Week 11 Part B
Matlab: Linear Indexing; Linear Equations, Curve Fitting, Short Circuit Evaluation
Linear indexing

• You have learnt how to index an element of a matrix using its row and column indices
• An alternative is to use linear indexing
• Sometimes linear indexing is more convenient
Linear indexing (Example)

- $M(2,3) = 10$
- $M(3,4) = 12$

Linear index in the top left hand corner

- $M(10) = 10$
- $M(15) = 12$
Using find with linear index

```
>> linearIndex = find(M == 12)

linearIndex =

    15

>> [rowIndex, colIndex] = ind2sub([4 4], linearIndex)

rowIndex =

    3

colIndex =

    4
```
Remarks

• find() can also return row and column indices
  – For 3-dimensional array or higher dimension, need to use linear indexing

• Linear indexing is convenient for finding the maximum or minimum in a matrix or higher dimensional array
Simultaneous equations

- Matrix arithmetic is ideally suited to solving $n$ (or more) equations in $n$ unknowns, such as
- In matrix notation,
  \[
  \begin{bmatrix} 3 & -2 \\ 5 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -3 \\ 14 \end{bmatrix}
  \]
  or
  \[
  A \ x = b
  \]
- to solve for $x$, premultiply by $A^{-1}$, the inverse of $A$

>> A = [3 -2; 5 3];  b = [-3; 14];  % b is a column vector
>> x = A \ b    % left-division is equivalent and preferred
x =
    1.0000
    3.0000

Never ever use this method! Why? Finite precision causes error!

>> x = inv(A) * b  % in theory correct, but not robust numerically
Reliability

Just because you can express equations in matrix form doesn’t mean they can be solved

- may be no solutions
  \[ x_1 + 2x_2 = 2; \quad 2x_1 + 4x_2 = 3 \]
- may be an infinite number of solutions
  \[ x_1 + 2x_2 = 2; \quad 2x_1 + 4x_2 = 4 \]
- linear algebra courses will cover determinants, singular matrices, rank, ill-conditioned equations etc
- later engineering courses will show how these can apply in engineering situations
- in some cases approximations can be obtained
  if solution is not unique \( \text{inv}(A) \) won’t exist, but one solution often can be obtained using the Moore-Penrose pseudo-inverse \( \text{pinv}(A) \) instead

The matrix \([1, 2; 2, 4]\) is the same in both cases, and it’s singular.
Example – cantilever forces*

A sign with mass $m$ kg is suspended from a cantilever truss as shown below.

- Each rigid member carries a force $F_i$ with directions shown
- The wall exerts forces at the attachment points, direction unknown (hence vector notation)
- The sign exerts a force $mg/2$ N at the points shown
- Forces are independent of $L$

The forces must balance at each junction, both horizontally and vertically.

* Holloway (2004). Sec 6.8.1
Cantilever equations

For the right-hand sign anchor point, in the $x$ direction the forces are $F_3$ and $1/\sqrt{2}$ of $F_2$; for $y$ there’s $1/\sqrt{2}$ of $F_2$, which balances the force exerted by half the mass:

$$F_3 + F_2/\sqrt{2} = 0$$
$$F_2/\sqrt{2} = -mg/2$$
$$F_4 + F_3 = 0$$
$$F_6 = -mg/2$$

$$F_1 + F_2/\sqrt{2} + F_5/\sqrt{2} = 0$$
$$F_5/\sqrt{2} - F_6 - F_2/\sqrt{2} = 0$$
$$W_{1x} + F_1 = 0$$
$$W_{1y} = 0$$

$$W_{2x} + F_4 + F_5/\sqrt{2} = 0$$
$$W_{2y} + F_5/\sqrt{2} = 0$$
Cantilever matrix formulation

- By expressing the equations this way the coefficient matrix contains only constants.

\[
\begin{bmatrix}
0 & 1/\sqrt{2} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1/\sqrt{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 1/\sqrt{2} & 0 & 0 & 1/\sqrt{2} & 0 & 0 & 0 & 0 & 0 \\
0 & -1/\sqrt{2} & 0 & 0 & 1/\sqrt{2} & -1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1/\sqrt{2} & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1/\sqrt{2} & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
F_1 \\
F_2 \\
F_3 \\
F_4 \\
F_5 \\
F_6 \\
W_{1x} \\
W_{1y} \\
W_{2x} \\
W_{2y} \\
\end{bmatrix} =
\begin{bmatrix}
0 \\
-\text{mg} / 2 \\
0 \\
-\text{mg} / 2 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
\end{bmatrix}
\]
Cantilever solution

\[
p\text{Mass} = 50; \quad \% \quad 50\text{kg} \sim 490.3 \text{N}
\]
\[
g = 9.81; \quad \% \quad \text{Acceleration due to gravity}
\]
\[
r_2 = \frac{1}{\sqrt{2}}; \quad \% \quad 45\text{-degree component}
\]
cantA = \[
\begin{bmatrix}
0 & r_2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & r_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & r_2 & 0 & 0 & r_2 & 0 & 0 & 0 & 0 & 0 \\
0 & -r_2 & 0 & 0 & r_2 & -1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & r_2 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & r_2 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]
b = \[
\begin{bmatrix}
0 & p\text{Mass}*g/2 & 0 & p\text{Mass}*g/2 & \ldots \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}.
\]
sol = cantA \backslash b; \quad \% \quad \text{this is the easy part!}

\text{printmat(sol, 'Cantilever solution', ...}
\text{'F1 F2 F3 F4 F5 F6 W1x W1y W2x W2y', 'N'})

\[
\begin{array}{l}
\text{Cantilever solution} = \\
N \\
F1 \quad 735.75000 \\
F2 \quad -346.83588 \\
F3 \quad 245.25000 \\
F4 \quad -245.25000 \\
F5 \quad -693.67175 \\
F6 \quad 245.25000 \\
W1x \quad -735.75000 \\
W1y \quad 0 \\
W2x \quad 735.75000 \\
W2y \quad 490.50000
\end{array}
\]

\textbf{Tip: printmat(m, title, col_labels, row_labels)} displays as above. Labels are space-separated.
Curve fitting

- Matlab can fit various curves (or trend lines) to noisy data under program or user control
- The standard method is a least squares fit
  - The same as trend lines in OpenOffice Calc
- Matlab provides two associated functions for polynomial trendlines, `polyfit` and `polyval`
- Using `polyval`

```matlab
% A polynomial of degree n is represented by vector of length (n+1)
% The vector contains the coefficients of the polynomial in
% descending order of its power
%
% Ex: 4x^3 - 2x + 1 is represented as [4 0 -2 1]
% To calculate the value of the polynomial at -1, 0, 1
polyval([4 0 -2 1], [-1 0 1])
```
Curve fitting

• Using `polyfit` and `polyval`

```matlab
% x, y - vectors representing data points
% n - polynomial order of the desired trendline
>> polyTrend = polyfit(x, y, n);  % polynomial coefficients
>> yTrendline = polyval(polyTrend, x);
    % evaluates trend line at each x (or other vector)
```

• An example is in `lsqfit_demo.m`

• Data file `magnetization_curve.dat`
  – data are noisy measurements of current in field windings of an AC generator, vs no-load output voltage
  – magnetic flux is non-linear at higher currents

• Fit a third order polynomial

• Data from Chapman (2013)
Interactive curve fitting

When you’ve plotted something like the magnetisation data, you can fit curves interactively, just like OpenOffice Calc or Excel.

Dialogue box provides lots of possible curves, displays equations, and can also show residuals (discrepancies between the data points and the fitted curve) on a subplot.
• Regression is polynomial or spline (curves that follow the data closely, not appropriate for noisy data)
Boolean operations

- Don’t use `==` or `~=` for strings, use `strcmp(s1, s2)` or `isempty(s)` instead.

- Built-in Boolean functions include:
  - `isempty(a)` – `a` has no elements (empty array), includes empty strings.
  - `ischar(s)` – `s` is a string (character array).
  - `isinf(v)` – `v` is infinite (the special value `Inf`).
  - `isnan(v)` – `v` is not a defined number (the special value `NaN`).
  - `isnumeric(v)` – `v` is a numeric array.
  - `logical(x)` – interpret `x` as a Boolean, any non-zero is true.
  - `exist(v, ‘var’)` – does the variable exist. There are other options for `exist`.
Short-circuit AND/OR

- **Elementwise (array)**
  
  ```
  & \textit{and} \\
  | \textit{or}
  ```

- **Scalar and short-circuit**
  
  ```
  && \textit{and} \\
  || \textit{or}
  ```

% Using non-short circuit AND
% This throws an error

% clear a % clear the variable a
exist('a','var') & a > 5

% Using short circuit AND
% This does NOT throw an error

% clear a % clear the variable a
exist('a','var') && a > 5

% This throws an error because
% a is not defined

% clear a % clear the variable a
a > 5

Code in
shortCircuitAndDemo.m
• Question: Let boo1 and boo2 be two Boolean expressions
  – If boo1 is FALSE, what is boo1 AND boo2?

• Doesn’t matter what boo2 is, the result is always FALSE

• Short circuit AND evaluation of boo1 && boo2
  – If boo1 is evaluated to be FALSE, stop the evaluation
  – This means boo2 is evaluated only if boo1 is true
Short-circuit AND

- Elementwise (array)
  & and
  | or

- Scalar and short-circuit
  && and
  || or

% Using non-short circuit AND
% This throws an error
clear a % clear the variable a
exist('a','var') & a > 5

Evaluated and caused an error
FALSE

% Using short circuit AND
% This does NOT throw an error
clear a % clear the variable a
exist('a','var') && a > 5

Not evaluated
Usage of Short Circuit AND/OR

• Shortcircuit AND can be used to prevent a program throwing an error because certain exception occurs

• Question: Given two Boolean expression boo1 and boo2. Under what condition can boo1 || boo2 be short circuited?

• Answer: When boo1 is TRUE