Part III – Numerical methods for time-dependent systems

MAT684 – Spring 2026 – Shukai Du

Topics:

- Numerical methods for diffusion equations
- Stability and Lax's equivalence theorem
- Stability analysis
- Numerical methods for hyperbolic equations

So far, we have studied numerical methods for ODEs (time discretization), as well as numerical methods for steady-state second-order elliptic equations (spatial discretization). In Part III, we will study numerical methods for time-dependent systems, which require us to combine what we have learned about time discretization for ODEs and spatial discretization for steady-state problems. The basic idea is as follows.

Suppose we are given a time-dependent partial differential equations (PDE) of the form

$$\partial_t u(x,t) = \mathcal{D}(u)(x,t) + g(x,t),$$

where u is the unknown solution, $\mathcal{D}$ is a linear differential operator (for example, $\mathcal{D}(u) = -\partial_{xx}u - \partial_{yy}u$), and g is a source term.

We first apply a spatial discretization, such as a finite difference method or a finite element method, as discussed in Part II. This leads to a system of ODEs:

$$\begin{pmatrix} u_1'(t) \\ u_2'(t) \\ u_3'(t) \\ \vdots \\ u_N'(t) \end{pmatrix} = D_h \begin{pmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \\ \vdots \\ u_N(t) \end{pmatrix} + \begin{pmatrix} g_1(t) \\ g_2(t) \\ g_3(t) \\ \vdots \\ g_N(t) \end{pmatrix}.$$

Here each $u_i(t)$ is a function of time and represents the ith degree of freedom in the spatial discretization at time t. For instance, $u_i(t)$ may represent the finite element solution at node i at time t. The matrix D_h represents the discretized differential operator $\mathcal{D}$, and $(g_1, \ldots, g_N)^\top$ is the vector obtained by discretizing the source term g.

Let

$$\boldsymbol{u}_h(t) = (u_1(t), \ldots, u_N(t))^\top, \qquad \boldsymbol{g}_h(t) = (g_1(t), \ldots, g_N(t))^\top,$$

where the subscript h indicates that $\boldsymbol{u}_h$ depends on the spatial mesh size h. Then the ODE system can be written more compactly as

$$\boldsymbol{u}_h'(t) = D_h \boldsymbol{u}_h(t) + \boldsymbol{g}_h(t). \tag{1}$$

The above equation (1) can be rewritten into a more standard form that appears in Part I:

$$\boldsymbol{u}_h'(t) = \boldsymbol{F}(t, \boldsymbol{u}_h), \quad \text{where} \quad \boldsymbol{F}(t, \boldsymbol{u}_h) := D_h \boldsymbol{u}_h + \boldsymbol{g}_h(t),$$

Note that the above formulation is exactly the type of ODE system that we studied systematically in Part I. Therefore, we can discretize the ODE system with a time-stepping method (e.g, Runge-Kutta):

$$\boldsymbol{u}_h^{n+1} = \boldsymbol{\Phi}_{\Delta t}(t_n, \boldsymbol{u}_h^n), \tag{2}$$

where Δt is the time step, $\boldsymbol{u}_h^n$ approximates $\boldsymbol{u}_h(t_n)$, and $\boldsymbol{\Phi}_{\Delta t}$ is the map defined by the chosen one-step method. For instance, if we use the forward Euler method, then we simply have

$$\boldsymbol{\Phi}_{\Delta t}(t_n, \boldsymbol{u}_h^n) = \boldsymbol{u}_h^n + \Delta t\big(D_h \boldsymbol{u}_h^n + \boldsymbol{g}_h(t_n)\big).$$

We call (1) a *semi-discretization* of the PDE, since only the spatial variable has been discretized. Correspondingly, we call (2) a *full discretization* of the PDE.

As in Parts I and II, the central themes in Part III are again consistency and stability. In particular, we care about the local truncation error of the full discretization (2), as well as the stability of the fully discrete scheme (2).

For the local truncation error, we will see that it depends on both the spatial discretization parameter h (or Δx) and the time step Δt. Thus, one important goal is to choose these parameters appropriately so that the total error is controlled efficiently.

We will also see that the stability of the full discretization depends on both the spatial discretization and the temporal discretization. Understanding this interaction is a central topic in Part III.

1 Numerical methods for diffusion equations

To begin, let us consider the one-dimensional diffusion equation:

$$\begin{aligned}
\partial_t u(x,t) - \partial_{xx} u(x,t) &= f(x,t) && \text{in} \quad (0,L)\times(0,T), && \text{(3a)}\\
u(x,0) &= u_{\text{init}}(x) && \text{for} \quad x \in (0,L), && \text{(3b)}\\
u(0,t) = g_0(t), \quad u(L,t) &= g_L(t) && \text{for} \quad t \in (0,T). && \text{(3c)}
\end{aligned}$$

The first equation (3a) is the governing PDE, describing the diffusion of a quantity $u(x,t)$ in space and time, with source term $f(x,t)$. The second equation specifies the initial condition, which prescribes the state of the system at time $t = 0$. The third equation represents the boundary conditions; here we impose Dirichlet boundary conditions, meaning that the value of u is prescribed at the endpoints by g_0 and g_L. Most importantly, the system (3) provides a mechanism for determining the solution u from the input data:

$$\text{Input data: } (f,\, u_{\text{init}},\, g_0,\, g_L) \longrightarrow \text{Output solution: } u.$$

This model arises in many physical contexts. For example, it describes the conduction of heat in a rod, where $u(x,t)$ represents the temperature distribution in space and time. It can also model the spreading of a substance, such as ink diffusing in water, where $u(x,t)$ represents the concentration of the substance.

From a mathematical point of view, this problem is well understood. Under suitable assumptions on the data f, u_{init}, g_0, and g_L, the problem is well posed in the sense that

a solution exists, is unique, and depends continuously on the given input data. This is a classical result in the theory of parabolic partial differential equations.

In practice, however, explicit solutions are rarely available, especially when the source term or boundary data are complicated. Therefore, our goal is to develop numerical methods for approximating the solution.

To begin, we consider perhaps the simplest numerical method for (3), based on a centered difference approximation for the spatial derivative and the forward Euler method for the time derivative.

Finite difference spatial discretization. Following the general idea introduced at the beginning of Part I, we discretize (3) in space using the finite difference method. To simplify the presentation, we assume homogeneous Dirichlet boundary conditions, that is,

$$u(0,t) = u(L,t) = 0.$$

We begin by introducing a uniform mesh on the interval $[0, L]$:

$$x_i = ih, \qquad i = 0, 1, \ldots, N, \qquad h = \frac{L}{N}.$$

Here, h is the mesh size and the points x_i are the grid points.

At each interior grid point x_i, $i = 1, \ldots, N-1$, we approximate the second derivative $\partial_{xx} u$ by the standard centered finite difference:

$$\partial_{xx} u(x_i, t) \;\approx\; \frac{u(x_{i+1}, t) - 2u(x_i, t) + u(x_{i-1}, t)}{h^2}.$$

We now introduce unknown functions $u_i(t)$ to approximate the exact values $u(x_i, t)$. Replacing $\partial_{xx} u(x_i, t)$ by the above finite difference approximation, and replacing $u(x_i, t)$ by $u_i(t)$, we obtain the semi-discrete system

$$\frac{d}{dt} u_i(t) + \frac{-u_{i+1}(t) + 2u_i(t) - u_{i-1}(t)}{h^2} = f(x_i, t), \qquad i = 1, \ldots, N-1.$$

The boundary conditions imply

$$u_0(t) = 0, \qquad u_N(t) = 0,$$

and the initial condition is discretized as

$$u_i(0) = u_{\text{init}}(x_i), \qquad i = 1, \ldots, N-1.$$

Thus, we obtain a system of ordinary differential equations in time for the unknown vector $(u_1(t), \ldots, u_{N-1}(t))$. In matrix form, this system can be written as

$$\frac{d}{dt}\mathbf{u}(t) + A\mathbf{u}(t) = \mathbf{f}(t), \tag{4}$$

where

$$\mathbf{u}(t) := \begin{pmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_{N-1}(t) \end{pmatrix}, \qquad \mathbf{f}(t) := \begin{pmatrix} f(x_1,t) \\ f(x_2,t) \\ \vdots \\ f(x_{N-1},t) \end{pmatrix},$$

and $A \in \mathbb{R}^{(N-1)\times(N-1)}$ is the tridiagonal matrix

$$A := \frac{1}{h^2} \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}.$$

The initial condition becomes

$$\mathbf{u}(0) = \begin{pmatrix} u_{\text{init}}(x_1) \\ u_{\text{init}}(x_2) \\ \vdots \\ u_{\text{init}}(x_{N-1}) \end{pmatrix}.$$

Note that both $\mathbf{f}(t)$ and the initial condition $(u_{\text{init}}(x_1), ..., u_{\text{init}}(x_{N-1}))$ are given as input data.

Remark 1.1. *For the tridiagonal matrix A, one can compute its eigenvalues and eigenvectors explicitly. Indeed, the eigenvalues are*

$$\lambda_k = \frac{1}{h^2}\Big(2 - 2\cos\frac{k\pi}{N}\Big) = \frac{4}{h^2}\sin^2\Big(\frac{k\pi}{2N}\Big), \qquad k = 1, \dots, N-1,$$

and the corresponding eigenvectors are given by

$$v_j^{(k)} = \sin\Big(\frac{jk\pi}{N}\Big), \qquad j = 1, \dots, N-1.$$

This explicit formula relies on the special structure of the matrix A. For more general spatial discretizations, it is usually not possible to compute the eigenvalues explicitly. Nevertheless, one can still derive useful qualitative properties and quantitative estimates for the resulting matrices.

Euler method for time discretization. Recall that we have obtained the system

$$\frac{d}{dt}\mathbf{u}(t) = \mathbf{f}(t) - A\mathbf{u}(t), \tag{5}$$

with initial condition

$$\mathbf{u}(0) = \mathbf{u}^0 := \begin{pmatrix} u_0(x_1) \\ u_0(x_2) \\ \vdots \\ u_0(x_{N-1}) \end{pmatrix}.$$

Here, $\mathbf{f}(t)$ and $\mathbf{u}^0$ are determined by the source term $f(x,t)$ and the initial condition $u_0(x)$, and are therefore given as input data.

Next, we discretize (5) in time using the forward Euler method. Let $\tau > 0$ be the time step size, and define

$$t_n = n\tau, \qquad n = 0, 1, 2, \dots$$

We use $\mathbf{u}^n$ to approximate $\mathbf{u}(t_n)$. Applying the forward Euler method to (5), we obtain

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\tau} = \mathbf{f}(t_n) - A\mathbf{u}^n.$$

Equivalently,

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \tau\big(\mathbf{f}(t_n) - A\mathbf{u}^n\big) = (I - \tau A)\mathbf{u}^n + \tau\mathbf{f}(t_n).$$

This gives a fully discrete scheme for approximating the solution of (3).

If we expand the above equation, we have

$$\begin{pmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{N-1}^{n+1} \end{pmatrix} = \begin{pmatrix} u_1^n \\ u_2^n \\ \vdots \\ u_{N-1}^n \end{pmatrix} + \tau \begin{pmatrix} f(x_1, t_n) \\ f(x_2, t_n) \\ \vdots \\ f(x_{N-1}, t_n) \end{pmatrix} - \frac{\tau}{h^2} \begin{pmatrix} 2u_1^n - u_2^n \\ -u_1^n + 2u_2^n - u_3^n \\ \vdots \\ -u_{N-3}^n + 2u_{N-2}^n - u_{N-1}^n \\ -u_{N-2}^n + 2u_{N-1}^n \end{pmatrix}.$$

Equivalently, we have

$$u_i^{n+1} = u_i^n + \tau f(x_i, t_n) - \frac{\tau}{h^2}\big(u_{i+1}^n - 2u_i^n + u_{i-1}^n\big), \qquad (6)$$
$$i = 1, \dots, N-1, \quad n = 0, 1, 2, \dots,$$

with the boundary values enforced by

$$u_0^n = 0, \qquad u_N^n = 0, \qquad n = 0, 1, 2, \dots.$$

Local truncation error. We next study the local truncation error of the fully discrete scheme (6). Recall that the local truncation error is obtained by substituting the exact solution $u(x_i, t_n)$ into (6) in place of the numerical values u_i^n, and then measuring the resulting residual. More precisely, we define

$$\mathrm{LTE}_i^n := u(x_i, t_{n+1}) - \left[u(x_i, t_n) + \tau f(x_i, t_n) - \frac{\tau}{h^2}\big(u(x_{i+1}, t_n) - 2u(x_i, t_n) + u(x_{i-1}, t_n)\big)\right],$$
$$i = 1, \dots, N-1, \quad n = 0, 1, 2, \dots.$$

Lemma 1.1 (Local truncation error)**.** *Assume that the exact solution u of (3) is sufficiently smooth. Then*

$$\mathrm{LTE}_i^n = \mathcal{O}(\tau^2 + \tau h^2).$$

Equivalently,

$$\frac{\mathrm{LTE}_i^n}{\tau} = \mathcal{O}(\tau + h^2).$$

Thus, the scheme is consistent of first order in time and second order in space.

Remark 1.2. *Recall that in Part I, we learned that for an ODE solver, if the local truncation error behaves like $\mathcal{O}(\tau^{p+1})$, then the global error is typically of order $\mathcal{O}(\tau^p)$, provided the method is convergent. For this reason, some authors define the local truncation error after dividing the residual by τ. In the present case, this alternative definition would be*

$$\frac{\mathrm{LTE}_i^n}{\tau}.$$

Under this convention, the local truncation error is of order $\mathcal{O}(\tau + h^2)$.

Proof of Lemma 1.1 . Since the exact solution satisfies

$$\partial_t u(x_i, t_n) - \partial_{xx} u(x_i, t_n) = f(x_i, t_n),$$

we may rewrite LTE_i^n as

$$\begin{aligned}\mathrm{LTE}_i^n = u(x_i, t_{n+1}) - u(x_i, t_n) - \tau \partial_t u(x_i, t_n) \\ - \tau \left(\frac{u(x_{i+1}, t_n) - 2u(x_i, t_n) + u(x_{i-1}, t_n)}{h^2} - \partial_{xx} u(x_i, t_n) \right).\end{aligned}$$

For the time discretization error, Taylor expansion in t around t_n gives

$$u(x_i, t_{n+1}) = u(x_i, t_n) + \tau \partial_t u(x_i, t_n) + \frac{\tau^2}{2} \partial_{tt} u(x_i, t_n) + \mathcal{O}(\tau^3).$$

Therefore,

$$u(x_i, t_{n+1}) - u(x_i, t_n) - \tau \partial_t u(x_i, t_n) = \mathcal{O}(\tau^2).$$

For the spatial discretization error, Taylor expansion in x around x_i at time t_n gives

$$\begin{aligned}u(x_{i+1}, t_n) &= u(x_i, t_n) + h\, \partial_x u(x_i, t_n) + \frac{h^2}{2} \partial_{xx} u(x_i, t_n) + \frac{h^3}{6} \partial_{xxx} u(x_i, t_n) + \mathcal{O}(h^4), \\ u(x_{i-1}, t_n) &= u(x_i, t_n) - h\, \partial_x u(x_i, t_n) + \frac{h^2}{2} \partial_{xx} u(x_i, t_n) - \frac{h^3}{6} \partial_{xxx} u(x_i, t_n) + \mathcal{O}(h^4).\end{aligned}$$

Adding these two expansions and subtracting $2u(x_i, t_n)$, we obtain

$$u(x_{i+1}, t_n) - 2u(x_i, t_n) + u(x_{i-1}, t_n) = h^2 \partial_{xx} u(x_i, t_n) + \mathcal{O}(h^4).$$

Dividing by h^2 and subtracting $\partial_{xx} u(x_i, t_n)$ gives

$$\frac{u(x_{i+1}, t_n) - 2u(x_i, t_n) + u(x_{i-1}, t_n)}{h^2} - \partial_{xx} u(x_i, t_n) = \mathcal{O}(h^2).$$

Multiplying by τ, we obtain

$$\tau \left(\frac{u(x_{i+1}, t_n) - 2u(x_i, t_n) + u(x_{i-1}, t_n)}{h^2} - \partial_{xx} u(x_i, t_n) \right) = \mathcal{O}(\tau h^2).$$

Combining the two estimates, we conclude that

$$\mathrm{LTE}_i^n = \mathcal{O}(\tau^2 + \tau h^2).$$

This completes the proof. □

From Lemma 1.1, we obtain

$$\frac{1}{\tau}\mathrm{LTE}_i^n = \mathcal{O}(\tau + h^2).$$

We see that this error consists of two contributions. The term

$$\frac{u(x_i, t_{n+1}) - u(x_i, t_n)}{\tau} - \partial_t u(x_i, t_n)$$

represents the error arising from the time discretization, while the term

$$\frac{u(x_{i+1}, t_n) - 2u(x_i, t_n) + u(x_{i-1}, t_n)}{h^2} - \partial_{xx} u(x_i, t_n)$$

represents the error arising from the spatial discretization.

Therefore, if we want these two contributions to be of the same order, it is natural to choose

$$\tau \sim h^2.$$

With this choice, the normalized local truncation error becomes

$$\frac{1}{\tau}\mathrm{LTE}_i^n = \mathcal{O}(h^2).$$

In fact, we have the following theorem.

Theorem 1.1 (Convergence of the explicit finite difference method)**.** *Assume that the exact solution u of (3) is sufficiently smooth on $[0, L] \times [0, T]$. Let u_i^n be the numerical solution generated by*

$$u_i^{n+1} = u_i^n + \tau f(x_i, t_n) + \frac{\tau}{h^2}\big(u_{i+1}^n - 2u_i^n + u_{i-1}^n\big), \qquad i = 1, \ldots, N-1, \tag{7}$$

with boundary values

$$u_0^n = 0, \qquad u_N^n = 0, \qquad n = 0, 1, 2, \ldots,$$

and initial values

$$u_i^0 = u(x_i, 0), \qquad i = 1, \ldots, N-1.$$

Let $\mu := \frac{\tau}{h^2}$. Then, if

$$\mu \le \frac{1}{2},$$

then there exists a constant C, independent of h and τ, such that

$$\max_{0 \le n \le T/\tau} \max_{1 \le i \le N-1} |u(x_i, t_n) - u_i^n| \le C(\tau + h^2).$$

In particular, the scheme is convergent, with first-order accuracy in time and second-order accuracy in space. Since the condition $\mu \le \frac{1}{2}$ implies $\tau \le \frac{1}{2}h^2$, it follows that

$$\max_{0 \le n \le T/\tau} \max_{1 \le i \le N-1} |u(x_i, t_n) - u_i^n| \le Ch^2.$$

Proof. Let

$$e_i^n := u(x_i, t_n) - u_i^n$$

be the error. Since the exact solution satisfies the scheme up to the local truncation error, we have

$$u(x_i, t_{n+1}) = u(x_i, t_n) + \tau f(x_i, t_n) + \frac{\tau}{h^2}\big(u(x_{i+1}, t_n) - 2u(x_i, t_n) + u(x_{i-1}, t_n)\big) + \mathrm{LTE}_i^n.$$

Subtracting (7) from this identity gives

$$e_i^{n+1} = e_i^n + \frac{\tau}{h^2}\big(e_{i+1}^n - 2e_i^n + e_{i-1}^n\big) + \mathrm{LTE}_i^n.$$

That is,

$$e_i^{n+1} = (1 - 2\mu)e_i^n + \mu e_{i-1}^n + \mu e_{i+1}^n + \mathrm{LTE}_i^n, \qquad i = 1, \ldots, N-1. \tag{8}$$

Also,

$$e_0^n = e_N^n = 0 \qquad \text{for all } n,$$

since both the exact and numerical solutions satisfy the same boundary conditions, and

$$e_i^0 = 0 \qquad \text{for } i = 1, \ldots, N-1,$$

because the initial values are exact.

Now define

$$E^n := \max_{0 \le i \le N} |e_i^n|.$$

Since $\mu \le \frac{1}{2}$, the coefficients in (8) are nonnegative and satisfy

$$(1 - 2\mu) + \mu + \mu = 1.$$

Therefore,

$$\begin{aligned} |e_i^{n+1}| &\le (1 - 2\mu)|e_i^n| + \mu|e_{i-1}^n| + \mu|e_{i+1}^n| + |\mathrm{LTE}_i^n| \\ &\le \big((1 - 2\mu) + \mu + \mu\big)E^n + \max_{1 \le j \le N-1} |\mathrm{LTE}_j^n| \\ &= E^n + \max_{1 \le j \le N-1} |\mathrm{LTE}_j^n|. \end{aligned}$$

Taking the maximum over $i = 1, \ldots, N-1$, we obtain

$$E^{n+1} \le E^n + \max_{1 \le j \le N-1} |\mathrm{LTE}_j^n|.$$

From the local truncation error estimate proved earlier in Lemma 1.1, we have

$$|\mathrm{LTE}_j^n| \le C(\tau^2 + \tau h^2).$$

Therefore,

$$E^{n+1} \le E^n + C(\tau^2 + \tau h^2).$$

Iterating this inequality and using $E^0 = 0$, we obtain

$$E^n \le n\, C(\tau^2 + \tau h^2).$$

Since $n \le T/\tau$, it follows that

$$\begin{aligned} E^n &\le C\frac{T}{\tau}(\tau^2 + \tau h^2) \\ &= CT(\tau + h^2). \end{aligned}$$

Absorbing T into the constant, we conclude that

$$\max_{0\le n\le T/\tau}\ \max_{1\le i\le N-1} |e_i^n| \le C(\tau + h^2).$$

This proves the result. □

Example. Let us consider the heat equation on the interval $[0, 1]$:

$$\partial_t u - \partial_{xx} u = 0, \qquad 0 < x < 1,\ t > 0,$$

with homogeneous boundary conditions

$$u(0, t) = u(1, t) = 0.$$

We choose the initial condition to be a Gaussian centered at $x = \frac{1}{2}$:

$$u(x, 0) = \exp\left(-\frac{(x - \frac{1}{2})^2}{2\sigma^2}\right),$$

where $\sigma > 0$ is small (for instance, $\sigma = 0.08$), so that the initial profile is localized.

We apply the explicit finite difference method

$$u_i^{n+1} = u_i^n + \mu\big(u_{i+1}^n - 2u_i^n + u_{i-1}^n\big), \qquad \mu = \frac{\tau}{h^2},$$

and compare two choices of μ:

$$\mu = \tfrac{1}{2}, \qquad \mu = \tfrac{1}{2} + 0.1.$$

The following MATLAB code generates both an animation of the numerical solution and a figure showing the final-time profile in the two cases.

```
clear; clc; close all;

% Spatial grid
L = 1;
N = 100;
h = L/N;
x = linspace(0,L,N+1)';
```

```
% Gaussian initial condition
sigma = 0.08;
u0 = exp(-((x-0.5).^2)/(2*sigma^2));

% Homogeneous Dirichlet boundary conditions
u0(1)   = 0;
u0(end) = 0;

% Final time
T = 0.006;

% Two choices of mu
mu1 = 0.5;
mu2 = 0.6;

tau1 = mu1*h^2;
tau2 = mu2*h^2;

M1 = floor(T/tau1);
M2 = floor(T/tau2);

% Actual final times reached
T1 = M1 * tau1;
T2 = M2 * tau2;

% Initialize solutions
u1 = u0;
u2 = u0;

u1_final = u1;
u2_final = u2;

figAnim = figure('Position',[100,100,1200,450]);
plotEvery = 2;   % reduce number of plotted frames

gifFile = 'heat_explicit_comparison.gif';

Mmax = max(M1,M2);

for n = 1:Mmax

    % Update first solution if still active
    if n <= M1
        u1_new = u1;
        u1_new(2:N) = u1(2:N) + mu1*(u1(3:N+1) - 2*u1(2:N) + u1(1:N-1));
        u1_new(1) = 0;
        u1_new(N+1) = 0;
        u1 = u1_new;
        u1_final = u1;
    end

    % Update second solution if still active
    if n <= M2
        u2_new = u2;
```

```
        u2_new(2:N) = u2(2:N) + mu2*(u2(3:N+1) - 2*u2(2:N) + u2(1:N-1));
        u2_new(1) = 0;
        u2_new(N+1) = 0;
        u2 = u2_new;
        u2_final = u2;
    end

    % Plot selected frames
    if mod(n,plotEvery)==0 || n==1 || n==Mmax
        subplot(1,2,1);
        plot(x,u1,'LineWidth',2);
        grid on;
        xlabel('x');
        ylabel('u');
        title(sprintf('\\mu = %.2f, t = %.4f',mu1,min(n,M1)*tau1));
        ylim([-0.2,1.1]);

        subplot(1,2,2);
        plot(x,u2,'LineWidth',2);
        grid on;
        xlabel('x');
        ylabel('u');
        title(sprintf('\\mu = %.2f, t = %.4f',mu2,min(n,M2)*tau2));

        Mplot = max(abs(u2));
        if isfinite(Mplot) && Mplot < 1
            ylim([-0.2,1.1]);
        elseif isfinite(Mplot)
            ylim([-1.1*Mplot,1.1*Mplot]);
        else
            ylim([-1,1]);
        end

        drawnow;

        % Capture frame for GIF
        frame = getframe(figAnim);
        im = frame2im(frame);
        [A,map] = rgb2ind(im,256);

        if n == 1
            imwrite(A,map,gifFile,'gif','LoopCount',Inf,'DelayTime',0.05);
        else
            imwrite(A,map,gifFile,'gif','WriteMode','append','DelayTime',0.05);
        end
    end
end

% Final-time comparison figure
figFinal = figure('Position',[100,100,1200,450]);

subplot(1,2,1);
plot(x,u1_final,'LineWidth',2);
grid on;
```

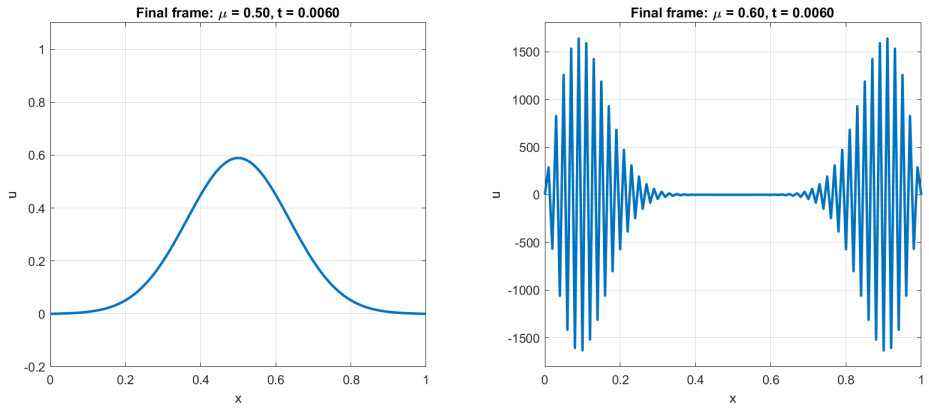
```
xlabel('x');
ylabel('u');
title(sprintf('Final frame: \\mu = %.2f, t = %.4f',mu1,T1));
ylim([-0.2,1.1]);

subplot(1,2,2);
plot(x,u2_final,'LineWidth',2);
grid on;
xlabel('x');
ylabel('u');
title(sprintf('Final frame: \\mu = %.2f, t = %.4f',mu2,T2));

Mfinal = max(abs(u2_final));
if isfinite(Mfinal) && Mfinal < 1
    ylim([-0.2,1.1]);
elseif isfinite(Mfinal)
    ylim([-1.1*Mfinal,1.1*Mfinal]);
else
    ylim([-1,1]);
end

saveas(figFinal,'heat_explicit_final_frame.png');
```

The following figure shows the numerical approximation at time $t = 0.006$ for two different choices of the parameter $\mu = \frac{\tau}{h^2}$:



This example indicates that the condition $\mu \leq \frac{1}{2}$ in Theorem 1.1 is important. For $\mu = \frac{1}{2}$, the numerical solution behaves well, while for $\mu = \frac{1}{2} + 0.1$, the method becomes unstable and does not provide convergent numerical solution.

Next, we shall study the stability and convergence of numerical methods for time-dependent systems in a more general setting.

2 Lax's equivalence theorem

Let us return to the general time-dependent problem:

$$\partial_t u(x,t) + \mathcal{D}u(x,t) = f(x,t) \qquad \text{in} \quad (0,L)\times(0,T), \tag{9a}$$

$$u(x,0) = u_{\text{init}}(x) \qquad \text{for} \quad x \in (0,L), \tag{9b}$$

$$u(0,t) = g_0(t), \qquad u(L,t) = g_L(t) \quad \text{for} \quad t \in (0,T). \tag{9c}$$

Here $\mathcal{D}$ is a spatial differential operator of the form:

$$(\mathcal{D}u)(x,t) = a_0(x)u(x,t) + a_1(x)\partial_x u(x,t) + a_2(x)\partial_{xx}u(x,t) + \cdots + a_n(x)\partial_x^n u(x,t).$$

For instance, in the diffusion equation considered previously, we had

$$\mathcal{D}u = -\partial_{xx}u.$$

If we wish to consider a heterogeneous diffusion problem, we may instead take

$$(\mathcal{D}u)(x,t) = -\partial_x\big(a(x)\partial_x u(x,t)\big),$$

where $a(x)$ is a diffusion coefficient. Later in this course, we shall study an important class of time-dependent systems called *hyperbolic equations*. A basic example is obtained by taking

$$\mathcal{D}u = \partial_x u.$$

For now, however, let us simply think of $\mathcal{D}$ as a general linear differential operator involving spatial derivatives.

It is easy to verify that such an operator is linear. Namely, if u_1 and u_2 are two functions and $\alpha, \beta \in \mathbb{R}$, then

$$\mathcal{D}(\alpha u_1 + \beta u_2) = \alpha \mathcal{D}u_1 + \beta \mathcal{D}u_2.$$

In PDE theory, one is interested in understanding the *well-posedness* of the system (9). More precisely, one asks the following questions:

1. Does a solution u exist? If so, in what function space does it exist, for example H^1, $W^{k,p}$, or $C^{k,\alpha}$?
2. Is the solution unique? For instance, can there exist multiple solutions satisfying the same equation together with the same initial and boundary conditions?
3. Does the solution depend continuously on the input data u_{init}, g_0, g_L, and f? If so, in what sense, and with respect to what norms or topologies?

If the answer to all of the above questions is yes, then we say that the problem is well-posed.

In this course, however, our primary interest is not the PDE theory itself, but rather how to solve such equations numerically. Thus, we shall assume that the continuous problem is well-posed, and focus instead on the design and analysis of numerical methods.

For a numerical method, the analogue of well-posedness is stability. Intuitively speaking, stability means that small perturbations in the numerical data, round-off errors, or intermediate errors generated during the computation do not grow uncontrollably.

On the other hand, consistency measures how accurately the numerical scheme approximates the original differential equation. A fundamental principle in numerical analysis is that, for a well-posed linear initial value problem, consistency together with stability implies convergence. This is the content of the celebrated Lax equivalence theorem.

In the rest of this section, we shall make these notions more precise.

2.1 Semi and full discrete scheme

Spatial discretization and the semi-discrete scheme. To keep the presentation simple, let us work on the same uniform mesh as before:

$$x_i = ih, \qquad i = 0, 1, \ldots, N, \qquad h = \frac{L}{N}.$$

Since the boundary conditions are prescribed by

$$u_0(t) = u(x_0, t) = g_0(t), \qquad u_N(t) = u(x_N, t) = g_L(t),$$

we only need to keep track of the unknowns associated with the interior grid points:

$$\boldsymbol{u}_h(t) = (u_1(t), \ldots, u_{N-1}(t))^\top,$$

where $u_i(t)$ approximates $u(x_i, t)$.

In general, after spatial discretization, the semi-discrete system takes the form

$$\boldsymbol{u}_h'(t) = D_h \boldsymbol{u}_h(t) + \boldsymbol{g}_h(t). \tag{10}$$

Here $D_h \in \mathbb{R}^{(N-1)\times(N-1)}$ is the matrix representing the spatial discretization of the differential operator in (9), and $\boldsymbol{g}_h(t)$ collects the contributions from the source term $f(x,t)$ together with those from the boundary data $g_0(t)$ and $g_L(t)$. The initial condition is

$$\boldsymbol{u}_h(0) = (u(x_1, 0), u(x_2, 0), \ldots, u(x_{N-1}, 0))^\top.$$

For instance, for the central difference scheme we considered previously, D_h takes the form:

$$D_h := -\frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}$$

Temporal discretization and the fully discrete scheme. We now discretize (10) in time. Let

$$t_n = n\tau, \qquad n = 0, 1, 2, \ldots,$$

where $\tau > 0$ is the time step, and let $\boldsymbol{u}_h^n$ denote an approximation to $\boldsymbol{u}_h(t_n)$. If we apply a one-step method such as a Runge–Kutta method, then the fully discrete scheme can be written in the form

$$\boldsymbol{u}_h^{n+1} = \boldsymbol{\Phi}_{h,\tau}(t_n, \boldsymbol{u}_h^n), \tag{11}$$

where $\boldsymbol{\Phi}_{h,\tau}$ is the *update map* determined by the spatial discretization together with the chosen time-stepping method. We also let $\boldsymbol{\Phi}_{h,\tau}^0$ be the update map in the case that $\boldsymbol{g}_h = 0$. We call $\boldsymbol{\Phi}_{h,\tau}^0$ the *homogeneous update map*.

For instance, if we apply the forward Euler method to the semi-discrete system (10), then the update map is given by

$$\boldsymbol{\Phi}_{h,\tau}(t_n, \boldsymbol{u}_h^n) = \boldsymbol{u}_h^n + \tau\big(D_h\boldsymbol{u}_h^n + \boldsymbol{g}_h(t_n)\big) = (I + \tau D_h)\boldsymbol{u}_h^n + \tau\boldsymbol{g}_h(t_n).$$

and $\boldsymbol{\Phi}_{h,\tau}^0(t_n, \boldsymbol{u}_h^n) = (I + \tau D_h)\boldsymbol{u}_h^n$. If instead we apply the backward Euler method, then the update map is given by

$$\boldsymbol{\Phi}_{h,\tau}(t_n, \boldsymbol{u}_h^n) = (I - \tau D_h)^{-1}\big(\boldsymbol{u}_h^n + \tau\boldsymbol{g}_h(t_{n+1})\big),$$

and $\boldsymbol{\Phi}_{h,\tau}^0(t_n, \boldsymbol{u}_h^n) = (I - \tau D_h)^{-1}\boldsymbol{u}_h^n$, provided that the matrix $I - \tau D_h$ is invertible.

Finally, we define the initial condition for the fully discrete scheme:

$$\boldsymbol{u}_h^0 = \big(u_{\text{init}}(x_1), u_{\text{init}}(x_2), \dots, u_{\text{init}}(x_{N-1})\big)^\top. \tag{12}$$

Equations (11) and (12) give the definition of the fully discrete scheme (with a flexible choice of spatial and temporal discretization) for the general time-dependent problem (9).

Discrete L^2-norm. For any vector $\boldsymbol{v} \in \mathbb{R}^M$ where $M \in \{N-1, N, N+1\}$, we introduce a discrete L^2-norm:

$$\|\boldsymbol{v}\|_h^2 := \sum_{j=1}^{M} v_j^2\, h,$$

where $h = L/N$. The reason for introducing this norm is that, if $f(x)$ is a sufficiently smooth function on $[0, L]$ and let $\boldsymbol{f}$ be any of the following vector of discretization:

$$\begin{aligned}\boldsymbol{f} = &(f(x_1), \dots, f(x_{N-1}))^\top, \quad \text{or} \quad (f(x_0), \dots, f(x_N))^\top,\\ &\text{or} \quad (f(x_0), \dots, f(x_{N-1}))^\top, \quad \text{or} \quad (f(x_1), \dots, f(x_N))^\top.\end{aligned}$$

Then

$$\|\boldsymbol{f}\|_h^2 \longrightarrow \int_0^L f^2(x)\, dx \qquad \text{as } h \to 0.$$

Namely, $\|\boldsymbol{f}\|_h$ approximates the continuous L^2-norm of f when h is small.

Note that the h-norm is just l_2 norm multiplying $\sqrt{h}$, namely, $\|\boldsymbol{v}\|_h = \sqrt{h}\|\mathbf{v}\|_2$. The above h-norm will be frequently used in the following texts.

2.2 Consistency

Let $\widetilde{\boldsymbol{u}}_h(t_n)$ denote the vector obtained by sampling the exact solution at the interior grid points at time $t = t_n$:

$$\widetilde{\boldsymbol{u}}_h(t_n) = \big(u(x_1, t_n), \dots, u(x_{N-1}, t_n)\big)^\top,$$

where $u(x,t)$ is the exact solution of (9).

We define the *local truncation error* at time t_n by

$$\boldsymbol{\xi}_h^n := \frac{1}{\tau}\Big(\widetilde{\boldsymbol{u}}_h(t_{n+1}) - \boldsymbol{\Phi}_{h,\tau}(t_n, \widetilde{\boldsymbol{u}}_h(t_n))\Big). \tag{13}$$

Note that for $\boldsymbol{\xi}_h^n$, we have divided the difference by the time-step size τ. This operation was explained in Remark 1.2.

Definition 2.1 (Consistency)**.** *Suppose that the PDE system (9) is well-posed and admits sufficient smooth solution. We say that the fully discrete scheme (11) is* consistent *if*

$$\sup_{0\le n\tau\le T} \|\boldsymbol{\xi}_h^n\|_h \to 0 \qquad \text{as } h \to 0,\ \tau \to 0.$$

More specifically, we say that the method (11) is of spatial order p and temporal order q if

$$\sup_{0\le n\tau\le T} \|\boldsymbol{\xi}_h^n\|_h = \mathcal{O}(h^p + \tau^q).$$

Simply put, a fully discrete scheme is consistent if its one-step error $\boldsymbol{\xi}_h^n$ (the local truncation error per time-step) tends to zero at any spatial and temporal grid location, as the mesh size h and the time step size τ both go to zero.

2.3 Stability

Let us define the error at $t = t_n$:

$$\boldsymbol{e}_h^n := \widetilde{\boldsymbol{u}}_h(t_n) - \boldsymbol{u}_h^n.$$

Then we have

$$\begin{aligned}
\boldsymbol{e}_h^{n+1} &= \widetilde{\boldsymbol{u}}_h(t_{n+1}) - \boldsymbol{u}_h^{n+1} \\
&= \widetilde{\boldsymbol{u}}_h(t_{n+1}) - \boldsymbol{\Phi}_{h,\tau}(t_n, \boldsymbol{u}_h^n) \\
&= \widetilde{\boldsymbol{u}}_h(t_{n+1}) - \boldsymbol{\Phi}_{h,\tau}(t_n, \widetilde{\boldsymbol{u}}_h(t_n)) + \boldsymbol{\Phi}_{h,\tau}(t_n, \widetilde{\boldsymbol{u}}_h(t_n)) - \boldsymbol{\Phi}_{h,\tau}(t_n, \boldsymbol{u}_h^n) \\
&= \tau\boldsymbol{\xi}_h^n + \boldsymbol{\Phi}_{h,\tau}(t_n, \widetilde{\boldsymbol{u}}_h(t_n)) - \boldsymbol{\Phi}_{h,\tau}(t_n, \boldsymbol{u}_h^n).
\end{aligned} \tag{14}$$

For all the time-stepping methods we will consider for the rest of this class, when applied to the linear semi-discrete system (10), the update map $\Phi_{h,\tau}$ is affine in the state variable. To be more precise, we have the following result.

Lemma 2.1. *For any well-defined Runge–Kutta methods (including forward Euler, backward Euler, and the trapezoidal method) applied to the ODE system (10), we have*

$$\boldsymbol{\Phi}_{h,\tau}(t_n, \boldsymbol{v}) - \boldsymbol{\Phi}_{h,\tau}(t_n, \boldsymbol{w}) = \boldsymbol{\Phi}^0_{h,\tau}(t_n, \boldsymbol{v} - \boldsymbol{w}).$$

In addition, the homogeneous update map $\boldsymbol{\Phi}^0_{h,\tau}$ is just a matrix-vector multiplication:

$$\boldsymbol{\Phi}^0_{h,\tau}(t, \boldsymbol{v}) = A_{h,\tau}\boldsymbol{v},$$

where $A_{h,\tau} \in \mathbb{R}^{(N-1)\times(N-1)}$.

Remark 2.1. *Recall that for explicit Euler, we have*

$$\boldsymbol{\Phi}_{h,\tau}(t_n, \boldsymbol{v}) = (I + \tau D_h)\boldsymbol{v} + \tau \boldsymbol{g}_h(t_n).$$

Therefore,

$$\boldsymbol{\Phi}_{h,\tau}(t_n, \boldsymbol{v}) - \boldsymbol{\Phi}_{h,\tau}(t_n, \boldsymbol{w}) = (I + \tau D_h)(\boldsymbol{v} - \boldsymbol{w}) = \boldsymbol{\Phi}^0_{h,\tau}(t_n, \boldsymbol{v} - \boldsymbol{w}).$$

So, in this case, $A_{h,\tau} = (I + \tau D_h)$. Similarly, for backward/implicit Euler, we can derive

$$A_{h,\tau} = (I - \tau D_h)^{-1}.$$

Proof of Lemma 2.1. Let the Runge–Kutta method have s stages and Butcher coefficients

$$(a_{ij})_{i,j=1}^s, \qquad (b_i)_{i=1}^s, \qquad (c_i)_{i=1}^s.$$

Applied to the linear nonhomogeneous system

$$\boldsymbol{u}_h'(t) = D_h \boldsymbol{u}_h(t) + \boldsymbol{g}_h(t),$$

the internal stages corresponding to an initial value $\boldsymbol{v} \in \mathbb{R}^{N-1}$ are given by

$$\boldsymbol{Y}_i(\boldsymbol{v}) = \boldsymbol{v} + \tau \sum_{j=1}^s a_{ij} \Big(D_h \boldsymbol{Y}_j(\boldsymbol{v}) + \boldsymbol{g}_h(t_n + c_j\tau) \Big), \qquad i = 1, \dots, s,$$

and the one-step update is

$$\boldsymbol{\Phi}_{h,\tau}(t_n, \boldsymbol{v}) = \boldsymbol{v} + \tau \sum_{i=1}^s b_i \Big(D_h \boldsymbol{Y}_i(\boldsymbol{v}) + \boldsymbol{g}_h(t_n + c_i\tau) \Big).$$

Now let $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^{N-1}$, and denote by $\boldsymbol{Y}_i(\boldsymbol{v})$ and $\boldsymbol{Y}_i(\boldsymbol{w})$ the corresponding stage vectors. Define the stage differences

$$\boldsymbol{Z}_i := \boldsymbol{Y}_i(\boldsymbol{v}) - \boldsymbol{Y}_i(\boldsymbol{w}), \qquad i = 1, \dots, s.$$

Subtracting the two stage equations, we obtain

$$\begin{aligned} \boldsymbol{Z}_i &= \boldsymbol{v} - \boldsymbol{w} + \tau \sum_{j=1}^s a_{ij} \Big(D_h \boldsymbol{Y}_j(\boldsymbol{v}) + \boldsymbol{g}_h(t_n + c_j\tau) \Big) - \tau \sum_{j=1}^s a_{ij} \Big(D_h \boldsymbol{Y}_j(\boldsymbol{w}) + \boldsymbol{g}_h(t_n + c_j\tau) \Big) \\ &= \boldsymbol{v} - \boldsymbol{w} + \tau \sum_{j=1}^s a_{ij} D_h \boldsymbol{Z}_j, \qquad i = 1, \dots, s, \end{aligned}$$

since the nonhomogeneous terms cancel.

Similarly, subtracting the two update formulas gives

$$\begin{aligned}\boldsymbol{\Phi}_{h,\tau}(t_n,\boldsymbol{v})-\boldsymbol{\Phi}_{h,\tau}(t_n,\boldsymbol{w}) &= \boldsymbol{v}-\boldsymbol{w}+\tau\sum_{i=1}^{s} b_i\Big(D_h\boldsymbol{Y}_i(\boldsymbol{v})+\boldsymbol{g}_h(t_n+c_i\tau)\Big)\\ &\quad -\tau\sum_{i=1}^{s} b_i\Big(D_h\boldsymbol{Y}_i(\boldsymbol{w})+\boldsymbol{g}_h(t_n+c_i\tau)\Big)\\ &= \boldsymbol{v}-\boldsymbol{w}+\tau\sum_{i=1}^{s} b_i D_h\boldsymbol{Z}_i.\end{aligned}$$

Observe that the system satisfied by $\boldsymbol{Z}_1,\dots,\boldsymbol{Z}_s$ is exactly the stage system obtained by applying the same Runge–Kutta method to the homogeneous problem

$$\boldsymbol{u}_h'(t)=D_h\boldsymbol{u}_h(t) \tag{15}$$

with initial value $\boldsymbol{v}-\boldsymbol{w}$. Therefore,

$$\boldsymbol{\Phi}_{h,\tau}(t_n,\boldsymbol{v})-\boldsymbol{\Phi}_{h,\tau}(t_n,\boldsymbol{w})=\boldsymbol{\Phi}^0_{h,\tau}(t_n,\boldsymbol{v}-\boldsymbol{w}),$$

where $\boldsymbol{\Phi}^0_{h,\tau}$ is the one-step output of the Runge–Kutta method applied to the homogeneous problem (15) with initial data $\boldsymbol{v}-\boldsymbol{w}$.

It thus remains to show that the one-step map $\boldsymbol{\Phi}^0_{h,\tau}$ is linear. Indeed, for the homogeneous system the stage equations with initial value $\boldsymbol{v}\in\mathbb{R}^{N-1}$ take the form

$$\boldsymbol{Y}_i(\boldsymbol{v})=\boldsymbol{v}+\tau\sum_{j=1}^{s} a_{ij}D_h\boldsymbol{Y}_j(\boldsymbol{v}),\qquad i=1,\dots,s,$$

and the update formula is

$$\boldsymbol{\Phi}^0_{h,\tau}(t_n,\boldsymbol{v})=\boldsymbol{v}+\tau\sum_{i=1}^{s} b_i D_h\boldsymbol{Y}_i(\boldsymbol{v}).$$

Since these equations depend only linearly on the initial value $\boldsymbol{v}$, it follows that the resulting one-step map is linear. Hence there exists a matrix

$$A_{h,\tau}\in\mathbb{R}^{(N-1)\times(N-1)}$$

such that

$$\boldsymbol{\Phi}^0_{h,\tau}(t_n,\boldsymbol{v})=A_{h,\tau}\boldsymbol{v},\qquad \forall\,\boldsymbol{v}\in\mathbb{R}^{N-1}.$$

Consequently,

$$\boldsymbol{\Phi}_{h,\tau}(t_n,\boldsymbol{v})-\boldsymbol{\Phi}_{h,\tau}(t_n,\boldsymbol{w})=A_{h,\tau}(\boldsymbol{v}-\boldsymbol{w}).$$

By construction, $A_{h,\tau}$ is exactly the update matrix corresponding to the homogeneous problem (15) with $\boldsymbol{g}_h=0$. This completes the proof.

□

Now, by Lemma 2.1, we have

$$\boldsymbol{\Phi}_{h,\tau}(t_n, \widetilde{\boldsymbol{u}}_h(t_n)) - \boldsymbol{\Phi}_{h,\tau}(t_n, \boldsymbol{u}_h^n) = A_{h,\tau}\left(\widetilde{\boldsymbol{u}}_h(t_n) - \boldsymbol{u}_h^n\right) = A_{h,\tau}\boldsymbol{e}_h^n,$$

and hence the error iteration (14) becomes:

$$\boldsymbol{e}_h^{n+1} = A_{h,\tau}\boldsymbol{e}_h^n + \tau\boldsymbol{\xi}_h^n. \tag{16}$$

From this relation, we see that in order for the error to remain controlled as the time steps advance, repeated applications of the matrix $A_{h,\tau}$ must remain uniformly bounded. This motivates the following definition of stability.

Definition 2.2 (Stability)**.** *We say that the fully discrete scheme is* stable *on the time interval* $[0,T]$*, if there exists a constant* $C > 0$*, independent of* h*,* τ*, and* n*, such that*

$$\|(A_{h,\tau})^n\|_h \le C \qquad \text{for all } n \text{ satisfying } n\tau \le T, \tag{17}$$

where $\|\cdot\|_h$ *denotes the induced matrix norm*

$$\|A\|_h := \sup_{\boldsymbol{v}\neq 0} \frac{\|A\boldsymbol{v}\|_h}{\|\boldsymbol{v}\|_h},$$

and $A_{h,\tau}$ *is the update matrix corresponding to the homogeneous update map* $\boldsymbol{\Phi}^0_{h,\tau}(t_n, \boldsymbol{v}) = A_{h,\tau}\boldsymbol{v}$*.*

Note that for this linear problem class (9), stability is independent of the input data (initial condition u_{init}, boundary condition g_0 and g_L, or source term $f(x,t)$). The stability is completely determined by the differential operator $\mathcal{D}$, and the spatial and temporal discretization.

2.4 Convergence and Lax's equivalence

We are now ready to state the main result of this Part III. For that purpose, let us first introduce the the formal definition of convergence.

Definition 2.3 (Convergence)**.** *We say that the fully discrete scheme* (11) *is* convergent *on* $[0,T]$ *if*

$$\sup_{0\le n\tau\le T} \|\widetilde{\boldsymbol{u}}_h(t_n) - \boldsymbol{u}_h^n\|_h \to 0 \qquad \text{as } h\to 0,\ \tau\to 0.$$

Theorem 2.1 (Lax equivalence theorem)**.** *If the fully discrete scheme* (11) *is consistent and stable, then it is convergent on* $[0,T]$*. More precisely, if for the local truncation error, we have*

$$\sup_{0\le n\tau\le T} \|\boldsymbol{\xi}_h^n\|_h = \mathcal{O}(h^p + \tau^q),$$

then together with stability, it implies

$$\sup_{0\le n\tau\le T} \|\boldsymbol{e}_h^n\|_h = \mathcal{O}(h^p + \tau^q).$$

Proof. By (14) and Lemma 2.1, we obtain the error equation:

$$\boldsymbol{e}_h^{n+1} = A_{h,\tau}\boldsymbol{e}_h^n + \tau\boldsymbol{\xi}_h^n.$$

Since the numerical solution is started with the exact initial value (see (12)), we have

$$\boldsymbol{u}_h^0 = (u_{\text{init}}(x_1), ..., u_{\text{init}}(x_{N-1})) = \widetilde{\boldsymbol{u}}_h(0).$$

Therefore

$$\boldsymbol{e}_h^0 = 0.$$

By iterating the error equation, we obtain

$$\boldsymbol{e}_h^n = \tau \sum_{j=0}^{n-1} A_{h,\tau}^{n-1-j}\boldsymbol{\xi}_h^j.$$

Hence, for every n with $n\tau \le T$, by the definition of stability (Definition 2.2), we have

$$\begin{aligned}
\|\boldsymbol{e}_h^n\|_h &\le \tau \sum_{j=0}^{n-1} \|A_{h,\tau}^{n-1-j}\|_h \|\boldsymbol{\xi}_h^j\|_h \\
&\le C\tau \sum_{j=0}^{n-1} \|\boldsymbol{\xi}_h^j\|_h \\
&\le C\, n\tau \sup_{0\le j\tau\le T} \|\boldsymbol{\xi}_h^j\|_h \\
&\le C\, T \sup_{0\le j\tau\le T} \|\boldsymbol{\xi}_h^j\|_h.
\end{aligned}$$

Therefore,

$$\sup_{0\le n\tau\le T} \|\boldsymbol{e}_h^n\|_h \le CT \sup_{0\le j\tau\le T} \|\boldsymbol{\xi}_h^j\|_h.$$

Now, by the definition of consistency (Definition 2.1), the right-hand side term tends to zero as $h, \tau \to 0$, and hence the scheme is convergent.

Moreover, by the above inequality, if

$$\sup_{0\le j\tau\le T} \|\boldsymbol{\xi}_h^j\|_h = \mathcal{O}(h^p + \tau^q),$$

then

$$\sup_{0\le n\tau\le T} \|\boldsymbol{e}_h^n\|_h = \mathcal{O}(h^p + \tau^q).$$

□

Remark 2.2. *The original Lax equivalence theorem also proves the converse direction: under the assumption of consistency, convergence implies stability. However, this direction is used less frequently in practice, since the verification of stability is more straightforward, as it depends only on the structure of the numerical method. Thus, the implication*

$$\textit{stability} + \textit{consistency} \;\Rightarrow\; \textit{convergence}$$

is therefore the direction most commonly applied in numerical analysis.

Optional reading: heuristic for the converse direction. The argument below provides some intuition for why convergence implies stability under suitable assumptions. It is not intended as a complete proof of the converse direction of the Lax equivalence theorem, but it illustrates the mechanism behind the result.

Roughly speaking, if a scheme is not stable, then one can construct a sequence of initial data for which the numerical solution stays away from zero, while the exact solution goes to zero, and therefore convergence fails.

Assume that the exact solution operator of the homogeneous semi-discrete problem satisfies the bound

$$\|e^{tD_h}\|_h \le M_T \qquad \text{for } 0 \le t \le T,$$

uniformly with respect to h.

We prove the contrapositive. Assume that the scheme is *not* stable on $[0,T]$. Then there exists a sequence (h_k, τ_k, n_k) with

$$n_k \tau_k \le T$$

such that

$$\|A_{h_k,\tau_k}^{n_k}\|_h \to \infty \qquad \text{as } k \to \infty.$$

For each k, by the definition of the operator norm, there exists $\boldsymbol{v}_k \in \mathbb{R}^{N-1}$ with

$$\|\boldsymbol{v}_k\|_h = 1$$

such that

$$\|A_{h_k,\tau_k}^{n_k} \boldsymbol{v}_k\|_h \ge \frac{1}{2}\|A_{h_k,\tau_k}^{n_k}\|_h.$$

Define

$$\boldsymbol{w}_k := \frac{\boldsymbol{v}_k}{\|A_{h_k,\tau_k}^{n_k}\|_h}.$$

Then

$$\|\boldsymbol{w}_k\|_h \to 0,$$

while

$$\|A_{h_k,\tau_k}^{n_k} \boldsymbol{w}_k\|_h = \frac{\|A_{h_k,\tau_k}^{n_k} \boldsymbol{v}_k\|_h}{\|A_{h_k,\tau_k}^{n_k}\|_h} \ge \frac{1}{2}.$$

Now consider the homogeneous semi-discrete problem

$$\boldsymbol{z}_h'(t) = D_h \boldsymbol{z}_h(t), \qquad \boldsymbol{z}_h(0) = \boldsymbol{w}_k.$$

Its exact solution is

$$\boldsymbol{z}_h(t) = e^{tD_h} \boldsymbol{w}_k.$$

Since $\|\boldsymbol{w}_k\|_h \to 0$, the corresponding exact solutions converge uniformly to the zero solution on $[0,T]$. In particular,

$$\|\boldsymbol{z}_h(t_{n_k})\|_h \le M_T \|\boldsymbol{w}_k\|_h \to 0.$$

On the other hand, the corresponding numerical solution is

$$\boldsymbol{z}_h^{n_k} = A_{h_k,\tau_k}^{n_k} \boldsymbol{w}_k,$$

and therefore

$$\|\boldsymbol{z}_h^{n_k}\|_h \geq \frac{1}{2}.$$

Hence

$$\begin{aligned}\|\boldsymbol{z}_h(t_{n_k}) - \boldsymbol{z}_h^{n_k}\|_h &\geq \|\boldsymbol{z}_h^{n_k}\|_h - \|\boldsymbol{z}_h(t_{n_k})\|_h \\ &\geq \frac{1}{2} - M_T\|\boldsymbol{w}_k\|_h.\end{aligned}$$

For all sufficiently large k, the right-hand side is bounded below by $1/4$. Therefore the numerical error does not tend to zero.

Thus we have constructed a sequence of grid parameters and initial data for which the numerical solution does not converge to the exact solution.

3 Stability analysis

The practicality of Lax's equivalence theorem can be understood as follows. The convergence of a numerical method is typically difficult to analyze directly, since it depends on both the PDE and the numerical discretization, as well as their interaction. Lax's result decomposes this complicated task into two mutually independent and easier-to-verify components: consistency and stability.

Consistency is a completely local property. It measures how well the numerical discretization approximates the original differential equation. We have already seen many examples of consistency analysis. A key technique is Taylor expansion, followed by estimating the remainder term and examining how it depends on the mesh size (whether spatial or temporal).

On the other hand, for the linear PDE setting considered here, stability is determined entirely by the homogeneous update operator associated with the differential operator and its discretization, and therefore does not depend on the data term.

In this section, we investigate stability in more detail. We examine the stability of several different combinations of spatial and temporal discretizations. But keep in mind that, throughout, our goal is simply to control $\|(A_{h,\tau})^n\|_h$, where $A_{h,\tau}$ is the update matrix corresponding to the homogeneous problem.

The first thing to notice is that the $\|\cdot\|_h$ is just the induced matrix-2 norm, since

$$\|A\|_h := \sup_{\boldsymbol{v}\neq 0} \frac{\|A\boldsymbol{v}\|_h}{\|\boldsymbol{v}\|_h} = \sup_{\boldsymbol{v}\neq 0} \frac{\sqrt{h}\|A\boldsymbol{v}\|_2}{\sqrt{h}\|\boldsymbol{v}\|_2} = \|A\|_2.$$

3.1 Stability analysis through eigenvalue techniques

Let us first recall a homework problem we have done. Consider the linear ODE system

$$\boldsymbol{y}' = A\boldsymbol{y}, \qquad \boldsymbol{y}(0) = \boldsymbol{y}_0. \tag{18}$$

Lemma 3.1. *Suppose that $A \in \mathbb{R}^{d\times d}$ is diagonalizable with eigenpairs $A\boldsymbol{v}_j = \lambda_j\boldsymbol{v}_j$ for $j = 1, 2, \ldots, d$, and that $\{\boldsymbol{v}_j\}_{j=1}^d$ forms an orthonormal basis of $\mathbb{R}^d$.*

Suppose that we use a Runge–Kutta method with stepsize τ to solve the equation (18). Then

$$\boldsymbol{y}_n = \sum_{j=1}^{d} c_j \, R(\tau\lambda_j)^n \, \boldsymbol{v}_j, \qquad \text{where} \quad c_j = \boldsymbol{v}_j \cdot \boldsymbol{y}_0,$$

and $R(z)$ is the stability function of the Runge–Kutta method.

With the above lemma, we can determine whether the fully discrete scheme (11) is stable by examining only the matrix D_h in the semi-discrete system

$$\boldsymbol{u}_h'(t) = D_h \boldsymbol{u}_h(t) + \boldsymbol{g}_h(t).$$

Proposition 3.1. *Suppose that $D_h \in \mathbb{R}^{(N-1)\times(N-1)}$ is diagonalizable with eigenpairs*

$$D_h \boldsymbol{v}_j = \lambda_j \boldsymbol{v}_j, \qquad j = 1, 2, \ldots, N-1,$$

and that $\{\boldsymbol{v}_j\}_{j=1}^{N-1}$ forms an orthonormal basis of $\mathbb{R}^{N-1}$. Assume that the stability function of the Runge–Kutta method satisfies

$$|R(\tau\lambda_j)| \le 1, \qquad j = 1, 2, \ldots, N-1.$$

Then the fully discrete scheme (11) is stable.

Proof. By Definition 2.2, we only need to verify that the update matrix $A_{h,\tau}$ corresponding to the homogeneous scheme satisfies

$$\|(A_{h,\tau})^n\|_h \le C \qquad \text{for all } n\tau \le T,$$

with a constant C independent of h, τ, and n.

Consider the homogeneous semi-discrete system

$$\boldsymbol{u}_h'(t) = D_h \boldsymbol{u}_h(t).$$

Applying the Runge–Kutta method with stepsize τ, the fully discrete solution satisfies

$$\boldsymbol{u}_h^n = A_{h,\tau}^n \boldsymbol{u}_h^0.$$

Now apply Lemma 3.1 with $A = D_h$ and $\boldsymbol{y} = \boldsymbol{u}_h$. Since $\{\boldsymbol{v}_j\}_{j=1}^{N-1}$ forms an orthonormal basis of $\mathbb{R}^{N-1}$, we may write

$$\boldsymbol{u}_h^0 = \sum_{j=1}^{N-1} c_j \boldsymbol{v}_j, \qquad c_j = \boldsymbol{v}_j \cdot \boldsymbol{u}_h^0.$$

Then the fully discrete solution satisfies

$$\boldsymbol{u}_h^n = \sum_{j=1}^{N-1} c_j \, R(\tau\lambda_j)^n \, \boldsymbol{v}_j.$$

Using the orthonormality of the eigenvectors, we obtain

$$\|\boldsymbol{u}_h^n\|_2^2 = \sum_{j=1}^{N-1} |c_j|^2 \, |R(\tau\lambda_j)|^{2n}.$$

Since by assumption

$$|R(\tau\lambda_j)| \le 1 \qquad \text{for all } j = 1, 2, \ldots, N-1,$$

it follows that

$$\|\boldsymbol{u}_h^n\|_2^2 \le \sum_{j=1}^{N-1} |c_j|^2 = \|\boldsymbol{u}_h^0\|_2^2.$$

Therefore,

$$\|\boldsymbol{u}_h^n\|_2 \le \|\boldsymbol{u}_h^0\|_2 \qquad \text{for all } n\tau \le T.$$

Since $\boldsymbol{u}_h^n = A_{h,\tau}^n \boldsymbol{u}_h^0$, this implies

$$\|A_{h,\tau}^n \boldsymbol{u}_h^0\|_2 \le \|\boldsymbol{u}_h^0\|_2 \qquad \text{for all } \boldsymbol{u}_h^0 \in \mathbb{R}^{N-1}.$$

Hence,

$$\|A_{h,\tau}^n\|_2 \le 1 \qquad \text{for all } n\tau \le T.$$

Taking $C = 1$, we conclude that

$$\|A_{h,\tau}^n\|_2 \le C \qquad \text{for all } n\tau \le T,$$

which proves the stability of the fully discrete scheme. □

Remark 3.1. *If A is a real symmetric matrix, then it admits an orthonormal basis of eigenvectors and hence is diagonalizable with real eigenvalues.*

More generally, a complex square matrix A admits an orthonormal basis of eigenvectors (i.e., it is unitarily diagonalizable) if and only if it is normal, that is,

$$A^*A = AA^*,$$

where A^* is the conjugate transpose of A.

Examples. Consider first the case where D_h is the standard second-order central difference discretization matrix approximating the second derivative operator ∂_{xx} for the diffusion equation with homogeneous Dirichlet boundary conditions. Thus,

$$D_h := -\frac{1}{h^2} \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}.$$

This matrix is real symmetric, so it admits an orthonormal basis of eigenvectors. Its eigenvalues are

$$\lambda_j = -\frac{4}{h^2}\sin^2\Big(\frac{j\pi}{2N}\Big), \qquad j = 1, \dots, N-1.$$

Now suppose that we use the explicit Euler method in time. Its stability function is

$$R(z) = 1 + z.$$

Therefore, by Proposition 3.1, stability holds provided

$$|1 + \tau\lambda_j| \le 1, \qquad j = 1, 2, \dots, N-1.$$

Since each $\lambda_j \le 0$, this is equivalent to

$$-2 \le \tau\lambda_j \le 0, \qquad j = 1, 2, \dots, N-1.$$

Hence we need

$$\tau|\lambda_j| \le 2, \qquad j = 1, 2, \dots, N-1.$$

Substituting the eigenvalue formula gives

$$\frac{\tau}{h^2} \le \frac{1}{2\sin^2\big(\frac{j\pi}{2N}\big)}, \qquad j = 1, 2, \dots, N-1.$$

The most restrictive condition occurs for the largest eigenvalue magnitude, namely $j = N-1$. Therefore,

$$\frac{\tau}{h^2} \le \frac{1}{2\sin^2\big(\frac{(N-1)\pi}{2N}\big)} = \frac{1}{2\cos^2\big(\frac{\pi}{2N}\big)}.$$

In particular, since $\cos^2\big(\frac{\pi}{2N}\big) \le 1$, a sufficient condition is

$$\frac{\tau}{h^2} \le \frac{1}{2}.$$

This recovers the condition for the convergence theorem (Theorem 1.1) for the diffusion equation that we have proved.

Next, suppose that we use a second-order explicit Runge–Kutta method. Its stability function is

$$R(z) = 1 + z + \frac{z^2}{2}.$$

Therefore, by Proposition 3.1, stability holds provided

$$\left|1 + \tau\lambda_j + \frac{(\tau\lambda_j)^2}{2}\right| \le 1, \qquad j = 1, 2, \dots, N-1.$$

Since $\lambda_j \le 0$, let

$$x := \tau\lambda_j \le 0.$$

Then we need

$$\left|1 + x + \frac{x^2}{2}\right| \le 1.$$

Now for real x, the quantity $1 + x + \frac{x^2}{2}$ is also real, so the above is equivalent to

$$-1 \le 1 + x + \frac{x^2}{2} \le 1.$$

The left inequality always holds, because

$$1 + x + \frac{x^2}{2} = \frac{1}{2}\big((x+1)^2 + 1\big) \ge \frac{1}{2}.$$

Thus we only need

$$1 + x + \frac{x^2}{2} \le 1,$$

that is,

$$x + \frac{x^2}{2} \le 0.$$

Factoring gives

$$x\left(1 + \frac{x}{2}\right) \le 0.$$

Since $x \le 0$, this holds exactly when

$$-2 \le x \le 0.$$

Therefore, the stability condition is again

$$-2 \le \tau\lambda_j \le 0, \qquad j = 1, 2, \ldots, N-1.$$

Proceeding exactly as in the explicit Euler case, we obtain

$$\frac{\tau}{h^2} \le \frac{1}{2\sin^2\!\big(\frac{(N-1)\pi}{2N}\big)} = \frac{1}{2\cos^2\!\big(\frac{\pi}{2N}\big)}.$$

In particular, a sufficient condition is

$$\frac{\tau}{h^2} \le \frac{1}{2}.$$

So for this diffusion problem, the second-order explicit Runge–Kutta method has the same stability restriction as the explicit Euler method.

Finally, let us briefly consider two implicit methods. For the implicit Euler method, the stability function is

$$R(z) = \frac{1}{1-z}.$$

For the trapezoidal rule, the stability function is

$$R(z) = \frac{1 + z/2}{1 - z/2}.$$

Since all eigenvalues λ_j of D_h are real and negative, we have

$$\mathrm{Re}(\tau\lambda_j) \le 0 \qquad \text{for all } j = 1, 2, \ldots, N-1.$$

Now implicit Euler and the trapezoidal rule are both A-stable, that is, their stability functions satisfy

$$|R(z)| \leq 1 \qquad \text{whenever } \operatorname{Re}(z) \leq 0.$$

Therefore,

$$|R(\tau\lambda_j)| \leq 1 \qquad \text{for all } j = 1, 2, \ldots, N-1$$

for any choice of $\tau > 0$. By Proposition 3.1, both implicit Euler and the trapezoidal rule are stable for any time step size. In other words, they are unconditionally stable for this diffusion problem.

Example. In this example, we implement a fully discrete scheme with central difference for space discretization and trapezoidal method for time discretization.

```
clear; clc; close all;

% =========================
% Plot appearance settings
% =========================
fs_axis  = 18;
fs_title = 20;
fs_tick  = 16;
fs_leg   = 16;
lw       = 2.5;

% Spatial grid
L = 1;
N = 100;
h = L/N;
x = linspace(0,L,N+1)';

% Initial condition
sigma = 0.08;
u0 = exp(-((x-0.5).^2)/(2*sigma^2));
u0([1,end]) = 0;

% Final time
T = 0.01;

% Interior unknowns and discrete Laplacian
Ni = N-1;
e  = ones(Ni,1);
A  = spdiags([e -2*e e], -1:1, Ni, Ni);
I  = speye(Ni);

% ====================================
% Figure 1: trapezoidal rule, large nu
% ====================================
nu_list = [10, 20, 50];
numCases = length(nu_list);

U_trap = zeros(N+1,numCases);
Tactual = zeros(numCases,1);
```

```
for k = 1:numCases
    nu  = nu_list(k);
    tau = nu*h^2;
    M   = max(1,floor(T/tau));

    u = u0(2:N);
    LHS = I - 0.5*nu*A;
    RHS = I + 0.5*nu*A;

    for n = 1:M
        u = LHS \ (RHS*u);
    end

    U_trap(2:N,k) = u;
    Tactual(k) = M*tau;
end

figure('Position',[100,100,900,600]);
plot(x,u0,'k--','LineWidth',lw); hold on;

for k = 1:numCases
    plot(x,U_trap(:,k),'LineWidth',lw);
end

grid on
xlabel('x','FontSize',fs_axis)
ylabel('u(x,t)','FontSize',fs_axis)
title('Heat equation (Trapezoidal rule with large nu)', ...
      'FontSize',fs_title)
set(gca,'FontSize',fs_tick)

legendEntries = cell(numCases+1,1);
legendEntries{1} = 'Initial data, t=0';
for k = 1:numCases
    legendEntries{k+1} = sprintf('nu=%g, t=%.4f',nu_list(k),Tactual(k));
end
legend(legendEntries,'FontSize',fs_leg,'Location','best')

% ============================================
% Figure 2: implicit Euler vs trapezoidal
% ============================================
nu  = 50;
tau = nu*h^2;
M   = max(1,floor(T/tau));
Tf  = M*tau;

u_impl = u0(2:N);
u_trap = u0(2:N);

LHS_impl = I - nu*A;
LHS_trap = I - 0.5*nu*A;
RHS_trap = I + 0.5*nu*A;
```

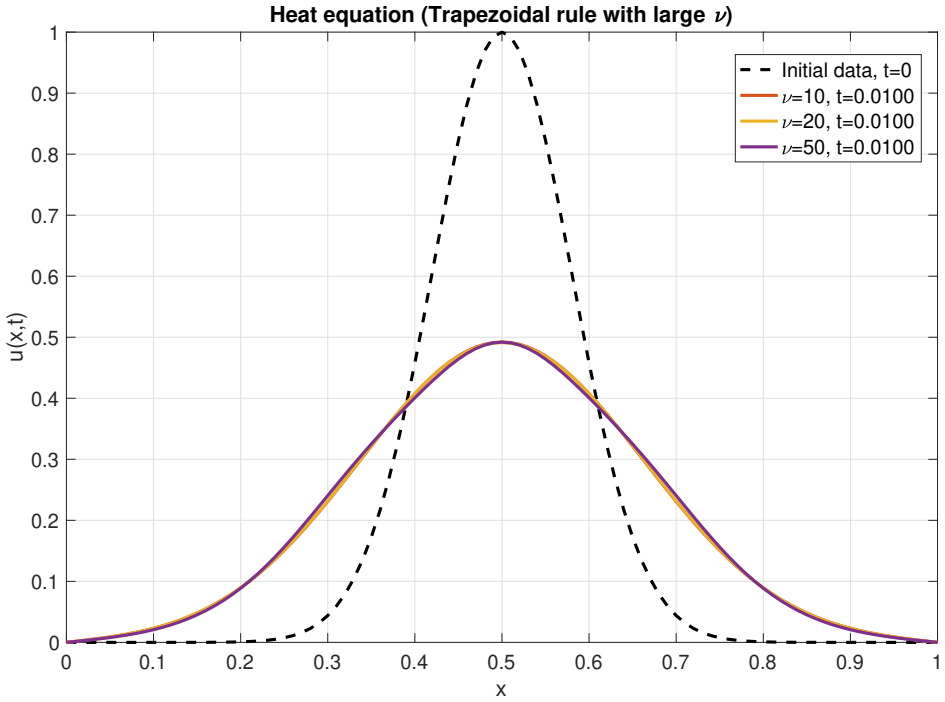
```
for n = 1:M
    u_impl = LHS_impl \ u_impl;
    u_trap = LHS_trap \ (RHS_trap*u_trap);
end

U_impl = zeros(N+1,1);
U_trap50 = zeros(N+1,1);
U_impl(2:N) = u_impl;
U_trap50(2:N) = u_trap;

figure('Position',[150,150,900,600]);
plot(x,u0,'k--','LineWidth',lw); hold on;
plot(x,U_impl,'LineWidth',lw);
plot(x,U_trap50,'LineWidth',lw);

grid on
xlabel('x','FontSize',fs_axis)
ylabel('u(x,t)','FontSize',fs_axis)
title(sprintf('Implicit Euler vs Trapezoidal method, nu=%g, t=%.4f', ...
      nu,Tf),'FontSize',fs_title)
set(gca,'FontSize',fs_tick)

legend({'Initial data, t=0','Implicit Euler','Trapezoidal'}, ...
       'FontSize',fs_leg,'Location','best')
```



The above figure shows that the method remains stable even when

$$\mu = \frac{\tau}{h^2} = 50.$$

Recall that for the explicit Euler method, the scheme becomes unstable when $\mu = 0.6$. By using the implicit trapezoidal method instead, we can therefore take much larger time steps τ without encountering stability issues.

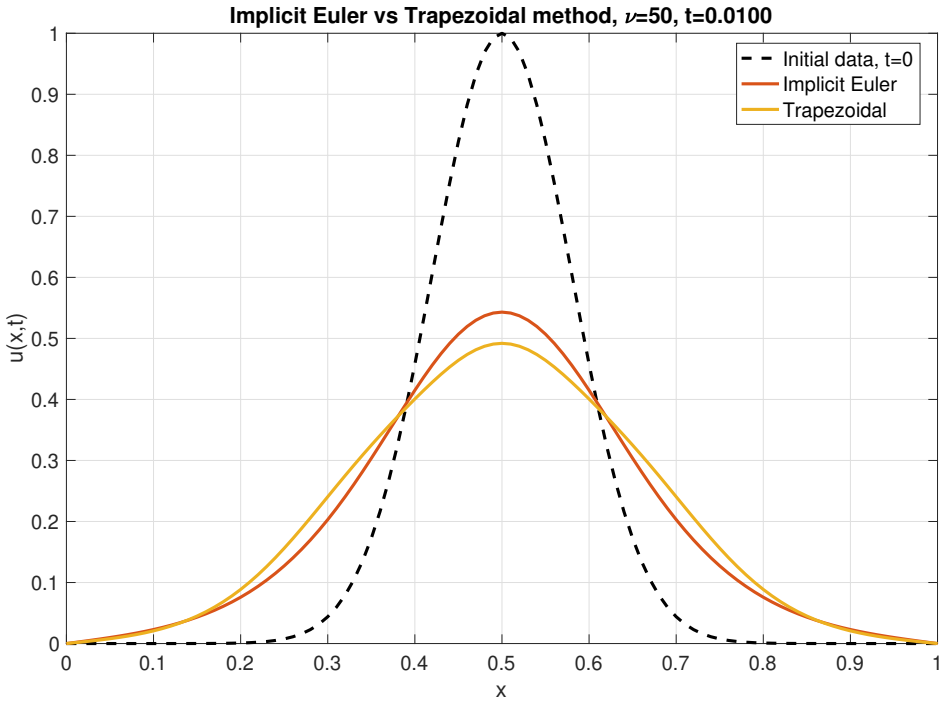
We also observe that the method produces a good approximation even with such a large time step. For instance, when $\mu = 50$, we have

$$\tau = 50h^2 = 50 \times 0.0001 = 0.005.$$

Thus, only two time steps are needed to reach the final time, yet the numerical solution is still very accurate. This example illustrates that, in this setting, temporal accuracy is not the main limitation. Rather, stability is the primary concern, and it is effectively resolved by switching to the trapezoidal method.

In this case, by using the trapezoidal method, we only need 2 time steps to obtain a good approximation. In contrast, the explicit Euler method requires about 200 time steps. As a result, one can expect a significantly faster solver.

We also compare the trapezoidal method with the implicit Euler method for $\mu = 50$, so that only two time steps are taken. In this case, both methods remain stable. However, the higher-order accuracy of the trapezoidal method clearly pays off: it produces a noticeably more accurate solution than the implicit Euler method.



3.2 Stability analysis through Fourier techniques

We now study stability using another technique. Again, recall that the key to stability is to ensure that the homogeneous update matrix $A_{h,\tau}^n$ remains uniformly bounded:

$$\|A_{h,\tau}^n\|_2 \leq C.$$

A sufficient condition for this is that the one-step update matrix is non-expansive:

$$\|A_{h,\tau}\boldsymbol{v}\|_2 \le \|\boldsymbol{v}\|_2 \qquad \text{for all } \boldsymbol{v} \in \mathbb{C}^N,$$

which implies

$$\|A_{h,\tau}^n\|_2 \le 1 \qquad \text{for all } n \ge 1.$$

Extending the setting to complex vectors $\mathbb{C}^N$ allows us to apply Fourier techniques more conveniently.

Also, to apply Fourier techniques, we now restrict our attention to the case of a uniform grid with periodic boundary conditions. More precisely, let

$$x_j = jh, \qquad j = 0, 1, \ldots, N-1,$$

and assume that the discrete solution satisfies

$$v_{j+N} = v_j.$$

In this setting, the boundary conditions no longer create additional complications, and the finite difference operators we consider take the same form at every grid point.

For example, the standard second-order central difference approximation of the second derivative takes the form

$$(D_h\boldsymbol{v})_j = \frac{v_{j-1} - 2v_j + v_{j+1}}{h^2}, \qquad j = 0, \ldots, N-1,$$

where the indices are understood periodically. The corresponding matrix is

$$D_h = \frac{1}{h^2}\begin{pmatrix} -2 & 1 & & & 1 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ 1 & & & 1 & -2 \end{pmatrix}.$$

Similarly, the fourth-order five-point central difference approximation takes the form

$$(D_h\boldsymbol{v})_j = \frac{-v_{j-2} + 16v_{j-1} - 30v_j + 16v_{j+1} - v_{j+2}}{12h^2}, \qquad j = 0, \ldots, N-1,$$

again with periodic indexing. The corresponding matrix is

$$D_h = \frac{1}{12h^2}\begin{pmatrix} -30 & 16 & -1 & & & -1 & 16 \\ 16 & -30 & 16 & -1 & & & -1 \\ -1 & 16 & -30 & 16 & -1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & -1 & 16 & -30 & 16 & -1 \\ -1 & & & -1 & 16 & -30 & 16 \\ 16 & -1 & & & -1 & 16 & -30 \end{pmatrix}.$$

Matrices of this type are called periodic (or circulant) matrices. Although such matrices arise naturally only under periodic boundary conditions, the corresponding stability analysis typically extends well to schemes with non-periodic boundary conditions. The reason is that the spectrum of the periodic matrix is usually very similar to that of the corresponding matrix with, for example, homogeneous boundary conditions.

Lemma 3.2 (Discrete Fourier inversion and Parseval identity)**.** *For any vector*

$$\boldsymbol{v} = (v_0, \dots, v_{N-1})^\top \in \mathbb{C}^N,$$

define its discrete Fourier coefficients by

$$\hat{v}_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} v_j e^{-ij\theta_k}, \qquad \theta_k = \frac{2\pi k}{N}, \qquad k = 0, 1, \dots, N-1.$$

Then the inverse formula is given by

$$v_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \hat{v}_k e^{ij\theta_k}, \qquad j = 0, 1, \dots, N-1. \tag{19}$$

Moreover, the discrete Fourier transform is unitary on $\mathbb{C}^N$:

$$\|\boldsymbol{v}\|_2^2 = \sum_{j=0}^{N-1} |v_j|^2 = \sum_{k=0}^{N-1} |\hat{v}_k|^2. \tag{20}$$

Proof. We first prove the inverse formula (19). By the definition of $\hat{v}_k$,

$$\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \hat{v}_k e^{ij\theta_k} = \frac{1}{N} \sum_{k=0}^{N-1} \sum_{\ell=0}^{N-1} v_\ell e^{-i\ell\theta_k} e^{ij\theta_k} = \frac{1}{N} \sum_{\ell=0}^{N-1} v_\ell \sum_{k=0}^{N-1} e^{i(j-\ell)\theta_k}.$$

Since $\theta_k = 2\pi k/N$, we have

$$\sum_{k=0}^{N-1} e^{i(j-\ell)\theta_k} = \sum_{k=0}^{N-1} \left(e^{2\pi i(j-\ell)/N}\right)^k.$$

If $j = \ell$, then each term equals 1, and hence

$$\sum_{k=0}^{N-1} e^{i(j-\ell)\theta_k} = N.$$

If $j \neq \ell$, then $e^{2\pi i(j-\ell)/N} \neq 1$, so this is a geometric series with sum

$$\sum_{k=0}^{N-1} \left(e^{2\pi i(j-\ell)/N}\right)^k = \frac{1 - e^{2\pi i(j-\ell)}}{1 - e^{2\pi i(j-\ell)/N}} = 0.$$

Therefore,

$$\sum_{k=0}^{N-1} e^{i(j-\ell)\theta_k} = N\delta_{j\ell},$$

where $\delta_{j\ell}$ is the Kronecker delta. Substituting this into the previous expression gives

$$\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \hat{v}_k e^{ij\theta_k} = \frac{1}{N} \sum_{\ell=0}^{N-1} v_\ell \, N\delta_{j\ell} = v_j.$$

This proves the inversion formula.

Next we prove the identity (20). Using the inversion formula,

$$v_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \hat{v}_k e^{ij\theta_k},$$

we compute

$$\sum_{j=0}^{N-1} |v_j|^2 = \sum_{j=0}^{N-1} \left(\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \hat{v}_k e^{ij\theta_k} \right) \overline{\left(\frac{1}{\sqrt{N}} \sum_{\ell=0}^{N-1} \hat{v}_\ell e^{ij\theta_\ell} \right)}.$$

Thus

$$\sum_{j=0}^{N-1} |v_j|^2 = \frac{1}{N} \sum_{k,\ell=0}^{N-1} \hat{v}_k \overline{\hat{v}_\ell} \sum_{j=0}^{N-1} e^{ij(\theta_k - \theta_\ell)}.$$

Again, by the same orthogonality relation,

$$\sum_{j=0}^{N-1} e^{ij(\theta_k - \theta_\ell)} = \sum_{j=0}^{N-1} e^{i\theta_j (k-\ell)} = N\delta_{k\ell}.$$

Hence

$$\sum_{j=0}^{N-1} |v_j|^2 = \frac{1}{N} \sum_{k,\ell=0}^{N-1} \hat{v}_k \overline{\hat{v}_\ell} \, N\delta_{k\ell} = \sum_{k=0}^{N-1} |\hat{v}_k|^2.$$

Therefore,

$$\|\boldsymbol{v}\|_2^2 = \sum_{j=0}^{N-1} |v_j|^2 = \sum_{k=0}^{N-1} |\hat{v}_k|^2.$$

This completes the proof. □

Remark 3.2. *For each $k = 0, \ldots, N-1$, let us define*

$$\theta_k = \frac{2\pi k}{N}, \qquad \boldsymbol{e}_k := (e^{i0\theta_k}, e^{i1\theta_k}, \ldots, e^{i(N-1)\theta_k})^\top.$$

Then, Lemma 3.2 can be rewritten in the following vector form:

$$\text{(project } \boldsymbol{v} \text{ in } \boldsymbol{e}_k \text{ to obtain Fourier coefficients)} \qquad \hat{v}_k = \frac{1}{\sqrt{N}}\langle \boldsymbol{v}, \boldsymbol{e}_k \rangle = \frac{1}{\sqrt{N}} \boldsymbol{v} \cdot \overline{\boldsymbol{e}}_k \tag{21a}$$

$$\text{(recover } \boldsymbol{v} \text{ with Fourier coefficients)} \qquad \boldsymbol{v} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \hat{v}_k \boldsymbol{e}_k \tag{21b}$$

$$\text{(Parseval identity)} \qquad \|\boldsymbol{v}\|_2^2 = \sum_{k=0}^{N-1} |\hat{v}_k|^2. \tag{21c}$$

Also, we have

$$\langle \boldsymbol{e}_k, \boldsymbol{e}_l \rangle = \boldsymbol{e}_k \cdot \overline{\boldsymbol{e}}_l = \sum_{j=0}^{N-1} e^{ij(\theta_k - \theta_l)} = N\delta_{kl}.$$

So, $\{\frac{1}{\sqrt{N}}\boldsymbol{e}_l\}_{l=0}^{N-1}$ are orthonormal basis in $\mathbb{C}^N$.

Lemma 3.3. *Let $A \in \mathbb{C}^{N\times N}$ be a circulant matrix, that is,*

$$A = \begin{pmatrix} a_0 & a_1 & a_2 & \cdots & a_{N-1} \\ a_{N-1} & a_0 & a_1 & \cdots & a_{N-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & a_3 & \cdots & a_0 \end{pmatrix}.$$

Then $\boldsymbol{e}_k$ is an eigenvector of A. More precisely,

$$A\boldsymbol{e}_k = \lambda_k \boldsymbol{e}_k,$$

where

$$\lambda_k = \sum_{m=0}^{N-1} a_m e^{im\theta_k} =: g(\theta_k).$$

Proof. Fix $k \in \{0, \dots, N-1\}$. Since A is circulant, its (j, ℓ) entry depends only on $\ell - j$ modulo N. More precisely,

$$A_{j\ell} = a_{\ell - j \,(\mathrm{mod}\, N)}.$$

Hence the jth component of $A\boldsymbol{e}_k$ is

$$(A\boldsymbol{e}_k)_j = \sum_{\ell=0}^{N-1} A_{j\ell} e^{i\ell\theta_k} = \sum_{\ell=0}^{N-1} a_{\ell - j \,(\mathrm{mod}\, N)} e^{i\ell\theta_k}.$$

Let

$$m = \ell - j \pmod{N}.$$

Then $\ell \equiv j + m \pmod{N}$, and as ℓ runs through $0, \dots, N-1$, so does m. Therefore,

$$(A\boldsymbol{e}_k)_j = \sum_{m=0}^{N-1} a_m e^{i(j+m)\theta_k} = e^{ij\theta_k} \sum_{m=0}^{N-1} a_m e^{im\theta_k}.$$

Since the quantity

$$\lambda_k := \sum_{m=0}^{N-1} a_m e^{im\theta_k}$$

is independent of j, we obtain

$$(A\boldsymbol{e}_k)_j = \lambda_k e^{ij\theta_k}.$$

Thus

$$A\boldsymbol{e}_k = \lambda_k \boldsymbol{e}_k,$$

which proves that $\boldsymbol{e}_k$ is an eigenvector of A. □

Proposition 3.2. *Assume that the one-step update matrix $A_{h,\tau} \in \mathbb{C}^{N\times N}$ is circulant. Then, there exists $g(\theta_k) \in \mathbb{C}$ such that*

$$A_{h,\tau}\boldsymbol{e}_k = g(\theta_k)\boldsymbol{e}_k.$$

If

$$|g(\theta_k)| \le 1 \qquad \text{for all } k = 0, \dots, N-1,$$

then

$$\|A_{h,\tau}\boldsymbol{v}\|_2 \le \|\boldsymbol{v}\|_2 \qquad \text{for all } \boldsymbol{v} \in \mathbb{C}^N.$$

Consequently, the scheme is stable.

Proof. Consider any $\boldsymbol{v} \in \mathbb{C}^N$:

$$\boldsymbol{v} = (v_0, v_1, v_2, ..., v_{N-1}).$$

By (21), we have

$$A_{h,\tau}\boldsymbol{v} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \hat{v}_k A_{h,\tau}\boldsymbol{e}_k.$$

Since $A_{h,\tau}$ is circulant. By Lemma 3.3, we have $A_{h,\tau}\boldsymbol{e}_k = g(\theta_k)\boldsymbol{e}_k$. Therefore,

$$A_{h,\tau}\boldsymbol{v} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \hat{v}_k g(\theta_k)\boldsymbol{e}_k.$$

By (21) again, we have

$$\|A_{h,\tau}\boldsymbol{v}\|_2^2 = \sum_{k=0}^{N-1} |\hat{v}_k g(\theta_k)|^2 \le \sum_{k=0}^{N-1} |\hat{v}_k|^2 = \|\mathbf{v}\|_2^2.$$

This completes the proof. □

Remark 3.3. *The above proposition suggests that, in order to determine whether a given scheme is stable, it suffices to compute the eigenvalues $g(\theta_k)$ of the update matrix $A_{h,\tau}$ and verify that*

$$|g(\theta_k)| \le 1 \qquad \text{for all } k = 0, \dots, N-1.$$

We now illustrate how to apply this proposition in the following examples.

Example I – Explicit Euler with central difference. Consider the scheme

$$u_j^{n+1} = u_j^n + \frac{\tau}{h^2}\left(u_{j-1}^n - 2u_j^n + u_{j+1}^n\right),$$

with periodic boundary conditions.

Let us compute $g(\theta_k)$. Let $\boldsymbol{u}^n = \boldsymbol{e}_k$. Then, $\boldsymbol{u}^{n+1} = A_{h,\tau}\boldsymbol{u}^n = A_{h,\tau}\boldsymbol{e}_k$, and $u_j^n = (\boldsymbol{e}_k)_j = e^{ij\theta_k}$. Therefore,

$$\begin{aligned}(A_{h,\tau}\boldsymbol{e}_k)_j = u_j^{n+1} &= e^{ij\theta_k} + \frac{\tau}{h^2}\left(e^{i(j-1)\theta_k} - 2e^{ij\theta_k} + e^{i(j+1)\theta_k}\right)\\ &= \left(1 + \frac{\tau}{h^2}(e^{-i\theta_k} - 2 + e^{i\theta_k})\right)e^{ij\theta_k} = \left(1 + \frac{\tau}{h^2}(e^{-i\theta_k} - 2 + e^{i\theta_k})\right)(\boldsymbol{e}_k)_j.\end{aligned}$$

Thus,

$$g(\theta_k) = 1 + \frac{\tau}{h^2}(e^{-i\theta_k} - 2 + e^{i\theta_k}) = 1 + \frac{\tau}{h^2}(-2 + 2\cos(\theta_k)) = 1 - \frac{4\tau}{h^2}\sin^2\Big(\frac{\theta_k}{2}\Big).$$

By Proposition 3.2, the scheme is stable provided that

$$|g(\theta_k)| \le 1 \qquad \text{for all } k = 0, \dots, N-1.$$

This is equivalent to

$$-1 \le 1 - \frac{4\tau}{h^2}\sin^2\Big(\frac{\theta_k}{2}\Big) \le 1.$$

Therefore, we need

$$\frac{\tau}{h^2} \le \frac{1}{2\sin^2(\theta_k/2)},$$

for all $k = 0, 1, ..., N-1$. Therefore, we need

$$\frac{\tau}{h^2} \le \frac{1}{2}.$$

Note that this condition is slightly more stringent than the condition with Dirichlet boundary condition.

Example II – Explicit Euler with five-point central difference. Consider the scheme

$$u_j^{n+1} = u_j^n + \frac{\tau}{12h^2}\left(-u_{j-2}^n + 16u_{j-1}^n - 30u_j^n + 16u_{j+1}^n - u_{j+2}^n\right),$$

with periodic boundary conditions.

Let us compute $g(\theta_k)$. Let $\boldsymbol{u}^n = \boldsymbol{e}_k$. Then, $\boldsymbol{u}^{n+1} = A_{h,\tau}\boldsymbol{u}^n = A_{h,\tau}\boldsymbol{e}_k$, and $u_j^n = (\boldsymbol{e}_k)_j = e^{ij\theta_k}$. Therefore,

$$\begin{aligned}(A_{h,\tau}\boldsymbol{e}_k)_j = u_j^{n+1} &= e^{ij\theta_k} + \frac{\tau}{12h^2}\left(-e^{i(j-2)\theta_k} + 16e^{i(j-1)\theta_k} - 30e^{ij\theta_k} + 16e^{i(j+1)\theta_k} - e^{i(j+2)\theta_k}\right)\\ &= \left(1 + \frac{\tau}{12h^2}\left(-e^{-2i\theta_k} + 16e^{-i\theta_k} - 30 + 16e^{i\theta_k} - e^{2i\theta_k}\right)\right)(\boldsymbol{e}_k)_j.\end{aligned}$$

Thus,

$$g(\theta_k) = 1 + \frac{\tau}{12h^2}\left(-e^{-2i\theta_k} + 16e^{-i\theta_k} - 30 + 16e^{i\theta_k} - e^{2i\theta_k}\right).$$

Using

$$e^{i\theta} + e^{-i\theta} = 2\cos\theta, \qquad e^{2i\theta} + e^{-2i\theta} = 2\cos(2\theta),$$

we obtain

$$g(\theta_k) = 1 + \frac{\tau}{12h^2}\left(-2\cos(2\theta_k) + 32\cos\theta_k - 30\right).$$

Let

$$s = \sin^2\Big(\frac{\theta_k}{2}\Big).$$

Using

$$\cos\theta_k = 1 - 2s, \qquad \cos(2\theta_k) = 1 - 8s + 8s^2,$$

we obtain

$$g(\theta_k) = 1 - \frac{4\tau}{h^2}s\Big(1 + \frac{1}{3}s\Big).$$

By Proposition 3.2, the scheme is stable provided that

$$|g(\theta_k)| \le 1 \qquad \text{for all } k = 0, \dots, N-1.$$

This is equivalent to

$$-1 \le 1 - \frac{4\tau}{h^2}s\Big(1 + \frac{1}{3}s\Big) \le 1.$$

Therefore, we need

$$\frac{4\tau}{h^2}s\Big(1 + \frac{1}{3}s\Big) \le 2, \qquad \text{for all } k = 0, 1, \dots, N-1.$$

The most restrictive case occurs when

$$\sin^2\Big(\frac{\theta_k}{2}\Big) = 1,$$

which gives

$$\frac{4\tau}{h^2}\left(1 + \frac{1}{3}\right) \le 2.$$

Therefore,

$$\frac{\tau}{h^2} \le \frac{3}{8}.$$

4 Advection equations

Finally, let us consider another class of time-dependent systems, namely hyperbolic equations, and in particular the advection equation.

$$\begin{aligned} \partial_t u(x,t) + c\,\partial_x u(x,t) &= g(x,t) \qquad \text{in } (0,L)\times(0,T), \\ u(x,0) &= u_{\text{init}}(x). \end{aligned}$$

Throughout this section we assume periodic boundary conditions:

$$u(x,t) = u(x+L,t) \qquad \text{for all } (x,t).$$

When $g = 0$, this equation describes transport of the solution from left to right with speed c. The exact solution in this case is

$$u(x,t) = u_{\text{init}}(x - ct).$$

In what follows, we consider three discretizations:

1. Central difference in space + explicit Euler in time
2. Upwind difference in space + explicit Euler in time
3. Central difference in space + RK4 in time

We shall study the stability and convergence of the above methods.

For simplicity, we focus on the homogeneous case $g = 0$. Note that the stability analysis automatically generalizes to non-homogeneous case by Lemma 2.1.

4.1 Central difference scheme + explicit Euler

For any $j = 0, 1, 2, \ldots, N-1$, let

$$x_j = jh, \qquad h = \frac{L}{N}.$$

We apply the central difference approximation for the spatial derivative:

$$\partial_x u(x_j, t^n) \approx \frac{u^n_{j+1} - u^n_{j-1}}{2h},$$

together with explicit Euler for time-discretization. This gives

$$u^{n+1}_j = u^n_j - \frac{c\tau}{2h}\left(u^n_{j+1} - u^n_{j-1}\right), \tag{22}$$

$$u^0_j = u_{\text{init}}(x_j). \tag{23}$$

Exercise. Derive the local truncation error.

Stability. To study stability, we shall use Proposition 3.2.

We first compute $g(\theta_k)$. Let $\boldsymbol{u}^n = \boldsymbol{e}_k$. Then $\boldsymbol{u}^{n+1} = A_{h,\tau}\boldsymbol{u}^n$, and

$$u^n_j = (\boldsymbol{e}_k)_j = e^{ij\theta_k}.$$

Therefore,

$$\begin{aligned}
(A_{h,\tau}\boldsymbol{e}_k)_j = (A_{h,\tau}\boldsymbol{u}^n)_j = (\boldsymbol{u}^{n+1})_j = u^{n+1}_j \\
&= e^{ij\theta_k} - \frac{c\tau}{2h}\left(e^{i(j+1)\theta_k} - e^{i(j-1)\theta_k}\right) \\
&= \left(1 - i\frac{c\tau}{h}\sin\theta_k\right) e^{ij\theta_k} \\
&= \left(1 - i\frac{c\tau}{h}\sin\theta_k\right)(\boldsymbol{e}_k)_j.
\end{aligned}$$

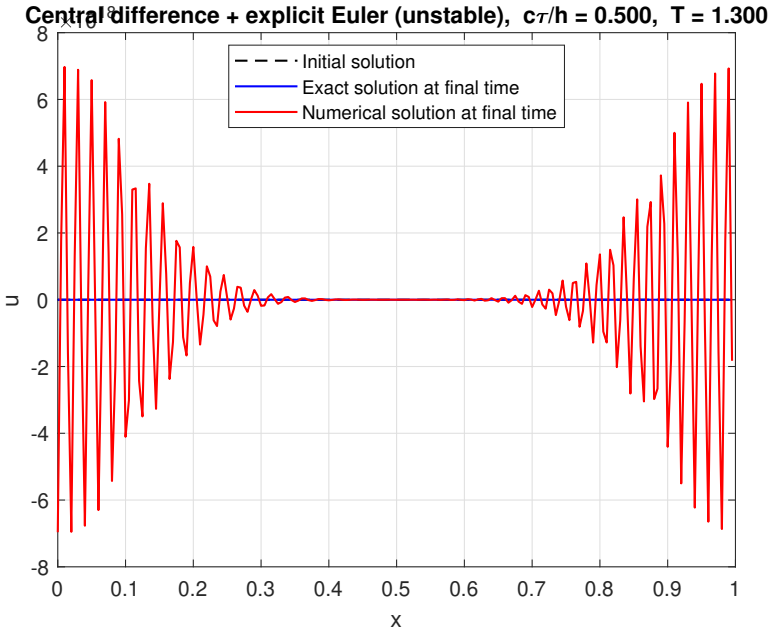
Thus,

$$g(\theta_k) = 1 - i\frac{c\tau}{h}\sin\theta_k.$$

Since

$$|g(\theta_k)|^2 = 1 + \left(\frac{c\tau}{h}\sin\theta_k\right)^2 > 1 \quad \text{for most modes } k,$$

the method is unstable.



4.2 Upwind difference scheme + explicit Euler

For any $j = 0, 1, 2, \ldots, N-1$, let

$$x_j = jh, \qquad h = \frac{L}{N}.$$

We apply the following finite difference approximation for the spatial derivative:

$$\partial_x u(x_j, t^n) \approx \frac{u_j^n - u_{j-1}^n}{h},$$

together with explicit Euler for time-discretization. This gives

$$u_j^{n+1} = u_j^n - \frac{c\tau}{h}\left(u_j^n - u_{j-1}^n\right), \tag{24}$$

$$u_j^0 = u_{\text{init}}(x_j). \tag{25}$$

Exercise. Derive the local truncation error.

Stability. To study stability, we shall use Proposition 3.2.

We first compute $g(\theta_k)$. Let $\boldsymbol{u}^n = \boldsymbol{e}_k$. Then $\boldsymbol{u}^{n+1} = A_{h,\tau}\boldsymbol{u}^n$, and

$$u_j^n = (\boldsymbol{e}_k)_j = e^{ij\theta_k}.$$

Therefore,

$$\begin{aligned}(A_{h,\tau}\boldsymbol{e}_k)_j = (A_{h,\tau}\boldsymbol{u}^n)_j = (\boldsymbol{u}^{n+1})_j = u_j^{n+1}\\ = e^{ij\theta_k} - \frac{c\tau}{h}\left(e^{ij\theta_k} - e^{i(j-1)\theta_k}\right)\\ = \left(1 - \frac{c\tau}{h}(1 - e^{-i\theta_k})\right)e^{ij\theta_k}\\ = \left(1 - \frac{c\tau}{h}(1 - e^{-i\theta_k})\right)(\boldsymbol{e}_k)_j.\end{aligned}$$

Thus,

$$g(\theta_k) = 1 - \frac{c\tau}{h}(1 - e^{-i\theta_k}).$$

Let

$$\lambda = \frac{c\tau}{h}.$$

Then

$$g(\theta_k) = (1 - \lambda) + \lambda e^{-i\theta_k}.$$

Let

$$\lambda = \frac{c\tau}{h}.$$

Then

$$g(\theta_k) = (1 - \lambda) + \lambda e^{-i\theta_k}.$$

Hence

$$\begin{aligned}|g(\theta_k)|^2 &= \left((1 - \lambda) + \lambda e^{-i\theta_k}\right)\left((1 - \lambda) + \lambda e^{i\theta_k}\right)\\ &= (1 - \lambda)^2 + \lambda(1 - \lambda)e^{i\theta_k} + \lambda(1 - \lambda)e^{-i\theta_k} + \lambda^2\\ &= (1 - \lambda)^2 + \lambda^2 + \lambda(1 - \lambda)\left(e^{i\theta_k} + e^{-i\theta_k}\right)\\ &= (1 - \lambda)^2 + \lambda^2 + 2\lambda(1 - \lambda)\cos\theta_k\\ &= 1 - 2\lambda + 2\lambda^2 + 2\lambda(1 - \lambda)\cos\theta_k\\ &= 1 - 2\lambda(1 - \lambda) + 2\lambda(1 - \lambda)\cos\theta_k\\ &= 1 + 2\lambda(1 - \lambda)(\cos\theta_k - 1).\end{aligned}$$

Now, if $0 \le \lambda \le 1$, then

$$\lambda(1 - \lambda) \ge 0, \qquad \cos\theta_k - 1 \le 0.$$

Therefore,

$$2\lambda(1 - \lambda)(\cos\theta_k - 1) \le 0,$$

and hence

$$|g(\theta_k)|^2 \le 1.$$

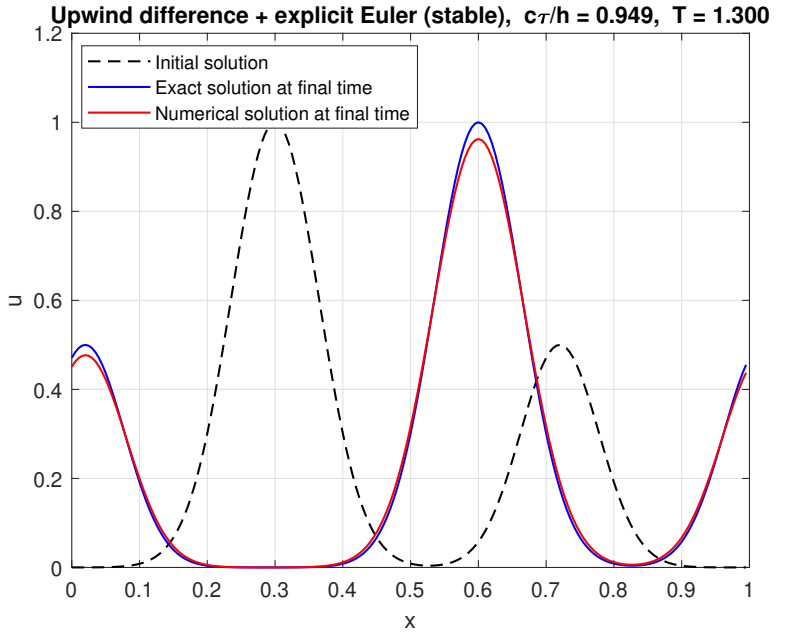
Thus,

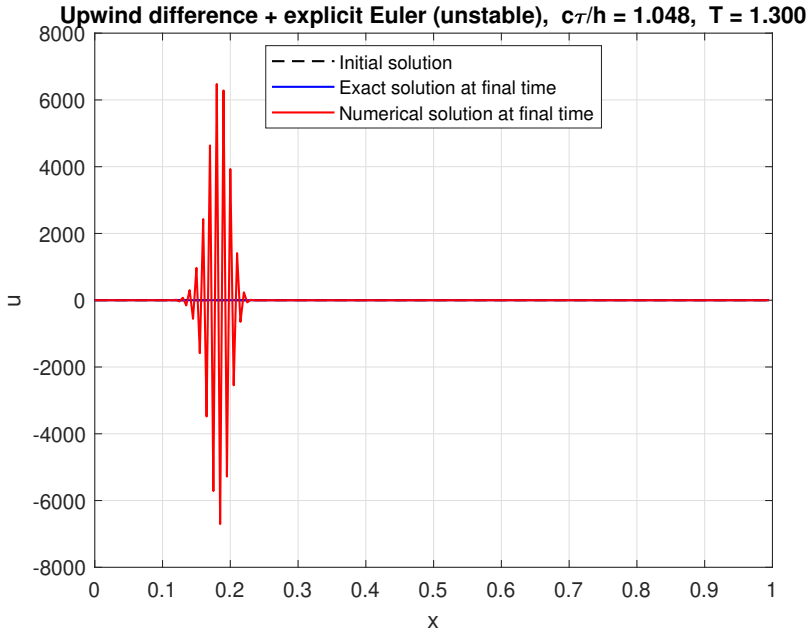
$$|g(\theta_k)| \leq 1 \quad \text{whenever} \quad 0 \leq \lambda \leq 1.$$

Therefore the scheme is stable provided

$$\frac{c\tau}{h} \leq 1.$$

This condition is called the CFL condition.





4.3 Central difference scheme + RK4

For any $j = 0, 1, 2, \ldots, N-1$, let

$$x_j = jh, \qquad h = \frac{L}{N}.$$

We again apply the central difference approximation for the spatial derivative:

$$\partial_x u(x_j, t) \approx \frac{u_{j+1} - u_{j-1}}{2h}.$$

This leads to the semi-discrete system

$$\frac{d}{dt} u_j(t) = -\frac{c}{2h}(u_{j+1} - u_{j-1}), \tag{26a}$$

$$u_j(0) = u_{\text{init}}(x_j). \tag{26b}$$

Thus, the spatial discretization operator D_h is

$$(D_h \boldsymbol{u})_j = -\frac{c}{2h}(u_{j+1} - u_{j-1}),$$

and (26) can be written in the vector form $\boldsymbol{u}'(t) = D_h \boldsymbol{u}(t)$ with $\boldsymbol{u}(t) = (u_0(t), u_1(t), ..., u_{N-1}(t))^\top$.

Exercise. Show, by using Fourier techniques, that the eigenvalues of D_h are

$$\lambda_k = -i\frac{c}{h}\sin\theta_k, \quad k = 0, 1, 2, \ldots, N-1,$$

and we have $D_h \boldsymbol{e}_k = \lambda_k \boldsymbol{e}_k$.

Stability. To study stability, we use the following modification of Proposition 3.1, adapted to periodic boundary conditions and complex-value setting.

Proposition 4.1. *Suppose that D_h admits an orthonormal basis of eigenvectors*

$$D_h \boldsymbol{v}_j = \lambda_j \boldsymbol{v}_j, \qquad j = 1, 2, \ldots, N,$$

where $\{\boldsymbol{v}_j\}_{j=1}^N$ is an orthonormal basis of $\mathbb{C}^N$. Assume that the stability function R of the Runge–Kutta method satisfies

$$|R(\tau\lambda_j)| \le 1, \qquad j = 1, 2, \ldots, N.$$

Then the fully discrete scheme is stable.

Applying RK4 to the semi-discrete system (26), we now study stability.
From the exercise above, the eigenvalues of D_h are

$$\lambda_k = -i\frac{c}{h}\sin\theta_k, \qquad k = 0, 1, 2, \ldots, N-1,$$

and the corresponding Fourier modes $\{\frac{1}{\sqrt{N}}\boldsymbol{e}_k\}_{k=0}^{N-1}$ form an orthonormal basis of $\mathbb{C}^N$. Therefore the assumptions of the proposition are satisfied.

For RK4, the stability function is

$$R(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24}.$$

Hence the amplification factor is

$$g(\theta_k) = R(\tau\lambda_k) = R\Big(-i\frac{c\tau}{h}\sin\theta_k\Big).$$

In Part I, we have plotted the linear stability region of RK4 method

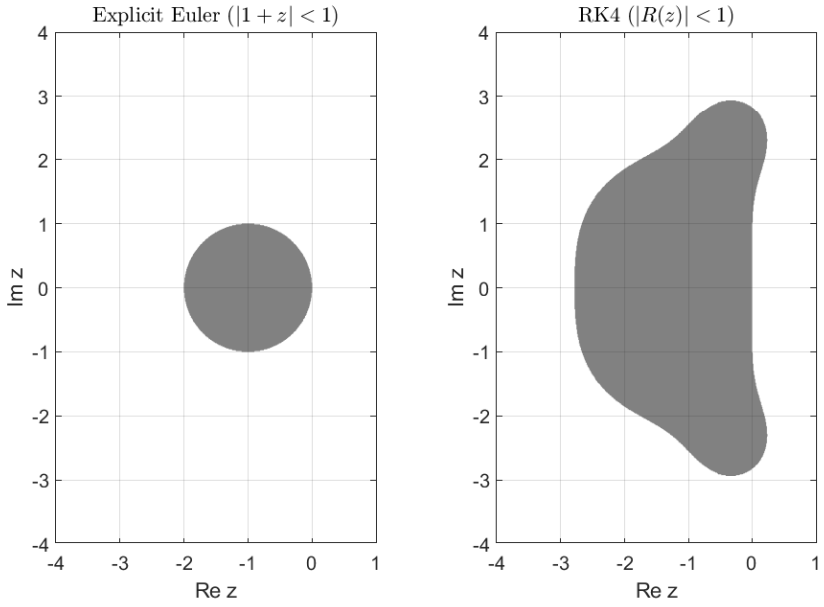


Figure 1: Visualization of the linear stability regions of explicit Euler and RK4.

We can prove that the stability region of RK4 contains the interval

$$[-i\sqrt{8},\, i\sqrt{8}]$$

on the imaginary axis, we have

$$|g(\theta_k)| = |R(\tau\lambda_k)| \le 1 \quad \text{whenever} \quad |\tau\lambda_k| \le \sqrt{8}.$$

Using

$$\lambda_k = -i\frac{c}{h}\sin\theta_k,$$

this condition becomes

$$\Big|\frac{c\tau}{h}\sin\theta_k\Big| \le \sqrt{8}.$$

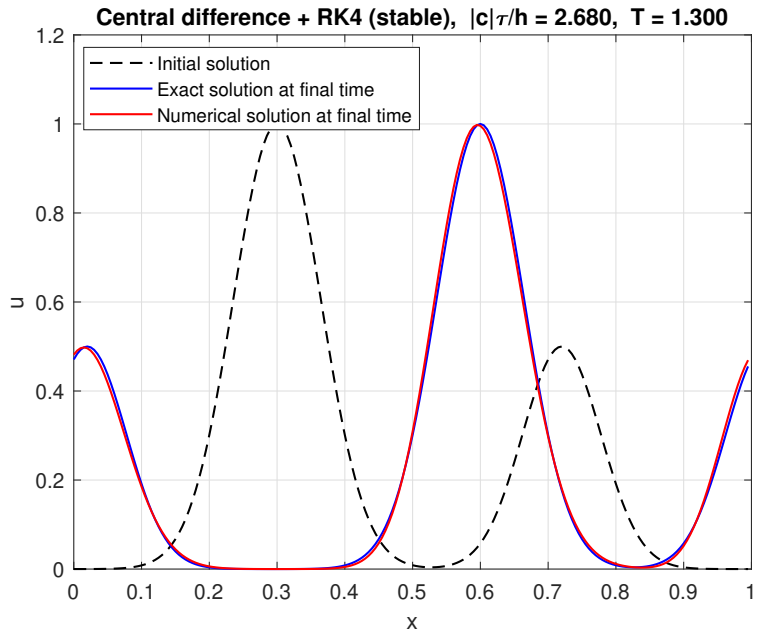
Hence, if

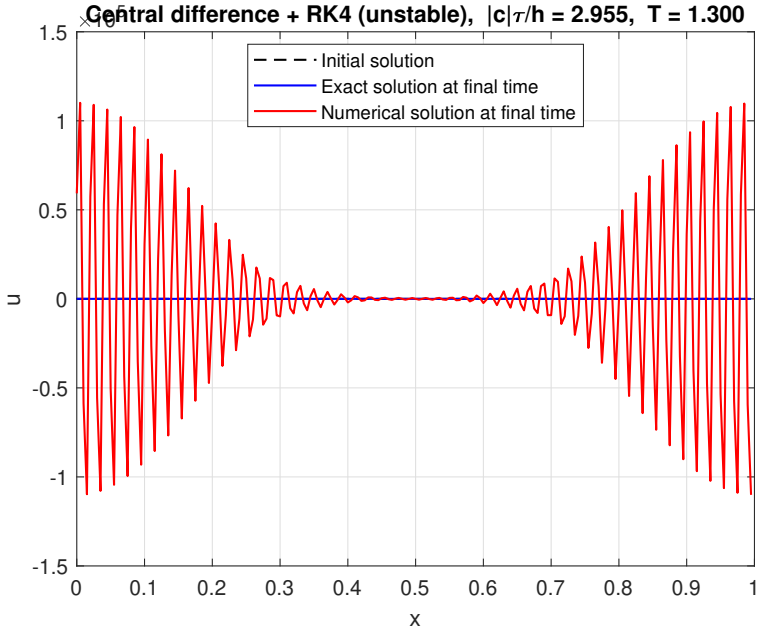
$$\frac{|c|\tau}{h} \le \sqrt{8} \approx 2.828,$$

then

$$|g(\theta_k)| = |R(\tau\lambda_k)| \leq 1 \qquad \text{for all } k = 0, 1, \ldots, N-1.$$

By the proposition above, the fully discrete scheme is stable.





Remark 4.1. *The above three examples show that the instability observed earlier in the first example can be removed by either adjusting the spatial discretization or adjusting the time discretization. The stability of the fully discrete scheme is therefore determined by the interaction between the space and time discretizations.*

The following code was used to generate the stability/instability test results that we have seen above.

```
clear; clc; close all;

% ============================================================
% 1D linear advection: u_t + c u_x = 0 on [0,L] with periodic BC
%
% Three cases:
%   1) Central difference + explicit Euler
%   2) Upwind difference  + explicit Euler
%   3) Central difference + RK4
%
% For cases 2 and 3:
%   - one stable example: dt slightly below the stability threshold
%   - one unstable example: dt slightly above the stability threshold
%
% All figures have the same format and include:
%   - initial solution
%   - exact solution at final time
%   - numerical solution at final time
%   - final time in the title
% ============================================================

% -------------------------
% Basic parameters
% -------------------------
L = 1.0;
c = 1.0;              % use c > 0 to match the upwind formula in the notes
N = 200;
h = L / N;
x = (0:N-1)' * h;

% -------------------------
% Initial condition
% -------------------------
% Smooth profile plus a moderate higher-frequency component so that
% instability is easier to see in case 1.
u0 =initial_profile(x, L);

% exact periodic solution
u_exact_fun = @(x,t) initial_profile(mod(x - c*t, L), L);

% -------------------------
% plotting parameters
% -------------------------
lw = 1.0;
fs = 10;

% ============================================================
% CASE 1: Central difference + explicit Euler
% ============================================================
% Scheme:
%   u_j^{n+1} = u_j^n - (c dt / (2h)) (u_{j+1}^n - u_{j-1}^n)
```

```
%
% This scheme is unstable.
%
% To make the instability visible, we use a moderate CFL and a longer T.
% ============================================================

T1 = 1.3;
cfl1 = 0.5;
dt1 = cfl1 * h / abs(c);

[u1, t1, dt1_used] = solve_central_euler(u0, c, h, dt1, T1);
u1_exact = u_exact_fun(x, t1);

make_plot(x, u0, u1_exact, u1, ...
    sprintf('Central difference + explicit Euler (unstable),  c\\tau/h = %.3f,  T = %.3f', ...
    abs(c)*dt1_used/h, t1), lw, fs);



% ============================================================
% CASE 2: Upwind difference + explicit Euler
% ============================================================
% Scheme (for c>0):
%   u_j^{n+1} = u_j^n - (c dt / h)(u_j^n - u_{j-1}^n)
%
% Stability condition:
%   0 <= c dt / h <= 1
% ============================================================

T2 = 1.3;

cfl2_stable   = 0.95;
cfl2_unstable = 1.05;

dt2_stable   = cfl2_stable   * h / abs(c);
dt2_unstable = cfl2_unstable * h / abs(c);

[u2s, t2s, dt2s_used] = solve_upwind_euler(u0, c, h, dt2_stable, T2);
[u2u, t2u, dt2u_used] = solve_upwind_euler(u0, c, h, dt2_unstable, T2);

u2s_exact = u_exact_fun(x, t2s);
u2u_exact = u_exact_fun(x, t2u);

make_plot(x, u0, u2s_exact, u2s, ...
    sprintf('Upwind difference + explicit Euler (stable),  c\\tau/h = %.3f,  T = %.3f', ...
    abs(c)*dt2s_used/h, t2s), lw, fs);

make_plot(x, u0, u2u_exact, u2u, ...
    sprintf('Upwind difference + explicit Euler (unstable),  c\\tau/h = %.3f,  T = %.3f', ...
    abs(c)*dt2u_used/h, t2u), lw, fs);



% ============================================================
```

```
% CASE 3: Central difference + RK4
% ============================================================
% Semi-discrete operator:
%   (D_h u)_j = -(c/(2h))(u_{j+1} - u_{j-1})
%
% Time stepping by RK4.
%
% Stability condition:
%   |c| dt / h <= sqrt(8)
% ============================================================

T3 = 1.3;

rk4_lim = sqrt(8);
cfl3_stable   = 0.95 * rk4_lim;
cfl3_unstable = 1.05 * rk4_lim;

dt3_stable   = cfl3_stable   * h / abs(c);
dt3_unstable = cfl3_unstable * h / abs(c);

[u3s, t3s, dt3s_used] = solve_central_rk4(u0, c, h, dt3_stable, T3);
[u3u, t3u, dt3u_used] = solve_central_rk4(u0, c, h, dt3_unstable, T3);

u3s_exact = u_exact_fun(x, t3s);
u3u_exact = u_exact_fun(x, t3u);

make_plot(x, u0, u3s_exact, u3s, ...
    sprintf('Central difference + RK4 (stable),  |c|\\tau/h = %.3f,  T = %.3f', ...
    abs(c)*dt3s_used/h, t3s), lw, fs);

make_plot(x, u0, u3u_exact, u3u, ...
    sprintf('Central difference + RK4 (unstable),  |c|\\tau/h = %.3f,  T = %.3f', ...
    abs(c)*dt3u_used/h, t3u), lw, fs);

% -------------------------
% print actual ratios
% -------------------------
fprintf('Case 1: Central + explicit Euler:\n');
fprintf('  actual c*dt/h = %.6f, T = %.6f\n\n', abs(c)*dt1_used/h, t1);

fprintf('Case 2: Upwind + explicit Euler:\n');
fprintf('  stable   c*dt/h = %.6f, T = %.6f\n', abs(c)*dt2s_used/h, t2s);
fprintf('  unstable c*dt/h = %.6f, T = %.6f\n\n', abs(c)*dt2u_used/h, t2u);

fprintf('Case 3: Central + RK4:\n');
fprintf('  stable   |c|*dt/h = %.6f, T = %.6f\n', abs(c)*dt3s_used/h, t3s);
fprintf('  unstable |c|*dt/h = %.6f, T = %.6f\n', abs(c)*dt3u_used/h, t3u);

% ============================================================
% Local functions
% ============================================================

function u = initial_profile(x, L)
    u = exp(-120*(x-0.30).^2) + 0.5*exp(-150*(x-0.72).^2);% + 0.08*sin(2*pi*25*x/L);
```

```
end

function [u, t, dt] = solve_central_euler(u0, c, h, dt_guess, T)
    nSteps = ceil(T / dt_guess);
    dt = T / nSteps;
    u = u0;
    t = 0;

    for n = 1:nSteps
        u = u - (c*dt/(2*h)) * (circshift(u,-1) - circshift(u,1));
        t = t + dt;
    end
end

function [u, t, dt] = solve_upwind_euler(u0, c, h, dt_guess, T)
    nSteps = ceil(T / dt_guess);
    dt = T / nSteps;
    u = u0;
    t = 0;

    if c >= 0
        for n = 1:nSteps
            u = u - (c*dt/h) * (u - circshift(u,1));
            t = t + dt;
        end
    else
        % correct upwind formula for c < 0
        for n = 1:nSteps
            u = u - (c*dt/h) * (circshift(u,-1) - u);
            t = t + dt;
        end
    end
end

function [u, t, dt] = solve_central_rk4(u0, c, h, dt_guess, T)
    nSteps = ceil(T / dt_guess);
    dt = T / nSteps;
    u = u0;
    t = 0;

    for n = 1:nSteps
        k1 = Dh_central(u, c, h);
        k2 = Dh_central(u + 0.5*dt*k1, c, h);
        k3 = Dh_central(u + 0.5*dt*k2, c, h);
        k4 = Dh_central(u + dt*k3, c, h);
        u = u + dt*(k1 + 2*k2 + 2*k3 + k4)/6;
        t = t + dt;
    end
end

function rhs = Dh_central(u, c, h)
    rhs = -(c/(2*h)) * (circshift(u,-1) - circshift(u,1));
end
```

```
function make_plot(x, u0, u_exact, u_num, ttl, lw, fs)
    figure;
    plot(x, u0, 'k--', 'LineWidth', lw); hold on;
    plot(x, u_exact, 'b-', 'LineWidth', lw);
    plot(x, u_num, 'r-', 'LineWidth', lw);
    grid on;
    box on;
    xlabel('x', 'FontSize', fs);
    ylabel('u', 'FontSize', fs);
    title(ttl, 'FontSize', fs);
    legend('Initial solution', 'Exact solution at final time', 'Numerical solution at final time', ...
        'Location', 'best', 'FontSize', fs-1);
    set(gca, 'FontSize', fs);
end
```