Computational methods and applications (AMS 147)

Homework 4 - Due Sunday February 24

Please submit to CANVAS a .zip file that includes the following Matlab functions:

`poly_least_squares.m`
`test_least_squares.m`

For the Extra Credit 1 and 2, scan your notes into one PDF file `scan.pdf`, and attach it to your submission (as a separate file, not as part of the .zip file)

Exercise 1 (50 points) Write a Matlab function `poly_least_squares.m` that implements the least squares method to approximate a data set in terms of a polynomial model of degree M. The function should be of the form

```
function [a,err] = poly_least_squares(xi,yi,M)
```

Input:
`xi`: vector of nodes `xi=[xi(1) ... xi(N)]`
`yi`: vector of data points `yi=[yi(1) ... yi(N)]` corresponding to `[xi(1) ... xi(N)]`
`M`: degree of the polynomial model

$$\psi(x) = \texttt{a(1)} + \texttt{a(2)}x + \texttt{a(3)}x^2 + \cdots + \texttt{a(M+1)}x^M \tag{1}$$

Output:
`a`: vector of coefficients representing the polynomial (1), i.e., `a=[a(1) ... a(M+1)]`
`err`: error between the model and the data in the 2-norm

$$\texttt{err} = \sum_{i=1}^{N} \left[y_i - \psi(x_i)\right]^2 . \tag{2}$$

Hint: You can compare the results of your least squares implementation with the output of the Matlab functions `polyfit()` and `polyval()` (see the Matlab/Octave documentation for details).

Exercise 2 (50 points) Use the function you coded in Exercise 1 to determine the least squares polynomial approximants of the attached data set `DJI_2014_2019.dat` representing the daily opening value of the Dow Jones Industrial Average from 1/1/14 to 2/17/19. To this end, write a Matlab/Octave function `test_least_squares.m` of the form

```
function [x,p1,p2,p4,p8] = test_least_squares()
```

Output:
`x`: vector of 1000 evenly-spaced evaluation nodes in $[0, 1]$ (including endpoints)
`p1`, `p2`, `p4`, `p8`: least squares polynomial models (1) of the DJI value with degrees $M = 1, 2, 4, 8$, respectively,

evaluated at `x`.

The function should also return one figure with the plots of the DJI opening prices $\{\texttt{x(i)},\texttt{y(i)}\}_{\texttt{i=1,2,...}}$ (from the data file `DJI_2014_2019.dat`) (in blue) and the least-squares polynomial models `p1, p2, p4` and `p8` you computed above (in red).

Hint: To load the DJI data in Matlab/Octave use the command `load` (see the Matlab/Octave documentation for further details).

Extra Credit 1 (10 points) Show that you can integrate exactly any polynomial $p_3(x)$ of degree 3 in $[0,1]$ by integrating the parabola $\Pi_2 p_3(x)$ that interpolates $p_3(x)$ at $\{0,1/2,1\}$, i.e.,

$$\int_0^1 p_3(x)dx = \int_0^1 \Pi_2 p_3(x)dx. \tag{3}$$

Extra Credit 2 (10 points) Let $x_j = j/n$ ($j = 0, ..., n$), be a set of $n+1$ evenly spaced points in $[0,1]$. We have seen in class that we can approximate any smooth function $f : [0,1] \mapsto \mathbb{R}$ with an interpolatory (not-a-knot) cubic spline $s_3(x)$. This means that we can approximate the integral of $f(x)$ by integrating the corresponding spline $s_3(x)$ (which can be done analytically). The error associated with such approximation can bounded as

$$\left| \int_0^1 f(x)dx - \int_0^1 s_3(x)dx \right| \leq \frac{5}{384n^4} \max_{x\in[0,1]} \left| f''''(x) \right|, \tag{4}$$

(see the lecture note `LECTURE_8_piecewise_interpolation_and_splines.pdf` posted in CANVAS). Derive a similar error estimate for quadratic cubic splines and compare the upper bound you obtain with the one of the Simpson rule[1]. To this end, note that if we interpolate the function $f(x)$ at the $n+2$ nodes $x_0 = 0$, $x_n = 1$ and $\overline{x}_j = (x_{j+1} + x_j)/2$ ($j = 0, .., n-1$), then the spline $s_2(x)$ is uniquely defined, and we have the error estimate

$$\max_{x\in[0,1]} \left| f(x) - s_2(x) \right| \leq \frac{1}{7n^3} \max_{x\in[0,1]} \left| f'''(x) \right|. \tag{5}$$

Which method between Simpson and quadratic spline interpolation converges faster to the integral of $f(x)$ in $[0,1]$ as we increase the number of points n?

[1]Note that in both cases you approximate the function $f(x)$ locally with a polynomial of degree 2. The difference between spline interpolation and simple piecewise polynomial interpolation (used in the Simpson rule) is that $s_2(x)$ is $C^1([0,1])$, while the classical piecewise polynomial interpolant is only $C^0([0,1])$.