Problem 9

$$y'' = -2y' - 3y^2 \qquad y(0) = 0 \qquad y(2) = -1$$

The following program is based on the function shoot2 in Example 8.1. The adaptive Runge-Kutta method (runKut5) is used for the intergration.

```
function p8_1_9
% Shooting method for 2nd-order boundary value problem
% in Problem 9, Problem Set 8.1.

xStart = 0; xStop = 2;      % Range of integration.
h = 0.1;                    % Step size.
freq = 2;                   % Frequency of printout.
u1 = -1.5; u2 = -0.5;       % Trial values of unkown
                            % initial condition u.
x = xStart;
u = ridder(@residual,u1,u2);
[xSol,ySol] = runKut5(@dEqs,x,inCond(u),xStop,h);
printSol(xSol,ySol,freq)

    function F = dEqs(x,y)      % First-order differential
    F = [y(2) -2*y(2)-3*y(1)^2]; % equations.
    end

    function y = inCond(u)  % Initial conditions (u is
    y = [0 u];              % the unknown condition).
    end

    function r = residual(u) % Boundary residual.
    x = xStart;
    [xSol,ySol] = runKut5(@dEqs,x,inCond(u),xStop,h);
    r = ySol(size(ySol,1),1) + 1;
    end
end
```

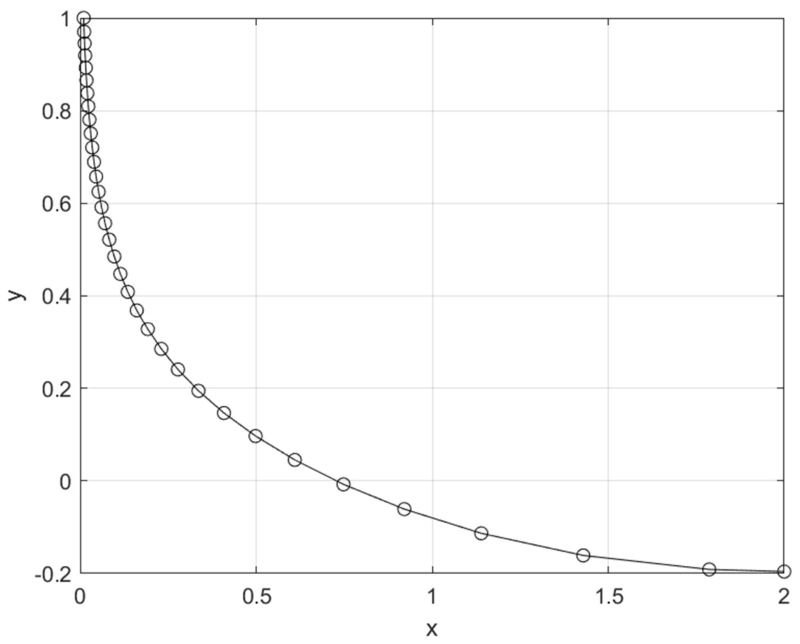
```
  x            y1            y2
0.0000e+00   0.0000e+00   -9.9420e-01
2.1272e-01  -1.7262e-01   -6.5598e-01
4.3180e-01  -2.9200e-01   -4.5460e-01
6.5808e-01  -3.8178e-01   -3.5340e-01
8.9993e-01  -4.6222e-01   -3.2249e-01
1.1626e+00  -5.4932e-01   -3.4986e-01
1.4475e+00  -6.6026e-01   -4.3934e-01
1.7457e+00  -8.1471e-01   -6.1311e-01
2.0000e+00  -1.0000e+00   -8.6542e-01
```

Problem 11

$$y'' = -\frac{1}{x}y' - y \qquad y(0.01) = 1 \qquad y'(2) = 0$$

y changes very rapidly near $x = 0$. So, assume $y(0.01) = 1$ for numerical integration.
The following program is based on the function `shoot2` in Example 8.1. The adaptive Runge-Kutta method (`runKut5`) is used for the integration. Linear interpolation (`linInterp`) is used for root finding.

```
function p8_1_11
% Shooting method for 2nd-order boundary value problem
% in Problem 11, Problem Set 8.1.

xStart = 0.01; xStop = 2;   % Range of integration.
h = 0.1;                    % Step size.
u1 = -50; u2 = 0;           % Trial values of unkown
                            % initial condition u.

x = xStart;
u = linInterp(@residual,u1,u2);
[xSol,ySol] = runKut5(@dEqs,x,inCond(u),xStop,h);
plot(xSol,ySol(:,1),'k-o'); grid on
xlabel('x'); ylabel('y')

    function F = dEqs(x,y)   % First-order differential
    F = [y(2) -y(2)/x-y(1)]; % equations.
    end

    function y = inCond(u)   % Initial conditions (u is
    y = [1 u];               % the unknown condition).
    end

    function r = residual(u) % Boundary residual.
    x = xStart;
    [xSol,ySol] = runKut5(@dEqs,x,inCond(u),xStop,h);
    r = ySol(size(ySol,1),2);
    end
end
```

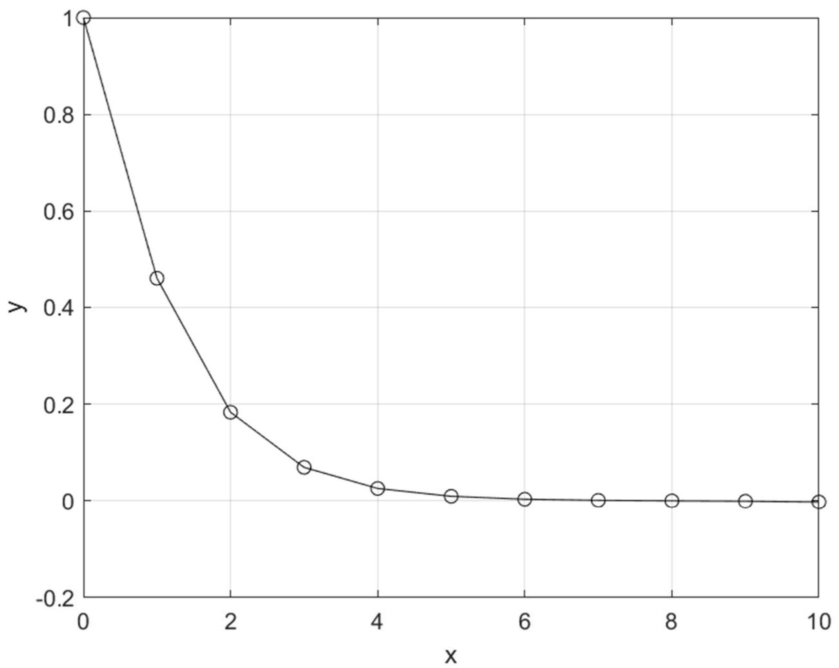


Problem 12

$$y'' = (1 - e^{-x})y \qquad y(0) = 1 \qquad y(\infty) = 0$$

The following program is based on the function shoot2 in Example 8.1.

```
function p8_1_12
% Shooting method for 2nd-order boundary value problem
% in Problem 12, Problem Set 8.1.

xStart = 0; xStop = 10; % Range of integration.
h = 1;                  % Step size.
u1 = 0; u2 = 2;         % Trial values of unkown
                        % initial condition u.

x = xStart;
u = linInterp(@residual,u1,u2);
[xSol,ySol] = bulStoer(@dEqs,x,inCond(u),xStop,h);
plot(xSol,ySol(:,1),'k-o');grid on
xlabel('x'); ylabel('y')

    function F = dEqs(x,y)       % First-order differential
    F = [y(2) (1-exp(-x))*y(1)]; % equations.
    end

    function y = inCond(u) % Initial conditions (u is
    y = [1 u];             % the unknown condition).
    end

    function r = residual(u) % Boundary residual.
    x = xStart;
    [xSol,ySol] = bulStoer(@dEqs,x,inCond(u),xStop,h);
    r = ySol(size(ySol,1),1);
    end
end
```



The results did not change significantly when the program was run with xStop = 7.5.

Problem 13

$$\begin{aligned} y''' &= -\frac{1}{x}y'' + \frac{1}{x^2}y' + 0.1(y')^3 \\ y(1) &= 0 \qquad y''(1) = 0 \qquad y(2) = 1 \end{aligned}$$

The following program is based on the function `shoot3` in Example 8.3. The adaptive Runge-Kutta method (`runKut5`) is used for the integration because the differential equation is nonlinear. Ridder's method (`ridder`) is used for root finding.

```
function p8_1_13
% Shooting method for 3rd-order boundary value
% problem in Problem 13, Problem Set 8.1.

xStart = 1; xStop = 2;  % Range of integration.
h = 0.2;                % Step size.
freq = 1;               % Frequency of printout.
u1 = 0; u2 = 2;         % Trial values of unknown
                        % initial condition u.
x = xStart;
u = ridder(@residual,u1,u2);
[xSol,ySol] = runKut5(@dEqs,x,inCond(u),xStop,h);
printSol(xSol,ySol,freq)

    function F = dEqs(x,y) % 1st-order differential eqs.
    F = [y(2) y(3) -y(3)/x+y(2)/x^2+0.1*y(2)^3];
    end

    function y = inCond(u) % Initial conditions.
    y = [0 u 0];
    end

    function r = residual(u) % Boundary residual.
    x = xStart;
    [xSol,ySol] = runKut5(@dEqs,x,inCond(u),xStop,h);
    r = ySol(size(ySol,1),1) - 1;
    end
end
```

```
>> p8_1_13
        x            y1           y2           y3
   1.0000e+00   0.0000e+00   9.0112e-01   0.0000e+00
   1.1571e+00   1.4218e-01   9.1160e-01   1.2496e-01
   1.3131e+00   2.8625e-01   9.3813e-01   2.1061e-01
   1.4963e+00   4.6205e-01   9.8375e-01   2.8379e-01
   1.7068e+00   6.7597e-01   1.0505e+00   3.4788e-01
   1.9488e+00   9.4093e-01   1.1422e+00   4.0906e-01
   2.0000e+00   1.0000e+00   1.1635e+00   4.2129e-01
```

Problem 15

$$\begin{aligned} y''' &= -2y'' - \sin y \\ y(-1) &= 0 \qquad y'(-1) = -1 \qquad y'(1) = 1 \end{aligned}$$

The following program is based on the function `shoot3` in Example 8.3. The adaptive Runge-Kutta method (`runKut5`) is used for the integration because the differential equation is nonlinear. Ridder's method (`ridder`) is used for root finding.

```
function p8_1_15
% Shooting method for 3rd-order boundary value
% problem in Problem 15, Problem Set 8.1.

xStart = -1; xStop = 1;      % Range of integration.
h = 0.05;                    % Step size.
freq = 2;                    % Frequency of printout.
u1 = 2; u2 = 6;              % Trial values of unknown
                             % initial condition u.
x = xStart;
u = ridder(@residual,u1,u2);
[xSol,ySol] = runKut5(@dEqs,x,inCond(u),xStop,h);
printSol(xSol,ySol,freq)

    function F = dEqs(x,y) % 1st-order differential eqs.
    F = [y(2) y(3) -2*y(3)-sin(y(1))];
    end

    function y = inCond(u) % Initial conditions.
    y = [0 -1 u];
    end

    function r = residual(u) % Boundary residual.
    x = xStart;
    [xSol,ySol] = runKut5(@dEqs,x,inCond(u),xStop,h);
    r = ySol(size(ySol,1),2) - 1;
    end
end
```

```
>> p8_1_15
        x            y1           y2           y3
   -1.0000e+00   0.0000e+00  -1.0000e+00   4.3791e+00
   -8.6166e-01  -1.0003e-01  -4.7044e-01   3.3278e+00
   -6.7867e-01  -1.3651e-01   4.1312e-02   2.3272e+00
   -4.8073e-01  -8.8056e-02   4.2389e-01   1.5855e+00
   -2.6549e-01   3.5252e-02   7.0222e-01   1.0356e+00
   -2.9648e-02   2.2546e-01   8.9462e-01   6.2107e-01
    2.3136e-01   4.7614e-01   1.0121e+00   2.9721e-01
    5.2346e-01   7.8042e-01   1.0585e+00   3.3759e-02
    8.5216e-01   1.1260e+00   1.0323e+00  -1.8063e-01
    1.0000e+00   1.2763e+00   1.0000e+00  -2.5381e-01
```

Problem 21

$$y''' = -yy'' \qquad y(0) = y'(0) = 0 \qquad y'(\infty) = 2$$

```
function p8_1_21
% Shooting method for 3rd-order boundary value
% problem in Problem 21, Problem Set 8.1.

xStart = 0; xStop = 5;      % Range of integration.
h = 0.1;                    % Step size.
u1 = 0; u2 = 2;             % Trial values of unknown
                            % initial conditions [u].
x = xStart;
u = ridder(@residual,u1,u2);
[xSol,ySol] = runKut5(@dEqs,x,inCond(u),xStop,h);
plot(xSol,ySol(:,1),'k-o'); hold on
plot(xSol,ySol(:,2),'k-s'); grid on
xlabel('x')
legend('y','dy/dx','Location','best')

    function F = dEqs(x,y) % 1st-order differential eqs.
    F = [y(2) y(3) -y(1)*y(3)];
    end

    function y = inCond(u) % Initial conditions.
    y = [0 0 u];
    end

    function r = residual(u) % Boundary residual.
    x = xStart;
    [xSol,ySol] = runKut5(@dEqs,x,inCond(u),xStop,h);
    n = size(ySol,1);
    r = ySol(n,2) - 2;
    end
end
```