ELSEVIER

Ecological Modelling 93 (1996) 155-173

ECOLOGICAL MODELLING

Simulation and evaluation with energy systems blocks

Howard T. Odum [a,*], Nils Peterson [b]

[a] Center for Environmental Policy and Environmental Engineering Sciences, University of Florida, Gainesville, FL 32611, USA
[b] College of Education, Washington State University, Pullman, WA 99164-2114, USA

Received 30 August 1994; accepted 4 December 1995

Abstract

By using blocks programmed for each symbol, models using energy systems language can now be computer-simulated directly, by-passing the mathematics, which is done automatically. Energy systems methodology was introduced in 1967 to help aggregate systems overview and connect human verbal thinking to quantitative models constrained by principles of energy and hierarchy. The GENSYS library of symbol blocks for the simulation program EXTEND, when connected on screen, sets up a system of equations and generates output graphs by sending information back and forth between the blocks. The process of programming helped define EMERGY (spelled with an 'm'), empower and transformity mathematically so that the simulations can plot quantity, flow, EMERGY, empower and transformity. Examples include ecological and economic systems. Blocks and models for elementary teaching use pictorial icons, which help bring systems modeling and simulation to general education without the necessity of writing equations. This paper explains the modeling concepts and how to prepare and use blocks.

Keywords: Symbol simulation; EXTEND; EMERGY; General systems

1. Introduction

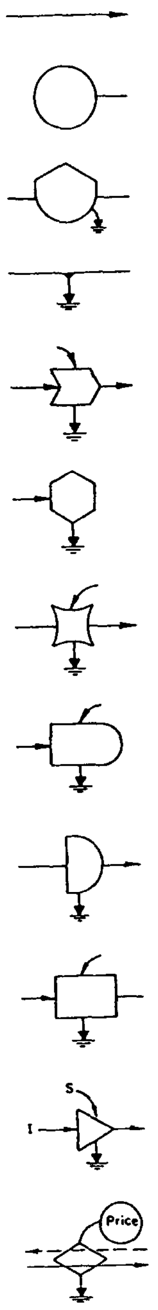
This paper introduces pre-programmed blocks with symbol icons of energy systems language for direct computer simulation and EMERGY evaluation using the program EXTEND. When blocks are connected on the computer screen, information is passed back and forth between blocks, facilitating output of simulation graphs and tables. Because the mathematics and energetics of each symbol are contained in the symbol blocks, a methodology results for simulating and evaluating systems diagrams without considering equations.

Modelling with energy systems language, introduced in 1967, is a methodology for converting verbal models into system network diagrams showing mathematic, energetic, cybernetic and hierarchical attributes simultaneously for many purposes, one of which is to facilitate the writing of computer programs for simulation. Another is the evaluation of EMERGY, defined as the available energy of one kind required directly and indirectly to generate a product or service. Fig. 1 shows the energy systems symbols from the book *Systems Ecology* (Odum, 1983, 1994a) where their mathematical relationships are given.

* Corresponding author. Tel.: +1-352-3920841; fax: +1-352-3923076.

SSDI 0304-3800(95)00221-9

In this paper, two sets of programmed blocks are described using EXTEND, one with the icons of the energy systems language (Fig. 2), the other with pictorial icons pre-calibrated to facilitate elementary teaching of systems simulation (see Section 3 below). First we describe modeling with energy systems symbols and the way these include hierarchical energy concepts, EMERGY and transformity.



Energy circuit. A pathway whose flow is proportional to the quantity in the storage or source upstream.

Source. Outside source of energy delivering forces according to a program controlled from outside; a forcing function.

Tank. A compartment of energy storage within the system storing a quantity as the balance of inflows and outflows; a state variable.

Heat sink. Dispersion of potential energy into heat that accompanies all real transformation processes and storages; loss of potential energy from further use by the system.

Interaction. Interactive intersection of two pathways coupled to produce an outflow in proportion to a function of both; control action of one flow on another; limiting factor action; work gate.

Consumer. Unit that transforms energy quality, stores it, and feeds it back autocatalytically to improve inflow.

Switching action. A symbol that indicates one or more switching actions.

Producer. Unit that collects and transforms low-quality energy under control interactions of high-quality flows.

Self-limiting energy receiver. A unit that has a self-limiting output when input drives are high because there is a limiting constant quality of material reacting on a circular pathway within.

Box. Miscellaneous symbol used for whatever unit or function is labeled.

Constant-gain amplifier. A unit that delivers an output in proportion to the input (I), but changed by a constant factor as long as the energy source (S) is sufficient.

Transaction. A unit that indicates a sale of goods or services (solid line) in exchange for payment of money (dashed line). Price is shown as an external source.

Fig. 1. Symbols of the energy systems language (Odum, 1967a,b,1983).

1.1. Energy systems language simulation

The set of energy systems language symbols was published in 1967 to represent, visually and together, energy, materials, mathematical relationships, hierarchy, the designs of self-organization and a means for calibrating models for simulation (Odum, 1967a,b,1972). A full explanation and application to

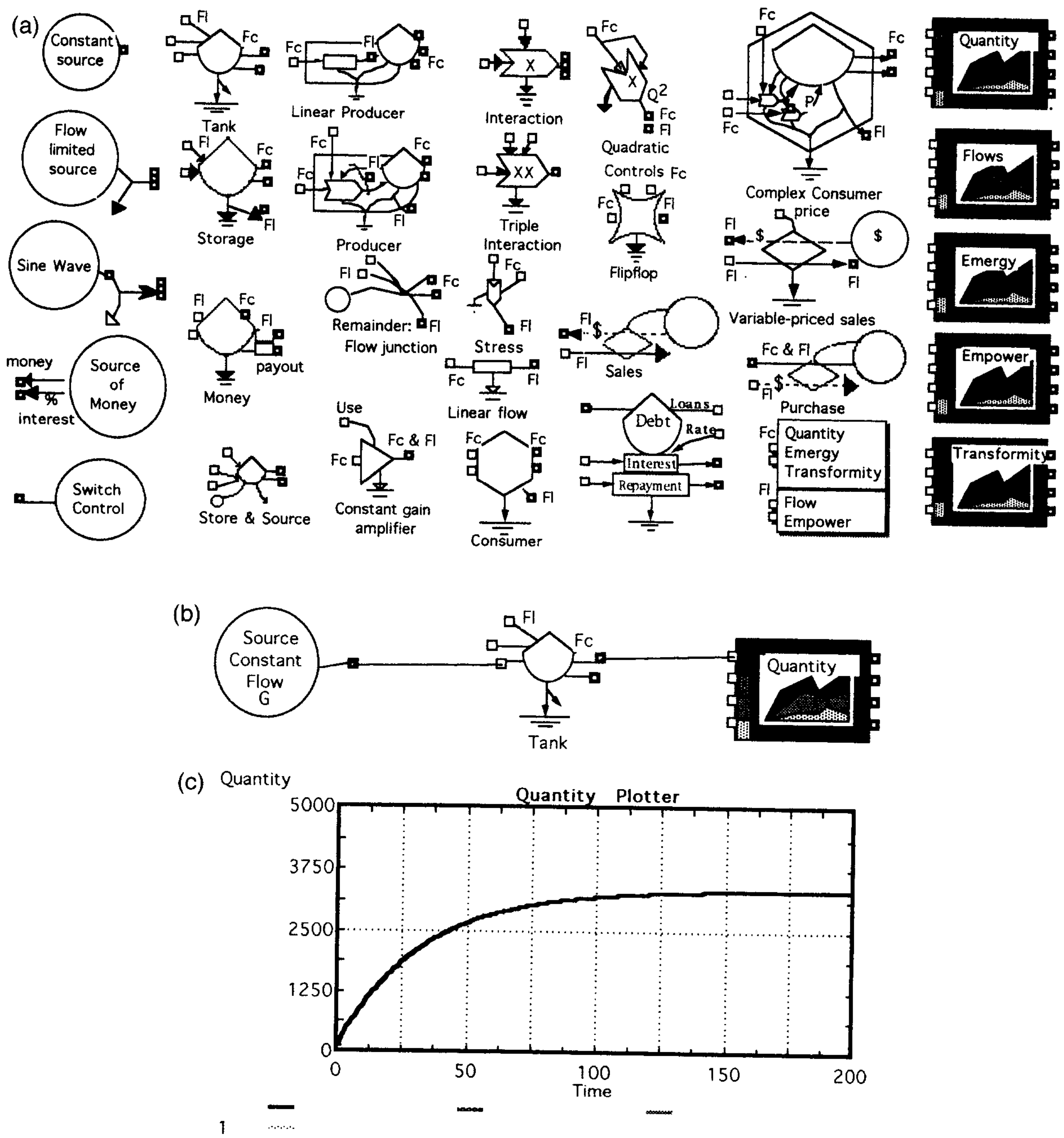


Fig. 2. Print-out of a computer screen with GENSYS blocks in use by EXTEND. (a) Unconnected icons of the energy systems symbol library (GENSYS); (b) simple operating model using two symbols and a plotter icon; (c) plot of simulation of tank model in (b).

many fields and scales was provided in book form (Odum, 1983, 1994a) and there are several summaries (Odum, 1972, 1975, 1989b; Odum and Odum, 1976, 1982; Odum et al., 1988; Jiang et al., 1993). Since the symbols (Fig. 1) represent kinetic relationships constrained by energy laws, the drawing of a model is equivalent to writing the equations for computer simulation. From modeling physical, chemical, biological, geological, ecological and economic systems, similar structures and network designs were found. Knowledge of these patterns helps modeling in new situations.

The latest version of our teaching workbook (Odum and Odum, 1989, 1994) has 46 minimodels of ecology and economics diagrammed with energy systems language, with equations derived and simulation code in BASIC. The new book, Ecological Microcosms (Beyers and Odum, 1993) has another 44. But for some time there has been a need to use computer graphical methods to diagram directly on computer screens in a way that creates a simulation without writing equations. This paper introduces a methodology and libraries of blocks for this purpose. An earlier set of blocks was used to model micro and macro-evolution and the EMERGY of Taxa (Odum, 1989a).

1.2. Energy, EMERGY, empower and transformity

Energy systems diagrams have pathways for all energy inflows to either be stored, flow out as exports, or drain away through the 'heat sink' symbol at the bottom, representing used energy (without available energy to do work). Where energy analysis is of interest, quantitative values of energy flows and storages may be written on pathways and tanks of energy systems diagrams, thus representing the energy conservation law and second law of energy depreciation. From such data procedures have been developed in recent years for evaluating EMERGY (spelled with an 'm' and capitalized to prevent confusion with energy). A revised maximum power hypotheses suggest that self organization tends to maximize useful EMERGY flow (emjoules per time) and therefore good policy involves planning that selects alternatives with higher EMERGY flows. Transformities calculated from EMERGY flows measure position in an energy hierarchy. Energy systems language is properly used when symbols for sources and components in energy systems diagrams are arranged from left to right in order of increasing transformity.

EMERGY is defined as the available energy of one kind required directly and indirectly to make a product or service. Its unit is the emjoule. As a common measure of work and real wealth, EMERGY measures everything: sun, wind, water, waves, fuels, materials, services, information, etc. For example, the solar EMERGY accumulated in forest growth of plantation pines was 9 E14 solar emjoules/ha/yr.

Transformity is defined as the EMERGY per unit energy, a measure of the position of something in the scale of energy hierarchy. Units are emjoules per joule. For example, the solar transformity of spruce forest wood was estimated as 3800 solar emjoules per joule. Transformities increase with successive energy transformations. Therefore, transformities measure position in the universal energy hierarchy. For example, the following are arranged in order of increasing transformities: sunlight, wind, rain, mechanical energy, wood, food, electric power, critical materials, drugs, human service, information sources and education. Tables of transformities from previous studies simplify EMERGY evaluations.

Empower (emjoules per time) is defined as the EMERGY flow per unit time. Units are emjoules per time. Data on inputs and energy flows in units of power (useful energy per time) are multiplied by their transformities to obtain flows in units of empower.

EMERGY, and its unit the emjoule, is a new dimension. The dimensions of transformity are EMERGY/energy (emjoule/J), not a dimensionless ratio. EMERGY/mass, EMERGY/$, EMERGY/individual and EMERGY/bit may be used also.

1.3. Equations for defining and programming EMERGY, empower and transformity

The behavior of EMERGY measures during processes of production and storage is shown in Figs. 3 and 4. Energy storage charges up and levels off as programmed EMERGY increases until the increment of increase is less than 5% of the level storage. So long as the storage is constant, the EMERGY is

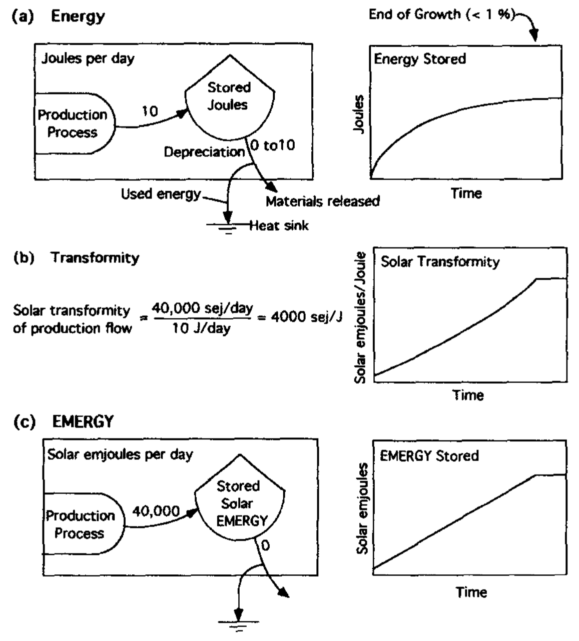


Fig. 3. Energy, EMERGY and transformity during a storing process. (a) Energy; (b) transformity; (c) EMERGY.

constant. The transformity increases during the storage process because it takes more and more EMERGY per unit of storage to offset the increasing depreciation.

Fig. 4 has the equations that go into a simulation model and the results of simulating growth of a stored quantity. The process of developing the simulation blocks was helpful in mathematically defining EMERGY. So long as a storage is growing, the

Energy Q : $dQ/dt = J - k1*Q - k2*Q$

EMERGY E: IF $dQ > 0$ then $dE/dt = Tr_j*J - Tr_q*k_2*Q$

IF $dQ = 0$ then $dE/dt = 0$

If $dQ < 0$ then $dE/dt = Tr_q*dQ/dt$

Where Transformity of J = Tr_j and

Transformity of Q = $Tr_q = E/Q$

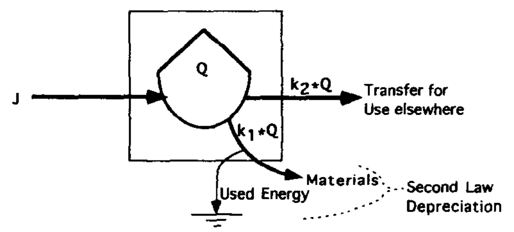


Fig. 4. Equations defining EMERGY and transformity of a storage unit.

stored EMERGY is the sum of inflowing EMERGY minus that exported to other systems (Fig. 4). But note that the energy drain (second law depreciation) is not included in the EMERGY equation since that flow is necessary to the EMERGY storing process (not subtracted from the integrated total). When the storage decreases, its EMERGY decreases in proportion, whether it is carried away or depreciates. EMERGY is conserved in a network up to the point the last available energy is used up, usually in an autocatalytic feedback interaction.

Other explanations of these measures and their uses are given in previous papers and books (Odum, 1986a,b,1988a,b,1989a,1991,1994b,1996; Huang and Odum, 1991; Odum and Arding, 1991). EMERGY, transformity and empower are automatically calculated in models assembled with general system blocks which are described next.

2. GENSYS: programmed blocks for general systems simulation

The symbols of the energy systems language (Fig. 1) were programmed for EXTEND as a general systems language for simulating any system. The icons of the object programmed blocks were collected on one screen in Fig. 2. When blocks are connected by drawing a pathway line using a mouse, the blocks become a simulation system with equations and energy constraints that follow the rules of the energy systems language. EMERGY is also calculated in each block, with transformities passed from one block to another accompanying the flows between blocks.

2.1. Force, flow and energy system thinking

Energy systems diagrams are a general systems way of visualizing system network of flows, storages and process interactions. They are not the flows of terms in successive mathematical operations. A pathway in this language is driven by a force (either a physical force or a concentration). A simple pathway with a barb (arrowhead) and without a backforce is a linear pathway with flow proportional to driving force (Fig. 5). An interaction (production process) has inflows and outflows in proportion to the interac-

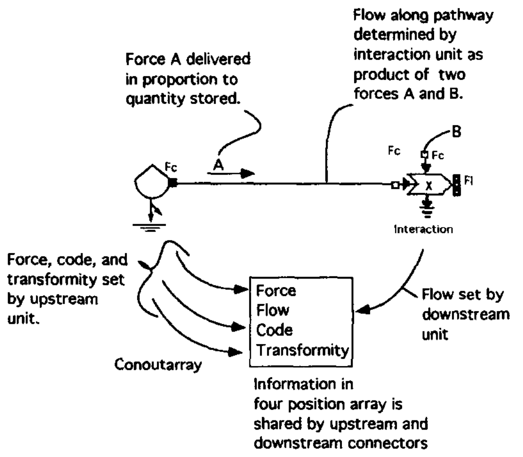

Flow along pathway determined by interaction unit as product of two forces A and B.
Force A delivered in proportion to quantity stored.
Fc
A
B
Fc
Fc
x
Fl
Interaction
Force, code, and transformity set by upstream unit.
Force
Flow
Code
Transformity
Flow set by downstream unit
Conoutarray
Information in four position array is shared by upstream and downstream connectors

Fig. 5. Force and flow in the energy systems language and its transfer between blocks with arrays.

tive function of two input forces (often the product). One unit of a system may drive the one downstream with a force, but the flow may depend on both units. In order to have on-screen objects represent the energy system way of thinking, the blocks must send information about the forces and flows back and forth between themselves.

Information is transferred between blocks along the lines shown in the EXTEND model diagrams. When a connector on one block is joined to the connector of another block, the connectors acquire the same numerical value. Thus, values put into connector in one block can be used by the connected block. If an array is used rather than a scalar, several values can be transmitted along a single pathway. For a system of blocks to function it is necessary for information to pass both ways over the joining pathway (Fig. 5). For example, the storage available to the next downstream unit is dependent on the upstream storage and this value has to be transmitted to the downstream user unit. Conversely, the use downstream has to be sent back upstream and subtracted from the upstream storage. In our first use of EXTEND, our symbols all had a cumbersome arrangement of two pathways for each inter-block connection and two pathway connections (one transmitting right and the other transmitting left).

Now, however, a four number array is used to pass information in both directions through one pathway using only one connector on each block. Information from one block is put into a designated position of the array shared by two blocks and is picked up from that position by the second block. Then the second block puts information into a different position in the array so it can be picked up and used by the first block. This array passing feature of the EXTEND program provides a great visual simplification on-screen. There is a receiving array in each block which is set equal to the inflow array that is passed. The first number in the array is the 'force' or concentration (example, biomass); the second number in the array is the flow (example, food consumed per unit time); the third number in the array is a code (1, 2, 3, etc.) available to instruct the connecting block for special purposes, or given zero values if not used. The fourth position carries the transformity which is used in downstream blocks (including plotters) to evaluate EMERGY and empower.

2.2. Procedure for use of symbol blocks in EXTEND for Macintosh and PC

The following description assumes the EXTEND program and our data files are loaded on the hard disk. Open one of the example work sheets, such as the one that merely portrays each of the general systems symbols unconnected (Fig. 2) or the model (worksheet) that has the elementary pictorial icons (Section 3), or create a new worksheet. A worksheet is a saved screen with its model system of blocks and settings for simulation. The 'library' menu provides access to a supply of energy symbol blocks which can be added to the simulation.

Energy symbols blocks are dragged to the desired position and connected to other copies of the same symbol or other symbols to make a systems model. Each symbol icon has connectors for joining the pathway lines. Connections must be from output connector (black cube) to input connector (clear cube). Some of the blocks have connectors labeled Fc for force and Fl for flow to help those who are uncertain about which kind of output or input should be connected. Only one pathway can go to an input. You can use more than one pathway from an output connector if it is going into a block that doesn't send anything back such as the plotters. However it is best not to use more than one to other blocks since most of them do send information back. A 'plotter' symbol from the library is included on the right side of the screen. Four variables to be plotted can be connected to the four input connectors of the plotter icon. Graph scaling is set within the plotter block.

When the model is run, a screen appears for setting time scale and iteration time characteristics for the simulation. As with all numerical integration, it is important to select an integration interval to avoid 'artificial chaos' when there is too high a ratio of flow to storage in a state variable (e.g., when the equations are stiff). After the graph is complete, or prematurely stopped, numerical values of the run are shown in a table below the graph. Both the graph and the numerical table may be printed or copied to the clipboard.

2.3. Simple examples of simulation of GENSYS models on EXTEND

Fig. 2b is a simple example of icons connected to form an energy systems model. In the center is a storage tank, which has a 3% depreciation outflow as part of its function. It receives a constant inflow from the source on the left. The line from the tank symbol to the plotter icon causes the plotter to generate the graph in Fig. 2c when the program runs. The user can set up this simulation in several minutes.

The model in Fig. 6a has a source symbol connected to provide a flow into the tank symbol that represents storage Q. The storage interacts with a force F from the second source to provide a production output P. The model has numerical values in Fig. 6b for calibration and simulation shown in Fig. 7.

2.4. Calibration methodology

One of the methodologies for model calibration that is frequently used with energy systems diagrams is to write flow values on the pathways of the diagram and storage values within the tanks (Fig. 6). Inflows and outflows to the same tank have to be the same quantity and in the same units of measure. One can study the values on the diagrams to see if these make sense. Usually some values are unknown and

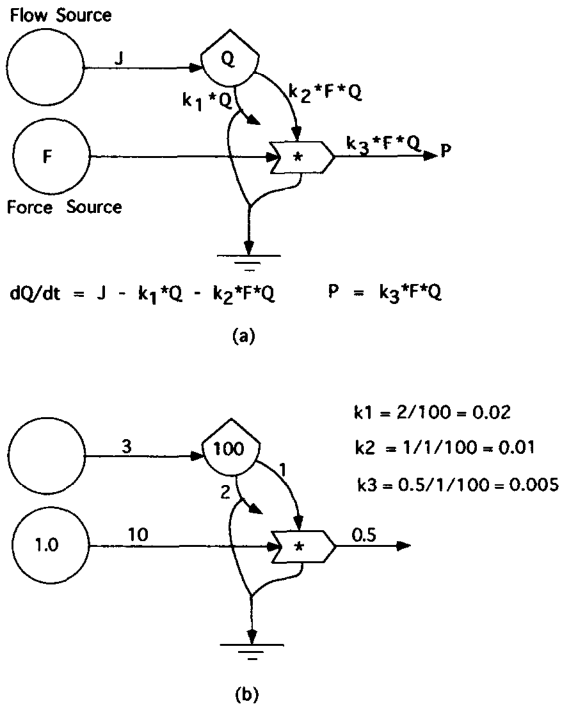


Fig. 6. Energy systems model TANKACT showing equations and method of assigning numbers for calibration. (a) Diagram with equations; (b) diagram with calibration values placed in storages and on pathways.

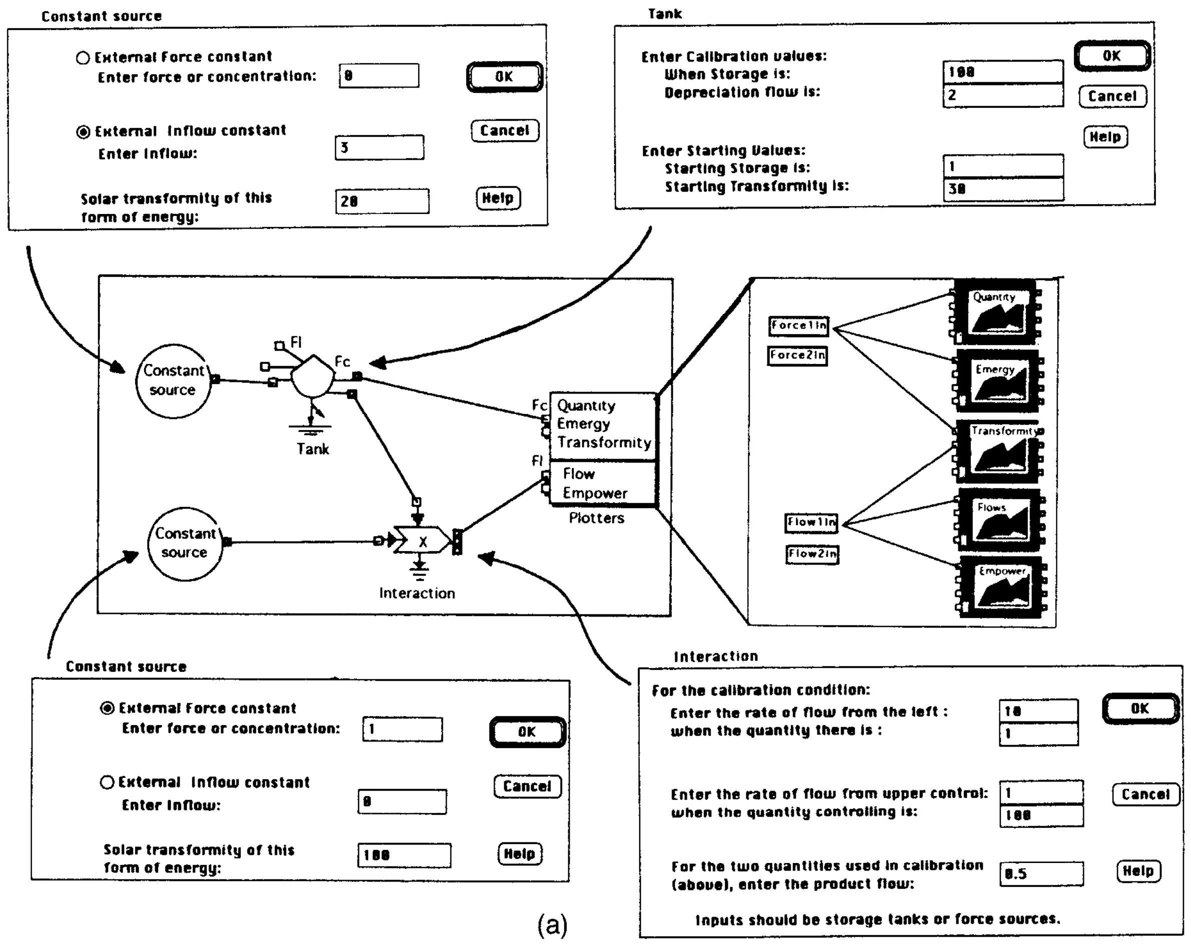
can be estimated by assuming a steady state (or using averages), balancing inflows and outflows. Or turnover times can be used to estimate flows, since they are readily visualized in many examples. If diagrams are done properly, items from left to right are in order of increasing transformity and thus increasing position in an energy hierarchy. From left to right they tend to have increasing turnover times and territories of support and effect.

After the flows and storages are written, coefficients are calculated by setting the mathematical term for each pathway equal to the flow and solving for the coefficient as in Fig. 6b. For example, if a linear depreciation pathway flow is 10 units per day, when the storage is 100 units, then solving for the coefficient results in a value of $k = 0.1$,

$K * Q = 10$ when $Q = 100$

Therefore

$k = 10/100 = 0.1$.



Constant source

○ External Force constant
Enter force or concentration: 0

◉ External Inflow constant
Enter Inflow: 3

Solar transformity of this form of energy: 20

Tank

Enter Calibration values:
When Storage is: 100
Depreciation flow is: 2

Enter Starting Values:
Starting Storage is: 1
Starting Transformity is: 30

Constant source

◉ External Force constant
Enter force or concentration: 1

○ External Inflow constant
Enter Inflow: 0

Solar transformity of this form of energy: 100

Interaction

For the calibration condition:
Enter the rate of flow from the left : 10
when the quantity there is : 1

Enter the rate of flow from upper control: 1
when the quantity controlling is: 100

For the two quantities used in calibration (above), enter the product flow: 0.5

Inputs should be storage tanks or force sources.

Fig. 7. (a) Macintosh screen from EXTEND with TANKACT model and dialog boxes. In the center is the model with connected icons ready to run. Calibration numbers in dialog boxes are those from Fig. 6. Items to be plotted are connected to the icon of the hierarchical group symbol PLOTTERS. To the right is the subsystem view obtained by double clicking on the PLOTTERS icon; (b) graphs obtained from simulation.

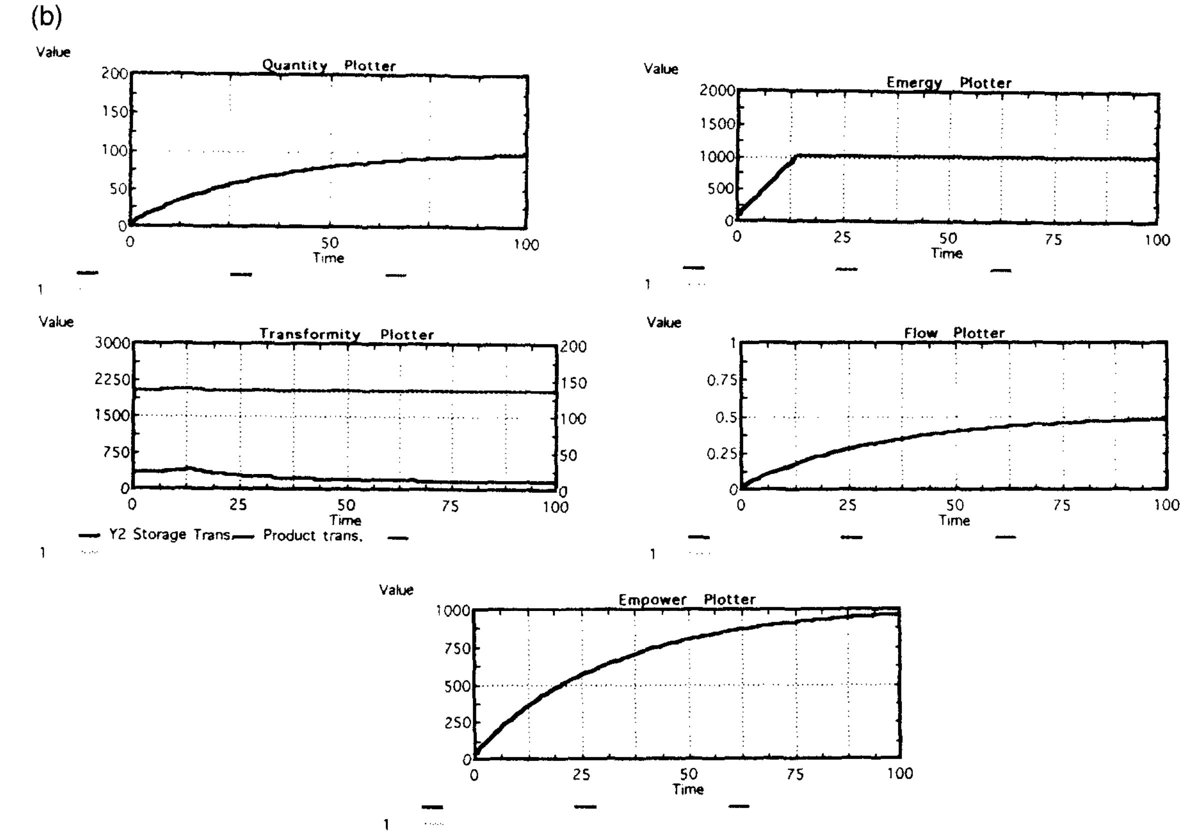


Fig. 7 (continued).

In our larger simulation models programmed in BASIC we use spreadsheets for calibration. With EXTEND, the calibration calculations are built into each block program and are done automatically once calibration flows and storages are entered in dialog boxes (Fig. 7).

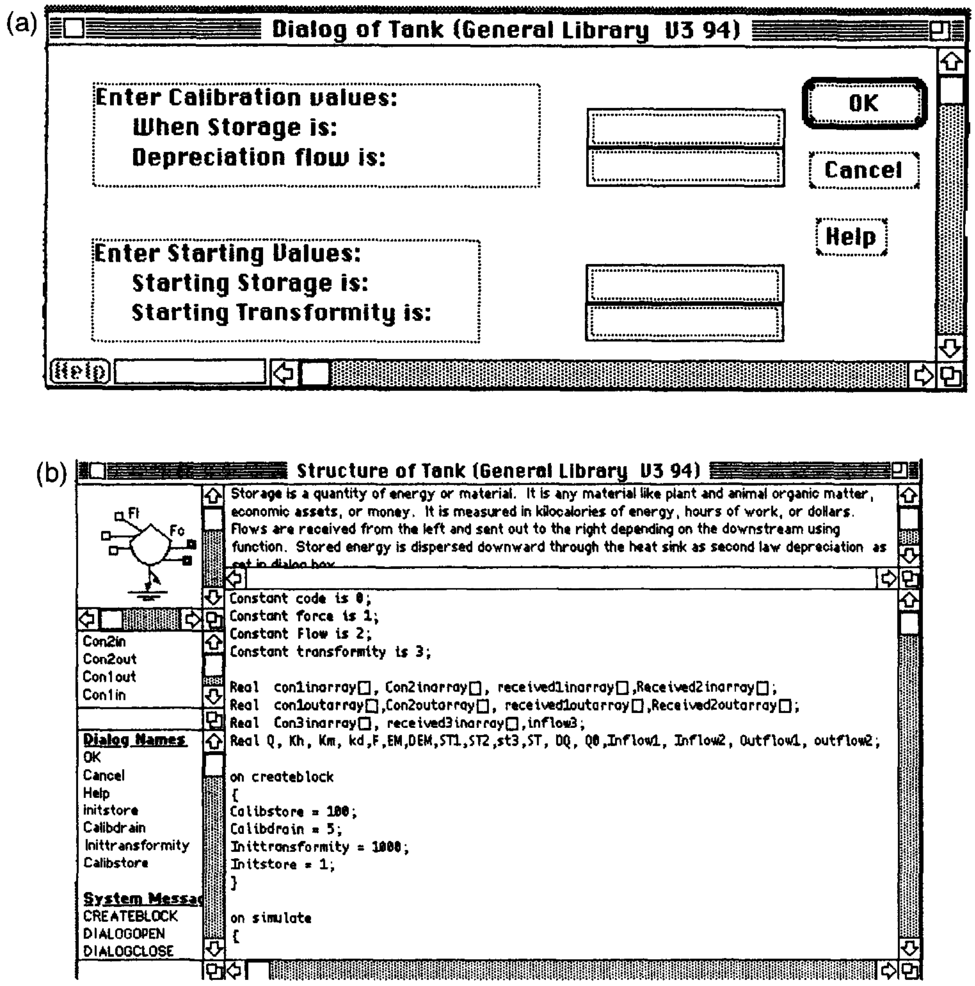
So long as all numbers on flows use the same time unit (days, years, etc.), the outputs will have the same time units. All flows into the same tank or merging pathways must have the same units. Flows coming out of storages have higher transformity than those going in. Flows into an interaction (production) process may have any units (but same time units). Note that an interaction is appropriate for the junction of two flows of different transformity with different effects per unit energy.

2.5. Calibration using dialog screens

The EXTEND program allows the user to make a dialog screen for each system block being programmed (Fig. 7) in which the user enters data and sets options for the simulation. The general energy systems symbols (Figs. 1 and 2) have been programmed so that all possible variables concerned with the block's modular functions can be set. The dialog screens were arranged for calibrating from an energy systems diagram on which flows and storages have been written. Included in Fig. 7a are four dialog boxes for the four symbols used. For example, the storing block TANK, has boxes for entering flows and storages for calibration. Since the initial condition values for the storage at the start of simulation is often different from the values for calibration, a box is supplied at the bottom of the dialog box for the starting value of the storage.

2.6. Programming in EXTEND

To program a new block or modify an existing block for EXTEND, an option key is used to bring the program to the screen, a linked pair of windows. One window allows design of the block's dialog box. Creating input fields in this dialog box also requires the definition of variable names used for user inputs. Contrast the user's view, Fig. 7a to the programmer's view, Fig. 8a of a TANK's dialog box. The other window used in programming a block is shown in Fig. 8b. In the upper left of the window, the block's icon can be drawn, or pasted from clipart developed elsewhere. The block's HELP text is typed into the upper right panel. Users can access this information by clicking on help buttons. Programming of the block's function is accomplished with a C-like language (MODL) in the main portion of the right window. Program of the TANK block (Figs. 1 and 2) is listed in Table 1.



Enter Calibration values:
When Storage is:
Depreciation flow is:

Enter Starting Values:
Starting Storage is:
Starting Transformity is:

Storage is a quantity of energy or material. It is any material like plant and animal organic matter, economic assets, or money. It is measured in kilocalories of energy, hours of work, or dollars. Flows are received from the left and sent out to the right depending on the downstream using function. Stored energy is dispersed downward through the heat sink as second law depreciation as set in dialog box

```
Constant code is 0;
Constant force is 1;
Constant Flow is 2;
Constant transformity is 3;

Real  con1inarray[], Con2inarray[], received1inarray[],Received2inarray[];
Real  con1outarray[],Con2outarray[], received1outarray[],Received2outarray[];
Real  Con3inarray[], received3inarray[],inflow3;
Real Q, Kh, Km, kd,F,EM,DEM,ST1,ST2,st3,ST, DQ, Q0,Inflow1, Inflow2, Outflow1, outflow2;

on createblock
{
Calibstore = 100;
Calibdrain = 5;
Inittransformity = 1000;
Initstore = 1;
}

on simulate
{
```

Fig. 8. Example of the Macintosh screen of the block TANK in programming mode. (a) Screen for setting up or changing dialog boxes and instructions; (b) screen showing the panes for typing in help statements, operational script, icons and connectors. Rest of script is in Table 1.

Table 1
Script of block TANK

```
Constant code is 0;
Constant force is 1;
Constant Flow is 2;
Constant transformity is 3;

Real con1inarray[], Con2inarray[], received1inarray[],
Received2inarray[];
Real con1outarray[],Con2outarray[], received1outarray[],
Received2outarray[];
Real Con3inarray[], received3inarray[],inflow3;

Real Q, Kh, Km, kd,F,EM,DEM,ST1,ST2,st3,ST, DQ, Q0,
Inflow1, Inflow2, Outflow1, outflow2;


on createblock


{
Calibstore = 100;
Calibdrain = 5;
Inittransformity = 100;
Initstore = 1;
}


on simulate
{
Em = St * Q;


If (Not Getpassedarray(Con1in, Received1inarray))

{
Con1inarray[code] = - 1;
Con1inarray[Force] = 0.0;
Con1inarray[Flow] = 0.0;
Con1inarray[transformity] = 0.0;
}


Else

{
Con1inarray[Flow] = Received1inarray[Flow];
Con1inarray[code] = received1inarray[code];
Con1inarray[transformity] = received1inarray[transformity];
}


If (Not Getpassedarray(Con2in, Received2inarray))

{
Con2inarray[code] = - 1;
```

Table 1 (continued)

```
Con2inarray[Force] = 0.0;
Con2inarray[Flow] = 0.0;
Con2inarray[transformity] = 0.0;
}


Else

{
Con2inarray[Flow] = Received2inarray[Flow];
Con2inarray[code] = received2inarray[Code];
Con2inarray[transformity] = received2inarray[transformity];
}


If (Not Getpassedarray(Con3in, Received3inarray))

{
Con3inarray[code] = -1;
Con3inarray[Force] = 0.0;
Con3inarray[Flow] = 0.0;
Con3inarray[transformity] = 0.0;
}


Else

{
Con3inarray[Flow] = Received3inarray[Flow];
Con3inarray[code] = received3inarray[code];
Con3inarray[transformity] = received3inarray[transformity];
}


If (Not Getpassedarray(Con1out, Received1outarray))

{
Con1outarray[code] = - 1;
Con1outarray[Force] = 0.0;
Con1outarray[Flow] = 0.0;
Con1outarray[transformity] = 0.0;
}


Else

{
Con1outarray[Flow] = Received1outarray[Flow];
}


If (Not Getpassedarray(Con2out, Received2outarray))
{
```

Table 1 (continued)

```
Con2outarray[code] = – 1;
Con2outarray[Force] = 0.0;
Con2outarray[Flow] = 0.0;
Con2outarray[transformity] = 0.0;
}
Else
{
Con2outarray[Flow] = Received2outarray[Flow];
}

Inflow1 = Con1inarray[Flow];
Inflow2 = Con2inarray[Flow];
Inflow3 = Con3inarray[Flow];
Outflow1 = Q * Con1outarray[Flow];
Outflow2 = Q * Con2outarray[Flow];

DQ = inflow1 + Inflow2 + inflow3 – Kd * Q – outflow1
– outflow2;
Q = Q + DQ * DeltaTime;
If (Q = = 0)
Q = 0.000000001;
If (Q < 0)
Q = 0.000000001;

St1 = con1inarray[transformity];
St2 = con2inarray[transformity];
St3 = Con3inarray[transformity];

If (DQ/Q > 0.05)
DEM = St1 * inflow1 + ST2 * inflow2 + st3 * inflow3 –
ST * outflow1 + ST * outflow2;

If (Dq/Q < 0.05 and Dq > 0.0)
DEM = 0;

If (Dq = = 0.0)
dEM = 0.0;

If (DQ < 0.0)
DEM = ST * DQ;

EM = EM + DEM * Deltatime;
If (EM = = 0.0)
EM = 1;

If (EM < 0.0)
EM = 1;

ST = EM/Q;
Con1inarray[force] = Q;
Con2inarray[force] = Q;
Con3inarray[force] = Q;

Con1outarray[transformity] = ST;
Con2outarray[transformity] = ST;
Con1outarray[code] = 1; * * code 1 is force out
Con2outarray[code] = 1; * * code 1 is force out
Con1outarray[Force] = Q;
Con2outarray[Force] = Q;
```

Table 1 (continued)

```
Con1in = Passarray(Con1inarray);
Con2in = Passarray(Con2inarray);
Con3in = Passarray(Con3inarray);

Con1out = passarray(Con1outarray);
Con2out = passarray(Con2outarray);
}

* * Initialize any simulation variables.
on initsim
{
Q = Initstore;
St = Inittransformity;
Kd = Calibdrain/Calibstore;
Em = Q * St;

Makearray(con1inarray,4);
Con1inarray[code] = – 1;
Con1inarray[Force] = 0.0;
Con1inarray[Flow] = 0.0;
Con1inarray[transformity] = 0.0;

Makearray(con2inarray,4);
Con2inarray[code] = – 1;
Con2inarray[Force] = 0.0;
Con2inarray[Flow] = 0.0;
Con2inarray[transformity] = 0.0;

Makearray(con3inarray,4);
Con3inarray[code] = – 1;
Con3inarray[Force] = 0.0;
Con3inarray[Flow] = 0.0;
Con3inarray[transformity] = 0.0;

Makearray(con1outarray,4);
Con1outarray[code] = – 1;
Con1outarray[Force] = 0.0;
Con1outarray[Flow] = 0.0;
Con1outarray[transformity] = 0.0;

Makearray(con2outarray,4);
Con2outarray[code] = – 1;
Con2outarray[Force] = 0.0;
Con2outarray[Flow] = 0.0;
Con2outarray[transformity] = 0.0;
}

* * User clicked the dialog HELP button.
on help
{
showHelp();
}
```

2.7. Features of the general symbol blocks (Fig. 2)

The following are the operations of the GENSYS blocks. Full explanation of the math of these relationships was given earlier (Odum, 1983, 1994a). Whether they send forces or flows, almost all blocks receive transformities on their inputs and send out transformities on their outputs.

2.7.1. Sources

There are several sources (inputs from outside the defined system window). There is a constant source that the user can set for either constant force or constant flow and set its value. The money source is a flow source that also transmits an interest rate that can be set. The switch control sends a logic pulse and can be set either for a regular period or for random number delivery.

With the flow limited source (example of application: the sun), the energy inflow (I) is set as externally limited. Whatever unit draws energy from that flow can use only the portion of the flow that is yet unused (remainder R). At each iteration the block sends R as the force and receives back the part of the use equation that is supplied from the downstream block. The source receives this usage term ($k * F$) and uses it to update the remainder. For example, if the downstream block is an interaction ($k * R * F$) with force F, the equations that are implemented are,

$$\text{Remainder} = R = I - k * R * F,$$

Notes to Table 1:
The CREATE handler executes when a block is first created on the EXTEND worksheet. In the example, default settings for the user variables are established.
The INITSIM handler is invoked prior to the start of the simulation. In the example, data from the user dialog are used to calculate coefficients and/or set intitial conditions.
The SIMULATE handler is invoked for each iteration of the simulation. It aquires current data from neighboring blocks, updates internal state variables (in this case using Euler integration) and transmits output data to neighboring blocks. If a model requires a small iteration time (such as Δtime = 0.1 or less), it takes a lot of memory when it runs and when it is saved to disk. One can increase the dt before saving in order to get more on disk. For more explanations of the MODL programming language see the EXTEND manual.

where k is a constant coefficient and therefore

$$R = I/(1 + k * F).$$

2.7.2. Storages

The module TANK receives inflows and sends out forces according to the storage, including backforce for those blocks upstream that are affected by backforce (example, water tanks). It has a depreciation whose rate can be set. The module STORAGE is like TANK except it has an outflow of materials as a fraction of the inside depreciation (example, nutrients). The money storage has a quantity connector for reading the storage. One of its two output connectors (payout) sends out a flow in proportion to storage. The other connector sends out its force (storage) with its outflow depending on the downstream block. There are also storages within the producer and consumer blocks (see below).

2.7.3. Interactions

The simplest INTERACTION receives two forces and sends out a flow. The TRIPLE INTERACTION receives three forces. The STRESS is the same as the simple interaction but with an icon that shows its negative connotation. The quadratic transformation interacts with itself (flows in proportion to the square of the input force). It has a connector for the squared force and one for the flow that is in proportion. The CONSTANT GAIN AMPLIFIER is a module generalized from electronics. Its output is a constant gain factor times one input force, but power for the output is only drawn from the upper pathway (use). This module is necessary if one is translating models that have infinite energy sources, such as those using intrinsic rate of growth that are the basis of most population models. To be more realistic these interactions should be replaced with a regular interaction that includes the real energy source.

2.7.4. Linear flows

LINEAR FLOW block flows in proportion to upstream force. Flow can be in proportion to the difference of upstream and downstream forces if that is appropriate (example, flow between two water tanks). FLOW JUNCTION merges flows from which other modules may interact (pump from) to draw off part, with the remainder going to a different place or being exported from the system. The mathematics of this block is similar to the flow limited source. What these blocks do is convert storages into average flows, thus eliminating them from integration. In this way variables which are important but whose oscillations are too fast to be relevant to the scale of the model are removed from integration. For example, phosphorus levels are important to yearly models of plant production, but the daily phosphorus fluctuation is not. This saves huge amounts of computer time and memory. It makes slow models into fast ones.

2.7.5. Economic exchanges

Several blocks receive commodities for sale and send out the money according to price. One of these has a dialog to set a constant price; another has an input for setting the price according to the force from another block. Other blocks receive money and send out commodities or services. One of these sends out a flow inverse to the price set; another sends out a

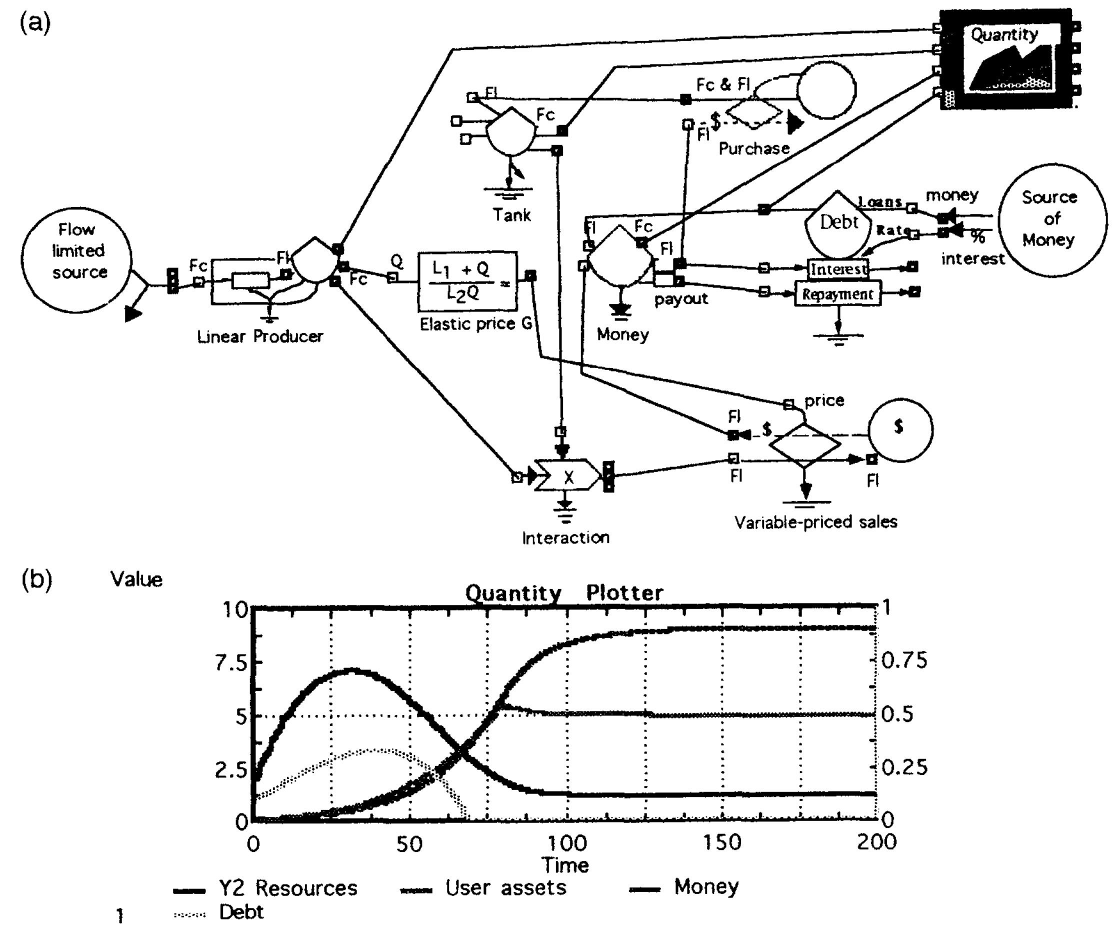


Fig. 9. A more complicated ecological economics model ENVUSE (Odum, 1996), which includes aspects of environmental and economic production and finance. (a) Macintosh screen with connected icons; (b) simulation results.

force that is inverse to the price set. The DEBT icon has connectors to receive interest rate and loans, storing the accounting of the debt while passing the money through to the user. Interest payments are received back and passed through to lender as are repayments of principal.

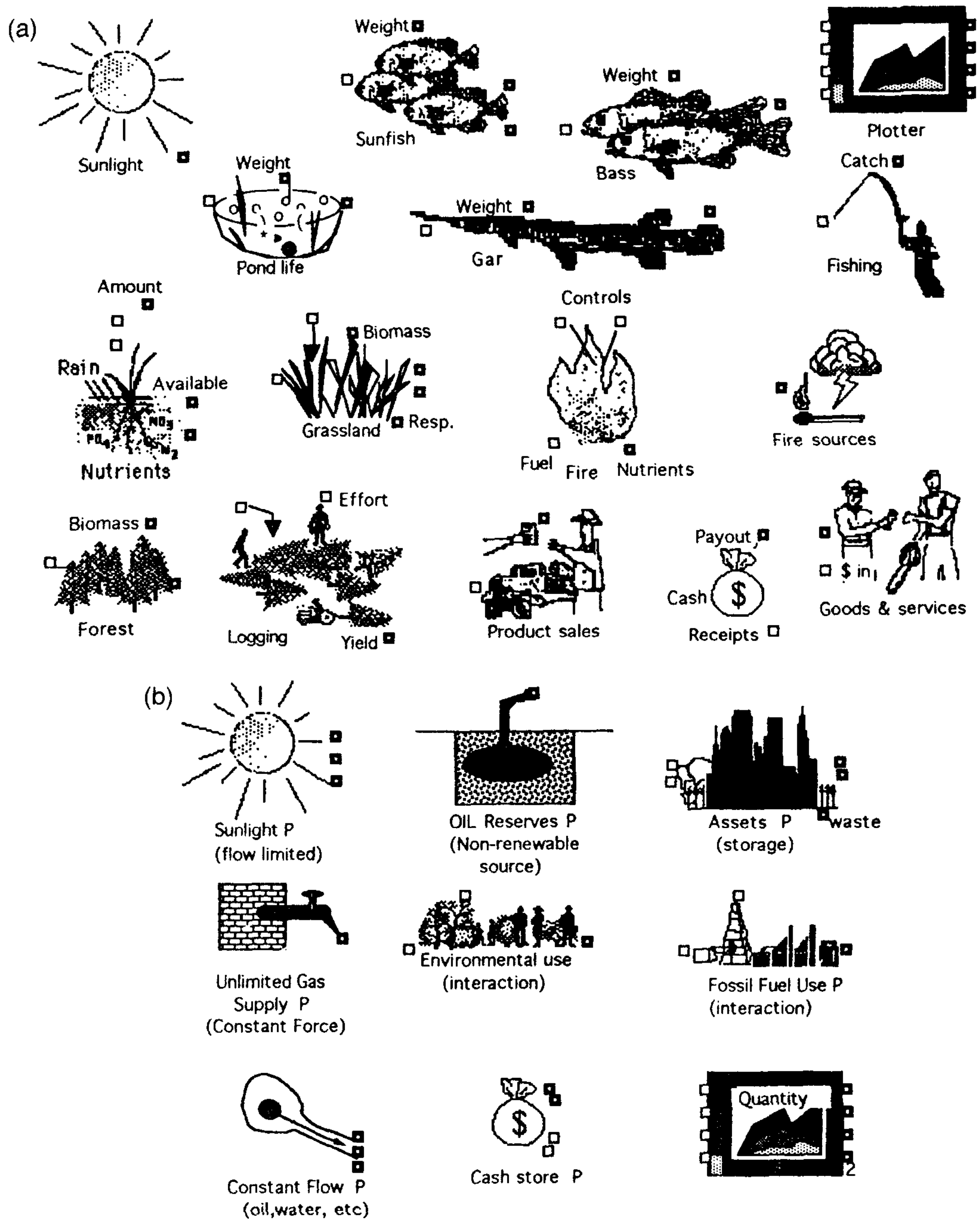


Fig. 10. Pictorial icons used in BioQUEST package "Environmental Decision Making" and in introducing global issues in introductory environmental science.

2.7.6. Logic functions

The incurved box is the group symbol for switching functions, which may include AND gates, OR gates, flip–flops, etc. The generic and discrete libraries for EXTEND, supplied with the software by the publisher has many blocks that can be used as the inside subsystem within an energy systems hierarchical block. In Fig. 2 a flip–flop is provided that turns on with a high threshold and off with a lower threshold. For example, it can be used to turn a fire on and off according the combustibles remaining.

2.7.7. Group symbols

Several symbols have combined some of the above functions. The PRODUCER symbol receives one or two input forces and sends out a flow of gross production to one connector and also to its internal storage, which is auto-catalytically part of the productive interaction. The internal storage also has a depreciation drain whose material release flows through the recycle connector. The CONSUMER block is similar except for the shape of its icon. Producers have lower transformities than the consumers they supply and should be to the left of their blocks. Energy systems language, following traditional engineering (but not Forrester's symbols), uses the rectangular box as miscellaneous, including subsystems. EXTEND version 3.1 has 'hierarchical models' for subsystems. These can be constructed on-screen by assembling primitive modelling blocks without programming a block from scratch. To be consistent with energy systems language, use a box-shaped icon for the subsystem and the regular icons for the subsystem within.

2.7.8. Plotters

Five plotters are available for GENSYS models. Quantity, EMERGY and transformity plotters should appropriately receive pathways from force-delivering icons (in other words from storages or force sources). Flow, empower and transformity plotters can appropriately receive pathways from flow pathways. Note that the transformity plotter reads transformity from force or flow-delivering connectors. If an inappropriate pathway goes to a plotter, there is no harm done. It just reads zero or no-value, which tells you that you may have an inappropriate connection.

The GENSYS package includes a hierarchical model PLOTTERS that has all 5 plotters inside as a subsystem (Fig. 7a). It has two channels of flow input and two channels of force input. It saves time to put this block in your model for all the plotting that may be needed. One may duplicate plotters if more than 4 plotting channels are needed.

2.8. Ecological economics example, ENVUSE

ENVUSE in Fig. 9 is a more complex example of simulation with GENSYS blocks. It is a general interface model between the work of environment generating high EMERGY from the left of the diagram interacting with the economic uses of environment products which brings purchased EMERGY into the interaction. Price of sales is inverse to

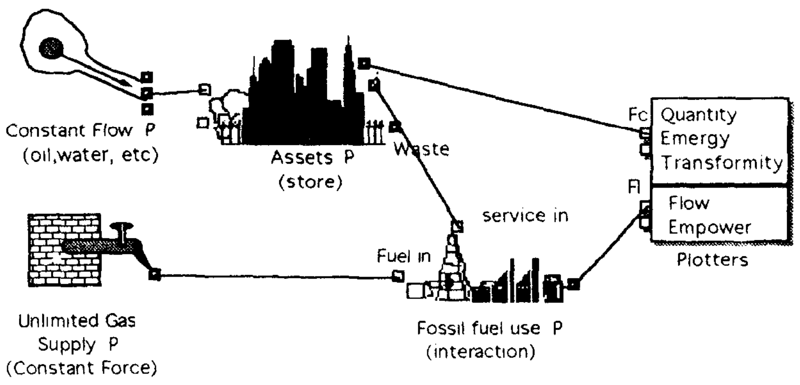


Fig. 11. Model TANKACT using pre calibrated, pictorial icons whose equations, calibrations and simulation graphs are similar to the general systems model in Figs. 6 and 7.

availability (with inelastic tail). The model has provision for loans, interest and repayment.

3. Pre-calibrated blocks with pictorial icons

Another style of simulation uses blocks pre-calibrated to represent a particular systems component (examples, bass, logging operation, oxygen storage). The dialog boxes are simpler, since more of the calibration is set within the script. The icon may be a picture (examples, bass, logging operation). Preprogrammed blocks are useful where the same system is under extended study. Blocks for research purposes may be internally complex.

The preprogrammed blocks may be used for introductory teaching where the student is only given one or two things to set in each block. Some of the preprogrammed blocks which we prepared for teaching purposes with pictorial icons were collected in Fig. 10. These pictorial blocks are compatible with the those of the general systems library. They include transformity-EMERGY calculations, although these may be ignored in elementary presentations. EMERGY, empower and transformity only appear in simulation if one of the special plotters is applied. For example, a pictorial teaching version of the storing example is shown in Fig. 11. The source is an icon for a fuel source and the storage is an icon for assets of the city. The calibrations are the same as the general systems model in Fig. 6a, but the dialog boxes are simpler. The simulation graph that results is the same as that using general systems blocks (Fig. 6b).

Another example is the forestry model in Fig. 12a, which was used in a published elementary

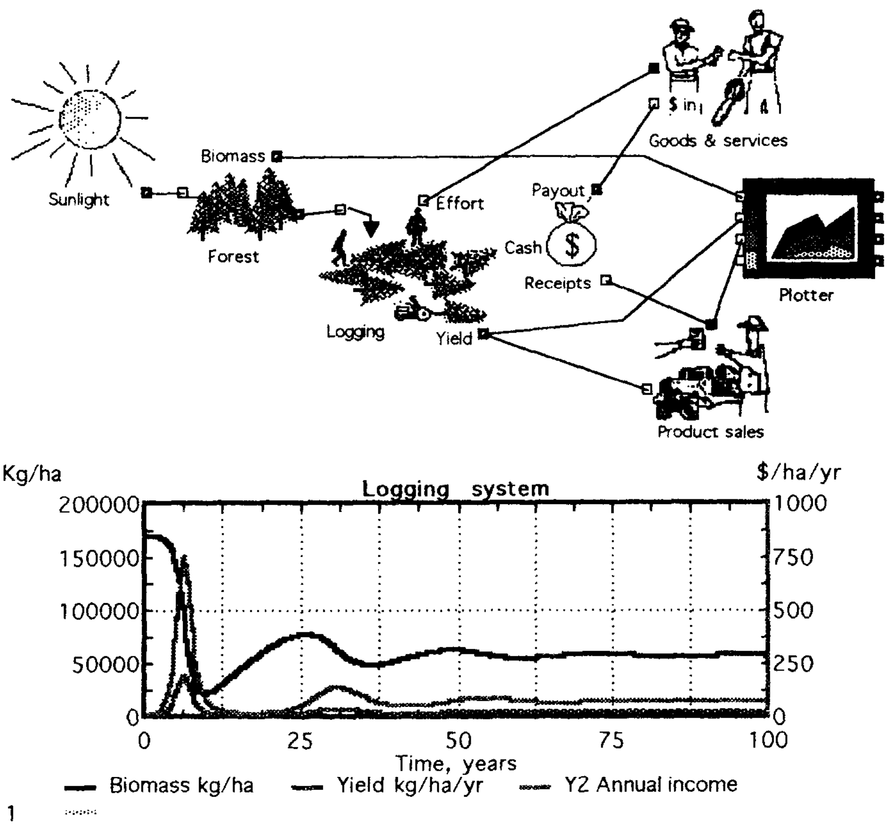


Fig. 12. Ecological economics teaching mini-model with pre-calibrated icons from BioQUEST (Odum et al., 1993). (a) Screen with connected pictorial icons and their simple dialog boxes; (b) simulation result.

teaching package (Environmental Decision Making in the *BioQUEST Collection*, Odum et al., 1993). Here the icons are the pictorial ones. Forest grows, is cut and sold as logs, with money used to pay for the goods and services for further cutting. The simulation (Fig. 12b) is sensitive to the price of the logs and the price of labor. The run has a surge of initial income from cutting a more mature forest initially, with a steady state developing in which the forest is cut continuously. Neither forest biomass or earnings are as high later.

To summarize, the programmed symbol blocks which we supply or which users can program for themselves for EXTEND appear to be an alternative way for simulating small and medium sized models, for research, teaching and new ways of thinking about energy and general systems relationships. Since the mathematics is hidden within the programs, it may be a preferable way for some with other training and connectivity aptitudes to simulate ecosystems and other networks.

Acknowledgements

We acknowledge suggestions of Bob Diamond and Pat Diamond of "Imagine That" and collaboration with Elisabeth C. Odum in developing and testing pictorial symbols in elementary teaching. Some of the elementary icon blocks were prepared for BioQUEST, a project developing software for undergraduate biology education, in which we acknowledge John Jungck and Patti Soderberg of Beloit College. Daeseok Kang assisted in checking scripts. EXTEND for MacIntosh and for Windows 3.1 on IBM compatible PC is available from "Imagine That", 6830 Via Del Oro, Suite 230, San Jose, CA 95119-1353.

References

Beyers, R.J. and Odum, H.T., 1993. Ecological Microcosms. Springer-Verlag, NY, 555 pp.

Huang, S.L. and Odum, H.T., 1991. Ecology and economy; EMERGY synthesis and public policy in Taiwan. J. Environ. Manage., 32: 313–333.

Jiang Y., Deying, X., Yamhui, W., Daoping, N., Zaiping, L., Yangang, Z., Yuiping, M., Fengyou, W., Honggi, W., Keping, M., Gruoting, Y., Shilin, D., Zhu, L., Shilin, D. and Jie, C. 1993. Chinese translation of H.T. Odum's Systems Ecology. Chinese Academy of Forestry, Beijing, China, 772 pp. English language original published in 1983, John Wiley, 544 pp.

Odum, H.T., 1967a. Biological circuits and the marine systems of Texas. In: ed. T.A. Olson and F.J. Burgess Pollution and Marine Ecology. Wiley Interscience, NY, pp. 99–157.

Odum, H.T., 1967b. Energetics of world food production. In: Problems of World Food Supply, President's Science Advisory Committee Report, Vol. 3. White House, Washington, DC, pp. 55–94.

Odum, H.T., 1972. An energy circuit language for ecological and social systems: Its physical Basis. In: ed. B. Patten, Systems Analysis and Simulation, Vol. II. Academic Press, NY, pp. 139–211.

Odum, H.T., 1975. Combining energy laws and corollaries of the maximum power principle with visual system mathematics. In: Ecosystems, Analysis and Prediction. Proc. of the Conf. on Ecosyst. at Alta, Utah, SIAM Institute for Mathematics and Society, pp. 239–26.

Odum, H.T., 1983. Systems Ecology: An Introduction. J. Wiley, NY, 644 pp.

Odum, H.T., 1986a. Enmergy in ecosystems. In: ed. N. Polunin, Environmental Monographs and Symposia. John Wiley, NY, pp. 337–369.

Odum, H.T., 1986b. Unifying Education with Environmental Systems Overviews. In: Environmental Science, Teaching and Practice. Proc. of the 3rd Int. Conf. on the Nat. and Teaching of Environ. Stud. and Sci. in Higher Edu. held at Sunderland Polytechnic, England, September 1985, pp. 181–199.

Odum, H.T., 1988a. Self organization, transformity and information. Science, 242 (Nov. 25, 1988): 1132–139.

Odum, H.T., 1988b. Living with complexity. In: The Crafoord Prize in the Biosciences 1987, Lectures. Royal Swedish Academy of Sciences, Stockholm, 87 pp, pp. 19–85.

Odum, H.T., 1989a. EMERGY and Evolution. In: ed. P.W.J. Ledington, Preprints of the 33rd Annual Meet. of the Int. Soc. for the Syst. Sci., Vol. III. Edinburgh, Scotland, 2–7 July 1989, pp. 10–18.

Odum, H.T., 1989b. Simulation models of ecological economics developed with energy language methods. Simulation, 1989 (August): 69–75.

Odum, H.T., 1991. EMERGY and biogeochemical cycles. In: ed. C. Rossi and E. Tiezzi, Ecological Physical Chemistry, Proc. of an Int. Workshop, Nov., 1990, Siena, Italy, Elsevier Science Publ., Amsterdam, pp. 25–65.

Odum, H.T., 1994a. Ecological and General Systems, revised edn. of Systems Ecology. Univ. Press of Colorado, Niwot, 644 pp.

Odum, H.T., 1994b. The EMERGY of natural capital. In: ed. A.M. Jansson, Investing in Natural Capital, Proc. of 1992 Int. Ecol. Econ. Conf., Stockholm, ch. 12. Island Press, Washington, DC.

Odum, H.T., 1996. Environmental Accounting, EMERGY and Decision Making. John Wiley, NY, 365 pp.

Odum, H.T. and Arding, J.E., 1991. EMERGY Analysis of Shrimp

Mariculture in Ecuador. Working Paper prepared for the Coastal Resources Center, Univ. of Rhode Island, Narragansett, RI, 114 pp.

Odum, H.T. and Odum, E.C., 1976, 1982. Energy Basis for Man and Nature. McGraw Hill, NY, 2nd Ed., 330 pp.

Odum, H.T. and Odum, E.C., 1989. Computer Minimodels and Simulation Exercises. Available for Apple II, PC, or MacIntosh. Center for Environmental Policy, Univ. of Florida, Gainesville, 32611.

Odum, H.T., Odum, E.C., Brown, M.T., LaHart, D., Bersok, C. and Sendzimir, J., 1988. Energy, Environment and Public Policy: A Guide to the Analysis of Systems. UNEP Regional Seas Reports and Studies No. 95., United States Environment Programme, Nairobi, 109 pp.

Odum, E.C., Odum H.T. and Peterson, N.S., 1993. Environmental Decision Making. A BioQUEST Collection In: ed. J.R. Jungck and P. Soderberg, Module for The BioQUEST Library. Academic Software Development Group, Univ. of Maryland, College Park.