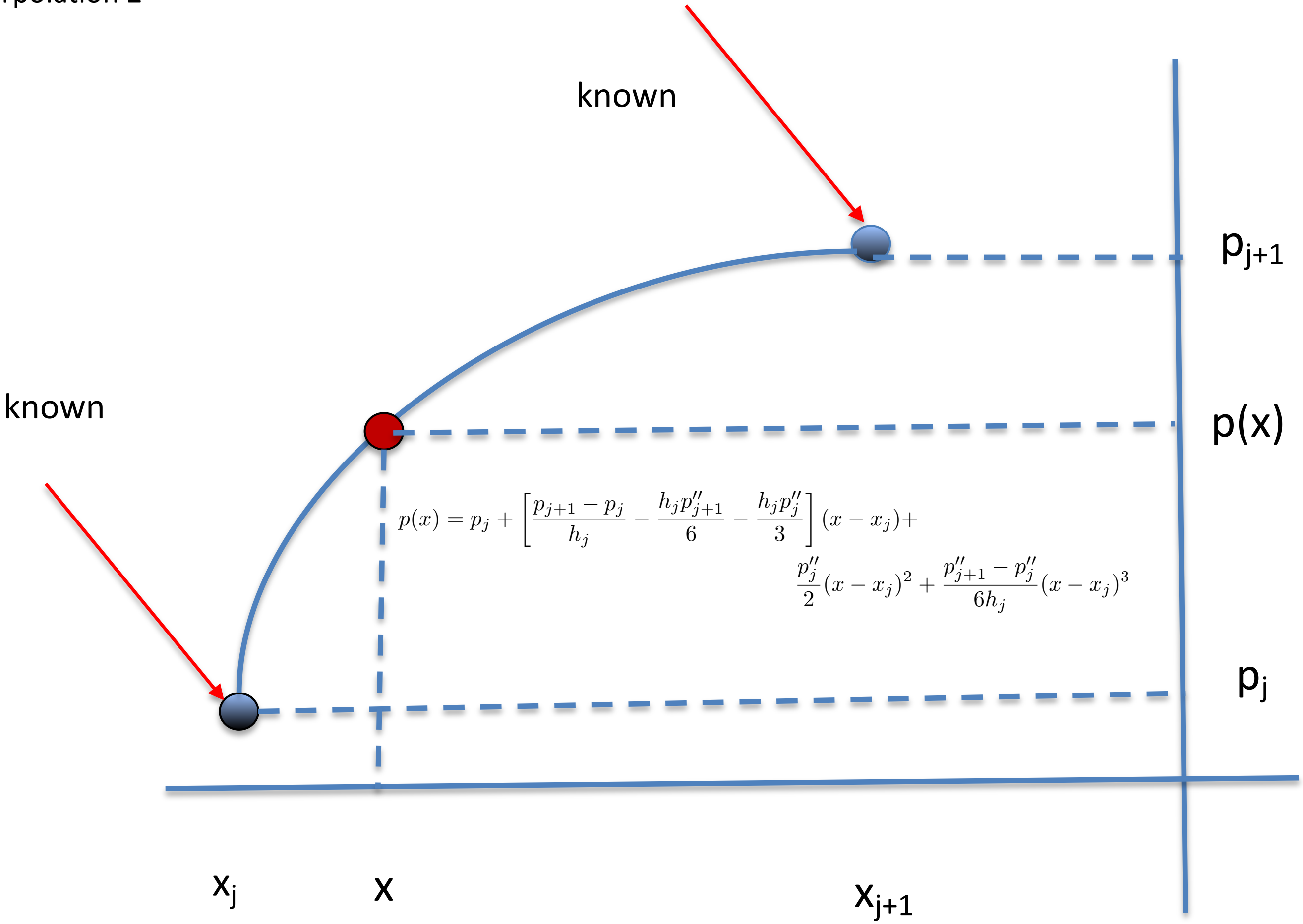
Interpolation 2



Notice that all we need is the second derivatives, *p''*

$$\begin{bmatrix} 2h_1 & h_1 & & & & & \\ h_1 & 2(h_1+h_2) & h_2 & & & & \\ & h_2 & 2(h_2+h_3) & h_3 & & & \\ & & & \ddots & & & \\ & & & & h_{N-2} & 2(h_{N-2}+h_{N-1}) & h_{N-1} \\ & & & & & h_{N-1} & 2h_{N-1} \end{bmatrix} \begin{bmatrix} p''_1 \\ p''_2 \\ p''_3 \\ \vdots \\ p''_{N-1} \\ p''_N \end{bmatrix}$$

$$= \begin{bmatrix} 6\frac{p_2-p_1}{h_1} - 6p'_1 \\ 6\frac{p_3-p_2}{h_2} - 6\frac{p_2-p_1}{h_1} \\ 6\frac{p_4-p_3}{h_3} - 6\frac{p_3-p_2}{h_2} \\ \vdots \\ 6\frac{p_n-p_{N-1}}{h_{N-1}} - 6\frac{p_{N-1}-p_{N-2}}{h_{N-2}} \\ -6\frac{p_N-p_{N-1}}{h_{N-1}} + 6p'_N \end{bmatrix} \quad (1)$$

Tridiagonal systems

$$\begin{pmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & c_2 & & & \\ & a_3 & b_3 & c_3 & & \\ & & & \ddots & & \\ & & & a_{N-1} & b_{N-1} & c_{N-1} \\ & & & & a_N & b_N \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{N-1} \\ x_N \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_{N-1} \\ r_N \end{pmatrix} \tag{5}$$

To solve, we use *Gaussian elimination*. For example, multiplying first row by a_2 / b_1 and subtracting from second row gives a new equation. We can substitute this equation in for the second row and get

$$\begin{array}{llll} b_1 x_1 & +c_1 x_2 & = r_1 & \text{original first row} \\ & \left(b_2 - \frac{a_2}{b_1} c_1\right) x_2 + c_2 x_3 & = r_2 - \frac{a_2}{b_1} r_1 & \text{modified second row} \end{array}$$

The process continues *N – 1* times after which the system has the form

$$
\begin{array}{cccccccc}
\beta_1 x_1 & +c_1 x_2 & & & & & = & \rho_1 \\
& \beta_2 x_2 & +c_2 x_3 & & & & = & \rho_2 \\
& & \beta_3 x_3 & +c_3 x_4 & & & = & \rho_3 \\
& & & & \ddots & & & \vdots \\
& & & & \beta_{N-1} x_{N-1} & +c_{N-1} x_N & = & \rho_{N-1} \\
& & & & & \beta_N x_N & = & \rho_N
\end{array}
\tag{6}
$$

where

$$\beta_1 = b_1, \quad \beta_j = b_j - \frac{a_j}{\beta_{j-1}} c_{j-1} \quad j = 2, \ldots, N \tag{7}$$

and

$$\rho_1 = r_1, \quad \rho_j = r_j - \frac{a_j}{\beta_{j-1}} \rho_{j-1} \quad j = 2, \ldots, N. \tag{8}$$

We can write the general form for the solution of x_j as,

$$x_{N-j} = \frac{(\rho_{N-j} - c_{N-j}x_{N-j+1})}{\beta_{N-j}}, \quad j = 1, \ldots N-1 \qquad (9)$$

That's it! You now have everything you need to interpolate using *cubic splines*. You use Eqs (6) -(9) to find the second derivatives, which are then substituted into Eq. (2) to find the values of the spline.

```
% Trisolve
%
% Inputs: The coefficients a,b,c and
%         r.h.s. r's
% Outputs: An array containing the solutions
%          to the tridiagonal system
%
% Input the  a, b, c, and r
% Loop 2 to N
%       calculate beta, and rho using
%       Eqs. 2.21 and 2.22
% End Loop
%
% Now back substitute
%
& Loop 1 to N-1
%       find x using Eq. 2.23
% End Loop
```

Notes: You will have to do the rho(1), beta(1), and x(N) outside their respective loops. You'll also have to make sure that no b's = 0. Why?

Matlab

```
function [x] = trisolve(A,r)
% This function solves the tridiagonal set of equations
%
%   / b(1) c(1)                           \  / x(1) \    / r(1) \
%   | a(2) b(2) c(2)                      |  | x(2) |    | r(2) |
%   |      a(3) b(3)  c(3)                |  | x(3) |    | r(3) |
%   |           ...                       |  | ...  | =  |      |
%   |        a(N-1) b(N-1) c(N-1)         |  |x(N-1)|    |r(N-1)|
%   \                a(N)   b(N)          /  \ x(N) /    \ r(N) /
%
% The entire matrix is provided, the appropriate diagonals are
% extracted using the MatLab diag command
N = length(A);
b = diag(A);
a = [0];
a = [a;diag(A,-1)];
c = diag(A,1);
c = [c;0];
if (b(1) == 0), error('Zero diagonal element in TRISOLVE');  end
beta(1) = b(1);
rho(1)  = r(1);
for j=2:N
    beta(j) = b(j) - a(j) *  c (j-1) / beta(j-1);
    rho(j)  = r(j) - a(j) * rho(j-1) / beta(j-1);
    if (b(j) == 0)
        error('Zero diagonal element in TRISOLVE');
    end
end
% Now, for the back substitution...
x(N) = rho(N) / beta(N);
for j = 1:N-1
    x(N-j) = ( rho(N-j)-c(N-j)*x(N-j+1) )/beta(N-j);
end
```

Interpolation is about fitting a function through all the data points. The function will necessarily pass through *all the points*

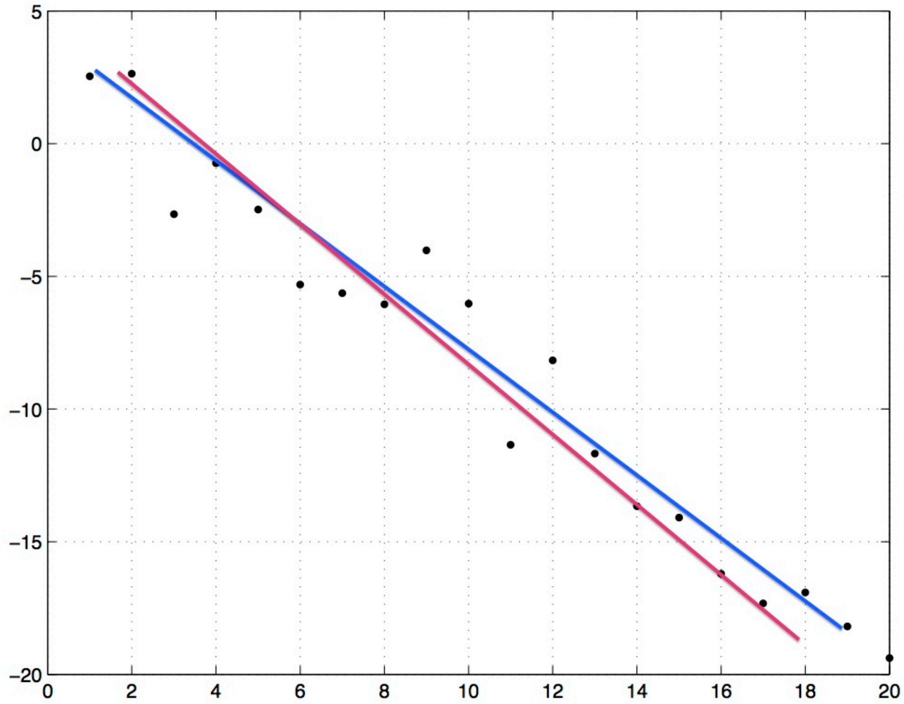
Curve fitting is about fitting the data points to a function. The function is often justified on theoretical grounds and in curve fitting, we seek the parameters that fix the function. The function will *not pass through all the points.*

Goals today.

- Generalized linear least square fitting
- *LU* factorization
- Non linear least square fitting

To get a feeling for what curve fitting entails, do questions 1—2 on the worksheet and then S T O P

(1)



(2) Several ideas.

- For example, pick the line that passes through *most* points;
- pick the line that has as many points *above the line as below the line*;
- pick the line that has the *least total distance* from the points to the line….

We will choose the line that *minimizes* the quantity,

$$\chi^2(a_1, a_2) = \sum_{i=1}^{N} \left(\frac{y_i - a_1 - a_2 x_i}{\sigma_i} \right)^2 \qquad (1)$$

That is, we choose the quantity that *minimizes* the sum of the square of the distances each data point is from a theoretical line.

Do questions 3—5 on the worksheet S T O P

(3) Eq. (1) is squared because we don't care if the distance of line to data point is positive or negative. That is we care about distance, not displacement.

(4) We are minimizing with respect to the parameters a_1 and a_2.

(5) We minimize by taking partial derivatives, $\frac{\partial \chi^2}{\partial a_1}$ and $\frac{\partial \chi^2}{\partial a_2}$ and setting them = 0

$$0 = \frac{\partial \chi^2}{\partial a_1} = -2\sum_{i=1}^{N} \frac{y_i - a_1 - a_2 x_i}{\sigma_i^2}$$

$$0 = \frac{\partial \chi^2}{\partial a_2} = -2\sum_{i=1}^{N} \frac{x_i(y_i - a_1 - a_2 x_i)}{\sigma_i^2}.$$

Notation is simplified if we use:

$$S \equiv \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \qquad S_x \equiv \sum_{i=1}^{N} \frac{x_i}{\sigma_i^2}; \qquad S_y \equiv \sum_{i=1}^{N} \frac{y_i}{\sigma_i^2}$$

$$S_{xx} \equiv \sum_{i=1}^{N} \frac{x_i^2}{\sigma_i^2} \qquad S_{xy} \equiv \sum_{i=1}^{N} \frac{x_i y_i}{\sigma_i^2}.$$

$$\begin{aligned} a_1 S + a_2 S_x &= S_y \\ a_1 S_x + a_2 S_{xx} &= S_{xy}. \end{aligned}$$

Solving:

$$\Delta \equiv S S_{xx} - (S_x)^2$$

$$a_1 = \frac{S_{xx} S_y - S_x S_{xy}}{\Delta}$$

$$a_2 = \frac{S S_{xy} - S_x S_y}{\Delta}.$$

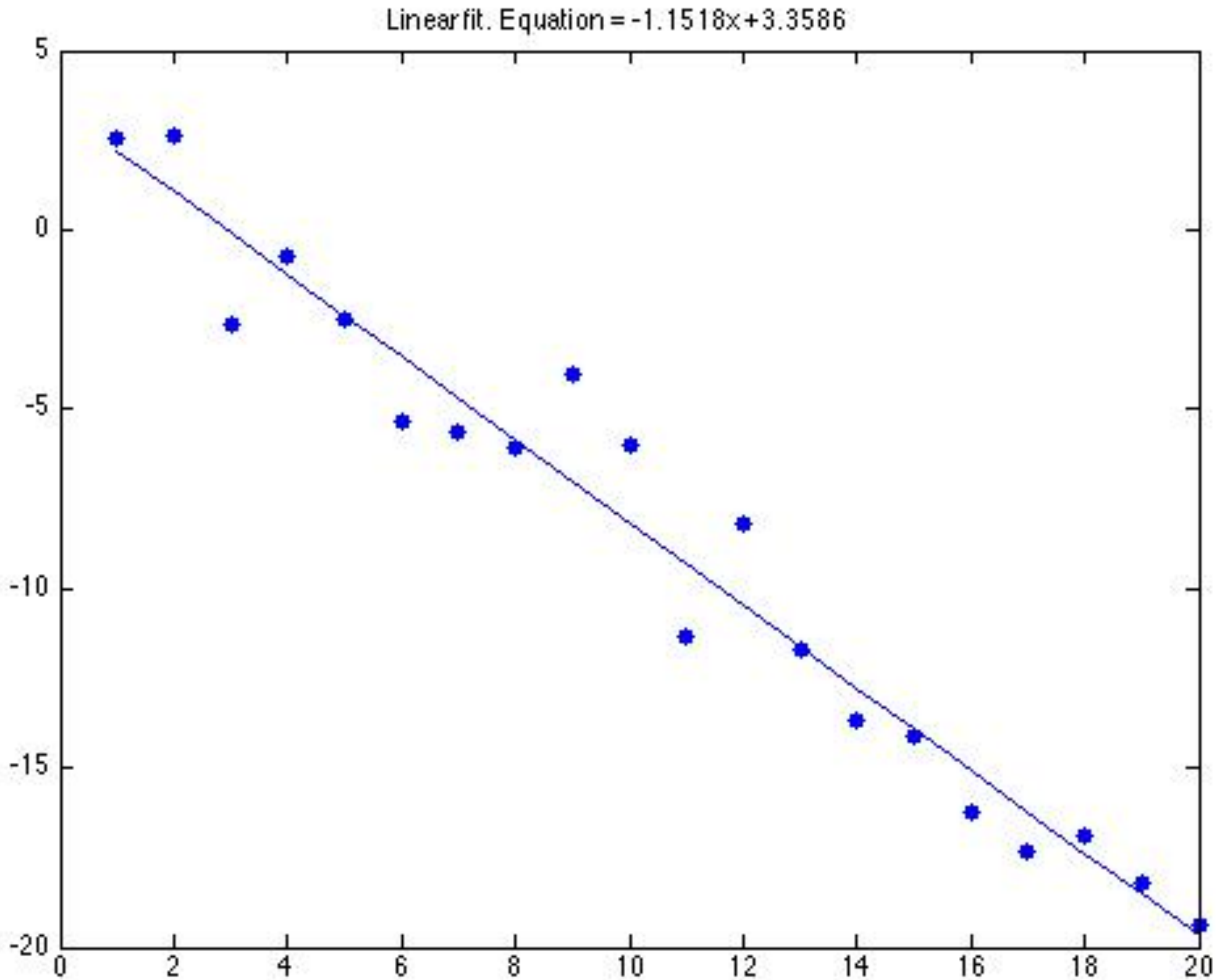
with

$$\sigma_{a_1}^2 = \frac{S_{xx}}{\Delta}$$

$$\sigma_{a_2}^2 = \frac{S}{\Delta}.$$

Do question (6) on the worksheet

Interpolation 2



When we studied *cubic splines*, we learned we needed to solve *tridiagonal linear systems.*

In order to *fit* data to *higher order polynomials or other functions* we will have to solve more *general types of linear systems.*

So before we go on to learning how to fit to these other functions, we study how to solve more general types of linear systems.

A general system of linear equations ,

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1N}x_N &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2N}x_N &= b_2 \\ &\vdots \\ a_{N1}x_1 + a_{N2}x_2 + \cdots + a_{NN}x_N &= b_2 \end{aligned}$$

Can be written in matrix form as

$$\boldsymbol{Ax} = \boldsymbol{b} \tag{2}$$

where $\boldsymbol{A}$ is a matrix containing the coefficients and $\boldsymbol{b}$ a vector containing the R.H.S. of the equations.

There are several ways to try to solve this matrix equation, one of the most effective is *LU factorization*

$$
\begin{pmatrix}
a_{11} & a_{12} & a_{13} & \cdots & a_{1N} \\
a_{21} & a_{22} & a_{23} & \cdots & a_{2N} \\
a_{31} & a_{32} & a_{33} & \cdots & a_{3N} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
a_{N1} & a_{N2} & a_{N3} & \cdots & a_{NN}
\end{pmatrix}
= \begin{pmatrix}
l_{11} & 0 & 0 & \cdots & 0 \\
l_{21} & l_{22} & 0 & \cdots & 0 \\
l_{31} & l_{32} & l_{33} & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
l_{N1} & l_{N2} & l_{N3} & \cdots & l_{NN}
\end{pmatrix}
\begin{pmatrix}
1 & u_{12} & u_{13} & \cdots & u_{1N} \\
0 & 1 & u_{23} & \cdots & u_{2N} \\
0 & 0 & 1 & \cdots & u_{3N} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 1
\end{pmatrix} \quad (3)
$$

or ***A = L U*** where ***L*** and ***U*** are the matrices given by the right hand side of Eq. (3)

Why one would *factor* a matrix in this way we will get to later, but for now do question (8-i) on the worksheet and S T O P to begin to figure out what the *l's* and *u's* are.

(8)—(i) $l_{11} = a_{11};\ l_{21} = a_{21};\ l_{31} = a_{31};\ \ldots l_{j,1} = a_{j,1}$

$$\begin{pmatrix} l_{11} = a_{11} & \ldots & \ldots \\ l_{21} = a_{21} & \ldots & \ldots \\ \vdots & \vdots & \vdots \\ l_{N1} = a_{N1} & \ldots & \ldots \end{pmatrix} \begin{pmatrix} 1 & u_{12} =? & \ldots \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \end{pmatrix}$$

Now do (ii) and STOP

(ii)

$$\begin{aligned} a_{12} &= l_{11}u_{12} \\ a_{13} &= l_{11}u_{13} \\ &\vdots \\ a_{1N} &= l_{11}u_{1N} \end{aligned}$$

or

$$u_{1j} = \frac{a_{1j}}{l_{11}} \quad j = 2, \ldots, N$$

Finally do (iii) and STOP

$$\begin{pmatrix} l_{11} = a_{11} & \ldots & \ldots \\ l_{21} = a_{21} & \ldots & \ldots \\ \vdots & \vdots & \vdots \\ l_{N1} = a_{N1} & \ldots & \ldots \end{pmatrix} \begin{pmatrix} 1 & u_{12} = \frac{a_{12}}{l_{11}} & \ldots u_{1j} = \frac{a_{1j}}{l_{11}} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \end{pmatrix}$$

(iii)

$$
\begin{aligned}
a_{22} &= l_{21}u_{12} + l_{22} \\
a_{32} &= l_{31}u12 + l_{32} \\
&\vdots \\
a_{N2} &= l_{N1}u_{12} + l_{N1} \\
&\text{or} \\
l_{i2} &= a_{i2} - l_{i1}u_{12} \quad i = 1, \ldots, N
\end{aligned}
$$

$$
\begin{pmatrix}
l_{11} = a_{11} & l_{12} = a_{12} - l_{11}u_{12} & \ldots \\
l_{21} = a_{21} & \vdots & \ldots \\
\vdots & \vdots & \vdots \\
l_{N1} = a_{N1} & l_{N2} = a_{N2} - l_{N1}u_{12} & \ldots
\end{pmatrix}
\begin{pmatrix}
1 & u_{12} = \frac{a_{12}}{l_{11}} & \ldots u_{1j} = \frac{a_{1j}}{l_{11}} \\
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots
\end{pmatrix}
$$

In general we find that

$$
\begin{aligned}
l_{ik} &= a_{ik} - \sum_{j=1}^{k-1} l_{ij}u_{jk}, \quad i = k, k+1, \ldots, N, \\
u_{kj} &= \frac{a_{kj} - \sum_{i=1}^{k-1} l_{ki}u_{ij}}{l_{kk}}, \quad j = k+1, k+2, \ldots, N
\end{aligned}
$$

So what has all this manipulation gotten us.

Since ***A x = b*** and we are assuming it is possible to write ***A = LU***, we have

LU x = b

Letting ***z = Ux*** we can rewrite the linear system as

L z = b. (1)

But *L* is lower triangular. This means that system of equations given by (1) is of the form:

$$
\begin{aligned}
l_{11}z_1 &= b_1 \\
l_{21}z_1 + l_{22}z_2 &= b_2 \\
l_{31}z_1 + l_{32}z_2 + l_{33}z_3 &= b_3 \\
\vdots \quad \vdots \quad \vdots \qquad & \qquad \vdots \\
l_{N1}z_1 + l_{N2}z2 + l_{N3}z_3 + \cdots + l_{NN}z_N &= b_N
\end{aligned}
$$

All the *l's* are known. So from the first row we can find z_1 . From the second, z_2 and so on

The general solution to *z* is: $z_i = \dfrac{b_i - \sum_{k=1}^{i-1} l_{ik} z_k}{l_{ii}}; \quad i = 2, \cdots, N$

But we're not interested in *z*, we want *x*. Recall however that

$$Ux = z$$

and *U* is upper triangular. This means that the system looks like

$$\begin{aligned} x_1 + u_{12}x_2 + u_{13}x_3 + \cdots + u_{1N}x_N &= z_1 \\ x_2 + u_{23}x_3 + \cdots + u_{2N}x_N &= z_2 \\ \ddots &\ \vdots \quad \vdots \\ x_N &= z_N \end{aligned}$$

And once again, we see it is straightforward to solve by "going up" the system.

The general solution for *x* is

$$x_{N-i} = z_{N-i} - \sum_{k=N-i+1}^{N} u_{N-i,k}x_k, \quad i = 1, \ldots, N-1$$

So solving a system of equations on a computer becomes straightforward