The Department of Engineering Science
The University of Auckland

Chapter 10

Linear Equations
and
Linear Algebra

Learning outcomes

- Solve systems of linear equations using MATLAB
- Solve basic linear algebra problems with MATLAB
- Use matrix transformations (rotation, translation, scaling, shearing)
- Apply transition matrices

Solving systems of linear equations

Consider the following system of linear equations:

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &= 1 \\ 2x_1 + 5x_2 + 3x_3 &= 6 \\ x_1 \quad\quad + 8x_3 &= -6 \end{aligned}$$

This system of linear equations can be written in matrix form as:

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 3 \\ 1 & 0 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 6 \\ -6 \end{bmatrix}$$

which is of the general form: $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 3 \\ 1 & 0 & 8 \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ $\mathbf{b} = \begin{bmatrix} 1 \\ 6 \\ -6 \end{bmatrix}$

The solution of $\mathbf{Ax} = \mathbf{b}$ is $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$, ie $\mathbf{x}$ is the inverse of $\mathbf{A}$ multiplied by $\mathbf{b}$.

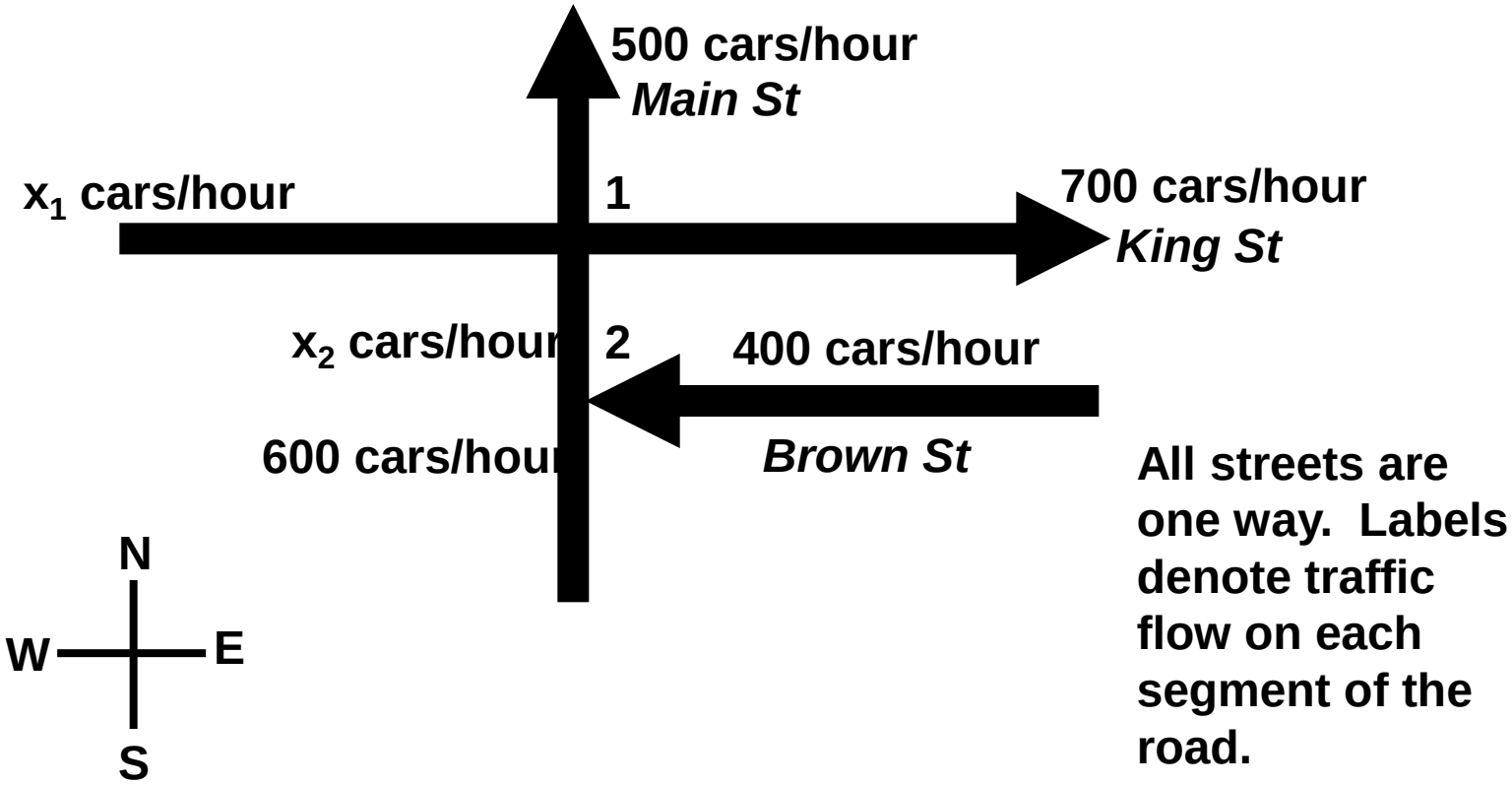
To solve this equation in MATLAB we simply type the following:

```
A = [1 2 3; 2 5 3; 1 0 8];
b = [1; 6; -6];
x = A \ b
```

The left division method performs the equivalent to Gaussian elimination

Systems of Linear Equations
A traffic flow example

- Many engineering problems can be modelled by a system of linear equations.
- Example: Traffic Flow Modelling



All streets are one way. Labels denote traffic flow on each segment of the road.

5 Steps for Problem Solving

1. State the problem clearly
2. Describe the input and output information
3. Work the problem by hand (or with a calculator) for a simple set of data
4. Develop a solution and convert it to a computer program
5. Test the solution with a variety of data

1. State the Problem Clearly

Determine x_1 and x_2 using the known traffic flows of 400, 500, 600 and 700 cars/hour on segments of Main St, King St and Brown St (which are one way streets).

2. Describe the input and output information

Input:

– Main St (north of intersection 1), 500 cars/hour

- King St (west of intersection 1), 700 cars/hour
- Brown St, 400 cars/hour
- Main St (south of intersection 2), 600 cars/ hour

Output:

- x_1 cars per hour travelling on King St (east of intersection 1)
- x_2 cars per hour travelling on Main St (between intersection 1 and intersection 2)

3. Work the Problem by Hand

- Balance the flow of cars into and out of each intersection.
- Intersection 1 $x_1 + x_2 = 500 + 700$
- Intersection 2 $x_2 = 600 + 400$

clearly x_2 is equal to 1000 cars/hour, so

$$x_1 + 1000 = 1200$$

$$\Rightarrow x_1 = 200$$

4. Develop a Solution and Convert it to a Computer Program

- The two equations we need to solve can be expressed as

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1200 \\ 1000 \end{bmatrix}$$

- To solve this problem in Matlab we can use the "left division" operator

```
A = [1, 1; 0, 1;]

b = [1200; 1000;]

x = A\b
```

Matrix Transformations

- A point (or points) can be transformed by using a square matrix. It is possible to create matrices representing stretches, enlargements, reflections and rotation.
- To transform a point we perform a matrix multiplication with the transformation matrix and the point.

For example, we can easily transform the unit square as follows:

Stretch the x direction by 3, reflect in the y axis and then rotate by 45 degrees.

The transformation matrices required are:

x stretch by 3

$$\mathbf{X} = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$$

reflection in y axis

$$\mathbf{Y} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

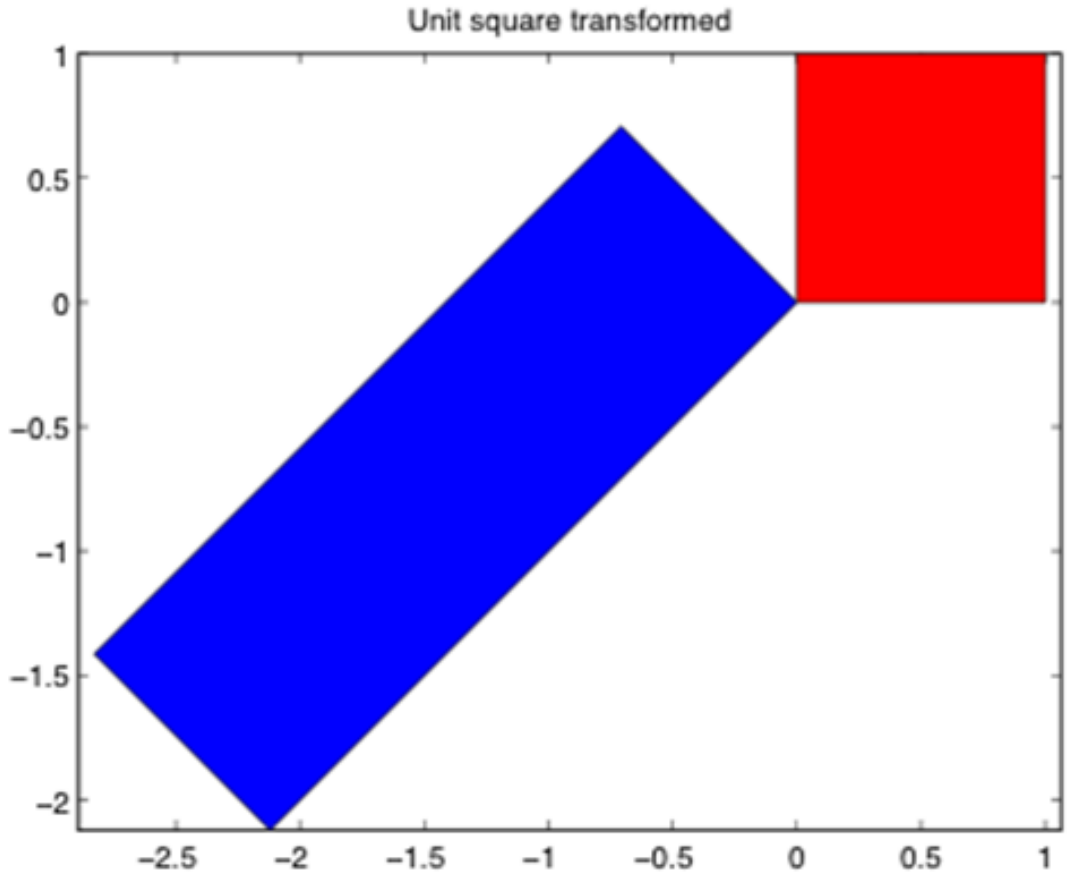
rotation by 45 degrees

$$\mathbf{R} = \begin{bmatrix} \cos(\pi/4) & -\sin(\pi/4) \\ \sin(\pi/4) & \cos(\pi/4) \end{bmatrix}$$

```
% matrix representing the unit square
S = [0 1 1 0; 0 0 1 1]

% plot square
fill( S(1,:), S(2,:), 'r');
title('Unit square transformed');
axis equal
```

```
% matrix to stretch x axis by a factor of 3
X = [3 0; 0 1]
% matrix to reflect in y axis
Y = [-1 0; 0 1]
% matrix to rotate by pi/4 radians (45
   degrees)
R = [cos(pi/4) -sin(pi/4); sin(pi/4)
   cos(pi/4)]
```

```
% perform transformations
T = R * Y * X *S

% plot transformed square
hold on
fill( T(1,:), T(2,:), 'b');
```


Unit square transformed
1
0.5
0
−0.5
−1
−1.5
−2
−2.5
−2
−1.5
−1
−0.5
0
0.5
1

The Department of Engineering Science
The University of Auckland

Chapter 11

Differential Equations
and
"Function" Functions

Learning outcomes

- Explain what a "function" function is
- Use fzero to find the root of a function
- Use ode45 to solve a first order differential equation
- Use feval to write your own "function" functions

What is a "function" function?

- Where a function needs to have access to another specified function in order to perform its task.
- This name of this function is provided as an input argument to the first function.

Finding roots with fzero

- The fzero function allows us to find *x* values for any function *f(x),* such that *f(x)=0* . These *x* values are called roots.

- Consider the problem of solving the equation: $x^2 - x - 12 = 0$

- This can be done by hand but it is also easy to solve using MATLAB.

```
function px = MyPolynomial(x)
   px = x.^2 - x - 12;
return
```

- For example to look for roots near x=5

```
root = fzero(@MyPolynomial, 5)
```

- This will produce

```
root =
            4
```

- To find the other root we need to start looking from a different value:

```
root = fzero(@MyPolynomial, -5)
```

- This will produce the following output:

```
root =
   -3
```

- The @ symbol indicates the name of the function for fzero to evaluate

Solving ODEs in MATLAB

- dy/dt = f(t,y)
- the derivative can be written as some function of the independent variable and dependent variable.

A simple ODE

Consider the problem of determining the volume of muddy water left in a 1000 litre tank, which has sprung a leak. We know that the rate of flow out of the tank will be proportional to the volume left in the tank.

$$\frac{dv}{dt} \propto v$$

We also know that as time goes by the mud will gradually coat the hole, reducing the amount of fluid that can flow out. Hence the rate of flow is inversely proportional to time.

$$\frac{dv}{dt} \propto \frac{1}{t}$$

Putting this together we have the following model:

$$\frac{dv}{dt} = \frac{kv}{t} \qquad v(0) = 1000$$

Where k is determined by experiment to have the value $k = 3$

Note that our dependent variable is v as it depends on time. Our independent variable is t.

ODE45

The MATLAB solver functions need:

1. A MATLAB function that calculates the derivative for any given values of the independent and dependent variables
2. time span array containing two values (a start time and finish time)
3. An initial value

It will produce two outputs:

1. An array of time values (the independent variable)
2. An array of corresponding solution values (the dependent variable)

Create our function definition

```
function dvdt = MuddyTankFlowRate(t,v)
% calculate the flow rate out of a muddy tank of
   water
% inputs: v the volume of water
%         t the time since the start of the flow
% output: dvdt   the flow rate

   k = 3;
   dvdt = k * v / t;
return
```

Set initial conditions. Call ODE45

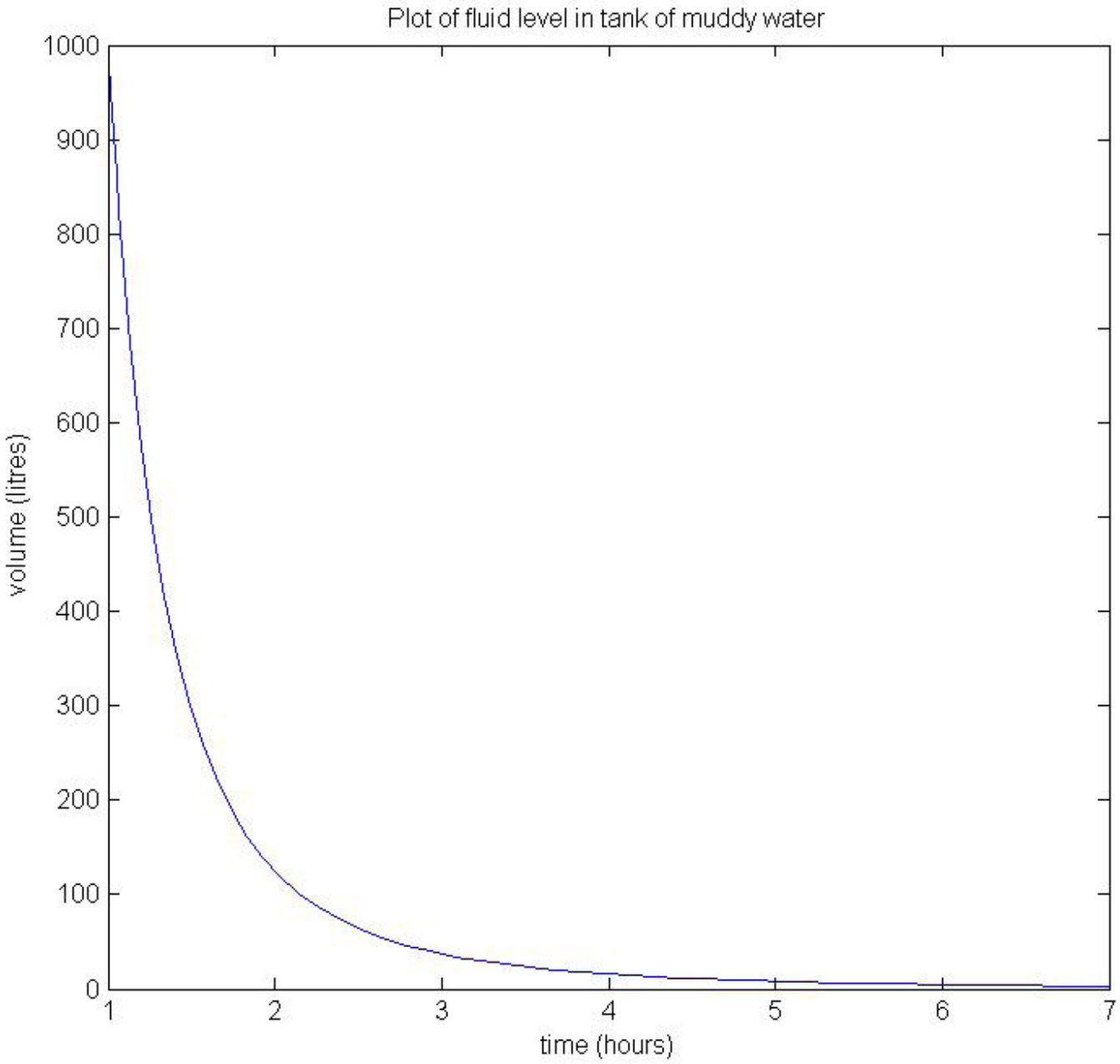
```
% Script to solve the volume of fluid left in a muddy tank
% that contains a hole

% set up a time span of 10 minutes (t is measured in
   seconds)
timeSpan = [0, 600]

% the initial volume at time t=0 is 1000 litres
vInit = 1000

% solve our ODE
[t,v] = ode45(@MuddyTankFlowRate, timeSpan, vInit);
```

Output of ODE45

t	v
1	1000
1.01674590954340	951.39868692652 8
1.03349181908679	905.89686765952 1
.....	863.25118918508 3
6.83913766802859	.....
6.91956883401430	3.12675517422768
7	3.01898421018001
	2.91610943086821

Use results

```
% plot result
plot(t,v)
title('Plot of fluid level in tank of muddy
   water')
xlabel('time (seconds)');
ylabel('volume (litres)')
```


Plot of fluid level in tank of muddy water
1000
900
800
700
600
500
400
300
200
100
0
volume (litres)
1
2
3
4
5
6
7
time (hours)

feval

- We can write our own "function" functions

```
function px = MyPolynomial(x)
   px = x.^2 - x - 12;
return
```

Call passing required arguments

```
p = feval(@MyPolynomial,2)
```

Function name stored as a variable

```
% create a variable to contain the address
   of our function
myFunctionAddress = @MyPolynomial;

p = feval(myFunctionAddress,2)
```

Suggested Reading

Chapter 11 Differential Equations and "Function" Functions	Introduction to Matlab 7 for Engineers (2nd ed)		A Concise Introduction to Matlab (1st ed)	
Topic	**Section**	**Pages**	**Section**	**Pages**
Using fzero	3.2	156-157	3.2	131-132
Solving ODEs	8.5	498-505		