

# Finite difference methods for diffusion processes

Hans Petter Langtangen<sup>1,2</sup>

<sup>1</sup>Center for Biomedical Computing, Simula Research Laboratory

<sup>2</sup>Department of Informatics, University of Oslo

Jul 14, 2014

VERY PRELIMINARY VERSION

## Contents

### 1D diffusion equation

The initial-boundary value problem for 1D diffusion . . . . .	
Forward Euler scheme . . . . .	
Backward Euler scheme . . . . .	
Sparse matrix implementation . . . . .	
The $\theta$ rule . . . . .	
The Laplace and Poisson equation . . . . .	
Extensions . . . . .	

### Analysis of schemes for the diffusion equation

Properties of the solution . . . . .	
Analysis of discrete equations . . . . .	
Analysis of the finite difference schemes . . . . .	
Analysis of the Forward Euler scheme . . . . .	
Analysis of the Backward Euler scheme . . . . .	
Analysis of the Crank-Nicolson scheme . . . . .	
Summary of accuracy of amplification factors . . . . .	

## Exercises

## List of Exercises and Projects

Exercise 1	Use an analytical solution to formulate a ...	
Exercise 2	Use an analytical solution to formulate a ...	
Exercise 3	Examine stability of a diffusion model with ...	p. 18
Exercise 4	Stabilizing the Crank-Nicolson method by Rannacher ...	p. 19
Project 5	Energy estimates for diffusion problems	p. 19

## the 1D diffusion equation

diffusion equation, 1D indexheat equation, 1D  
amous *diffusion equation*, also known as the *heat equation*, reads

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

where  $u(x, t)$  is the unknown function to be solved for,  $x$  is a coordinate in space, and  $t$  is time. The coefficient  $\alpha$  is the *diffusion coefficient* and determines how fast  $u$  changes in time.

For the diffusion equation is  $u_t = \alpha u_{xx}$ .

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

For all diffusion problems may experience rapid change in the very beginning, but then the change of  $u$  becomes slower and slower. The solution is usually very smooth, and after a long time cannot recognize the initial shape of  $u$ . This is in sharp contrast to solution of the wave equation where the initial shape is preserved - the solution is basically a moving pulse. The standard wave equation  $u_{tt} = c^2 u_{xx}$  has solutions that propagate without changing shape, while the diffusion equation converges to a *stationary* solution  $u \rightarrow \infty$ . In this limit,  $u_t = 0$ , and  $u$  is governed by  $u''(x) = 0$ . This stationary limit equation is called the *Laplace* equation and arises in a very wide range of applications in the sciences.

It is possible to solve for  $u(x, t)$  using an explicit scheme, but the time step restriction is much less favorable than for an explicit scheme for the wave equation. And since the solution  $u$  of the diffusion equation is very smooth and changes slowly, it is more convenient and not required by accuracy as the diffusion process converges to a steady state.

## the initial-boundary value problem for 1D diffusion

To find a unique solution of the diffusion equation, or equivalently, to apply numerical methods, we need initial and boundary conditions. The diffusion equation goes with one initial condition  $u(x, 0) = I(x)$ , where  $I$  is a prescribed function. One boundary condition is required at each boundary, which in 1D means that  $u$  must be known,  $u_x$  must be known, or a combination of them.

We will start with the simplest boundary condition:  $u = 0$ . The complete initial-boundary value problem in one space dimension can then be specified as

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

(1) is known as a one-dimensional *diffusion equation*, also often referred to as the *heat equation*. With only a first-order derivative in time, only one *initial condition* is needed, while a second-order derivative in space leads to a demand for two *boundary conditions*. The parameter  $\alpha$  is given and is referred to as the *diffusion coefficient*.

Diffusion equations like (1) have a wide range of applications throughout physics and financial sciences. One of the most common applications is propagation of heat.  $u$  represents the temperature of some substance at point  $x$  and time  $t$ . Section ?? discusses several other interesting applications.

## 2 Forward Euler scheme

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

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

and

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

Moreover,  $u_i^n$  denotes the mesh function that approximates  $u(x_i, t_n)$  for  $i = 0, \dots, N_x$ . Requiring the PDE (1) to be fulfilled at a mesh point  $(x_i, t_n)$  leads to

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

The next step is to replace the derivatives by finite difference approximations. The simplest method arises from using a forward difference in 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 discrete equations. An important property of the equations is that they are algebraic, which makes them easy to solve. We anticipate that  $u_i^n$  is already computed such that  $u_i^{n+1}$  is the only unknown in the equation with respect to this unknown is easy:

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

The computational algorithm then becomes

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

The algorithm is compactly fully specified in Python:

```

pace(0, L, Nx+1)    # mesh points in space
] - x[0]
pace(0, T, Nt+1)    # mesh points in time
] - t[0]
t/dx**2
ros(Nx+1)
ros(Nx+1)

itial condition u(x,0) = I(x)
range(0, Nx+1):
i] = I(x[i])

range(0, Nt):
mpute u at inner mesh points
i in range(1, Nx):
u[i] = u_1[i] + Fo*(u_1[i-1] - 2*u_1[i] + u_1[i+1])

sert boundary conditions
= 0; u[Nx] = 0

date u_1 before next step
:] = u

```

## Backward Euler scheme

Apply a backward difference in time in (5), but the same central difference in space.

$$[D_t^- u = D_x D_x u]_i^n,$$

Equation (10) then out reads

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

Assume  $u_i^{n-1}$  is computed, but all quantities at the "new" time level  $n$  are unknown. It is not possible to solve with respect to  $u_i^n$  because this value couples to its neighbors  $u_{i-1}^n$  and  $u_{i+1}^n$ , which are also unknown. Let us examine this fact for the case of Equation (10) written for  $i = 1, \dots, Nx - 1 = 1, 2$  becomes

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

Boundary values  $u_0^n$  and  $u_{Nx}^n$  are known as zero. Collecting the unknown new values on the left-hand side gives

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

A coupled  $2 \times 2$  system of algebraic equations for the unknowns  $u_1^n$  and  $u_2^n$ . Discretizing the rest of the domain leads to a coupled system of equations for the unknown function at a new

level are said to be *implicit methods*. The counterpart, *explicit methods*, refers to methods where there is a simple explicit formula for the values of the unknown function at the spatial mesh points at the new time level. From an implementational point of view, implicit methods are more comprehensive to code since they require the solution of a coupled system, a matrix system, at each time level.

In the general case, (10) gives rise to a coupled  $(Nx - 1) \times (Nx - 1)$  system of equations for all the unknown  $u_i^n$  at the interior spatial points  $i = 1, \dots, Nx - 1$ . The right-hand side contains knowns on the left-hand side, and introducing the *numerical Fourier number*

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

Equation (10) can be written

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

for  $i = 1, \dots, Nx - 1$ . One can either view these equations as a system for where the interior grid points,  $i = 1, \dots, Nx - 1$ , are unknown, or we may append the boundary values  $u_0^n$  and  $u_{Nx}^n$  to the system. In the latter case, all  $u_i^n$  for  $i = 0, \dots, Nx$  are unknown. We add the boundary equations to the  $Nx - 1$  equations in (16):

$$\begin{aligned} u_0^n &= 0, \\ u_{Nx}^n &= 0. \end{aligned}$$

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

$$AU = b$$

where  $U = (u_0^n, \dots, u_{Nx}^n)$ , and the matrix  $A$  has the following structure:

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

The nonzero elements are given by

$$\begin{aligned} A_{i,i-1} &= -F_o \\ A_{i,i} &= 1 + 2F_o \\ A_{i,i+1} &= -F_o \end{aligned}$$

equations for internal points,  $i = 1, \dots, N_x - 1$ . The equations for the boundary are added to

$$\begin{aligned} A_{0,0} &= 1, \\ A_{0,1} &= 0, \\ A_{N_x, N_x-1} &= 0, \\ A_{N_x, N_x} &= 1. \end{aligned}$$

The right-hand side  $b$  is written as

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

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

Observe that the matrix  $A$  contains quantities that do not change in time. They are computed once and for all before we enter the recursive formulas for the time evolution. The right-hand side  $b$ , however, must be updated at each time step. This leads to the following algorithm, here sketched with Python code:

```

Nx, Nt = 100, 100 # mesh points in space and time
x = x[0]
t = t[0]
ros(Nx+1)
ros(Nx+1)

# Structures for the linear system
s((Nx+1, Nx+1))
s(Nx+1)

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

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

# 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

```

#### 4 Sparse matrix implementation

We have seen from (19) that the matrix  $A$  is tridiagonal. The code segment above uses a dense matrix representation of  $A$ , which stores a lot of values we know are zero. Otherwise, the solution algorithm computes with all these zeros. With  $N_x + 1$  unknowns, the solution algorithm is  $\frac{1}{3}(N_x + 1)^3$  and the storage requirements  $(N_x + 1)^2$ . In fact, because  $A$  is tridiagonal and employing corresponding software tools, the work requirements can be proportional to  $N_x$  only.

The key idea is to apply a data structure for a tridiagonal or sparse matrix. The SciPy package has relevant utilities. For example, we can store the nonzero diagonals of  $A$ . The SciPy package also has linear system solvers that operate on sparse matrix data structures. The following code illustrates how we can store only the main diagonal and the upper and lower

```

# 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*Fo
lower[:] = -Fo #1
upper[:] = -Fo #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

```

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

#### 5 The $\theta$ rule

The  $\theta$  rule provides a family of finite difference approximations in time:

0 gives the Forward Euler scheme in time

1 gives the Backward Euler scheme in time

$\frac{1}{2}$  gives the Crank-Nicolson scheme in time

o the 1D diffusion problem we have

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

me also leads to a matrix system with entries  $1 + 2F_o\theta$  on the main diagonal and  $-F_o\theta$  on the super- and sub-diagonal. The right-hand side entry  $b_i$  is

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

## he Laplace and Poisson equation

ace equation,  $\nabla^2 u = 0$ , or the Poisson equation,  $-\nabla^2 u = f$ , occur in nu  
ms throughout science and engineering. We can solve 1D variants of the  
with the listed software, because we can interpret  $u_{xx} = 0$  as the limiting  
 $u_{xx}$  when  $u$  reach a steady state limit where  $u_t \rightarrow 0$ . Similarly, Poisson's e  
arises from solving  $u_t = u_{xx} + f$  and letting  $t \rightarrow \infty$  so  $u_t \rightarrow 0$ .

ically in a program, we can simulate  $t \rightarrow \infty$  by just taking one large time  
tly, set  $\alpha$  to a large value. All we need is to have  $F_o$  large. As  $F_o \rightarrow \infty$ , we c  
ies see that the limiting discrete equation becomes

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

othing but the discretization  $[D_x D_x u]_i^{n+1} = 0$  of  $u_{xx} = 0$ .

ackward Euler scheme can solve the limit equation directly and hence produce a  
Laplace equation. With the Forward Euler scheme we must do the time steppi  
is illegal and leads to instability. We may interpret this time stepping as solv  
system from  $u_{xx}$  by iterating on a time pseudo time variable.

## xtensions

ensions are performed exactly as for a wave equation as they only affect the  
s (which are the same as in the wave equation).

table coefficients

mann and Robin conditions

and 3D

rsions of this document will for completeness and independence of the wave e  
feature info on the three points. The Robin condition is new, but straightfor

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

# Analysis of schemes for the diffusion equation

## .1 Properties of the solution

particular characteristic of diffusive processes, governed by an equation like

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

that the initial shape  $u(x, 0) = I(x)$  spreads out in space with time, along  
mplitude. Three different examples will illustrate the spreading of  $u$  in space  
me.

**imilarity solution.** The diffusion equation (31) admits solutions that dep  
 $)/\sqrt{4\alpha t}$  for a given value of  $c$ . One particular solution is

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

here

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

the *error function*, and  $a$  and  $b$  are arbitrary constants. The error function l  
dd around  $\eta = 0$ , and goes relatively quickly to  $\pm 1$ :

$$\lim_{\eta \rightarrow -\infty} \operatorname{erf}(\eta) = -1,$$

$$\lim_{\eta \rightarrow \infty} \operatorname{erf}(\eta) = 1,$$

$$\operatorname{erf}(\eta) = -\operatorname{erf}(-\eta),$$

$$\operatorname{erf}(0) = 0,$$

$$\operatorname{erf}(2) = 0.99532227,$$

$$\operatorname{erf}(3) = 0.99997791.$$

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

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

here we have introduced the *complementary error function*  $\operatorname{erfc}(\eta) = 1 - \operatorname{erf}(\eta)$   
4) implies the boundary conditions

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

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

or small enough  $t$ ,  $u(0, t) \approx 1$  and  $u(1, t) \approx 1$ , but as  $t \rightarrow \infty$ ,  $u(x, t) \rightarrow 1/2$  on

**for a Gaussian pulse.** The standard diffusion equation  $u_t = \alpha u_{xx}$  admits a Gaussian solution:

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

This is a Dirac delta function, so for computational purposes one must start at some time  $t = t_\epsilon > 0$ . Replacing  $t$  by  $t_\epsilon + t$  in (37) makes it easy to operate at starts at  $t = 0$  with an initial condition with a finite width. The important feature is that the standard deviation  $\sigma$  of a sharp initial Gaussian pulse increases in time as  $\sqrt{2\alpha t}$ , making the pulse diffuse and flatten out.

**for a sine component.** For example, (31) admits a solution of the form

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

Parameters  $Q$  and  $k$  can be freely chosen, while inserting (38) in (31) gives the constraint

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

One important feature is that the initial shape  $I(x) = Q \sin kx$  undergoes a damping (exponential decay) with time, meaning that rapid oscillations in space, corresponding to large  $k$ , are damped more rapidly than slow oscillations in space, corresponding to small  $k$ . This feature is a direct consequence of the diffusion equation.

The following examples illustrate the damping properties of (38). We consider the

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

The initial condition has been chosen such that adding two solutions like (38) constructs a solution to the problem:

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

This illustrates the rapid damping of rapid oscillations  $\sin(100\pi x)$  and the very much slower damping of the slowly varying  $\sin(\pi x)$  term. After about  $t = 0.5 \cdot 10^{-4}$  the rapid oscillations are no longer visible, while we have to wait until  $t \sim 0.5$  before the amplitude of the  $\sin(\pi x)$  term becomes very small.

## Analysis of discrete equations

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

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

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

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

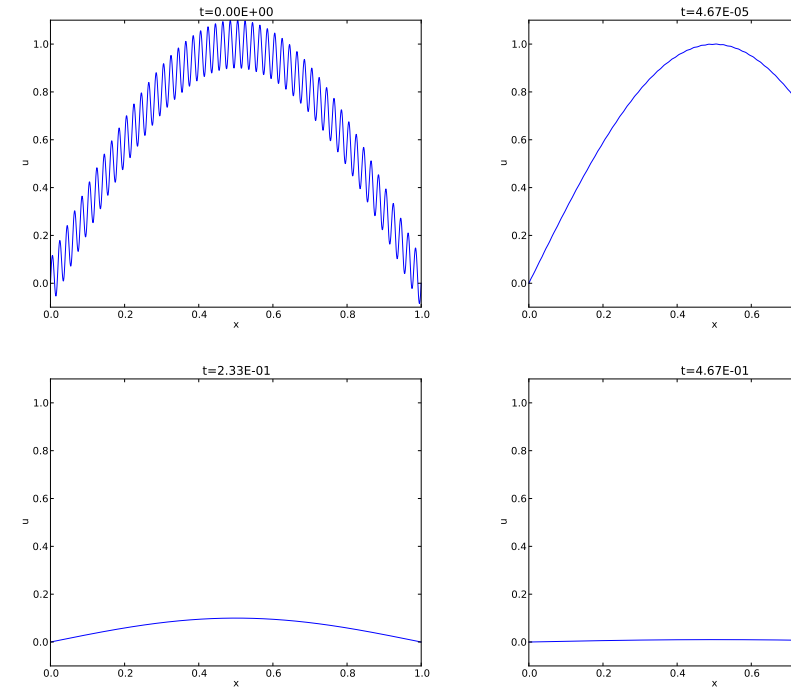


Figure 1: Evolution of the solution of a diffusion problem: initial condition (upper left), 1/10 reduction of the small waves (upper right), 1/10 reduction of the long wave (lower left), 1/100 reduction of the long wave (lower right).

where  $K$  is a set of an infinite number of  $k$  values needed to construct the solution. However, the series is truncated and  $K$  is a finite set of  $k$  values needed to build a good approximation. Note that (39) is a special case of (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 the initial condition. At  $t = 0$  we have  $u \approx \sum_k b_k \exp(ikx)$  and find  $K$  and  $b_k$  such that

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

The relevant formulas for  $b_k$  come from Fourier analysis, or equivalently, a least squares fit or approximating  $I(x)$  in a function space with basis  $\exp(ikx)$ .

Much insight about the behavior of numerical methods can be obtained by analyzing how a single wave component  $\exp(-\alpha k^2 t) \exp(ikx)$  is treated by the numerical scheme. Each wave component is also a solution of the schemes, but the damping factor varies among the schemes. To ease the forthcoming algebra, we write the damping factor as  $A = \exp(-\alpha k^2 \Delta t)$ .

## analysis of the finite difference schemes

een that a general solution of the diffusion equation can be built as a linear combination of components

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

entral question is whether such components are also solutions of the finite difference scheme. This is indeed the case, but the amplitude  $\exp(-\alpha k^2 t)$  might be modified (when solving the ODE counterpart  $u' = -\alpha u$ ). 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 component into the finite difference equation.

The exact amplification factor is  $A_e = \exp(-\alpha k^2 \Delta t)$ . We should therefore have a decaying numerical solution as well. If  $-1 \leq A < 0$ ,  $A^n$  will change sign at time level  $n$ , and we get stable, non-physical oscillations in the numerical solution not present in the exact solution.

To determine how accurately a finite difference scheme treats one wave component, we note that the basic deviation from the exact solution is reflected in how well  $A^n$  approximates  $A_e$ .

## analysis of the Forward Euler scheme

The forward Euler finite difference scheme for  $u_t = \alpha u_{xx}$  can be written as

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

For a wave component (42) in the scheme demands calculating the terms

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

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

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

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

Consequently

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

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

where  $F_o$  is the numerical Fourier number. The complete numerical solution is then

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

**Stability.** We easily see that  $A \leq 1$ . However, the  $A$  can be less than  $-1$ , with growth of a numerical wave component. The criterion  $A \geq -1$  implies

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

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

$$F_o \leq \frac{1}{2},$$

or expressed as a condition on  $\Delta t$ :

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

Note that halving the spatial mesh size,  $\Delta x \rightarrow \frac{1}{2}\Delta x$ , requires  $\Delta t$  to be reduced by a factor of 4. The method hence becomes very expensive for fine spatial meshes.

**Accuracy.** Since  $A$  is expressed in terms of  $F_o$  and the parameter we now call  $p$ , we can also express  $A_e$  by  $F_o$  and  $p$ :

$$A_e = \exp(-\alpha k^2 \Delta t) = \exp(-4F_o p^2).$$

Computing the Taylor series expansion of  $A/A_e$  in terms of  $F_o$  can easily be done with sympy:

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

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

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

The result is

$$\frac{A}{A_e} = 1 - 4F_o \sin^2 p + 2F_o^2 p^2 - 16F_o^3 p^2 \sin^2 p + 8F_o^4 p^4 + \dots$$

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

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

## analysis of the Backward Euler scheme

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

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

ting a wave component (42), leads to calculations similar to those arising fi Euler scheme, but since

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

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

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

lete numerical solution can be written

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

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

## analysis of the Crank-Nicolson scheme

k-Nicolson scheme can be written as

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

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

(42) in the time derivative approximation leads to

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

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

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

some more algebra,

$$A = \frac{1 - 2F_o \sin^2 p}{1 + 2F_o \sin^2 p}.$$

numerical solution is hence

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

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 c arameter. Figures 2, 3, and 4 show how long and small waves are damped by the ompared to the exact damping. As long as all schemes are stable, the amplif ositive, except for Crank-Nicolson when  $F_o > 0.5$ .

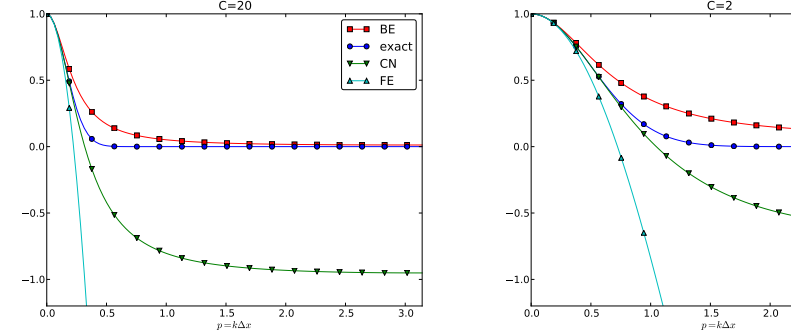


Figure 2: Amplification factors for large time steps.

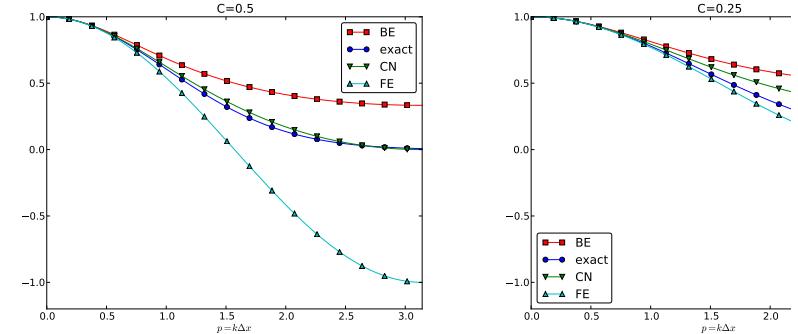


Figure 3: Amplification factors for time steps around the Forward Euler sta

The effect of negative amplification factors is that  $A^n$  changes sign from o re next, thereby giving rise to oscillations in time in an animation of the soluti figure 2 that for  $F_o = 20$ , waves with  $p \geq \pi/2$  undergo a damping close to  $-$  at the amplitude does not decay and that the wave component jumps up an or  $F_o = 2$  we have a damping of a factor of 0.5 from one time level to the nex uch smaller than the exact damping. Short waves will therefore fail to be effect hese waves will manifest themselves as high frequency oscillatory noise in the

A value  $p = \pi/4$  corresponds to four mesh points per wave length of  $e^{ikx}$  mplies only two points per wave length, which is the smallest number of points p represent the wave on the mesh.



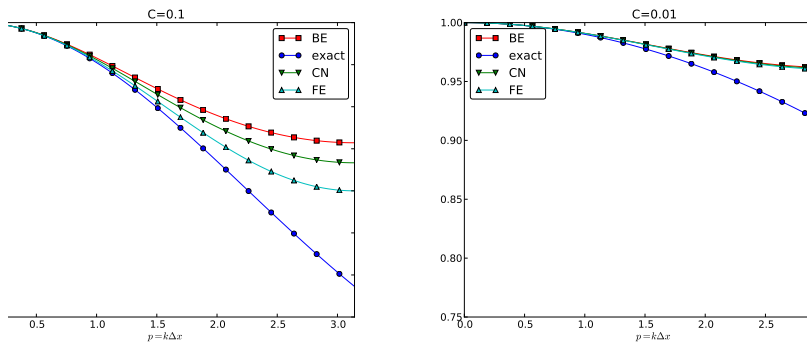


Figure 4: Amplification factors for small time steps.

nonstrate the oscillatory behavior of the Crank-Nicolson scheme, we choose a that leads to short waves with significant amplitude. A discontinuous  $I(x)$  serve this purpose.  
 $\gamma_o = \dots$

### Exercise 1: Use an analytical solution to formulate a 1D test

Exercise explores the exact solution (37). We shall formulate a diffusion problem in 1D for half of the Gaussian pulse. Then we shall investigate the impact of a boundary condition, which we in general cases often are forced due if the solution is rough finite boundaries undisturbed.

The solution (37) is seen to be symmetric at  $x = c$ , because  $\partial u / \partial x = 0$  always vanishes. Use this property to formulate a complete initial boundary value problem in 1D in the diffusion equation  $u_t = \alpha u_{xx}$  on  $[0, L]$  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 of the scheme. Check if a one-sided difference for  $u_x(0, t)$ , say  $u_0 = u_1$ , destroys the second-order accuracy.

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

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

What is a suitable error measure for the present problem?

) Instead of using  $u(L, t) = 0$  as approximate boundary condition for letting a Gaussian pulse out of our finite domain, one may try  $u_x(L, t) = 0$  since the solution is quite flat. Argue that this condition gives a completely wrong asymptotic behavior. To do this, integrate the diffusion equation from 0 to  $L$ , integrate  $u_{xx}$  by parts, and use the divergence theorem 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

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

Since the integral of the initial pulse is 1.

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

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

where  $q$  is an unknown heat transfer coefficient and  $u_s$  is the surrounding temperature of the medium outside of  $[0, L]$ . (Note that arguing that  $u_s$  is approximately  $u(L, t)$  is not a good condition from the previous subexercise that is qualitatively wrong for large  $t$ .) Derive a problem for the error in the solution using (52) as boundary condition. Assume  $u_s = 0$  "outside the domain" as  $u \rightarrow 0$  for  $x \rightarrow \infty$ . Find a function  $q = q(t)$  such that the exact solution obeys the condition (52). Test some constant values of  $q$  and analyze the corresponding error function behavior. Also compute  $E(t)$  curves as suggested in the previous subexercise. Filename: `diffu_symmetric_gaussian.py`.

### Exercise 2: Use an analytical solution to formulate a 2D test

Generalize (37) to multi dimensions by assuming that one-dimensional solutions can be used to solve  $u_t = \alpha \nabla^2 u$ . Use this solution to formulate a 2D test case where the peak is at the origin and where the domain is a rectangle in the first quadrant. Use homogeneous boundary conditions  $\partial u / \partial n = 0$  wherever possible, and use exact Dirichlet conditions on the remaining boundaries. Filename: `diffu_symmetric_gaussian_2D.pdf`.

### Exercise 3: Examine stability of a diffusion model with a source term

Consider a diffusion equation with a linear  $u$  term:

$$u_t = \alpha u_{xx} + \beta u.$$

) Derive in detail a Forward Euler scheme, a Backward Euler scheme, and a Crank-Nicolson type of diffusion model. Thereafter, formulate a  $\theta$ -rule to summarize the three schemes.

) Assume a solution like (38) and find the relation between  $a$ ,  $k$ ,  $\alpha$ , and  $\beta$ .

) Calculate the stability of the Forward Euler scheme. Design numerical experiments to verify the results.

) Repeat c) for the Backward Euler scheme.

) Repeat c) for the Crank-Nicolson scheme.

oes the extra term  $bu$  impact the accuracy of the three schemes?

Compare the numerical and exact amplification factors, either in graphs or by hand (or both).

`diffu_stab_uterm.pdf`.

## Exercises

### Exercise 4: Stabilizing the Crank-Nicolson method by Rannacher

It is known that the Crank-Nicolson method may give rise to non-physical oscillations for diffusion equations if the initial data exhibit jumps (see Section 2.6). Rannacher proposed a stabilizing technique consisting of using the Backward Euler scheme for the first  $m$  steps with step length  $\frac{1}{2}\Delta t$ . One can generalize this idea to taking  $2m$  time steps of the Backward Euler method and then continuing with the Crank-Nicolson method, second-order in time. The idea is that the high frequencies of the initial solution are damped out, and the Backward Euler scheme treats these high frequencies correctly. The frequency content of the solution is gone and the Crank-Nicolson method will inherit this idea for  $m = 1, 2, 3$  on a diffusion problem with a discontinuous initial condition. Compute the convergence rate using the solution (34) with the boundary conditions (35)-(36) such that the conditions are in the vicinity of  $\pm 1$ . For example,  $t < 5 \cdot 10^{-2}$  m. Compute the analytical solution.

### Exercise 5: Energy estimates for diffusion problems

One often concerns so-called *energy estimates* for diffusion problems that can be obtained by analytical insight and for verification of implementations.

Consider first with a 1D homogeneous diffusion equation with zero Dirichlet conditions:

$$\begin{aligned} u_t &= \alpha u_{xx}, & x \in \Omega = (0, L), \quad t \in (0, T], \\ u(0, t) &= u(L, t) = 0, & t \in (0, T], \\ u(x, 0) &= I(x), & x \in [0, L]. \end{aligned}$$

The energy estimate for this problem reads

$$\|u\|_{L^2} \leq \|I\|_{L^2},$$

where the  $\|\cdot\|_{L^2}$  norm is defined by

$$\|g\|_{L^2} = \sqrt{\int_0^L g^2 dx}.$$

The quantity  $\|u\|_{L^2}$  or  $\frac{1}{2}\|u\|_{L^2}^2$  is known as the *energy* of the solution, although it is not the energy of the system. A mathematical tradition has introduced the notion *energy*.

The estimate (56) says that the "size of  $u$ " never exceeds that of the initial condition. Equivalently, that the area under the  $u$  curve decreases with time.

To show (56), multiply the PDE by  $u$  and integrate from 0 to  $L$ . Use that  $u u_t$  is the time derivative of  $\frac{1}{2}u^2$  and that  $u_x u$  can be integrated by parts to form an identity. That the time derivative of  $\|u\|_{L^2}^2$  must be less than or equal to zero. Integrate and derive (56).

Now we address a slightly different problem,

$$\begin{aligned} u_t &= \alpha u_{xx} + f(x, t), & x \in \Omega = (0, L), \quad t \in (0, T], \\ u(0, t) &= u(L, t) = 0, & t \in (0, T], \\ u(x, 0) &= 0, & x \in [0, L]. \end{aligned}$$

The associated energy estimate is

$$\|u\|_{L^2} \leq \|f\|_{L^2}.$$

(This result is more difficult to derive.)

Now consider the compound problem with an initial condition  $I(x)$  and a source term  $f(x, t)$ :

$$\begin{aligned} u_t &= \alpha u_{xx} + f(x, t), & x \in \Omega = (0, L), \quad t \in (0, T], \\ u(0, t) &= u(L, t) = 0, & t \in (0, T], \\ u(x, 0) &= I(x), & x \in [0, L]. \end{aligned}$$

Show that if  $w_1$  fulfills (53)-(55) and  $w_2$  fulfills (58)-(60), then  $u = w_1 + w_2$  is the solution to (62)-(64). Using the triangle inequality for norms,

$$\|a + b\| \leq \|a\| + \|b\|,$$

show that the energy estimate for (62)-(64) becomes

$$\|u\|_{L^2} \leq \|I\|_{L^2} + \|f\|_{L^2}.$$

One application of (65) is to prove uniqueness of the solution. Suppose  $u_1$  and  $u_2$  are two solutions to (62)-(64). Show that  $u = u_1 - u_2$  then fulfills (62)-(64) with  $f = 0$  and  $I = 0$ . Use the energy estimate to show that the energy must be zero for all times and therefore that  $u_1 = u_2$ , which proves the solution is unique.

Generalize (65) to a 2D/3D diffusion equation  $u_t = \nabla \cdot (\alpha \nabla u)$  for  $x \in \Omega$ .

**Hint.** Use integration by parts in multi dimensions:

$$\int_{\Omega} u \nabla \cdot (\alpha \nabla u) dx = - \int_{\Omega} \alpha \nabla u \cdot \nabla u dx + \int_{\partial \Omega} u \alpha \frac{\partial u}{\partial n},$$

where  $\frac{\partial u}{\partial n} = \mathbf{n} \cdot \nabla u$ ,  $\mathbf{n}$  being the outward unit normal to the boundary  $\partial \Omega$  of the domain  $\Omega$ .

re also consider the multi-dimensional PDE  $u_t = \nabla \cdot (\alpha \nabla u)$ . Integrate both sides Gauss' divergence theorem,  $\int_{\Omega} \nabla \cdot \mathbf{q} \, dx = \int_{\partial\Omega} \mathbf{q} \cdot \mathbf{n} \, ds$  for a vector field  $\mathbf{q}$ . Show that homogeneous Neumann conditions on the boundary,  $\partial u / \partial n = 0$ , area under the  $u$  constant in time and

$$\int_{\Omega} u \, dx = \int_{\Omega} I \, dx.$$

Write a code in 1D, 2D, or 3D that can solve a diffusion equation with a source addition  $I$ , and zero Dirichlet or Neumann conditions on the whole boundary. Then use (65) and (66) as a partial verification of the code. Choose some functions  $I$  that (65) is obeyed at any time when zero Dirichlet conditions are used. Iterate  $I$  functions and check that (66) is fulfilled when using zero Neumann conditions.

Write a list of some possible bugs in the code, such as indexing errors in arrays, failure at boundary conditions, evaluation of a term at a wrong time level, and similar bugs, see if the verification tests from the previous subexercise pass or fail. The analysis shows how strong the energy estimates and the estimate (66) are for pointwise the implementation.

`diffu_energy.pdf`.

## References

W. S. C. Chen. Finite element solution of diffusion problems with irregular data. *Numerische Mathematik*, 43:309–327, 1984.

## Index

amplification factor, 13

energy estimates (diffusion), 19  
explicit discretization methods, 4

implicit discretization methods, 5

stationary solution, 3