SYSTEM DYNAMICS BY VISICALC
Simulation Models Using the New Spreadsheet Programs

Alan McK Shorb
Micro Dynamics
58 Northgate Road
Wellesley, MA 02181

ABSTRACT

Visicalc, the original spreadsheet program for microcomputers is explored for suitability as a vehicle for system dynamics models.

It is shown that Visicalc can support most of the model features that DYNAMO allows, but at the price of some inconvenience and care needed when setting up the model. However, with the widespread distribution of spreadsheet programs among decision makers, there may be situations where they may be the vehicle of choice for implementing a dynamic model.

0. WHY SYSTEM DYNAMICS BY VISICALC?

In many situations, organizations wish to develop policies for allocating and controlling resources, and are concerned about the effect of these policies over an extended period of time. As most of the audience of this paper would agree, system dynamics is the technology of choice for developing these policies, and computer simulation is an essential element of this technology. Often, however, these organizations don't have access to the larger computers that support DYNAMO, but may have access to today's smaller microcomputers.

The most successful single piece of software written for microcomputers is Visicalc (TM), a spreadsheet program. As far as I know, Visicalc or one of its emulators, is available on every single microcomputer for sale today, and probably has been purchased by most owners of microcomputers outside the home. Spreadsheet programs are designed to be simulation aids for budgeting and planning, but I have not yet seen them used within the context of the system dynamics technology.

The main reason I am writing this paper is that I wondered if the spreadsheet programs could be used for making system dynamics models. I tried it, found they could be, and wanted to let other system dynamicists know about it. But there are other reasons than mere curiosity why the ability to construct models with spreadsheet programs is of interest to system dynamicists. For one thing, the fact that spreadsheet programs are available on a wide variety of microcomputers allows simulation models to be easily written for any of them without having to rely on special purpose languages like DYNAMO, which may not be available on the computer at hand. A related advantage is that since the spreadsheet programs are similar, a model developed on one computer can be easily transferred to other computers. This ability is particularly important when a prototype model is developed on one computer to be implemented on a variety of other computers owned by the ultimate users of the model. If the user is interested in modifying the model, he may not want to go to the trouble of learning the formalism of DYNAMO, even if it were

available, but may be quite willing to use his knowledge of the spreadsheet program he has already learned to use. Even if the model is not to be altered by the user, if it is written in a spreadsheet format it can be quickly and easily distributed to its intended audience.

This paper presents one way that spreadsheet programs can be used to construct system dynamics models. After a brief overview of the features of spreadshet programs, a sketch of the issues to be dealt with is made. Then three of these issues are dealt with in greater depth: how the correct computational sequence is assured; how time is handled; and how table lookups are accomplished. Finally, a fragment of a resource management model is presented, first in DYNAMO, then as a spreadsheet program. The paper then closes with a brief discussion of the merits and shortcomings of this approach.

1. OVERVIEW OF SPREADSHEET PROGRAMS

Spreadsheet programs utilize a two-dimensional rectangular array to analyze problems of interest. Each position in this array can be either a string of characters, an expliticly assigned numerical value or a formula. A formula has two different aspects: the algebraic expression for the formula, and the value calculated by the formula. The algebraic expression can involve the combination of values from other positions by means of the four usual arithmetic operations, as well as special functions such as maximum, extended sum, and table lookup. In these algebraic equations, special expressions are used to

indicate positions on the spreadsheet array. The most popular method is to label the rows of the array by numbers and the columns by letters or pairs of letters. Then the expression indicating the value for a position is just the column designator followed by the row designator. Thus the expression "B6" indicates the value at column B, row 6 on the spreadsheet, while the expression "AC18" indicates the value in column AC and row 18.

In a wide class of spreadsheet program applications, the array is aranged with different rows representing different variables of interest, and different columns representing the values at different times. This general arrangement will be used in the discussions in this paper. More specifically, in the terminology of DYNAMO, the rows will represent levels, rates, auxiliaries and constants, and successive columns will represent their values at instants in TIME that are DT apart. In addition, some special areas in the array will be set aside for table function values and for computations to assure the correctness of the computation order.

The capabilities of the spreadsheet programs that allow this approach to modeling to work are the following:

The order of computation can be either row-dominant or column-dominant. In column-dominant computations, the values for each element in a column are computed, from top to bottom, then

the computation proceeds to the next column. Thus for the format described above, with column-dominant computations, all the variables for one instant are computed, and then all the variables for the instant DT later are calculated.

Entire rows of formulas can be moved around easily. This feature greatly aids in getting the computations to be performed in the correct order.

Formulas can be easily duplicated across columns or rows, with only the appropriate changes in the subscripts. This feature is what allows the same formula relating variables to be used for all instants during the simulation.

2. ISSUES TO BE HANDLED

In order for a computational technique to successfully support system dynamics models, a number of technical issues must be dealt with.

Two of the most important of these issues are: How to assure that the computation of variables at a given timestep is in the correct order; and how to assure that the correct values are passed from one timestep to the next for the entire duration of the simulation. Another issue which is nearly as important, if not as fundamental, is how to provide for table lookups within the model. Each of these three issues is dealt with in turn in the sections below.

There are other modelling issues that can be handled within the spreadsheet program or by easy-to-write companion programs. The most important of these are PLOTTING model results, using random functions, computed initialization of levels, and boxcars. None of these issues is further discussed in this paper.

3. COMPUTATIONAL ORDER

The task of assuring the correct computational order in a system dynamics model is a two-fold one. First, it must be assured that there are no cyclical causal paths in the model without alternation of levels and rates along these paths. Second, it must be assured that the order of computation follows the order of causation. That is, no auxiliary or rate variable should be computed for a given timestep before all the levels and auxiliaries on which it depends are computed for that timestep.

3.1 Levels and Rates

As the first step toward assuring the correct computation sequence, all level variables will be computed before all auxiliary variables, which will be computed before all the rate variables. In addition, as a technical convenience to be explained later, all constants other than level initializers will be computed anew at each timestep. Thus the rows of the spreadsheet will be arranged as in the diagram below.

Constants
Levels
Auxiliaries
Rates

The order described above will cause no problem for the proper computation of level variables, since they only depend on constants and on variables from a previous timestep. Also this order will cause no problems in the computation of rates, since the should only depend on the values of constants and on levels and auxiliaries from the current timestep.

3.2 Auxiliaries

The only problem that could occur is in the order of computation of auxiliary variables. The correct order of computation is accomplished in two steps. First, it is assured that there are no causal cycles among the auxiliary variables. Then, the rows corresponding to the variables are rearranged one at a time until the computation is in the correct order. The same computational device is used in both these steps. Namely, a few columns are set aside as scratch space for keeping track of the order of computation. The spreadsheet array will then be organized as shown below.

Scratch	Constants
	Levels
	Auxiliaries
	Rates

In the first scratch column, the maximum number of causal steps from a level to that variable is computed. This computation is accomplished as follows. All values in this column are initially set to zero. Then for each auxiliary and rate variable, the value in this column is set to one more than the maximum value in this column of the variables that have a causal influence on it. (A simple 0,1 switch is used to distinguish between the two different ways of setting the values in the first scratch column. 0,1 switches are discused in detail in the section below concerned with initialization of level variables.)

The second scratch column is used to calculate the maximum value in the first scratch column. For each auxiliary and rate variable the value in this column is set equal to the maximum of the value in the first scratch column for that row and the value in the second scratch column for the previous row.

The values in these two columns are calculated repeatedly, all the while observing the behavior of the last value in the second column. One of two things will happen. Either this value will be the same for two successive calculations, or it will

continue to increase with each new calculation. In the first case, the number that ends up in the first scratch column for each variable is the maximum number of causal steps from a level to that variable. The rows can then be arranged as explained several paragraphs hence.

If the first case occurs, it must occur before the last value in the second scratch column exceeds the total number of non-level variables. Otherwise, the second case holds, and there is a causal cycle of variables with no intervening level variable. To find the cycles, scan up the second scratch column to find the first row in which this last value occurs. (That is, if there are 12 auxiliary and rate variables and the last value in the second column has reached 14, then scan up the second column to find the first occurence of 14 in that column.) The variable associated with this row is in a causal cycle.

To find the other variables in the cycle(s), trace back through the cycle as follows. At each variable, find the variable(s) that this variable depends on which has(have) values in the first column greater than or equal to one less than the first column value of this variable. There will be at least one such variable. If (any of) the new variable(s) has(have) been encountered before in this search, a cycle has been found. Otherwise perform the same procedure on each of the new variables found, repeating until all variables found have been investigated in this manner. This process must find all the cycles that

contain the variable started with, because there are not enough non-level variables in the model to trace back to a level variable by the procedure described, yet the procedure will always lead to either a new variable or one that has been encountered earlier in the search.

After any cycles have been eliminated, then the model builder must assure that the equations are in the correct computational order. That is, for every auxiliary and rate variable, every other variable it depends on must have already been computed on an earlier row. There are several strategies that can be used to assure that the variables are in the correct order. The easiest of these strategies involves the values in the first scratch column described above. Merely move the rows for auxiliary variables around until this value is in ascending order. One acceptable strategy, if there are only a few auxiliary variables, is the reordering of rows done by inspection.

When variables are found to be out of order, they can be rearranged using the editing features of the particular spreadsheet program being used. If the spreadsheet program allows rows to be moved around readily, keeping track of the proper row references in its equations, then the rows should be rearranged using whatever strategy is preferred until they are in the proper order according to the values of the first scratch column.

If, as is the case with most spreadsheet programs, the editing facilities are limited to adding blank rows and copying one row onto another, more care needs to be taken. The row that is to be moved is first copied onto a blank row that has been created in the proper position. Then references to that row in the other equations of the spreadsheet array must be adjusted. A process using auxiliary columns reminiscent of the process for eliminating cycles can be used to check on this adjustment.

4. HANDLING TIME

Spreadsheet programs are well designed to handle time in a system dynamics model, but there are a few technical details that merit close attention.

The general arrangement of the spreadsheet array is to let each column of the model portion of the array represent one timestep, with time increasing to the right. The level variables are toward the top of the spreadsheet, with TIME being the topmost of these level variables. The auxiliary and rate variables come below the level variables, as already described. Then, at any given time step, the computations for the level variables use values from the previous column, and the computations for auxiliary and rate variables use values higher up in the same column.

The three issues in handling time that need to be discussed in more detail are: how to do long simulations when there are a

limited number of columns available for computation; how to handle the initialization of the level variables, and how to reference the correct positions at each time step.

4.1 Handling Long Simulations

Most spreadsheet programs are limited to less than a hundred columns, and for any sizable application, the limitations on computer memory are reached long before the formal limit on the number of columns. On the other hand, a typical system dynamics model will use well over a hundred timesteps. The problem is how to use the columns to represent timesteps without either exceding the limitations of the spreadsheet program, or putting undesirable limitations on the characteristics of the simulation.

The solution is to run the simulation only for a convenient number of time steps at once, and then to "wrap around" the computations, going back to the first timestep column and continuing the simulation.

For example, consider a simulation with 5 timesteps per month which is required to display results quarterly over a four year period. That is a total of 241 timesteps (including both beginning and end) that must be computed, well beyond the capability of most spreadsheet programs. This simulation can be handled by doing the simulation a quarter at a time. More precisely, there would be sixteen active columns in the

simulation. In the first run of the simulation, these columns would represent values of TIME running from 0 to 3 months in steps of 0.2 months. At the end of this first run, the initial values for the simulation will be displayed in the first active column, and the end-of-quarter values would be displayed in the last active column.

Then the simulation for the second quarter would be run by first placing the values of the level variables from the end of the first quarter back into the first active column, and then proceeding with the computation for one more quarter. At the tnd of these computations, the columns would represent values of TIME running from 3 to 6 months. This process could be continued indefinitely, producing a simulation that lasts as many quarters as desired.

In the general case, after the simulation of the first stretch of time, the values of the level variables at the end of that stretch would be placed back into the first active column, and the simulation repeated as often as necessary.

The placement of the level variables back into the first active column would be controlled by a simple zero-one switch parameter under the control of the person doing the simulation. At the start of the simulation the switch would be set to zero and would select the initial values for the levels and put them into the first column. Then the switch would be set to one, and

in subsequent runs it would select values of the level variables at the end of the previous simulation, and place them into the first active column. The formulation of this zero-one switch is described in the next section.

4.2 Initializing Levels

As described above, the initialization of level variables for a simulation run depends on whether the run is the first run of the overall simulation, or is a continuation of previous runs. This initialization is done in the first active column of the level variable rows by spreadsheet equations corresponding to the psuedo-DYNAMO equation:

LEVELN=(1-SWITCH)*LEVEL0+SWITCH*LEVELF,

where LEVELN is the value of the level variable selected to start the current run, LEVEL0 is the initial value of the level for the total simulation, LEVELF is the value of the level at the end of the previous run of the simulation (active column 16 in the earlier example), and SWITCH is either 0 or 1. The initial values of the level variables should be placed in a convenient location, such as in the column just before the first active column.

4.3 Equations at Each Time Step

Unlike in DYNAMO, the equations of a spreadsheet representation of a simulation model cannot be written once for each variable, and then forgotten. For each variable they must be reproduced for each timestep. Fortunately, the editing

capabilities of most spreadsheet programs ease this task considerably.

First, the equations for the variables and the values for the constants must be written into the spreadsheet in an "initial" position. For level variables, the initial equations must go into the second active column, written in terms of auxiliary and rate variables from the first active column. (The selection equations for initializing the variables are in the first active column.) For auxiliary and rate variables, the initial equations go into the first active column. In addition, values for constants, other than those initializing level variables, should go into the first active column. The reason for this will be seen shortly.

Then each of these equations should be replicated along its row, all the way to the final active column. In this replication the "relative" attribute should be applied to all position references in these equations (except for table functions, as discussed later). This attribute ensures that all references to a position move along the row with the position for which the equation is being written, so that levels are always computed in terms of variables from the previous timestep, etc.

Since there is no way for the spreadsheet program to automatically distinguish between constants and variables, the relative replication of equations will generate relative

references to constants as well as variables. In order to relieve the user of the burden of worrying about this distinction for each equation, it turns out to be easier to replicate the values of the constants (other than level initializers) across all time steps. To do this, with the constant value entered into the first active column, it is only necessary to enter the equation in the second active column saying that the value in that column is equal to the value in the previous column in the same row. Relative replication of these equations through the last active column will assure that the constants are really constant.

Thus the layout for the active, simulating part of the spreadsheet for a system dynamics model will be organized according to the diagram below.

Active Columns		1	2 . . . . . Last
Constants		Values	Initial Equations
Levels	Initial Values	Selection Equations	Initial Equations
Auxiliary Variables		Initial Equations	Replications
Rates		Initial Equations	

5. TABLE LOOKUPS

All spreadsheet programs I am familiar with provide for table lookups by comparing two rows (or columns) of numbers on the spreadsheet. The first of these rows must be listed in ascending order. A key value for the table lookup is given and the program searches along the first row of the table from left to right until it finds the last number in that row that is less than or equal to the given key. The value in the second row just below that value is the value returned by the table lookup function. If the key is less than the first number or greater than the last number in the first row then an error message is given.

Unfortunately, this type of table lookup is not suitable for most system dynamics applications, since an interpolated value is usually needed. However, the interpolated value can be obtained by using a trick. To see how this trick works, we first look more closely at the interpolation formula:

TBVALU=LOVALU+(HIVALU-LOVALU)*(KEY-LOWKEY)/GAP,

where

TBVALU is the desired interpolated table value

KEY is the value given the table lookup function

GAP is the difference between successive values in lst row (assumed constant)

LOWKEY is the largest value in the first table row not greater than KEY

LOVALU is the value returned by the lookup function, ie the value in the second row below LOWKEY

HIVALU is the value in the second table row just after LOVALU and is the value returned for KEY+GAP.

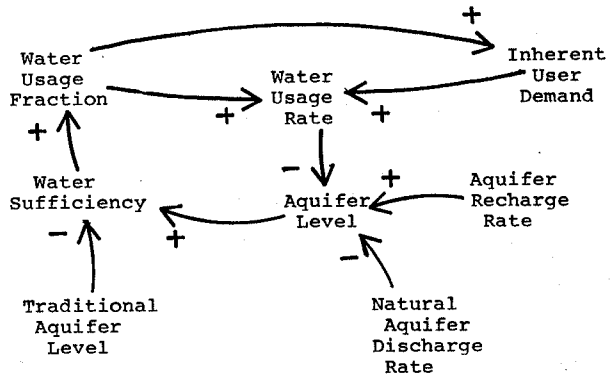
The trick is to duplicate the first row and use it as both the first and second row in a lookup function. Then LOWKEY is the value returned for KEY by this lookup and all the values in this formula can be computed. It is suggested that in actual practice the table lookup using this trick be divided into four steps. The first three steps will be the table lookups for LOWKEY, LOVALU and HIVALU, and the fourth step is the above interpolation formula.

6. EXAMPLE MODEL

As an example of this technique, I have written a system dynamics model using Visicalc, the original spreadsheet program. This model is a simplified representation of the basic interactions between groundwater supply and usage in an area relying heavily on groundwater supplies, like many regions in the Middle East.

6.1 Causal Diagram

This model is based on the causal diagram:



6.2 DYNAMO Version of Model

The DYNAMO equations for the model are displayed below. The parameters were chosen to have a Traditional Aquifer Level of 500 million galons (MG), a Water Usage Rate of 90 MG/YR and an Inherent User Demand of 100 MG/YR.

```
R     WUR.KL=IUD.K*WUF.K
          Water Usage Rate (MG/YR)
L     IUD.K=IUD.J+(DT/DAT)(1-DAS*(1-WUF.J))
          Inherent User Demand (MG/YR)
N     IUD=NIUD
C     NIUD=100
          Initial Inherent User Demand (MG/YR)
C     DAT=4
          Demand Adjustment Time (YRS)
C     DAS=10
          Demand Adjustment Sensitivity (Dimensionless)
```



```
A     WUF.K=TABLE(WUFT,WS.K,0,1,.2)
          Water Usage Fraction (Dimensionless)
T     WUFT=0/.39/.61/.77/.9/1
          Water Usage Fraction Table
A     WS.K=AQLEV.K/TAQLEV
          Water Sufficiency (Dimensionless)
C     TAQLEV=500
          Traditional Aquifer Level (MG)

L     AQLEV.K=AQLEV.J+DT(AQRR.JK-WUR.JK-NAQDR.JK)
          Aquifer Level (MG)
N     AQLEV=NAQLEV
C     NAQLEV=400
          Initial Aquifer Level (MG)
R     AQRR.KL=CAQRR
          Aquifer Recharge Rate (MG/YR)
C     CAQRR=130
          Constant Aquifer Recharge Rate (MG/YR)
R     NAQDR.KL=AQLEV.K/AQRT
          Natural Aquifer Discharge Rate (MG/YR)
C     AQRT=10
          Aquifer Residence Time (YRS)

N     TIME=0
          Time (YRS)
C     DT=.1
          Simulation Increment (YRS)
C     LENGTH=10
          Simulation Length (YRS)
```

The reader is invited to try out this model on his own computer.

6.3 Visicalc Version of Model

The model presented above can be entered into a spreadsheet according to the discussion in earlier sections. The way this can be done is indicated below, using the notation for Visicalc.

In this formulation, since there are only two auxiliary variables, the correct computational order was easily obtained by inspection, so no scratch columns were used. The computation order and the rows the quantities appear on are:

QUANTITY	ROW
DT	2
DAT	3
DAS	4
TAQLEV	5
CAQRR	6
AQRT	7
TIME	9
IUD	10
AQLEV	11
WS	13
(LOVALU for WUF)	14
(HIVALU for WUF)	15
(LOWKEY for WUF)	16
WUF	17
WUR	19
AQRR	20
NAQDR	21

The gaps in the sequence are for purposes of punctuation only. In addition, space was set aside for the table listing in rows 23 through 25, columns C through H, and for GAP in row 26, columns C through M.

There are eleven active columns, to cover an entire year with DT=.1. The first active column is column C, and the last one is column M. The values indicated above for constants were put into column C, rows 2 through 7, and for the level initializing constants in column B, rows 9 through 11.

The initial equations for the constants are in column D. They each say that the value in column D is the same as the value in column C:

Di = Ci, for i from 2 to 7.

The equations used to select the initialization of the level variables were in column C, rows 9 through 11. The value of the switch for this selection was stored in column B, row 3. These equations were of the form:

Ci = (1 - B3) * Bi + B3 * Mi, i from 9 to 11.

For each level, the first active equation appeared in column D. The actual equations for rows 9 to 11 are displayed below, acompanied by their corresponding DYNAMO equation.

```
D9 = C9 + D2
(TIME.K=TIME.J+DT)

D10 = C10 + (D2/D3)*(1 - D4*(1 - C17))
(IUD.K=IUD.J+(DT/DAT)(1-DAS*(1-WUF.J)))

D11 = C11 + D2*(C20 - C19 - C21)
(AQLEV.K=AQLEV.J+DT*(AQRR.JK-WUR.JK-NAQDR.JK))
```

The two auxiliary variable equations are presented below in the same style, but for column C, the first column for these variables:

```
C13 = C11/C5
(WS.K=AQLEV.K/TAQLEV)

C14 = @LOOKUP(C13,C24.H24)
(LOVALU in the formulation in the section on table lookups)

C15 = @LOOKUP (C13+D26,C24.H24)
(HIVALU from earlier section)

C16 = @LOOKUP (C13,C23.H23)
(LOWKEY)
```

```
C17 = C14 + (C15 - C14) * (C13 - C16) / C26
(WUF.K=LOVALU+(HIVALU-LOVALU)*(KEY-LOWKEY)/GAP, or
      =TABLE(WUFT,WS.K,0,1,.2)).
```

The rate equations are given below in the same style.

```
C19 = C10 * C17
(WUR.KL=IUD.K*WUF.K)

C20 = C6
(AQRR.KL=CAQRR)

C21 = C11 / C7
(NAQDR.KL=AQLEV.K/AQRT).
```

These equations for rows 2 through 21 are then propagated across to column M, as described above, using the relative method of copying blocks of equations, but being careful not to change the ranges in the table lookups.

The model as described runs on Visicalc. Along column M on successive computation runs will be found exactly those numbers that would be printed out for the corresponding variables in a DYNAMO simulation of the same model with DT = .1 and PRTPER = 1. The reader is invited to try this comparison himself, or to attend my presentation of this paper for a demonstration.

7. CONCLUSION

System dynamics models can indeed be implemented using Visicalc or similar spreadsheet programs. But there is a price to be paid in terms of ease of formulation of the model, and the convenience of obtaining the results, as compared to DYNAMO implementations of the same model. However, the widespread availability of spreadsheet programs may make them the vehicle of choice for models to be run on computers for which DYNAMO is not available. Also, turnkey programs which are to be used by a person not familiar with DYNAMO and are not to be altered can reach a wide audience very quickly if they are formulated for spreadsheet programs.