NAKHODA

Journal of Algorithms and Computation

journal homepage: http://jac.ut.ac.ir

A Closed-Form Solution for Two-Dimensional Diffusion Equation Using Crank-Nicolson Finite Difference Method

Iman Shojaei*[1] and Hossein Rahami†[2]

[1]Align Technology Inc, San Jose, CA 95134, USA.

[2]School of Engineering Science, College of Engineering, University of Tehran, Tehran, Iran.

ABSTRACT

In this paper a finite difference method for solving 2-dimensional diffusion equation is presented. The method employs Crank-Nicolson scheme to improve finite difference formulation and its convergence and stability. The obtained solution will be a recursive formula in each step of which a system of linear equations should be solved. Given the specific form of obtained matrices, rather than solving the problem in each step using conventional iterative methods, a closed-form solution is formulated.

ARTICLE INFO

Article history:
Received 15, Agust 2018
Received in revised form 18, April 2019
Accepted 4 May 2019
Available online 01, June 2019

Keyword: diffusion equation, finite difference method, convergence and stability.

AMS subject Classification: 05C88.

1 Introduction

Analytical solution of partial differential equations (PDEs) can be laborious or impossible in many cases, particularly when the domain of the problem is complex. Therefore, various numerical methods (e.g., finite difference methods, finite element methods, boundary

*shojaei.iman@uky.edu

†Corresponding author: H. Rahami. Email:hrahami@ut.ac.ir

element methods, ...) have been developed to solve PDEs. These numerical methods are different in their complexity of formulation, rate of convergence, accuracy, and numerical stability. Specifically, finite difference methods are considered among the simplest numerical methods (i.e., in terms of formulation) that are extensively used for solving PDEs. Finite difference methods, however, are inferior to other numerical methods for the rate of convergence. As such, advanced algorithms like Crank-Nicolson method [1, 2] have been developed and used (e.g., wave, heat, and Laplace equations) to improve the conventional finite difference solutions. Crank-Nicolson method is an implicit finite difference method that is numerically stable and uses a time step of second order accuracy.

Sweilam et al. (2012) and Jankowska (2012) used Crank-Nicolson method to solve time fractional diffusion equation and one-dimension heat equation with Robin boundary condition, respectively [5, 8]. Also, Crank-Nicolson method has been used to by Umair et al. (2017) to solve two-dimensional fractional sub-diffusion equation [10].

When employing Crank-Nicolson method to solve, for instance, wave equation of first order, implicit equations involving three different time steps are obtained. As such, to find unknown of the problem in a specific time step, we need to know it in the other two earlier time steps. Therefore, it is convenient to find unknown of the problem in the first two time steps using another method and then apply Crank-Nicolson method to solve the problem in other time steps. The unknown of the problem at the beginning is found using the initial condition and at the next step is found using a method like Backward difference. While using Crank-Nicolson method, matrices of tridiagonal form are gained solution of which is obtained using Gauss-Seidel or Thomas algorithms.

In following sections we first elaborate on formulating diffusion equation using Crank-Nicolson method and then, given the specific form of obtained matrices, develop a closed-form solution for this equation using eigenvalues and eigenvectors [3, 4, 6, 7, 9].

2 Finite Difference Formulation of Diffusion Equation

Developing a simple algorithm for the vibration analysis of global near-regular mechanical systems from available vibration solution of their corresponding regular A wide range of diffusion equations can be solved using analytical solutions (e.g., separation of variables, integral transformation, etc.). However, in more complicated cases numerical solution techniques are required to solve the problem. For instance, the problem systems

$$\begin{aligned} &\alpha^2 u_{xx} = u_t, \quad (0 < x < L,\ 0 < t < \infty) \\ &u\left(0,\ t\right) = p\left(t\right), \quad (0 < t < \infty) \\ &u\left(L,\ t\right) = q\left(t\right), \quad (0 < t < \infty) \\ &u\left(x,\ 0\right) = f\left(x\right), \quad (0 < x < L) \end{aligned} \tag{1}$$

is difficult to be solved using analytical solutions if $p(t)$ and $q(t)$ are not constant, but can readily be solved using a numerical solution like finite difference method.

Discretizing the problem using grid points and approximating $u_t(x,t)$ and $u_{xx}(x,t)$ as

$$\begin{aligned} u_t(x,t) &\approx \frac{u(x,\ t+\Delta t)-u(x,\ t)}{\Delta t} \\ u_{xx}(x,t) &\approx \frac{\frac{u(x+\Delta x,\ t)-u(x,\ t)}{\Delta x}-\frac{u(x,\ t)-u(x-\Delta x,\ t)}{\Delta x}}{\Delta x} \end{aligned} \tag{2}$$

result in finite-difference approximation as follows

$$U_{j,k+1}=rU_{j-1,k}+(1-2r)\,U_{j,k}+rU_{j+1,k} \tag{3}$$

where

$$r=\alpha^2\frac{\Delta t}{(\Delta x)^2} \tag{4}$$

Using Eq. 3 we can calculate U (time k+1) at a grid point as a linear combination of U's at the preceding time (k). As the equations are decoupled, the method is computationally efficient. However, the finite difference method (2) is convergent and stable only if Δt and Δx satisfy the equation

$$r=\alpha^2\frac{\Delta t}{(\Delta x)^2}\le\frac{1}{2} \tag{5}$$

Satisfying the condition in Eq. 5 may result in high computational cost if we choose a small Δx, for the sake of accuracy, that in turn requires a very small Δt (i.e., many time steps). To remove such a restriction, the finite difference formulation can be modified.

3 Implicit Finite Difference Method: Crank-Nicolson Formulation

In construction of Eq. 3 we have approximated $u_x x(x,t)$ using Eq. 2. However, this is not the only possible way of approximation of $u_x x(x,t)$. For instance, if we use some weighted average over the time interval, we will have

$$\begin{aligned} u_{xx}(x,t) \approx\ & (1-\theta)\frac{u(x+\Delta x,\ t)-2u(x,\ t)+u(x-\Delta x,\ t)}{(\Delta x)^2} \\ &+\theta\frac{u(x+\Delta x,\ t+\Delta t)-2u(x,\ t+\Delta t)+u(x-\Delta x,\ t+\Delta t)}{(\Delta x)^2} \end{aligned} \tag{6}$$

where θ is a number in interval [0, 1]. The finite difference formulation will be

$$\begin{aligned} &\alpha^2\left[(1-\theta)\frac{U_{j-1,\ \ k}-2U_{j,k}+U_{j+1,k}}{(\Delta x)^2}+\theta\frac{U_{j-1,\ \ k+1}-2U_{j,k+1}+U_{j+1,k+1}}{(\Delta x)^2}\right] \\ &=\frac{U_{j,k+1}-U_{j,k}}{\Delta t} \end{aligned} \tag{7}$$

which will reduce to Eq. 3 for $\theta = 1$. It can be shown that if $\theta \geq \frac{1}{2}$, Eq. 7 is both convergent and stable for all values of $r > 0$ (i.e., the condition in Eq. 5 is discarded). If we set $\theta = \frac{1}{2}$, we will have

$$\begin{aligned} &- rU_{j-1,k+1} + 2\left(1+r\right)U_{j,k+1} - rU_{j+1,k+1} = rU_{j-1,k} + 2\left(1-r\right)U_{j,k} + rU_{j+1,k} \\ &j = 1,\ 2, \ldots,\ N-1 \qquad \text{and} \qquad k = 0,\ 1,\ 2, \ldots \end{aligned} \tag{8}$$

that is the Crank-Nicolson scheme. Expressing Eq. 8 in matrix form and moving boundary and U values from step k (i.e., $U_(j,k)$) to the right hand side of the matrix equation lead to

$$\begin{bmatrix} 2\left(1+r\right) & -r & & \cdots & 0 \\ -r & 2\left(1+r\right) & -r & & \vdots \\ & -r & \ddots & -r & \\ \vdots & & -r & 2\left(1+r\right) & -r \\ 0 & \cdots & & -r & 2\left(1+r\right) \end{bmatrix} \begin{bmatrix} U_{1,k+1} \\ U_{2,k+1} \\ \vdots \\ U_{N-2,k+1} \\ U_{N-1,k+1} \end{bmatrix} = \begin{bmatrix} rp_{k+1} + rp_k + 2\left(1-r\right)U_{1,k} + rU_{2,k} \\ rU_{1,k} + 2\left(1-r\right)U_{2,k} + rU_{3,k} \\ \vdots \\ rU_{N-3,k} + 2\left(1-r\right)U_{N-2,k} + rU_{N-1,k} \\ rU_{N-2,k} + 2\left(1-r\right)U_{N-1,k} + rq_{k+1} + rq_k \end{bmatrix} \tag{9}$$

or in a compact form

$$\mathbf{AU}_{\mathrm{k+1}} = \mathbf{c} \tag{10}$$

Now starting with $k = 0$, we can solve the matrix equation for unknowns in the first time step (i.e., $U_{1,1}$, $U_{2,1}$, ..., $U_{(N-1,1))}$. Then setting $k = 1$ and solving the matrix equation, the next line of unknowns is found (i.e., $U_1, 2$, $U_2, 2$, , $U_(N-1, 2)$). The procedure is repeated until unknowns in all time steps are calculated [2]. The conventional method of solving Eq. 10 is through an iterative algorithm outlined below:

1. Rewriting A and matrix equation as

$$A = \begin{bmatrix} 2\left(1+r\right) & 0 & & \cdots & 0 \\ 0 & 2\left(1+r\right) & 0 & & \vdots \\ & 0 & \ddots & 0 & \\ \vdots & & 0 & 2\left(1+r\right) & 0 \\ 0 & \cdots & & 0 & 2\left(1+r\right) \end{bmatrix} + \begin{bmatrix} 0 & -r & & \cdots & 0 \\ -r & 0 & -r & & \vdots \\ & -r & \ddots & -r & \\ \vdots & & -r & 0 & -r \\ 0 & \cdots & & -r & 0 \end{bmatrix} = 2\left(1+r\right)\mathbf{I} + \mathbf{A}' \tag{11}$$

and

$$[2\,(1+r)\,\mathbf{I}+\mathbf{A}']\,\mathbf{U}_{\mathrm{k+1}}=\mathbf{c} \tag{12}$$

or

$$\mathbf{U}_{\mathrm{k+1}}=\frac{1}{2\,(1+r)}\mathbf{c}-\frac{1}{2\,(1+r)}\mathbf{A}'\mathbf{U}_{\mathrm{k+1}} \tag{13}$$

2. Setting an initial approximation as $\mathbf{U}^0_{\mathrm{k+1}}=\frac{1}{2(1+r)}\mathbf{c}$ and obtaining a solution

$$\mathbf{U}^1_{\mathrm{k+1}}=\frac{1}{2\,(1+r)}\mathbf{c}-\frac{1}{2\,(1+r)}\mathbf{A}'\mathbf{U}^0_{\mathrm{k+1}} \tag{14}$$

3. Repeating the procedure to define an iterative algorithm

$$\mathbf{U}^{\mathrm{n+1}}_{\mathrm{k+1}}=\frac{1}{2\,(1+r)}\left[\mathbf{c}-\mathbf{A}'\mathbf{U}^{\mathrm{n}}_{\mathrm{k+1}}\right]\qquad \text{and}\qquad n=0,\ 1,\ 2,\ \dots \tag{15}$$

It can be shown that the solution in Eq. 15 converges to exact solution of $\mathbf{AU}_{\mathrm{k+1}}=\mathbf{c}$. However, due to the iterative nature of the algorithm, achieving an accurate enough solution can be time-consuming. In the next section an efficient closed-from solution for Eq. 10 is presented.

4 Closed-Form Solution of $\mathbf{AU}_{\mathbf{k+1}}=\mathbf{c}$ in Implicit Finite Difference Method

Eigenvalues and eigenvectors of a tridiagonal matrix, of dimension N - 1, of the form

$$M=\begin{bmatrix} b & c & & \cdots & 0\\ a & b & c & & \vdots\\ & a & \ddots & c & \\ \vdots & & a & b & c\\ 0 & \cdots & & a & b \end{bmatrix} \tag{16}$$

are calculated [4, 8] using

$$\begin{aligned}
&\lambda_n=b+2\sqrt{ac}\cos\frac{n\pi}{N}\qquad \textit{and}\quad n=1,\ 2\ ,\dots,\ N-1\\
&v^n_j=\rho^{j-1}\sin\frac{nj\pi}{N}\quad ;\quad \rho=\sqrt{\frac{a}{c}}\quad ;\quad j=1,\ 2\ ,\dots,\ N-1\\
&v^n=\left[v^n_1,\ v^n_2,\dots,\ v^n_{N-1}\right]^t
\end{aligned} \tag{17}$$

Therefore, for matrix A we will have

$$\begin{aligned}
&\lambda_n=2\,(1+r)+2r\cos\frac{n\pi}{N}\qquad \text{and}\quad n=1,\ 2\ ,\dots,\ N-1\\
&v^n_j=\sin\frac{nj\pi}{N}\qquad \text{and}\quad j=1,\ 2\ ,\dots,\ N-1\\
&v^n=\left[v^n_1,\ v^n_2,\dots,\ v^n_{N-1}\right]^t
\end{aligned} \tag{18}$$

Since A is a symmetric matrix (of dimension $N-1$), its eigenvectors $(v^1, v^2, ..., v^{(N-1)})$ provide an orthogonal basis for N-1 space. Therefore, we can expand $\mathbf{U}_{\mathrm{k+1}}$ and c in terms of $(v^1, v^2, ..., v^{(N-1)})$ basis:

$$\mathbf{U}_{\mathrm{k+1}} = \sum_{i=1}^{N-1} a_i v^i \quad \text{and} \quad c = \sum_{i=1}^{N-1} c_i v^i \tag{19}$$

where c_i's are known (i.e., can readily be computed)

$$c_i = \left(v^i\right)^t c \tag{20}$$

and a_i's are our unknowns that should be computed through substitution in $\mathbf{U_{k+1}} = \mathbf{c}$

$$\mathbf{A}\sum_{i=1}^{N-1} a_i v^i = \sum_{i=1}^{N-1} c_i v^i \tag{21}$$

where we can write

$$\mathbf{A}\sum_{i=1}^{N-1} a_i v^i = \sum_{i=1}^{N-1} a_i \mathbf{A} v^i = \sum_{i=1}^{N-1} a_i \lambda_i v^i \tag{22}$$

We can now re-express Eq. 21 as

$$\sum_{i=1}^{N-1} \lambda_i a_i v^i = \sum_{i=1}^{N-1} c_i v^i \tag{23}$$

Because v^i's are linearly independent (they form a basis), we will have

$$\lambda_i a_i = c_i \ \rightarrow a_i = \frac{c_i}{\lambda_i} \qquad \text{and} \quad i = 1,\ 2, \ldots,\ N-1 \tag{24}$$

Therefore,

$$\mathbf{U}_{\mathrm{k+1}} = \sum_{i=1}^{N-1} \frac{c_i}{\lambda_i} v^i = \sum_{i=1}^{N-1} v^i \frac{c_i}{\lambda_i} = \sum_{i=1}^{N-1} \frac{v^i \left(v^i\right)^t}{\lambda_i} c \tag{25}$$

Finally,

$$\mathbf{U}_{\mathrm{k+1}} = \sum_{i=1}^{N-1} \frac{\left[\sin\frac{i\pi}{N},\ \sin\frac{2i\pi}{N}, \ldots, \sin\frac{(N-1)i\pi}{N}\right]^t \left[\sin\frac{i\pi}{N},\ \sin\frac{2i\pi}{N}, \ldots, \sin\frac{(N-1)i\pi}{N}\right]}{2\left(1+r\right) + 2r\cos\frac{i\pi}{N}} c \tag{26}$$

References

[1] Crank J. , Nicolson Ph., A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type, Advances in Computational Mathematics, 6(1) (1996) 207-226.

[2] Greenberg M.D., Advanced engineering mathematics, volume 2, Prentice Hall Upper Saddle River, NJ, (1998).

[3] Kaveh A. and Rahami H., Tri-diagonal and penta-diagonal block matrices for efficient eigensolutions of problems in structural mechanics, Acta Mechanica, 192(1-4) (2007) 77-87.

[4] Kaveh A. and Shojaei I. and Rahami H., New developments in the optimal analysis of regular and near-regular structures: decomposition, graph products, force method, Acta Mechanica, 226(3) (2015) 665-681.

[5] Malgorzata J., An interval finite difference method of crank-nicolson type for solving the one-dimensional heat conduction equation with mixed boundary conditions, In International Workshop on Applied Parallel Computing, Springer (2010) 157-167.

[6] Shojaei I. and Rahami H. and Kaveh A., A numerical solution for Laplace and Poisson's equations using geometrical transformation and graph products, Applied Mathematical Modelling, 40(17-18) (2016) 7768-7783.

[7] Shojaei I. and Rahami H. and Kaveh A., Efficient finite element solution of regular and near-regular systems using graph products, Acta Mechanica, 226(7) (2015) 2393-2405.

[8] Sweilam NH. and Khader MM. and Mahdy AMS., Crank-Nicolson finite difference method for solving time-fractional diffusion equation, Journal of Fractional Calculus and Applications, 2(2) (2012) 1-9.

[9] Yueh W.C., Eigenvalues of several tridiagonal matrices, Applied mathematics e-notes,5(66-74) (2005) 210-230.

[10] Umair A. and Farah Aini A. and Ahmad Izani I., Crank-Nicolson finite difference method for two-dimensional fractional sub-diffusion equation, Journal of Interpolation and Approximation in Scientific Computing, 2 (2017) 18-29.