CS412, Fall 19
Prof. Amos Ron

Assignment #5
Due: Decmeber 2 , 2019
(after that day assignments should be put in the mbox of Jihye Choi)

(1) Your recent exposition on ill-conditioning was fabulous! As a result, you were promoted to the position of a J.V.P. in the firm `Ail, Utup & Azy` (cf. previous assignments for more details). In your new position, you report directly to the president I.M.L. Azy. Lynn asks you to propose ways for removing noise from scientific measurements. You immediately assign this problem to two of your new subordinates, R.T. Arty and S.T. Esty. Dick suggests that, in order to approximate noisy measurements, you should use the cubic spline least squares (`spap2` in `matlab`). Sarah disagrees and suggests to use least squares approximation by polynomials. You feel unsure, and decide to take the matter into your hands, and run a large scale experiment:

(a) The experiment will consist of running many tests in which you compare least squares polynomial approximation to least squares spline approximation. Each single test is executed as follows:
  (i) Pick one (by one each) of the four functions that were involved in the previous two assignments.
  (ii) Evaluate the given function at m equidistant points on the interval $[-1, 2]$.
  (iii) Add "noise" to the above m values.
  (iv) Approximate the noisy values by a polynomial of degree $\leq n$, using the least squares method.
  (v) Approximate the same noisy values by a cubic spline with $n+1$ equidistant knots over $[-1, 2]$
  (vi) Compute the error of each method and compare thereby Dick's suggestion to Sarah's.

(b) In order to create "noise" you can use the command
`noise=a*(rand(1,m)-.5)`
Note that a controls here the level of noise. In your experiments, always start with $a = 0$ and increase the level of the noise gradually. The above noise is *uniform.* If you know how to create other types of noise (e.g., *white* noise), you are more than welcome to try those as well.

(c) In order to find the least squares polynomial approximation to your noisy data y, you first create an $m \times (n+1)$ Vandermonde matrix V, and then find the coefficients of the polynomial by `a=V\y`. Remember that these are coefficients in the monomial representation, and you can use nested evaluation to evaluate this polynomial.

(d) In order to find the least square cubic spline approximation, use `matlab`'s `spap2`. Type `help spap2` for details. Note that the order of your spline is 4 (and not 3). In order to evaluate your spline you should use `spval` (and not ppval). *Note: for technical reasons, the first knot and the last knot in your sequence should be listed each 4 times such as in* $0, 0, 0, 0, 0.5, 1, 1.5, 2, 2, 2, 2$. *This sequence contains* 7 *knots, (count each endpoint twice), hence corresponds to* $n = 6$.

(e) Since you use least squares, the "error" here is best interpreted as the 2-norm of your error vector. Your error vector is the difference between the m (noise-free) values of your function and the m values produced by the approximating polynomial or spline. Note that the error is expected to grow with m, and you can "neutralize" this (clearly undesired) artifact by dividing each error by $\sqrt{m}$ (why $\sqrt{m}$? if all the entries of your error vector are 1's, the 2-norm of the vector is $\sqrt{m}$, while the "correct" value should be 1.)

(f) Your objective is to learn as much as possible from your tests about least square fitting by polynomials and splines. In particular, you wish to know whether taking a large m helps or not, how fast the error grows with the noise (you may wish to know that for the kind of noise suggested above, the "size" of the noise (i.e., its standard deviation) is $\sim a^{3/2}$.), how to 'play" correctly with n, i.e., the number of degree of freedoms, and how far the results are influenced by the "fidelity" of the underlying function. Since you have four parameters to play with (m, n, a and the function we "chase"), you need to organize your experiment very carefully. For example, you may wish to fix three of the above, to vary the fourth one, and to plot graphs of the error's norm as a function of the parameter you vary.

(h) Turn in every piece of code you wrote here, a short description of the types of experiments you ran, and a careful list of all the conclusions you draw. It is useful and recommended to provide samples from your experiments. You are limited here to a *total* of no more than 10 graphs/plots/histograms/tables combined.

Finally, there are many ways to run this experiment successfully, and the question does not attempt to suggest any particular way, only to give you some general guidelines. Thus, simply try to be creative and effective in running your experiment.

(2) (This problem deals with numerical integration.) Lynn also asks you to find a way to approximate integrals of the form

$$\int_1^2 f(t)\,dt,$$

by expressions of the form

$$w_0 f(1) + w_1 f(1.5) + w_2 f(2).$$

You, again, ask Dick and Sarah for help. Dick suggests that you use the scheme

$$\int_1^2 f(t)\,dt \approx \frac{f(1)+f(2)+4f(1.5)}{6},$$

while Sarah believes that the right scheme is

$$\int_1^2 f(t)\,dt \approx \frac{f(1)+f(2)+2f(1.5)}{4}.$$

(a) Identify each one of the suggested schemes with a known quadrature scheme discussed in class ("quadrature scheme" is a numerical integration rule). Write the error formula for each one of these schemes (a typical error formula should be of the form *error* $= f^{(j)}(c) \times \#$, where c is in the interval $[1,2]$; so you need to determine j and the value of the number $\#$ for each case).

(b) You are now given five functions f: x^2-3x+2, $\arctan(3x-4)$, $\sin(x/2)$, $(2-x)^{5/2}$, and $\sqrt{|x-1.5|}$. Based only on your error formulæ from (a), which of these you assign to Dick, and which to Sarah (no more than three functions to one person; you may assign one of the functions to both). Show the work that led you to your choice. (Of course, you would like to assign a given function to the person whose scheme is expected to do better for that function).

(c) Next, you would like to check your decision making abilities. For that, you first need to find the correct values of the integrals of the above five functions (with an error no larger than 10^{-6}). You must follow the following rules: (1) At least four of these five integrals should be computed by numerical integration, hence at most one may be computed with the aid of anti-derivatives. (2) The numerical integration should be a composite rule studied in class. You are free to choose the rule, and to assign different rules to different functions. However, you must be able to calculate in advance the correct number of subintervals required for the 10^{-6} accuracy. Thus, for example, you will not be able to use Composite Simpson for a function whose 4th order derivative does not exist at some point of the interval, since for such a function, you cannot use the error formula of Composite Simpson, hence cannot find in advance the number of subintervals to be used. On the other hand, you would not like to use the inferior Composite Rectangle rule for 'nice functions', since you will need a large number of subintervals in order to get the required accuracy. Explain briefly how you selected the appropriate rule for each function, and how you computed the correct number of subintervals. Then, write a short `matlab` code that computes these four (or five) numbers. (Why "four or five"? If you choose to use anti-derivatives for one of the integrals, you really won't need `matlab` for that case.)

(d) Now find the errors in Dick's scheme and Sarah's scheme for each one of the functions above (you found the correct value in (c)!) Compare these to your error estimates and predictions. (Hint: if you find out that you goofed, you might wish to go back to (b), and to revisit the analysis you did there.)