Description of A Simple Agent-Based Capital Exchange Model
Entropic Index Laboratory (EiLab) – Model I
Using the ODD Protocol

Author: Garvin H. Boyle
Date: 21 January 2014

Description of A Simple Agent-Based Capital Exchange Model Entropic Index Study EiLab – Model I Using the ODD Protocol

Table of Contents

Description of A Simple Agent-Based Capital Exchange Model Entropic Index Laboratory (EiLab) – Model I Using the ODD Protocol

Abstract

EiLab is a software application in which a user can configure and run capital exchange models. Models A through H are implementations of the models of Drăgulescu and Yakovenko (2000), but Model I is specially designed to demonstrate the simple bounded-wealth model described in the paper entitled "A Definition and Examination of an Entropic Index for Agent-Based Models". In that paper, Model I of the EiLab application is described informally, the entropic index of a histogram is defined, and the implications of the use of entropic index with Model I is examined in some detail. Finally, the predicted behaviour of the model, based on an analysis of possible state transitions, is compared with the actual behaviour of the model and the agreement is found to be extremely close. This paper, written in parallel, is a formal description of "Model I", as used in that study of its entropic index.

Keywords

Agent-Based Model, Entropic Index, Entropy, Maximum Entropy Principle, Model description, Model replication, ODD Protocol, Second Law of Thermodynamics

1 Introduction

This paper will use the ODD protocol [1, 2] to describe a capital exchange model that can be accessed as "EStudy – Model I" in the EiLab application. [3] The full name of this application is the "Entropic Index Laboratory for Agent-Based Models". The EiLab application was written to demonstrate a variety of agent-based capital exchange models as described by Drăgulescu and Yakovenko. [4] As physical scientists, they drew on knowledge of energy distributions in gas models to examine and explain wealth distributions in economic capital exchange models. Yakovenko later posted to his website a demonstration of one of his capital exchange models in which entropy was calculated as the model approached an equilibrium distribution. This was both expected and surprising at the same time. Of course, entropy rises to a maximum as the system approaches an equilibrium distribution. But, what kind of entropy can possibly exist within a capital exchange model?

Several circumstances inspired the design of Model I with the intent to produce an extremely simple agent-based capital exchange model:

- Entropy is a concept that is poorly understood by most people, and even generally misunderstood. In a recent CBC radio show it was explained that Boltzmann, when he made his original public arguments about thermodynamic entropy, had tried to use the distinction between order and disorder to interpret the implications of thermodynamic entropy. Many years later, so the story goes, he retracted that explanation due to the confusion it caused. Unfortunately, for over 100 years, university texts and other popular science books have all explained entropy in terms of order and disorder. According to a presentation at a recent Biophysical Economics conference in Vermont, that explanation is now being removed from university texts, but the strong link in people's minds between the concept of entropy, and the concept of disorder continues to confuse our understanding.
- In spite of this ongoing confusion, there is an increasingly strong belief that the production of thermodynamic entropy drives all physical dynamic systems through which energy is flowing. In

addition to the propensity of energy to move and spread through conduction, convection, radiation and transformation from one form to another as it moves through a system is also a propensity to produce entropy. Energy is never created or destroyed, but the ability of energy to do 'useful' work is destroyed, and that decline in the ability of the energy to do work is marked by an increase in thermodynamic entropy. As the ability to do work degrades, thermodynamic entropy rises. In a world in which energy flows, energy uses, and energy conservation are so important, an understanding of the role of entropy is becoming exceedingly important.

- And even more so, in a world in which it is becoming increasingly clear that we people, as a species, need to learn to live sustainably, and in which the flow of energy from fossil fuels has enabled us to increase our numbers well above the carrying capacity of the Earth on which we live, and in which pollution from our consumption of fossil fuels is destroying our life-supporting environment, a much deeper understanding of thermodynamic entropy production would seem to be a key piece of knowledge required for our survival through the coming years.
- The tell-tale signs of thermodynamic entropy production (e.g. the migration of a system towards equilibrium states, accompanied by signature distributions of energy) are now being found in systems which are not fundamentally thermodynamic in nature, but are dominated by data structures and data transfers such as communications or economic systems. This would lead one to believe that entropy is not just a thermodynamic phenomenon, but, rather, it is a fundamental statistical process which is manifested in these many and varied types of systems.
- Finally, the fact that an analogy of entropy was apparently easily demonstrated in such simple agent-based models (ABMs) as those designed and described by Drăgulescu and Yakovenko makes it clear that entropy is a much broader concept than is currently captured in the words "thermodynamic entropy".

So, Model I was added to the EiLab suite of capital exchange models for the sole purpose of providing a simple ABM in which the process of entropy production could be studied in detail. [3] The analysis of entropy production in this model does not require new mathematical formulae. The formulae of Boltzmann, Gibbs and Shannon were modified and applied in their discrete form, and as such can be understood by high school students. In fact the goal of the analysis is to be consistent with previous concepts of entropy, but to generalize the concept in such a way that reference to thermodynamic entropy or Shannon entropy (i.e. informational entropy) are not required in order to understand the production of entropy within this ABM.

I distinguish at least four somewhat different types of entropy, based on the type of system that produces it:

- Thermodynamic entropy – that is produced by dynamic physical systems as energy is used, and shapes the distribution of energy in the atoms of a gas;
- Shannon entropy – that describes the new information content in a message;
- Economic entropy – that is produced by major economies as capital is used, and shapes the distribution of wealth amongst the economic players; and
- ABM entropy – that is produced in an extremely simple capital exchange model, and shapes the distribution of wealth amongst the agents in the ABM.

Reductionism is the word used to describe the tendency of scientists to abstract away the irrelevant and minor differences between varied systems in order to find a general and simple description of our complex reality. Reductionism has been a powerful tool in the development of modern science. However, I think it would be a mistake to say that entropy as demonstrated by Yakovenko in his capital exchange models is "just a model of thermodynamic entropy" (which it isn't) or "just an instance of

Shannon entropy". I would argue the other way, that Dr Yakovenko has demonstrated a different kind of entropy that I am calling ABM entropy, and I would argue further that thermodynamic entropy, Shannon entropy, economic entropy and ABM entropy are all manifestations of an elemental time-asymmetric stochastic mechanism. I believe that entropy can be defined for a simple histogram, and that this is the most fundamental version of entropy. All others (thermodynamic entropy, Shannon entropy, and economic entropy (which, admittedly, is not yet fully recognized by economists as a valid manifestation of entropy) are just more complicated manifestations of this mathematical phenomenon from which the physical irrelevancies have not been fully abstracted. My concept of ABM entropy is a direct application of entropy as defined on a histogram.

With such a shift from thermodynamic or informational or economic considerations to mathematical considerations, the nature and role of entropy can be more easily studied, and the potential for its applicability to a much wider range of dynamic phenomena opens up.

So, for agent-based models we do not really need a definition of a new kind of entropy that might be called ABM entropy. Rather, what we really need is a shift in perspective in which we view entropy as a fundamental mathematical phenomenon. With such a shift, this author believes that such varied processes as Darwin's natural selection, Smith's invisible hand, Boltzmann's thermodynamic entropy, Shannon's information entropy, and the emerging concept of economic entropy will all be seen to rise from the same source. If this is true, then a deeper understanding of that source would be a wise thing to seek.

Model I of the EiLab application is an ABM designed to support an initial study of an entropic index, as defined in the accompanying papers. [5, 6, 7] Model I is a modification of one of the capital exchange models described by Drăgulescu and Yakovenko. In their models, wealth has a lower bound, but no upper bound. The initial study of an entropic index in such an unbounded domain was problematic, so, to simplify the initial study, an upper bound was added. Ultimately, the arbitrary imposition of an upper bound will have to be removed, but, for now, it is there. The number of possible states in the state space of the model rises almost exponentially (it's a combination of factorial functions) as the extent of the domain is increased, and as the number of agents is increased, so, again to keep the analysis tractable for an initial study, the parameters have a very limited extent. And, again, to keep the mathematics simple, only discrete values are allowed for all variables. A more detailed study of an entropic index in ABMs would require some or all of these restrictions to be addressed.

2 The ODD Protocol

2.1 History of the ODD protocol

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

Early agent-based models (ABMs) were described in an ad hoc fashion. The ODD protocol was recently proposed as a means to encourage organized, complete and replicable descriptions. [1] The proposed protocol was well received, and a number of ABMs 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. [2]

2.2 Use of the ODD protocol

This paper follows the revised description of 2010. The ODD protocol was designed specifically for the description of ABMs, and, for the most part, it was easy to complete the ODD template. However, the model described below has two qualities that made use of the protocol difficult.

First, EiLab is a generalized application which demonstrates a variety of capital exchange models, and it has a large parameter space, and the model which is described in this paper uses a relatively small subset of the possible parameters. There is therefore some complexity in the software that has not been included in this description.

Second, EiLab 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 ABMs, 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 OOP objects found in EiLab.

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

3 The “Entropic Index” Study (Model I)

3.1 Overview

EiLab V1 (Alpha) has nine models labelled A through J. The first eight models labelled A through H are implementations of most of the models described by Drăgulescu and Yakovenko at [4]. The ninth and tenth models, Models I and J, were added for the purpose of supporting an initial study of an entropic index in ABMs at [5, 6, 7]. Model I is a closed economic system. Model J is an open economic system. To access Model I:

- Start the software, and click through the splash screen and welcome message.
- Left-click once on the IWiz toolbar button. IWiz stands for Initiation Wizard. A dialogue will open.
- Left-click on the tab in the Wizard dialogue labelled “Model I”.
- The software will automatically set the default parameter values for Model I.
- In the IWiz dialogue you will notice three paramters than control the model: K (number of bins), A (number of agents), and W (total wealth). Alter the parameters if you wish. For a first time, leave them as is. Its not simple to find good parms, but, as a rule, these all work: choose a K, set A = qK where q is an integer; W = A(K+1)/2.
- Left-click on the OK button, and the IWiz dialogue will close as the model is initialized with the correct parameters.

To execute a run of the model:

- Left-click on the Reset toolbar button to have access to the PRNG seed.
- Choose a seeding method, and a seed, and then click on OK.
- Left-click on the Go toolbar button.
- To control the display options, use the “T/M”, L, R, S and N toolbar buttons.

In this model agents possessing capital meet and exchange that capital in a highly idealized economy that is distantly analogous to the exchange of energy between atoms in an idealized gas. This type of

economy is therefore referred to as a "gas model" in the Econophysics literature. A histogram of the distribution of agents, categorized by amount of capital each agent controls, is constructed after each capital exchange, and the entropic index for ABMs, as defined at [5], is calculated, and recorded. The equation used is:

$$S(h) \equiv \left(\frac{1}{\ln(K)}\right) \sum_{i=1}^{K} \left(\frac{a_i}{A}\right) \ln\left(\frac{A}{a_i}\right) \qquad \textbf{Equ (1)}$$

where:

- S(h) is defined as the 'entropic index' of h;
- h is an ordered K-tuple having i coordinates a_i, representing a histogram having K bins in which the order of the bins is integral to the meaning of the histogram; and
- A is the sum of all of the a_i.

There are a small variety of ways to watch the development of the model in real time. A time series of values of entropic index is available, as is a histogram of recent values of entropic index. Also textual descriptions of components of the model can be presented. Finally, a bar graph showing the probability distribution of states visited by the model can be displayed.

3.1.1 Purpose

The purpose of Model I is to enable and support an initial study of the concept of entropic index in closed capital exchange ABMs, as described at [5, 6, 7], without reference to other types of entropy.

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
• *EiLabSys*	3.1.2.1	The capital exchange system
• The capital exchange agents	3.1.2.2	Agents
• The temporal scale	3.1.2.3	Relative meanings
• Non-participating entities	3.1.2.4	Supporting entities that play no role in the capital exchange activities
◦ Agent list	3.1.2.4.1	Fixed allocations of memory for agents
◦ Wealth database	3.1.2.4.2	Capture and display of historic data
◦ Entropy database	3.1.2.4.3	Capture and display of historic data
◦ Configuration database	3.1.2.4.4	Capture and display of historic data

3.1.2.1 EiLabSys

EiLabSys represents the capital exchange system. It contains within itself all of the other entities of interest in the model, and manages the rules of capital exchange. In addition, in the EiLab implementation, *EiLabSys* contains a number of variables that disable features not required for Model I. These are not relevant to replication of Model I, and so, are not described here. Finally, there are a number of variables that control entities that play a supporting role for Model I, but which have no material effect on the outcome of a run. These non-participating entities are discussed in section 3.1.2.4 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 may be used by many entities. The age of the system can be written as *EiLabSys*:*Age*.

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

Name	Type	Brief Description
Age	Variable	The count of time increments since inception.
A or *NoOfAgents*	Variable	The count of *Agents* in the model.
K or *NoOfBins*	Variable	The count of *Bins* or classifications in the histogram used to calculate entropic index. Also, the number of discrete levels of personal wealth available to the agents in the model.
W or *TotalWealth*	Variable	The sum of the capital of all *Agents* in the model.
AgentList	Entity	A non-participating entity that contains *Agents* (see section 3.1.2.4.1).

The variables K, A and W each go by two names. The shorter single-letter names are used extensively in the analysis of the model at [5,6, 7]. The longer names are used extensively in the C++ code at [3].

3.1.2.2 The capital exchange agents

Within Model I there is one type of agent, each of which has a single characteristic, and that is an amount of wealth or capital. These agents are immortal, in that they never die and they are never removed from the model at any step. The total number of agents in the model is conserved in every transaction and at the system level throughout each tick of the model.

Each agent has capital (money) ranging from \$1 to \$*K* in discrete whole numbers. The total capital of all agents in the model is \$*W*.

Agent Name	Description	Geographic characteristics	Temporal characteristics	Breadth of Knowledge (geographic and temporal)
Agent	Holds and exchanges capital when able.	None	Immortal	None

Agents are characterized by the following state variables:

Name	Type	Brief Description
Capital	Variable	The amount of money, in discrete dollars. Synonyms for capital are wealth, cash, or money.

3.1.2.6 Temporal scale

With respect to the temporal scale, time in EiLab is measured in discrete time units called ticks. All activities (outlined in section 3.1.3) are executed once during each tick. A tick is the amount of time it takes for two economic agents to meet, make a deal, and exchange cash. It could represent minutes or months, but the time scale is not in any way relevant to the running or interpretation of the model results.

3.1.2.4 Non-Participating entities

EiLab 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 (ADE), 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 EiLab, many such objects exist which are not part of the design of the model, but exist due to the goals of the programmer to provide real-time data display, 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 useful for complete understanding of the pseudocode provided in section 3.1.3. Furthermore, any attempt to replicate Model I will require some attention to similar functions.

3.1.2.4.1 Agent list

The system contains a pre-allocated list of *Agents*. This is part of the *EiLabSys* entity as *EiLabSys:AgentList*. The list is of fixed size and location in memory. The *Agents* are unlinked and are in no particular order in the list. All are identical, except for the amount of capital they hold.

3.1.2.4.2 Wealth database

The computation of entropic index, while integral to the study and purpose of the model, is not, in fact, a functioning part of the model. For example, if no entropic index was ever calculated, it would not affect the trajectory of the model within its state space. Nevertheless, in order to calculate entropic index of the model during any tick to track its rise, it is necessary to sort the agents into the bins of a histogram, and then apply the formula for entropic index against that histogram. The wealth database is the entity within which the wealth histogram resides, and it is into that wealth histogram that the classified agents are sorted in each tick. An EiLab user can see the wealth histogram changing over time in the main panel. Under some conditions, the main panel can be minimized or maximized using the "T/M" toolbar button.

3.1.2.4.3 Entropy database

When the entropic index is calculated using the wealth histogram embedded in the wealth database, that current index is added to a time series of such indices. This time series contains a historic record of past indices, and is located in the entropy database. For display purposes, in fact, a collection of three such series are maintained. The most recent is graphed in magenta, the older two are graphed in light blue. An EiLab user can see an extensive history of the changing entropic index in graphic form. These time series are displayed in the "Notes" panel at the bottom of the screen, and it can be minimized or maximized using the "N" toolbar button.

In addition, the entropy data base also contains a "brief look" entropy histogram in which the most recent data points (up to 3000 values of the entropic index) are sorted, again for each tick, and a "long look" entropy histogram in which entropy data for extended runs can be collected and presented. These histograms enable an EiLab user to see how the entropic index is asymptotic to a maximal value, and then is subject to random perturbations as time passes, producing a distribution about a mean value. The nature of that distribution can be examined. These histograms are displayed in the "Left" panel of the screen, and can be minimized or maximized using the "L" toolbar button.

3.1.2.4.4 Configuration database

Due to the fact that the key variable '***wealth'*** is restricted to discrete whole numbers, the state space of Model I is discrete and finite in size. As the model moves from state to state through its state space, a record of the number of times each state is visited is maintained in yet another histogram located in the configuration database, and accessible via the "L" toolbar button. This shows the user the probability that the model is in a specific configuration, based on past behaviour. Data can be collected over extended runs, and the actual results were found to agree to a remarkable degree with the predictions developed at [5]. An EiLab user is able to watch the signature distribution develop as the model progresses.

3.1.3 Process Overview and Scheduling

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

- ***Setup*** – in which lists and display indicators are reset, if needed;
- ***Exchange of capital*** – in which a pair of *Agents* exchange capital.

- ***Cleanup*** – in which distributions of wealth are tabulated, the entropic index is computed, and databases are updated for display, visual displays are updated, and termination or pause conditions are checked.

An EiLab user will notice two additional steps listed in the monitor display when running model I: Insertion and Extraction. These were added to support a follow-on study of open ABMs in Model J (whereas Model I is closed with respect to wealth) and do not affect a run of Model I.

The following sections describe each of these functions in detail. Please note, however, that because Model I exists in the context of a larger application, EiLab, that can collect and display data from many different capital exchange models, there is a certain amount of scaffolding code that is shown below that is not a needed part of Model I. Also, there is some associated code used to collect and display data about this model, but which is not part of this model. In the code that follows, black code is generic scaffolding, green code is the code distinctive to Model I, and red code is unexplained code used by the EiLab application to collect and display data. That collection and display process is not part of the ODD description, and is not explained in this document.

3.1.3.1 Setup

This is an almost empty function in which the nag screen may be activated, the flag for suppression of activities mid-tick is managed, and the speed governor is adjusted. These activities have no relevance to the model itself, but manage the framework of the application in which each tick of the model executes. In C++ code, this process looks like this:

```
long AEiLabSys::DoSetup( long CurrentFunction )
{
    // Declarations
    SNagScreen    NagScreen;

    ASSERT( CurrentFunction >= 0 );
    ASSERT( CurrentFunction < _EL_Fn_NoOfFns );

    // Execute only if required.
    if( CurrentFunction != _EL_Fn_Setup )
        return CurrentFunction;

    if( ReleaseType == _EL_ReleaseType_Demo )
    {
        if( ( Age.GetHours()   == 0 ) &&
            ( Age.GetMinutes() == 0 ) &&
            ( Age.GetSeconds() == 0 ) )
        {
            NagScreen.ReleaseType = ReleaseType;
            NagScreen.DoModal();
        }
    } // end if ReleaseType_Demo

    // Note that a tick has just started.  No processing has yet been done.
    // Access to certain modification techniques must be restricted if
    //   processing of a tick has started but not completed.
    TogglesPtr->BetweenTicks = _EL_BetweenTicks_False;

    // First, read the clock and note when this EiLab tick started.
    Governor.SetStartTick();

    CurrentFunction++;
    return CurrentFunction;
```

```
}
```

The variables that start with the letters "_EL_" are compiler switches set at compile time and unchanging during execution. The code in bold red is not part of Model I, but is used in the EiLab application to collect and/or display real-time data.

3.1.3.2 Exchange of capital

This is the primary function in the execution of Model I. The relevant C++ code for this function is as follows:

```
long AEiLabSys::DoCapitalExchange( long CurrentFunction )
{
    // Declarations

    ASSERT( CurrentFunction >= 0 );
    ASSERT( CurrentFunction < _EL_Fn_NoOfFns );

    // Execute only if required.
    if( CurrentFunction != _EL_Fn_CapitalExchange )
        return CurrentFunction;

    switch( TogglesPtr->ModelNo )
    {
    default:
    // case _EL_Model_A:
        // DoCapitalExchange_A();
        // break;
//    Many similar non-relevant cases elided.
    case _EL_Model_I:
        DoCapitalExchange_I();
        break;
    }

    CurrentFunction++;
    return CurrentFunction;
}
```

The variables that start with the letters "_EL_" are compiler switches set at compile time and unchanging during execution. The object "Toggles" is an instance of a class used to store control variables that are accessible throughout the EiLab application. Here, the model number that is currently activated is stored, and used to switch to the code that executes Model I. The call to DoCapitalExchange_I() executes the following C++ code:

```
long AEiLabSys::DoCapitalExchange_I( void )
{
        // Declarations
        long            NoOfAgents;
        long            AgentNo;
        AAgent*         AgentPtr = NULL;
        AAgent*         WinnerAgentPtr = NULL;
        AAgent*         LoserAgentPtr = NULL;
        ASlotNoList     AgentSlotNoList;
        double          DeltaCapital;
        BEStudyConfig   NewEStudyConfig;

        // Establish a random order for processing the Agents.
        // Ensure the list is empty, first.
        AgentSlotNoList.InitASlotNoList();
        AgentSlotNoList.RsPtr = RsPtr;
        NoOfAgents = AgentList.NoOfAgents;
        for( AgentNo = 0; AgentNo < NoOfAgents; AgentNo++ )
        {
```

```
                AgentPtr = &AgentList.Agents[ AgentNo ];
                AgentSlotNoList.AddSlotNo( AgentPtr->AgentSlotNo );
        }

        // Process two Agents.
        // Select two random agents from those not yet processed.
        // The SlotNoList deletes each agent's SlotNo when used once.
        WinnerAgentPtr = AgentList.GetAgentPtr( AgentSlotNoList.GetRandomSlotNo() );
        LoserAgentPtr  = AgentList.GetAgentPtr( AgentSlotNoList.GetRandomSlotNo() );

        // Compute delta c (amount of capital to change hands).
        DeltaCapital = 1;

        if( ( WinnerAgentPtr->Capital < TogglesPtr->Scem_EStudy_ModelI_Nobim ) &&
                ( LoserAgentPtr->Capital > 1 ) )
        {
                LoserAgentPtr->Capital -= DeltaCapital;
                WinnerAgentPtr->Capital += DeltaCapital;
                EStudyDataBase.AllowedTransitions++;
                EStudyDataBase.SortAgentsIntoPassiveEStudyHistogram( TodaysDate(), &AgentList );
                if( TogglesPtr->Model_X == _EL_Model_I )
                {
                        //NewEStudyConfig.InitBEStudyConfig( TogglesPtr );   // Redundant
                        ExtractModelIConfig( &NewEStudyConfig );
                        Model_I_ConfigList.IncCountByHash( &NewEStudyConfig );
                }
        }
        else
        {
                EStudyDataBase.DisallowedTransitions++;
        }
        EStudyDataBase.AttemptedTransitions++;
        ASSERT( EStudyDataBase.AttemptedTransitions ==
                ( EStudyDataBase.AllowedTransitions +
                  EStudyDataBase.DisallowedTransitions ) );


        return TRUE;
}
```

The code in red is not part of Model I, but is used in the EiLab application to collect and display real-time data. Possibly similar but different modules would be used in a different implementation. The "AgentSlotNoList" object is used to hold the placement indices (or slot numbers) of all of the agents. A slot number is randomly selected and retrieved, and that is used to reach into the AgentList[] array and pull out the memory address of the associated agent.

When a capital exchange is completed, either successfully or unsuccessfully, the two agents are implicitly returned to the pool of agents, and they are free to participate in the next exchange, if selected. Previous activity has no impact on the probability of selection of an agent during each tick.

3.1.3.3 Cleanup

The Cleanup() function is needed for the EiLab application, and is used to tie up administrative activities started in the Setup() function, but is only barely relevant to the functioning of the model. Data collection and display code is in red text. The C++ code for the DoCleanup() function follows:

```
long AEiLabSys::DoCleanup( long CurrentFunction )
{
     // Declarations

     ASSERT( CurrentFunction >= 0 );
     ASSERT( CurrentFunction < _EL_Fn_NoOfFns );
     // Execute only if required.
     if( CurrentFunction != _EL_Fn_Cleanup )
          return CurrentFunction;
```

```
    // The tick is complete. Age and display.
    Age.AgeByOneSecond();

    // The EntropyDataBase is used for the entropic index displays.
    // Process the new data, produce histogram data.
    if( ( TodaysDate() % EntropyDataBase.TicksBetweenDataPts ) == 0 )
    {
        // The wealth histograms in the WStudyDataBase use varying numbers of
        //   bins in an attempt to maximize the use of the display area.  However,
        //   to compute entropic index consistently, you need a consistent
        //   number of bins.
        // Sort the wealth data into bins for computation of entropic index
        //   and store in the histogram in the wrapper.
        EntropyDataBase.EdbWealthHistWrapper.SortAgentsIntoEdbWealthHistogram( &AgentList );
        // Compute the entropic index using the above histogram, store it in
        //   the array of time series (for the notes panel).
        EntropyDataBase.ComputeEntropy( TodaysDate() );    // For notes panel.

        if( TogglesPtr->DisplayLeftPanel == _EL_PanelDisplayMode_Open )
        {
            // To reduce overhead per tick, only create the histograms
            //   when needed for display.
            EntropyDataBase.ComputeEntropyBriefLook();     // For left panel.
        }
    }
    EntropyDataBase.LoadEntropyLongLook();   // For left panel.

    // Update the screen.
    // Much code irrelevant to operation of Model I elided.

    // Set the flag that prevents activities during a second.
    TogglesPtr->BetweenTicks = _EL_BetweenTicks_False;

    // And finally, use the governer to use up clock ticks if
    // this EiLab tick has completed too quickly.
    Governor.GovernSpeed();

    CurrentFunction = _EL_Fn_Setup;  // Reset to setup mode.
    return CurrentFunction;
}
```

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 Conserved Quantities

Three quantities are conserved: K (the number of bins in a histogram), A (the number of agents) and W (the total wealth, or total capital). The role of agents in a capital exchange model is distantly analogous to the role of mass in a thermodynamic system. Similarly, the role of capital (or wealth) is distantly analogous to the role of energy in a thermodynamic system. In thermodynamics, both mass and energy are conserved. These variables, A and W, must remain constant after every exchange, and after every tick if the capital exchange model is to be considered a closed economic system with respect to A and W. K is a more fundamental characteristic of the system, determining the upper and lower bounds on agent wealth. As such, it defines the nature of the economy being executed, and is held constant.

3.2.1.1 Small discrete parameter space

There are a minimum number of parameters: the two conserved quantities A and W, and the number of bins in the wealth histogram, a number which defines boundaries, and also plays a key role in the mathematics of entropic index.
The variables have discrete values, which ensures that the state space has a countable number of states.
The variables have very limited ranges, which ensures that the number of states in the state space is small.

3.2.1.2 Level of Abstraction

The level of abstraction is extremely high. A capital exchange model is very abstract as it does not model the goods and services being exchanged. It just models the exchange of capital. Model I goes well beyond this. There is no attempt to make the amount of capital exchanged realistic in any way. Nor are the boundaries on amounts of money possessed realistic. Studies of entropy in thermodynamics involve numbers of agents (atoms or molecules) of the order of 10^{23}, and a virtual infinity of potential states. The system for which the initial study of entropic index for ABMs was undertaken at [5] had 8 agents, and had a total of 13 possible states, each having a computable entropic index. The goal of Model I was to enable such a study. The techniques used in the studies at [5, 6, 7] would be quickly overwhelmed if applied to a less abstracted model.

3.2.2 Emergence

An emergent property is a behaviour which appears in a complex adaptive system but which was not explicitly part of the design. Several interesting characteristics are emergent in Model I, including:

- Automatic asymptotic rise of average entropic index towards its maximal value as time passes.
- Automatic conformity of the model to predicted behaviour with respect to percentage of time spent in any specific state.
- Evidence for three different causes of an “arrow of time” to be functional in the model.

3.2.3 Adaptation

In all EiLab models, agents are extremely simple and unable to adapt in any way, having no knowledge, no strategy, no talent, or no intentions as might be based on greed. The wonderful distributions of wealth that appear in the models of Drăgulescu and Yakovenko are purely phenomenological and are manifestations of deep entropy-driven processes. The agents of Model I, which is a simplification of those models, are similarly without any ability to adapt.

3.2.4 Agent objectives

In Model I, the agents have no implicit objectives. There is nothing within the agent which would prefer one condition of existence over another. All causal forces that shape the distribution of wealth are external to the agents.

3.2.5 Learning

In Model I, the agents are unable to learn.

3.2.6 Prediction

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

3.2.7 Sensing

In Model I, agents do not sense anything, and have no knowledge of anything other than the amount of capital in their possession at this tick.

3.2.8 Interaction

All interactions between agents are controlled by the Pseudo Random Number Generator (PRNG). The PRNG used in EiLab is the Mersenne Twister [8, 9]. When two agents are selected for an attempted exchange of capital, the PRNG is used to select them from the list of agents. The Mersenne Twister is considered to be almost entirely without bias for such purposes.

3.2.9 Stochasticity

EiLab 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. An EiLab model is a finite state machine (FSM), and, when given precisely the same starting parameters, a run should be 100% repeatable over millions of tick. EiLab has a "Halt At" feature by which you can run a model for a pre-determined number of ticks (e.g. until steady-state is achieved), stop the run to examine the results, 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 [8, 9].

Model I 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.

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 an EiLab-based model, the following types of action are randomized:

- Selection of each of a pair of agents for possible exchange of capital.
- Selection of a winner and loser in the pair of agents.

Hypothesis: It is herein hypothesized that, if a specific initialization of Model I parameters produces a stable probability distribution across all states in its state space after a sufficient number of ticks (say 500,000), then that same distribution will be produced by all runs having identical parameters, except for the PRNG seed. That is, the probability distribution across states in the state space is independent of the seed used, and independent of the trajectory of the model through its state space.

This hypothesis has been consistent with observations over many runs, although it has not been formally tested at this point.

3.2.10 Collectives

There is a single collective of immortal agents. The concept of collectives is not of significant interest in EiLab-based models.

3.2.11 Observations

A variety of real-time data observation techniques are available in version 1.36, the most recent version available as of this date. There is also a capability to write data to CSV files, which can then be loaded into MS Excel for analysis. This includes data for the main histogram (not very useful, since Model I histograms are so simple), and data for the probability distributions for entropy (the brief look and long look histograms) and the probability distribution for configurations visited by the model. This simple output can be used for a relatively wide variety of analytic activities using MS Excel.

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 Model I.

3.3.1 Initialization

Model I can be initialized in one scenario called the Entropic Index Study (EStudy). The settings of parameters and state variables are given here.

Entity:Parameter	Variable name	Default values	Minimim values	Maximum values
EiLabSys:Seed	Seed	1	1	32000
EiLabSys:NoOfBins	K	4	2	5
EiLabSys:NoOfAgents	A	8	2	20
EiLabSys:TotalWealth	W	20	2	100

Note that the parameters need to be selected with care. Not all combinations provide a viable model. For best results:

- Set K;
- Then set A = qK where q is an integer and q >= 2;
- Then set W half-way between A and AK, at W = A(K+1)/2.

Access to the seed is via the Reset toolbar button.

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 models without user intervention. This type of feature has not been implemented in EiLab, and is not available for Model I.

3.3.3 Sub-Models

There are no sub-models in Model I. It is an extremely simple ABM.

3.3.3.1 Data Collection and Display

While this does not affect the outcome of the model, and therefore does not constitute a sub-model, it does enable visualization of activities in the model in the course of a run, or examination after a run.

Real-time display is enabled for:
a. - a description of the size and status of all internal histograms and time series.
b. - a display of the current status of the primary wealth histogram used to calculate entropic index of the model.
c. - a display of the time series of historic values of entropic index, in two colours (most recent magenta, historic in blue).
d. - a display of a histogram of the most recent (up to 3000) values of entropic index.
e. - a display of a histogram of the values of entropic index collected over an extended run, comprising potentially the results of millions of ticks.
f. - a display of a bar chart (histogram) showing the probability of occupation for each configuration of state space that is visited by the model.

The data for displays a, d, e and f can be sent to CSV files for access and study using MS Excel.

References

[1] 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.

[2] 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.

[3] Boyle, G.H., "Capital Exchange Models – With Investigations of Entropic Index." (Version 3). CoMSES Computational Model Library.
Retrieved from: https://www.openabm.org/model/3860/version/3

[4] Drăgulescu, A., Yakovenko, V., "Statistical Mechanics of Money", Eur. Phys. J. B. 2000 17,723-729.

[5] Boyle, G.H., "A Definition and Examination of an Entropic Index for Agent-Based Models". CoMSES Computational Model Library.
Retrieved from: https://www.openabm.org/model/3860/version/3

[6] Boyle, G.H., "Maximum Entropic Index in Bounded Capital Exchange Models". CoMSES Computational Model Library.
Retrieved from: https://www.openabm.org/model/3860/version/3

[7] Boyle, G.H., "Bounded Capital Exchange Models and Entropic Index: Densities and Tarry Times". CoMSES Computational Model Library.
Retrieved from: https://www.openabm.org/model/3860/version/3

[8] Mersenne Twister. *Wikipedia*. Retrieved April 23, 2011, from http://en.wikipedia.org/wiki/Mersenne_twister

[9] Matsumoto, M.; Nishimura, T. (1998). "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator". ACM Transactions on Modeling and Computer Simulation 8 (1): 3–30.

[10] Cockshott, W.P., Cottrell, A.F., Michaelson, G.J., Wright, I.P., Yakovenko, V.M., "Classical Econophysics", Routledge (2009).