Numerical Analysis – Lab 7

1 Data fitting by least-squares (and more)

Linear regression. We want to find a degree one polynomial $y = a_1 x + a_0$ to best fit the data $\{(x_i, y_i)\}_{i=1}^m$. The least-squares

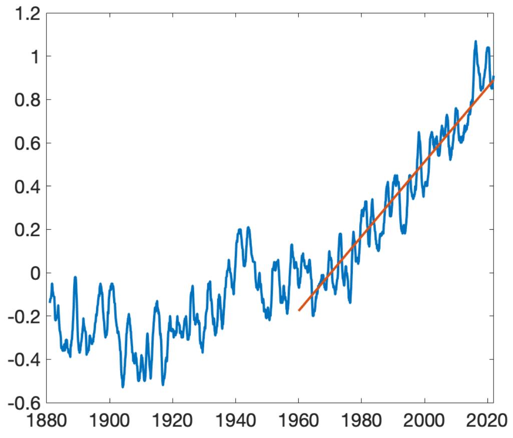
$$\min_{a_0,a_1} \sum_{i=1}^{m} |y_i - (a_1 x_i + a_0)|^2.$$

is the most used method, which results the normal system

$$\begin{pmatrix} \mathbf{1}\cdot\mathbf{1} & \mathbf{1}\cdot\mathbf{x} \\ \mathbf{x}\cdot\mathbf{1} & \mathbf{x}\cdot\mathbf{x} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} \mathbf{1}\cdot\mathbf{y} \\ \mathbf{x}\cdot\mathbf{y} \end{pmatrix}, \qquad \text{denoted } L\mathbf{a} = \mathbf{z},$$

where $\mathbf{1}$ is a vector with all its entries being 1, $\mathbf{x} = (x_i)$ and $\mathbf{y} = (y_i)$. We can solve the system by the MATLAB command `a=L\z`. Let us try it on a data set. The global mean temperatures of the surface air in each month of 1881-2021 are listed in "global.csv". We can read out and visualize the data:

```
T = readtable('global.csv');
XY = T{13:1704,:};
plot(XY(:,1),XY(:,2),'linewidth',2);
```



It appears to us that since the year 1960 there is a linear growth of the temperature against the year.

```
x = XY(949:end,1); y = XY(949:end,2);
m = numel(x); e = ones(m,1);
L = [e'*e  e'*x; x'*e x'*x];
z = [e'*y; x'*y]; a = L\z;
yh = a(1)+a(2)*x;
hold on; plot(x,yh,'linewidth',2);
rmse(y,yh) % only since 2022b and install
% Statistics and Machine Learning toolbox
```

Polynomial regression (optional). Let $\mathbb{P}_n$ be the space of polynomials of degree at most n and $\{\phi_k\}_{k=0}^n$ be a basis of $\mathbb{P}_n$. We want to find a polynomial in the form $p_n(x) = a_0\phi_0(x) + \ldots + a_n\phi_n(x)$ that fits the best to the data $\{(x_i, y_i)\}_{i=1}^m$. This is usually done by the least-squares criterion:

$$\min_{a_0,\ldots,a_n} \sum_{i=1}^{m} |y_i - p_n(x_i)|^2.$$

In linear algebra, we solve

$$\min_{\mathbf{a}} \|A\mathbf{a} - \mathbf{y}\|_2^2$$

with $\mathbf{a} = (a_k)$, $\mathbf{y} = (y_i)$ and

$$A = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \ldots & \phi_n(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \ldots & \phi_n(x_1) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_0(x_m) & \phi_1(x_m) & \ldots & \phi_n(x_m) \end{pmatrix}.$$

A solution is readily obtained by the MATLAB command `a=A\y`. Note that if the columns of A are linearly dependent (which may be incurred by that $m < n$), then the solution is not unique. One needs also to be cautious about the condition number of A. For example, if the monimial basis $\{\phi_k(x) = x^k\}_{k=0}^n$

is used, the resulting matrix A (called Vandermonde matrix) most likely has a condition number growing exponentially with n. A much better basis for large n exists, e.g. the Chebyshev basis $\{T_k(x) = \cos(k \arccos x)\}_{k=0}^n$ for $x \in [-1, 1]$. The Chebyshev basis is usually evaluated recursively

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

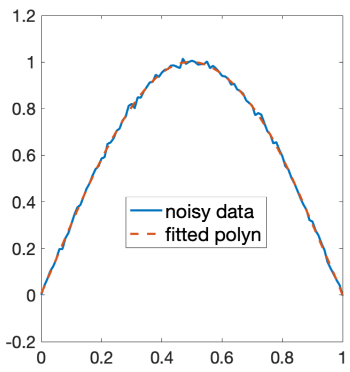
because cos and arccos are more expensive to compute. Our programs are two functions:

```
function a = chebfit(x,y,n)
x = x(:);
A = zeros(numel(x),n+1);
A(:,1) = 1; A(:,2) = x;
for k=2:n
    A(:,k+1) = 2*x.*A(:,k)-A(:,k-1);
end
a = A\y(:);
```

```
function y = chebval(a,x)
x = x(:); n = numel(a)-1;
A = zeros(numel(x),n+1);
A(:,1) = 1; A(:,2) = x;
for k=2:n
    A(:,k+1) = 2*x.*A(:,k)-A(:,k-1);
end
y = A*a(:);
```

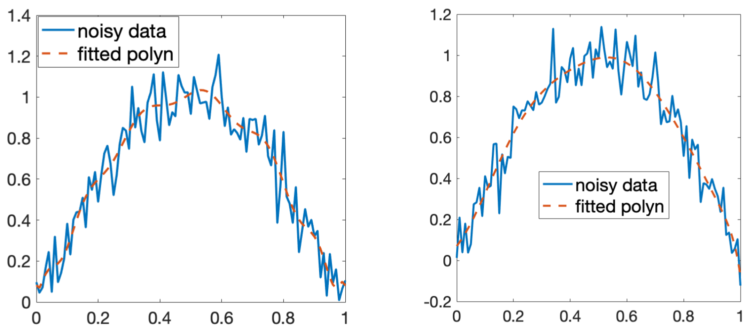
We try it on a manufactured example. The function values are sampled from $f(x) = \sin(\pi x)$ and with some noises added into the samples.

```
x = 0:0.01:1;
f = @(x) sin(pi*x);
y = f(x) + 1e-2*randn(size(x));
a = chebfit(x,y,10);
yy = chebval(a,x);
plot(x,y,x,yy,'--','linewidth',2)
set(gca,'fontsize',16);
legend('noisy data','fitted polyn','fontsize',18);
```



We see that the degree 10 polynomial regressed very well to the true model f underlying the noisy data.

You may increase the level of noises and the degree of polynomial to see what happens. You may find the warning message from MATLAB that “the matrix is rank deficient” i.e. the columns of A are linearly dependent. Moreover, with a very high degree, you may find the polynomial is learning (or affected significantly by) the noises– a phenomenon called *overfitting*. For example, with the maximum noise 0.1, the least-squares fitted 15th polynomial is shown below on the left.



A commonly used trick for treating the rank deficiency and the overfitting issue is to add a regularizer. For example, solve $\min_{\mathbf{a}} \|A\mathbf{a} - \mathbf{y}\|_2^2 + \epsilon\|\mathbf{a}\|_2^2$ for which the normal system reads $(A^T A + \epsilon I)\mathbf{a} = A^T\mathbf{y}$, where $\epsilon > 0$ is a parameter one can choose. The regularizer $\|\mathbf{a}\|_2^2$ is named after the Russian mathematician Andrey Tikhonov (1906-1993), and the regularized least-squares problem is also known as ridge regression. For $\epsilon = 1$, the maximum noise 0.1, the regularized least-squares fitted 15th polynomial is shown above on the right. It can be seen that the curve is smoother than in the left (ordinary least-squares without regularization).

Periodic model (optional). As we have seen for the interpolation in Lab 4, a periodic function may be modelled by trigonometric polynomials or periodic splines. Here, rather than interpolating the data, we fit the model to the data. We think of $\{(x_s, y_s)\}_{s=1}^m$ as sampled (maybe with noises) from a 1-period function f (the ground truth). We seek a trigonometric sum

$$p_n(x) = \sum_{k=-n}^{n} c_k \mathrm{e}^{\mathrm{i}k2\pi x}$$

that best fits the data $\{(x_s, y_s)\}_{s=1}^m$. Using the least-squares method, we would solve

$$\min_{\{c_k\}} \sum_{s=1}^{m} |p_n(x_s) - y_s|^2 .$$

In linear algebra, this has no difference from before for the polynomial regression:

$$\min_{\mathbf{c}} \|A\mathbf{c} - \mathbf{y}\|_2^2$$

with $\mathbf{c} = (c_k)$, $\mathbf{y} = (y_s)$ and for $\phi_k = \mathrm{e}^{\mathrm{i}k2\pi x}$,

$$A = \begin{pmatrix} \phi_{-n}(x_1) & \phi_{-n+1}(x_1) & \dots & \phi_n(x_1) \\ \phi_{-n}(x_2) & \phi_{-n+1}(x_2) & \dots & \phi_n(x_1) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_{-n}(x_m) & \phi_{-n+1}(x_m) & \dots & \phi_n(x_m) \end{pmatrix}.$$

Our programs are two functions:

```
function c = trigfit(x,y,n)
x = x(:);
A = zeros(numel(x),2*n+1);
A(:,n+1) = 1; w = exp(1i*2*pi*x);
for k=-1:-1:-n
    A(:,n+1+k) = A(:,n+2+k)./w;
end
for k=1:n
    A(:,n+1+k) = A(:,n+k).*w;
end
c = A\y(:);
```

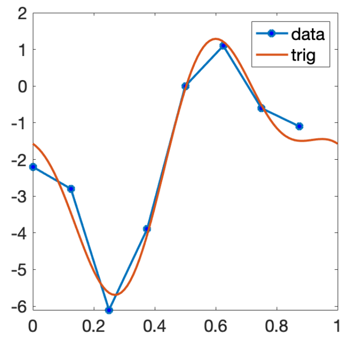
```
function y = trigval(c,x)
x = x(:); n = floor((numel(c)-1)/2);
A = zeros(numel(x),2*n+1);
A(:,n+1) = 1; w = exp(1i*2*pi*x);
for k=-1:-1:-n
    A(:,n+1+k) = A(:,n+2+k)./w;
end
for k=1:n
    A(:,n+1+k) = A(:,n+k).*w;
end
y = A*c(:);
```

The recorded temperatures in Washington, D.C., on January 1, 2001 are listed in the table.

time	12mid	3am	6am	9am	12noon	3pm	6pm	9pm
t	0	$\frac{1}{8}$	$\frac{2}{8}$	$\frac{3}{8}$	$\frac{4}{8}$	$\frac{5}{8}$	$\frac{6}{8}$	$\frac{7}{8}$
temp (C)	-2.2	-2.8	-6.1	-3.9	0.0	1.1	-0.6	-1.1

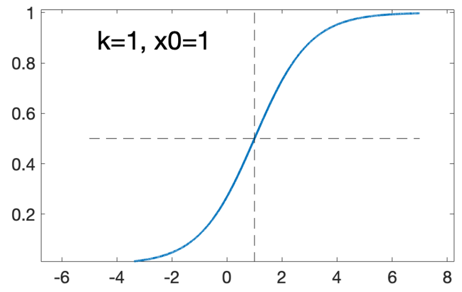
We fit a second degree trigonometric polynomial to the data:

```
t = 0:1/8:7/8;
T = [-2.2 -2.8 -6.1 -3.9 0 1.1 -0.6 -1.1];
c = trigfit(t,T,2);
tt = 0:0.01:1;
TT = real(trigval(c,tt));
plot(t,T,'o-',tt,TT,'linewidth',2,'MarkerFaceColor','b');
legend('data','trig','fontsize',16);
set(gca,'fontsize',16);
```



Logistic regression (optional). This actually goes beyond the least-squares method. The logistic function (also known as sigmoid)

$$p(x) = \frac{1}{1+\mathrm{e}^{-k(x-x_0)}}$$



resembles the shape of arctan. It can represent the conditional probability of an event given an x value. So it is often used in a binary classifier, and since $p > 0.5$ if and only if $x > x_0$, the point x_0 can be thought of as the separating boundary for the two classes $\{x > x_0\}$ ($\{x < x_0\}$) corresponding to that the event does (not) likely happen.

The logistic regression is to find the two parameters k and x_0 that best fits the given data $(x_i, y_i)_{i=1}^m$. For the classification problem, $y_i = 1, 0$ representing that the event happens or not. Note that one can also replace $k(x - x_0)$ for the linear classifier with a polynomial $p_n(x)$ for a nonlinear classifier (then the two classes become $\{p_n(x) < 0\}$ and $\{p_n(x) > 0\}$). The common criterion for fitting the logistic model is the maximum likelihood. That is, maximizing the probability of observing the data:

$$\left(\prod_{i:\ y_i=1} p(x_i)\right)\left(\prod_{i:\ y_i=0} (1-p(x_i))\right).$$

After taking the logarithm and making use of $y_i \in \{0, 1\}$, we can rewrite the objective function as

$$\ell = \left(\sum_{i:\ y_i=1} \log p(x_i)\right) + \left(\sum_{i:\ y_i=0} \log(1-p(x_i))\right) = \sum_{i=1}^{m} \left(y_i \log p(x_i) + (1-y_i)\log(1-p(x_i))\right).$$

For convenience, the logistic function is rewritten as

$$p(x) = \frac{1}{1+\mathrm{e}^{-(a_1x+a_0)}}.$$

So our goal is to

$$\max_{a_0,a_1} \ell(a_0, a_1).$$

The gradient of ℓ about (a_0, a_1) is

$$\nabla\ell = (\frac{\partial\ell}{\partial a_0}, \frac{\partial\ell}{\partial a_1}).$$

To facilitate the calculations, let $z = a_1x + a_0$ and note that

$$1-p(x) = \frac{\mathrm{e}^{-z}}{1+\mathrm{e}^{-z}} = \frac{1}{1+\mathrm{e}^{z}}, \quad \frac{\mathrm{e}^{z}}{1+\mathrm{e}^{z}} = p(x), \quad \frac{\partial(\log p(x))}{\partial z} = (1+\mathrm{e}^{-z})\frac{\mathrm{e}^{-z}}{(1+\mathrm{e}^{-z})^2} = 1-p(x),$$

$$\frac{\partial(\log(1-p(x)))}{\partial z} = (1+\mathrm{e}^{z})\frac{-\mathrm{e}^{z}}{(1+\mathrm{e}^{z})^2} = -p(x).$$

Hence,

$$\frac{\partial\ell}{\partial a_0} = \sum_{i=1}^{m}(y_i - p(x_i)), \quad \frac{\partial\ell}{\partial a_1} = \sum_{i=1}^{m}(y_i - p(x_i))x_i, \quad \frac{\partial^2\ell}{\partial a_0^2} = -\sum_{i=1}^{m}(1-p(x_i))p(x_i),$$

$$\frac{\partial^2\ell}{\partial a_0\partial a_1} = -\sum_{i=1}^{m}(1-p(x_i))p(x_i)x_i, \quad \frac{\partial^2\ell}{\partial a_1^2} = -\sum_{i=1}^{m}(1-p(x_i))p(x_i)x_i^2.$$

So the Hessian matrix

$$H = \begin{pmatrix} \frac{\partial^2 \ell}{\partial a_0^2} & \frac{\partial^2 \ell}{\partial a_0 \partial a_1} \\ \frac{\partial^2 \ell}{\partial a_0 \partial a_1} & \frac{\partial^2 \ell}{\partial a_1^2} \end{pmatrix} = - \begin{pmatrix} \sum_{i=1}^m (1-p(x_i))p(x_i) & \sum_{i=1}^m (1-p(x_i))p(x_i)x_i \\ \sum_{i=1}^m (1-p(x_i))p(x_i)x_i & \sum_{i=1}^m (1-p(x_i))p(x_i)x_i^2 \end{pmatrix}$$

is negative definite. Indeed, we need only to verify all the principal minor determinants of the minus Hessian $-H$ are positive. First, the (1,1)-entry of $-H$ is positive. Second, denote by $u_i = \sqrt{(1-p(x_i))p(x_i)}$, $v_i = u_i x_i$, then the determinant of $-H$ is

$$(\sum_{i=1}^m u_i^2)(\sum_{i=1}^m v_i^2) - (\sum_{i=1}^m u_i v_i)^2 = \|\mathbf{u}\|_2^2 \|\mathbf{v}\|_2^2 - (\mathbf{u}\cdot\mathbf{v})^2 \geq 0$$

where the equality can not happen for $u_i > 0$ and "$x_i,\ i = 1, .., m$ are not all equal". That proved, the objective function $\ell = \ell(a_0, a_1)$ is strictly concave, and the maximum of it is attained at the unique stationary point i.e. the zero point of the gradient:

$$0 = \frac{\partial \ell}{\partial a_0} = \sum_{i=1}^m (y_i - p(x_i)) = \sum_{i=1}^m (y_i - \frac{1}{1+\mathrm{e}^{-(a_1 x_i + a_0)}}),$$

$$0 = \frac{\partial \ell}{\partial a_1} = \sum_{i=1}^m (y_i - p(x_i))x_i = \sum_{i=1}^m (y_i - \frac{1}{1+\mathrm{e}^{-(a_1 x_i + a_0)}})x_i.$$

This nonlinear system of $\mathbf{a} = (a_0, a_1)^T$ can be solved by the (multivariate) Newton method

$$\mathbf{a}_{n+1} = \mathbf{a}_n - H(\mathbf{a}_n)^{-1} \nabla \ell(\mathbf{a}_n).$$

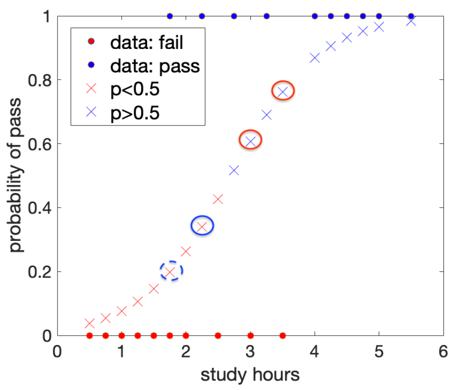
It is also common in machine learning to use the slower but simpler method– gradient descent. But the descent is for minimization, while we are maximizing ℓ. So we can think of minimizing $-\ell$:

$$\mathbf{a}_{n+1} = \mathbf{a}_n - \alpha_n \nabla(-\ell)(\mathbf{a}_n) = \mathbf{a}_n + \alpha_n \nabla \ell(\mathbf{a}_n)$$

with the step size (learning rate) $\alpha_n > 0$. We implement the gradient method as follows.

```
function a = logisfit(x,y,alpha,maxIt,gradtol)
a = [1; 1]; x= x(:); y = y(:);
for it = 1:maxIt
  p = 1./(1+exp(-(a(2)*x+a(1))));
  dl = [sum(y-p); sum((y-p).*x)];
  a = a + alpha*dl;
  if norm(dl,2)<gradtol
      break;
  end
end

function p = logisval(a,x)
p = 1./(1+exp(-(a(2)*x+a(1))));
```



Now let us consider the following data:

Hours	0.50	0.75	1.00	1.25	1.50	1.75	1.75	2.00	2.25	2.50	2.75	3.00	3.25	3.50	4.00	4.25	4.50	4.75	5.00	5.50
Pass	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1

We fit the logistic model to the data and check whether its prediction matches the data (codes are below). The result is visualized in the above figure. We see that the model has made four wrong predictions on the data (all of which are used as the training set). Note that there are two data at $x = 1.75$: one is pass and the other is fail. You may try to feed part of the data to the model training (fitting) and test the model on the rest part of the data. The mismatch of the model and the data indicates the single explanatory variable "study hours" is not sufficient for explaining the categorical variable "pass exam".

```
x = [0.50 0.75 1.00 1.25 1.50 1.75 1.75 2.00 2.25 2.50 2.75 3.00 3.25 3.50 4.00 ...
    4.25 4.50 4.75 5.00 5.50];
y = [0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 1 1 1 1];
a = logisfit(x,y,0.01,1000,1e-10); % feed all of (x,y) to the model training
p = logisval(a,x); % check the prediction of the model on the training set

plot(x(y==0),y(y==0),'o','MarkerFaceColor','r'); hold on
plot(x(y~=0),y(y~=0),'o','MarkerFaceColor','b');

plot(x(p<0.5),p(p<0.5),'rx','MarkerSize',10);
plot(x(p>0.5),p(p>0.5),'bx','MarkerSize',10);
xlabel('study hours'); ylabel('probability of pass'); set(gca,'fontsize',16);
legend('data: fail', 'data: pass', 'p<0.5', 'p>0.5', 'fontsize',18);
```

2 Function approximation by continuous least-squares

From discrete to continuous least-squares (optional). For data fitting problems, we have m data points $(x_i, f(x_i))_{i=1}^m$ on an interval $[a,b] \ni x_i$ to fit a degree n polynomial. What happens if we fix n but let $m \to \infty$? For example, if we have the x-coordinates of the m data points equally spaced from a to b, then the least-squares problem

$$\min_{p_n \in \mathbb{P}_n} \sum_{i=1}^m |f(x_i) - p_n(x_i)|^2$$

is equivalent to minimizing the mean square error (the so-called RMSE is just square-root of the mean square error)

$$\min_{p_n \in \mathbb{P}_n} \frac{1}{m} \sum_{i=1}^m |f(x_i) - p_n(x_i)|^2,$$

of which the objective function is exactly the Riemann sum of the integral $\int_a^b |f(x) - p_n(x)|^2 \, \mathrm{d}x$ and

$$\frac{1}{m} \sum_{i=1}^m |f(x_i) - p_n(x_i)|^2 \overset{m\to\infty}{\to} \int_a^b |f(x) - p_n(x)|^2 \, \mathrm{d}x.$$

So, in the limit of $m \to \infty$, the least-squares problem becomes continuous:

$$\min_{p_n \in \mathbb{P}_n} \int_a^b |f(x) - p_n(x)|^2 \, \mathrm{d}x.$$

Such limits can also be taken in the probability sense. If $\{x_i\}_{i=1}^m$ is sampled independently from the uniform distribution on $[a,b]$, then by the law of large numbers, for any fixed $\varepsilon > 0$, it holds that

$$\Pr\left(\left| \frac{1}{m} \sum_{i=1}^m |f(x_i) - p_n(x_i)|^2 - \int_a^b |f(x) - p_n(x)|^2 \, \mathrm{d}x \right| > \varepsilon \right) \overset{m\to\infty}{\to} 0$$

where $\frac{1}{m}\sum_{i=1}^m |f(x_i) - p_n(x_i)|^2$ is the sample mean and $\int_a^b |f(x) - p_n(x)|^2 \, \mathrm{d}x$ is the expectation. If $\{x_i\}_{i=1}^m$ is sampled independently from the probability density function

$$\rho(x) = \begin{cases} w(x), & x \in [a,b] \\ 0, & \text{otherwise} \end{cases},$$

then

$$\Pr\left(\left| \frac{1}{m} \sum_{i=1}^m |f(x_i) - p_n(x_i)|^2 - \int_a^b |f(x) - p_n(x)|^2 w(x) \, \mathrm{d}x \right| > \varepsilon \right) \overset{m\to\infty}{\to} 0.$$

That is, in the limit of $m \to \infty$, the least-squares problem becomes continous with weight $w(x)$:

$$\min_{p_n \in \mathbb{P}_n} \int_a^b |f(x) - p_n(x)|^2 w(x) \, \mathrm{d}x.$$

To fix the idea, from now on, let $[a, b] = [-1, 1]$. When the weight is

$$w(x) = \frac{1}{\pi} \frac{1}{\sqrt{1 - x^2}},$$

we get the inner product

$$\langle f, g \rangle := \int_{-1}^1 f(x) g(x) w(x) \, \mathrm{d}x$$

with respect to which the orthogonal polynomials are the Chebyshev polynomials

$$T_k(x) = \cos(k \arccos x), \quad k = 0, 1, \ldots$$

which can be used as the basis for solving the weighted least-squares problem

$$\min_{c_0, \ldots, c_n} \int_{-1}^1 |f(x) - \sum_{k=0}^n c_k T_k(x)|^2 w(x) \, \mathrm{d}x,$$

with the solution given by

$$c_k = \frac{\int_{-1}^1 f(x) T_k(x) w(x) \, \mathrm{d}x}{\int_{-1}^1 T_k^2(x) w(x) \, \mathrm{d}x}.$$

Let $\theta = \arccos x$. We have

$$\int_{-1}^1 f(x) T_k(x) w(x) \, \mathrm{d}x = \frac{1}{\pi} \int_0^\pi f(\cos\theta) \cos(k\theta) \, \mathrm{d}\theta,$$

$$\int_{-1}^1 T_k^2(x) w(x) \, \mathrm{d}x = \frac{1}{\pi} \int_0^\pi \cos^2(k\theta) \, \mathrm{d}\theta = \begin{cases} 1, & k = 0 \\ \frac{1}{2}, & k = 1, 2, \ldots \end{cases}$$

In practice, the integral involving f may not have a closed form result, and a numerical integration (to be learned in Lecture 9-10) is needed. Here, we shall simply use the Riemann sum

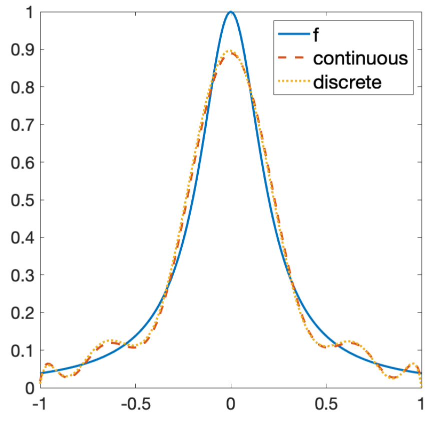
$$\frac{1}{\pi} \int_0^\pi f(\cos\theta) \cos(k\theta) \, \mathrm{d}\theta \approx \frac{1}{m} \sum_{i=1}^m f(\cos\frac{i\pi}{m}) \cos\frac{ik\pi}{m}.$$

We implement the weighted orthogonal projection (continuous weighted least-squares) as follows. After $\mathbf{c}$ is obtained, the same function `chebval` in the previous section can be used to evaluate $p_n(x)$.

```
function c = chebprj(f,n,m)
t = linspace(0,1,m+1); t = t(2:end)*pi;
k = (0:n)'; c = 1/m*cos(k*t)*f(cos(t'));
c(2:end) = c(2:end)*2; % function file ends

f=@(x) 1./(1+25*x.^2); n=10; c=chebprj(f,n,1000);
x = -1:0.01:1; y = chebval(c,x);
plot(x,f(x),x,y,'--','linewidth',2);
set(gca,'fontsize',16);

t = rand(101,1)*pi;
a = chebfit(cos(t),f(cos(t)),n);
yy=chebval(a,x);hold on;plot(x,yy,':','linewidth',2);
legend('f','continuous','discrete','fontsize',18)
```



For comparison, we also did the discrete least-squares with $\{x_i\}_{i=1}^m$ sampled from the p. d. f. ρ defined above, which is equivalent to sample $\theta_i = \arccos x_i$ from the uniform distribution on $[0, \pi]$.

3 Minimax: best approximation in maximum norm (optional)

Insufficiency of least-squares approximation. Mark Embree in his lecture notes says: *"In many applications, the L^2-norm has a physical interpretation – often associated with some measure of energy – and this makes continuous least-squares approximation particularly appealing (e.g., the best approximation minimizes the energy of the error). ... However, like discrete least-squares, this approach suffers from a potential problem: it minimizes the influence of outlying data, i.e., points where the function f varies wildly over a small portion of the interval [a, b]. Such an example is shown below."*

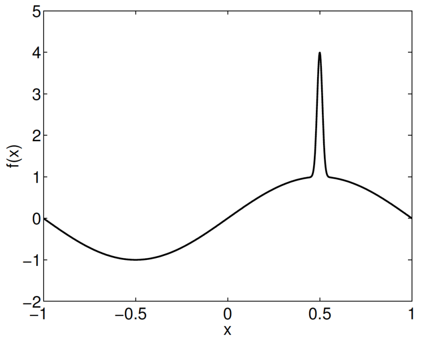


Figure from Mark Embree's lecture notes.

The function in the figure is something like

$$f(x) = \sin(\pi x) + 3\exp(-50(x - \frac{1}{2})^2).$$

The L^2-norm of a function g on $[-1, 1]$ is just

$$\sqrt{\int_{-1}^{1} g^2(x)\ \mathrm{d}x}$$

but the precise meaning of L^2 requires Lebesgue's definition of measures and integrals. Here, our function $f \in C^\infty[-1, 1]$ is smooth so the integral can be understood as Riemann's integral, and we seek the least-squares polynomial as

$$\min_{p_n \in \mathbb{P}_n} \int_a^b |f(x) - p_n(x)|^2\ \mathrm{d}x.$$

In this case, we need to use the Legendre polynomials

$$L_0(x) = 1, \quad L_1(x) = x, \quad L_{n+1}(x) = \frac{2n+1}{n+1} x L_n(x) - \frac{n}{n+1} L_{n-1}(x) \text{ for } n = 2, 3, \dots$$

which are orthogonal with respect to the inner product

$$\langle u, v \rangle = \int_{-1}^{1} u(x)v(x)\ \mathrm{d}x.$$

Indeed,

$$\int_{-1}^{1} L_m L_n(x)\ \mathrm{d}x = \begin{cases} \frac{2}{2n+1}, & m = n \\ 0, & m \neq n \end{cases}.$$

The solution to the least-squares problem is then

$$p_n(x) = \sum_{k=0}^{n} c_k L_k(x), \quad c_k = \frac{2k+1}{2} \int_{-1}^{1} f(x) L_k(x)\ \mathrm{d}x.$$

We approximate the integral involving f by the Riemann sum

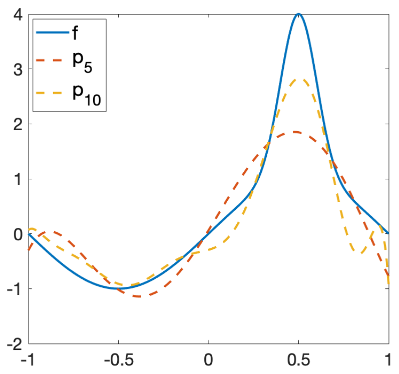
$$\int_{-1}^{1} f(x)L_k(x)\ \mathrm{d}x \approx \frac{1}{m}\sum_{i=1}^{m} f(-1+\frac{2i}{m})L_k(-1+\frac{2i}{m}).$$

This gives the following implementation:

```
function c = legendreprj(f,n,m)
x = linspace(-1,1,m+1)'; x = x(2:end);
L = zeros(m,n+1); L(:,1) = 1; L(:,2) = x;
for k=2:n
  L(:,k+1)=(2*k+1)/(k+1)*x.*L(:,k)-...
           k/(k+1)*L(:,k-1);
end
k=(0:n)'; c = (2*k+1)/2/m.*(L'*f(x));
```

```
function y = legendreval(c,x)
x = x(:); n = numel(c)-1;
L=zeros(numel(x),n+1);L(:,1)=1;L(:,2)=x;
for k=2:n
  L(:,k+1)=(2*k+1)/(k+1)*x.*L(:,k)-...
           k/(k+1)*L(:,k-1);
end
y = L*c(:);
```

```
f = @(x) sin(pi*x) + 3*exp(-50*(x-0.5).^2);
xc = logspace(log10(0.46),log10(0.55),500);
x = [-1:0.01:0.45 xc 0.56:0.01:1];
c = legendreprj(f,5,1e7);
y = legendreval(c,x);
plot(x,f(x),x,y,'--','linewidth',2);
c = legendreprj(f,10,1e7);
y = legendreval(c,x);
hold on; plot(x,y,'--','linewidth',2);
set(gca,'fontsize',16);
legend('f','p_5','p_{10}','fontsize',18);
```



We see that the continuous least-squares approximation has difficultiy in capturing the Gaussian peak around $x = 0.5$.

Minimax approximation. Given the drawback of least-squares in controlling the maximum error

$$\|f - p_n\|_{\infty,\,[-1,1]} := \max_{x\in[-1,1]} |f(x) - p_n(x)|^1,$$

it is desirable to consider the minimax problem

$$\min_{p_n\in\mathbb{P}_n} \max_{x\in[-1,1]} |f(x) - p_n(x)|.$$

Theorem 1 (de la Vallée Poussin Theorem). *Let $f \in C[a,b]$ and suppose $r \in \mathbb{P}_n$ is some polynomial for which there exist $n+2$ points $\{x_j\}_{j=0}^{n+1}$ with $a \le x_0 < x_1 < \cdots < x_{n+1} \le b$ at which the error $f(x) - r(x)$ oscillates signs, i.e.,*

$$\operatorname{sign}(f(x_j) - r(x_j)) = -\operatorname{sign}(f(x_{j+1}) - r(x_{j+1}))$$

for $j = 0,\ldots,n$. Then

$$\min_{p\in\mathbb{P}_n} \|f - p\|_\infty \ge \min_{0\le j\le n+1} |f(x_j) - r(x_j)|\,.$$

Theorem 2 (Equi-Oscillation Theorem). *Suppose $f \in C[a,b]$. Then the minimax problem has a unique solution, and $p_* \in \mathcal{P}_n$ is a minimax approximation to f from $\mathcal{P}_n$ on $[a,b]$ if and only if there exist $n+2$ points $x_0 < x_1 < \cdots < x_{n+1}$ such that*

$$|f(x_j) - p_*(x_j)| = \|f - p_*\|_\infty\,, \quad j = 0,\ldots,n+1$$

and

$$f(x_j) - p_*(x_j) = -(f(x_{j+1}) - p_*(x_{j+1}))\,, \quad j = 0,\ldots,n.$$

[1]The maximum absolute value of a function is called the maximum norm of the function. It is also called the infinity norm of the function because it is the limit of the p-norm: $(\int_{-1}^{1} |f(x)|^p)^{1/p} \overset{p\to\infty}{\to} \|f\|_\infty$.

The Remez algorithm can be used to find the minimax solution. Let $\{\phi_k\}_{k=0}^n$ be a basis of $\mathbb{P}_n$. One first initializes $n+2$ points $x_0 < x_1 < \ldots < x_{n+1}$ intended to eventually become the equioscillation points of $f - p_n$. Then one repeats the following two steps until $f - p_n$ equioscillates at $\{x_i\}_{i=0}^{n+1}$:

Step 1. Find $\{c_k\}_{k=0}^m$ and $E \in \mathbb{R}$ such that

$$\sum_{k=0}^{m} c_k \phi_k(x_i) + (-1)^i E = f(x_i), \qquad \text{for } i = 0, 1, \ldots, n+1.$$

Step 2. Let $p_n(x) = \sum_{k=0}^m c_k \phi_k(x_i)$. Since $f(x_i) - p_n(x_i) = (-1)^i E$, $i = 0, 1, \ldots, n+1$ have alternating signs, $f(x) - p_n(x)$ has $n+1$ roots between x_i and x_{i+1}, $i = 0, \ldots, n$. Hence, $f(x) - p(x)$ has n local extrema between the adjacent roots, plus two more between the endpoints of the interval and the first and last roots. Find all the local extrema, and relocate $x_0 < x_1 < \ldots < x_{n+1}$ to those local extrema.

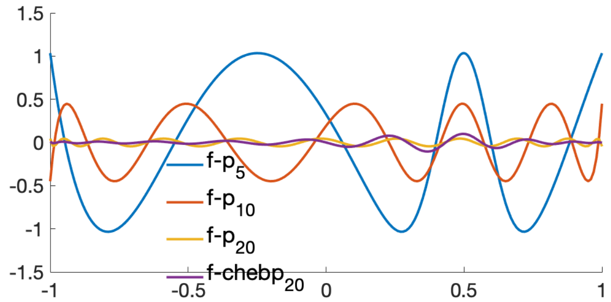
The program is as follows.

```
function [c,E] = chebremez(f,n)
% assume the interval is [-1,1]
k = (0:n+1)'; x = cos((2*k+1)*pi/(2*n+4));
x = flip(x);

while true
    % Step 1. solve the oscillation problem
    A = zeros(n+2,n+2);
    A(:,1) = 1; A(:,2) = x;
    for k=2:n
      A(:,k+1) = 2*x.*A(:,k)-A(:,k-1);
    end
    A(:,end) = (-1).^(0:n+1)';
    c = A\f(x); c = c(1:end-1);
    g = @(x)f(x)-chebval(c,x);

    % Step 2.
    % find roots of f-p by bisection
    r = zeros(n+1,1);
    for k=1:n+1
        a = x(k); b = x(k+1);
        r(k) = bisect(g,a,b,1e-10,100);
    end
    % find extrema of f-p by golden section
    tomin = @(x) -g(-1)*g(x);
    x(1) = golden_section(tomin,-1,r(1),1e-10,100);
    tomin = @(x) -g(1)*g(x);
    x(end) = golden_section(tomin,r(end),1,1e-10,100);
    for k=2:n+1
        mi = (r(k-1)+r(k))/2;
        tomin = @(x) -g(mi)*g(x);
        x(k)=golden_section(tomin,r(k-1),r(k),1e-10,100);
    end
    % stopping criterion
    E = abs(g(x));
    if max(abs(diff(E)))<1e-12
       break;
    end
end  % function file ends

f = @(x) sin(pi*x) + 3*exp(-50*(x-0.5).^2);
xc = logspace(log10(0.46),log10(0.55),500);
x = [-1:0.01:0.45 xc 0.56:0.01:1]'; hold on;
for n = [5 10 20]
   [c, E] = chebremez(f,n);
   y = chebval(c,x);
   plot(x,f(x)-y,'linewidth',2);
end
k = (0:n)'; xi = cos((2*k+1)*pi/(2*n+2));
y = lagrange(xi,f(xi),x);
plot(x,f(x)-y,'linewidth',2);
```



We see that the Lagrange interpolation at the Chebyshev nodes is nearly the minimax solution!