SYSTEM DYNAMICS SIMULATION LANGUAGE--- DYNAMOC

Han Jiuqiang, Sun Guoji, Wu Biao

Institute of Systems Engineering
Xi'an Jiaotong University
Xi'an, P.R. of China

Abstract

This paper introduces a system dynamics simulation language---DYNAMOC developed by authers. In the paper, we first focus on discussing the program structure, software functions and features . Secondly, a simulation example is provided to illustrate our DYNAMOC.

Introduction

Since system dynamics was found by Prof.J.W. Forrester in 1956 at MIT, it has been extensively used for the study on society, economics, population and environment system etc.. At the same time, system dynamics simulation language was also developed as a supporting software. MICRO-DYNAMO is an earliest SD(system dynamics) simulation language that can be run on microcomputer IBM-PC/XT.Although it has played an important role in the research and application for SD method,there also exists some disadvantages, such as, limited model size,sigle integration rule,low compiling and running speed and lack of interactive capacity etc.. In recent years, because of popularization for microcomputer and enhancement of its performance, developing SD simulation language with high performance will certainly promote the study and application of this technique.In order to overcome all those shortcomings in MICRO-DYNAMO, and meet the needs of simulating many large-scale systems in society economics, we have developed the SD simulation software DYNAMOC successfully. DYNAMOC's performance is superior to MICRO-DYNAMO in many aspects, such as,fast compiling and running speed,friendly interactive,various integration methods and larger model size etc.. Since 1989,DYNAMOC has been extensively used in over sixty universities and scientific research organizations in our country. Many users think DYNAMOC to be a typical SD simulation software on microcomputer IBM-PC/XT,AT and compatible computer, in china.

Functions and Features

1. Model Size

In DYNAMOC, SD model size is only limited by memory space in computer because of using dynamic memory allocation technology. DYNAMOC can simulate SD model with 500 equations on computer IBM-PC/XT with 312kb memory space. And when computer's memory spaces increase by 64kb each time, the model size to be simulated can increase by 400 -- 500 equations.

2. Integration Method

DYNAMOC provides three kinds of integration rules included Euler, fourth order Runge-Kutta and changable step. These integration methods can entirely meet the needs of different simulation accuracy and speed.

3. Equation and Function

DYNAMOC can solve all kinds of equations and functions stipulated in standard DYNAMO language.In addition, the independent variable of table function may be both equal and unequal interval, and it provides relative table functions too.

4. Operation Sign

Standard DYNAMO language only provides arithmetic operation. But DYNAMOC can solve both arithmetic and logic operation.

5. File Management

User can directly use most of commands provided by DOS(Disk Operation System) during DYNAMOC operation, such as,edit text,print and copy file and so on.

6. Interactive

Because of using menu technique full and providing large scale assistance operation informations,when user only enters a few commands, simulation process may be completed.During simulation run, user can pause the operation of DYNAMOC at any time,and modify parameter and display data etc.. After simulation end,user can also modify parameter and rerun,but it is unnecessary to recompile model.

7. Parameter Optimization

DYNAMOC provides parameter optimization function that can optimize five mode parameters each time. After user enters the parameters to be opyimized and objective function according to the grammmar rule of DYNAMOC ,the parameters can be optimized immetiately.

8. Output Mode

DYNAMOC provides many kinds of output modes such as data print,screen graph with low and medium precision,low precision graph file and plotting on SR-6602 plotter.These output functions provide great convenience to user.

9. Support Environment

DYNAMOC is programmed in C and assembler language and can run on microcomputer IBM-PC/XT,AT and compatible computer. In order to modify and regulate source program easily and maintain the software conveniently, the whole program consists of seven main modules.

Overall Structure

Overall structure of the software program is shown in Fig.1. The system controller,together with file management, model compiler,parameter modifying, result output,error diagnosis and model run module are combined into the overall structure of the software program. We introduce the program structure and function of every module respectively.

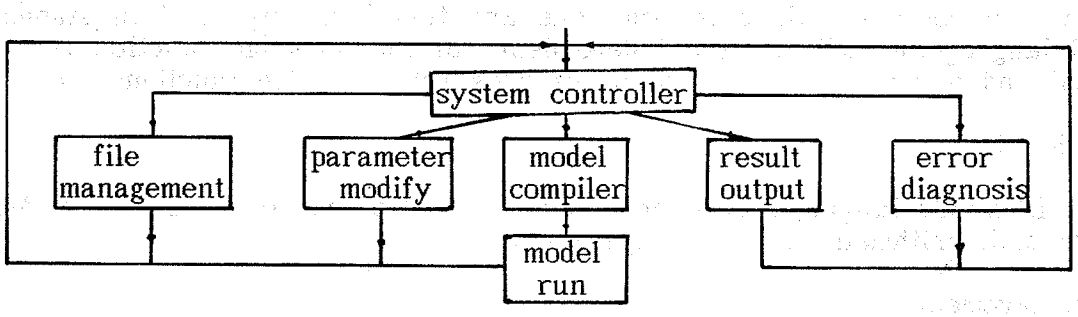

Fig. 1. DYNAMOC Overall Structure

1. System Controller Module

System controller harmonizes the operation of every module in overall structure,controls the system run and manages the operation menu of the whole system according to user's choice.

2. File Management Module

In DYNAMOC operation process, when user needs to operate file such as Edit, Copy and Print and so on,they have to pause simulation operation and exit DYNAMOC. In this way, the simulation data generated in run is easily lost, and this type of operation is very unconvenient too. Hence, file management module accomplishes most of functions in DOS.However,it is not a true operation system, but only uses full various functions provided by operation system,its whole program is written in essembler language

3. Compiler Module

Compiler module is a nucleus in DYNAMOC, its performance affects directly the speed of compiling and running model. In order to enable DYNAMOC's compiler program to be independent of a special computer, we use a medium language that is a executable code generated by compiling source program. The design thought of the medium language is shown in Fig. 2.

(1) Source program
What is called source program means SD simulation program written by user according to DYNAMOC's grammar regulation. For instance,Eq(1) is a SD simulation source program, and it is sometimes called a SD equation.

R RATE.KL=A.K*B.K+C.K (1)

(2). Grammar analysis and pseudo-machine code
Grammar analysis is that checks whether source program accords with grammar regulation or not. Pseudo-machine code is a medium code generated by analysing and treating source program. In grammar analysis process, compiler dispatch program reads in a SD equation (source program) from SD simulation model program each time, and checks its type and name. When the equation name and its type accord with grammar regulation, dynamic memory allocation program will build a equation tree for the equation. The structure of equation tree uses a two-intersection tree structure that uses structure name as node. When souce program exists grammar wrong,error diagnosis will display error type and modifying information.After the equation name is treated, expression

treatment program will analyse and treat the expression at right side in the equation.Because the expression includes either arithmetic operation or logic operation , or both, the analysis algorithm of expression uses a operation sign priority rule of expression-oriented. During expression treatment , contant, variable and function are treated by a semantics analysis program. First, the semantics analysis program takes a operation symbol and relative operand from the expression in order,through analysis and treatment, and yields a operation instruction that is called medium (pseudo-machine) code, and,at the same time.also builds a same medium code tree with equation tree structure . So that variable check and the generation of optimization objective code can be done in late.Next,the operands that can not temporarily become medium code are temporarily pushed in a operand stack.After the whole equation is treated, compiler stipatch program will treat next equation continually. In this way, until meeting RUN equation or the source program end, grammar analysis of the source progarm for a model will be completed and all pseudo-machine codes generated too.

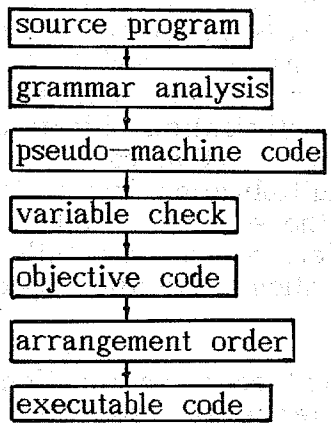

Fig.2. Flow chart of medium language

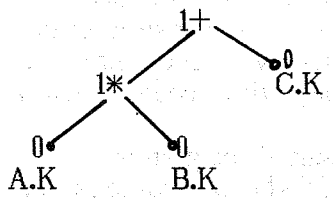

Fig.3. medium code tree

For instance,in treatment Eq(1) process, the semantics analysis program first generates two instructions stored A.K and B.K variable, and pushes the two address pointers stored the two temporary instructions in operation stack. Next, it generate a multiplication operation instruction. In order to generate medium code instruction for the multiplication operation, the two address pointers kept A.K and B.K variable in operation stack are poped from the stack , and, generates into a medium code instruction for A.K*B.K . But the medium code instruction is as temporary operation instruction in generation addition instruction, and its address pointer will be pushed in the stack , then generates another instruction stored C.K variable. Last,it genertes a addtion instruction, its two operands are the address pointer in the stack and the instruct stored C.K respectively .At this moment, the address pointer of the addtion instruction is only pushed in the stack, and it is the root of medium code tree built by A.K*B.k+C.k experssion. The medium code tree structure is shown in Fig. 3.

From Fig.3.,the mark of every variable is zero. If there is a operand in a operation instruction, the mark of the instruction equals that the mark of the operand adds 1. If there are two operands, and their mark is same, the mark of the instruction is that the mark of operand adds 1, else its mark takes a bigger mark in two operand marks.

(2) Variable check
After medium code tree for all equations are correctly generated, variable check starts. As mentioned above, because DYNAMOC's grammar stipulate that undefined variable can be first used, it is nesessary to check whether all variables at right side of a equation appear at left side of the equation or not, and whether their subscripts accord with grammar regulation or not. In variable check, variable check program first searches those medium code instructions stored variables. and seeks same name equation by looking for relative equation tree according to the variable name found already, Then will obtain the permitable subscript of the variable by looking for variable table according to the equation and variable type found already, and compares permitable subscript with actual subscript, if both subscripts are same, the variable has been defined, else will search continuosly

(3) Objective code
Generation mode for objective code instruction is same with that of pseudo-machine code instruction.As the interpreter program written in C language executes objective code instruction, the medium code instruction passed variable check should be transformed into objective code instruction,and in transformation process , operands may be both immediate number, register, and addess pointer. Because of using the generation algorithm of optimization objective code tree and generating objective code chain by producing objective code of medium code tree ,the operation for a*b only generates a multiplication instruction, but does not generate transmission data instruction, and the equation can be calculated as long as objective code chain is executed in order. so that can both shorten the length of objective code chain and enhance equation's execution speed largely.

(4) Arrangement sequence
Because DYNAMOC provides automatic arrangement sequence function, user does not need to consider the programming sequence for source program. In DYNAMOC, there are four kinds of equations that need to arrange sequence. They are R,A,N and S equation.Arrangement sequence is made by scanning objective code equation tree.After all equation trees are scanned once, if the equation to be arranged order can not be found in the equations of unarrangement sequence ,this arrangement order is failure,otherwise, it is successful. In arranement order, the arrangement sequence program arranges same kind of equations at continuous place in sequence table, and every type of equation has start address and number.Only after all equations are arranged sequence successfuly , the simulation for the model can start.

4. Run Module

The program structure diagram for run module is shown in Fig.4.

(1) Run initiating
Run initiating program is used to set the initial value of zero for every equation except C(Contant),B(train),SPEC and T(table) equation,and keeps the initial values of train and SPEC variable so that are used when rerun.

(2) Equation calculation
The calculation program for every kind of equation is almost similar. For instance, the calculation program for R(rate) and A(assistance) equation written in C language is as follows.

```
run_raequt()
    {
    for( i=0; i<racount;i++)
        {curt_equid=*(sorted_order+rastart+i);
        runobj=curt_equid->root.obj_root;
        do{
            (*runobj->func_name) ();
              runobj=runobj->next;
          }
        while (runobj!=NULL);
            curt_equid->value=reqist[16];
        }
    }
```

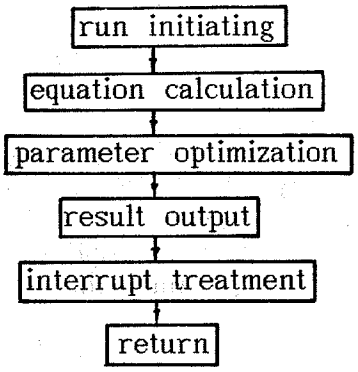

Fig.4. run structure diagram

Where,

racount: the number of R and A equation
rastart: the number of the equations between the start point for R and A equation and the table head in calculation sequence table
sorted_order: the start address in calculation sequence table
curt_equid: address of equation tree
runobj: the address of objective code instruct
func_name: threaded code of the instruct to be executed

For L(level) equation, its calculation program is almost same too. We know that L equation is a difference form of differential equation,and because DYNAMOC provides many kind of integration rules,in solving process, L equation have to have one of two forms as follows.

L LEVEL.K=LEVEL.J+DT*..... (2)

or L LEVEL.K=LEVEL.J+(DT)(.....) (3)

In Eq(2) and Eq(3),the part behind * or in brackets is the right part of a differential equation. Under restriction above, differential equation can be solved by any existent integration rule. In fact, main difference between L and A or R equation only consists in their simulation time.

(3) Parameter optimization

Parameter optimization uses DFP optimization algorithm that is sometimes called fastest fall rule. supposing: $Q(\underline{\alpha})$ represents targage function, and $\underline{\alpha}$ represents parameter to be optimized, then

$$-\nabla Q = -\frac{\partial Q}{\partial \alpha}\Big|_{\alpha=\alpha_k} \qquad (4)$$

Where,∇Q represents $Q(\underline{\alpha})$'s ladder agree when $\underline{\alpha}=\underline{\alpha}_k$. If h_k represents search step, the parameter $\underline{\alpha}_{k+1}$ can be calculated by using following formula.

$$\underline{\alpha}_{k+1}=\underline{\alpha}_k - h_k^* \nabla Q(\underline{\alpha}_k) \qquad (5)$$

Where, h_k^* represents optimization step, and it can be obtained by gold severing method. If the optimal region of step is between A_0 and B_0, and $H_k = A_0-B_0$, then the values on H_{k1} and H_{k2} point can be calculated by gold severing way as follows.

$$H_{k1} = 0.382 * H_k + A_0 \quad (6)$$

$$H_{k2} = 0.618 * H_k + B_0 \quad (7)$$

Source function can be overlaply calculated once by using Eq(5) and regarding H_{k1} and H_{k2} as step respectively, and obtain relative objective value: D(2) and D(3). If D(2)<D(3), next severing region is (A_0, H_{k2}). If D(2)>D(3), next severing region is (H_{k1}, B_0). So repeated severing, until $H_k < E_0$(smallest severing region), optimization step will be found as Eq(8). If it is nesessary to calculate Q value, the whole model should be run once.

$$h_k^* = \frac{A_0 + B_0}{2} \quad (8)$$

5. Parameter modifying module

During simulation run,not only can modify any parameter in model and all system variables, such as constant, run time and output variable, but also change the output mode of each output variable, but not change integration rule. If have changed the output mode for some variables, and will output the variables by new output mode, it is nesessary to rerun. However, rerun does not need to compile source program because parameter modifying only changes objective code. In this way,it enhances the speed of the whole system run largely.

6. Output module

Output part is independent of run and compiler part, so it is very flexible. In simulation process, because DYNAMOC does not limit the number of print points and the number of output variables,for large-scale system, it will generate large scale data so that there is not the space to store the data. In order to sesolve the contradiction between large scale data saving and fast calling in data, database is used in output module.

Example

We simulated many SD models,such as world model (WORLD DYNAMICS W5 and W2) and the planning model for some area(over 1200 equations) and so on. For example, we simulated the mode of researching the organisms environment relationship between sea grass and limps in near sea.Its SD flow chart is shown in Fig.6.

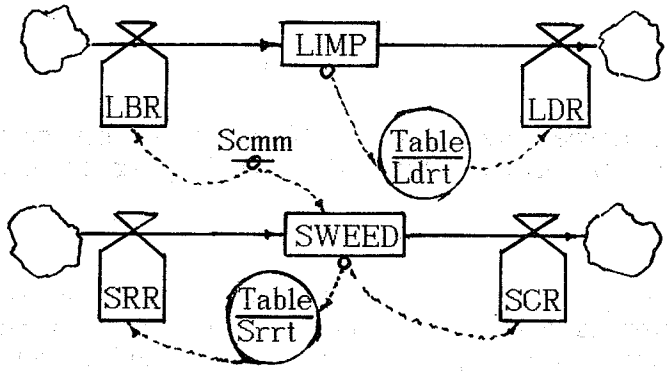

Fig.6. Flow chart of SD

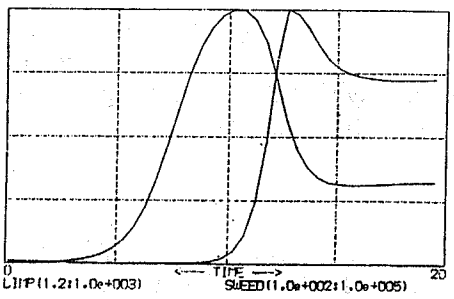

Fig.7. simulation result

Its SD simulation source program is as follows:

```
L       SWEED.K=SWEED.J=DT*(SRR.JK-SCR.JK)
N       SWEED=100
R       SRR.KL=TABLE.(SRRT,SWEED.K,0,125000,25000)
T       SRRT=0/22000/30000/27000/10000/-20000
L       LIMP.K=LIMP.J+DT*(LBR.JK-LDR.JK)
N       LIMP=10
R       LDR.KL=TABLE(LDRT,LIMP.K,0,1250,250)
T       LDRT=0/100/250/500/1000/10000
R       SCR.KL=SCRM*SWEED.K*LIMP.K
R       LBR.KL=LBRM*SCRM*SWEED.K*LIMP.K
C       SCRM=0.001
C       LBRM=0.02
PRINT   LIMP,SWEED
SPEC    DT=0.1, LENGTH=20, PRINTER=0.5
PLOT    LIMP,SWEED
END
```

Simulation result ploted by ploter is shown in Fig.7.

Concluusion

DYNAMOC is a SD simulation language suitable for microcomputer, and it has been extensively used in practical system. Many users think that DYNAMOC's performence is superior to MICRO-DYNAMO in many aspects. Its extension will certainly promote the study and application of system dynamics simulation technique in many fields.

Refference

1. J.W. Forrester, Principles of Systems Dynamics, MIT. Press,1986.
2. User Guide and Reference Manual for Micro-DYNAMO System Dynamics Modelling Language, IBM-PC Version,by Addisson Weslay Publishing Company Inc,1983.
3. J.J.UHRAN, JR. AND W.I. DAVISSON: The Structure of NDTRAN---- A System Simulation Language, IEEE TRANSACTIONS ON SYSTEM VOL SMC-14 NO.6 1984.
4. J.B.PHILLIPS,M.F.BURKE AND G.S WILSON. Threaded Code for Laboratory Computers, SOFTWARE-PRACTICE AND EXPERIENCE, VOL.8, PP257-263,1978.
5. A.V.AHO AND S.C.JOHNSON: Optimal Code Generation for Expression Trees. Journal of the Association for Computing Machinery vol.23. NO.3 pp 488-507, July 1976.