## Table of Contents

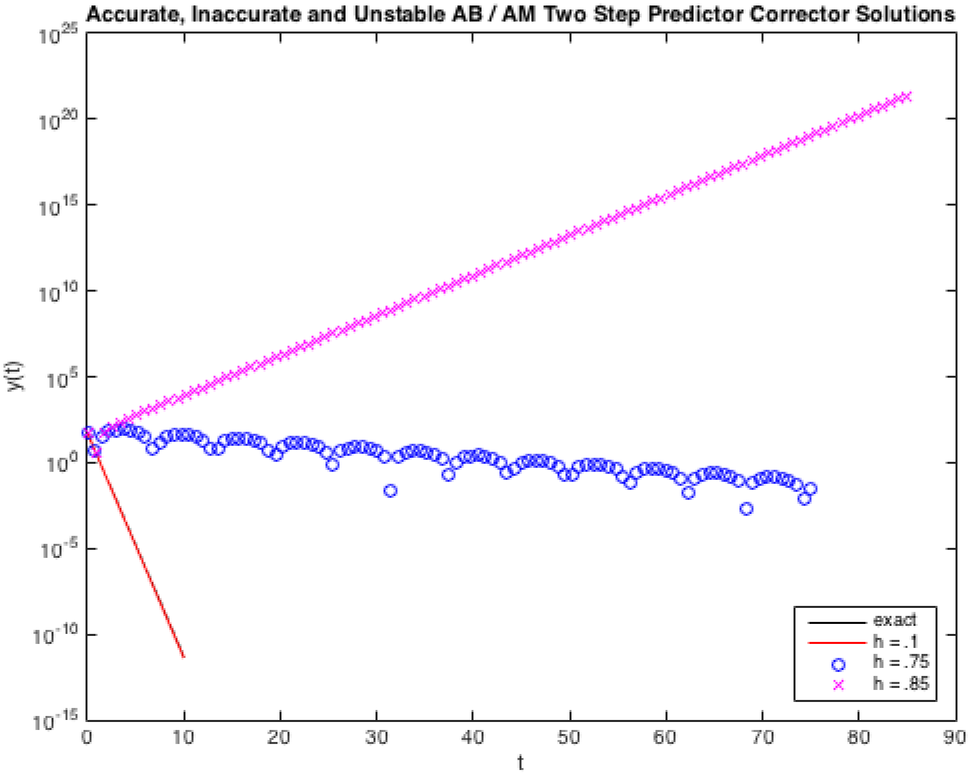
# Matt McFarland

ENGS 91, lab 6, question 2

```
function [] = q2()

% Write AB / AM Two Step Predictor Corrector Scheme
close all; clear all;
```

# Define constants and functions

```
y0      = 50;
t_start = 0;

h       = [.1 .75 .85];              % accurate, stable but inaccurate
 and unstable time steps

yFunc       = @(t) (y0 * exp(-3.*t));   % analytical solution
dydt        = @(y, t) (-3.*y);             % ODE
points      = 100;

% Solve ODE for different step sizes
[accurate_t, accurate_y]        = TwoStep(dydt, h(1), yFunc,y0,
 t_start, h(1)*points, 0);
[inaccurate_t, inaccurate_y]    = TwoStep(dydt, h(2), yFunc,y0,
 t_start, h(2)*points, 0);
[unstable_t, unstable_y]        = TwoStep(dydt, h(3), yFunc,y0,
 t_start, h(3)*points, 0);

exact_t = linspace(t_start, h(1)*points, 1000);
exact_y = yFunc(exact_t);

figure()
semilogy(exact_t,       abs(exact_y), 'k',...
        accurate_t,     abs(accurate_y),'r',...
        inaccurate_t,   abs(inaccurate_y),'bo',...
        unstable_t,     abs(unstable_y),'mx')
xlabel('t')
ylabel('y(t)')
title('Accurate, Inaccurate and Unstable AB / AM Two Step Predictor
 Corrector Solutions')
legend('exact','h = .1','h = .75','h = .85','Location','southeast')
```



```
end
```

# A-B / A-M 2-Step Predictor Corrector Solution to ODE

```
        Uses Analytic solution to get first point beyond initial value

function [TwoStep_t, TwoStep_y] = TwoStep(RateFunc, step_size,
 AnalyticFunc,y_0, t_0, t_end, n_max)

    max_len = int64((t_end - t_0) / step_size);
    TwoStep_t = zeros(n_max + 1, max_len);
    TwoStep_y = zeros(n_max + 1, max_len);

    TwoStep_t(1,1) = t_0;
    TwoStep_y(1,1) = y_0;

    delta_t         = step_size;
    steps           = (t_end - t_0) / delta_t;

    % Use analytic function to get first point beyond initial value
    TwoStep_y(1,2)  = AnalyticFunc(TwoStep_t(1,1) + delta_t);
    TwoStep_t(1,2)  = TwoStep_t(1,1) + delta_t;

    % Calculate using prediction - correction with 1 correction
    for j = 2:steps
        TwoStep_t(1,j+1) = TwoStep_t(1,1) + j * delta_t;

        f_cur       = RateFunc( TwoStep_y(1,j),   TwoStep_t(1,j)   );
        f_back      = RateFunc( TwoStep_y(1,j-1), TwoStep_t(1,j-1) );
        predict     = TwoStep_y(1,j) + delta_t/2 * (3 * f_cur -
 f_back);
        correct     = RateFunc( predict, TwoStep_t(1,j+1) );

        TwoStep_y(1,j+1) = TwoStep_y(1,j) + ...
            delta_t/12 * (5 * correct + 8 * f_cur - f_back);
    end
end
```

*Published with MATLAB® R2015a*