Homework 2.(a)

Modeling Complex Systems, Xing Jin & Javier Lobato

Due date: Thursday, February 15, 2018

A Optimal h for the central differences method

The Taylor series expansions for the central differences method are:

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(x) + \ldots \\ f(x-h) &= f(x) - hf'(x) + \frac{1}{2}h^2 f''(x) - \frac{1}{6}h^3 f'''(x) + \ldots \end{aligned}$$

Subtracting and rearranging these two equations, the scheme is defined:

$$f'(x) \simeq \frac{f(x+h) - f(x-h)}{2h} + E(f,h)$$

where $E(f,h)$ represents the truncation and round-off error.

Expressing the scheme in another way, we have that:

- $D_h = \dfrac{f(x+h) - f(x-h)}{2h}$
- $D_h = f'(x) + \dfrac{h^3}{3} f'''(\eta) \longrightarrow f'''(\eta) = \dfrac{3}{h^3}[D_h - f'(x)]$

In this case. the upper bound is the third derivative:

$$|f'''(x)| \leq M_3$$

Rearranging and including the previous equation:

$$\left| D_h - \frac{f'(x)}{2h} \right| \leq \frac{M_3}{2h}\frac{h^3}{3} \longrightarrow \left| D_h - \frac{f'(x)}{2h} \right| \simeq \frac{M_3 h^2}{6} + \frac{2\delta}{2h} \longrightarrow \text{errorD}(h) = \frac{M_3 h^2}{6} + \frac{\delta}{h}$$

Minimizing that function:

$$\frac{d(\text{errorD}(h))}{dh} = \frac{M_3 h}{3} - \frac{\delta}{h^2} = 0 \rightarrow \frac{M_3 h^3}{3} - \delta = 0 \rightarrow \boxed{h_{opt} = \sqrt[3]{\frac{3\delta}{M_3}}}$$

Substituting this h_{opt} value into the error function, it yields:

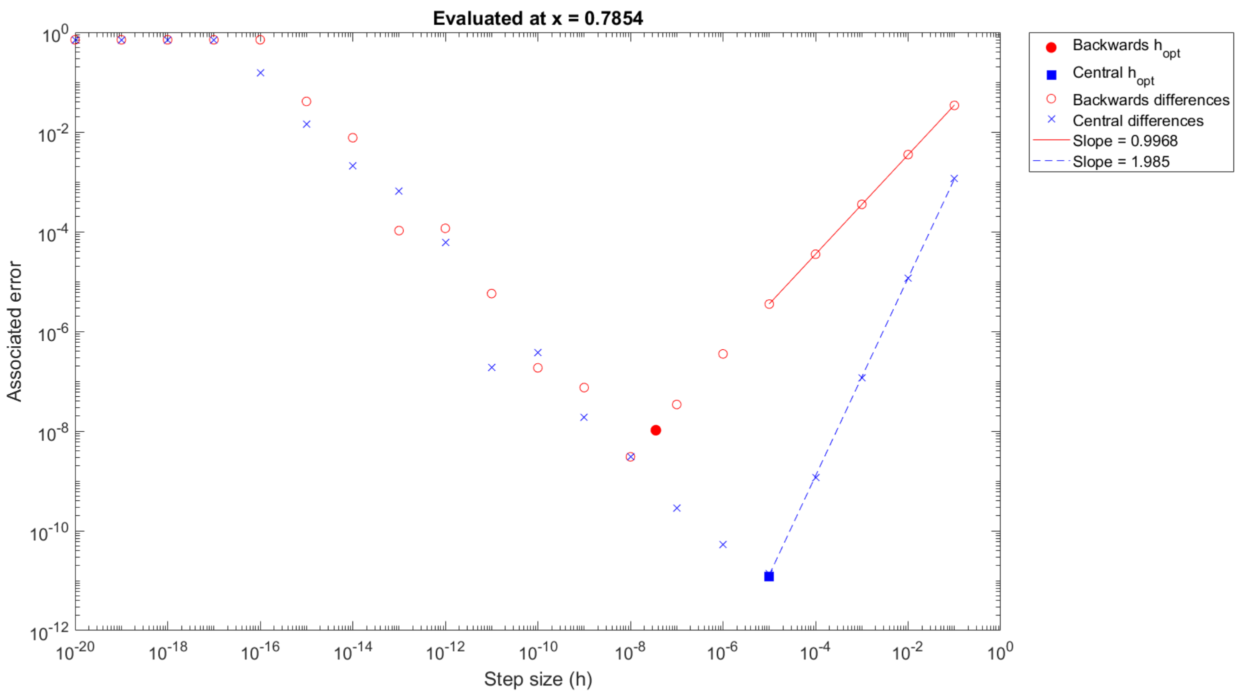
$$\text{errorD}(h_{opt}) = \frac{M_3 \left(\frac{3\delta}{M_3}\right)^{2/3}}{6} + \frac{\delta}{\left(\frac{3\delta}{M_3}\right)^{1/3}} = \frac{M_3 \frac{3\delta}{M_3} + 6\delta}{6\left(\frac{3\delta}{M_3}\right)^{1/3}} \longrightarrow \text{errorD}(h_{opt}) = \frac{\sqrt[3]{9\delta^2 M_3}}{2}$$

B Implementation of optimal step-size and error

The listing of the code is shown at the end of this report. When implementing the optimal step-size, one while-loop for each difference method has been used: given that the optimal step-size is needed to get the optimal step-size, an iterative process is used to get h_{opt}.

When creating the graphs, the way the error is computed with the functions `errorD(`h_{opt}`)` is different from the absolute value of the subtraction between the real value and the approximated one. That's why the latter is chosen in order to represent the same data in the plot.

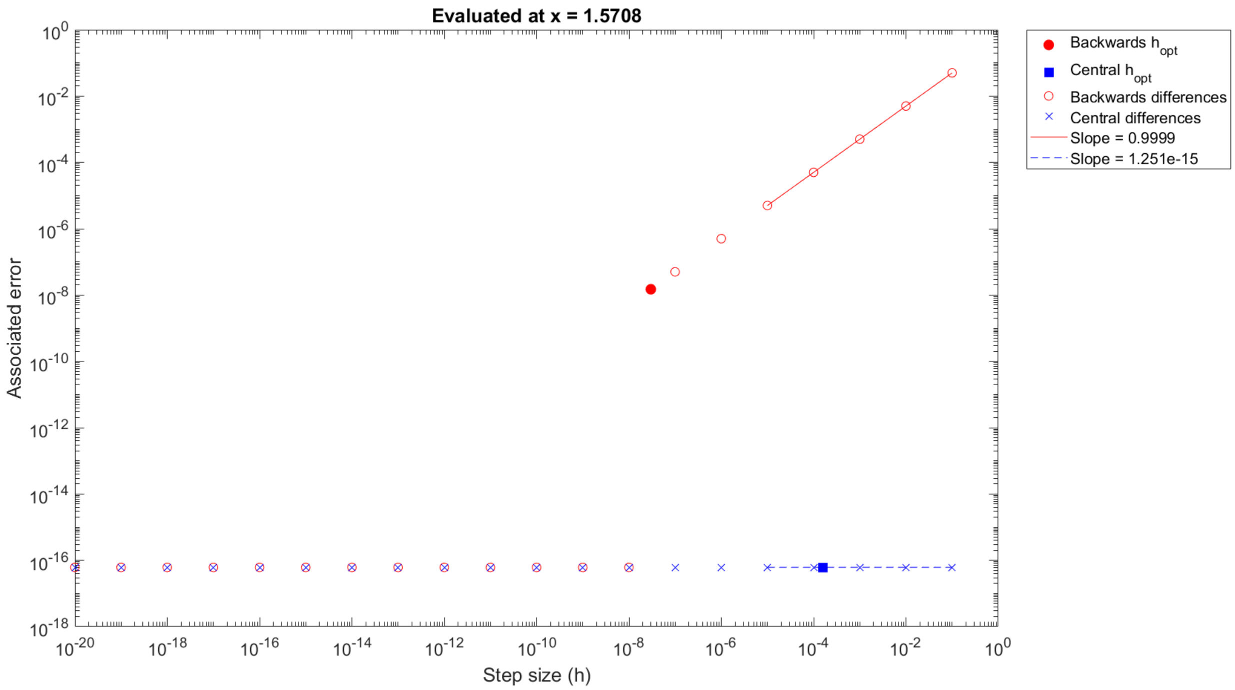
C Results for $x = \pi/4$



In the case when $x = \pi/4$, the plot has three very distinct regions. The regions can be separated depending on the value of the step-size:

- $h > h_{opt}$: if the step-size is greater than the optimal value, the error increases due to truncation errors. $E(h)$ decreases as h decreases. This region can be fitted with a line whose slope is the order of the difference scheme (having $\simeq 1$ for backwards method and $\simeq 2$ for central differences).
- $h_{opt} > h > eps$: if h is between h_{opt} and the machine epsilon of the machine (which for MATLAB is `eps=2.2204e-16`) the region is dominated by round-off errors. $E(h)$ increases as h decreases.
- $eps < h$: making a step smaller than `eps` makes no sense because MATLAB can't represent values smaller than `eps`. Just when $h \simeq$ `eps`, the round-off error is of the order of $\mathcal{O}(1)$ while the truncation error is negligible - that's why for this region the dots follow an horizontal line.

D Results for $x = \pi/2$ and $x = \pi$



When $x = \pi/2$ the value of the derivative is $f'(x) = 0$. Analyzing each difference method:

- Backward difference: h will always be subtracted to $x = \pi/2$, so the maximum will be $M_2 = 1$ and therefore $h_{opt} = 2\sqrt{\texttt{eps}} \simeq 2 \times 10^{-8}$. Analyzing again as before:
  - $h > h_{opt}$: the value of $E(h)$ decreases as the step-size decreases.
  - $h_{opt} > h$: if the step-size is smaller than the optimal, the value of the error goes to $E(h) = \texttt{eps}$
- Central difference: given that $sin(x)$ is symmetrical around the point $x = \pi/2$ (having that $\sin(\pi/2 + h) = \sin(\pi/2 - h)$ so $f'(x) = 0$), the central difference method will not be able to compute $f'(x)$. That's why the error goes straight down to $E(h) = \texttt{eps}$.

When $x = \pi$ (this figure is in the next page), the derivative is $f'(x) = -1$. In this case, the results for both difference methods are the same. Given that $\sin(\pi + h) = -\sin(\pi - h)$ and that $\sin(\pi) = 0$, each method yields:

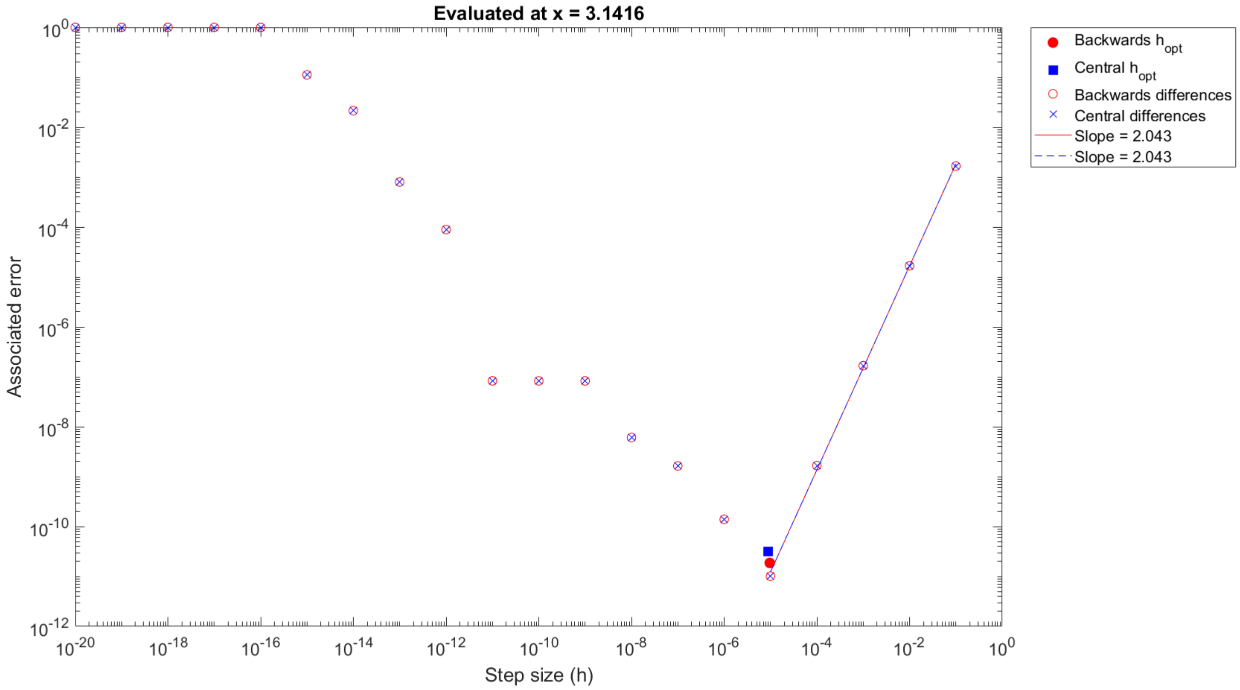
- Backward difference:

$$f'(x) \simeq \frac{\sin(\pi) - \sin(\pi - h)}{h} = -\frac{\sin(\pi - h)}{h}$$

- Central difference:

$$f'(x) \simeq \frac{\sin(\pi + h) - \sin(\pi - h)}{2h} = \frac{-2\sin(\pi - h)}{2h} = -\frac{\sin(\pi - h)}{h}$$

The three zones are the same as the ones shown in the case of $x = \pi/4$. There is one flat region around $h = 10^{-10}$ that can be due to a sampling error.



E Comparison of analytical calculations with empirical data

The set of equations for each method are:

- Backward difference:

$$h_{opt} = \sqrt{\frac{4\texttt{eps}}{M_2}} \qquad E(h_{opt}) = 2\sqrt{\texttt{eps}M_2}$$

- Central difference:

$$h_{opt} = \sqrt[3]{\frac{3\texttt{eps}}{M_3}} \qquad E(h_{opt}) = \frac{\sqrt[3]{9\texttt{eps}^2 M_3}}{2}$$

The behavior of each one of the three cases is:

- $x = \pi/4$: the optimal points match up the empirical data very well. They are located following the slopes for both methods.
- $x = \pi/2$: in this case, for the backward difference method the optimal h is located on the slope line (in case it would be made longer). The central difference method represents the error with its value of `eps`.
- $x = \pi$: both analytical points have small discrepancies with the empirical data obtained.

F Numerical approximation of the second order derivative

To get the approximation, we will use three points: the point in which the second derivative is wanted (x) and two points at the same distance h from that point (having $x+h$ and $x-h$). Using the Taylor series expansion for the function $f(x)$ at each one of the points:

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(x) + \dots \\ f(x) &= f(x) \\ f(x-h) &= f(x) - hf'(x) + \frac{1}{2}h^2 f''(x) - \frac{1}{6}h^3 f'''(x) + \dots \end{aligned}$$

Operating with these equations, it yields:

$$\begin{aligned} f(x+h) - 2f(x) + f(x-h) \simeq\ & f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(x) - \\ & -2f(x) + \\ & +f(x) - hf'(x) + \frac{1}{2}h^2 f''(x) - \frac{1}{6}h^3 f'''(x) \end{aligned}$$

Rearranging and cancelling out the terms:

$$f(x+h) - 2f(x) + f(x-h) \simeq h^2 f''(x) + \mathcal{O}(h^4)$$

Solving for $f''(x)$:

$$\boxed{f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + \mathcal{O}(h^2)}$$

The order of accuracy of the approximation of the second derivative (without knowing the first one) is of second order (having $\mathcal{O}(h^2)$).

Listing of the code used

`comparederivs.m` function

```
function comparederivs(allx, f, truedf1st, truedf2nd, truedf3rd)
% COMPAREDERIVS: % Compares true, 1st-order, and 2nd-order slope approximations
%
% INPUTS:
%   allx: scalar or vector of values to look at
%   f: function handle of the function to differentiate
%   truedf1st: function handle of the analytic 1st derivative function of f
%   truedf2nd: function handle of the analytic 2nd derivative function of f
%   truedf3rd: function handle of the analytic 3rd derivative function of f
%
% OUTPUTS:
%   This program produces one plot for each value in the vector allx
%   The plots show the error in the approximations as a function of step
%   size, including the optimal stepsize and its associated error
%
% sample call: comparederivs([0 5],@exp, @exp, @exp, @exp)

% Maggie Eppstein, 02/10/08; documentation improved 02/15/11
% Xing Jin and Javier Lobato, modified 02/14/18

% THIS IS A GOOD EXAMPLE OF A WELL-DOCUMENTED FILE;  NOTE THE FOLLOWING:
%   a) contents and consistent organization of function headers; first
%   comment line for LOOKFOR command, 1st contiguous comment block for HELP
%   command; always define inputs/outputs, including size constraints or
%   other pre-/post-conditions;
%   b) in-line comments should always be at one level of abstraction higher
%   than the code itself;
%   c) use of full-line UPPER-CASE in-line comments to give a high-level
%   description of what each logically-related code block does; you can
%   read through these alone to get a good understanding of what the code
%   does, without even looking at the code;
%   d) additional lower-case comments at the ends of potentially confusing
%   lines for clarification;


% FOR EACH X-VALUE, PLOT THE APPROXIMATION ERRORS AS A FUNCTION OF STEPSIZE
h = logspace(-20,-1,20); %logarithmically-spaced step sizes

for xi = 1:length(allx) %each x-value will get its own plot
    x = allx(xi); %get the i-th value of tthe allx vector

    % COMPUTE TRUE DERIVATIVE AND ITS APPROXIMATIONS
    df = truedf1st(x); % compute true derivative at x
    bdf = backdiff(x, f, h); %approximate with 1st order backwards difference
    cdf = centraldiff(x, f, h); %approximate with 2nd order central difference

    % COMPUTE APPROXIMATION ERRORS BY COMPARING TO TRUE DERIVATIVES
    berr = abs(df - bdf);
    cerr = abs(df - cdf);

    % In order to compute the optimal error, an iterative process is
```

```
    % required. An initial guess on the stepsize is made. With that initial
    % guess, a value of h_opt is computed, and evaluated again in M2 - in
    % order to get the optimal step size
    opt_stepsize = ones([2,1]);
    opt_stepsize(2) = 1e-6;
    while abs(opt_stepsize(2) - opt_stepsize(1)) > eps
        opt_stepsize(1) = opt_stepsize(2);
        M2 = max(abs(truedf2nd((x-opt_stepsize(1)):x)));
        back_hopt = 2*sqrt(eps/M2);
        opt_stepsize(2) = back_hopt;
    end
    % The error of the backwards difference is 2*sqrt(eps*M2) but it does
    % not give the same result as evaluating the function with h_opt, so
    % the second method is used
    back_optError = abs(df - backdiff(x, f, back_hopt));

    %Following the same procedure for the central differences...
    opt_stepsize = ones([2,1]);
    opt_stepsize(2) = 1e-6;
    while abs(opt_stepsize(2) - opt_stepsize(1)) > eps
        opt_stepsize(1) = opt_stepsize(2);
        M3 = max(abs(truedf3rd((x-opt_stepsize(1)):(x+opt_stepsize(1)))));
        central_hopt = (3*eps/M3)^(1/3);
        opt_stepsize(2) = central_hopt;
    end
    % The error for central differences from the mathematical derivation
    % does not give the same value as if the function is evaluated with
    % h_opt - the second method is again choosen
    central_optError = abs(df - centraldiff(x, f, central_hopt));

    % PLOT THE APPROXIMATION ERRORS AS A FUNCTION OF STEPSIZE
    figure

    % Plotting the optimal step sizes and its associated errors
    loglog(back_hopt, back_optError, 'ro', 'MarkerSize', 8, 'MarkerFaceColor', 'r');
    hold on
    loglog(central_hopt, central_optError, 'bs', 'MarkerSize', 9, 'MarkerFaceColor', 'b');

    % LINEAR REGRESSION OF LOG-LOG RELATIONSHIPS
    % (ONLY IN REGION GOVERNED BY TRUNCATION ERROR)
    coef1 = polyfit(log(h(end-4:end)), log(berr(end-4:end)), 1);
    coef2 = polyfit(log(h(end-4:end)), log(cerr(end-4:end)), 1);

    % Plotting of the different empirical errors for all stepsizes in h
    loglog(h, berr, 'ro', 'MarkerSize', 7);
    loglog(h, cerr, 'bx', 'MarkerSize', 8);

    % Plotting the linear regression lines
    loglog(h(end-4:end), exp(coef1(2))*h(end-4:end).^coef1(1), 'r-');
    loglog(h(end-4:end), exp(coef2(2))*h(end-4:end).^coef2(1), 'b--');

    % Labeling of the plots
    set(gca, 'fontsize', 14) % be kind to the instructor's aging eyes!
    xlabel('Step size (h)')
    ylabel('Associated error')
     legend('Backwards h_{opt}', 'Central h_{opt}','Backwards differences','Central
     ↪  differences',...
```

```
        ['Slope = ',num2str(coef1(1),4)],['Slope = ',num2str(coef2(1),4)],...
        'Location','BestOutside');
    title(['Evaluated at x = ',num2str(x)])

    %ALLOW USER TO VIEW EACH PLOT BEFORE MOVING ON TO THE NEXT
    if xi < length(allx)
        disp('Hit any key to continue...')
        pause
    end

end

figure(gcf)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NOTE: the following functions are placed here for convenience for this
% demo code, but cannot be called from outside this file; in general,
% these should be in their own files (e.g. in a directory for your personal
% library "toolbox" that you add to the Matlab path to access your own
% handy utility functions)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function df1 = backdiff(x, f, h)
% BACKDIFF: 1st order backwards difference approximation to first derivative
%
% INPUTS:
% x: location(s) of where in domain to approximate the derivative
% f: handle of function to approximate derivative of
% h: stepsize(s) to use in approximation
%
% SIZE CONSTRAINTS: at least one of x or h must be a scalar, but the other
% can be of any other dimension (scalar, vector, matrix)
%
% OUTPUTS:
% df1: 1st order approximation to first deriv (slope) of f at x
%      (same size as largest of x or h)
%
% SAMPLE CALLS:
%   df1 = backdiff(0, @sin, [1e-3 1e-2 1e-1]) % vector of stepsizes
%   df1 = backdiff(0:.5:3, @sin, 1e-3) % vector of domain values
%
% AUTHOR: Maggie Eppstein, 2/15/2011

df1 = (f(x)-f(x-h))./h;


function df2=centraldiff(x,f,h)
% CENTRALDIFF: 2nd order central difference approximation to first derivative
%
% INPUTS:
% x: location(s) of where in domain to approximate the derivative
% f: handle of function to approximate derivative of
% h: stepsize(s) to use in approximation
%
% SIZE CONSTRAINTS: at least one of x or h must be a scalar, but the other
% can be of any dimension (scalar, vector, matrix)
```

```
%
% OUTPUTS:
% df2: 2nd order approximation to first deriv (slope) of f at x
%     (same size as largest of x or h)
%
% SAMPLE CALLS:
%   df2 = backdiff(0, @sin, [1e-3 1e-2 1e-1]) % vector of stepsizes
%   df2 = backdiff(0:.5:3, @sin, 1e-3) % vector of domain values
%
% AUTHOR: Maggie Eppstein, 2/15/2011

df2 = (f(x+h)-f(x-h))./(2*h);
```

`truedf2nd.m` function

```
function y=truedf2nd(x)
% Xing Jin and Javier Lobato, modified 02/14/18
% The true analytic value of the 2nd derivative of sin(x) is -sin(x)
y = -sin(x);
```

`truedf3rd.m` function

```
function y=truedf3rd(x)
% Xing Jin and Javier Lobato, modified 02/14/18
% The true analytic value of the 3rd derivative of sin(x) is -cos(x)
y = -cos(x);
```

`hw2aDriver.m` script

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                              HOMEWORK #1.A                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Xing Jin and Javier Lobato, modified 02/14/18

% By executing this driver, the function comparederivs is called for three
% different points (pi/4, pi/2 and pi) applied to the function @sin. The
% first derivative of @sin is known and it is also used as input for the
% function as @cos. The second derivative of sin(x) is -sin(x) and the
% third derivative of sin(x) is -cos(x) (some functions have been created
% to account for the signs)

% Let's clear the workspace
clear all; close all; clc

% Calling to comparederivs with its arguments
comparederivs([pi/4 pi/2 pi], @sin, @cos, @truedf2nd, @truedf3rd);
```