PHY407 – University of Toronto
Lecture 9: Partial Differential Equations, part 2/2

Nicolas Grisouard, nicolas.grisouard@utoronto.ca

16 November 2020

Supporting textbook chapters for week 8: §§ 9.3.3, 9.3.4

Lecture 9, topics: * Stability, * Implicit and Crank-Nicholson methods, * Spectral methods.

1 Reminders

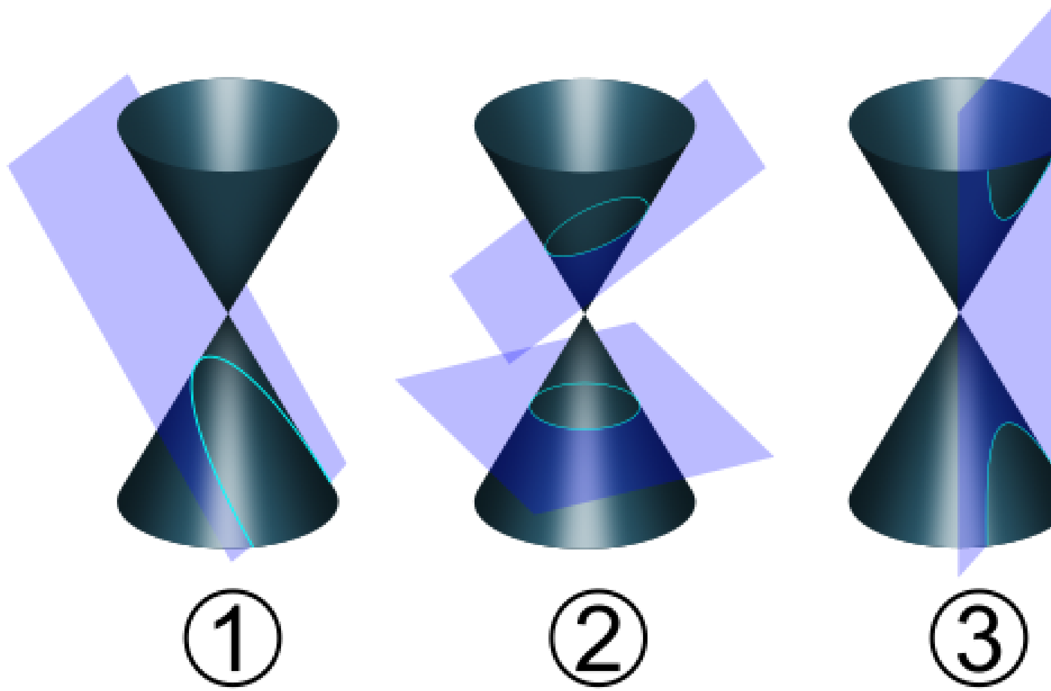
1.1 Classifying PDEs

$$\alpha\frac{\partial^2\phi}{\partial x^2} + \beta\frac{\partial^2\phi}{\partial x\partial y} + \gamma\frac{\partial^2\phi}{\partial y^2} + \delta\frac{\partial\phi}{\partial x} + \varepsilon\frac{\partial\phi}{\partial y} = f.$$

Classification based on

$$\Delta = \beta^2 - 4\alpha\gamma.$$

1. $\Delta = 0$: parabolic PDE,
2. $\Delta < 0$: elliptic PDE,
3. $\Delta > 0$: hyperbolic PDE.



1. Canonical parabolic PDE: the diffusion equation, $\kappa\nabla^2\phi - \frac{\partial T}{\partial t} = 0$,
2. Canonical elliptic PDE: the Poisson equation, $\nabla^2\phi = \rho$,
3. Canonical hyperbolic PDE: the wave equation, $c^2\nabla^2\phi - \frac{\partial^2 T}{\partial t^2} = 0$.

1.2 Calculating the second derivative

- Recall central difference calculation of 2nd derivative (§5.10.5):

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} - \frac{1}{12}h^2 f^{(4)}(x) + \ldots$$

1.3 General approach

- Discretize system spatially and temporally. Can use
  - Finite difference
  - Spectral coefficients
  - Etc.
- $\Rightarrow$ set of coupled ODEs that you need to solve in an efficient way.
- Spatial derivatives bring information in from neighbouring points $\Rightarrow$ coupling,
- $\Rightarrow$ errors depend on space and time and can get wave-like characteristics.

- Elliptical equations (e.g., Poisson eqn.):
  - Jacobi relaxation (always stable),
  - Speed-up with overrelaxation (not always stable),
  - Gauss-Seidel (overrelaxed or not): replace on the fly; more stable than Jacobi when overrelaxing.

- Parabolic PDEs (e.g., heat eqn):
  - FTCS (Forward Time, Centred Space) scheme: centred finite-diff. in space, forward Euler in time
  - Von Neumann analysis says stable if sufficient resolution space.
- Hyperbolic PDEs (e.g., wave eqn.):
  - Von Neumann analysis says FTCS never stable.
  - See next week for better schemes.
- Von Neumann stability analysis: plug a Fourier mode, see if it grows or not.

Today: stable and accurate schemes for Hyperbolic PDEs?

2 The implicit method

We have other choices on how to discretize in time the set of ODEs

$$\frac{d\phi_m}{dt} = \psi_m, \quad \text{and} \quad \frac{d\psi_m}{dt} = \frac{c^2}{a^2}\left(\phi_{m+1} - 2\phi_m + \phi_{m-1}\right)$$

What if we evaluated the RHS at time $t + h$ instead of t?

Explicit method we saw last time was

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

To compute with the implicit method, * first do $h \to -h$:

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

- Then, $n \to n+1$ (one shift forward in time):

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

  or

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

Why would we compute n if we know $n+1$?

Because all RHS's combined (i.e., at all n's) form is a matrix expression that we can invert to get the LHS.

2.1 Stability

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

If we do the Von Neumann substitution, $\widehat{\phi}_k^n \exp(ikma)$ and $\widehat{\psi}_k^n \exp(ikma)$, we get

$$\mathbf{B} \begin{bmatrix} \widehat{\phi}_k^{m+1} \\ \widehat{\psi}_k^{m+1} \end{bmatrix} = \begin{bmatrix} \widehat{\phi}_k^m \\ \widehat{\psi}_k^m \end{bmatrix} \exp(ikma),$$

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

$$\Rightarrow \begin{bmatrix} \widehat{\phi}_k^{m+1} \\ \widehat{\psi}_k^{m+1} \end{bmatrix} = \mathbf{B}^{-1} \begin{bmatrix} \widehat{\phi}_k^m \\ \widehat{\psi}_k^m \end{bmatrix} \exp(ikma).$$

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

[2]:

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

[3]:
```
# inverse of B
B**-1
```

[3]:

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

[4]:
```
# eigenvalues as a list
L = list((B**-1).eigenvals().keys())
```

[5]:
```
# First eigenvalue
L[0].factor()
```

[5]:

$$-\frac{ihr-1}{h^2r^2+1}$$

[6]:
```
# Magnitude of first eigenvalue
abs(L[0].factor())
```

[6]:

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

[7]:
```
# Magnitude of 2nd eigenvalue
abs(L[1].factor())
```

[7]: $\frac{1}{\sqrt{h^2r^2+1}}$

$$\begin{bmatrix}\widehat{\phi}_k^{m+1}\\ \widehat{\psi}_k^{m+1}\end{bmatrix} = \mathbf{B}^{-1}\begin{bmatrix}\widehat{\phi}_k^{m}\\ \widehat{\psi}_k^{m}\end{bmatrix}\exp(ikma).$$

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

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

- The eigenvalues are the growth factors and these are less than one.
- So the implicit method is unconditionally stable.
- But solutions decay exponentially! This is a big problem for the wave equation!

3 Crank-Nicholson

Crank-Nicholson: average of explicit (fwd Euler) and implicit methods

Euler:

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

Implicit:

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

Crank-Nicholson (C-N):

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

If we do the Von Neumann substitution, $\widehat{\phi}_k^n\exp(ikma)$ and $\widehat{\psi}_k^n\exp(ikma)$, 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

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

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

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

- For Euler-Forward, the growth factors are greater than one and the solution diverges.
- For Implicit, the growth factors are less than one and the solution decays to zero.
- For CN, the growth factors are one so the solution neither grows nor decays.
- It is also 2nd-order accurate in time, while both explicit and implicit methods are 1st-order accurate.

4 Spectral methods

4.1 General idea

- Find yourself a set of orthogonal functions forming a basis of your function space
  - $\sin$ if quantity is zero at boundaries or function is odd w.r.t. midline of domain,
  - $\cos$ if quantity has zero derivatives at boundaries or function is even w.r.t. midline of domain,
  - $\exp$ 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,
  - ...
- Project your initial conditions and forcing on that basis,
- Iterate in time for linear PDEs,
- Iterate in time, and do FFTs and iFFTs to compute the non-linear terms if PDEs are non-linear,

- We focus on $\sin/\cos/\exp$ bases, sometimes called "Fourier spectral methods" (*perhaps only by me*),
- Usually, all of these methods require computing FFTs (even for non-Fourier spectral methods),
- $\oplus$ FFTs: large cost of computing them, but a large return on investment usually:
  - linear PDEs: all modes oscillate independently, without coupling $\Rightarrow$ computing the FFTs of the initial conditions give you the solutions at all times,
  - non-linear PDEs: elliptic PDEs can be solved without the need of an

iterative solver like relaxation method

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

$$\begin{pmatrix}\phi\\ \rho\end{pmatrix}=\sum_i\sum_j\begin{pmatrix}\widehat{\phi}_{ij}\\ \widehat{\rho}_{ij}\end{pmatrix}\exp i(k_ix+l_jy),$$

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

and you are just one iFFT away form getting the solution $\Rightarrow$ no need to use an iterative solver! This is particularly useful with large sets of coupled PDEs, for which just one elliptic PDE can be the main bottleneck of a non-spectral implementation.

- $\ominus$ Spectral methods are really difficult to implement in complicated geometries.

5 Practical implementation tricks

Periodic BCs are simpler, let's focus on them.

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

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 fo examples of how to implement it.

[9]:
```
# Based on derivative_fft.py
# calculate derivative of a function using fourier transform method
# import required routines
from numpy import arange, exp, pi
from pylab import plot, legend, show, subplot, xlabel, ylabel, tight_layout
from numpy.fft import rfft, irfft
#define function and its derivative
def f(x):
    return exp(-(x-L/2)**2/Delta**2)
def dfdx(x):
    return exp(-(x-L/2)**2/Delta**2)*-2*(x-L/2)/Delta**2

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

# define x, f(x), f'(x)
x=arange(0,L,L/nx)
```

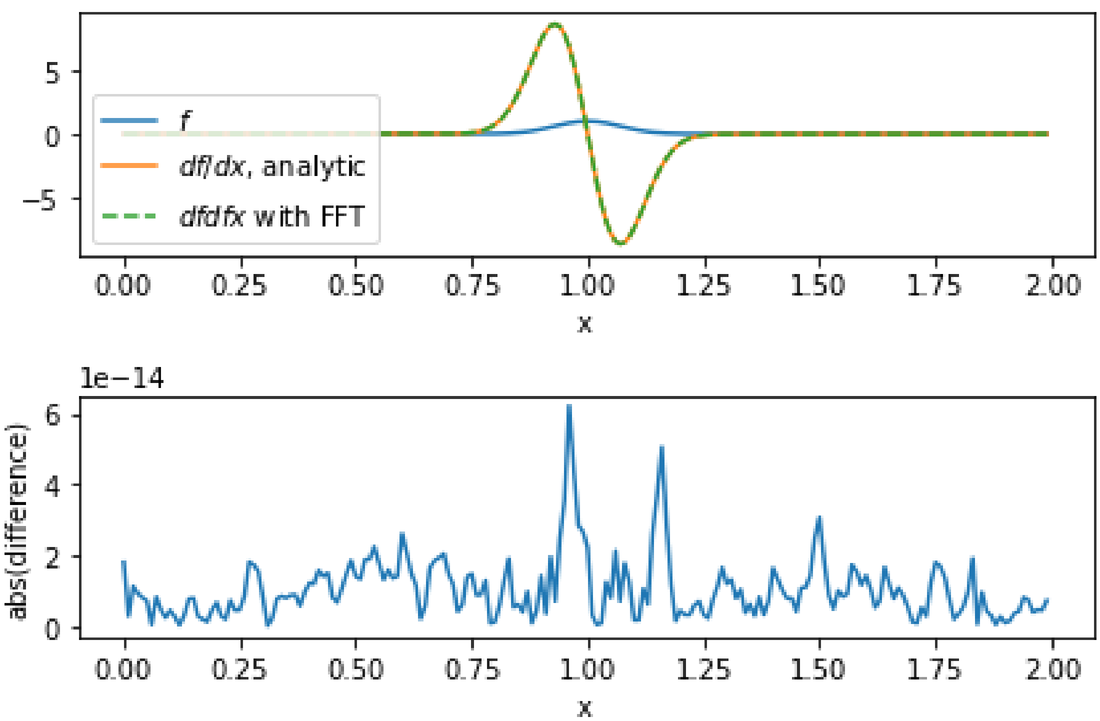
```
f = f(x)

f_derivative = dfdx(x)

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

subplot(2,1,1)
plot(x, f, label='$f$')
plot(x, f_derivative, label='$df/dx$, analytic')
plot(x, f_derivative_fft, '--', label='$dfdfx$ with FFT')
legend(loc=3)
xlabel('x')
subplot(2,1,2)
plot(x,abs(f_derivative-f_derivative_fft))
xlabel('x')
ylabel('abs(difference)')

tight_layout()
show()
```



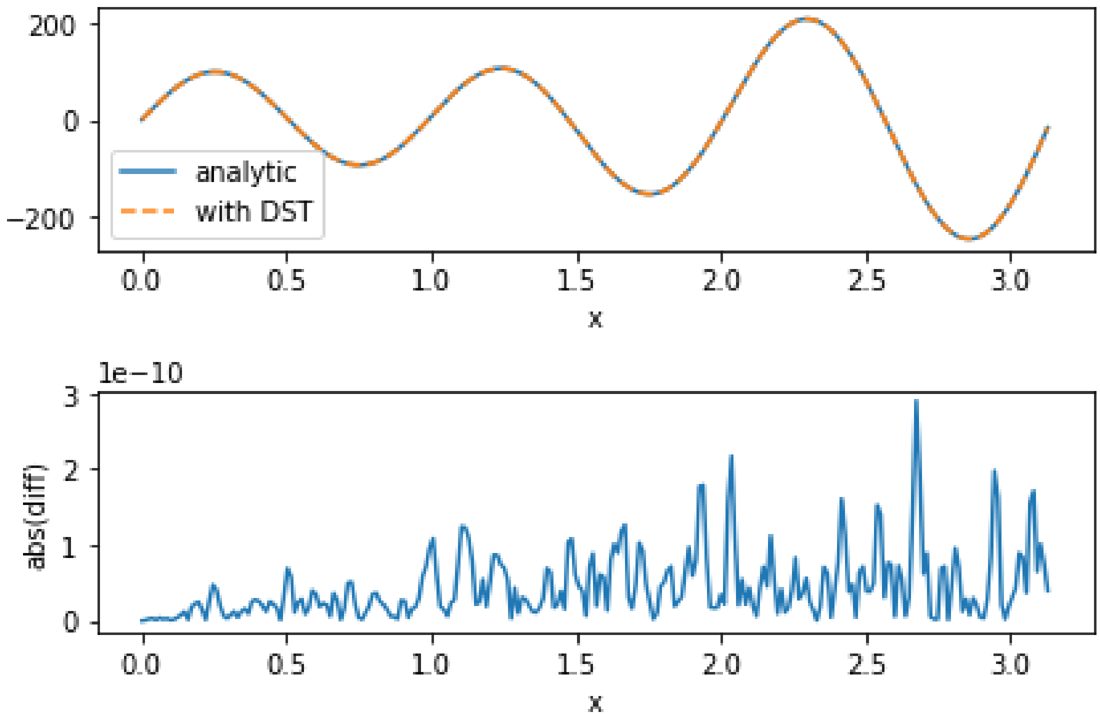
[10]:
```python
from pylab import plot, show, legend, subplot, xlabel, ylabel, tight_layout
from numpy import zeros, empty, linspace, exp, arange, minimum, pi, sin, cos,␣
↪array
from dcst import dst, idst, dct, idct

N = 256
x = arange(N)*pi/N  # x = pi*n/N
f = sin(x) - 2*sin(4*x) + 3*sin(5*x) - 4*sin(6*x)  # function is a sine series
fCoeffs = dst(f)  # 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))

print('See Figure for calculating second derivative')

# Below: 2nd derivative also a sine series
d2f_dx2_a = -sin(x) + 32*sin(4*x) - 75*sin(5*x) + 144*sin(6*x)
DerivativeCoeffs = -arange(N)**2*fCoeffs  # 2nd derivative using Fourier␣
↪transform
d2f_dx2_b = idst(DerivativeCoeffs)
subplot(2, 1, 1)
plot(x, d2f_dx2_a, label='analytic')
plot(x, d2f_dx2_b, '--', label='with DST')
xlabel('x')
legend()
subplot(2, 1, 2)
plot(x,abs(d2f_dx2_a - d2f_dx2_b))
xlabel('x')
ylabel('abs(diff)')
tight_layout()
show()
```

```
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): 3.67e-17
Coefficient of sin(3x): -5.96e-16
Coefficient of sin(4x): -2.00e+00
Coefficient of sin(5x): 3.00e+00
Coefficient of sin(6x): -4.00e+00
See Figure for calculating second derivative
```



6 Summary

- Last week: FTCS was
  - and explicit scheme,
  - unstable for hyperbolic PDEs (wave eqn.)
- FTCS with implicit time stepping:
  - infers what RHS of next step is based on present step, and inverts.
  - stable for hyperbnolic PDEs, but decays (bad accuracy)
- Crank-Nicholson:
  - average of both, also requires matrix inversion,
  - neither grows nor decays
- Spectral methods:
  - leverage $\partial_x f \to ik \times \hat{f}$ and powerful FFT methods,
  - can be much faster than grid-based schemes (though it depends),
  - super-duper accurate
  - Not too flexible when it comes to domain shape.