LIÈGE université
Sciences Appliquées

Lecture 4b Approximations of diffusions

Mathématiques appliquées (MATH0504-1)
B. Dewals, C. Geuzaine

Numerical solutions of PDEs

We have found simple formulas for solutions to some first and second order PDEs, but the solution of many PDEs cannot be written simply

Even when there is a formula, it might be so complicated that we would prefer to visualize a typical solution by looking at its graph

Let us start investigating if we can reduce the process of solving a PDE with its auxiliary conditions to a finite number of arithmetic calculations that can be carried out by computer

Numerical solutions of PDEs

There are dangers in doing so:

- If the numerical method is not carefully designed, the computed solution may substantially differ from the actual one.
- For difficult problems the computation could easily take so long (years…) that it would not be tractable

Two well known numerical methods are Finite Difference (FD) and Finite Element (FE): we will introduce both.

Finite Volume (FV) is another numerical method, widely used in fluid flow computations.

Learning objectives

Review elementary finite difference (FD) schemes

Become aware of the existence of stability conditions for the design of numerical schemes

Derive formally the stability condition for the 1D diffusion equation

Understand the concepts of amplification factor, and stable / unstable modes

Discover that numerical schemes may be unstable, conditionally stable or unconditionally stable

Outline

1. Review of Finite Difference Schemes
2. Application to the Diffusion Equation
3. Numerical experiments
4. Derivation of a stability criterion for an explicit discretization of the 1D diffusion equation
5. Crank-Nicholson scheme

LIÈGE université
Sciences Appliquées

1 – Review of Finite Difference Schemes

In this section we review the basics of finite difference numerical schemes.

Finite differences

Finite difference schemes consist in replacing each derivative by a difference quotient.

Consider a function $u(x)$ of one variable.

Choose a *mesh size* Δx.

Approximate the value $u(j\Delta x)$ for $x = j\Delta x$ by a number u_j indexed by an integer j:

$$u_j \sim u(j\Delta x)$$

Approximations for the first derivative

The three standard approximations for the first derivative $\frac{\partial u}{\partial x}(j\Delta x)$ are

The backward difference: $\frac{u_j - u_{j-1}}{\Delta x}$

The centered difference: $\frac{u_{j+1} - u_{j-1}}{2\Delta x}$

The forward difference: $\frac{u_{j+1} - u_j}{\Delta x}$

Approximations for the first derivative

Each of them is a correct approximation, as shown by a Taylor expansion (assuming u is a C^4 function):

$$u(x+\Delta x) = u(x) + u'(x)\Delta x + \tfrac{1}{2}u''(x)(\Delta x)^2 + \tfrac{1}{6}u'''(x)(\Delta x)^3 + O(\Delta x)^4$$

Replacing Δx by $-\Delta x$, we get

$$u(x-\Delta x) = u(x) - u'(x)\Delta x + \tfrac{1}{2}u''(x)(\Delta x)^2 - \tfrac{1}{6}u'''(x)(\Delta x)^3 + O(\Delta x)^4$$

We deduce that

$$u'(x) = \frac{u(x) - u(x-\Delta x)}{\Delta x} + O(\Delta x)$$

↓ Taking $x = j\,\Delta x$

The backward difference: $\dfrac{u_j - u_{j-1}}{\Delta x}$

Approximations for the first derivative

Similarly for all three standard approximations:

$$u'(x) = \frac{u(x) - u(x - \Delta x)}{\Delta x} + O(\Delta x)$$ Backward difference

$$= \frac{u(x + \Delta x) - u(x)}{\Delta x} + O(\Delta x)$$ Forward difference

$$= \frac{u(x + \Delta x) - u(x - \Delta x)}{2\Delta x} + O(\Delta x)^2$$ Centred difference

Taking $x = j\Delta x$ we see that backward and forward differences are correct approximations to the order $O(\Delta x)$; and centered differences is a correct approximation to the order $O(\Delta x)^2$

Approximations for the second derivative

The simplest approximation for the second derivative is the

$$\text{centered second difference:} \quad u''(j\Delta x) \sim \frac{u_{j+1} - 2u_j + u_{j-1}}{(\Delta x)^2}$$

which is justified by the same Taylor expansions as used before, which when added give:

$$u''(x) = \frac{u(x + \Delta x) - 2u(x) + u(x - \Delta x)}{(\Delta x)^2} + O(\Delta x)^2$$

Centered second difference is thus valid with an error of $O(\Delta x)^2$

Functions of two variables

For a function of 2 variables $u(x,t)$, we choose a mesh size for both variables:

$$u(j\,\Delta x, n\,\Delta t) \sim u_j^n$$

The forward difference approximations of the first order partial derivatives are then for example

$$\frac{\partial u}{\partial t}(j\,\Delta x,\ n\,\Delta t) \sim \frac{u_j^{n+1} - u_j^n}{\Delta t} \qquad \frac{\partial u}{\partial x}(j\,\Delta x,\ n\,\Delta t) \sim \frac{u_{j+1}^n - u_j^n}{\Delta x}$$

Two types of errors are generally introduced in computations based on such approximations

① *Truncation errors* refers to the error introduced in the solutions by the approximations themselves, that is, the $O(\Delta x)$ terms

- local truncation error: on one term
- global truncation error: on the actual solution, combining all local contributions

② *Round off errors* occurs in a real computation because only a certain number of digits, typically 8 or 16, are retained by the computer at each step of the computation

LIÈGE université
Sciences Appliquées

2 – Application to the Diffusion Equation

In this section we apply the finite difference method to the 1D diffusion equation.

Difference equation for Diffusion

Let us solve

$$u_t = u_{xx}, \qquad u(x,0) = \phi(x)$$

with a forward difference for u_t and a centered difference for u_{xx}.

We obtain the following difference equation:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\Delta x)^2}$$

The local truncation error is $O(\Delta t)$ for the left-hand side and $O(\Delta x)^2$ for the right-hand side.

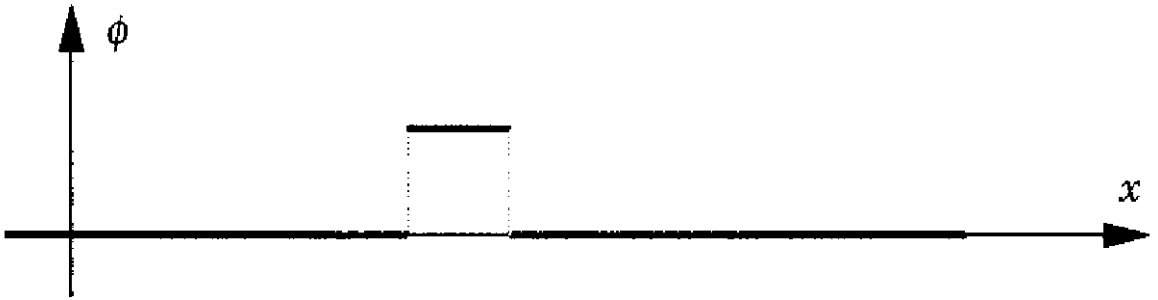
Difference equation for Diffusion

Let us choose a very small Δx and $\Delta t = (\Delta x)^2$.

This leads to the simplified equation:

$$u_j^{n+1} = u_{j+1}^n - u_j^n + u_{j-1}^n$$

Consider the following initial data $\phi(x)$



approximated by the following ϕ_j :

$$0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \qquad \rightarrow x$$

Difference equation for Diffusion

Applying the scheme we obtain:

t ↑

$n=4$	1	−4	10	−16	19	−16	10	−4	1	
$n=3$	0	1	−3	6	−7	6	−3	1	0	
$n=2$	0	0	1	−2	3	−2	1	0	0	
$n=1$	0	0	0	1	−1	1	0	0	0	
$n=0$	0	0	0	0	1	0	0	0	0	$\rightarrow x$

This is *clearly* not correct: the initial data gets amplified and oscillates – which is not expected for diffusion!

Indeed, the maximum principle tells us that the true solution is bounded by the minimum and maximum values in the initial condition (0 and 1 here) .

Difference equation for Diffusion

We should analyse this, because we see that small truncation errors do not guarantee that the solution will be close to the true solution!

LIÈGE université
Sciences Appliquées

3 – Numerical experiments

In this section we perform some numerical experiments to highlight the influence of the choice of the time step (with respect to the grid spacing) on the stability of the computation.

Let us have another try!

We again use a forward difference for u_t and a centered difference for u_{xx} . In contrast with the first attempt, though, we do not specify *yet* the choice of the time step Δt and the mesh size Δx.

We introduce the following notation:

$$s = \frac{\Delta t}{(\Delta x)^2}$$

The difference equation

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\Delta x)^2}$$

becomes $u_j^{n+1} = s\left(u_{j+1}^n + u_{j-1}^n\right) + (1 - 2s)u_j^n$

Explicit *vs.* implicit numerical schemes

The scheme

$$u_j^{n+1} = s\left(u_{j+1}^n + u_{j-1}^n\right) + (1-2s)u_j^n$$

is said to be explicit because the values at time step $n+1$ are given explicitly in terms of the values at the earlier times.

In contrast, one example of an implicit scheme would write:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{(\Delta x)^2}$$

Explicit *vs.* implicit numerical schemes

A scheme may also be (semi-)implicit, such as:

$$\frac{u_j^{n+1}-u_j^n}{\Delta t}=(1-\theta)\frac{u_{j+1}^n-2u_j^n+u_{j-1}^n}{(\Delta x)^2}+\theta\frac{u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1}}{(\Delta x)^2}$$

where θ is a parameter usually set between 0 an 1.

Opting for such a scheme generally improves the stability of the scheme; but it requires the resolution of (large) systems of algebraic equations.

The value of θ may be chosen to enhance the scheme accuracy. We will come back to this later.

Now, let us use the explicit scheme to solve with Matlab or Python a standard diffusion problem

The explicit scheme writes:

$$u_j^{n+1} = s\left(u_{j+1}^n + u_{j-1}^n\right) + (1-2s)u_j^n$$

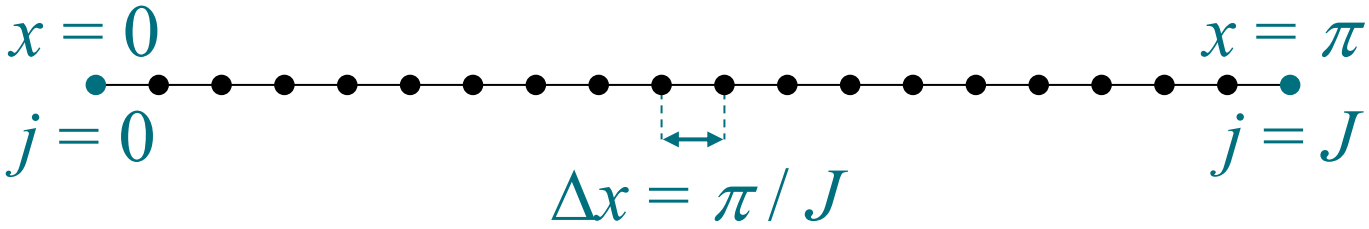
Consider the following standard problem:

$$\begin{aligned} u_t &= u_{xx} && \text{for } 0 < x < \pi, t > 0 \\ u &= 0 && \text{at } x = 0, \pi \end{aligned}$$

$$u(x,0) = \phi(x) = \begin{cases} x & \text{in } \left(0, \frac{\pi}{2}\right) \\ \pi - x & \text{in } \left(\frac{\pi}{2}, \pi\right). \end{cases}$$

Now, let us use the explicit scheme to solve with Matlab or Python a standard diffusion problem

First, discretize in space the 1D domain $(0, \pi)$, with a grid of $J + 1$ nodes and a spacing $\Delta x = \pi / J$.



The discrete boundary and initial conditions are

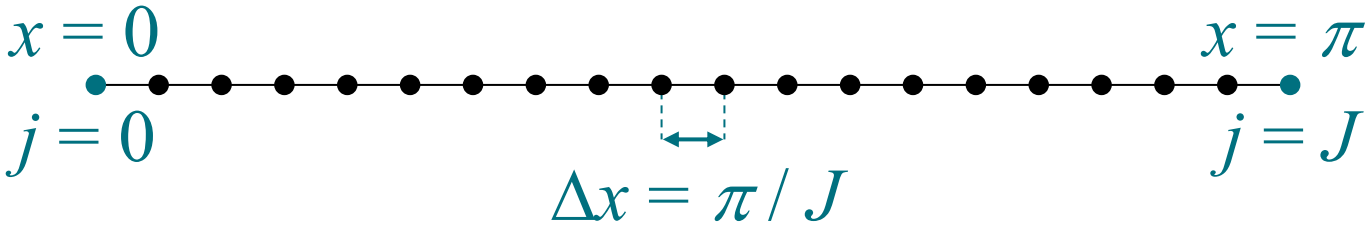
$$u_0^n = u_J^n = 0 \qquad \text{and} \qquad u_j^0 = \phi(j\Delta x)$$

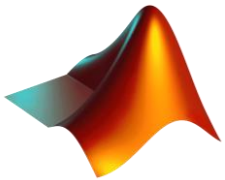
$$\text{with } \phi(x) = \begin{cases} x & \text{in } \left(0, \frac{\pi}{2}\right) \\ \pi - x & \text{in } \left(\frac{\pi}{2}, \pi\right). \end{cases}$$

Hands on: launch Matlab or Python and examine how the computation behaves…

① Create the domain

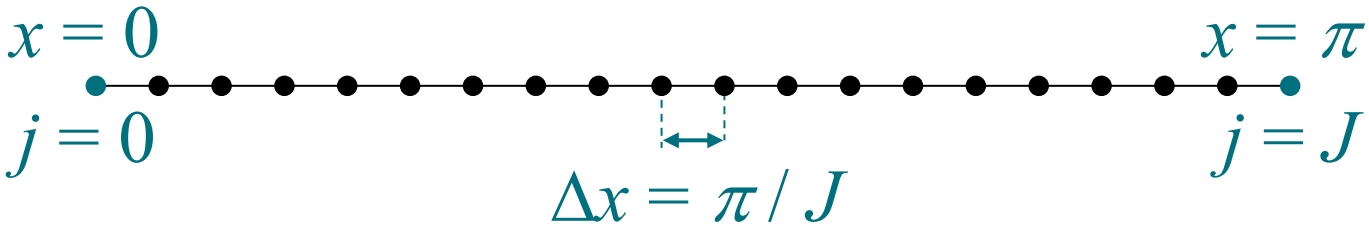
Consider $J = 20$





```
x_start = 0; x_end = pi; J = 20;
dx = (x_end - x_start) / J;
X = [x_start:dx:x_end];
```

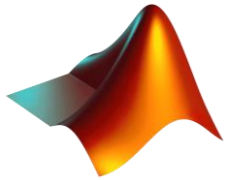
Hands on: launch Matlab or Python and examine how the computation behaves…



```
import numpy as np
import matplotlib.pyplot as plt
x_start = 0.0
x_end = np.pi
J = np.uint32(20)
dx = (x_end-x_start)/J
X = np.arange(x_start,x_end+dx, dx)
```

Hands on: launch Matlab or Python and examine how the computation behaves…

② You will compute the solution for $t = \pi^2 / 25$ by considering $s = \Delta t / (\Delta x)^2 = 0.4$.

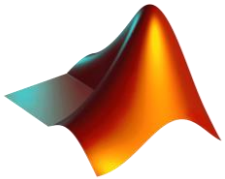


```
t_end = pi^2 / 25;
s = 0.4;
dt = s * dx^2;
```

```
t_end = np.pi**2 / 25
s = 0.4
dt = s * dx**2
```

Hands on: launch Matlab or Python and examine how the computation behaves…

③ Initialize a matrix to store the solution:

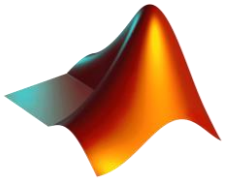


```
N = uint32(t_end / dt);
U = zeros(N+1, J+1);
```

```
N = np.uint32(t_end/dt)
U = np.zeros((N+1, J+1))
```

Hands on: launch Matlab or Python and examine how the computation behaves…

④ Prescribe the initial condition …

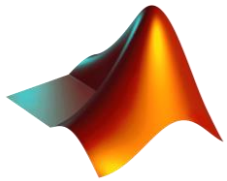


```
II = find(X < pi/2);
U(1, II) = X(II);
II = find(X >= pi/2);
U(1, II) = pi - X(II);
```

```
II = np.argwhere(X<np.pi/2)
U[0,II] = X[II]
II = np.argwhere(X>=np.pi/2)
U[0, II] = np.pi - X[II]
```

Hands on: launch Matlab or Python and examine how the computation behaves…

⑤ … and the boundary conditions

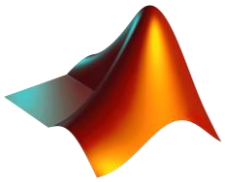


```
U(:, 1) = 0;
U(:, end) = 0;
```

```
U[:,0] = 0.0
U[:, -1] = 0.0
```

Hands on: launch Matlab or Python and examine how the computation behaves…

⑥ Now, you are ready for your first numerical resolution of a PDE:

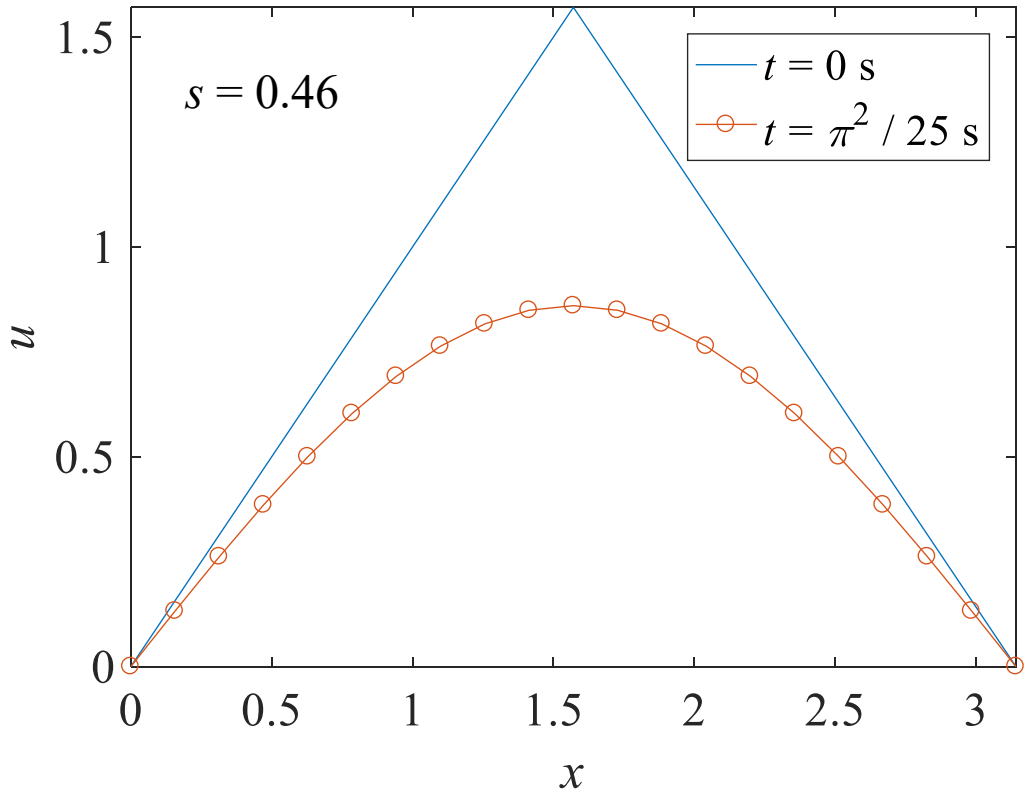


```
for n = 1:N
   U(n+1, 2:end-1) …
   = s*(U(n, 3:end) + U(n, 1:end-2))…
      + (1-2*s) * U(n, 2:end-1);
End
```

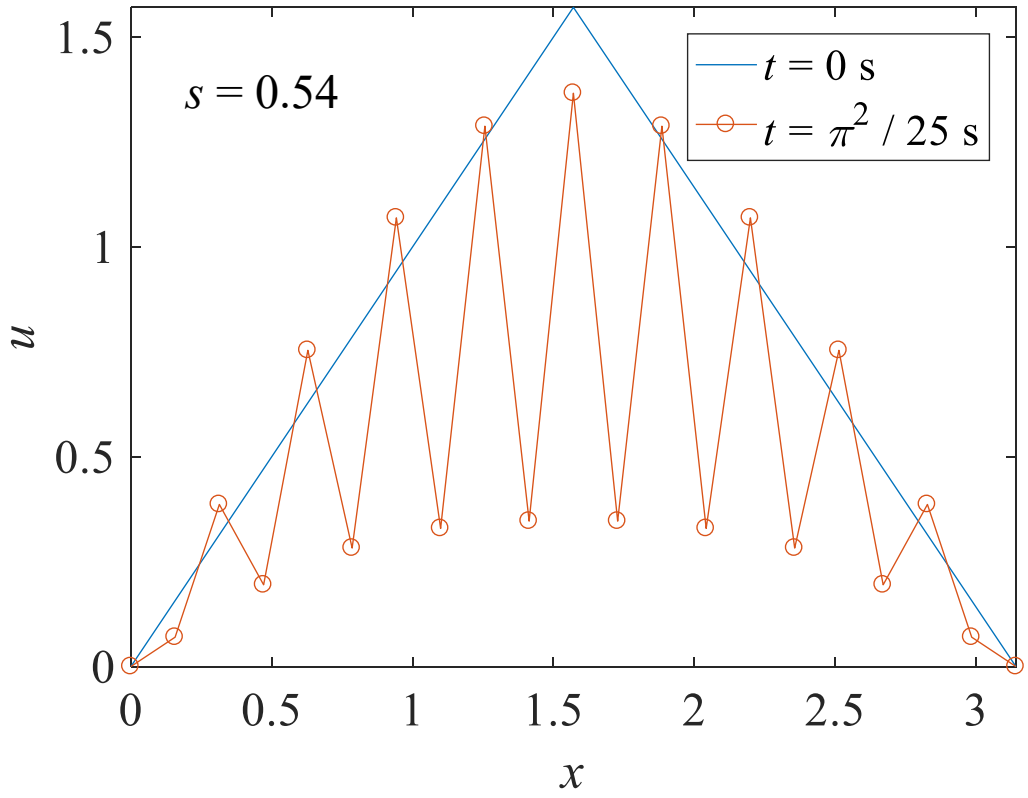
```
for n in range(N):
   U[n+1, 1:-1] = s*(U[n,2:] + U[n,0:-2])
+ (1-2*s) * U[n, 1:-1]
```

$$u_j^{n+1} = s\left(u_{j+1}^n + u_{j-1}^n\right) + (1-2s)u_j^n$$

Hands on: launch Matlab or Python and examine how the computation behaves…



Hands on: launch Matlab or Python and examine how the computation behaves…



LIÈGE université
Sciences Appliquées

4 – Derivation of a stability criterion

In this section, we derive a theoretical stability criterion for an explicit discretization of the 1D diffusion wave equation. The theoretical criterion agrees very well with the observations made in the numerical experiments described in the previous section.

A stability criterion can be formally derived

Heuristically, we find that the computation remains stable provided that $s < 1/2$.

A hint on this can be found in the discretized equation

$$u_j^{n+1} = s\left(u_{j+1}^n + u_{j-1}^n\right) + (1 - 2s)u_j^n$$

where the coefficient $1 - 2\,s$ becomes negative for $s > 1/2$.

Let us demonstrate now the stability condition

$$s < 1/2.$$

We proceed in six simple steps.

① Separate the variables in the difference equation

We look for solutions of the difference equation

$$u_j^{n+1} = s\left(u_{j+1}^n + u_{j-1}^n\right) + (1-2s)u_j^n$$

of the form

$$u_j^n = X_j\, T_n,$$

with X_j a function of space only ($x = j\,\Delta x$), and

T_n a function of time only ($t = n\,\Delta t$).

Substituting in the difference equation, we get

$$X_j\, T_{n+1} = s\left(X_{j+1}\, T_n + X_{j-1}\, T_n\right) + \left(1-2s\right) X_j\, T_n$$

① Separate the variables in the difference equation

The difference equation may be rewritten as

$$\frac{T_{n+1}}{T_n} = 1 - 2s + s\frac{X_{j+1} + X_{j-1}}{X_j} = \xi\left(\cancel{n}, \cancel{j}\right)$$

The left-hand side (LHS) and the right-hand side (RHS) of this equation are functions of independent variables (respectively n and j).

Therefore, the equality may hold only if each side is a constant *independent* of n and j.

We note this constant ξ.

② Solve the time equation

From

$$\frac{T_{n+1}}{T_n} = \xi$$

we get

$$T_n = \xi^n . T_0$$

The factor ξ plays a major part in the assessment of the stability of numerical schemes.

It is called the *amplification factor*.

③ Solve the spatial equation

The difference equation

$$1-2s+s\frac{X_{j+1}+X_{j-1}}{X_j}=\xi$$

may be rewritten

$$s\left(X_{j+1}-2X_j+X_{j-1}\right)+\left(1-\xi\right)X_j=0$$

This is a discretized form of a second-order ODE, which has sine and cosine solutions.

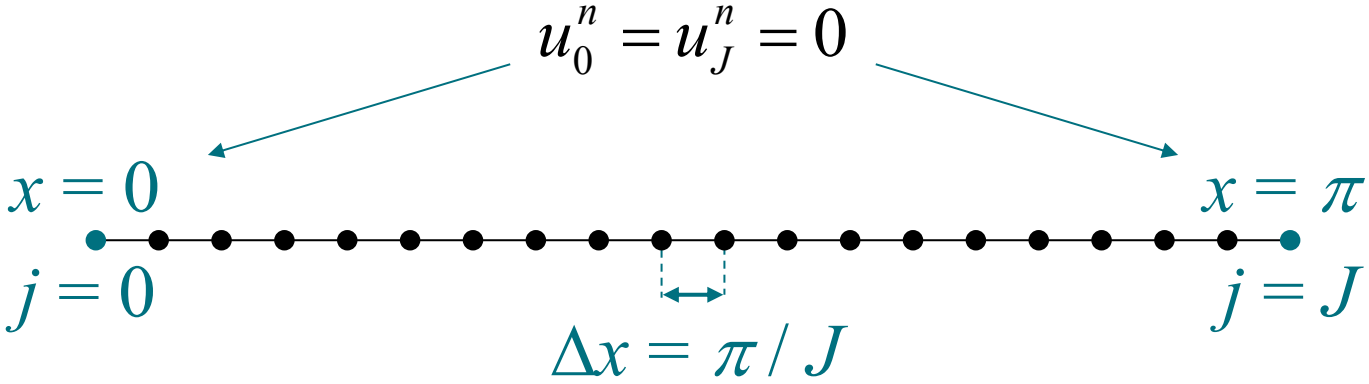
Therefore, we guess solutions of the form

$$X_j=A\cos\theta j+B\sin\theta j$$

with A, B arbitrary constants and θ to be determined.

④ Prescribe the boundary conditions

The boundary conditions of the problem were formulated as



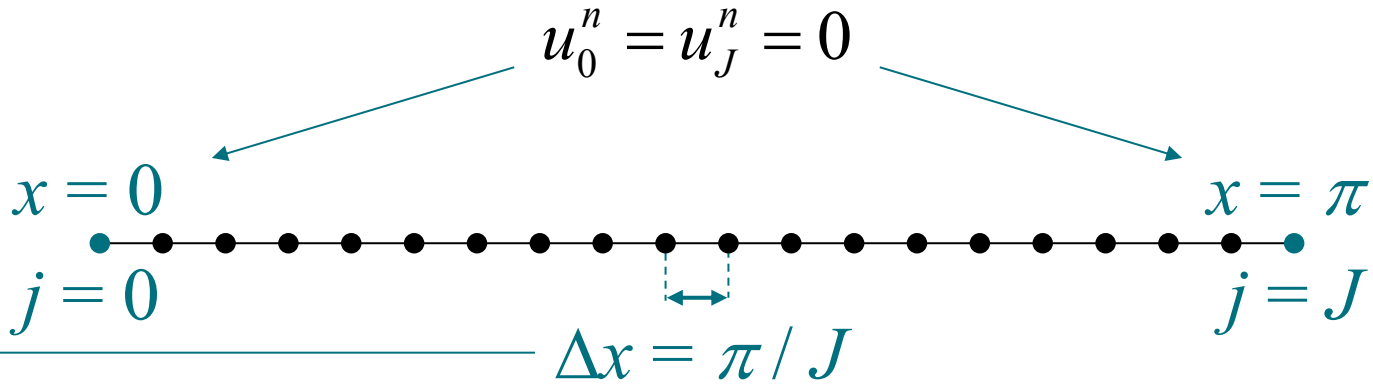
Setting $X_0 = 0$ at $j = 0$ implies that $A = 0$.

We can freely set $B = 1$, so that X_j becomes

$$X_j = A\cos\theta j + B\sin\theta j \quad \Rightarrow \quad X_j = \sin\theta j$$

④ Prescribe the boundary conditions

The boundary conditions of the problem were formulated as



Setting $X_J = 0$ at $j = J$ implies that $\sin J\theta = 0$.

Thus, $J\theta = k\,\pi$, with k an integer (*wave number*).

Combining with $J = \pi / \Delta x$ leads to $\theta = k\,\Delta x$, and

$$X_j = \sin\left(jk\Delta x\right)$$

⑤ Determine ξ from the spatial equation

Let us substitute $X_j = \sin(j\,k\,\Delta x)$ into the spatial equation

$$1 - 2s + s\frac{X_{j+1} + X_{j-1}}{X_j} = \xi$$

It leads to

$$1 - 2s + s\frac{\sin\left[(j+1)k\Delta x\right] + \sin\left[(j-1)k\Delta x\right]}{\sin(jk\Delta x)} = \xi$$

or

$$1 - 2s + s\frac{2\sin(jk\Delta x)\cos(k\Delta x)}{\sin(jk\Delta x)} = \xi$$

Hence, $\xi = \xi(k) = 1 - 2s\left[1 - \cos(k\Delta x)\right]$

⑥ Discuss the value of ξ as a function of s

We know from the solution of the time equation

$$T_n = \xi^n T_0$$

that $|\xi(k)|$ must remain below 1; otherwise

- the numerical solution would amplify with time, and we have no chance to get a stable solution
- we have no chance that the numerical solution converges towards the true solution

$$u(x,t) \to 0 \qquad \text{for} \qquad t = n\,\Delta t \to \infty.$$

Therefore let us look at the condition(s) to be prescribed on s so that $|\xi|$ remains below 1 for all k.

⑥ Discuss the value of ξ as a function of s

Since the factor $1-\cos(k\Delta x)$ in

$$\xi = \xi(k) = 1 - 2s\left[1-\cos(k\Delta x)\right]$$

varies between 0 and 2, we have

$$1 - 4\,s \le \xi(k) \le 1.$$

So, stability requires that $1 - 4\,s \ge -1$, hence

$$\boxed{s = \frac{\Delta t}{\Delta x^2} \le \frac{1}{2}}$$

This is the condition required for stability of the computation!

Which is the most unstable mode?

The analysis above shows that the "most dangerous" mode is the mode for which $\xi(k) = -1$.

From

$$\xi = \xi(k) = 1 - 2s\left[1 - \cos(k\Delta x)\right]$$

we can infer that this "most dangerous" mode corresponds to $\cos(k\,\Delta x) = -1$.

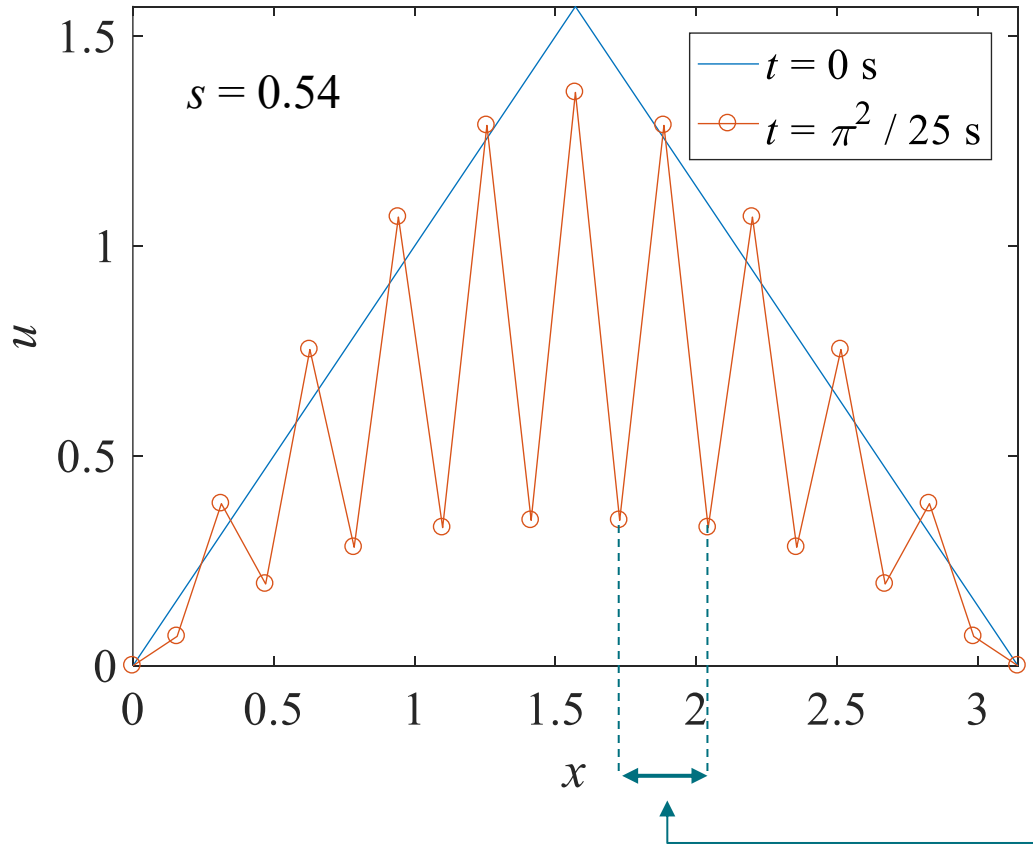
The corresponding wave number k is

$$k = \frac{\pi}{\Delta x}$$

which is a fairly high wave number.

Which is the most unstable mode?

This theoretical result is, again, fully consistent with the observations in the "failed" computation.



Period of the "most dangerous" mode according to theory

$= 2\pi / k$

$= 2\pi / (\pi / \Delta x)$

$= 2\,\Delta x$

$$X_j = \sin(jk\Delta x)$$

Looking back at what we just did…

From the difference equation

$$1-2s+s\frac{X_{j+1}+X_{j-1}}{X_j}=\xi$$

we could have been obtained

$$\xi=\xi(k)=1-2s\left[1-\cos(k\Delta x)\right]$$

by simply substituting into the difference equation an *exponential mode* of the form:

$$X_j=\left(e^{ik\Delta x}\right)^j$$

Looking back at what we just did…

So, in summary, what we did was to plug the separated solution mode

$$u_j^n = \left(e^{ik\Delta x}\right)^j \left[\xi(k)\right]^n$$

into the scheme and look for which values of

$$s = \frac{\Delta t}{(\Delta x)^2}$$

we have the amplification factor $|\xi(k)| \leq 1.$

LIÈGE université
Sciences Appliquées

5 – Crank-Nicholson scheme

In this section we analyse a different numerical scheme that exhibits unconditional stability

Implication for engineering

The stability criterion

$$s = \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2}$$

for the explicit scheme means that in practice the time steps must be taken very short.

Particularly, for the numerical scheme considered so far, Δt scales with the square of Δx!

Let us investigate whether a slightly different scheme may lead to a less restrictive stability condition … or even no stability condition at all.

Crank-Nicolson scheme

Let us come back to the (semi-)implicit scheme introduced earlier

$$\frac{u_j^{n+1}-u_j^n}{\Delta t}=(1-\theta)\frac{u_{j+1}^n-2u_j^n+u_{j-1}^n}{(\Delta x)^2}+\theta\frac{u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1}}{(\Delta x)^2}$$

with θ a number between 0 and 1.

If $\theta=0$, it reduces to the previous explicit scheme.

Otherwise, it is implicit since u^{n+1} appears on both sides of the equation. This means that, at each time step, a system of linear algebraic equations must be solved.

Let us analyze the stability of the scheme

As before, we plug the separated solution

$$u_j^n = \left(e^{ik\Delta x}\right)^j \left[\xi(k)\right]^n$$

into the difference equation

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = (1-\theta)\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\Delta x)^2} + \theta\frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{(\Delta x)^2}$$

It leads to

$$\frac{\xi - 1}{\Delta t} = (1-\theta)\frac{e^{ik\Delta x} - 2 + e^{-ik\Delta x}}{(\Delta x)^2} + \theta\xi\frac{e^{ik\Delta x} - 2 + e^{-ik\Delta x}}{(\Delta x)^2}$$

$$\xi - 1 = 2(1-\theta)s\left[\cos(k\Delta x) - 1\right] + \xi 2\theta s\left[\cos(k\Delta x) - 1\right]$$

Let us analyze the stability of the scheme

From

$$\xi - 1 = 2(1-\theta)s\left[\cos(k\Delta x) - 1\right] + \xi 2\theta s\left[\cos(k\Delta x) - 1\right]$$

we get the following expression for $\xi(k)$:

$$\xi = \frac{1 - 2(1-\theta)s\left[1 - \cos(k\Delta x)\right]}{1 + 2\theta s\left[1 - \cos(k\Delta x)\right]}$$

By examining this result, we find out that

- $\xi(k) \le 1$ is always true;
- while $\xi(k) \ge -1$ requires that:

$$s(1-2\theta)[1-\cos(k\Delta x)] \le 1$$

Let us analyze the stability of the scheme

If $1 - 2\ \theta \leq 0$, the condition

$$s(1-2\theta)[1-\cos(k\Delta x)] \leq 1$$

is fulfilled whatever the value of s

This means that for $\theta \geq 1/2$, the scheme is **unconditionally stable**.

The particular case of $\theta = 1/2$ is called the Crank-Nicholson scheme. It is second-order accurate in Δt.

For $\theta < 1/2$, the stability condition writes:

$$s \leq \frac{1}{2(1-2\theta)}$$

Take-home messages

We empirically observed a stability criterion, which we also demonstrated theoretically.

In a simple explicit discretization of the diffusion equation, the stability criterion requires that Δt has to scale with Δx^2, which may be very restrictive.

In contrast, implicit schemes (like the Crank-Nicholson scheme) can be unconditionally stable.

What's next

We will derive a general stability criterion for linear PDEs – the Von Neumann criterion – which applies systematically.