Lecture 8: Advanced DEs

Lecture 9, topics:

- Stability
  - FTCS explicit
  - FTCS implicit
  - C-N
- Verlet method
- Spectral methods
- Boundary Value Problems

Stability

How can we determine stability in PDEs?

Von Neumann Analysis: consider a single Fourier mode k, and see how it evolves.

FTCS Explicit Method for Parabolic Equations

Consider the diffusion equation:

$$\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2},$$

Consider T_m^n (spatial grid point m with spacing a, timestep n with interval h) as an inverse DFT: $\quad T_m^n = \sum_k \widehat{T}_k^n \exp(ikx_m)$

$$T_m^{n+1} = T_m^n + \frac{\kappa h}{a^2}\left(T_{m+1}^n - 2T_m^n + T_{m-1}^n\right)$$

Then

$$\widehat{T}_k^{n+1} \mathrm{e}^{ikam} = \left(1 - \frac{2\kappa h}{a^2}\right) \widehat{T}_k^n \mathrm{e}^{ikam} + \frac{\kappa h}{a^2}\left(\widehat{T}_k^n \mathrm{e}^{ika(m+1)} - \widehat{T}_k^n \mathrm{e}^{ika(m-1)}\right)$$

$$\Rightarrow \left|\frac{\widehat{T}_k^{n+1}}{\widehat{T}_k^n}\right| = 1 + \frac{\kappa h}{a^2}\left(\mathrm{e}^{ika} + \mathrm{e}^{-ika} - 2\right) = \left|1 - \frac{4h\kappa}{a^2}\sin^2\left(\frac{ka}{2}\right)\right|.$$

- This is the growth factor, and it should be less than unity if the solution is not meant to grow

Stability criterion:

$$\boxed{h \leq \frac{a^2}{2\kappa}}. \qquad \textit{(independent of k!)}$$

FTCS stable for the parabolic equation, provided temporal resolution is adequate ($a \geq \sqrt{2\kappa h}$).

Slide Type

FTCS Explicit Method for Hyperbolic Equations

Slide Type Slide

- Consider the wave equation:

$$\frac{\partial^2 \phi}{\partial t^2} = c^2 \frac{\partial^2 \phi}{\partial x^2},$$

- Discretize as set of ODEs:

Slide Type Sub-Slide

$$\phi_m^{n+1} = \phi_m^n + h\psi_m^n,$$
$$\psi_m^{n+1} = \psi_m^n + h\frac{c^2}{a^2}\left(\phi_{m-1}^n + \phi_{m+1}^n - 2\phi_m^n\right).$$

or, equivalently:

$$\begin{bmatrix} \phi_m^{n+1} \\ \psi_m^{n+1} \end{bmatrix} = \begin{bmatrix} 1 & h \\ -\frac{2hc^2}{a^2} & 1 \end{bmatrix} \begin{bmatrix} \phi_m^n \\ \psi_m^n \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{c^2 h}{a^2}\left(\phi_{m+1}^n + \phi_{m-1}^n\right) \end{bmatrix}$$

Slide Type Sub-Slide

- Consider single Fourier mode, $\begin{bmatrix} \phi_m^n \\ \psi_m^n \end{bmatrix} = \begin{bmatrix} \widehat{\phi}_k^n \\ \widehat{\psi}_k^n \end{bmatrix} \exp(ikma)$. Obtain, after some algebra,

$$\begin{bmatrix} \widehat{\phi}_k^{n+1} \\ \widehat{\psi}_k^{n+1} \end{bmatrix} = \mathbf{A} \begin{bmatrix} \widehat{\phi}_k^n \\ \widehat{\psi}_k^n \end{bmatrix},$$

$$\text{with} \quad \mathbf{A} = \begin{bmatrix} 1 & h \\ -hr^2 & 1 \end{bmatrix} \quad \text{and} \quad r^2 = \frac{2c}{a}\sin\frac{ka}{2},$$

  which **does** depend on k.

Slide Type Sub-Slide

- Eigenvalues of $\mathbf{A}$ are $\lambda_1 = 1 + ihr$ and $\lambda_2 = 1 - ihr$,
- therefore, $|\lambda_\pm|^2 = 1 + h^2 r^2 \geq 1$.
- Define corresponding eigenvectors $\mathbf{V}_1$ and $\mathbf{V}_2$, project initial condition on eigenvectors, i.e., write $\alpha_1 \mathbf{V}_1 + \alpha_2 \mathbf{V}_2$.
- After p time steps, solution becomes $\alpha_1 \lambda_1^p \mathbf{V}_1 + \alpha_2 \lambda_2^p \mathbf{V}_2$, which grows unbounded!

⇒ **FTCS always unstable for the wave equation!**

FTCS Implicit Method for Hyperbolic Equations

Slide Type Sub-Slide

Using the implicit method for the wave equation, discretization gives:

$$\begin{bmatrix}\phi_m^n\\ \psi_m^n\end{bmatrix} = \begin{bmatrix}1 & -h\\ +\frac{2hc^2}{a^2} & 1\end{bmatrix}\begin{bmatrix}\phi_m^{n+1}\\ \psi_m^{n+1}\end{bmatrix} - \begin{bmatrix}0\\ \frac{c^2h}{a^2}\left(\phi_{m+1}^{n+1}+\phi_{m-1}^{n+1}\right)\end{bmatrix}$$

If we do the Von Neumann substitution, $(\phi_m^n, \psi_m^n) = \left(\widehat{\phi}_k^n, \widehat{\psi}_k^n\right)\exp(ikma)$, we get

$$\mathbf{B}\begin{bmatrix}\widehat{\phi}_k^{n+1}\\ \widehat{\psi}_k^{n+1}\end{bmatrix} = \begin{bmatrix}\widehat{\phi}_k^{n}\\ \widehat{\psi}_k^{n}\end{bmatrix} \Rightarrow \begin{bmatrix}\widehat{\phi}_k^{n+1}\\ \widehat{\psi}_k^{n+1}\end{bmatrix} = \mathbf{B}^{-1}\begin{bmatrix}\widehat{\phi}_k^{n}\\ \widehat{\psi}_k^{n}\end{bmatrix}.$$

with:

$$\mathbf{B} = \begin{bmatrix}1 & -h\\ hr^2 & 1\end{bmatrix}, r = \frac{2c}{a}\sin\frac{ka}{2}$$

Slide Type Sub-Slide

```
from sympy import *
init_printing()
h, r = symbols('h, r', positive=True)
B = Matrix([[1, -h], [h*r**2, 1]])
B
```

$$\begin{bmatrix}1 & -h\\ hr^2 & 1\end{bmatrix}$$

Slide Type Fragment

```
# inverse of B
B**-1
```

$$\begin{bmatrix}\frac{1}{h^2r^2+1} & \frac{h}{h^2r^2+1}\\ -\frac{hr^2}{h^2r^2+1} & \frac{1}{h^2r^2+1}\end{bmatrix}$$

Slide Type Fragment

```
# eigenvalues as a list
L = list((B**-1).eigenvals().keys())
```

Slide Type Fragment

```
# First eigenvalue
L[0].factor()
```

$$-\frac{i(hr+i)}{h^2r^2+1}$$

Slide Type Sub-Slide

```
# Magnitude of first eigenvalue
abs(L[0].factor())
```

$$\frac{1}{\sqrt{h^2r^2+1}}$$

Slide Type Fragment

```
# Magnitude of 2nd eigenvalue
abs(L[1].factor())
```

$$\frac{1}{\sqrt{h^2r^2+1}}$$

Slide Type Sub-Slide

$$\text{Recall} \quad \begin{bmatrix} \widehat{\phi}_k^{m+1} \\ \widehat{\psi}_k^{m+1} \end{bmatrix} = \mathbf{B}^{-1} \begin{bmatrix} \widehat{\phi}_k^{m} \\ \widehat{\psi}_k^{m} \end{bmatrix}.$$

The eigenvalues of $\mathbf{B}^{-1}$ are

$$\lambda_\pm = \frac{1 \pm ihr}{1 + h^2 r^2}, \qquad |\lambda_\pm| = \frac{1}{\sqrt{1 + h^2 r^2}} \le 1.$$

- The eigenvalues are the growth factors and these are less than one.
- So the implicit method is unconditionally stable, but solutions decay exponentially!
- For the wave equation, all Fourier components of our solution (except k=0) die away... meaning the wave solution cannot propagate!

Slide Type

Crank-Nicolson Method for Hyperbolic Equations

Slide Type Slide

Using C-N method for wave equation, discretization gives:

$$\phi_m^{n+1} - \frac{h}{2}\psi_m^{n+1} = \phi_m^n + \frac{h}{2}\psi_m^n$$
$$\psi_m^{n+1} - \frac{h}{2}\frac{c^2}{a^2}\left(\phi_{m-1}^{n+1} + \phi_{m+1}^{n+1} - 2\phi_m^{n+1}\right) = \psi_m^n + \frac{h}{2}\frac{c^2}{a^2}\left(\phi_{m-1}^{n} + \phi_{m+1}^{n} - 2\phi_m^{n}\right).$$

Slide Type Sub-Slide

If we do the Von Neumann substitution, $(\phi_m^n, \psi_m^n) = \left(\widehat{\phi}_k^n, \widehat{\psi}_k^n\right)\exp(ikma)$, we get

$$\mathbf{B}' \begin{bmatrix} \widehat{\phi}_m^{n+1} \\ \widehat{\psi}_m^{n+1} \end{bmatrix} = \mathbf{A}' \begin{bmatrix} \widehat{\phi}_m^{n} \\ \widehat{\psi}_m^{n} \end{bmatrix},$$

or

$$\begin{bmatrix} \widehat{\phi}_m^{n+1} \\ \widehat{\psi}_m^{n+1} \end{bmatrix} = \mathbf{B}'^{-1}\mathbf{A}' \begin{bmatrix} \widehat{\phi}_m^{n} \\ \widehat{\psi}_m^{n}, \end{bmatrix}$$

with

$$\mathbf{B}'^{-1}\mathbf{A}' = \frac{a}{1 + h^2 r'^2}\begin{bmatrix} 1 - h^2 r'^2 & 2h \\ -2hr'^2 & 1 - h^2 r'^2 \end{bmatrix}, \qquad r' = \frac{c}{a}\sin\frac{ka}{2}$$

Slide Type Sub-Slide

Growth factors of Crank-Nicolson are eigenvalues of $\mathbf{B}^{-1}\mathbf{A}$:

$$\lambda_\pm = \frac{1 \pm 2ihr' - h^2 r'^2}{1 + h^2 r'^2}, \qquad \boxed{|\lambda_\pm| = 1}.$$

Growth factors are one, so the solution neither grows nor decays.

Verlet Method

Slide Type Slide

- Guaranteed time-reversible and energy-conserving method for the special case of two coupled ODEs, with LHS and RHS having separated variables
- e.g. Newton's 2nd law for conservative forces:
$$\frac{d^2x}{dt^2} = \frac{F(x)}{m} \quad \Rightarrow \quad \frac{dx}{dt} = v \quad \text{and} \quad \frac{dv}{dt} = \frac{F(x)}{m}.$$
1st ODE: x on LHS, v on RHS; 2nd ODE: v on LHS, x on RHS.

First, take an Euler step:

$$v\left(t+\frac{h}{2}\right) = v(t) + \frac{h}{2}\frac{F(x(t))}{m}$$

Then, for all subsequent timesteps:

$$x(t+h) = x(t) + hv\left(t+\frac{h}{2}\right),$$
$$v(t+h) = v\left(t+\frac{h}{2}\right) + \frac{h}{2}\frac{F(x(t+h))}{m}$$
$$v\left(t+\frac{3}{2}h\right) = v\left(t+\frac{h}{2}\right) + h\frac{F(x(t+h))}{m}$$

Slide Type Sub-Slide

- It's a 2-variable leapfrog-based method, at half the cost of leapfrog
- If diagnostics (like energy) are needed at specific time steps, we need the half-step quantities

Slide Type

Spectral Methods

Slide Type Slide

For linear PDEs:

- Find yourself a set of orthogonal functions forming a basis of your function space
- Use transforms to express trial solution and its derivative(s) in this basis, with unknown coefficients
  - Remember, any linear combination of solutions is also itself a solution
- Use transforms to project initial conditions onto that basis, and use them to determine the coefficients
- Use inverse transforms to directly obtain the solution at any specified coordinates (e.g. at any time t, without stepping through all the previous time-steps)

Features:

- Particularly useful with large sets of coupled PDEs, for which just one elliptic PDE can be the main bottleneck of a non-spectral implementation
- Very accurate
- But inflexible when it comes to domain shape.

Basis of function space

Slide Type Sub-Slide

Examples of "set of orthogonal functions forming a basis of your function space":

- $\sin(n\pi x/L)$, $n \in \mathbb{N}$ if quantity is zero at boundaries (assuming $x = 0, L$ are the boundaries) or function is odd w.r.t. midline of domain (assuming $x = 0$ at midline),
- $\cos(n\pi x/L)$, $n \in \mathbb{N}$ if quantity has zero derivatives at boundaries (assuming $x = 0, L$ are the boundaries) or function is even w.r.t. midline of domain (assuming $x = 0$ at midline),
- $\exp(in\pi x/L)$, $n \in \mathbb{N}$ if quantity is periodic,
- Chebyshev polynomials for more flexible combinations of boundary conditions or non-periodic, closed domains,
- Hermite polynomials on the $(-\infty, \infty)$ real line,
- Laguerre polynomials on the $(0, \infty)$ real half-line

Slide Type Sub-Slide

We focus on $\sin/\cos/\exp$ bases, sometimes called "Fourier spectral methods"

- ⊖ large down-payment cost of computing FFTs
- ⊕ large return on investment: gives you the solution at any times without stepping through previous times
  - e.g. elliptic PDEs can be solved without the need of an iterative solver like relaxation method
- ⊕ numerical stability
- ⊖ difficult or impossible to implement in complicated geometries.
- ⊖ problematic for non-linear equations

Slide Type Sub-Slide

e.g. for elliptic PDEs:

$$\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} = \rho;$$

$$\begin{pmatrix}\phi\\ \rho\end{pmatrix} = \sum_i\sum_j \begin{pmatrix}\widehat{\phi}_{ij}\\ \widehat{\rho}_{ij}\end{pmatrix}\exp(i(k_i x + \ell_j y)),$$

$$\text{Use orthogonality to project } \Rightarrow \widehat{\phi}_{ij} = -\frac{\widehat{\rho}_{ij}}{k_i^2 + \ell_j^2}$$

and you are just one iFFT away from getting the solution ⇒ no need to use an iterative solver!

Practical implementation in sin/cos/exp basis

Slide Type Slide

$$f = \sum_{n=-\infty}^{\infty} \hat{f}_n \exp(ik_n x) \Rightarrow \frac{\partial f}{\partial x} = \sum_{n=-\infty}^{\infty} ik_n \hat{f}_n \exp(ik_n x),$$

or, in shorthand,

$$\frac{\partial f}{\partial x} \to ik_n\hat{f}_n, \quad \frac{\partial^2 f}{\partial x^2} \to -k_n^2\hat{f}_n$$

Next are a couple of examples of how to express functions and their derivatives in sin/cos/exp basis

Slide Typ

First derivative

$$f(x) = \exp\left(\frac{-(x - L/2)^2}{\Delta^2}\right)$$

Slide Typ

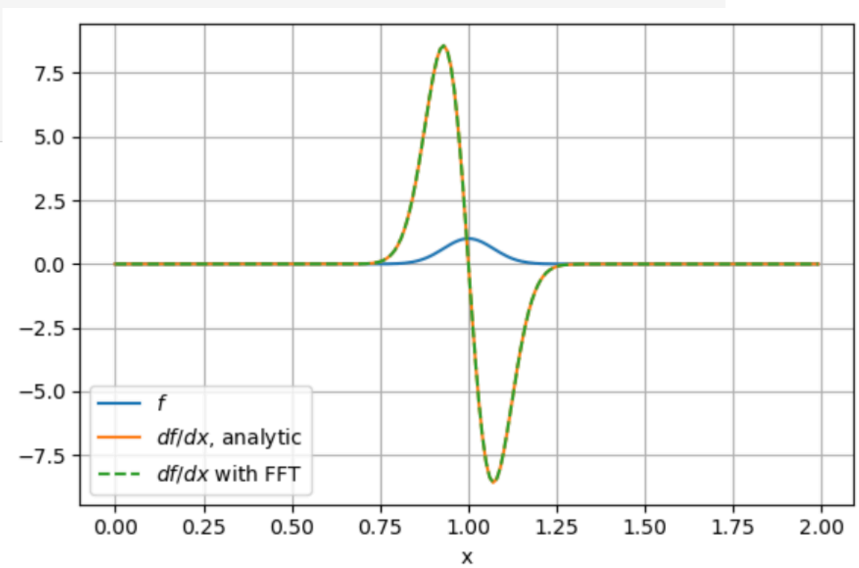
```
# Based on derivative_fft.py
import numpy as np
import matplotlib.pyplot as plt
from numpy.fft import rfft, irfft

#define function and its derivative
def f(x):
    return np.exp(-(x-L/2)**2/Delta**2)
def dfdx(x):
    return -2*(x-L/2)/Delta**2*np.exp(-(x-L/2)**2/Delta**2)

# define problem parameters
L = 2.0
Delta = 0.1
nx = 200

# define x, f(x), f'(x)
x = np.arange(0, L, L/nx)
farr = f(x)
farr_derivative = dfdx(x)
```

Slide Typ

```
# now do the same thing spectrally:
fhat = rfft(farr)  # fourier transform
karray = np.arange(nx/2+1)*2*np.pi/L  # define k
fhat_derivative = complex(0, 1)*karray*fhat  # define ik*fhat
f_derivative_fft = irfft(fhat_derivative)  # and transform back
```

Slide Typ

```
plt.figure(dpi=100)
plt.plot(x, farr, label='$f$')
plt.plot(x, farr_derivative, label='$df/dx$, analytic')
plt.plot(x, f_derivative_fft, '--', label='$df/dx$ with FFT')
plt.legend(loc=3)
plt.xlabel('x')
plt.grid()
plt.tight_layout()
plt.show()
```



Second derivative

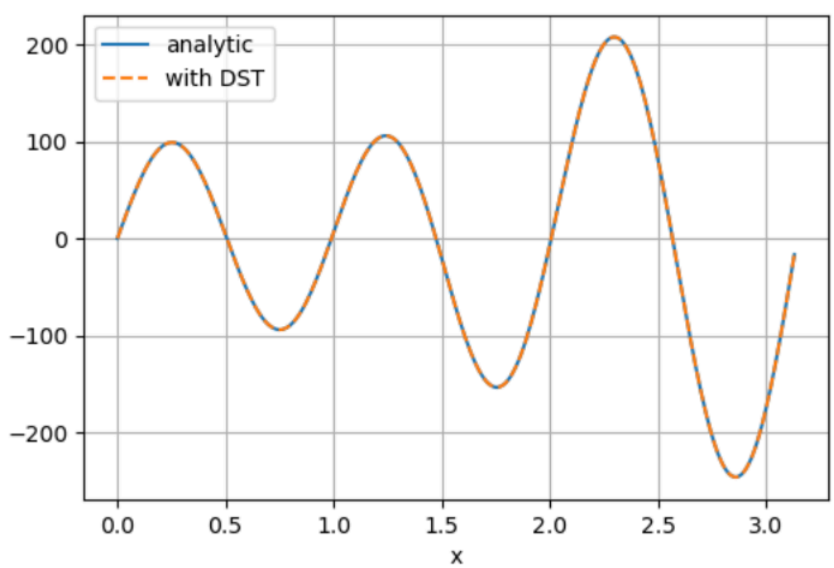
$$f = \sin(x) - 2\sin(4x) + 3\sin(5x) - 4\sin(6x)$$

```
import numpy as np
from dcst import dst, idst, dct, idct  # From Newman's dcst.py

N = 256
x = np.arange(N)*np.pi/N  # x = pi*n/N
farr = np.sin(x) - 2*np.sin(4*x) + 3*np.sin(5*x) - 4*np.sin(6*x)  # function is a sine series
fCoeffs = dst(farr)  # do fourier sine series
print('Original series: f = sin(x) - 2sin(4x) + 3sin(5x) - 4sin(6x)')
for j in range(7):
    print('Coefficient of sin({0}x): {1:.2e}'.format(j, fCoeffs[j]/N))
```

```
Original series: f = sin(x) - 2sin(4x) + 3sin(5x) - 4sin(6x)
Coefficient of sin(0x): 0.00e+00
Coefficient of sin(1x): 1.00e+00
Coefficient of sin(2x): -4.58e-17
Coefficient of sin(3x): -5.64e-16
Coefficient of sin(4x): -2.00e+00
Coefficient of sin(5x): 3.00e+00
Coefficient of sin(6x): -4.00e+00
```

```
# Below: 2nd derivative also a sine series
d2f_dx2_a = -np.sin(x) + 32*np.sin(4*x) - 75*np.sin(5*x) + 144*np.sin(6*x)

# 2nd derivative using Fourier transform
DerivativeCoeffs = -np.arange(N)**2*fCoeffs
d2f_dx2_b = idst(DerivativeCoeffs)
```

```
plt.figure(dpi=100)
plt.plot(x, d2f_dx2_a, label='analytic')
plt.plot(x, d2f_dx2_b, '--', label='with DST')
plt.xlabel('x')
plt.legend()
plt.grid()
plt.show()
```



Slide Type

Boundary Value Problems

Slide Type

A combination of ODE-solving and root-finding

Slide Type

Shooting method

Slide Type Slide

- Suppose we want to choose an initial velocity v_0 for a projectile to land after a certain elapsed time (e.g. $t_L = 10$ s), where the projectile obeys Newton's 2nd Law:
$$\frac{d^2x}{dt^2} = \frac{F(x)}{m} \quad \Rightarrow \quad \frac{dx}{dt} = v \quad \text{and} \quad \frac{dv}{dt} = \frac{F(x)}{m}.$$
- $x(v_0, t)$ is a nonlinear function of v_0, and finding $x(v_0, t = t_L)$ can be done using root finding method (binary search, secant...)
- **Shooting method:** integrate the equations and adjust v_0 until you locate root.

Slide Type Sub-Slide

Newman fig. 8.11

Slide Type

Eigenvalue Problems

Slide Type Sub-Slide

$$-\frac{\hbar}{2m}\frac{d^2\psi}{dx^2} + V(x)\psi = E\psi,$$
$$\psi(x = 0) = \psi(x = L) = 0.$$

- Shooting method does not work for finding wavefunctions that satisfy two boundary conditions, as in QM square well, except for valid eigenvalues E.
- So for these problems, eigenvalue E is the parameter that must be varied, instead of the boundary values (e.g. slope of ψ at the leftmost end or rightmost end of the domain)