PHYS 325 Scientific Computing — Fall 2018

Solutions Exam 3

Problem 1: General questions

a) Applications of Fourier transforms include signal processing, data smoothing, solving differential equations, data analysis, etc. (see lecture notes)

b) The numerical derivative, defined e.g. as

$$\frac{f(x+\Delta x)-f(x)}{\Delta x}$$

suffers from both a discretization error which increases with increasing Δx, as well as a roundoff error, which increases with decreasing Δx (because we need to subtract two very similar numbers from each other). We cannot reduce both errors simultaneously by changing Δx.

c) `x = numpy.arange(3,17,2)` has elements: 3, 5, 7, 9, 11, 13, 15
`x[-1]` returns 15
`x[2:5]` returns 7, 9, 11
`x[5:]` returns 13, 15

d) The output of the program is 11, because we add $2+3+6$ via the loops.

Problem 2: Fourier transforms

a) $\omega_k = \frac{2\pi}{\Delta N}k = \frac{2\pi}{T}k$

b) Increasing the time resolution leads to an increase in the frequency **range**, since $\omega_{\max} \propto N/T = 1/\Delta$. Increasing the time interval length leads to an increase in the frequency **resolution**, since $\Delta\omega = 2\pi/T$.

c) There will be two equally high sharp peaks in the Fourier transform located at frequencies a and b (on the imaginary axis). Since the signal is real there will also be mirrored peaks at the corresponding negative frequencies.

d) After performing the Fourier transform, set part of the Fourier components (those at higher frequencies, keeping the components close to zero frequency) to zero and then Fourier transform back. The more components one sets to zero the stronger the smoothing.

e) FFT scales like $\mathcal{O}(N \log N)$

Problem 3: Ordinary differential equations

a) Define $I = dQ/dt$ (the current). Then

$$\frac{dI}{dt} = -\frac{R}{L}I - \frac{Q}{CL} \tag{1}$$

$$\frac{dQ}{dt} = I \tag{2}$$

b) See lecture notes. Possible examples include radioactive decay, Schrödinger equation, Newton's equations, wave equation, etc.

c) Euler's method is based on the Taylor Series expansion for x around some time t_0:

$$x(t_0+\Delta t) = x(t_0) + \left.\frac{dx(t)}{dt}\right|_{t=t_0}\Delta t + \mathcal{O}(\Delta t^2)$$

Substitute the differential equation $\frac{dx}{dt} = f$ to get

$$x(t_0+\Delta t) = x(t_0) + \Delta t f(x(t_0), t_0)$$

- start with the initial condition $x(t_0) = x_0$
- repeatedly apply the above formula from the Taylor expansion
- each time replace $x(t)$ with the new value of $x(t + \Delta t)$
- each step is accurate to $\mathcal{O}(\Delta t^2)$, but the total error is $\mathcal{O}(\Delta t)$ because error accumulate

d) The idea behind adaptive methods is to spend more computational time on regions where the ODE function changes a lot and less time on regions where the function is essentially flat. This is crucial for numerical efficiency and stability. In practice this is a two part process: first we need to estimate the error on each step and then adjust the step size if necessary. For this we need to take each step twice, typically once as a full step and then independently as two half steps. The difference between the respective results gives us an estimate on the error, which we can compare with our accuracy goal. If the goal is not met we need to repeat again with an even smaller step. In addition we adjust the step size for the next step, depending on how close we were to the accuracy goal. We need to be careful not to increase the step too much because sometimes the results for the two different step sizes agree by chance.

Problem 4: Runge-Kutta method with Python

```
import numpy as np
import matplotlib.pyplot as plt

def f(x,t):
    return -x**3 + np.sin(t)

start = 0.0
end = 10.0          # or some other value, but needs to be defined
numSteps = 200
stepSize = (end-start)/numSteps

x = 2.0
tpoints = np.arange(start, end, stepSize)
xpoints = []

# Runge-Kutta
for t in tpoints:
    xpoints.append(x)
    k1 = stepSize*f(x,t)
    k2 = stepSize*f(x+0.5*k1, t+0.5*stepSize)
    x += k2

# Make plots
plt.plot(tpoints,xpoints)
plt.xlabel("t")
plt.ylabel("x(t)")
plt.show()
```

The errors were corrected in the code in red:

- incorrect syntax in the definition of `f(x,t)`

- end time needs to be defined
- incorrect initial condition
- incorrect definition of `tpoints`
- typo in the last line of the Runge-Kutta implementation
- need to plot the solution `xpoints` versus the correct times given by `tpoints`
- quotation marks missing around ylabel

Problem 5: Random numbers

a) Pseudorandom numbers are faster, less expensive to generate, and allow to reproduce the exact sequence of numbers by choosing the same random seed.
b) See lecture notes on good random number practices.
c) $y_i = 4x_i + 4$
d) You can use either the transformation method or the rejection method. Both are described in the lecture notes for this distribution.

Problem 6: Monte Carlo

a) The error scales like $1/\sqrt{N}$
b) For example in higher dimensions, since the error scaling is independent of dimension, while for deterministic methods the scaling gets worse: $1/N^{a/d}$ where the constant a depends on the method (for example, $a = 4$ for Simpson's rule) and d is the number of dimensions. Also very strongly oscillating functions are often difficult to integrate with deterministic methods.
c) Example for the calculation of π is in the lecture notes.
d) Ising model example is also in the lecture notes.

Extra credit

a) For small values of x the value of y approaches $1 - 1 = 0$. Subtraction of two similar numbers suffers from large roundoff errors.
b) $\frac{\delta f}{f} = \sqrt{\left(\frac{\delta x}{x}\right)^2 + \left(\frac{\delta y}{y}\right)^2}$ or $\left|\frac{\delta x}{x}\right| + \left|\frac{\delta y}{y}\right|$ depending on the method you use.
$\delta f = \delta x e^{-x}$
c) Q is the probability that we would have obtained the measured data from the best fit function. The higher this probability the better we deem our fit. Typically we would like values $\gtrsim 0.1$, but we might accept fits with smaller values of Q. Fits with $Q \lesssim 0.001$ should certainly be rejected.
d) We typically have many more data points than fit parameters. Also there is statistical noise in the data, which means many different fits may be almost equally suitable to describe the data. After calculating the singular values d_i, we need to check which ones are too small, typically $d_i < 10^{-12}\max(d_i)$. For these we set $1/d_i = 0$, which means that they will not contribute to the best fit solution.
e) The integration error is of $\mathcal{O}(\Delta x^4)$.
f) The quoted formula contains a telescoping sum. One should always group identical terms together, like this:

$$\int_a^b f(x)dx \approx \Delta x \left(\frac{f(a)}{2} + \frac{f(b)}{2} + \sum_{i=1}^{N-1} f(x_i) \right)$$