```
% ### EXintegrate2.m ###        12.01.16

% Example to show how Riemann sums can be used to determine the area
of a
% circle. Particular attention is given to examining how a "rectangle"
% compares to the area under the curves

% Uses several different "methods":
% Method A - Approximate the integral via brute force LEFT and RIGHT
Riemann sums
% Method B - Visualize the Riemann sums for different # of points for
% different methods (Fig.2); also show rect. width re "area" to
determine (Fig.1)
% Method C - Estimates via Matlab's built-in functions

% -----
% Notes
% o considers just a quarter of a circle (in the upper right
quadrant), the
% rest of the argument stemming from symmetry
% o provides an easy way to numerically estimate pi (e.g., if r=1,
this
% area should be pi/4)
% o modified version of original source code:
% http://ef.engr.utk.edu/ef230-2011-01/modules/matlab-integration/
% o For Method C, trapz.m requires user to specify # of intervals (Nt)
% while quad.m (which uses "adaptive Simpson quadrature")
automatically
% assumes a default error tolerance (i.e., user does not need to
specify #
% of points)
% o In some places below, different syntaxes were used to demonstrate
% different ways of achieving the same result {e.g., disp([ + num2str
vs disp(sprintf }



clear;
% ----------------------
% User parameters
r= 1;   % radius of circle
N= 100;    % Method A - # of "rectangles" for LEFT and RIGHT
pts= [3 4 5 6 10 25]; % Method B - # of points to consider integrating
(via trapz function)
dur= 1;     % Method B - pause duration [s] for visualization loop
(Fig.2)
rectW= 0.01;     % Method B - width for Riemann sum rect. (Fig.1)
{0.1} (kludge: should be less than ~0.2)
Nt= 100;     % Method C - # of points for trapz
```

```
precV= 10;  % # of vals. past decimal place for displaying built-in
function results {10}
% ----------------------

% ***************
% set up the relevant pieces
F = @(x)sqrt(r^2-x.^2); % eqn. for (quarter) circle
xL= [0 r]; % integration limits

% ***************
% Method A
% Approximate the integral via brute force LEFT and RIGHT Riemann sums
sumL= 0; sumR=0;
delX= (xL(2)-xL(1))/N;    % step-size
x= linspace(xL(1),xL(2),N+1);  % add one since N is # of 'boxes' and
is really N-1
for nn=1:N
    sumL= sumL + F(x(nn))*delX;
    sumR= sumR + F(x(nn+1))*delX;
end
disp(['Meth. A used ',num2str(N),' steps']);
disp(['Meth. A - area LEFT rule= ',num2str(sumL,precV),' (pi est.
=',num2str(4*sumL/r^2,precV),')']);
disp(sprintf('Meth. A - area RIGHT rule=  %g (pi est. = %g)',
sumR,4*sumR/r^2));
% MID is simply 1/2*(LEFT+RIGHT)
midV= 0.5*(sumL+sumR);
disp(['Meth. A - area MID= ',num2str(midV),' (pi est.
=',num2str(4*midV/r^2,precV),')']);



% ***************
% Method B
% Visualize the Riemann sums for different # of points for different
methods

% -------
% Fig.1 - Show the curve and some rect. Riemann sums
figure(1); clf; hold on; grid on;
fplot(F,[xL(1),xL(2)]) % a quick way to plot a function via functional
definition
set(findobj(gca, 'Type', 'Line', 'Linestyle', '-'), 'LineWidth',
2,'Color',[0 0 0]);
xlabel('x'); ylabel('F(x)');
% ---
pos= [0.2 0.4 0.6 0.8];     % centers for the three different Riemann
sums
% LEFT [centered about pos(1)]
px=[pos(1) pos(1)+rectW pos(1)+rectW pos(1)];
```

```
py=[0 0 F(pos(1)) F(pos(1))];
b1= fill(px,py,5,'FaceAlpha',0.5);
% RIGHT [centered about pos(2)]
px=[pos(2) pos(2)+rectW pos(2)+rectW pos(2)];
py=[0 0 F(pos(2)+rectW) F(pos(2)+rectW)];
b2= fill(px,py,3,'FaceAlpha',0.5);
% MID [centered about pos(3)]
px=[pos(3) pos(3)+rectW pos(3)+rectW pos(3)];
py=[0 0 F(pos(3)+rectW/2) F(pos(3)+rectW/2)];
b3= fill(px,py,1,'FaceAlpha',0.5);
% TRAP [centered about pos(4)]
px=[pos(4) pos(4)+rectW pos(4)+rectW pos(4)];
py=[0 0 F(pos(4)+rectW) F(pos(4))];
b4= fill(px,py,7,'FaceAlpha',0.5);
% ---
legend([b1 b2 b3 b4],'LEFT','RIGHT','MID','TRAP');
title('Comparison of different ways to compute Riemann sums');

% -------
% Fig.2 - animated version for the different methods
for np=pts
    figure(2); clf % clear the current figure
    hold on % allow stuff to be added to this plot
    x = linspace(xL(1),xL(2),np); % generate x values
    y = F(x); % generate y values
    a2 = trapz(x,y); % use trapz to integrate
    % Generate and display the trapezoids used by trapz
    for ii=1:length(x)-1
        % --- (rectangles: LEFT)
        subplot(221); hold on; grid on;
        title('Left-hand rule');
        px=[x(ii) x(ii+1) x(ii+1) x(ii)];   py=[0 0 y(ii) y(ii)];
        fill(px,py,ii)
        fplot(F,[xL(1),xL(2)]); xlabel('x'); ylabel('F(x)');
        % --- (rectangles: RIGHT)
        subplot(222); hold on; grid on;
        title('Rightt-hand rule');
        px=[x(ii) x(ii+1) x(ii+1) x(ii)];   py=[0 0 y(ii+1) y(ii+1)];
        fill(px,py,ii)
        fplot(F,[xL(1),xL(2)]); xlabel('x'); ylabel('F(x)');
        % --- (rectangles: MID)
        subplot(223); hold on; grid on;
        title('Mid-point rule');
        tempX= (x(ii+1)+x(ii))/2;
        tempY= F(tempX);
        px=[x(ii) x(ii+1) x(ii+1) x(ii)];   py=[0 0 tempY tempY];
        fill(px,py,ii)
        fplot(F,[xL(1),xL(2)]); xlabel('x'); ylabel('F(x)');
        % --- (trapezoids)
        subplot(224); hold on; grid on;
```

```
        title('Trapezoid rule');
        px=[x(ii) x(ii+1) x(ii+1) x(ii)];   py=[0 0 y(ii+1) y(ii)];
        fill(px,py,ii)
        fplot(F,[xL(1),xL(2)]); xlabel('x'); ylabel('F(x)');
    end
    %fplot(F,[xL(1),xL(2)]); xlabel('x'); ylabel('F(x)');
    %disp(['area calculated by trapz.m for ',num2str(np),' points
='',num2str(a2)]);
    %title(['area calculated by trapz.m for ',num2str(np),' points
='',num2str(a2)]);
    pause(dur);   % wait a bit
end

% ***************
% Method C
% Display estimates via Matlab's built-in functions
% ---
% quad
a1 = quad(F,xL(1),xL(2)); % use quad to integrate (default # of
points)
disp(['Meth. C - area via quad.m= ' num2str(a1,precV),' (pi est.
=',num2str(4*a1/r^2,precV),')']);
% ---
% trapz
xT = linspace(xL(1),xL(2),Nt); % generate x values
a2 = trapz(xT,F(xT)); % use trapz to integrate  (default # of points)
disp(['Meth. C - area via trapz.m= ' num2str(a2,precV),' (pi est.
=',num2str(4*a2/r^2,precV),')']);
```