```
% Problem 1
clc;
clear;

% Initial conditions
y_0 = [1/2; 0];
t_0 = 0;
t_f = 3*acos(-1);

% Testing purposes - make sure the trapezoidal method returns a numerical
% value that's close to the actual solution
% n = 10000;
% h = (t_f-t_0)/n;
% y = trapezoid_2(@odefun, t_0, t_f, h, y_0);
% fprintf("Absolute error between numerical approximation and actual solution:
 %0.16e\n", ...
%         abs(besselj(1, t_f) - t_f*y(1)));

% Use repeated Richardson extrapolation to obtain a numerical solution
% that's accurate to 10 digits of the actual answer
actual_soln = besselj(1, t_f);
max_rows = 20;
tol = 1e-11;
h = (t_f - t_0);

A = zeros(max_rows, max_rows);
y = trapezoid_2(@odefun, t_0, t_f, h, y_0);
A(1, 1) = y(1);

for i = 1:(max_rows - 1)
    h = h/2;

    y = trapezoid_2(@odefun, t_0, t_f, h, y_0);
    A(i + 1, 1) = y(1);

    for j = 1:i
        A(i + 1, j + 1) = ((4^j)*A(i + 1, j) - A(i, j))/((4^j) - 1);
    end

    if abs(A(i + 1, i + 1) - A(i, i)) < tol
        break;
    end
end

num_approx = t_f*A(max_rows, max_rows);
abs_err    = abs(num_approx - actual_soln);
rel_err    = abs_err/abs(actual_soln);

fprintf("Numerical approximation of solution: %0.16f\n", num_approx);
% fprintf("Absolute error between numerical approximation computed via
 Richardson extrapolation and actual solution: %0.16e\n", ...
%         abs_err);
% fprintf("Relative error between numerical approximation computed via
 Richardson extrapolation and actual solution: %0.16e\n", ...
%          rel_err);

function [f, J_f] = odefun(t, y)
    if norm(t) < 1e-10
        f = [0; (-1/8)];
    else
        f = [y(2); (-3./t).*y(2) - y(1)];
    end

    % Return the Jacobian of f if requested
    if nargout > 1
        J_f = [0, 1; ...
               -1, -3./t];
    end
end

function [y] = trapezoid(odefun, t_0, t_f, h, y_0)
    % Tolerance for Newton's method
    tol = 1e-10;
    y = y_0;

    while t_0 <= t_f
        % Compute the explicit part of the trapezoidal method
        E_y = y + (h/2)*odefun(t_0, y);

        % Perform one step of forward Euler to kick off Newton's method
        FE_y = y + h*odefun(t_0, y);

        % Apply Newton's method to calculate the implicit part of the
        % trapezoidal method
        while true
            [f, J_f] = odefun(t_0 + h, y);
            update = ((h/2)*J_f - eye(size(J_f)))\(E_y + (h/2)*f - FE_y);
            FE_y = FE_y - update;
            if norm(update) < tol
                break;
            end
        end

        % Compute y_{n+1} using the explicit and implicit parts of the
        % trapezoidal method
        y = E_y + (h/2)*odefun(t_0 + h, FE_y);

        t_0 = t_0 + h;
    end
end

function [y] = trapezoid_2(odefun, t_0, t_f, h, y_0)
    y = y_0;

    while t_0 <= t_f
        E_y = y + (h/2)*odefun(t_0, y);
        I_m = [1, -h/2; ...
               h/2, 1 + (3/2)*h/(t_0 + h)];
        I_y = I_m\E_y;

        y = E_y + (h/2)*odefun(t_0 + h, I_y);
        t_0 = t_0 + h;
    end
end
```

```
Numerical approximation of solution: 0.17672521522753480
```

Published with MATLAB® R2022a