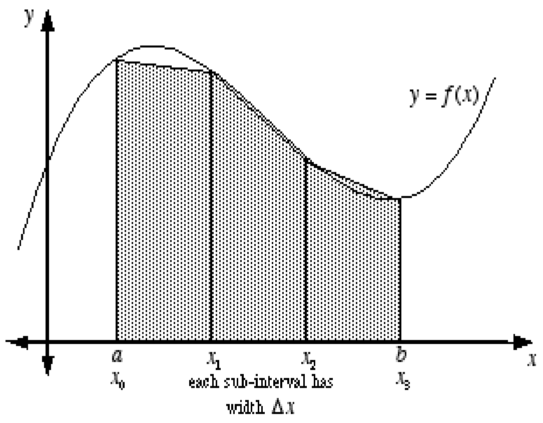
NUMERICAL INTEGRATION AND FUNCTION FUNCTIONS

The arguments of functions previously considered had constants, or vector or matrix variables as arguments. In this section we will see how to set up a function to process another function as one of its arguments, as well as, explore the integration process with a computer.

% You should recall basic numerical integration concepts including simple rectangular approximations, as well as, the trapezoidal rule. The next function will use this rule to calculate the definite integration of any well behaved function.
NOTE: Any well behaved function means you do not have to memorize the indefinite integral of a function like the integral of cos() is sin() etc or look in ancient texts for worked out integrals of exotic function which was done a great deal in the 19th century.
Let's review the Trapezoidal rule, study the figures below to the left .
Remember integral = area under the curve. We want the area from x = a to x = b in this case. We have divided the wanted area into 3 parts. So the interval between the x values is simple $\Delta x=(b-a)/3$. Each area is a rectangle plus triangle we call the combination a trapezoid.
A magnified view of two sections is below.

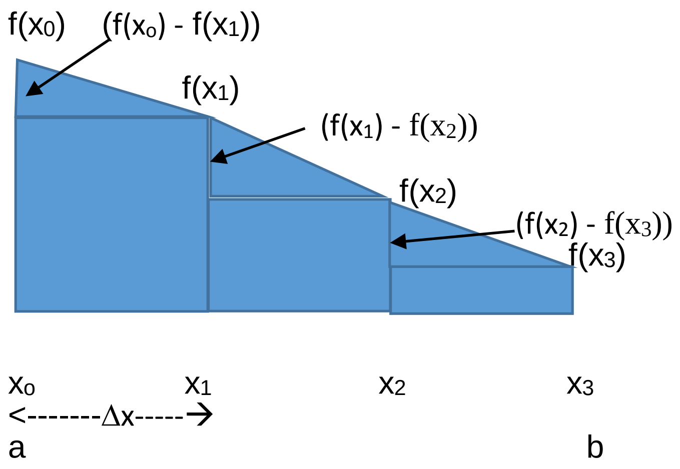
Area = area rectangle +area of triangle which becomes for the magnified view
Area = area rect + area of triabgle + next rect etc
Area= $\Delta X*f(x_1)+1/2\ \Delta X*(f(x_o) - f(x_1))+ \Delta X*f(x_2)+ 1/2\ \Delta X*(f(x_1) - f(x_2))\ \Delta X*f(x_3)+ 1/2\ \Delta X*(f(x_2) - f(x_3))$
$= \Delta X*\{ f(x_o)/2+ f(x_1)- f(x_1)/2+ 1/2f(x_1) + f(x_2)- f(x_2)/2+ 1/2f(x_2)+ f(x_3)- f(x_3)/2\}$
$= \Delta X*\{ f(x_o)/2+ f(x_1)+ f(x_2)+ f(x_3)/2\}$ = formula under the curve!



The area of the trapezoids (shaded) approximately equals the area bounded by $y = f(x)$.

$$\int_a^b f(x)dx = \frac{\Delta x}{2}[f(x_0) + 2f(x_1) + 2f(x_2) + f(x_3)].$$

=



Formally, we get for n divisions which is best. The more divisions,n, the better the integral value compared to the definite integral of known functions, except too many could create problems.

1. Partition $[a,b]$ into the set $\{x_0, x_1, x_2, x_3, ..., x_n\}$ so that there are n sub-intervals of equal width.

2. The integral $\int_a^b f(x)dx$ is estimated by adding the areas of all the trapezoids as illustrated below.
The width Δx of each sub-interval is given by $\Delta x = \frac{b-a}{n}$.

$$\int_a^b f(x)dx = \frac{\Delta x}{2}[f(x_0) + 2f(x_1) + 2f(x_2) + 2f(x_3) + \cdots + 2f(x_{n-1}) + f(x_n)]$$

Note: The larger the value of n, the better the approximation. Also, Simpson's rule generally gives a better approximation than the trapezoid rule.

TRAPEZOIDAL INTEGRATION PROGRAM (example of a function of a function)
We now construct an m-file to do trapezoidal integration.
» type area1

% This program estimates area under curve using trapezoids
% USING A SPECIFIED n number of trapezoids, over the limits a to b the function func!

```
>> type area1
function y=area1(func,a,b,n)
total=0;
base=(b-a)/n;
for k=2:n
  xk= a+base*(k-1);
  total =total +feval(func,xk);
end
y = 0.5 .* base .* (feval(func,a) + 2.0 .* total + feval(func,b));
```

func is the name of the function we will take the integral of
% The special function **feval** as defined below permits us to use a function name when we have a function that needs another function as an argument.

» help feval

```
FEVAL        Function evaluation.
        If F is a string containing the name of a function (usually
        defined by an M-file), then  FEVAL(F,x1,...,xn)  evaluates
        that function at the given arguments.  For example,
        F = 'foo',  FEVAL(F,9.64) is the same as  foo(9.64).
        FEVAL is usually used inside functions which have the names
        of other functions as arguments.
```

So the function we are going to integrate must be passed the FEVAL as a
String.
% lets start with the built in function   sin()
Using area1 to evaluate the integral of sin() between 0 and pi
DO THIS NOW IN CLASS! The answer is ?
Remember the trapezoidal rule is a basic but primitive calculation we see this with
The answers.
Taking 20 divisions first then a 100 improves the results.

```
>> area1('sin',0,pi,20)
ans =    1.9959

>> area1('sin',0,pi,100)
ans =    1.9998
```

%Since we are passing a function or our function **area1 to be evaluated**
We say area1 is said to be a **function function**.

Another example is to use area1 to evaluate a function we define.
Suppose we define an exponential function called **fexp** as

```
» type fexp

function y = fexp(x)
y=4* exp(-x);
```

% We now can get the integral of this function using our area1 function which needs the name of the function to integrate as well as the limits on the x-axis as well as how many trapezoids to use. So for examples

```
» area1('fexp',0,1,5)
ans =    2.5369

» area1('fexp',0,1,10)
ans =    2.5306

» area1('fexp',0,1,50)
ans =    2.5286

» area1('fexp',0,1,100)
ans =    2.5285
```

%Here we have been getting answers in 4 decimal accuracy. As you know MATLAB actually does calculations to a high precision (double precision) in memory. We can increase the number of decimals, we output, by using the MATLAB **format** command. As in:

```
» format long
```

% Then when we run a program we get, for example

```
» area1('fexp',0,1,100)
ans =   2.52850330596441
```

% We note an, we can consider an exact value to the integral can be calculated and to electronic calculator accuracy, my Iphone! **2.52848223514231**
our answer with 100 subdivisions of the area only agrees to 3 decimals in long form.
This is because our integral function uses the trapezoidal approximation. We can do a lot Better.

Matlab has more advance built in functions.
Let us focus on the intrinsic routine **quad()**

help quad
quad Numerically evaluate integral, adaptive Simpson quadrature.
Q = quad(FUN,A,B) tries to approximate the integral of scalar-valued function FUN from A to B to **within an error of 1.e-6 using recursive adaptive Simpson quadrature.** FUN is a function handle. The function Y=FUN(X) should accept a vector argument X and return a vector result Y, the integrand evaluated at each element of X.

Q = quad(FUN,A,B,TOL) uses an absolute error tolerance of TOL instead of the **default, which is 1.e-6**. **Larger values of TOL result in fewer function evaluations and faster computation, but less accurate results.** The quad function in MATLAB 5.3 used a less reliable algorithm and a default tolerance of 1.e-3.
Etc.
NOTE. Some advanced functions change when new versions of Matlab become available.

Use **help quad** to see other integration schemes MATLAB has.

% redoing the sin() function example with quad gives us a much better ansew to the exact value of 2 for the integration from 0 to pi.

```
>> quad('sin',0,pi)
ans =    2.0000

>> format long
>> quad('sin',0,pi)
ans =   1.999999996398431
```

% Note again above the value from my ipone to the integral of **fexp() 2.52848223514231**
  Compare the answers below with this last value.
RETURNING FOR COMPARISON TO LESS DECIMALS

```
» format short

» q=quad('fexp',0,1)
q =    2.5285

» format long
» q=quad('fexp',0,1)
q =   2.528482235687844
```

% agreeing to **9 decimals** to the iphone calculator value

Even picking a large tolerance gives a good value since the routine is very robust in its convergence.

```
>>quad('fexp',0,1,10)     ans    2.528482240821508
    % How good is this one?
```

% we see the last is best but the others are not bad
Noting again the value for comparison the trapezoidal rule.

```
» area1('fexp',0,1,100)
ans =   2.52850330596441   a 3 decimal match
```

We can push the trapezoidal area1 like having 10,000 trapezoids get a better matching value. **After a while one has to wonder how good is our calculator value?**

```
>> area1('fexp',0,1,10000)
ans =   2.528482237421307
```

% In class let’s look over the quad.m files
  with the 'type quad' and note how a professional mathematical package is done!
% Be sure to check out in your careers the many numerical integration and differential functions Matlab provides.
Run   the “demo” to see the wealth of mathematical power you have with MATLAB

 >> demo

LABORATORY TASKS

Show all work, especially any files you construct as well as the executions.

41. (2 pts) Try to get an exact value of the area with your calculator of the function above "fexp" between 0 and 1. What is this value and how many decimal places does the quad evaluation agree?

42. (10 pts) Recall function humps

```
» type humps
function y = humps(x)
%       Humps(x) is a function used by QUADDEMO and ZERODEMO.
y = 1 ./ ((x-.3).^2 + .01) + 1 ./ ((x-.9).^2 + .04) - 6;
```

a. Write a short m-file that will call upon the function

And integrate it from 0 to 1 both by **first constructing the function of function area1.m** as above, using 100 trapezoids and then

b. calling on the built in quad function to get the area.
c. The program also plots humps from 0 to 1 very finely with 100 values of x.

Looking at the graph

d. WHICH OF THE two methods gave the best answer? Why or why not?

Show all work!

43(10 pts) . Integration fun.

a. -Set up a function file for y(x)=34 exp(-0.2x) sin(3x+pi/8) call it whatever you want.

In an one m-file do the following

b. -Integrate this function from 0 to 30 using the quad() functions.
c. - next use **area1()** to integrate this function.
d. - Plot the function over the interval in steps of 0.01: Do a rough estimate of the area from the plot.
e. -Compare the integrations over the same interval. Which is probably the best answer. Why? Or why not

44. (8 pts) Write a **function function** that sums all values of the function in 43 y(x)=… for each value in the interval -8 to +8 with steps of .1. Note: This function called SUMFUN calls the function as an argument and uses **feval()** to get its answers (SEE THE area1 for feval use). Then run an m-file containing this function that gets the sum as specified.

45. (8 pts) EXTRA CREDIT

Construct a **function function** that calculates the area from "a" to "b" when it divides any curve into "n" rectangular boxes which it sums up to get the area.
Run it for the sin() as above as well as fexp() as above. Comment on its accuracy!
For the sin() keep increasing n to see what happens. (0 to pi range). Comment!