Problem 1.

Find the characteristic equation and reduced characteristic equation of the 3-step method

$$y_{n+1} = -\frac{27}{11}y_n + \frac{27}{11}y_{n-1} + y_{n-2} + h\left[\frac{3}{11}f_{n+1} + \frac{27}{11}f_n + \frac{27}{11}f_{n-1} + \frac{2}{11}f_{n-2}\right]$$

Find the (exact) roots of the reduced characteristic equation, and determine if this method is stable for small $h\lambda$ or not.

Solution

First let us rewrite the method as

$$y_{n+1} + \frac{27}{11}y_n - \frac{27}{11}y_{n-1} - y_{n-2} = h\left[\frac{3}{11}f_{n+1} + \frac{27}{11}f_n + \frac{27}{11}f_{n-1} + \frac{2}{11}f_{n-2}\right]$$

Note that for this method we have

$$a_0 = -1, a_1 = -\frac{27}{11}, a_2 = \frac{27}{11}$$

Which produces the characteristic polynomial

$$\begin{aligned} p(z) &= z^3 + \frac{27}{11}z^2 - \frac{27}{11} - 1 \\ &= \frac{1}{11}(z-1)(11z^2 + 38z + 11) \\ &= \frac{1}{11}(z-1)(z + \frac{1}{11}(19 + 4\sqrt{15}))(z - \frac{1}{11}(4\sqrt{15} - 19)) \end{aligned}$$

From this we find the zeros for the characteristic polynomial

$$z = 1, z = -\frac{1}{11}(19 + 4\sqrt{15}), z = \frac{1}{11}(4\sqrt{15} - 19)$$

Note that $|-\frac{1}{11}(19 + 4\sqrt{15})| \approx 3.1356303 > 1$.
This implies that this method is not stable for small values of $h\lambda$.

Problem 2.

The IVP

$$y' = -y + 2e^{-t}\cos(2t)$$
$$y(0) = 0$$

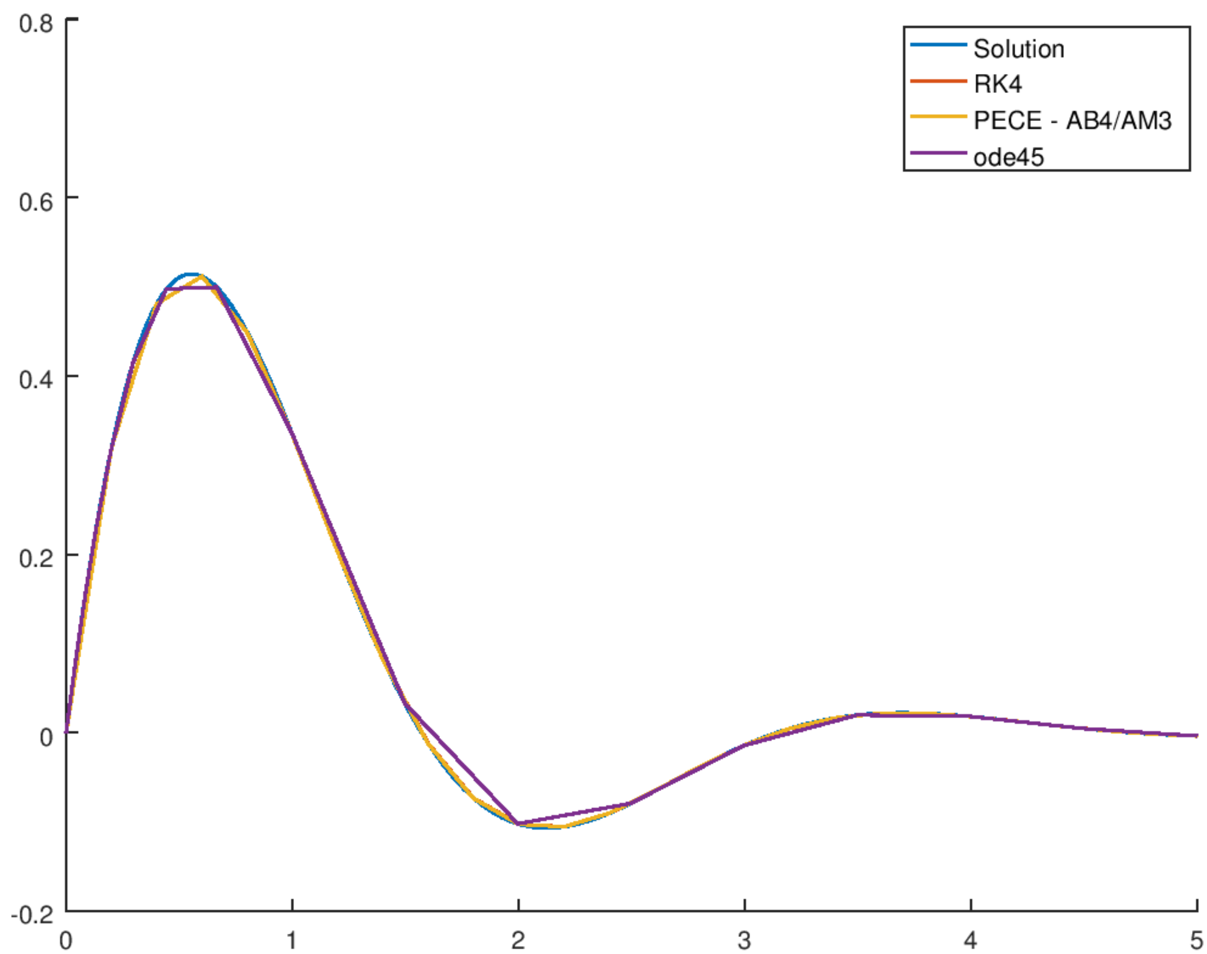
has the true solution $y(t) = e^{-t}\sin(2t)$.
Solve it numerically from t=0 to t=5, using

(a) the classical 4-stage Runge-Kutta method with h=0.2

(b) a PECE predictor-corrector method based on 4-step AB, 3-step AM, h=0.2
Use RK values from part (a) for the startup.

(c) the Matlab built-in ODE45. This routine will pick its own steps.

Solution



	Actual	RK4	PECE - AB4/AM3	ODE45 (h unknown)
h = 0.2				
Value	-0.013911	-0.013911	-0.013776	-0.014068
Error		$2.1822 * 10^{-7}$	$1.3560 * 10^{-4}$	$1.5663 * 10^{-4}$
h = 0.1				
Value	-0.013911	-0.013911	-0.013908	
Error		$1.2907 * 10^{-8}$	$3.3871 * 10^{-6}$	

```
function p2

clear all;
close all;
hold on;

%% Actual solution.
_f = @(t) e.^(-t) .* sin(2 .* t);
_t = linspace(0, 5, 200);
_y = _f(_t);

plot(_t, _y, 'DisplayName', 'Solution', 'LineWidth', 1);
val = _f(3)

%% Numerical Solutions
f = @(t, y) -y + 2 .* e^(-t) * cos(2*t);
h = 0.2; % Step Size
t = 0:h:5; % Interval
y = [0]; % Initial point
steps = length(t);

%% RK4
for n = 1:(steps-1)

    % 4-Step
    k1 = f(t(n),          y(n));
    k2 = f(t(n) + h/2,          y(n) + (h/2) * k1);
    k3 = f(t(n) + h/2,          y(n) + (h/2) * k2);
    k4 = f(t(n) + h,          y(n) + h * k3);

    % Compute y_{n+1}
    y(n+1) = y(n) + (h/6) * (k1 + 2*k2 + 2*k3 + k4);

end

val = y((3/h) + 1)
err = norm(_f(3) - val, inf)
```

```
plot(t, y, 'DisplayName', 'RK4', 'LineWidth', 1);

%% PECE
y = y(1:4); % Pull startup values from RK4

for n = 5:steps

    % 4-Step AB
    y(n) = y(n-1) + h * (
        (55/24) * f(t(n-1), y(n-1)) -
        (59/24) * f(t(n-2), y(n-2)) +
        (37/24) * f(t(n-3), y(n-3)) -
        (3/8)        * f(t(n-4), y(n-4))
    );

    % 3-Step AM
    y(n) = y(n-1) + h * (
        (3/8)        * f(t(n), y(n)) +
        (19/24)        * f(t(n-1), y(n-1)) -
        (5/24)        * f(t(n-2), y(n-2)) +
        (1/24)        * f(t(n-3), y(n-3))
    );

end

val = y((3/h) + 1)
err = norm(_f(3) - val, inf)
plot(t, y, 'DisplayName', 'PECE - AB4/AM3', 'LineWidth', 1);

%% ODE45

[solx, soly] = ode45(f, [0 5], [0 0]);
val = interp1(solx, soly(:,1), 3, method='linear')
err = norm(_f(3) - val, inf)
plot(solx, soly(:,1), 'DisplayName', 'ode45', 'LineWidth', 1);

legend('show');

end
```

Problem 3.

Use the Matlab routine $fzero$ or something comparable to find the two solutions of

$$e^x - x - 2 = 0.$$

There is one positive and one negative solution.

Solution

Zeros exist at the points x = -1.8414 and x = 1.1462.

```
function p3

y = @(x) (e.^x) - x - 2;
fzero(y, [-10, 0]) % -1.8414
fzero(y, [0, 10])  %  1.1462

end
```

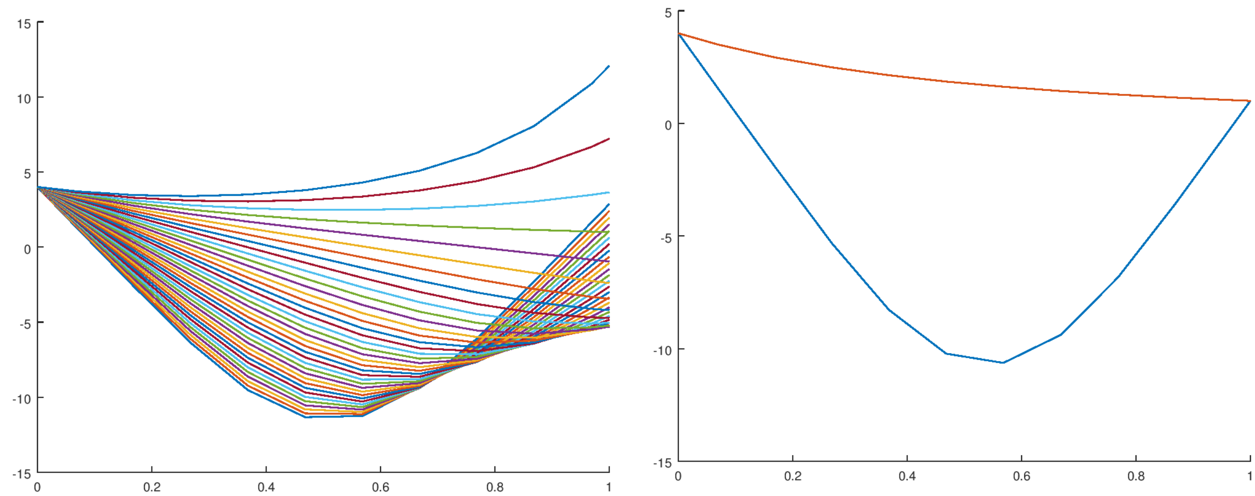
Problem 4.

Solve the boundary value problem

$$\begin{aligned} y''(t) &= \frac{3}{2}y^2(t) \\ y(0) &= 4 \\ y(1) &= 1 \end{aligned}$$

by a shooting method.

Solution



(a) s $\in [-40, -5]$ (b) Solutions

Using fzero to find zeros of the function phi(s) produces values of s = -35.859 and s = -8. Figure (a) shows the solutions for integer s values in the interval $[-40, -5]$, where figure (b) displays only the solutions for s = -35.859 and s = -8 that satisfy the IVP with y(0) = 4 and y(1) = 1. The code to produce these solutions may be found below.

```
function p4

clear all;
close all;
hold on;

for s = -40:-5
    [err, solx, soly] = phi(s);
    plot(solx, soly(:,1), 'DisplayName', 'ode45', 'LineWidth', 1);
end

% -35.859
zero0 = fzero(@phi, -40)
% -8
zero1 = fzero(@phi, -5)

[err, solx, soly] = phi(zero0);
plot(solx, soly(:,1), 'DisplayName', 'ode45', 'LineWidth', 1);
[err, solx, soly] = phi(zero1);
plot(solx, soly(:,1), 'DisplayName', 'ode45', 'LineWidth', 1);

end
```

```
function [err, solx, soly] = phi(s)

dy = @(t, y) [y(2), (3/2) * (y(1) .^ 2)];

[solx, soly] = ode45(dy, [0, 1], [4, s]);
est = interp1(solx, soly(:,1), 1, method='linear');
err = est - 1;

end
```

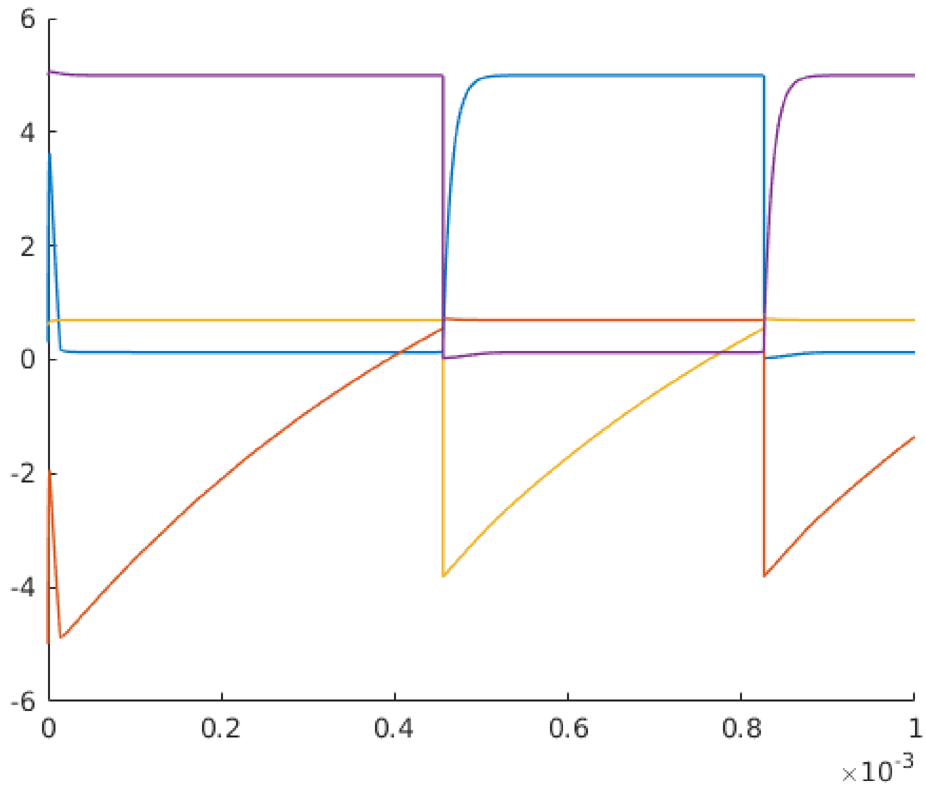
Problem 5.

(a) This is a stiff ODE. Solev the equations on the interval [0, 0.001], using a stiff solver such as ode15s. Plot the result.

(b) Answer the following questions.

- How many steps did the stiff solver take? How much computer time?
- How many steps does a non-stiff solver, such as ode113 take? How much computer time does it take?
- What is the clock period?

Solution



From the code included below, we can tell that ode15s took 393 steps and solved the stiff ODE in 0.721958 seconds whereas ode113 took 19484 steps to solve the stiff ODE in 1.1636209 seconds. The graph gives us a clock width of approximately $(8.25 * 10^{-4} - 4.45 * 10^{-4}) = 0.00038$ seconds. Thus the clock period is approximately 0.00076 seconds.

```
function p5

clear all;
close all;
hold on;

% Given parameters
VT        = 0.026;
Vcc        = 5.0;
p1        = 1100000;
p2         = 20000;
p3         = 0.000011;
p4         = 0.000022;
p5         = 0.0000001;
p6         = 0.00001;
p7         = 1000000;
p8         = 0.00001;
p9         = 0.00002;
V0         = [0.3; -5; 0.6; 5];

% Stiff ODE We want to solve.
dV = @(t, V) [ ...
    % dV1 / dt
    p1 * (Vcc - V(1)) + p2 * (Vcc - V(2)) - p3 * (exp(V(3)./VT) - 1) + ...
        p4 * (exp((V(3)-V(1))./VT) - 1) - p5 * (exp(V(2)/VT) - 1) - p6 * (exp((V(2)-V(4)
    % dV2 / dt
    p7 * (Vcc - V(1)) + p2 * (Vcc - V(2)) - p8 * (exp(V(3)./VT) - 1) + ...
        p9 * (exp((V(3)-V(1))./VT) - 1) - p5 * (exp(V(2)/VT) - 1) - p6 * (exp((V(2)-V(4)
    % dV3 / dt
    p2 * (Vcc - V(3)) + p7 * (Vcc - V(4)) - p5 * (exp(V(3)./VT) - 1) - ...
        p6 * (exp((V(3)-V(1))./VT) - 1) - p8 * (exp(V(2)/VT) - 1) + p9 * (exp((V(2)-V(4)
    % dv4 / dt
    p2 * (Vcc - V(3)) + p1 * (Vcc - V(4)) - p5 * (exp(V(3)./VT) - 1) - ...
        p6 * (exp((V(3)-V(1))./VT) - 1) - p3 * (exp(V(2)/VT) - 1) + p4 * (exp((V(2)-V(4)
];

% Solve with ode15s.
tic;
[solx, soly] = ode15s(dV, [0 0.001], V0);
toc; % Elapsed time is 0.721958 seconds.
disp(length(solx)); % 393 Steps in solution.

% Plot the solution.
plot(solx, soly(:,1), 'DisplayName', 'V1', 'LineWidth', 1);
```

```
plot(solx, soly(:,2), 'DisplayName', 'V2', 'LineWidth', 1);
plot(solx, soly(:,3), 'DisplayName', 'V3', 'LineWidth', 1);
plot(solx, soly(:,4), 'DisplayName', 'V4', 'LineWidth', 1);

% Solve with ode113.
tic;
[solx, soly] = ode113(dV, [0 0.001], V0);
toc; % Elapsed time is 1.1636209 seconds.
disp(length(solx)); % 19484 Steps in solution.

end
```