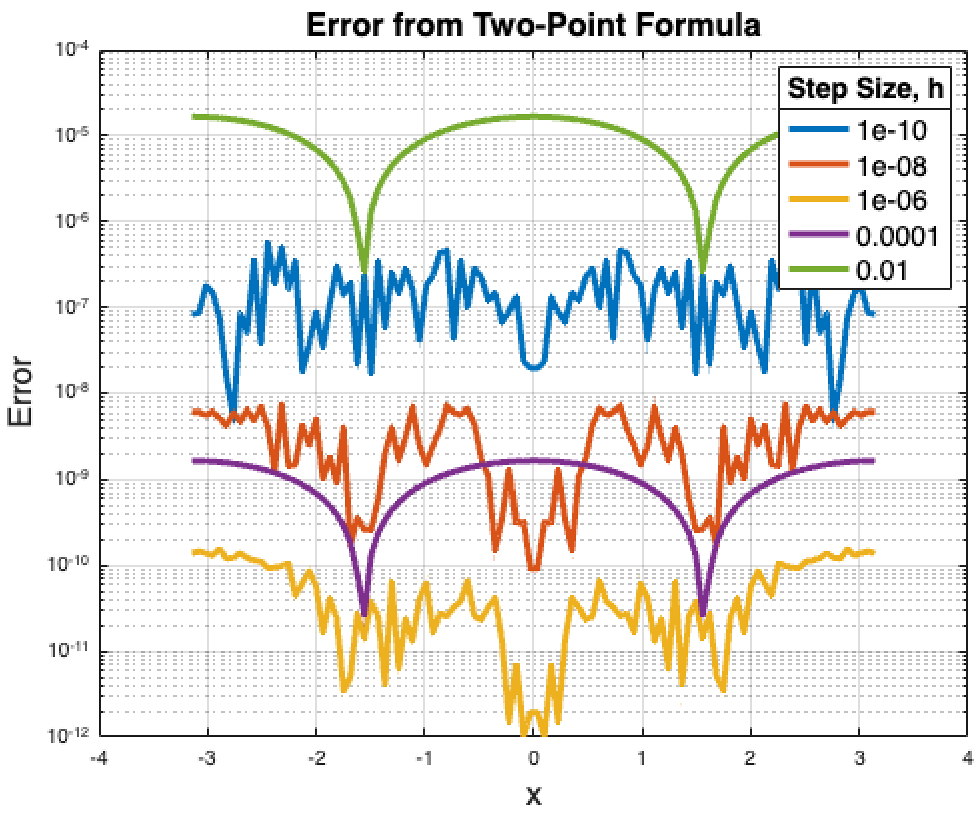
Numerical Derivatives Group Assignment

Bethany Wu & Alex Stapely

Problem 1: Finite Differences

```
syms x0 h0
x = linspace(-pi,pi,100);
y = @sin;
dydx = @(x0,h0)  (y(x0+h0)-y(x0-h0))/(2*h0);

hs = [1e-10 1e-8 1e-6 1e-4 1e-2];
figure();
for h = hs
    yp = dydx(x,h);
    semilogy(x,abs(yp-cos(x)),'LineWidth',3); hold on;
end
title('Error from Two-Point Formula','FontSize',16); xlabel('x','FontSize',16);
ylabel('Error','FontSize',16);
leg = legend(string(hs),'FontSize',14); title(leg,'Step Size, h'); grid on; hold
off;
```



Looking at the error plot, as h is decreased the error decreases until h is around 1e-6, then the error begins to increase!

Therefore, h = 1e-6 seems to be the optimal step size for this specific case where the error is between 10^-12 and 10^-10.

Problem 2: More sophisticated derivatives...

Part (a) (Find written functions in bottom section!)

Richardson extrapolation of the finite difference formulas: Alex

Chebyshev methods: Bethany

Part (b): (See 'compare' function in bottom section!)

```
% %Write a driver function that allows you to use a common interface to apply the
methods
% % you implemented in part (a) with an arbitrary function.
%
% % Set up arbitrary function and parameters
% f=@(x) sin(x);
% % x = Point(s) to evaluate at
% x = pi;
% % Step
% h = 0.4;
% % Order of extrapolation
% n = 4;
% % Calling that function here
% compare(f,x,h,n)
```

Part (c)

```
% sin(x) for 100 points between -pi and pi
f1=@(x) sin(x);
xs1 = linspace(-pi,pi,100);
h = 0.4;
n = 4;
compare(f1,xs1,h,n)
```

```
RMSE of the Richardson method with actual derivative: 0.67
Took 0.0421 seconds to compute.
RMSE of the Chebyshev method with actual derivative: 0.67
Took 0.7491 seconds to compute.
```

```
% sin(1/x) for 100 points between [-1,1]
f2=@(x) 1./sin(x);
xs2 = linspace(-1,1,100);
compare(f2,xs2,h,n)
```

```
Warning: Function not resolved using 65537 pts. Have you tried 'splitting on'?
RMSE of the Richardson method with actual derivative: 12094.39
Took 0.0065 seconds to compute.
RMSE of the Chebyshev method with actual derivative: 2785.80
Took 0.6873 seconds to compute.
```

```
% 3x^2 + 1/(pi^2)*ln((pi-x)^2)+1 for 100 points between [3.14,3.16]
f3=@(x) 3*x.^2 + 1./(pi^2).*log((pi-x).^2)+1;
xs3 = linspace(3.14,3.16,100);
compare(f3,xs3,h,n)
```

```
Warning: Function not resolved using 65537 pts. Have you tried 'splitting on'?
RMSE of the Richardson method with actual derivative: 44.99
Took 0.0125 seconds to compute.
RMSE of the Chebyshev method with actual derivative: 262.75
Took 0.6880 seconds to compute.
```

```
% sin(sin(sin....sin(x))...) 100 nested sines, 100 points between [-1,1]
num_sines = 100;
sines = cell(num_sines, 1);
sine = @(x) sin(x);
for i = 2:num_sines
    sine = @(x) sin(sine(x));
    sines{i} = sine;
end

f4 = sines{end};
xs4 = linspace(-1,1,100);
compare(f4,xs4,h,n)
```

```
RMSE of the Richardson method with actual derivative: 0.31
Took 158.3536 seconds to compute.
RMSE of the Chebyshev method with actual derivative: 0.31
Took 34.0710 seconds to compute.
```

```
% 1/|y| where y satisfies the Ax=b equation, 100 points between [-3,3]
f5 = @(x) 1./norm([1 x; 2 x.^2].\[1; 1]);
xs5 = linspace(-3,3,100);
compare(f5,xs5,h,n)
```

```
RMSE of the Richardson method with actual derivative: 0.41
Took 0.0214 seconds to compute.
RMSE of the Chebyshev method with actual derivative: 0.41
Took 0.5698 seconds to compute.
```

Part (d)

Overall, Chebyshev should perform better than Richardson extrapolation for finding the derivative of a function when the function is 'smooth' or periodic. However, the Richardson method performs better than the Chebyshev for functions of higher order and greater numerical stability.

Functions

```
function der  = richardson(f, x, h, n)
```

```
    R(1,1) = (f(x+h)-f(x))/h;
    for i = 1:n
        h = h/2;
        R(i+1,1)=(f(x+h)-f(x))/h;
        for j=1:i
            R(i+1,j+1)=(2^j*R(i+1,j)-R(i,j))/(2^j-1);
        end
    der = R(i+1,j+1);
    end
end

function deriv = chebyshev_diff(func,a,b,numPts)
    % Uses open-source package, Chebfun!
    cf = chebfun(func,linspace(a,b,numPts));
    df = diff(cf);
    deriv = df.pointValues';
end

% This function is written for part b. Used to compare the different methods
% from part a
function compare(arbitrary_function,xs,h,n)
    % Richardson Method
    der_rich=zeros(1,100); tic;
    for i = 1:100
        x = xs(i);
        der = richardson(arbitrary_function, x, h, n);
        % Put next function here
        %fprintf('According to the Richardson extrapolation method, the derivative
is: %.4f\n', der);
        % fprintf('According to the other method, the derivative is: %.9f\n', der);
        der_rich(i) = der;
    end
    t_rich = toc;

    % Chebyshev method
    tic;
    der_cheb = chebyshev_diff(arbitrary_function,xs(1),xs(end),length(xs));
    t_cheb = toc;

    % Compute actual derivative
    func_deriv = diff([arbitrary_function(xs) 0]);

    % Compare RMSE of each method with actual derivative
    fprintf("RMSE of the Richardson method with actual derivative: %.2f\nTook %.4f
seconds to compute.\n" + ...
        "RMSE of the Chebyshev method with
actual derivative: %.2f\nTook %.4f seconds to
compute.\n",rmse(der_rich,func_deriv),t_rich,rmse(der_cheb,func_deriv),t_cheb);

end
```