Numerical Analysis and Computing

Lecture Notes #12
— Approximation Theory —
Chebyshev Polynomials & Least Squares, redux

Peter Blomgren,
⟨blomgren.peter@gmail.com⟩

Department of Mathematics and Statistics
Dynamical Systems Group
Computational Sciences Research Center
San Diego State University
San Diego, CA 92182-7720

http://terminus.sdsu.edu/

Fall 2014

Outline

Orthogonal Polynomials: A Quick Summary

So far we have seen the use of orthogonal polynomials can help us solve the **normal equations** which arise in discrete and continuous least squares problems, **without** the need for expensive and numerically difficult matrix inversions.

The ideas and techniques we developed — *i.e.* **Gram-Schmidt orthogonalization** with respect to a weight function over any interval have applications far beyond least squares problems.

The Legendre Polynomials are orthogonal on the interval $[-1, 1]$ with respect to the weight function $w(x) = 1$. — One curious property of the Legendre polynomials is that their roots (all real) yield the optimal node placement for Gaussian quadrature.

The Legendre Polynomials — Background

The Legendre polynomials are solutions to the Legendre Differential Equation (which arises in numerous problems exhibiting spherical symmetry)

$$(1-x^2)\frac{d^2y}{dx^2} - 2x\frac{dy}{dx} + \ell(\ell+1)y = 0, \quad \ell \in \mathbb{N}$$

or equivalently

$$\frac{d}{dx}\left[(1-x^2)\frac{dy}{dx}\right] + \ell(\ell+1)y = 0, \quad \ell \in \mathbb{N}$$

Applications: Celestial Mechanics (Legendre's original application), Electrodynamics, etc...

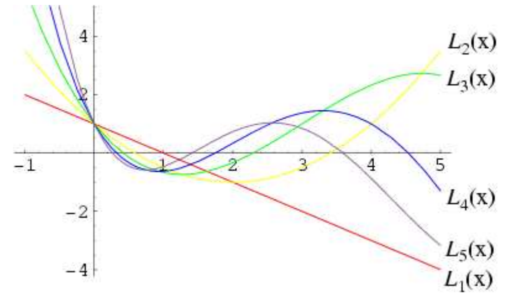
Quote

"Orthogonal polynomials have very useful properties in the solution of mathematical and physical problems. [... They] provide a natural way to solve, expand, and interpret solutions to many types of important differential equations. Orthogonal polynomials are especially **easy*** *to generate using Gram-Schmidt orthonormalization."*

"The roots of orthogonal polynomials possess many rather surprising and useful properties."

(http://mathworld.wolfram.com/OrthogonalPolynomials.html)

* Definition of "easy" may vary.



The **Laguerre polynomials** are solutions to the Laguerre differential equation

$$x\frac{d^2}{dx^2} + (1-x)\frac{dy}{dx} + \lambda y = 0$$

They are associated with the radial solution to the Schrödinger equation for the Hydrogen atom's electron (Spherical Harmonics).

Polynomials	**Interval**	**w(x)**
Chebyshev (1st)	$[-1,1]$	$1/\sqrt{1-x^2}$
Chebyshev (2nd)	$[-1,1]$	$\sqrt{1-x^2}$
Gegenbauer	$[-1,1]$	$(1-x^2)^{\alpha-1/2}$
Hermite*	$(-\infty,\infty)$	e^{-x^2}
Jacobi	$(-1,1)$	$(1-x)^\alpha(1+x)^\beta$
Legendre	$[-1,1]$	1
Laguerre	$[0,\infty)$	e^{-x}
Laguerre (assoc)	$[0,\infty)$	$x^k e^{-x}$

Today we'll take a closer look at Chebyshev polynomials of the first kind.

* These are the Hermite orthogonal polynomials, not to be confused with the Hermite interpolating polynomials...

The Chebyshev polynomials, of first $T_n(x)$ and second $U_n(x)$ are solutions to the Chebyshev differential equations:

$$(1-x^2)y'' - xy' + n^2 y = 0 \qquad \text{(1st)}$$

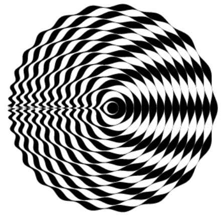
$$(1-x^2)y'' - 3xy' + n(n+2)y = 0 \qquad \text{(2nd)}$$

The Chebyshev polynomials are a special case of the Gegenbauer polynomials, which themselves are a special case of the Jacobi polynomials.

$\{\, T_0(x), \ldots, T_{11}(x)\,\}$ and $\{\, U_0(x), \ldots, U_9(x)\,\}$ and all kinds of other exciting information can be found at
http://en.wikipedia.org/wiki/Chebyshev_polynomials

Chebyshev Polynomials: The Sales Pitch

$$T_n(z) = \frac{1}{2\pi i} \oint \frac{(1-t^2)\, t^{-(n+1)}}{(1-2tz+t^2)}\, dt$$



Chebyshev Polynomials are used to **minimize approximation error.** We will use them to solve the following problems:

[1] Find an optimal placement of the interpolating points $\{x_0, x_1, \ldots, x_n\}$ to minimize the error in Lagrange interpolation.

[2] Find a means of reducing the degree of an approximating polynomial with minimal loss of accuracy.

Chebyshev Polynomials: Definitions.

The Chebyshev polynomials $\{T_n(x)\}$ *are orthogonal on the interval* $(-1, 1)$ *with respect to the weight function* $w(x) = 1/\sqrt{1-x^2}$, *i.e.*

$$\langle T_i(x),\ T_j(x) \rangle_{w(x)} \equiv \int_{-1}^{1} T_i(x) T_j(x)^* \, w(x) dx = \alpha_i \delta_{i,j}.$$

We could use the *Gram-Schmidt* orthogonalization process to find them, but it is easier to give the definition and then check the properties...

Definition (Chebyshev Polynomials)

For $x \in [-1, 1]$, define

$$T_n(x) = \cos(n \arccos x), \quad \forall n \geq 0.$$

Note: $T_0(x) = \cos(0) = 1, \quad T_1(x) = x.$

Chebyshev Polynomials, $T_n(x)$, $n \geq 2$.

We introduce the notation $\theta = \arccos x$, and get

$$T_n(\theta(x)) \equiv T_n(\theta) = \cos(n\theta), \quad \text{where } \theta \in [0, \pi].$$

We can find a recurrence relation, using these observations:

$$\begin{aligned}
T_{n+1}(\theta) &= \cos((n+1)\theta) = \cos(n\theta)\cos(\theta) - \sin(n\theta)\sin(\theta) \\
T_{n-1}(\theta) &= \cos((n-1)\theta) = \cos(n\theta)\cos(\theta) + \sin(n\theta)\sin(\theta) \\
\mathbf{T_{n+1}}(\theta) + \mathbf{T_{n-1}}(\theta) &= \mathbf{2}\cos(\mathbf{n}\theta)\cos(\theta).
\end{aligned}$$

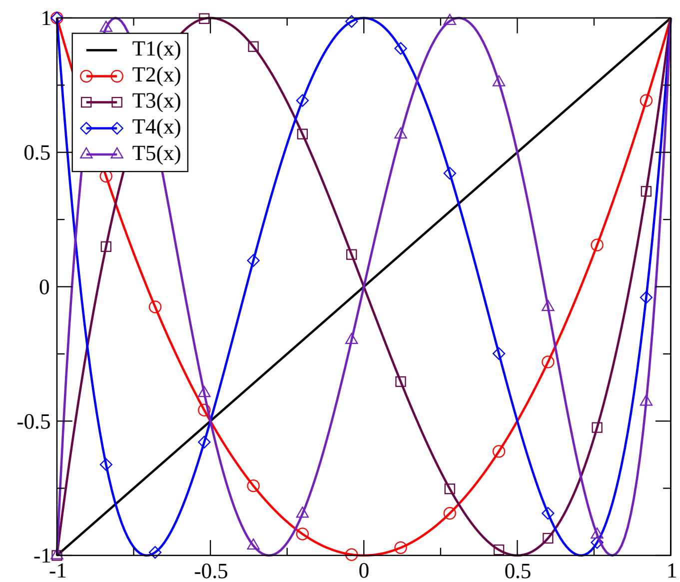
Returning to the original variable x, we have

$$T_{n+1}(x) = 2x\cos(n \arccos x) - T_{n-1}(x),$$

or

$$\mathbf{T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x).}$$

The Chebyshev Polynomials



Orthogonality of the Chebyshev Polynomials, I

$$\int_{-1}^{1} \frac{T_n(x)\,T_m(x)}{\sqrt{1-x^2}}\,dx = \int_{-1}^{1} \cos(n\arccos x)\cos(m\arccos x)\,\frac{dx}{\sqrt{1-x^2}}.$$

Reintroducing $\theta = \arccos x$ gives,

$$d\theta = -\frac{dx}{\sqrt{1-x^2}},$$

and the integral becomes

$$-\int_{\pi}^{0} \cos(n\theta)\cos(m\theta)\,d\theta = \int_{0}^{\pi} \cos(n\theta)\cos(m\theta)\,d\theta.$$

Now, we use the fact that

$$\cos(n\theta)\cos(m\theta) = \frac{\cos(n+m)\theta + \cos(n-m)\theta}{2}\ \ldots$$

Orthogonality of the Chebyshev Polynomials, II

We have:
$$\int_{0}^{\pi} \frac{\cos(n+m)\theta + \cos(n-m)\theta}{2}\,d\theta.$$

If $m \neq n$, we get

$$\left[\frac{1}{2(n+m)}\sin((n+m)\theta) + \frac{1}{2(n-m)}\sin((n-m)\theta)\right]_0^{\pi} = 0,$$

if $m = n$, we have

$$\left[\frac{1}{2(n+m)}\sin((n+m)\theta) + \frac{x}{2}\right]_0^{\pi} = \frac{\pi}{2}.$$

Hence, the Chebyshev polynomials are **orthogonal.**

Zeros and Extrema of Chebyshev Polynomials.

Theorem

The Chebyshev polynomial of degree $n \geq 1$ has n simple zeros in $[-1,1]$ at

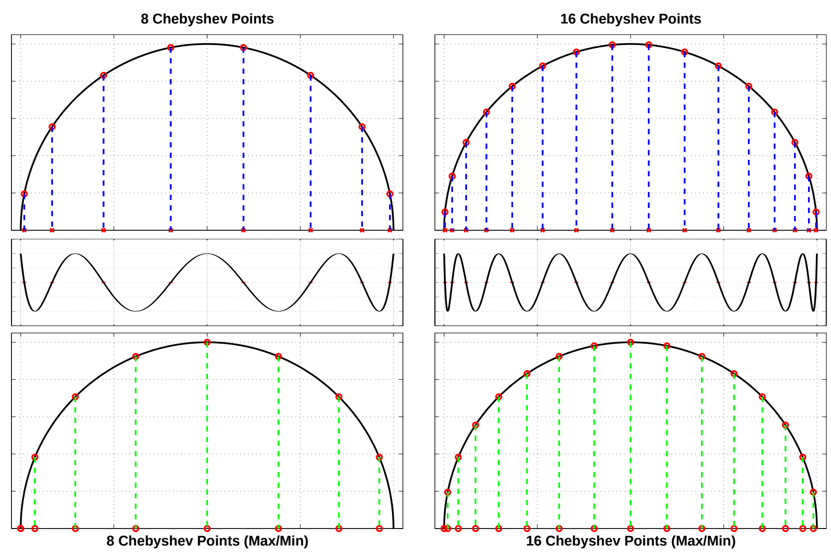
$$x_k = \cos\left(\frac{2k-1}{2n}\pi\right), \quad k = 1,\ldots,n.$$

Moreover, $T_n(x)$ assumes its absolute extrema at

$$x'_k = \cos\left(\frac{k\pi}{n}\right), \quad \textit{with} \quad T_n(x'_k) = (-1)^k, \quad k = 1,\ldots,n-1.$$

Payoff: No matter what the degree of the polynomial, the oscillations are kept under control!!!

Zeros and Extrema of Chebyshev Polynomials   Zeros $\{x_k\}$, Max/Min $\{x'_k\}$



Zeros and Extrema of Chebyshev Polynomials — Proof.

Proof.

Let:

$$x_k = \cos\left(\frac{2k-1}{2n}\pi\right), \quad x'_k = \cos\left(\frac{k\pi}{n}\right).$$

Then:

$$\begin{aligned} T_n(x_k) &= \cos(n \arccos(x_k)) = \cos\left(n \arccos\left(\cos\left(\tfrac{2k-1}{2n}\pi\right)\right)\right) \\ &= \cos\left(\tfrac{2k-1}{2}\pi\right) = 0, \quad \surd \end{aligned}$$

$$T'_n(x) = \frac{d}{dx}[\cos(n\arccos(x))] = \frac{n\sin(n\arccos(x))}{\sqrt{1-x^2}},$$

$$T'_n(x'_k) = \frac{n\sin\left(n\arccos\left(\cos\left(\frac{k\pi}{n}\right)\right)\right)}{\sqrt{1-\cos^2\left(\frac{k\pi}{n}\right)}} = \frac{n\sin(k\pi)}{\sin\left(\frac{k\pi}{n}\right)} = 0, \quad \surd$$

$$T_n(x'_k) = \cos\left(n\arccos\left(\cos\left(\tfrac{k\pi}{n}\right)\right)\right) = \cos(k\pi) = (-1)^k. \quad \surd$$

□

Monic Chebyshev Polynomials, I

Definition (Monic Polynomial)

A monic polynomial is a polynomial with leading coefficient 1.

We get the monic Chebyshev polynomials $\tilde{T}_n(x)$ by dividing $T_n(x)$ by 2^{n-1}, $n \geq 1$. Hence,

$$\tilde{T}_0(x) = 1, \quad \tilde{T}_n(x) = \frac{1}{2^{n-1}} T_n(x), \quad n \geq 1.$$

They satisfy the following recurrence relations

$$\begin{aligned} \tilde{T}_2(x) &= x\tilde{T}_1(x) - \tfrac{1}{2}\tilde{T}_0(x) \\ \tilde{T}_{n+1}(x) &= x\tilde{T}_n(x) - \tfrac{1}{4}\tilde{T}_{n-1}(x). \end{aligned}$$

Monic Chebyshev Polynomials, II

The location of the zeros and extrema of $\tilde{T}_n(x)$ coincides with those of $T_n(x)$, however the extreme values are

$$\tilde{T}_n(x'_k) = \frac{(-1)^k}{2^{n-1}}, \quad k = 1, \ldots, n-1.$$

Definition

Let $\tilde{\mathcal{P}}_n$ denote the set of all monic polynomials of degree n.

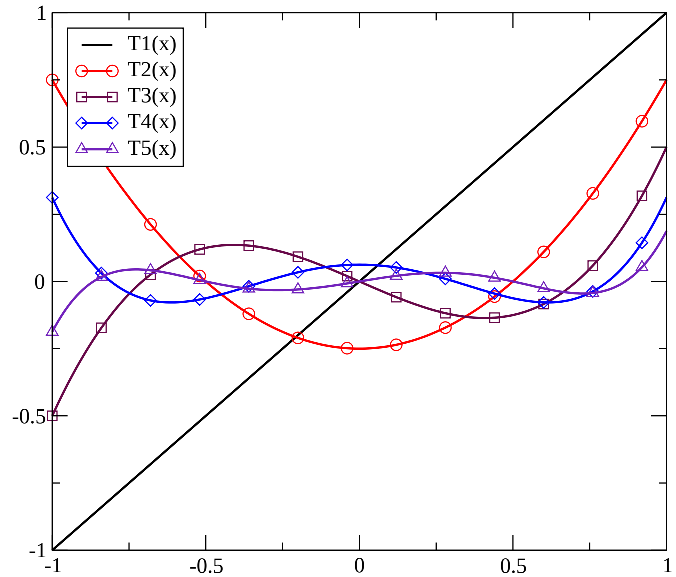
Theorem (Min-Max)

The monic Chebyshev polynomials $\tilde{T}_n(x)$, have the property that

$$\frac{1}{2^{n-1}} = \max_{x\in[-1,1]} |\tilde{T}_n(x)| \leq \max_{x\in[-1,1]} |P_n(x)|, \quad \forall P_n(x) \in \tilde{\mathcal{P}}_n.$$

Moreover, equality can only occur if $P_n(x) \equiv \tilde{T}_n(x)$.

The Monic Chebyshev Polynomials



Optimal Node Placement in Lagrange Interpolation, I

If $x_0, x_1, \ldots, x_n$ are distinct points in the interval $[-1, 1]$ and $f \in C^{n+1}[-1, 1]$, and $P(x)$ the n^{th} degree interpolating Lagrange polynomial, then $\forall x \in [-1, 1]\ \exists \xi(x) \in (-1, 1)$ so that

$$f(x) - P(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!} \prod_{k=0}^{n} (x - x_k).$$

We have no control over $f^{(n+1)}(\xi(x))$, but we can place the nodes in a clever way as to minimize the maximum of $\prod_{k=0}^{n}(x - x_k)$. Since $\prod_{k=0}^{n}(x - x_k)$ is a monic polynomial of degree $(n+1)$, we know the min-max is obtained when the nodes are chosen so that

$$\prod_{k=0}^{n} (x - x_k) = \tilde{T}_{n+1}(x), \quad i.e. \quad x_k = \cos\left(\frac{2k+1}{2(n+1)}\pi\right).$$

Optimal Node Placement in Lagrange Interpolation, II

Theorem

If $P(x)$ is the interpolating polynomial of degree at most n with nodes at the roots of $T_{n+1}(x)$, then

$$\max_{x \in [-1,1]} |f(x) - P(x)| \le \frac{1}{2^n (n+1)!} \max_{x \in [-1,1]} |f^{(n+1)}(x)|,$$

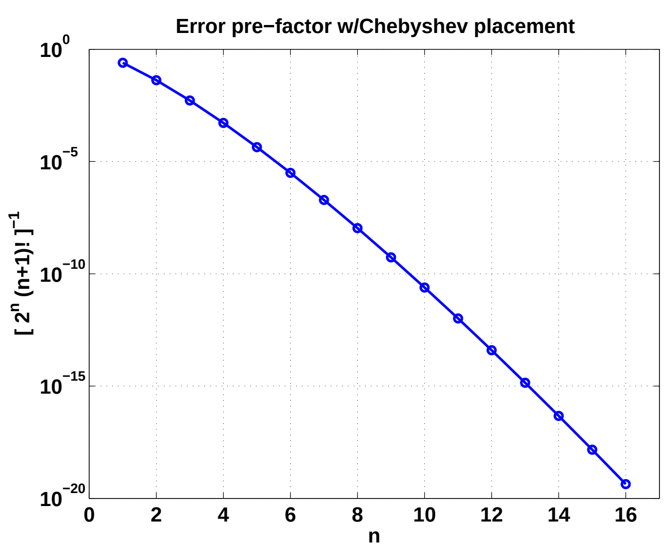
$$\forall f \in C^{n+1}[-1, 1].$$

Extending to any interval: The transformation

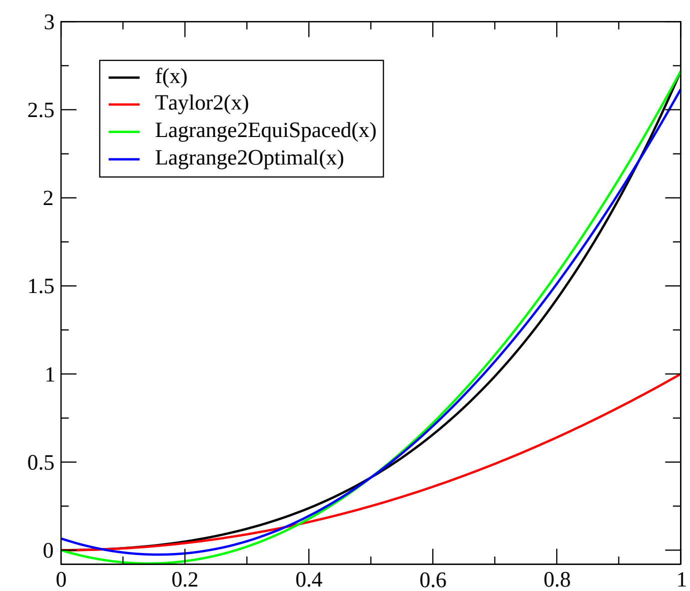
$$\tilde{x} = \frac{1}{2}\left[(b-a)x + (a+b)\right]$$

transforms the nodes x_k in $[-1, 1]$ into the corresponding nodes $\tilde{x}_k$ in $[a, b]$.

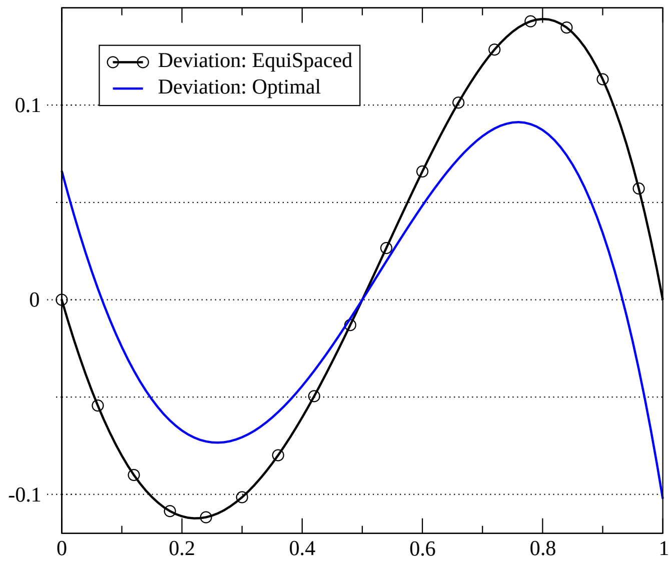
Error "Pre-factor" $\frac{1}{2^n(n+1)!}$ Visualized



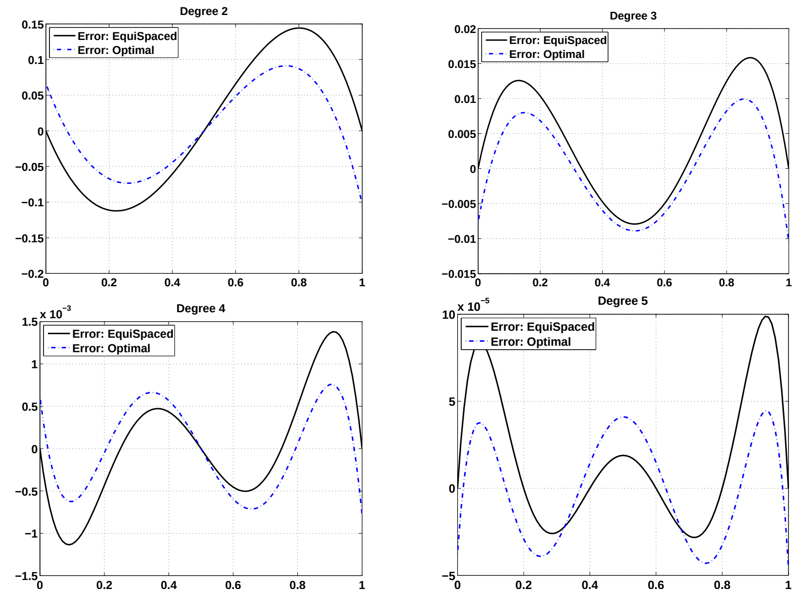
Example: Interpolating $f(x) = x^2\, e^x$.



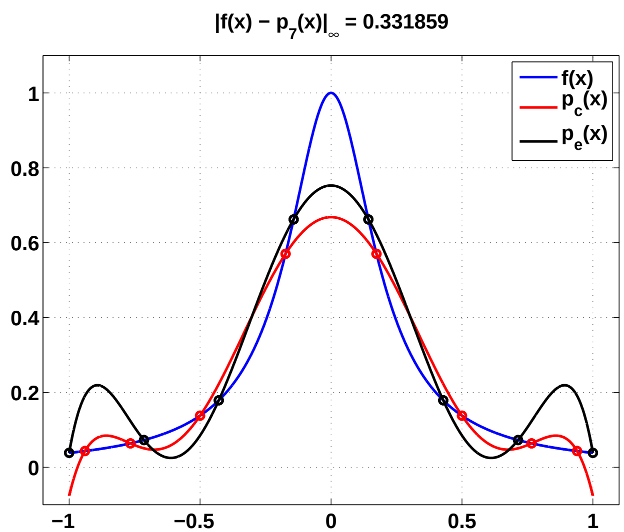
Example: Interpolating $f(x) = x^2 e^x$ — The Error.



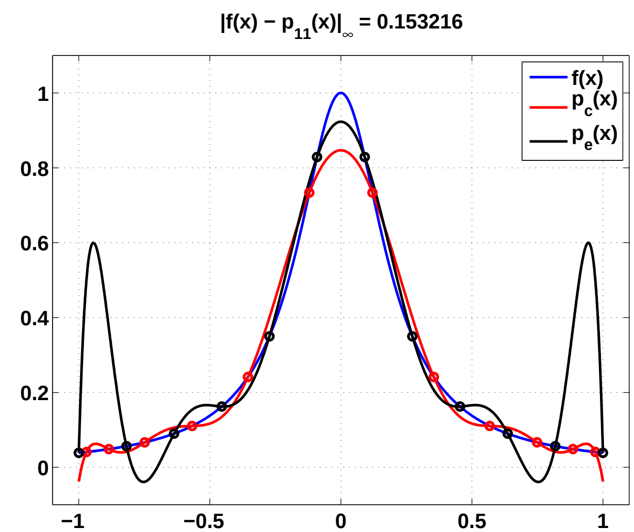
Example: Interpolating $f(x) = x^2 e^x$ — The Error.



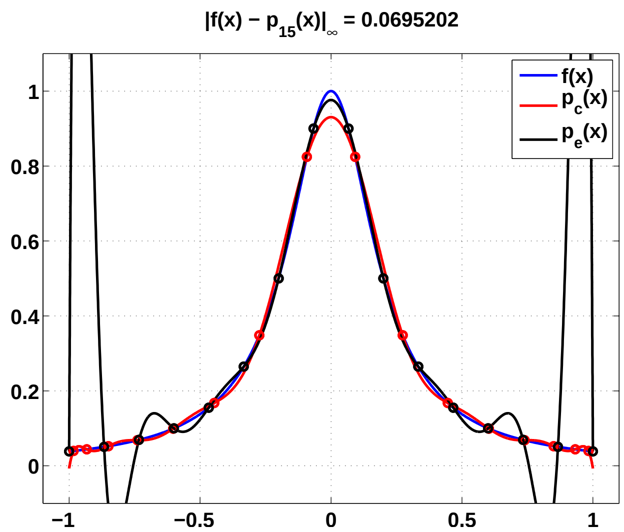
$P_7(x)$ Polynomial Interpolation of Runge's Function $f(x) = \frac{1}{1+25\,x^2}$



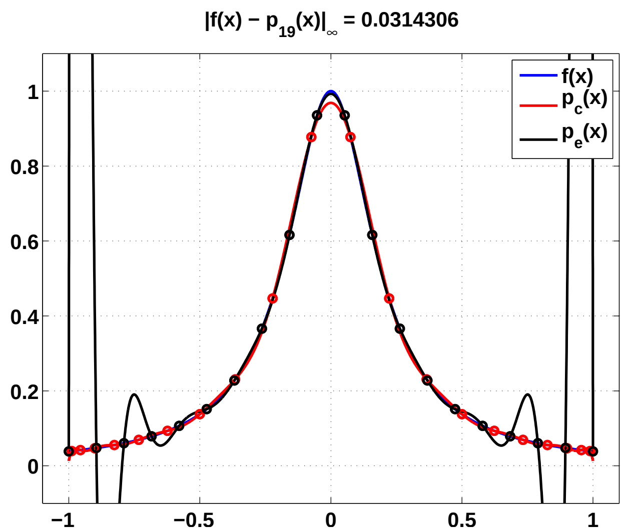
$P_{11}(x)$ Polynomial Interpolation of Runge's Function $f(x) = \frac{1}{1+25\,x^2}$



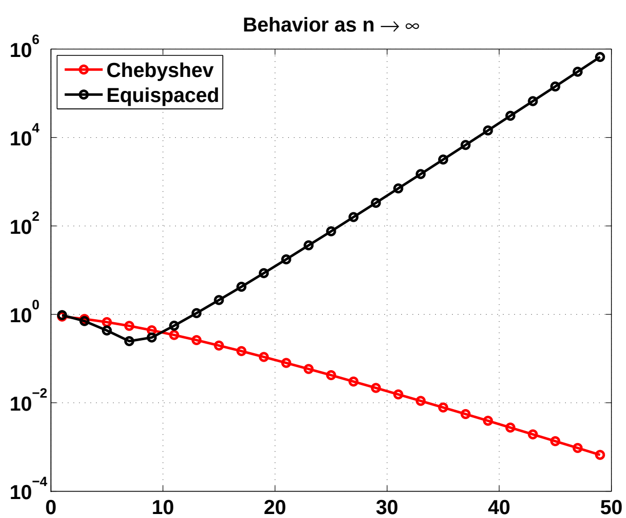
$P_{15}(x)$ Polynomial Interpolation of Runge's Function $f(x) = \frac{1}{1+25\,x^2}$



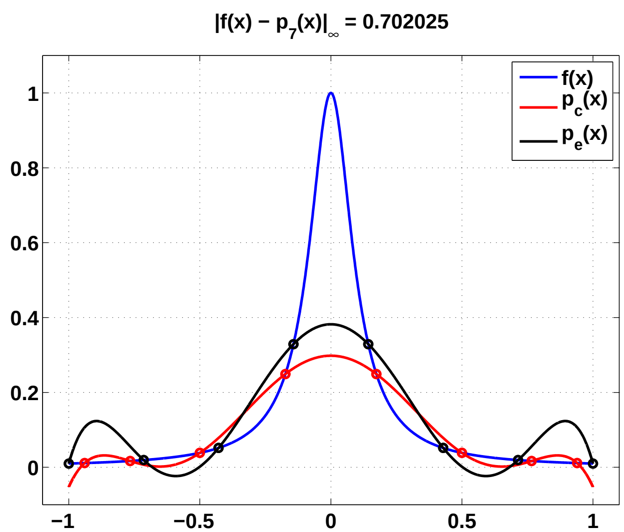
$P_{19}(x)$ Polynomial Interpolation of Runge's Function $f(x) = \frac{1}{1+25\,x^2}$



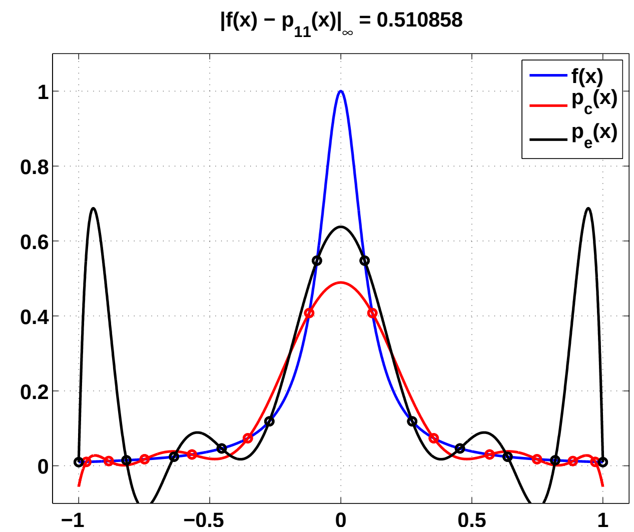
$P_{\infty}(x)$ Polynomial Interpolation of Runge's Function $f(x) = \frac{1}{1+25\,x^2}$



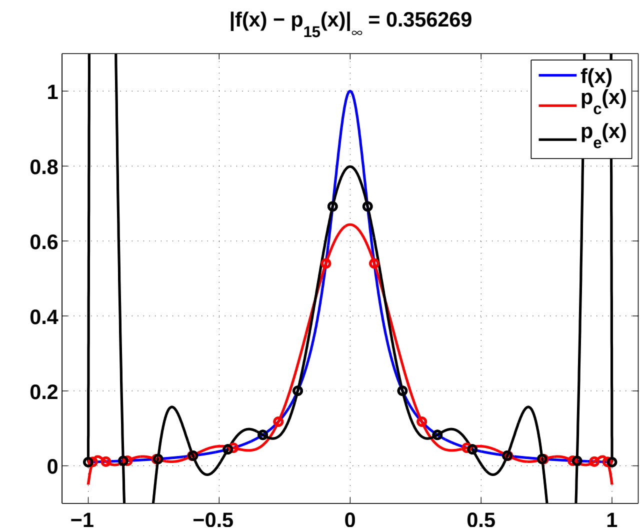
$P_{7}(x)$ Polynomial Interpolation of Runge's Function $f(x) = \frac{1}{1+100\,x^2}$



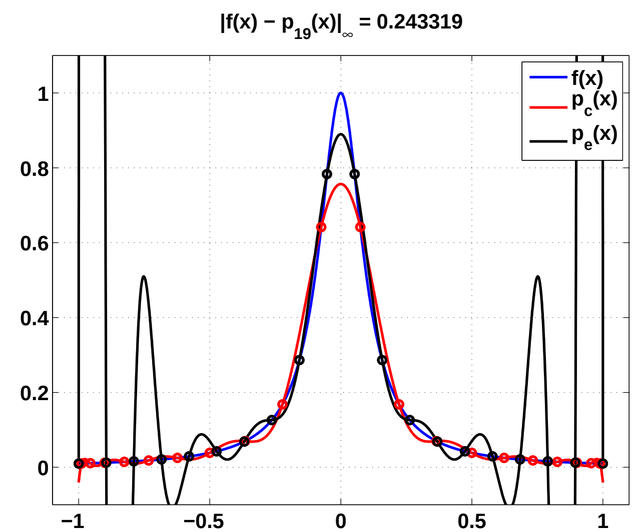
$P_{11}(x)$ Polynomial Interpolation of Runge's Function $f(x) = \frac{1}{1+100\,x^2}$



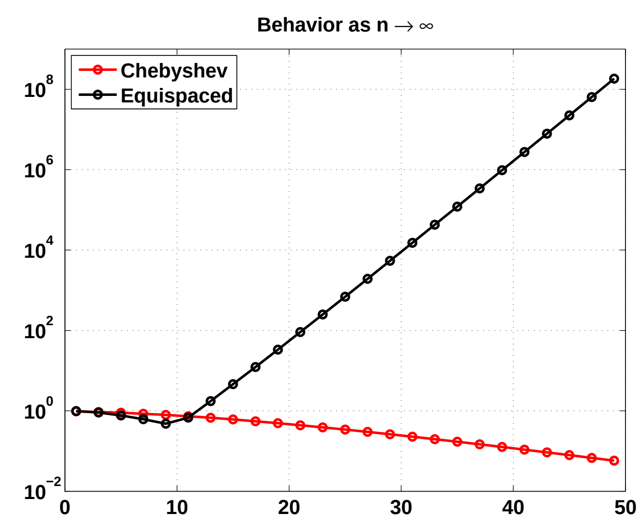
$P_{15}(x)$ Polynomial Interpolation of Runge's Function $f(x) = \frac{1}{1+100\,x^2}$



$P_{19}(x)$ Polynomial Interpolation of Runge's Function $f(x) = \frac{1}{1+100\,x^2}$



$P_{\infty}(x)$ Polynomial Interpolation of Runge's Function $f(x) = \frac{1}{1+100\,x^2}$



Level Curves of $|\ell_n(z)|$

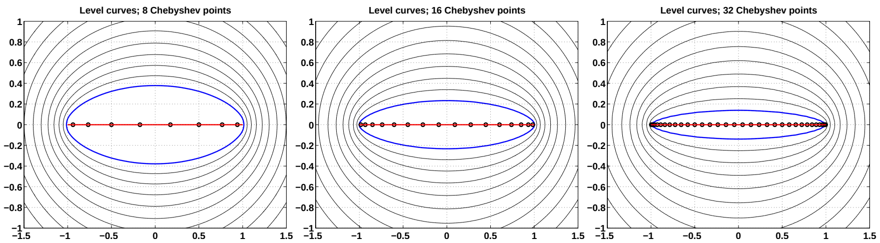


Figure: Chebyshev Placement.

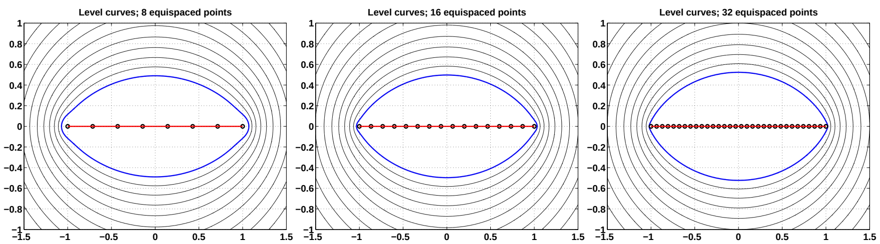


Figure: Equi-spaced Placement.

Level Curves of $|\ell_n(z)|$

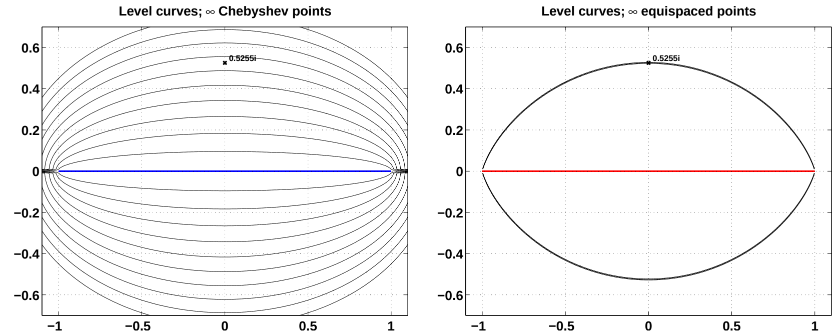


Figure: For Chebyshev placement, the level-set "eye" closes completely when $n \to \infty$, thus guaranteeing convergence for all Runge-type functions no matter how close to the real line the complex poles are; recall that in the Equi-spaced case, the "eye" opens to roughly $\pm 0.5255i$ we only get interpolation covergence for $1/(1+ax^2)$ when $a \in [0,\ 3.2795)$.

Least Squares: Revisited

Before we move on to new and exciting orthogonal polynomials with exotic names... Let's take a moment (or two) and look at the usage of Least Squares Approximation.

This section is a "how-to" with quite a few applied example of least squares approximation...

Example #1 — Warm-up

First we consider the problem of fitting 1st, 2nd, and 3rd degree polynomials to the following data:

```
x = [1.0 1.1 1.3 1.5 1.9 2.1]'

y = [1.84 1.90 2.31 2.65 2.74 3.18]'
```

matlab¿¿
```
[First we define the matrices]
A1 = [ones(size(x)) x];
A2 = [A1 x.*x];
A3 = [A2 x.*x.*x ];
[Then we solve the Normal Equations]
pcoef1 = A1\y;
pcoef2 = A2\y;
pcoef3 = A3\y;
```

Note: The matrices `A1`, `A2`, and `A3` are "tall and skinny." Normally we would compute $(An' \cdot An)^{-1}(An' \cdot y)$, however when matlab encounters `An\y` it automatically gives us a solution in the least squares sense.

We now have the coefficients for the polynomials, let's plot:

```
matlab>> xv = 1.0:0.01:2.1;
         p1 = polyval(flipud(pcoef1),xv);
         p2 = polyval(flipud(pcoef2),xv);
         p3 = polyval(flipud(pcoef3),xv);
         plot(xv,p3,'k-','linewidth',3); hold on;
         plot(x,y,'ko','linewidth',3); hold off
```

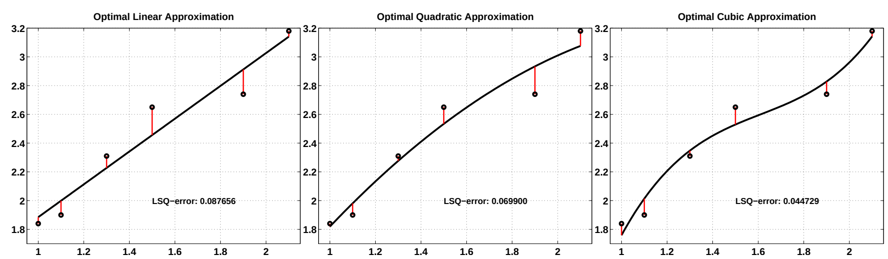


Figure: The least squares polynomials $p_1(x)$, $p_2(x)$, and $p_3(x)$.

Finally, we compute the error

```
matlab>> p1err = polyval(flipud(pcoef1),x) - y;
         p2err = polyval(flipud(pcoef2),x) - y;
         p3err = polyval(flipud(pcoef3),x) - y;
         disp([sum(p1err.*p1err) sum(p2err.*p2err)
         sum(p3err.*p3err)])
```

Which gives us the fitting errors

$$P_1^{\text{Err}} = 0.0877, \quad P_2^{\text{Err}} = 0.0699, \quad P_3^{\text{Err}} = 0.0447$$

Consider the same data:

```
x = [1.0 1.1 1.3 1.5 1.9 2.1]'
y = [1.84 1.90 2.31 2.65 2.74 3.18]'
```

But let's find the best fit of the form $a + b\sqrt{x}$ to this data! Notice that this expression is linear in its parameters a, b, so we can solve the corresponding least squares problem!

```
matlab>> A = [ones(size(x)) sqrt(x)];
         pcoef = A\y;
         xv = 1.0:0.01:2.1;
         fv = pcoef(1) + pcoef(2)*sqrt(xv);
         plot(xv,fv,'k-','linewidth',3); hold on;
         plot(x,y,'ko','linewidth',3); hold off;
```

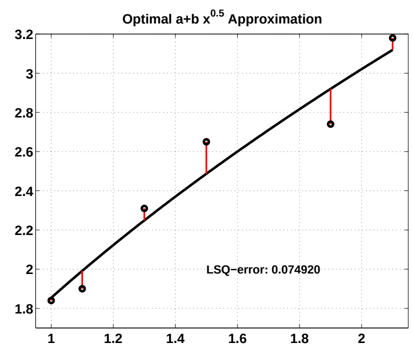



Figure: The best fit of the form $a + b\sqrt{x}$.

We compute the fitting error:

```
matlab>> ferr = pcoef(1) + pcoef(2)*sqrt(x) - y;
         disp(sum(ferr.*ferr))
```

Which gives us $P_{\{a+b\sqrt{x}\}}^{\text{Err}} = 0.0749$

As long as the model is **linear in its parameters**, we can solve the least squares problem.

Non-linear dependence will have to wait until `Math 693a`.

We can fit this model:

$$M_1(a,b,c,d) = a + bx^{3/2} + c/\sqrt{x} + de^{\sin(x)}$$

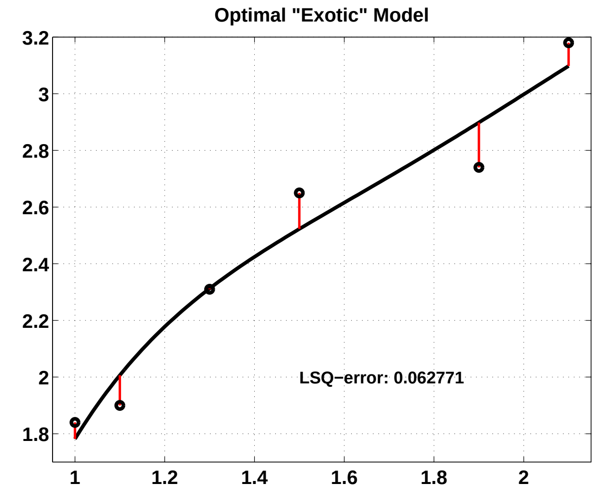
Just define the matrix

```
matlab>> A = [ones(size(x)) x.^(3/2) 1./sqrt(x) exp(sin(x))];
```

and compute the coefficients

```
matlab>> coef = A\y;
```

etc...



The optimal approximation of this form is

$$16.4133 - 0.9970x^{3/2} - \frac{11.0059}{\sqrt{x}} - 1.1332e^{\sin(x)}$$

It seems quite unlikely the model

$$M_1(a,b,c,d) = a + bx^{3/2} + c/\sqrt{x} + de^{\sin(x)}$$

will ever be useful.

However, we have forgotten about one important aspect of the problem — so far our models have depended on only one variable, x.

How do we go about fitting multi-dimensional data?


```
matlab>> x=1:0.25:5;
         y=1:0.25:5;
         [X,Y]=meshgrid(x,y);
         Fxy=1+sqrt(X)+Y.^3+0.05*randn(size(X));
         surf(x,y,Fxy)
```

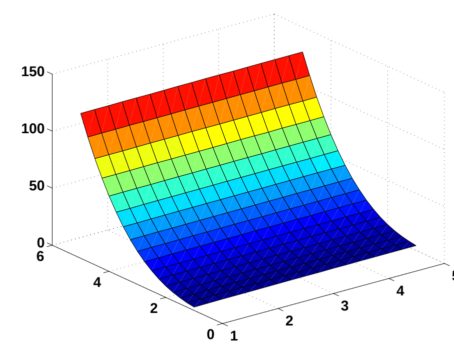


Figure: 2D-data set, the vertexes on the surface are our data points.

Lets try to fit a simple 3-parameter model to this data

$$M(a, b, c) = a + bx + cy$$

```
matlab>> sz   = size(X);
         Bm   = reshape(X,prod(sz),1);
         Cm   = reshape(Y,prod(sz),1);
         Am   = ones(size(Bm));
         RHS  = reshape(Fxy,prod(sz),1);
         A    = [Am Bm Cm];
         coef = A \ RHS;
         fit  = coef(1) + coef(2)*X + coef(3)*Y;
         fitError = Fxy - fit;
         surf(x,y,fitError)
```

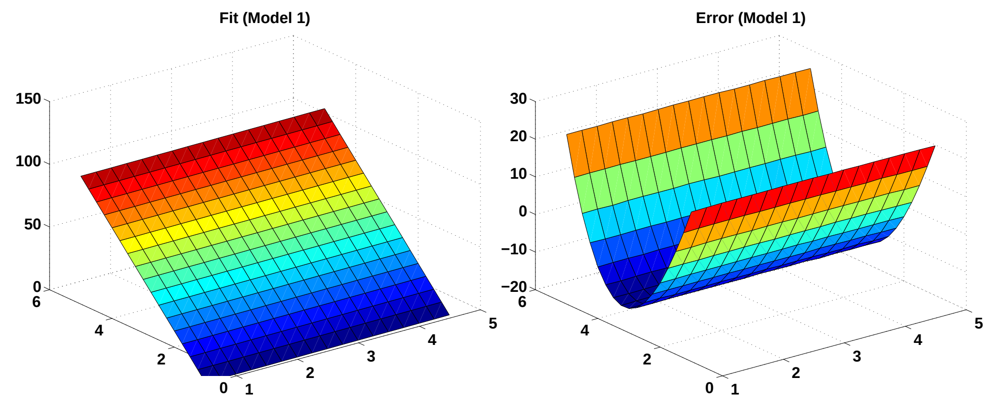



Figure: The optimal model fit, and the fitting error for the least squares best-fit in the model space $M(a, b, c) = a + bx + cy$. Here, the total LSQ-error is 42,282.

Lets try to fit a simple 4-parameter (bi-linear) model to this data

$$M(a, b, c) = a + bx + cy + dxy$$

```
matlab>> sz   = size(X);
         Bm   = reshape(X,prod(sz),1);
         Cm   = reshape(Y,prod(sz),1);
         Dm   = reshape(X.*Y,prod(sz),1);
         Am   = ones(size(Bm));
         RHS  = reshape(Fxy,prod(sz),1);
         A    = [Am Bm Cm Dm];
         coef = A \ RHS;
         fit  = coef(1) + coef(2)*X + coef(3)*Y + coef(4)*X.*Y;
         fitError = Fxy - fit;
         surf(x,y,fitError)
```

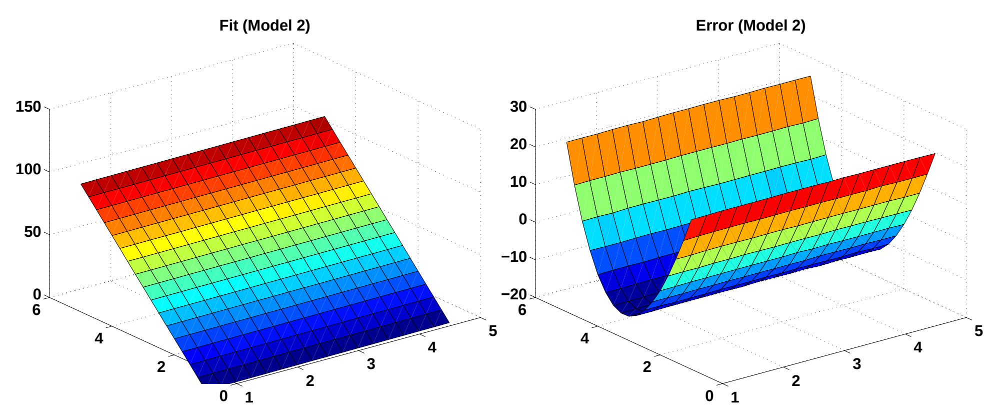



Figure: The fitting error for the least squares best-fit in the model space $M(a, b, c) = a + bx + cy + dxy$. — Still a pretty bad fit. Here, the total LSQ-error is still 42,282.

Since the main problem is in the y-direction, we try a 4-parameter model with a quadratic term in y

$$M(a,b,c) = a + bx + cy + dy^2$$

```
matlab>> sz    = size(X);
         Bm    = reshape(X,prod(sz),1);
         Cm    = reshape(Y,prod(sz),1);
         Dm    = reshape(Y.*Y,prod(sz),1);
         Am    = ones(size(Bm));
         RHS   = reshape(Fxy,prod(sz),1);
         A     = [Am Bm Cm Dm];
         coef  = A \ RHS;
         fit   = coef(1) + coef(2)*X + coef(3)*Y + coef(4)*Y.*Y;
         fitError = Fxy - fit;
         surf(x,y,fitError)
```

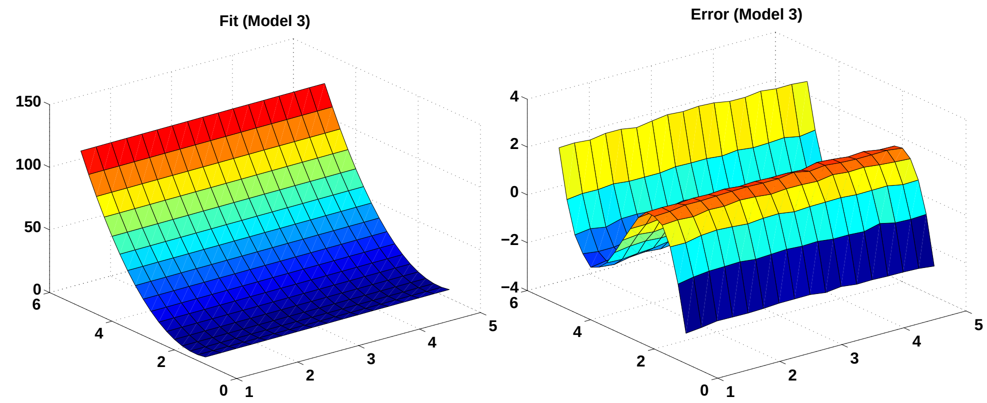



Figure: The fitting error for the least squares best-fit in the model space $M(a,b,c) = a + bx + cy + dy^2$. — We see a significant drop in the error (one order of magnitude); and the total LSQ-error has dropped to 578.8.

We notice something interesting: the addition of the xy-term to the model did not produce a drop in the LSQ-error. However, the y^2 allowed us to capture a lot more of the action.

The change in the LSQ-error as a function of an added term is one way to decide what is a useful addition to the model.

Why not add both the xy and y^2 always?

	xy	y^2	**Both**
$\kappa(A)$	86.2	107.3	170.5
$\kappa(A^TA)$	7,422	11,515	29,066

Table: Condition numbers for the A-matrices (and associated Normal Equations) for the different models.

We fit a 5-parameter model with a quadratic term in y

$$M(a,b,c) = a + bx + cy + dy^2 + ey^3$$

```
matlab>> sz    = size(X);
         Bm    = reshape(X,prod(sz),1);
         Cm    = reshape(Y,prod(sz),1);
         Dm    = reshape(Y.*Y,prod(sz),1);
         Em    = reshape(Y.*Y.*Y,prod(sz),1);
         Am    = ones(size(Bm));
         RHS   = reshape(Fxy,prod(sz),1);
         A     = [Am Bm Cm Dm Em];
         coef  = A \ RHS;
         fit   = coef(1) + coef(2)*X + coef(3)*Y + coef(4)*Y.*Y +
         coef(5)*Y.^3;
         fitError = Fxy - fit;
         surf(x,y,fitError)
```

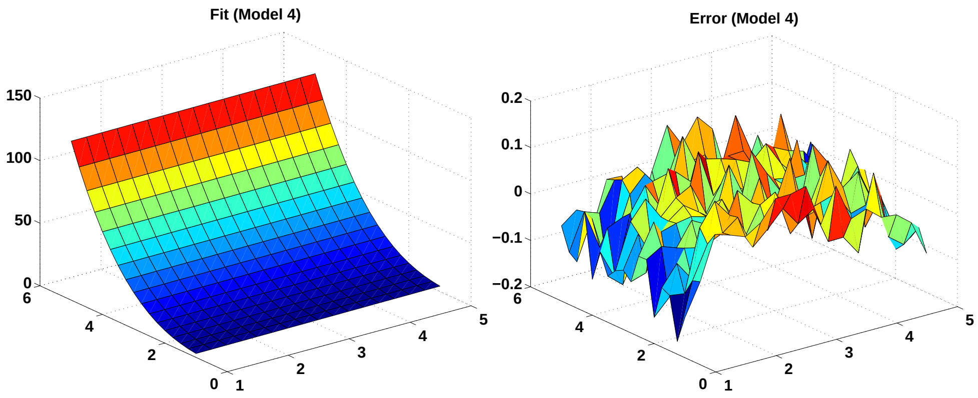


Figure: The fitting error for the least squares best-fit in the model space $M(a,b,c) = a + bx + cy + dy^2 + ey^3$. — We now have a pretty good fit. The LSQ-error is now down to 0.9864.

Model	**LSQ-error**	$\kappa(A^T A)$
$a + bx + cy$	42,282	278
$a + bx + cy + dxy$	42,282	7,422
$a + bx + cy + dy^2$	578.8	11,515
$a + bx + cy + ey^3$	2.695	107,204
$a + bx + cy + dy^2 + ey^3$	0.9864	1,873,124

Table: Summary of LSQ-error and conditioning of the Normal Equations for the various models. We notice that additional columns in the A-matrix (additional model parameters) have a **severe** effect on the conditioning of the Normal Equations.

Moving to Even Higher Dimensions

At this point we can state the Linear Least Squares fitting problem in any number of dimensions, and we can use exotic models if we want to.

In 3D we need 10 parameters to fit a model with all linear, and second order terms

$$\begin{aligned} &M(a,b,c,d,e,f,g,h,i,j) = \\ &\quad a + bx + cy + dz + ex^2 + fy^2 + gz^2 + hxy + ixz + jyz \end{aligned}$$

With n_x, n_y, and n_z data points in the x-, y-, and z-directions (respectively) we end up with a matrix A of dimension $(n_x \cdot n_y \cdot n_z) \times 10$.

Ill-conditioning of the Normal Equations

Needless(?) to say, the normal equations can be quite ill-conditioned in this case. The ill-conditioning can be eased by searching for sets of orthogonal functions with respect to the inner products

$$\langle f(x), g(x) \rangle = \int_{x_a}^{x_b} \int_{y_a}^{y_b} \int_{z_a}^{z_b} f(x,y,z) g(x,y,z)^* \, dx \, dy \, dz$$

$$[f(x), g(x)] = \sum_{x_k \in \mathbb{X}} \sum_{y_\ell \in \mathbb{Y}} \sum_{z_m \in \mathbb{Z}} f(x_k, y_\ell, z_m) g(x_k, y_\ell, z_m)^*$$

That's *sometimes* possible, but we'll leave the details as an exercise for a dark and stormy night...