THE DEVELOPMENT OF DYSMAP2

Olga Vapenikova
Computing Services Section
Salford University
England

Abstract. This paper introduces a new implementation of the System Dynamics simulation language DYSMAP, a current project at Salford University. A short overview of the history, syntax and features of DYSMAP will be given. The new DYSMAP2 system operates as an interpreter. The relative merits of computer interpretation vs. compilation into FORTRAN will be discussed. An outline of the operation of this new portable package (written in FORTRAN77) will include high level descriptions of the parser, run-time interpreter,the interactive environment, dimensional analyser and the optimiser. The performance of the new DYSMAP2 package will be illustrated.

OVERVIEW OF THE DYSMAP MODELLING LANGUAGE

Modern control theory relies on rigorous mathematics, such as the use of differential equations, and drastic simplification of the dynamic system under scrutiny. System Dynamics is the application of control theory to large, non-linear socio-economic systems where the main objectives are overall understanding of system behaviour and the impact of strategic policy options, rather than hardware design. For the operational research scientist working as a stock control engineer or a marketing executive in a large company the use of rigorous advanced mathematics has been eliminated by defining the syntax of System Dynamics in a way which follows closely the intuitive analysis of the system in the influence diagram. The System Dynamics simulation language DYSMAP has been originally developed at Bradford University. The following is a very simple DYSMAP model:

```
* POPULATION EXPLOSION MODEL
L POP.K=POP.J+DT*NBR.JK
N POP=0.5E11
R NBR.KL=NGF*POP.K
C NGF=0.03
C DT=0.5
C LENGTH=300
C PRTPER=10
PRINT POP,NBR,NGF
D NBR=(PEOPLE/YEAR) NET BIRTH RATE
D NGF=(1/YEAR) NET GROWTH FACTOR
D POP=(PEOPLE) POPULATION
D DT=(YEAR) SOLUTION INTERVAL
D LENGTH=(YEAR) LENGTH OF SIMULATION RUN
D PRTPER=(YEAR) PRINTING INTERVAL
RUN BASIC POPULATION EXPLOSION
```

In this model the rate equation R represents the rate at which the system will change during the next solution interval. The constant variable, NGF, reflects managerial policy options and the level equation, L, is the DYSMAP equivalent of an integral equation describing the state of the system at any instant in time. In this example we have a complete set of dimension statements and the model is dimensionally correct. The N statement gives the initial value of the level variable, POP. Sophisticated DYSMAP models might contain auxiliary, supplementary and table equations and they would certainly make use of some of the fifty or so functions which are available as part of the DYSMAP syntax. The user can employ limiting functions, time related functions, delay functions, table functions, random number functions and mathematical functions. DYNAMO and DYSMAP are the two main computer implementations of a simulation language for System Dynamics modelling. DYSMAP has been widely used since its original implementation started in 1972 [4]. This package consisted of a suite of FORTRAN programs which included a program to read the source model and translate it into a FORTRAN program, a loop analyser, documentor, a dimensional analyser and a library of functions. While this package performed well enough in the era of batch processing, it was inherently clumsy (so far as it relied on the resident FORTRAN compiler and linker). In view of the expertise in compiler writing available at Salford University [1,2] it was felt that DYSMAP users would benefit from a complete re-write of this implementation and DYSMAP2 is a completely new package written with current trends in software design very much in mind.

PARSING AND INTERPRETING

The structure of the DYSMAP language is such that the label of a line determines the type of equation one is dealing with. The parsing process converts the expression on the R.H.S. into an encoded Reverse Polish Strip. The encoding caters for operators, functions, variables and numbers. For example, the equation

L HOUSES.K=HOUSES.J+DT*(HCR.JK-SALES.JK)

will be converted into the following Reverse Polish Strip (RPS):

HOUSES DT HCR SALES - * +

The hash table entry for the L.H.S. variable, HOUSES in this example, contains a pointer to the start of the strip, which is terminated by zero. This encoded representation is interpreted at run time and a small stack is used to hold the intermediate results. Since the aim of the project has been to keep to standard FORTRAN 77 it is not feasible to trap execution errors as they happen and then recover gracefully. This has been overcome by including various checks in the Interpreter to trap overflows, underflows and disc full conditions BEFORE they happen.

INTERPRETATION VS. TRANSLATION INTO FORTRAN

A programming team faced with the task of implementing a modelling language such as DYSMAP on the computer would have three approaches open to them. These are:

1) translation into FORTRAN i.e. 'old' DYSMAP package strategy

2) interpreting i.e. DYSMAP2 approach

3) translation into machine code

Considering the relative merits of the three approaches it can be said immediately that (3) may well be the optimal implementation, but its great disadvantage is that it is very machine specific. One would expect (2) and (3) to share code up to the stage where Reverse Polish Strips are encoded based on the model expressions. After that, (3) would convert these RPS's into machine instructions and it is this part of the package that would have to be implemented every time the package was moved to a different machine. One can summarise the comparisons in the following table:

	Method 1	Method 2	Method 3
Portability	I	G	B
Speed of execution	G	I	G
Speed of overheads	B	G	G
Diagnostics: compile time	I	G	G
Diagnostics: run time	B	G	B
Interactive flexibility	B	G	I

TABLE 1. Comparison of features

(G = good, I = indifferent and B = bad)

Generally one would find that with method (1) if any compile time diagnostics leaked into the FORTRAN stage then they would not relate directly to the model and confusing error messages would appear.

THE INTERACTIVE ENVIRONMENT OF DYSMAP2

Throughout the development of DYSMAP2 the emphasis has been on providing extensions to the modelling language in keeping with current trends in user friendliness. The consequence of this philosophy is that there is a basic DYSMAP model in which most of the old 'control' statements are redundant. The user then decides what options to 'switch on' during his/her particular interactive session at the terminal. In particular, the command line could contain, for example,
DYSMAP2 <model> -I -NODIM
where -I requests entry into the Interactive Environment at the end of processing and -NODIM specifies that Dimensional Analysis should not be performed (this could be useful when user is in early stages of investigation and the dimensionality of variables is not finalised). Other options can be specified on the command line. In the Interactive Environment the user has the option of plotting up to six model variables and all the necessary scaling is performed automatically. A co-plot displays a single variable over a number of different runs and a scatter-plot is a display of two variables as an X,Y pair. This overrides the default plotting assumption of X=TIME and is particularly

useful for ascertaining which portion of a table function has been employed during a simulation run. Changing any model parameter and, in fact, adding fresh variables and model equations is also available as an interactive facility, as well as the re-running of the model. Any optimising work on a DYSMAP model would be performed from within the Interactive Environment. The ease of the implementation of the Interactive Environment has been due to the general design of the DYSMAP2 package.

DIMENSIONAL ANALYSER

It has been recognised [3,5] that a very useful check on the correctness of a DYSMAP model is obtained by confirming that it is dimensionally sound. In fact, dimensional analysis can also be of service in model development. In order to invoke the automatic Dimensional Analyser feature of DYSMAP2 all the variables and constants must be declared on a D statement with their correct dimensions. Thus,

```
D FCPD=(UNITS/DEER*YRS) FOOD CONSUMPTION PER DEER
D TIWC=($/M) TARGET RATE OF GROWTH IN WORKING CAPITAL
```

are examples of valid D statements. When DYSMAP2 processes D statements in a model it treats every unit name as being distinct. Therefore, DYSMAP2 sees no connection between MONTHS,YEARS and YRS, for example, and units such as 1/YEAR and ONE/YEAR are treated as distinct. Provided the modeller is consistent, however, he/she can choose his/her own unit names. The associativity rules in dimension statements must be those of FORTRAN, i.e. standard ones. Rules governing the dimensions of functions and their arguments are clearly spelled out by Cavana & Coyle [6] and those have been implemented with some 'softening' to take into account common usage. The process of dimensional analysis is very much analogous to the process of numerical interpretation. In both cases we are scanning the Reverse Polish Strips representing the original expressions. At the end of the scan we must have the same powers of units on both sides of the equation, otherwise the Dimensional Analyser outputs an error message.

PERFORMANCE

The performance of 'old' DYSMAP and DYSMAP2 on a supermini, PRIME 9955, can be illustrated by the following table of execution speeds (CPU time in seconds):

Model name	LENGTH	DT	DYSMAP2	DYSMAP
MODEL 1	24	1.0	0.7	16.2
MODEL 2	50	0.0625	2.2	17.3
MODEL 3	60	0.25	1.9	16.5
MODEL 3	600	0.25	13.9	22.3
MODEL 4	125	0.125	5.1	20.5
MODEL 5	600	1.0	2.9	17.4
MODEL 6	100	0.0625	13.8	20.3

Table 2. Comparison of execution speeds

The table speaks for itself as far the superiority of DYSMAP2 is concerned. However, we can go a stage further and try to analyse the respective contributions to running time from

1) overheads (a)

2) actual evaluation of the model (b)

Writing the time to process a DYSMAP model as

$$T = a + b * LENGTH \qquad (1)$$

and using MODEL 3 with different LENGTH's, the following figures emerge

	a	b
DYSMAP2	0.6	0.02
'old' DYSMAP	15.9	0.01

TABLE 3. Values of coefficients

As one would expect this set of results confirms that the overheads in executing a model are much smaller with DYSMAP2. Somewhat surprisingly the cost of interpretation in DYSMAP2 seems only a factor of 2 worse than the cost of executing compiled FORTRAN code as in 'old' DYSMAP'. The figure for b reflects the fact that most time is spent printing out tables of results.

CONCLUSION

While it has not been possible to deal in great detail with the

implementation of DYSMAP2 it is hoped that the reader will have gained some feel for the working of this package. Friendly diagnostics and the interactive flexibility of DYSMAP2 should encourage experimentation with existing models and the development of new ones.

REFERENCES

[1] Bailey, D., (June 1985). The University of Salford LISP/PROLOG system. Software Practice & Experience, Vol 15 (6), pp 595-609.
[2] Bailey, D., Vallance, D.M.,Ross N., University of Salford FTN77. Reference Manual.
[3] Ballico-Lay,B. A Dimensional Analyser for System Dynamics Models. Dynamica 3 (1).
[4] Ratnatunga, A. K., (1980). DYSMAP User Manual. University of Bradford.
[5] Coyle, R. G., (1977). Management System Dynamics. Wiley, New York
[6] Cavana, R.Y. and Coyle R.G., (1982). DYSMAP User Manual. University of Bradford.