PH2150 Scientific Computing Skills

Andrew Casey

October 29, 2012

1 Exercise Wk5Ex1:

Last week in PH2130 Mathematical Methods you were introduced to the concept of a *fourier series*, the expansion of an arbitrary periodic function f(x) as a linear combination of sines and cosines, or in an exponential form. In sine and cosine form this takes the expression:

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty}[a_n \cos(\frac{n\pi x}{L}) + b_n \sin(\frac{n\pi x}{L})]$$

a) Write a user defined function to sum the values up to the n^{th} term of a fourier series to approximate a square wave of the form:

$$f(x) = \begin{cases} -1 & \pi \leq x \leq 2\pi \\ 1 & 0 \leq x < \pi \end{cases}$$

b) Modify your program to include a function to sum the values up to the n^{th} term of a fourier series to approximate a saw-tooth wave of the form:

$$f(x) = x, \qquad 0 \leq x \leq \pi$$

Plot both functions for the sum of n up to 9, 99 and 999 as two subplots on the same figure, see example figure 1. *NOTE: to observe the Gibbs Phenomena you must have enough resolution in x.*

BONUS Marks: Animate a plot of the square wave so that it shows the evolution of the fourier series as more elements are added up to n=20.

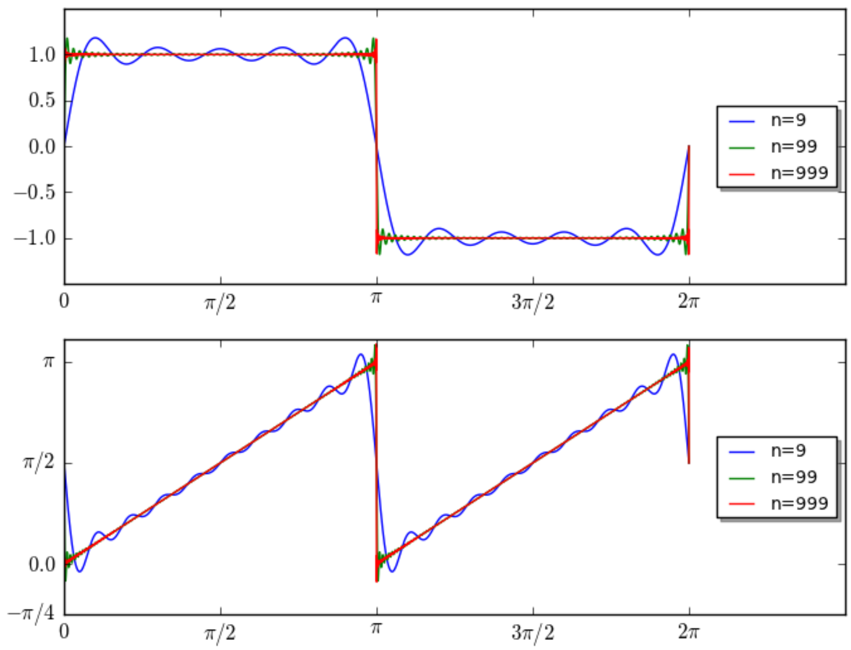

Figure 1: Output of Fourier series python code for square and sawtooth

2 Exercise Wk5Ex2:

The code curvefit.py demonstrates the *scipy* function *curve_fit()*, in this example data is generated with the functional form: $y = a\exp(-b * x) + c$ with the addition of a noise term. A curve fitting routine based on a non-linear least squares fit using a Levenburg-Marquardt algorithm returns the fitting parameters popt. pcov returns a 2D array which is the estimated covariance of popt. The diagonals provide the variance of the parameter estimate. The syntax for curve_fit and other commonly used optimisation algorithms can be found here:

http://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html

a) Run the example curvefit.py shown below, record the fit values obtained.

b) Modify the program curvefit.py such that for each point it calculates a term *yerr = sqrt(abs(yn-y))* add this to the plot as a set of error bars, use the *yerr* array as an argument for sigma to weight the least squares fit.*[curve_fit(f, xdata, ydata, p0=None, sigma=None, **kw)]* Compare the result of the fit with that of part a.

c) The data in file fitting_week5.dat on the moodle page, is of the functional form $y = a\sin(bx)$, write a program to find a and b. Plot the data, along with your best fit result.

```
# curvefit.py week 5 example
import numpy as np
from scipy.optimize import curve_fit
from pylab import *
import matplotlib.pyplot as plt

def func(x, a, b, c):
    return a*np.exp(-b*x) + c # function to generate data for curve fit
x = np.linspace(0,4,50)
y = func(x, 2.5, 1.3, 0.5)
yn = y + 0.2*np.random.normal(size=len(x)) # adding some noise to the data points
popt, pcov = curve_fit(func, x, yn) # performing curve fit, and returning parameters
print 'Parameters : ', popt
print 'Covariance : ', pcov
# graphical output of results
fig=plt.figure()
plt.scatter(x,y, label='data')
plt.plot(x,yn, label='data + noise')
plt.plot(x, func(x,popt[0],popt[1],popt[2]), label='best fit')
plt.legend()
plt.show()
```

curvefit.py