MAT3110/MAT4110, Autumn 2025, Compulsory assignment 1

Deadline 25 September, 14:30

Consider the linear system of equations $A\mathbf{x} = \mathbf{b}$,

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix},$$

where $n > m$. Since there are more equations than unknowns, we cannot usually solve this system, but we can instead find the vector $\mathbf{x} \in \mathbb{R}^m$ that minimizes $\|A\mathbf{x} - \mathbf{b}\|_2$, which is known as the least squares method.

In this assignment we are going to approximate two data sets with a polynomial curve. The first data set is

```
n = 30
start = -2;
stop = 2;
x = linspace(start,stop,n);
eps = 1;
rng(1);
r = rand(1,n) * eps;
y = x.*(cos(r+0.5*x.^3)+sin(0.5*x.^3));
plot(x,y,'o');
```

The second data set is

```
n = 30
start = -2;
stop = 2;
```

```
x = linspace(start,stop,n);
eps = 1;
rng(1);
r = rand(1,n) * eps;
y = 4*x.^5 - 5*x.^4 - 20*x.^3 + 10*x.^2 + 40*x + 10 + r;
plot(x,y,'o');
```

These data sets can be seen in Figure 1 below. Note that in this code, you can change the value of ϵ to generate more or less noise. Letting $\epsilon = 0$ will remove the noise.

Given n observations of data $y_1, \ldots, y_n$, possibly noisy data, at distinct points $x_1, \ldots, x_n$, it is often desirable to fit a smooth function. Such a smooth function could be a polynomial, piecewise polynomial (spline), trigonometric function, sum of exponentials, and so on. In this assignment we will fit a polynomial of degree $< m$ for some chosen m, i.e.,

$$p(x) = \sum_{j=1}^{m} c_j x^{j-1},$$

and we will minimize the sum of the squares of the errors

$$S = \sum_{i=1}^{n} (p(x_i) - y_i)^2.$$

Substituting the formula for p into S gives

$$S = \|A\mathbf{x} - \mathbf{b}\|_2^2,$$

where

$$A = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{m-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{m-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{m-1} \end{bmatrix},$$

a so-called Vandermonde matrix, $\mathbf{x} = [c_1, c_2, \ldots, c_m]^T$, the vector of coefficients of p, and $\mathbf{b} = [y_1, y_2, \ldots, y_n]^T$, the data observations.

There are two methods of finding $\mathbf{x}$ that minimizes $\|A\mathbf{x} - \mathbf{b}\|_2^2$. One is to use the QR factorization of A as explained in Lectures 3 and 4. The other is to solve the *normal equations*. Since

$$F(\mathbf{x}) = \|A\mathbf{x} - \mathbf{b}\|_2^2$$

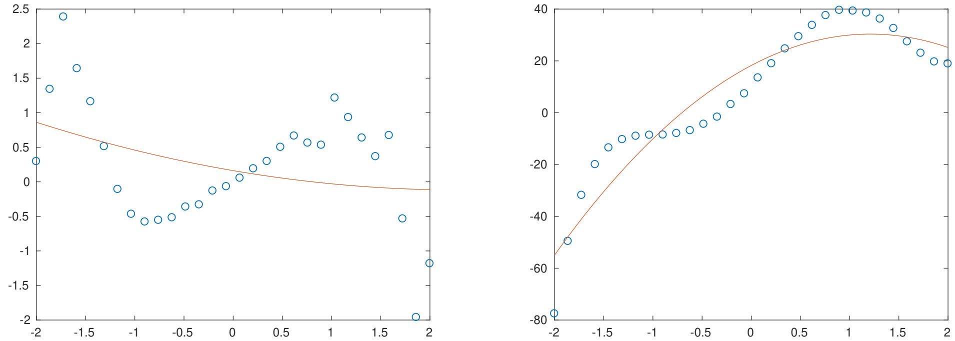

Figure 1: Least squares polynomial fitting

is a convex function of the vector $\mathbf{x}$, it has a unique minimium which occurs when its gradient, $\nabla F(\mathbf{x})$, is zero. Setting this gradient to zero leads to the normal equations

$$A^T A\mathbf{x} = A^T\mathbf{b}.$$

Your task is to implement both these methods.

1. Use the QR factorization of A and apply back substitution to R_1 to find $\mathbf{x}$. You will need to write your own routine for back substitution but you can use the matlab function

   ```
   [Q,R] = qr(A);
   ```

   to find Q and R.

2. The $m \times m$ matrix $B = A^T A$ is symmetric and positive definite. Solve the normal equations using the Cholesky factorization RR^T of B. To do this you also need to implement forward substitution and the Cholesky algorithm explained in Lecture 3.

3. Discuss the difference in the two methods, for example, from the point of view of conditioning if m is large.

Then let $m = 3$ and $m = 8$ and plot the two polynomials corresponding to the two data sets. Your two plots for $m = 3$ should look something like Figure 1.

Your delivery should be a short report showing the four plots and summarizing your work as a **single pdf file**, submitted through canvas.