# Chapter 13 - Function Handles

## Table of Contents



So far we have seen functions defined symbolically. Matlab also has functions which are defined as function handles. These are used for certain Matlab commands as we'll see and are used when we need to pass functions as parameters to function m-files as we will do in the last chapter.

# Function Handles

Function handles are defined using the `@` operator. Consider the following example:

```
f = @(x) x^5-3*x
```

```
f =
  function_handle with value:
    @(x)x^5-3*x
```

This tells Matlab to create a function for which `x` is the variable (this is what the `@(x)` does) and for which the rule is `x^5-3*x`. Here are some things we can do with function handles. We can plug things in:

```
f(3)
```

```
ans =
   234
```

# Differentiation and Integration

Differentiating is a bit obscure. We cannot simply do `diff(f)` because `diff` doesn't work as-is on function handles. However there is a workaround. If we define `x` as symbolic then Matlab will accept `diff(f(x))`. What's happening here is that `f` is a function handle but `f(x)` is symbolic because symbolic `x` is plugged in:

```
clear all;
f = @(x) x^5-3*x;
syms x;
diff(f(x))
```

```
ans =
5*x^4 - 3
```

and for a second derivative:

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

ans =
20*x^3
```

Integration works the same way in that `int(f)` and `int(f,1,2)` will not work but provided `x` is symbolic:

```
clear all;
f = @(x) x^5-3*x;
syms x;
int(f(x))

ans =
(x^2*(x^4 - 9))/6
```

and

```
int(f(x),1,2)

ans =
6
```

will work fine, as will:

```
int(f,x,1,2)

ans =
6
```

# Numerical Integration of Function Handles

Now that we have function handles we can use `integral` to do numerical integration. However there's another caveat. The Matlab `integral` command does its work with vectors and consequently for reasons we will not go into right now we must replace `*` and `/` and `^` with `.*` and `./` and `.^`. For example:

```
clear all;
f = @(x) exp(x.^2);
integral(f,1,2)

ans =
   14.9900
```

# The `matlabFunction` Command

If you're confused or frustrated there is a command, `matlabFunction` (notice the weird capitalization) which converts symbolic functions into function handles. This command is named in a totally irresponsible fashion and should have the term *handle* in its name but it doesn't.

```
clear all;
syms x;
f = matlabFunction(exp(x^2));
```

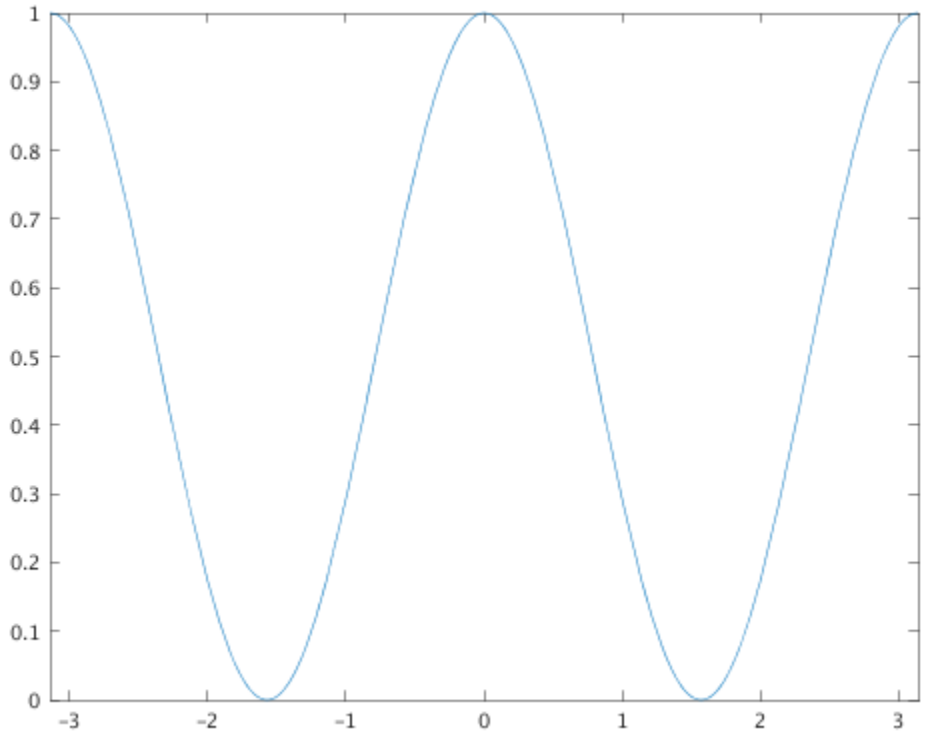
```
integral(f,1,2)

ans =
   14.9900
```

# Plotting Function Handles

The `fplot` command however works fine:

```
clear all;
syms x;
f = matlabFunction(cos(x)^2);
fplot(f,[-pi,pi])
```




# More Complicated Calculations - IMPORTANT!

It's important to realize that if `f` is a function handle and `x` is symbolic then `f(x)` and consequently things like `diff(f(x))` are symbolic too. This means we need to use `subs` in places. For example if we wish to find a second derivative of a function handle and *then* plug in:

```
clear all;
syms x;
f = matlabFunction(1/x^3+log(x-2));
subs(diff(f(x),2),3)

ans =
```

```
-77/81
```

Take a moment to understand what this does, keeping in mind the fact that `x` is nested inside `f(x)` which is inside `diff` which is inside `subs`.

Similarly suppose we wanted to start with a function handle, differentiate, divide by `x` and then plug in:

```
clear all;
syms x;
f = matlabFunction(x^3+2/x);
subs(diff(f(x))/x,5)
```

```
ans =
1873/125
```

*Published with MATLAB® R2016b*