Numerical Analysis Hilary Term 2024
Lecture 4: Least-squares problem

So far the linear systems we treated had the same number of equations as unknowns (variables), so the problem was $Ax = b$ for a square matrix A. Very often in practice, we have more equations that we would like to satisfy than variables to fit them. It is then usually impossible to obtain $Ax = b$; a common approach is then to try minimise the difference between Ax and b. If we choose to minimise the Euclidean length of the vector, this leads to a *least-squares problem*:

$$\min_x \|Ax - b\|, \qquad A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m, m \geq n. \tag{1}$$

Here $\|y\| := \sqrt{y_1^2 + y_2^2 + \cdots + y_m^2} = \sqrt{y^T y}$.

Least-squares problems (also known as *overdetermined* systems) are ubiquitous in applied mathematics and data science; linear regression is a basic example.

Solution of least-squares by the QR factorisation:

Let $A = [Q\ Q_\perp]\left[\begin{smallmatrix} R \\ 0 \end{smallmatrix}\right] = Q_F\left[\begin{smallmatrix} R \\ 0 \end{smallmatrix}\right]$ be a 'full' QR factorization, computed e.g. via the Householder QR factorization. We assume R is nonsingular (i.e., A has full column rank); this is a generic condition. Noting that $\|Q_F^T y\| = \sqrt{y^T Q_F Q_F^T y} = \sqrt{y^T y} = \|y\|$ for any vector y, we have

$$\|Ax - b\| = \|Q_F^T(Ax - b)\| = \left\| \begin{bmatrix} R \\ 0 \end{bmatrix} x - \begin{bmatrix} Q^T b \\ Q_\perp^T b \end{bmatrix} \right\|.$$

The bottom part is $-Q_\perp^T b$, no matter what x is. The top part can be made 0 by taking $x = R^{-1}Q^T b$—this is therefore the solution.

The argument also suggests an algorithm: compute the "thin" QR factorization $A = QR$, then solve $Rx = Q^T b$ for x, which is obtained by backward substitution as R is triangular. Note that while we used the full QR for the derivation, we only need the thin QR for the solution of (1).

Later we will see that a general linear least-squares problem has solution characterised by the orthogonality condition, which in our context reduces to $A^T(Ax - b) = 0$, so $x = (A^T A)^{-1} A^T b$; one can verify this is the same as $R^{-1}Q^T b$ obtained above.

Illustration of least-squares for polynomial approximation: We treated Lagrange interpolation in Lecture 1. While Lagrange polynomials give a clean expression for the interpolating polynomial, the interpolating polynomial is not always a good approximation to the original underlying function f. For example, suppose $f(x) = 1/(25x^2 + 1)$ (this is a famous function called the *Runge function*), and take a degree-n polynomial interpolant p_n at $n + 1$ equispaced points in $[-1, 1]$. The interpolating polynomials for varying n are shown in Figure 1.

As we increase n, we hope that $p_n \to f$—but this is far from the truth! p_n is diverging as n grows near the endpoints ± 1, and the divergence is actually exponential (very bad); note the vertical scales of the final plots! This is called Runge's phenomenon.

How can we avoid the divergence, and get $p_n \to f$ as we hope? One approach is to *oversample*: take (many) more points than the degree n. With $m(> n + 1)$ data

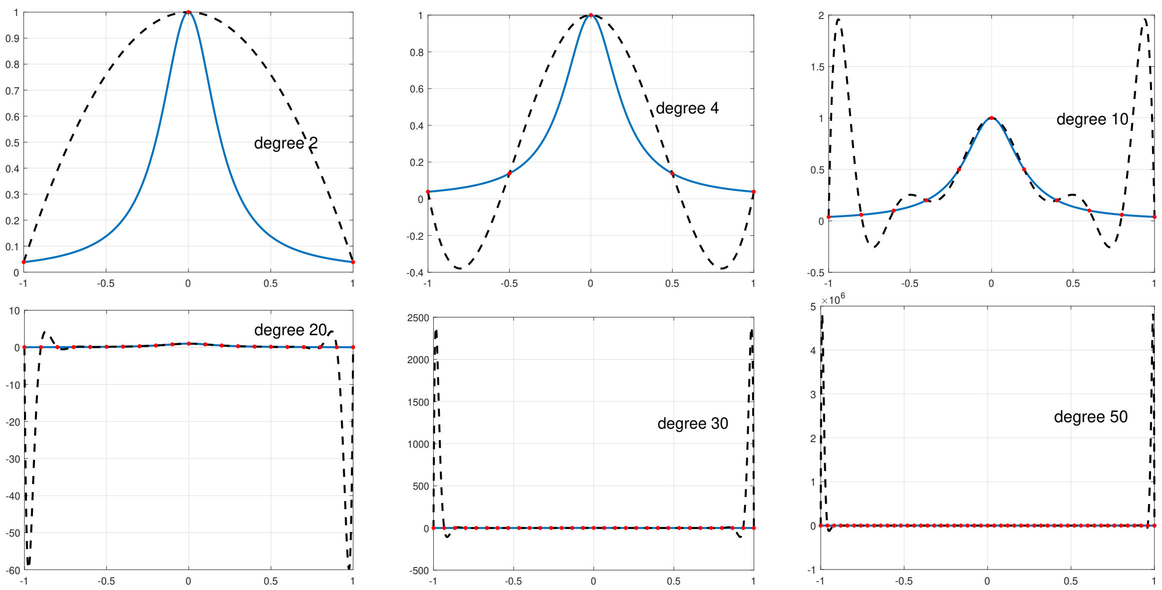

Figure 1: Polynomial interpolants (dashed black curves) of $f(x) = 1/(25x^2+1)$ (blue). The red dots are the interpolation points.

points $x_1, \ldots, x_m$, this will lead to the least-squares problem $\min_c \|Ac - b\|$, wherein $c = [c_0, c_1, \ldots, c_n]^T$ represents the coefficients of the polynomial $p_n(x) = \sum_{j=0}^{n} c_j x^j$, $A \in \mathbb{R}^{m\times(n+1)}$ with $A_{ij} = (x_i)^{j-1}$ and $b = [f(x_1), \ldots, f(x_m)]^T$.

We illustrate this in Figure 2 with the example above, but now fixing $n = 20$ and varying the number of data points m. This time, for large enough m the polynomial p_n is close to f across the whole interval $[-1, 1]$.

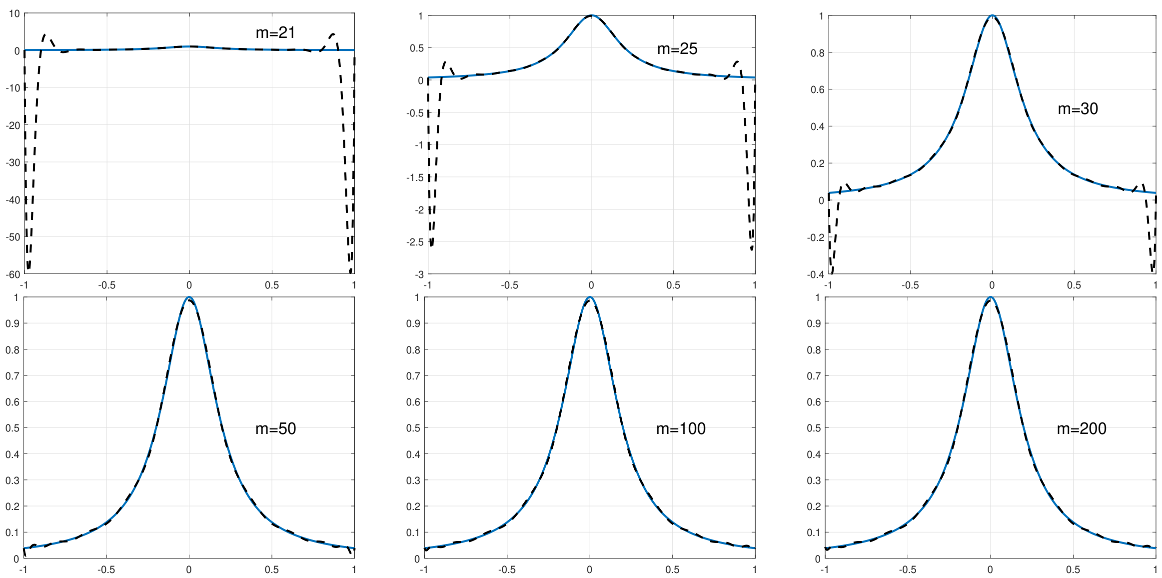

Figure 2: Least-squares polynomial fits of degree 20 (black dashed curves) of $f(x) = 1/(25x^2+1)$ (blue).

Extensions and related facts (Non-examinable):

- Instead of $p_n(x) = \sum_{j=0}^{n} c_j x^j$, it is actually much better to use a different polynomial basis involving *orthogonal polynomials* $\{\phi_i\}_{i=0}^n$ such as the Chebyshev polynomials, a topic discussed later. Then we would express $p_n(x) = \sum_{j=0}^{n} c_j \phi_j(x)$ and $A_{ij} = (\phi_{j-1}(x_i))$, and the least-squares problem will be beter-conditioned (easier to solve accurately). However, Runge's phenomenon still persists unless $m \gg n$.

- Note that we do not have $p_n \to f$ in Figure 2 as $m \to \infty$ because the polynomial degree $n = 20$ is fixed; to get $p_n \to f$ one needs to increase n together with m. It can be shown that if one takes $m = n^2$, we do have $p_n \to f$ for any analytic function f (the convergence is exponential in n).

- Another—more elegant—solution to overcome the instability in Figure 1 is to change the interpolation points. If one chooses them to be the so-called Chebyshev points $x_j = \cos(j\pi/n)$ for $j = 0, 1, \ldots, n$, the interpolating polynomial can be shown to be an excellent approximation to f, in fact nearly the best-possible polynomial approximation for any continuous f. This is a fundamental fact in approximation theory; for a rigorous and extended discussions (including an explanation of Runge's phenomenon), check out the Part C course Approximation of Functions.

Underdetermined case (Non-examinable): One might wonder, what if we have *fewer* equations than variables? That is, if we have $Ax = b$ with $A \in \mathbb{R}^{m \times n}$, $m < n$. This *underdetermined* system of equations has infinitely many solutions (if there is one). The natural question becomes, which one should we look for? One possibility is to find the minimum-norm solution minimize $\|x\|$ subject to $Ax = b$; the solution can be computed again via the QR factorization (of A^T). This problem has connections to the hot topic of *deep learning*. Another fascinating approach that has had enormous impact is to minimise the 1-norm $\|x\|_1$ subject to $Ax = b$, where $\|x\|_1 = \sum_{i=1}^{n} |x_i|$. It turns out that the solution x then tends to be sparse, i.e., most of its entries are 0. This is the basis of the exciting field of *compressed sensing*.