

Note for the Instructors:

This Group Work is part of three series of Complex Numerical Analysis. Complex Numerical Analysis is applying complex variables, instead of usual real variables, to commonly known numerical algorithms. However, often we might have some advantages due to the complex numbers and (analytic) functions properties. Please feel free to modify or edit the Group Work to fit your class. It may further develop to be used as an undergraduate research project in computational mathematics or scientific computing.

The presented Group Work is less known complex step differentiation, where we can avoid the subtractive cancellation due to the independence of real and imaginary axis. It also gives an opportunity to students to understand the effect of round off errors for any finite precision machine. For our experience, adequate group size is two to three students in each group.

Here are some references for numerical differentiation, complex step differentiation, and examples of subtractive cancellation:

Numerical Differentiation - Wikipedia

https://en.wikipedia.org/wiki/Numerical_differentiation

Lyness, James N., and Moler, Cleve B., Numerical Differentiation of Analytic Functions, SIAM J. Numerical Analysis 4, 1967, pp. 202-210. <https://epubs.siam.org/doi/abs/10.1137/0704019>

Squire, William, and Trapp, George, Using complex variables to estimate derivatives of real functions, SIAM Review 40, 1998, pp. 110-112.

<https://epubs.siam.org/doi/abs/10.1137/S003614459631241X>

There are many AI tools that students can access (some are free and some need subscription fee). If your students use the AI tools for the coding, reviewing the code in group or in class is important. (From vibe coding to peer coding) Here is an example of *numerical differentiation with forward finite difference scheme*, generated by Google Gemini (free account):

✦ AI Overview

The forward finite difference approximation for a first derivative has a **first-order convergence** ($O(h)$), meaning the error decreases linearly as the step size h shrinks.  MathWorks

Below is a complete MATLAB code that evaluates the derivative of a sample function $f(x) = \sin(x)$ at a specific point $x_0 = 1$ using successively smaller step sizes, calculates the error, and displays the empirical order of convergence.

MATLAB Code

matlab

```
clear; clc; close all;
```

```

% 1. Define the function and its exact derivative analytically
f = @(x) sin(x);
df = @(x) cos(x);

% Point of interest
x0 = 1.0;
exact_val = df(x0);

% 2. Define a sequence of decreasing step sizes (h)
% e.g., h = 10^-1, 10^-2, ..., 10^-12
h_values = logspace(-1, -12, 12);

% Preallocate arrays for storage
num_derivs = zeros(size(h_values));
errors      = zeros(size(h_values));

% 3. Compute numerical derivative using forward difference for each h
for i = 1:length(h_values)
    h = h_values(i);
    % Forward difference formula: f'(x) ≈ (f(x+h) - f(x)) / h
    num_derivs(i) = (f(x0 + h) - f(x0)) / h;
    errors(i)      = abs(num_derivs(i) - exact_val);
end

% 4. Estimate the order of convergence (slope of Log(error) vs Log(h))
% Error ~ C * h^p => Log(Error) ~ p * Log(h) + Log(C)
p = diff(log(errors)) ./ diff(log(h_values));

% 5. Display results in a formatted table
fprintf('      h          Derivative      Absolute Error      Est. Order (p)\n')
fprintf('-----\n')
fprintf('%10.2e      %12.8f      %14.2e\n', h_values(1), num_derivs(1), errors(1))
for i = 2:length(h_values)
    fprintf('%10.2e      %12.8f      %14.2e      %14.2f\n', ...
        h_values(i), num_derivs(i), errors(i), p(i-1));
end

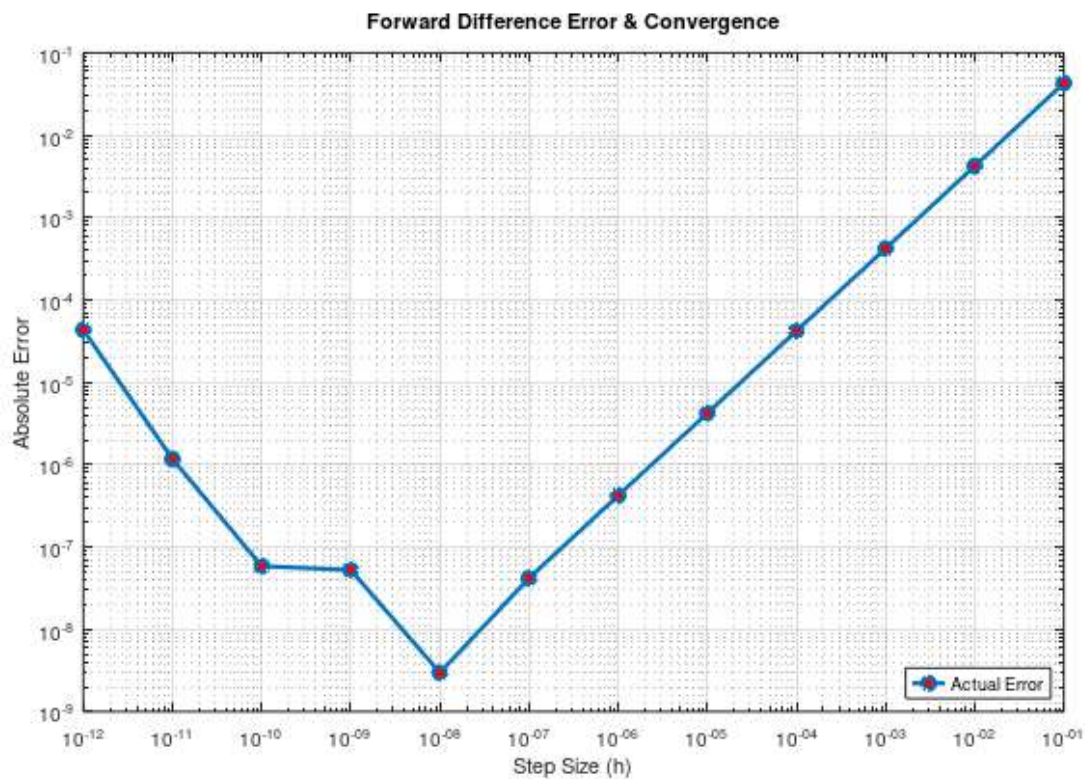
% 6. Plot the error convergence Log-Log graph
figure;
loglog(h_values, errors, '-o', 'LineWidth', 2, 'MarkerFaceColor', 'r');
grid on;
xlabel('Step Size (h)');
ylabel('Absolute Error');
title('Forward Difference Error & Convergence');
legend('Actual Error', 'Location', 'Southeast');

```

And here is the result of the computation of the above Matlab code, which shows both the errors for each step size and the order of convergence. It also shows graphical result of the order of convergence, which shows the forward finite difference scheme is ill-conditioned.

h	Derivative	Absolute Error	Est. Order (p)
1.00e-01	0.49736375	4.29e-02	
1.00e-02	0.53608598	4.22e-03	1.01
1.00e-03	0.53988148	4.21e-04	1.00
1.00e-04	0.54026023	4.21e-05	1.00
1.00e-05	0.54029810	4.21e-06	1.00
1.00e-06	0.54030189	4.21e-07	1.00
1.00e-07	0.54030226	4.18e-08	1.00
1.00e-08	0.54030230	2.97e-09	1.15
1.00e-09	0.54030236	5.25e-08	-1.25
1.00e-10	0.54030225	5.85e-08	-0.05
1.00e-11	0.54030114	1.17e-06	-1.30
1.00e-12	0.54034555	4.32e-05	-1.57

Table 1: Absolute Errors and the Estimated Order of Convergence



Graph 1: The Order of Convergence in Absolute Error & Step Size in Log-Log scale

Note that *numerical differentiation with forward finite difference scheme* can be derived from the Taylor series expansion, and the order of convergence is 1, as one can see from the truncation error as follows:

For a smooth function $f(x)$, by Taylor series expansion,

$$f(x+h) = f(x) + f'(x)h + \frac{f''(\xi)h^2}{2!},$$

where ξ is between x and $x+h$. Rewriting for $f'(x)$, we have

$$f'(x) = \frac{f(x+h) - f(x)}{h} + \frac{f''(\xi)h}{2!} = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h).$$

Now for the Complex Step Differentiation, similar computation shows the order of convergence as 2 as follows. In addition, it has no subtractive cancellation, and as a result, the algorithm is more stable numerical method.

By Taylor series expansion again, but with complex variable, we have

$$f(x+ih) = f(x) + f'(x)ih - \frac{f''(x)h^2}{2!} - \frac{f'''(x)ih^3}{3!} + \dots$$

Which shows,

$$f'(x) = \frac{\text{Im}(f(x+ih))}{h} + \mathcal{O}(h^2).$$

Here is a sample Matlab code for Complex Step Differentiation:

```
% Complex Step Differentiation Example with
% Centered Finite Difference Scheme for comparison
clear; clc; close all;

% define a test function
F = @(x) exp(x)./((cos(x)).^3 + (sin(x)).^3)

% complex step vs centered finite difference approximation
Fpc = @(x,h) imag(F(x+i*h))/h;
Fpd = @(x,h) (F(x+h) - F(x-h))/(2*h);

%Compare them for a sequence of decreasing h
format long
disp([' h complex step finite difference'])
for h = 10.^(-(1:16))
    disp([h Fpc(pi/4,h) Fpd(pi/4,h)])
end

% Differentiate and evaluate the derivative at pi/4
% with the Symbolic Toolbox
syms x
F = exp(x)/((cos(x))^3 + (sin(x))^3);
Fps = simple(diff(F));
```

```

exact = subs(Fps,pi/4);
flexact = double(exact);

% Error Plot in Log-Log scale
h = 10.^(-(1:16)');
errs = zeros(16,2);
for k = 1:16
    errs(k,1) = abs(Fpc(pi/4,h(k))-exact);
    errs(k,2) = abs(Fpd(pi/4,h(k))-exact);
end
loglog(h,errs)
axis([eps 1 eps 1])
set(gca,'xdir','rev')
legend('complex step','finite difference','location','southwest')
xlabel('step size h')
ylabel('error')

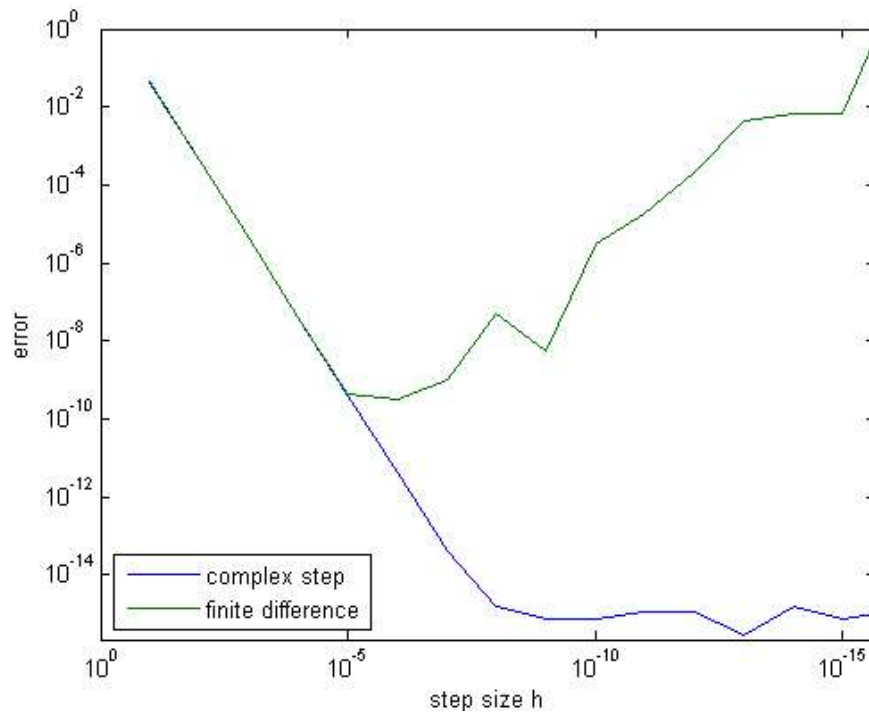
```

Here is the result of the above Matlab code: Note that the Centered Finite Difference Scheme also has the order of convergence as 2.

h	complex step	finite difference
0.1000000000000000	3.144276040634560	3.061511866568119
0.0100000000000000	3.102180075411270	3.101352937655877
0.0010000000000000	3.101770529535847	3.101762258158169
0.0001000000000000	3.101766435192940	3.101766352480162
0.0000100000000000	3.101766394249620	3.101766393398542
0.0000010000000000	3.101766393840188	3.101766393509564
0.0000001000000000	3.101766393836091	3.101766394841832
0.0000000100000000	3.101766393836053	3.101766443691645
0.0000000010000000	3.101766393836052	3.101766399282724
0.0000000001000000	3.101766393836052	3.101763290658255
0.0000000000100000	3.101766393836053	3.101785495118747
0.0000000000010000	3.101766393836053	3.101963130802687
0.0000000000001000	3.101766393836052	3.097522238704187
0.0000000000000100	3.101766393836053	3.108624468950438
0.0000000000000001	3.101766393836052	3.108624468950438
0.0000000000000000	3.101766393836053	8.881784197001252

Table 2: Complex Step vs. Centered Finite Difference Scheme

Finally, the order of convergence for both Complex Step Differentiation, and the Centered Finite Difference Scheme are plotted in a log-log scale graph for comparison. Notice that both algorithms start with the same order of convergence, but the Centered Finite Difference Scheme starts to fail to maintain the accuracy when the step size is getting smaller.



Graph 2: The order of convergence comparison for the Complex Step Differentiation, and the Centered Finite Difference Scheme