Last time: piecewise polynomial interpolation, least-squares fitting
Today: underdetermined least squares, nonlinear least squares
Find least-squares fit for degree 11 polynomial to 50 samples of $y = \cos(4x)$ for $x \in [0, 1]$

Let’s express the best-fit polynomial using the monomial basis: $p(x; b) = \sum_{k=0}^{11} b_k x^k$

(Why not use the Lagrange basis? Lagrange loses its nice properties here since $m > n$, so we may as well use monomials)

The $i$th condition we’d like to satisfy is $p(x_i; b) = \cos(4x_i) \implies$ over-determined system with “50 × 12 Vandermonde matrix”
Least-squares polynomial fit

But solving the normal equations still yields a small residual, hence we obtain a good fit to the data

$$\| r(b_{\text{normal}}) \|_2 = \| y - A b_{\text{normal}} \|_2 = 1.09 \times 10^{-8}$$

$$\| r(b_{\text{lst.sq.}}) \|_2 = \| y - A b_{\text{lst.sq.}} \|_2 = 8.00 \times 10^{-9}$$
Non-polynomial Least-squares fitting

So far we have dealt with approximations based on polynomials, but we can also develop non-polynomial approximations. We just need the model to depend linearly on parameters.

**Example:** Approximate $e^{-x} \cos(4x)$ using $f_n(x; b) \equiv \sum_{k=-n}^{n} b_k e^{kx}$

(Note that $f_n$ is linear in $b$: $f_n(x; \gamma a + \sigma b) = \gamma f_n(x; a) + \sigma f_n(x; b)$)
Non-polynomial Least-squares fitting

3 terms in approximation

\[ e^{-x}\cos(4x) \]

samples

best fit

\[ n = 1, \quad \frac{\|r(b)\|_2}{\|b\|_2} = 4.16 \times 10^{-1} \]
Non-polynomial Least-squares fitting

\[ n = 3, \quad \frac{\|r(b)\|_2}{\|b\|_2} = 1.44 \times 10^{-3} \]
Non-polynomial Least-squares fitting

11 terms in approximation

\[ e^{-x \cos(4x)} \]
samples
best fit

\[ n = 5, \quad \frac{\| r(b) \|_2}{\| b \|_2} = 7.46 \times 10^{-6} \]
Pseudoinverse

Recall that from the normal equations we have:

\[ A^T Ab = A^T y \]

This motivates the idea of the “pseudoinverse” for \( A \in \mathbb{R}^{m \times n} \):

\[ A^+ \equiv (A^T A)^{-1} A^T \in \mathbb{R}^{n \times m} \]

Key point: \( A^+ \) generalizes \( A^{-1} \), i.e. if \( A \in \mathbb{R}^{n \times n} \) is invertible, then \( A^+ = A^{-1} \)

Proof: \( A^+ = (A^T A)^{-1} A^T = A^{-1} (A^T)^{-1} A^T = A^{-1} \)
Pseudoinverse

Also:
- Even when $A$ is not invertible we still have still have $A^+ A = I$
- In general $AA^+ \neq I$ (hence this is called a “left inverse”)

And it follows from our definition that $b = A^+ y$, i.e. $A^+ \in \mathbb{R}^{n \times m}$ gives the least-squares solution

Note that we define the pseudoinverse differently in different contexts
Underdetermined Least Squares

So far we have focused on overconstrained systems (more constraints than parameters)

But least-squares also applies to underconstrained systems: \( Ab = y \) with \( A \in \mathbb{R}^{m \times n}, m < n \)

\[
\begin{bmatrix}
A
\end{bmatrix}
\begin{bmatrix}
b
\end{bmatrix}
= 
\begin{bmatrix}
y
\end{bmatrix}
\]

i.e. we have a “short, wide” matrix \( A \)
Underdetermined Least Squares

For $\phi(b) = \|r(b)\|_2^2 = \|y - Ab\|_2^2$, we can apply the same argument as before (i.e. set $\nabla \phi = 0$) to again obtain

$$A^T Ab = A^T y$$

But in this case $A^T A \in \mathbb{R}^{n \times n}$ has rank at most $m$ (where $m < n$), why?

Therefore $A^T A$ must be singular!

Typical case: There are infinitely vectors $b$ that give $r(b) = 0$, we want to be able to select one of them
First idea, pose as a constrained optimization problem to find the feasible \( b \) with minimum 2-norm:

\[
\begin{align*}
\text{minimize} & \quad b^T b \\
\text{subject to} & \quad Ab = y
\end{align*}
\]

This can be treated using Lagrange multipliers (discussed later in the Optimization section)

Idea is that the constraint restricts us to an \((n - m)\)-dimensional hyperplane of \( \mathbb{R}^n \) on which \( b^T b \) has a unique minimum
We will show later that the Lagrange multiplier approach for the above problem gives:

\[ b = A^T (AA^T)^{-1} y \]

As a result, in the underdetermined case the pseudoinverse is defined as \( A^+ = A^T (AA^T)^{-1} \in \mathbb{R}^{n \times m} \)

Note that now \( AA^+ = I \), but \( A^+ A \neq I \) in general (i.e. this is a “right inverse”)
Underdetermined Least Squares

Here we consider an alternative approach for solving the underconstrained case

Let’s modify \( \phi \) so that there is a unique minimum!

For example, let

\[
\phi(b) \equiv \|r(b)\|_2^2 + \|Sb\|_2^2
\]

where \( S \in \mathbb{R}^{n \times n} \) is a scaling matrix

This is called regularization: we make the problem well-posed (“more regular”) by modifying the objective function
Calculating $\nabla \phi = 0$ in the same way as before leads to the system

$$(A^T A + S^T S) b = A^T y$$

We need to choose $S$ in some way to ensure $(A^T A + S^T S)$ is invertible.

Can be proved that if $S^T S$ is positive definite then $(A^T A + S^T S)$ is invertible.

Simplest positive definite regularizer: $S = \mu I \in \mathbb{R}^{n \times n}$ for $\mu \in \mathbb{R}_{>0}$
Underdetermined Least Squares

**Example:** Find least-squares fit for degree 11 polynomial to 5 samples of \( y = \cos(4x) \) for \( x \in [0, 1] \)

12 parameters, 5 constraints \( \implies A \in \mathbb{R}^{5 \times 12} \)

We express the polynomial using the monomial basis (can’t use Lagrange since \( m \neq n \)): \( A \) is a submatrix of a Vandermonde matrix

If we naively use the normal equations we see that \( \text{cond}(A^T A) = 4.78 \times 10^{17} \), i.e. “singular to machine precision”!

Let’s see what happens when we regularize the problem with some different choices of \( S \).
Underdetermined Least Squares

Find least-squares fit for degree 11 polynomial to 5 samples of $y = \cos(4x)$ for $x \in [0, 1]$

Try $S = 0.001I$ (i.e. $\mu = 0.001$)

\[ \| r(b) \|_2 = 1.07 \times 10^{-4} \]
\[ \| b \|_2 = 4.40 \]
\[ \text{cond}(A^T A + S^T S) = 1.54 \times 10^7 \]

Fit is good since regularization term is small but condition number is still large
Underdetermined Least Squares

Find least-squares fit for degree 11 polynomial to 5 samples of $y = \cos(4x)$ for $x \in [0, 1]$

Try $S = 0.5I$ (i.e. $\mu = 0.5$)

\[
\|r(b)\|_2 = 6.60 \times 10^{-1}
\]

\[
\|b\|_2 = 1.15
\]

\[
\text{cond}(A^T A + S^T S) = 62.3
\]

Regularization term now dominates: small condition number and small $\|b\|_2$, but poor fit to the data!
Underdetermined Least Squares

Find least-squares fit for degree 11 polynomial to 5 samples of $y = \cos(4x)$ for $x \in [0, 1]$

Try $S = \text{diag}(0.1, 0.1, 0.1, 10, 10 \ldots, 10)$

$$\| r(b) \|_2 = 4.78 \times 10^{-1}$$

$$\| b \|_2 = 4.27$$

$$\text{cond}(A^T A + S^T S) = 5.90 \times 10^3$$

We strongly penalize $b_3, b_4, \ldots, b_{11}$, hence the fit is close to parabolic
Underdetermined Least Squares

Find least-squares fit for degree 11 polynomial to 5 samples of $y = \cos(4x)$ for $x \in [0, 1]$

Python routine gives Lagrange multiplier based solution, hence satisfies the constraints to machine precision

$$\|r(b)\|_2 = 1.03 \times 10^{-15}$$

$$\|b\|_2 = 7.18$$
Nonlinear Least Squares

So far we have looked at finding a “best fit” solution to a linear system (linear least-squares).

A more difficult situation is when we consider least-squares for nonlinear systems.

**Key point**: We are referring to linearity in the *parameters*, not linearity of the *model*.

(e.g. polynomial $p_n(x; b) = b_0 + b_1 x + \ldots + b_n x^n$ is nonlinear in $x$, but linear in $b$!)

In **nonlinear least-squares**, we fit functions that are nonlinear in the parameters.
Nonlinear Least Squares: Example

Example: Suppose we have a radio transmitter at \( \hat{b} = (\hat{b}_1, \hat{b}_2) \) somewhere in \([0, 1]^2 \) (\( \times \))

Suppose that we have 10 receivers at locations \((x_1^1, x_2^1), (x_1^2, x_2^2), \ldots, (x_1^{10}, x_2^{10}) \in [0, 1]^2 \) (\( \bullet \))

Receiver \( i \) returns a measurement for the distance \( y_i \) to the transmitter, but there is some error/noise (\( \epsilon \))
Nonlinear Least Squares: Example

Let $b$ be a candidate location for the transmitter

The distance from $b$ to $(x_1^i, x_2^i)$ is

$$d_i(b) \equiv \sqrt{(b_1 - x_1^i)^2 + (b_2 - x_2^i)^2}$$

We want to choose $b$ to match the data as well as possible, hence minimize the residual $r(b) \in \mathbb{R}^{10}$ where $r_i(b) = y_i - d_i(b)$
Nonlinear Least Squares: Example

In this case, $r_i(\alpha + \beta) \neq r_i(\alpha) + r_i(\beta)$, hence nonlinear least-squares!

Define the objective function $\phi(b) = \frac{1}{2} \| r(b) \|_2^2$, where $r(b) \in \mathbb{R}^{10}$ is the residual vector.

The $1/2$ factor in $\phi(b)$ has no effect on the minimizing $b$, but leads to slightly cleaner formulae later on.
As in the linear case, we seek to minimize $\phi$ by finding $b$ such that $\nabla \phi = 0$

We have $\phi(b) = \frac{1}{2} \sum_{j=1}^{m} [r_j(b)]^2$

Hence for the $i^{th}$ component of the gradient vector, we have

$$\frac{\partial \phi}{\partial b_i} = \frac{\partial}{\partial b_i} \frac{1}{2} \sum_{j=1}^{m} r_j^2 = \sum_{j=1}^{m} r_j \frac{\partial r_j}{\partial b_i}$$
Nonlinear Least Squares

This is equivalent to $\nabla \phi = J_r(b)^T r(b)$ where $J_r(b) \in \mathbb{R}^{m \times n}$ is the Jacobian matrix of the residual

$$\{J_r(b)\}_{ij} = \frac{\partial r_i(b)}{\partial b_j}$$

Exercise: Show that $J_r(b)^T r(b) = 0$ reduces to the normal equations when the residual is linear.
Nonlinear Least Squares

Hence we seek \( b \in \mathbb{R}^n \) such that:

\[
J_r(b)^T r(b) = 0
\]

This has \( n \) equations, \( n \) unknowns; in general this is a nonlinear system that we have to solve iteratively.

An important recurring theme is that linear systems can be solved in “one shot,” whereas nonlinear generally requires iteration.

We will briefly introduce Newton’s method for solving this system and defer detailed discussion until the optimization section.
Nonlinear Least Squares

Recall Newton’s method for a function of one variable: find $x \in \mathbb{R}$ such that $f(x) = 0$

Let $x_k$ be our current guess, and $x_k + \Delta x = x$, then Taylor expansion gives

$$0 = f(x_k + \Delta x) = f(x_k) + \Delta x f'(x_k) + O((\Delta x)^2)$$

It follows that $f'(x_k) \Delta x \approx -f(x_k)$ (approx. since we neglect the higher order terms)

This motivates Newton’s method: $f'(x_k) \Delta x_k = -f(x_k)$, where $x_{k+1} = x_k + \Delta x_k$
Nonlinear Least Squares

This argument generalizes directly to functions of several variables

For example, suppose $F : \mathbb{R}^n \to \mathbb{R}^n$, then find $x$ s.t. $F(x) = 0$ by

$$J_F(x_k) \Delta x_k = -F(x_k)$$

where $J_F$ is the Jacobian of $F$, $\Delta x_k \in \mathbb{R}^n$, $x_{k+1} = x_k + \Delta x_k$
Nonlinear Least Squares

In the case of nonlinear least squares, to find a stationary point of $\phi$ we need to find $b$ such that $J_r(b)^T r(b) = 0$

That is, we want to solve $F(b) = 0$ for $F(b) \equiv J_r(b)^T r(b)$

We apply Newton’s Method, hence need to find the Jacobian, $J_F$, of the function $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$
Consider \( \frac{\partial F_i}{\partial b_j} \) (this will be the \( ij \) entry of \( J_F \)):

\[
\frac{\partial F_i}{\partial b_j} = \frac{\partial}{\partial b_j} \left( J_r(b)^T r(b) \right)_i
= \frac{\partial}{\partial b_j} \sum_{k=1}^{m} \frac{\partial r_k}{\partial b_i} r_k
= \sum_{k=1}^{m} \frac{\partial r_k}{\partial b_i} \frac{\partial r_k}{\partial b_j} + \sum_{k=1}^{m} \frac{\partial^2 r_k}{\partial b_i \partial b_j} r_k
\]
Gauss–Newton Method

It is generally a pain to deal with the second derivatives in the previous formula, second derivatives get messy!

Key observation: As we approach a good fit to the data, the residual values $r_k(b)$, $1 \leq k \leq m$, should be small

Hence we omit the term $\sum_{k=1}^{m} r_k \frac{\partial^2 r_k}{\partial b_i \partial b_j}$.
Gauss–Newton Method

Note that \( \sum_{k=1}^{m} \frac{\partial r_k}{\partial b_j} \frac{\partial r_k}{\partial b_i} = (J_r(b)^T J_r(b))_{ij} \), so that when the residual is small \( J_F(b) \approx J_r(b)^T J_r(b) \)

Then putting all the pieces together, we obtain the iteration:
\[
b_{k+1} = b_k + \Delta b_k
\]
where
\[
J_r(b_k)^T J_r(b_k) \Delta b_k = -J(b_k)^T r(b_k), \quad k = 1, 2, 3, \ldots
\]

This is known as the Gauss–Newton Algorithm for nonlinear least squares
This looks similar to Normal Equations at each iteration, except now the matrix $J_r(b_k)$ comes from linearizing the residual

Gauss–Newton is equivalent to solving the linear least squares problem $J_r(b_k)\Delta b_k \simeq -r(b_k)$ at each iteration

This is a common refrain in Scientific Computing: Replace a nonlinear problem with a sequence of linearized problems
Computing the Jacobian

To use Gauss–Newton in practice, we need to be able to compute the Jacobian matrix $J_r(b_k)$ for any $b_k \in \mathbb{R}^n$

We can do this “by hand”, e.g. in our transmitter/receiver problem we would have:

$$[J_r(b)]_{ij} = -\frac{\partial}{\partial b_j} \sqrt{(b_1 - x_1^i)^2 + (b_2 - x_2^i)^2}$$

Differentiating by hand is feasible in this case, but it can become impractical if $r(b)$ is more complicated.

Or perhaps our mapping $b \rightarrow y$ is a “black box” — no closed form equations hence not possible to differentiate the residual!
So, what is the alternative to “differentiation by hand”?  

Finite difference approximation: for $h \ll 1$ we have

$$[J_r(b_k)]_{ij} \approx \frac{r_i(b_k + e_j h) - r_i(b_k)}{h}$$

Avoids tedious, error prone differentiation of $r$ by hand!

Also, can be used for differentiating “black box” mappings since we only need to be able to evaluate $r(b)$.
Gauss–Newton Method

We derived the Gauss–Newton algorithm method in a natural way:

- apply Newton’s method to solve $\nabla \phi = 0$
- neglect the second derivative terms that arise

However, Gauss–Newton is not widely used in practice since it doesn’t always converge reliably
A more robust variation of Gauss–Newton is the Levenberg–Marquardt Algorithm, which uses the update

$$[J^T(b_k)J(b_k) + \mu_k \text{diag}(S^T S)]\Delta b = -J(b_k)^T r(b_k)$$

where\(^1\) \(S = I\) or \(S = J(b_k)\), and some heuristic is used to choose \(\mu_k\).

This looks like our “regularized” underdetermined linear least squares formulation!

---

\(^1\)In this context \(\text{diag}(A)\) means “zero the off-diagonal part of \(A\)”
Levenberg–Marquardt Method

**Key point**: The regularization term $\mu_k \text{diag}(S^T S)$ improves the reliability of the algorithm in practice.

Levenberg–Marquardt is implemented in Python and Matlab’s optimization toolbox.

We need to pass the residual to the routine, and we can also pass the Jacobian matrix or ask for a finite-differenced Jacobian.

Now let’s solve our transmitter/receiver problem.
Python example: Using lsqnonlin.py we provide an initial guess (●), and converge to the solution (×)
Nonlinear Least Squares: Example

Levenberg–Marquardt minimizes $\phi(b)$, as we see from the contour plot of $\phi(b)$ below.

Recall $\times$ is the true transmitter location, $\times$ is our best-fit to the data; $\phi(\times) = 0.0248 < 0.0386 = \phi(\times)$.

These contours are quite different from what we get in linear problems.
Linear Least-Squares Contours

Two examples of linear least squares contours for
\[ \phi(b) = \| y - Ab \|_2^2, \ b \in \mathbb{R}^2 \]

In linear least squares \( \phi(b) \) is quadratic, hence contours are “hyperellipses”