Description of An Agent-Based Model of a Sustainable Economy, Using the ODD Protocol

Author: Garvin H. Boyle
Date: 30 January 2013

Description of An Agent-Based Model of a Sustainable Economy, Using the ODD Protocol

Table of Contents

Description of An Agent-Based Model of a Sustainable Economy, 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 simple sustainable economies, and ultimately, to identify the necessary and sufficient conditions for sustainability. The first sustainable ModEco-based economy, and the only one achieved to date, has been dubbed the Perpetual Motion Machine #1 (PMM #1). 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 #1 is described 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 #1" (PMM #1) which was constructed using the ModEco application, which is currently under further development. The description will, of necessity, include some information about the ModEco application itself, as may be necessary, but the focus is on PMM #1. 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 with associated technical notes can be downloaded from the website 'modeco-software.webs.com'.

Most modern economies are clearly unsustainable in the long term. Any regime based on the exploitation and consumption of finite stores of non-renewable resources must, of necessity, eventually come to an end. Furthermore, any regime additionally based on the 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 towards one monolithic unsustainable global economy.

The lesson from history is that all economies eventually become sustainable, but usually at a subsistence level, and usually after collapse of the society and irreversible damage to the environment. What is particularly alarming now is the scale of damage caused by our emerging global economy. Previous societies that collapsed caused irreversible damage to their locale. Our modern society is causing irreversible damage on a global scale. So, the question is not "Will the modern global economy, which is now blossoming, ever be sustainable?" Rather, the question is "Can humankind learn how to curb its practices and live sustainably with an elevated standard of living, or will nature return us to sustainable subsistence-level living against our will?

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). 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). An ecological system is a complex adaptive system, the principal nature of which is physical. An economy is also a complex adaptive system, the principal nature of which is, by contrast, social, embedded into and intertwined with the human biophysical system. 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 services which are uncosted by the economic system and are for the most part not well understood. These include the rain forests which provide oxygen in immense quantities, the the hydrological cycle which provides fresh water for food crops, the oceans with their ability to produce immense quantities of food, and the dirt itself with its ability to produce emmense quantities of food. These services are used by all as we draw freely from a common fund, but in recent years we find these funds are being exhausted. We must learn to manage these necessary commons.

Putting all of this together, we define a sustainable economy as one which is conservative of non-renewable resources, restorative of renewable resources, and socially and economically self-perpetuating. In addition, it is mindful of planetary limits, and is accountable for its use of commonly-needed natural resources. Please note that social justice is not part of this definition. The vision of sustainability that includes shared prosperity (Jackson, 2009) adds an additional dimension to the definition. While the ModEco-based economies are designed only to investigate economic sustainability without reference to social justice, some of the output of PMM #1 shows that these two high visions are not totally compatible. But that will be addressed in another paper.

So, how do we discover the characteristics of a sustainable economy? The goal of the ModEco project is to discover the necessary and sufficient characteristics that make an economy sustainable. We are drawing 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 ecological processes. The concepts of SSE are generally considered to have first appeared in the 1966 essay "*Analystical Economics*" by Nicholas Georgescu-Roegen (Georgescu-Roegen, 1986), and was 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 recent study has shown ABM 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 being a very simple and very highly abstracted agricultural economic system, together with the lower-level ecological/biophysical system and processes 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 it step-by-step as the level of abstraction is reduced and the level of realism is increased.

The ModEco application is conceived as a laboratory in which a number of different model economies can be enabled through toggled features and 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. 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 toggle might be used to select one of several pricing mechanisms.

In very general terms a ModEco-based economy has the following characteristics:

- At the lower metabolic level, agents harvest food, eat it, and recycle the waste.
- 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.
- Agents are born (via fission, the most simple form of reproduction), live a life of limited length, and successfully reproduce (via fission) when they are old enough and wealthy enough, or they die of starvation, old age, or bankruptcy. Death can come from metabolic or economic insufficiencies.
- Both mass and energy are conserved in every commercial transaction, meaning that the total mass owned collectively by two agents before a transaction equals the total mass owned by them collectively after the transaction, and the same for energy. There may be a transfer of mass and energy between the agents, but the total amounts do not change.
- Intrinsic value, based on metabolic needs, is assigned to mass and energy, and intrinsic value is also conserved implicitly in all commercial transactions.
- All transfers of goods, and all deliveries of services, are mediated by the reciprocal transfer of cash, which is conserved in each transaction. There is no bartering.
- Farmers and workers do not have global knowledge of the market, but, rather, (a) have knowledge of the existence of only those other agents that reside within a local commuting area, and (b) retain a record of their own average historic costs.

An economy as described above is easy to model using agent-based techniques, and, using the ModEco application, the behaviour of a number of interesting little economies was explored. However, it was discovered that, even with this small set of constraints, a sustainable 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 price paid for all goods and services must be precisely equal to the intrinsic value, based on metabolic needs. When a price/value gap is allowed (i.e. when price negotiations are allowed) then inflation creeps in, and positive feedback mechanisms cause the collapse of the economy, somewhat similar to the way friction eventually brings a spinning wheel to a stop.

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. PMM #1 is a ModEco-based economy in which many of the available ModEco functions and features are toggled off. The purpose of this paper is to describe PMM #1 using the techniques of the ODD protocol. A more general ModEco-based economy will be described in a later paper.

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 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 agent-based models (ABMs), and, for the most part, it was easy to complete most parts of the ODD template. However, the model described below has two qualities that made use of the protocol difficult.

First, ModEco is a generalized application with a large parameter space, and the sustainable economy which is described in this paper uses a relatively small subset of the possible parameters. There is therefore some complexity in the software that need not be included in this description.

Second, ModEco is developed using object-oriented programming (OOP) techniques, in which most concepts are encapsulated in objects containing both computer code and data. This type of approach is particularly suitable for the design of agent-based models, because each agent is an object and each object contains its own state variables. However, when combined with the additional complexity mentioned above, there is not an easy one-to-one mapping between the entities described in this document and the actual objects found in ModEco.

With the goal of making the description as simple and straight-forward as possible, much of the complexity of the OOP structure of ModEco has been suppressed. A comparison of the following description with the actual ModEco C++ code, therefore, would require some care.

3 The Perpetual Motion Machine #1 (PMM #1)

3.1 Overview

PMM #1 is a specific, highly abstracted, agent-based, model agricultural economy that can be established using the ModEco application. It contains four types of agents. *Wrkr*s and *Frmr*s exist in multiple copies scattered through a toroidal 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 #1 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.

The design of PMM #1 requires some explanation. The goal was to produce a sustainable economy that was “the most simple” but one step above a barter system, employing money to facilitate exchanges of value. ModEco has many built-in features that can be turned on and off. These were added over the course of the project with the purpose of understanding why the model economies were, in fact, not sustainable, but always ended in collapse. A substantial amount of experimentation was undertaken,

much of which enhanced the ModEco program, but failed to produce sustainability. The following paragraphs describe some of the thought processes that lead to the discovery of PMM #1.

After spending some time thinking about first-year physics lessons in which students learn about friction-free mechanics, we 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. Friction causes the dissipation of energy, causing the system to run down and stop. Similarly, we reasoned, 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 price-value gap caused by price-negotiations and the resulting 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.

So, in PMM #1 we removed the price/value gap. Every commercial transaction involves a precisely equal exchange of value for 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 will have unchanging wealth throughout their lives. The necessary addition of the central agents, the *MMgr* and the *EMgr*, to the mix changes this, as explained below.

Sustainability in an ecosystem or an economy is closely associated with the concept of resilience, which is the ability of a system to return to a normed state following a severe disturbance. A resiliant system exists 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 resiliant system will eventually return to orbit near the attractor set.

For example, let's imagine an agent-based model of a simple ecological system. Suppose you have a finite toroidal automaton in which logical bugs eat energy-carrying algae. 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 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. But, if the population is too small, then most of the energy will be returned to algae in the field making food easily available and encouraging a return to a higher population number. The number of bugs in such a population is resilient, and the population is sustainable.

In PMM #1 we have chosen to make the system closed with respect to four conserved quantities, 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* (see section 3.1.2.4.2.1) can purchase waste using money it does not have. This ability to buy up assets and inject money into the economy is similar to the concept of "quantitative easing" by which "toxic assets" are purchased by a government. In ModEco, such a technique decouples the sub-systems and allows the economic system to approach its own economy-related steady state while the metabolic system approaches a physics-related steady state.

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 cycle. Two problems arose.

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. While the possibility of having the consumers sell it directly to the farmers was tried, it failed because the mass tended to become more and more centrally concentrated causing a geographical pooling of the scarce resource. This was an emergent phenomenon which was not part of the design intent, and detrimental to the sustainability of the society. To counter this, the material manager (*MMgr*) was invented, having the ability to buy waste from any consumer at any time, and sell it to any *Frmr*, regardless of their location on the board. The role of the *MMgr* seems to be necessary due to (a) its ability to inject money into the system when needed, and also due to (b) its ability to avoid geographic pooling of mass.

Second, when agents die without offspring, where do the assets go? If the economy is to be sustainable, those assets need to be placed back into the economy. The assets were handed over to the estate manager (*EMgr*, see section 3.1.2.4.2.2) to be distributed to needy 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 most needy agents, but includes most agents most of the time. A "needy" agent is one which has sufficient resources to participate in a commercial opportunity, but not enough to maximize participation. Such a needy 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.

3.1.1 Purpose

The purpose of PMM #1 is to demonstrate the construction of a very simple but complete agricultural economy which is truly sustainable from both an ecological and an economic point of view. By "complete" we mean an economy which (a) consists of two subsystems, the economic and the ecological, merged together; (b) conserves mass and energy at every step of the process; and (c) forms a closed conservative mass/energy system at the highest level. The ultimate purpose is to use such complete agent-based models to investigate the behaviour of complete sustainable economies, and to determine the necessary and sufficient conditions of sustainability.

The purpose of this description of PMM #1 using the ODD protocol is to enable others to replicate PMM #1 on a different platform.

3.1.2 Entities, State Variables and Scales

The following entities, together with their state variables and scales of application are described in this section under the indicated subsections:

Entity	Sub-section	Representing
• *ModEcoSys*	3.1.2.1	The dual ecological/economic system
• *Township*	3.1.2.2	The geographic area
• *Lot*s	3.1.2.3	A portion of the *Township*
○ Residences	3.1.2.3.1	A portion of a residential lot
• The economic agents	3.1.2.4	Agents
○ The mortal agents	3.1.2.4.1	Agents in the private sector
▪ *Wrkr*s	3.1.2.4.1.1	Agents who are workers
▪ *Frmr*s	3.1.2.4.1.2	Agents who own land
○ The immortal agents	3.1.2.4.2	Agents in the public sector
▪ The material manager (*MMgr*)	3.1.2.4.2.1	Agent who buys and sells waste
▪ The estate manager (*EMgr*)	3.1.2.4.2.2	Agent who receives and distributes assets
• The geographic scale	3.1.2.5	Relative meanings

- The temporal scale 3.1.2.6 Relative meanings
- Non-participating entities 3.1.2.7 Supporting entities that play no role in the ecological or economic activities
    - Agent lists 3.1.2.7.1 Fixed allocations of memory for agents
    - Contact lists 3.1.2.7.2 Fixed allocations of memory for *Contact*s
    - Real-time data display entities 3.1.2.7.3 Display data in real time on the screen
    - Data capture entities 3.1.2.7.4 Capture data in CSV files

3.1.2.1 ModEcoSys

ModEcoSys represents the combined ecological and economic system. It contains within itself all of the other entities of interest in the model. In addition, in the ModEco implementation, *ModEcoSys* contains a number of variables that disable features not required for PMM #1. These are not relevant to replication of PMM #1, and so, are not described here. Finally, there are a number of variables that control entities that play a supporting role for PMM #1, but which have no material effect on the outcome of a run. These non-participating entities are discussed in section 3.1.2.7 below.

A note about format: All state variables and entities are in italics throughout this document. furthermore, where clarification is useful, the owning entity is prefixed. For example, the *Age* variable is used by many entities. The age of the system can be written as *ModEcoSys*:*Age*.

The *ModEcoSys* entity is characterized by the following state variables and contained entities:

Name	Type	Brief Description
Age	Variable	The count of time increments since inception.
NoOfWrkrs	Variable	The count of *Wrkr*s in the economy.
NoOfFrmrs	Variable	The count of *Frmr*s in the economy.
Township	Entity	The entity which is divided into *Lot*s in which the action happens (see section 3.1.2.2 below).
WrkrList	Entity	A non-participating entity that contains *Wrkr*s (see sections 3.1.2.4.1.1 and 3.1.2.7 below).
FrmrList	Entity	A non-participating entity that contains *Frmr*s (see sections 3.1.2.4.1.2 and 3.1.2.7 below).
ContactList	Entity	A non-participating entity that contains *Contact*s (see section 3.1.2.7 below).

3.1.2.2 Township

PMM #1 is a hybrid model, being part cellular automaton and part agent-based model. The cellular automaton is a rectangular array called the *Township*, having two optional user-selectable sizes. The *Township* simulates the geographic dispersion of agents and assets. All of the action happens within the *Township*. Visually, the *Township* is a rectangular area tiled by green squares called *Lot*s (see section 3.1.2.3 below). 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 geographic effects dramatically more simple.

The *Township* entity is characterized by the following parameters and contained entities:

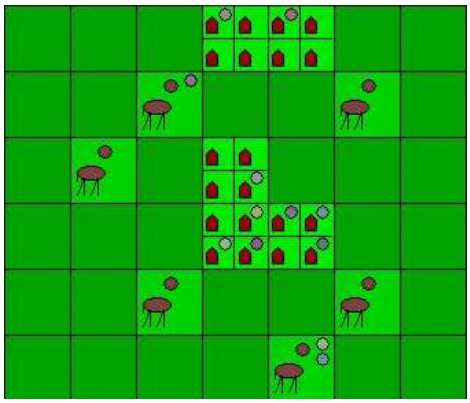
Name	Type	Brief Description
Height	Parameter	The height of the *Township*, in *Lot*s. Integer, 6 or 15.
Width	Parameter	The width of the *Township*, in *Lot*s. Integer, 10 or 20.
k_Size	Parameter	Size of the k-neighbourhood. Is fixed at 2 on compilation. Commuting area is then 5x5 *Lot*s. (See section 3.1.2.4 for a definition of *Lot* and commuting area.)
MMgr	Entity	The material manager who buys waste and sells recycled mass. (See section 3.1.2.4.2.1)
EMgr	Entity	The estate manager who receives the assets of dead agents and redistributes them. (See section 3.1.2.4.2.2)

The values for *Height* and *Width* are set on initialization when the *Township* is constructed, and do not change as the economy changes. The value for *k_Size* is fixed at time of compilation and is not user selectable. In PMM #1, when in steady state, the agents will huddle in a small cluster which tends to wander around the *Township* as time progresses.

The concept of a k-neighbourhood is drawn from the design of cellular automata. All cells, and only those cells, within a distance of k of the cell of interest have the ability to affect the next state of that cell, and those agents within the cell. In PMM #1, the value of *k_Size* controls the geographic extent of the economic knowledge of the agents. This value implicitly controls emergent behaviour such as agent huddling, population density, and delayed reproduction due to overcrowding.

3.1.2.3 Lots

Each cell represents a piece of property in the *Township*. There are three types of *Lot*: unspecified (default), residential and farmed. State variables differ slightly by type of *Lot*. The cellular automaton is "degenerate" in the sense that the cells are passive. Their states are controlled entirely by the action of the mortal agents. The cells have no direct effect on each other, but the concept of a k-neighbourhood is borrowed from cellular automata and used to define commuting areas for the mortal agents (see description of *k_Size* in section 3.1.2.2 above).



Some lots. Unspecified lots are clear green. Farm lots have a horse. Residential lots have four houses. Dots indicate Wrkr's home and job locations. Three Wrkrs got a job today at two farms.

For the purposes of efficiency, each *Lot* contains a logical list of all other *Lot*s within its k-neighbourhood. This list is established on start-up and does not change. It is called the commuting area of the *Lot*. It recognizes the toroidal nature of the *Township* and embodies the tricky task of figuring out exactly which *Lot*s are in the k-neighbourhood of which other *Lot*s. Two mortal agents found within each other's commuting areas are able to interact, via the medium of their *ContactLists*. The size of the commuting area is determined by the variable *k_Size*.

A *Lot* is characterized by the following state variables and values:

Name	Type	Brief Description
LotDevType	Variable	Unspecified, farm or residence. Determined by the type of agent that first settles the *Lot*, and returned to unspecified whenever the *Lot* is vacated.
NoOfResidents	Variable	Zero for unspecified and farm lots. For residential lots, from 1 to 4, the number of *Wrkr*s residing on the residential *Lot*.
X and *Y*	Fixed values	X and Y coordinates of the *Lot*. Values are fixed from the start.

3.1.2.4.1 Residences

Within each residential lot there are four residences, numbered 0 through 3, in which Wrkrs can live. When the first Wrkr moves into an unspecified lot, it becomes a residential lot, and the Wrkr takes up abode in residence number 0. Thereafter up to three Wrkrs may live in the empty residences. When the last Wrkr in such a lot dies or moves out, the lot reverts to type unspecified.

Name	Type	Brief Description
Lot:ResNo	Fixed value	The residence number of each residence is fixed at time of initialization.

3.1.2.4 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 workers (*Wrkr*s) and farmers (*Frmr*s), each of which must reproduce before it dies of old age or starvation, and each of which has limited knowledge of the economy. The immortal agents are the material manager (*MMgr*) and the estate manager (*EMgr*), which are both singular, ageless and have knowledge of the economy not limited to any commuting area.

An understanding of agents requires explanation of the classifications of resources in PMM #1, and the units of measure used for each. PMM #1 has four key resources that are conserved in all transactions. The four conserved quantities are as follows:

- Cash – the units of which are dollars.
- Mass – the units of which are “metabolism-based mass units” or MbMus.
- Metabolic energy – the units of which are “metabolism-based energy units” or MbEus.
- Intrinsic value – the units of which are “dollars”. The intrinsic value of an MbMu and an MbEu are fixed from the start.

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 the cash. However, an optional toggle will allow the MMgr to go into debt, effectively opening the system with respect to cash. In this case, cash is still conserved on a transaction-by-transaction basis, but is not conserved at the system level.

In PMM #1, 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 contained in food and consumed by agents. For the purposes of this description of PMM #1,

let's refer to this kind of energy as metabolically accessible energy, or just metabolic energy. Then, in PMM #1, mass plays the role of being the vehicle by which metabolic energy is collected during harvesting, delivered to the consumer, and eaten.

Mass which is devoid of metabolic energy is measured in metabolism-based mass units (MbMus). One MbMu has, arbitrarily, an intrinsic value of exactly two dollars.

Metabolic energy is measured in metabolism-based energy units (MbEus). One MbEu has, arbitrarily, an intrinsic value of exactly eight dollars.

Mass which is carrying its full complement of metabolic energy is measured in metabolism-based mass/energy units (MbMEus). One MbMEu is equal to one MbMu plus one MbEu. The intrinsic value of an MBMEu is 2 $ (the intrinsic value of the MbMu) + 8 $ (the intrinsic value of the MbEu) = 10 $.

The concept of an MbMEu 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. The metabolic value of food does not change dramatically over time, while, in most ModEco-based economies (but not PMM #1) the monetary value of food can change dramatically. An agent 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 MbMEus per tick to live, comprising 4 MbMus and 4 MbEus, having an arbitrary value of $40.

In ModEco we track both intrinsic value based on metabolic needs and monetary value based on weighted average cost. Within PMM #1 there is redundancy between monetary value and intrinsic value, but both are mentioned in this description to clarify the need for and role of intrinsic value in other less restrictive ModEco-based economies.

The relative characteristics of the four types of agents are as follows:

Agent Name	**Description**	**Geographic characteristics**	**Temporal characteristics**	**Breadth of Knowledge (geographic and temporal)**
Wrkr	A worker; a consumer.	Lives in a house on a residential *Lot* with up to three other *Wrkr*s; works on any farm within commuting area; can move if hungry and unemployed.	Mortal; reproduces via fission; dies of starvation or old age or bankruptcy controlled by user-set parameters.	Local, within its own commuting area; historic, no temporal knowledge.
Frmr	A farmer who works the land; a consumer; hires *Wrkr*s as workers; sells inventory to consumers; buys waste from the *MMgr*.	Lives alone on a farmed *Lot*; cannot move or relocate; reacts with mortal agents within its commuting area.	Mortal; reproduces via fission; dies of starvation or old age or bankruptcy controlled by user-set parameters.	Local, within its own commuting area; historic, no temporal knowledge.

Agent Name	Description	Geographic characteristics	Temporal characteristics	Breadth of Knowledge (geographic and temporal)
MMgr	A central agent; buys waste from consumers (*Frmr*s and *Wrkr*s) and sells it to farmers (*Frmr*s).	Exists everywhere; No visible location in *Township*.	Immortal; neither reproduces nor dies;	Global; can derive knowledge from any agent; can maintain historic records.
EMgr	A central agent; processes the assets of any agent that dies without issue; distributes those assets to needy-but-deserving agents.	Exists everywhere; No visible location in *Township*.	Immortal; neither reproduces nor dies;	Global; moment in time; responds to requests for assets, and provides grants.

3.1.2.4.1 The mortal agents

These agents are the work horses of the economy, and a thriving economy contains many of both types of mortal agent. They inhabit *Lot*s in the *Township*, causing them to differentiate between unspecified *Lot*s, farmed *Lots*, or residential *Lot*s. They actively engage in commercial transactions with other agents, both mortal and immortal. They have metabolic and economic needs which, if not met, will cause them to die. There are two types, workers (called *Wrkr*s) and farmers (called *Frmr*s), and these groups are co-dependent, which is to say, if one type dies out, the other cannot survive long.

*3.1.2.4.1.1 Workers (*Wrkr*s)*

*Wrkr*s are workers and consumers. They hire themselves out to farmers to do work, collect a salary, buy supplies, consume supplies and sell waste. They reproduce via fission, or die of starvation, bankruptcy or old age. *Wrkr*s have both metabolic and economic needs. Farmers (i.e. *Frmr*s) and workers are co-dependant in the economy.

*Wrkr*s are characterized by the following state variables:

Name	**Type**	**Brief Description**
SerNo	Variable	Serial number, unique for each agent, used to collate data.
X, Y and ResNo	Variables	The *Lot* location and residence number used to locate the home base of the *Wrkr*.
Age	Variable	The age of the *Wrkr*, in ticks.
DateLastHired	Variable	The value of *ModEcoSys*:*Age* when the *Wrkr* was last hired.
DateLastSupplied	Variable	The value of *ModEcoSys*:*Age* when the *Wrkr* was last supplied with food.
Cash	Variable	The amount of cash, in dollars.
NoOfMbEus	Variable	The total number of metabolism-based energy units.
NoOfSupplyMbMEus	Variable	The total number of supply mass/energy units.
NoOfWasteMbMus	Variable	The total number of waste metabolism-based mass units.
EmployerList	Entity	A ContactList for all potential employers in the commuting area.
SupplierList	Entity	A ContactList for all potential suppliers in the commuting area.

The fixed relationships between dollars, MbMEus, MbEus and MbMus is described in section 3.2.1.3.

*3.1.2.4.1.2 Farmers (*Frmrs*)*

*Frmr*s are hiring agents, as well as workers and consumers. They hire workers to work with them in the field harvesting food, sell inventory, consume supplies and sell waste. They reproduce via fission, or die of starvation, bankruptcy or old age. Farmers have both metabolic and economic needs. Farmers and workers are co-dependant in the economy.

*Frmr*s are characterized by the following state variables:

Name	Type	Brief Description
SerNo	Variable	Serial number, unique for each agent, used to collate data.
X and Y	Variables	X, Y and Residence number, used to locate the home base of the agent.
Age	Variable	Maintains the age of the agents, in ticks.
NoOfHires	Variable	The count of *Wrkr*s hired so far this tick. Reset to 0 every tick in setup.
DateLastSupplied	Variable	The value of the *Township* clock when the agent was last supplied with food.
Cash	Variable	The amount of cash, in dollars.
NoOfMbEus	Variable	The total number of metabolism-based energy units.
NoOfRecycledMbMus	Variable	The total number of recycled metabolism-based mass units.
NoOfInventoryMbMEus	Variable	The total number of material units in inventory.
NoOfSupplyMbMEus	Variable	The total number of material units in supply.
NoOfWasteMbMus	Variable	The total number of waste metabolism-based mass units.
UnionList	Entity	A ContactList of all potential employees in the commuting area.
SupplierList	Entity	A ContactList of all potential suppliers in the commuting area.
CustomerList	Entity	A ContactList of all potential customers in the commuting area.

The fixed relationships between dollars, MbMEus, MbEus and MbMus is described in section 3.2.1.3.

3.1.2.4.2 The immortal agents

During the development of PMM #1 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 extensive knowledge of the market.

*3.1.2.4.2.1 The material manager (*MMgr*)*

You can imagine this agent as the central manager of the 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.

The *MMgr* is characterized by the following state variables:

Name	Type	Brief Description
Cash	Variable	The amount of cash, in dollars.
NoOfMbMus	Variable	The total number of metabolism-based mass units.

*3.1.2.4.2.2 The estate manager (*EMgr*)*

You can imagine this agent as the central manager who levies estate taxes and redistributes the acquired assets as grants. His role is to strip all conserved assets from a dead agent, and store them, and dole them out in response to requests for grants.

The *EMgr* is characterized by the following state variables:

Name	Type	Brief Description
Cash	Variable	The amount of cash, in dollars.
NoOfMbMus	Variable	The total number of metabolism-based mass units.
NoOfMbEus	Variable	The total number of metabolism-based energy units.

3.1.2.5 Geographic scale

The geographic scale is user selectable, having two options. The "Basic" version is 6 *Lot*s by 10 *Lot*s. The "Crowded" version is 15 *Lot*s by 20 *Lot*s. These sizes are easily adjusted by the programmer, pre-compile, to any larger geographic size in either *Township:Height* or *Township:Width*, as the display is self-scaling to the screen. But, since resources do not belong to *Lot*s, but rather, to agents, the geographic scale has little effect on the outcome of an economic run.

By analogy, one can say a *Lot* (a cell) approximates the amount of farmland required to support five people (i.e. one farmer and four workers).

A commuting area (the k-neighbourhood of the cellular automaton) is easily adjusted by the programmer during compilation of the code, 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. This is controlled by the variable *Township*:*k_Size*.

3.1.2.6 Temporal scale

With respect to the temporal scale, time in ModEco is measured in ticks. All functions (outlined in section 3.1.3) are executed once for all agents during each tick.

The analogy to real-world time is not very relevant to the interpretation of the model, since there is no intention 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 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.

3.1.2.7 Non-Participating entities

ModEco is implemented in Microsoft's C++ with MFC, as found in the Visual Studio Professional 2010 product. This is an object-oriented programming (OOP) application development environment, and it requires that each significant concept be encapsulated in an individualized OOP-style object. These OOP-style "objects" correspond only roughly to the "entities" to be described as part of this ODD protocol.

In ModEco, many such objects 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. These objects contain no

parameters or toggles, no state variables, and no embedded entities which affect the outcome of a run of any economy. They are therefore considered non-participating entities.

Nevertheless, some understanding of these non-participating entities is required for complete understanding of the pseudocode provided in section 3.1.3. Furthermore, any attempt to replicate PMM #1 will require some attention to similar functions.

3.1.2.7.1 Agent lists

For reasons of efficiency (to avoid the use of excessive computer resources allocating and releasing space in memory) the system contains a pre-allocated list of *Wrkr*s (both activated and deactivated) and a list of *Frmr*s (both activated and deactivated). These are both part of the *ModEcoSys* entity as *ModEcoSys:WrkrList* and *ModEcoSys:FrmrList*. Both lists are of fixed size and location in memory. When an agent is born, one of the inactive agents in memory is initialized appropriately and activated. When an agent dies, its assets are processed by the estate manager (see section 3.1.2.4.2.2) and it is deactivated. The activated *Wrkr*s form a linked list which comprises a subset of the total list. When the code loops through all *Wrkr*s, it only visits the active linked *Wrkr*s. The *ModEcoSys:FrmrList* is treated similarly.

The benefit of this approach is this: even though the number of *Wrkr*s and *Frmr*s fluctuates regularly, the computer need not allocate and deallocate space with each change in number. The drawback is in the overall limitation on the number of agents. Populations cannot grow beyond the built-in limitation on the list size.

3.1.2.7.2 Contact lists

Again, for reasons of efficiency, each agent must have access to a variable sized *ContactList* for each type of function that involves transactions with other agents. For example, *Frmr*s need lists of all potential hires (*Frmr*:*UnionList*), all potential customers for its inventory (*Frmr*:*CustomerList*) or all potential suppliers (*Frmr*:*SupplierList*). In ModEco, a single massive contact card list is built, and each agent is responsible to maintain its contacts using a linked list which is a subset of the massive list. When an agent contacts another agent, it is via a contact card in its private linked list, but which is stored in the common *ContactList*. These implementation details have no effect on the behaviour of PMM #1, except for speed of execution. But, when reading the pseudocode in section 3.1.3, the various contact lists are mentioned.

3.1.2.7.3 Real-time data display entities

ModEco offers a variety of real-time data displays of individual agent data and aggregated economic data. There are three types of entities that address this type of need:

- Individual agent-level data about the current status of agents is drawn directly from the agents and displayed. This is handled by a drawing module.
- Aggregated data about the current status of the economy is maintained in an “aggregator” entity that is updated with every tick, and displayed by the same drawing module. This is called “per tick” data.
- Historic aggregated data of a selected subset of data in the Aggregator can also be displayed. This data is stored in a data base with revolving records. That is, there are typically 60 records in a linked list. As a new record is added, it is written over the oldest record, and marked as the newest record. This data is also displayed by the same real-time drawing model. This is called “per increment” data, and a record is taken at regular increments.

3.1.2.7.4 Data capture entities

ModEco also offers a variety of data capture options. Data is sent in a stream to CSV (comma-separated value) files which are accessible via spreadsheet software such as Microsoft’s Excel 2010 package.

Each function, as outlined in section 3.1.3, can generate micro-economic data, transaction-by-transaction, tick-by-tick, in a unique file. So, if a user wishes to study the "ConsumeSupplies" function, that data stream can be turned on, and just over 1 million records of that type collected.

For hybrid cross-sectional/longitudinal population studies, a set of micro-economic readings can be taken once every generation (determined by the *Frmr:ART* and *Wrkr:ART* parameters).

This rather large 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 pseudocode 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 #1, it greatly complicates an already complex set of pseudocode modules, and so was intentionally left out.

3.1.3 Process Overview and Scheduling

With each tick of the ModEco clock the following schedule of functions is executed:

- ***Setup*** – in which lists and display indicators are reset, if needed;
- ***Hire Wrkrs*** – in which *Frmr*s canvass *Wrkr*s within their commuting area in a random order and hire them to harvest food from the *Lot*.
- ***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.
- ***Sell Inventory*** – in which *Frmr*s (i.e. farmers) sell inventory to 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.
- ***Consume supplies*** – in which *Frmr*s and *Wrkr*s consume an amount of supplies determined by a user-controlled parameter. Those which have insufficient supplies to meet the day's requirements are marked as starved, and later, removed.
- ***Sell waste*** – in which *Frmr*s and *Wrkr*s sell waste to the *MMgr* at value.
- ***Buy recycled*** – in which *Frmr*s purchase waste from the *MMgr* to enable a good harvest in the next tick.
- ***Agents reproduce*** – in which *Frmr*s and *Wrkr*s undergo fission if they are mature enough (i.e. old enough) and healthy enough. One daughter agent replaces the parent, geographically. The other must find an empty *Lot* or residence to occupy.
- ***Agents die*** – in which *Frmr*s and *Wrkr*s die if they are too poor, too hungry or too old.
- ***Cleanup*** – in which aggregate micro- and macro-economic and ecological data is compiled, visual displays are updated, termination or pause conditions are checked.

There are a number of exogenous parameters which play a significant role in PMM #1. Most of these are under the user's control. However, note that PMM #1 is sustainable for the default values, but unsustainable for many settings of these parameters. In the pseudocode that follows, you will identify the parameters by the prefixes *Parm:* or *Quota:*. These parameters are described in section 3.3.3.1.

The following sections describe each of these functions in detail.

3.1.3.1 Setup

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

```
Setup()
   If necessary, rebuild contact lists
      For each active Frmr in ModEcoSys:FrmrList
         Build Frmr:UnionList - Wrkrs within commuting area
         Build Frmr:CustomerList – Wrkrs and Frmrs within commuting area
         Build Frmr:SupplierList - Frmrs within commuting area
      End For
      For each active Wrkr in ModEcoSys:WrkrList
         Build Wrkr:EmployerList - Frmrs within commuting area
         Build Wrkr:SupplierList - Frmrs within commuting area
      End For
   End If
   For each active Frmr
      ClearDisplayedDots() // Removing indications of previous hires.
   End For
End Setup
```

The building of these lists is different from initialization (see section 3.3.1), in which the memory for the lists is allocated. Here, only if needed, each agent canvasses each *Lot* within its commuting area and activates a contact record, inserting it into its own linked list of such records. The contact record is activated in the common memory area. However, when each contact record is activated it is linked to the activating agent, and all additional records "inserted" into the agent's list are actually activated in the common memory and linked into the ring of records belonging to the associated list of this agent. So, for example, a *Frmr* will build a linked list of all potential hires, and call it the *Frmr:UnionList*. Another *Frmr* in the neighbourhood will build its own such list. A *Wrkr* in the neighborhood will have two records in it's *Wrkr:EmployerList*, one for each *Frmr*. Each *Frmr* will have one record each in their respective *Frmr:UnionList*. Each contact thus results in two reciprocating entries in the common contact list, but those two entries are linked into the linked lists of the two agents involved in the contact, one per agent.

3.1.3.2 Hire Wrkrs

```
Hire Wrkrs()
   For each active Frmr in ModEcoSys:FrmrList, in random order
      For each potential employee in Frmr:UnionList, in random order
         If Frmr:Cash is zero, break
         If Frmr:NoOfMbEus is zero, break
         If Frmr:NoOfRecycledMbMus is zero, break
         If Frmr:NoOfHires equals four, break
         // Contact the potential employee, a Wrkr
         If (Wrkr:NoOfMbEus > zero) and (Wrkr:DateLastHired is not ModEcoSys:Age)
            MakeJobOffer() (see below)
            If successful
               Set Wrkr:DateLastHired = ModEcoSys:Age
               Increment Frmr:NoOfHires
            End If
         End If
      End For
   End For
End Hire 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 his 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 stores (*Frmr:Cash*,

Frmr:NoOfRecycledMbMus; *Frmr:NoOfMbEus* and *Wrkr:NoOfMbEus*). The "Drawn" variables are temporary draws from: *Frmr:NoOfMbEus, Frmr:NoOfRecycledMbMus, Frmr:Cash*, and *Wrkr:NoOfMbEus*.

The requirements for completion of the work (i.e. the 'completion conditions') are:

Wrkr:MbEusDrawn <= Quota: DailyWorkRate
Frmr:MbEusDrawn = Wrkr:MbEusDrawn
Frmr:RecycledMbMusDrawn = Frmr:MbEusDrawn + Wrkr:MbEusDrawn
Frmr:CashDrawn = Intrinsic value of *Wrkr:MbEusDrawn*

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*.

So, the concept of the following routine is (a) address cash insufficiency first, if any, since there are no cash grants; (b) apply for grants to address other insufficiencies; (c) pro-rate all downwards to the lowest remaining insufficiency after available grants are included; (d) draw the grants, if any; and (e) do the work, draw the resources, and convert the recycled mass into food and place it in the inventory.

The MakeJobOffer routine is the most complicated and problematic routine in the model. Pseudocode for the "MakeJobOffer" routine looks like this:

```
MakeJobOffer()
   Local variables: F_RecycledMbMusDrawn, F_MbEusDrawn, W_MbEusDrawn
   Local variables: ExtendedPrice
   Local variables: F_MbEuGrant, W_MbEuGrant, F_MbMuGrant
   Set F_RecycledMbMusDrawn = 2 x Quota:DailyWorkRate
   Set F_MbEusDrawn = Quota:DailyWorkRate
   Set W_MbEusDrawn = Quota:DailyWorkRate
   Compute ExtendedPrice = 8 * W_MbEusDrawn

   If ExtendedPrice > Frmr:Cash,
      Set ExtendPrice = Frmr:Cash
      Prorate three drawn variables downwards
   End If

   If (EMgr:NoOfMbEus > 0) or (EMgr:NoOfMbMus > 0)
      // Grants are available
      If Wrkr:NoOfMbEus < W_MbEusDrawn, apply for W_MbEuGrant
      If Frmr:NoOfMbEus < F_MbEusDrawn, apply for F_MbEuGrant
      If Frmr:NoOfRecycledMbMus < F_RecycledMbMusDrawn, apply for F_MbMuGrant
      AdjustAllForCompletionConditions()
      // Wrkr gets W_MbEuGrant, if appropriate
         Subtract W_MbEuGrant from EMgr:NoOfMbEus
         Add W_MbEuGrant to Wrkr:NoOfMbEus
      // Frmr gets F_MbEuGrant, if appropriate
         Subtract F_MbEuGrant from EMgr:NoOfMbEus
         Add F_MbEuGrant to Frmr:NoOfMbEus
      // Frmr gets F_MbMuGrant, if appropriate
         Subtract F_MbMuGrant from EMgr:NoOfMbMus
         Add F_MbMuGrant to Frmr:NoOfRecycledMbMus
   End If

   Else
      AdjustDrawsForCompletionConditions()
   End Else
```

```
   Subtract F_MbEusDrawn from Frmr:NoOfMbEus
   Subtract F_MbMusDrawn from Frmr:NoOfRecycledMbMus
   Subtract W_MbEusDrawn from Wrkr:NoOfMbEus
   Add F_MbMusDrawn to Frmr:NoOfInventoryMbMEus
   Subtract ExtendedPrice from Frmr:Cash
   Add ExtendPrice to Wrkr:Cash
End MakeJobOffer()
```

The low-level pseudocode for AdjustAllForCompletionConditions() and AdjustDrawsForCompletionConditions() is not shown here. One or some of the draws may be less than allowed under the completion conditions, and all draws need to be reduced to match the least sufficient draw. This may require adjustment of some grants, as well. It's not tricky, but it takes quite a bit of code. Find the least sufficient draw after grants are counted, pro-rate all draws downwards to meet completion conditions, then adjust grants. The first version addresses grants and associated draws. The second version addresses only draws when no grants are involved.

3.1.3.3 Move Wrkrs

The concept is, a worker who is unemployed and broke will move in search of better employment opportunities. *Wrkr:Networth* is a temporary aggregate defined as the total monetary value of the Wrkr. The pseudocode for the Move*Wrkr*s() routine is as follows:

```
MoveWrkrs()
   Local variables: NetWorth
   For each active Wrkr in ModEoSys:WrkrList, in random order
      Compute NetWorth of the Wrkr
      If (Wrkr:DateLastHired <> ModEcoSys:Age) and (NetWorth < Parm:W_CMT)
         // Find a vacant residential slot in the commuting area
            Make a list of empty (Lot, ResidenceNo) pairs in the commuting area
            Select a random pair from the list
         // Move the Wrkr from old lot to new lot
            Remove Wrkr from all lists of all agents in old commuting area
            Move Wrkr to new location, updating Wrkr:X, Wrkr:Y, Wrkr:ResNo
            Add Wrkr to all lists of all agents in new commuting area
            Rebuild Contact lists of Wrkr for new commuting area
      End If
End MoveWrkrs()
```

3.1.3.4 Sell inventory

The pseudocode for the SellInventory() function follows:

```
SellInventory()
   For each active Frmr in ModEcoSys:FrmrList, in random order
      If Frmr:NoOfInventoryMbMEus > 0
         Rebuild Frmr:CustomerList, if necessary
         IssueSalesPitches()
      End If
   End For
End SellInventory()
```

A customer may be a *Wrkr* or a *Frmr*. The pseudocode for the IssueSalesPitches() routine follows:

```
IssueSalesPitches()
   For each active Customer in Frmr:CustomerList, in random order
      If Frmr:NoOfInventoryMbMEus = 0, break
      If ( Customer:Cash > 0 ) And ( Customer:IsPlanningToBuySupplies() )
         IssueSalesPitch()
```

```
      End If
   End For
End IssueSalesPitches()
```

Networth is a temporary variable. The pseudocode for the IsPlanningToBuySupplies() routine follows:

```
IsPlanningToBuySupplies()
   Local variables: NetWorth
   Compute NetWorth of customer
   If Customer:NoOfSupplyMbMEus < ( Parm:BSF * NetWorth )
      {Yes}
   Else
      {No}
End IsPlanningToBuySupplies()
```

The pseudocode for the IssueSalesPitch() routine follows:

```
IssueSalesPitch()
   Local variables: MbMEusSold, Grant, ExtendedPrice
   Set MbMEusSold equal to Quota: MaxInvMbMEusSold
   If Frmr:NoOfInventoryMbMEus < MbMEusSold {Reduce MbMEusSold to match}
   Compute ExtendedPrice
   If Customer:Cash is insufficient, apply for a Grant

   If Customer:Cash + Grant is still insufficient
      Reduce MbMEusSold
      Reduce Grant
   End If

   // Draw the Grant, if it is available and needed
      Subtract Grant from EMgr:Cash
      Add Grant to Customer:Cash

   // Complete the transaction
      Subtract ExtendedPrice from Customer:Cash
      Add ExtendPrice to Frmr:Cash
      Subtract MbMEusSold from Frmr:NoOfInventoryMbMEus
      Add MbMEusSold to Customer:NoOfSupplyMbMEus
End IssueSalesPitch()
```

3.1.3.5 Consume supplies

The pseudocode for the ConsumeSupplies() function follows:

```
ConsumeSupplies()
   Local variables: MbMEusToConsume, MbMusWithdrawn, and MbEusWithdrawn
   For each active Wrkr in ModEcoSys:WrkrList, in random order
      Set MbMEusToConsume = Parm:W_MPT
      If Wrkr:NoOfSupplyMbMEus < MbMEusToConsume {Set Wrkr:HasStarvedFlag = True}
      Else
         Subtract MbMEusToConsume from Wrkr:NoOfSupplyMbMEus
            as MbMusWithdrawn and MbEusWithdrawn
         Add MbMusWithdrawn in Wrkr:NoOfWasteMbMus
         Add MbEusWithdrawn in Wrkr:NoOfMbEus
      End Else
   End For

   For each active Frmr in ModEcoSys:FrmrList, in random order
      Set MbMEusToConsume = Parm:F_MPT
      If Frmr:NoOfSupplyMbMEus < MbMEusToConsume {Set Frmr:HasStarvedFlag = True}
```

```
      Else
         Subtract MbMEusToConsume from Frmr:NoOfSupplyMbMEus
            as MbMusWithdrawn and MbEusWithdrawn
         Add MbMusWithdrawn to Frmr:NoOfWasteMbMus
         Add MbEusWithdrawn to Frmr:NoOfMbEus
      End Else
   End For
End ConsumeSupplies()
```

*3.1.3.6 S*ell Waste

The pseudocode for the SellWaste() function follows:

```
SellWaste()
   Local variables: ExtendedPrice, MbMusSold
   For each active Wrkr in ModEcoSys:WrkrList, in random order
      Set MbMusSold = Quota:MaxWsteMbMusSold
      If Wrkr:NoOfWasteMbMus < MbMusSold {Set MbMusSold = Wrkr:NoOfWasteMbMus}
      Compute ExtendedPrice
      Complete the transaction
         Withdraw ExtendedPrice from MMgr:Cash
         Deposit ExtendedPrice into Wrkr:Cash
         Withdraw MbMusSold from Wrkr:NoOfWasteMbMus
         Store MbMusSold into MMgr:NoOfMbMus
   End For

   For each active Frmr in ModEcoSys:FrmrList, in random order
      Set MbMusSold = Quota:MaxWsteMbMusSold
      If Frmr:NoOfWasteMbMus < MbMusSold {Set MbMusSold = Frmr:NoOfWasteMbMus}
      Compute ExtendedPrice
      // Complete the transaction
         Subtract ExtendedPrice from MMgr:Cash
         Add ExtendedPrice to Frmr:Cash
         Subtract MbMusSold from Frmr:NoOfWasteMbMus
         Add MbMusSold to MMgr:NoOfMbMus
   End For
End SellWaste()
```

3.1.3.7 Buy Recycled

The pseudocode for the BuyRecycled() function follows:

```
BuyRecycled()
   Local variables: Networth, MbMusSold, ExtendedPrice
   For each active Frmr in ModEcoSys:FrmrList, in random order
      Compute NetWorth
      If ( Frmr:NoOfRecycledMbMus < ( Parm:F_BRF * Frmr:NetWorth) )
        and ( Frmr:Cash > 0 )
        and ( MMgr:NoOfMbMus > 0 )
         Set MbMusSold = Quota:MaxRecMbMusSold
         If MMgr:NoOfMbMus < MbMusSold {Set MbMusSold = MMgr:NoOfMbMus}
         Compute ExtendedPrice
         If ExtendedPrice > Frmr:Cash {Prorate MbMusSold to match}
         // Complete the transaction
            Subtract ExtendedPrice from Frmr:Cash
            Add ExtendedPrice to MMgr:Cash
            Subtract MbMusSold from MMgr:NoOfMbMus
            Add MbMusSold to Frmr:NoOfRecycledMbMus
      End If
   End For
```

```
End BuyRecycled()
```

3.1.3.8 Agents reproduce

The pseudocode for the DoAgentsRepro() function follows:

```
DoAgentsRepro()
   Local variables: NetWorth
   For each active Frmr in ModEcoSys:FrmrList, in random order
      Compute NetWorth
      If    ( NetWorth > Parm:F_CRT )
        and ( Frmr:Age > Parm:F_ART ) {ReproduceFrmr()}
   End For

   For each active Wrkr in ModEcoSys:WrkrList in random order
      Compute NetWorth
      If    ( NetWorth > Parm:W_CRT )
        and ( Wrkr:Age > Parm:W_ART ) {ReproduceWrkr()}
   End For
End DoAgentsRepro()
```

The pseudocode for the Reproduce*Wrkr*() function follows:

```
Reproduce Wrkr()
   Compute location of a random vacant residence within commuting area.
   If there are no such residences {break}
   Activate daughters 1 and 2 (D1 and D2)
   Assign D1:SerNo and D2:SerNo
   Withdraw parent Wrkr from all lists
   Move parent Wrkr out of residence, and move D1 into parent's residence
   Move D2 into newly found vacant residence
      If residence is in a vacant Lot {Set Lot:LotDevType}
   Increment Lot:NoOfResidents

   For both D1 and D2, do the following:
      Set Dx:DateLastSupplied = 0
      Set Dx:Cash = Wrkr:Cash / 2
      Set Dx:NoOfMbEus = Wrkr:NoOfMbEus / 2
      Set Dx:NoOfSupplyMbMEus = Wrkr:NoOfSupplyMbMEus / 2
      Set Dx:NoOfWasteMbMus = Wrkr:NoOfWasteMbMus / 2
      Add Dx to list of all agents in the new commuting area
      Build Dx contact lists
   End For
End Reproduce Wrkr()
```

The pseudocode for the Reproduce*Frmr*() function is a little different, and it follows:

```
Reproduce Frmr()
   Compute location of a random vacant Lot within commuting area.
   If there are no such Lots {break}
   Activate daughters 1 and 2 (D1 and D2)
   Assign D1:SerNo and D2:SerNo
   Withdraw parent Frmr from all lists
   Move parent Frmr out of Lot, and move D1 into parent's Lot
   Move D2 into newly found vacant Lot
   Set Lot:LotDevType = Farmed Lot
   For both D1 and D2, do the following:
      Set Dx:NoOfHires = 0
      Set Dx:DateLastSupplied = 0
```

```
      Set Dx:Cash = Frmr:Cash / 2
      Set Dx:NoOfRecycledMbMus = Frmr:NoOfRecycledMbMus / 2
      Set Dx:NoOfMbEus = Frmr:NoOfMbEus / 2
      Set Dx:NoOfSupplyMbMEus = Frmr:NoOfSupplyMbMEus / 2
      Set Dx:NoOfInventoryMbMEus = Frmr:NoOfInventoryMbMEus / 2
      Set Dx:NoOfWasteMbMus = Frmr:NoOfWasteMbMus / 2
      Add Dx to list of all agents in the new commuting area
      Build Dx contact lists
   End For
End ReproduceFrmr()
```

3.1.3.9 Agents die

The pseudocode for the DoAgentsDeath() function is a little different, and it follows:

```
DoAgentsDeath()
   Local variables: NetWorth,
   For each active Frmr, in random order
      Compute NetWorth
      If   ( NetWorth < Parm:F_CDT )
        or ( Frmr:Age > Parm:F_ADT )
        or ( Frmr:HasStarvedFlag = True ) { KillFrmr() }
   End For
   For each active Wrkr, in random order
      Compute NetWorth
      If   ( NetWorth < Parm:W_CDT )
        or ( Wrkr:Age > Parm:W_ADT )
        or ( Wrkr:HasStarvedFlag = True ) { KillWrkr() }
   End For
End DoAgentsDeath()
```

The pseudocode for the Kill*Wrkr*() function follows:

```
KillWrkr()
   Local variables: SupplyMbEus, SupplyMbMus
   Withdraw Wrkr from all lists
   Move Wrkr out of residence
   Decrement Lot:NoOfResidents
   If residence is now a vacant Lot {Set Lot:LotDevType = Unspec}
   // Process the estate assets
      Add Wrkr:Cash to EMgr:Cash
      Add Wrkr:MbEus to EMgr:MbEus:NoOfMbEus
      Separate Wrkr:Supplies:NoOfMbMEus to SupplyMbMus and SupplyMbEus
      Add SupplyMbMus to EMgr:MbMus:NoOfMbMus
      Add SupplyMbEus to EMgr:MbEus:NoOfMbEus
      Add Wrkr:NoOfWasteMbMus to EMgr: MbMus:NoOfMbMus
   Deactivate Wrkr in ModEcoSys:WrkrList
End KillWrkr()
```

The pseudocode for the Kill*Frmr*() function is a little different, and it follows:

```
KillFrmr()
   Local variables: SupplyMbEus, SupplyMbMus
   Local variables: InventoryMbEus, InventoryMbMus
   Withdraw Frmr from all lists
   Move Frmr out of Lot
   Set Lot:LotDevType = Unspec
   // Process the estate assets
      Add Frmr:Cash to EMgr:Cash
```

```
      Add Frmr:MbEus to EMgr:NoOfMbEus
      Add Frmr:NoOfRecycledMbMus to EMgr:NoOfMbMus
      Separate Frmr:NoOfInventoryMbMEus to InventoryMbEus and InventoryMbMus
      Add InventoryMbMus to EMgr:NoOfMbMus
      Add InventoryMbEus to EMgr:NoOfMbEus
      Separate Frmr:Supplies:NoOfSupplyMbMEus to SupplyMbMus and SupplyMbEus
      Add SupplyMbMus to EMgr:NoOfMbMus
      Add SupplyMbEus to EMgr:NoOfMbEus
      Add Frmr:NoOfWasteMbMus to EMgr:NoOfMbMus
   Deactivate Wrkr in ModEcoSys:WrkrList
End KillFrmr()
```

3.1.3.10 Cleanup

The pseudocode for the DoCleanup() function is a little different, and it follows:

```
DoCleanup()
   Update all real-time visual displays
   Increment ModEcoSys:Age
   For each Wrkr {Imcrement Wrkr:Age}
   For each Frmr {Increment Wrkr:Age}
End DoCleanup()
```

3.2 Design Concepts

The following are the eleven design concepts of which a description is required in the ODD protocol specification.

3.2.1 Basic Principles

There are several basic design principles, as follows:

3.2.1.1 Two systems co-evolving

Society will be modelled at two levels: at the metabolic level and at the economic level.

At the metabolic level all mortal agents, both *Wrkr*s and *Frmr*s, experience daily cycles and a life cycle.

The daily cycle will happen once per tick, as described above in section 3.1.3. That is, all agents will work in the fields to harvest recycled mass and imbue it with added value, converting it into food. They will eat the food, transferring the energy into their bodies and producing waste. The waste will be spread back on the fields, via the intermediary action of the *MMgr*, ready for reharvesting.

The life cycle happens over many ticks, as determined by the "metabolic parameters" as described in section 3.3.3.2. A mature and wealthy agent will reproduce via fission after it comes of age, at which time it will cease to exist. The newly born agents will eat and work, absorbing energy and expending energy, until they also come of age to reproduce. If they are unable to reproduce due to poor finances when they come of age, they will eventually die of old age. They can die of lack of wealth, lack of food, or old age.

At the economic level, farmers hire workers, and then they work in the field together to gather recycled mass and imbue it with added value, turning it into food, at which point it is placed into inventory. Essentially, the farmer pays the worker for its energy expenditure, and receives the benefit of energized food in its inventory. Inventory is then sold to consumers (both *Frmr*s and *Wrkr*s) and placed into their supply stores. Food is then consumed by the agents, and the waste is sold to the *MMgr*. Finally, the *MMgr* sells the waste to the farmers to be spread on the fields.

3.2.1.2 Level of Abstraction

The level of abstraction is very high. A ModEco economy is the most simple economy possible, while being complete. Simplicity comes from extremely high-level abstraction of metabolic changes and economic changes. By "complete" we mean resources move from the field into the economy and are recycled back into the field.

3.2.1.3 Conserved Quantities

Four quantities (mass, energy, intrinsic value and cash) are conserved in every metabolic change and every commercial transaction, and these four conserved quantities cycle endlessly through the economy. Mass is measured in metabolism-based mass units (MbMus). Energy is measured in metabolism-based energy units (MbEus). A combination of mass and energy that forms food is measured in metabolism-based mass/energy units (MbMEus). Instrinsic value is measured in dollars ($). And cash is measured in dollars ($). In PMM #1 these units have fixed relationships.

1 MbMEu = 1 MbMu + 1 MbEu
1 MbMu costs 2 $
1 MbEu costs 8 $

3.2.1.4 Cycles

Mass (goods) and energy (services) and food (combined mass and energy) will carry pre-determined intrinsic value, and will cycle through the economy in the opposite direction from cash. Agents will use cash to facilitate all exchanges of goods and services, which will be sold at their intrinsic value.

3.2.2 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 #1, including:

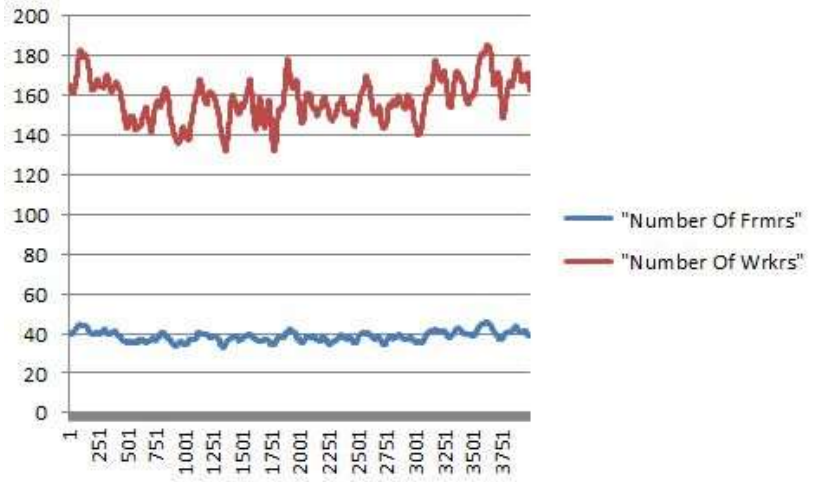
- Automatic relative scaling of *Wrkr* and *Frmr* population sizes;
- Automatic relative scaling of *Wrkr* and *Frmr* division of wealth between sectors;
- Log-normal distribution of wealth;
- The geographic formation of concentrations of agents; and
- A variety of macro-economic indicators such as velocity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

3.2.2.1 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 approximately 4 Wrkrs per Frmr, which I am happy to say was the design intent. So the ratio is as expected, but the absolute size is a surprise.

The steady-state population sizes are determined by initialization parameters, particularly those which specify levels 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 at 4:1. 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.

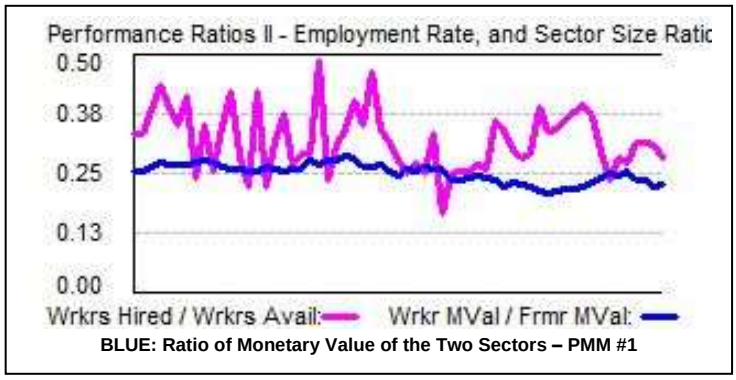
It could be argued that this is not a truly emergent property, since, of necessity, any sustained population living on a finite but recycled pool of resources will be of finite non-zero size. Particularly so, because agents with too little wealth die, under the control of a parameter. And if true for 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.



Population sizes of Wrkrs and Frmrs rise and fall over time within the PMM. The time values are in tens of ticks, 0-40,000 ticks.

3.2.2.2 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.



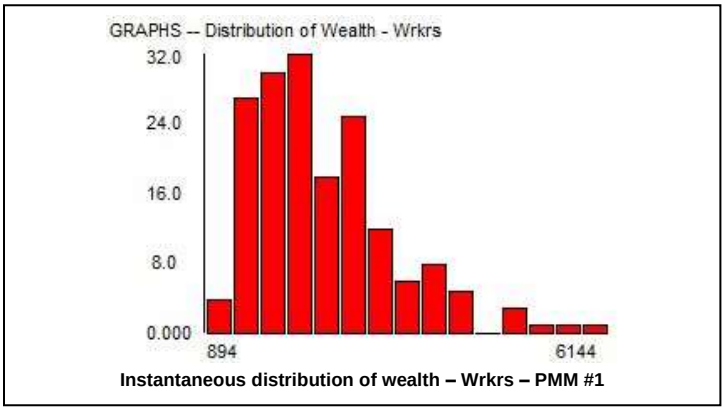
BLUE: Ratio of Monetary Value of the Two Sectors – PMM #1

The relative size of the two sectors seems to stabilize at a ratio of approximately 1$ in the Wrkr sector to every 4$ or 5$ in the Frmr sector. On average, the wealth of the Frmr is from 16 to 20 times the wealth of the Wrkr.

3.2.2.3 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.

It is highly reminiscent of the Boltzmann distribution of energies (Yakovenko, 2010). A phenomenon analogous to the "Maximum Entropy Principle" (MEP) may

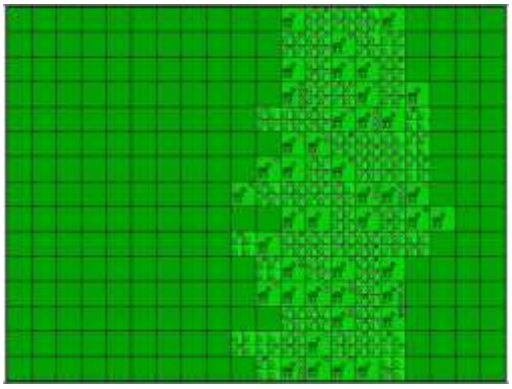


Instantaneous distribution of wealth – Wrkrs – PMM #1

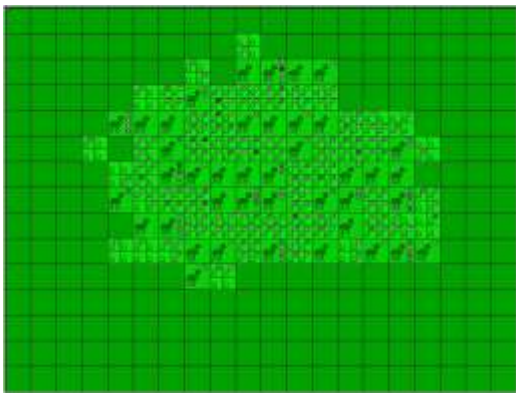
be causing this distribution.

3.2.2.4 Geographic concentrations

Regardless of the geographic dispersion of agents on initialization, when PMM #1 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 films.
This phenomenon has some deep implications worth exploring.



Example of a film-like geographic distribution.



Example of a drop-like geographic distribution.

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.

3.2.2.5 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 #1.

3.2.3 Adaptation

In ModEco-based economies, agents can evolve pricing strategies. However, in PMM #1 these adaptive features are, of necessity turned off. To date, all economies in which there is a price-value gap have collapsed and been unsustainable. Therefore, explicitly adaptive features are turned off, and the agents in PMM #1 have no adaptive traits.

However, the economy, as a whole, does seem to have adaptive capailities. For the parameter settings published herein, the economy has negative feedback mechanisms which enable it to remain 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 #1 has not yet been undertaken.

3.2.4 Agent objectives

When in stationary state, 50% of all agents die of bankruptcy, starvation or old age. Implicitly, the goal of each agent is to survive to reproduce. These objectives are implicit in the metabolic parameters described in section 3.3.3.2.

- To avoid bancruptcy the agent must maintain a minimum level of wealth. (See *W_CDT* and *F_CDT*.)
- To avoid starvation an agent must maintain a minimum level of supplies. (See *W_MPT* and *F_MPT*.)
- To reproduce an agent must be old enough and wealthy enough. (See *W_CRT*, *W_ART*, *F_CRT* and *F_ART*).
- To avoid death by old age an agent must reproduce before reaching old age (See *W_ADT* and *F_ADT*).

Implicitly, agents also have the objective of keeping some content in each and every store. If an agent has an opportinity to participate in a transaction, but has zero content in any store that is required as an input to that transaction, the agent cannot participate, but must sit out the opportunity. Those agents which have empty stores 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 stores well above zero levels are therefore fully able to participate in all commercial opportunties that come their way. These objectives are implicit in the "business factors" described in section 3.3.3.1.

- Frmrs use the *Parm:F_BRF* parameter to decide when to buy recycled MbMus;
- Frmrs use the *Parm:F_HWF* parameter to decide when to hire workers;
- Frmrs and Wrkrs use the *Parm:F_BSF* and *Parm:W_BSF* parameters 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 store, and keeps the waste flowing out. Second, if an agent engages in a commercial opportunity but has insufficient amounts of a resource to meet quota for that transaction, it applies for a grant of that kind of resource, and, on receipt of a grant, it increases its overwealth. Such an increase in wealth (a) reduces the probability of death; (b) increases the probability of being reproductive; and (c) increaes 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.

3.2.5 Learning

In a typical ModEco-based economy 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.

3.2.6 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 stores.

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 nature of prediction in PMM #1.

3.2.7 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.

3.2.8 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 *Wrkr*s move or agents reproduce, a new site is found by a randomized draw from a list of sites.

The only interactions between agents in a ModEco-based economy are commercial in nature. Agents do not mate for reproduction or for other metabolic or social purposes. Whenever agents interact, one agent is the active agent, and the other is passive. Interactions can be characterized as follows:

Active Agent	List Type	Description
MMgr	*ModEcoSys:WrkrList* *ModEcoSys:FrmrList*	Used to purchase waste.
MMgr	*ModEcosys:FrmrList*	Used to sell recycled mass (i.e. recycled waste).
Frmr	*Frmr:UnionList*	Used to make a job offer to a *Wrkr*.
Frmr	*Frmr:ConsumerList*	Used to sell inventory to a *Wrkr* or a *Frmr*. It includes itself on this list.
Frmr	*Frmr:SupplierList*	Used to remove itself from all consumer lists on the death of this *Frmr*.
Wrkr	*Wrkr:EmployerList*	Used to remove itself from all union lists on the move or death of this *Wrkr*.
Wrkr	*Wrkr:SupplierList*	Used to remove itself from all consumer lists on the move or death of this *Wrkr*.

All contact records are triple-linked. First, all contact records in ModEcoSys:ContactList are primarily linked in a loop, making searches within the large list possible. Next, for example, suppose a Frmr and a Wrkr are within each other's commuting areas. The records in *Frmr:UnionList* will form a linked list owned and managed by the *Frmr*. Similarly, the records in the *Wrkr:EmployerList* will form a linked list owned and managed by the *Wrkr*. These are virtual linked lists, and the links are different from the primary links. Finally, however, in *Frmr:UnionList* list there will be a contact record for the *Wrkr*, and in *Wrkr:EmployerList* there will be a contact record for the *Frmr*. These two records are directly linked to each other. This forms the third kind of link. When a *Wrkr* moves or dies or undergoes fission, the *Frmr* is advised to remove this record from its *Frmr:UnionList*. Or, when the *Frmr* dies or undergoes fission, the *Wrkr* is advised to remove this record from its *Wrkr:EmployerList*.

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.

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 resource levels to determine if it should enter into negotiations;
- A target agent is *Contact*ed;
- An offer is made (a sales pitch, or a job offer);
- The target agent determines 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 price is negotiated and, if agreement is reached, a deal is made (note that in PMM #1 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;

- All amounts for the transaction are set at quota, or pro-rated to the resource level of the lowest fractional resource;
- Resources are transferred between agents.

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

3.2.9 Stochasticity

ModEco is being developed using C++ as implemented in Microsoft's Visual Studio 2010. 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 tick. ModEco has a "Halt At" feature by which you can run a scenario for a pre-determined number of ticks (e.g. until steady-state is achieved), stop the run to turn on data collection, then restart. The built-in PRNG reseeds on halt, making such actions non-reproducible. The built-in PRNG was therefore replaced with the Mersenne Twister (citation needed).

PMM #1 will produce the same results for every run, if it is started with 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 (i.e. the Mersenne Twister). In a ModEco-based economy, the following types of action are randomized:

- When hiring *Wrkr*s, a *Frmr* will make job offers to *Wrkr*s on its union list,in random order.
- When moving, a *Wrkr* will make a list of available sites and select a random site.
- When selling inventory, a *Frmr* will make sales pitches to agents on its consumer list, in random order.
- When purchasing waste or selling recycled mass, the *MMgr* will deal with agents in random order.
- Since grants are provided as part of commercial transactions, and such transaction are available to all agents in random order, then grants are implicitly given to agents in random order.
- When generating price quotes, agents will introduce some randomness into each quote (not used in PMM #1).

Hypothesis: It is hereby hypothesized that, if a specific initialization of PMM #1 is sustainable with one seed, it is sustainable for all possible seeds. This hypothesis is largely untested, and, possibly, unproveable. 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.

3.2.10 Collectives

The obvious identification of collectives in PMM #1 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 the the other 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 3.1) In PMM #1 such geographic concentrations alter the macro-metabolic characteristics of the economy, causing higher rates of failure to reproduce. This is an emergent behavior and not part of the design intent.

3.2.11 Observations

A large number of data output options, for both real-time and later observations, are available for PMM #1. (See section 3.3.3.5.)

3.3 Other Details

The ODD protocol defers description of some of the less theoretical implementation details to the final section. Here we describe initialization of an economy, sources of input data, and sub-models.

3.3.1 Initialization

PMM #1 can be initialized in two scenarios called the basic model, and the crowded model. The settings of parameters and state variables which are unique to each scenario are given here. All other parameters and variables have the default values indicated elsewhere. Variable names and values are in corresponding order.

Entity:Variable	**Basic Model**	**Crowded Model**
ModEcoSys:Seed	1 (or your choice)	1 (or your choice)
Township:Height	6	15
Township:Width	10	20
ModEcoSys:NoOfWrkrs	16	320
ModEcoSys:NoOfFrmrs	4	80

The initializaion of each *Wrkr* and each *Frmr* for each type of scenario is given below.

For *Wrkr*s, we need to specify the following state variables: *Wrkr*:SN, *Age*, X, Y, ResNo, Cash, NoOfMbEus, NoOfSupplyMbMEus, NoOfWasteMbMus.

For *Frmr*s, we need to specify the following state variables: *Frmr*:SN, *Age*, X, Y, ResNo, Cash, NoOfRecycledMbMus, NoOfMbEus, NoOfInventoryMbMEus, NoOfSupplyMbMEus, NoOfWasteMbMus.

Some of these will be the same for all agents of the same type, and some will vary per agent.

3.3.1.1 Intialization values – basic scenario

For the basic scenario, these four variables are assigned uniquely for the 16 *Wrkr*s.

SN	X	Y	ResNo
3	4	2	0
4	4	2	1
5	4	2	2
6	4	2	3
7	4	3	0
8	4	3	1
9	4	3	2
10	4	3	3
11	5	2	0
12	5	2	1
13	5	2	2
14	5	2	3
15	5	3	0
16	5	3	1
17	5	3	2
18	5	3	3

In addition, *Wrkr*:*Age* is assigned stochastically. The 16 *Wrkr*s are selected in random order and assigned ages starting at 0 ticks and incrementing by 50 up to 750 ticks.

The other state variables are common for all *Wrkr*s.

Wrkr:Cash	1340
Wrkr:NoOfMbEus	134
Wrkr:NoOfSupplyMbMEus	134
Wrkr:NoOfWasteMbMus	134

For the basic scenario, these three variables are assigned uniquely for the *Frmr*s.

SN	X	Y
1	3	2
2	3	3
19	6	2
20	6	3

In addition, *Frmr*:*Age* is assigned stochastically. The 4 *Frmr*s are selected in random order and assigned ages starting at 0 ticks and incrementing by 200 up to 600 ticks.

The other state variables are common for all *Frmr*s.

Frmr:Cash	9600
Frmr:NoOfMbEus	480
Frmr:NoOfRecycledMbMus	480
Frmr:NoOfInventoryMbMEus	480
Frmr:NoOfSupplyMbMEus	480
Frmr:NoOfWasteMbMus	0

An explanation of the X and Y coordinate assignments will clarify them. 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 apears as a cluster of houses in a village with four outlying farms. This pattern is repeated in the crowded scenario.

In the RVP, every *Wrkr* has four farms within its commuting area, and every farm has two farms (including itself) within its commuting area.

3.3.1.3 Intialization values – crowded scenario

For the crowded scenario, the RVP (see section 3.3.1.2 immediately above) is used as a template for twenty small villages. If we start with the RVP and add a row of four empty *Lot*s at the bottom, we now have a rectangle of size three *Lot*s high by four *Lot*s wide. Now, add three empty *Lot*s to the right, forming a rectangle three *Lot*s high by five *Lot*s wide. We will use this expanded RVP as a template to form a township 15 *Lot*s high by twenty *Lot*s wide. The crowded scenario therefore has 20 "villages" in a pattern five villages high and four villages wide. The RVPs are all separated by an empty column or empty row. Also, the RVPs are snug against the left and top edges of the township, and have an empty column/row to the right/bottom of the township. Due to the wrapping features (toroidal shape) of the township, all villages are equitably situated.

These state variables are assigned to form this pattern: *Wrkr*:SN, X, Y, ResNo, *Frmr*:SN, X, Y.

Wrkr:SN is assigned in serial order, incremented by 1, as agents are placed.
Frmr:SN is assigned in serial order, incremented by 1, as agents are placed.

Frmr:*Age* is assigned stochastically. The 80 *Frmr*s are selected in random order and assigned ages starting at 0 ticks and incrementing by 2.5, rounded down, up to 797 ticks.

Wrkr:*Age* is assigned stochastically. The 320 *Wrkr*s are selected in random order and assigned ages starting at 0 ticks and incrementing by 10, up to 790 ticks.

The other state variables are common for all *Wrkr*s and *Frmr*s as indicated below:

Wrkr:Cash	840
Wrkr:NoOfMbEus	84
Wrkr:NoOfSupplyMbMEus	84
Wrkr:NoOfWasteMbMus	84
Frmr:Cash	4600
Frmr:NoOfMbEus	230
Frmr:NoOfRecycledMbMus	230
Frmr:NoOfInventoryMbMEus	230
Frmr:NoOfSupplyMbMEus	230
Frmr:NoOfWasteMbMus	0

3.3.2 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 #1.

3.3.3 Sub-Models

3.3.3.1 Business Factors and Quotas

So, even though the system is closed (so conserved quantities cannot leak out) and the existence of the *MMgr* and *EMgr* complete the needed cyclic loops through which the four conserved quantities can flow, the system still fails. It is possible for the system to go into a state of deadlock in which, for example, agents cannot work due to lack of energy, but cannot get energy due to lack of funds, and cannot get funds due to lack of work. This usually occurs because agents amass wealth in one asset class to the exclusion of others. We have devised two techniques to address this problem. These are sufficient to achieve sustainability, but the necessity of each of them is yet to be determined.

A business factor is a relative threshold applied to a set of stores (pools of intrinsic value) which are under the control of a single agent. For example, a *Frmr* should not buy up all waste every time an opportunity arises. If it does, it may have a huge pile of hoarded recycled mass, and no money to hire workers to convert it into inventory. When its inventory runs out, it will no longer be able to obtain cash, and it will starve, even though it is very wealthy. The "Buy Recycled Factor" (*Parm:F_BRF*) is a percentage indicator. If the value of its store of recycled mass is greater *Parm:F_BRF*% of its overall wealth, then it deems it has enough recycled mass and declines to participate in commercial opportunities until the store of recycled mass has been reduced. This approach works well for both wealthy and poor agents.

The business factors used in PMM #1, prefixed with *Parm:* in the pseudocode, are as follows:

Name	Brief Description	Default Value
F_BRF	"Buy Recycled Factor" – if the value of a *Frmr*'s pool of recycled mass is less than *F_BRF*% of the agent's total wealth, then it will buy recycled mass when the opportunity presents.	2.5
F_HWF	"Hire Wrkr Factor" – if the value of a *Frmr*'s pool of inventory is less than *F_HWF*% of the agent's total wealth, then the *Frmr* will hire a *Wrkr* when the opportunity presents.	25
F_BSF *W_BSF*	"Buy Supplies Factor" – if the value of an agent's (*Frmr*'s or *Wrkr*'s) pool of supplies is less than *F_BSF*% (or *W_BSF*%) of the agent's total wealth, then the agent will buy supplies when the opportunity presents.	25

These four numbers are exogenous values which can be set under user control.

Business factors alone do not seem to be sufficient to avoid excessive pooling and collapse. Because they are relative thresholds, a very wealthy agent may sop up all available resources in each transaction, leaving nothing for the rest. This kind of concentration of wealth in the hands of a very few agents seems to lead to collapse.

A quota is an absolute threshold setting a maximum or minimum amount of resource. There are fundamentally two different kinds of quota that can be applied:

- Pool quota - a quota on the amount of resource that can be held in a single pool of resource (an agent's asset class).
- Process quota - a quota on the amount of resource transferred in a commercial transaction, or converted in a conversion process.

Pool quotas were the subject of experimentation prior to the implementation of business factors. They did not work well at all, and were decommissioned in ModEco. Further research should be done to investigate their role in increasing or reducing resiliency and sustainability. For example, they might play a useful role if the total wealth of an agent is set by an absolute threshold. This was tried, but the perceived problem with this quota was the inability of the most wealthy agents to play a role in the economy, effectively sequestering those held resources and making the economy smaller for the poor agents. The absolute pool quotas were replaced by the relative business factors.

On the other hand, process quotas work surprisingly well. A process quota can be specific to a type of agent (*Frmr* or *Wrkr*), a type of transaction (buy/sell, hire, or consume), or a type of resource (mass, energy, cash). These quotas, together with the metabolic parameters described below, significantly shape the nature of the economy produced. The process quotas used in PMM #1 are as follows:

The following process quotas, prefixed as *Quota:* in the pseudocode, are exogenous variables and are user selectable.

Process Name	**Process Quota Name**	**Type of Agent**	**Type of Resource**	**Value**
Hire *Wrkr*	*DailyWorkRate*	*Wrkr* and *Frmr*	MbEus	20 MbEus
	When a *Frmr* hires a *Wrkr*, both must contribute equal portions of energy to produce inventory. Each can contribute a maximum of 20 MbEus, for a total of 40 MbEus per hire, per tick. This maximum is achieved only if the *Frmr* also has 40 MbMus of recycled mass, and $20x8 = $160 to pay for the *Wrkr*'s MbEus. If either agent has 0 units of a required resource, they cannot make a deal. But, if both agents have a small amount of each required resource (and so are "deserving"), but insufficient to make quota or match the contribution of the other agent (and so are "needy"), they can apply for a grant from the *EMgr*.			
Sell Inventory	*MaxInvMbMEusSold*	*Wrkr* and *Frmr*	MbMEus	40 MbMEus
	When a *Frmr* sells inventory to a consumer (*Wrkr* or *Frmr*) the maximum that can be sold is 40 MbMEus, valued at $400. If the buyer has some cash to initiate the transaction, but insufficient to pay for the amount of MbMEus offered for sale, it can apply to the *EMgr* for a CashGrant.			
Consume Supplies	*W_MPT*	*Wrkr*	MbMEus	4 MbMEus
	F_MPT	*Frmr*	MbMEus	16 MbMEus
	MPT stands for "Material Per Tick" and is described under "Metabolic Parameters.			
Sell Waste	MaxWsteMbMusSold	*Wrkr* or *Frmr*	MbMus	Unlimited MbMus
	When a consumer (*Wrkr* or *Frmr*) buys recycled mass from the *MMgr* the maximum that can be purchased is unlimited.			
Buy Recycled	*MaxRecMbMusSold*	*Wrkr* and *Frmr*	MbMus	40 MbMus
	When a *Frmr* buys recycled mass from the *MMgr* the maximum that can be purchased is 40 MbMus, valued at $80. If the buyer has some cash to initiate the transaction, but insufficient to pay for the amount of MbMus offered for sale, it can apply to the *EMgr* for a CashGrant.			

Mass tends to pool in the agent's "waste material" stores. To eliminate this problem we (a) allowed the *MMgr* to go into debt to purchase surplus MbMus, and (b) removed the former quota on number of MbMus that could be sold to the *MMgr* per tick. These two actions were sufficient to achieve sustainability. Perhaps only one is necessary.

3.3.3.2 Metabolic Parameters

A ModEco-based economy is an amalgamation of two complex adaptive systems, welded together to form one co-developing system. The business factors and quota described above can be used to shape and control the economic system. The metabolic parameters are used to control the ecosystem in which the agents exist, and to provide the link between the two systems.

These metabolic parameters are not original to ModEco, but were derived from the work of Dr Michael Palmiter (citation).

Metabolic Parameters for *Wrkr*s:

Parm Name	Default Value	Brief Description
W_ART	800 ticks	"Age Reproductive Threshold" – a *Wrkr* cannot reproduce prior to this age.
W_ADT	1600 ticks	"Age Death Threshold" – a *Wrkr* cannot live beyond this age.
W_CRT	$1000	"Cash Reproductive Threshold" – a *Wrkr* must have this level of wealth to reproduce. This links the economic system to the metabolic system.
W_CMT	$300	"Cash Move Threshold" – A *Wrkr* having less than this level of wealth, and finding itself umemployed in the recent hiring round, will move to a randomly selected site within its commuting in the hopes of finding employment in the next tick.
W_CDT	$20	"Cash Death Threshold" – a *Wrkr* with less wealth will die of bankruptcy. This links the economic system to the metabolic system.
W_MPT	4 MbMEus/tick	"Material Per Tick" – A *Wrkr* must consume precisely this amount of food each tick to live; no more, and no less. Failure to consume this amount results in death by starvation.

Metabolic Parameters for *Frmr*s

Parm Name	Default Value	Brief Description
F_ART	800 ticks	"Age Reproductive Threshold" – a *Frmr* cannot reproduce prior to this age.
F_ADT	1600 ticks	"Age Death Threshold" – a *Frmr* cannot live beyond this age.
F_CRT	$1000	"Cash Reproductive Threshold" – a *Frmr* must have this level of wealth to reproduce. This links the economic system to the metabolic system.
F_CDT	$20	"Cash Death Threshold" – a *Frmr* with less wealth will die of bankruptcy. This links the economic system to the metabolic system.
F_MPT	16 MbMEus/tick	"Material Per Tick" – A *Frmr* must consume precisely this amount of food each tick to live; no more, and no less. Failure to consume this amount results in death by starvation.

3.3.3.3 Municipal Grants

Grant Code Name	Type of Transac-tion	Type of Agent	Type of Resource	Value (0 to quota)	Brief Description
F_EnergyGrants	Hire *Wrkr*	*Frmr*	MbEus	0-20	A *Frmr* having some energy, but insufficient to match the input of a hired *Wrkr*, can apply for a grant of energy from the *EMgr*.
F_MassGrants	Hire *Wrkr*	*Frmr*	MbMus	0-40	A *Frmr* having some recycled mass, but insufficient to meet the harvest quota, can apply for a grant of recycled mass from the *EMgr*.

Grant Code Name	Type of Transac-tion	Type of Agent	Type of Resource	Value (0 to quota)	Brief Description
*W*_EnergyGrants	Hire *Wrkr*	*Wrkr*	MbEus	0-40	A *Wrkr* having some energy, but insufficient to match the input of a farmer (*Frmr*), can apply for a grant of energy from the *EMgr*.
CashGrants	Buy Supplies	*Wrkr* or *Frmr*	Cash	0-40	An agent having insufficient cash to pay for a full quota can apply for a grant of cash from the *EMgr*.

PMM #1 requires that these grant mechanisms be turned on in order to complete the cash and resource cycles.

3.3.3.4 Additional Relevant Toggles

Parameters have an effect on a model when a value is changed. You might also want to simply remove all logic and effects associated with a parameter from the model. A toggle is a switch that turns a feature or capability on or off. Usually it is presented as radio buttons (a pair of little circles with a dot in the middle) in a dialogue, or as a toolbar button that can be depressed or released, or as a set of mutually exclusive menu items. Those toggles which are active active during a run of PMM #1 are shown in bold. They are not a necessary part of the model, but are useful to experiment with the effects of the feature under control.

Toggle Name	Brief Description
B_ Economic Engine (Cyclic Engine)	ModEco has two different economic engines. A **cyclic** engine (having a cyclic energy flow), and a linear engine (having a linear energy flow). PMM #1, which is the topic of this paper, uses the **cyclic** engine. The engine type is selectable by menu.
Model Type	PMM #1 comes in two scenarios: Basic and Crowded. The Scenario type is selectable by menu.
***F_BRF* Switch** ***F_HWF* Switch** ***F_BSF* Switch** ***W_BSF* Switch**	ModEco allows the user to turn the business factors on or off via radio buttons in the Initialization Wizard (IWiz) and Economic Wizard (EWiz) dialogues.
***F*_EnergyGrantSwitch** ***F*_MassGrantSwitch** ***W*_EnergyGrantSwitch** **CashGrantSwitch**	ModEco allows the user to turn the four municipal grants programs on or off by radio buttons located in the IWiz and EWiz dialogues.
***MMgr*QuEasingSwitch**	ModEco allows the user to control whether the *MMgr* can go into debt to purchase surplus waste. This is controlled via a pair of radio buttons in the IWiz and EWiz dialogues.

PMM #1 is sustainable if (a) the cyclic engine is selected; (b) the crowded model is chosen; (c) all business factors are operational; (d) all grant programs are enabled; and (e) quantative easing is enabled. These settings represent sufficiency for sustainable operation. Testing of necessary toggles for sustainability has not yet been completed.

3.3.3.5 Data Collection and Display

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.

Real-time display is enabled for:
- a view of the current values of the state variables for all agents.
- a variety of tabulations of aggregations of mass, energy, intrinsic value and cash, by economic sector.
- a histogram showing the current distribution of wealth across agents, by sector.
- histograms showing the current distribution of wealth among asset classes for each individual agent, and by sector.
- line graphs showing a small selected number of macro-economic indicators such as employment rates, velocity of cash, and relative sector sizes.
- counts of events, transactions, and agents, with derived success rates for “deals” made.

Optionally, data can be collected in “comma-separated value” (CSV) files which can be imported into MS Excel. Up to 1,048,573 records can be collected for each type of data. This is the limit that Excel 2010 can handle. The following types of data can be collected:
- For each transaction type, for each transaction, a record of input values and results. This includes hires, sales of inventory, consumption of supplies, purchase of recycled mass, sales of waste, agent births, agent deaths, grants applied for and received.
- For each generation, aggregate data can be collected on a generation-by-generation basis (i.e. every 800 ticks, since a generation is determined by F_ART and/or W_ART).

References

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.

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

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

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

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.

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.

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

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

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

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