We have found that in order to interpolate data using a cubic spline that tri-diagonal matrices like,

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

need to be solved. We found that with

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

and

$$\rho_1 = r_1, \quad \rho_j = r_j - \frac{a_j}{\beta_{j-1}} \rho_{j-1} \quad j = 2, \ldots, N. \tag{3}$$

the solution is given by

$$x_{N-j} = \frac{(\rho_{N-j} - c_{N-j} x_{N-j+1})}{\beta_{N-j}}, \quad j = 1, \ldots, N-1 \tag{4}$$

A pseudo code for this is shown below. Translate the pseudo code to `MatLab`. Then discuss how you would this to find cubic splines

```
% 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?

Figure 1: Pseudo Code to solve tri-diagonal systems

1 Curve Fitting

Interpolation approximates a function in such a way that the interpolation passes through a known set of points. *Curve fitting* is a procedure that determines the parameters to a function that *best fits* a set of points. This necessarily means that the function will not pass through all the points. What best means and what we mean by parameters are discussed next.

Table 1: Some data which may be linear

x	y	x	y
1	2.5377	11	-11.3500
2	2.6339	12	-8.1651
3	-2.6588	13	-11.3500
4	-0.7378	14	-13.6630
5	-2.4812	15	-14.0850
6	-5.3077	16	-16.2050
7	-5.6336	17	-17.3240
8	-6.0574	18	-16.9100
9	-4.0216	19	-18.1910
10	-6.0306	20	-19.3830

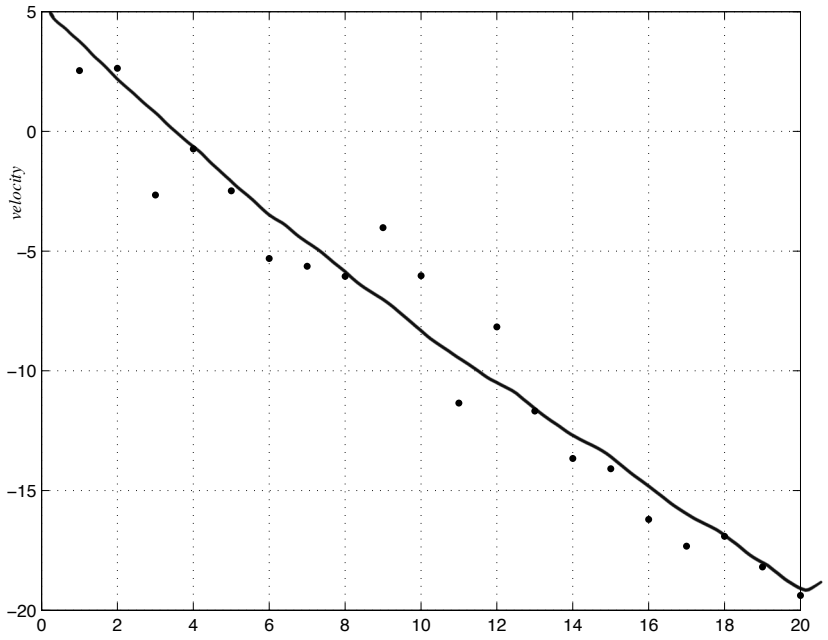

Figure 2: Data perhaps representing a linear function.

(1) The data from table 1 is plotted in figure 1. Suppose you have reason to believe data should be linear. Sketch some lines that you think best represent the data as a line.

(2) How might one determine the "best" line through the data

One way to determine the best fit line through some data is to minimize the error between a theoretical line, y_i and the data. If we write the line associated

$$y(x) = y(x; a_1, a_2) = a_1 + a_2 x \tag{1}$$

where we are seeking to find a_1, a_2 that give the best line to a particular data set. We do this by minimizing the chi-square function

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

where the σ_i are the measurement errors and we seek to find the parameters a_1, a_2.

(3) Why is the right hand side of Eq. (2) shown as squared?

(4) In equation (2), what term(s) is one varying in order to find a minimum.

(5) How does one minimize a function like that given in Eq. (2).

(6) Translate the pseudo code given in box 2.2.1 in the Course Notes to `MatLab`.

When we studied interpolation using cubic splines, we found we needed to solve tridiagonal systems of linear equations. In order to *fit* data to higher order polynomials, we'll need to solve general systems of linear equations. We introduce now a method for handling such equations, $\boldsymbol{LU}$ factorization.

A general system of 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

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{3}$$

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

In $\boldsymbol{LU}$ factorization, we try to write the matrix $\boldsymbol{A}$ as the product of lower diagonal ($\boldsymbol{L}$) and upper diagonal ($\boldsymbol{U}$) matrices. That is,

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1N} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2N} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & a_{N3} & \cdots & a_{NN} \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & 0 & \cdots & 0 \\ l_{21} & l_{22} & 0 & \cdots & 0 \\ l_{31} & l_{32} & l_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{N1} & l_{N2} & l_{N3} & \cdots & l_{NN} \end{pmatrix} \begin{pmatrix} 1 & u_{12} & u_{13} & \cdots & u_{1N} \\ 0 & 1 & u_{23} & \cdots & u_{2N} \\ 0 & 0 & 1 & \cdots & u_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} \tag{4}$$

or

$$\boldsymbol{A} = \boldsymbol{LU}$$

where the matrices $\boldsymbol{L}, \boldsymbol{U}$ are given by the right hand side of Eq. (3).

(8) If the factorization of a known matrix A by Eq. (3) is possible, we need to find the coefficients, $l_{11}, l_{21}, \ldots, u_{12}, u_{13}, \ldots$. It turns out this is not too difficult. Let's start with the easiest

(i) What are l_{j1} in terms of the known a_{ij}.

(ii) One down, now find an expression for $u_{12}, u_{13}, \ldots$.

(iii) Now move to the second column of A and *solve* for unknown l_{2j}.

(9) Translate the pseudo code given in box 2.2.2 in the Course Notes to `MatLab`.