LECTURE 5:

In this chapter you will learn about:

- different ways of evaluating expressions;
- solving equations symbolically and numerically;
- using MATLAB to differentiate expressions;
- using MATLAB to evaluate integrals (symbolically and numerically);
- dealing with limits.

2.1 Evaluating expressions

We previously saw a small example of the behaviour of the MATLAB Symbolic Math Toolbox when redefining variables in an expression. Recall:

```
>> syms x
>> y = 2*x
>> x = sym(6)
>> y            % still 2x
```

The subs command is useful here.

```
>> subs(y)
```

By default subs substitutes the current value of all variables into the expression. In this case, x was 6 so it substitutes 6 in for x to give y = 12.

The subs command has a more specific form subs(expr, old, new) which replaces all occurrences of the expressions old in expr by the expression new. For example, suppose we want to evaluate the expression x^2+3x-2 at $x=1$. This is achieved by using the commands

```
>> syms x
>> y = x^2 + 3*x - 2;
>> subs(y, x, 1)
```

A useful feature is that y itself has not changed, so that if we now wish to evaluate y at a different value of x then this is easily done:

```
>> % first, note y unchanged
>> y
>> subs(y, x, 10)
>> subs(y, x, sym(pi))
>> y2 = subs(y, x, atan(x))
```

Exercise 2.1 In each of the following, evaluate the expression at the given value.

i) $x^3 - 3x^2 + 2x - 1$, at $x = 5$;

ii) $\sin x \cos^3 x$, at $x = \pi/4$;

iii) $\ln\left(u + 1 + \sqrt{u^2 - 3}\right)$, at $u = 2$. □

A single call of `subs` can be used to perform several substitutions. As an example, suppose that we wish to evaluate $r = \sqrt{x^2 + y^2 + z^2}$ at the point $(x, y, z) = (1, 2, 3)$. This is achieved by typing

```
>> syms x y z
>> r = sqrt(x^2 + y^2 + z^2)
>> A = subs(r, [x y z], [sym(pi) 10 20])
```

The `[]` notation indicates a vector: more on this later. Recall that you can use `double()` to evaluate to a decimal approximation:

```
>> double(A)
```

Sometimes after using `subs` you might need `simplify` to encourage MATLAB to cleanup the result.

That completes this brief introduction to evaluation. We now move on to look at solving equations. The main aim is to introduce `solve()`, which can be used to solve equations, inequalities, and systems of these. First, recall that "`=`" is used in MATLAB to express assignment. To express equality we use "==".

```
>> a = sym(6)  % assign 6 to a
>> a == 6      % returns 1 for true
>> a == 7      % returns 0 for false
>> a < 10      % returns 1 for true
>> a <= 6      % returns 1 for true
>> a ~= 5      % returns 1 for true
>> A ~= 6      % returns 0 for false
```

These are mostly self-explanatory except for "`~=`" (tilde followed by equals) which means "not equal to".

2.2 The `solve` command

2.2.1 Solving equations symbolically

The `solve` command can be used to rearrange simple algebraic expressions to arrive at a new expression. For example, the solution of the equation

$$2x + 3 = 0 \tag{2.1}$$

for x can be obtained using the Symbolic Math Toolbox as follows:

```
>> syms x
>> eq = 2*x + 3 == 0
>> solve(eq, x)
```

Again please note the different uses of = and == here: eq is *assigned* the expression of Equation (2.1) which happens to contain an *equality* operator. Messing this up makes subtle bugs that can be hard to track down.

Note that the solution, $-\frac{3}{2}$, is not assigned to x. You can do that yourself if you wish:

```
>> x
>> x = solve(eq, x)
>> eq
>> eq2 = subs(eq)
```

Suppose that we wish to solve the simultaneous equations

$$x + y = 2, \qquad -x + 3y = 3.$$

This is done as follows:[1]

```
>> clear
>> syms x y
>> eq1 = x+y == 2;
>> eq2 = -x+3*y == 3;
>> [xsoln, ysoln] = solve(eq1, eq2, x, y)
```

An alternative:

```
>> sol = solve(eq1, eq2, x, y)
>> xsoln = sol.x
>> ysoln = sol.y
```

where sol is something called a structure: roughly speaking it can contain "sub-variables" known as fields.

What about problems with multiple solutions?

```
>> clear
>> syms x
>> eqn = x^2 == -25
>> sol = solve(eqn, x)
>> sol(1)
>> sol(2)
```

Here sol is a vector of two solutions, $-5i$ and $5i$.

Exercise 2.2 Find the solutions of the following equations:

(a) $x^2 - x - 2025 = 0$, (b) $x^3 - 6x^2 - 19x + 24 = 0$,

(c) $2x^4 - 11x^3 - 20x^2 + 113x + 60 = 0$. □

Exercise 2.3 Use the solve command to find the point of intersection, in the (x, y)-plane, of the two lines

$$ax + by = A, \quad cx + dy = B.$$

Having found the intersection, find an expression for the distance of the point of intersection from the origin. □

Note in the previous exercise that the Symbolic Math Toolbox made various assumptions (for example that $ad - bc \neq 0$). Some other systems (for example MuPAD in its worksheet mode) are very good at finding a wider variety of solution possibilities: this can be useful (and/or annoying) depending on what you're trying to accomplish.

2.2.2 Assumptions

Sometimes one makes assumptions about equations and variables. For example, you might be interested only in the real roots of a quadratic. Or you might have a mathematical model where you know a particular variable must stay positive (say it represents a positive physical quantity like concentration or density). The Symbolic Math Toolbox allows you to add assumptions to symbolic variables.

```
>> syms x
>> assume(x, 'real')
>> ineq = x^2 <= 25
>> assumeAlso(ineq)
>> assumeAlso(x > 0)
```

To list the current assumptions, type:

```
>> assumptions
```

You can remove the assumptions on a particular variable using

```
>> x = sym('x', 'clear')
```

Recall that if you want to clear everything and start again, the usual approach is the `clear` command. But it is worth mentioning that this will not remove assumptions from variables.[2] The `clear all` command will do the right thing.

Exercise 2.4 Define $z = x + iy$ where x and y are symbolic variables. Take the complex conjugate of z. Now modify your code to assume that both x and y are real and run your code again. □