

CAAM 453/553 Fall 2010
Lecture 19: Continuous Least Squares
Approximation

T. Warburton [†]

[†]Adapted from original lecture notes by Prof. M. Embree

We began §3.1 with the problem of approximating some $f \in C[a, b]$ with a polynomial $p \in \mathcal{P}_n$ at the discrete points x_0, x_1, \dots, x_m for some $m \geq n$.

This example motivated our study of discrete least squares problems (a subject with many other diverse applications), but the choice of the m points is somewhat arbitrary.

Suppose we uniformly distribute these approximation points over $[a, b]$: set $h_m := (b - a)/m$ and let $x_k = a + kh_m$.

The least squares error formula, when scaled by h_m , takes the form of a Riemann sum that, in the $m \rightarrow \infty$ limit, approximates an integral:

$$\lim_{m \rightarrow \infty} h_m \sum_{k=0}^m (f(x_k) - p(x_k))^2 = \int_a^b (f(x) - p(x))^2 dx.$$

That is, in the limit of infinitely many uniformly spaced approximation points, we are actually minimizing an integral, rather than a sum.

This Lecture

In this lecture, we will see how to pose such problems as a matrix problem of dimension $(n + 1)$ -by- $(n + 1)$, instead of a discrete least squares problem with matrix of dimension ' ∞ -by- $(n + 1)$ '.

3.3.2. Least squares minimization via calculus

Given some $f \in C[a, b]$, the basic L^2 approximation problem seeks the polynomial $p \in \mathcal{P}_n$ that minimizes the error $f - p$ in the L^2 norm.

In symbols:

$$\min_{p \in \mathcal{P}_n} \|f - p\|_{L^2}.$$

We shall denote the polynomial that attains this minimum by p_* .

We can solve this minimization problem using basic calculus.

Consider this example for $n = 1$, where we optimize the error over polynomials of the form $p(x) = c_0 + c_1x$.

Note that $\|f - p\|_{L^2}$ will be minimized by the same polynomial as $\|f - p\|_{L^2}^2$.

Continuous Least Squares Polynomial Fit with Linear Polynomials

For any given $p \in \mathcal{P}_1$, the error function is given by

$$\begin{aligned}
 E(c_0, c_1) &:= \|f(x) - (c_0 + c_1x)\|_{L^2}^2 \\
 &= \int_a^b (f(x) - c_0 - c_1x)^2 dx \\
 &= \int_a^b \left(f(x)^2 - 2f(x)(c_0 + c_1x) + (c_0^2 + 2c_0c_1x + c_1^2x^2) \right) dx \\
 &= \int_a^b f(x)^2 dx - 2c_0 \int_a^b f(x) dx - 2c_1 \int_a^b xf(x) dx \\
 &\quad + c_0^2(b-a) + c_0c_1(b^2 - a^2) + \frac{1}{3}c_1^2(b^3 - a^3).
 \end{aligned}$$

To find the optimal polynomial, p_* , we need to optimize E over c_0 and c_1 , i.e., we must find the values of c_0 and c_1 for which

$$\frac{\partial E}{\partial c_0} = \frac{\partial E}{\partial c_1} = 0.$$

First, compute

$$\frac{\partial E}{\partial c_0} = -2 \int_a^b f(x) dx + 2c_0(b-a) + c_1(b^2 - a^2)$$

$$\frac{\partial E}{\partial c_1} = -2 \int_a^b xf(x) dx + c_0(b^2 - a^2) + \frac{2}{3}c_1(b^3 - a^3).$$

Setting these partial derivatives equal to zero yields

$$2c_0(b-a) + c_1(b^2 - a^2) = 2 \int_a^b f(x) dx$$

$$c_0(b^2 - a^2) + \frac{2}{3}c_1(b^3 - a^3) = 2 \int_a^b xf(x) dx.$$

$$2c_0(b-a) + c_1(b^2 - a^2) = 2 \int_a^b f(x) dx$$

$$c_0(b^2 - a^2) + \frac{2}{3}c_1(b^3 - a^3) = 2 \int_a^b xf(x) dx.$$

These equations, linear in the unknowns c_0 and c_1 , can be written in the matrix form

$$\begin{bmatrix} 2(b-a) & b^2 - a^2 \\ b^2 - a^2 & \frac{2}{3}(b^3 - a^3) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} 2 \int_a^b f(x) dx \\ 2 \int_a^b xf(x) dx \end{bmatrix}.$$

When $b \neq a$ this system always has a unique solution.

i.e. we solve

$$\begin{bmatrix} 2(b-a) & b^2 - a^2 \\ b^2 - a^2 & \frac{2}{3}(b^3 - a^3) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} 2 \int_a^b f(x) dx \\ 2 \int_a^b xf(x) dx \end{bmatrix}.$$

and the resulting c_0 and c_1 are the coefficients for the monomial-basis expansion of the least squares approximation $p_* \in \mathcal{P}_1$ to f on $[a, b]$.

Example: $f(x) = e^x$

We apply this result to the function $f(x) = e^x$ for $x \in [0, 1]$.

Since

$$\int_0^1 e^x dx = e - 1, \quad \int_0^1 xe^x dx = [e^x(x-1)]_{x=0}^1 = 1,$$

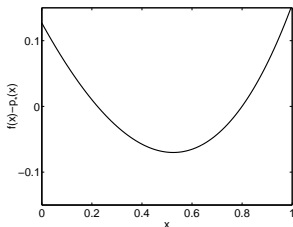
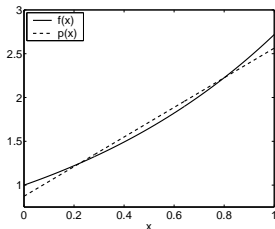
we must solve the system

$$\begin{bmatrix} 2 & 1 \\ 1 & \frac{2}{3} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} 2e - 2 \\ 2 \end{bmatrix}.$$

The desired solution is

$$c_0 = 4e - 10, \quad c_1 = 18 - 6e.$$

Below we show a plot of this approximation (left), and the error $f(x) - p_*(x)$.



We can see from these pictures that the approximation looks decent to the eye, but the error is not terribly small.

In fact, $\|f - p_*\|_{L^2} = 0.06277 \dots$

We can decrease that error by increasing the degree of the approximating polynomial.

Just as we used a 2-by-2 linear system to find the best linear approximation, a general $(n + 1)$ -by- $(n + 1)$ linear system can be constructed to yield the L^2 -optimal degree- n approximation.

3.3.3. General polynomial bases

Note that we performed the above minimization in the monomial basis: $p(x) = c_0 + c_1x$ is a linear combination of 1 and x .

Our experience with interpolation suggests that different choices for the basis may yield approximation algorithms with superior numerical properties.

Thus, we develop the form of the approximating polynomial in an arbitrary basis.

Suppose $\{\phi_k\}_{k=0}^n$ is a basis for \mathcal{P}_n .

Then any $p \in \mathcal{P}_n$ can be written as

$$p(x) = \sum_{k=0}^n c_k \phi_k(x).$$

The error expression takes the form

$$\begin{aligned} E(c_0, \dots, c_n) &:= \|f(x) - p(x)\|_{L^2}^2 \\ &= \int_a^b \left(f(x) - \sum_{k=0}^n c_k \phi_k(x) \right)^2 dx \\ &= \langle f, f \rangle - 2 \sum_{k=0}^n c_k \langle f, \phi_k \rangle + \sum_{k=0}^n \sum_{\ell=0}^n c_k c_\ell \langle \phi_k, \phi_\ell \rangle. \end{aligned}$$

Stationarity Conditions

$$E(c_0, \dots, c_n) = \langle f, f \rangle - 2 \sum_{k=0}^n c_k \langle f, \phi_k \rangle + \sum_{k=0}^n \sum_{\ell=0}^n c_k c_\ell \langle \phi_k, \phi_\ell \rangle.$$

As before, compute $\partial E / \partial c_j$ for $j = 0, \dots, n$:

$$\frac{\partial E}{\partial c_j} = -2 \langle f, \phi_j \rangle + \sum_{k=0}^n 2c_k \langle \phi_k, \phi_j \rangle.$$

Setting $\partial E / \partial c_j = 0$ gives the $n + 1$ equations

$$\langle f, \phi_j \rangle = \sum_{k=0}^n c_k \langle \phi_k, \phi_j \rangle.$$

The $n + 1$ equations $\langle f, \phi_j \rangle = \sum_{k=0}^n c_k \langle \phi_k, \phi_j \rangle$ are simply a system of linear algebraic equations, which can be written in the matrix form

$$\begin{bmatrix} \langle \phi_0, \phi_0 \rangle & \langle \phi_0, \phi_1 \rangle & \cdots & \langle \phi_0, \phi_n \rangle \\ \langle \phi_1, \phi_0 \rangle & \langle \phi_1, \phi_1 \rangle & & \vdots \\ \vdots & & \ddots & \vdots \\ \langle \phi_n, \phi_0 \rangle & \langle \phi_n, \phi_1 \rangle & \cdots & \langle \phi_n, \phi_n \rangle \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} \langle f, \phi_0 \rangle \\ \langle f, \phi_1 \rangle \\ \vdots \\ \langle f, \phi_n \rangle \end{bmatrix},$$

which we shall denote as $\mathbf{H}\mathbf{c} = \mathbf{b}$.

\mathbf{H} is typically referred to as a *mass matrix* in finite element literature.

The Hilbert Matrix (Mass Matrix for Monomials)

Suppose we apply this method on the interval $[a, b] = [0, 1]$ with the monomial basis, $\phi_k(x) = x^k$.

In that case,

$$\langle \phi_k, \phi_j \rangle = \langle x^k, x^j \rangle = \int_0^1 x^{j+k} dx = \frac{1}{j+k+1},$$

and the coefficient matrix has an elementary structure.

In fact, this is a form of the notorious *Hilbert matrix*.[§]

[§]See M.-D. Choi, 'Tricks or treats with the Hilbert matrix,' *American Math. Monthly* 90 (1983) 301–312.

It is exceptionally difficult to obtain accurate solutions to $\mathbf{H}\mathbf{c} = \mathbf{b}$ with the Hilbert matrix in floating point arithmetic, reflecting the fact that the monomials are a poor basis for \mathcal{P}_n on $[0, 1]$.

Let \mathbf{H} denote the $n + 1$ -dimensional Hilbert matrix, and suppose \mathbf{b} is constructed so that the exact solution to the system $\mathbf{H}\mathbf{c} = \mathbf{b}$ is $\mathbf{c} = (1, 1, \dots, 1)^T$. Let $\hat{\mathbf{c}}$ denote computed solution to the system in MATLAB.

Ideally the forward error $\|\mathbf{c} - \hat{\mathbf{c}}\|_2$ will be nearly zero (if the rounding errors incurred while constructing \mathbf{b} and solving the system are small).

Ideally the forward error $\|\mathbf{c} - \hat{\mathbf{c}}\|_2$ will be nearly zero.

Unfortunately, this is not the case – entirely consistent with our analysis of the sensitivity of linear systems, studied in Section 1.4.2.

n	$\kappa(\mathbf{H})$	$\ \mathbf{c} - \hat{\mathbf{c}}\ _2$
5	1.495×10^7	7.548×10^{-11}
10	1.603×10^{14}	0.01288
15	4.380×10^{17}	12.61
20	1.251×10^{18}	46.9

Clearly these errors are not acceptable!

The last few 2-norm condition numbers are in fact smaller than they ought to be, a consequence of the fact that MATLAB is not computing the singular value decomposition of the Hilbert matrix exactly.

The standard algorithm for computing singular values obtains answers with small *absolute* accuracy, but not small *relative* accuracy.

Thus we expect that singular values smaller than about $10^{-16} \|\mathbf{H}\|_2$ may not even be computed to the correct order of magnitude.

In the next lecture, we will see how better-conditioned bases for \mathcal{P}_n yield matrices \mathbf{H} for which we can solve $\mathbf{H}\mathbf{x} = \mathbf{b}$ much more accurately.

3.3.4. Connection to discrete least squares

bf Q: Why did the continuous least squares approximation problem studied above directly lead to a square $(n + 1) \times (n + 1)$ linear system, while the discrete least squares problem introduced in Lecture 16 led to an $(m + 1) \times (n + 1)$ least squares problem?

In the discrete case, we seek to minimize $\|\mathbf{c} - \mathbf{A}\mathbf{f}\|_2$, where
(using the monomial basis)

$$\mathbf{A} = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \cdots & x_m^n \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ \vdots \\ f(x_m) \end{bmatrix}.$$

Normal Equations for Discrete Least Squares

We have seen that this discrete problem can be solved via the *normal equations*

$$\mathbf{A}^* \mathbf{A} \mathbf{c} = \mathbf{A}^* \mathbf{f}.$$

Now compute

$$\mathbf{A}^* \mathbf{f} = \begin{bmatrix} \sum_{k=0}^n f(x_k) \\ \sum_{k=0}^n x_k f(x_k) \\ \sum_{k=0}^n x_k^2 f(x_k) \\ \vdots \\ \sum_{k=0}^n x_k^n f(x_k) \end{bmatrix} \in \mathbb{C}^{n+1}.$$

Notice that if $m + 1$ approximation points are uniformly spaced over $[a, b]$, $x_k = a + kh_m$ for $h_m = (b - a)/m$, we have

$$\lim_{m \rightarrow \infty} h_m \mathbf{A}^* \mathbf{f} = \begin{bmatrix} \int_a^b f(x) dx \\ \int_a^b x f(x) dx \\ \int_a^b x^2 f(x) dx \\ \vdots \\ \int_a^b x^n f(x) dx \end{bmatrix} = \begin{bmatrix} \langle f, 1 \rangle \\ \langle f, x \rangle \\ \langle f, x^2 \rangle \\ \vdots \\ \langle f, x^n \rangle \end{bmatrix},$$

which is precisely the right hand side vector $\mathbf{b} \in \mathbb{C}^{n+1}$ obtained for the *continuous least squares problem*.

Similarly, the $(j + 1, k + 1)$ entry of the matrix $\mathbf{A}^* \mathbf{A} \in \mathbb{C}^{(n+1) \times (n+1)}$ for the discrete problem can be formed as

$$(\mathbf{A}^* \mathbf{A})_{j+1, k+1} = \sum_{\ell=0}^m x_{\ell}^j x_{\ell}^k = \sum_{\ell=0}^m x_{\ell}^{j+k},$$

and thus for uniform grids, we have in the limit that

$$\lim_{m \rightarrow \infty} h_m (\mathbf{A}^* \mathbf{A})_{j+1, k+1} = \int_a^b x^{j+k} dx = \langle x^j, x^k \rangle.$$

Thus in aggregate we have

$$\lim_{m \rightarrow \infty} h_m \mathbf{A}^* \mathbf{A} = \mathbf{H},$$

where \mathbf{H} is the matrix that arose in the continuous least squares problem.

We arrive at the following beautiful conclusion:

Asymptotic Equivalence of Discrete Least Squares to Continuous Least Squares

The normal equations $\mathbf{A}^* \mathbf{A} \mathbf{c} = \mathbf{A}^* \mathbf{f}$ formed for polynomial approximation by *discrete least squares* converges to *exactly the same* $(n + 1) \times (n + 1)$ system $\mathbf{H} \mathbf{c} = \mathbf{b}$ as we independently derived for polynomial approximation by *continuous least squares*.

In the latter case, calculus led us directly to the normal equation form of the solution.