# Numerical Methods ITGS219

## Lecture 1

*By: Zahra Abdalla Elashaal*

## Prerequisites Numerical Methods - ITGS219

- Mathematics ITMM121.
- C programming ITGS122.
- MatLab R2017b Application.
  - This program should be installed in your computer to do your work with it,

## Evaluations

| Activity | Grade |
|---|---|
| Homework's and Quizzes<br>- a Matlab program every week with 5marks, then averaged overall to 20<br>- a quiz a week with 5marks | 20 |
| Midterm exam. | 30 |
| Final Exam | 50 |
| **Total** | **100** |

**Reference Book:** An Introduction to Programming and Numerical Methods in MATLAB
*By: S.R. Otto and J.P. Denier*

**Helping Book:**

- Introduction to Numerical Methods and Matlab Programming for Engineers
- Numerical Methods for Engineers 7th_Edit

# Course Contents:

Chapter 1 Simple Calculations with Matlab.
Chapter 2 Writing Scripts and Functions.
Chapter 3 Loops and Conditional Statements.
Chapter 4 Root Finding.
Chapter 5 Interpolation and Extrapolation.
Chapter 6 Matrices.
Chapter 7 Numerical Integration.
Chapter 8 Solving Differential Equations.

Chapter 1 Simple Calculations with Matlab.

1 Introduction
2 Scalar Quantities and Variables
  2.1 Rules for Naming of Variables.
  2.2 Precedence: The Order in Which Calculations Are Performed
  2.3 Mathematical Functions
3 Format: The Way in Which Numbers Appear
4 Vectors in MATLAB
  4.1 Initialising Vector Objects
  4.2 Manipulating Vectors and Dot Arithmetic
5 Setting Up Mathematical Functions
6 Some MATLAB Specific Commands
  6.1 Looking at Variables and Their Sizes
7 Accessing Elements of Arrays .
8 Tasks

# Matlab and Solving Equations

## Lecture 1

## Simple Calculations with Matlab

**Reference Book:** An Introduction to Programming and Numerical Methods in MATLAB
*By: S.R. Otto and J.P. Denier*

# Introduction:

## Why Do We Need Numerical Methods?

- Some numerical answers are required to rely on approximate methods to obtain useable answers.
- There are many problems whose exact solution is beyond our current state of knowledge.
- There are also many problems which are too long to solve by hand.

When such problems arise we can exploit numerical analysis to reduce the problem to one involving a finite number of unknowns and use a computer to solve the resulting equations.

- The computer then used to solve problems which cannot be solved by hand.
- In this subject we elect to express our ideas in terms of the syntax of the computer package MATLAB.

The name **MATLAB** suggested from **(MATrix LABoratory)**.

In Matlab, the basic objects are **matrices**, i.e. **arrays** of numbers. **Vectors** can be thought of as special matrices.

# Scalar Quantities and Variables

The MATLAB prompt is denoted by **>>** (which does not need to be typed),

```
>> a = 3
a =
    3
>> b = 4;
```

Which mean:
set a equal to 3
set b equal to 4 (and suppress output)

**Note:**
- it is not possible to have **>> 7 = x** (set 7 equal to x)
- whereas we could have **>> x = 7** (set x equal to 7)

These variables can now be used again,

```
>> a = 3;
>> b = a+1;
>> x = a+b;
```

MATLAB can be used into two basic groups: **unary** and **binary** operations,

```
>> 3*4
ans =
      12
>> ans*2
ans =
      24
```

We could have typed the commands on one line as:

```
>> a = 3; b = 4; x = a*b
```

which can be and read as:
set a equal to 3 (don't output anything),
set b equal to 4 (don't output anything)
and set x equal to a times b

# Examples on Scalar Quantities and Variables:

*Example-1* Try entering the following commands into MATLAB, but before you do so try to work out what output you would expect.

```
>> 3*5*6
>> z1 = 34;
>> z2 = 17;
>> z3 = -8;
>> z1/z2
>> z1-z3
>> z2+z3-z1
```

The answers, 90, 2, 42 and −25.

*Example-2* Here we give an example of the simple use of brackets:

```
>> format rat
>> a = 2; b = 3; c = 4;
>> a*(b+c)
>> a*b+c
>> a/b+c
>> a/(b+c)
>> format
```

The answers, 14, 10, 14/3 and 2/7.

(The command format rat has been used to force the results to be shown as rationales, the final command format reverts to the default, which happens to be format short.)

## 1- Rules for Naming of Variables

The rules for naming variables in MATLAB can be summarized as follows:

1. Variable names must start with a letter and can be up to 31 characters long. The trailing characters can be numbers, letters or underscores.
2. There are many choices which are forbidden as variable names, for some reasons:
   - (such as **a*b** which signifies a multiplication of the a and b)
   - (and **a.b**). MATLAB supports object orientated programming. Because of this a.b refers to the value of the "b" component of the object a.
   - The naming of MATLAB files have a single dot. However, in this case a single dot is allowed within the name of the file; everything after the dot is used to tell MATLAB what type of file it is dealing with (whether it be a file containing MATLAB code, or data etc).
3. Variable names are case sensitive, so that a and A are two different objects.
4. It is good programming practice to employ meaningful variable names. however as the examples become more complex our variable names will be more informative.
5. Variables names should not corresponds to or coincide with a predefined MATLAB command or with any user-defined subroutines.

# 2- Precedence: The Order in Which Calculations Are Performed

a(b + c) = a*(b+c) ≠ a*b+c

c+a*b = c + ab ≠ (c + a)b

a/b*c = $\frac{a}{b}C \neq \frac{a}{bc}$

a/(b*c) = $\frac{a}{bc} \neq \frac{a}{b}C$

**Note:**
- all multiplications must be denoted by an asterisk *.
- brackets used to force precedence of the operation.
- operations of division and multiplication take precedence over addition and subtraction.

*Example-3* Determine the value of the expression
a(b + c(c + d))a, where a = 2, b = 3, c = −4, d = −3.

The MATLAB statement to evaluate the expression:

```
>> a = 2; b = 3; c = -4; d = -3;
>> a*(b+c*(c+d))*a
```

The answer 124

*Example-4* Evaluate the following expressions by hand and then check answers with MATLAB.

```
1+2/3*4-5
1/2/3/4
1/2+3/4*5
5-2*3*(2+7)
(1+3)*(2-3)/3*4
(2-3*(4-3))*4/5
```

The answer in the book

**Note:** (type *help precedence* at the MATLAB prompt for more details).

# 2- Precedence:

It is also possible to enter numbers using the exponent-mantissa form. This uses the fact that numbers can be written as **"mantissa × 10^exponent"**

| Number | mantissa - exponent | MATLAB form |
|---|---|---|
| 789.34 . | $7.8934 \times 10^2$ | 7.8934e2 |
| 0.0001 | $1 \times 10^{-4}$ | 1e-4 |
| 4 | $4 \times 10^0$ | 4 |
| 400000000000 | $4 \times 10^{11}$ | 4e11 |

*Example-5* Write 3432.6 in exponent-mantissa form and write $100 \times 10^{10}$ in normal form..

We have $3432.6 \equiv 3.4326 \times 10^3$
And $100 \times 10^{10} \equiv 1,000,000,000,000$.

**Note:**
The **smallest positive number** that MATLAB can store which is different from zero is realmin ≈ $10^{-308}$, whilst the **largest number** is realmax ≈ $10^{308}$.

*Example-6* Use MATLAB to calculate the expression. $b - \frac{a}{b + \frac{b+a}{ca}}$
*where* $a = 3$, $b = 5$ *and* $c = -3$.

```
>> a = 3; b = 5; c = -3;
>> x = b-a/(b+(b+a)/(c*a));
```

the solution will contained in the variable x.

*Example-7* Enter the numbers $x = 45 \times 10^9$ and y = 0.0000003123 using the exponent-mantissa syntax described above. Calculate the quantity xy using MATLAB and by hand...

```
>> x = 45e9;
>> y = 3.123e-7;
>> xy = x*y;
```

# 3- Mathematical Functions

1. **Arithmetic functions:** +, -, / and *.
2. **Trigonometric functions:** sin (sine), cos (cosine) and tan (tangent) (with their inverses as asin, acos or atan). the syntax of the commands is sin(x)
3. **Exponential functions:** exp, log, log10 and ^, which is a binary operation so that a^b = $a^b$
4. Other functions available in MATLAB like:

| | |
|---|---|
| round(x) | Rounds a number to the nearest integer |
| ceil(x) | Rounds a number up to the nearest integer |
| floor(x) | Rounds a number down to the nearest integer |
| fix(x) | Rounds a number to the nearest integer towards zero |
| rem(x,y) | The remainder left after division |
| mod(x,y) | The signed remainder left after division |
| abs(x) | The absolute value of x |
| sign(x) | The sign of x |
| factor(x) | The prime factors of x (factor gives multiple outputs) |
| sqrt(x) | The squared root of x function. |
| size(x) | The size of the vector or matrix, returns the number of rows and columns. |

**Important Point**

It is essential that arguments for functions are contained within round brackets, as cos(x) and when functions are multiplied together an asterisk is used, as f(x) = (x+2) cos x should be written (x+2)*cos(x)

# 3- Mathematical Functions

*Example-8* Calculate the expressions: sin 60◦ (and the same quantity squared), exp(ln(4)), cos 45◦ −sin 45◦, ln exp(2+cos π) and tan 30◦/(tan π/4+tan π/3)..

```
>> x = sin(60/180*pi)
          x =0.8660
>> y = x^2
          y =0.7500
>> exp(log(4))
          ans =4
>> z = 45/180*pi; cos(z)-sin(z)
          ans =1.1102e-16
>> log(exp(2+cos(pi)))
          ans =1
>> tan(30/180*pi)/(tan(pi/4)+tan(pi/3))
          ans =0.2113
```

The values of these expressions should be √3/2, 3/4, 4, 0, 1 and 1/(3 +√3).

**Note:** zero has been approximated by 1.1102e-16 which is smaller than the MATLAB variable, which reflects the accuracy of the calculations.

**Note:**

Some functions takes multiple inputs and returns a single output. others which takes a single input and returns multiple outputs..

*Example-9* the number 12345 = 9 × 1371 + 6, so the remainder of 12345 / 9 = 6. by MATLAB using:

```
>> rem(12345,9);
```

example of function takes a single input and returns multiple outputs is factor which provides the prime decomposition of an integer.

```
>> x = factor(24)
          x = [2 2 2 3]
```

the solution is returned as an array x

# Format: The Way in Which Numbers Appear

*Example-10* Consider the following code on the vector s

```
>> s = [1/2  1/3  pi  sqrt(2)];
>> format short; s
  s = 0.5000   0.3333   3.1416   1.4142
>> format long; s
  s = 0.50000000000000   0.33333333333333   3.14159265358979   1.41421356237310
>> format rat; s
  s = 1/2   1/3   355/113   1393/985
>> format; s
  s = 0.5000   0.3333   3.1416   1.4142
```

short – 5 digits
long – 15 digits
rat – try to represent the answer as a rational.

For more info. type help format

# Vectors in MATLAB

## 1- Initializing Vector Objects

One of the most powerful aspects of MATLAB is its use of vectors (and matrices) as objects.
In Matlab, the basic objects are **matrices**, i.e. **arrays** of numbers. **Vectors** can be thought of as special matrices. A row vector is recorded as a **1 × n** matrix and a column vector is recorded as a **m × 1** matrix.

```
>> r = 1:5;
    r= [1   2   3   4   5]
>> v = [0  1  2  3]
    v= [0   1   2   3]
>> u = [9; 10; 11; 12; 13]
```

the vector u will be column vector

You can access an entry in a vector with

```
>> u(2)
    ans =   10
```

You can change the value of that entry with

```
>> u(2)=47
```

You can extract a slice out of a vector with

```
>> u(2:4)
```

# Vectors in MATLAB

## 1- Initializing Vector Objects

By transposing the vector we can change a row vector into a column vector, and ' call the transpose operator.

**>> w = u'**
**>> x = -1 : .1 : 1**

r = a : h : b, creates the vector r running from a to b in steps of h,

**>> r = 1 : 2 : 5;**

r= [1 3 5]

**>> t = 1 : 2 : 6;**

t= [1 3 5]

Why r and t have the same answer? Which is:
[1 3 5]

**>> s = 1: 0.5 : 3.5;**

s = 1.0000 1.5000 2.0000 2.5000 3.0000 3.5000

**>> m = linspace(0,1);**
**>> y = linspace(0,1,5)**

y = 0.0000 0.2500 0.5000 0.7500 1.0000

m is a row vector runs from 0 to 1 and has 100 elements and y again runs from 0 to 1 but now has 5 elements.

**Note:**
To set up a vector which runs from zero to one in steps of 1/N, we can use:

w = 0:1/N:1
or w = linspace(0,1,N+1)

Example:
typing s=0:0.1:1.0; length(s) . You will find that s has 11 elements.

Example: if N=5 both of the vectors will be:
w = 0 0.2000 0.4000 0.6000 0.8000 1

# Vectors in MATLAB

## 2- Manipulating Vectors and Dot Arithmetic

Dot arithmetic allows us to manipulate vectors in an element-wise fashion rather than treating them as mathematical objects (in fact for addition and subtraction this is the same thing).

*Example:* mathematical objects to multiply a value by a vector,

>> a = [1 2 3];
>> 2*a;

ans = 2 4 6

*Example:* mathematical objects to multiply a vector by a vector,

>> a = [1 2 3];
>> b = [4 5 6];
>> **a*b**
??? Error using ==> *
Inner matrix dimensions must agree.

An error message appears because both a and b are **row vectors** and therefore cannot be multiplied together. to multiply the elements of vector **a** by the elements of vector **b** in an element by element sense. by using dot arithmetic as follows

The **.** indicates to MATLAB to perform the operation term by term and the ***** indicates we require a multiplication.

>> a = [1 2 3];
>> b = [4 5 6];
>> **a.*b**
ans = 4 10 18

The returned vector containing [$a_1b_1$, $a_2b_2$, $a_3b_3$].

# Vectors in MATLAB

## 2- Manipulating Vectors and Dot Arithmetic

*Example:* We can also do a term by term division with

```
>> a = [1 2 3];
>> b = [4 5 6];
>> a./b
      ans = 0.2500 0.4000 0.5000
```

The result is, $\left[\frac{a_1}{b_1}, \frac{a_2}{b_2}, \frac{a_3}{b_3}\right]$

*Example-11* We shall create two vectors running from 1 to 6 and from 6 to 1 and then demonstrate the use of the dot arithmetical operations:

The produces output

```
>> s = 1:6
   s =  1  2  3  4  5  6
>> t = 6:-1:1
   t =  6  5  4  3  2  1
>> s+t
    ans = 7 7 7 7 7 7
>> s-t
    ans = -5  -3  -1  1  3  5
>> s.*t
    ans = 6 10 12 12 10 6
>> s./t
   ans = 0.1667 0.4000 0.7500 1.3333 2.5000 6.0000
>> s.^2
   ans = 1 4 9 16 25 36
>> 1./s
   ans = 1.0000 0.5000 0.3333 0.2500 0.2000 0.1667
>> s/2
   ans = 0.5000 1.0000 1.5000 2.0000 2.5000 3.0000
>> s+1
   ans = 2 3 4 5 6 7
```

Note:
the vectors need to be the same size (or one of them is a scalar – as in the last three examples).

# Setting Up Mathematical Functions

It discuss the ways in which you can set up the input to the function

*Example-12* Set up a vector *x* which contains the values from 0 to 1 in steps of one tenth =1/10.

This can be done in a variety of ways:

```
   % Firstly just list all the values:
>> x = [0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0];
   % Use the colon construction
>> x = 0 : 0.1 : 1.0;
   % Or use the command linspace
>> x = linspace(0,1,11);
```

What is the output of linspace(0,1,10)? And why?

Note: The piece of code after the % is treated by Matlab as a comment and so is ignored.

now set up a mathematical function, $y = x^2$.
Initially you may want to type x^2
but this will generate the error message

```
??? Error using ==> ^
Matrix must be square.
```

Why?????

Error using ^
One argument must be a square matrix and the other must be a scalar. Use POWER (.^) for elementwise power.

```
>> y = x. ^2
y = 0 0.01 0.04 0.09 0.16 0.25 0.360.49 0.64 0.81 1.00
```

Equivalently we could use `y = x.*x;`

# Setting Up Mathematical Functions

More Examples…

*Example-13* Construct the polynomial

$y = (x+2)^2(x^3+1)$

for values of x from -1 to 1 in steps of 0.1.

```
>> x = -1:0.1:1;
>> f = x+2;
>> g = x.^3+1;
>> y = (f.^2).*(g);
```

good idea to use intermediate functions when constructing complicated functions.

Or you can use the next command instead of the last 3 commands

```
>> y = ((x+2).^2).*(x.^3+1)
```

*Example-14* Construct the function $y = \frac{x^2}{x^3+1}$ for x from 1 to 2 in steps of 0.01.

```
>> x = 1:0.01:2;
>> f = x.^2;
>> g = x.^3+1;
>> y = f./g;
```

Or you can use

```
y = x.^2./(x.^3+1);
```

*Example-15* Construct the function

$$y(x) = \sin\left(\frac{x\cos x}{x^2+3x+1}\right)$$

for values of x from 1 to 3 in steps of 0.02.

```
>> x = 1:0.02:3;
>> f = x.*cos(x);
>> g = x.^2+3*x+1;
>> y = sin(f./g)
```

# Some MATLAB Specific Commands

We would make calculations where the *input can take a variety of forms*. The first command is **polyval**. This command takes two inputs, namely the coefficients of a polynomial and the values at which you want to evaluate it.

*Example-16* Evaluate the cubic

$y = x^3 + 3x^2 - x - 1$

at the points x = (1, 2, 3, 4, 5, 6).

```
% Firstly set up the points at which the polynomial
% is to be evaluated
>> x = 1:6;
% Enter the coefficients of the cubic (note that
% these are entered starting with the
% coefficient of the highest power first
>> c = [1 3 -1 -1];
% Now perform the evaluation using polyval
>> y = polyval(c,x)
    y =    2   17   50   107   194   317
```

Important Point

It is important that you remember to enter the coefficients of the polynomial starting with the one associated with the highest power and that zeros are included in the sequence.

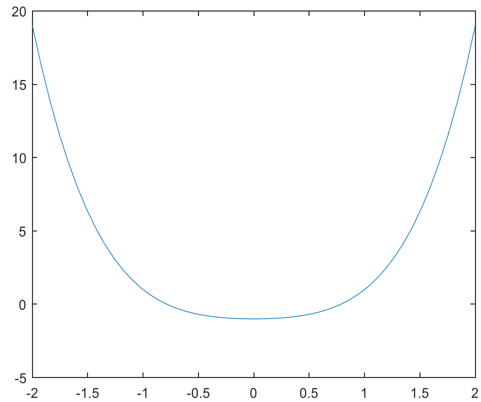
# Some MATLAB Specific Commands

We might want to **plot the results** of this calculation and this can be simply accomplished using the **plot** command.

*This produces the output*

*Example-17* Plot the polynomial $y = x^4+x^2-1$ between $x = -2$ and $x = 2$ (using fifty points).

```
>> x = linspace(-2,2,50);
>> c = [1 0 1 0 -1];
>> y = polyval(c,x);
>> plot(x,y)
```



# Example of Plotting Data

Consider the following table, obtained from experiments on the viscosity of a liquid.1 We can enter

| T (C◦) | 5 | 20 | 30 | 50 | 55 |
|---|---|---|---|---|---|
| μ | 0.08 | 0.015 | 0.009 | 0.006 | 0.0055 |

this data into Matlab with the following commands entered in the command window:

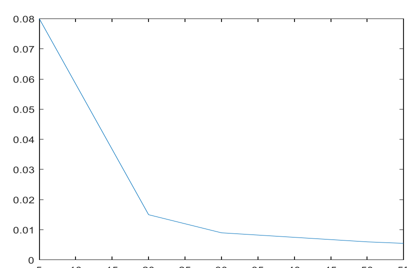
```
>> x = [ 5  20  30  50  55 ]
>> y = [ 0.08  0.015  0.009  0.006  0.0055]
```

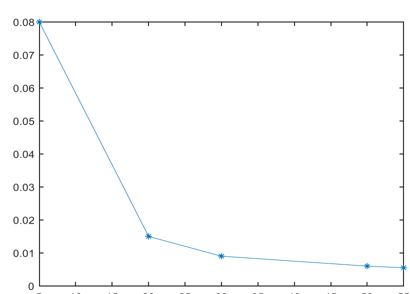
We can plot data in the form of vectors using the plot command:

```
>> plot(x,y)
```

This will produce a graph with the data points connected by lines. If you would prefer that the data points be represented by symbols you can do so. as:

```
>> plot(x,y,'-*')
>> plot(x,y,'-o')
>> plot(x,y,'-.')
```





Type help plot for more info. about plot

# Some MATLAB Specific Commands

One of the most useful commands is the **roots** to manipulate polynomials. The *input* to the routine is simply these coefficients and the *output* is the roots of the polynomial.

*Example-18* Find the roots of the polynomial $y = x^3 - 3x^2 + 2x$ using the command roots.

```
>> c = [1  -3  2  0];
>> r = roots(c)
     r =
          0
          2
          1
```

**poly** convert roots to polynomial. This takes the roots and generates the coefficients of the polynomial having those roots.

poly(r), when r is a vector, is returns a vector whose elements are the coefficients of the polynomial whose roots are the elements of r.

```
>> poly(r)
ans =     1    -3    2    0
```

# Some MATLAB Specific Commands

## 1- Looking at Variables and Their Sizes

To list the variables which are currently defined we can use the command **whos**. This will give a list of the variables which are currently defined. And the command **who** used to obtain a shorter output.
This command **whos re*** used to list certain variables only, lists the variables whose names start with **re**.

*Example-19* The following code

```
>> clear all
>> a = linspace(0,1,20);
>> b = 0:0.3:5;
>> c = 1.;
>> whos
```

gives the output

| Name | Size | Bytes | Class |
|---|---|---|---|
| a | 1x20 | 160 | double |
| b | 1x17 | 136 | double |
| c | 1x1 | 8 | double |

Grand total is 38 elements using 304 bytes

- Here we have used the **clear all** command to remove all previously defined variables.
- To look at the size of one variable we can use the command **length**, as example with the previous **length(a)** will give the answer 20.
- We note that the command **size(a)** will give two dimensions of the array, that is in this case [1 20].

# Accessing Elements of Arrays

considering a simple array x = 0:0.1:1.;. The elements of this array can be recalled by using the format x(1) through to x(11). The number in the bracket is the index and refers to which value of x we require. A convenient mathematical notation for this would be $x_j$ where j = 1, · · · , 11. This programming notation should not be confused with *x(j);* that is x is a function of j.

*Example-20* Construct the function f(x) = $x^2$+2 on the set of points x = 0 to 2 in steps of 0.1 and give the value of *f(x)* at x = 0, x = 1 and x = 2. The code to construct the function is:

```
>> x = 0:0.1:2;
>> f = x.^2+2;
% Function at x=0
>> f(1)
   ans = 2
% Function at x=1
>> f(11)
   ans = 3
% Function at x=2
>> f(21)
   ans = 6
```

Note that the three points are not f(0), f(1) and f(2)!

In this example we have noted that

$$x_j = (j-1)/10$$

and hence $x_1$ = 0, $x_{11}$ = 1 and $x_{21}$ = 2. These three indices are the ones we have used to find the value of the function.

Important Point

In MATLAB *f(j)* the value of *j* refers to the index within the array rather than the function *f(.)* evaluated at the value *j*!

*Example-21* We now show how to extract various parts of the array x.

```
>> x = linspace(0,1,10);
>> y = x(1:end);      % Whole of x
>> y = x(1:end/2);    % First half
>> y = x(2:2:end);    % Even indices only
>> y = x(2:end-1);    % All but the first & last one
```

# Accessing Elements of Arrays

*Example-22* Debug the code which is supposed to set up the function f(x) = $x^3$ cos(x + 1) on the grid x = 0 to 3 in steps of 0.1 and give the value of the function at x = 2 and x = 3.

```
x = linspace(0,3);
f = x^3.*cos x+1;
% x = 2
f(2)
% x = 3
f(End)
```

*The corrected code should be*

```
x = 0:0.1:3;  % or x = linspace(0,3,31);
f = x.^3.*cos(x+1);
% x = 2
f(21)
   ans =   -7.9199
% x = 3
f(end)
   ans =  -17.6484
```

**Homework:**
**Tasks on page 24 is homework**

# Tasks

**Task 1.2** *Calculate the value of the function*

$$y(x) = |x| \sin x^2$$

*for values of* $x = \pi/3$ *and* $\pi/6$ *(use the MATLAB command abs(x) to calculate* $|x|$*).*

**Solution 1.2** To calculate the function $y(x) = |x| \sin x^2$ we use the code:

```
x = pi/3;
y = abs(x)*sin(x^2);
```

and similarly for $x = \pi/6$. Notice care is needed with the brackets and the syntax.

**Task 1.7** *Evaluate the function*

$$y = \frac{x}{x + \frac{1}{x^2}}$$

*for* $x = 3$ *to* $x = 5$ *in steps of 0.01.*

**Solution 1.7**

```
x = 3:0.01:5;
y = x./(x+1./x.^2);
```

**Task 1.8** *Evaluate the function*

$$y = \frac{1}{x^3} + \frac{1}{x^2} + \frac{3}{x}$$

*for* $x = -2$ *to* $x = -1$ *in steps of 0.1.*

**Solution 1.8**

```
x = -2:0.1:-1;
f = 1./x;
y = f.^3+f.^2+3*f;
```

## You are welcome for

## Any Question?