Finite difference methods for diffusion processes

Hans Petter Langtangen[1,2]

[1]Center for Biomedical Computing, Simula Research Laboratory
[2]Department of Informatics, University of Oslo

Dec 8, 2013

Note: **VERY PRELIMINARY VERSION**

Contents

List of Exercises

The 1D diffusion equation

ıdexdiffusion equation, 1D indexheat equation, 1D

The famous *diffusion equation*, also known as the *heat equation*, reads

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2},$$

here $u(x,t)$ is the unknown function to be solved for, x is a coordinate in ɔace, and t is time. The coefficient α is the *diffusion coefficient* and determines ow fast u changes in time. A quick short form for the diffusion equation is $_t = \alpha u_{xx}$.

Compared to the wave equation, $u_{tt} = c^2 u_{xx}$, which looks very similar, but ıe diffusion equation features solutions that are very different from those of the ave equation. Also, the diffusion equation makes quite different demands to ıe numerical methods.

Typical diffusion problems may experience rapid change in the very beginning, ut then the evolution of u becomes slower and slower. The solution is usually ery smooth, and after some time, one cannot recognize the initial shape of u. his is in sharp contrast to solutions of the wave equation where the initial shape preserved - the solution is basically a moving initial condition. The standard ave equation $u_{tt} = c^2 u_{xx}$ has solutions that propagates with speed c forever, ithout changing shape, while the diffusion equation converges to a *stationary olution* $\bar{u}(x)$ as $t \to \infty$. In this limit, $u_t = 0$, and $\bar{u}$ is governed by $\bar{u}''(x) = 0$. his stationary limit of the diffusion equation is called the *Laplace* equation and rises in a very wide range of applications throughout the sciences.

It is possible to solve for $u(x,t)$ using a explicit scheme, but the time step estrictions soon become much less favorable than for an explicit scheme for the ave equation. And of more importance, since the solution u of the diffusion quation is very smooth and changes slowly, small time steps are not convenient nd not required by accuracy as the diffusion process converges to a stationary ate.

.1 The initial-boundary value problem for 1D diffusion

o obtain a unique solution of the diffusion equation, or equivalently, to apply umerical methods, we need initial and boundary conditions. The diffusion quation goes with one initial condition $u(x,0) = I(x)$, where I is a prescribed unction. One boundary condition is required at each point on the boundry, which in 1D means that u must be known, u_x must be known, or some ombination of them.

We shall start with the simplest boundary condition: $u = 0$. The complete itial-boundary value diffusion problem in one space dimension can then be ɔecified as

$$\begin{aligned}
\frac{\partial u}{\partial t} &= \alpha \frac{\partial^2 u}{\partial x^2}, \quad x \in (0,L),\ t \in (0,T] \\
u(x,0) &= I(x), && x \in [0,L] \\
u(0,t) &= 0, && t > 0, \\
u(L,t) &= 0, && t > 0\,.
\end{aligned}$$

Equation (1) is known as a one-dimensional *diffusion equation*, also often to as a *heat equation*. With only a first-order derivative in time, only on *condition* is needed, while the second-order derivative in space leads to a for two *boundary conditions*. The parameter α must be given and is ref as the *diffusion coefficient*.

Diffusion equations like (1) have a wide range of applications thr physical, biological, and financial sciences. One of the most common appl is propagation of heat, where $u(x,t)$ represents the temperature of some su at point x and time t. Section **??** goes into several widely occurring appl

1.2 Forward Euler scheme

The first step in the discretization procedure is to replace the domain $[0, L$ by a set of mesh points. Here we apply equally spaced mesh points

$$x_i = i\Delta x, \quad i = 0, \ldots, N_x,$$

and

$$t_n = n\Delta t, \quad n = 0, \ldots, N_t\,.$$

Moreover, u_i^n denotes the mesh function that approximates $u(x_i, t_n)$ $0, \ldots, N_x$ and $n = 0, \ldots, N_t$. Requiring the PDE (1) to be fulfilled at point (x_i, t_n) leads to the equation

$$\frac{\partial}{\partial t} u(x_i, t_n) = \alpha \frac{\partial^2}{\partial x^2} u(x_i, t_n),$$

The next step is to replace the derivatives by finite difference approxi The computationally simplest method arises from using a forward diffe time and a central difference in space:

$$[D_t^+ u = \alpha D_x D_x u]_i^n\,.$$

Written out,

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \alpha \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}\,.$$

We have turned the PDE into algebraic equations, also often called equations. The key property of the equations is that they are algebrai

ıakes them easy to solve. As usual, we anticipate that u_i^n is already computed ıch that u_i^{n+1} is the only unknown in (7). Solving with respect to this unknown easy:

$$u_i^{n+1} = u_i^n + \alpha\frac{\Delta t}{\Delta x^2}\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right). \tag{8}$$

The computational algorithm then becomes

1. compute $u_i^0 = I(x_i)$for $i = 0, \ldots, N_x$
2. for $n = 0, 1, \ldots, N_t$:
   (a) apply (8) for all the internal spatial points $i = 1, \ldots, N_x - 1$
   (b) set the boundary values $u_i^{n+1} = 0$ for $i = 0$ and $i = N_x$

he algorithm is compactly fully specified in Python:

```
= linspace(0, L, Nx+1)    # mesh points in space
dx = x[1] - x[0]
= linspace(0, T, Nt+1)    # mesh points in time
dt = t[1] - t[0]
= a*dt/dx**2
u   = zeros(Nx+1)
u_1 = zeros(Nx+1)

# Set initial condition u(x,0) = I(x)
for i in range(0, Nx+1):
    u_1[i] = I(x[i])

for n in range(0, Nt):
    # Compute u at inner mesh points
    for i in range(1, Nx):
        u[i] = u_1[i] + C*(u_1[i-1] - 2*u_1[i] + u_1[i+1])

    # Insert boundary conditions
    u[0] = 0;  u[Nx] = 0

    # Update u_1 before next step
    u_1[:]= u
```

.3 Backward Euler Scheme

/e now apply a backward difference in time in (5), but the same central difference ı space:

$$[D_t^- u = D_x D_x u]_i^n, \tag{9}$$

hich written out reads

$$\frac{u_i^n - u_i^{n-1}}{\Delta t} = \alpha\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}. \tag{10}$$

Now we assume u_i^{n-1} is computed, but all quantities at the "new" time are unknown. This time it is not possible to solve with respect to u_i^n beca value couples to its neighbors in space, u_{i-1}^n and u_{i+1}^n, which are also u Let us examine this fact for the case when $N_x = 3$. Equation (10) wri $i = 1, \ldots, Nx - 1 = 1, 2$ becomes

$$\frac{u_1^n - u_1^{n-1}}{\Delta t} = \alpha\frac{u_2^n - 2u_1^n + u_0^n}{\Delta x^2}$$
$$\frac{u_2^n - u_2^{n-1}}{\Delta t} = \alpha\frac{u_3^n - 2u_2^n + u_1^n}{\Delta x^2}$$

The boundary values u_0^n and u_3^n are known as zero. Collecting the unkn values u_1^n and u_2^n on the left-hand side gives

$$\left(1 + 2\alpha\frac{\Delta t}{\Delta x^2}\right)u_1^n - \alpha\frac{\Delta t}{\Delta x^2}u_2^n = u_1^{n-1},$$
$$-\alpha\frac{\Delta t}{\Delta x^2}u_1^n + \left(1 + 2\alpha\frac{\Delta t}{\Delta x^2}\right)u_2^n = u_2^{n-1}.$$

This is a coupled 2×2 system of algebraic equations for the unknowns u_2^n. Discretization methods that lead to a coupled system of equations unknown function at a new time level are said to be *implicit metho* counterpart, *explicit methods*, refers to discretization methods where th simple explicit formula for the values of the unknown function at eac spatial mesh points at the new time level. From an implementational view, implicit methods are more comprehensive to code since they req solution of coupled equations, i.e., a matrix system, at each time level.

In the general case, (10) gives rise to a coupled $(Nx - 1) \times (Nx - 1)$ of algebraic equations for all the unknown u_i^n at the interior spatia $i = 1, \ldots, Nx-1$. Collecting the unknowns on the left-hand side, and intr the quantity

$$C = \alpha\frac{\Delta t}{\Delta x^2},$$

(10) can be written

$$-Cu_{i-1}^n + (1 + 2C)\,u_i^n - Cu_{i+1}^n = u_{i-1}^{n-1},$$

for $i = 1, \ldots, Nx - 1$. One can either view these equations as a system fo the u_i^n values at the internal grid points, $i = 1, \ldots, N_x - 1$, are unknow may append the boundary values u_0^n and $u_{N_x}^n$ to the system. In the lat all u_i^n for $i = 0, \ldots, N_x$ are unknown and we must add the boundary eq to the $N_x - 1$ equations in (16):

$$u_0^n = 0,$$
$$u_{N_x}^n = 0.$$

A coupled system of algebraic equations can be written on matrix form, nd this is important if we want to call up ready-made software for solving the ystem. The equations (16) and (17)–(18) correspond to the matrix equation

$$AU = b$$

here $U = (u_0^n, \ldots, u_{N_x}^n)$, and the matrix A has the following structure:

$$= \begin{pmatrix} A_{0,0} & A_{0,1} & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ A_{1,0} & A_{1,1} & 0 & \ddots & & & & & \vdots \\ 0 & A_{2,1} & A_{2,2} & A_{2,3} & \ddots & & & & \vdots \\ \vdots & \ddots & & \ddots & \ddots & 0 & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & 0 & A_{i,i-1} & A_{i,i} & A_{i,i+1} & \ddots & \vdots \\ \vdots & & & & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & & & \ddots & \ddots & \ddots & A_{N_x-1,N_x} \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & A_{N_x,N_x-1} & A_{N_x,N_x} \end{pmatrix} \tag{19}$$

he nonzero elements are given by

$$A_{i,i-1} = -C \tag{20}$$
$$A_{i,i} = 1 + 2C \tag{21}$$
$$A_{i,i+1} = -C \tag{22}$$

or the equations for internal points, $i = 1, \ldots, N_x - 1$. The equations for the oundary points correspond to

$$A_{0,0} = 1, \tag{23}$$
$$A_{0,1} = 0, \tag{24}$$
$$A_{N_x,N_x-1} = 0, \tag{25}$$
$$A_{N_x,N_x} = 1\,. \tag{26}$$

he right-hand side b is written as

$$b = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_i \\ \vdots \\ b_{N_x} \end{pmatrix} \tag{27}$$

with

$$\begin{aligned} b_0 &= 0, \\ b_i &= u_i^{n-1}, \quad i = 1, \ldots, N_x - 1, \\ b_{N_x} &= 0\,. \end{aligned}$$

We observe that the matrix A contains quantities that do not ch time. Therefore, A can be formed once and for all before we enter the r formulas for the time evolution. The right-hand side b, however, must be at each time step. This leads to the following computational algorith sketched with Python code:

```
x = linspace(0, L, Nx+1)   # mesh points in space
dx = x[1] - x[0]
t = linspace(0, T, N+1)    # mesh points in time
u   = zeros(Nx+1)
u_1 = zeros(Nx+1)

# Data structures for the linear system
A = zeros((Nx+1, Nx+1))
b = zeros(Nx+1)

for i in range(1, Nx):
    A[i,i-1] = -C
    A[i,i+1] = -C
    A[i,i] = 1 + 2*C
A[0,0] = A[Nx,Nx] = 1

# Set initial condition u(x,0) = I(x)
for i in range(0, Nx+1):
    u_1[i] = I(x[i])

import scipy.linalg

for n in range(0, Nt):
    # Compute b and solve linear system
    for i in range(1, Nx):
        b[i] = -u_1[i]
    b[0] = b[Nx] = 0
    u[:] = scipy.linalg.solve(A, b)

    # Update u_1 before next step
    u_1[:] = u
```

1.4 Sparse matrix implementation

We have seen from (19) that the matrix A is tridiagonal. The code s above used a full, dense matrix representation of A, which stores a lot c we know are zero beforehand, and worse, the solution algorithm comput all these zeros. With $N_x + 1$ unknowns, the work by the solution algo $\frac{1}{3}(N_x + 1)^3$ and the storage requirements $(N_x + 1)^2$. By utilizing the fact

tridiagonal and employing corresponding software tools, the work and storage emands can be proportional to N_x only.

The key idea is to apply a data structure for a tridiagonal or sparse matrix. he `scipy.sparse` package has relevant utilities. For example, we can store the onzero diagonals of a matrix. The package also has linear system solvers that perate on sparse matrix data structures. The code below illustrates how we an store only the main diagonal and the upper and lower diagonals.

```
# Representation of sparse matrix and right-hand side
main  = zeros(Nx+1)
lower = zeros(Nx-1)
upper = zeros(Nx-1)
b     = zeros(Nx+1)

# Precompute sparse matrix
main[:] = 1 + 2*C
lower[:] = -C  #1
upper[:] = -C  #1
# Insert boundary conditions
main[0] = 1
main[Nx] = 1

A = scipy.sparse.diags(
    diagonals=[main, lower, upper],
    offsets=[0, -1, 1], shape=(Nx+1, Nx+1),
    format='csr')
print A.todense()

# Set initial condition
for i in range(0,Nx+1):
    u_1[i] = I(x[i])

for n in range(0, Nt):
    b = u_1
    b[0] = b[-1] = 0.0  # boundary conditions
    u[:] = scipy.sparse.linalg.spsolve(A, b)
    u_1[:] = u
```

he `scipy.sparse.linalg.spsolve` function utilizes the sparse storage structure of `A` and performs in this case a very efficient Gaussian elimination solve.

.5 The θ rule

he θ rule provides a family of finite difference approximations in time:

- $\theta = 0$ gives the Forward Euler scheme in time
- $\theta = 1$ gives the Backward Euler scheme in time
- $\theta = \frac{1}{2}$ gives the Crank-Nicolson scheme in time

pplied to the 1D diffusion problem we have

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \alpha \left(\theta \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2} + (1-\theta)\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} \right) .$$

This scheme also leads to a matrix system with entries $1 + 2C\theta$ on tl diagonal of the matrix, and $-C\theta$ on the super- and sub-diagonal. The rig side entry b_i is

$$b_i = u_i^n + C(1-\theta)\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} .$$

1.6 The Laplace and Poisson equation

The Laplace equation, $\nabla^2 u = 0$, or the Poisson equation, $-\nabla^2 u = f$ in numerous applications throughout science and engineering. We ca 1D variants of the Laplace equations with the listed software, because interpret $u_{xx} = 0$ as the limiting solution of $u_t = \alpha u_{xx}$ when u reach a state limit where $u_t \to 0$. Similarly, Poisson's equation $-u_{xx} = f$ aris solving $u_t = u_{xx} + f$ and letting $t \to$ so $u_t \to 0$.

Technically in a program, we can simulate $t \to \infty$ by just taking o time step, or equivalently, set α to a large value. All we need is to have As $C \to \infty$, we can from the schemes see that the limiting discrete e becomes

$$\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2} = 0,$$

which is nothing but the discretization $[D_x D_x u]_i^{n+1} = 0$ of $u_{xx} = 0$.

The Backward Euler scheme can solve the limit equation directly an produce a solution of the 1D Laplace equation. With the Forward Euler we must do the time stepping since $C > 1/2$ is illegal and leads to ins We may interpret this time stepping as solving the equation system from iterating on a time pseudo time variable.

1.7 Extensions

These extensions are performed exactly as for a wave equation as they on the spatial derivatives (which are the same as in the wave equation).

- Variable coefficients
- Neumann and Robin conditions
- 2D and 3D

Future versions of this document will for completeness and independenc wave equation document feature info on the three points. The Robin c is new, but straightforward to handle:

$$-\alpha \frac{\partial u}{\partial n} = h_T(u - U_s), \quad [-\alpha D_x u = h_T(u - U_s)]_i^n$$

Analysis of schemes for the diffusion equation

.1 Properties of the solution

particular characteristic of diffusive processes, governed by an equation like

$$u_t = \alpha u_{xx}, \tag{31}$$

that the initial shape $u(x,0) = I(x)$ spreads out in space with time, along ith a decaying amplitude. Three different examples will illustrate the spreading f u in space and the decay in time.

imilarity solution. The diffusion equation (31) admits solutions that depend n $\eta = (x-c)/\sqrt{4\alpha t}$ for a given value of c. One particular solution is

$$u(x,t) = a\,\mathrm{erf}(\eta) + b, \tag{32}$$

here

$$\mathrm{erf}(\eta) = \frac{2}{\sqrt{\pi}}\int_0^\eta e^{-\zeta^2}d\zeta, \tag{33}$$

the *error function*, and a and b are arbitrary constants. The error function es in $(-1,1)$, is odd around $\eta = 0$, and goes relatively quickly to ± 1:

$$\begin{aligned}
\lim_{\eta\to-\infty}\mathrm{erf}(\eta) &= -1,\\
\lim_{\eta\to\infty}\mathrm{erf}(\eta) &= 1,\\
\mathrm{erf}(\eta) &= -\mathrm{erf}(-\eta),\\
\mathrm{erf}(0) &= 0,\\
\mathrm{erf}(2) &= 0.99532227,\\
\mathrm{erf}(3) &= 0.99997791\,.
\end{aligned}$$

As $t \to 0$, the error function approaches a step function centered at $x = c$. or a diffusion problem posed on the unit interval $[0,1]$, we may choose the step t $x = 1/2$ (meaning $c = 1/2$), $a = -1/2$, $b = 1/2$. Then

$$u(x,t) = \frac{1}{2}\left(1 - \mathrm{erf}\left(\frac{x-\frac{1}{2}}{\sqrt{4\alpha t}}\right)\right) = \frac{1}{2}\mathrm{erfc}\left(\frac{x-\frac{1}{2}}{\sqrt{4\alpha t}}\right), \tag{34}$$

here we have introduced the *complementary error function* $\mathrm{erfc}(\eta) = 1 - \mathrm{erf}(\eta)$. he solution (34) implies the boundary conditions

$$u(0,t) = \frac{1}{2}\left(1 - \mathrm{erf}\left(\frac{-1/2}{\sqrt{4\alpha t}}\right)\right), \tag{35}$$

$$u(1,t) = \frac{1}{2}\left(1 - \mathrm{erf}\left(\frac{1/2}{\sqrt{4\alpha t}}\right)\right). \tag{36}$$

For small enough t, $u(0,t) \approx 1$ and $u(1,t) \approx 1$, but as $t \to \infty$, $u(x,t) \to$ $[0,1]$.

Solution for a Gaussian pulse. The standard diffusion equation u_t admits a Gaussian function as solution:

$$u(x,t) = \frac{1}{\sqrt{4\pi\alpha t}}\exp\left(-\frac{(x-c)^2}{4\alpha t}\right).$$

At $t = 0$ this is a Dirac delta function, so for computational purposes o start to view the solution at some time $t = t_\epsilon > 0$. Replacing t by t (37) makes it easy to operate with a (new) t that starts at $t = 0$ with a condition with a finite width. The important feature of (37) is that the s deviation σ of a sharp initial Gaussian pulse increases in time accor $\sigma = \sqrt{2\alpha t}$, making the pulse diffuse and flatten out.

Solution for a sine component. For example, (31) admits a solutio form

$$u(x,t) = Qe^{-at}\sin(kx)\,.$$

The parameters Q and k can be freely chosen, while inserting (38) in (3 the constraint

$$a = -\alpha k^2\,.$$

A very important feature is that the initial shape $I(x) = Q\sin kx$ ur a damping $\exp(-\alpha k^2 t)$, meaning that rapid oscillations in space, corres to large k, are very much faster dampened than slow oscillations i corresponding to small k. This feature leads to a smoothing of th condition with time.

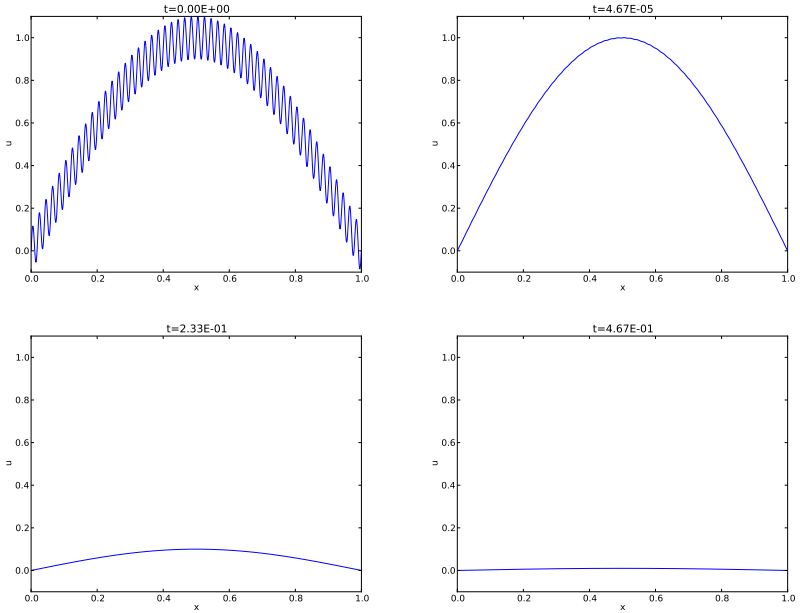
The following examples illustrates the damping properties of (3 consider the specific problem

$$\begin{aligned}
u_t = u_{xx}, \quad & x \in (0,1),\ t \in (0,T],\\
u(0,t) = u(1,t) = 0, \quad & t \in (0,T],\\
u(x,0) = \sin(\pi x) + 0.1\sin(100\pi x)\,.&
\end{aligned}$$

The initial condition has been chosen such that adding two solutions l constructs an analytical solution to the problem:

$$u(x,t) = e^{-\pi^2 t}\sin(\pi x) + 0.1e^{-\pi^2 10^4 t}\sin(100\pi x)\,.$$

Figure 1 illustrates the rapid damping of rapid oscillations $\sin(100\pi x)$ very much slower damping of the slowly varying $\sin(\pi x)$ term. Afte $t = 0.5\cdot 10^{-4}$ the rapid oscillations do not have a visible amplitude, w have to wait until $t \sim 0.5$ before the amplitude of the long wave $\sin(\pi x)$ l very small.



igure 1: Evolution of the solution of a diffusion problem: initial condition
ıpper left), 1/100 reduction of the small waves (upper right), 1/10 reduction of
ıe long wave (lower left), and 1/100 reduction of the long wave (lower right).

.2 Analysis of discrete equations

counterpart to (38) is the complex representation of the same function:

$$u(x,t) = Qe^{-at}e^{ikx},$$

here $i = \sqrt{-1}$ is the imaginary unit. We can add such functions, often referred
› as wave components, to make a Fourier representation of a general solution of
ıe diffusion equation:

$$u(x,t) \approx \sum_{k\in K} b_k e^{-\alpha k^2 t}e^{ikx}, \tag{40}$$

here K is a set of an infinite number of k values needed to construct the
›lution. In practice, however, the series is truncated and K is a finite set of k
ılues need build a good approximate solution. Note that (39) is a special case
f (40) where $K = \{\pi, 100\pi\}$, $b_\pi = 1$, and $b_{100\pi} = 0.1$.

The amplitudes b_k of the individual Fourier waves must be determined from
ıe initial condition. At $t = 0$ we have $u \approx \sum_k b_k \exp(ikx)$ and find K and b_k
ıch that

$$I(x) \approx \sum_{k\in K} b_k e^{ikx}.$$

(The relevant formulas for b_k come from Fourier analysis, or equival
least-squares method for approximating $I(x)$ in a function space wit
$\exp(ikx)$.)

Much insight about the behavior of numerical methods can be obta
investigating how a wave component $\exp(-\alpha k^2 t)\exp(ikx)$ is treated
numerical scheme. It appears that such wave components are also solu
the schemes, but the damping factor $\exp(-\alpha k^2 t)$ varies among the sche
ease the forthcoming algebra, we write the damping factor as A^n. Th
amplification factor corresponding to A is $A_\mathrm{e} = \exp(-\alpha k^2 \Delta t)$.

2.3 Analysis of the finite difference schemes

We have seen that a general solution of the diffusion equation can be bı
linear combination of basic components

$$e^{-\alpha k^2 t}e^{ikx}.$$

A fundamental question is whether such components are also solution
finite difference schemes. This is indeed the case, but the amplitude exp
might be modified (which also happens when solving the ODE cour
$u' = -\alpha u$). We therefore look for numerical solutions of the form

$$u_q^n = A^n e^{ikq\Delta x} = A^n e^{ikx},$$

where the amplification factor A must be determined by inserting the cor
into an actual scheme.

Stability. The exact amplification factor is $A_\mathrm{e} = \exp(-\alpha^2 k^2 \Delta t)$. We
therefore require $|A| < 1$ to have a decaying numerical solution as
$-1 \leq A < 0$, A^n will change sign from time level to time level, and we ge
non-physical oscillations in the numerical solutions that are not presen
exact solution.

Accuracy. To determine how accurately a finite difference scheme tre
wave component (42), we see that the basic deviation from the exact sol
reflected in how well A^n approximates A_e^n, or how well A approximates

2.4 Analysis of the Forward Euler scheme

The Forward Euler finite difference scheme for $u_t = \alpha u_{xx}$ can be writte

$$[D_t^+ u = \alpha D_x D_x u]_q^n.$$

Inserting a wave component (42) in the scheme demands calculating th

$$e^{ikq\Delta x}[D_t^+ A]^n = e^{ikq\Delta x} A^n \frac{A-1}{\Delta t},$$

nd

$$A^n D_x D_x [e^{ikx}]_q = A^n \left(-e^{ikq\Delta x} \frac{4}{\Delta x^2} \sin^2\left(\frac{k\Delta x}{2}\right)\right) .$$

nserting these terms in the discrete equation and dividing by $A^n e^{ikq\Delta x}$ leads to

$$\frac{A-1}{\Delta t} = -\alpha \frac{4}{\Delta x^2} \sin^2\left(\frac{k\Delta x}{2}\right),$$

nd consequently

$$A = 1 - 4C \sin^2\left(\frac{k\Delta x}{2}\right), \tag{43}$$

here

$$C = \frac{\alpha \Delta t}{\Delta x^2}. \tag{44}$$

he complete numerical solution is then

$$u_q^n = \left(1 - 4C \sin^2\left(\frac{k\Delta x}{2}\right)\right)^n e^{ikq\Delta x}. \tag{45}$$

tability. We easily see that $A \leq 1$. However, the A can be less than -1, hich will lead to growth of a numerical wave component. The criterion $A \geq -1$ nplies

$$4C \sin^2(p/2) \leq 2.$$

he worst case is when $\sin^2(p/2) = 1$, so a sufficient criterion for stability is

$$C \leq \frac{1}{2}, \tag{46}$$

r expressed as a condition on Δt:

$$\Delta t \leq \frac{\Delta x^2}{2\alpha}. \tag{47}$$

ote that halving the spatial mesh size, $\Delta x \to \frac{1}{2}\Delta x$, requires Δt to be reduced y a factor of $1/4$. The method hence becomes very expensive for fine spatial eshes.

Accuracy. Since A is expressed in terms of C and the parameter we $p = k\Delta x/2$, we also express A_e by C and p:

$$A_\mathrm{e} = \exp\left(-\alpha k^2 \Delta t\right) = \exp\left(-4Cp^2\right).$$

Computing the Taylor series expansion of A/A_e in terms of C can easily with aid of `sympy`:

```
def A_exact(C, p):
    return exp(-4*C*p**2)

def A_FE(C, p):
    return 1 - 4*C*sin(p)**2

from sympy import *
C, p = symbols('C p')
A_err_FE = A_FE(C, p)/A_exact(C, p)
print A_err_FE.series(C, 0, 6)
```

The result is

$$\frac{A}{A_\mathrm{e}} = 1 - 4C\sin^2 p + 2Cp^2 - 16C^2p^2\sin^2 p + 8C^2p^4 + \cdots$$

Recalling that $C = \alpha\Delta t/\Delta x$, $p = k\Delta x/2$, and that $\sin^2 p \leq 1$, we real the dominating error terms are at most

$$1 - 4\alpha\frac{\Delta t}{\Delta x^2} + \alpha\Delta t - 4\alpha^2\Delta t^2 + \alpha^2\Delta t^2\Delta x^2 + \cdots .$$

2.5 Analysis of the Backward Euler scheme

Discretizing $u_t = \alpha u_{xx}$ by a Backward Euler scheme,

$$[D_t^- u = \alpha D_x D_x u]_q^n,$$

and inserting a wave component (42), leads to calculations similar t arising from the Forward Euler scheme, but since

$$e^{ikq\Delta x}[D_t^- A]^n = A^n e^{ikq\Delta x}\frac{1 - A^{-1}}{\Delta t},$$

we get

$$\frac{1 - A^{-1}}{\Delta t} = -\alpha\frac{4}{\Delta x^2}\sin^2\left(\frac{k\Delta x}{2}\right),$$

and then

$$A = \left(1 + 4C\sin^2 p\right)^{-1}.$$

The complete numerical solution can be written

$$u_q^n = \left(1 + 4C\sin^2 p\right)^{-n} e^{ikq\Delta x}.$$

tability. We see from (48) that $0 < A < 1$, which means that all numerical ave components are stable and non-oscillatory for any $\Delta t > 0$.

.6 Analysis of the Crank-Nicolson scheme

he Crank-Nicolson scheme can be written as

$$[D_t u = \alpha D_x D_x \overline{u}^x]_q^{n+\frac{1}{2}},$$

$$[D_t u]_q^{n+\frac{1}{2}} = \frac{1}{2}\alpha\left([D_x D_x u]_q^n + [D_x D_x u]_q^{n+1}\right).$$

nserting (42) in the time derivative approximation leads to

$$[D_t A^n e^{ikq\Delta x}]^{n+\frac{1}{2}} = A^{n+\frac{1}{2}} e^{ikq\Delta x}\frac{A^{\frac{1}{2}} - A^{-\frac{1}{2}}}{\Delta t} = A^n e^{ikq\Delta x}\frac{A-1}{\Delta t}.$$

nserting (42) in the other terms and dividing by $A^n e^{ikq\Delta x}$ gives the relation

$$\frac{A-1}{\Delta t} = -\frac{1}{2}\alpha\frac{4}{\Delta x^2}\sin^2\left(\frac{k\Delta x}{2}\right)(1+A),$$

nd after some more algebra,

$$A = \frac{1 - 2C\sin^2 p}{1 + 2C\sin^2 p}. \tag{50}$$

he exact numerical solution is hence

$$u_q^n = \left(\frac{1 - 2C\sin^2 p}{1 + 2C\sin^2 p}\right)^n e^{ikp\Delta x}. \tag{51}$$

tability. The criteria $A > -1$ and $A < 1$ are fulfilled for any $\Delta t > 0$.

.7 Summary of accuracy of amplification factors

Ve can plot the various amplification factors against $p = k\Delta x/2$ for different noices of the C parameter. Figures 2, 3, and 4 show how long and small aves are damped by the various schemes compared to the exact damping. As ong as all schemes are stable, the amplification factor is positive, except for rank-Nicolson when $C > 0.5$.

The effect of negative amplification factors is that A^n changes sign from one me level to the next, thereby giving rise to oscillations in time in an animation f the solution. We see from Figure 2 that for $C = 20$, waves with $p \geq \pi/2$ ndergo a damping close to -1, which means that the amplitude does not decay nd that the wave component jumps up and down in time. For $C = 2$ we have a amping of a factor of 0.5 from one time level to the next, which is very much naller than the exact damping. Short waves will therefore fail to be effectively

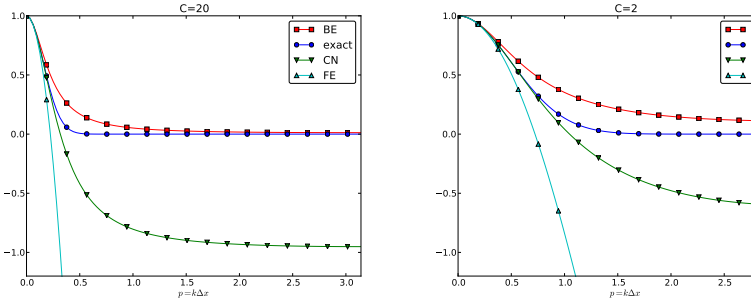


Figure 2: Amplification factors for large time steps.

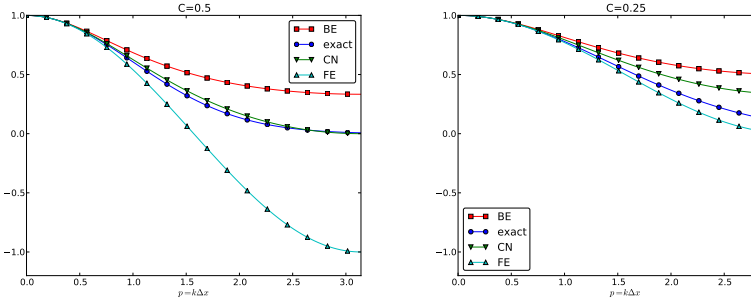


Figure 3: Amplification factors for time steps around the Forward Euler limit.

dampened. These waves will manifest themselves as high frequency osc noise in the solution.

A value $p = \pi/4$ corresponds to four mesh points per wave length while $p = \pi/2$ implies only two points per wave length, which is the s number of points we can have to represent the wave on the mesh.

To demonstrate the oscillatory behavior of the Crank-Nicolson sch choose an initial condition that leads to short waves with significant am A discontinuous $I(x)$ will in particular serve this purpose.

Run $C =$ .....

Exercise 1: Use an analytical solution to formulate test

This exercise explores the exact solution (37). We shall formulate a d problem in half of the domain for half of the Gaussian pulse. Then investigate the impact of using an incorrect boundary condition, whic general cases often are forced due if the solution needs to pass throug boundaries undisturbed.

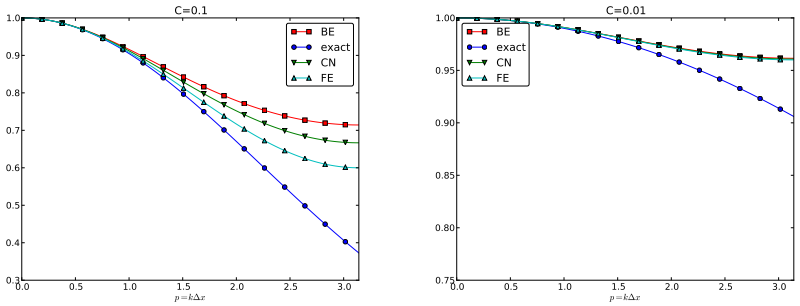


Figure 4: Amplification factors for small time steps.

) The solution (37) is seen to be symmetric at $x = c$, because $\partial u/\partial x = 0$ lways vanishes for $x = c$. Use this property to formulate a complete initial oundary value problem in 1D involving the diffusion equation $u_t = \alpha u_{xx}$ on $, L]$ with $u_x(0,t) = 0$ and $u(L,t)$ known.

) Use the exact solution to set up a convergence rate test for an implementation f the problem. Investigate if a one-sided difference for $u_x(0,t)$, say $u_0 = u_1$, estroys the second-order accuracy in space.

) Imagine that we want to solve the problem numerically on $[0, L]$, but we do ot know the exact solution and cannot of that reason assign a correct Dirichlet ondition at $x = L$. One idea is to simply set $u(L,t) = 0$ since this will be n accurate approximation before the diffused pulse reaches $x = L$ and even ereafter it might be a satisfactory condition. Let u_e be the exact solution nd let u be the solution of $u_t = \alpha u_{xx}$ with an initial Gaussian pulse and the oundary conditions $u_x(0,t) = u(L,t) = 0$. Derive a diffusion problem for ne error $e = u_e - u$. Solve this problem numerically using an exact Dirichlet ondition at $x = L$. Animate the evolution of the error and make a curve plot of ne error measure

$$E(t) = \sqrt{\frac{\int_0^L e^2 dx}{\int_0^L u dx}}.$$

; this a suitable error measure for the present problem?

) Instead of using $u(L,t) = 0$ as approximate boundary condition for letting ne diffused Gaussian pulse out of our finite domain, one may try $u_x(L,t) = 0$ nce the solution for large t is quite flat. Argue that this condition gives a ompletely wrong asymptotic solution as $t \to 0$. To do this, integrate the iffusion equation from 0 to L, integrate u_{xx} by parts (or use Gauss' divergence eorem in 1D) to arrive at the important property

$$\frac{d}{dt}\int_0^L u(x,t)dx = 0,$$

implying that $\int_0^L u dx$ must be constant in time, and therefore

$$\int_0^L u(x,t)dx = \int_0^L I(x)dx\,.$$

The integral of the initial pulse is 1.

e) Another idea for an artificial boundary condition at $x = L$ is to use a law

$$-\alpha u_x = q(u - u_S),$$

where q is an unknown heat transfer coefficient and u_S is the surr temperature in the medium outside of $[0, L]$. (Note that arguing th approximately $u(L,t)$ gives the $u_x = 0$ condition from the previous sub that is qualitatively wrong for large t.) Develop a diffusion problem for t in the solution using (52) as boundary condition. Assume one can take "outside the domain" as $u \to 0$ for $x \to \infty$. Find a function $q = q(t)$ su the exact solution obeys the condition (52). Test some constant val and animate how the corresponding error function behaves. Also comp curves as suggested in subexercise b).

Filename: `diffu_symmetric_gaussian.py`.

Exercise 2: Use an analytical solution to formulate test

Generalize (37) to multi dimensions by assuming that one-dimensional s can be multiplied to solve $u_t = \alpha\nabla^2 u$. Use this solution to formulate a case where the peak of the Gaussian is at the origin and where the dom rectangle in the first quadrant. Use symmetry boundary conditions ∂ 0 whereever possible, and use exact Dirichlet conditions on the re boundaries.

The solution is seen to be symmetric Filename: `diffu_symmetric_ga`

ndex