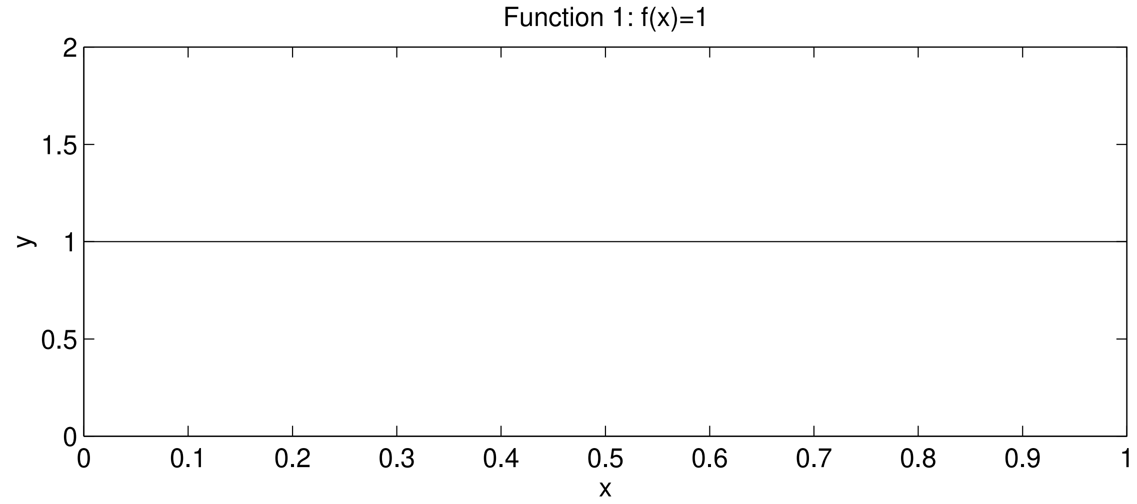
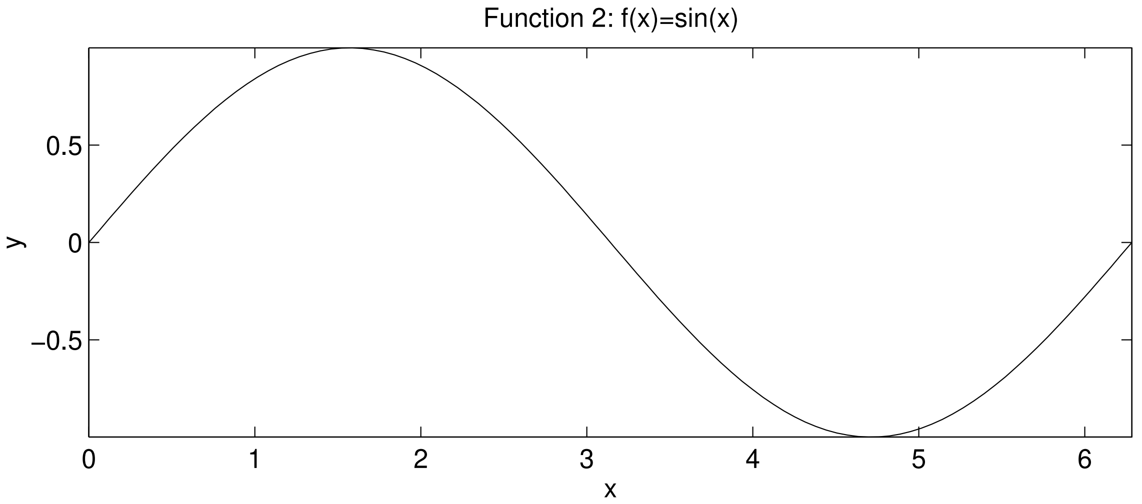
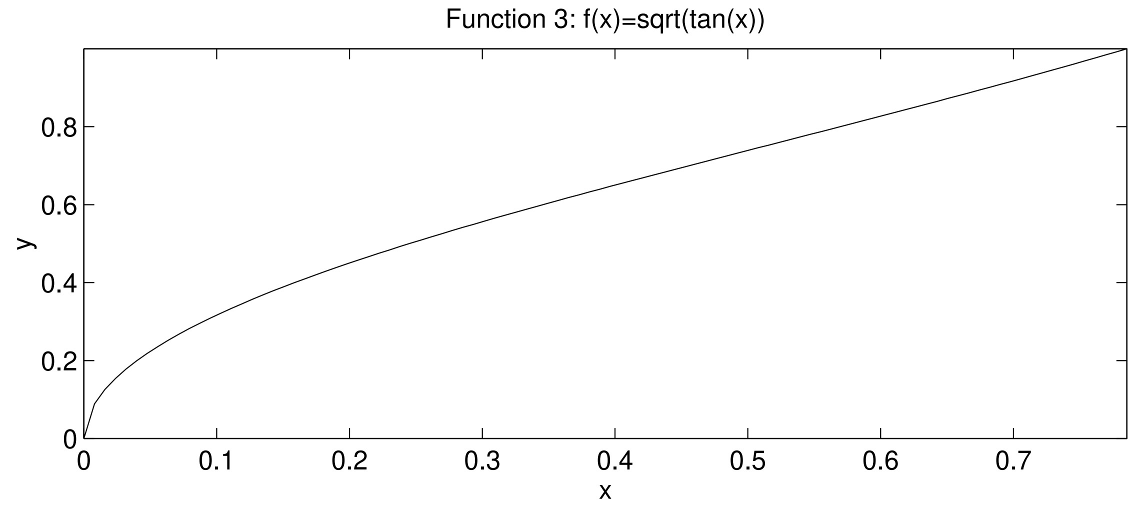
CS100b Program 6 Sample Solution

Fall 1999

1 Matlab Plot







2 M-File trapnew.m that plots function and performs the Trapezoidal Rule

```
%   a:  lower Integral limit
%   b:  upper Integral limit
%
%   f1: y=1
%   f2: y=sin(x)
%   f3: y=sqrt(tan(x))

%
%    INPUT
%
clear; format compact

f_name=input(' Please enter the function to integrate: ','s');
a     =input(' Enter lower integral bound (a):');
b     =input(' Enter upper integral bound (b):');

%
%    PLOT
%
orient tall
x=linspace(a,b,100); y=eval(f_name);
if length(y) == 1
     y=ones(1,length(x))*y;
end
plot(x,y)
if length(eval(f_name)) == 1
   c=eval(f_name)-1; d=eval(f_name)+1;
   axis([a b c d]);
else
   axis([a b -inf inf]);
end
title(['Function: f(x)=',f_name]); xlabel('x'); ylabel('y');

%
%    CHECK FOR ROOTS USING LHS-RHS METHOD
%
%    tol : tolerance, stopping criteria for lhs/rhs method
tol=sqrt(eps);
root=lhs_rhs(f_name,a,b,tol);

%
%    DETERMINE THE INTEGRAL WITH WRITTEN TRAPEZOIDAL FUNCTION
%                       AND WITH TRAPZ FUNCTION
fprintf('\n *** Trapezoidal rule for Function %2s: ***\n',f_name)
fprintf('     n    mytrapez        trapz\n')
```

```
for i=0:3
   n=10^i;
   % Use written trapezoidal rule
   if (root==a)
     I1 = abs(mytrapez(f_name,a,b,n));
   else
     I1a =abs(mytrapez(f_name,a,root,n));
     I1b =abs(mytrapez(f_name,root,b,n));
     I1=I1a+I1b;
   end

   % Use Matlab's trapz function
   x=linspace(a,b,n+1);y=eval(f_name);
   if length(y) == 1
      y=ones(1,length(x))*y;
   end
   I2 = trapz(x,y);

   % Output
   fprintf(' %5.0f  %10.6f   %10.6f\n',n,I1,I2)
end

%
%    DETERMINE THE INTEGRAL WITH INT FUNCTION
%
F=int(f_name,'x',a,b);
I=eval(F);
fprintf('\n Solution using MatLab''s int-function: %7.6f\n',I)
```

3 Sample Program Listing for function mytrapez.m

```
% Purpose:  Integration of a function by extended
%           trapezoidal rule.
% Synopsis: I = trapez(f_name,a,b,n)
%           f_name: function M-file name of integrand
%           a: lower limit
%           b: upper limit
%           n: number of intervals

function I = mytrapez(f_name,a,b,n)
     h=(b-a)/n;
     x=a+(0:n)*h;
     f=eval(f_name);
     if length(f)==1
        f=ones(1,length(x))*f;
     end
     I=h*(sum(f)-0.5*(f(1)+f(length(f))));
```

4 Sample Program Listing for LHS-RHS Method

```
function root=lhs_rhs(f_name,a,b,tol)
% determines the root of a function
%
% Input:  f_name: function name
%         a:      lower integral bound
%         b:      upper integral bound
%         tol:    tolerance for stopping criteria
%
% Output: root:   root of function or a if there is no root
%                 in interval [a,b]

step=0.001;
x=a;
x=x+step;
while (eval(f_name) > tol & x<b)
  x=x+step;
end

if (abs(b-x)<=step)   % no root
  root=a;
else
  root=x;             % root at x
end
```

5 Output (output.txt)

```
>> diary output.txt
>> trapnew
 Please enter the function to integrate: 1
 Enter lower integral bound (a):0
 Enter upper integral bound (b):1

 *** Trapezoidal rule for Function  1: ***
     n    mytrapez       trapz
     1    1.000000     1.000000
    10    1.000000     1.000000
   100    1.000000     1.000000
  1000    1.000000     1.000000

 Solution using MatLab's int-function: 1.000000
>>
>> trapnew
 Please enter the function to integrate: sin(x)
 Enter lower integral bound (a):0
 Enter upper integral bound (b):2*pi

 *** Trapezoidal rule for Function sin(x): ***
     n    mytrapez       trapz
     1    0.001280    -0.000000
    10    3.967047    -0.000000
   100    3.999671     0.000000
  1000    3.999997    -0.000000

 Solution using MatLab's int-function: 0.000000
>>
>> trapnew
 Please enter the function to integrate: sqrt(tan(x))
 Enter lower integral bound (a):0
 Enter upper integral bound (b):pi/4

 *** Trapezoidal rule for Function sqrt(tan(x)): ***
     n    mytrapez       trapz
     1    0.392699     0.392699
    10    0.483434     0.483434
   100    0.487356     0.487356
  1000    0.487491     0.487491

 Solution using MatLab's int-function: 0.487495
>> diary off
```

6 Questions

Increasing the values of **n** improves the accuracy. You could also say that the truncation error decreases. (However, at a very high **n** the accuracy starts getting worse, or, the rounding error increases.)