

System of linear equations:

$$a_{00}x_0 + a_{01}x_1 + a_{02}x_2 + \cdots + a_{0,N-1}x_{N-1} = b_0$$

$$a_{10}x_0 + a_{11}x_1 + a_{12}x_2 + \cdots + a_{1,N-1}x_{N-1} = b_1$$

$$a_{20}x_0 + a_{21}x_1 + a_{22}x_2 + \cdots + a_{2,N-1}x_{N-1} = b_2$$

...

...

$$a_{M-1,0}x_0 + a_{M-1,1}x_1 + \cdots + a_{M-1,N-1}x_{N-1} = b_{M-1}$$

Here the N unknowns x_j , $j = 0, 1, \dots, N - 1$ are related by M equations. The coefficients a_{ij} with $i = 0, 1, \dots, M - 1$ and $j = 0, 1, \dots, N - 1$ are known numbers, as are the *right-hand side* quantities b_i , $i = 0, 1, \dots, M - 1$.

Matrix-vector notation:

$$a_{00}x_0 + a_{01}x_1 + a_{02}x_2 + \cdots + a_{0,N-1}x_{N-1} = b_0$$

$$a_{10}x_0 + a_{11}x_1 + a_{12}x_2 + \cdots + a_{1,N-1}x_{N-1} = b_1$$

$$a_{20}x_0 + a_{21}x_1 + a_{22}x_2 + \cdots + a_{2,N-1}x_{N-1} = b_2$$

...

...

$$a_{M-1,0}x_0 + a_{M-1,1}x_1 + \cdots + a_{M-1,N-1}x_{N-1} = b_{M-1}$$

$$\mathbf{A} = \begin{bmatrix} a_{00} & a_{01} & \cdots & a_{0,N-1} \\ a_{10} & a_{11} & \cdots & a_{1,N-1} \\ \cdots & \cdots & \cdots & \cdots \\ a_{M-1,0} & a_{M-1,1} & \cdots & a_{M-1,N-1} \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ \cdots \\ b_{M-1} \end{bmatrix}$$

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$$

We start with: $N = M$

LU decomposition:

Suppose we are able to write the matrix \mathbf{A} as a product of two matrices,

$$\mathbf{L} \cdot \mathbf{U} = \mathbf{A} \quad (2.3.1)$$

where \mathbf{L} is *lower triangular* (has elements only on the diagonal and below) and \mathbf{U} is *upper triangular* (has elements only on the diagonal and above). For the case of a 4×4 matrix \mathbf{A} , for example, equation (2.3.1) would look like this:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ \alpha_{10} & 1 & 0 & 0 \\ \alpha_{20} & \alpha_{21} & 1 & 0 \\ \alpha_{30} & \alpha_{31} & \alpha_{32} & 1 \end{bmatrix} \cdot \begin{bmatrix} \beta_{00} & \beta_{01} & \beta_{02} & \beta_{03} \\ 0 & \beta_{11} & \beta_{12} & \beta_{13} \\ 0 & 0 & \beta_{22} & \beta_{23} \\ 0 & 0 & 0 & \beta_{33} \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$\mathbf{A} \cdot \mathbf{x} = (\mathbf{L} \cdot \mathbf{U}) \cdot \mathbf{x} = \mathbf{L} \cdot (\mathbf{U} \cdot \mathbf{x}) = \mathbf{b}$$

$$\mathbf{L} \cdot \mathbf{y} = \mathbf{b}$$

$$\mathbf{U} \cdot \mathbf{x} = \mathbf{y}$$

LU decomposition including complexity ($O()$)

LU decomposition:

For each $j = 0, 1, 2, \dots, N - 1$ do these two procedures:

$$\text{for } i = 0, 1, \dots, j \quad \beta_{ij} = a_{ij} - \sum_{k=0}^{i-1} \alpha_{ik} \beta_{kj} \quad O(N^3)$$

$$\text{for } i = j + 1, j + 2, \dots, N - 1 \quad \alpha_{ij} = \frac{1}{\beta_{jj}} \left(a_{ij} - \sum_{k=0}^{j-1} \alpha_{ik} \beta_{kj} \right)$$

Forward substitution:

$$\mathbf{L} \cdot \mathbf{y} = \mathbf{b}$$

$$y_0 = \frac{b_0}{\cancel{\alpha_{00}}} \\ y_i = \frac{1}{\cancel{\alpha_{ii}}} \left[b_i - \sum_{j=0}^{i-1} \alpha_{ij} y_j \right] \quad i = 1, 2, \dots, N - 1 \quad O(N^2)$$

Back substitution:

$$\mathbf{U} \cdot \mathbf{x} = \mathbf{y}$$

$$x_{N-1} = \frac{y_{N-1}}{\beta_{N-1,N-1}} \\ x_i = \frac{1}{\beta_{ii}} \left[y_i - \sum_{j=i+1}^{N-1} \beta_{ij} x_j \right] \quad i = N - 2, N - 3, \dots, 0 \quad O(N^2)$$

Gaussian elimination is also $O(N^3)$

Advantage of LU decomposition when comparing to Gaussian elimination ?

Assume we have to solve a system of linear equations with the **same matrix** and a **new right hand side**.

Then we only need to do forward and back substitutions (assuming we stored L and U). Hence $O(N^2)$ rather than $O(N^3)$.

Application: Linear least squares problems:

N data points $(x_i, y_i), i = 0, \dots, N - 1$

Model:
$$y(x) = \sum_{k=0}^{M-1} a_k X_k(x)$$

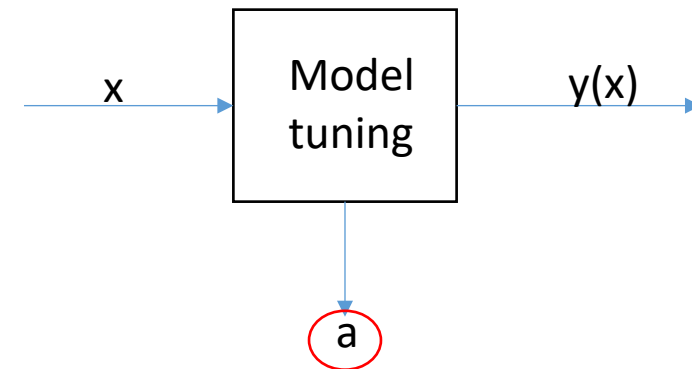
$X_0(x), \dots, X_{M-1}(x)$ are arbitrary fixed functions of x , called the *basis functions*.

M adjustable parameters $a_j, j = 0, \dots, M - 1$

Input-output relation



Model calibration



Example:

Data generated by introducing
fluctuations to ground truth parabola:

$$a_0 = 1; \quad a_1 = 1; \quad a_2 = 0.5$$

Example:

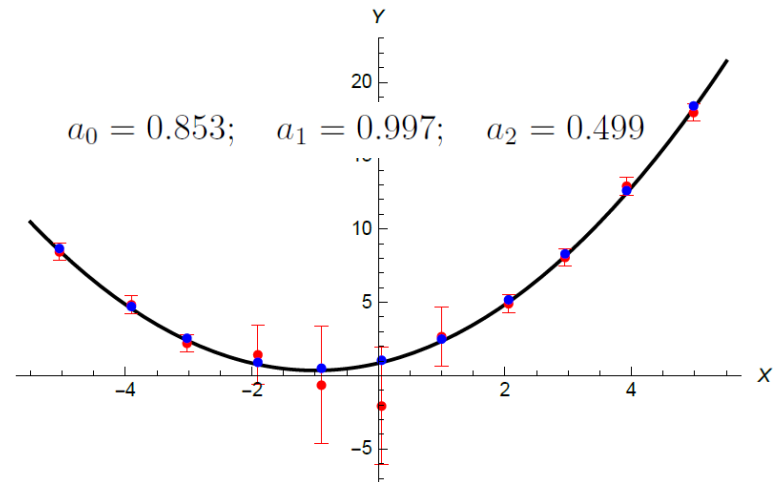
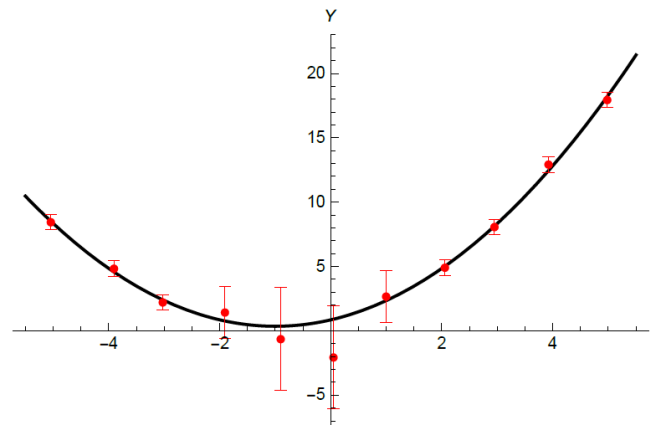
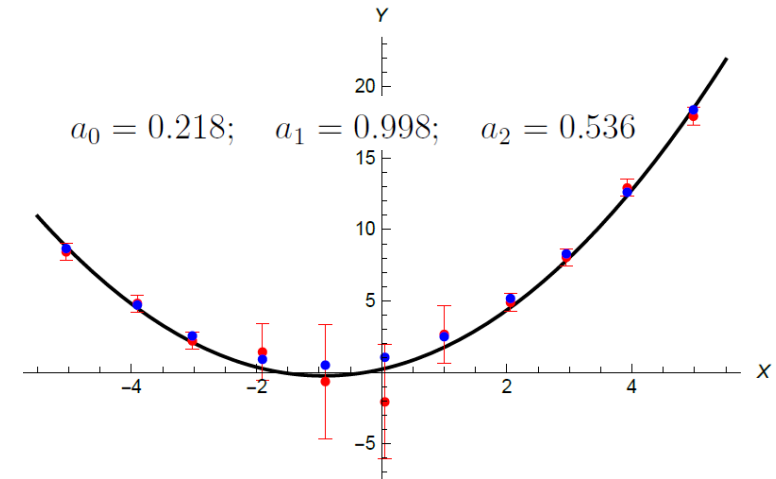
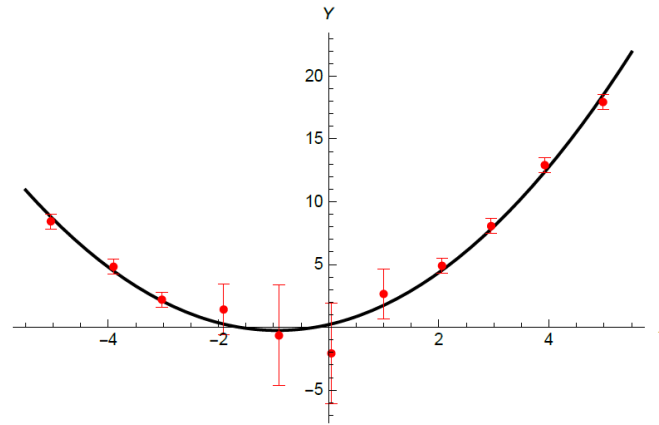
Fitting a parabola to a set of data points

$$y(x) = a_0 + a_1x + a_2x^2$$

Btw: Data points must have accuracy info.

$$\min \sum_{i=0}^{N-1} [a_0 + a_1x_i + a_2x_i^2 - y_i]^2$$

$$\min \sum_{i=0}^{N-1} \left[\frac{a_0 + a_1x_i + a_2x_i^2 - y_i}{\sigma_i} \right]^2$$



$$N \text{ data points } (x_i, y_i), i = 0, \dots, N-1 \quad \text{Model:} \quad y(x) = \sum_{k=0}^{M-1} a_k X_k(x)$$

From statistics, we hence know that we must **minimize**:

$$\chi^2 = \sum_{i=0}^{N-1} \left[\frac{y_i - \sum_{k=0}^{M-1} a_k X_k(x_i)}{\sigma_i} \right]^2$$

σ_i is the measurement error (standard deviation) of the i th data point

First order conditions are

$$\frac{\partial \chi^2}{\partial a_k} = 0 \quad k = 0, \dots, M-1$$

from which we get

$$0 = \sum_{i=0}^{N-1} \frac{1}{\sigma_i^2} \left[y_i - \sum_{j=0}^{M-1} a_j X_j(x_i) \right] X_k(x_i) \quad k = 0, \dots, M-1 \quad (15.4.6)$$

$$0 = \sum_{i=0}^{N-1} \frac{1}{\sigma_i^2} \left[y_i - \sum_{j=0}^{M-1} a_j X_j(x_i) \right] X_k(x_i) \quad k = 0, \dots, M-1 \quad (15.4.6)$$

Interchanging the order of summations, we can write (15.4.6) as the matrix equation

$$\sum_{j=0}^{M-1} \alpha_{kj} a_j = \beta_k \quad (15.4.7)$$

$$\text{where } \alpha_{kj} = \sum_{i=0}^{N-1} \frac{X_j(x_i) X_k(x_i)}{\sigma_i^2} \quad \text{and} \quad \beta_k = \sum_{i=0}^{N-1} \frac{y_i X_k(x_i)}{\sigma_i^2}$$

We can now formulate this with a "design matrix" \mathbf{A} and a right hand side \mathbf{b}

$$A_{ij} = \frac{X_j(x_i)}{\sigma_i} \quad b_i = \frac{y_i}{\sigma_i} \quad \mathbf{a} \equiv (a_0, \dots, a_{M-1}) \quad (\mathbf{A}^T \cdot \mathbf{A}) \cdot \mathbf{a} = \mathbf{A}^T \cdot \mathbf{b}$$

$$\alpha_{kj} = \sum_{i=0}^{N-1} [A^T]_{ki} [A]_{ij} \quad \beta_k = \sum_{i=0}^{N-1} [A^T]_{ki} b_i$$

"Normal equations"

Design matrix A (and right hand side b):

$$A_{ij} = \frac{X_j(x_i)}{\sigma_i}$$

$$b_i = \frac{y_i}{\sigma_i}$$

$$\begin{array}{c}
 \begin{array}{cccc}
 X_0(\quad) & X_1(\quad) & \dots & X_{M-1}(\quad)
 \end{array} \\
 \begin{array}{c} \longleftarrow \text{Basis functions} \longrightarrow \end{array} \\
 \begin{array}{c}
 \begin{array}{c}
 \uparrow x_0 \\
 \vdots \\
 x_1 \\
 \vdots \\
 x_{N-1} \\
 \downarrow
 \end{array}
 \begin{array}{c}
 \left(\begin{array}{cccc}
 \frac{X_0(x_0)}{\sigma_0} & \frac{X_1(x_0)}{\sigma_0} & \dots & \frac{X_{M-1}(x_0)}{\sigma_0} \\
 \frac{X_0(x_1)}{\sigma_1} & \frac{X_1(x_1)}{\sigma_1} & \dots & \frac{X_{M-1}(x_1)}{\sigma_1} \\
 \vdots & \vdots & & \vdots \\
 \vdots & \vdots & & \vdots \\
 \vdots & \vdots & & \vdots \\
 \frac{X_0(x_{N-1})}{\sigma_{N-1}} & \frac{X_1(x_{N-1})}{\sigma_{N-1}} & \dots & \frac{X_{M-1}(x_{N-1})}{\sigma_{N-1}}
 \end{array} \right)
 \end{array}
 \end{array}
 \end{array}$$

data points

Algorithm for Linear Least Square Problems (Normal Equations):

N data points $(x_i, y_i), i = 0, \dots, N - 1$

Model:
$$y(x) = \sum_{k=0}^{M-1} a_k X_k(x)$$

$$A_{ij} = \frac{X_j(x_i)}{\sigma_i} \qquad b_i = \frac{y_i}{\sigma_i} \qquad \mathbf{a} \equiv (a_0, \dots, a_{M-1})$$

$$\mathbf{C} \equiv \mathbf{A}^T \cdot \mathbf{A}$$

$$\mathbf{c} \equiv \mathbf{A}^T \cdot \mathbf{b}$$

Solve $\mathbf{C} \cdot \mathbf{a} = \mathbf{c}$

Symmetric and positive (semi-)definite matrices

An $N \times N$ matrix \mathbf{B} is called *symmetric* if $B_{ij} = B_{ji}$ for all i, j .

A symmetric $N \times N$ matrix \mathbf{B} is called *positive definite* if $\mathbf{v} \cdot \mathbf{B} \cdot \mathbf{v} > 0$ for all **nonzero** vectors $\mathbf{v} \in \mathbb{R}^N$. (It is called *positive semi-definite* if $\mathbf{v} \cdot \mathbf{B} \cdot \mathbf{v} \geq 0$ for all **nonzero** vectors $\mathbf{v} \in \mathbb{R}^N$).

Consider now the matrix $\mathbf{C} = \mathbf{A}^T \mathbf{A}$ from the Normal equations.
We now get that

$$C_{kj} = \sum_{i=0}^{M-1} [A^T]_{ki} [A]_{ij} = \sum_{i=0}^{M-1} [A]_{ik} [A]_{ij}$$

$$\begin{aligned} \mathbf{v} \cdot \mathbf{C} \cdot \mathbf{v} &= \sum_{k=0}^{N-1} v_k \left[\sum_{j=0}^{N-1} C_{kj} v_j \right] \\ &= \sum_{k=0}^{N-1} v_k \left[\sum_{j=0}^{N-1} \left(\sum_{i=0}^{M-1} [A]_{ik} [A]_{ij} \right) v_j \right] \\ &= \sum_{i=0}^{M-1} \left[\sum_{k=0}^{N-1} A_{ik} v_k \right] \left[\sum_{j=0}^{N-1} A_{ij} v_j \right] \\ &= \sum_{i=0}^{M-1} \left[\sum_{k=0}^{N-1} A_{ik} v_k \right]^2 \geq 0 \end{aligned}$$

Hence \mathbf{C} is at least positive semi-definite. Mostly, it will be positive definite, and you may assume this to begin with.

Cholesky decomposition (for symmetric and pos. def. matrices):

Assume that \mathbf{A} is symmetric and positive definite. We now first write the LU decomposition of A :

$$\mathbf{A} = \mathbf{L}^* \mathbf{U}^* \equiv \mathbf{L}^* \mathbf{D} \mathbf{D} \mathbf{V}$$

where $\mathbf{D} = \text{diag}\{\sqrt{U_{ii}^*}\}$ contains the square roots of the diagonal elements of \mathbf{U}^* (when \mathbf{A} is positive definite, it can be shown (we skip that) that all diagonal elements in \mathbf{U}^* are positive). Hence, \mathbf{V} is an upper triangular matrix with $V_{ii} = 1$ for all i .

Since the LU -matrices are unique $\mathbf{L}^*, \mathbf{D}, \mathbf{V}$ are also unique. We now use that \mathbf{A} is symmetric to obtain

$$\mathbf{A} = \mathbf{A}^T = \mathbf{V}^T \mathbf{D} \mathbf{D} (\mathbf{L}^*)^T$$

As $\mathbf{L}^*, \mathbf{D}, \mathbf{V}$ are unique, we get $\mathbf{V} = (\mathbf{L}^*)^T$ and hence

$$\mathbf{A} = \mathbf{L}^* \mathbf{D} \mathbf{D} (\mathbf{L}^*)^T$$

Hence, we can uniquely write

$$\mathbf{A} = \mathbf{L}^* \mathbf{D} (\mathbf{L}^* \mathbf{D})^T \equiv \mathbf{L} \mathbf{L}^T$$

where $\mathbf{L} = \mathbf{L}^* \mathbf{D}$ is lower triangular, but *no longer* with $L_{ii} = 1$

Cholesky decomposition (for symmetric and pos. def. matrices):

Similar to LU decomposition:

$$\mathbf{L} \cdot \mathbf{L}^T = \mathbf{A}$$

For $i=0$ to $N-1$

$$L_{ii} = \left(a_{ii} - \sum_{k=0}^{i-1} L_{ik}^2 \right)^{1/2}$$

$$L_{ji} = \frac{1}{L_{ii}} \left(a_{ij} - \sum_{k=0}^{i-1} L_{ik} L_{jk} \right) \quad j = i + 1, i + 2, \dots, N - 1$$

Notice that it is only needed to compute matrix elements at and below the diagonal. Hence Cholesky-decomposition is approximately twice as fast as LU-decomposition.

```
In[1]:= A = Table[2 Random[] - 1, {i, 20}, {j, 6}];
```

```
In[2]:= B = Transpose[A].A;
```

```
B // MatrixForm
```

Out[3]//MatrixForm=

$$\begin{pmatrix} 5.80211 & -2.4489 & -0.360758 & -2.97801 & 0.172635 & 1.67832 \\ -2.4489 & 6.43294 & -1.86467 & 0.606378 & -0.556698 & -1.30022 \\ -0.360758 & -1.86467 & 6.53501 & 1.13656 & 0.579452 & -1.81166 \\ -2.97801 & 0.606378 & 1.13656 & 6.26633 & -1.02099 & -0.278366 \\ 0.172635 & -0.556698 & 0.579452 & -1.02099 & 5.11047 & -3.16761 \\ 1.67832 & -1.30022 & -1.81166 & -0.278366 & -3.16761 & 8.51815 \end{pmatrix}$$

```
In[4]:= L = Transpose[CholeskyDecomposition[B]];
```

```
L // MatrixForm
```

Out[5]//MatrixForm=

$$\begin{pmatrix} 2.40876 & 0. & 0. & 0. & 0. & 0. \\ -1.01667 & 2.32365 & 0. & 0. & 0. & 0. \\ -0.14977 & -0.868006 & 2.39982 & 0. & 0. & 0. \\ -1.23633 & -0.27997 & 0.29518 & 2.1383 & 0. & 0. \\ 0.0716698 & -0.208222 & 0.170616 & -0.486856 & 2.18994 & 0. \\ 0.696757 & -0.254708 & -0.803557 & 0.350249 & -1.35299 & 2.31708 \end{pmatrix}$$

```
In[6]:= L.Transpose[L] - B // MatrixForm
```

Out[6]//MatrixForm=

$$\begin{pmatrix} 8.88178 \times 10^{-16} & -4.44089 \times 10^{-16} & 0. & 0. & 2.77556 \times 10^{-17} & 0. \\ -4.44089 \times 10^{-16} & 8.88178 \times 10^{-16} & 0. & 1.11022 \times 10^{-16} & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 2.22045 \times 10^{-16} \\ 0. & 1.11022 \times 10^{-16} & 0. & -8.88178 \times 10^{-16} & 0. & 2.22045 \times 10^{-16} \\ 2.77556 \times 10^{-17} & 0. & 0. & 0. & -8.88178 \times 10^{-16} & 4.44089 \times 10^{-16} \\ 0. & 0. & 2.22045 \times 10^{-16} & 2.22045 \times 10^{-16} & 4.44089 \times 10^{-16} & -1.77636 \times 10^{-15} \end{pmatrix}$$

Cholesky decomposition vs. LU decomposition:

- Cholesky decomposition can only be used for symmetric and positive definite matrices.
- Compared to LU decomposition, Cholesky decomposition is approximately twice as fast and requires only half the storage
- Pivoting is unnecessary in Cholesky decomposition and should NEVER be carried out!!!
- Cholesky decomposition is numerically more stable than LU decomposition (reason is subtle and beyond the scope of NR)

We must still discuss the following:

- How do we compute the accuracy on a solution x to the linear equation $Ax=b$?
(We define the accuracy on the solution x as $\|x-x_{\text{True}}\|$ where x_{True} is the true solution. Unfortunately, we don't know x_{True})

Exercise:

Two data sets from real applications will be provided by Jens

Establish the design matrix $A_{ij} = \frac{X_j(x_i)}{\sigma_i}$ and right hand side $b_i = \frac{y_i}{\sigma_i}$ for each of the two applications

Establish the Normal equations and solve these with Cholesky decomposition.

Compare your results to the results and error bounds stated on the web pages.