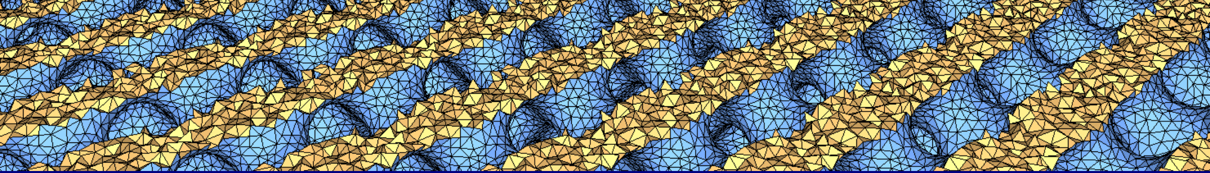
$$u\left(x,t\right)=\sum_{\nu=0}^{\infty}a_{\nu}^{0}e^{i\nu x}e^{\nu^{2}t}$$

$$U_{n}^{m}=\sum_{\nu=0}^{N}a_{\nu}^{m}e^{i\nu nh}\coloneqq\sum_{\nu=0}^{N}a_{\nu}^{0}\omega_{\nu}^{m}e^{i\beta_{\nu}n}.$$



Fourier stability analysis

Fourier stability analysis

We will introduce some simple spatial discretizations for the periodic constant-coefficient advection-diffusion problem on a uniform grid $\Omega_h = \{x_1, \ldots, x_m\}$ with grid points $x_j = jh$ and mesh width $h = \frac{1}{m}$.
Discrete Fourier decomposition is an important tool to analize linear difference schemes with spatial periodic conditions. It can be used to study fundamental properties like stability, dissipation and dispersion.

Definition

Fourier modes

$$\forall k \in \mathbb{Z} : \varphi_k\left(x\right) = \exp\left(2\pi i k x\right).$$

These modes form an orthonormal basis for $L^2\left[0,1\right]$ space, i.e.,

$$\forall v \in L^2\left[0,1\right] : \sum_{k\in\mathbb{Z}} \alpha_k \varphi_k\left(x\right),$$

where the right-hand side is a convergent series, which we now call the Fourier series. The Fourier coefficients are given by $\alpha_k = \langle \varphi_k, v \rangle$.

Definition

Discrete Fourier modes

$$\forall k \in \mathbb{Z} : \phi_k\left(x\right) = \left(\varphi_k\left(x_1\right), \ldots, \varphi_k\left(x_m\right)\right) \in \mathbb{C}^m.$$

The Courant-Friedrichs-Lewy (CFL) condition

Explicit schemes for the advection equation $\partial_t u + c\partial_x u = 0$ give rise to step size restrictions (stability conditions) of the form

$$|c| \frac{\Delta t}{\Delta x} \leq C,$$

with C an appropiate positive constant independent of Δx and Δt.

Remark

In the beginning of numerical analysis, finite difference approximations were used to prove the existence of PDE solutions. For convergence of finite difference approximations are necessary in order to guarantee that the mathematical domain of dependence of a PDE problem lies within the numerical counterpart of the finite difference method.

$$U_j^{n+1} = \sum_{k=-r}^{r} \gamma_k U_{j+k}^n.$$

A necessary condition for stability is that the mathematical domain of dependence of PDE is contained in the numerical domain of dependence.

Fourier stability analysis

In the 1940's, John von Neumann introduced Fourier analysis in the theory of finite difference schemes for time-dependent PDEs. We are interested in the propagation of small errors at different grid points. If these errors are not controlled at each stage of the time iteration, they can grow and create a solution completely different from the desired solution at a later time. An initial pulse which has bounded size but oscillates with frequency k. Take a complex exponential for some $k \in \mathbb{R}$

$$e^{ikx} = \cos(kx) + i\sin(kx).$$

The size of the complex-valued pulse is given by its modulus, and

$$\left|e^{ikx}\right| = \left|\cos(kx) + i\sin(kx)\right| = 1.$$

The growth in size when applying the scheme once to e^{ikx}, captured by an amplification constant called the growth factor. At time step n, for some $j \in \mathbb{Z}$ we take

$$U_j^n(k) \equiv \lambda(k)^n e^{ik(j\Delta x)} \implies \begin{cases} U_j^{n+1}(k) & = \lambda(k)^{n+1} e^{ik(j\Delta x)}. \\ U_{j+1}^n(k) & = \lambda(k)^n e^{ikj\Delta x} e^{ik\Delta x}. \\ U_{j-1}^n(k) & = \lambda(k)^n e^{ikj\Delta x} e^{-ik\Delta x}. \\ U_j^{n-1}(k) & = \lambda(k)^{n-1} e^{ikj\Delta x}. \end{cases}$$

Remark

$$\operatorname{Re}\left(e^{ik\Delta x}\right) = \cos(k\Delta x) = \frac{e^{ik\Delta x} + e^{-ik\Delta x}}{2}, \qquad \operatorname{Im}\left(e^{ik\Delta x}\right) = \sin(k\Delta x) = \frac{e^{ik\Delta x} - e^{-ik\Delta x}}{2i}.$$

Fourier stability analysis

Example (Transport equation on a periodic domain)

For some constant $c > 0$ and some continuous and bounded function $g\colon [0,1] \to \mathbb{R}$.

$$\begin{cases} \partial_t u + c\partial_x u = 0, & x \in (0,1)\,,\ t > 0. \\ u(0,t) = u(1,t)\,, & t > 0. \\ u(x,0) = g(x)\,, & x \in [0,1]\,. \end{cases}$$

We consider the implicit FDM

(6)
$$\begin{cases} \frac{v_j^{m+1} - v_j^m}{\Delta t} + c\frac{v_j^{m+1} - v_{j-1}^{m+1}}{\Delta x} = 0 & j = 1, \dots, n+1,\ m = 0, 1, \dots \\ v_0^{m+1} = v_{n+1}^{m+1} & m = 0, 1, \dots \\ v_j^0 = g(x_j) & j \in \mathbb{Z}. \end{cases}$$

Show that for any choice of $\Delta t, \Delta x > 0$, any solution of (6) satisfies

$$\inf_{x\in[0,1]} g(x) \leqslant v_j^m \leqslant \sup_{x\in[0,1]} g(x)\,.$$

Fourier stability analysis

Solution

Let J be a point at which $\left\{v_J^{m+1}\right\}_{j=1}^{n+1}$ attains its maximum. Then

$$v_J^{m+1} = v_J^m - c\frac{\Delta t}{\Delta x}\left(v_J^{m+1} - v_{J-1}^{m+1}\right).$$

By assumption, $c > 0$, and by the choice of J we have $v_J^{m+1} - v_{J-1}^{m+1} \geq 0$. Iterating the inequality over all m yields

$$v_j^m \leq \max_{j=0,\ldots,n+1} v_j^0 \leq \sup_{x\in[0,1]} g\left(x\right).$$

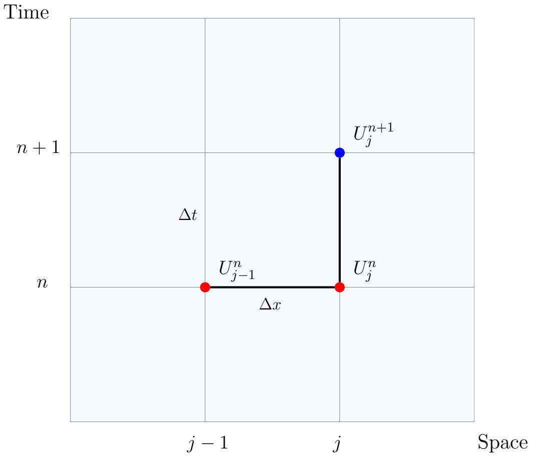
Fourier stability analysis

Example (First-Order Upwind (FOU))

Proposed by Richard Courant, Eugene Isaacson and Mina Rees. Let $c(x,t) = c > 0$.

$$0 = \frac{U_j^{n+1} - U_j^n}{\Delta t} + c\frac{U_j^n - U_{j-1}^n}{\Delta x}.$$

$$U_j^{n+1} = U_j^n - r\left(U_j^n - U_{j-1}^n\right), \quad r = c\frac{\Delta t}{\Delta x}.$$



Fourier stability analysis

Theorem (Stability analysis for FOU scheme)

$$r \in (0,1] \iff |\lambda(k)| \leq 1.$$

Proof.

$$\begin{aligned} U_j^{n+1} &= U_j^n - r\left(U_j^n - U_{j-1}^n\right). \\ \lambda(k)^{n+1} e^{ik(j\Delta x)} &= \lambda(k)^n e^{ik(j\Delta x)} - r\lambda(k)^n \left(e^{ik(j\Delta x)} - e^{ik(j-1)\Delta x}\right). \\ \lambda(k) &= 1 - r\left(1 - e^{-ik\Delta x}\right). \\ |\lambda(k)| &= \left|1 - r + re^{-ik\Delta x}\right| \\ &\leq |1-r| + \left|re^{-ik\Delta x}\right| \\ &= |1-r| + r \\ &\leq 1 \iff r \in (0,1]. \end{aligned}$$

□

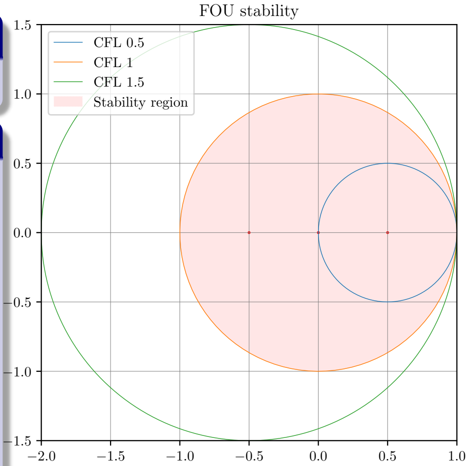


Figure: The FOU scheme is dissipative.

```toml
label = "FOU scheme for 1D linear advection PDE"
space = [0, 10]
spacepoints = [6]
speed = 5
time = [0, 1]
timesteps = 5
piecewise = [2, 4]
```

Program : Parameters file fou.toml.

```python
from tomllib import load

import numpy as np

with open(file="fou.toml", mode="rb") as f:
    data = load(f)

c = data["speed"]
x, Δx = np.linspace(
    start=data["space"][0],
    stop=data["space"][1],
    num=data["spacepoints"][0],
    retstep=True,
)
t, Δt = np.linspace(
    start=data["time"][0], stop=data["time"][1], num=data["timesteps"], retstep=True
)
cfl = c * Δt / Δx

u = np.where((data["piecewise"][0] ⩽ x) & (data["piecewise"][1] ⩽ 4), 1.0, 0)
u = np.insert(u, 0, u[0])  # left hand side ghost node
u = np.append(u, u[-1])  # right hand side ghost node

print(f"CFL number: {cfl}")

print("u0     u1     u2     u3     u4     u5     u6")
for timestep in t:
    print(u)
    u[1:] -= cfl * (u - np.roll(u, 1))[1:]
    u[0] = u[1]
    u[-1] = u[-2]
```

Program : fou.py.

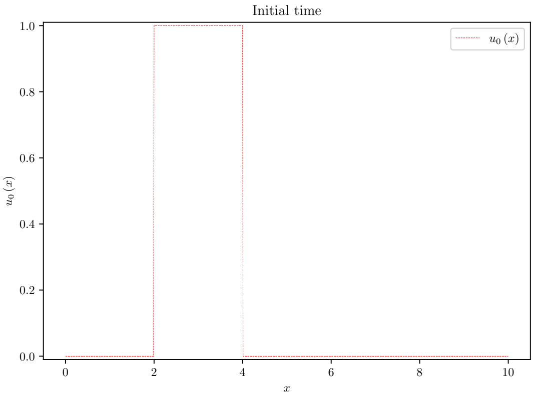


Figure: Initial profile.

```
CFL number: 0.625

u0     u1     u2     u3     u4     u5     u6
[0. 0. 1. 1. 1. 1. 1. 1.]
[0.    0.    0.375 1.    1.    1.    1.    1.   ]
[0.       0.       0.140625 0.609375 1.       1.       1.       1.       ]
[0.       0.       0.052734 0.316406 0.755859 1.       1.       1.       ]
[0.       0.       0.019775 0.151611 0.481201 0.847412 1.       1.       ]
```