Ay 190 Assignment 2

Differentiation and Interpolation

January 15, 2013

1 In-Class Exercise 1

Finite difference approximations of the deriative of the function $f(x) = x^3 - 5x^2 + x$ on the interval $[-2, 6]$.

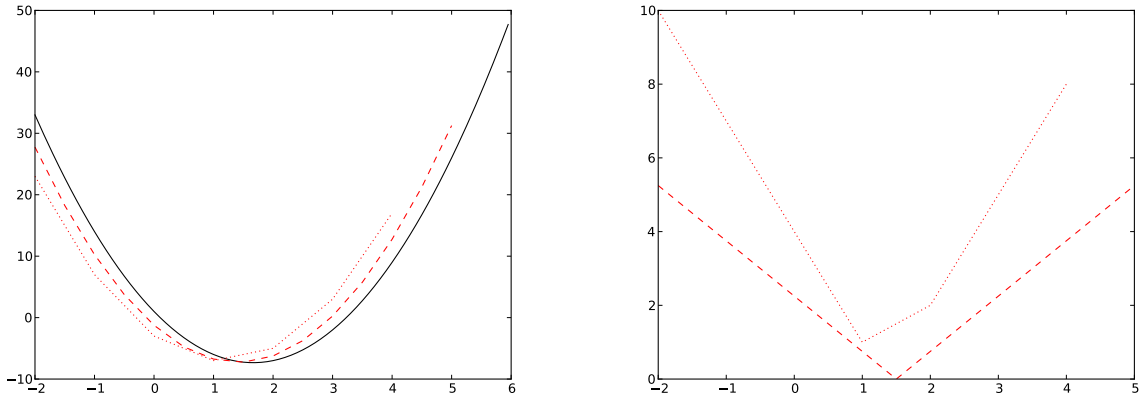

Figure 1: Approximation using the forward difference method. On the right is the exact derivative, overplotted with two approximations using the forward difference method with step size $h_1 = 1$ (dotted line) and $h_2 = 0.5$ (dashed line). The absolute error for both approximations is plotted on the left. The error decreases by a factor of 2 when the step size is halved by 2, demonstrating first order convergence.

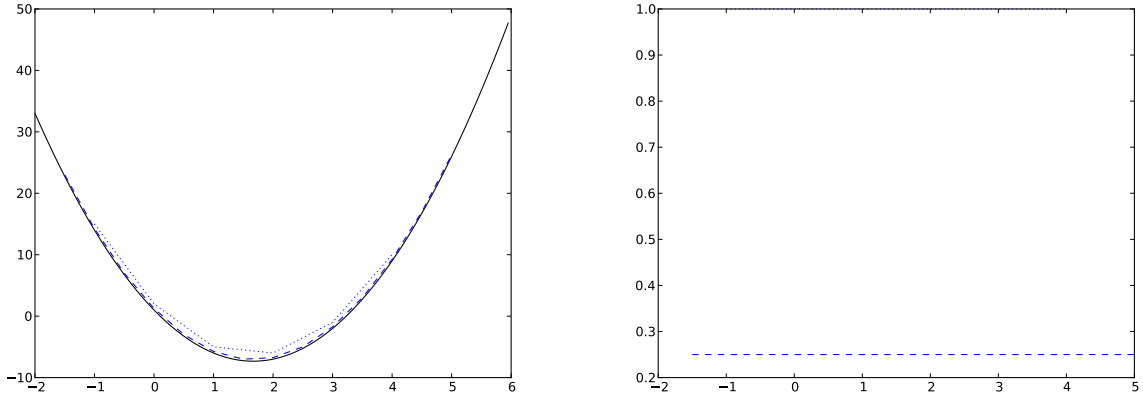

Figure 2: The same as in Figure 1, except for the central difference method. The error decreases by a factor of 4 when the step size is halved, demonstrating second order convergence.

2 In-Class Exercise 2

Interpolation of a Cepheid lightcurve using the Lagarange, piecewise linear, and piecewise quadratic methods.

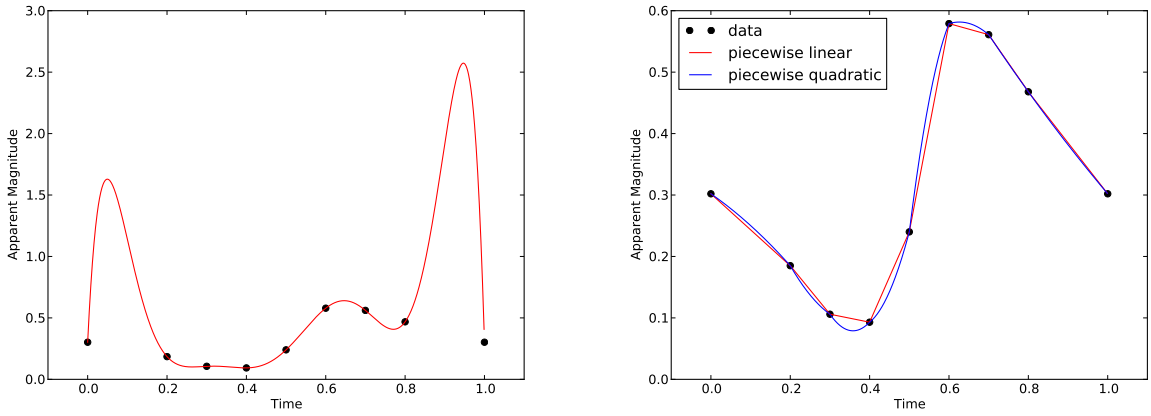

Figure 3: Data is plotted in black. On the right is the approximation to the continuous lightcurve using Lagrange interpolation, and on the left, piecewise linear and quadratic interpolations.

3 Homework Exercise 2

A second-order central finite difference approximation for the second derivative of a function $f(x)$, assuming a fixed step size h can be derived like the central finite difference approximation for the first derivative:

$$f''(x_0) = \frac{f'(x_0 + \frac{h}{2}) - f'(x_0 - \frac{h}{2})}{h} + \mathcal{O}(h^2) \tag{1}$$

$$= \frac{f(x_0 + h) - f(x_0)}{h^2} - \frac{f(x_0) - f(x_0 - h)}{h^2} + \mathcal{O}(h^2) \tag{2}$$

$$= \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2} + \mathcal{O}(h^2) \tag{3}$$

4 Homework Exercise 3

Interpolation of a Cepheid lightcurve using the cubic Hermite and cubic spline methods.

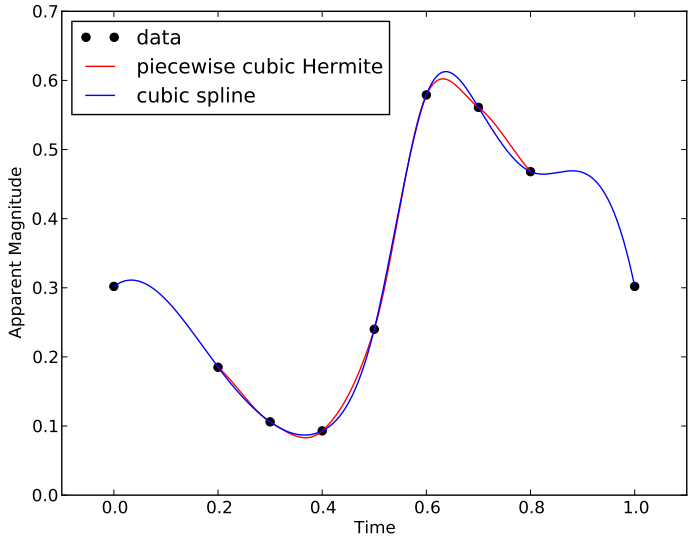

Figure 4: Piecewise cubic Hermite and cubic spline interpolation of the lightcurve. The derivative of the lightcurve, required for piecewise cubic Hermite interpolation, was calculated using the central differencing method. As the derivatives cannot be calculated for the endpoints under the central differencing method, the lightcurve was only interpolated between the remaining points.

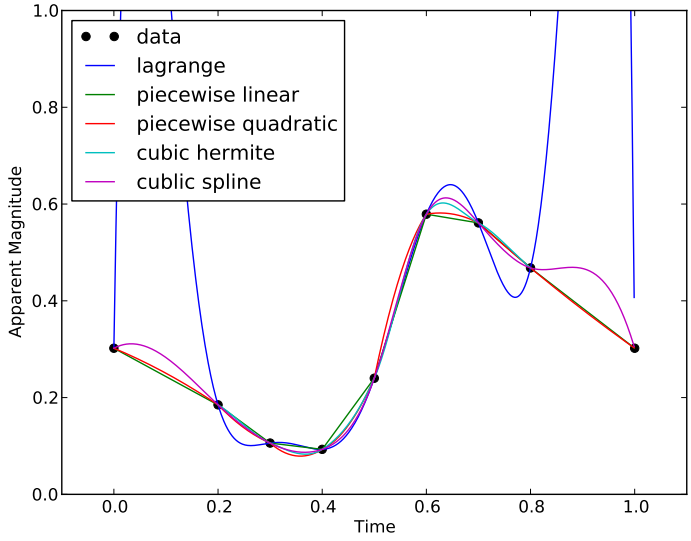

Figure 5: Results from all interpolation methods together. Cubic spline, followed by piecewise linear and quadratic interpolation produce the closest fits to the data. Piecewise cubic Hermite and particularly Lagrangian interpolation produce spurious bumps near the endpoints of the dataset.

Appendices

The Python modules and scripts used in this assignment include

A The Interpolation Routines: `interp.py` (see pages 4-5)

B Script for the assignment: `set2.py` (see pages 6-8)

A The Interpolation Routines: interp.py

```
# interp.py
# created 1/14/12 by stacy kim
#
# Contains a suite of interpolation routines.
#
# NOTE: In all of the folowing routines, x is an array of values to interpolate the
#  discretized function given by arrays xdat, ydat.

import sys, math
import numpy as np
from operator import mul


def lagrange(xdat, ydat, x):
    """Lagrange interpolation."""
    xdat=np.array(xdat,dtype=np.float)
    ydat=np.array(ydat,dtype=np.float)

    return [sum([ydat[j]*reduce(mul,[(xx-xk)/(xdat[j]-xk) for xk in np.concatenate((xdat[:j],xdat[j+1:]))])
                 for j in range(len(xdat))]) for xx in x]


def pw_linear(xdat, ydat, x):
    """Piecewise linear interpolation (for non-piecewise, supply 2 data pts)."""
    # sort data (x,y) pairs by xdat values
    a=np.array([xdat,ydat],dtype=np.float)
    dat=a.T[a.T[:,0].argsort()]
    xsrt,ysrt=dat[:,0],dat[:,1]

    # interpolate at given locations x
    y=[] # interpolated values
    for xx in x:
        for i in range(len(xsrt)):
            if xx < xsrt[i]: break
        i-=1
        y.append(lagrange(xsrt[i:i+2],ysrt[i:i+2],[xx]))
        #y.append(ysrt[i]+(ysrt[i+1]-ysrt[i])/(xsrt[i+1]-xsrt[i])*(xx-xsrt[i]))

    return y


def pw_quadratic(xdat, ydat, x):
    """Piecewise quadratic interpolation (for non-piecewise, supply 3 data pts)."""
```

```
    # sort data (x,y) pairs by xdat values
    a=np.array([xdat,ydat],dtype=np.float)
    dat=a.T[a.T[:,0].argsort()]
    xsrt,ysrt=dat[:,0],dat[:,1]

    # interpolate at given locations x
    y=[] # interpolated values
    print len(x)
    for xx in x:
        for i in range(len(xsrt)-1):
            if xx< xsrt[i]: break
        i-= 1
        y.append(lagrange(xsrt[i:i+3],ysrt[i:i+3],[xx]))

    return y


def cubic_hermite(xdat, ydat, ypdat, x):
    """Piecewise cubic Hermite interpolation."""
    psi0 = lambda z: 2*z**3-3*z**2+1
    psi1 = lambda z: z**3-2*z**2+z

    # sort data (x,y,yp) triples by xdat values
    a=np.array([xdat,ydat,ypdat],dtype=np.float)
    dat=a.T[a.T[:,0].argsort()]
    xsrt,ysrt,ypsrt=dat[:,0],dat[:,1],dat[:,2]

    # interpolate at given locations x
    y=[] # interpolated values
    for xx in x:
        for i in range(len(xsrt)-1):
            if xx < xsrt[i]:
                i-=1
                break
        z=(xx-xsrt[i])/(xsrt[i+1]-xsrt[i])
        y.append(ysrt[i]*psi0(z)+ysrt[i+1]*psi0(1-z)
                 +ypsrt[i]*(xsrt[i+1]-xsrt[i])*psi1(z)
                 -ypsrt[i+1]*(xsrt[i+1]-xsrt[i])*psi1(1-z))

    return y
```

B Script for the assignment: set2.py

```
# set2.py
# created 1/14/12 by stacy kim

import interp
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d

# Cephieid Lighcurve data
xdat=[0.0,0.2,0.3,0.4,0.5,0.6,0.7,0.8,1.0]
ydat=[0.302,0.185,0.106,0.093,0.24,0.579,0.561,0.468,0.302]


# IN-CLASS EXERCISE 1 ------------------------------------------------------------
# Finite difference approximation and convergence
h=1.
x1=np.arange(-2,6,h)
f1=x1**3-5*x1*x1+x1

# calculate first derivative
fp1=3*x1*x1-10*x1+1                                          # exact
fp_fd1=[(f1[i+1]-f1[i])/h for i in range(len(x1)-1)]          # forward diff, h
fp_cd1=[(f1[i+1]-f1[i-1])/(2*h) for i in range(1,len(x1)-1)] # central diff, h

# redo with reduced step size
h=.5
x2=np.arange(-2,6,h)
f2=x2**3-5*x2*x2+x2

fp2=3*x2*x2-10*x2+1                                          # exact
fp_fd2=[(f2[i+1]-f2[i])/h for i in range(len(x2)-1)]          # forward diff, h/2
fp_cd2=[(f2[i+1]-f2[i-1])/(2*h) for i in range(1,len(x2)-1)] # central diff, h/2

# plot approximations
x=np.arange(-2,6,0.05)
fp=3*x*x-10*x+1

plt.plot(x,fp,c='k')
plt.plot(x1[:-1],fp_fd1,c='r',ls=':')
plt.plot(x2[:-1],fp_fd2,c='r',ls='--')
plt.savefig('forward_diff.pdf')
plt.show()

plt.plot(x,fp,c='k')
plt.plot(x1[1:-1],fp_cd1,c='b',ls=':')
plt.plot(x2[1:-1],fp_cd2,c='b',ls='--')
plt.savefig('central_diff.pdf')
plt.show()

# convergence plots
plt.plot(x1[:-1],abs(fp_fd1-fp1[:-1]),c='r',ls=':')
```

```
plt.plot(x2[:-1],abs(fp_fd2-fp2[:-1]),c='r',ls='--')
plt.savefig('forward_diff_converg.pdf')
plt.show()

plt.plot(x1[1:-1],abs(fp_cd1-fp1[1:-1]),c='b',ls=':')
plt.plot(x2[1:-1],abs(fp_cd2-fp2[1:-1]),c='b',ls='--')
plt.savefig('central_diff_converg.pdf')
plt.show()


# IN-CLASS EXERCISE 2a ------------------------------------------------------
# Lagrange interpolation of Cepheid lightcurve
x=np.arange(0,1,0.001)
y_lgr=interp.lagrange(xdat,ydat,x)

# plot interpolated function and original data
plt.plot(xdat,ydat,'ko',mec='k')
plt.plot(x,y_lgr,c='r')
plt.xlim([-0.1,1.1])
plt.xlabel('Time')
plt.ylabel('Apparent Magnitude')
plt.savefig('cepheid_lagrange.pdf')
plt.show()

# IN-CLASS EXERCISE 2b ------------------------------------------------------
y_pwl=interp.pw_linear(xdat,ydat,x) # piecewise linear interp
y_pwq=interp.pw_quadratic(xdat,ydat,x) # piecewise quadratic

plt.plot(xdat,ydat,'ko',mec='k')
plt.plot(x,y_pwl,c='r')
plt.plot(x,y_pwq,c='b')
plt.legend(['data','piecewise linear','piecewise quadratic'],loc='best')
plt.xlim([-0.1,1.1])
plt.xlabel('Time')
plt.ylabel('Apparent Magnitude')
plt.savefig('cepheid_pwl_pwq.pdf')
plt.show()


# HOMEWORK EXERCISE 3 -------------------------------------------------------
# piecewise cubic Hermite
x_pwch=np.arange(0.2,0.8,0.001)
ypdat=[(ydat[i+1]-ydat[i-1])/(xdat[i+1]-xdat[i-1]) for i in range(1,len(xdat)-1)]
y_pwch=interp.cubic_hermite(xdat[1:-1],ydat[1:-1],ypdat,x_pwch)

# cubic spline
f_cs=interp1d(xdat,ydat,kind='cubic')

plt.plot(xdat,ydat,'ko',mec='k')
plt.plot(x_pwch,y_pwch,c='r')
plt.plot(x,f_cs(x),c='b')
plt.legend(['data','piecewise cubic Hermite','cubic spline'],loc='best')
plt.xlim([-0.1,1.1])
plt.xlabel('Time')
```

```
plt.ylabel('Apparent Magnitude')
plt.savefig('cepheid_pwch_cs.pdf')
plt.show()


# plot all results
plt.plot(xdat,ydat,'ko',mec='k')
plt.plot(x,y_lgr, x,y_pwl, x,y_pwq, x_pwch,y_pwch, x,f_cs(x))
plt.legend(['data','lagrange','piecewise linear','piecewise quadratic','cubic hermite','cublic spline']
plt.xlim([-0.1,1.1])
plt.ylim([0,1])
plt.xlabel('Time')
plt.ylabel('Apparent Magnitude')
plt.savefig('cepheid_all.pdf')
plt.show()
```