FACULTY OF BUSINESS

RESEARCH AND WORKING PAPER SERIES

STRUCTURING PROBLEMS FOR ANALYSIS WITH ELECTRONIC SPREADSHEETS

by

Norman P. Archer

McMaster University

Faculty of Business

Hamilton, Ontario

Canada L8S 4M4

Working Paper #267

McMASTER UNIVERSITY

1280 Main Street West
Hamilton, Ontario, Canada L8S 4M4
Telephone (416) 525-9140

STRUCTURING PROBLEMS FOR ANALYSIS WITH ELECTRONIC SPREADSHEETS

Norman P. Archer

McMaster University

Hamilton, Ontario

Canada L8S 4M4

Abstract- This paper discusses a method which uses directed graphs to reveal the underlying structure spreadsheet problems. Advantage is taken of the fact that there is a great deal of similarity among the two main classes of spreadsheet problems: those which are acyclic, and those involving cycles. For acyclic problems, a logical numbering can be assigned to cells, and the cell relationships may be shown in the form of a Cell Relationship Diagram (CRD. For cyclic problems, a pseudo CRD is used in order to break the cycles and derive pseudo-logical numberings for the cells. Cyclic spreadsheets are shown to contain fixed point problems, and three iterative numerical solution techniques (successive substitution, Newton-Raphson, and Gauss-Seidel) are described in terms of implementation and convergence properties, and demonstrated with several examples.

STRUCTURING PROBLEMS FOR ANALYSIS WITH ELECTRONIC SPREADSHEETS†

Norman P. Archer*

McMaster University

Hamilton, Ontario

Canada L8S 4M4

1. INTRODUCTION

Electronic spreadsheet packages originated with the development of Visicalc (c) for the Apple II (c) microcomputer by Bricklin and Frankston in 1979. Since then the growth in the variety and use of electronic

---

† This work was supported by a grant from the Natural Sciences and Engineering Research Council of Canada.

* Norm Archer is an Associate Professor of Management Science and Information Systems in the Faculty of Business at McMaster University. His research interests include office productivity evaluation and the management, measurement and evaluation of computer systems performance.

spreadsheet packages has been at a tremendous rate, paralleling and spurring the growth in the business microcomputer population. Modern development of the capability of electronic spreadsheets and related software has also continued to narrow the gap between the capability of spreadsheets and the financial planning packages which are the big brothers of the spreadsheet packages, but which have additional capabilities in forecasting and modeling techniques. Because of the expanding use of electronic spreadsheets, a wide variety of spreadsheet support software has also been developed to assist business users in detecting and preventing spreadsheet errors caused by incorrect formula and/or data entry (for example, [1,4]). This has become a major issue due to the importance and the complexity of some of the problems handled by these packages [6,8,11].

While the initial use of electronic spreadsheets was in business financial analysis and planning, there have been many other uses found for these packages in engineering, science and business [13,14]. In particular, they are increasingly being used to solve problems which do not have closed form solutions [2]. To aid users, (particularly scientists and engineers) in solving analytical problems, certain special purpose commercial packages have been developed as adjuncts to spreadsheets. For example, What's Best! (c) uses a linear programming algorithm to optimize certain spreadsheet formulations developed using Lotus 123 (c). Other commercially available packages such as TK!Solver (c), which is not a spreadsheet program, use tabular formulations for ease of use, and are flexible enough to solve a wide variety of both linear and non-linear problems [10].

While there is clearly a widespread and growing use of electronic spreadsheets for problem solution, there is a certain amount of coherence in the apparent variety of problems which are being solved by these

methodologies. This paper explores some of the common structures, algorithms and methodologies which can be used to aid in solving problems using electronic spreadsheets.

2. ELECTRONIC SPREADSHEET OPERATIONS

An electronic spreadsheet is simply a multi-dimensional table (most applications are two-dimensional) in which data and/or operators are stored in various cells or addresses within the table. We will restrict this discussion to two-dimensional tables, without loss of generality. Operators within a particular cell may define operations to be performed upon the contents of other cells (and sometimes upon the same cell) within the spreadsheet. To address a cell, we will specify its location within the table at address ij, which is the intersection of the i-th column and the j-th row (in most spreadsheets, the columns are actually ordered alphabetically, and the rows numerically). The content of a cell will be referred to as its value, which is the numerical result of calculations defined by the formula formed from the operators and/or constants which are also stored at that cell address.

In general, a cell will be differentiated as being one of two types: Constant, denoted by C(ij), and Formula, denoted by F(ij). Two other cell types, Script and Null, are not relevant to this discussion. Constant cells have values which are self-determined entirely from the operators and/or constants within that cell. Formula cells contain formulas which are built from numerical constants, mathematical and relational operators, and functions, and depend in some way on the values of one or more other cells. Table 1 is a list of some of the operators and a few of the built-in

functions available in most spreadsheet packages. The rules for combining such operators and functions into formulas are similar to the rules used in Fortran and Basic.

Table 1.

Sample Spreadsheet Operators And Functions

Arithmetic Operators	Relational Operators	Built-In Functions
+ Add	= Equality	SUM(list)
- Subtract	<> Not Equal To	ABS(exprn)
* Multiply	< Less Than	COUNT(list)
/ Divide	> Greater Than	LOOKUP(exprn,list)
^ Exponentiate	<= Less Than Or Equal To	MAX(list)
	>= Greater Than Or Equal To	OR(exprnl,exprn2)
		AND(exprnl,exprn2)
		PMT(prn,int,term)
		AVG(list)

In Table 1, "list" is a set of cell addresses which may be written to indicate a set of adjacent cells, with notation such as ij:kl or ij..kl, or as individual cell addresses which may or may not be in order. "exprn" may be an operator, constant, function or formula.

3. PROBLEM STRUCTURE

Figure 1 shows the relationships among the cells in a typical spreadsheet, in terms of directed arcs from each cell which is referenced to the cell(s) from which it is referenced. This graphical method of showing inter-cell relationships, along with the formulas and constants for the cells, will be referred to as a Cell Relationship Diagram (CRD). The numbers shown in the

bubbles at the ends of the cells represent a logical ordering which will be discussed below. Self-referencing cells are sometimes also desirable in spreadsheet control functions, as will be seen in some other examples, and these lead to self-loops in the corresponding cell relationship diagram. The lower part of Figure 1 shows the values calculated for the cells in this particular configuration.

Because of the directional inter-cell relationships in a spreadsheet, a Cell Relationship Diagram such as the one shown in Figure 1 will have the same properties as a digraph (directed graph) in which a Formula or Constant cell is a vertex in the graph, and the arrows indicating the directional relationships in the CRD are arcs in the graph.

Before continuing, some definitions from the theory of digraphs [15,16] are useful:

1) A walk W in a CRD is a finite sequence, consisting of vertices and arcs of the CRD alternately, and beginning and ending with vertices.

2) A walk, all of whose vertices are distinct, is called a path.

3) A walk whose first and last vertices coincide is called a closed walk.

4) A closed walk all of whose vertices are distinct except the first and last is called a cycle.

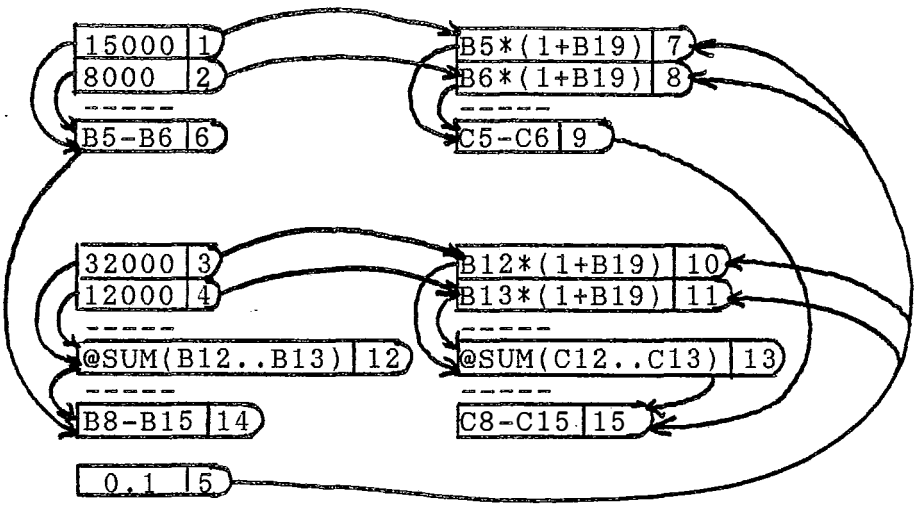
5) The number of arcs in a path is called the length of the path. The length of a cycle is similarly defined.

6) The number of arcs beginning at a vertex u is called its outdegree, and the number of arcs ending at u is called its indegree.

FIGURE 1

Sample Acyclic Cell Relationship Diagram

	A	B	C
1			
2		1986	1987
3	INCOME		
4			
5	SALES	15000 | 1	B5*(1+B19) | 7
6	SALES COST	8000 | 2	B6*(1+B19) | 8
7		-----	-----
8	NET SALES	B5-B6 | 6	C5-C6 | 9
9			
10	EXPENSE		
11			
12	OPERATIONS	32000 | 3	B12*(1+B19) | 10
13	ADMINISTR.	12000 | 4	B13*(1+B19) | 11
14		-----	-----
15	TOTAL EXPENSE	@SUM(B12..B13) | 12	@SUM(C12..C13) | 13
16		-----	-----
17	NET OP. REV.	B8-B15 | 14	C8-C15 | 15
18			
19	GROWTH RATE	0.1 | 5	



Calculated Values For Above Cell Relationship Diagram

	A	B	C
1			
2		1986	1987
3	INCOME		
4			
5	SALES	150000	165000
6	SALES COST	80000	88000
7		------	------
8	NET SALES	70000	77000
9			
10	EXPENSE		
11			
12	OPERATIONS	32000	35200
13	ADMINISTR.	12000	13200
14		-----	-----
15	TOTAL EXPENSE	44000	48400
16		-----	-----
17	NET OP. REV.	26000	28600
18			
19	GROWTH RATE	0.1	

7) A graph that ignores the orientation of a directed graph is called its underlying undirected graph. A directed graph is said to be connected if the underlying undirected graph is connected. That is, there is a path between every pair of vertices in the underlying undirected graph. In this discussion, only connected CRDs will be considered.

The following theorems [15] are stated without proof:

1) A CRD is acyclic if it has no non-trivial closed walks.

2) A CRD is cyclic if it has at least one non-trivial closed walk.

3) Every walk is a path in an acyclic CRD.

4) Every acyclic CRD has at least one source and at least one sink.

Making use of these properties, we note that a Constant cell is the only cell type which has an indegree of zero (it does not reference other cells). Hence, if one or more sources exist in a CRD then they must be Constant cells. Likewise, if sinks exist in a CRD then they must be Formula cells because Formula cells can have an outdegree of zero but must have a non-zero indegree.

4. ACYCLIC CRDs

A logical numbering can be assigned to an acyclic CRD, where the logical numbering is defined as a function N for a CRD with p vertices. N assigns to each vertex u of the CRD an integer N(u) such that each of the integers 1,2,...p is assigned to exactly one vertex, and if <u,v> is an arc, then N(u) < N(v). The CRD of Figure 1 is acyclic, and the numbers in the end

bubbles for all of the vertices (cells) represent one possible logical numbering of the CRD (there are usually many such numberings possible).

In an acyclic CRD, the various relationships which are defined in the spreadsheet represent a set of recursive, causally ordered equations.

Theorem: The cell values in a spreadsheet with no cycles can be determined correctly if the order in which the cell formulas are calculated is defined by a logical ordering on the associated acyclic CRD.

Proof: Each Formula cell has one or more precursor cells. If the Formula cell value is to be calculated correctly, then the values of all its precursor cells must first be calculated correctly. If calculations are carried out in the order in which the cells are numbered, then any logical numbering on an acyclic CRD will automatically ensure that the precursor cell values will be calculated first. Thus, all cell values will be calculated correctly.

The default order of calculation in most modern spreadsheet packages will follow a pattern which is equivalent to a logical numbering of the corresponding CRD (it is often referred to as "natural order"). Older spreadsheet versions may, however, calculate cell values row-wise or column-wise, and this can give rise to "forward references" and resulting erroneous results. The only cure for such a problem, assuming that the spreadsheet formulas cannot be rearranged to eliminate forward references, is to re-calculate the spreadsheet at least as many times as the maximum forward reference depth. This depth can be determined by tracing major cell

relationship patterns to determine how frequently these patterns "double back" through the spreadsheet. For example, a financial plan for a number of time periods with the formulas for each time period in a separate column, where there are dependencies involving revenue carryover from one period to the next, will often have a forward reference depth equal to the number of planning periods.

For a variety of reasons we will want to consider paths between two vertices in a CRD. Let us denote the terminating vertex as t and the vertex from which the paths originate as the initial vertex s. The set of integers which describe each distinct path from s to t in a logically numbered CRD will have the property that they will be monotonically increasing. Suppose that there are a total of J paths from s to t, and that there are r_j vertices in the j-th path of this set. Then $u_j(i)$ $1 <= i <= r_j$ will refer to the logically ordered index assigned to the i-th vertex in the j-th path.

To determine the impact on the value calculated for a terminal cell t due to changes in the value of an initial cell s, every one of the J paths which join these two vertices must be identified. The function $V_j(st)$ which describes the relationship of vertex s to vertex t due to the j-th path between the vertices is given by

$$V_j(st) = F_{u_j(r_j)}\{F_{u_j(r_j-1)}\{F_{u_j(r_j-2)}\{ \ldots (F_{u_j(2)}\{F_{u_j(1)}\}) \ldots\}\}\} \quad (1)$$

Here, F_k refers to the formula in cell k which calculates the cell value based upon internal cell parameters and constants as well as the values of other cells referenced by the formula.

Note that $V_j(st)$ is a complex analytical expression, and the actual value attributed to cell t as a consequence of the value of cell s will be impossible to determine unless the variables in this expression are separable. There are, in general, many such complex contributions to the value of a particular cell in a CRD. However, the main interest here is in the relative sensitivity at cell t due to value <u>changes</u> at cell s.

If $V_j(st)$ $(1 <= j <= J)$ is differentiable over the range of interest of the various functions which describe it, then the sensitivity of the value in cell t to changes in a parameter P(s) of the formula for cell s can be determined by calculating the derivative of V(st) with respect to P(s). This is a straightforward application of the chain rule for differentiation, and the result is

$$\frac{\partial V(st)}{\partial P(s)} = \sum_{j=1}^{J} \{\partial F_{u_j(r_j)} / \partial F_{u_j(r_j-1)}\} . \{\partial F_{u_j(r_j-1)} / \partial F_{u_j(r_j-2)}\} \ldots .$$

$$. \{\partial F_{u_j(2)} / \partial F_{u_j(1)}\} . \{\partial F_{u_j(1)} / \partial P(s)\} \qquad (2)$$

While equation 2 measures the absolute sensitivity, we may also be interested in the relative sensitivity, given by

$$\frac{\partial V(st)}{\partial P(s)} \times \frac{P(s)}{V(st)} \qquad (3).$$

Note that equation 2 is a differential equation formulation of the well-known "what-if?" question often asked of financial spreadsheet models. In a "what-if?" scenario, the sensitivity is measured numerically by substituting different values for the original value of the constant or parameter in question so the effect on other cell values may be observed.

5. CYCLIC CRDs

A cyclic CRD contains at least one cycle. For the cells on the cyclical walk(s), there is no logical numbering possible and hence the cell values on cyclical walks will (except in very unusual circumstances) not be calculated correctly with one cycle of calculations. The existence of a cycle in a spreadsheet is detected and signaled automatically by many spreadsheet packages, since this is an error indicator if the cycle has not been deliberately inserted into the spreadsheet. However, there are many analytical problems which lead to cycles if they are to be solved with electronic spreadsheets. Typically these involve coupled linear or non-linear equations which are otherwise solved by matrix inversion or by some iterative technique. With some care in developing the problem, solutions

can often be found easily through the adaptation for spreadsheet use of a variety of classical techniques.

To simplify the initial discussion, let us consider CRDs with only one cycle. The extension to more than one cycle can be made easily. A CRD which contains either a cycle or a cell with a self-reference (loop) cannot be assigned a logical numbering in the usual sense. Self-referencing cells will be used in the examples discussed below, but we will ignore any extensions to graph theory which may arise from their use. However, we will need to assign a pseudo-ordering to CRDs with cycles through an artifice which will allow us to discuss the order of the calculations. To assign a pseudo-ordering to a CRD containing a cycle, we select a suitable arc which is in the cycle path, and break this arc so it both enters and exits a pseudo-cell labeled c. The cell which this pseudo-cell now references will be labeled u(r-1) (an arc leads from u(r-1) to c). The cell which references the pseudo-cell will be labeled u(m). The pseudo-cell c has the properties that it:

a) will be the last cell in the CRD for which a value is calculated, and

b) retains its value which can then be released by some external control mechanism (such as a "re-calculate") for use in cell u(m) which references this pseudo-cell.

By inserting a pseudo-cell, the cycle is removed from the CRD, allowing a pseudo-logical numbering to be assigned to the cells in the CRD. However, the pseudo-cell does not really exist in the implementation of the spreadsheet, and of course this artifice does not in any way resolve the

convergence difficulties associated with the cycle. A CRD with pseudo-logical numbering will be referred to as a pseudo CRD.

The following sections discuss various techniques which may be used to solve analytical spreadsheet problems with cyclical CRDs.

5.1 SUCCESSIVE SUBSTITUTION

Since the value of a pseudo-cell is the last one to be calculated in the CRD, equation 1 will give the value calculated for cell c based on the current values of cells along the pseudo-path which was the former cycle from u(m) to c. That is, with (r-m+1) cells along the path,

$$V(c) = F_c\{F_{u(r-1)}\{F_{u(r-2)}\{ \cdots \{F_{u(m+1)}\{F_{u(m)}(V(c'))\}\} \ldots\}\}\} \quad (4)$$

Here, V(c') is a value passed on to cell u(m) from a previous calculation for cell c. Note that, since the pseudo CRD is connected, the values of all cells in the CRD (including those not in the pseudo-path) will be implicitly included in the results due to the interconnecting relationships expressed in the cell formulas along the path from u(m) to c.

If we choose, we can now transfer the calculated value V(c) for the pseudo-cell c on to cell u(m) and re-calculate the result in an iterative manner. This iterative process obviously applies to any arbitrary cell in the pseudo-path. Thus, for example, $V(u(m))^n$ becomes $V(u(m))^{n+1}$ based on the value calculated in the previous iteration, giving rise to the equation

$$V((u(m))^{n+1} = F_{u(m)}\{F_c\ \{F_{u(r-1)}\{\ \ldots\ \{F_{u(m+1)}\{V(u(m))^n\}\}\ \ldots\}\}\} \qquad (5)$$

Equation 5 is clearly of the form

$$x^{n+1} = G(x^n)$$

If the solution converges to a value x^*, then x^* is a fixed point of G. The method of replacing the current value of a cell with a new value calculated during the next iteration, in a spreadsheet containing cycles, is equivalent to the iterative method of "successive substitution" (also known as Picard's method). This technique will converge under the following conditions:

If G is continuous and differentiable, the solution will converge to a fixed point x^* provided that the initial value x^0 is in an interval I which contains both x^* and G(x), and there is some $\alpha < 1$ such that $|G'(x)| <= \alpha$ for all x in I [17, p. 274].

Successive substitution is a technique which is automatically invoked whenever a cycle appears in a spreadsheet, since no additional mathematical formulation is required. It is also easy to use standard spreadsheet functions to "break" a cycle and insert starting values to serve as initial inputs for one or more cells to get the process started close to the expected solution. In most advanced spreadsheet packages the user may specify how many iterations should be performed on a spreadsheet containing one or more cycles. However, convergence may be slow and the choice of starting values will have a effect on the number of iterations required to

converge. Convergence is not possible at all in a spreadsheet with a single cycle unless the previously described conditions hold.

Figure 2 is a pseudo CRD for an example of successive substitution used to solve a problem which has much application in multi-attribute utility theory (MAUT) [9] . Here, the weights k_i for various model attributes are known,

and a multiplicative model is assumed. In the model, the total of the L attribute weights is

$$S = \sum_{i=1}^{i=L} k_i$$

The equation which must be satisfied for the model scaling constant k is

$$k^* = \prod_{i=1}^{i=L} (1+k^* k_i) - 1 = G(k^*) \qquad (6).$$

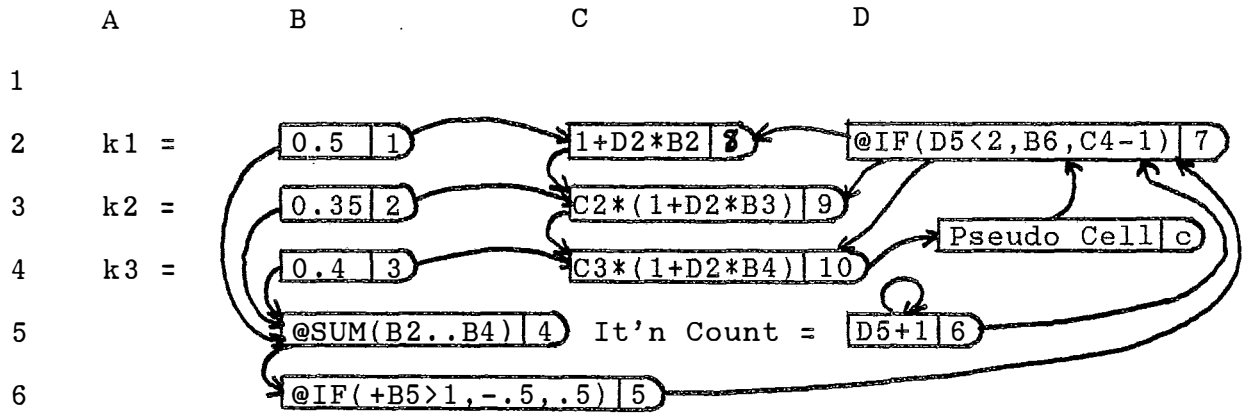
The scaling constant k^* is the fixed point of G, and must be evaluated iteratively. The unique root k^* lies in the ranges $-1 < k^* < 0$, 0 or $0 < k^* < \infty$ when $S > 1$, $= 1$, or < 1 respectively [9].

For $-1 < k^* < 0$, G' is less than 1 and the successive substitution method will converge. For $k^*=0$, $G'=1$ and $k^* > 0$, $G' > 1$, and successive substitution will not converge, so other techniques must be used in this region (the Newton-Raphson technique is applied to this problem below). The results for a particular set of weights k_i when $L = 3$ are given in the

FIGURE 2

Iterative Solution Using Successive Substitution

Pseudo Cell Relationship Diagram



Convergent Values For Above CRD

	A	B	C	D
1				
2	k1 =	0.5	0.738724	-0.52255
3	k2 =	0.35	0.603616	
4	k3 =	0.4	0.477448	
5		1.25	It'n Count =	139
6		-0.5		

second part of Figure 2. This particular problem contains one cycle with three pseudo-paths: <7,8,9,10,c,7>, <7,9,10,c,7>, and <7,10,c,7>. The problem is easily modified to contain any desired number of weights L, and the number of pseudo-paths will be equal to this number. All these paths are bisected by the pseudo-cell c. Note that the number of iterations necessary for convergence is shown, and is calculated by a self-referencing cell which is also used to control initial value assignment. The solution of $k^* = -.5225$ is given by the convergent value in cell 7.

5.2 NEWTON-RAPHSON METHOD

While there are many iterative techniques which are applicable to the solution of non-linear equations and which therefore can be used to solve fixed point spreadsheet problems, the most attractive of these is the direct substitution method since it requires no additional programming. However, this method has a linear rate of convergence if it converges at all. There are other techniques which have higher convergence rates but which require additional programming to implement. The most attractive and widely used of these is the Newton-Raphson (NR) technique which has a quadratic convergence rate [17, p. 292], and is frequently used for non-linear problem solution.

The NR iteration formulation for fixed point problems is of the form

$$x^{n+1} = x^n - \{G(x^n) - x^n\} / \{G'(x^n) - 1\} \qquad (7)$$

If we let G(x) equal the right hand side and x the left hand side of equation 1 respectively, combining this with the result of equation 2 for the partial first derivative of G, we obtain

$$\frac{\partial G(x)}{\partial x} = \sum_{j=1}^{j=J} \{\partial F_{u_j(m)}/\partial F_c\}.\{\partial F_c/\partial F_{u_j(r-1)}\}.$$

$$\{\partial F_{u_j(r-1)}/\partial F_{u_j(r_j-2)}\}.....\{\partial F_{u_j(m+1)}/\partial V_{u_j(m)}\} \qquad (8)$$

where the partial derivatives are summed over all J pseudo-paths in the cycle. Equation 8 is normally not difficult to calculate for a given pseudo-CRD.

From equation 7, let

$$h(x) = G(x) - x ,$$

so that

$$h(x^*) = 0.$$

If h(x) is twice differentiable on some interval I containing a zero point x^* and where $h'(x) \neq 0$, then the NR method will converge if x^0 is in some neighborhood of x^* [17, p. 275].

While the NR method usually converges faster than direct substitution, the price is the formulation of the partial first derivative of G and its recalculation at each iteration. The NR technique also tends to be unstable, in that it may not converge without some initial searching of trial values until a starting value is selected which is near enough to a fixed point to allow convergence. There may be more than one fixed point, and each can be determined by appropriate selection of starting values. The conditions for convergence are not nearly as restrictive as they are for

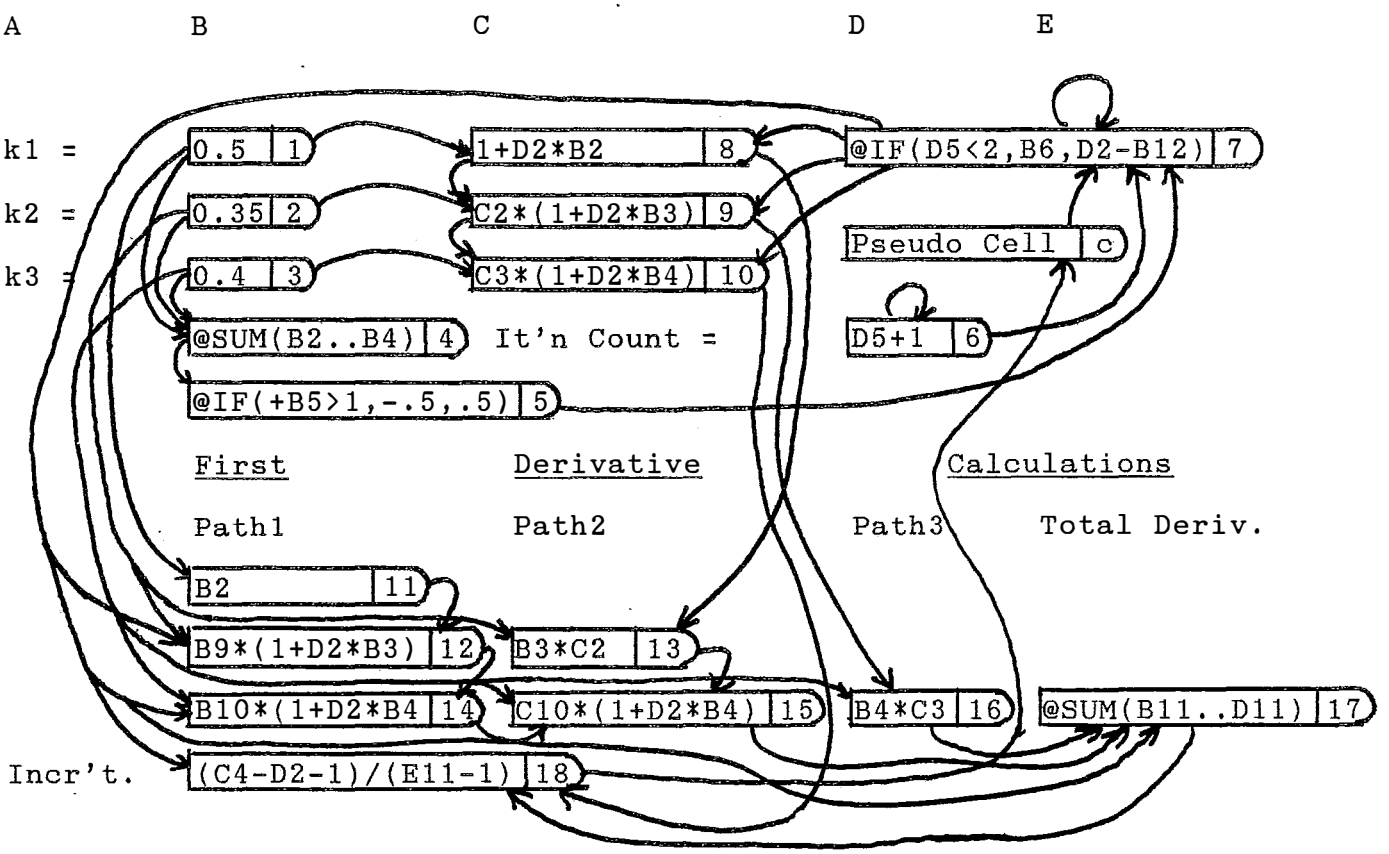
successive substitution, which may converge much more slowly or not at all, no matter what starting value is selected.

Figure 3 shows an application of the NR technique to the solution of the same MAUT problem solved in Figure 2. The pseudo-CRD shows the formulation of the problem in the same manner, but an additional section has been added for the calculation of the partial first derivative. There are three pseudo-paths as in Figure 2. The partial first derivatives on these pseudo-paths, as described in equation 8, are calculated in cells <11,12,14>, <13,15> and <16> respectively, with the total partial derivative calculated in cell <17>. This result may be verified by differentiating equation 6 directly. The self-loop in cell 7 allows the previously calculated result for the value of the cell to be used in the next cycle of calculations. A change in this value is due to the result calculated through the pseudo-cycle and entering this cell from the pre-cursor cell c. Note that the NR technique converged in 10 iterations, as compared to the 139 required for the successive substitution method in Figure 2, for the same problem. The solution of -.5225 is again shown in cell 7 in the lower part of the Figure. For this problem type, the NR technique will converge for any value of $-1 < k^* < \infty$, provided that an appropriate starting value near enough to k^* is selected.

FIGURE 3

Iterative Solution Using Newton-Raphson Method

Pseudo Cell Relationship Diagram



Convergent Values For Above CRD

	A	B	C	D	E
1					
2	k1 =	0.5	0.738724	-0.52255	
3	k2 =	0.35	0.603616		
4	k3 =	0.4	0.477448		
5		1.25	It'n Count =	10	
6		-0.5			
7		First	Derivative	Calculations	
8		Path1	Path2	Path3	Total Deriv.
9		0.5			
10		0.408553	0.258553		
11		0.323157	0.204510	0.241446	0.769114
12	Incr't.	0.0			

5.3 GAUSS-SEIDEL METHOD

The Gauss-Seidel (GS) method is a well-known iterative technique for the solution of systems of linear equations, and may be preferred to Gaussian elimination or matrix inversion for certain very large systems.

The standard form for a system of linear equations is

$$a_{i1}x_1+a_{i2}x_2+ \ldots\ldots +a_{ik}x_k = b_i \qquad (9)$$

where there are k equations in k variables. Any such system may be transformed to

$$x_i = (1/a_{ii})(b_i-a_{i1}x_1-a_{i2}x_2-\ldots-a_{i,i-1}x_{i-1}-a_{i,i+1}x_{i+1}-\ldots\ldots-a_{ik}x_k) \qquad (10)$$

for $1 <= i <= k$. This is a multi-dimensional form of a fixed point problem. The GS iterative approach to the solution of this equation set is:

$$x_i^{n+1} = (1/a_{ii})(b_i-a_{i1}x_1^{n+1}-a_{i2}x_2^{n+1}-\ldots-a_{i,i-1}x_{i-1}^{n+1}-a_{i,i+1}x_{i+1}^{n}-\ldots\ldots-a_{ik}x_k^{n}) \qquad (11)$$

for $1 <= i <= k$. The GS method will converge for any starting values of the variables, providing that the system of equations defined by equation 9 is strictly diagonally dominant [17, p. 251]. Strict diagonal dominance is defined by

$$\sum_{j\neq m}^{k} |a_{mj}| < |a_{mm}| \qquad \text{for } 1 <= m <= k$$

Figure 4 shows an application of the Gauss-Seidel method to the system

$$5x_1 + x_2 - 2x_3 = -12$$

$$3x_1 - 7x_2 + x_3 = 23$$

$$2x_1 - x_2 + 4x_3 = 15$$

This system is seen to be diagonally dominant by inspection, so the Gauss-Seidel method will converge. The equations, transformed according to equation 11, appear in the pseudo CRD in Figure 4. Note that, since there are three unknowns, there are three interleaved cycles which are broken by the three pseudo-cells shown in the Figure. The solutions for the three variables are (-.4970, -3.0359, 3.2395), and appear in the lower part of the Figure. This system converged in 24 iterations from starting values of 0 for each of the three variables.

This method is closely related to the successive substitution method used for single variable problems, and requires very little work to set up. Also, diagonal dominance may be achieved in some cases simply by the way the equations are ordered, improving the possibility of applying Gauss-Seidel iteration successfully to a wide range of linear systems.

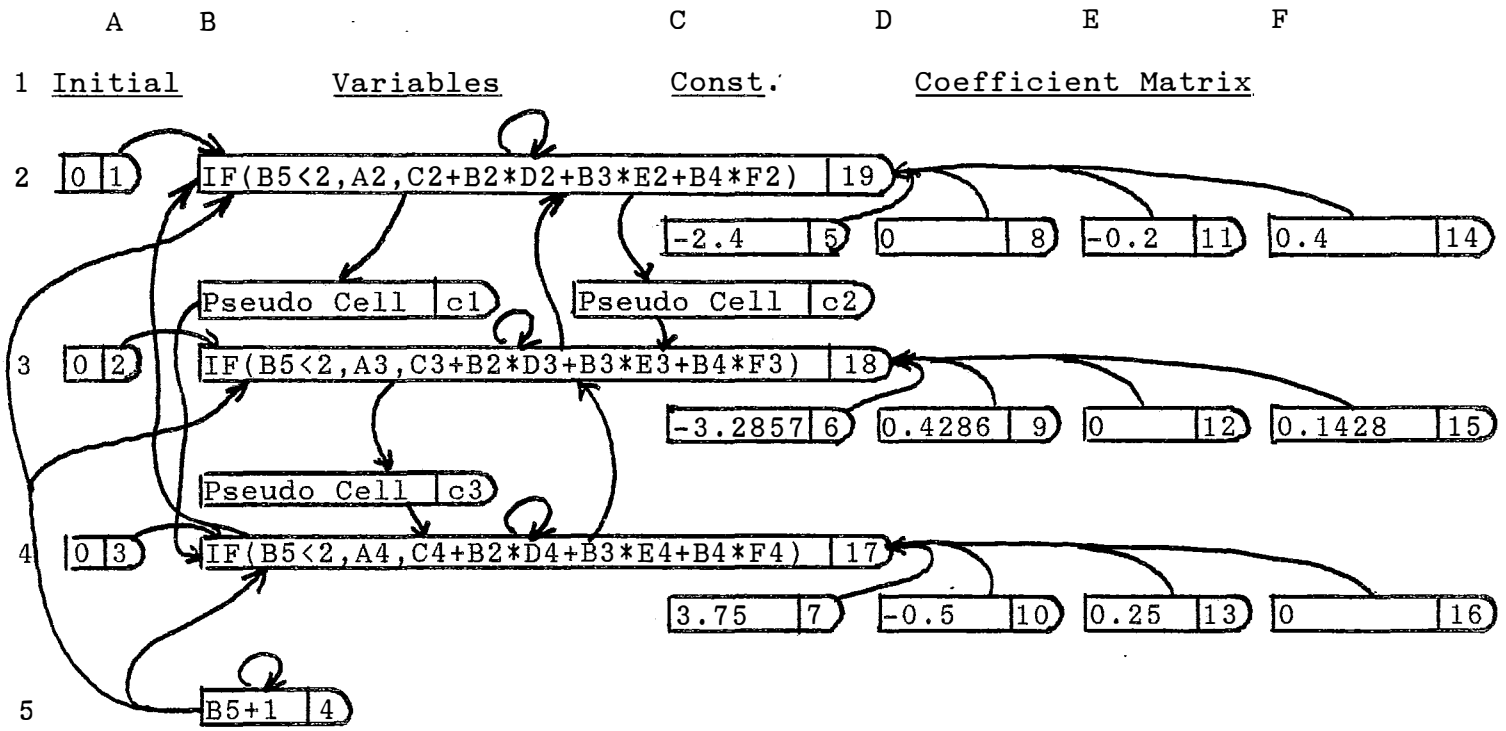
6. DISCUSSION

This paper has discussed only a few of the well-known numerical techniques which may be adapted to spreadsheet solutions. For example, systems of non-linear equations may also be solved by iterative techniques in much the same manner as systems of linear equations, with convergence depending upon certain local conditions on the related Jacobian matrix [12, p. 93]. This requires arranging the non-linear equation set into the form of a fixed point problem. For a broader viewpoint, the book by Arganbright [2] offers many hints on the development of solutions to a variety of related problems.

FIGURE 4

Iterative Solution Using Gauss-Seidel Method

Pseudo Cell Relationship Diagram



Convergent Values For Above CRD

	A	B	C	D	E	F
1	Initial	Variables	Const.	Coefficient Matrix		
2	0	-0.4970	-2.4	0	-0.2	0.4
3	0	-3.0359	-3.2857	0.4286	0	0.1428
4	0	3.3239	3.75	-0.5	0.25	0
5		24				

While constraint equations were not discussed in this paper, a typical spreadsheet may include a wide variety of various types of constraints. For example, the templates provided by Cobb and Cobb [5] for a set of acyclic business spreadsheet problems almost all have numerous constraints invoked through cell formulas. Any of the methods discussed in this paper may be used for problems involving formula constraints, but the convergence conditions will generally not apply since the functions may not be continuous over the range of interest. In fact, because some of the constraints are usually tight in the solution, it has been found from experience that constraints usually enhance convergence. It is also necessary that there be a feasible region for each variable to be determined and that the starting value for each variable is in a feasible region. However, if the feasible region(s) is(are) not convex, there is no guarantee that the result(s) will converge to global solution(s).

There are many spreadsheet packages available, but there are also many microcomputer packages on the market which are designed specifically to solve nonlinear optimization problems [7]. Most standard spreadsheet packages do not at the present time have the versatility or the capability to solve many of these problems directly, but with a little creativity along the lines discussed in this paper it is possible to handle some of the more tractable ones. At the same time, the capability of spreadsheets is still developing so that there is little doubt that we will eventually see professional spreadsheet packages which will handle a much greater variety of optimization problems [3]. For example, the Smart Spreadsheet (c) from the related integrated package currently includes matrix manipulation

routines. An advantage of such extensions is the fact that many spreadsheet routines have been integrated with word processing, graphics and database software which allows much more inherent flexibility in the manipulation and presentation of data.

References

1. Anonymous, "The Spreadsheet Auditor", PC World, (May, 1985), pp.95-98.
2. Deane Arganbright, Mathematical Applications Of Electronic Spreadsheets, McGraw-Hill: New York, (1985).
3. Samuel E. Bodily, :"Spreadsheet Modeling As A Stepping Stone", Interfaces 16 (Sept. - Oct. 1986), pp. 34-52.
4. Steven Burke, "Program Closes Window On Spreadsheet Pitfalls", Infoworld, (Sept. 9, 1985), pp. 33-34.
5. Douglas Ford Cobb and Gena Berg Cobb, SuperCalc SuperModels For Business, Cue Corp.: Indianapolis, Indiana (1983).
6. Robert Firmin, "Spreadsheet Use: High Anxiety?", Computer World, (June 10, 1985), p.75, 92.
7. Bruce L. Golden and Edward A. Wasil, "Nonlinear Programming On A Microcomputer", Computers And Operations Research 13, (1986), pp. 149-166.
8. Doran Howitt, "Avoiding Bottom-Line Disaster", Infoworld, (February 11, 1985), pp. 26-30.
9. Ralph L. Keeney and Howard Raiffa, Decisions With Multiple Objectives: Preference And value Tradeoffs, Wiley : New York (1976), pp. 347-348.
10. Milos Konopasek and Sundaresan Jayaraman, The TK!Solver Book, Osborne/ McGraw-Hill: Berkley, Calif. (1984).

11. Harry Miller, "The Matrix Menace", PC World, (March, 1985) pp. 19-20.

12. David G. Moursund and Charles S. Duris, Elementary Theory And Application Of Numerical Analysis, McGraw-Hill: New York (1967).

13. M.E. Palmer III, M.G. Pecht, and J.V. Horan, "Adapting The Spreadsheet To Engineering Problems", Computers In Mechanical Engineering, (September 1985), pp. 49-56.

14. Mahmut Parlar, "Dynamic Programming On An Electronic Spreadsheet",4 Computers And Industrial Engineering, V. 10, (1986), pp. 203-213.

15. R.F. Robinson and L.R. Foulds, Digraphs: Theory And Techniques, Gordon and Breach Publishers: New York (1980).

16. M.N.S. Swamy and K. Thulasiraman, Graphs, Networks And Algorithms, Wiley: New York (1981).

17. James S. Vandergraft, Introduction To Numerical Computations, Second Edition, Academic Press: New York (1983).

Faculty of Business

McMaster University

WORKING PAPERS - RECENT RELEASES

247. Itzhak Krinsky and A. Leslie Robb, "Approximately the Statistical Properties of Elasticities Derived from Translog and Generalized Leontief Cost Functions", February 1986.

248. John W. Medcof, "An Integration of Kelley's Covariation and Configuration Theories", March 1986.

249. Isik Urla Zeytinoglu, "The International Labour Organization's Standards and the African Labour Laws", March 1986.

250. Robert F. Love and Paul D. Dowling, "A Generalized Bounding Method for Facilities Location Models", April 1986.

251. Roy J. Adams, "The Role of Management in a Political Conception of The Employment Relationship", June 1986.

252. Roy J. Adams, "Employment Standards As Industrial Relations Sub-System: The Ontario Case", June 1986.

253. Harish C. Jain, "Recruitment and Selection of Visible Minorities in Canadian Police Forces: A Survey of Selected Police Agencies", June, 1986.

254. George Steiner, "An Algorithm To Generate the Ideals of a Partial Order", July 1986.

255. John Miltenburg, "A Theoretical Basis For Scheduling Mixed-Model Assembly Lines For Just-In-Time Production Systems", July 1986.

256. John Miltenburg, "Scheduling Mixed Model Multi-Level Just-In-Time Production Systems", July 1986.

257. Joseph B. Rose, "Statutory Expedited Grievance Arbitration: The Case of Ontario", July, 1986.

258. John W. Medcof, "The Effects of Event Valence and Degree of Attribution Upon Action Recommendations", September 1986.

259. Paul D. Dowling and Robert F. Love, "One-Dimensional Floor Layout Solutions Using Squared Distances", September, 1986.

260. Harish C. Jain, "Affirmative Action/Employment Equity in Canada: Findings of a Survey", September, 1986.

261. Min Basadur, "Catalyzing Interfunctional Efforts to Find and Creatively Solve Important Business Problems", September, 1986.

262. Roy J. Adams and Isik Zeytinoglu, "Labour-Management Dispute Resolution in Canadian Heavy Industry: The Hilton Works Case", September, 1986.

263. Rick D. Hackett, "New Directions in The Study of Employee Absenteeism: A Research Example", October 1986.

264. Ali-Reza Montazemi, "An Analysis of Information Technology Assessment and Adoption in Small Business Environments", October, 1986.

265. Isik U. Zeytinoglu, "Part-time Workers: Unionization and Collective Bargaining in Canada", November, 1986.

266. Norman P. Archer and M.W. Luke Chan, "The Architecture of Chinese-English Microcomputer Systems", December, 1986.