Int. J. Engng Ed. Vol. 22, No. 4, pp. 868–875, 2006
Printed in Great Britain.

0949-149X/91 $3.00+0.00

A Partial Differential Equation Solver for the Classroom*

CHUNG-YAU LAM and F. H. ALAN KOH
School of Mechanical and Aerospace Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798. E-mail: mcylam@ntu.edu.sg

This paper presents a solver for partial differential equations that was developed in Microsoft Excel. The solver consists of selected finite-difference numerical methods for the three types of partial differential equations: namely the elliptic, parabolic and hyperbolic equations. The partial differential equations and the finite-difference methods implemented are commonly used in classroom teaching. Regular cell arrangement in worksheets represents the finite-difference grid. The computational procedures were translated into Visual Basic for Application code to automate the methods. The solver can handle problems with Dirichlet and Neumann boundary conditions. A graphical user interface accepts problem parameters while worksheets and charts display the numerical solutions. The solver requires minimal spreadsheet knowledge from the users. Various numerical experiments such as divergence and stability testing can be performed easily. These allow users to concentrate on the numerical aspects of their problems, which enhances the learning of the numerical methods.

Keywords: partial differential equations; spreadsheet; differential equation solver; numerical method

INTRODUCTION

THERE ARE a number of commercial software packages available for solving partial differential equations (PDEs). While they are fast and powerful solvers, which can tackle problems with complex geometries and complicated boundary conditions, they tend to have a steep learning curve and are expensive. For classroom purposes, where simpler problems are used, a more cost-effective and simple solution is desired.

Spreadsheets were used in recent years as an educational tool in various engineering problems. Some examples in heat transfer are Planck's black-body radiation [1] and thermal radiation in enclosures [2]. For electrical engineering, an example is an antenna design [3]. Mathematics examples include discrete and fast Fourier transforms [4] and complex transformations [5]. Many features of the spreadsheet such as circular reference, matrix inversion and solver have been used successfully in these applications. A drawback of spreadsheets is the inability to symbolically manipulate formulas.

Through the use of macros, spreadsheet programs can be made to function similar to programs written in conventional programming languages. Harnessing this ability, a spreadsheet application for solving PDEs was developed in Microsoft Excel. The availability of user forms for graphical user interface and macros for automation are the main contributors to make this application functioning like a programme. Finite-difference methods transform partial derivatives into difference expressions, allowing PDEs to be recast as simple algebraic expressions. A total of seven finite difference methods commonly used in the classroom were selected and implemented for the elliptic, hyperbolic and parabolic PDEs. The solving procedures were coded in Visual Basic for Application (VBA) and programmed into macros that run automatically. User forms provide an interactive and familiar appearance to the application. This separates user from the worksheets and directs the user to place problem parameters in the right places. The user forms request all problem parameters before solving begins.

The native layout of rows and columns of spreadsheet cells is ideally suited to represent the finite-difference grid. The spreadsheet cells of the worksheet also behave like 'variables' and hold data. Calculation goes on in either the cells or the background and the problem solutions are presented on the worksheets. Charts are also used to present the solution graphically.

NUMERICAL FORMULATIONS

The finite-difference methods implemented in the solver are commonly used in the classroom. Details of the methods can be found in many textbooks, for examples see [6–8]. The essences of the methods are given below.

* Accepted 4 November 2005.

Poisson equation

The Poisson equation is:

$$u_{xx} + u_{yy} = f(x,y) \tag{1}$$

Using central-differencing, the finite-difference equation for (1) is given by:

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta x)^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta y)^2} = f(x,y) \tag{2}$$

where Δx and Δy are the grid sizes in the x and y directions respectively; the subscripts i and j denote the x and y stations respectively.

For the **Gauss Seidel method**, Equation (2) is rearranged to:

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

where the superscript represents the iteration number.

For the **Successive Over-Relaxation method** (SOR), we use:

$$u_{i,j}^{k+1} = \omega[u_{i,j}^{k+1}]_{GS} + (1-\omega)u_{i,j}^{k} \tag{4}$$

where $\omega[1 < \omega < 2]$ is the over-relaxation factor and $[u_{i,j}^{k+1}]_{GS}$ are the Gauss Seidel values as calculated from Equation (3).

For the **Alternate Direction Implicit method**, Equation (2) is rearranged to:

$$\begin{aligned}(\Delta y)^2 u_{i+1,j}^{k+1} - 2[(\Delta x)^2 + (\Delta y)^2]u_{i,j}^{k+1} + (\Delta y)^2 u_{i-1,j}^{k+1} \\ = -(\Delta x)^2(u_{i,j+1}^{k} + u_{i,j-1}^{k}) + (\Delta x)^2(\Delta y)^2 f(x,y)\end{aligned} \tag{5a}$$

and

$$\begin{aligned}(\Delta x)^2 u_{i,j+1}^{k+1} - 2[(\Delta x)^2 + (\Delta y)^2]u_{i,j}^{k+1} + (\Delta x)^2 u_{i,j-1}^{k+1} \\ = -(\Delta y)^2(u_{i+1,j}^{k} + u_{i-1,j}^{k}) + (\Delta x)^2(\Delta y)^2 f(x,y)\end{aligned} \tag{5b}$$

Equations (5a) and (5b) represent calculations along the j-th row and i-th column of grid points respectively. These equations are used alternatively until solution converges.

Heat (diffusion) equation

Consider the heat equation:

$$u_t = c^2 u_{xx} \tag{6}$$

where c is a constant.

In the **Explicit Forward-Time-Centered-Space method**, the forward-difference approximation is used to replace the time derivative and the centered-difference approximation is used to replace the spatial derivative. This yields:

$$\frac{u_i^{j+1} - u_i^j}{\Delta t} = c^2 \frac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{(\Delta x)^2} \tag{7}$$

where the subscript represents the station in the spatial x-direction and the superscript represents the step in the time t-direction.

Rearranging Equation (7) explicitly,

$$u_i^{j+1} = \left[1 - 2c^2\frac{\Delta t}{(\Delta x)^2}\right]u_i^j + c^2\frac{\Delta t}{(\Delta x)^2}(u_{i+1}^j + u_{i-1}^j) \tag{8}$$

This method is stable when the grid Fourier number:

$$c^2\frac{\Delta t}{(\Delta x)^2} \le \frac{1}{2}$$

In the **Crank Nicolson method**, the centered-difference approximation, calculated from time steps j to $j+1$, is used to replace the time derivative while the average of the centered-difference approximation at time steps j and $j+1$ replaces the spatial derivative. This yields:

$$\frac{u_i^{j+1} - u_i^j}{\Delta t} = c^2\frac{1}{2}\left[\frac{u_{i+1}^{j+1} - 2u_i^{j+1} + u_{i-1}^{j+1}}{(\Delta x)^2} + \frac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{(\Delta x)^2}\right] \tag{9}$$

Rearranging,

$$\begin{aligned}-c^2\frac{\Delta t}{(\Delta x)^2}u_{i-1}^{j+1} + 2\left[1 + c^2\frac{\Delta t}{(\Delta x)^2}\right]u_i^{j+1} - c^2\frac{\Delta t}{(\Delta x)^2}u_{i+1}^{j+1} \\ = c^2\frac{\Delta t}{(\Delta x)^2}u_{i-1}^j + 2\left[1 - c^2\frac{\Delta t}{(\Delta x)^2}\right]u_i^j + c^2\frac{\Delta t}{(\Delta x)^2}u_{i+1}^j\end{aligned} \tag{10}$$

At each time step, a system of simultaneous equation results from applying Equation (10) at each station in the spatial direction. The solution of system of simultaneous equation gives the functional values at the next time step. The Crank Nicolson method is unconditionally stable.

In the **Backward-Time-Centered-Space method,** the backward-difference approximation is used to replace the time derivative while the centered-difference approximation is used to replace the spatial derivative:

$$\frac{u_i^{j+1} - u_i^j}{\Delta t} = c^2\frac{u_{i+1}^{j+1} - 2u_i^{j+1} + u_{i-1}^{j+1}}{(\Delta x)^2} \tag{11}$$

The method yields

$$-c^2\frac{\Delta t}{(\Delta x)^2}u_{i-1}^{j+1}+\left[1+2c^2\frac{\Delta t}{(\Delta x)^2}\right]u_i^{j+1}$$
$$-c^2\frac{\Delta t}{(\Delta x)^2}u_{i+1}^{j+1}=u^{j_i} \qquad (12)$$

Similar to the Crank Nicolson method, a system of simultaneous equation results from applying Equation (12) at each station in the spatial direction at each time step. The solution of system of simultaneous equation gives the functional values at the next time step. This implicit method is unconditionally stable.

Wave equation

The wave equation is:

$$u_{tt}=c^2u_{xx} \qquad (13)$$

where c is a constant.

Using centered-difference approximation for both the time and spatial derivatives, the finite-difference equation for Equation (13) is:

$$\frac{u_i^{j+1}-2u_i^j+u_i^{j-1}}{(\Delta t)^2}=c^2\frac{u_{i+1}^j-2u_i^j+u_{i-1}^j}{(\Delta x)^2} \qquad (14)$$

where the subscript represents the station in the spatial x-direction and the superscript represents the step in the time t-direction. Rearranging:

$$u_i^{j+1}=2\left[1-c^2\left(\frac{\Delta t}{\Delta x}\right)^2\right]u_i^j$$
$$+c^2\left(\frac{\Delta t}{\Delta x}\right)^2(u_{i+1}^j+u_{i-1}^j)-u_i^{j-1} \qquad (15)$$

This **Centered-Time-Centered-Space method** is stable for:

$$c^2\left(\frac{\Delta t}{\Delta x}\right)^2\le 1$$

USER INTERFACE DESIGN

The user interface is based on the requirement that it must be simple, easy to use and foolproof. It should not hinder the learning of the numerical methods.

Form-based user interface

Form-based graphical user interface (GUI) is adopted for this application in Excel since such interface tends to improve user friendliness. The GUI provides an interactive environment for user input. The form for selecting the PDE is shown in Fig. 1. As the user cycles through the choices on the list located on the top left of the form, the corresponding equation is displayed to the right of the list. The fields on the lower half of the form also respond to the selected equation by displaying a white or gray background, indicating if the field is applicable or not. The GUI's response to the user guides the user and increases user confidence in using the solver.

The form in Fig. 2 shows an elliptic equation being selected, for which the user must specify the boundary conditions along the edges of the solution domain and the starting values for iteration.

Input of mathematical expressions

To enhance user friendliness, a mathematical function input form as shown in Fig. 3 has been created for the user to input mathematical

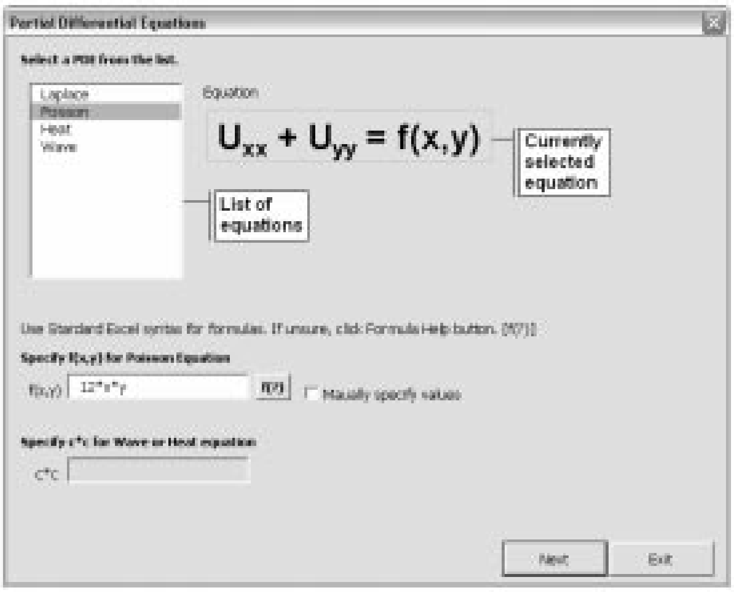

Fig. 1. Form for selecting PDE.

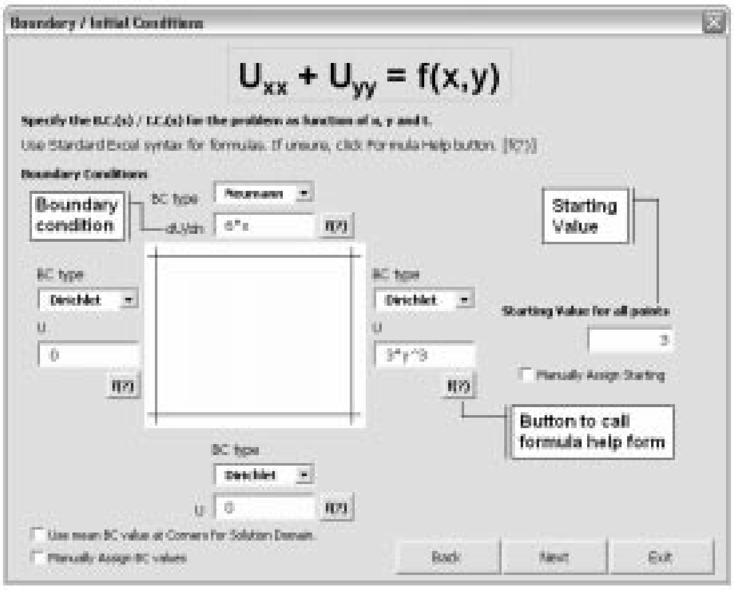

Fig. 2. Input form for elliptic PDE.

functions by selection. Upon selection, the corresponding function is translated and displayed in Excel syntax automatically. This is helpful when specifying boundary or initial conditions prescribed by mathematical functions. With this form, the user is not required to have a prior knowledge of excel syntax for mathematical functions. This form is called by clicking on the 'f(?)' button, as indicated in Fig. 2. Such buttons are provided when needed.

Spreadsheet display

The spreadsheet represents an array, which stores the functional values at the grid points in the solution domain. The mathematical functions take values form the cells of the spreadsheet and return the calculated values to the appropriate cells according to the respective numerical schemes. Other essential information pertaining to the respective scheme is also included. Examples for the SOR scheme for solving the Poisson equation $\nabla^2 u = 12xy$, and the Crank Nicolson scheme for solving the parabolic heat equation $u_t = c^2 u_{xx}$, are shown in Figs 4 and 5 respectively. In these figures, in addition to the numerical solution, all user input such as step sizes, boundary conditions, initial conditions, mathematical function needed, like $f(x, y)$ of the Poisson Equation, are displayed. Colur schemes indicate the x- and y-axis, the type of boundary conditions along the edges of the solution domain and the relative magnitude of the value of $f(x, y)$ over the solution domain. The numerical solution and the associated boundary and initial conditions are displayed at their

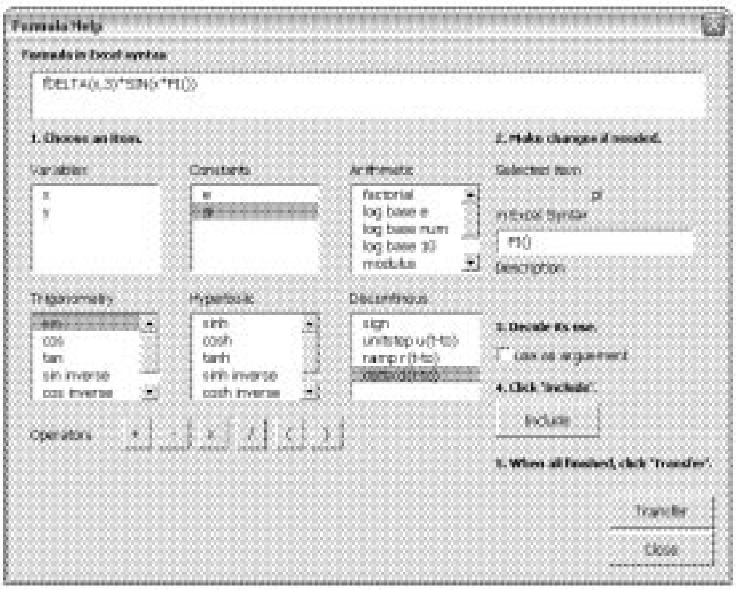

Fig. 3. Input form for mathematical functions.

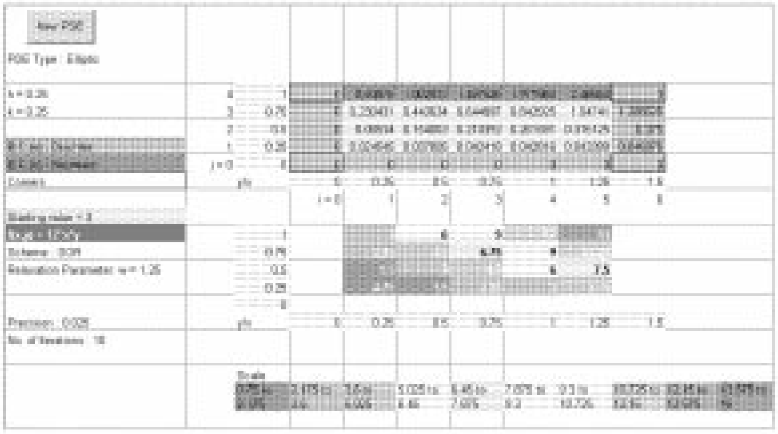

Fig. 4. Spreadsheet display for solution of $\nabla^2 u = 12xy$ using SOR method, with $\Delta x = \Delta y = 0.25$ and $\omega = 1.25$.

respective geometric locations. This, together with the colour scheme used, aids interpretation of the numerical solution.

For implicit methods such as the Crank Nicolson scheme, it is necessary to solve a system of simultaneous equations. In such a situation, the worksheet functions minverse( ) and mmult( ) are extremely useful. The system of linear equations is put into matrix form to obtain the coefficient matrix, which is inverted using minverse( ). The coefficient matrix is displayed on the spreadsheet to facilitate the use of these worksheet functions. This intermediate working is retained to reflect the implicit nature of the selected method and to facilitate checking.

Graphical display

A graphical form of the solution greatly improves its interpretation. The charts in Excel are used to display the numerical solutions graphically. These charts are generated automatically by macros, when requested by the users. The numerical solutions in Figs 4 and 5 are shown graphically in Figs 6 and 7 respectively. A color scheme is also provided. Hyperbolic PDEs are displayed in the same style as parabolic PDEs, where the solution is plotted at each station in the spatial direction for each time step.

MACROS IN VISUAL BASIC FOR APPLICATION

Visual Basic for Application (VBA) is the scripting language used to create all macros in this application. The macros were written either from scratch or by editing pre-recorded macros. The first example of code describes a function that replaces variables in a mathematical expression with numerical values, which was very helpful in the evaluation of $f(x, y)$ for the Poisson equation. The second example is a segment of code that breaks up a surface plot into ten regions and assigns a specific colour to each region. The outcome is shown in Fig. 6.

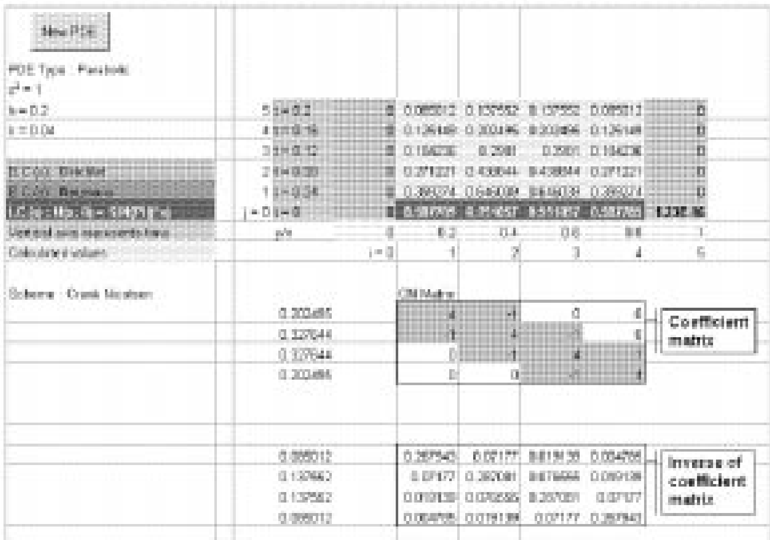

Fig. 5. Spreadsheet display for solution of $u_t = u_{xx}$ using the Crank Nicolson method, with $\Delta x = 0.2$ and $\Delta y = 0.04$.

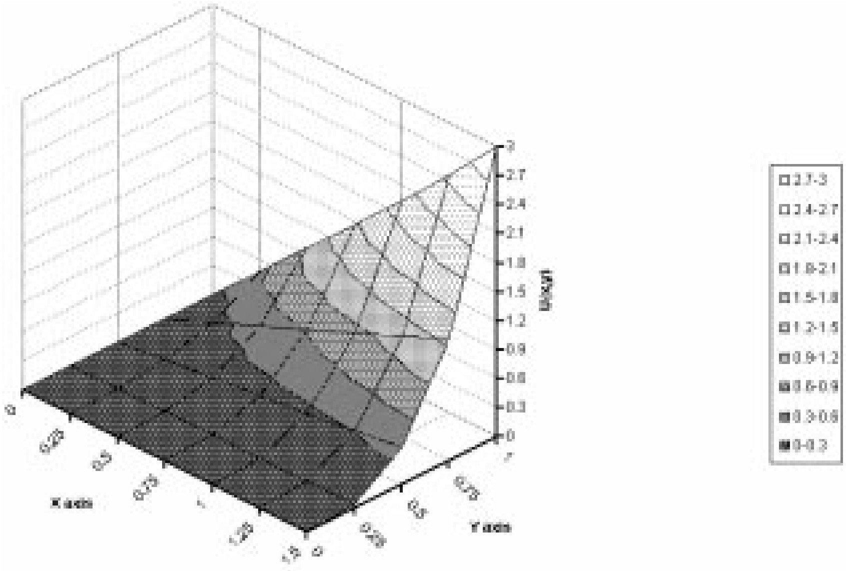

Fig. 6. Chart display for solution of $\nabla^2 u = 12xy$ using SOR method, with $\Delta x = \Delta y = 0.25$ and $\omega = 1.25$.

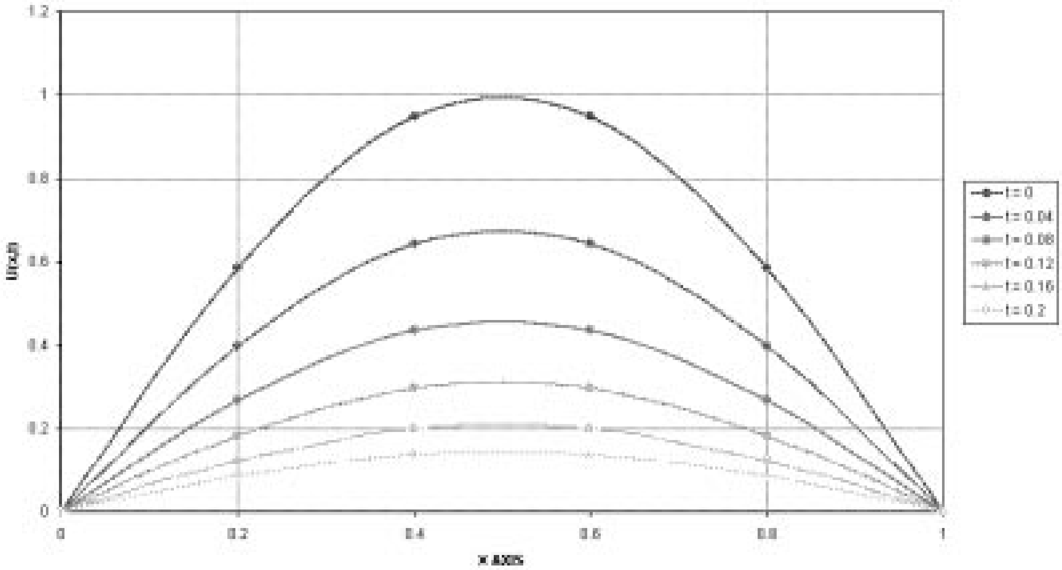

Fig. 7. Chart display for solution of $u_t = u_{xx}$ using the Crank Nicolson method, with $\Delta x = 0.2$ and $\Delta t = 0.04$.

```
Public Function fReplace(formulaWithVariable As String, replacementVariable
As String, replaceValue As String) As String

Dim varPosition As Integer
Dim fragment1 As String
Dim fragment2 As String
Dim formulaToEdit As String
Dim lenReplaceVar As Integer

varPosition = 0
lenReplaceVar = 0

formulaToEdit = formulaWithVariable

varPosition = InStr(1, formulaToEdit, replacementVariable)
lenReplaceVar = Len(replacementVariable)

If varPosition = 0 Then
  fReplace = formulaToEdit
  Exit Function

Else

  Do While varPosition > 0
    fragment1 = Left(formulaToEdit, varPosition - 1)
    fragment2 = Mid(formulaToEdit, varPosition + lenReplaceVar)

    formulaToEdit = fragment1 & replaceValue & fragment2
    varPosition = InStr(1, formulaToEdit, replacementVariable)
  Loop
  fReplace = formulaToEdit

End If

End Function
```

Example Code 1

```
ActiveChart.Axes(xlValue).MajorUnit = 0.1 * (maxScale - minScale)
.
.
.
' Legend Key Color
    ActiveChart.Legend.LegendEntries(1).LegendKey.Interior.ColorIndex = 17

If Not maxScale = minScale Then
    ActiveChart.Legend.LegendEntries(2).LegendKey.Interior.ColorIndex = 37
    ActiveChart.Legend.LegendEntries(3).LegendKey.Interior.ColorIndex = 24
    ActiveChart.Legend.LegendEntries(4).LegendKey.Interior.ColorIndex = 34
    ActiveChart.Legend.LegendEntries(5).LegendKey.Interior.ColorIndex = 35
    ActiveChart.Legend.LegendEntries(6).LegendKey.Interior.ColorIndex = 15
    ActiveChart.Legend.LegendEntries(7).LegendKey.Interior.ColorIndex = 36
    ActiveChart.Legend.LegendEntries(8).LegendKey.Interior.ColorIndex = 40
    ActiveChart.Legend.LegendEntries(9).LegendKey.Interior.ColorIndex = 38
    ActiveChart.Legend.LegendEntries(10).LegendKey.Interior.ColorIndex = 12
End If
```

Example Code 2

CONCLUDING REMARKS

The solver, developed in a spreadsheet program, provides seven finite-difference methods for solving all three different types of PDEs. Macros were created to execute repetitive calculations and

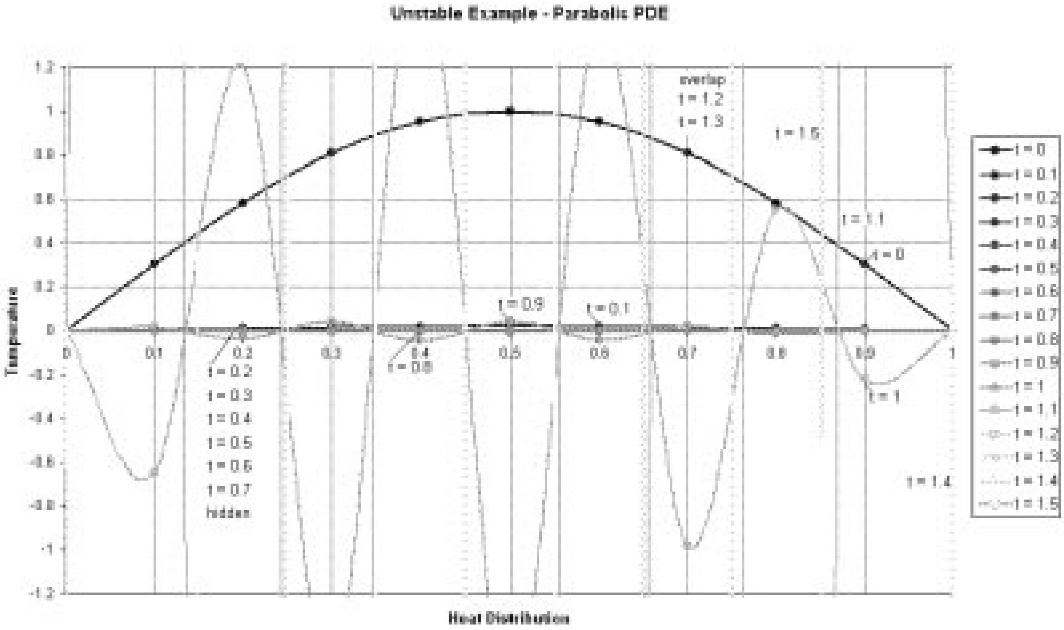

Fig. 8. Chart display for solution of $u_t = u_{xx}$ using Forward-Time-Centered-Space method, with grid Fourier number = 10 and $\Delta t = 0.1\,s$.

procedures, which removes the burden of programming the solving methods. The resemblance of the layout of worksheet cells to the finite-difference grid facilitates placing the numerical solution on the worksheet, providing a simple and quick visualisation of the behaviour of the solution. Graphical display of the numerical solutions is made possible through standard chart options in Excel. The use of colours enhances presentation of solutions. With the aid of the user-friendly graphical user interface, little prior knowledge is required to use the application. Users are freed to focus on their PDE problems, which must be completely defined before solving begins.

Various aspects of the numerical methods can be studied easily. Comparisons can be made in the accuracy between different numerical methods. The stability and convergence characteristics of a numerical method can also be examined easily with the solver. An example to study the stability of the Explicit Forward-Time-Centered-Space method for the heat equation $u_t = u_{xx}$ is shown in Fig. 8 which was obtained by solving the equation deliberately with a grid Fourier number of 10, which violates the stability requirement.

To study the convergence characteristic of the solution for Elliptic PDEs, a feature has been incorporated in the program to automatically generate the convergence history by displaying the sum of the magnitudes of the differences between successive iterates at all grid points in the solution domain at every iterative loop. An

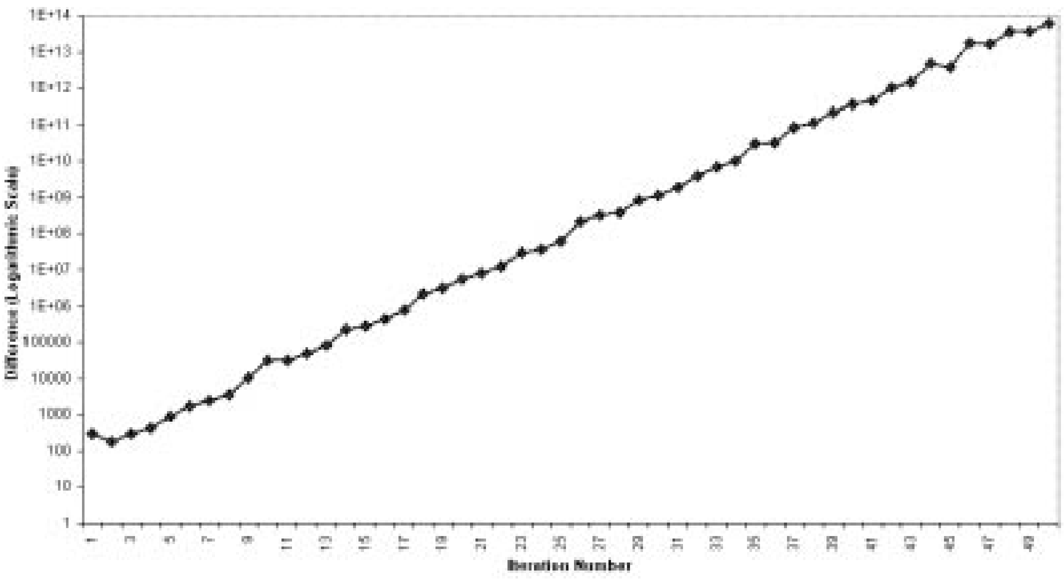

Fig. 9. Divergence of solution for $\nabla^2 u = 12xy$ obtained by SOR with $\omega = 2.75$.

example for the solution of $\nabla^2 u = 12xy$ obtained by SOR with $\omega = 2.75$ is shown in Fig. 9 which clearly shows the diverging behavior of the solution.

As Excel is readily available, using such a solver as an educational tool can be suitable for the classroom without incurring the cost of acquiring additional commercially developed software which is also harder to learn. Copies of the solver may be requested from the author.

REFERENCES

1. Duncan Lawson, Black body fraction infinite series and spreadsheets, *Int. J. Eng. Educ.*, **20**, 2004, pp. 984–990.
2. P. J. Jordan, Spreadsheet-based Method for Thermal Radiation Calculations, *Int. J. Eng. Educ.*, **20**, 2004, pp. 991–998.
3. M. Hagler, Spreadsheet program in antenna design application, *IEEE Trans.*, AP-30, 1986, pp. 585–587
4. D. A. Chapman, Spreadsheet demonstration of discrete and fast Fourier transforms, *Int. J. Elect. Eng. Educ.*, **30**, 1993, pp. 211–215.
5. R. Jackson, A spreadsheet approach to complex transformations, *Teaching Maths and its application*, **12**, 1993, p. 174.
6. J. D. Hoffman, *Numerical Methods for Engineers and Scientists*, McGraw-Hill International, USA (1993).
7. C. Y. Lam, *Applied Numerical Methods for Partial Differential Equations*, Prentice-Hall (1994).
8. E. Kreyszig, *Advanced Engineering Mathematics*, 8th Ed., John Wiley & Sons Inc. USA (1998).

C. Y. Lam holds a B.Sc.(Eng) degree with first class honors and a Ph.D. degree, both in Aeronautical Engineering from the University of London. He is an Associate Professor in the School of Mechanical and Aerospace Engineering at the Nanyang Technological University, Singapore. His current research interests include computational fluid dynamics, mathematical modeling, computer applications and anthropometry.

F. H. Alan Koh graduated in Mechanical Engineering from the Nanyang Technological University, Singapore with a Bachelor degree. His interests are in mathematics and aerospace related topics pursued at leisure time.