IGRASP - A SYSTEM DYNAMICS SOFTWARE PACKAGE WITH AUTOMATIC CODE GENERATION FACILITY

Souvik Banerjee, Mohit Juneja, and Pratap K. J. Mohapatra

Department of Industrial Engineering & Management
Indian Institute of Technology, Kharagpur
Kharagpur - 721 302, India

ABSTRACT

In this paper, we describe the features of IGRASP (Interactive Graphic Simulation Package), a software package developed by the authors, which helps in developing and simulating system dynamics models and which has the facility of automatically developing codes for system dynamics models from their flow diagrams. The package integrates the following four modules: (1) The graphic user-interface module, (2) The automatic code generation module, (3) The simulation module, and (4) The output module. The key to the automatic code generation lies in the dimensional matching of the variable for which the code is generated and the set of variables and parameters which influence it.

INTRODUCTION

Since the days of DYNAMO and DYSMAP, great improvements have taken place recently in software packages that help development and simulation of system dynamics models. Software packages now allow drawing causal loop diagrams (COSMOS and VENSIM) and flow diagrams (STELLA, ITHINK and POWERSIM) on the screen using icons. Automatic generation of codes now seems both a natural extension and a distinct possibility, particularly because system dynamics model equations generally use only basic arithmetic operators and because both sides of each of these equations are always dimensionally matched.

THE IGRASP PACKAGE

IGRASP is developed keeping in mind the requirements of both beginners as well as experienced system dynamics users. A prime concern is to incorporate the facilities of system dynamics modelling with the minimum hardware requirement so as to enhance its scope of application. It is developed for a DOS environment. It gives various facilities which are comparable to those provided by packages in a WINDOWS/MACINTOSH environment. These facilities include drawing of various diagrams like causal loop, policy structure, sectorial overview and flow diagrams, as well as graphical output of results. Having these facilities in the widely available DOS environment is a major advantage of IGRASP. Additionally, the hardware and software requirements of the other packages make them expensive and out of reach of many system dynamicists, especially in countries like India. IGRASP has also another unique feature - the automatic code generation facility.

IGRASP consists of four modules:

1. The Graphical User Interface Module,
2. The Automatic Code Generation Module,
3. The Simulation Module, and
4. The Output Module.

These are discussed below with the help of an example.

THE GRAPHICAL USER INTERFACE

This module facilitates drawing of various diagrams by selecting appropriate icons. The icons represent various symbols normally used for drawing flow diagrams. Figure 1 shows a flow diagram constructed using these icons. The module stores the icon descriptors and their relationships (in the form of a connectivity matrix) using the graph theory approach. It facilitates the saving, editing (such as modifying and moving) and retrieving of the diagrams and also provides the user with the basic file functions. The module also facilitates drawing diagrams other than the flow diagram, using the same icons. Figure 2 and Figure 3 respectively show a causal loop diagram and two policy structure diagrams, drawn using the module. However, no code is generated for these diagrams. It provides error messages and has in-built checks to ensure

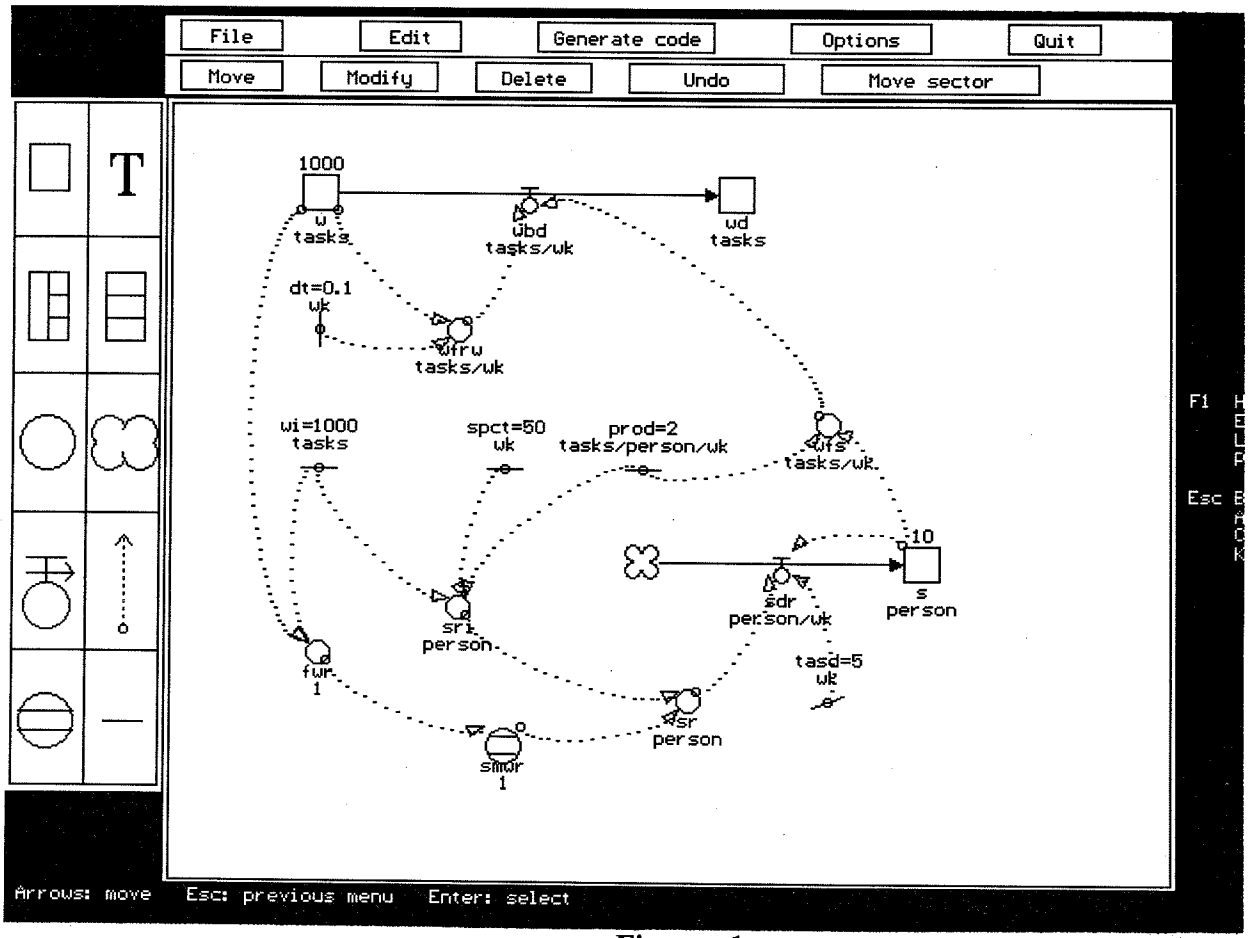


Figure 1

correct modelling. It also facilitates drawing of graphs for various table functions. Figure 4 shows the screen layout for a table function.

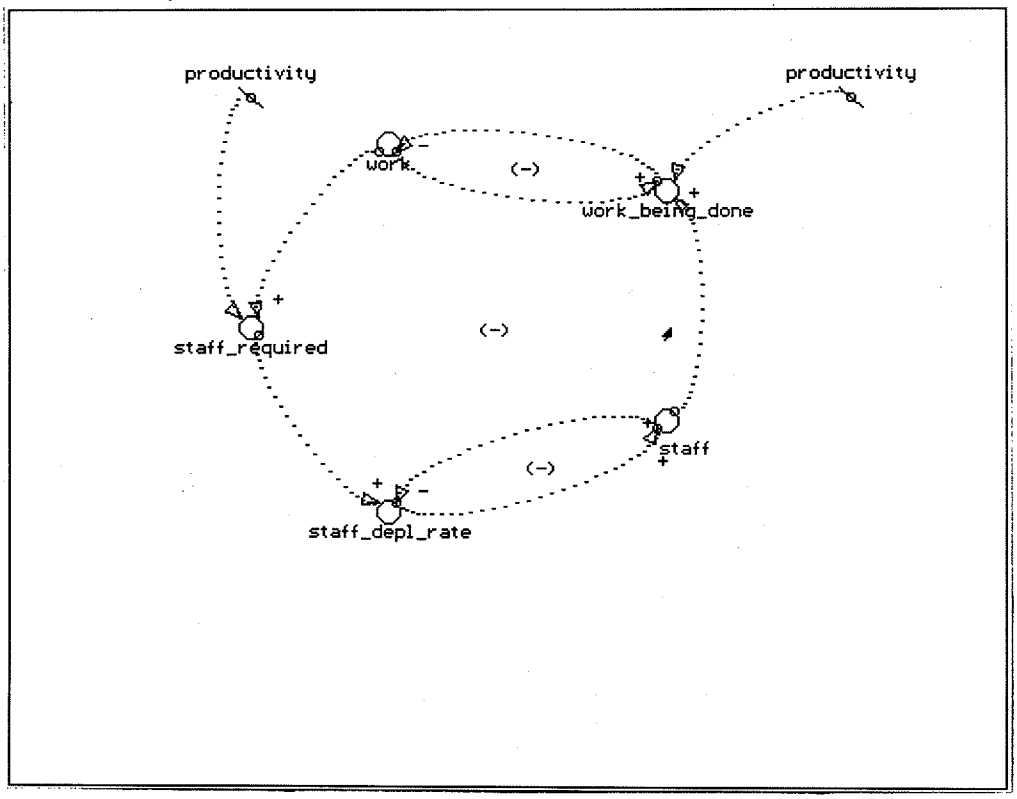


Figure 2

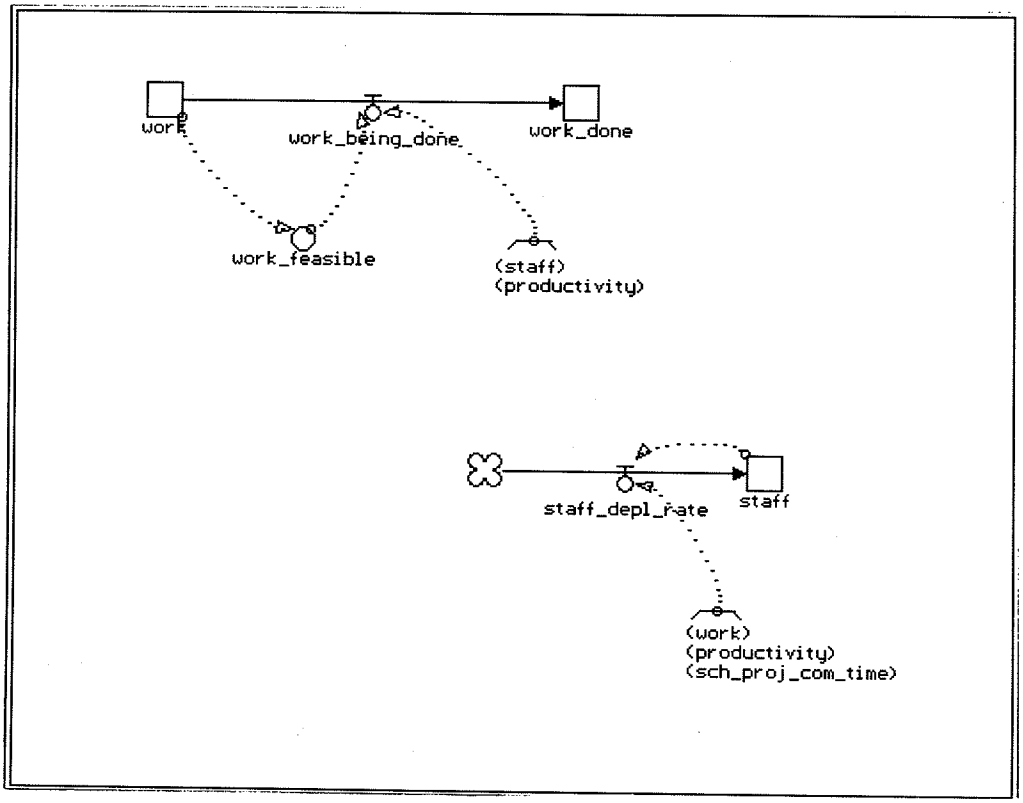


Figure 3

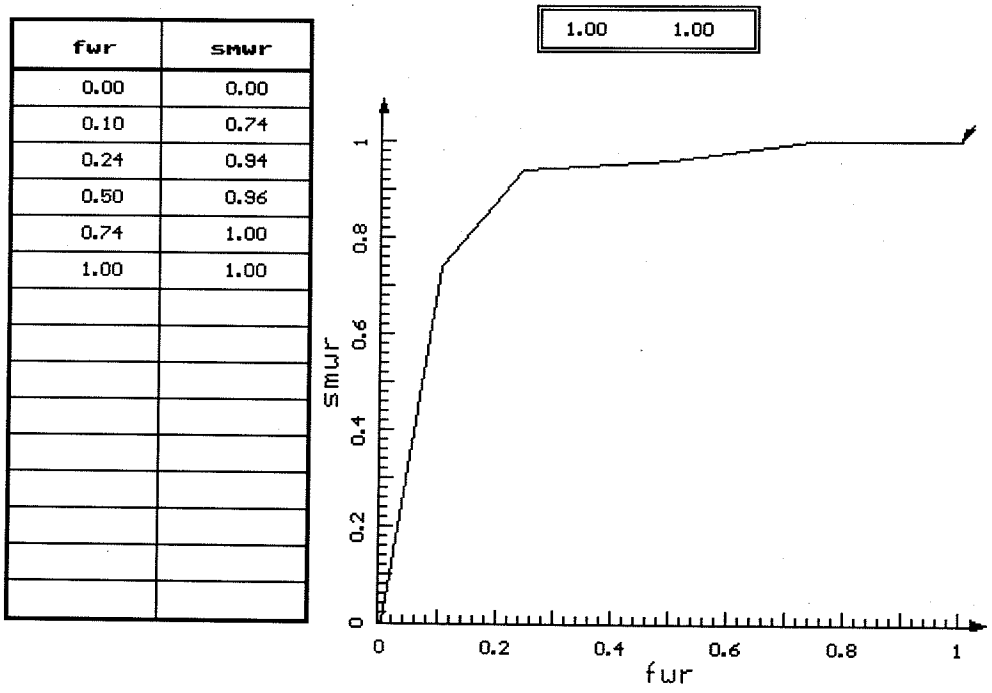


Figure 4

THE AUTOMATIC CODE GENERATION MODULE

This module makes use of icon descriptors and the connectivity matrix to perform dimensional analysis and generate model code for each object in the flow diagram. It gives appropriate error messages in case of dimensional mismatch. Since it is based on the principle of dimensional equivalence it cannot generate equations of trigonometric, exponential and non-linear algebraic functions. Further, it fails to distinguish between an addition and a subtraction between two entities that have the same dimension. The user can also modify the model equations and write additional code, if required.

The Principles of Automatic Code Generation

Suppose we wish to generate the code for a variable (entity) y which is known to be influenced by a set of n causal variables (some or all of them may be constants) x_1, x_2, x_3, ..., x_n:

$$y = f(x_1, x_2, x_3, ..., x_n).$$

We assume a simple algebraic function for f(.); in fact we have assumed that f(.) is composed of only addition, multiplication, and reciprocation operators. At the moment, we have not been able to consider the case of the subtraction operator.

The problem lies in combinatorially generating all possible algebraic forms of expression that connect the causal variables x_1, x_2, x_3, ..., x_n and test each expression for its dimensional

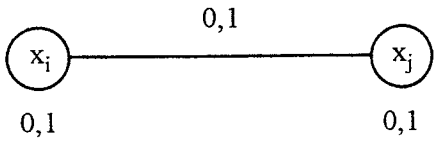
matching with the affected variable, y.

To develop these expressions, we have assumed that each variable, x_i, can be in two states: (1) x_i and (2) its reciprocal x_i^{-1}. Further, we have considered that each variable, in any one of its two states, has a binary relationship with each of the remaining variables, in any of its two states. This binary relationship can be either additive or multiplicative.

Thus, the different forms of expression connecting two variables x_i and x_j would be as follows:

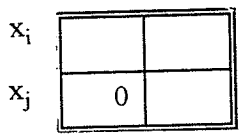
Additive Relationships	Multiplicative Relationships
$x_i + x_j$	$x_i * x_j$
$x_i + x_j^{-1}$	$x_i * x_j^{-1}$
$x_i^{-1} + x_j$	$x_i^{-1} * x_j$
$x_i^{-1} + x_j^{-1}$	$x_i^{-1} * x_j^{-1}$

We can depict the relationships in the form of an undirected weighted graph:

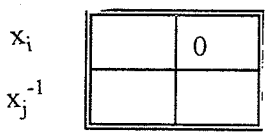


Here the nodes denote the variables x_i and x_j, and the edges denote the relationships. Each node can take the binary values 0 and 1, 0 indicating the variable and 1 its reciprocal. Each edge similarly takes the binary values 0 and 1, 0 indicating an additive relationship and 1 a multiplicative relationship.

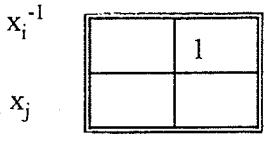
Another way to represent the relationships is in the form of a square edge matrix, where the rows (and the columns) of the matrix represent the nodes of the graph and the entries in the matrix represent the binary values that the edges can take. The matrix is symmetric and only the upper triangular part (excluding the diagonal) is adequate to represent a relationship. We however need as many matrices as there are relationships. Matrix representations for a few relationships between x_i and x_j are the following:



$x_i + x_j$

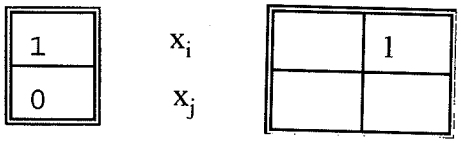


$x_i + x_j^{-1}$



$x_i^{-1} * x_j$

Another representation of a relationship is to define the matrix rows (and columns) by their natural states , but associate the rows (and columns) with a node array representing either of the binary states which each node can occupy. For example, the relation $x_i^{-1} * x_j$ can be depicted as under:

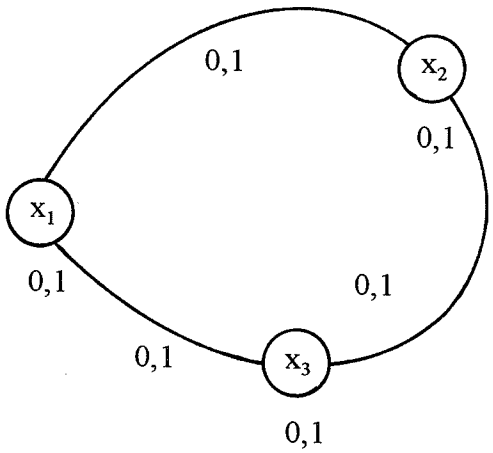


Edge Matrix

Now, consider that the affected variable is a function of three causal variables x_1, x_2, and x_3:

$$y = f(x_1, x_2, x_3).$$

The graphical representation is given by:



The matrix representations of the following relationships

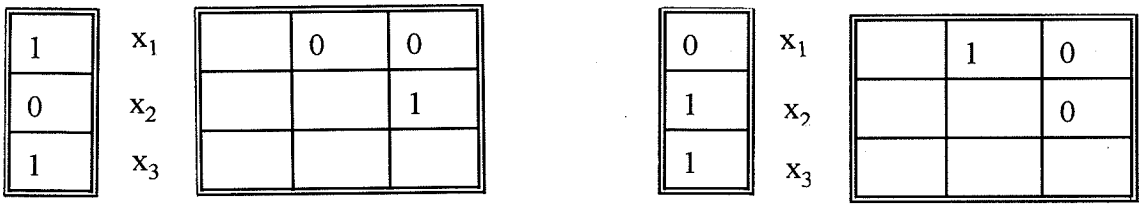
$$y = x_1^{-1} + x_2 * x_3^{-1}$$

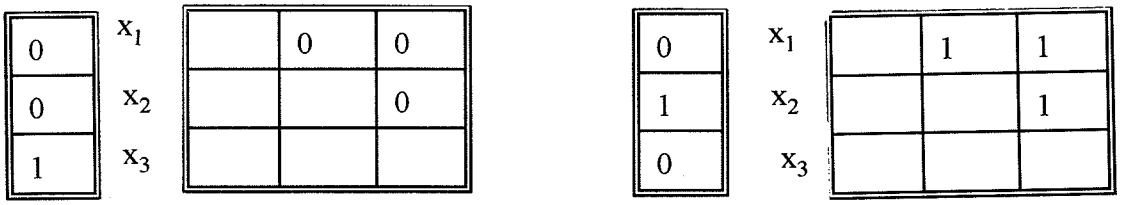
$$y = x_1 * x_2^{-1} + x_3^{-1}$$

$$y = x_1 + x_2 + x_3^{-1}$$

$$y = x_1 * x_2^{-1} * x_3$$

are shown below :





The total number of possible relationships for this case would be equal to

(Total no. of combinations of the node array) * (Total no. of combinations of edge matrix)

$= 2^3 * 2^{3*2/2} = 2^3 * 2^3 = 64.$

The steps for the algorithm are given below:

1. Generate 0,1 states for each causal variable.
2. Generate relationships using the entities.
3. Build distinct sets of multiplicative relationships.
4. Evaluate each of them in terms of dimensions.
5. Check for dimensional matching using these sets. If dimensions are matched, then the search is successful and the equation is written. The search is stopped; else, go to step 6.
6. Check if all the combination relationships are exhausted. If yes, go to step 7, else go to step 2.
7. Check if all combinations of states are exhausted. If yes, then the search has failed, meaning that no valid relationships are existing. The search is stopped; else go step 1.

In general, if there are n causal variables then the number of possible relationships is given by:

$$2^n * 2^{n(n-1)/2} = 2^{n(n+1)/2}$$

A limitation of the algorithm is as under:

Suppose dim(a) = dim(b), and the desired relationship is 1/(a+b). Using the algorithm, however, the generated relationship will be 1/a + 1/b , which, though dimensionally congruent, denotes a different relationship.

Effort is under way to modify the algorithm to take care of this problem. The alternative method groups all equal dimensions as one entity and looks for only multiplicative or divisional relationship between groups to match dimensions. Inside the groups additive relationships are taken. Thus if a and b are taken as one unit, say c (dim(c) = dim(a) = dim(b)) , and 1/c is found to be matching , then 1/c is retranslated to either 1/a + 1/b or 1/(a+b).

The code generated for the model (fig. 1) is given below.

```
*
* Demonstration for IGRASP
*
L w=w-dt*(wbd)
N w=1000
L wd=wd+dt*(wbd)
N wd=0
R wbd=wfs+wfrw
L s=s+dt*(sdr)
N s=10
R sdr=sr/tasd+s/tasd
A wfrw=w/dt
A wfs=s*prod
A fwr=w/wi
A smwr=TABHL(TVsmwr,TVfwr,fwr)
T TVsmwr=0/0.75/0.95/1/1/1
T TVfwr=0/0.1/0.25/0.5/0.75/1
A sr=smwr*sri
A sri=wi/spct/prod
C wi=1000
C spct=50
C prod=2
C dt=0.1
C tasd=5
C dt = 0.1
C length = 100
C prtper = 1
print w wd s
run DEMONSTRATION
```

On inspection of the generated code we find that the equation for *wbd* and *sdr* should be as under:

R wbd=min(wfs,wfrw)
R sdr=sr/tasd-s/tasd

Further, we note that *dt* has been defined twice since the model has used the value of dt. So we must delete one of these two equations.

We can, of course, add rerun statements.

THE SIMULATION MODULE

This module checks for syntax errors, loops without levels, gives appropriate error messages, and provides a large library of built-in functions such as DELAY3, DELAYP, SMOOTH, DLINF3, TABHL, TABLE, CLIP, STEP, RAMP, MAX, MIN, and ABS. The module automatically sorts the equations into a computable sequence and carries out simulation using the EULER algorithm. It stores the output in two data files, one for tabular output and the other for graphical output.

THE OUTPUT MODULE

The tabular output gives the values of the variables that the user wants to print for different runs. Up to 15 variables can be tabulated for each run. A table header appears for every screen. A run header identifies every run.

The graphical output is menu-driven and has two options i.e. normal plots and coplots. Normal plots can have up to 4 variables plotted against any other variable, such as TIME. A coplot shows a variable against any other variable for all simulation runs up to a maximum of four successive simulation runs in one figure.

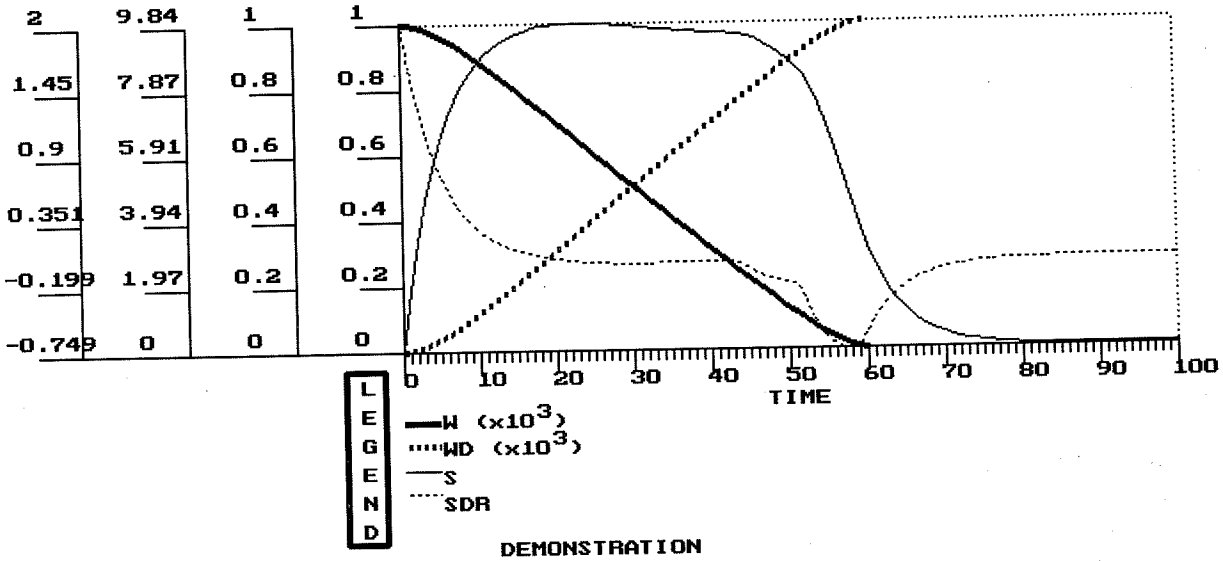


Figure 5

Figures 5 shows a time-based plot and Figure 6 shows a coplot. The user can select X-axis and Y-axis variables from the menu that appears before plotting. The module also has several on-screen options like Explode (vary the X-axis range), Set range (vary the Y-axis range), Edit (change caption and X-, Y-axis descriptors), Default (reverts to default ranges), Zoom (increases the graph size (height) by 50%) and Print (which outputs the graph on the screen to the printer).

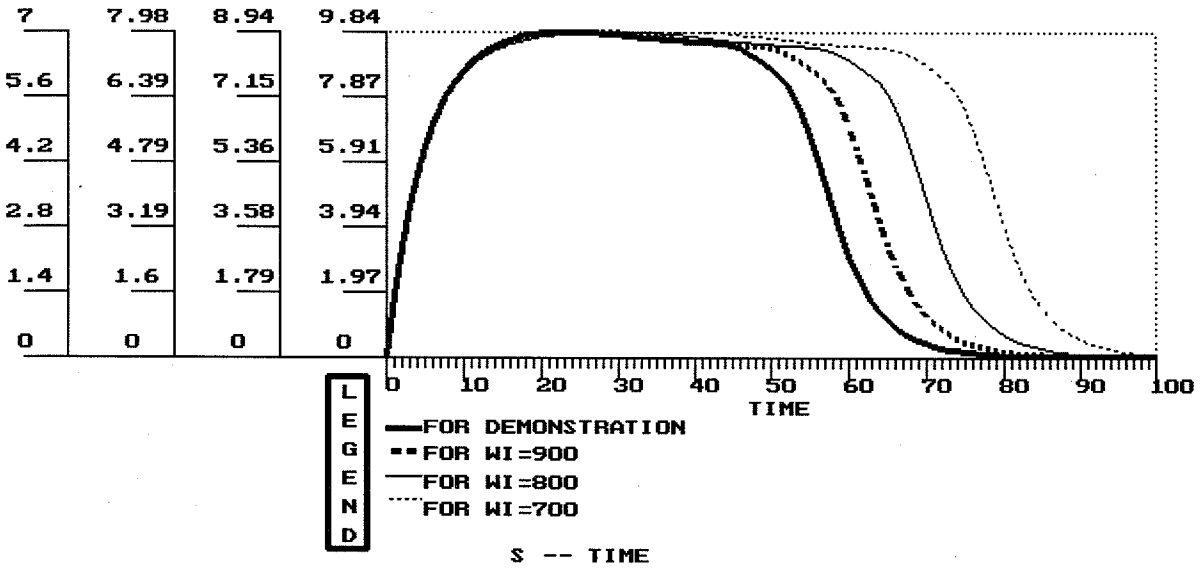


Figure 6

REFERENCES

COGNITUS (1994), Business strategy and process design with ITHINK.

COSMIC Holding Company (1994), COSMIC and COSMOS, Version 4.1, Professional power for Model Development, Simulation, Optimization in system dynamics models.

Dangerfield B. (1992), The System Dynamics Modelling Process and DYSMAP2, European Journal of Operations Research, vol. 59, pp.203-209 .

Eberlein R. L. and D. W. Peterson (1992), Understanding Models with VENSIM, European Journal of Operations Research, vol. 59, pp.216-219.

High Performance Systems Inc. (1992), STELLA II, Tutorial and Technical documentation.

Peterson S. (1992) Software for model-building and simulation: An illustration of design philosophy, European Journal of Operations Research, vol. 59, pp.197-202 .

POWERSIM newsletter, number 2, 1994.

Pugh and Roberts , Asssociates,Inc.(1978), GAMING DYNAMO/F, DYNAMO newsletter, vol. 13.