Int. J. Engng Ed. Vol. 8, No. 4, pp. 278–287, 1992
Printed in Great Britain.

0742–0269/91 $3.00+0.00

Spreadsheet Approach to Partial Differential Equations. Part 1: Elliptic Equation*

C. Y. LAM
School of Mechanical and Production Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 2263

Two educational spreadsheet programs have been successfully developed by using the popular spreadsheet software Lotus 1-2-3 to solve the two-dimensional elliptic Laplace equation in rectangular domains subject to Dirichlet boundary conditions. Finite difference methods are used. They include the Gauss–Seidel method, the successive over-relaxation method and the alternating direction implicit method. Extensive macro commands have been used to construct these customized programs so as to provide interactiveness, user-friendliness as well as some distinct and useful features. The advantages of these spreadhsheet programs over the traditional computer programs are highlighted. Experience in classroom adoption of these programs has confirmed that they are of great educational value.

INTRODUCTION

ALL engineering and science disciplines involve partial differential equations. These equations therefore form an important part of the engineering and science curriculum. With the complex nature of today's physical problems and the advances in computer technology, the numerical techniques of solving partial differential equations have grown in importance.

Computer programs are usually used as tools in the teaching of the numerical methods for solving partial differential equations. With these programs, a user can perform numerous numerical experiments (e.g. to investigate the effects of varying the grid sizes or the over-relaxation parameter for the successive over-relaxation method) which are otherwise impossible. Traditionally, they are written in a computer language (e.g. Fortran) and the codes are easily available from many textbooks (for examples, see [1, 2]). However, before they can be used, a substantial amount of effort and time is required to spend on modifying (in order to suit the particular compiler to be used), keying-in, compiling and debugging them. Moreover, they do not provide graphics capability and their outputs are sets of numerical values that are often difficult to interpret. For iterative methods, they do not provide intermediate values or errors which are necessary for the study of the convergence characteristics of such methods. While the tedious and time-consuming traditional approach may improve the computer programming techniques, it is ineffective as a learning tool for the numerical methods concerned.

To overcome the drawbacks of the traditional approach, a new approach has been made to solve the partial differential equations by using the electronic spreadsheet in a microcomputer. The objective is to enhance the learning effectiveness of the numerical methods, rather than the computer programming techniques. The electronic spreadsheet, originally developed for business applications, has proved itself to be useful in various engineering and science educational purposes [4–8]. Its cell structure, being similar to the domain discretization used in finite difference methods, together with its many advanced features (e.g. the advanced macro commands and the graphics capability) make it a good choice for the current application in solving partial differential equations.

In this paper, two spreadsheet programs developed by using the popular Lotus 1-2-3 Release 3.1 spreadsheet software to solve the two-dimensional Laplace equation in rectangular domains, subject to Dirichlet conditions, are described. The Laplace equation is considered because it is the simplest elliptic equation that models a variety of physical problems; for example, heat conduction, fluid mechanics and electrostatic problems. Three finite difference numerical methods have been implemented: the Gauss Seidel (GS) method, the successive over-relaxation (SOR) method and the alternating direction implicit (ADI) method. The solving of the Laplace equation by these basic methods has been commonly used in the teaching of elliptic partial differential equations. The spreadsheet approach to the parabolic and hyperbolic equations is presented in [9].

* Paper accepted 3 August 1992.

PROBLEM FORMULATION

The two-dimensional Laplace equation is

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{1}$$

where u is the dependent variable, x and y are the spatial coordinates.

To solve the Laplace equation by using finite difference methods, the domain is discretized and the grid points are labelled as shown in Fig. 1 for a rectangular domain of dimensions $L \times H$ with constant grid sizes Δx and Δy. To obtain the numerical solution, four Dirichlet boundary conditions $u(0, y)$, $u(L, y)$, $u(x, 0)$ and $u(x, H)$ are specified.

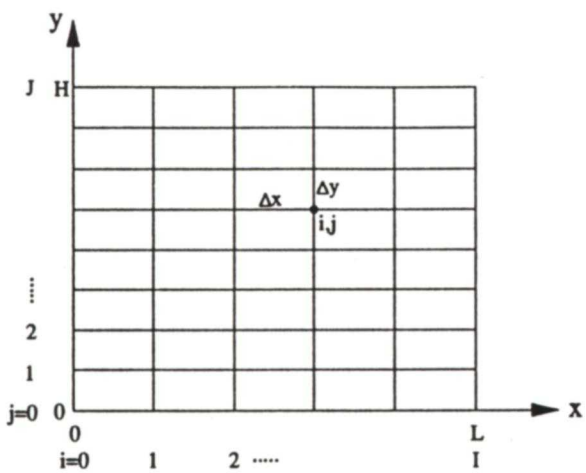


Fig. 1. The discretized computational domain

The finite difference methods implemented in the spreadsheet programs can be found in many standard texts (for examples, see [1–3]). They are briefly summarized below.

The GS method

Using central differences, the finite difference form of equation (1) for the GS method is

$$(u_{i,j}^{k+1})_{GS} = \frac{u_{i+1,j}^{k} + u_{i-1,j}^{k+1} + (\Delta x/\Delta y)^2(u_{i,j+1}^{k} + u_{i,j-1}^{k+1})}{2[1 + (\Delta x/\Delta y)^2]} \tag{2}$$

where Δx, Δy are the grid sizes in the x and y directions respectively (a) the double subscripts (for example, $[i, j]$) denote the number of the x- and y-grid points respectively, (b) the superscripts denote the numbers of iterates, and (c) the subscript GS denotes the GS value.

The SOR method

For the SOR method, the finite difference equation is

$$u_{i,j}^{k+1} = \omega(u_{i,j}^{k+1})_{GS} + (1 - \omega)u_{i,j}^{k}$$

$$= \frac{\omega}{2[1 + (\Delta x/\Delta y)^2]}\,[u_{i+1,j}^{k} + u_{i-1,j}^{k+1} + (\Delta x/\Delta y)^2(u_{i,j+1}^{k} + u_{i,j-1}^{k+1})] + (1 - \omega)u_{i,j}^{k} \tag{3}$$

where $\omega(1 < \omega \leqslant 2)$ is the over-relaxation parameter and all other parameters are the same as those defined for equation (2).

The ADI method

In the ADI method, we take $\Delta x = \Delta y$ for simplicity, which is usually the case in the teaching of this method. The ADI method involves calculations along the rows of interior grid points successively and then along the columns of interior grid points successively.

For the row calculations, the finite difference equation is

$$u_{i-1,j}^{k+1} - (2 + \zeta)u_{i,j}^{k+1} + u_{i+1,j}^{k+1} = -u_{i,j-1}^{k} + (2 - \zeta)u_{i,j}^{k} - u_{i,j+1}^{k} \tag{4a}$$

where $\zeta(\zeta > 0)$ is an acceleration parameter and the other parameters are the same as those defined in equation (2).

Along the j-th row, applying this at all interior grid points $i = 1, \ldots, I - 1$ yields

$$\begin{bmatrix} -(2+\zeta) & 1 & 0 & 0 & 0 & \ldots & 0 \\ 1 & -(2+\zeta) & 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & -(2+\zeta) & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & -(2+\zeta) & 1 & \ldots & 0 \\ . & . & . & . & . & & . \\ . & . & . & . & . & & . \\ 0 & 0 & 0 & 0 & 0 & \ldots & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ . \\ . \\ -(2+\zeta) \end{bmatrix} \begin{bmatrix} u_{1,j}^{k+1} \\ u_{2,j}^{k+1} \\ u_{3,j}^{k+1} \\ u_{4,j}^{k+1} \\ . \\ . \\ u_{I-1,j}^{k+1} \end{bmatrix} = \begin{bmatrix} -u_{1,j-1}^{k} + (2-\zeta)u_{1,j}^{k} - u_{1,j+1}^{k} - u_{0,j} \\ -u_{2,j-1}^{k} + (2-\zeta)u_{2,j}^{k} - u_{2,j+1}^{k} \\ -u_{3,j-1}^{k} + (2-\zeta)u_{3,j}^{k} - u_{3,j+1}^{k} \\ -u_{4,j-1}^{k} + (2-\zeta)u_{4,j}^{k} - u_{4,j+1}^{k} \\ . \\ . \\ -u_{I-1,j-1}^{k} + (2-\zeta)u_{I-1,j}^{k} - u_{I-1,j+1}^{k} - u_{I,j} \end{bmatrix} \tag{4b}$$

where I is the number of x-intervals.

For the column calculations, the finite difference equation is

$$u_{i,j-1}^{k+2} - (2 + \zeta)u_{i,j}^{k+2} + u_{i,j+1}^{k+2} = -u_{i-1,j}^{k+1} + (2 - \zeta)u_{i,j}^{k+1} - u_{i+1,j}^{k+1} \tag{5a}$$

where the parameters are the same as those defined for equation (4a).

Along the i-th column, applying this at all interior grid points $j = J - 1, J - 2, \ldots, 1$ yields

$$\begin{bmatrix} -(2+\zeta) & 1 & 0 & 0 & 0 & \dots & 0 \\ 1 & -(2+\zeta) & 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & -(2+\zeta) & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & -(2+\zeta) & 1 & \dots & 0 \\ . & . & . & . & . & & . \\ . & . & . & . & . & & . \\ 0 & 0 & 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ . \\ . \\ -(2+\zeta) \end{bmatrix} \begin{bmatrix} u_{i,J-1}^{k+2} \\ u_{i,J-2}^{k+2} \\ u_{i,J-3}^{k+2} \\ u_{i,J-4}^{k+2} \\ . \\ . \\ u_{i,1}^{k+2} \end{bmatrix} = \begin{bmatrix} -u_{i-1,J-1}^{k+1} + (2-\zeta)u_{i,J-1}^{k+1} - u_{i+1,J-1}^{k+1} - u_{i,j} \\ -u_{i-1,J-2}^{k+1} + (2-\zeta)u_{i,J-2}^{k+1} - u_{i+1,J-2}^{k+1} \\ -u_{i-1,J-3}^{k+1} + (2-\zeta)u_{i,J-3}^{k+1} - u_{i+1,J-3}^{k+1} \\ -u_{i-1,J-4}^{k+1} + (2-\zeta)u_{i,J-4}^{k+1} - u_{i+1,J-4}^{k+1} \\ . \\ . \\ -u_{i-1,1}^{k+1} + (2-\zeta)u_{i,1}^{k+1} - u_{i+1,1}^{k+1} - u_{i,0} \end{bmatrix} \quad (5b)$$

where J is the number of y-intervals.

Note that in equation (5b), the unknown column matrix is arranged such that the j-subscript is increasing upward along the matrix. This simplifies the construction of the macros used in the programs for the display of the solution, such that the y-axis is pointing upward in the same direction as that in the physical domain shown in Fig. 1.

THE SPREADSHEET PROGRAMS

The spreadsheet software Lotus 1-2-3 Release 3.1 offers many useful advanced features for the present application; for example, the three-dimensional worksheet feature and the graphics capability [10]. To facilitate user-friendliness and interactiveness, macro commands have been extensively used in the programs developed.

The structure of the spreadsheet comprises sheets of rows and columns of cells as shown in Fig. 2. From Figs 1 and 2, it can be seen that the rows and columns of cells of a sheet closely resemble the discretized computational domain. Therefore, a particular cell range of a sheet, which varies in size according to I and J, can be allocated to store and display the numerical values at all grid points. The allocation has been automated by using macros. Since the worksheet is three-dimensional, the working area (which contains the input data and the computed results), the macros and other data (e.g. flow control flags and matrices) can be stored on different sheets. Sheet A has been chosen for the working area. This avoids the possibility of damaging the macros (if unprotected by the user for any reason) and the distraction to the user since only working sheet A is displayed at all times.

The spreadsheet programs developed are menu-driven. The menu, which is common for the spreadsheet programs developed, is shown in Fig. 3 and the descriptions of the program commands are given in Table 1. As can be seen in Fig. 3 and Table 1, all required tasks can be done within the

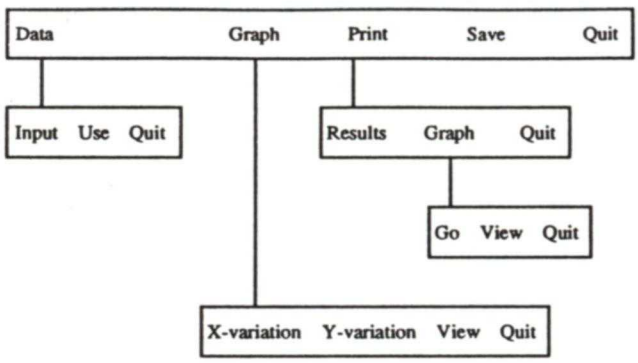


Fig. 3. The program menu tree

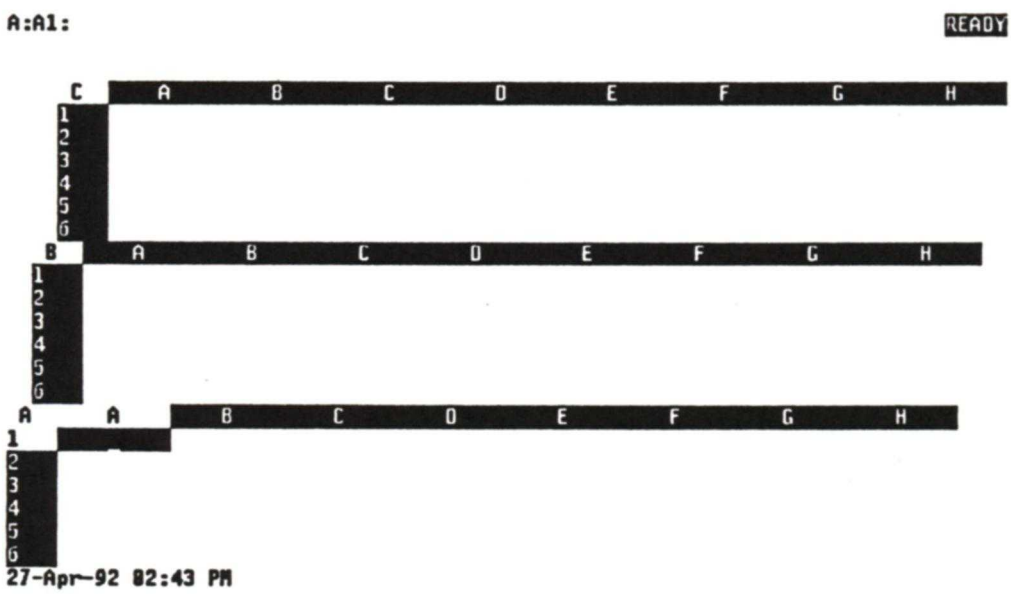


Fig. 2. The structure of the three-dimensional worksheet of Lotus 1-2-3 (showing sheets A, B and C)

Table 1. The program commands

Program command	Description
Data Input	To enter new data for subsequent computation
Data Use	To continue iteration using the existing data
Data Quit	To return to the main program menu
Graph X-variation	To plot and display the graph of the dependent variable u against the x-direction
Graph Y-variation	To plot and display the graph of the dependent variable u against the y-direction
Graph View	To display the current graph as and when the user desires.
Graph Quit	To return to the main program menu
Print Results	To send the input data and the computed results to a printer
Print Graph Go	To send the current graph to a printer
Print Graph View	To display the current graph. This allows the user to view the graph before printing
Print Graph Quit	To return to the main program menu
Print Quite	To return to the main program menu
Save	To save the spreadsheet program with the existing data and results in a file
Quit	To return to 1-2-3's READY mode

program menu. To select a program command, the user can either enter the first letter of that command or highlight that command by using the arrow keys and press the Enter key.

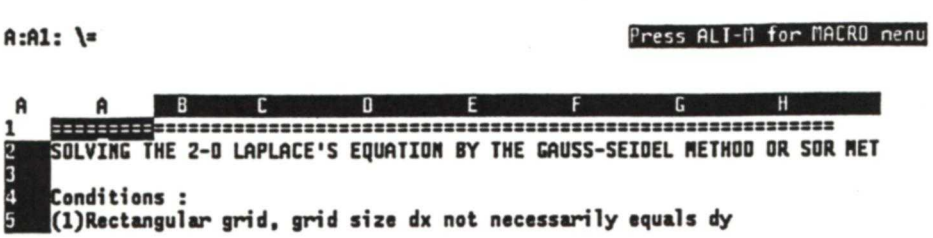
After a spreadsheet program is retrieved into the 1-2-3 environment, the user is prompted self-explained messages at the top of the monitor screen at all stages. These messages include user instructions or questions, error messages and current actions taken by the computer. If an unacceptable answer (e.g. an alphabetical answer instead of an expected numerical answer) is entered, the programs will either reject it and pose the same question again for re-entering or allow the user to change it later. Examples of these messages are shown in Fig. 4. Audible signals are also provided when an error occurs or when computation stops. With very little spreadsheet knowledge, the user is able to solve Laplace equation including displaying and printing the results in graphical form. The user can thus concentrate on his numerical experiments rather than on how to use the spreadsheet programs. The only 1-2-3 commands required are the /File Retrieve command which retrieves a spreadsheet program into the 1-2-3 environment and the /Quit command which leaves 1-2-3 and returns to the operating system.

As the numerical methods considered are all iterative, a safety feature has been incorporated into the programs to prevent them from running into an infinite iterative loop should the solution be diverging. This is done by limiting the maximum number of iterations to a value input by the user. After each iterative loop, a convergence test is done at all interior grid points and the criterion used is

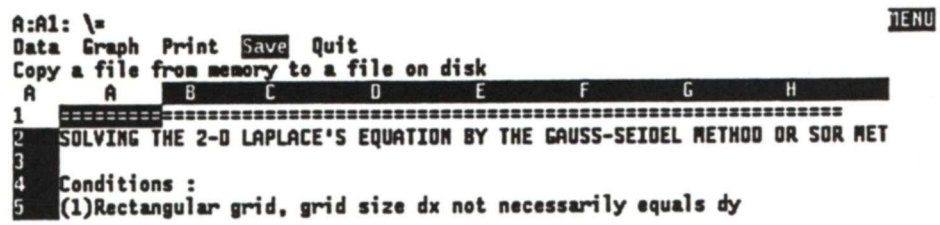
$$\left| \frac{(u_{i,j})_{\text{current}} - (u_{i,j})_{\text{previous}}}{(u_{i,j})_{\text{previous}}} \right| < \varepsilon$$

where ε is a small prescribed convergence parameter. Also, the starting values are taken to be the same at all interior grid points.

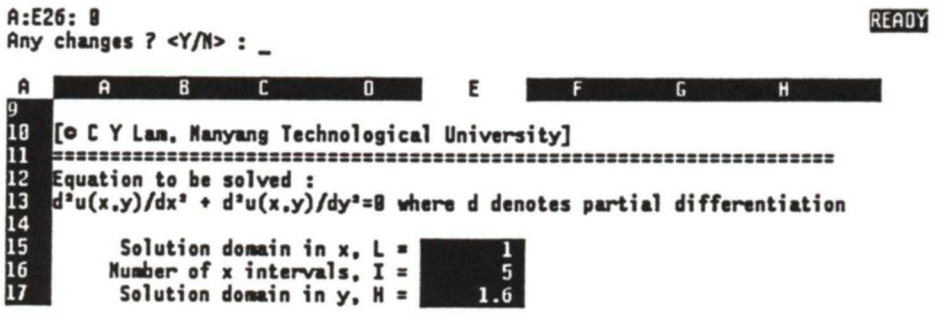
In the programs, the step sizes Δx and Δy are computed automatically based on the input data L, I, H and J.



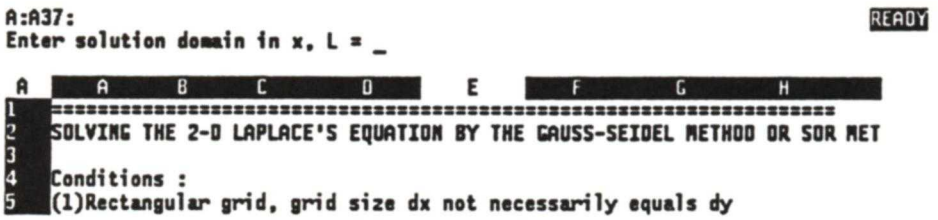
(a) Instruction to invoke the main program menu



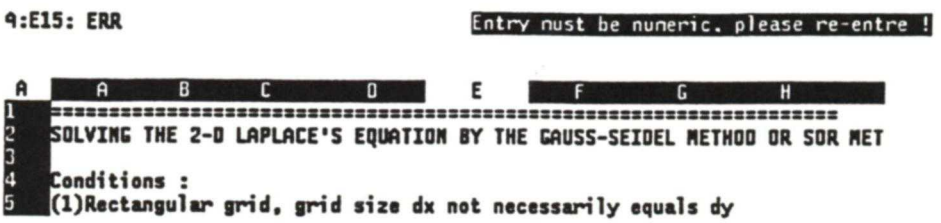
(b) Program menu commands with explanatory notes



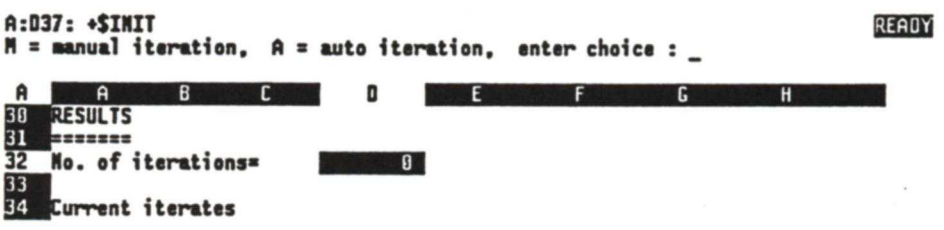
(c) Options for data changing and correction



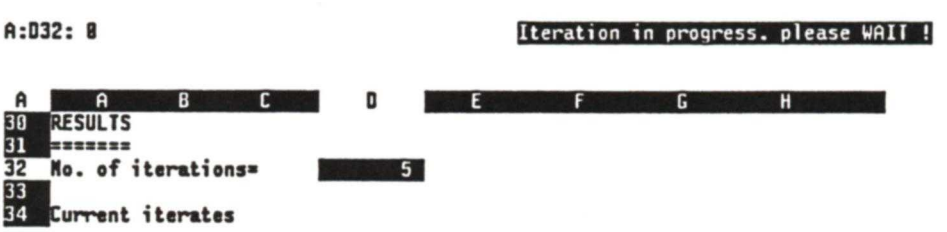
(d) Input of a typical data



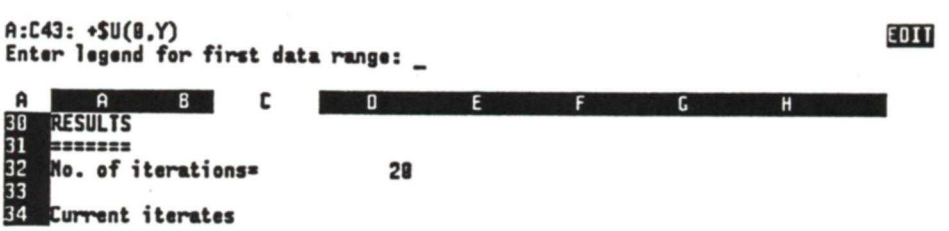
(e) An error message



(f) Selection of a mode of iteration



(g) Current action taken by the program



(h) Input of a legend for graphing the results

Fig. 4. Examples of the user-friendly and interactive features

The spreadsheet program GSSOR.WK3

This spreadsheet program employs both the GS and SOR methods. This is possible because equation (3) reduces to equation (2) for $\omega = 1$. Therefore, the GS method applies if one is entered for the over-relaxation parameter ω. Other values of $\omega (1 < \omega \leqslant 2)$ entered would imply that the SOR method is being used.

Example. Consider the two-dimensional heat conduction problem of a rectangular plate of dimensions $L = 1$ m and $H = 1.6$ m subject to the boundary conditions $u(0, y) = 10°C$, $u(1, y) = -10°C$, $u(x, 0) = 0°C$ and $u(x, 1.6) = 30°C$. Using the program command Data Input, the data are entered as shown in Fig. 5. The ω-value entered is 1.34 and thus the SOR method is used. After confirming the data, the program automatically reserves a range of cells on working sheet A to display the current iterates. This includes calculating and displaying the two axes, laying the boundary conditions at the cells corresponding to the boundary grid points (see Fig. 6(a)) and assigning equation (3) to the cells corresponding to the interior grid points. An option is then provided for the user to select a mode of iteration (see Fig. 4(f)). In the manual mode, iteration stops every loop and the user indicates whether iteration is to be continued. In the automatic mode, iteration continues until the solution converges or the number of iteration exceeds the maximum value entered. After selection, the program then reserves two ranges of cells also on sheet A to display the previous iterates and the relative percentage errors. For this problem, the converged solution is obtained after 20 iterations as shown in Fig. 6(a), the corresponding relative percentage errors and the previous iterates are shown in Figs 6(b) and (c)

```
A:E26: 0                                                                READY
Any changes ? <Y/N> : _

A      A       B       C       D      E        F        G        H
9
10 [© C Y Lam, Nanyang Technological University]
11 ======================================================================
12 Equation to be solved :
13 d²u(x,y)/dx² + d²u(x,y)/dy²=0 where d denotes partial differentiation
14
15          Solution domain in x, L =        1
16         Number of x intervals, I =        5
17          Solution domain in y, H =      1.6
18         Number of y intervals, J =        8
19       Boundary condition, u(0,y) =       10
20       Boundary condition, u(L,y) =      -10
21       Boundary condition, u(x,0) =        0
22       Boundary condition, u(x,H) =       30
23          Over-relaxation factor =     1.34
24           Convergence parameter =   0.0001
25           Max. no. of iteration =       30
26           Initial values u(x,y) =        0
27                     Step in x, dx =    0.2
28                     Step in y, dy =    0.2
GSSOR.WK3                                 CMD
```

Fig. 5. The input data in GSSOR.WK3

```
A:D32: 0                                  Solution converged. press Enter !

A    A       B       C        D        E        F        G        H
30 RESULTS
31 =======
32 No. of iterations=            20
33
34 Current iterates
35
36       1.6   |                30       30       30       30
37       1.4   |       10  17.8421 19.10871 16.77491 9.555378      -10
38       1.2   |       10  12.2597 11.81782 8.435539 1.446607      -10
39         1   |       10 9.378879 7.467347 3.702816  -2.2045      -10
40       0.8   |       10 7.788511 4.969903 1.112904 -3.96743      -10
41       0.6   |       10 6.805261 3.510898 -0.25364 -4.77813      -10
42       0.4   |       10 5.921655 2.522039 -0.86023 -4.89142      -10
43       0.2   |       10 4.359349 1.515858 -0.81794 -3.92734      -10
44         0   |                 0        0        0        0
45         y      ---------------------------------------------------
46             x        0      0.2      0.4      0.6      0.8        1
```

(a) The converged solution

A:D48: READY

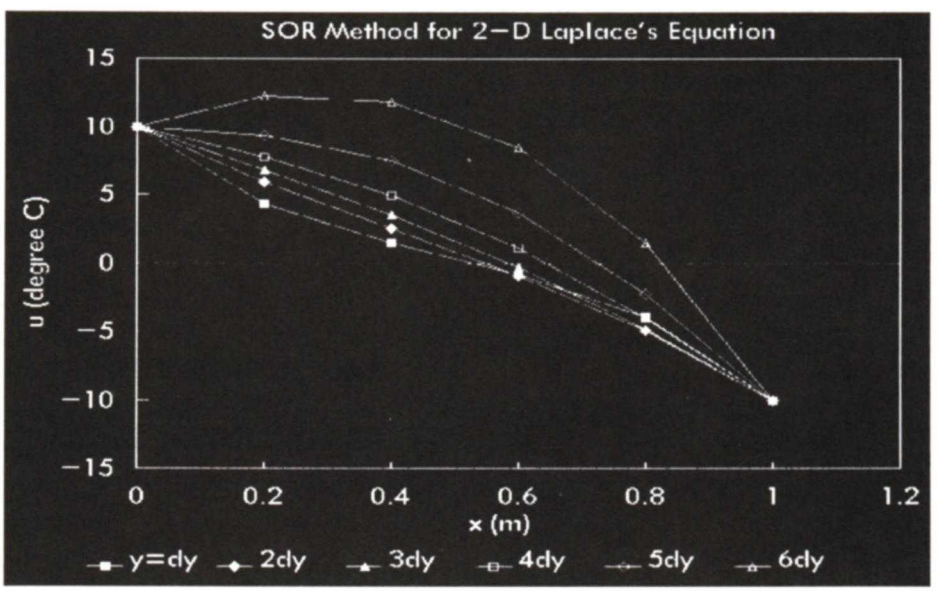
	A	B	C	D	E	F	G	H
48	% errors							
49								
50	1.6			0	0	0	0	
51	1.4		0	4.5E-07	5.7E-07	4.3E-07	2.9E-07	0
52	1.2		0	1.4E-06	2.1E-06	2.9E-06	8.9E-06	0
53	1		0	9.3E-07	3.9E-06	7.7E-06	8.7E-06	0
54	0.8		0	2.2E-06	1.8E-06	0.00002	4.1E-06	0
55	0.6		0	5.4E-06	0.000012	0.000031	3.4E-07	0
56	0.4		0	0.000013	0.000027	0.000055	2.4E-06	0
57	0.2		0	8.5E-06	0.000053	0.000041	4.0E-06	0
58	0			0	0	0	0	
59	y							
60		x	0	0.2	0.4	0.6	0.8	1

(b) The relative percentage errors

A:D62: READY

	A	B	C	D	E	F	G	H
62	Previous iterates							
63								
64	1.6			30	30	30	30	
65	1.4		10	17.84289	19.1087	16.7749	9.555377	-10
66	1.2		10	12.25968	11.8178	8.435518	1.446596	-10
67	1		10	9.378873	7.467324	3.702794	-2.20452	-10
68	0.8		10	7.788495	4.969899	1.112887	-3.96745	-10
69	0.6		10	6.805224	3.510855	-0.25365	-4.77813	-10
70	0.4		10	5.92158	2.521971	-0.86028	-4.89143	-10
71	0.2		10	4.359312	1.515778	-0.81797	-3.92735	-10
72	0			0	0	0	0	
73	y							
74		x	0	0.2	0.4	0.6	0.8	1

(c) The previous iterates



(d) Variations of i with x as displayed on the screen

Fig. 6. The solution of the heat conduction problem using GSSOR.WK3

respectively. The variations of temperature with x are graphed as shown in Fig. 6(d) by using the Graph X-variation command. The results and the graphs can be printed by using the program's Print commands.

The spreadsheet program ADI.WK3

This spreadsheet program employs the ADI method. The usual procedure for running this program is identical to that for the GSSOR.WK3 since the program menu of both is the same. For

this program, $\Delta x = \Delta y$ but I is not necessarily equal to J. Therefore, after the data L, H, I and J are entered, the program will automatically check and request for re-entering these data if $\Delta x(=L/I) \neq \Delta y(=H/J)$. For the calculations along a row or a column, equations (4b) or (5b) is required to be solved. This is conveniently done by the matrix inversion technique since the inverse of the coefficient matrices of these equations are the same and remain unchanged for every loop. By using 1-2-3's built-in matrix operations (the 1-2-3's /Data Matrix Invert and /Data Matrix Multiply commands), the solving has been easily programmed in the macros. Since one complete iteration consists of the calculations of the rows followed by the calculations of the columns, the results from the calculations of the rows of each loop are also stored and displayed as the intermediate iterates in addition to the previous and current iterates on working sheet A.

Example. Consider the same problem solved previously. The input data are entered as shown in Fig. 7. The converged solution, the relative percentage errors, the previous iterates, the intermediate iterates and the variations of u with x after 12 iterations are obtained as shown in Fig. 8.

CONCLUSIONS

The spreadsheet programs have been developed by using the Lotus 1-2-3 package, thus its usefulness is basically governed by 1-2-3's capability. The user-friendliness, interactiveness and graphics capability have been easily achieved by using the macros and the 1-2-3's built-in functions. Flexibility and safety features have also been incorporated. These include the automatic error detection, the maximum number of iterations specification and the iteration mode selection. The user is therefore able to use the programs without much spreadsheet knowledge. The iteration mode selection also allows the user to view the intermediate values and errors of any loop and thus helps the user to better understand the numerical methods. After iteration stops (either the solution has converged or the number of iteration exceeds the maximum value specified), it is possible to continue the iteration by invoking the Data Use command after manually changing the input data maximum number of iterations (if more iterations are required) or the convergence parameter (if better accuracy is desired). By using a common program menu, the user can easily master the programs in the shortest possible time. Furthermore, owing to the close resemblance of the row and column layout of the spreadsheet cells to the discretization of the physical domains, both spreadsheet programs are able to display results at the correct physical geographical locations of the problems. This helps the user to visualize and to interpret the numerical results. These features, which are difficult or impossible to achieve by the traditional programming techniques, are the key elements for effective learning of the numerical methods. Indeed, testing of these programs in the classroom has verified that most students have no difficulty in using the programs after a brief demonstration, including some who have never used a spreadsheet before. Feedback from the students shows that these spreadsheet programs are preferred to the traditional computer programs.

However, like any other computer programs, the spreadsheet programs developed also have their own limitations. The program must be used after Lotus 1-2-3 is loaded into the memory, the computer thus has less storage capacity for the computation. The number of grid points (cells) that can be

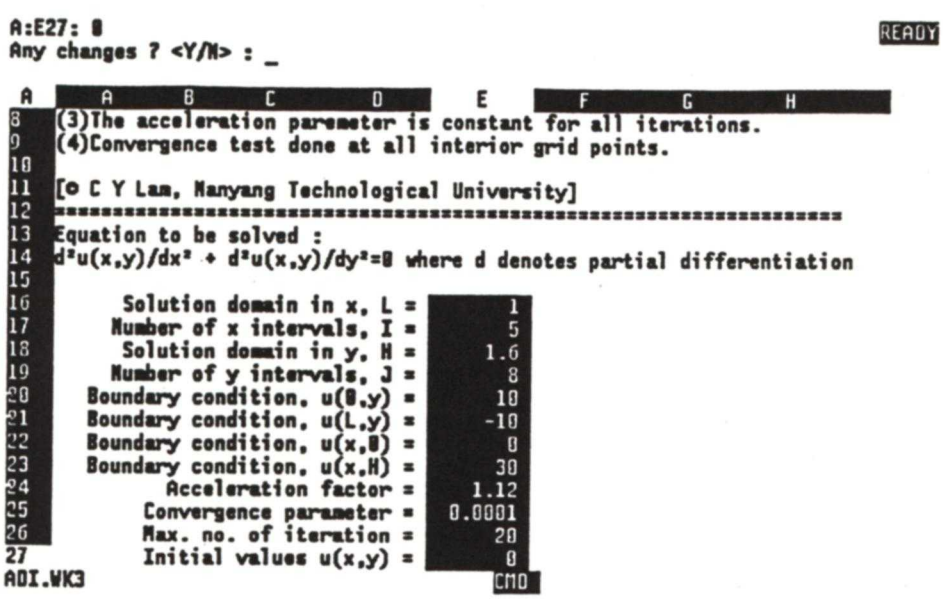


Fig. 7. The input data in ADI.WK3

displayed on the screen at the same time is limited. For a large number of grid points, in order to see the values at the same grid points outside the screen, the user has to return to 1-2-3's READY mode by using the program command Quit and then uses the arrow keys to move around. The maximum number of curves that can be displayed is six, being limited by the capability of 1-2-3. For ADI.WK3, the maximum number of x- and y-intervals is limited to 81 so that the coefficient matrices of equations (4b) and (5b) are at most 80×80, which is the maximum size invertable by 1-2-3. While these limitations may cause some problems in practical engineering application purposes, they do not pose real constraints for educational purposes in which the requirements are not so stringent. Owing to the rapid software and hardware development, it is anticipated that these limitations will eventually become less severe.

With the experience gained, the author is cur-

A:G38: 9.55537583548651742 Solution converged. press Enter !

	A	B	C	D	E	F	G	H
31	RESULTS							
32	=======							
33	No. of iterations=			12				
34								
35	Current (k+2) iterates							
36								
37		1.6			30	30	30	30	
38		1.4		10	17.84289	19.10871	16.77488	9.555376	-10
39		1.2		10	12.25968	11.81783	8.435582	1.446602	-10
40		1		10	9.378867	7.467367	3.70278	-2.2045	-10
41		0.8		10	7.78849	4.969924	1.112867	-3.96743	-10
42		0.6		10	6.885244	3.518981	-0.25369	-4.77812	-10
43		0.4		10	5.921655	2.522853	-0.86027	-4.89143	-10
44		0.2		10	4.359376	1.515878	-0.81794	-3.92734	-10
45		0			0	0	0	0	
46		y							
47			x	0	0.2	0.4	0.6	0.8	1

(a) The converged solution

A:G49: READY

	A	B	C	D	E	F	G	H
49	% errors							
50								
51		1.6			30	30	30	30	
52		1.4		10	1.9E-07	3.4E-06	2.9E-07	4.1E-06	-10
53		1.2		10	4.2E-07	0.00001	8.4E-07	0.000051	-10
54		1		10	8.5E-07	0.000021	3.0E-06	0.000043	-10
55		0.8		10	1.0E-06	0.000034	9.8E-06	0.000026	-10
56		0.6		10	1.1E-06	0.000044	0.000042	0.00002	-10
57		0.4		10	9.7E-07	0.000047	9.3E-06	0.000015	-10
58		0.2		10	7.0E-07	0.000043	5.3E-06	0.00001	-10
59		0			0	0	0	0	
60		y							
61			x	0	0.2	0.4	0.6	0.8	1

(b) The relative percentage errors

A:G63: READY

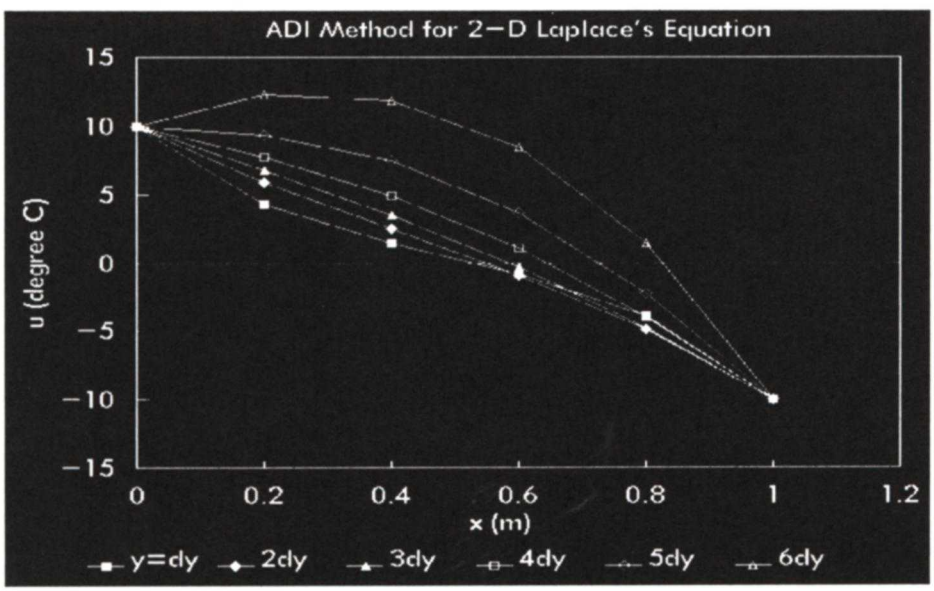
	A	B	C	D	E	F	G	H
63	Previous (k) iterates							
64								
65		1.6			30	30	30	30	
66		1.4		10	17.84289	19.10864	16.77489	9.555337	-10
67		1.2		10	12.25969	11.81771	8.435589	1.446528	-10
68		1		10	9.378875	7.467211	3.702791	-2.2046	-10
69		0.8		10	7.788498	4.969755	1.112878	-3.96753	-10
70		0.6		10	6.885251	3.518745	-0.25368	-4.77822	-10
71		0.4		10	5.92166	2.521933	-0.86026	-4.8915	-10
72		0.2		10	4.359379	1.515813	-0.81794	-3.92738	-10
73		0			0	0	0	0	
74		y							
75			x	0	0.2	0.4	0.6	0.8	1

(c) The previous iterates

A:G77: READY

	A	B	C	D	E	F	G	H
77	Intermediate (k+1) iterates							
78								
79	1.6			30	30	30	30	
80	1.4		10	17.84209	19.10868	16.77489	9.555358	-10
81	1.2		10	12.25969	11.81778	8.435507	1.446571	-10
82	1		10	9.37887	7.467298	3.702784	-2.20454	-10
83	0.8		10	7.788494	4.969852	1.112873	-3.96747	-10
84	0.6		10	6.805247	3.510833	-0.25368	-4.77817	-10
85	0.4		10	5.921657	2.522001	-0.86027	-4.89146	-10
86	0.2		10	4.359377	1.51585	-0.81794	-3.92736	-10
87	0			0	0	0	0	
88	y							
89		x	0	0.2	0.4	0.6	0.8	1

(d) The intermediate iterates



(e) Variation of u with x displayed on the screen

Fig. 8. The solution of the heat conduction problem using ADI.WK3

rently working on the extensions of these programs to problems with Neumann conditions and/or irregular boundaries and other elliptic equations. The implementation of other numerical methods using the spreadsheet approach is also being considered.

REFERENCES

1. J. H. Ferziger, *Numerical Methods for Engineering Applications*. Wiley-Interscience, New York (1981).
2. W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, *Numerical Recipes—The Art of Scientific Computing (Fortran Version)*. Cambridge University Press, Cambridge (1989).
3. J. D. Hoffman, *Numerical Methods for Engineers and Scientists*. McGraw-Hill, New York (1992).
4. N. D. Rao, Typical applications of microcomputer spreadsheets to electrical engineering problems. *IEEE Trans. Educ.*, **E-27**, 237–242 (1984).
5. Y. L. Kuo and W. Kuo, Application of electronic spreadsheets to linear and integer programming. *Int. J. Appl. Engng. Educ.*, **3**, 563–576 (1987).
6. S. K. Chan and C. Y. Lam, The electronic spreadsheet as a tool for course co-ordination in a school of engineering. *Comput. Educ.*, **14**, 231–238 (1990).
7. R. Kari, Spreadsheets in advanced physical chemistry. *J. Comput. Math. Sci. Teaching*, **10**, 39–48 (1990).
8. C. Pinter-Lucke, Rootfinding with a spreadsheet in pre-calculus. *J. Comput. Math. Sci. Teaching*, **11**, 85–93 (1992).
9. C. Y. Lam, Spreadsheet approach to partial differential equations. Part 2: Parabolic and hyperbolic equations. *Int. J. Engng Educ.*, to be published (1992).
10. Lotus Development Corporation, *Lotus 1-2-3 Release 3.0 Reference*. Lotus Development Corporation, Cambridge (1989).