Applied Mathematics 205

Unit I: Data Fitting

Lecturer: Dr. David Knezevic

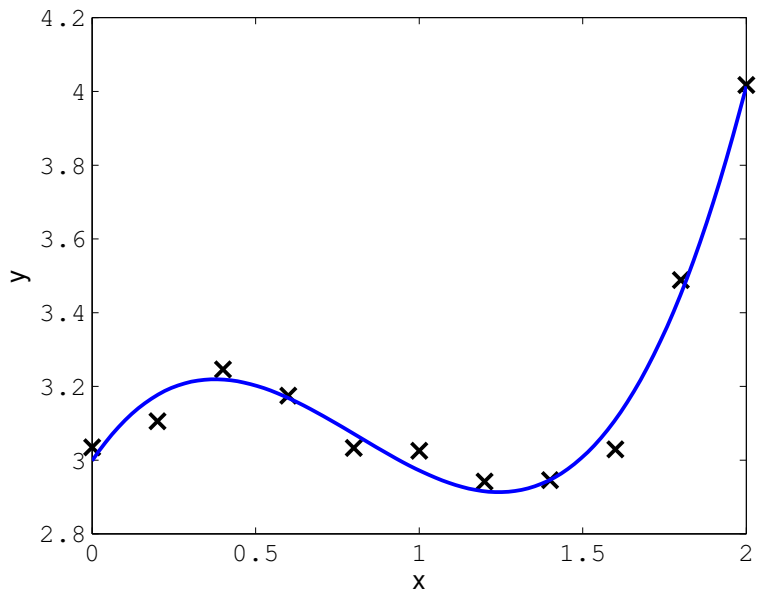
Unit I: Data Fitting

Chapter I.3: Linear Least Squares

The Problem Formulation

Recall that it can be advantageous to not fit data points exactly (e.g. due to experimental error), we don't want to "overfit"

Suppose we want to fit a cubic polynomial to 11 data points



Question: How do we do this?

The Problem Formulation

Suppose we have m constraints and n parameters with $m > n$ (e.g. $m = 11$, $n = 4$ on previous slide)

In terms of linear algebra, this is an overdetermined system $Ab = y$, where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^n$ (parameters), $y \in \mathbb{R}^m$ (data)

$$\begin{bmatrix} & \\ & A \\ & \end{bmatrix} \begin{bmatrix} \\ b \\ \\ \end{bmatrix} = \begin{bmatrix} \\ y \\ \\ \end{bmatrix}$$

i.e. we have a "tall, thin" matrix A

The Problem Formulation

In general, cannot be solved exactly (hence we will write $Ab \simeq y$); instead our goal is to minimize the residual, $r(b) \in \mathbb{R}^m$

$$r(b) \equiv y - Ab$$

A very effective approach for this is the method of least squares:[1]
Find parameter vector $b \in \mathbb{R}^n$ that minimizes $\|r(b)\|_2$

As we shall see, we minimize the 2-norm above since it gives us a differentiable function (we can then use calculus)

[1] Developed by Gauss and Legendre for fitting astronomical observations in the presence of experimental error

The Normal Equations

Goal is to minimize $\|r(b)\|_2$, recall that $\|r(b)\|_2 = \sqrt{\sum_{i=1}^{n} r_i(b)^2}$

The minimizing b is the same for $\|r(b)\|_2$ and $\|r(b)\|_2^2$, hence we consider the differentiable "objective function" $\phi(b) = \|r(b)\|_2^2$

$$\begin{aligned} \phi(b) &= \|r\|_2^2 = r^T r = (y - Ab)^T (y - Ab) \\ &= y^T y - y^T Ab - b^T A^T y + b^T A^T Ab \\ &= y^T y - 2b^T A^T y + b^T A^T Ab \end{aligned}$$

where last line follows from $y^T Ab = (y^T Ab)^T$, since $y^T Ab \in \mathbb{R}$

ϕ is a quadratic function of b, and is non-negative, hence a minimum must exist, (but not nec. unique, e.g. $f(b_1, b_2) = b_1^2$)

The Normal Equations

To find minimum of $\phi(b) = y^T y - 2b^T A^T y + b^T A^T A b$, differentiate wrt b and set to zero[2]

First, let's differentiate $b^T A^T y$ wrt b

That is, we want $\nabla(b^T c)$ where $c \equiv A^T y \in \mathbb{R}^n$:

$$b^T c = \sum_{i=1}^{n} b_i c_i \implies \frac{\partial}{\partial b_i}(b^T c) = c_i \implies \nabla(b^T c) = c$$

Hence $\nabla(b^T A^T y) = A^T y$

[2] We will discuss numerical optimization of functions of many variables in detail in Unit IV

The Normal Equations

Next consider $\nabla(b^T A^T A b)$ (note $A^T A$ is symmetric)

Consider $b^T M b$ for symmetric matrix $M \in \mathbb{R}^{n \times n}$

$$b^T M b = b^T \left(\sum_{j=1}^{n} m_{(:,j)} b_j \right)$$

From the product rule

$$\begin{aligned} \frac{\partial}{\partial b_k}(b^T M b) &= e_k^T \sum_{j=1}^{n} m_{(:,j)} b_j + b^T m_{(:,k)} \\ &= \sum_{j=1}^{n} m_{(k,j)} b_j + b^T m_{(:,k)} \\ &= m_{(k,:)} b + b^T m_{(:,k)} \\ &= 2 m_{(k,:)} b, \end{aligned}$$

where the last line follows from symmetry of M

Therefore, $\nabla(b^T M b) = 2Mb$, so that $\nabla(b^T A^T A b) = 2A^T A b$

The Normal Equations

Putting it all together, we obtain

$$\nabla\phi(b) = -2A^T y + 2A^T Ab$$

We set $\nabla\phi(b) = 0$ to obtain

$$-2A^T y + 2A^T Ab = 0 \Longrightarrow A^T Ab = A^T y$$

This square $n \times n$ system $A^T Ab = A^T y$ is known as the normal equations

The Normal Equations

For $A \in \mathbb{R}^{m \times n}$ with $m > n$, $A^T A$ is not invertible if and only if A is rank-deficient.[3]

Proof:

($\Rightarrow$) Suppose $A^T A$ is not invertible, then $\exists z \neq 0$ such that $A^T A z = 0$. Hence $z^T A^T A z = \|Az\|_2^2 = 0$, so that $Az = 0$. Therefore A is rank-deficient.

($\Leftarrow$) Suppose A is rank-deficient. $\exists z \neq 0$ such that $Az = 0$, hence $A^T A z = 0$, so that $A^T A$ is not invertible.

[3] Recall $A \in \mathbb{R}^{m \times n}$, $m > n$ is rank-deficient if columns are linearly dependent, i.e. $\exists z \neq 0$ s.t. $Az = 0$

The Normal Equations

We use the contrapositive to state this result in an equivalent, but perhaps clearer, way

Contrapositive: $P \implies Q \iff \neg Q \implies \neg P$

For example, Legal Seafoods slogan:
"If it isn't fresh, it isn't Legal" $\iff$ "If it's Legal then it's fresh"

Using the contrapositive in "both directions" in the result above gives: $A^T A$ is invertible if and only if A has full rank

The Normal Equations

Hence if A has full rank (i.e. $\text{rank}(A) = n$) we can solve the normal equations to find the unique least-squares minimizer b

However, in general it is a bad idea to solve the normal equations directly; it is not as numerically stable as some alternative methods

We will discuss better methods (QR factorization, SVD) in Unit II

Question: If we shouldn't use the normal equations directly, how do we actually solve least-squares problems in Matlab?

Matlab "backslash"

"Backslash" (\) is one of the most useful operators in Matlab

"Overloaded" to do different calculations in different contexts

Most common situation is "solve $Ax = b$" via $x = A\backslash b$ for square system (uses LU decomposition, cf. Unit II)

If system is over-determined, "backslash" finds least squares solution in a stable manner (uses QR factorization, cf. Unit II)

Least-squares polynomial fit

Example: 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 Lagrange like in I.2? 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

Matlab code for comparing "backslash" and normal equations for this least-squares problem:

```
format long
x = linspace(0,1,50)';
A = fliplr(vander(x));
A = A(:,1:12);
y = cos(4*x);

% solve normal equations
fprintf('cond(A'*A) = %d\n\n ', cond(A'*A))
b_normal = (A'*A) \ (A'*y)

% solve using 'backslash' (less rounding error)
b_backslash = A \ y
```

Least-squares polynomial fit

$\texttt{cond}(A^T A) = 1.354 \times 10^{16}$, hence normal equations are "singular to machine precision"

$$b_{\text{normal}} = \begin{bmatrix} 1.000000051508329 \\ -0.000015133093351 \\ -7.999431402147580 \\ -0.008391428014185 \\ 10.731053092678904 \\ -0.291236426826351 \\ -4.862157012040036 \\ -1.510203667008908 \\ 3.386344100793780 \\ -1.238285407662096 \\ 0.144069879639166 \\ -0.005390299320099 \end{bmatrix}, \quad b_{\text{backslash}} = \begin{bmatrix} 1.000000000996605 \\ -0.000000422742734 \\ -7.999981235694049 \\ -0.000318763130923 \\ 10.669430795224949 \\ -0.013820285245551 \\ -5.647075634537363 \\ -0.075316011985823 \\ 1.693606949690125 \\ 0.006032118434138 \\ -0.374241707313253 \\ 0.088040576742115 \end{bmatrix}$$

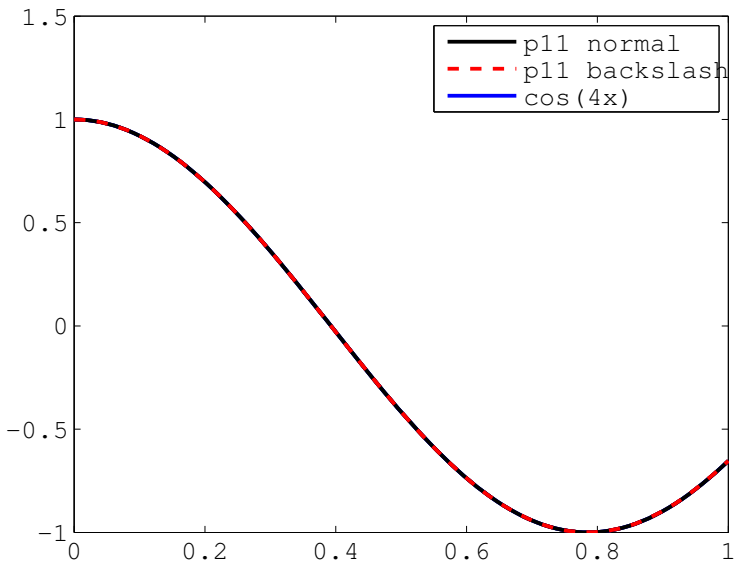
Rel. error wrt $b_{\text{backslash}}$ is large due to lack of numerical stability of the normal equations approach

Least-squares polynomial fit

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

$$\|r(b_{\text{normal}})\|_2 = \|y - Ab_{\text{normal}}\|_2 = 2.24 \times 10^{-7}$$

$$\|r(b_{\text{backslash}})\|_2 = \|y - Ab_{\text{backslash}}\|_2 = 8.00 \times 10^{-9}$$



We will discuss the distinction between *small residual* and *small error* in Unit II

Non-polynomial Least-squares fitting

So far we have dealt with approximations based on polynomials, but we can also develop non-polynomial approximations[4]

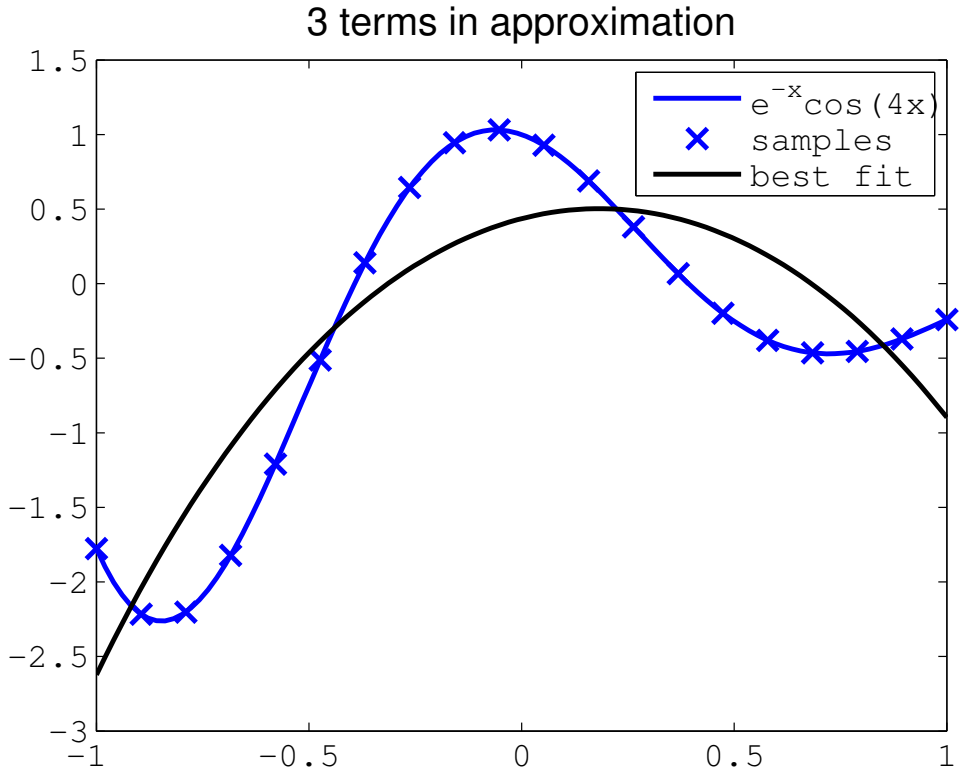
We just need the model to depend linearly on parameters

Matlab 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: $f_n(x;\gamma a + \sigma b) = \gamma f_n(x;a) + \sigma f_n(x;b)$)

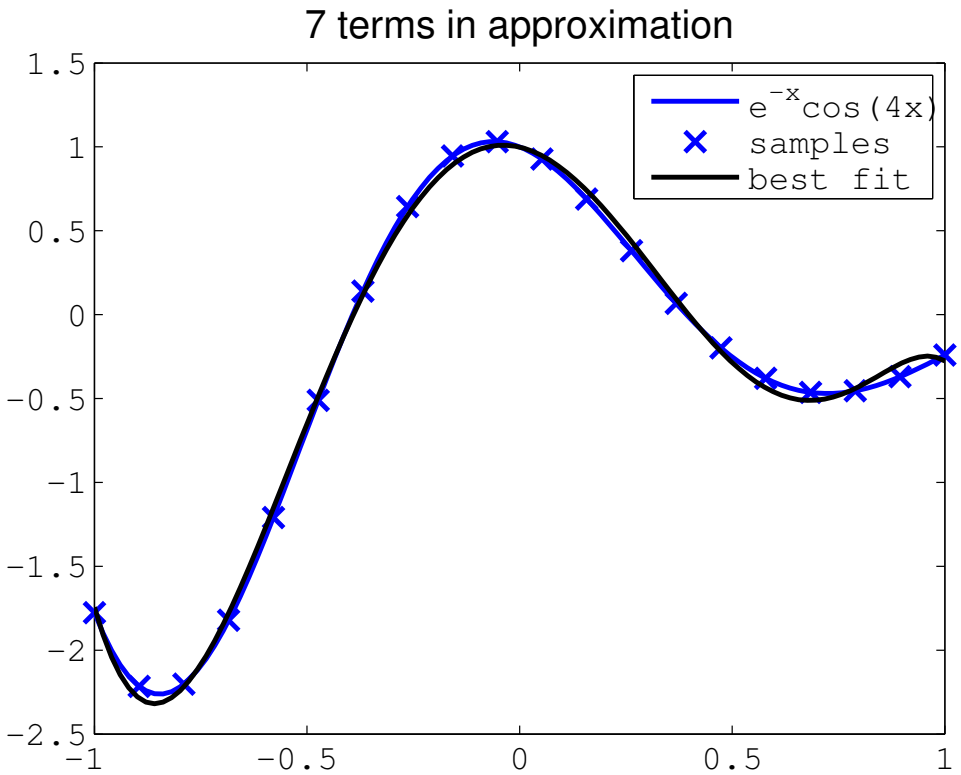
[4] Polynomials are natural for interpolation due to uniqueness of polynomial interpolant, but least-squares works well for other types of functions too

Non-polynomial Least-squares fitting



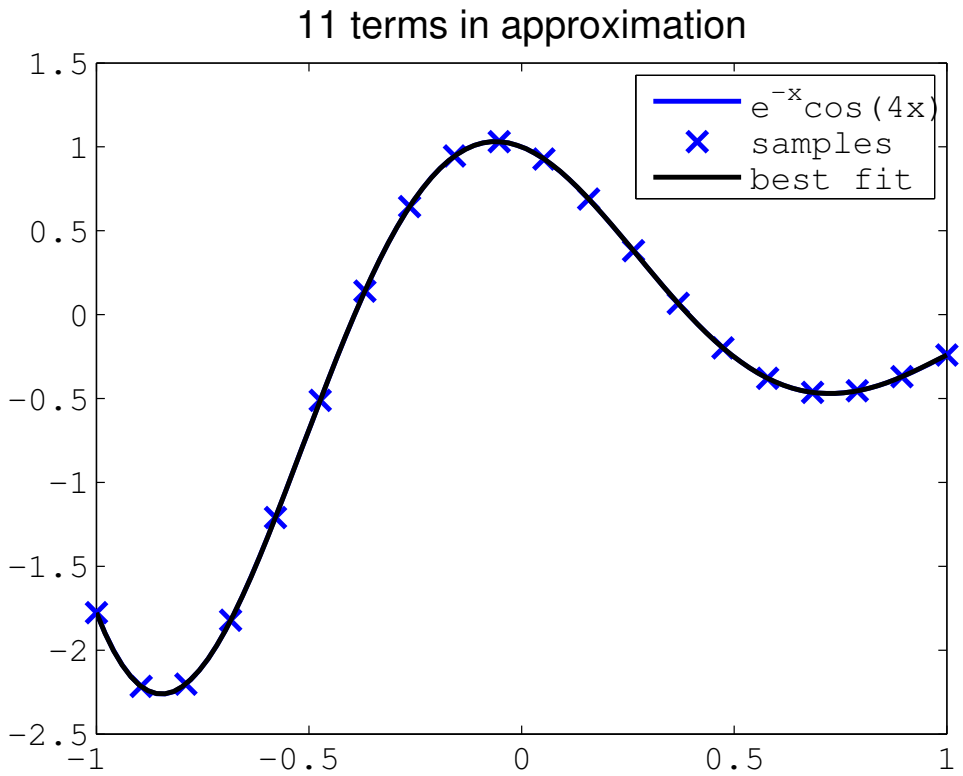
$$n = 1, \quad \frac{\|r(b)\|_2}{\|b\|_2} = 4.16 \times 10^{-1}$$

Non-polynomial Least-squares fitting



$$n = 3, \quad \frac{\|r(b)\|_2}{\|b\|_2} = 1.44 \times 10^{-3}$$

Non-polynomial Least-squares fitting



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

Pseudoinverse

Recall that from the normal equations we have:

$$A^T A b = A^T y$$

This motivates the idea of the "pseudoinverse" (`pinv` in Matlab) 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

We consider pseudoinverse in the "underdetermined" case shortly, and a more general definition based on the SVD is given in Unit II

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 $A^T A$ cannot be invertible!

Typical case: There are infinitely many vectors b that give $r(b) = 0$, we want to be able to select one of them

Underdetermined Least Squares

First idea, pose as a constrained optimization problem to find the feasible b with minimum 2-norm:

$$\begin{array}{ll} \text{minimize} & b^T b \\ \text{subject to} & Ab = y \end{array}$$

This can be treated using Lagrange multipliers (we will not discuss this now, see Unit IV)

Underdetermined Least Squares

We will show in Unit IV 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

Since we defer Lagrange multipliers until Unit IV, here we consider an alternative approach for "solving" the underconstrained case

Let's modify ϕ 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

Underdetermined Least Squares

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

If $S^T S$ is positive definite[5] then $(A^T A + S^T S)$ is invertible (this follows from the Rayleigh quotient, Unit V)

Simplest positive definite regularizer: $S = \mu \mathrm{I} \in \mathbb{R}^{n \times n}$ for $\mu \neq 0$

[5] $b \neq 0 \implies b^T S^T S b > 0$, see Assignment 1

Underdetermined Least Squares

Matlab 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 $\implies A \in \mathbb{R}^{5\times 12}$

As before, we express the polynomial using the monomial basis, hence A is a submatrix of a Vandermonde matrix

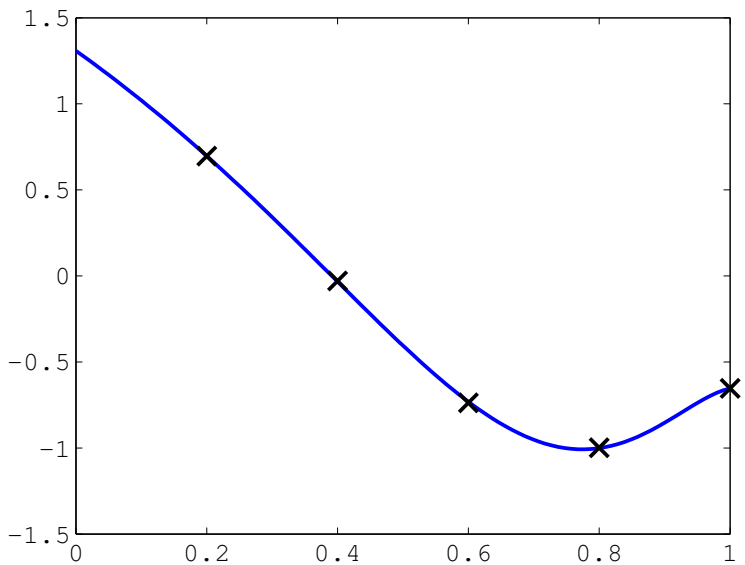
If we naively use the normal equations we see that $\text{cond}(A^T A) = 4.78 \times 10^{17}$, i.e. "singular to machine precision"!

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 $S = 0.001\mathrm{I}$ (i.e. $\mu = 0.001$)



$$\|r(b)\|_2 = 1.07 \times 10^{-4}$$

$$\|b\|_2 = 4.40$$

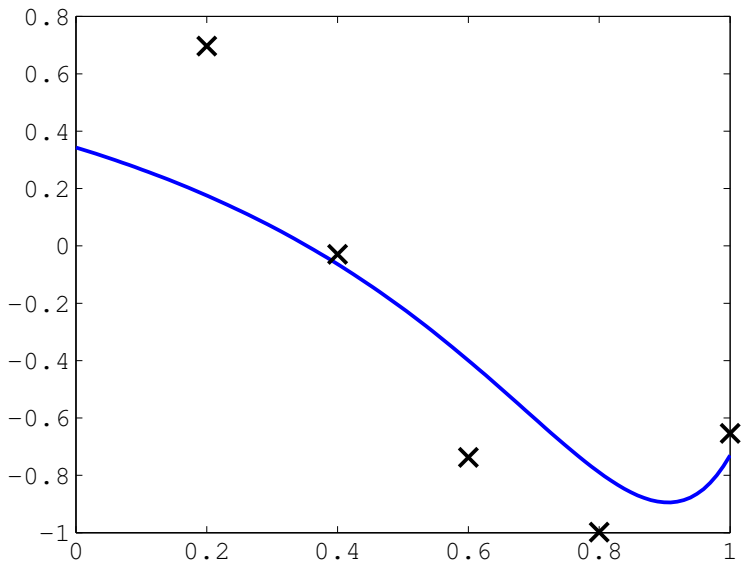
$$\mathrm{cond}(A^T A + S^T S) = 1.54 \times 10^7$$

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 $S = 0.5\mathrm{I}$ (i.e. $\mu = 0.5$)



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

$$\|b\|_2 = 1.15$$

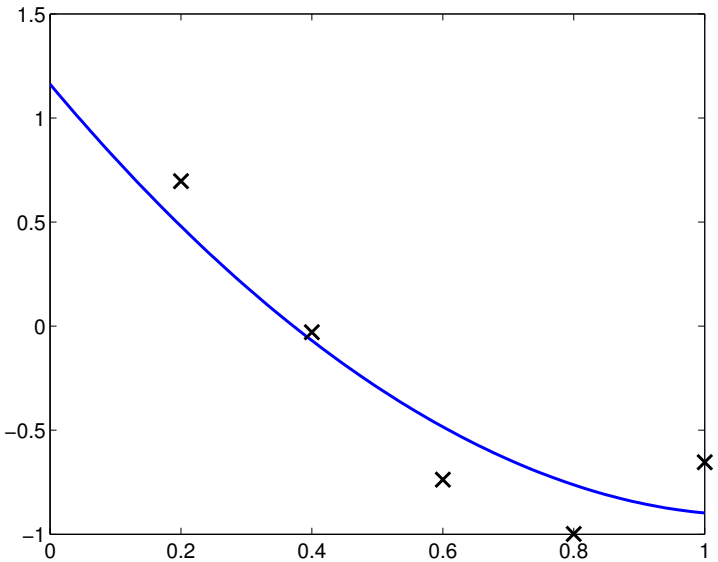
$$\mathrm{cond}(A^T A + S^T S) = 62.3$$

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



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

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

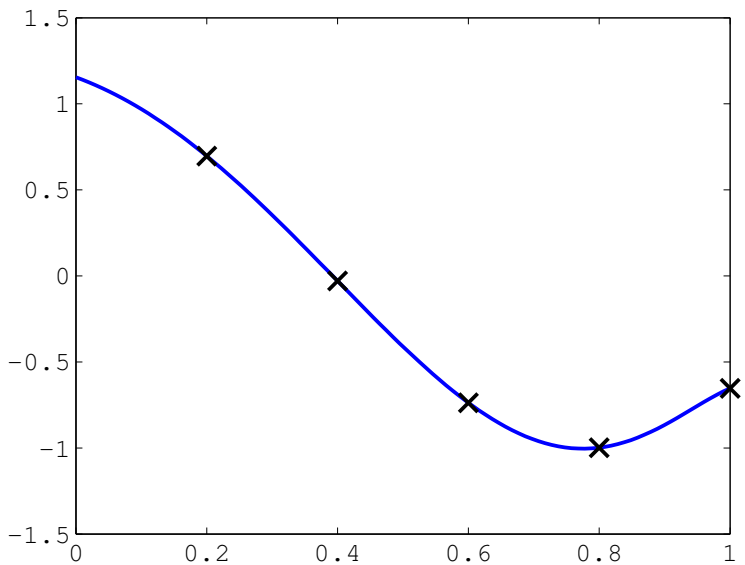
$$\text{cond}(A^T A + S^T S) = 5.90 \times 10^3$$

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 $x \in [0, 1]$

Try using Matlab's "backslash"



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

$$\|b\|_2 = 7.18$$

"Backslash" gives Lagrange multiplier based solution, hence satisfies the constraints to machine precision