stocks of energy in individual agents did not rise high enough to meet the 'health' requirement for reproduction. **Prescription**: A ratcheting mechanism that prevents older agents from using too much energy in 'work' activities was introduced.

Curiously, neither of these techniques were required in PMM #1. The additional restraint on older agents (enabling more effective reproduction), together with the reduced restraint on extremely poor agents (enabling higher levels of economic activity), achieved a stable system dynamic. It appears that these additional variations on the regulation of the activities of individual agents are necessary to achieve sustainability in the NetLogo implementation of the PMM.

So, with these two innovations, and some tinkering with the EMgr's welfare policies, a stable and sustainable economy was finally achieved using the NetLogo ADE in the form of the PMM #2 as of October 2014, ending a six month search for sustainability using that ADE.

3.2.4 Emergence

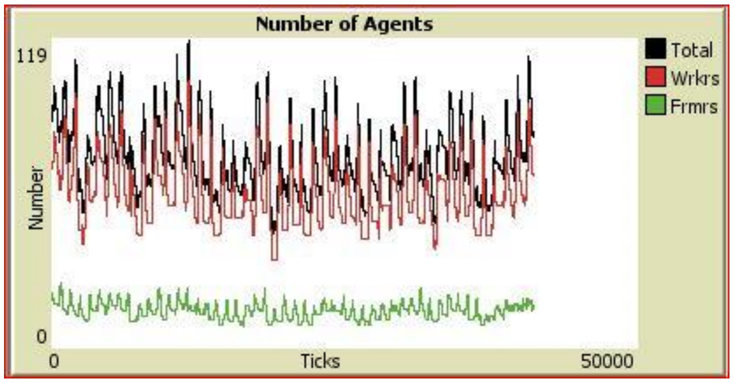
An emergent property is a behaviour which is not intended – a behaviour which appears in a complex adaptive system but which was not part of the design. Several interesting characteristics are emergent in PMM #2, including:

- Automatic relative scaling of *wrkr* and *frmr* population sizes;
- Automatic relative scaling of *wrkr* and *frmr* division of wealth between sectors;
- Log-normal distribution of wealth;
- The geographic formation of concentrations of agents; and
- A variety of macro-economic indicators such as velocity of cash, employment levels, velocity of goods and services.

3.2.4.1 Population scaling

The question of how to scale a steady-state economy is of interest in many circles. In the PMM, the steady-state population ratio stabilizes at from approximately 4 to 13 wrkrs per frmr. (See section 3.3.3.7.)

Figure 03 – Populations automatically scale.



The steady-state population sizes are determined by initialization parameters, particularly those which specify endowments of conserved renewable resources and consumption rates. The relative sizes of the two populations (*wrkr*s and *frmr*s) co-evolve until an uneasy balance is found between them. These populations rise and fall in a pattern somewhat similar to a predator-prey relationship, but the cause of death is competition for scarce common resources rather than predation. Each of the two populations must be composed of members that fight desperately among themselves for resources in order to survive, but do not drive the other population to extinction.

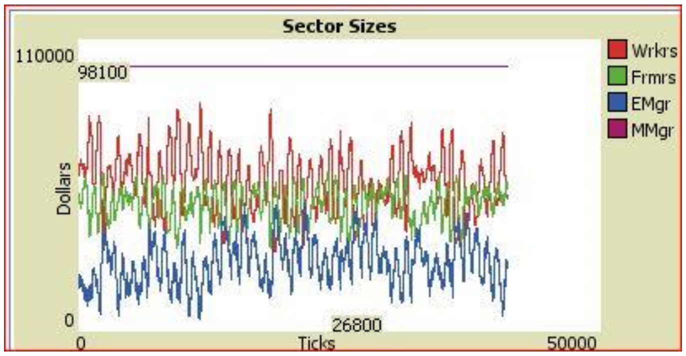
It could be argued that this is not a truly emergent property, since, of necessity, any sustained population living on a finite but recycled pool of resources will be of finite non-zero size. Particularly so, because agents with too little wealth die, under the control of a parameter. And if true for the population as a whole, this must also be true for both sub-sets of the population. And, therefore, they must achieve some type of relative and sustainable size relationship. However, the regular pulsing interplay between the populations is an unintended and unpredicted emergent property. There is a tradeoff between two competing forces that hold the population near an equilibrium size. (See section 3.3.7.7.)

3.2.4.2 Inter-sectoral wealth scaling

Similar to the scaling of population sizes, the amount of wealth held by the two competing sectors of the economy, the personal (*wrkr*s) and business (*frmr*s) sectors, is automatically scaled at levels which are difficult to predict from the parameters.

The relative size of the two sectors seems to stabilize at a ratio of approximately $1 in the wrkr sector to $1 in the frmr sector. On average, the wealth of the frmr is from 4 to 10 times the wealth of the wrkr. (See section 3.3.7.7.)

Figure 04 – Sectors automatically scale.



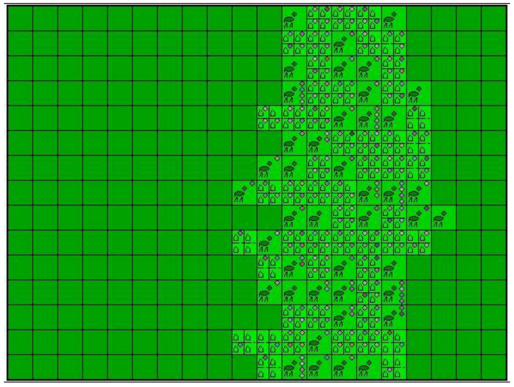
3.2.4.3 Intra-sectoral wealth scaling

Within a sector, for example, within the population of *wrkr*s, there is a distribution of wealth that is similar in shape to a log-normal curve but which has some very distinctive characteristics. (See section 3.3.7.4.)
It is highly reminiscent of the Boltzmann distribution of energies (Yakovenko, 2010). A phenomenon analogous to the "Maximum Entropy Principle" (MEP) may be causing this distribution.

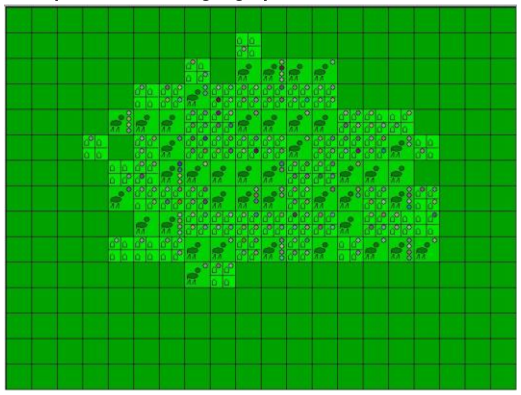
3.2.4.4 Geographic/spatial concentrations

Regardless of the spatial dispersion of agents on initialization, when PMM #2 reaches steady state, the mortal agents are huddled in a circular concentration in one place in the township, or locate in a film that connects the top and bottom edges of the township. The effect is highly reminiscent of the behaviour of water under the influence of surface tension, and the formation of droplets or thin soap films. This phenomenon has some deep implications worth exploring.

Figure 05 – Emergent Patterns from PMM#1



Example of a film-like geographic distribution.



Example of a drop-like geographic distribution, taken from PMM #1.

Those agents in the middle of the concentration have difficulty reproducing for lack of a site to locate the offspring. Reproductive opportunities vary by relative location of an agent in the cluster. Those at the edge often locate offspring too far from the concentration to be viable. Also, commercial opportunities vary by relative location. Those agents in the middle have a densely populated commuting area. Those agents on the edge have a partly vacant commuting area. So, relative location in the cluster has significant benefits and drawbacks, wherever the agent is located.

Hypothesis: A change in the size of the k-neighbourhood (commuting area) will reduce the tendency of agents to pack into a tight area. This hypothesis is as yet untested.

If the *Height* and *Width* of the *Township* are large enough, so that the economic agents huddle in a small area within the *Township*, then these two parameters play no role in the economy. But, if these values are roughly equal in size to the diameter of the cluster (an emergent property) then the huddle of agents will span the *Township* from top to bottom, the edges of the cluster will merge, and a small edge effect will alter the behaviour of the economy. This edge effect has not been explored.

This clustering also results in a concentration of waste mass in the centre of the cluster, making it unavailable to farms at the edges, and causing farms on the edges of the cluster to struggle and die. The cluster would shrink, and the population would shrink. This was viewed as counter-productive to the goal of sustainability, and the role of the MMgr as a non-local agent was introduced to reduce the effect. (See section 3.2.3.1.)

3.2.4.5 Macro-economic indicators

While macro-economic indicators may be attached to any agent-based model, and are not, in and of themselves, emergent, the behaviours that they track and measure are emergent. So, things like the velocity of cash, the velocity of goods and services, unemployment rates, or economic sector sizes, all exhibit emergent behaviour. Similarly, metabolic measures such as average age and level of health and wealth at reproduction, average age and level of health and wealth at starvation, or average level of health and wealth at point of death by old age are all measurable emergent characteristics. These characteristics are not planned, and are only indirectly controlled by any one parameter. The relationship between parameters and macro-economic and macro-metabolic behaviours have not yet been explored in PMM #2.

3.2.5 Adaptation

In ModEco-based economies (C++ ADE), agents can evolve pricing strategies. However, in PMM #2 these adaptive features have not been implemented. To date, all economies in which there is a price/value gap have collapsed and been unsustainable. Therefore, explicitly adaptive features have not been implemented here, and the agents in PMM #2 have no adaptive traits.

However, the economy, as a whole, does seem to have adaptive capabilities. For the parameter settings published herein, the economy has negative feedback mechanisms which enable it to remain resilient and sustainable in the long term. These negative feedback mechanisms are not, at present, fully understood. Many parameter settings lead to economic collapse. A study of the viable parameter space for sustainability of PMM #2 has not yet been undertaken.

3.2.6 Agent objectives

When in stationary state, 50% of all agents die of starvation or old age. Implicitly, the goal of each agent is to survive to reproduce. These objectives are implicit in the parameters of the biophysical sub-system described in section 3.3.6.1.

- To avoid starvation an agent must maintain a minimum level of supplies and energy.
- To reproduce an agent must be old enough and healthy enough.
- To avoid death by old age an agent must reproduce before reaching old age.

Implicitly, agents also have the objective of keeping its stocks of assets diversified to remain fully economically enfranchised. If an agent has an opportunity to participate in a transaction, but has zero content in any stock that is required as an input to that transaction, the agent cannot participate, but must sit out the opportunity. Those agents which have empty stocks of any kind from time to time are therefore hindered in their participation in the economy. Agents which are wealthy and able to maintain all stocks well above zero levels are therefore fully able to participate in all commercial opportunities that come their way. These objectives are implicit in the "business factors" described in sections 3.3.2.4.1.2 and 3.3.6.2. In short:

- frmrs use the *frmr:HRL* derived dynamic variable to decide when to buy recycled mass;
- frmrs use the *frmr:HIL* derived dynamic variable to decide when to hire workers;
- wrkrs and frmrs use the *wrkr:HSL* and *frmr:HSL* derived dynamic variables to decide when to buy supplies; and
- All agents consume supplies and sell waste at every opportunity.

Agents benefit from commercial transactions in two ways. First, it keeps the goods flowing into its supply stock, and keeps the waste flowing out. Second, if an agent can engage in a commercial opportunity but has insufficient amounts of a resource to meet quota for that transaction, it applies to the EMgr for a grant of that kind of resource, and, on receipt of a grant, it increases its overall wealth. Such an increase in wealth (a) reduces the probability of death; (b) increases the probability of being reproductive; and (c) increases the percentage of commercial transactions in which it can engage in the future.

It is somewhat anthropocentric to say the agents have goals, which is somewhat similar in style to saying mother nature designs creatures. The reality is, the poor agents will probably starve. The wealthy agents will probably become more wealthy, more healthy, and reproduce as soon as they come of age. However, poor agents that benefit from several low-probability high-impact grants can nevertheless become wealthy, and middle-class agents that fail to receive any grants will produce sickly offspring that will probably die of starvation.

3.2.7 Learning

In a typical ModEco-based economy (C++ ADE) agents modify their pricing behaviour via genetic evolution, and via a memory of recent gene-mediated prices. In PMM #1 such genetic and learning techniques are toggled off. The agents do not learn. In PMM #2 (NetLogo ADE) such learning abilities are not implemented.

3.2.8 Prediction

ModEco-based economies do not use predictive algorithms of any type. All decisions within the system are based on current-state information compared with parameters.

However, the business factors and system of quotas can be considered to be a parameter-controlled resource allocation system that has implicit decision-making built in. The economy uses the system of quotas to prevent any one agent from usurping all of the available resources in any one tick. Similarly, the agents use the business factors to prevent the unnecessary hoarding of any resource in excessive relative amounts in any one of its stocks.

Prevention of actions which will lead to collapse of the system or collapse of the agent can be viewed as predictive in the negative, as in “If you do this you will perish.” This is the limited nature of prediction in PMM #2.

3.2.9 Sensing

Mortal agents sense the existence of other agents within a k-neighbourhood of 25 *Lot*s (5 *Lot*s by 5 *Lot*s). This is called the “commuting area” of the agent. The commuting area is implemented as an object associated with each *Lot*, and is a (2k+1) by (2k+1) array of pointers to *Lot*s in the local vicinity within the toroidal township. I.e. the pointers are consistent with the wrapped edges of the township.

Mortal agents maintain no knowledge of the metabolic or commercial characteristics of other agents.

The *MMgr* and *EMgr*, on the other hand, sense the existence of all mortal agents and can do business with any one of them.

3.2.10 Interaction

All interactions between agents are controlled by lists. For example, each *frmr* maintains a list of workers (the union list), of suppliers, and of customers, each containing the address of all relevant agents within its commuting area (the 5x5 square around it). The ordering of agents on the lists can seriously affect the participation of the agents; those being first on the list having the best access to commercial opportunities. This arbitrary bias is removed by doing a random draw of agents from the appropriate list for each transaction. Also, when agents reproduce, a new site is found by a randomized draw from a list of sites. This randomization happens implicitly in NetLogo, and is controlled by the PRNG, seeded during initialization. Two runs using the same seed will have identical results.

The only interactions between agents in a ModEco-based economy are commercial in nature. Agents do not mate for reproduction or interact for other metabolic or social purposes. Whenever agents interact, one agent is the active agent, and the other is passive. Table 03 summarizes the interactions.

Table 03 – Types of Commercial Interaction Between Agents

Active Agent	List Type	Description
MMgr	*ModEcoSys:turtles*	Purchases waste from consumers (all agents).
MMgr	*ModEcosys:frmrs*	Sells recycled mass to frmrs.
frmr	*frmr:union-set*	Hires wrkrs within its commuting area.
frmr	*frmr:customer-set*	Sells inventory to customers (consumers) within its commuting area (both wrkrs and frmrs, including itself).
frmr	*frmr:supplier-set*	Maintained, but not used, passive.
wrkr	*wrkr:employer-set*	Maintained, but not used, passive.
wrkr	*wrkr:supplier-set*	Maintained, but not used, passive.

Agents may, on occasion, refuse or be unable to participate in a commercial opportunity when it presents itself (via a randomized draw). An absolute lack, or an over-sufficiency, of a relevant resource may result in non-participation in a deal.

Mortal agents tend to follow a protocol when an interaction is imminent, as follows:

- The active agent determines if it has the pre-requisite resources to enter negotiations;
- The active agent consults its business factors and stock levels to determine if it should enter into negotiations;
- A target agent is Contacted;

- An offer is made (a sales pitch, or a job offer);
- The target agent determines if it has the pre-requisite resources to enter negotiations;
- The target agent consults its business factors and resource levels to determine if it should enter into negotiations;
- A deal is made (note that in PMM #2 agreement is always reached based on the intrinsic value);
- Both agents check resource levels, and if any are below quota, applications are made to the *EMgr* for grants of estate assets from dead agents;
- The *EMgr* grants resources on a first-come, first-served basis, while stocks last;
- Appropriate resources are transferred between agents, pro-rated to a fraction of quota based on the lowest available stock.
- Ununsed grants are returned to the EMgr, if any.

Note that the estate manager (*EMgr*) does not use a list. It provides grants in response to grant applications. In order to be eligible for a grant, an agent must have successfully negotiated a deal (deserving), and is simply topping up the levels of resources that fail to meet quota (is poor). Grants are approved while the *EMgr* has assets to grant. Resources are committed by the *EMgr*, and then either distributed and used, or returned to be granted to the next applicant.

See section 3.3.2.4.2.2 for more about the EMgr and section 3.3.6.3 for more about the municipal grant sub-system.

3.2.11 Stochasticity

ModEco (NetLogo) was developed using NetLogo 5.0.5. The built-in pseudo-random number generator (PRNG) is not well documented, and offers implementation problems. A ModEco economy is a finite state machine (FSM), and, when given precisely the same starting parameters, a run should be 100% repeatable over millions of ticks. The PRNG is called implicitly by certain native NetLogo commands, and those commands may be embedded in optional executables such as plots, buttons, or toggled features. I have taken care to ensure that all such optional executables do not make a call to the PRNG. I am therefore relatively certain that all runs starting with the same seed should proceed along an identical trajectory in state space. However, any commands issued by the user during a halt may invoke the PRNG and change the trajectory.

In short, PMM #2 should produce the same results for every run, if it is started with the same initial conditions, and the same seed. This is true whether the run is briefly halted to check real-time output, or to start or stop collection of data to CSV files.

However, it is also stochastic, in that a variety of actions are “random” in nature, mediated by the PRNG. In a ModEco-based economy, the following types of action are randomized:

- When buying waste, the MMgr will purchase waste from all consumers in random order.
- When selling waste to frmrs, the MMgr will make sells offers to frmrs in random order.
- When hiring *wrkr*s, frmrs will be activated in random order, and each *frmr* will make job offers to *wrkr*s on its union list, in random order.
- When selling inventory, frmrs will be activated in random order, and a *frmr* will make sales pitches to agents on its consumer list, in random order.

Hypothesis: It is hereby hypothesized that, if a specific initialization of PMM #2 is sustainable with one seed, it is sustainable for all possible seeds. This hypothesis is largely untested, and, possibly, unprovable. It has been tested for a few seed values. It would be interesting if a ‘theory of economic sustainability' could be developed and used to prove or disprove such hypotheses. Perhaps the new studies of resilience of complex adaptive systems will have something to say on such matters, given sufficient time.

3.2.12 Collectives

The obvious identification of collectives in PMM #2 comes from the distinction between *wrkr*s and *frmr*s. Both are needed for a ModEco-based economy to avoid collapse. It is relatively easy to construct an economy in which one sector dominates, while the other collapses, only to be followed inevitably by the collapse of both sectors. These two collectives must mutually support each other for long-term sustainability.

But, there is another type of collective. Agents will naturally pack into dense concentrations of agents, often to their own detriment. (See section 3.2.4.4) In PMM #2 such spatial concentrations alter the biophysical environment, causing higher rates of failure to reproduce. This is an emergent behavior and not part of the design intent.

3.2.13 Observations

A large number of data output options, for both real-time and later observations, are available for PMM #2. (See sections 3.3.6.5 and 3.3.6.6.)

3.3 Details

The ODD protocol defers description of the technical details of implementation to the final section.

3.3.1 Terminology

For the purposes of this paper, the concepts of variable, parameter, entity, state variable, derived variable and scale are understood to have meaning as follows:

- **Variable** – This can have two similar but not identical meanings:
  - **Computer variable** – In computer science literature a variable is a memory address in which a number or string can be stored and accessed by code as it executes. A number or string stored at such an address is said to be soft-coded. A number or string that is stored in the code itself, rather than in an address accessible by the code, is said to be hard-coded. When emulating a real-world system, a computer program might use computer variables to store system constants or system variables.
  - **System variable** – In systems science literature a variable is a measurement of a characteristic of the system that changes dynamically with time.
- **Explicit parameter** - This is a computer variable which is included in the user interface. It is given a value before a run of a model, and its value does not usually change during the run. Setting the value of the parameter would require the user to adjust a switch or slider, or select an option from a chooser device, or change a value in an input field. No changes to the code are needed to alter such explicit parameters.
- **Entity** – This is an ODD-specific word that means a logical collection of computer variables and code in the model that together perform a function which, in some fashion, is analogous to the form and/or function of a real-world concept. This definition of an entity applies to the model as a whole, or conditionally to a subset of the model. The model changes as time passes and the variables within the entities may change values.
- **State variable** – In systems science a state variable is distinguished from other variables by the fact that it is a member of a subset of all possible variables of the system, which subset spans the state space non-redundantly. That is to day, a set of **state variables** is a smallest subset of the set of all

variables needed to fully describe the state of the system. When a computer model is viewed as a system, in its own right, then the state variables of the model must be stored in computer variables.

- **Scale** – Scale is an required ODD-specific word. The concept of scale has two closely related meanings in association with a computer variable:
  - It can refer to the range of values that a state variable of a model is able to assume. For, example, the state variable length may have values ranging from 10^{-20} to 10^{+20}, depending on units of measure and the nature of the model. So, scale might refer to the extent of the domain of a variable. The state space does not extend infinitely in all dimensions. It has practical finite limits in every dimension, in every state variable.
  - But, the concept of scale also refers to the analogies that we draw between entities in the model and the things they simulate in the real world. How does a small change in a state variable in the model relate to an analogous change in the real world? For example, does one tick of discrete time in the model represent one microsecond in the real world, one second, one day, one year?
- **Parameter space** – This is the Cartesian cross product of all explicit parameters. For example, if there are twenty-nine explicit parameters, then the parameter space has 29 dimensions with limited extent in each dimension determined by the allowed values of the associated parameter. A point in the parameter space is formed as the ordered 29-tuple of 29 parametric values. A point in the parameter space explicitly defines one model economy.
- **State space, or Phase space** – This is Cartesian cross product of all state variables of a system or computer model. For example, if there are 920 state variables in a model, then the state space has 920 dimensions with limited extent in each dimension determined by the allowed values of the associated state variable. A point in the state space is formed as the ordered 920-tuple of measured values. A point in state space explicitly defines one state of the model.
- **Trajectory of a model** – This is the time series of states, (of n-tuples, of points in the state space) that are assumed by the model as a run proceeds, one n-tuple per tick of the model.

The ODD requires only that state variables be identified and described. I have taken some effort to describe all variables relevant to the replication of this model because of its complexity, but to explicitly identify the state variables among them.

As much as is possible, I have used the names given to the associated computer variables as the names of the explicit parameters, dynamic state variables, dynamic derived variables. Those names follow the NetLogo naming standards I developed while writing the code. (Boyle, 2014 [07])

3.3.2 Entities, State Variables and Scales

A high-level summary of the entities, state variables and scales is provided in section 3.1.2. (See Table 01.) Here I provide substantially more detail.

3.3.2.1 ModEcoSys

The word “ModEco” is a shortened version of noun phrase “Model Economies”. This implementation of ModEco (NetLogo ADE) can demonstrate several slightly different models as features are toggled on or off. This is a subset of the set of model economies that can be demonstrated using ModEco (C++ ADE). Within this document, unless explicitly differentiated, *ModEco* refers to the NetLogo application that presents all such models, whether sustainable or not, with all of their parameters, optional toggled features, and contents.

There is, in fact, no entity within ModEco (NetLogo ADE) that is called ModEcoSys. *ModEcoSys* can be considered to exist as the collection of explicit and implicit parameters, static and dynamic variables, static and dynamic entities that make up the ModEco application.

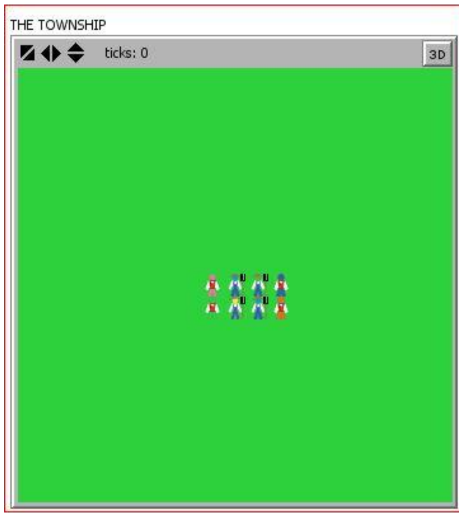
There are a number of variables that control entities that play a supporting role for PMM #2, but which have no material effect on the outcome of a run. For example, they may control real-time data display, data collection for later analysis, or debug-time output. These non-participating entities are discussed in section 3.3.2.7 below.

The static and dynamic characters of the fictional entity call ModEcoSys are listed in Table 04

Table 04 – The Static and Dynamic Characters of ModEcoSys

Name	Type	Brief Description
ticks	built-in state variable	The count of time increments since inception, implemented as a built-in NetLogo reporter function. It advances by 1 each tick. A tick is divided into eight steps that follow each other in order within each tick.
township	Dynamic entity	The 2-dimensional rectangular set of patches which is divided into *Lot*s (patches) in which the action happens (see section 3.1.2.2 below).
wrkrs	Dynamic entity	An agent-set, a breed of agent (see sections 3.1.2.4.1.1 and 3.1.2.7 below).
frmrs	Dynamic entity	An agent-set, a breed of agent (see sections 3.1.2.4.1.2 and 3.1.2.7 below).

Figure 06 – The Township



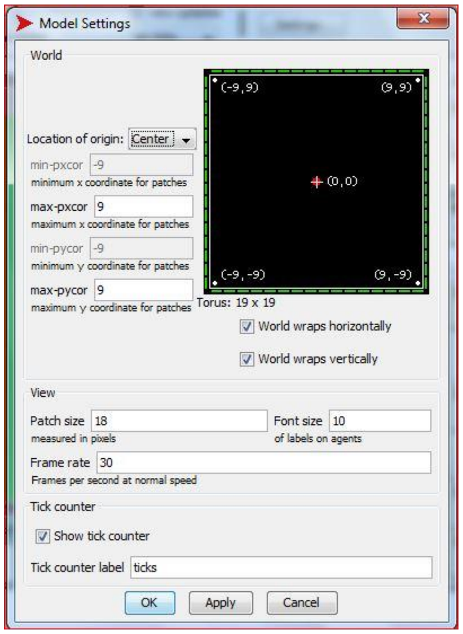
The *Township* forms a green square consisting of *Lots* which can be occupied by *wrkrs* and *frmrs*.

3.3.2.2 Township

The action happens in a rectangular array of patches (or Lots) called the *township*. Such a rectangular array is a standard component of most NetLogo models. The left and right sides of the *township* are wrapped together, so that a *Lot* at the left edge of the *township* is considered to be immediately to the right of a *Lot* at the right edge, and vice versa. Similarly, the top and bottom edges are wrapped. The *Township* then forms the topological equivalent of a toroid. This technique, while seemingly a bizarre practice, reduces the location-specific effects for *Lot*s situated at the edges of the *Township*, and makes analysis of spatial effects dramatically more simple.

The *township* entity is a standard built-in component of a NetLogo user interface, and is characterized by parameters that are set in the associated built-in dialogue box. These include:

Figure 07 – Model Settings



Model Settings

World

Location of origin: Center

min-pxcor -9
minimum x coordinate for patches

max-pxcor 9
maximum x coordinate for patches

min-pycor -9
minimum y coordinate for patches

max-pycor 9
maximum y coordinate for patches

Torus: 19 x 19

[✓] World wraps horizontally

[✓] World wraps vertically

View

Patch size 18
measured in pixels

Font size 10
of labels on agents

Frame rate 30
Frames per second at normal speed

Tick counter

[✓] Show tick counter

Tick counter label ticks

The standard "Model Settings" dialogue of NetLogo controls the dimensions and behaviour of the township.

Table 05 – Model Settings – Static Characters of the Township

Name	Type	Brief Description or Setting
Location of Origin	Parameter	Center
min-pxcor	Parameter	The minimum x coördinate for patches is -9.
max-pxcor	Parameter	The maximum x coördinate for patches is 9.
min-pycor	Parameter	The minimum y coördinate for patches is -9.
max-pycor	Parameter	The maximum y coördinate for patches is 9.
Horizontal wrap	Toggle	This is a tick box, marked as ticked.
Vertical wrap	Toggle	This is a tick box, marked as ticked.
Patch size	Parameter	The width of the square patches, in pixels is 18.
Label font size	Parameter	Point size of labels used for labels on agents.
Frame rate	Parameter	The frames per second at ‘normal’ speed is 30.
Tick counter display	Toggle	This is a tick box, marked as ticked.
Tick counter label	Label	This is a user-entered text, “ticks”, which is the default value.
Dimension	Toggle	2 Dimensional.

These values are all set on initialization when the *township* is constructed, and do not change as the economy changes. They are not state variables.

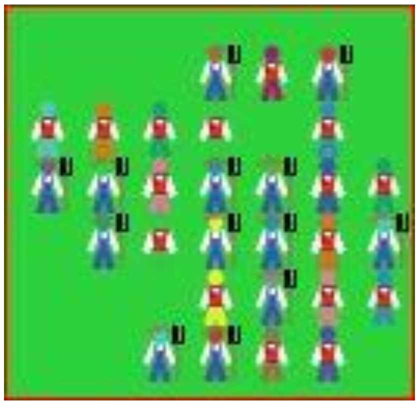
3.3.2.3 Lots

These entities have different names arising from different types of jargon:

- In the NetLogo ADE these are called patches.
- In a ModEco model economy they are called lots.

Each NetLogo patch represents a piece of real-estate property or *lot* within the *Township*. There are three types of *lot*: unspecified (default), residential and farmed. The lots are passive. Their states are controlled entirely by the action of the mortal agents that may or may not occupy them. The purpose of the spatial structure is to simulate geographic dispersion of the agents.

Figure 07 – Lots



Some lots. Unspecified lots are clear green. Farm lots have a *frmr* with a pitch fork. Residential lots have up to four *wrkrs*, only one of which is visible

In the NetLogo ADE much of the technical machinery for managing the associations between agents and *lots* is built-in. Table 06 lists the static and derived dynamic variables that are associated with a lot, together with the one explicit parameter from Table 21 that is relevant to lots.

There are no resources associated with a *lot*. Most relevant variables are assigned a value on setup and do not change during a run. The two derived variables p-no-of-wrkrs and p-no-of-frmrs represent states of occupation only. Figure 07 shows a small group of lots that are either of unspecified use, farms, or residences.

For the purposes of efficiency, each *lot* contains a logical list of all other *lot*s within its 2-neighbourhood. This list is established on start-up and does not change. It is called the commuting area of the *lot* and is implemented in the entity *com-area*. It recognizes the toroidal nature of the *Township* and embodies the tricky task of figuring out exactly which agents are in the 2-neighbourhood of which other agents. Two mortal agents found within each other's commuting areas are able to interact, via the medium of their associated *lot:com-area*.

Table 06 – Static and Dynamic Characters of a Lot

Name	Type	Brief Description
Relevant Explicit Parameters		
g-no-of-rents-max	explicit parameter	This is a slider parameter with a default value of 4. It constrains the maximum number of wrkrs that can live together on a single residential lot, and influences other aspects of frmr/wrkr relationships. A frmr and four wrkrs can live on two lots, and, by design, this is intended to be a sustainable economic unit. One of many parameters identified in Table 21.
Built-In NetLogo Variables		
pxcor	fixed value	The x coordinate of the patch.
pycor	fixed value	The y coordinate of the patch.
pcolor	fixed value	The colour of the patch.
plabel	fixed value	The label that will be used. Not invoked in ModEco.
plabel-color	fixed value	The colour of the label. Not invoked in ModEco.
ModEco-Specific Variables		
p-no-of-wrkrs	dynamic derived variable	Zero for unspecified and farm lots. For residential lots, from 1 to g-no-of-rents-max, the number of *wrkr*s residing on the residential *lot*.
p-no-of-frmrs	dynamic derived variable	Zero for unspecified and residential lots. For farm *lots*, it's always 1, the number of *frmr*s residing on the farm *lot*.
com-area	fixed entity	A patch-set (set of NetLogo patches) containing the 25 lots in the 2-neighbourhood of the lot. This is established when the township is constructed, and does not change.

The *lots* have no direct effect on each other, but the concept of a 2-neighbourhood is borrowed from cellular automata and used to define the entity *lot:com-area* (commuting area) for the mortal agents. In a cellular automaton, the k-neighbourhood of cell A is the set of all cells within a distance of k of cell A, measured moving through both edges and corners. In *ModEco*, the commuting area of a lot is the 2-neighbourhood of the lot. A 2-neighbourhood contains 25 lots, including the central lot of interest. The size of the commuting area is implicitly hard-coded in PMM #2 with implied k=2, one of the few instances of a hard-coded parameter. I was unable to easily determine how to parameterize this concept in NetLogo.

In PMM #2, the commuting area of a lot controls the geographic extent of the economic knowledge of the co-located agents. This value for k (i.e. k=2), together with the *g-no-of-rents-max* global variable (described below), implicitly control social effects such as economic enfranchisement, agent huddling, population density, and delayed reproduction due to overcrowding.

Up to *g-no-of-rents-max* (=4) *wrkrs* can live in a residential *lot*. When the first wrkr moves into an unspecified *lot*, it becomes a residential *lot*. Thereafter up to three more wrkrs may live in the *lot*. When the last *wrkr* in such a *lot* dies, the *lot* reverts to type unspecified.

3.3.2.4 The economic agents

Within the *township* there are four different types of economic agents. These agents can be classified as mortal or immortal, each having slightly different characteristics. The mortal agents are *wrkr*s and *frmr*s, each of which must reproduce before it dies of old age or starvation, and each of which has access limited only to agents within its commuting area. The immortal agents are the material manager (*MMgr*) and the estate manager (*EMgr*), which are both singular, ageless and can interact with all agents in the economy not limited to any commuting area.

An understanding of agents requires explanation of the classifications of resources in the PMM #2, and the units of measure used for each. The PMM #2 has four key resources that are conserved in all transactions. The four conserved quantities are as follows:

- **Cash** – the units of which are dollars (singular: dollar).
- **Mass** – the units of which are "metabolism-based mass units" or mus (singular: mu).
- **Metabolic energy** – the units of which are "metabolism-based energy units" or eus (singular: eu).
- **Intrinsic value** – the units of which are "dollars" (singular: dollar). The intrinsic metabolic value of an mu and an eu are each fixed from the start, as explicit parameters.

At the metabolic layer, mass and energy (with their accompanying intrinsic value) flow in cycles through the system, being continually recycled. At the economic layer, cash flows in cycles through the system, also being continually recycled. We say the system is closed with respect to these four conserved quantities. Mass, energy and intrinsic value flow in the opposite direction to cash. However, an optional toggle will allow the MMgr to go into debt, effectively opening the system with respect to cash, while at the same time partially decoupling the two sub-systems. In this case, cash is still conserved on a transaction-by-transaction basis and at a system level, but includes negative amounts.

In ModEco, mass is categorized as being in one of two possible conditions. Either it contains energy which can be used to sustain the life of agents, or it does not. The energy referred to does not include kinetic energy, heat, or gravitational potential energy, but specifically refers to chemical potential energy that can be (a) contained in food and metabolized by agents, or (b) contained in the muscles of the agents. For the purposes of this description of PMM #2, all references to energy mean this kind of metabolically accessible energy. Then, in PMM #2, mass plays the role of being the vehicle by which energy is collected during harvesting, delivered to the consumer, and delivered to the muscles when eaten. In a wonder of physics, rather than this energy being degraded and emanated into space on use, it is embedded back into the product of work (harvested grain) as it is expended, thus completing the energy cycle. Mass can then be viewed as a vector that carries metabolic energy with a fixed capacity to do so.

NOTE: I do realize that such an 'energy cycle' as described above is ludicrously unreal. At the same time, I argue that it is not a totally unrealistic model of renewable energy sources. The real focus of the model is the economic sub-system. But, a more realistic biophysical sub-system with linear flows of mass and energy developed as part of ModEco (C++ ADE) has, so far, proven to be unsustainable. I have not been able to model an open mass-energy sub-system coupled to a sustainable economic sub-system.

Mass which is devoid of metabolic energy is measured in metabolism-based mass units (mus). One mu has, arbitrarily, an intrinsic value of exactly $1, as set in the global variable *g-mu-price*.

Metabolic energy is measured in metabolism-based energy units (eus). One eu has, arbitrarily, an intrinsic value of exactly $1, as set in the global variable *g-eu-price*.

Mass which is carrying its full complement of metabolic energy is measured in metabolism-based mass/energy units (meus; singular: meu). One meu is equal to one mu plus one eu. The intrinsic value of an meu is $2 = $1 (the intrinsic value of the mu) + $1 (the intrinsic value of the eu), as set in the global variable *g-meu-price*.

NOTE: This is a slightly simplified version of the pricing used in the PMM #1.

The concept of an meu is based on metabolic needs. An agent needs to consume a pre-determined amount of mass and energy with each tick (time increment) in order to survive. Metabolic processes require a mixture of mass and energy. While each can be measured separately, a common unit of measure was required to model their relative roles, and more importantly, to value them economically. The metabolic value of food does not change dramatically over time, while, in most economies (but not PMM #2) the monetary value of food can change dramatically. A *wrkr* needs a pre-determined number of life-sustaining units of mass each tick, and another pre-determined number of life-sustaining units of metabolic energy each tick. With some arbitrariness, it was decided that a typical worker will require 4 meus to live (as set in the global variable *g-EPT*), comprising 4 mus and 4 eus, having an arbitrary value of $8. Ergo, a *wrkr* must earn and spend, on average, $8 per tick to stay alive. To balance that, a *frmr* must consume 16 meus (*g-no-of-hires-max* * *g-EPT*) per tick to live, and so must earn and spend $16 per tick.

In ModEco (C++ ADE) we track both intrinsic value based on metabolic needs and monetary value based on weighted average cost. In ModEco (NetLogo ADE) there is no distinction between intrinsic and monetary value, as the focus is on replication of the PMM#1 in the PMM #2. Both kinds of value are mentioned in this description to clarify the need for and role of intrinsic value in other less restrictive ModEco-based economies.

3.3.2.4.1 The mortal agents

There are two types or breeds of mortal agents: *wrkr*s and *frmr*s. These breeds are co-dependent, which is to say, if one breed dies out, the other cannot survive long. These agents are the atomic components of the economy, and a thriving economy contains many of both breeds.

Table 07 – A Comparison of the Mortal Agents

Agent Name	Description	Geographic characteristics	Temporal characteristics	Breadth of Knowledge (geographic and temporal)
wrkr	A worker; a consumer; sells energy to frmrs; buys supplies from frmrs; sells waste to the *MMgr*.	Lives on a residential *lot* with up to three other *wrkr*s; cannot move or relocate; reacts with frmrs within its commuting area.	Mortal; reproduces via fission; dies of starvation or old age or bankruptcy.	Local; agents within its own commuting area; no temporal knowledge.
frmr	A farmer who works the land; a consumer; buys recycled mass from the *MMgr*; hires *wrkr*s; sells inventory to consumers; sells waste to the *MMgr*; buys recycled mass from the *MMgr*.	Lives alone on a farmed *lot*; cannot move or relocate; reacts with wrkrs and frmrs within its commuting area.	Mortal; reproduces via fission; dies of starvation or old age or bankruptcy.	Local; agents within its own commuting area; no temporal knowledge.

The mortal agents inhabit *lot*s in the *township*, causing the *lots* to differentiate between unspecified *lot*s, farmed *lots*, or residential *lot*s. They actively engage in commercial transactions with other mortal agents within their respective commuting areas (*lot:com-area*). They have metabolic needs which, when met, allow them to reproduce, and if not met, will cause them to die. Their only purpose is to live long enough to reproduce. In ModEco, an economy is considered collapsed (and unsustainable) when either breed dies out. Table 07 outlines the similarities and differences between the breeds of mortal agents. Compare this with Table 14.

The activities of both *wrkr*s and *frmrs* are restrained by a large number of explicit parameters. Most of these were soft-coded during the design cycle, and then made explicit as I tidied up the code for others to see. They are presented in the sections on sub-models, but you might want to scan them prior to reading about the details of the mortal agents. See:

- Biophysical sub-system – Section 3.3.6.1 – Table 19.
- Economic sub-system – Section 3.3.6.2 – Table 21.

NetLogo is designed for ABMs, so the agents are all native structures having a minimal set of characters (variables with pre-defined purpose), some of which can vary, and some of which do not. In addition, the modeller can define 'breeds' of agents and assign breed-specific variables to each type of agent. In ModEco, the two breeds are wrkrs and frmrs. In the tables indicated, I identify four kinds of variables for these breeds as follows:

- **Static Characters** – Those variables that are assigned a value for the agent at birth, and never change, being largely descriptive, but possibly having some role in determining the ability of the agent to survive to reproduction. Four of these variables are state variables from the perspective of the model. (See Table 08).
- **Dynamic State Variables** – Those dynamic state variables that can change from tick to tick, determining the current state of the agent moment-by-moment. All of these are state variables. (See Tables 09 and 11.)
- **Derived Dynamic Participation Booleans** – Such variables can be considered temporary flags that serve an immediate purpose and then are returned to a default value. I call these participation Booleans. An example would be the flag wrkr:b-can-buy-supplies to indicate whether a wrkr can or has purchased supplies from some frmr so far in the 'sell inventory' step of this tick. A negative flag would exclude that wrkr from further purchases of supplies in this step of this tick. None of these are state variables. (See Table 09.)
- **Other Derived Dynamic Variables** – Those dynamic derived variables that record some fleeting aspect of the state of the agent at some point during the tick. None of these are state variables. (See Tables 10, 12 and 21.)

The wrkrs and frmrs (the two breeds of mortal agents) have a number of characteristics in common which are fixed at birth, identified in Table 08. Many of these, labelled 'built-in', are determined automatically by the NetLogo ADE when an agent is sprouted, and play no real role in the model. They are listed in Table 08 for completeness.

Table 08 – Static Characters of Mortal Agents Fixed at Birth

Name	**Type**	**Brief Description**
who	built-in	A unique identifying handle for the agent.
color	built-in	A color chosen from a palette of colors used to paint the agent.
heading	built-in	Not used in ModEco.
xcor	built-in state variable	The x coordinate of the lot in which it lives.
ycor	built-in state variable	The y coordinate of the lot in which it lives.

(Continued on next page.)

Table 08 – Static Characters of Mortal Agents Fixed at Birth (Continued)

Name	Type	Brief Description
shape	built-in	The shape, selected from the shapes library, used to display the agent.
label	built-in	A suitable label for display with the agent.
label-color	built-in	The color used to paint the label.
breed	built-in state variable	Always wrkr – defines this agent as a wrkr
hidden?	built-in	A Boolean flag – Not used in ModEco
size	built-in	Scales the size of the shape
pen-size	built-in	Not used in ModEco, as the agents do not move.
pen-mode	built-in	Not used in ModEco, as the agents do not move.
age-to-reproduce	ME-specific state variable	The earliest age, in ticks, at which this agent can possibly reproduce. It is computed at time of birth as (g-RAT - (g-GTT / 2) + (random g-GGT))
ma-who	ME-specific	The who number of the mother of this agent.
da-who	ME-specific	The who numbers of the daughters of this agent, assigned fleetingly as the agent reproduces, and just before this agent is removed from the model.
my-patch	Static entity	The patch which is the residential lot in which this agent sits. Redundant, due to xcor and ycor. Used for efficiency.

Four of these characteristics (shaded blue) are like dynamic state variables from the point of view of the model as a whole, but have a static or fixed value from the point of view of the agent. For example, if an agent is born into a lot, and lives in that lot until it dies, xcor and ycor do not change throughout the life of the agent and are not dynamic variables from the point of view of the agent. But they do change the state of the model when the agent is placed there, and again when the agent is removed. The xcor and ycor variables are then state variables, though static in value for each agent.

The mortal agents, wrkrs and frmrs, are also characterized by a set of Boolean flags that are used, each within a single step. I call these ‘participation Booleans’, since they indicate the ability of an agent to participate in a particular type of biophysical action or commercial transaction (collectively referred to as x-actions). These flags are either on (1 or true) or off (0 or false). These are derived variables that flip on and off rapidly, having only fleeting meaning. The variables for both wrkrs and frmrs are presented in Table 09 for ease of reference.

Table 09 – The Derived Dynamic Participation Booleans of Mortal Agents

Variable Name	Type of agent	Step	Brief Description
b-can-sell-waste	both	buy recycled sell waste	A consumer has waste available to be sold in the waste market’, and has not yet sold any this tick.
b-can-buy-recycled	frmr	buy recycled sell waste	A frmr has a shortage of recycled, and has cash available to purchase recycled mass in the market’, and has not yet purchased any this tick.
b-can-work	wrkr	hire wrkrs	A wrkr has energy and is available for hire in the labour market’ and has not yet been hired this tick.
b-can-sell-inventory	frmr	sell inventory	A frmr has inventory to sell in the inventory / supplies market’.
b-can-buy-supplies	both	sell inventory	A consumer has a shortage of supplies, and has cash available to purchase supplies in the inventory/supplies market’, and has not yet purchased any this tick.

(Continued on next page.)

Table 09 – The Derived Dynamic Participation Booleans of Mortal Agents (Continued)

Variable Name	Type of agent	Step	Brief Description
b-can-eat	both	eat supplies	A consumer has sufficient supplies (greater than or equal to g-EPT) to consume this tick, and has not yet eaten any this tick.
b-can-reproduce	both	reproduce	An agent is old enough (Age > g-RAT) and healthy enough (energy > g-RET) to reproduce, and has not yet reproduced this tick.
b-just-sprouted	both	reproduce	An agent was sprouted and has not yet been initialized.
b-can-die	both	death	An agent is emaciated (energy < g-DET) or overaged (age > g-DAT) or starved (g-can-eat = false) but has not yet died.
b-can-die-of-hunger	both	death	An agent is starved (g-can-eat = false) and will die this tick, but has not died yet.
b-can-die-of-age	both	death	An agent is too old (age > g-DAT) and will die this tick, but has not died yet.
b-can-die-of-lownrg	both	death	An agent is emaciated (energy < g-DET) and will die this tick, but has not died yet.

There is one more set of variables in the wrkrs and frmrs that are used to manage municipal grants. See the description of the role and actions of the EMgr in section 3.3.2.4.2.2, or the description of the grant management system in section 3.3.6.3 for details about these variables and how they are used.

3.3.2.4.1.1 wrkr*s*

wrkrs are mortal agents, so everything described in section 3.3.2.4.1 applies to them as well as to the *frmrs*. As biophysical agents they work as they harvest grain, they eat supplies, they reproduce via fission when mature and healthy, or they die of old age. As economic agents they sell waste mass, they hire themselves out to work (selling their energy), they purchase supplies for their own consumption. wrkrs and frmrs are co-dependant in the economy.

The behaviour of wrkrs is constrained by:

- The static variables associated with each agent, including four state variables (Table 08);
- The dynamic participation Booleans associated with each agent (Table 09); and
- The explicit parameters of the biophysical sub-system (Table 19);
- The explicit parameters of the economic sub-system (Table 21);
- The dynamic variables associated with each agent as part of the grant management sub-system (Table 22).

Of all of these variables, only four of the static variables from Table 09 (*breed*, *xcor*, *ycor* and *age-to-reproduce*) are dynamic state variables. The rest are parameters or dynamic derived variables. The four state variables are determined at time of birth of each agent as follows:

- The breed of each of the two daughters produced via fission are the same as the breed of the parent that produced them. wrkrs beget wrkrs. frmrs beget frmrs.
- The first daughter occupies the lot occupied by the parent, and so inherits the same xcor and ycor. The second daughter occupies another available lot within the commuting area of the parent. If there is no available lot, the parent cannot reproduce.
- The variable called age-to-reproduce is calculated at the birth of each agent and then fixed for life. It is calculated as a base value plus a random value. The base value is (g-RAT – (g-GTT / 2)). The random value is g-GTT. The values are then randomly distributed about the parameter g-RAT with a

uniform probability distribution. The birth and death of agents can become highly synchronized, causing a increasingly large undulation in the population, making it unstable. This randomizing procedure was introduced into the biophysical sub-system to break up the population waves and stabilize the economy. This particular technique was one of several minor tweaks made in the search for sustainability. Once sustainability was achieved, it was not removed. It may or may not be necessary.

In Tables 09 and 10 the remaining dynamic variables and entities associated with each wrkr are listed. Table 10 is a key table, listing the five remaining state variables associated with a single wrkr. These are state variables from the point of view of the agent, as well as from the point of view of the system. That is to say, they change dynamically during the life of the agent. The variables in Table 11 are derived variables, the values of which are determined using state variables as inputs to mathematical or logical processes.

Table 10 – The Dynamic State Variables of wrkrs

Name	Units	Brief Description
State Variables		
age	Ticks	The time, in ticks, since birth of this agent.
cash	$	The agent's stock of cash, measured in dollars.
energy	eus	The agent's stock of energy, measured in eus.
supplies	meus	The agent's stock of supplies, measured in meus.
waste	mus	The agent's stock of waste mass, measured in mus.

Other than age, these variables are the stocks wherein are held the assets of the wrkr – four asset classes in all.

Table 11 – The Dynamic Derived Variables and Entities of wrkrs

Name	Type	Units	Brief Description
State Variables			
net-value	derived variable	$	The dollar value of all of the agent's stocks, computed as needed, changing throughout each tick.
real-value	derived variable	$	Vestigial – The dollar value of all of the agent's stocks, except for cash, computed as needed, changing throughout each tick.
HSL	derived variable	meus	Hold Supplies Limit – The threshold level of supplies that triggers non-purchase of supplies. If an agent has an economic opportunity to purchase supplies, but the current stock exceeds this level, the opportunity is declined. This is computed as needed using HSL = g-HSF * net-value, but cannot be smaller than g-HSL-w-min.
energy-set-point	derived variable	eus	A variable that starts at zero and increases with each tick that is used to protect energy from early use, saving it for use in reproduction. When a wrkr is hired to work, only stocks of energy above the set point is made available for work.
available-energy	derived variable	ues	Computed each tick as available-energy = energy – energy-set-point.
employer-set	derived entity	agents	An agent-set of all frmrs within the commuting area (com-area) of my-patch. The wrkr can work for only one of these frmrs per tick, at most. Since wrkrs are passive purchasers, this is maintained but not used.
supplier-set	entity	agents	An agent-set of all frmrs within the commuting area (com-area) of my-patch. The wrkr can purchase supplies from only one of these frmrs per tick, at most.

wrkrs are directly affected by the two explicit parameters g-HSF-w and g-HSL-w-min. These two parameters are given a fixed value at setup and do not change throughout a run. They are used, together with the derived variable wrkr:net-value, to compute the state variable wrkr:HSL during the 'buy supplies' step of each tick. This is an extremely important state variable used to enable/enforce diversification of the holdings of an agent across key asset classes. It is used to prevent an agent from putting too large of a percentage of its net-value into supplies, or too few meus into supplies. There is an upper limit on the value of supplies, relative to other asset classes, and a lower limit on the absolute number of supplies, relative to metabolic needs. If supplies rise above the relative limit, the agent stops buying supplies. If the supplies fall below the absolute limit, the agent ignores the relative limit and purchases supplies at every opportunity.

frmrs have three similar derived variables (*frmr:HRL*, *frmr:HIL* and *frmr:HSL*). All four of these are handled in a similar way, so the following paragraphs are equally applicable to the frmr's three variables. The many explicit parameters used to manage this process are sumarized in Table 21. The process used to compute, and then use this *wrkr:HSL* variable to regulate the purchase of supplies is as follows:

- **EVALUATE HSL** – At the beginning of the 'buy supplies' step, the variable HSL is computed for each consumer, both wrkrs and frmrs.
  - First, using wrkr:net-value as the example, the derived variable wrkr:net-value is calculated as the sum of the intrinsic value of all stocks owned by the agent: wrkr:net-value = wrkr:cash + (wrkr:energy * g-eu-price) + (wrkr:supplies * g-meu-price) + (wrkr:waste * g-mu-price).
  - Then, an interim value of HSL is calculated as a product of the business factor and the net value: *wrkr:HSL* = (*g-HSF-w* * *wrkr:net-value*).
  - Then, if the relatively-sized limit *wrkr:HSL* is less than *g-HSL-w-min*, *wrkr:HSL* is set equal to *g-HSL-w-min*, the absolute lowest limit.
  - Thus, HSL is set as a fraction of the net-value, but no lower than the allowed minimum.
- **EVALUATE ABILITY TO PARTICIPATE IN MARKET** – Then the wrkr is evaluated for suitability to enter the 'buy supplies' market. There are two conditions that must be met – a wrkr must demonstrate the need to rebalance its portfolio of assets through purchasing supplies, and it must demonstrate its ability to afford the needed assets.
  - Default decision: It is assumed the wrkr is able to participate in the market. It is flagged as so able, and the flag variable wrkr:b-can-purchase-supplies is set to 1 (true). This is a participation Boolean that has only two states: 0 (false) or 1 (true).
  - Rebalancing the portfolio decision: The size of the *wrkr*'s stock of supplies is then compared to the 'hold supplies limit'. If *wrkr:supplies* is more than wrkr:HSL, then the wrkr has an overage of supplies on hand and is flagged as not able to purchase more supplies. The variable wrkr:b-can-purchase-supplies is set to 0 (false).
  - Availability of cash on hand decision: The *wrkr* must also have some money. So, if wrkr:cash is zero, the variable wrkr:b-can-purchase-supplies is set to 0 (false).
  - Other consumers: An identical process is applied to the *frmrs* and their state variables *frmr:net-value*, *frmr:supplies*, and *frmr:HSL*, frmr:cash and frmr:b-can-purchase-supplies.
- **OPERATE THE MARKET** – Once all consumers have been flagged as able or unable to participate in the sell inventory/buy supplies market, the frmrs, in random order, each attempt to sell their inventory to able consumers, each in their own commuting area.
  - A list of potential customers (consumers) within the frmr's commuting area is built. Note that a frmr is eligible to be its own customer, and they are otherwise disallowed from consuming their own inventory. They can only consume supplies purchased through the inventory/supplies market.
  - In random order, a customer is selected and serviced.

        - The frmr and consumer both check with the EMgr to see if grants are available, and receive them, if needed and available.
        - A transaction up to the allowed quota per transaction (g-supplies-purchase-quota) is completed.
        - The frmr and consumer each return unused grants to the EMgr, if any.
        - The consumer is flagged as no longer in the market as the flag b-can-purchase-supplies is set to 0 (false). This excludes them from being included in the customer list of any other frmr as the process proceeds.
    - If the frmr still has unserviced consumers within its commuting area and it still has inventory to sell, the next consumer is selected randomly from its list, and serviced, until all consumers have been so serviced, or until all of its inventory is sold.

Understandably, this appears complicated, because it is complicated. But, it is necessarily so to meet the design goals of demonstrating a market for food in a most simple, complete, conservative, and sustainable agricultural economy. I hypothesize that each of the parameters and state variables mentioned above plays a necessary role if sustainability is to be achieved.

3.3.2.4.1.2 frmr*s*

frmrs are mortal agents, so everything described in section 3.3.2.4.1 applies to them as well as to the *wrkrs*. As biophysical agents they work as they harvest grain, they eat supplies, they reproduce via fission when mature and healthy, or they die of old age. As economic agents they sell waste mass, they buy recycled mass for their farm, they hire wrkrs, they sell inventory to consumers, they purchase supplies for their own consumption. frmrs and wrkrs are co-dependant in the economy.

The behaviour of frmrs is constrained by:

- The static variables associated with each agent, including four state variables (Table 08);
- The dynamic participation Booleans associated with each agent (Table 09); and
- The explicit parameters of the biophysical sub-system (Table 19);
- The explicit parameters of the economic sub-system (Table 21);
- The dynamic variables associated with each agent as part of the grant management sub-system (Table 22)

In Tables 11 and 12 the remaining dynamic variables and entities associated with each frmr are listed. Table 12 is a key table, listing the seven remaining state variables associated with a single frmr (i.e. in addition to the four identified in Table 08). They change dynamically during the life of the agent. The variables in Table 13 are derived variables, the values of which are determined using state variables as inputs to mathematical or logical processes.

Table 12 – The Dynamic State Variables of frmrs

Name	**Type**	**Brief Description**
age	state variable	The time, in ticks, since birth of this agent.
cash	state variable	The agent’s stock of cash, measured in dollars.
energy	state variable	The agent’s stock of energy, measured in eus.
recycled	state variable	The agent’s stock of recycled mass, measured in mus.
inventory	state variable	The agent’s stock of inventory, measured in meus.
supplies	state variable	The agent’s stock of supplies, measured in meus.
waste	state variable	The agent’s stock of waste mass, measured in mus.

Other than age, these variables are the stocks wherein are held the assets of the frmr – six asset classes in all.

Table 13 summarizes the dynamic derived variables in frmrs. The HRL, HIL and HSL variables are used to manage the participation of a frmr in three markets and enable a frmr to hold a diversified portfolio of economic assets based on current net-value. frmrs participate in a market only if the relevant stock (e.g. recycled mass) is less than the computed limit (e.g. HRL = Hold Recycled Limit). While such a mechanism is necessary to enable wealthy frmrs to thrive, unfortunately, it tends to hinder the survival of the extremely poor frmrs, disenfranchising them in critical markets when they are in desperate circumstances. For such agents, diversification of the stock portfolio is not the main concern. A system-wide set of fixed minimum values for these 'hold stock limits' seems to have improved the ability of poor frmrs, in particular, to survive and grow wealthy.

Table 13 – The Dynamic Derived Variables and Entities of frmrs

Name	**Type**	**Brief Description**
net-value	derived variable	The dollar value of all of the agent's stocks, computed as needed, changing throughout each tick.
real-value	derived variable	The dollar value of all of the agent's stocks, except for cash, computed as needed, changing throughout each tick.
HRL	derived variable	Hold Recycled Limit – The threshold level of recycled mass that triggers non-purchase of recycled mass. If an agent has an economic opportunity to purchase recycled mass, but the current stock exceeds this level, the opportunity is declined. This is computed as needed using HRL = g-HRF * net-value, but cannot be smaller than g-HRL-w-min.
HIL	derived variable	Hold Inventory Limit – The threshold level of inventory that triggers non-hiring of wrkrs to harvest inventory. If an agent has an economic opportunity to hire wrkrs, but the current stock exceeds this level, the opportunity is declined. This is computed as needed using HIL = g-HIF * net-value, but cannot be smaller than g-HIL-w-min.
HSL	derived variable	Hold Supplies Limit – The threshold level of supplies that triggers non-purchase of supplies. If an agent has an economic opportunity to purchase supplies, but the current stock exceeds this level, the opportunity is declined. This is computed as needed using HSL = g-HSF * net-value, but cannot be smaller than g-HSL-w-min.
energy-set-point	derived variable	A variable that starts at zero and increases with each tick that is used to protect energy from early use, saving it for use in reproduction. When a wrkr is hired to work, only stocks of energy above the set point is made available for work.
available-energy	derived variable	Computed each tick as available-energy = energy – energy-set-point.
union-set	derived entity	An agent-set of all wrkrs within the commuting area (com-area) of my-patch. The frmr can hire only g-no-of-hires-max of these wrkrs per tick, at most.
customer-set	derived entity	An agent-set of all consumers within the commuting area (com-area) of my-patch. The frmr can sell supplies to as many consumers as are able to purchase them until the stock runs out.
supplier-set	derived entity	An agent-set of all frmrs within the commuting area (com-area) of my-patch. The frmr can purchase supplies from only one of these frmrs per tick, at most. As a purchaser of supplies, a frmr is a passive agent. So this entity is maintained but not used.

In section 3.3.2.4.1.1 (wrkrs), the roles of HSL and the other variables that manage wrkr participation in the inventory/supplies market was described in some detail. wrkrs have one such set of variables to manage their participation in that one market. On the other hand, frmrs have three such sets of variables, with associated parameters, to manage their participation in three of the four markets in which they are involved:

- the waste/recycled mass market (HRL et al.),
- the labour/inventory market (HIL et al.), and
- the inventory/supplies market (HSL et al.).

Table 21 identifies the six explicit parameters that manage the participation of the frmrs in each of those three markets (g-HRF-f, g-HIF-f, g-HSF-f, g-HRL-f-min, g-HIL-f-min and g-HSL-f-min). Please refer to that table for details.

3.3.2.4.2 The immortal agents

During the development of PMM #1 and PMM #2 certain roles and procedures were added to the design in the search for a sustainable economy. Eventually, these roles were encapsulated into two singular immortal agents having somewhat more extensive knowledge of the market than is available to the mortal agents. Table 14 summarizes the general characteristics of the two immortal agents, in contrast to the characteristics of the immortal agents. Compare this with Table 07.

Table 14 – A Comparison of the Immortal Agents

Agent Name	**Description**	**Geographic characteristics**	**Temporal characteristics**	**Breadth of Knowledge (geographic and temporal)**
MMgr	A central agent; buys waste from consumers (*wrkr*s and *frmr*s) and sells it to farmers (*frmr*s).	Exists everywhere; no visible location in *township*.	Immortal; neither reproduces nor dies.	Global; moment in time, any agent.
EMgr	A central agent; receives the assets of any agent that dies without issue; distributes those assets to poor but deserving agents.	Exists everywhere; no visible location in *township*.	Immortal; neither reproduces nor dies.	Global; moment in time; any agent.

*3.3.2.4.2.1 The material manager (*MMgr*)*

You can imagine this agent as the central manager of the garbage collection, composting and recycling depot of the *township*. His role is to purchase waste from consumers when they wish to sell it, and resell it to farmers as recycled compost for their fields, when they wish to buy it.

Table 15 – ModEcoSys state variables implementing the MMgr

Name	**Type**	**Brief Description**
g-MMgr-cash	ModEcoSys state variable	The stock of cash, measured in dollars, during a run.
g-MMgr-mass	ModEcoSys state variable	The stock of mass, measured in mus, during a run.

The MMgr is not implemented as a native NetLogo agent. It is not a breed of agent. It is implemented simply as a pair of global state variables, shown in Table 15. Initialization values in PMM #2 are not under user control, but in the 'GROWTH' scenario it is under user control via a pair of explicit parameters (sliders) g-MMgr-cash-at-setup and g-MMgr-mass-at-setup. (See Table 21.)

The MMgr is an active agent, in the sense that it takes the lead in actively buying and selling mass on the market from/to participating agents. However, it is not eligible to receive grants from the EMgr. It cannot die. It does not age. It does not eat. It does not reproduce. It is the only agent that can go into debt, and only if the switch is set to allow that.

Since it cannot receive grants from the EMgr (increasing its net value), nor can it reproduce (decreasing its net value), and since net value of both participants in a transaction is conserved in every commercial transaction, the MMgr has the peculiar property that it's net value is constant throughout a run.

The MMgr is a cornerstone agent in the economy, however, since it completes the cycle for the flow of mass in the economy on a tick-by-tick basis. It must be given a supply of either cash or mass or both during setup, as a float, in order to perform its role. That float must be large enough to handle the per-tick flow of mass from consumers to frmrs. For example, 100 agents may produce 500 mus of waste per tick. The MMgr must have enough float to handle a flow of mass of this volume. This need for a float of resources can be addressed in ModEco in at least three ways:

- If the MMgr is allowed to spend money it does not have (i.e. to go into debt) at any time (default option), then it always has sufficient cash. This is the logical equivalent of giving it a virtually infinite cash float. Then no real cash float is needed due to the ready access to cash. There is no bottleneck on the flow of mass and the MMgr can afford to purchase all the mass made available in the waste/recycled market. This is the setup for the PMM #2. So, the ability of the MMgr to go into debt is sufficient for sustainability, but it's necessity is as yet undetermined.
- If the MMgr is not allowed to go into debt (via a switch), then it must be supplied a large float of cash, or mass, or both on setup. If the float is too small, the MMgr becomes a bottleneck in the flow of mass through the economy putting it in peril of collapse. But, how small is too small? This question has not yet been explored, but there are sliders that allow user experimentation in the 'GROWTH' scenario.
- If the MMgr function is turned off (via a switch), then frmrs buy recycled mass directly from consumers who are selling it as waste mass. Then there is an inefficiency in the mass market which appears to be due to long-term concentration of mass in the centre of the model, and all such economies tested so far have quickly or eventually collapsed. So, it seems the role of the MMgr is necessary for sustainability.

*3.3.2.4.2.2 The estate manager (*EMgr*)*

You can imagine this agent as the central manager who levies estate taxes and redistributes the acquired assets as grants. Its role is to strip all conserved resources (i.e. cash, mass, energy) from a dead agent, store them, and then dole them out in response to requests for grants.

Table 16 – ModEcoSys state variables implementing the EMgr

Name	**Type**	**Brief Description**
g-EMgr-cash	ModEcoSys state variable	The stock of cash, measured in dollars, during a run.
g-EMgr-mass	ModEcoSys state variable	The stock of mass, measured in mus, during a run.
g-EMgr-energy	ModEcoSys state variable	The stock of energy, measured in eus, during a run.

The EMgr is not implemented as a native NetLogo agent. It is not a breed of agent. It is implemented simply as a triplet of global state variables, shown in Table 16. Initialization is under user control via a triplet of explicit parameters (sliders) g-EMgr-cash-at-setup, g-EMgr-energy-at-setup and g-EMgr-mass-at-setup. (See Table 21.)

The EMgr is a passive agent, in the sense that it has no other needs or goals than to respond to the needs of the mortal agents. It is not a commercial agent, never buying or selling anything. It cannot die. It does not age. It does not eat. It does not reproduce.

Like the MMgr, the EMgr plays a critical role in providing a closed loop for the flow of resources. However their roles differ in some significant ways.

- The EMgr handles energy, as well as cash and mass. When an agent dies other than in the process of reproduction, it has no 'heirs' and all of its assets are granted (given) to the EMgr.
- This transaction is not a commercial transaction. The net value of the dying agent plummets from something to nothing, while the net value of the EMgr jumps by an equivalent amount. There is no 'quid pro quo'. The transfer of wealth is a grant from dying agent to EMgr.
- These assets must therefore be given back to surviving mortal agents via non-commercial transactions, called grants, in which the net value of the EMgr drops as the grant is transferred, while the net value of the recipient jumps. The decision respecting to whom these grants are made, and when, becomes a significant kind of economic policy.
- The EMgr needs no initial float. It can simply wait for the first death, then start doling out grants. However, if it is given an initial float, that becomes an endowment that pours additional free resources into an economy as quickly as the economy can sop them up. The 'GROWTH' scenario is an example of this role of the EMgr.
- While the MMgr completes the cycle for the flow of mass and cash on a tick-by-tick basis, the EMgr completes the cycle for the flow of all three resources on a generation-by-generation basis.

The EMgr's economic policy, as implemented in PMM #1 and PMM #2 is to give grants to poor but deserving agents:

- DESERVING – An agent must be able to participate in the economy. Agents that have no cash cannot buy anything. Agents that have no energy cannot work. Frmrs that have no recycled mass cannot hire any wrkrs. If they cannot participate in the economy on their own strengths, they cannot request grants from the EMgr. So, agents must have some modicum of resources in key asset classes in order to remain fully enfranchised. As the net-value of an agent declines, the probability that it will be able to participate in a commercial opportunity declines.
- POOR – Every commercial transaction is regulated (throttled, controlled) by a per-transaction quota of some kind. If an agent is deserving (as defined above) but has insufficient assets to meet quota, they can apply for a grant of the deficient asset class from the EMgr. If it is available, the EMgr will grant sufficient of that asset to meet quota. There may be circumstances in which an agent, nevertheless, is unable to use the entire grant. (E.g. the other agent in the transaction failed to get a matching grant due to lack of resources in the EMgr.) In that case, the unused portion of the grant is returned to the EMgr.

The effect of this 'poor but deserving' economic policy is that assets are given to the poor, but not to the poorest of the poor. When given to the poorest (i.e. to those who are totally economically disenfranchised due to poverty) it prevents the death of many agents, causing populations to rise, and causing poverty to spread, until few are able to reproduce, and collapse ensues. Clearly, such economic policies have ethical implications, as well as implications for sustainability.

So, the role of the EMgr seems necessary for sustainability. It has an intergenerational quality. It would also seem that there are a number of ways that this role can be implemented, and those are imbued with implications for economic and social policy in a sustainable economy.

3.3.2.5 Spatial scale

The spatial (or geographic) scale is set at a square of 19 Lots by 19 Lots. This size is easily adjusted by the user prior to setup. But, since resources do not belong to *lot*s, but rather, they belong to agents, the geographic scale has little effect on the outcome of an economic run unless most of the lots are all fully occupied.

By analogy, one can say a *lot* approximates the amount of farmland required to support five people (i.e. one farmer and g-no-of-hires-max=4 workers).

A commuting area is easily adjusted by a programmer, but is not currently user-selectable. A k-neighbourhood of 2 (a 5x5 area) causes a sustainable economy to be tightly packed into a small area of the landscape.

3.3.2.6 Temporal scale

The question of temporal scales has two aspects: What processes happen in what order during a run? And, how does this relate to timings and durations in the real world?

3.3.2.6.1 Processes and order of execution

Within ModEco, there are several levels of aggregation for time:

- RUN – A run of the application is the execution of a single economic model from setup to collapse, or to user-termination. A run consists of the execution of many ticks in serial order. A run consists of many generations of agents following each other.
- GENERATION – Most agents reproduce at or shortly after they are aged 800 ticks (g-RAT default value is 800 ticks) and none live past 1600 ticks (g-DAT default value is 1600 ticks). Somewhat arbitrarily, a generation is considered to be g-RAT ticks in length. All new-born agents have half the wealth of a mature agent, and a generation must be long enough to allow them to gain back the wealth lost in reproduction.
- TICK – A tick consists of the execution of eight steps in a predefined order. The eight steps of a tick are (1) do-pre-tick; (2) do-sell-waste-buy-recycled; (3) do-hire-wrkrs; (4) do-sell-inventory; (5) d-eat; (6) do-reproduce; (7) do-death; (8) do-post-tick.
- STEP – A step consists of the completion by all agents of a type of economic transaction (e.g. buy supplies) or a type of biophysical action (e.g. eat). Collectively, these are called x-actions. Within a step, each agent may or may not be given the opportunity to participate in an x-action, depending on circumstances.
- X-ACTION – An x-action may involve a single agent (biophysical action) or two agents (a commercial transaction). This is the atomic level at which the biophysical and economic sub-systems operate. An agent able to participate in a commercial opportunity is said to be economically enfranchised.

3.3.2.6.2 Relation to real-world timings and durations

The analogy to real-world time is not very relevant to the interpretation of the model, since there is no intention or need to validate the model by comparing it with real-world economies. The goal is to understand sustainable dynamics in a stand-alone logical economy first. Better simulation of real-world economies would possibly be a later refinement. Nevertheless, there are a couple of ways in which one can draw an analogy to real-world time.

First, if you consider one tick to represent a working day, then 20 ticks is a month (counting only working days) and 250 ticks is approximately a year.

Second, if you consider a generation in the real world to be about 20 years, then 800 ticks (the time required for a young agent to mature) represents about 20 years, and 40 ticks represents one year, and one tick represents about a week.

The second analogy seems to be more useful in interpretations, but as stated before, validation against real-world systems is not a current goal. If needed, when the parameter space of PMM #2 has been more fully explored, it may be possible to make these two interpretations each more fully commensurate with real-world metabolic consumption rates, and with each other.

3.3.2.7 Non-Participating entities

In the NetLogo implementation of ModEco, a few entities exist which are not part of the design of the economic engine or ecological system being modeled, but exist due to the goals of the programmer to provide real-time data display, data capture to file, efficiency of operation, or maintainability of code. By intent, these entities contain no state variables, and no embedded entities which affect the outcome of a run of any economy. They are therefore considered non-participating entities.

It was necessary to be sure that none of the code in the non-participating entities used the NetLogo commands that implicitly invoke the PRNG. This is important, if you want a specific random seed to produce the same results, regardless of the status or use of the non-participating entities. I think I have succeeded in this. Plots, debug routines, monitors, and data dumps do not cause a change in the trajectory of the economy through its phase space.

Nevertheless, some understanding of these non-participating entities is required for complete understanding of the pseudo-code provided in section 3.3.3. Furthermore, any attempt to replicate PMM #2 will require some attention to similar functions.

3.3.2.7.1 Real-time data display entities

ModEco offers a variety of real-time data displays of aggregated biophysical or economic data. There are several types of entities that address this type of need:

- Monitors are used to display the aggregates of biophysical materiel classes (cash, mass, energy) by breed. (See section 3.3.7.1.)
- Monitors are used to display the aggregates of economic asset classes (cash, energy, recycled, inventory, supplies, waste) by breed. (See section 3.3.7.1.)
- A variety of histograms and line graphs are used to display data for wealth distribution, age distribution, economic asset classes (both aggregate and averaged), and biophysical and economic indicators such as entropic indices, GDP, sector size, frmrs per wrkr ratio, employment rate, and resource flow rates. (See sections 3.3.7.4 through 3.3.7.7.) Care must be taken to ensure that the embedded code in plots and other user interface controls does not implicitly call the PRNG, or these controls do become ‘participating’ entities. They can alter the trajectory of a run through state space.

3.3.2.7.2 Data capture entities

ModEco also offers several data capture options:

- Each native NetLogo plot automatically collects a data base of all plot instructions so it can redraw itself. A right-click on any plot invokes a pop-up menu with an export option. The data can be exported to a .txt file which can be loaded into MS Excel as a CSV file.
- In addition, there are three toggles that control the collection of data into CSV files:
  - DPX – Data Per X-Action – Each biophysical action (eat, reproduce, die) or each commercial transaction (buy, hire, sell) can be logged. This is micro-data.
  - DPT – Data Per Tick – At the end of each tick, aggregate (macro) data is logged.
  - DPG – Data Per Generation – At the end of each g-RAT ticks, data for each mortal agent is logged. This is micro-data.
- Finally, a debugging facility was developed to enable the capture and/or real-time display of trace information for each step, or for all steps, and this trace data is captured in a debug log. There is an 'agent in focus' feature that enables a trace of all actions of a specific agent.

These data capture activities are designed in such a way as not to alter the trajectory of a run through its state space, whether on or off.

This range of options was developed as a means to debug the code and engineer a sustainable economy. They have been left active in the software. The pseudo-code described in section 3.1.3 below does not mention the data collection points and conditions. While is it realized that such information would be highly useful to those who would wish to replicate PMM #2, it greatly complicates an already complex set of pseudo-code modules, and so was intentionally left out.

3.3.3 Process and Scheduling Details with High-Level Pseudo-Code

On first loading the model, the following module is run automatically:

- ***startup*** – in which default values for certain sliders, choosers and switches are set. These native NetLogo controls have persistent values through the save/load cycle. This routine over-rides those persistent values, then the 'setup' routine is called. The only control not so over-ridden is the scenario chooser.

On startup, or when clicking on the 'Setup' button, the following routine is run:

- ***setup*** – in which the township and its inhabitants are constructed, and derived parameters are calculated. On startup, the default values for explicit parameters are used. When run from a button, the adjusted settings of all explicit parameters (switches, sliders and choosers) are used.

Once setup has been executed, the model can be run one tick at a time, using the 'One Tick' button(s), or it can be run continuously, using the 'Go' button(s). In either case, the **go** routine is called.

NOTE: **startup**, **setup**, and **go** are standard names for top-level routines according to NetLogo practice.

With each execution of the **go** routine the following schedule of steps is executed:

- **do-pre-tick** – in which termination or pause conditions are checked, the tick counter is advanced, derived parameters are re-calculated (in case a slider has been moved).
- **do-buy-recycled-sell-waste** – in which one of three optional approaches is used, depending on the settings in the gb-MMgr and gb-MMgr-debt switches:

  - If the gb-MMgr switch is on, the MMgr canvasses all participating consumers to purchase their waste, then canvasses all frmrs to sell them recycled waste. If the gb-MMgr-debt switch is on, the MMgr can go into debt when purchasing waste.
  - Otherwise, all frmrs canvass all participating consumer within their commuting areas with an offer to purchase waste directly, without intervention of the MMgr.
- ***do-hire-wrkrs*** – in which *frmr*s canvass participating *wrkr*s within their commuting area in a random order and hire them to harvest food from the *lot*.
- ***do-move-wrkrs*** – in which any *wrkr* that is recently unemployed and for which food is in short supply can move to any randomly-selected empty residential spot within its commuting area. (*Not implemented in PMM#2 using the NetLogo ADE***.**)
- ***do-sell-inventory*** – in which *frmr*s sell inventory to participating consumers, thereby re-labelling the goods to 'supplies'. A *frmr* may not consume its own inventory. However, it may sell it's inventory to itself, and so that inventory becomes its own supplies which it may now consume. (This characteristic is needed for compatibility with other economies not described here, in which price negotiation is a fundamental part of the model.)
- ***do-eat-supplies*** – in which *wrkr*s and *frmr*s consume an amount of supplies determined by an explicit parameter. Those which have insufficient supplies to meet the day's requirements are marked as starved, and later, removed.
- ***do-reproduce*** – in which *wrkr*s and *frmr*s undergo fission if they are mature enough and healthy enough. One daughter agent replaces the parent, geographically. The other must find an empty l*ot* or residence to occupy.
- ***do-death*** – in which *wrkr*s and *frmr*s die if they are too hungry or too old.
- ***do-post-tick*** – in which aggregate micro- and macro-economic and ecological data is compiled, visual displays are updated.

There are a number of explicit parameters which play a significant role in PMM #2. However, note that PMM #2 is sustainable for the default values, but unsustainable for many settings of these parameters.

The following sections describe each of these routines at a very high level only, compressing over a hundred pages of code into a few pages of pseudo-code. For details, one should refer to the code.

3.3.3.1 startup pseudo-code

The manual describes this routine as follows: "This procedure, if it exists, will be called when a model is first loaded in the NetLogo application. Startup does not run when a model is run headless from the command line, or by parallel BehaviorSpace." The pseudocode for this routine looks like this:

```
begin startup
  ;; This routine is to be executed by the observer.
  Ensure all data collection is turned off.
  Ensure that key persistent parameters (i.e. those associated with switches, choosers,
    and sliders, are set to default values. (See Table 21.)
  Run the setup routine to initialize other non-persistent global variables.
end startup
```

3.3.3.2 setup pseudo-code

This is the routine in which lists and display indicators are reset, if needed, in each tick. In pseudocode, this process looks like this:

```
begin setup
  ;; This routine is to be executed by the observer.
  Clear all ticks, turtles, patches, drawing, plots, and output, but not the globals.
  Individually initialize the globals that are not associated with controls.
   - Version number
```

```
      - Scenario, using the input from the chooser.
      - Derived parameters, calculated from values in explicit parameter controls.
     Initialize the PRNG, using the seed from the slider.
     Set the shapes for wrkrs and frmrs.
     Initialize the patches with 0 wrkrs, 0 frmrs, but correct comm-area.
     Build a Rectangular Village Pattern (RVP) of 4 frmrs and 16 wrkrs.
     Initialize the MMgr and EMgr appropriately for the scenario and sliders.
   end setup
```

There are two key sub-routines of setup:

To initialize the patches:

```
begin f-establish-patch-commuting-area
  For each patch (in random order)
    Build an agent set containing the 25 patches centred on this patch.
    Indicate the number of wrkrs and frmrs present as zero.
  end For
end f-establish-patch-commuting-area
```

To build the Rectangular Village Pattern (RVP)

```
begin f-establish-rvp
  ;; This routine is to be executed by the observer.
  Create 16 wrkrs and place in a square formation.
  Create 4 frmrs and place in 4 lots to side of wrkrs.
  Initialize the 4 frmrs.
  Initialize the 16 wrkrs.
end f-establish-rvp
```

3.3.3.3 do-pre-tick pseudo-code

Maintain the model framework parameters.

```
begin do-pre-tick
  ;; This routine is to be executed by the observer.
  Advance the tick counter by 1 tick.
  Ensure that the derived parameters are properly valued, based on current values
    of explicit parameters in switches, sliders and choosers.
end do-pre-tick
```

3.3.3.4 do-sell-waste-buy-recycled pseudo-code

There are two optional approaches, the first of which involves the MMgr, the second of which does not. If the MMgr is involved, a switch determines whether it may go into debt to purchase waste.

```
begin do-sell-waste-buy-recycled
  ;; This routine is to be executed by the observer.
  if the MMgr feature is turned on
    For each consumer, set the participation flag, end for
    For each participating consumer (in random order)
      f-sell-waste-to-mmgr
    end for
    For each frmr, set the participation flag, end for.
    For each participating frmr (in random order)
      f-buy-recycled-from-mmgr
    end for
  end if
  else
    f-sell-waste-buy-recycled-via-frmrs
  end else
end do-buy-recycled-sell-waste
```

There are three key sub-routines of do-buy-recycled-sell-waste:

To sell waste to the MMgr:

```
begin f-sell-waste-to-mmgr
  ;; A consumer executes this routine.
  if g-MMgr-cash > 0 and gb-MMgr = true
    f-consumer-requests-Sew-mass-grant
    if MMgr has sufficient cash to meet quota
      exchange waste for cash
    end if
    f-consumer-returns-unused-grants
  end if
end f-sell-waste-to-mmgr
```

To buy recycled mass from the MMgr:

```
begin f-buy-recycled-from-mmgr
  ;; Only frmrs execute this routine.
  if the MMgr has mass to sell
    f-set-frmr-hrl
    if recycled < hrl
      f-frmr-requests-ByR-cash-grant
      determine the amount and price
      exchange cash for recycled mass
      f-frmr-returns-unused-grants
    end if
  end if
end f-buy-recycled-from-mmgr
```

To have frmrs buy recycled mass directly from consumers requires two sub-routines – one that loops through participating frmrs, and one that loops through participating wrkrs for each frmr:

```
begin f-sell-waste-buy-recycled-via-frmrs
  ;; This routine is to be executed by the observer.
  for each frmr, set participation flag based on hrl and cash, end for
  for each consumer, set participation flag based on waste, end for
  for each participating frmr (in random order)
    f-buy-recycled-from-consumers
  end for
end f-sell-waste-buy-recycled-via-frmrs

begin f-buy-recycled-from-consumers
  ;; Only frmrs execute this routine.
  for each participating customer within frmr’s commuting area(in random order)
      continue until sufficient transactions are completed
      or until cash is depleted
      or until customers are depleted
    f-frmr-requests-ByR-cash-grant
    f-consumer-requests-Sew-mass-grant
    determine the amount and price
    exchange the waste for cash
    turn off the consumer’s participation flag, suppressing further sales
    f-consumer-returns-unused-grants
    f-frmr-returns-unused-grants
    increment the number of transactions
    determine if the frmr can continue
  end for
end f-buy-recycled-from-consumers
```

3.3.3.5 do-hire-wrkrs pseudo-code

```
begin do-hire-wrkrs
  ;; This routine is to be executed by the observer.
  f-set-frmr-hil
  f-set-consumer-hsl
  set frmrs' participation flags based on recycled, cash, energy, hil, hsl, set-point
  set wrkrs' participation flags based on energy, hsl, set-point
  for each participating frmr (in random order)
    f-frmr-hires-wrkrs
  end for
end do-hire-wrkrs
```

There is one key sub-routine:

```
begin f-frmr-hires-wrkrs
  ;; only frmrs execute this routine – self is a frmr
  for each participating wrkr within frmr's commuting area (in random order)
      continue until union list is depleted
      or until cash is depleted
      or until recycled is depleted
      or until energy is depleted
      or until inventory is over hil
    f-frmr-requests-Hwk-cash-grant
    f-frmr-requests-Hwk-mass-grant
    f-wrkr-requests-Hwk-energy-grant
    f-frmr-requests-Hwk-energy-grant
    determine the amount to harvest and the salary to be paid (tricky)
    do the work, pay the salary, harvest the mass/energy and convert to inventory
    f-frmr-returns-unused-grants
    f-wrkr-returns-unused-grants
    turn wrkr's participation flag off to suppress further hirings
    determine if frmr can continue
  end for
end f-frmr-hires-wrkrs
```

When a job offer is successfully negotiated according to the above routine, then the two parties must decide on the amount of work to be done, which determines the price to be paid. The frmr provides the recycled mass and half of the energy. The wrkr provides half of the energy. The frmr pays the wrkr for its energy expended on behalf of the frmr. The process to rationalize the amount to be done is complicated due to the need to match the amount of resources from four stocks (*frmr:cash*, *frmr:recycled*. *frmr:energy* and *wrkr:energy*. It is further complicated by the participation of the EMgr topping up none, some, or all of these stocks. The fundamental equation of harvesting is 2 recycled mus + 1 energy eu from frmr + 1 energy eu from wrkr => 2 inventory meus for frmr. The frmr pays the wrkr a salary of $1 for the 1 energy eu.

If both parties have more than sufficient resources to meet quota, there is no problem. However, if one or more of the resources are in insufficient amount, then the rest must be reduced accordingly. But first, if grants are available, the insufficiencies might be addressed by grants from the *EMgr*.

3.3.3.6 do-move-wrkrs pseudo-code

A wrkr who is unemployed and broke will move in search of better employment opportunities. This is not implemented in PMM #2 using the NetLogo ADE.

3.3.3.7 do-sell-inventory pseudo-code

Each participating frmr sells inventory to participating consumers within its commuting area.

```
begin do-sell-inventory
  ;; This routine is to be executed by the observer.
  set frmrs' participation flags based on inventory
  for each consumer
    f-set-consumer-hsl
    set consumer's participation flag based on cash, supplies, hsl
  end for
```

```
    for each participating frmr (in random order)
      f-sell-inventory-to-consumers
      turn off frmr’s participation flag
    end for
  end do-sell-inventory
```

A consumer is a w*rkr* or a f*rmr*. A frmr may sell inventory to itself, converting them to supplies. The high level pseudo-code for the f-sell-inventory-to-consumers routine follows:

```
begin f-sell-inventory-to-consumers
  ;; Frmrs execute this routine – self is a frmr.
  while frmr has inventory and customers
    for each participating customer within the frmr’s commuting area (in random order)
      f-frmr-requests-SlI-meu-grant
      f-consumer-requests-SlI-cash-grant
      determine the amount and price (note frmrs need 4 times the amount)
      exchange the cash and inventory/supplies
      f-frmr-returns-unused-grants
      f-consumer-returns-unused-grants
      turn consumer’s participation flag off, suppressing further purchases
      determine if frmr can continue
    end for
  end while
end f-sell-inventory-to-consumers
```

3.3.3.8 do-eat-supplies pseudo-code

The high level pseudo-code for the do-eat-supplies routine and its two sub-routines follows:

```
begin do-eat-supplies
  ;; This routine is to be executed by the observer.
  ;; [not needed] ask turtles [ set b-is-ready-to-die 0 ]
  for each frmr (in random order)
    f-frmr-eats-own-supplies
  end for
  for each wrkr (in random order)
    f-wrkr-eats-own-supplies
  end for
end do-eat-supplies

begin f-wrkr-eats-own-supplies
  ;; Wrkrs execute this routine – self is a wrkr.
  determine amount of supplies available to eat, g-EPT or less.
  consume those supplies, increasing energy and waste appropriately
  if amount eaten < g-EPT
    flag the wrkr as ready to die of hunger
  end if
  if age of wrkr < g-RAT
    adjust the energy-set-point upwards by one
    f-set-available-energy for working
  end if
end f-wrkr-eats-own-supplies

begin f-frmr-eats-own-supplies
  ;; Frmrs execute this routine – self is a frmr.
  determine amount of supplies available to eat, ( 4 * g-EPT ) or less.
  consume those supplies, increasing energy and waste appropriately
  if amount eaten < ( 4 * g-EPT )
    flag the frmr as ready to die of hunger
  end if
  if age of frmr < g-RAT
    adjust the energy-set-point upwards by one
    f-set-available-energy for working
  end if
end f-frmr-eats-own-supplies
```

3.3.3.9 do-reproduction pseudo-code

The high level pseudo-code for the do-reproduction routine and its two key sub-routines follows:

```
begin do-reproduction
  ;; This routine is to be executed by the observer.
  set the participation flags for all turtles, based on age and energy
  for each participating frmr (in random order)
    f-reproduce-frmr
  end for
  for each participating wrkr (in random order)
    f-reproduce-wrkr
  end for
end do-reproduction

begin f-reproduce-frmr
  ;; Frmrs execute this routine – self is a frmr.
  ;; Reproduction is by fission, producing daughters D1 and D2.
  if there are any empty lots within commuting area of parent.
    select a lot for daughter D2 (D1 takes parent’s place)
    create D2 there and attach it to the lot
    initialize each of D1 and D2,
      giving half of parent’s assets to each.
      fr-get-random-age-to-reproduce
      zero the energy-set-point
      f-set-available-energy
      zero all participation flags
    end initialize
    take parent off all contact lists within its commuting area
    for each of D1 and D2
      add it to contact lists of all agents within its commuting area
      build each its own contact lists of agents within its commuting area
    end for
  end if
end f-reproduce-frmr

begin f-reproduce-wrkr
  ;; Wrkrs execute this routine – self is a wrkr.
  ;; Reproduction is by fission, producing daughters D1 and D2.
  if there are an empty residences (rents) within commuting area of parent.
    select a residence for daughter D2 (D1 takes parent’s place)
    create D2 there and attach it to the lot/residence
    initialize each of D1 and D2,
      giving half of parent’s assets to each
      fr-get-random-age-to-reproduce
      zero the energy-set-point
      f-set-available-energy
      zero all participation flags
    end initialize
    take parent off all contact lists within its commuting area
    for each of D1 and D2
      add it to contact lists of all agents within its commuting area
      build each its own contact lists of agents within its commuting area
    end for
  end if
end f-reproduce-wrkr
```

3.3.3.10 do-death pseudo-code

The high level pseudo-code for the do-death routine is a little different, and it follows:

```
begin do-death
  ;; This routine is to be executed by the observer.
  ;; Check to see which agents are ready-to-die.
  set participation flags for turtles based on old age (Age > g-DAT),
      hunger (supplies < g-EPT) or emaciation (energy < g-DET)
  for each participating turtle (in random order)
    transfer assets to the EMgr
    remove it from the contact lists of the agents within its commuting area
    release the lot or residence in which it is found
    kill the logical shell of the agent
  end for
end do-death
```

3.3.3.11 do-post-tick pseudo-code

The high level pseudo-code for the do-post-tick routine:

```
to do-post-tick
  ;; This routine is to be executed by the observer.
  for each turtle, increase age by one, end for
end
```

There are a number of actions relating to debug routines, real-time data display, real-time data capture, plotting, etc. that are not directly relevant to the model. These have been elided here.

3.3.4 Initialization

ModEco (NetLogo ADE) can be initialized in two scenarios called the PMM and the GROWTH model. Both are able to run endlessly. The settings of parameters which are unique to each scenario are given in Table 17. All other parameters and variables have the default values indicated elsewhere. Variable names and values are in corresponding order.

Table 17 – Initial Endowments of Cash, Mass and Energy – MMgr and EMgr

Initial Endowments		
g-EMgr-cash-at-setup	$0 $1,000,000	PMM #2 scenario – Initial endowment. GROWTH scenario – Initial endowment.
g-EMgr-mass-at-setup	0 mus 1,000,000 mus	PMM #2 scenario – Initial endowment. GROWTH scenario – Initial endowment.
g-EMgr-energy-at-setup	0 eus 1,000,000 eus	PMM #2 scenario – Initial endowment. GROWTH scenario – Initial endowment.
g-MMgr-cash-at-setup	$100,000	Both scenarios – Initial endowment.
g-MMgr-mass-at-setup	0 mus	Both scenarios – Initial endowment.

The initialization of each *wrkr* and each *frmr* for each type of scenario is given below.

Both scenarios start in a cluster of 4 frmrs and 16 wrkrs.

Figure 08 – The Rectangular Village Pattern (RVP)

```
|--------|---------|---------|---------|
| 1 frmr | 4 wrkrs | 4 wrkrs | 1 frmr  |
| 1 frmr | 4 wrkrs | 4 wrkrs | 1 frmr  |
|--------|---------|---------|---------|
```

The arrangement of *wrkr*s and *frmr*s forms a pattern called the "Rectangular Village Pattern" (or RVP), in which four residential *Lot*s form a square housing 16 *wrkr*s, and there are two farmed *Lot*s each to the left and right, for a total of four farmed *Lot*s. Altogether, the RVP occupies a rectangular area two *Lot*s high by four *Lot*s wide which appears as a cluster of houses in a village with four outlying farms. This pattern is used in both the PMM #2 and GROWTH scenarios.

In the RVP, every *wrkr* has four farms within its commuting area where is may work or purchase supplies, and every farm has two frmrs (including itself) from which it can purchase supplies, and 16 wrkrs within its commuting area whom it may hire.

In Table 18 the details of the calculation of state variables for wrkrs and frmrs on initialization are given. The first and second daughters are indicated by D1 and D2 respectively. Note that stock sizes are scaled by the consumption rates. The variables *wrkr*:*age* and *frmr*:*age* are assigned stochastically. They are between zero and the age of reproduction (g-RAT) with a uniform probability distribution. This avoids starting in synchronization. The age-to-reproduce is also assigned randomly in a uniform probability distribution. It is centred on g-RAT having a width of g-GTT. Age is only randomized on initialization of the economy. Age-to-reproduce is randomized on initialization, and also on the birth of new agents, when age is set to zero.

Table 18 – Formulae for Initialization of Agent's state variables

Variable	wrkr	frmr
breed	same as parent's	same as parent's
(xcor, ycor)	D1 inherits from parent D2 gets a lot in parent's lot:com-area	D1 inherits from parent D2 gets a lot in parent's lot:com-area
age	randomized between 0 and g-RAT	randomized between 0 and g-RAT
age-to-reproduce	Randomized between (g-RAT – g-GTT/2) and (g-RAT + g-GTT/2)	Randomized between (g-RAT – g-GTT/2) and (g-RAT + g-GTT/2)
cash	$1,000	$1,000
energy	(1,000 * g-EPT) eus	(1,000 * g-EPT) eus
recycled	N/A	(500 * g-EPT) mus
inventory	N/A	(100 * g-EPT) meus
supplies	(100 * g-EPT) meus	(200 * g-EPT) meus
waste	(0 * g-EPT) mus	(0 * g-EPT) mus

3.3.5 Input Data

In the ODD protocol, this section is assigned for the description of all state variables and parameters that are read automatically from input files. Such a feature could be used to enable multiple runs of various economies without user intervention. This type of feature has not been implemented in ModEco, and is not available for PMM #2. In its place, I have converted many soft-coded parameters into user-interface controls. These controls can be managed by the 'Behaviour Space' facility in NetLogo, allowing for a number of runs with selected parameter settings.

3.3.6 Sub-Models

3.3.6.1 Biophysical Sub-System Parameters

A ModEco-based economy is an amalgamation of two complex adaptive systems, welded together to form one co-adaptive system. The business factors and quota described above can be used to shape and control the economic sub-system. The metabolic parameters are used to control the biophysical sub-system. These biophysical parameters are not original to ModEco, but were derived from the work of Dr. Michael Palmiter (Dewdney, 1989) and used the application PSoup demonstrating evolution (Boyle, 2014 [06]).

In Table 19 we have descriptions of those explicit biophysical parameters that control the life functions of the mortal agents as they live and die in the biophysical sub-system. All of these global variables are assigned a fixed value during setup, based on sliders, and that value does not change during a run. These are not state variables.

Table 19 – Explicit Parameters of the Biophysical Sub-System

Name	**Default Value**	**Brief Description**
g-DAT	*1600 ticks*	Death Age Threshold – Agents that exceed this age, in ticks, die of old age.
g-DET	*4 eus*	Death Energy Threshold – Agents whose energy drops below this level die of emaciation.
g-RAT	*800 ticks*	Reproductive Age Threshold – Used to compute the earliest possible age for reproduction of an agent.
g-GTT	*200 ticks*	Gestation Time Threshold – Used to compute the earliest age possible for reproduction of an agent.
g-RET	*1000 eus*	Reproductive energy threshold – Agents that are of age but hold less energy than this cannot reproduce.
g-EPT	*4 meus* 16 meus	Energy Per Tick – wrkrs must consume this amount of food per tick, or they die of starvation. frmrs must consume a multiple of this to match the number of potential hires. (16 meus = 4 hires * 4 meus per hire)

The biophysical economics of the RVP per tick can be expressed in maximum possible units, where E is g-EPT (=4), H is the parameter g-no-of-hires-max (=4, see Table 20.), and F is the number of farms with associated residential lots in the RVP (=4, hard-coded).

Table 20 – Maximum Possible Biophysical Flows of Resources

Type of Flow of Resources	**Quantity of Resources – Per Farm**		**Per RVP**
	H=4 wrkrs	**1 frmr**	**total * F**
waste mass sold to MMgr	=H*E=16 mus	=H*E=16 mus	=128 mus
recycled mass purchased from MMgr		=128 mus	=128 mus
energy spent harvesting	=10*H*E/2=80 eus	=10*H*E/2=80 eus	=640 eus
recycled mass embedded into inventory	=80 mus	=80 mus	=640 mus
inventory harvested	=80 meus	=80 meus	=640 meus
inventory sold as supplies	=10*H*E=160 meus	=10*H*E=160 meus	=320 meus
waste produced in consuming supplies	=H*E=16 mus	=H*E=16 mus	=128 mus
energy produced in consuming supplies	=H*E=16 eus	=H*E=16 eus	=128 eus

The factor ‘10’ is one of the few hard-coded parameters in the model. It can be changed in the code where it occurs. Roughly speaking, a wrkr must be hired once every five ticks, on average, to survive. (10*E/2=20 eus spent per job is generated at a rate of E=4 eus per tick. 20 eus per hire / 4 eus per tick = 5 ticks per hire.)

3.3.6.2 Economic Sub-System Parameters

In Table 21 we have a description of the explicit parameters that control the economic sub-system. Each of these variables are assigned a fixed value, based on sliders, during setup, and that value does not change during a run. These are not state variables.

Table 21 – Explicit Parameters of the Economic Sub-system

Name	Default Value	Brief Description
Type of parameter: Iteration limits of Various Kinds		
g-no-of-rents-max	4 wrkrs	This is a slider parameter with a default value of 4. It constrains the maximum number of wrkrs that can live together on a single residential lot, and influences other aspects of frmr/wrkr relationships. A frmr and four wrkrs can live on two lots, and, by design, this is intended to be a sustainable economic unit. This parameters is also listed in Table 06 in connection with lots.
g-no-of-hires-max	4 wrkrs	The maximum number of wrkrs that a frmr can hire to do work per tick. Note that a wrkr can be hired a maximum of once per tick. This is hard-coded for wrkrs and soft-coded for frmrs.
g-no-of-waste-xactions-max	5 trans-actions	The maximum number of consumers from which a frmr can purchase waste when the MMgr is not involved. To date, there are no versions of the PMM #2 that do not involve the MMgr. This parameter is not used in the PMM #2 but exists in the ModEco application.
Type of parameter: Prices Determining Both Monetary and Intrinsic Values		
g-mu-price	$1	The price of one mu of mass (waste or recycled).
g-eu-price	$1	The price of one eu of energy.
g-meu-price	$2	The price of one meu of inventory or food (inventory or supplies). g-meu-price = g-mu-price + g-eu-price since 1 meu = 1 mu + 1 eu.
Type of parameter: Transaction Quotas – Preventing Monopolization		
g-waste-sales-quota	40 mus	The maximum amount of waste mass, measured in mus, that a consumer can sell to the MMgr (or to a frmr) per tick.
g-recycled-purchase-quota	200 mus	The maximum amount of recycled mass, measured in mus, that a frmr can purchase from the MMgr per tick. The quota on purchased by frmrs from consumers is, de facto, the quota on sales by consumers to frmrs, i.e. g-waste-sales-quota.
g-harvest-quota	40 meus	The maximum amount of food, measured in meus, that a wrkr (a hire) can harvest from the field per tick. A frmr can harvest a multiple of this amount, based on g-no-of-hires-max.
g-supplies-purchase-quota	40 meus	The maximum amount of food, measured in meus, that a consumer can purchase from a frmr per tick.
Type of parameter: Business Factors – Enabling Diversification of Asset Holdings		
g-HRF-f	0.50	frmrs – Hold Recycled Factor – Target value of recycled mass is 50% of frmr:net-value.
g-HIF-f	0.25	frmrs – Hold Inventory Factor – Target value of inventory is 25% of frmr:net-value.
g-HSF-f	0.25	frmrs – Hold Supplies Factor – Target value of supplies is 25% of frmr:net-value.
g-HSF-w	0.35	wrkrs – Hold Supplies Factor – Target value of supplies is 35% of wrkr:net-value.
Type of parameter: Asset Floors – Used in Combination with Business Factors		
g-HRL-f-min	500 mus	frmrs – Hold Recycled Limit Minimum – Minimum target.
g-HIL-f-min	300 meus	frmrs – Hold Inventory Limit Minimum – Minimum target.
g-HSL-f-min	200 meus	frmrs – Hold Supplies Limit Minimum – Minimum target.
g-HSL-w-min	100 meus	wrkrs – Hold Supplies Limit Minimum – Minimum target.

The ‘interation limits’ are general parameters closely associated with the concept of a village of people consisting of one frmr and four wrkrs living on two lots, which is the building block of the Rectangular Village Principle (RVP).

The pricing scheme arbitrarily considers mass and energy to be of equal value, as both are needed to make food.

The transaction quotas prevent monopolization of an asset by a single wealthy agent that could buy up all available quantities of an asset, causing the others to starve, and the economy to collapse.

The business factors and asset floors work together to encourage diversification of assets within the holdings of a single agent. For example, a business factor of g-HRF-f = 0.25 indicates that a frmr will not purchase recycled mass if the present holding is over 25% of its current net value. This g-HRF-f is used to calculate the 'hold recycled limit' (frmr:HRL) for the tick. The 'asset floor' g-HRL-f-min is a lower limit on the setting for frmr:HRL. A poor frmr will purchase whenever possible, as it must stay involved in the market to survive.

3.3.6.3 Municipal Grant Sub-System

The need for the role of the EMgr (Estate Manager) was described in sections 3.2.3.1 and 3.3.2.4.2 above. However, the recycling of the assets of dead agents could be done in many different ways. These assets are from agents without offspring, having no heirs, and the municipality is the recipient of these assets. The assets are transferred without a commercial quid-pro-quo, as gifts. They must be gifted back into the commercial flows of goods, services, and cash. Exactly how you do it is a kind of economic policy decision. They can be viewed as business development grants, or welfare grants. I call them municipal grants. I have decided to gift these resources to agents who are in some sense poor but deserving. They target the lower middle class. This makes them a combination of business development and welfare.

An agent is deemed 'deserving' if it can and has entered into an agreement to sell or purchase a good or a service. Those who, for some reason, are not engaged in a successful commercial transaction cannot apply for a grant.

An agent is deemed 'poor' if it does not have sufficient stocks of a needed resource, however, to complete the transaction at the full amount of the transaction quota for that type of transaction. It cannot benefit fully from the commercial transaction.

Every agent involved in a commercial transaction can apply for a municipal grant. The EMgr assesses its state, and if it is both poor and deserving, assigns it a grant to meet quota, but only from available stocks. If for some reason one of the two agents still cannot meet quota, there is the possibility that some of the grant may go unused. The unused grant is returned to the EMgr.

There is a set of variables that are used to characterize the grant status of wrkrs and frmrs on a step-by-step basis within a tick. These variables are used to track the grants received from the EMgr and used by the agent. Any surplus grant not used by the agent in the transaction for which it was granted is calculated and returned to the EMgr at the end of each transaction. So, again, these variables are of fleeting interest, used to make the code more simple and efficient. They are not state variables. They are derived variables. The variables come in pairs of the form grt-mgmt-<resource>-granted and grt-mgmt-<resource>-used, where <resource> is a substitute for type of resource. For example, if the resource is cash, the variables are grt-mgmt-cash-granted and grt-mgmt-cash-used. The amount returned to the EMgr is any positive result of the difference 'granted' minus 'used'. (See Table 22.)

Table 22 – Derived Dynamic 'Grant Management' Variables of Mortal Agents

Resource (type of grant)	Type of agent	Market action	Brief Description of When Grant is Given.
cash	frmr frmr consumer frmr	buy recycled hire wrkr buy supplies buy waste	A frmr needs cash to buy recycled mass from the MMgr. A frmr needs cash to hire a wrkr. A consumer needs cash to buy supplies. A frmr needs cash to buy waste from a consumer (MMgr off).
energy	frmr wrkr	hire wrkr work	A frmr needs energy to work with a hired wrkr. A wrkr needs energy to work.
recycled	frmr	hire wrkr	A frmr needs recycled mass to hire a wrkr.
inventory	frmr	sell inventory	A frmr needs inventory to sell to consumers
waste	consumer consumer	buy recycled sell waste	A consumer needs waste to sell to the MMgr (or a frmr, if the MMgr function is turned off).

In all there are ten types of municipal grant. PMM #2 requires that these grant mechanisms be active in order to complete the cash and resource cycles. Perhaps not all are necessary. That question has not been explored.

3.3.6.4 Debug tools

Establishing sustainability was difficult, and a number of non-standard debug tools were built around the model in the search for sustainability. These have been organized in a panel in the user interface and left in place, should others wish to use them. Primarily, the debug tools allow a user to trace the detailed action as it happens. Two kinds of traces are possible: (a) full trace of all X-actions; and (b) trace of a single 'agent in focus'. The debug data will be sent to a debug log file, but may also be sent to the NetLogo 'command center'.

3.3.6.5 Real-time Data Display

There are a variety of plots in the user interface. While this does not affect the outcome of the model, it does enable visualization of activities in the model, and it enables detailed analysis of micro- and macro-economic and metabolic data generated in the course of a run. (See section 3.3.7 for more details.)

3.3.6.6 Data Collection for Analysis Post-Run

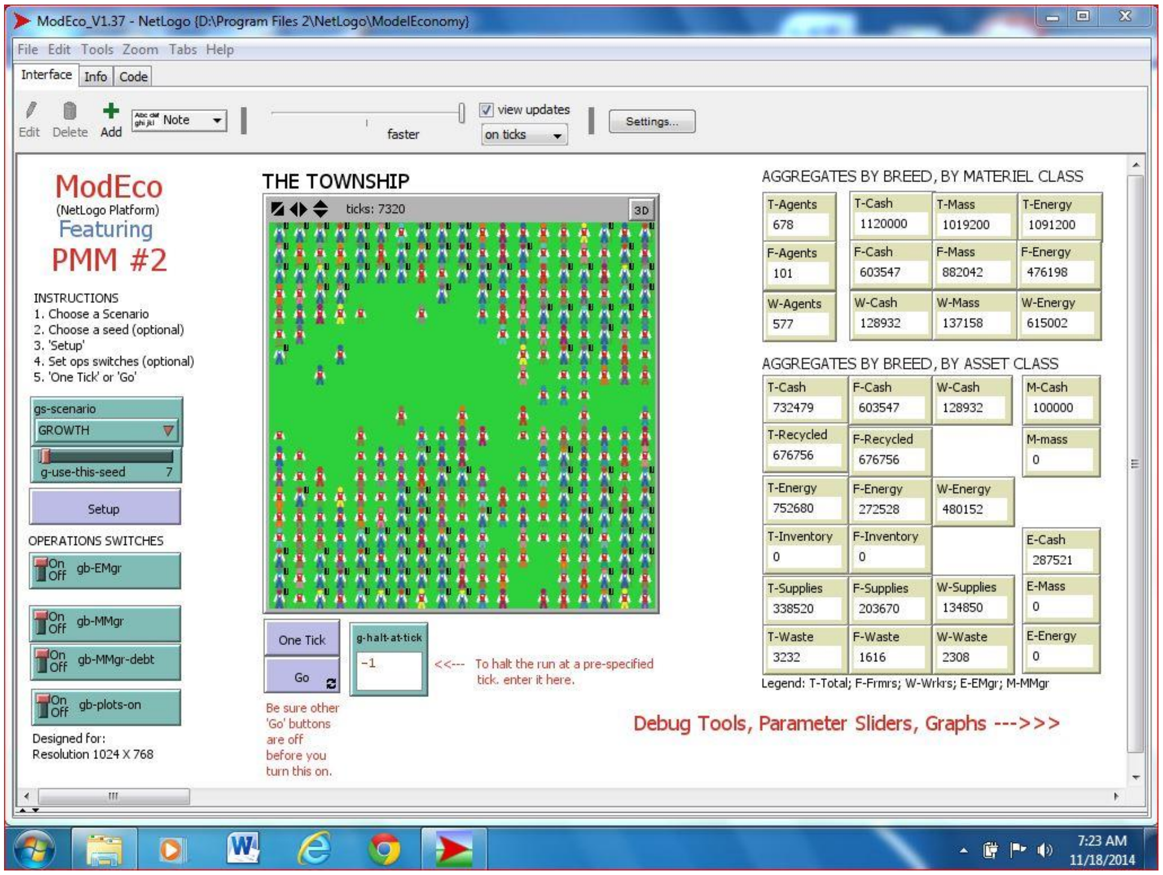
Optionally, data can be collected in "comma-separated value" (CSV) files which can be imported into MS Excel. Up to 1,000,000 records can be collected for each type of data. This is within the limit that Excel 2010 can handle. Data can be collected for analysis after a run has been completed in two distinctly different ways:

- First, NetLogo plots each contain a built-in data-base of instruction on how to redraw the plot. If the instructions are complicated, the data in the data-base is also complicated. If the plot instructions are simple, the data in the data-base looks like simple comma-delimited data. If you right-click on the face of the plot, a pop-up menu appears with a selection called 'Export'. Export the data to a .txt file, then use MS Excel to load it.
- Second, I have built in a somewhat slightly more formal set of tools that accomplish much the same thing.
    - DPX – (Data per X-action) – writes data to file for every biophysical action, or for every commercial transaction.
    - DPT – (Data per Tick) – writes data to file for every tick; and
    - DPG – (Data per Generation) – writes data to file for every generation of agents.

3.3.7 User Interface Example

The user interface has been organized into seven informal panels: main panel; debug panel; parameter panel; histogram panel; assets panel; indices panel; and sectors panel.

Figure 09 – The Main Panel of the User Interface



INSTRUCTIONS
1. Choose a Scenario
2. Choose a seed (optional)
3. 'Setup'
4. Set ops switches (optional)
5. 'One Tick' or 'Go'

gs-scenario: GROWTH
g-use-this-seed: 7

OPERATIONS SWITCHES
On Off gb-EMgr
On Off gb-MMgr
On Off gb-MMgr-debt
On Off gb-plots-on

Designed for:
Resolution 1024 X 768

g-halt-at-tick: -1

<<--- To halt the run at a pre-specified tick, enter it here.

Be sure other 'Go' buttons are off before you turn this on.

AGGREGATES BY BREED, BY MATERIEL CLASS

T-Agents 678	T-Cash 1120000	T-Mass 1019200	T-Energy 1091200
F-Agents 101	F-Cash 603547	F-Mass 882042	F-Energy 476198
W-Agents 577	W-Cash 128932	W-Mass 137158	W-Energy 615002

AGGREGATES BY BREED, BY ASSET CLASS

T-Cash 732479	F-Cash 603547	W-Cash 128932	M-Cash 100000
T-Recycled 676756	F-Recycled 676756		M-mass 0
T-Energy 752680	F-Energy 272528	W-Energy 480152	
T-Inventory 0	F-Inventory 0		E-Cash 287521
T-Supplies 338520	F-Supplies 203670	W-Supplies 134850	E-Mass 0
T-Waste 3232	F-Waste 1616	W-Waste 2308	E-Energy 0

Legend: T-Total; F-Frmrs; W-Wrkrs; E-EMgr; M-MMgr

Debug Tools, Parameter Sliders, Graphs --->>>

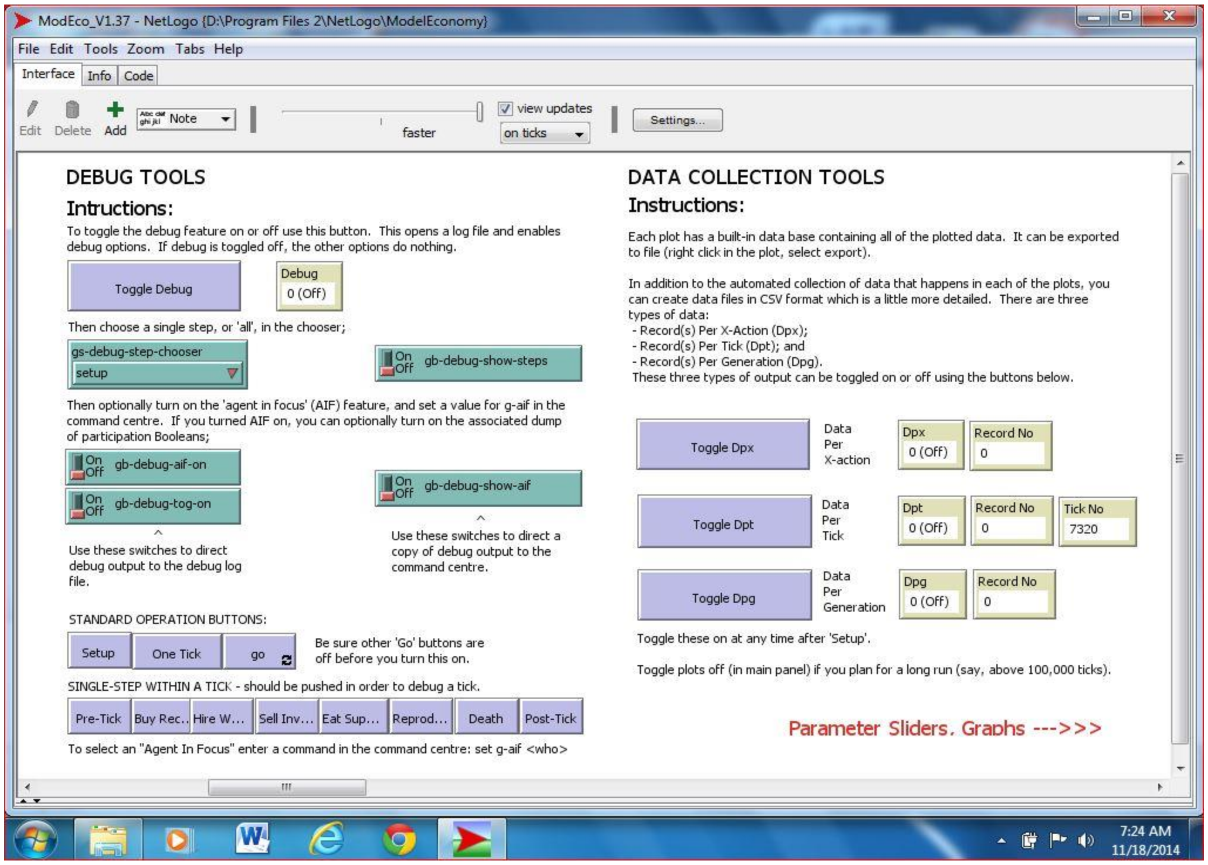
Figure 09 shows the main panel. It includes instructions to execute a simple run, for users who are not familiar with the model. The operational switches do the following:

- gb-EMgr turns the actions of the EMgr off or on.
- gb-MMgr turns the actions of the MMgr off or on.
- gb-MMgr turns the ability of the MMgr to go into debt off or on. This is an interesting switch.
- gb-plots-on disables or enables all plots.

To execute a fast long run, set the slider at the top to 'faster', untick the 'view updates' tick box, and turn of plotting. The economy can be advanced one tick at a time using the 'One Tick' button, or continuously using the 'Go' button. There are, in fact, many such 'Go' buttons in the user interface. To stop a run you must un-click the same Go button that is clicked, and darkened. There is a 'Halt at' data input box. If you want a run to stop at, say, tick 100, enter 100 into this box before clicking on the Go button. To the right there are a number of monitor boxes that display two kinds of aggregate data. Note that the totals of the conserved quantities (cash, mass and energy) will not vary. You can monitor that here. If they vary, there is a bug in the biophysical/economic engine.

Figure 10 shows the panel of the user interface that clusters the debug controls and real-time data collection controls.

Figure 10 – The Debug and Data Collection Panel



ModEco_V1.37 - NetLogo {D:\Program Files 2\NetLogo\ModelEconomy}

File Edit Tools Zoom Tabs Help

Interface Info Code

Edit Delete Add Note faster view updates on ticks Settings...

DEBUG TOOLS

Intructions:

To toggle the debug feature on or off use this button. This opens a log file and enables debug options. If debug is toggled off, the other options do nothing.

Toggle Debug

Debug 0 (Off)

Then choose a single step, or 'all', in the chooser;

gs-debug-step-chooser setup

On Off gb-debug-show-steps

Then optionally turn on the 'agent in focus' (AIF) feature, and set a value for g-aif in the command centre. If you turned AIF on, you can optionally turn on the associated dump of participation Booleans;

On Off gb-debug-aif-on

On Off gb-debug-tog-on

On Off gb-debug-show-aif

Use these switches to direct debug output to the debug log file.

Use these switches to direct a copy of debug output to the command centre.

STANDARD OPERATION BUTTONS:

Setup One Tick go

Be sure other 'Go' buttons are off before you turn this on.

SINGLE-STEP WITHIN A TICK - should be pushed in order to debug a tick.

Pre-Tick Buy Rec... Hire W... Sell Inv... Eat Sup... Reprod... Death Post-Tick

To select an "Agent In Focus" enter a command in the command centre: set g-aif <who>

DATA COLLECTION TOOLS

Instructions:

Each plot has a built-in data base containing all of the plotted data. It can be exported to file (right click in the plot, select export).

In addition to the automated collection of data that happens in each of the plots, you can create data files in CSV format which is a little more detailed. There are three types of data:
- Record(s) Per X-Action (Dpx);
- Record(s) Per Tick (Dpt); and
- Record(s) Per Generation (Dpg).
These three types of output can be toggled on or off using the buttons below.

Toggle Dpx — Data Per X-action — Dpx 0 (Off) — Record No 0

Toggle Dpt — Data Per Tick — Dpt 0 (Off) — Record No 0 — Tick No 7320

Toggle Dpg — Data Per Generation — Dpg 0 (Off) — Record No 0

Toggle these on at any time after 'Setup'.

Toggle plots off (in main panel) if you plan for a long run (say, above 100,000 ticks).

Parameter Sliders, Graphs --->>>

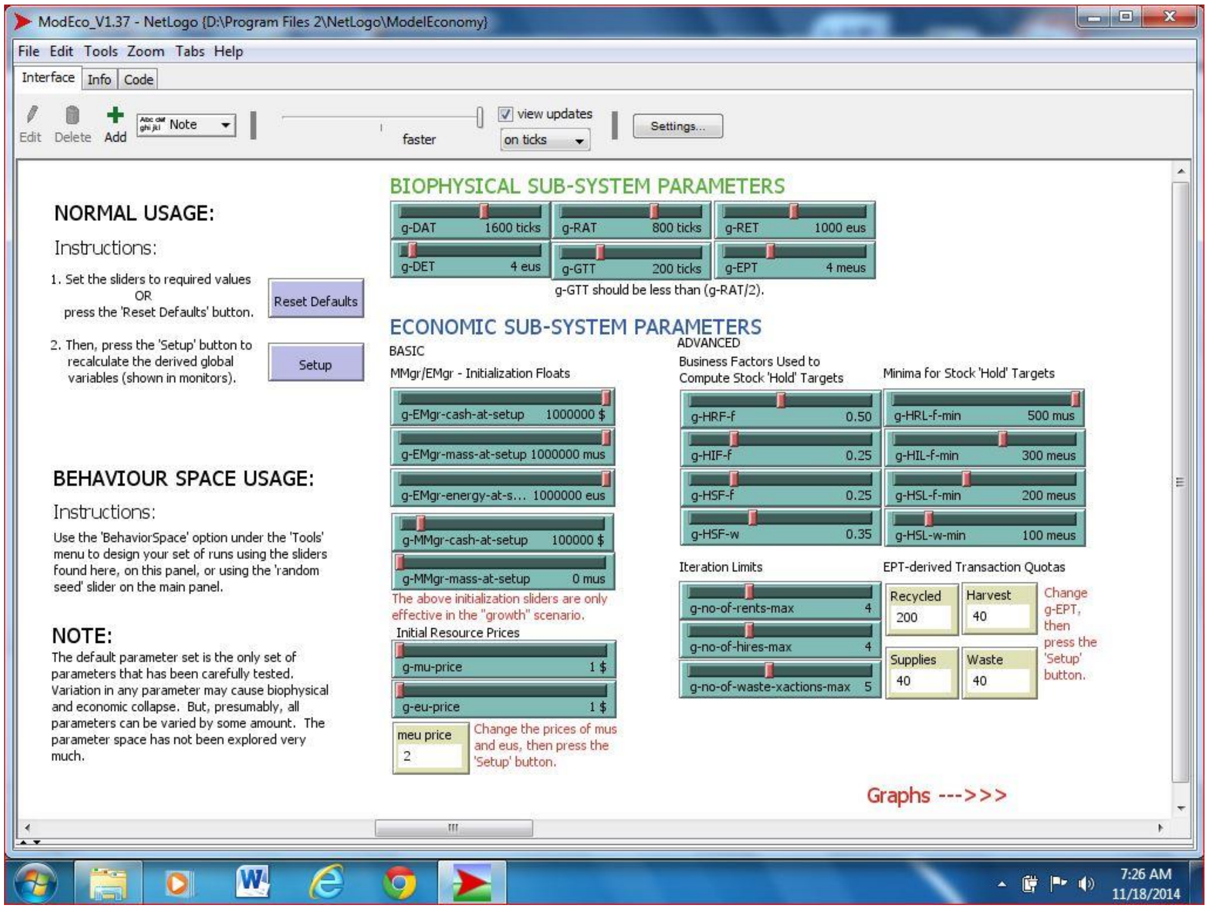
7:24 AM 11/18/2014

DEBUG TOOLS: Use the ‘Toggle Debug’ button to toggle the debug feature on or off, opening a log file. Use the chooser to select ‘Setup’, or one of the eight steps, or ‘all’. This determines which sections of code will be traced to the log file. (See section 3.3.6.4.) Use the ‘aif’ and ‘tog’ switches to dump ‘agent-in-focus’ and the associated ‘participation flags’ or toggles trace data. The two toggle to the right of those direct a copy of the debug trace data to the command center. There is a copy of the setup, one step and go buttons from the main panel here, for convenience. Then there are ‘step’ buttons so you can execute a single step and see what is happening in that step, useful when the data is streaming to the command center.

DATA COLLECTION TOOLS: Each toggle will open a data file and collect CSV data therein. The number of records written to each file is indicated. If a file is full, simply close it and open another. Each is time stamped. (See section 3.3.6.6.)

Figure 11 shows the panel of the user interface where the parameters that can be used with the behaviour space facility are clustered.

Figure 11 – The Parameters and Behaviour Space Panel



NORMAL USAGE:

Instructions:

1. Set the sliders to required values OR press the 'Reset Defaults' button.
2. Then, press the 'Setup' button to recalculate the derived global variables (shown in monitors).

BEHAVIOUR SPACE USAGE:

Instructions:

Use the 'BehaviorSpace' option under the 'Tools' menu to design your set of runs using the sliders found here, on this panel, or using the 'random seed' slider on the main panel.

NOTE:

The default parameter set is the only set of parameters that has been carefully tested. Variation in any parameter may cause biophysical and economic collapse. But, presumably, all parameters can be varied by some amount. The parameter space has not been explored very much.

BIOPHYSICAL SUB-SYSTEM PARAMETERS

g-DAT 1600 ticks | g-RAT 800 ticks | g-RET 1000 eus

g-DET 4 eus | g-GTT 200 ticks | g-EPT 4 meus

g-GTT should be less than (g-RAT/2).

ECONOMIC SUB-SYSTEM PARAMETERS

BASIC

MMgr/EMgr - Initialization Floats

g-EMgr-cash-at-setup 1000000 $

g-EMgr-mass-at-setup 1000000 mus

g-EMgr-energy-at-s... 1000000 eus

g-MMgr-cash-at-setup 100000 $

g-MMgr-mass-at-setup 0 mus

The above initialization sliders are only effective in the "growth" scenario.

Initial Resource Prices

g-mu-price 1 $

g-eu-price 1 $

meu price 2

Change the prices of mus and eus, then press the 'Setup' button.

ADVANCED

Business Factors Used to Compute Stock 'Hold' Targets

g-HRF-f 0.50

g-HIF-f 0.25

g-HSF-f 0.25

g-HSF-w 0.35

Minima for Stock 'Hold' Targets

g-HRL-f-min 500 mus

g-HIL-f-min 300 meus

g-HSL-f-min 200 meus

g-HSL-w-min 100 meus

Iteration Limits

g-no-of-rents-max 4

g-no-of-hires-max 4

g-no-of-waste-xactions-max 5

EPT-derived Transaction Quotas

Recycled 200 | Harvest 40

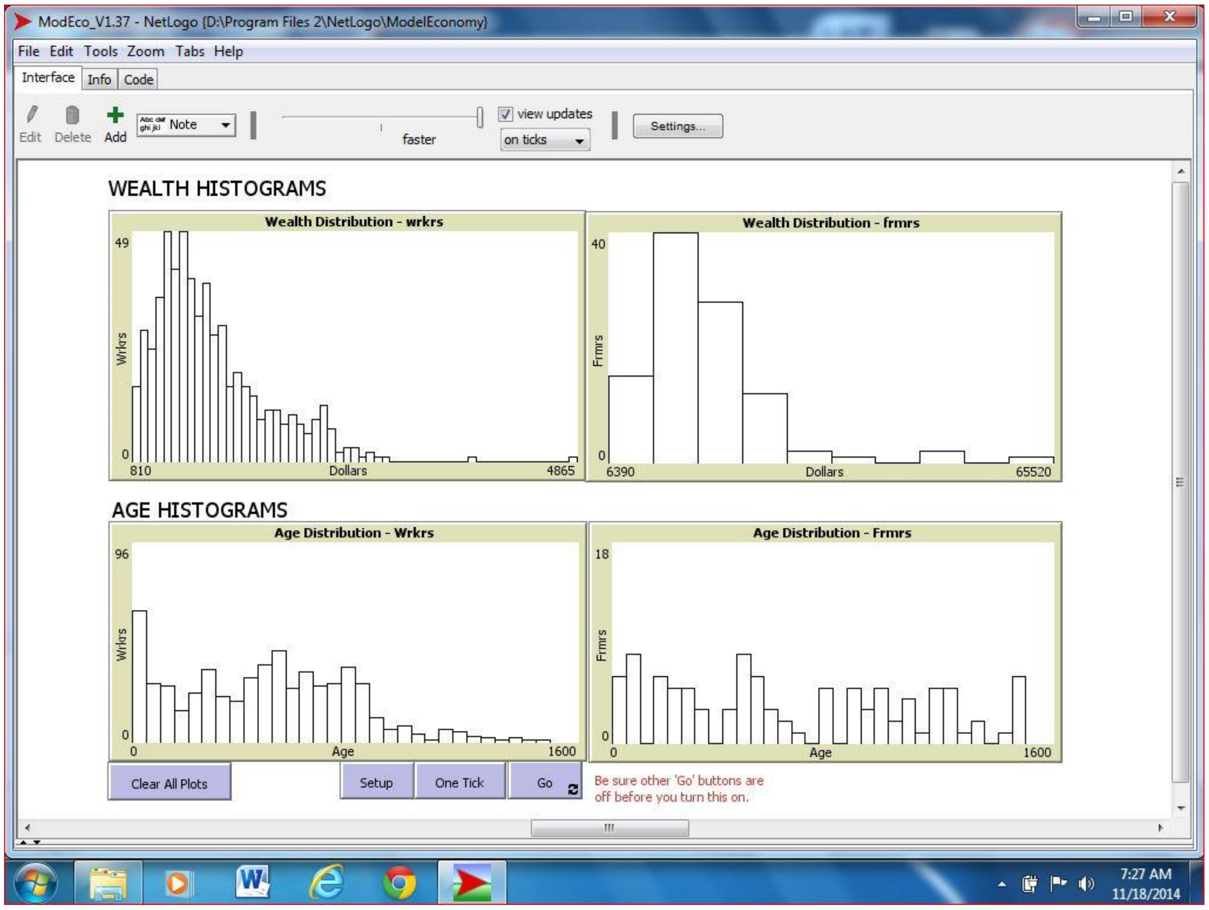
Supplies 40 | Waste 40

Change g-EPT, then press the 'Setup' button.

PARAMETERS AND BEHAVIOUR SPACE TOOLS: The parameter space has not been explored, and it is uncertain what setting are, in fact part of the PMM, and sustainable. Use the ‘Reset Defaults’ to recover the known PMM settings. When parameter switches are changed, some derived parameters need to be recalculated. Use the ‘Setup’ button to do that. The parameters of the biophysical sub-system have been described in section 3.3.6.1. The parameters of the economic sub-system have been described in section 3.3.6.2. Read the instruction for behaviour space facility, and design your own runs there.

Figure 12 shows the panel of the user interface where the real-time plots of wealth and age distributions are clustered.

Figure 12 – The Wealth And Age Histogram Panel


ModEco_V1.37 - NetLogo {D:\Program Files 2\NetLogo\ModelEconomy}
File Edit Tools Zoom Tabs Help
Interface Info Code
Edit Delete Add Note
faster
view updates
on ticks
Settings...
WEALTH HISTOGRAMS
Wealth Distribution - wrkrs
49 Wrkrs 0 810 Dollars 4865
Wealth Distribution - frmrs
40 Frmrs 0 6390 Dollars 65520
AGE HISTOGRAMS
Age Distribution - Wrkrs
96 Wrkrs 0 0 Age 1600
Age Distribution - Frmrs
18 Frmrs 0 0 Age 1600
Clear All Plots
Setup
One Tick
Go
Be sure other 'Go' buttons are off before you turn this on.
7:27 AM 11/18/2014

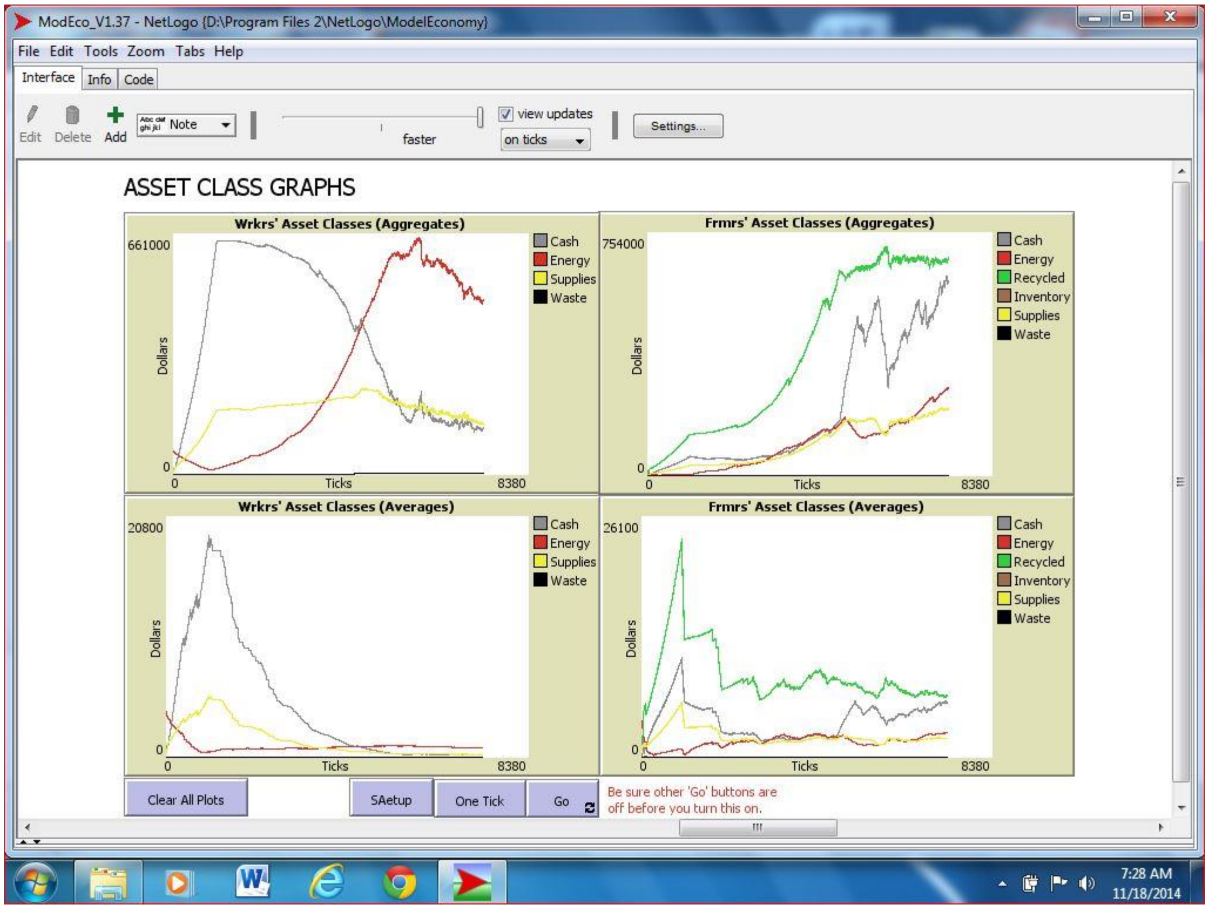
WEALTH AND AGE HISTOGRAMS: Use the gb-plots-on toggle on the main panel to disable these to enable a faster run. If you want to use these plots for data collection, they must be active. Use the 'Clear All Plots' button to clear these and all other plots of their contained data. Right-click on a plot and select 'export' to transfer the data to a CSV file. The standard setup, one tick and go buttons are replicated here. Note that the wealth histogram has a log-normal shape.

Note that the most wealthy frmr is 13 times as wealthy as the most wealthy wrkr. Note that the oldest agent cannot be older than g-DAT = 1,600 ticks, and they can reproduce after age g-RAT = 800 ticks. Those over age g-RAT are either unhealthy (energy less than g-RET = 1,000 eus), or are crowded in the centre of a mass of agents, and have no place to put their offspring, and therefore have deferred reproduction.

These and the following examples come from the 'GROWTH' scenario, g-use-this-seed = 7.

Figure 13 shows the panel of the user interface where the time series plots of asset classes, both aggregate and average per agent, are clustered. These graphs track the key biophysical and economic state variables, when translated into dollar values in the economic system.

Figure 13 – The Asset Class Panel

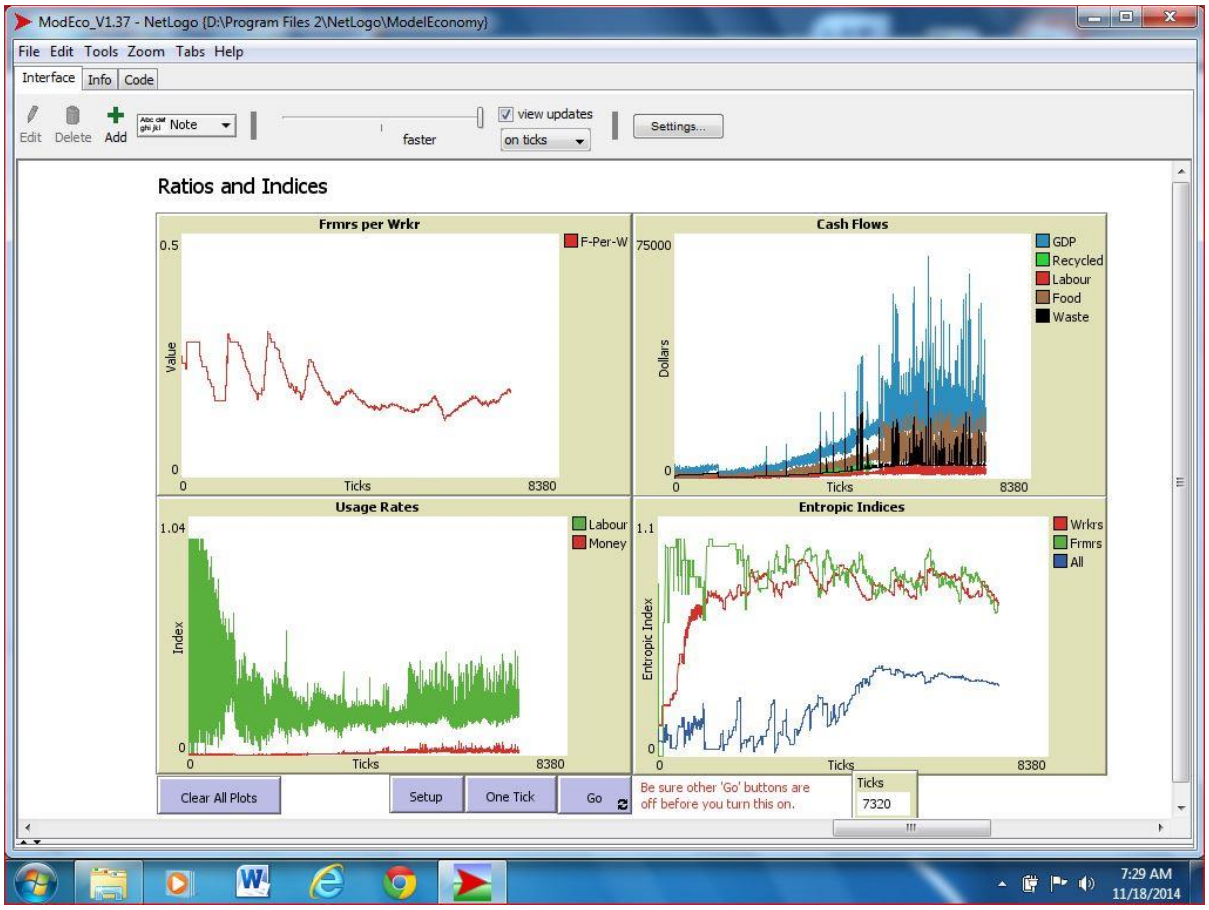

ModEco_V1.37 - NetLogo {D:\Program Files 2\NetLogo\ModelEconomy}
File Edit Tools Zoom Tabs Help
Interface Info Code
Edit Delete Add Note
faster
view updates
on ticks
Settings...
ASSET CLASS GRAPHS
Wrkrs' Asset Classes (Aggregates)
661000
0
Ticks
8380
Dollars
Cash
Energy
Supplies
Waste
Frmrs' Asset Classes (Aggregates)
754000
Recycled
Inventory
Wrkrs' Asset Classes (Averages)
20800
Frmrs' Asset Classes (Averages)
26100
Clear All Plots
SAetup
One Tick
Go
Be sure other 'Go' buttons are off before you turn this on.
7:28 AM
11/18/2014

ASSET CLASS TIME SERIES: Use the gb-plots-on toggle on the main panel to disable these to enable a faster run. If you want to use these plots for data collection, they must be active. Use the 'Clear All Plots' button to clear these and all other plots of their contained data. Right-click on a plot and select 'export' to transfer the data to a CSV file. The standard setup, one tick and go buttons are replicated here.

Note that this data is all pre-steady-state. Nevertheless, you can see two distinct changes of phase in the data. These are most noticeable in the top left plot, when the wrkrs' aggregate cash peaks, and then when the wrkrs' aggregate energy peaks.

Figure 14 shows the panel of the user interface where the time-series plots of various ratios and indices are clustered.

Figure 14 – The Ratios and Indices Time Series Panel

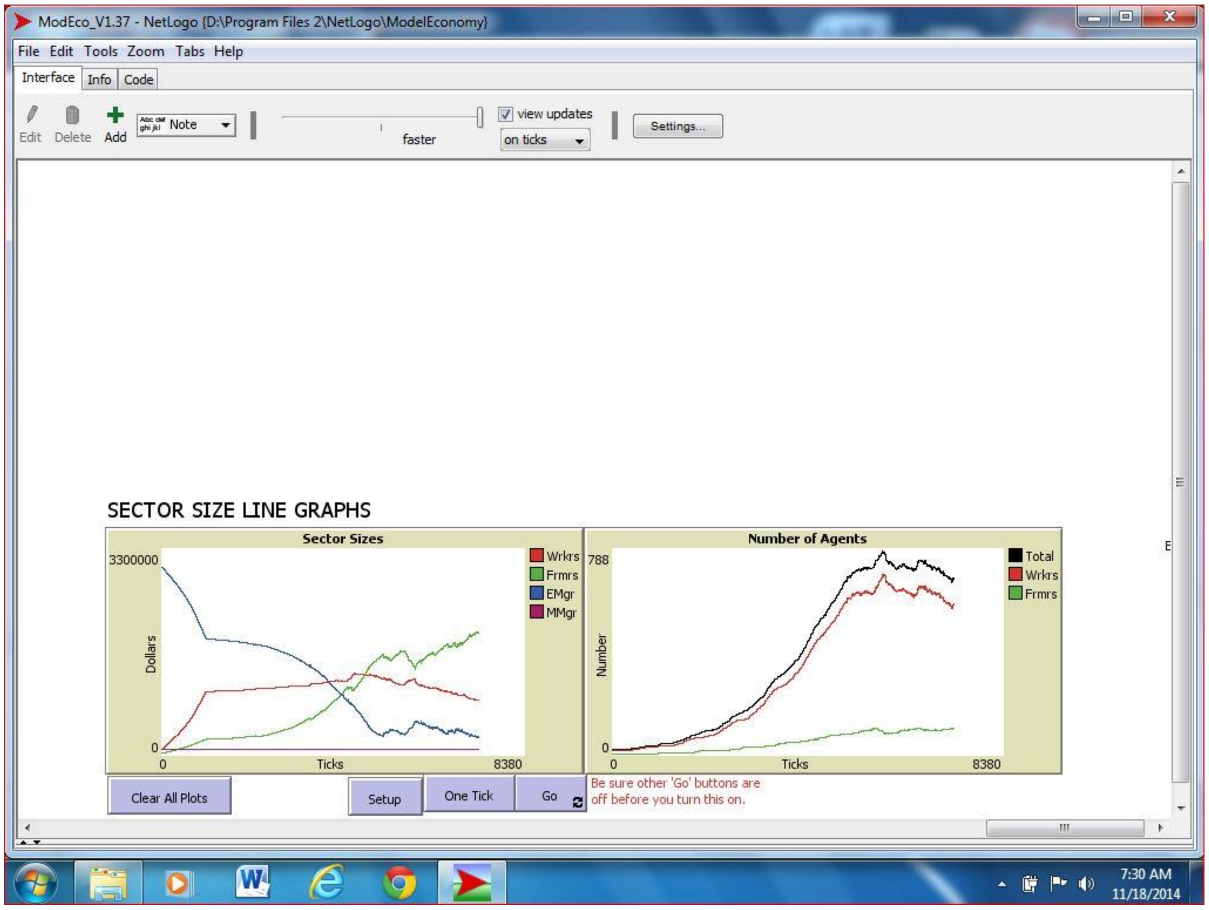


WEALTH AND AGE HISTOGRAMS: Use the gb-plots-on toggle on the main panel to disable these to enable a faster run. If you want to use these plots for data collection, they must be active. Use the 'Clear All Plots' button to clear these and all other plots of their contained data. Right-click on a plot and select 'export' to transfer the data to a CSV file. The standard setup, one tick and go buttons are replicated here.

Note that the phase shifts seen in the previous graphs are exhibited here as well, but are somewhat less obvious. Note that employment rate, the flow of cash, and the GDP become very volatile as the economy approaches steady state. The entropic index is a concept I am developing. (Boyle, 2014 [05]) Due to the small number of agents (under 1,000 – as compared to 10^{23} molecules per mole) it is quite volatile. Like thermodynamic entropy, it has risen asymptotic to a maximal value, and then unlike thermodynamic entropy, it declines as steady state is approached. Note the correlation in the volatility of the wrkr and frmr entropic indices, and the relative stability of the combined entropic index as steady state is approached.

Figure 15 shows the panel of the user interface where the real-time plots of comparative sector statistics are clustered.

Figure 15 – The Sector Size Time Series Panel



SECTOR SIZE TIME SERIES: Use the gb-plots-on toggle on the main panel to disable these to enable a faster run. If you want to use these plots for data collection, they must be active. Use the 'Clear All Plots' button to clear these and all other plots of their contained data. Right-click on a plot and select 'export' to transfer the data to a CSV file. The standard setup, one tick and go buttons are replicated here.

Note that the phase changes mentioned with respect to the previous plots are very clear in the graph showing the value of the four economic sectors (wrkrs, frmrs, EMgr endowment fund, and MMgr). In the first phase, the wealth of the wrkr and frmr sectors rise rapidly as the EMgr gives away a massive portion of its initial endowment (this is the GROWTH scenario), even as the populations remain small, growing exponentially. Suddenly the wrkrs assume linear financial growth while the frmrs continue to experience exponential financial growth, though at a much slower rate. Then, about the time that the population curve hits the inflexion point to level off at carrying capacity (as per the right-hand plot), the value of the wrkr's sector starts to decline, even as the wrkrs' population numbers continue to escalate.

References

01 Boulanger, P.-M., Bréchet, T., 2005. *Models for policy-making in sustainable development: The state of the art and perspectives for research*. Ecological Economics 55, 337-350.

02 Boyle, Garvin H (2013). *Description of An Agent-Based Model of a Sustainable Economy, Using the ODD Protocol.* (C++ ADE). Unpublished note to file.

03 Boyle, Garvin H (2013). *A Simple Complete Conservative Sustainable Agent-Based Model Economy – Some Hypotheses.* Unpublished paper.

04 Boyle, Garvin H (2013). *ModEco and the PMM - A simple physically conservative complete sustainable economy.* (Version 2). *CoMSES Computational Model Library*. Retrieved from: http://www.openabm.org/model/3613/version/2. C++ software from website.

05 Boyle, Garvin H (2014). *A Definition and Examination of an Entropic Index for Agent-Based Models.* Unpublished paper.

06 Boyle, Garvin H (2014). *Primordial Soup (PSoup)* Application download site. Unpublished software. http://orrery-software.webs.com/psoup

07 Boyle, Garvin H (2014). *NetLogo Coding Standards for Orrery Software, Note To File*, Unpublished note to file.

08 Daly, H., 1991. *Steady-State Economics: Second Edition with New Essays*. Island Press, Washington, DC, USA.

09 Dewdney, A. K., May 1989. *Simulated Evolution*. Computer Recreations, Scientific American.

10 Georgescu-Roegen, N., 1986. *The Entropy Law and the Economic Process in Retrospect*. Eastern Economic Journal, Volume XII, No. 1, January-March 1986.

11 Grimm, V., Berger, U., Bastiensen, F., Eliassen, S., Ginot, V., Giske, J., Goss-Custard, J., Grand, T., Heinz, S., Huse, G., Huth, A., Jepsen, J.U., Jørgensen, C., Mooij, W.M., Müller, B., Pe'er, G., Piou, C., Railsback, S.F., Robbins, A.M., Robbins, M.M., Rossmanith, E., Rüger, N., Strand, E., Souissi, S., Stillman, R.A., Vabø, R., Visser, U., DeAngelis, D.L., 2006. *A standard protocol for describing individual-based and agent-based models*. Ecolological Modelling 198, 115-126.

12 Grimm, V., Berger, U., DeAngelis, D.L., Polhill, J.G., Giske, J., Railsback, S.F., 2010. *The ODD protocol: A review and first update*. Ecological Modelling 221, 2760-2768.

13 Hawken, P., 1993. *The Ecology of Commerce: A Declaration of Sustainability*. Collins Business, Harper Collins Publishers, New York, USA.

14 Jackson, T., 2009. *Prosperity Without Growth: Economics For a Finite Planet*. Earthscan, London, UK.

15 Tesfatsion, L., 2002. *Agent-Based Computational Economics: Growing Economies from the Bottom Up*. Artificial Life, Vol. 8, No. 1, 55-82.

16 Yakovenko, V. M., 2010. *Statistical Mechanics Approach to the Probability Distribution of Money*. Department of Physics, University of Maryland, College Park, Maryland, USA.