LECTURE 3: USING SYMBOLS IN MATLAB

Numbers are nice and all, but the `sym()` command makes more interesting things possible because symbolic expressions can contain symbols:

```
>> x = sym('x')
>> y = asin(x^3)
>> z = sin(y)
```

Another example:

```
>> x = sym('x')
>> y = sym('y')
>> f = (x^3 + x^2) * sqrt(y) * exp(x)
```

In fact, there is a shortcut—`syms`—for defined symbolic variables:

```
>> syms x y z a b c r s t
>> f = sin(a*x) * sqrt(tan(r*t-x) + sin(t-1) * tanh(s^2*y))
```

Consider

```
>>   a = sym(6)
>>   b = sym('2*a')
>>   c = sym(2)*a
```

What happened? The result in `b` might not be what you wanted! This suggests you should avoid putting expressions inside the quotes and instead build them out of symbolic variables or symbolic numbers.[1] For example, rather than writing `f = sym('a * x + b')`, it's probably better to do

```
>>   syms a x b
>>   f = a*x + b
```

In Section 2.1, we will learn about the `subs` command which will help push assigned variables into expressions.

Variable types and casting

We have encountered two types ("classes") of objects so far: `doubles` which store numbers with a finite precision and `syms` which store symbolic expressions. It will often be important to keep track of which is which. The `whos` command can help.

```
>> clear
>> a = sym('6/9')
>> b = 6/9
>> whos
```

Now we can ask some useful questions about what happens when you combine objects.

Exercise 1.2 Use MATLAB to evaluate the following exactly:

```
>>  a = sym(2/3)
>>  b = 5/3
>>  c = a + b
>>  d = b*a
>>  e = a^b
>>  f = (2/3) ^ (sym(5/3))
>>  e - f
>>  whos
```

From these results, what does MATLAB typically do when it performs a binary operation which combines a `double` and a `sym`? This is known as casting. □

So very roughly speaking, as long as some symbols in your expression are of class `sym`, you can expect the result to be symbolic too. This is good news for entering complex expressions, for example:

```
>> x = sym('x');
>> f = 512*x^4 + 13/10*x^2 + 256*x^(2/3)
>> % because the alternative is just horrible:
>> g = sym(512)*x^(sym(4))  +  sym(13)/sym(10)*x^sym(2)  +  ...
>>     sym(256)*x^(sym(2)/sym(3))
>> f - g      % but they are the same in the end
```

The previous example also shows how to split a very long line in two: simply interrupt the line with "..." (three full stops) and continue on the next. You'll also note that we have been using the character "%" to comment our code. This is a good idea to help the reader understand what the code does. As you've probably figured out, MATLAB ignores these comments when executing the code.

Sometimes it's hard to tell if an expression like `f` above was entered correctly; the output often looks as bad—or worse—than the input. Try this:

```
>> pretty(f)
```

That will render the expression in way that you might find easier to decipher.[2] Similarly, `latex(f)` will output code that can be pasted into LaTeX, which is the standard tool most mathematicians use for typesetting mathematics.[3]

Just as `sym()` converts to symbolic expressions, `double()` converts to double-precision numerical values.

Exercise 1.3 Use MATLAB to evaluate the following exactly:

$$19 \times 99, \quad 3^{20}, \quad 2^{1000}, \quad 25!, \quad \frac{3}{13} - \frac{26}{27}.$$

Now calculate the decimal value of following:

$$\frac{21}{23}, \quad 17^{1/4}, \quad \frac{1}{99!}, \quad 0.216^{100}.$$

Finally, explain what MATLAB does in each of these cases:

```
>> a = 2/3
>> b = sym(2)/sym(3)
>> c = double(b)
```

□

MATLAB has some names reserved for constants, some of which are shown in the following table

`pi`	$3.1415\ldots$
`i` (or "1i")	$\sqrt{-1}$
`inf`	∞
`nan` (Not a Number)	$\frac{0}{0}$, $\sin(\infty)$

The number e, that is $2.7182\ldots$, can be entered as

```
>> double_e = exp(1)
>> symbolic_e = exp(sym(1))
>> log(symbolic_e)
```

Exercise 1.4 *Careful* it's not `sym(exp(1))`: try that and see what happens. At least in MATLAB version R2013b, it fails: can you explain why?

As stated above, MATLAB denotes the square root of -1 by `i` (actually `1i` or `j` work too). Suppose that we wish to express $(2+3i)^4$ in the complex form $a+bi$, where a and b are real:

```
>> (2 + 3*i)^4
```

Some MATLAB programmers don't like this because they want to use `i` for a variable. Thus it is very common to instead write:

```
>> (2 + 3*1i)^4
>> % or
>> (2 + 3i)^4
```

These examples used class `double` but symbolic complex numbers work too:

```
>> syms x y
>> z = x + 1i*y
>> sin(z)
```

You might have a sense of unease at this point: you and I might be tacitly assuming `x` and `y` are real but of course MATLAB doesn't know that. Indeed the complex conjugate of $z = x+iy$ is $\bar{z} = x - iy$ but MATLAB says:

```
>> syms x y
>> z = x + 1i*y
>> conj(z)
```

We will see how to fix this later using assumptions in Section 2.2.2.

Exercise 1.5 Use MATLAB to express the following in the approximate decimal form $a+bi$

$$(5-8i)^2, \quad i^{-1}, \quad i^2, \quad i^{\frac{1}{2}}, \quad \frac{1+i}{5+2i}, \quad \text{and} \quad \left(\frac{4}{3}+\frac{2i}{5}\right)^4.$$

Exercise 1.6 Use symbolic complex numbers to verify that $(a+bi)^2 = a^2 - b^2 + 2abi$. (Hint: one approach would be to construct the left-hand side and the right-hand side. Sometimes the `simplify()` command will give MATLAB a nudge...). □

1.2.7 Built-in functions and help

As we have seen already, MATLAB has many in-built functions, many of which are fairly obvious. The table below lists some of those that you should be familiar with.

`cos`, `cosh`	cosine and hyperbolic cosine
`sin`, `sinh`	sine and hyperbolic sine
`tan`, `tanh`	tangent and hyperbolic tangent
`sec`, `sech`	secant and hyperbolic secant
`csc`, `csch`	cosecant and hyperbolic cosecant
`cot`, `coth`	cotangent and hyperbolic cotangent
`exp`	exponential
`log`	natural logarithm
`log10`	base-10 logarithm
`abs`	absolute value (and magnitude of complex number)
`sqrt`	square root

MATLAB has extensive built-in help which you can access from the command prompt:

```
>> help cos
```

One important note is that help often gives you the function that operates on `double` values. For example, with the command above, you'll probably see

```
Overloaded methods:
   sym/cos
```

If you are working with symbolic expressions, you can access the symbolic specific help:

```
>> help sym/cos
```

(An object-oriented programmer would say "cos is overloaded for `sym` input".) Other commands have associated help too:

```
>> help clear
>> help syms
>> help whos
```

You can also search for keywords:

```
>> lookfor hyperbolic
```

MATLAB has more detailed hyperlinked help accessed with the `doc` command (and also available online).

Exercise 1.7 Use MATLAB to evaluate the following:

$$\tan^2(\pi/3),\ \sec(\pi/6),\ 1+\cot^2(\pi/4),\ \operatorname{cosec}^2(\pi/4)\ e^{3\ln 4}.$$

Exercise 1.8 Use the help facility to learn how to compute the binomial coefficient ${}^nC_r$ in MATLAB. Evaluate ${}^{10}C_4$ and ${}^{250}C_{12}$. Do this both symbolically and numerically with double precision. □

Exercise 1.9 Evaluate $\arcsin(1/2)$. Evaluate $\sec(\arctan x)$ in terms of x. □