Data Modeling and Analysis Techniques: Interpolation and Approximation

Hong Qin

Rm. 151, NEW CS Building

Department of Computer Science

Stony Brook University (State University of New York)

Stony Brook, New York 11794-2424

Tel: (631)632-8450; Fax: (631)632-8334

qin@cs.sunysb.edu or qin@cs.stonybrook.edu

http://www.cs.stonybrook.edu/~qin

Data Interpolation

- Why interpolation?
- We acquire discrete observations/measurements for continuous systems, and we would like to convert discrete measurements to continuous representations
- We definitely need the ability to interpolate values "in-between" discrete points

Data Interpolation

- One simple example
- Our goal is to find the value of a function between known values
- Let us consider the two pairs of values (x,y):

$$(0.0, 1.0), \text{ and } (1.0, 2.0)$$

What is y at $x = 0.5$? That is, what's $(0.5, y)$?

Linear Interpolation

- Given two points, (x_1,y_1), (x_2,y_2):

  Fit a straight line between the points

$$y(x) = a\,x + b$$

$$a=(y_2-y_1)/(x_2-x_1), \quad b= (y_1\,x_2-y_2\,x_1)/(x_2-x_1),$$

Use this equation to find y values for any

$$x_1 < x < x_2$$

Another Example

- What about four points ?
- (0, 2), (1, 0.3975), (2, -0.1126), (3, -0.0986)

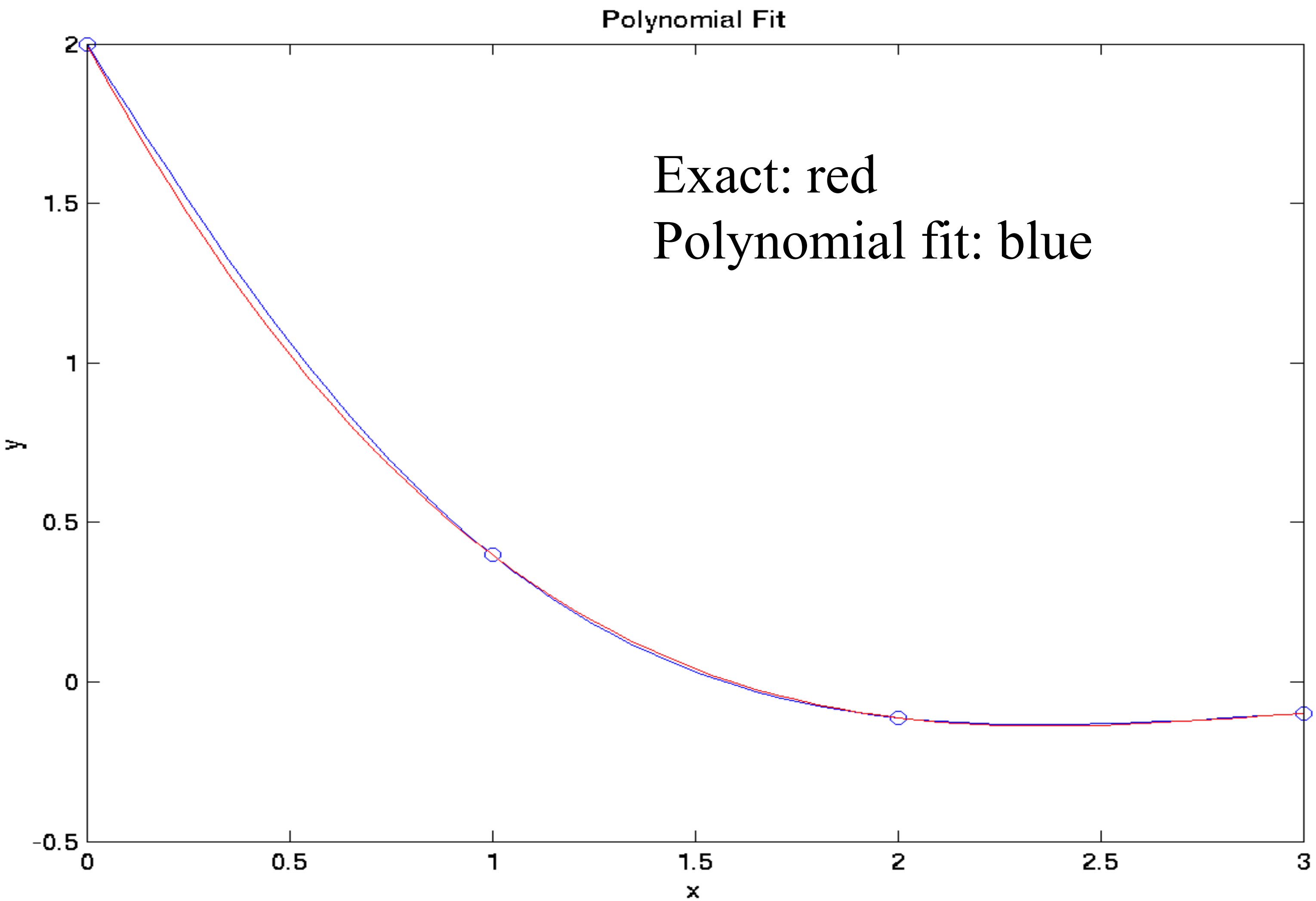
Another Example

Data points are: (0,2), (1,0.3975), (2, -0.1126), (3, -0.0986).

Fitting a cubic polynomial through the four points gives:

$$y_p(x) = 2.0 - 2.3380x + 0.8302x^2 - 0.0947x^3$$

Polynomial Fit to Example



Polynomial Interpolants

- Now given n (n=4) data points $(x_i, y_i), i = 1,4$
- Find the interpolating function that goes through these points, will need a cubic polynomial

$$y_p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$$

- If there are n+1 data points, the function will become (with n+1 unknown variables)

$$y_p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots + a_N x^N$$

Polynomial Interpolant

- The polynomial must pass through the four points, resulting in the following constraints

$$\begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ 1 & x_4 & x_4^2 & x_4^3 \end{pmatrix} \begin{pmatrix} a_o \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$$

$$\mathbf{p}\,\mathbf{a} = \mathbf{y}$$

$$\mathbf{a} = \mathbf{p}^{-1}\,\mathbf{y}$$

Ordinary Least-Squares

Least Squares Interpolant

- For n points, we only have a fitting polynomial of order m (m < (n-1)), we want the least squares fitting polynomial is similar to the exact fit form:

$$\mathbf{y_p(x) = p\,a}$$

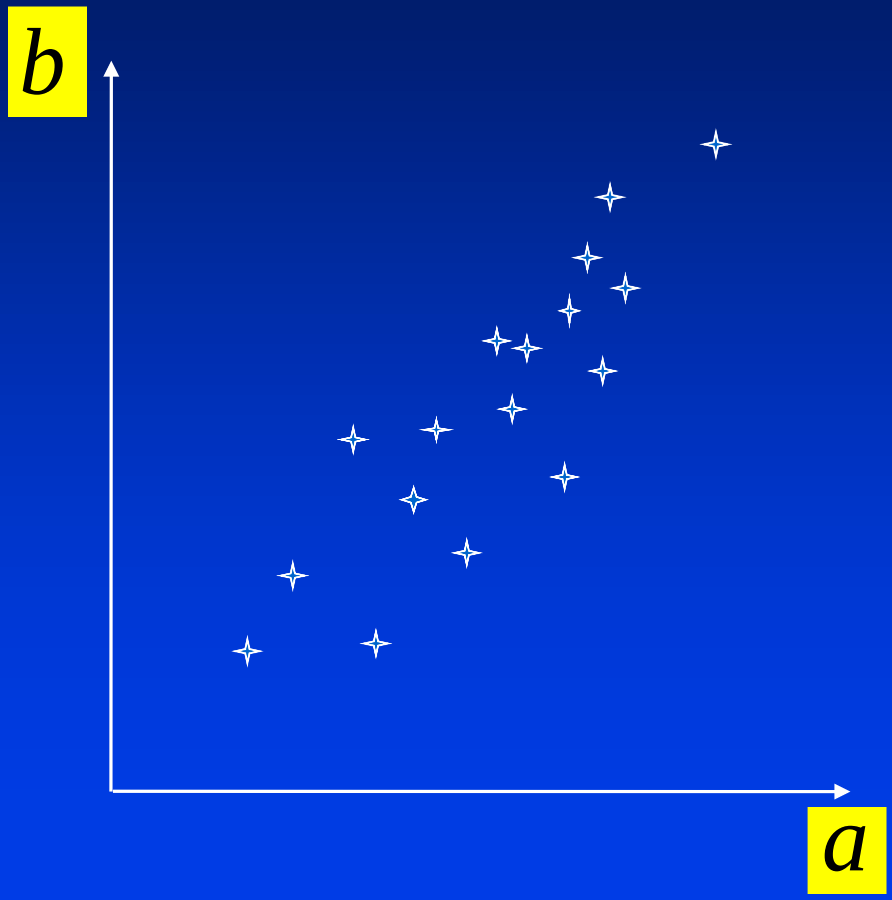
- Now p is becoming a n * m matrix. We have fewer unknowns than data points, the interpolant can not go through all the points exactly, we need to measure the total error

$$\epsilon_i = y_p(x_i) - y_i \qquad \sum_{i=1}^{N} \epsilon_i^2$$

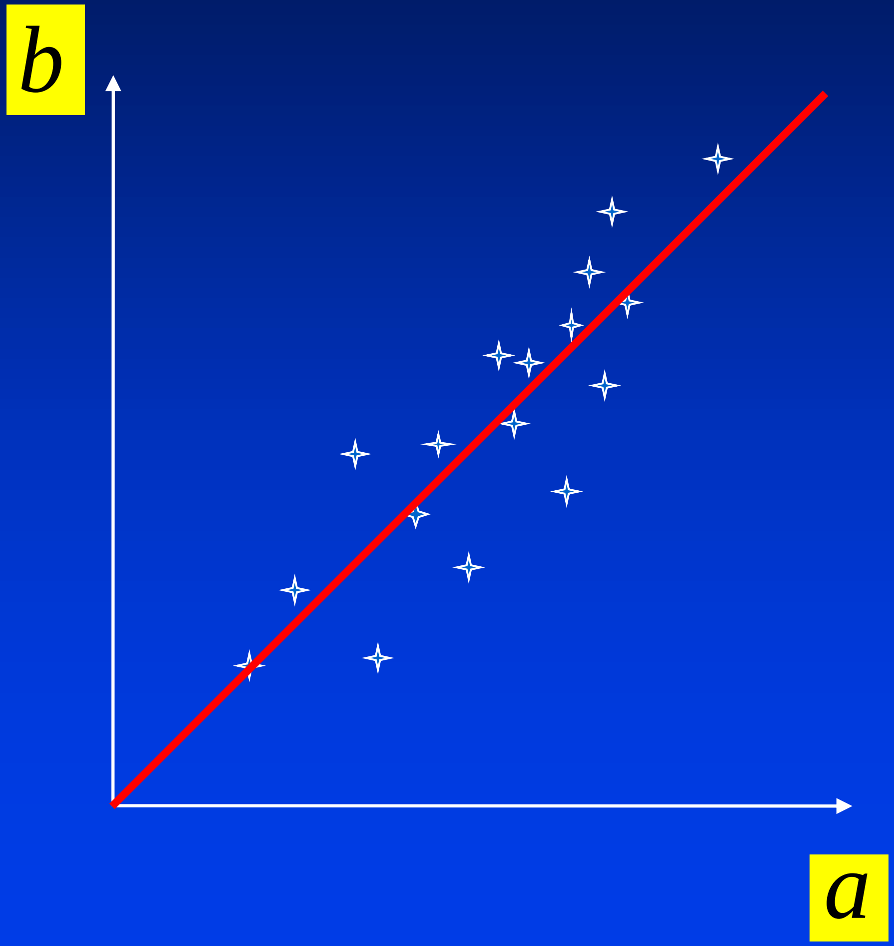
Outline

- Linear regression
- Geometry of least-squares
- Discussion of the Gauss-Markov theorem

Ordinary Least-Squares



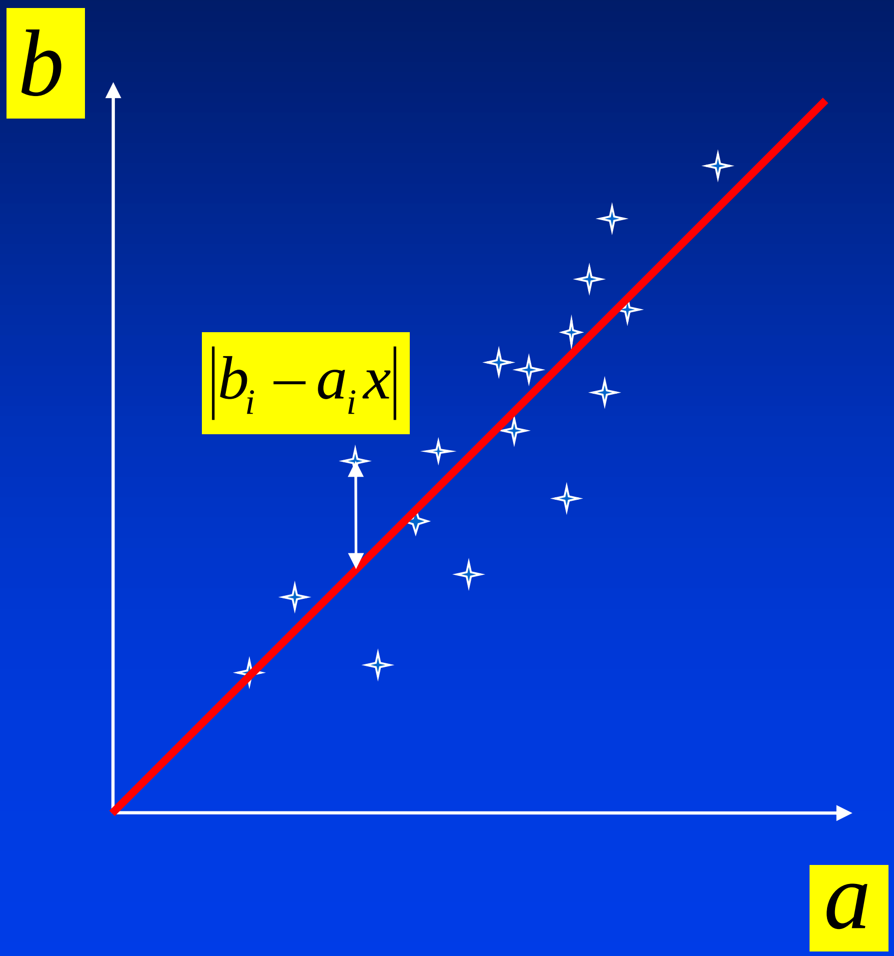
One-dimensional Regression



Find a line that represent the ”best” linear relationship:

$$b = ax$$

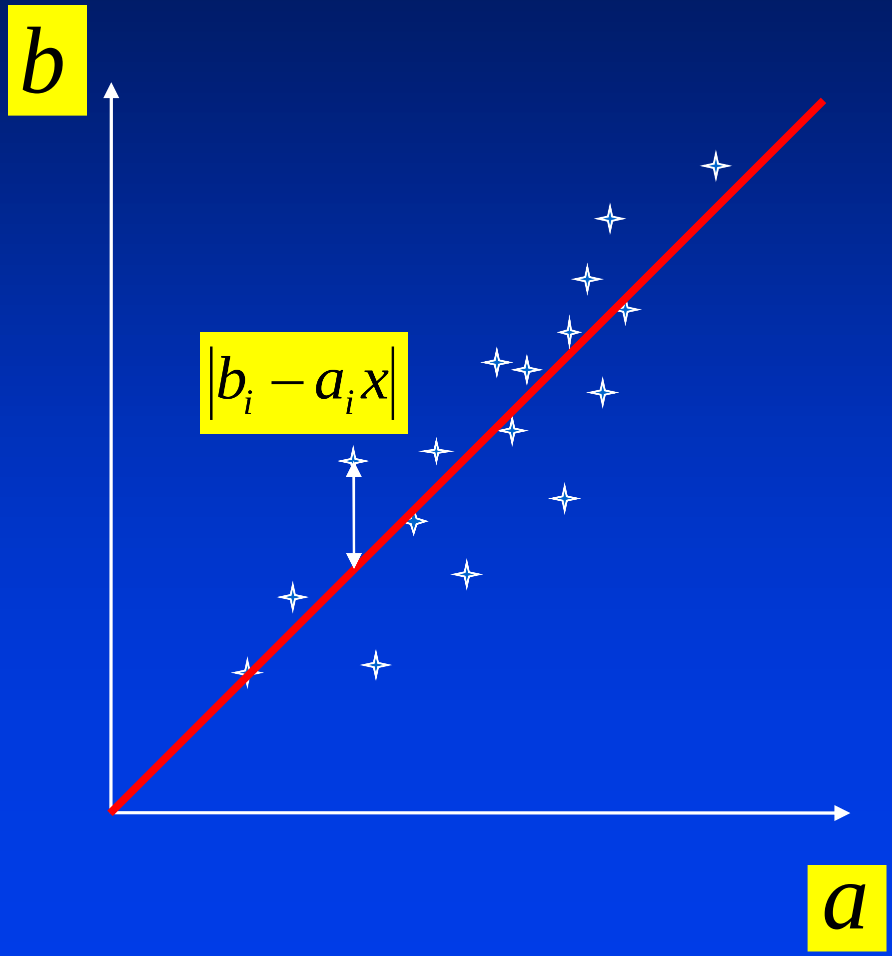
One-dimensional Regression



- Problem: the line does NOT go through all the data points exactly, so only approximation

$$e_i = b_i - a_i x$$

One-dimensional Regression



- Find the line that minimizes the sum of error squared:

$$\sum_i (b_i - a_i x)^2$$

Matrix Notation

Using the following notations

$$\mathbf{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

we can rewrite the error function using linear algebra as:

$$e(x) = \sum_i (b_i - a_i x)^2$$

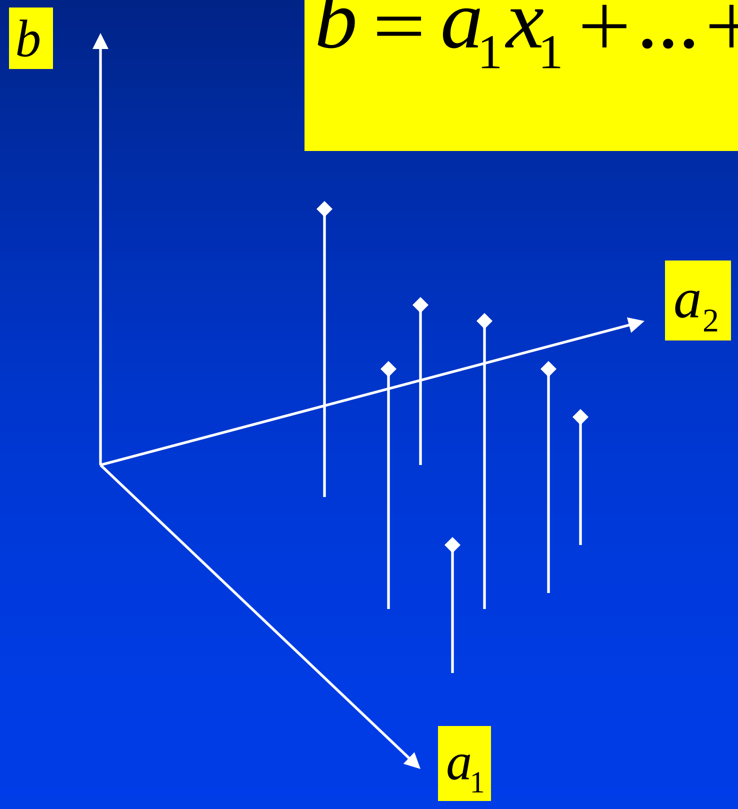
$$= (\mathbf{b} - x\mathbf{a})^T (\mathbf{b} - x\mathbf{a})$$

$$e(x) = \|\mathbf{b} - x\mathbf{a}\|^2$$

Multidimensional Linear Regression

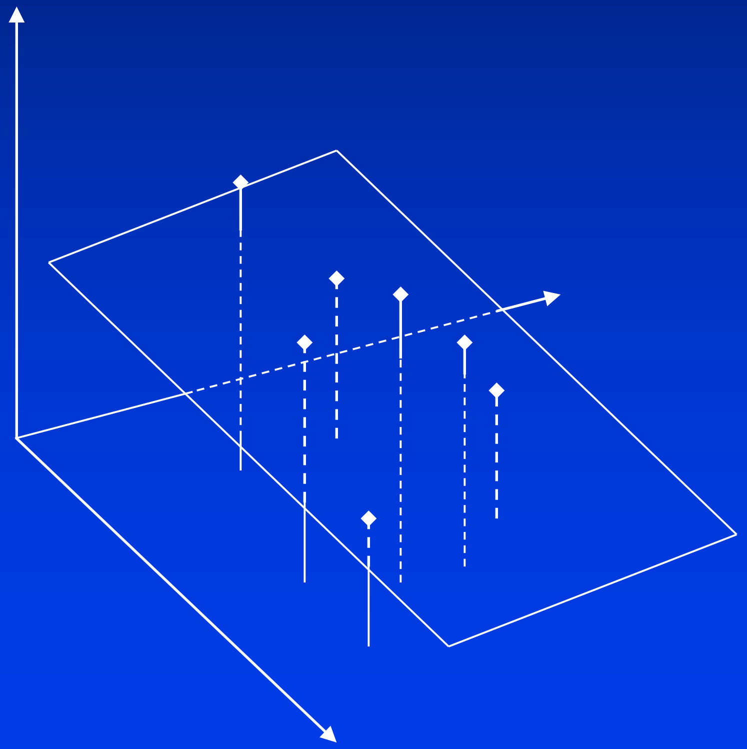
Using a model with *m* parameters

$$b = a_1 x_1 + \ldots + a_m x_m = \sum_j a_j x_j$$



Multidimensional Linear Regression

Using a model with *m* parameters



Multidimensional Linear Regression

Using a model with *m* parameters

$$b = a_1 x_1 + \ldots + a_m x_m = \sum_j a_j x_j$$

and *n* measurements

$$e(\mathbf{x}) = \sum_{i=1}^{n} \left(b_i - \sum_{j=1}^{m} a_{i,j} x_j \right)^2$$

$$= \left\| \mathbf{b} - \left[\sum_{j=1}^{m} a_{i,j} x_j \right] \right\|^2$$

$$e(\mathbf{x}) = \left\| \mathbf{b} - \mathbf{A}\mathbf{x} \right\|^2$$

Matrix Notation

$$\boldsymbol{b} - \boldsymbol{Ax}$$

$$\boldsymbol{b} - \boldsymbol{Ax} = \begin{bmatrix} b_1 \\ : \\ b_n \end{bmatrix} - \begin{bmatrix} a_{1,1} & .. & a_{1,m} \\ : & & : \\ a_{n,1} & .. & a_{n,m} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ : \\ x_m \end{bmatrix}$$

Matrix Notation

$$\boldsymbol{b} - \boldsymbol{Ax} = \begin{bmatrix} b_1 \\ : \\ b_n \end{bmatrix} - \begin{bmatrix} a_{1,1} & .. & a_{1,m} \\ : & & : \\ a_{n,1} & .. & a_{n,m} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ : \\ x_m \end{bmatrix}$$

$$= \begin{bmatrix} b_1 - (a_{1,1}x_1 + \ldots + a_{1,m}x_m) \\ : \\ b_n - (a_{n,1}x_1 + \ldots + a_{n,m}x_m) \end{bmatrix}$$

$\boldsymbol{b} - \boldsymbol{Ax}$

parameter 1

$$\boldsymbol{b} - \boldsymbol{Ax} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} - \begin{bmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

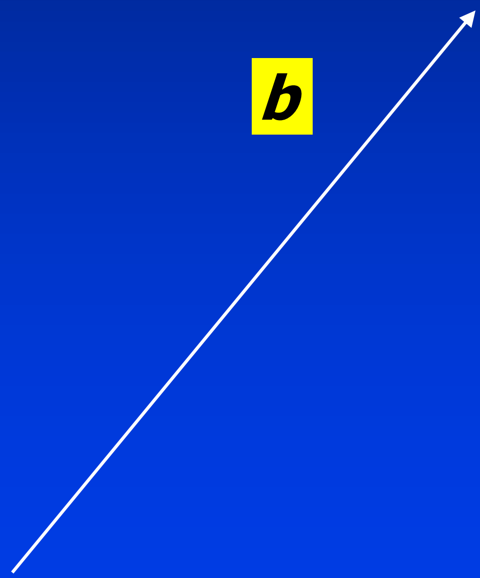
measurement *n*

$$= \begin{bmatrix} b_1 - (a_{1,1}x_1 + \ldots + a_{1,m}x_m) \\ \vdots \\ b_n - (a_{n,1}x_1 + \ldots + a_{n,m}x_m) \end{bmatrix}$$

Geometric Interpretation

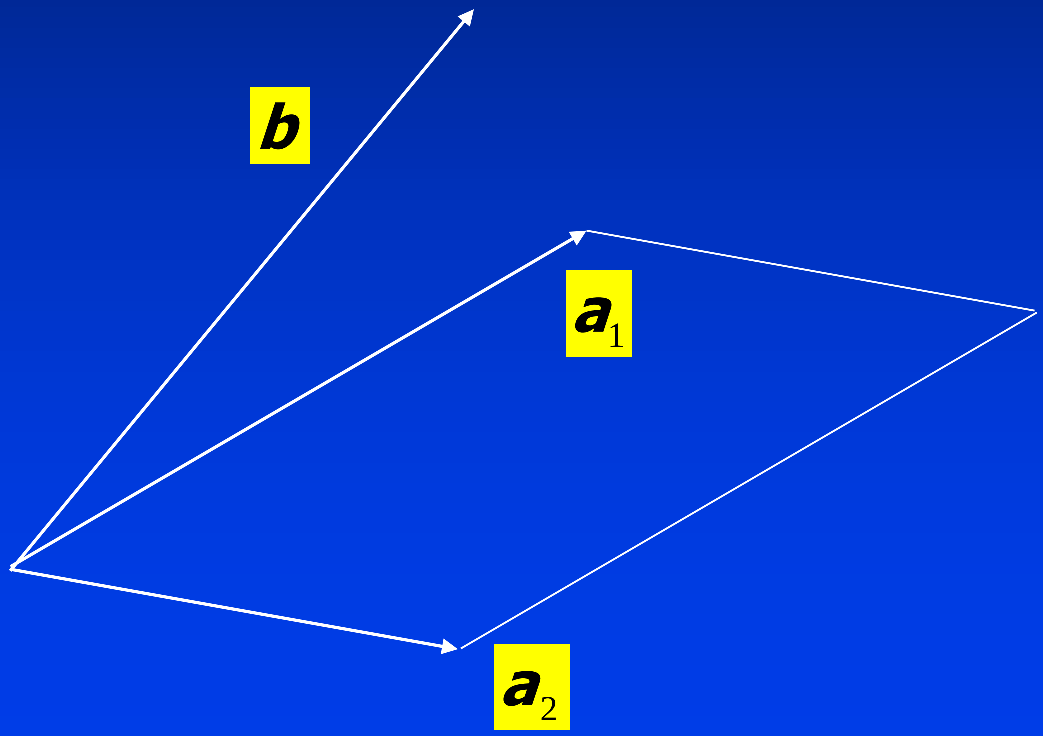
Geometric Interpretation

- ***b*** is a vector in R^n



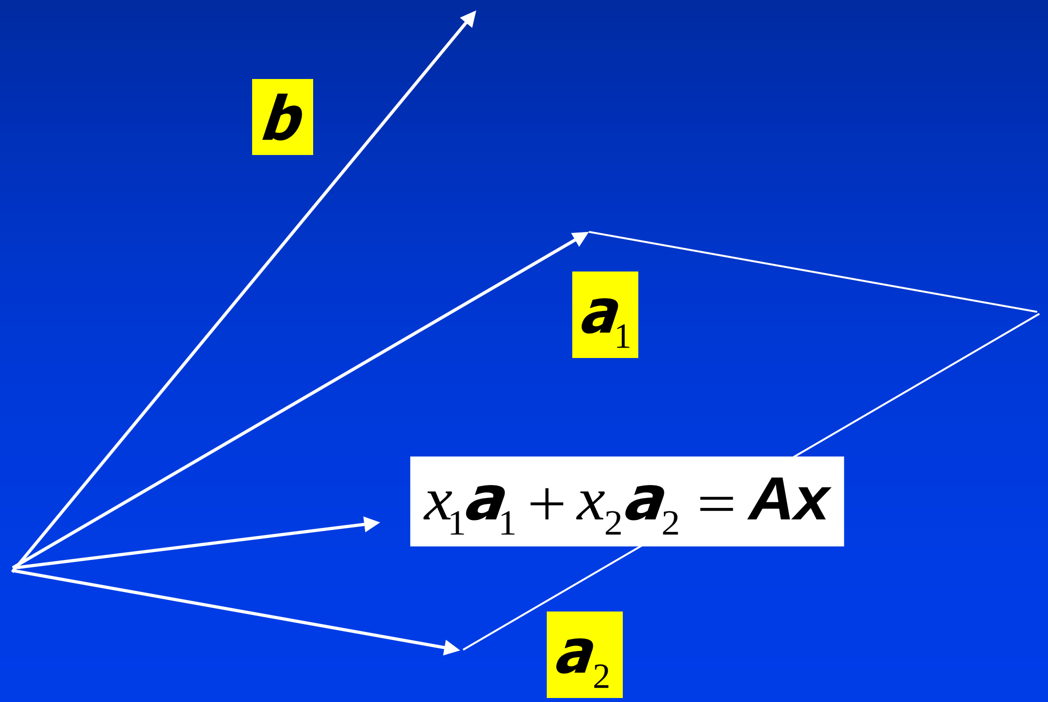
Geometric interpretation

- ***b*** is a vector in R^n
- The columns of ***A*** define a vector space *range(A)*



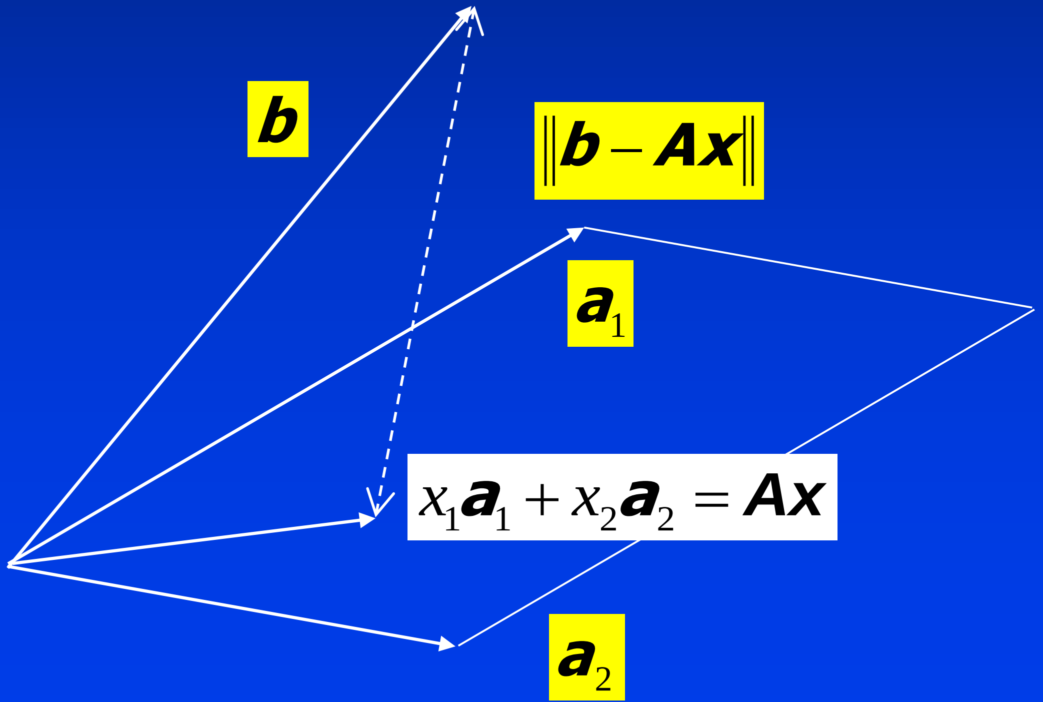
Geometric Interpretation

- ***b*** is a vector in R^n
- The columns of ***A*** define a vector space *range(A)*
- ***Ax*** is an arbitrary vector in *range(A)*



Geometric Interpretation

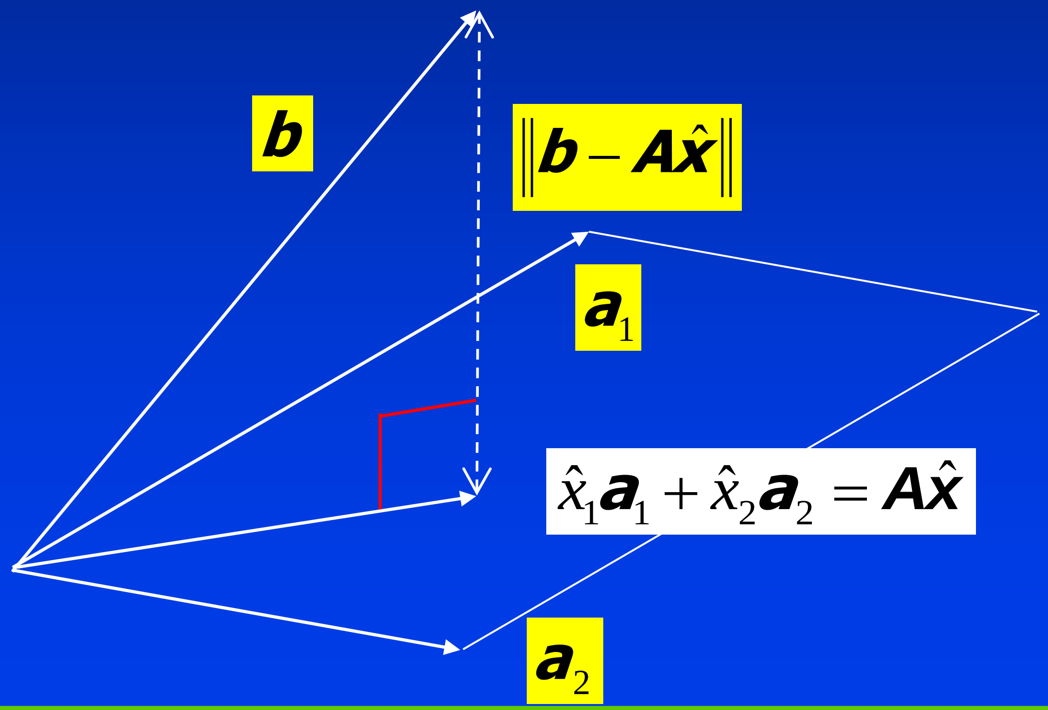
- ***b*** is a vector in R^n
- The columns of ***A*** define a vector space *range(A)*
- ***Ax*** is an arbitrary vector in *range(A)*



Geometric Interpretation

- $A\hat{x}$ is the orthogonal projection of ***b*** onto *range(A)*

$$\Leftrightarrow A^T(b - A\hat{x}) = 0 \Leftrightarrow A^T A\hat{x} = A^T b$$



The Normal Equation

$$\boldsymbol{A}^T \boldsymbol{A}\hat{\boldsymbol{x}} = \boldsymbol{A}^T \boldsymbol{b}$$

The Normal Equation: $A^T A\hat{x} = A^T b$

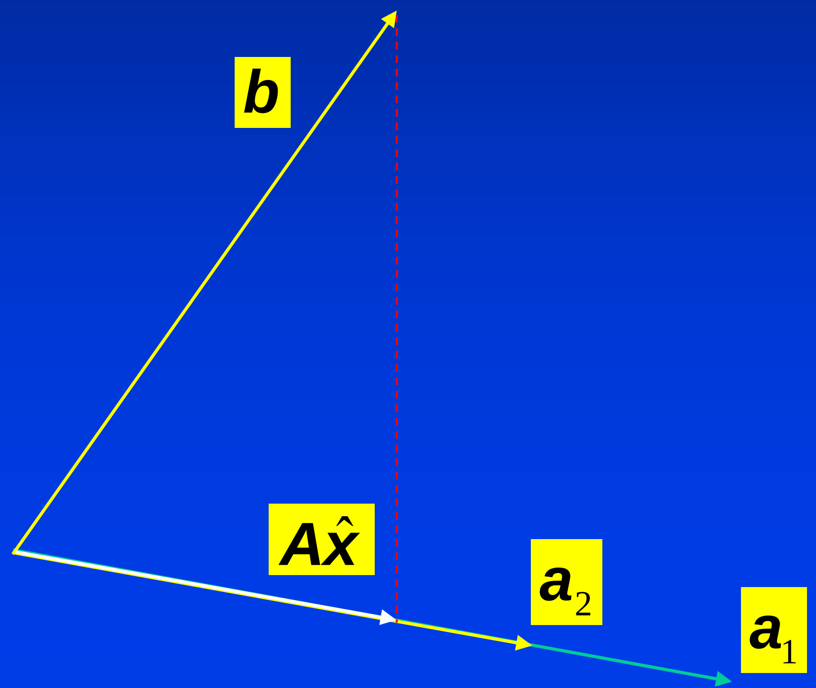
- Existence: $A^T A\hat{x} = A^T b$ has always a solution

The Normal Equation: $A^T A\hat{x} = A^T b$

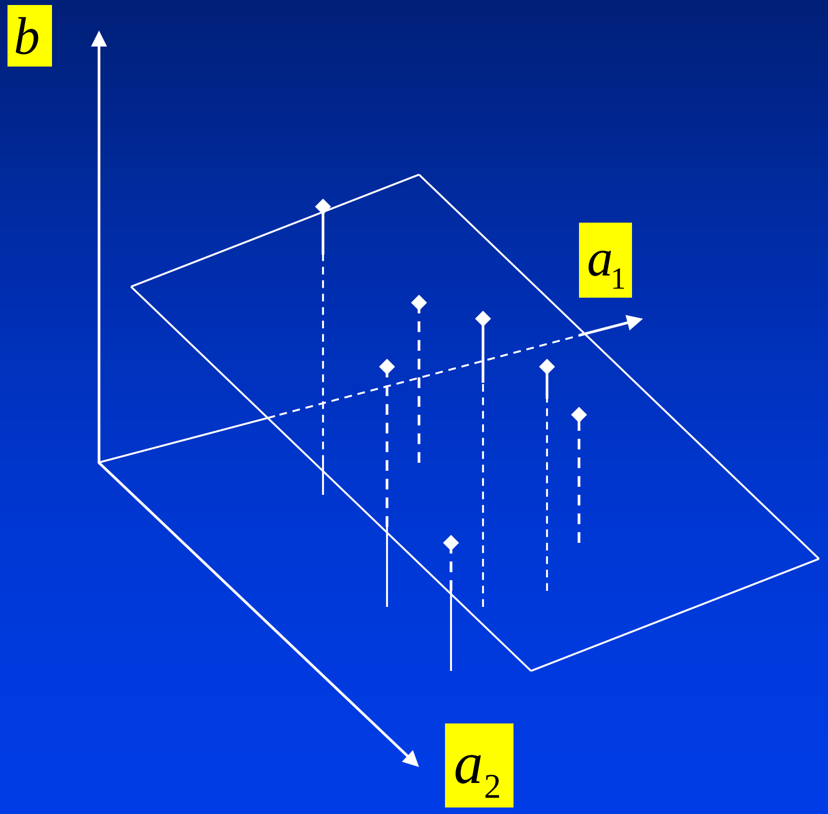
- Existence: $A^T A\hat{x} = A^T b$ has always a solution
- Uniqueness: the solution is unique if the columns of A are linearly independent

The Normal Equation: $\boldsymbol{A}^T \boldsymbol{A}\hat{\boldsymbol{x}} = \boldsymbol{A}^T \boldsymbol{b}$

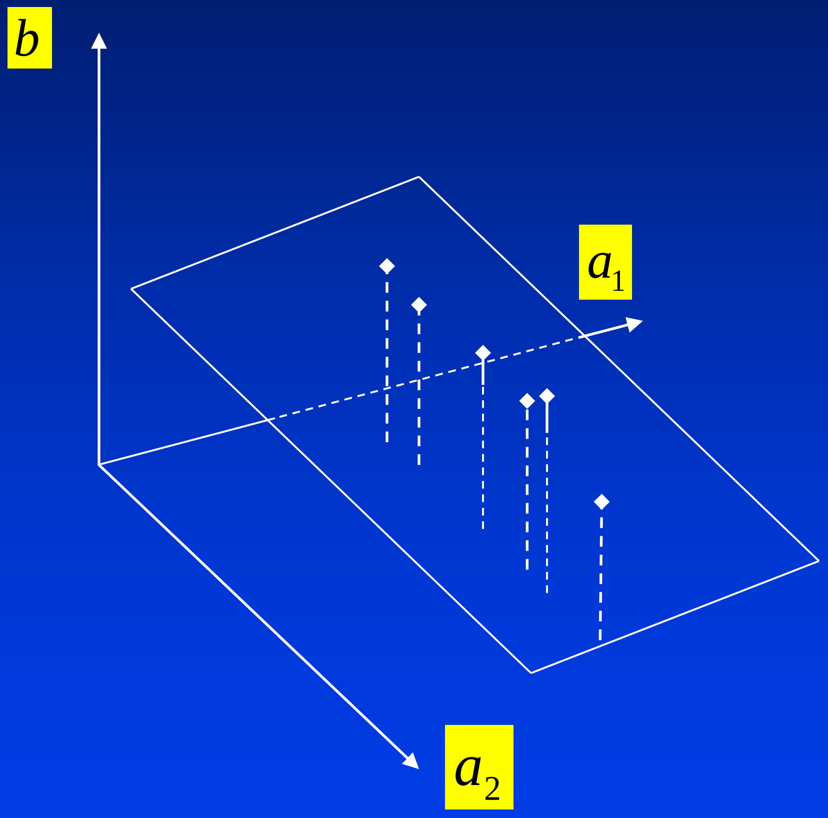
- Existence: $\boldsymbol{A}^T \boldsymbol{A}\hat{\boldsymbol{x}} = \boldsymbol{A}^T \boldsymbol{b}$ has always a solution
- Uniqueness : the solution is unique if the columns of $\boldsymbol{A}$ are linearly independent



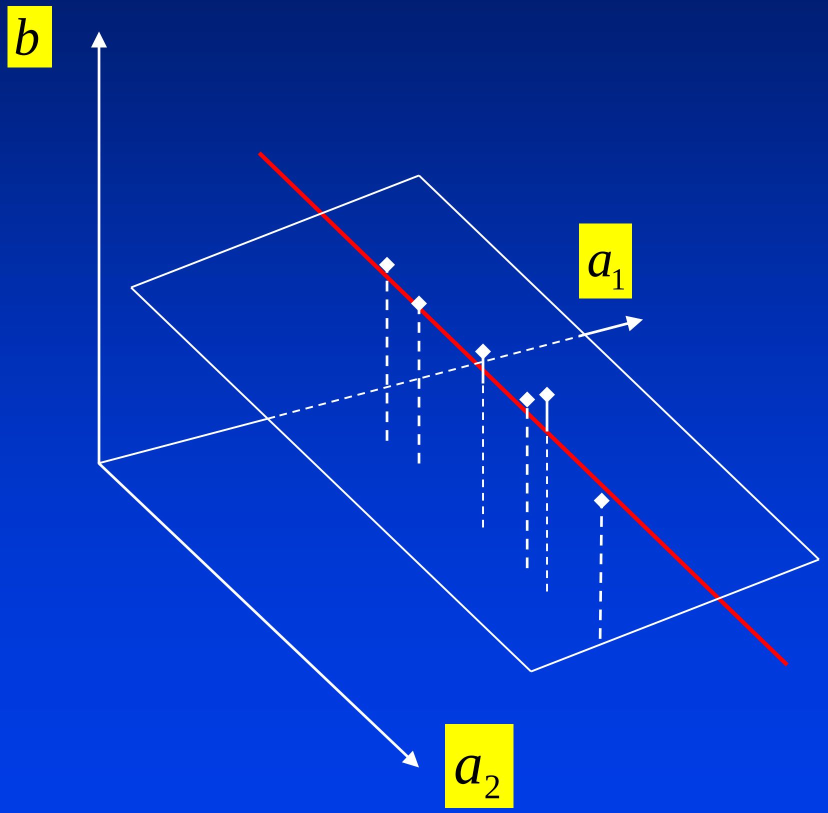
Under-constrained Problem



Under-constrained Problem

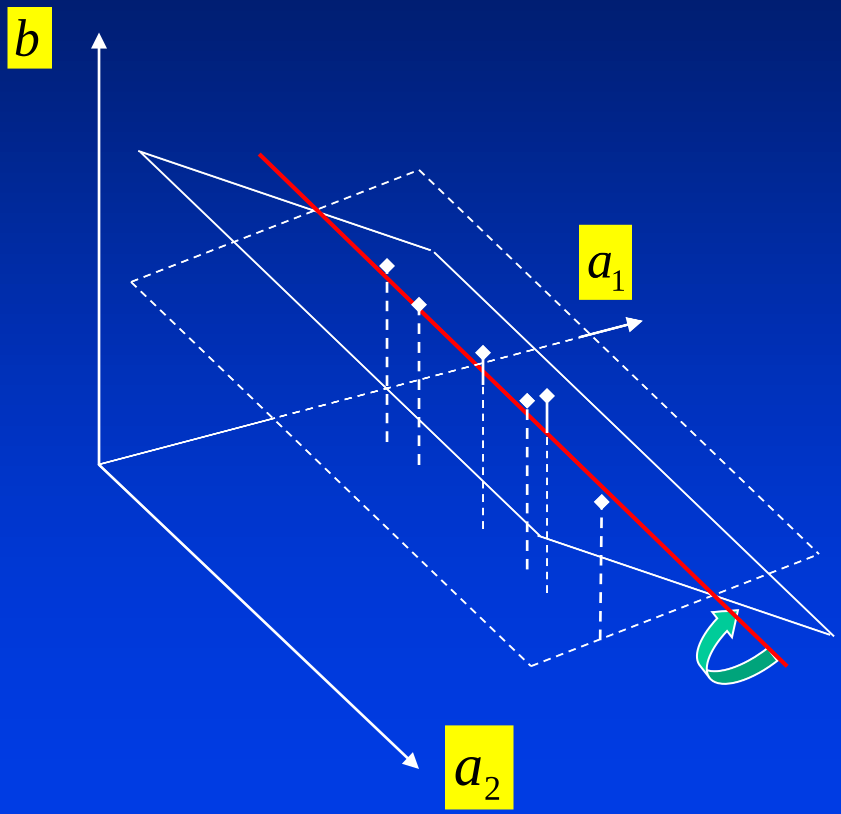


Under-constrained Problem



Under-constrained Problem

- Poorly selected data
- One or more of the parameters are redundant



Under-constrained Problem

- Poorly selected data
- One or more of the parameters are redundant

Add constraints

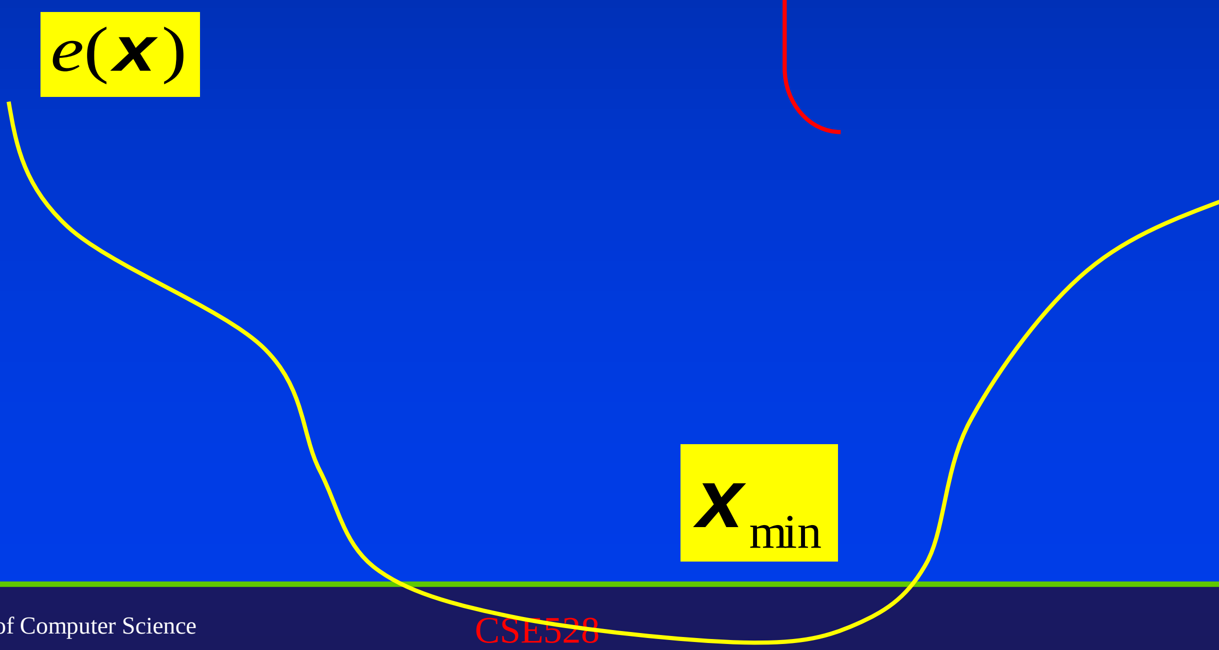
$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b} \ \text{ with} \min_{\boldsymbol{x}} \|\boldsymbol{x}\|$$

Minimizing $e(\boldsymbol{x})$

$\boldsymbol{x}_{\min}$ minimizes $e(\boldsymbol{x})$ if

Minimizing $e(\boldsymbol{x})$

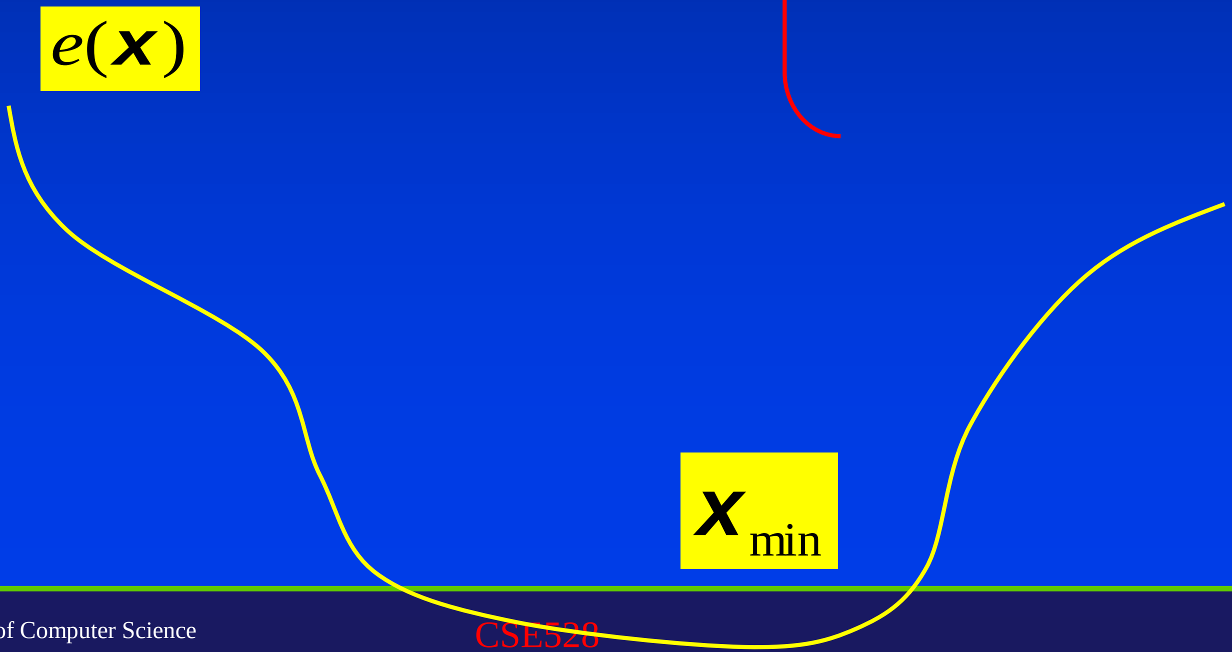
$\boldsymbol{x}_{\min}$ minimizes $e(\boldsymbol{x})$ if



Minimizing $e(\boldsymbol{x})$

$\boldsymbol{x}_{\min}$ minimizes $e(\boldsymbol{x})$ if

$e(\boldsymbol{x})$ is flat at $\boldsymbol{x}_{\min}$

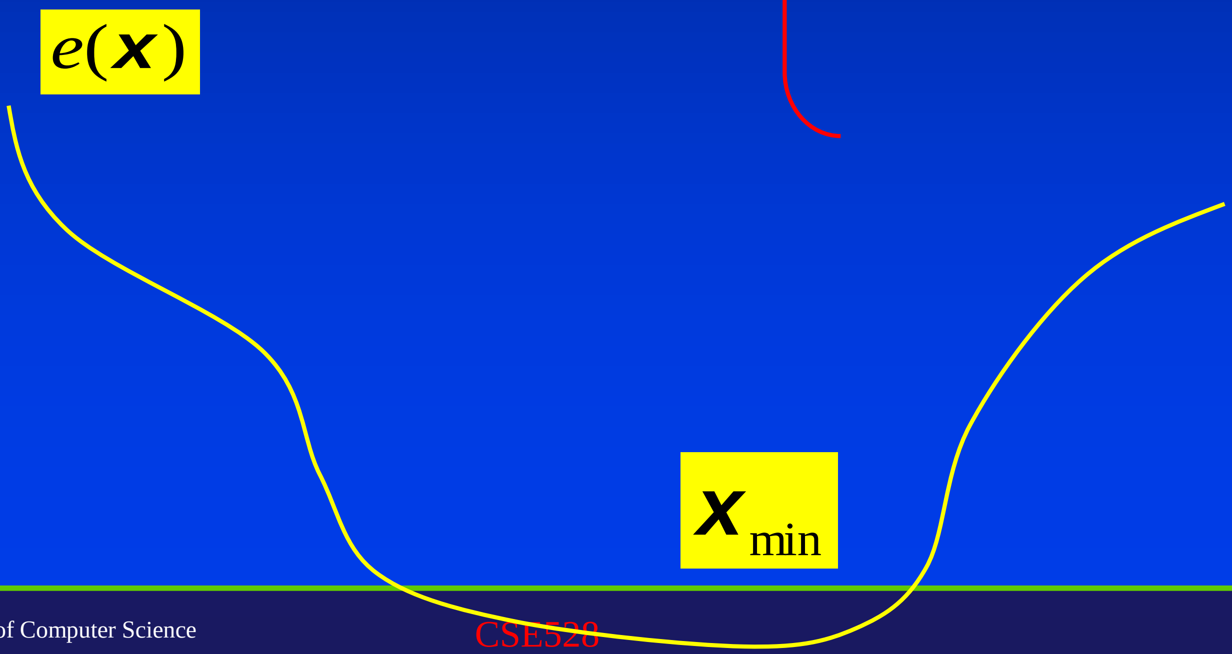


Minimizing $e(\boldsymbol{x})$

$\boldsymbol{x}_{\min}$ minimizes $e(\boldsymbol{x})$ if

$e(\boldsymbol{x})$ is flat at $\boldsymbol{x}_{\min}$

$$\nabla e(\boldsymbol{x}_{\min}) = \boldsymbol{0}$$



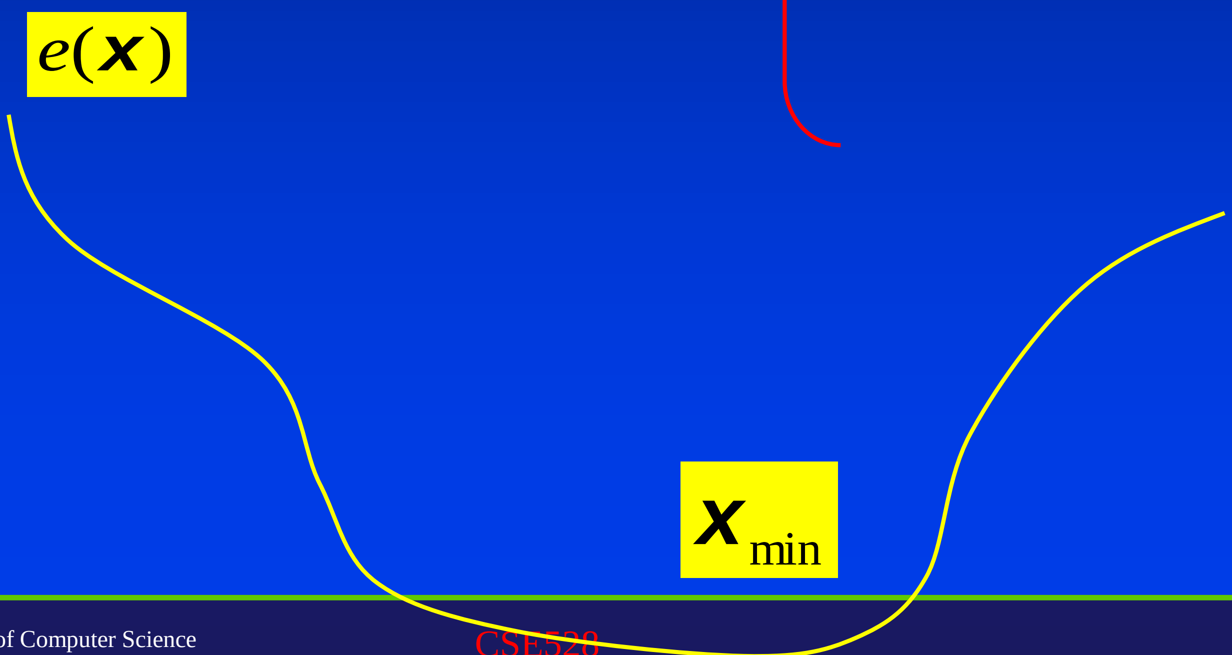
Minimizing $e(\boldsymbol{x})$

$\boldsymbol{x}_{\min}$ minimizes $e(\boldsymbol{x})$ if

- $e(\boldsymbol{x})$ is flat at $\boldsymbol{x}_{\min}$

$$\nabla e(\boldsymbol{x}_{\min}) = \boldsymbol{0}$$

- $e(\boldsymbol{x})$ does not go down around $\boldsymbol{x}_{\min}$



Minimizing $e(\mathbf{x})$

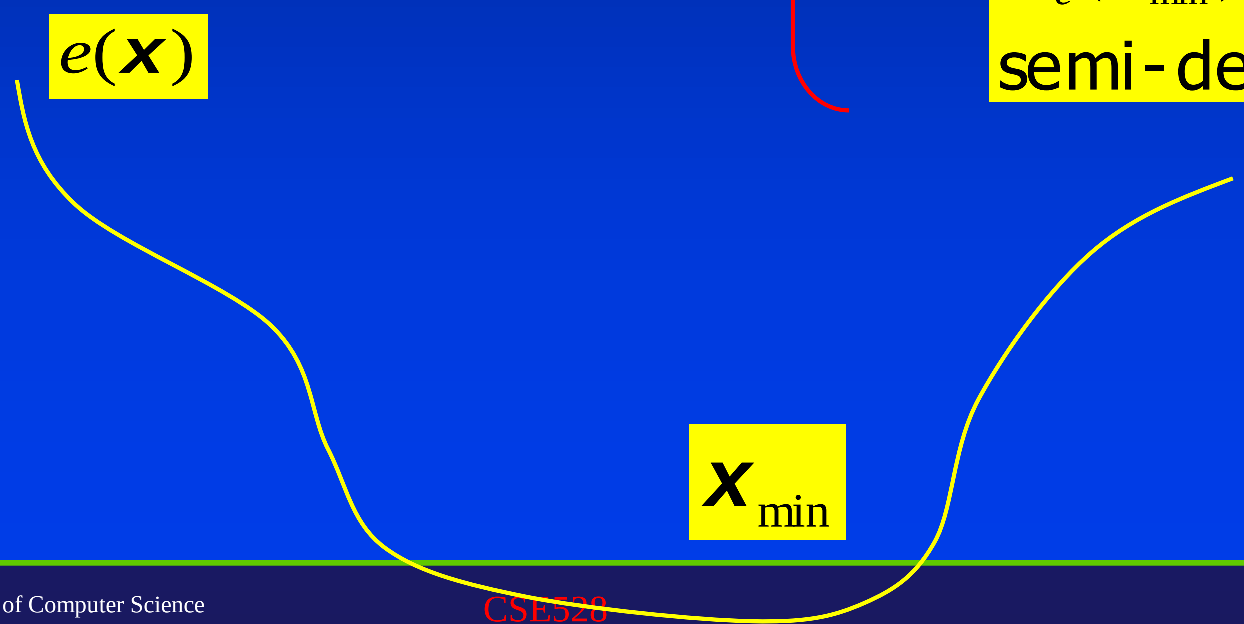
$\mathbf{x}_{\min}$ minimizes $e(\mathbf{x})$ if

- $e(\mathbf{x})$ is flat at $\mathbf{x}_{\min}$

  $$\nabla e(\mathbf{x}_{\min}) = \mathbf{0}$$

- $e(\mathbf{x})$ does not go down around $\mathbf{x}_{\min}$

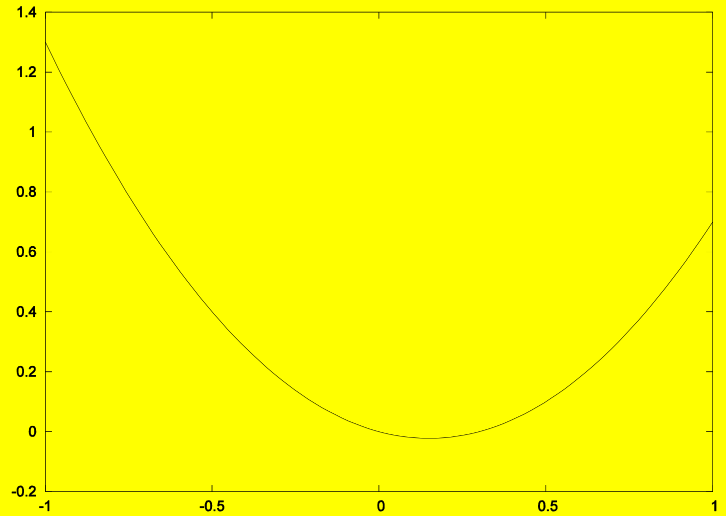
  $H_e(\mathbf{x}_{\min})$ is positive semi-definite



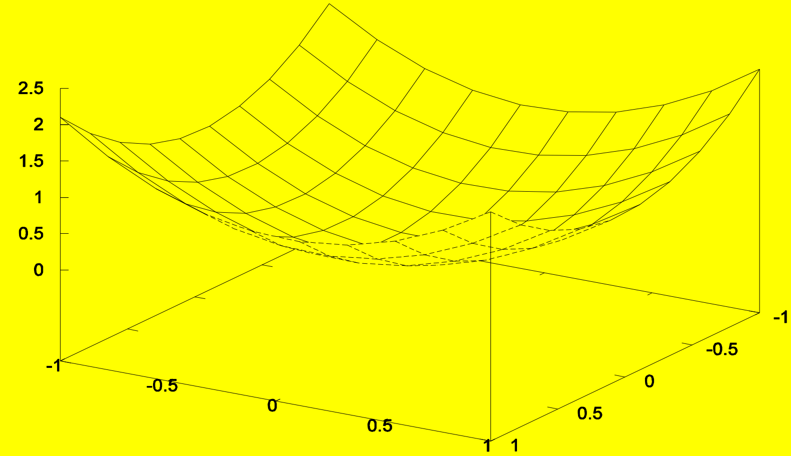
Positive Semi-definite

$$\boldsymbol{A} \text{ is positive semi-definite} \Leftrightarrow \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{0}, \text{ for all } \boldsymbol{x}$$

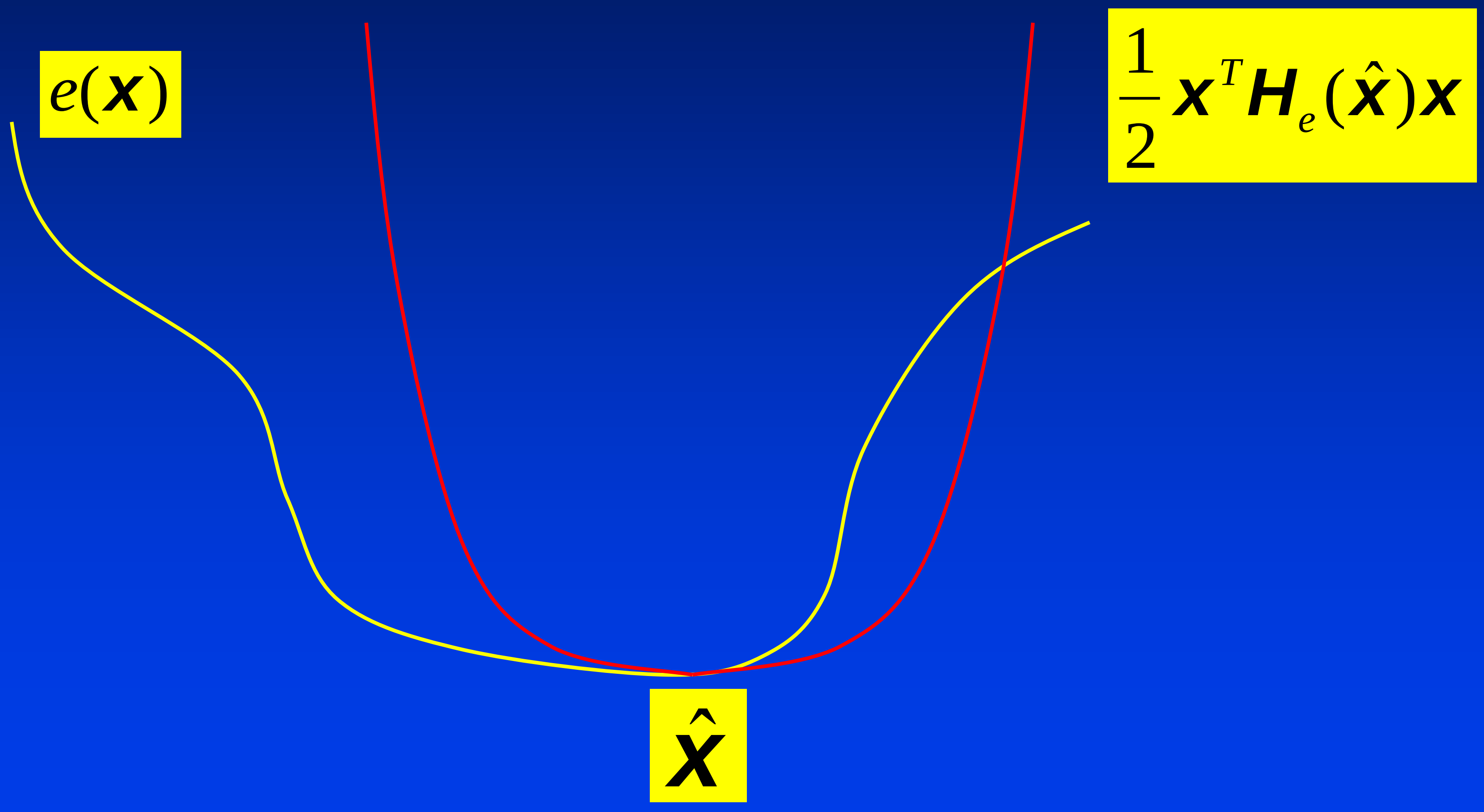
In 1-D



In 2-D

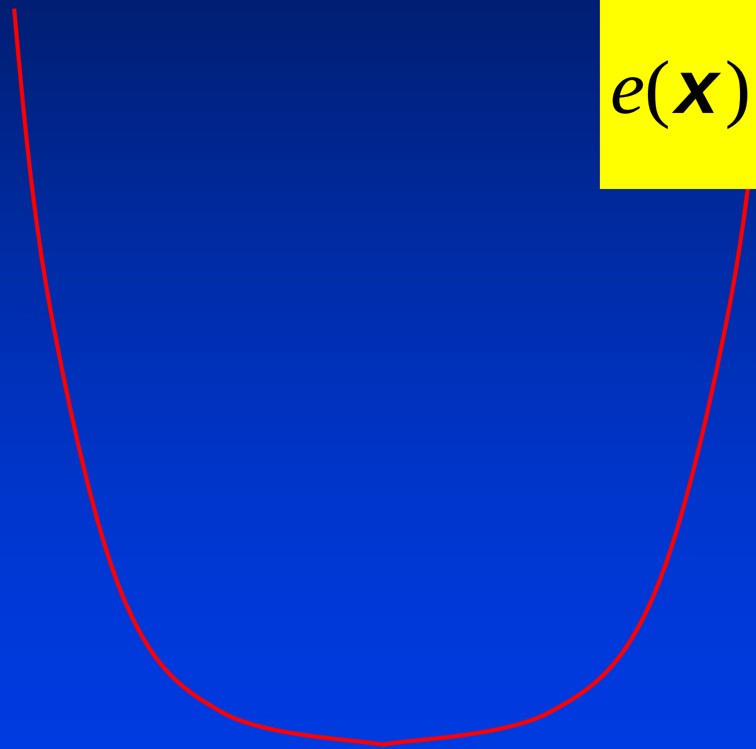


Minimizing $e(\boldsymbol{x})$



Minimizing $e(\mathbf{x}) = \|\mathbf{b} - \mathbf{A}\mathbf{x}\|^2$

$$e(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{H}_e(\hat{\mathbf{x}})\mathbf{x}$$



$\hat{\mathbf{x}}$

Minimizing $e(\boldsymbol{x}) = \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|^2$

$\hat{\boldsymbol{x}}$ minimizes $e(\boldsymbol{x})$ if

- $\boldsymbol{A}^T \boldsymbol{A} \hat{\boldsymbol{x}} = \boldsymbol{A}^T \boldsymbol{b}$
- $2\boldsymbol{A}^T \boldsymbol{A}$ is positive semi-definite

Minimizing $e(\boldsymbol{x}) = \left\| \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} \right\|^2$

$\hat{\boldsymbol{x}}$ minimizes $e(\boldsymbol{x})$ if

- $\boldsymbol{A}^T \boldsymbol{A} \hat{\boldsymbol{x}} = \boldsymbol{A}^T \boldsymbol{b}$
- $2\boldsymbol{A}^T \boldsymbol{A}$ is positive semi-definite

Always true

Minimizing $e(\mathbf{x}) = \|\mathbf{b} - \mathbf{A}\mathbf{x}\|^2$

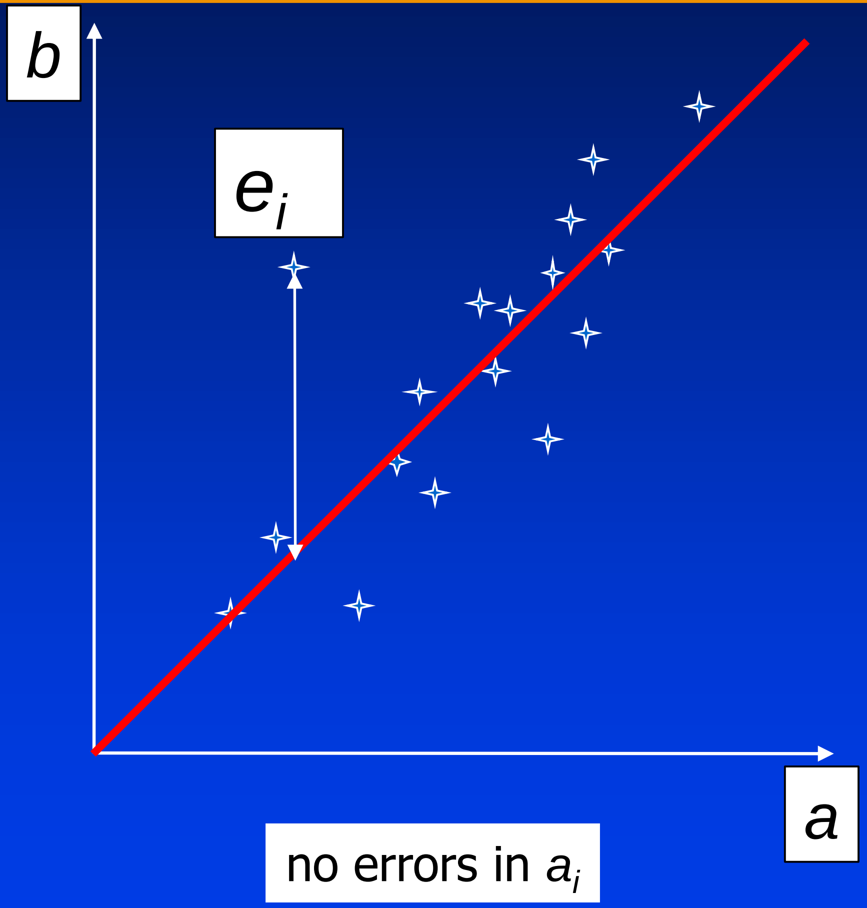
$\hat{\mathbf{x}}$ minimizes $e(\mathbf{x})$ if

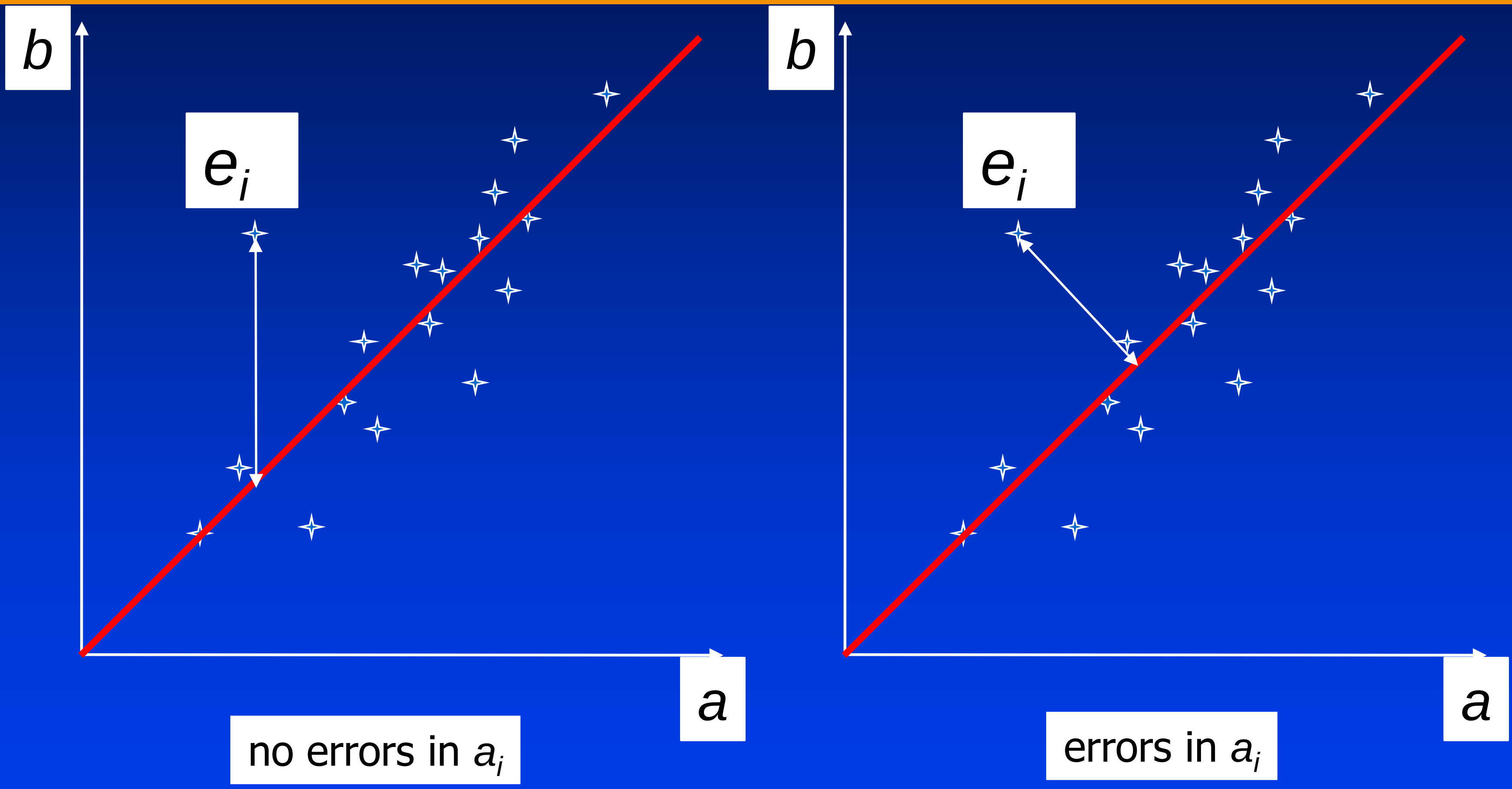
$$\mathbf{A}^T \mathbf{A} \hat{\mathbf{x}} = \mathbf{A}^T \mathbf{b}$$

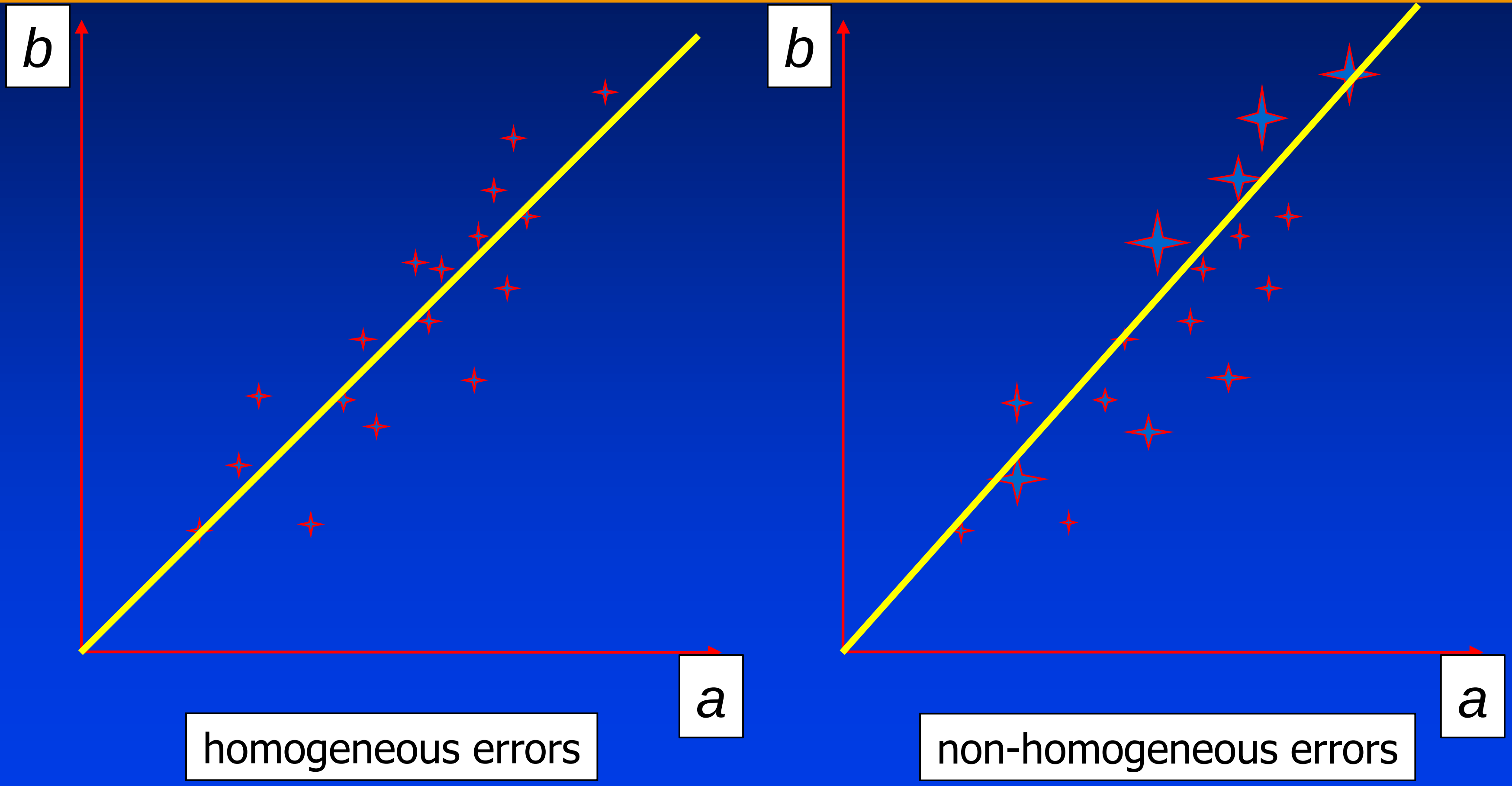
The ***Normal Equation***

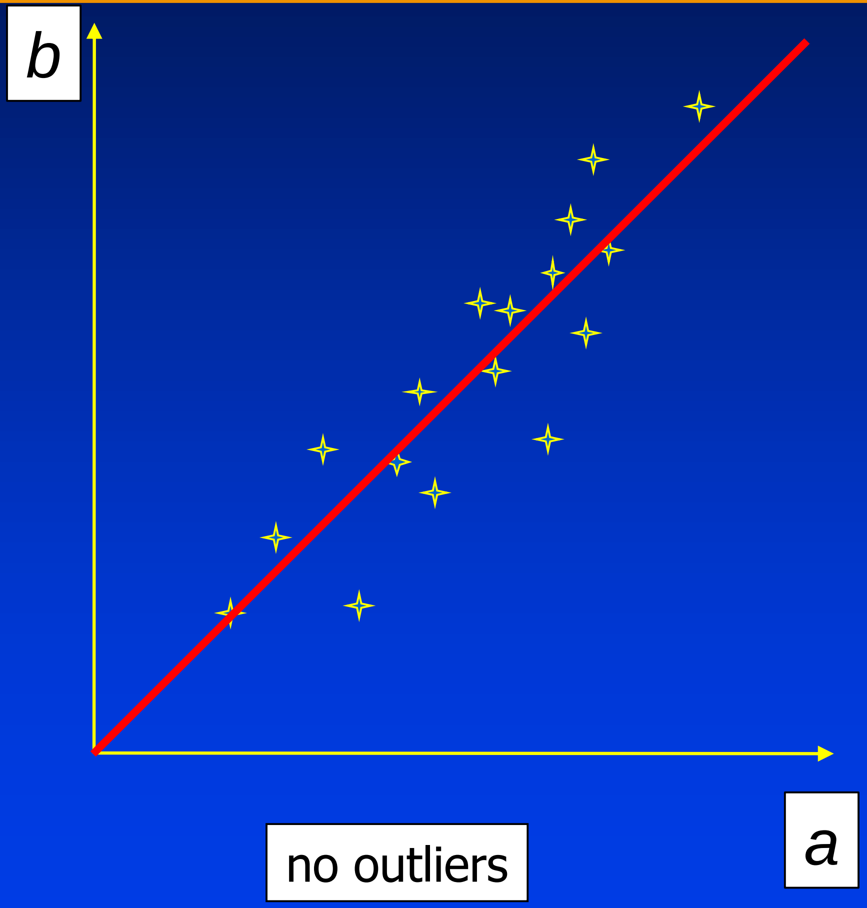
$2\mathbf{A}^T \mathbf{A}$ is positive semi-definite

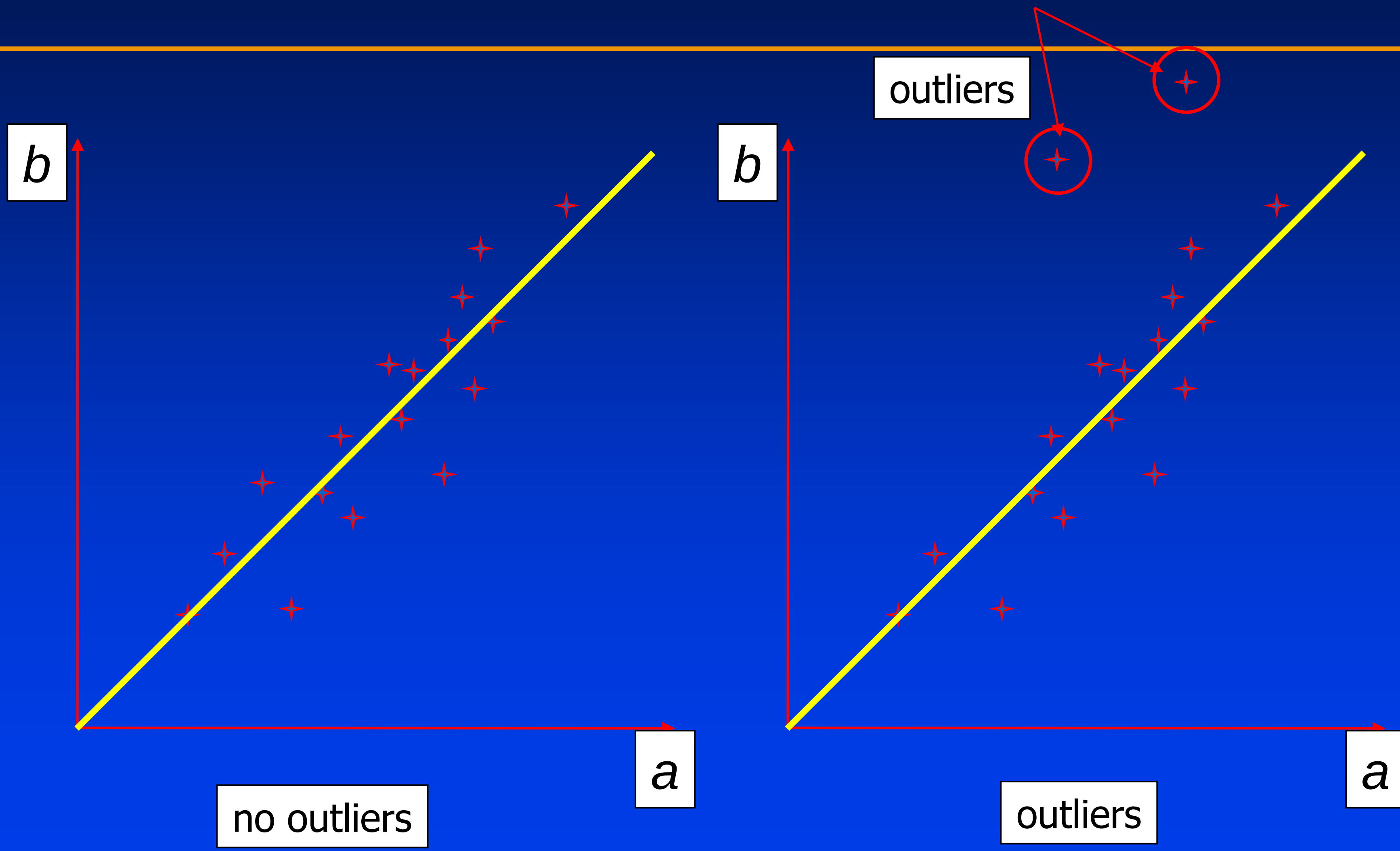
Always true


b
e_i
a
no errors in a_i


b
e_i
a
no errors in a_i
b
e_i
a
errors in a_i


b
a
homogeneous errors
b
a
non-homogeneous errors


b
a
no outliers


outliers
b
b
a
a
no outliers
outliers

Question

You should be able to prove that the equation above leads to the following expression for the best fit straight line:

$$y_p(x) = mx + b$$

$$m = \frac{\left(N \sum_i^N x_i y_i - \sum_i x_i \sum_i y_i\right)}{N \sum_i x_i^2 - \left(\sum_i x_i\right)^2}$$

$$b = \frac{\sum_i^N y_i - m \sum_i x_i}{N}$$

How good is the least-squares criteria?

- Optimality: the Gauss-Markov theorem

How good is the least-squares criteria?

- Optimality: the Gauss-Markov theorem

  Let $\{b_i\}$ and $\{x_j\}$ be two sets of random variables and define:

$$e_i = b_i - a_{i,1}x_1 - \ldots - a_{i,m}x_m$$

How good is the least-squares criteria?

- Optimality: the Gauss-Markov theorem

  Let $\{b_i\}$ and $\{x_j\}$ be two sets of random variables and define:

$$e_i = b_i - a_{i,1}x_1 - \ldots - a_{i,m}x_m$$

If

A1: $\{a_{i,j}\}$ are not random variables,

A2: $E(e_i) = 0$, for all i,

A3: $\text{var}(e_i) = \sigma$, for all i,

A4: $\text{cov}(e_i, e_j) = 0$, for all i and j,

How good is the least-squares criteria?

- Optimality: the Gauss-Markov theorem

Let $\{b_i\}$ and $\{x_j\}$ be two sets of random variables and define:

$$e_i = b_i - a_{i,1}x_1 - \ldots - a_{i,m}x_m$$

If

A1: $\{a_{i,j}\}$ are not random variables,

A2: $E(e_i) = 0$, for all i,

A3: $\mathrm{var}(e_i) = \sigma$, for all i,

A4: $\mathrm{cov}(e_i, e_j) = 0$, for all i and j,

Then $\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \sum e_i^2$ is the best unbiased linear estimator

Least Squares Interpolant

- We arrive at a system of equations through function minimization

$$2\mathbf{p}^T\mathbf{p}\mathbf{a} - 2\mathbf{p}^T\mathbf{y} = 0 \qquad \mathbf{a} = \left(\mathbf{p}^T\mathbf{p}\right)^{-1}\mathbf{p}^T\,\mathbf{y}^T$$

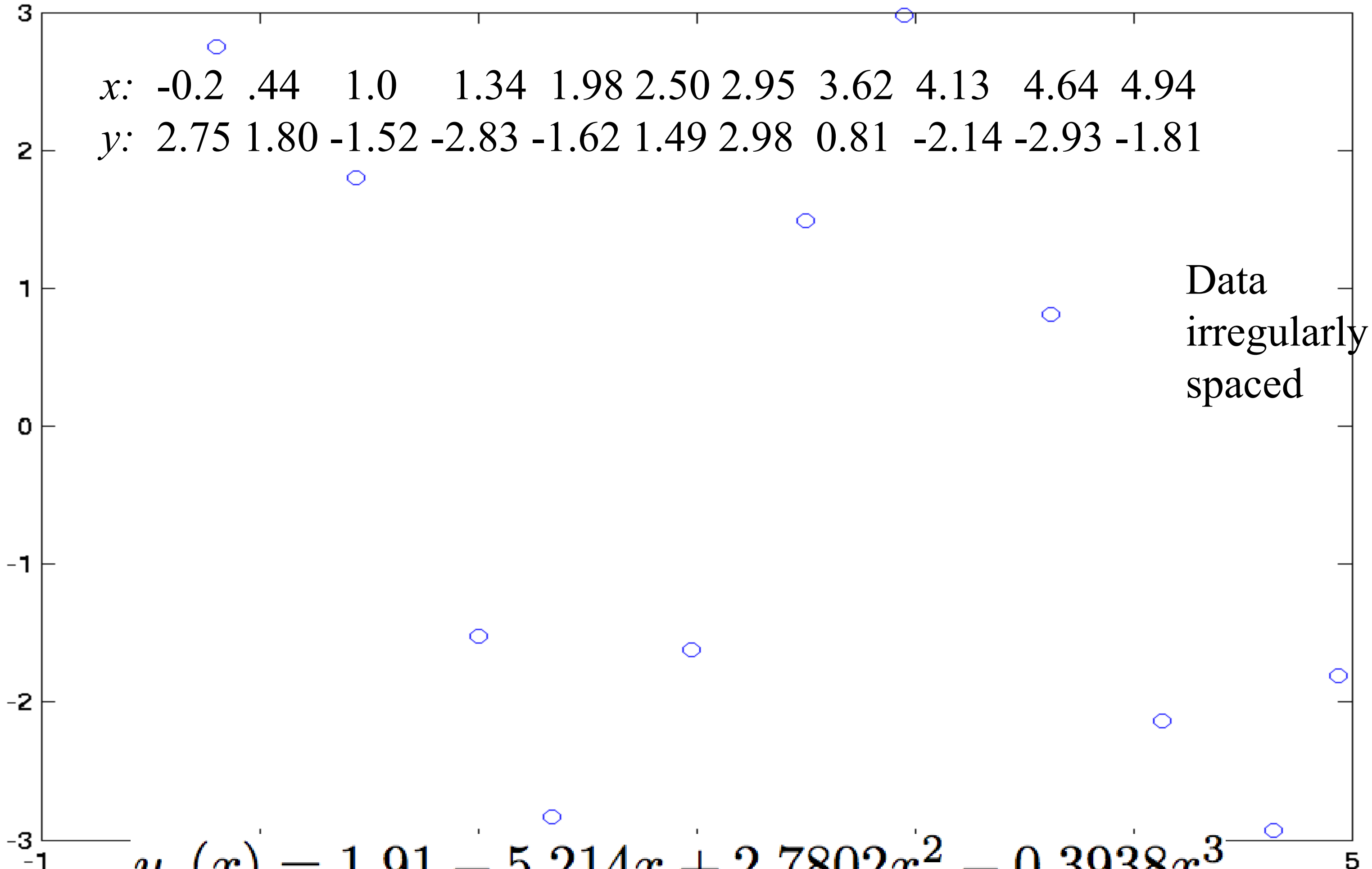
- We can introduce a pseudo-inverse

$$\mathbf{a} = \mathbf{p}^{+}\mathbf{y}^T$$

- For four points with a cubic polynomial

$$\mathbf{p} = \begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ 1 & x_4 & x_4^2 & x_4^3 \end{pmatrix}$$

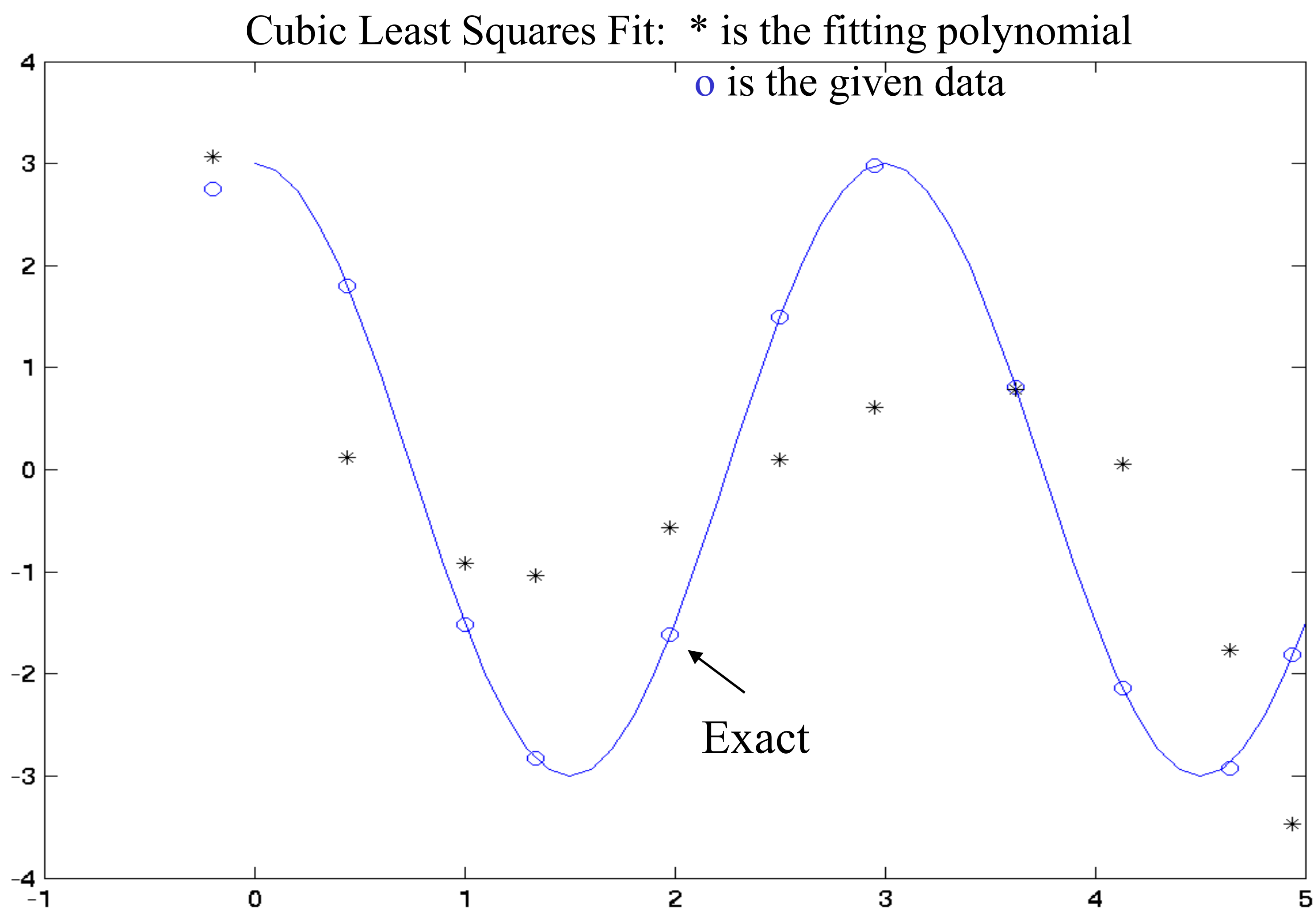
Cubic Least Squares Example



x:	-0.2	.44	1.0	1.34	1.98	2.50	2.95	3.62	4.13	4.64	4.94
y:	2.75	1.80	-1.52	-2.83	-1.62	1.49	2.98	0.81	-2.14	-2.93	-1.81

$$y_p(x) = 1.91 - 5.214x + 2.7802x^2 - 0.3938x^3$$

Least Squares Interpolant



Piecewise Interpolation

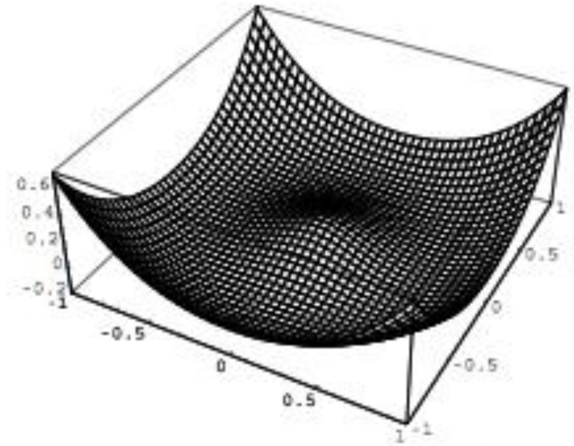
- Piecewise polynomials: a collection of polynomials to fit all the data points
- Different choices: linear, quadratic, cubic
- Non-polynomials: radial basis functions (RBFs)

Radial Basis Functions

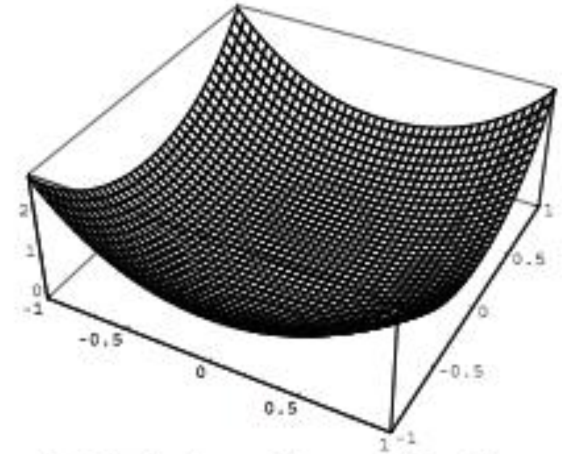
Developed to interpolate 2-D data: think bathymetry.

Given depths: $\mathbf{x}_i, i = 1, N$, interpolate to a rectangular grid.

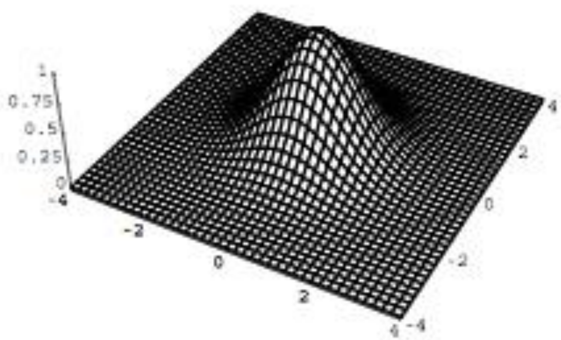
RBF



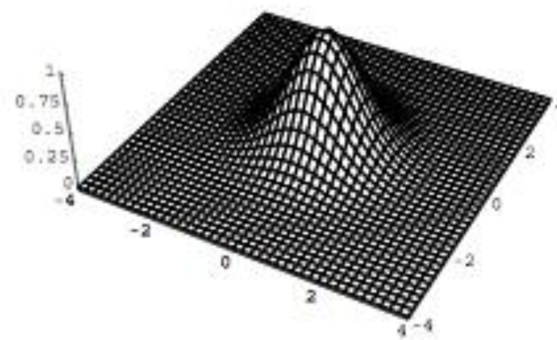
a) Thin-plate (2-d)
$\phi(r) = r^2 \log r$



b) Thin-plate (3-d)
$\phi(r) = r^3$



c) Gaussian
$\phi(r) = e^{r^2 \sigma^2}$



d) Compactly Supported
$\phi(r) = (1 - r)_+^4 (4r + 1)$

$r = |\mathbf{x} - \mathbf{x}_j|$

Radial Basis Functions

- Data points: $\mathbf{x}_i, i = 1, N$
- For each position, there is an associated value:

$$u_i, i = 1, N$$

- Radial basis function (located at each point):

$$g_j(\mathbf{x}) \equiv g(|\mathbf{x} - \mathbf{x}_j|), j = 1, N$$

$$u_p(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \, g_j(\mathbf{x})$$

Radial Basis Function for Data Fitting

- To find the unknown coefficients, we force the interpolant to go through all the data points:

$$\sum_{j=1}^{N} \alpha_j \, g_j(\mathbf{x}_i) = u_i, \quad i = 1, N$$

$$\mathbf{x}_i \equiv \left| \mathbf{x}_i - \mathbf{x}_j \right|$$

- We have n equations for the n unknown coefficients

Multiquadric RBF

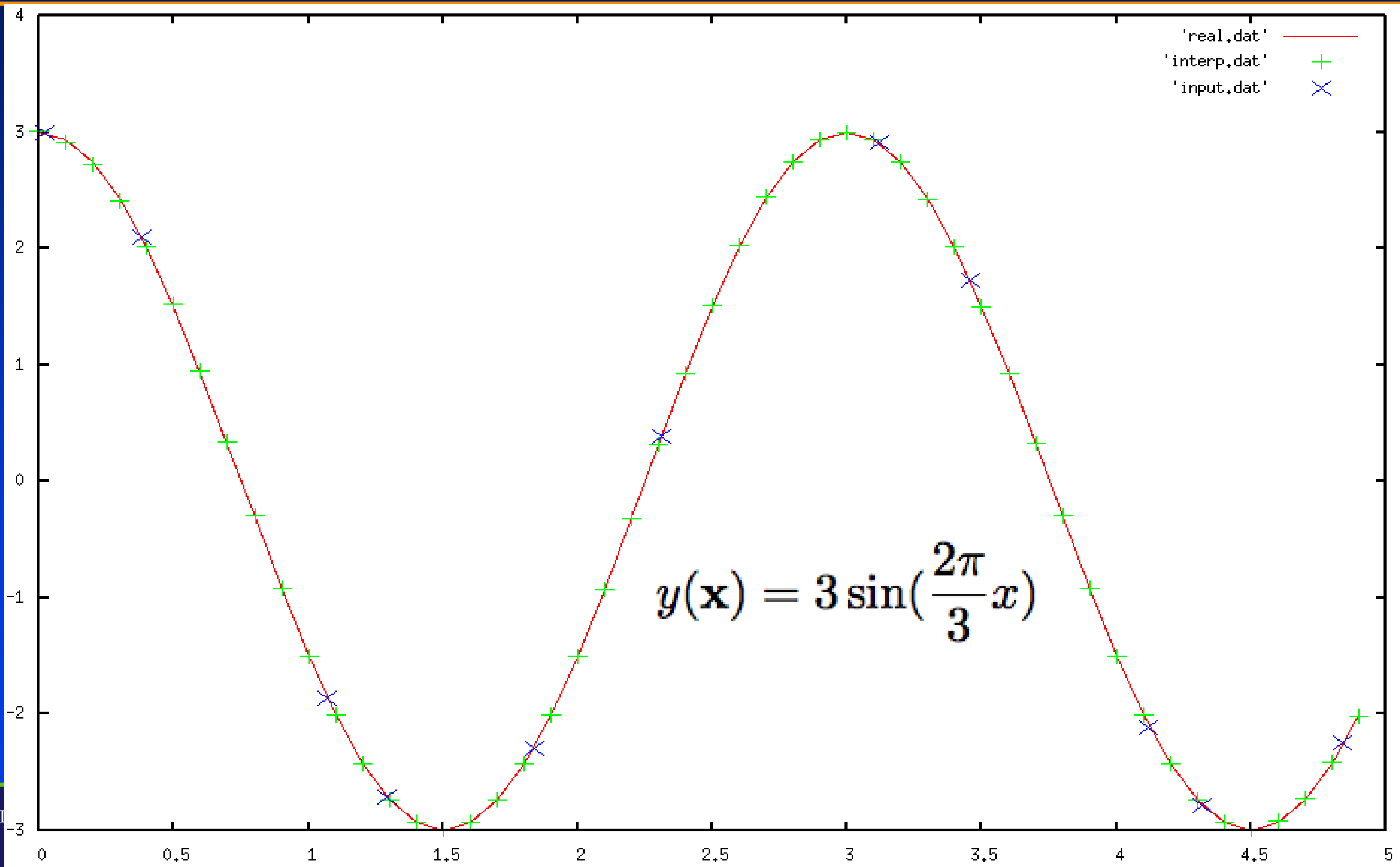
MQ: $$g_j(\mathbf{x}) = \sqrt{c_j^2 + r^2}$$

$$r = |\mathbf{x} - \mathbf{x}_j|$$

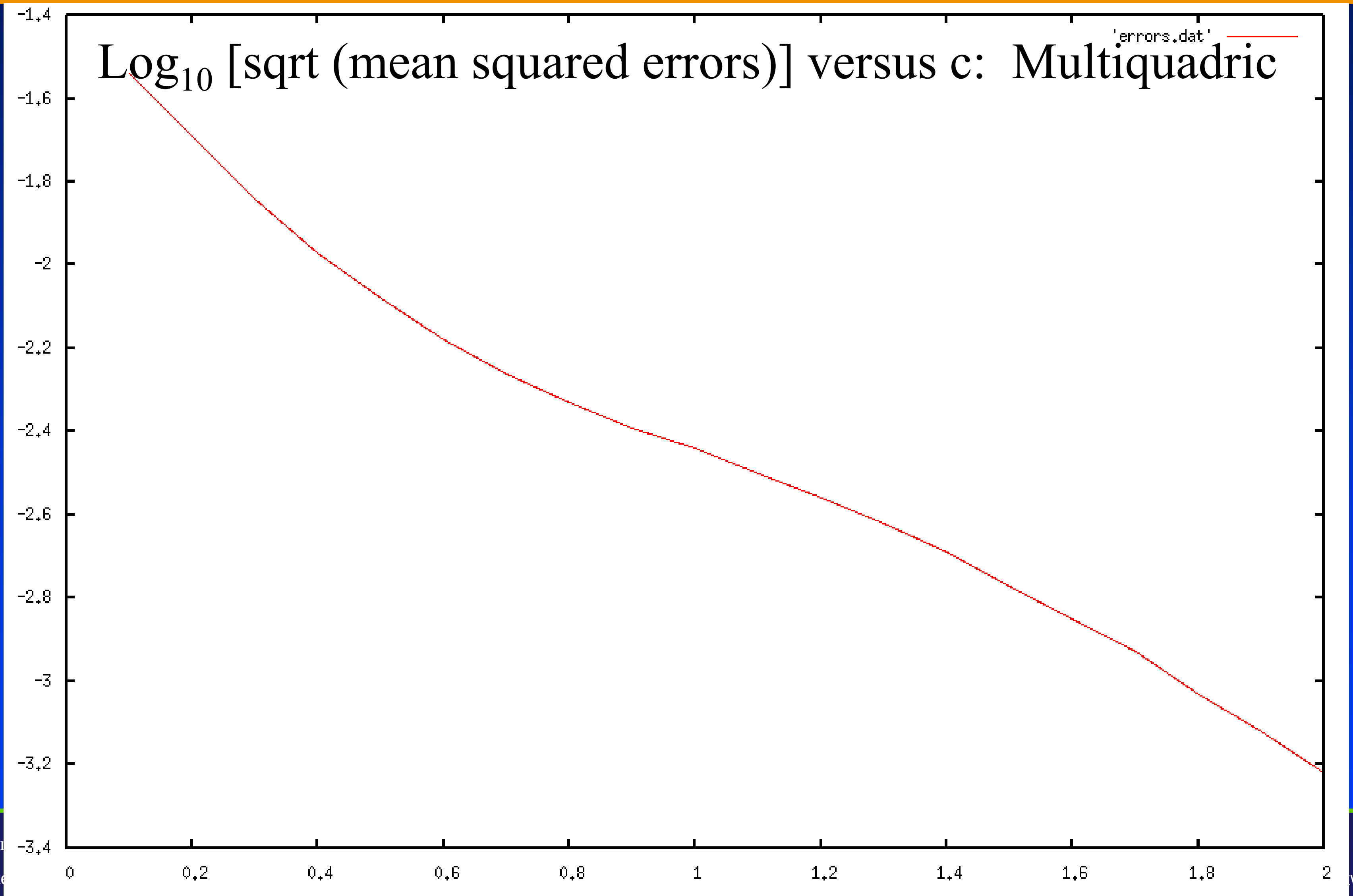
RMQ: $$g_j(\mathbf{x}) = \frac{1}{\sqrt{c_j^2 + r^2}}$$

Hardy, 1971; Kansa, 1990

11 (x,y) pairs: (0.2, 3.00), (0.38, 2.10), (1.07, -1.86), (1.29, -2.71), (1.84, -2.29), (2.31, 0.39), (3.12, 2.91), (3.46, 1.73), (4.12, -2.11), (4.32, -2.79), (4.84, -2.25) **SAME AS BEFORE**



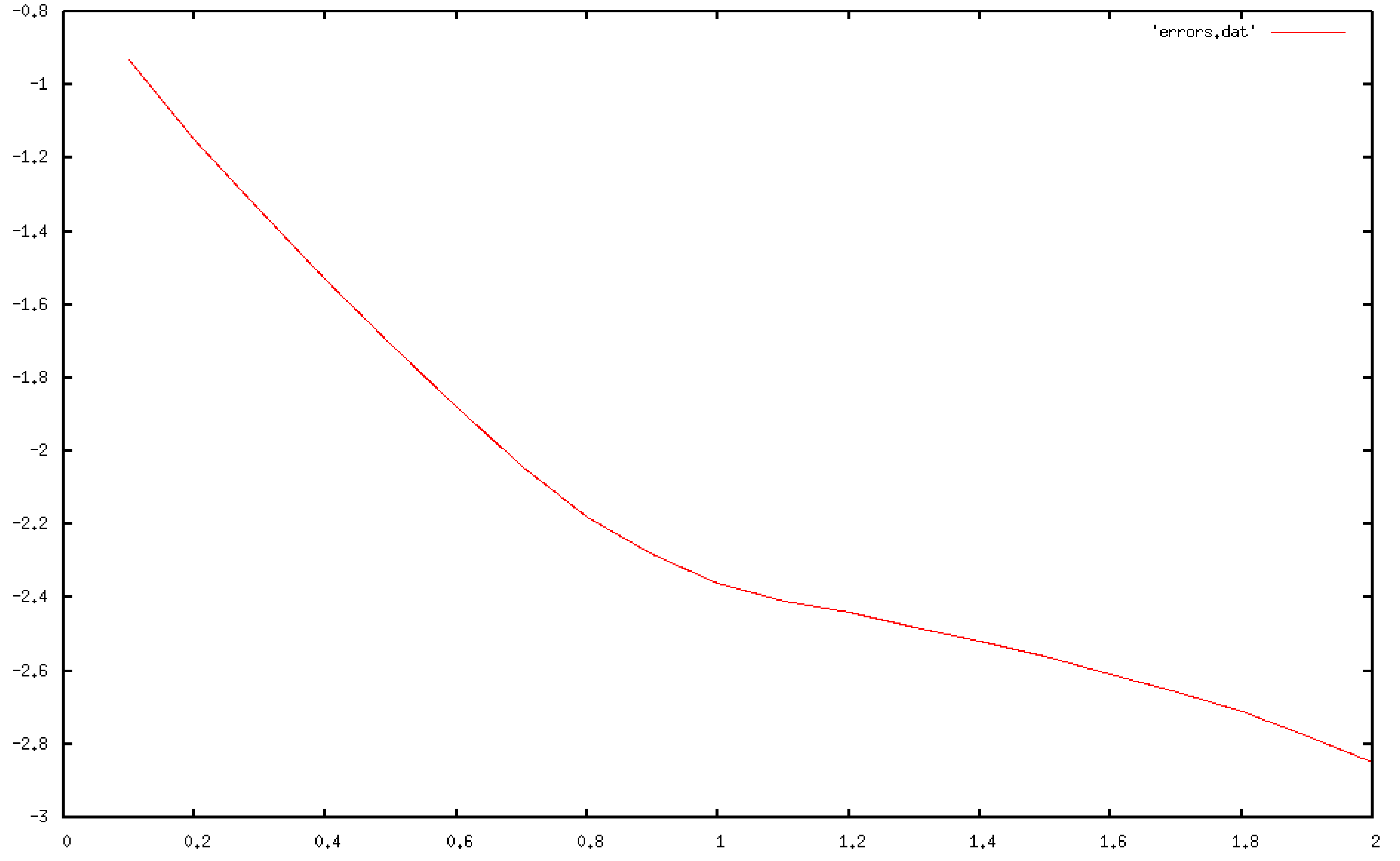
RBF Errors



Log10 [sqrt (mean squared errors)] versus c: Multiquadric
'errors.dat'
-1.4
-1.6
-1.8
-2
-2.2
-2.4
-2.6
-2.8
-3
-3.2
-3.4
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

RBF Errors

Log_{10} [sqrt (mean squared errors)] versus c: Reciprocal Multiquadric



Consistency (Property)

- Consistency is the ability of an interpolating function to reproduce a polynomial of a given order, the simplest consistency is constant consistency (reproduce unity)

$$\mathbf{x}_i \equiv \left|\mathbf{x}_i - \mathbf{x}_j\right|$$

$$\sum_{j=1}^{N} \alpha_j \, g_j(\mathbf{x}_i) = 1, \quad i = 1, N$$

If $g_j(0) = 1$, then a constraint results:

$$\sum_{j=1}^{N} \alpha_j = 1$$

Note: Not all RBFs have $g_j(0) = 1$

RBFs and PDEs

- Solve a boundary value problem: $\nabla^2 \phi(x, y) = 0$

$$\phi(x, y)\big|_{\text{on the boundary}} = f(x, y)$$

- We make use of RBFs as a possible solution

$$\phi_h(\mathbf{x}) = \sum_{j=1,N} \alpha_j \, g_j(\mathbf{x})$$

RBFs and PDEs

- The governing equation and boundary conditions

$$\phi_h(\mathbf{x}) = \sum_{j=1,N} \alpha_j \, g_j(\mathbf{x})$$

$$\sum_{j=1}^{N} \alpha_j \nabla^2 g_j(x_i) = 0 \quad \text{for all the interior points}$$

$$\sum_{j=1}^{N} \alpha_j g_j(x_i) = f_i \quad \text{for the boundary points}$$

These are N equations for the N unknown constants, α_j

RBFs and PDEs

- One common problem with many RBFs is that the n * n matrix is dense, one easy-fix is to use a RBF with compact support (matrix becomes sparse)

1D:

$$\begin{cases} (1-r/h)^3(3r/h+1) & \text{for } |r|<h \\ 0, & \text{otherwise} \end{cases}$$

3D:

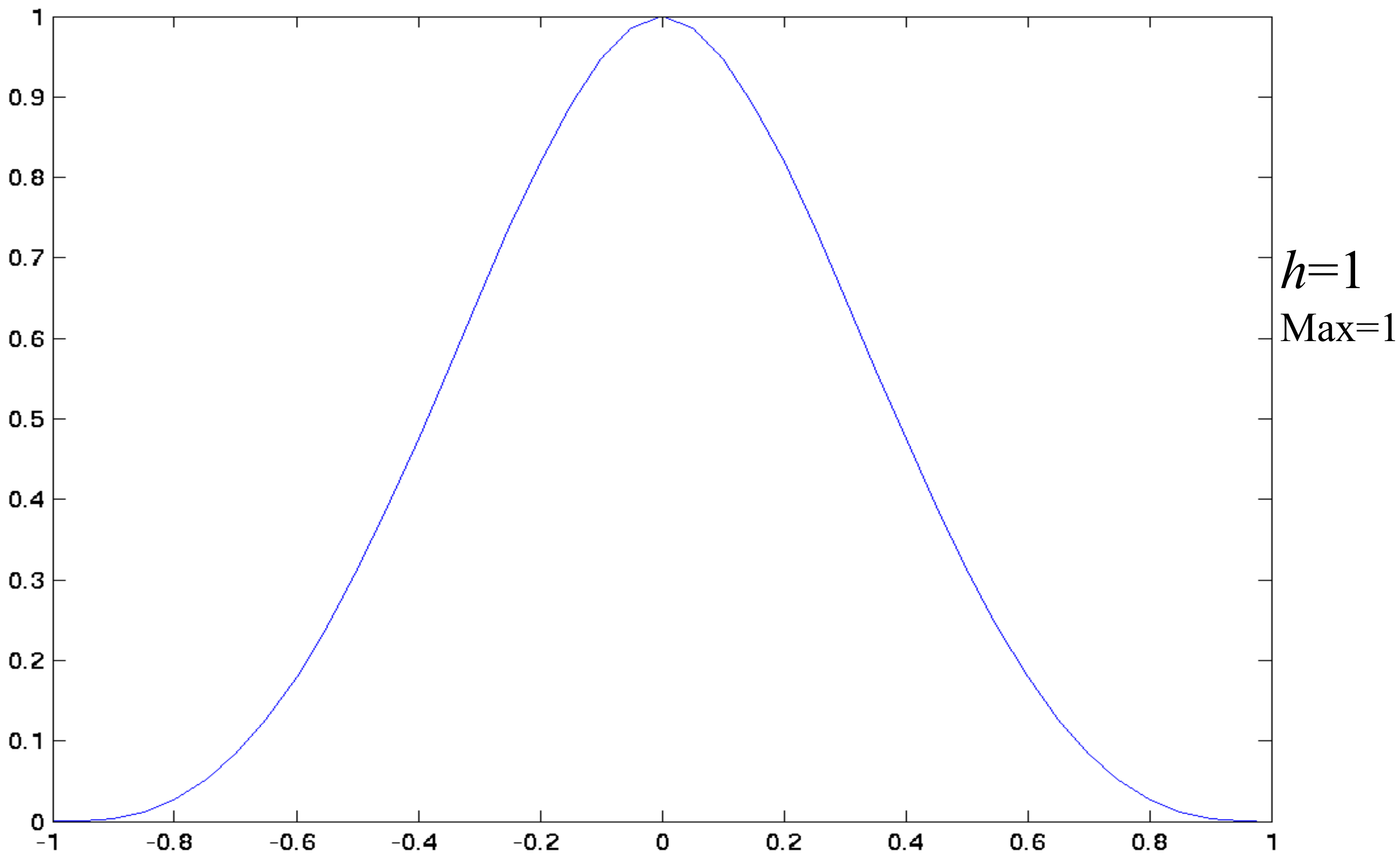
$$\begin{cases} (1-r/h)^4(4r/h+1) & \text{for } |r|<h \\ 0, & \text{otherwise} \end{cases}$$

$$(1-r/h)_+^4(4r/h+1)$$

RBFs with small 'footprints' (Wendland, 2005)

Advantages: matrix is sparse, but still $n * n$

Wendland 1-D RBF with Compact Support



Moving Least Squares Interpolant

$$u_p(\mathbf{x}) = \sum_{j}^{N} a_j(\mathbf{x}) p_j(\mathbf{x}) \equiv \mathbf{p}^T(\mathbf{x})\,\mathbf{a}(\mathbf{x})$$

$$p^T(\mathbf{x})$$

are monomials in x for 1D $(1, x, x^2, x^3)$
x,y in 2D, e.g. $(1, x, y, x^2, xy, y^2 \ldots)$

Note a_j are functions of $\boldsymbol{x}$

Moving Least Squares Interpolant

$$E(\mathbf{x}) = \sum_{i=1}^{N} W(\mathbf{x} - \mathbf{x}_i)\left(\mathbf{p}^T(\mathbf{x}_i)\,\mathbf{a}(\mathbf{x}) - u_i\right)^2$$

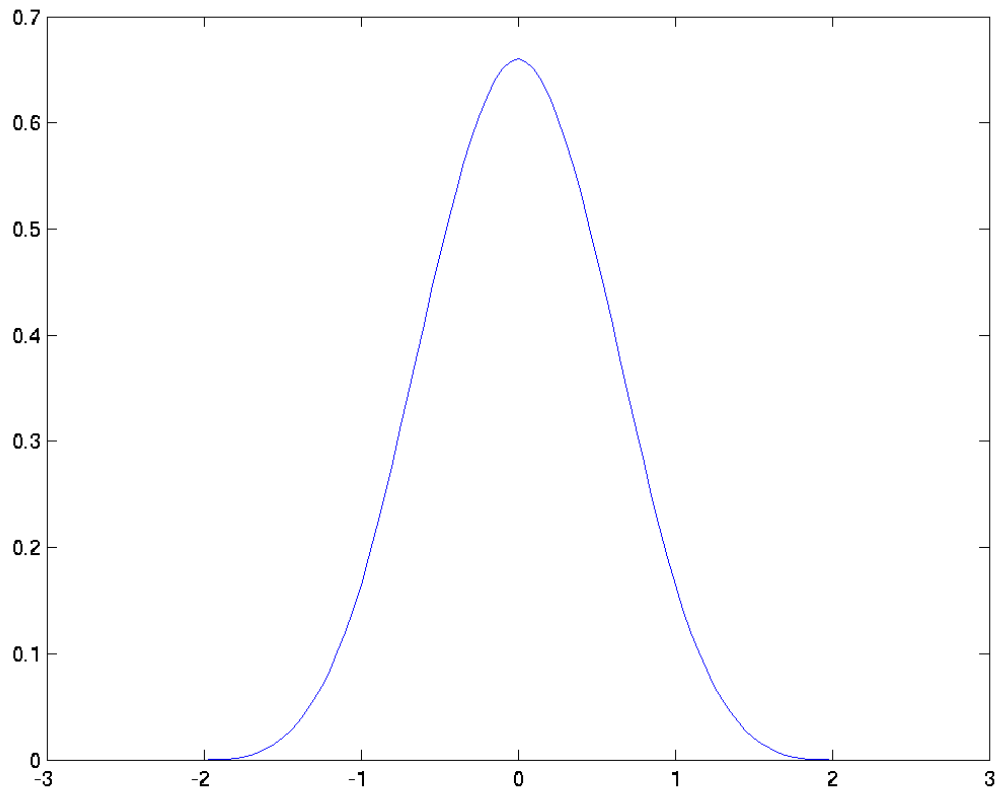
We define a weighted mean-squared error

where $W(\boldsymbol{x}-\boldsymbol{x}_i)$ is a weighting function that decays with increasing $\boldsymbol{x}-\boldsymbol{x}_i$.

Same as previous least squares approach, except for $W(\boldsymbol{x}-\boldsymbol{x}_i)$

Weighting Function

$$W(q) = \frac{2}{3h} \begin{cases} 1 - \frac{3}{2}q^2 + \frac{3}{4}q^3, & \text{for } q \leq 1 \\ \frac{1}{4}(2-q)^3, & \text{for } 1 \leq q \leq 2 \\ 0, & \text{for } q > 2 \end{cases}$$



$q = x/h$

Moving Least Squares Interpolant

Minimizing the weighted squared errors for the coefficients:

$$\frac{\partial E}{\partial \mathbf{a}} = \mathbf{A}(\mathbf{x})\mathbf{a}(\mathbf{x}) - \mathbf{B}(\mathbf{x})\mathbf{u} = 0$$

where $\mathbf{u}^T = (u_1, u_2, ...u_n)$ $\mathbf{A} = \mathbf{P}^T\mathbf{W}(\mathbf{x})\mathbf{P}$, $\mathbf{B} = \mathbf{P}^T\mathbf{W}(\mathbf{x})$

$$\mathbf{P} = \begin{bmatrix} p_1(\mathbf{x}_1) & p_2(\mathbf{x}_1) & \dots & p_m(\mathbf{x}_1) \\ p_1(\mathbf{x}_2) & p_2(\mathbf{x}_2) & \dots & p_m(\mathbf{x}_2) \\ \dots & \dots & \dots & \dots \\ p_1(\mathbf{x}_n) & p_2(\mathbf{x}_n) & \dots & p_m(\mathbf{x}_n) \end{bmatrix}$$

$$\mathbf{W}(\mathbf{x}) = \begin{bmatrix} W(\mathbf{x}-\mathbf{x}_1) & 0 & \dots & 0 \\ 0 & W(\mathbf{x}-\mathbf{x}_2) & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & W(\mathbf{x}-\mathbf{x}_n) \end{bmatrix}$$

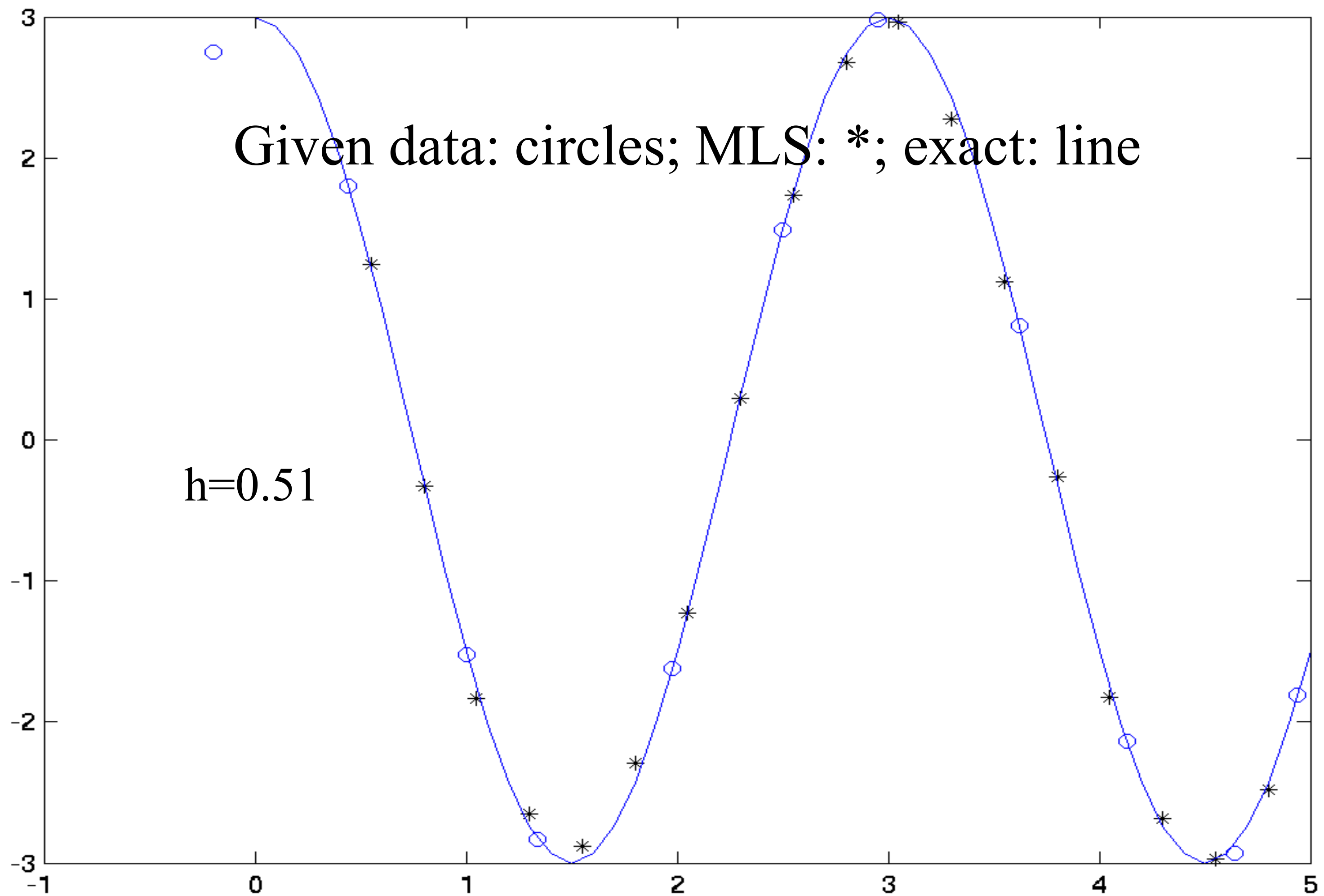
Moving Least Squares Interpolant

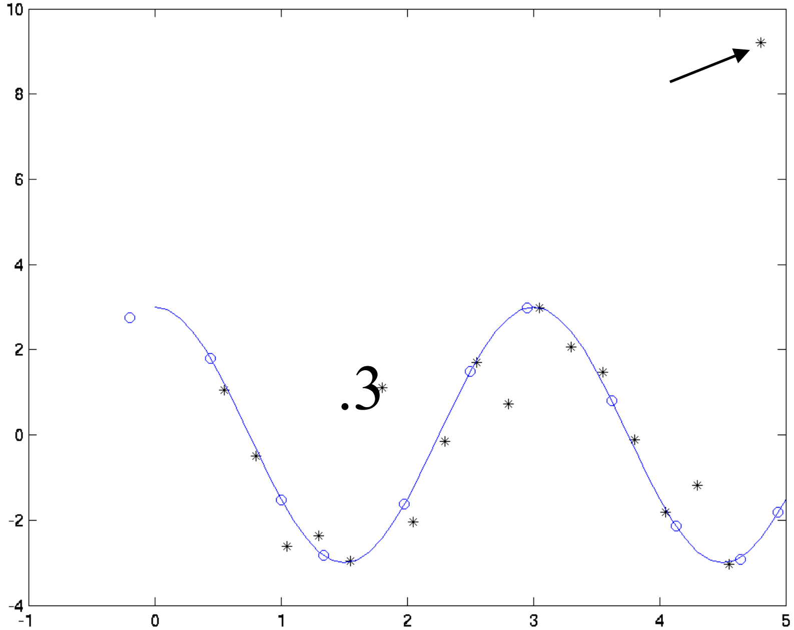
$$\mathbf{a}(\mathbf{x}) = \mathbf{A}^{-1}(\mathbf{x})\,\mathbf{B}(\mathbf{x})\,\mathbf{u}$$

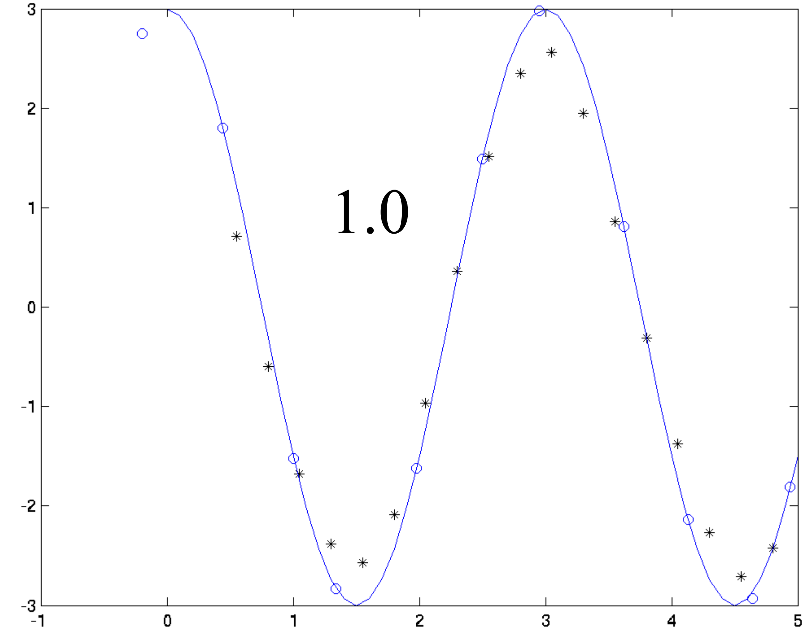
The final locally valid interpolant is:

$$u_p(\mathbf{x}) = \sum_{j}^{N} a_j(\mathbf{x}) p_j(\mathbf{x}) \equiv \mathbf{p}^T(\mathbf{x})\,\mathbf{a}(\mathbf{x})$$

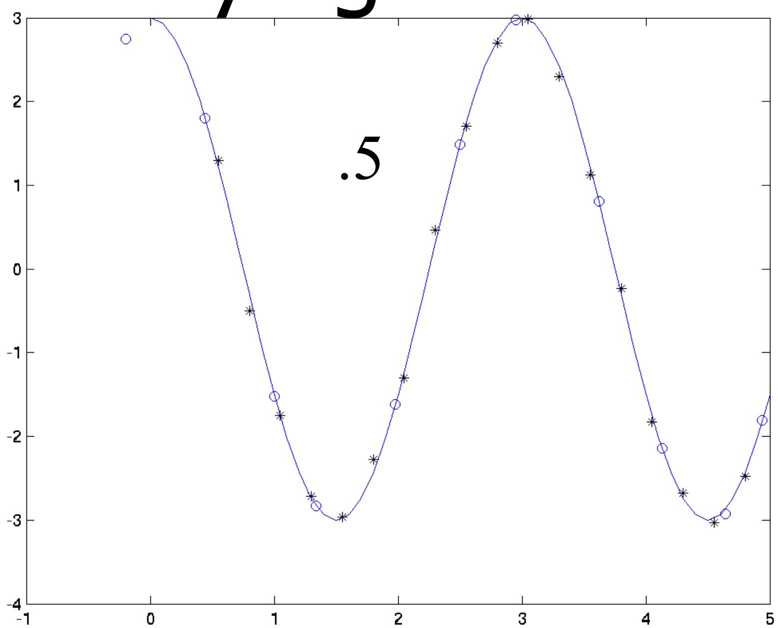
MLS Fit to (Same) Irregular Data

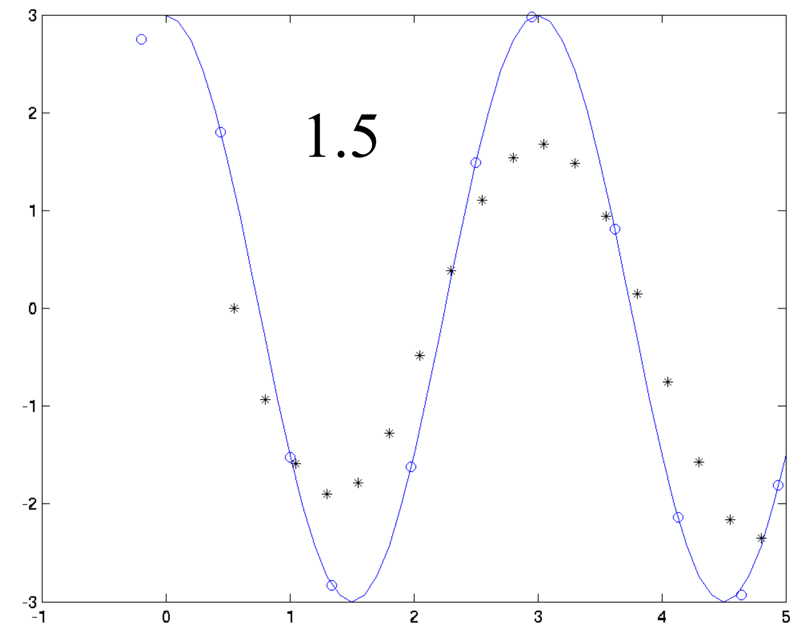






Varying h Values





Conclusion

There are a variety of interpolation techniques for irregularly spaced data:

- Polynomial fits
- Best fit polynomials
- Piecewise polynomials
- Radial basis functions
- Moving least squares