Finite-Difference Approximations to the Heat/Black-Scholes Equation

JEREMY LIAN

Aug 2020

Abstract

This paper presents a practical overview of the numerical solution to a parabolic linear partial differential equation (e.g. heat/Black-Scholes equation) using the finite-difference method. The explicit forward time, centered space (FTCS) scheme, as well as the implicit backward time, centered space (BTCS), and Crank-Nicolson (CN) schemes are developed for a simple problem involving the one-dimensional heat equation. Working PYTHON 3 codes for each scheme are also presented, with the results of running the codes on finer meshes, and with smaller time steps being demonstrated. These sample calculations realize the theoretical predictions of the relationship of truncation errors and stability limits on spatial and temporal discretisation step size.

Contents

1 Heat Diffusion Equation

Partial differential equations (PDEs) can be linear or non-linear. In this article, we will look at a specific case of a linear PDE that is widely recognisable in Physics - the heat diffusion equation[1]. For a linear PDE, 3 classifications exist: ellipse, parabola and hyperbola, very much analogous to the conic sections in Euclidean geometry described by quadratic equations. Knowledge of these classes are important for the computational scientist because different numerical methods are required for the different classes of linear PDEs.

1.1 On (Parabolic) Linear PDE

Let us consider a function in two variables $F(t, x)$. The general linear PDE of second order can then be written as:

$$aF_{xx} + 2bF_{x,t} + cF_{tt} + dF_x + eF_t + gF = h \tag{1}$$

where the subscripts denote the derivatives taken at the respective variables.

It is easy to see that Eq.1 is associated with a polynomial of degree 2. This polynomial, and the corresponding PDE, is then classified as hyperbolic, parabolic, or elliptic according as its discriminant:

$$b^2 - 4ac > 0 \quad \textit{hyperbolic} \tag{2}$$

$$b^2 - 4ac = 0 \quad \textit{parabolic} \tag{3}$$

$$b^2 - 4ac < 0 \quad \textit{elliptic} \tag{4}$$

The one dimensional (1D) heat diffusion equation is known to be:

$$\boxed{\frac{\partial \phi}{\partial t} = \alpha \frac{\partial^2 \phi}{\partial x^2}, \quad 0 \leq x \leq L, \quad t \geq 0} \tag{5}$$

where $\phi(t, x)$ is a function of transient heat conduction in a 1D material of length L, and α is a constant coefficient representing the thermal diffusivity of the material.

It is obvious that the heat equation in 5 is a parabolic linear PDE, which will be the focus of the remaining sections of this article. The interested reader can refer to Ref. 1 for a better treatment of the other classes of linear PDEs.

The domain of the solution to the heat equation is a semi-infinite strip of width L that continues indefinitely in time. However, under practical computation, the solution is obtained only for some finite time $t_{\max}$. An attempt at a unique solution requires the specification of boundary conditions at $x = 0$ and $x = L$, as well as an initial value at $t = 0$. Many different types of boundary consitions can be specified, e.g. Neumann (gradient), Dirichlet (constant) etc. In the rest of this article, we consider the following Dirichlet boundary conditions and initial value:

$$\boxed{\phi(t, 0) = \phi_0, \quad \phi(t, L) = \phi_L, \quad \phi(0, x) = f(x)} \tag{6}$$

[1]The Black–Scholes equation in options pricing has the exact form of a heat diffusion equation!

2 Finite-Difference Method

Numerical solutions to the heat equation in Eq.5 can be obtained through finite-difference method. In this methods, solutions to the continuous PDE is replaced with a discrete approximation, i.e. the numerical solution is known only at a finite number of points in the physical domain (mesh). In general, increasing the number of points can increase resolution and therefore accuracy of the numerical solution. This discrete approximation results in a set of algebraic equations that can then be easily evaluated with linear algebra. The core idea of the finite-difference method is to replace continuous derivatives with difference formulae that involve only the discrete values associated with positions on the mesh.

In our heat equation, derivative with respect to time $\frac{\partial}{\partial t}$ exists alongside derivative with respect to space $\frac{\partial}{\partial x}$. Using different combinations of mesh points in our difference formulae results in vastly different schemes. In the limit as the mesh spacings ($\Delta x and \Delta t$) go to zero, the numerical solution obtained with any useful scheme will approach the true solution to the original differential equation. However, the rate at which this numerical solution approaches the true solution varies with the choice of scheme. In addition, there are some schemes which may be computationally practical but can fail to yield a solution for bad combinations of Δx and Δt. In this paper, we introduce 3 different schemes for the solution to 1-D heat equation.

The numerical solutions to different classes of PDEs are discussed in many textbooks. The inquisitive reader should refer to Cooper[2] for a modern introduction to the theory of PDEs along with a brief coverage of different numerical methods. Fletcher[3], Golub and Ortega[4], and Hoffman[5] offer a more applied approach that also introduces some code implementation issues.

2.1 About the Discrete Mesh

Let us now discretise our spatial domain into N uniformly spaced nodes x_i in the interval $0 \leq x \leq L$ such that

$$x_i = (i-1)\Delta x, \quad i = 1, 2, \ldots N \tag{7}$$

It follows then that

$$\Delta x = \frac{L}{N-1} \tag{8}$$

We can do the same for our temporal domain in the interval $0 \leq t \leq t_{\max}$:

$$t_m = (m-1)\Delta t, \quad m = 1, 2, \ldots M \tag{9}$$

where M is the total number of time steps and the size of one time step Δt is

$$\Delta t = \frac{t_{\max}}{M-1} \tag{10}$$

Figure 1 depicts the discretisation of the solution space (mesh). In Table 1, we introduce some notations used in later sections of this paper.

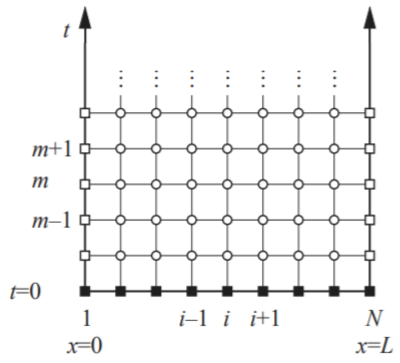

Figure 1: A semi-infinite strip depicting the mesh used for numerical solution to the one-dimensional heat equation. The solid squares indicate the (known) initial values. The open squares indicate the (known) boundary values. The open circles indicate the position of the interior points where the finite difference approximation is computed.

Notation	Meaning
$\phi(t,x)$	Continuous (true) solution
$\phi(t_m,x_i)$	Continuous solution evaluated at a mesh point.
ϕ_i^m	Approximate numerical solution obtained by solving the finite-difference equations.

Table 1: Notations used in later sections of this paper.

2.2 On Finite-Difference Approximations

It is easier to first develop the finite difference formulae with only one independent variable, x, i.e. $\phi = \phi(x)$. The resulting formulae can then be used to approximate derivatives with respect to either space or time without any loss of generality in the result.

2.2.1 First-Order Forward Difference

Let us proceed by considering a Taylor series expansion of $\phi(x)$ about the point x_i:

$$\phi(x_i+\delta x) = \phi(x_i) + \delta x \left.\frac{\partial \phi}{\partial x}\right|_{x_i} + \frac{\delta x^2}{2}\left.\frac{\partial^2\phi}{\partial x^2}\right|_{x_i} + \frac{\delta x^3}{3!}\left.\frac{\partial^3\phi}{\partial x^3}\right|_{x_i} + \cdots \tag{11}$$

where δx is a change in x relative to x_i. Let $\delta x = \Delta x$ in Eq.11, i.e. consider the value of ϕ at the location of the x_{i+1} mesh line:

$$\phi(x_i+\Delta x) = \phi(x_i) + \Delta x \left.\frac{\partial \phi}{\partial x}\right|_{x_i} + \frac{\Delta x^2}{2}\left.\frac{\partial^2\phi}{\partial x^2}\right|_{x_i} + \frac{\Delta x^3}{3!}\left.\frac{\partial^3\phi}{\partial x^3}\right|_{x_i} + \cdots \tag{12}$$

Making $\left.\frac{\partial\phi}{\partial x}\right|_{x_i}$ the subject of the equation,

$$\left.\frac{\partial \phi}{\partial x}\right|_{x_i} = \frac{\phi(x_i+\Delta x) - \phi(x_i)}{\Delta x} - \frac{\Delta x}{2}\left.\frac{\partial^2\phi}{\partial x^2}\right|_{x_i} - \frac{\Delta x^2}{3!}\left.\frac{\partial^3\phi}{\partial x^3}\right|_{x_i} + \cdots \tag{13}$$

Substituting the approximate solution for the exact solution, i.e., use $\phi_i \approx \phi(x_i)$ and $\phi_{x+1} \approx \phi(x_i+\Delta x)$,

$$\left.\frac{\partial \phi}{\partial x}\right|_{x_i} \approx \frac{\phi_{i+1} - \phi_i}{\Delta x} - \frac{\Delta x}{2}\left.\frac{\partial^2\phi}{\partial x^2}\right|_{x_i} - \frac{\Delta x^2}{3!}\left.\frac{\partial^3\phi}{\partial x^3}\right|_{x_i} + \cdots \tag{14}$$

The derivative terms on the right hand side of Eq.14 make up the truncation error of the finite difference approximation. This is the error that results from truncating the series. Since the function $\phi(t,x)$ is unknown, $\frac{\partial^2\phi}{\partial x^2}$ cannot be computed. Although the exact magnitude of the truncation error cannot be determined, we can use the "big O" notation to express the dependence of the truncation error on the mesh spacing. Thus, we rewrite Eq.14:

$$\boxed{\left.\frac{\partial \phi}{\partial x}\right|_{x_i} = \frac{\phi_{i+1} - \phi_i}{\Delta x} + \mathcal{O}(\Delta x)} \tag{15}$$

Eq.15 is called the *forward difference* or *forward Euler* formula for $\left.\frac{\partial\phi}{\partial x}\right|_{x_i}$ because it involves nodes x_i and x_{i+1}. The forward difference approximation has a truncation error that is significantly dependent on the magnitude of Δx.

2.2.2 First-Order Backward Difference

An alternative first order finite difference formula is obtained if we rewrite the Taylor series in Eq.11 with $\delta x = -\Delta x$. Thus, we have

$$\left.\frac{\partial \phi}{\partial x}\right|_{x_i} = \frac{\phi(x_i) - \phi(x_i-\Delta x)}{\Delta x} + \frac{\Delta x}{2}\left.\frac{\partial^2\phi}{\partial x^2}\right|_{x_i} - \frac{\Delta x^2}{3!}\left.\frac{\partial^3\phi}{\partial x^3}\right|_{x_i} + \cdots \tag{16}$$

Using the discrete mesh variables in place of all the unknowns, the *backward difference* or *backward Euler* formula using "big O" notation is then

$$\boxed{\left.\frac{\partial \phi}{\partial x}\right|_{x_i} = \frac{\phi_i - \phi_{i-1}}{\Delta x} + \mathcal{O}(\Delta x)} \tag{17}$$

where the order of magnitude of the truncation error for the backward difference approximation is the same as that of the forward difference approximation. In the next section, we obtain a first order difference formula for $\left.\frac{\partial \phi}{\partial x}\right|_{x_i}$ with a smaller truncation error.

2.2.3 First-Order Central Difference

We begin with the Taylor series expansions for ϕ_{i+1} and ϕ_{i-1}:

$$\phi_{i+1} = \phi_i + \Delta x \left.\frac{\partial \phi}{\partial x}\right|_{x_i} + \frac{\Delta x^2}{2}\left.\frac{\partial^2 \phi}{\partial x^2}\right|_{x_i} + \frac{(\Delta x)^3}{3!}\left.\frac{\partial^3 \phi}{\partial x^3}\right|_{x_i} + \cdots \tag{18}$$

$$\phi_{i-1} = \phi_i - \Delta x \left.\frac{\partial \phi}{\partial x}\right|_{x_i} + \frac{\Delta x^2}{2}\left.\frac{\partial^2 \phi}{\partial x^2}\right|_{x_i} - \frac{(\Delta x)^3}{3!}\left.\frac{\partial^3 \phi}{\partial x^3}\right|_{x_i} + \cdots \tag{19}$$

Subtracting Eq.19 from Eq.18, and making $\left.\frac{\partial \phi}{\partial x}\right|_{x_i}$ the subject of the equation then yields:

$$\left.\frac{\partial \phi}{\partial x}\right|_{x_i} = \frac{\phi_{i+1} - \phi_{i-1}}{2\Delta x} - \frac{(\Delta x)^2}{3!}\left.\frac{\partial^3 \phi}{\partial x^3}\right|_{x_i} + \cdots \tag{20}$$

or, in order notation,

$$\boxed{\left.\frac{\partial \phi}{\partial x}\right|_{x_i} = \frac{\phi_{i+1} - \phi_{i-1}}{2\Delta x} + \mathcal{O}\left(\Delta x^2\right)} \tag{21}$$

This is the *central difference* approximation to $\left.\frac{\partial \phi}{\partial x}\right|_{x_i}$. When $\Delta x \ll 1$, the truncation error for the central difference approximation becomes much smaller than the truncation errors in the forward or backward difference approximations. However, there exist a complication with Eq.21: it does not account for the value of ϕ_i. This may cause problems when the central difference formula is included in an approximation to a differential equation.

2.2.4 Second-Order Central Difference

Finite difference approximations to higher order derivatives can be obtained by manipulating the Taylor Series expansions. Adding Eq.18 and 19, for example, yields

$$\phi_{i+1} + \phi_{i-1} = 2\phi_i + (\Delta x)^2 \left.\frac{\partial^2 \phi}{\partial x^2}\right|_{x_i} + \frac{2(\Delta x)^4}{4!}\left.\frac{\partial^4 \phi}{\partial x^4}\right|_{x_i} + \cdots \tag{22}$$

Solving for $\left.\frac{\partial^2 \phi}{\partial x^2}\right|_{x_i}$ gives

$$\left.\frac{\partial^2 \phi}{\partial x^2}\right|_{x_i} = \frac{\phi_{i+1} - 2\phi_i + \phi_{i+1}}{\Delta x^2} + \frac{(\Delta x)^2}{12}\left.\frac{\partial^4 \phi}{\partial x^4}\right|_{x_i} + \cdots \tag{23}$$

i.e.,

$$\boxed{\left.\frac{\partial^2 \phi}{\partial x^2}\right|_{x_i} = \frac{\phi_{i+1} - 2\phi_i + \phi_{i+1}}{\Delta x^2} + \mathcal{O}\left(\Delta x^2\right)} \tag{24}$$

This is the central difference approximation to the second derivative.

3 Numerical Schemes for the Heat Equation

The finite difference approximations developed in the preceding section can now be assembled into a discrete approximation for the heat equation in Eq.5. Both the time and space derivatives are replaced by finite differences. Here, we introduce superscript m to designate the time step of the discrete solution, in addition to a subscript i denoting the spatial step. We shall soon see that choosing the time step at which the spatial derivatives are evaluated will have a large impact on the performance and ease of implementation of the finite difference model. Figure 2 depicts the various finite-difference approximation schemes to the heat equation using their computational molecules or stencil representations.

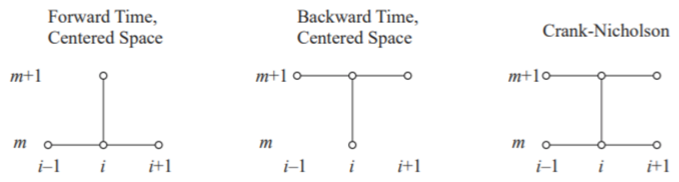

Figure 2: Stencil representations of various finite-difference approximations to the heat equation.

3.1 Forward Time, Centered Space

Under the Forward Time, Centered Space (FTCS) scheme, we approximate the time derivative in Eq.5 with a forward difference:

$$\left.\frac{\partial \phi}{\partial t}\right|_{t_{m+1},x_i} = \frac{\phi_i^{m+1} - \phi_i^m}{\Delta t} + \mathcal{O}(\Delta t) \tag{25}$$

We then use the central difference approximation for the spatial second-order derivative $\left.\frac{\partial^2 \phi}{\partial x^2}\right|_{x_i}$:

$$\left.\frac{\partial^2 \phi}{\partial x^2}\right|_{x_i} = \frac{\phi_{i-1}^m - 2\phi_i^m + \phi_{i+1}^m}{\Delta x^2} + \mathcal{O}\left(\Delta x^2\right) \tag{26}$$

Substituting Eq.25 and 26 into the heat equation in 5 gives

$$\frac{\phi_i^{m+1} - \phi_i^m}{\Delta t} = \alpha \frac{\phi_{i-1}^m - 2\phi_i^m + \phi_{i+1}^m}{\Delta x^2} + \mathcal{O}(\Delta t) + \mathcal{O}\left(\Delta x^2\right) \tag{27}$$

Ignoring the truncation error terms for the moment and solving for ϕ_i^{m+1} yields

$$\boxed{\phi_i^{m+1} = r\phi_{i+1}^m + (1-2r)\phi_i^m + r\phi_{i-1}^m} \tag{28}$$

where $r = \frac{\Delta t}{\Delta x^2}$.

The FTCS scheme is easy to implement; the entire solution is contained within two loops: an outer loop over all time steps, and an inner loop over all interior spatial nodes[2]. Notice that the value of ϕ_i^{m+1} does not depend on its direct spatial neighbours ϕ_{i-1}^{m+1} or ϕ_{i+1}^{m+1}, i.e. the FTCS scheme is an explicit method. Thus, this scheme behaves more like a the solution to a hyperbolic differential equation than a parabolic differential equation. It is known that the solution to a parabolic linear PDE like Eq.5 subject to the initial value and boundary conditions in Eq.6 will be a bounded, decaying function. In other words, the magnitude of the solution will decrease from an initial value to a constant. The FTCS can yield unstable solutions that oscillate and grow if Δt is too large. Stable solutions with the FTCS scheme are only obtained if the following condition is satisfied:

$$r = \frac{\alpha \Delta t}{\Delta x^2} \leq \frac{1}{2} \tag{29}$$

The stability condition for FTCS scheme will not be proven in this paper. The interested reader can refer to Ref. 6.

[2]The inner *for*-loop could be easily vectorized in MATLAB.

Finally, we note that Eq.28 can be expressed as a matrix multiplication:

$$\phi^{(m+1)} = A\phi^{(m)} \tag{30}$$

where $\phi^{(m+1)}$ is the vector of ϕ values of at time step $m+1$, and $\phi^{(m)}$ is the vector of ϕ values at time step m.
The matrix A is then the tridiagonal matrix with its first and last rows adjusted to reflect the Dirichlet boundary conditions in Eq.6:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ r & (1-2r) & r & 0 & 0 & 0 \\ 0 & r & (1-2r) & r & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & r & (1-2r) & r \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{31}$$

3.2 Backward Time, Centered Space

The Backward Time, Centered Space (BTCS) scheme differs from FTC in that we approximate the time derivative in Eq.5 with a backward difference instead. Hence, this yields for finite difference approximation to the heat equation:

$$\frac{\phi_i^m - \phi_i^{m-1}}{\Delta t} = \alpha \frac{\phi_{i-1}^m - 2\phi_i^m + \phi_{i+1}^m}{\Delta x^2} + \mathcal{O}(\Delta t) + \mathcal{O}\left(\Delta x^2\right) \tag{32}$$

The truncation errors in this BTCS scheme have the same order of magnitude as the truncation errors in the FTCS method. Ignoring the truncation error terms for now and rearranging the equation then yields:

$$\boxed{-\frac{\alpha}{\Delta x^2}\phi_{i-1}^m + \left(\frac{1}{\Delta t} + \frac{2\alpha}{\Delta x^2}\right)\phi_i^m - \frac{\alpha}{\Delta x^2}\phi_{i+1}^m = \frac{1}{\Delta t}\phi_i^{m-1}} \tag{33}$$

Notice that the value of ϕ_i^m depends on its direct spatial neighbours ϕ_{i-1}^m and ϕ_{i+1}^m, i.e. the BTCS scheme is an implicit method. Thus, Eq.33 actually represents a system of equations for the values of ϕ at the internal nodes of the spatial mesh ($i = 2, 3, \cdots, N-1$). This system of equations can be represented in matrix form as

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & a_{N-1} & b_{N-1} & c_{N-1} \\ 0 & 0 & 0 & 0 & a_N & b_N \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \vdots \\ \phi_{N-1} \\ \phi_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{N-1} \\ d_N \end{bmatrix} \tag{34}$$

where the interior nodes have the values

$$a_i = -\alpha/\Delta x^2 \tag{35}$$
$$b_i = (1/\Delta t) + \left(2\alpha/\Delta x^2\right) \tag{36}$$
$$c_i = -\alpha/\Delta x^2 \tag{37}$$
$$d_i = (1/\Delta t)\phi_i^{m-1} \tag{38}$$
$$i = 2, 3, \ldots, N-1 \tag{39}$$

Under the Dirichlet boundary conditions in Eq.6,

$$b_1 = 1, \quad c_1 = 0, \quad d_1 = \phi_0 \tag{40}$$
$$a_N = 0, \quad b_N = 1, \quad d_n = \phi_L \tag{41}$$

The first and last rows of the matrix will be different under different types of boundary conditions. The interested reader can refer to Ref. 7 for a treatment of the Neumann boundary conditions.

Implementation of the BTCS scheme requires solving a system of equations at each time step. This makes the BTCS scheme much more computationaly expensive than the FTCS scheme. However, the BTCS scheme has one huge advantage: it is unconditionally stable for numerical solutions to parabolic linear PDEs and yet just as accurate as the FTCS scheme. Therefore, the BTCS scheme can yield a computational model that is robust to choices of Δt and Δx. But this advantage is not always overwhelming, however, and the FTCS scheme can still be very useful for some situations.

3.3 Crank-Nicolson Method

The FTCS and BTCS schemes have a temporal truncation error of $\mathcal{O}(\Delta t)$. When time-accurate solutions are important, the Crank-Nicolson (CN) method has significant advantages. The CN scheme is not significantly more difficult to implement than the BTCS scheme, and it has a temporal truncation error that is $\mathcal{O}(\Delta t^2)$. The CN scheme, like BTCS, is an implicit method – it is also unconditionally stable for numerical solutions to parabolic linear PDEs[8, 9, 10].

Under the CN scheme, the time derivative is approximated with backward difference similar to the BTCS scheme. The second-order spatial derivative is, however, approximated with the average of the central difference scheme evaluated at the current and previous time step. Thus, the heat equation in 5 becomes

$$\frac{\phi_i^m - \phi_i^{m-1}}{\Delta t} = \frac{\alpha}{2}\left[\frac{\phi_{i-1}^m - 2\phi_i^m + \phi_{i+1}^m}{\Delta x^2} + \frac{\phi_{i-1}^{m-1} - 2\phi_i^{m-1} + \phi_{i+1}^{m-1}}{\Delta x^2}\right] \tag{42}$$

Rearranging Eq.42 so that values of ϕ at time m are on the left, and values of ϕ at time $m-1$ are on the right gives:

$$-\frac{\alpha}{2\Delta x^2}\phi_{i-1}^m + \left(\frac{1}{\Delta t} + \frac{\alpha}{\Delta x^2}\right)\phi_i^m - \frac{\alpha}{2\Delta x^2}\phi_{i+1}^m = \tag{43}$$

$$\frac{\alpha}{2\Delta x^2}\phi_{i-1}^{m-1} + \left(\frac{1}{\Delta t} - \frac{\alpha}{\Delta x^2}\right)\phi_i^{m-1} + \frac{\alpha}{2\Delta x^2}\phi_{i+1}^{m-1} \tag{44}$$

Uncoincidentally, the implicit CN method yields a system of equations for ϕ that has to be solved at each time step, mirroring the form in Eq.34 but with differing coefficients of the interior nodes:

$$a_i = -\alpha/\left(2\Delta x^2\right) \tag{45}$$

$$b_i = (1/\Delta t) + \left(\alpha/\Delta x^2\right) \tag{46}$$

$$c_i = -\alpha/\left(2\Delta x^2\right) \tag{47}$$

$$d_i = (1/\Delta t)\phi_i^{m-1} - a_i\phi_{i-1}^{m-1} + (a_i + c_i)\phi_i^{m-1} - c_i\phi_{i+1}^{m-1} \tag{48}$$

$$i = 2, 3, \ldots, N-1 \tag{49}$$

It should be obvious that nodes along the boundaries are similar to the ones in BTCS scheme since the same boundary conditions apply. Thus, the values of $a_N, b_1 = 1, d_1$ and d_n remain similar to the ones in Eq.41. The CN scheme has a truncation error of $\mathcal{O}\left(\Delta t^2\right) + \mathcal{O}\left(\Delta x^2\right)$ i.e., the temporal truncation error is significantly smaller than that of the BTCS scheme.

4 Adventures with code - Implementing the Schemes

PYTHON 3 versions of the FTCS, BTCS, and CN schemes are presented and demonstrated in this section.

4.1 The Analytical Solution

Consider the heat equation with boundary conditions $\phi(t, 0) = \phi(t, L) = 0$, and initial values $f_0(x) = \sin(\pi x/L)$. The general solution for the equation is found to be[11]

$$\phi(t, x) = \sum_{n=1}^{\infty} b_n \sin\left(\frac{n\pi x}{L}\right) \exp\left(-\frac{\alpha n^2 \pi^2 t}{L^2}\right) \tag{50}$$

where the values of b_n are obtained as follows

$$b_n = \frac{2}{L} \int_0^L f_0(x) \sin\left(\frac{n\pi x}{L}\right) dx \tag{51}$$

The exact solution to the heat equation subject to our given initial and boundary conditions is then easily determined to be

$$\phi(x, t) = \sin\left(\frac{\pi x}{L}\right) \exp\left(-\frac{\alpha \pi^2 t}{L^2}\right) \tag{52}$$

4.2 Code Implementation

All 3 numerical schemes schemes are implemented in the PYTHON 3 scripts HEATFTCS, HEATBTCS and HEATCN of Listing 1, 2, and 3 of Appendix A respectively. The systems of equations for both the BTCS and CN methods can be solved using matrix decompositions. For the interested reader, Ref.12 provides a derivation of the solution method for a tridiagonal system of equations using *LU decomposition*, which can then be easily implemented in PYTHON 3. However, we took the easier path and chose not to “reinvent the wheel” by making use of the ever so helpful linear algebra package SCIPY.LINALG.

4.3 On Convergence & Stability

The heat equation is a model of diffusive systems. For all problems with constant boundary values, the solutions of the heat equation decay from an initial state to a non-varying steady state condition. The transient behavior of these solutions are smooth and bounded: the solution does not develop local or global maxima that are outside the range of the initial values.

In section 3.1, it was asserted that the FTCS scheme can exhibit instability. An unstable numerical solution is one in which the values of ϕ at the interior nodes may oscillate or the magnitude grow outside the bounds of the initial and boundary conditions. The convergence of the numerical solution towards the exact analytical solution will depend on the stability condition of Eq.29.

5 On Verification of Error Estimates

Let us focus on the truncation error for FTCS/BTCS scheme – $\mathcal{O}(\Delta t) + \mathcal{O}\left(\Delta x^2\right)$. The order notation expresses the rate at which the truncation error goes to zero, i.e. we are only interested in the order of magnitude of the truncation error. For code validation, however, we need to work with the magnitude of the truncation error. Let TE denote the true magnitude of the truncation error for a given Δt and Δx:

$$\mathrm{TE} = K_t \Delta t + K_x \Delta x^2 \tag{53}$$

where K_t and K_x are constants that depend on the accuracy of the finite difference approximations and the problem being solved.

To verify that a FTCS/BTCS code is working properly, we wish to determine whether a linear reduction in Δt causes a linear reduction in TE. Likewise, we want to determine whether a linear reduction in Δx causes a quadratic reduction in TE.

Given the numerical solution ϕ_i^m at time step m, one scalar measure of the truncation error is the L_2 norm[3]:

$$e = \left\| \phi_i^m - \phi\left(x_i, t_m\right) \right\|_2 \tag{54}$$

We could have also used the L_1 or L_∞ norms. The value of e is used to verify that the FTCS and BTCS solutions have truncation errors that scales as $\mathcal{O}(\Delta t) + \mathcal{O}\left(\Delta x^2\right)$. In other words, e can be used in place of TE when validating our code implementation.

5.1 Towards Error Measurements

[3]In Linear Algebra, there are different ways to measure the magnitude of vectors. L_2 norm is the common Euclidean norm – the shortest distance to go from one point to another.

References

[1] Verena M. Umar. 3 classification of linear pdes in two independent variables. URL `https://www.phy.ornl.gov/csep/pde/node3.html`.

[2] Jeffery Cooper. *Introductin to Partial Differential Equations with Matlab*. Birkhauser, 1998.

[3] Clive A.J. Fletcher. *Computational Techniquess for Fluid Dynamics*. Springer-Verlag, 1998.

[4] Gene Golub; James M. Ortega. *Scientific Computing: An Introduction with Parallel Computing*. Academic Press, Inc., 1993.

[5] Joe D. Hoffman. *Numerical Methods for Engineers and Scientists*. McGrawHill, 1992.

[6] K.W. Morton; D.F. Mayers. *Numerical Solution of Partial Differential Equations: An Introduction*. Cambridge University Press, 1994.

[7] Vivi Andasari. Finite difference methods ii (time-dependent pdes). URL `http://people.bu.edu/andasari/courses/Fall2015/LectureNotes/Lecture15_29Oct2015.pdf`.

[8] William F. Ames. *Numerical Methods for Partial Differential Equations*. Academic Press, Inc., 1992.

[9] R. L. Burden; J. D. Faires. *Numerical Analysis*. Brooks/Cole Publishing Co., 1997.

[10] Eugene Isaacson; Herbert Bishop Keller. *Analysis of Numerical Methods*. Dover, 1994.

[11] Erwin Kreyszig. *Advanced Engineering Mathematics*. Wiley, 1993.

[12] Gerald W. Recktenwald. Finite-difference approximations to the heat equation. URL `http://dma.dima.uniroma1.it/users/lsa_adn/MATERIALE/FDheat.pdf`.

A Code Listings

heatFTCS	PYTHON 3 implementation of the FTCS scheme to solve the problem defined in section 4.1
heatBTCS	PYTHON 3 implementation of the BTCS scheme to solve the problem defined in section 4.1
heatCN	PYTHON 3 implementation of the CN scheme to solve the problem defined in section 4.1
matrix_solve	Solve the system of equations from the BTCS and CN schemes.
show_heat	Plot the results of solving the heat equation.