Definite Integrals

Despite your training in Calculus, **most** definite integrals cannot be computed exactly, and must be approximated numerically. You learned a number of rules for doing this: the trapezoidal rule and Simpson's rule come to mind. These are tedious computations that are best left to a machine.

Octave has an algorithm for computing numerical approximations of definite integrals. Suppose we want to compute

$$\int_0^3 e^{1-x^3}\,dx.$$

Here's how to find a numerical approximation with Octave

```
octave-3.2.3:3> f=@(x) exp(1-x.^3)
f =

@(x) exp (1 - x .^ 3)

octave-3.2.3:4> quad(f,0,3)
ans =  2.4274
```

The first argument to quad is the function to integrate, the second is the lower bound of integration, and the third is the upper bound of integration.

Area-Under-The-Curve Functions

In this course we keep running into area-under-the-curve functions. For example, consider

$$F(x) = \int_3^x \sin(\sqrt{1+s^2})\,ds$$

This is an unusual looking function. The value $F(3) = 0$ since $\int_3^3 \sin(\sqrt{1+s^2})\,ds = 0$. The value $F(4)$ is $\int_3^4 \sin(\sqrt{1+s^2})\,ds$. Although we cannot compute this value exactly, we can obtain a numerical approximation with Octave using the quad function.

```
octave-3.2.3:14> f=@(x) sin(sqrt(1+x.^2));
octave-3.2.3:15> quad(f,3,4)
ans = -0.46060
```

If we need to evaluate the area-under-the-curve function at a lot of different points, we can make an Octave function to help out:

```
octave-3.2.3:26> f=@(x) sin(sqrt(1+x.^2));
octave-3.2.3:27> F=@(x) quad(f,3,x);
```

```
octave-3.2.3:28> F(3)
ans = 0
octave-3.2.3:29> F(4)
ans = -0.46060
```

Unfortunately, we cannot feed `F` a vector:

```
octave-3.2.3:30> F([0,1,2,3])
warning: implicit conversion from real matrix to real scalar
ans = -2.2656
```

Frankly, this should have been an error message. Octave is complaining that it can't have a vector as the upper bound of integration: `quad(f,0,[0,1,2,3])` doesn't make sense. To make a vector containing `[F(0),F(1),F(2),F(3)]` we can use the `arrayfun` command. Consider:

```
octave-3.2.3:32> x=[0,1,2,3]
x =

   0   1   2   3

octave-3.2.3:33> arrayfun(F,x)
ans =

  -2.26556  -1.36100  -0.41686   0.00000
```

versus

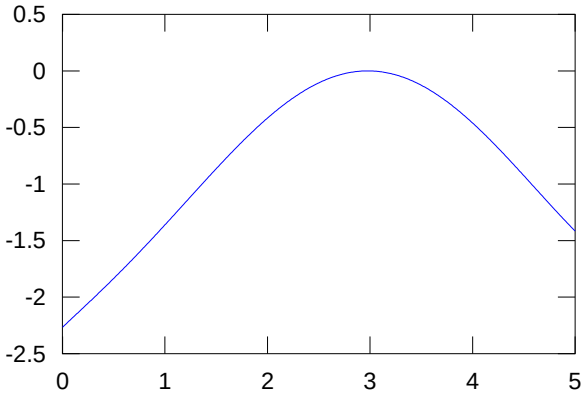
```
octave-3.2.3:34> [F(0),F(1),F(2),F(3)]
ans =

  -2.26556  -1.36100  -0.41686   0.00000
```

In general, the result of `arrayfun(f,x)` is a vector of the same size as `x` with the function `f` applied to each entry of `x`. This is helpful for plotting area-under-the-curve functions:

```
octave-3.2.3:37> x=0:0.01:5;
octave-3.2.3:38> plot(x,arrayfun(F,x))
```

which generates:



Finding Roots

Octave has a function that can (often) find a numerical approximation of a solution of an equation of the form

$$f(x) = 0.$$

For example, suppose we want to find the solution of the equation

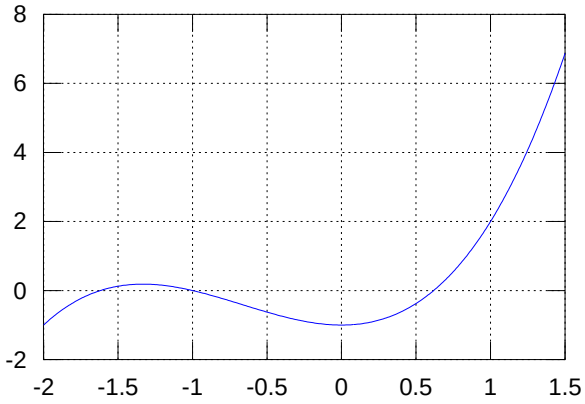
$$x^3 + 2x^2 = 1.$$

This equation can be rewritten as

$$x^3 + 2x^2 - 1 = 0.$$

So we want to solve $f(x) = 0$ where $f(x) = x^3 + 2x^2 - 1$. It will be helpful to look at a graph of f.

```
octave-3.2.3:76> x=-2:0.01:1.5;
octave-3.2.3:77> plot(x,f(x))
octave-3.2.3:78> grid
```



This is a cubic polynomial, and we see that there are three roots, one each near -1.6, -1, and 0.6. To find a root we use `fzero`, which takes two arguments: the function, and an initial guess for the root. For example, to find a root near $x = 0.5$ we use:

```
octave-3.2.3:80> f=@(x) x.^3 + 2*x.^2-1
f =

@(x) x .^ 3 + 2 * x .^ 2 - 1

octave-3.2.3:81> fzero(f,0.5)
ans =  0.61803
```

You can see more digits in your answers by adjusting the formatting:

```
octave-3.2.3:82> fzero(f,0.5)
ans =  0.61803
octave-3.2.3:83> format long
octave-3.2.3:84> fzero(f,0.5)
ans =  0.618033988749895
```

The other two roots can be found as follows:

```
octave-3.2.3:84> fzero(f,-0.5)
ans = -1
octave-3.2.3:85> fzero(f,-1.5)
ans = -1.6180
```

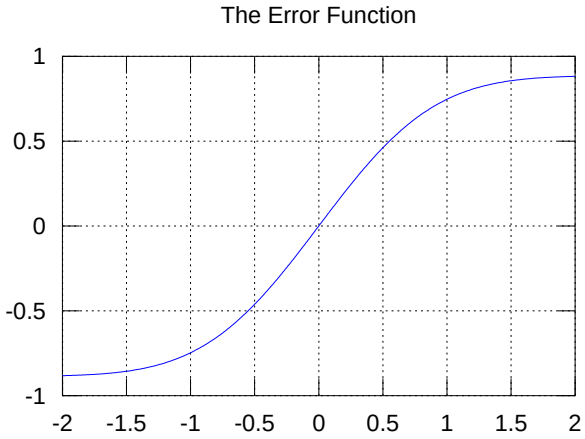
Making Scripts

A script in Octave is a sequence of commands to be run in order. For example, we might want to make a nice graph of the function $F(x) = \int_0^x e^{-s^2}\, ds$ over the interval $-2 \le x \le 2$. The commands are:

```
f = @(x) exp(-x.^2);
F = @(x) quad(f,0,x);
x=-2:0.01:2;
plot(x,arrayfun(F,x));
title('The Error Function')
grid
```

If you save these commands in a script you can make changes to them later if, for example, want to adjust the title or change the domain. To do this, you'll need a text editor. On Windows, Notepad will do. On the Mac, you can use TextEdit. Open your application and enter the commands above. If you are using the TextEdit, you'll also need to select "Make Plain Text" from the "Format" menu (or change this preference permanently under "Preferences"). Now save the document as `myplot.m` in a location that Octave knows about (via its path). The directory where you saved your file `df.m` will work, for example. Now within

Octave you can simply type `myplot` and your plot will appear. If you adjust the commands in the file `myplot.m` and rerun `myplot` in Octave, a new plot will appear. The plot from these commands looks as follows:



Exercises

Exercise 1:

Compute $\int_{-3}^{5} x\cos(x)\,dx$.

Exercise 2:

Let $F(x) = \int_0^x \sqrt{1+\cos(s)}\,ds$. Compute $F(2)$. Graph this function for $2 \le x \le 4$.

Exercise 3:

Let $F(x) = \int_0^x e^{-x^2}\,ds$. Find an x such that $F(x) = 0.8$.

Exercise 4:

Make a script that plots the function $F(x)$ from Exercise 3 on the domain $-1 \le x \le 2$, puts a blue square on the location $(x, F(x))$ where x is the solution of $F(x) = 0.8$, and adds a grid. Run your script to make sure it works. Then edit it to add a red square at the location $(x, F(x))$ where x is the solution of $F(x) = 0.4$

Your final plot should look like