Assignment 7

Corrections: [none yet]

1. We have points x_k and values f_k. We want a polynomial of degree d so that $p(x_k) = f_k$ for $k = 0, \ldots, d$. Write a Python module that constructs $p(x) = p_0 + \_1 x + \cdots + p_d x^d$ by solving a linear system of equations for the coefficients p_k. Use $d+1$ points uniformly spaced in the interval $[-1, 1]$. Create a Python module `pFit.py` that finds the coefficients, given the x_k and f_k by solving a linear system with the Vander Monde matrix. It also should return the condition number of the Vander Monde matrix. You can evaluate the condition number using a Nympy routine or from the singular values. Use the appropriate `Numpy` linear algebra routine to do the solving. Write another Python module `pEval.py` that takes the coefficients p_k and a point x and evaluates $p(x)$. Do this with Horner's rule.

   (a) Write a module `pTest.py` that does several tests to see that the other two routines work correctly. It should test both routines.

   (b) Write a module that plots the condition number of the VanderMonde matrix as a function of d. Make a plot (log-log or semi-log or whatever) that makes clear the quantitative behavior of the condition number as d grows. Does it grow with d? Does it grow polynomially or exponentially? Think about what kind of plot would answer these questions and make it.

   (c) Plot the polynomial fits of
   $$F(x) = e^{-\frac{1}{2}x^2} \ .$$
   Plot the polynomial of degree d that interpolates F at $d+1$ uniformly spaced points in the interval $[-2, 2]$. (Warning: this looks like the Runge example but it isn't.) For plotting, you need a large number of uniformly spaced points (maybe a few hundred?), even if d is much smaller than that. Comment on the error as a function of d. Comment on the error inside and outside of the interval $[-1, 1]$ that was used for fitting. What effect does the conditioning of the Vander Monde matrix have? Please answer quantitatively.

2. Write Python modules `sFit.py`, `sEval.py` and `sTest.py` that repeat much of Exercise 1, but with cubic B-splines. The fitting routine takes $d+1$ knot points $x_0 < x_1 \leq x_d$ and constructs a piecewise cubic polynomial $s(x)$ so that for $x_k \leq x \leq x_{k+1}$ the spline is a cubic polynomial
   $$s(x) = q_k(x) = a_k + b_k(x - x_k) + c_k(x - x_k)^2 + d_k(x - x_k)^3 \quad \text{for} \quad x_k \leq x \leq x_{k+1} \ .$$

We use $(x - x_k)$ instead of x to make the condition number better. The coefficients should be found from the following conditions

$$\begin{aligned} s(x_k) &= f_k \quad \text{for} \quad k = 0, \ldots, d \\ s' \text{ and } s'' &\text{ are continuous} \\ s''(x_0) &= 0 \\ s''(x_d) &= 0 \end{aligned}$$

Show that this is $4d$ linear equations for the $4d$ unknown coefficients. Repeat the steps from Exercise 1 to create spline interpolation software.

(a) The module `sText.py` should use finite differences on the computed $s(x)$ to verify that the fourth derivative is zero (for a cubic) except at knot points x_k. It should verify that $s(x)$ and $s'(x)$ and $s''(x)$ is continuous at knot points (finite differences). It should verify that the interpolation conditions are satisfied.

(b) Plot the condition number of the $4d \times 4d$ matrix as a function of d. Contrast this condition number to the condition number of the matrix used for high order polynomial interpolation.

(c) Test the accuracy on $e^{-\frac{1}{2}x^2}$ on $[-1, 1]$. Make a plot with a small but not too small number of knots where you can see the error. Make a plot of the max error as a function of d. If you have time, try to find the order of accuracy.

3. *Radial basis function* interpolation finds a function

$$r(x) = \sum_{k=0}^{d} w_k \phi(x - x_k) \ .$$

The function $\phi(x)$ is the radial basis function. Take it to be the "quadratic exponential"

$$\phi(x) = e^{-\frac{x^2}{2L^2}} \ .$$

In more than one dimension, this would be $e^{-\frac{|x|^2}{2L^2}}$, which is a function only of $r = |x|$ and explains the term "radial" basis function. Radial basis function interpolation involves a "length scale" parameter L. Write a set of Python modules `rFit.py`, and `rEval.py`, and `rTest.py`. The fitting routine should choose weights w_k to satisfy interpolation conditions $r(x_k) = f_k$.

(a) The tester `rTest.py` should check that the interpolation conditions are satisfied.

(b) Explore the condition number as a function of d and L for uniformly spaced points.

(c) Do radial basis function interpolation on $e^{-\frac{1}{2}x^2}$ with $d+1$ points and length scale L to see which combinations give good results.