Functions

Table of Contents

What is a Function?

There are two different ways of thinking of functions in calculus and hence in Matlab. One is as a symbolic expression, meaning a bunch of symbols. The other is as a function, meaning a rule. The difference between these can be captured by looking at the difference between the expressions `f=x^2` and `f(x)=x^2`. In the former we're saying that `x` is a symbol and `f` is just the square of that symbol. In the latter we're saying `f` is a function into which `x` is plugged.

Matlab treats these quite differently. Each has advantages and disadvantages.

Symbolic Expressions

We start by defining a letter as a symbolic variable and then the function as a symbolic expression. Something like:

```
syms x;
f=x^4+7*x^2+x+exp(2*x)
```

```
f =

x + exp(2*x) + 7*x^2 + x^4
```

At this point it's important to note that `x` is to be treated symbolically (as a symbol) and so is `f`. In other words Matlab specifically treats `x` as a symbol and `f` as just another symbol which happens to be equal to `x^4+7*x^2+x+exp(2*x)`.

In any case we can now do some things with `f`.

Plugging into Symbolic Functions

We can substitute a constant for the symbol `x`:

```
subs(f,x,3)
```

```
ans =

    5.504287934927352e+02
```

Derivatives of Symbolic Functions

We can find the first derivative using either of these:

```
diff(f,x)
```

```
ans =

14*x + 2*exp(2*x) + 4*x^3 + 1
```

or the following where the `1` means the first derivative.

```
diff(f,x,1)
```

```
ans =

14*x + 2*exp(2*x) + 4*x^3 + 1
```

And the second derivative with either of these:

```
diff(diff(f,x),x)
```

```
ans =

4*exp(2*x) + 12*x^2 + 14
```

Which is, as defined, the derivative of the derivative, or:

```
diff(f,x,2)
```

```
ans =

4*exp(2*x) + 12*x^2 + 14
```

Which is conceptually straight to the second derivative.

Derivatives and Plugging

We can easily take the third derivative then plug in `0.1`:

```
subs(diff(f,x,3),x,0.1)
```

```
ans =

  12.171222065281359
```

Note that this is very different than what people often do, which is:

```
diff(subs(f,x,0.1),x,3)
```

```
ans =

0
```

Do you see what this does? This *first* substitutes in `x=0.1` (yielding a constant) and then takes the derivative. Not what we wanted at all.

We can integrate from `1` to `3` by:

```
int(f,x,1,3)
```

```
ans =

(exp(2)*(exp(4) - 1))/2 + 1696/15
```

Or just find the standard antiderivative as follows. As before note that Matlab does not bother with the +C but you should keep in mind that really it should be there.

```
int(f,x)
```

```
ans =

exp(2*x)/2 + x^2/2 + (7*x^3)/3 + x^5/5
```

Function Handles

The preferred way to define functions is with the `@` operator. Technically speaking this is creating a function handle but for many practical purposes we can gloss over the difference. Consider the following example:

```
f = @(x) x^2-x
```

```
f =

    @(x)x^2-x
```

This tells Matlab to create a function for which `x` is the variable and for which the rule is `x^2-x`. This function can be treated in some ways very much like an inline function. For example the following do as you'd expect:

```
f(3)
fzero(f,2)
```

```
ans =

     6


ans =

     1
```

Function M-Files

This last way of defining functions will be introduced later.

Published with MATLAB® 8.0