Homework 6: Least Squares

Michael Overton, Numerical Computing, Spring 2017

April 3, 2017

This homework builds on Homework 3, where you built a Vandermonde matrix and then used MATLAB's backslash operator to implement polynomial *interpolation* (when the degree of the polynomial was one less than the number of data points) or *least squares approximation* (when the degree was smaller).

Write a MATLAB function `approxpoly` as follows. The input consists of `t` and `b`, which specify the data $(t_i, b_i), i = 1, \dots, m$, and the polynomial degree `d`, as before. The `nPlot` parameter specifying the length of the vector on which the approximating polynomials are going to be plotted should not be an input to `approxpoly`: the plotting is still important, but it should be done by a different function.

For each choice of d, which will be specified further below, the program should construct the $m \times (d+1)$ Vandermonde matrix A and compute the coefficients x for the least-squares approximating polynomial of degree d, which is the vector x minimizing $\|Ax - y\|$, by four different methods:

1. Using vanilla backslash: `x=A\b`. This is the *only* place in the program where you should use `\`.

2. The normal equations, as on p. 145 of A&G, using the Cholesky factorization (computed by `chol`). You can use the codes `forsub` (modified appropriately, since the Cholesky factor does *not* have a unit diagonal) and `backsub` given on p. 112 for forward and back substitution. Do not use the backslash operator `\`. In particular, do *not* compute `B\y` as on p. 149, where $B = A^T A$, because MATLAB will probably not recognize that B is symmetric positive definite and will therefore probably use LU factorization instead of Cholesky, which is twice as much work and potentially less accurate.

3. Using the economy-size QR factorization, as on p. 156, version (a). This is done via `[Q,R]=qr(A,0)`. Again you can use `backsub` to solve the triangular system.

4. Using the economy-size SVD, as on p. 235, *with a cutoff tolerance*, which specifies the size of "small" singular values to be omitted in

step 4. In other words, instead of prespecifying r, determine r according to whether or not the singular values are bigger or smaller than the cutoff tolerance. For now, set the cutoff tolerance to zero: we will come back to this later. Since you are computing the SVD here, it's convenient to compute $\kappa(A)$, the condition number of A at the same time: this is σ_1/σ_n. Remember that the economy-size SVD is computed by `[U,S,V]=svd(A,0)`.

The outputs from `approxpoly` should be the four coefficient vectors computed by these four methods, along with the computed condition number $\kappa(A)$. All four methods are computing the same least squares solution x, but numerically, they may be different.

Now experiment, **using the data given in the** MATLAB **data file linked from the course web page**. The first thing you should do is to make sure the four coefficient vectors are nearly, if not exactly, the same, for small degree polynomials. If not, you have a bug.

Using the coefficient vectors computed by the SVD method, call the plotting routine to plot the approximating polynomial on `tt`, a grid of `nPlot` ordered points, equally spaced between t_1 and t_m, as well as the original data points (t_i, y_i), as in Homework 3, for various choices of d. How big do you need to raise d in order to make the polynomial approximate the data reasonably well in the "eyeball" norm, in other words so it looks reasonable to you? In order to make this quantitative, write another routine to call `approxpoly` repeatedly to compute the residual $r = b - Ax$ for increasing degrees d, again using the SVD-computed coefficient vectors, and plot the residual 2-norms in another figure. As d gets larger you will see these residual norms decrease. What seems to be a reasonable choice of d, in your opinion? (There is no "right" answer to this.)

Before going further, carefully think about the following question and **write the answer out carefully in your submitted homework**. Intuitively, it seems that raising the degree from d to $d + 1$ should result in a smaller residual norm, or, possibly, the same, but not bigger. Why is this? Give a proof that, mathematically speaking, this is true: this is not difficult once you see how to do it. However, if you make *d too big*, the residual may actually increase. If so, for what d does this occur? The reason is the huge condition number of A for large d. In particular, the residual should be zero mathematically if $d = m - 1$, because in this case x defines the *interpolating* polynomial, but the condition number of A is so large that the computation is inaccurate and the residual will not be zero.

Now write another routine that plots the relative error norms

$$\eta_j = \frac{\|x^{(j)} - x^{(4)}\|}{\|x^{(4)}\|}, j = 1, 2, 3,$$

where $x^{(j)}$ denotes the coefficient vector computed by method j in the list of four methods above. The idea here is that the SVD method should be the most accurate, so we are comparing the computed coefficients for the other methods with the SVD method. As you keep raising the degree, because $\kappa(B)$ is the square of $\kappa(A)$, where $B = A^T A$, you should eventually see the normal equations coefficient vector $x^{(2)}$ differ significantly from the others, and hence η_2 increase. Since these numbers will vary a lot in magnitude as η_2 increases, plot them with `semilogy`. Plot the condition numbers $\kappa(A)$ and $\kappa(B) = \kappa(A)^2$ too (it may be better to use another semilogy figure for this). Think how best to display the results. Make good use of `legend`, `xlabel`, `ylabel`, `title`, etc., in your plots.

Answer the following questions:

1. Are there any significant differences between $x^{(1)}$, $x^{(3)}$ and $x^{(4)}$? In other words, do η_1 and η_3 remain small as you increase d, compared to η_2?

2. How big does d have to get for the normal equations coefficient vector $x^{(2)}$ to agree with the SVD coefficient vector to only about half the machine precision (8 digits): in other words, $\eta_2 > 10^{-8}$? What are the condition numbers $\kappa(A)$ and $\kappa(B) = \kappa(A)^2$ for this degree?

3. If you make d sufficiently large, is the polynomial plotted using the normal equations coefficient vector $x^{(2)}$ visibly different from the polynomial plotted using the SVD coefficient vector $x^{(4)}$? Plot them both in the same figure with different colors and/or symbols (type `help plot`). You may have to use the zoom tool to see a difference. If you can see a difference, what are the condition numbers $\kappa(A)$ and $\kappa(B) = \kappa(A)^2$ for this degree?

4. If you make d larger still, the condition number $\kappa(A)$, and hence the condition number $\kappa(B) = \kappa(A)^2$, becomes so large that Cholesky "breaks down": it decides that the matrix B is not positive definite. You can prevent an error return by requesting the second output of `chol` and, if it is nonzero, skipping the forward and back substitution and setting the coefficient vector to NaNs, so the residual will then also be NaN and will not appear in the plot. How big is d when this happens, how big are the condition numbers $\kappa(A)$ and $\kappa(B)$, and how big is the norm of the SVD coefficient vector?

5. Now, experiment with changing the singular value cutoff tolerance to be positive. The bigger the tolerance, the more the SVD coefficient vector should be reduced in norm, because we are no longer dividing by the small singular values. We say the computation is *regularized.* Does this have any effect, good or bad, on (a) the residual norms or

(b) the plot of the approximating polynomial? Plot another figure showing the norm of the SVD coefficient vector and the norm of the residual as the tolerance cutoff is increased.

Submit written answers to the questions above, with supporting code listings and plots. You do not need to submit a huge pile of output. Just submit enough to support your answers.