EE5471: Assignment on Spline Interpolation and Rational Interpolation

Harishankar Ramachandran

August 13, 2014

Linking C functions into python

The problem with numerical recipes code is that it uses nrutil.h. I have put a version of spline.c that does not require nrutil.h which you can use in weave.

Even so, use of spline is non-trivial. So here is an example code that uses spline.c and does interpolation. This can be downloaded as testspline.py

```
# script to test the spline routine
from scipy import *
from matplotlib.pyplot import *
from scipy import weave
# define support code
with open("spline.c","r") as f:
  scode=f.read()
h=logspace(-3,0,16)
# h=input("Enter h as a numpy vector: ")
N=(2.0*pi)/h
err=zeros(h.shape)
for i in range(len(h)):
  x=linspace(0,2*pi,N[i])
  y=sin(x)
  n=int(N[i])
  xx=linspace(0,2*pi,10*n+1)
  y2=zeros(x.size)
  u=zeros(x.size)
  yy=zeros(xx.size)
  code="""
  #include<math.h>
  int i;
  double xp;
  spline(x,y,n,cos(x[0]),cos(x[n-1]),y2,u);
  for(i=0;i<=10*n;i++){
    xp=xx[i];
    splint(x,y,y2,n,xp,yy+i);
  }
"""
  weave.inline(code,["x","y","n","y2","u","xx","yy"],support_code=scode,ext
```

```
  if i==0:
    figure(0)
    plot(x,y,'ro')
    plot(xx,yy,'g')
    title("Interpolated values and data points for n=%d" % N[i])
  z=(yy-sin(xx))
  err[i]=z.max()
figure(1)
loglog(h,err)
title("Error vs. spacing")
show()
```

The Assignment

We require a 6 digit accurate method to compute the function

$$f(x) = \frac{x^{1+J_0(x)}}{\sqrt{(1+100x^2)(1-x)}} = \frac{x^{1+J_0(x)}}{\sqrt{1-x+100x^2-100x^3}} \tag{1}$$

between 0 and 0.9. The function is known *exactly* (to 15 digits) at certain locations, $x_k = x_0 + kdx$, $k = 0, \ldots, n$ where n is the order of interpolation required.

1. Convert the function to a table, spaced 0.05 apart, sampling it from 0.1 to 0.9.

2. In *python*, plot this function and determine its general behaviour. Is it analytic in that region? What is the radius of convergence at $x = 0$ and at $x = 0.9$?

3. Use spline interpolation on the function in Eq. (1). Use the *not-a-knot* method and obtain the spline coefficients and hence interpolate as above. How does the error vary with the number of uniformly spaced spline points? How many points are required to achieve six digit accuracy?

4. Use Eq. 1 to *analytically* compute the derivitives at the end points. Note that $J_0'(x)$ can be written in terms of $J_1(x)$ and hence computed. Use that information to obtain the spline coefficients. How does the error in the interpolated value compare with the *not-a-knot* method? This is an important issue for an experimentalist since he usually does not know y' at x_0 and x_n.

5. Use hundred times the actual derivitive as your boundary condition. How does the spline fit change? Plot the points near the edge and show the way that the errors decay. This is what you need to explain in the theory assignment.

6. Can a non-uniform spacing of spline points help reduce the number of spline points required? Try to determine the optimal spacing at which error appears to be uniform across the domain.

7. Use rational interpolation for this problem and study the dependence of error on order.

8. Create a 5×5 table of data by sampling $\sin \pi x \cos \pi y$ uniformly between 0 and 1 along both x and y. Use the known derivitives of this function to generate the spline matrix.

   (a) Generate a 50×50 mesh of points over the same domain and use bilinera interpolation to evaluate the function. Plot a surface plot of the function.

   (b) Generate a bicubic spline interpolation at these points. Plot the same. Compare with what the bilinear gave.

   (c) Genearte a straight cubic interpolation at these points and compare. Compare.