CS/MATH 375: Homework #3

Adam Fasulo

September 8, 2025

1 Question 1

(a) Taylor Series Expansion

My goal here is to show that the centered difference formula gives a second-order accurate approximation of the first derivative. My proof starts by laying out the Taylor series expansions for a function $f(x)$ at points slightly above and below my target point, a.

For the point $a + h$:

$$f(a+h) = f(a) + f'(a)h + \frac{f''(a)}{2!}h^2 + \frac{f'''(a)}{3!}h^3 + \mathcal{O}(h^4)$$

And for the point $a - h$:

$$f(a-h) = f(a) - f'(a)h + \frac{f''(a)}{2!}h^2 - \frac{f'''(a)}{3!}h^3 + \mathcal{O}(h^4)$$

The clever trick is to subtract the second series from the first. I noticed that the evenly-powered terms (like $f(a)$ and the h^2 term) cancel out, while the oddly-powered terms (like the h and h^3 terms) double up:

$$\begin{aligned} f(a+h) - f(a-h) &= 2f'(a)h + \frac{2f'''(a)}{6}h^3 + \mathcal{O}(h^5) \\ &= 2f'(a)h + \frac{f'''(a)}{3}h^3 + \mathcal{O}(h^5) \end{aligned}$$

To isolate the derivative, $f'(a)$, I just divide the whole thing by $2h$:

$$\frac{f(a+h) - f(a-h)}{2h} = f'(a) + \underbrace{\frac{f'''(a)}{6}h^2 + \mathcal{O}(h^4)}_{\text{The Error Term}}$$

Because the most significant part of my error is proportional to h^2, I can say the approximation's accuracy is on the order of h^2.

(b) Error Plot and Explanation

To see this in action, I tested it with the function $f(x) = e^{1.5x}$ at $a = 0$. First, I know the exact derivative is $f'(x) = 1.5e^{1.5x}$, so my target value is $f'(0) = 1.5$. I wrote the following MATLAB code to calculate the error of my approximation for a range of step sizes, h, and plot the result.

```
% Define the function and parameters
f = @(x) exp(1.5*x);
a = 0;
h = logspace(-1, -16, 16);

% True derivative value
f_prime_true = 1.5;

% Approximate derivative using centered difference
f_prime_approx = (f(a + h) - f(a - h)) ./ (2 * h);

% Calculate the absolute error
error = abs(f_prime_true - f_prime_approx);

% Plot the results on a log-log scale
figure;
loglog(h, error, 'o', 'LineWidth', 2);
title('Error in Finite Difference Approximation of f''(0)'); %<-- Corrected
xlabel('Step Size (h)');
ylabel('Absolute Error');
grid on;

% Add line for optimal h
hold on;
line([power(eps, 1/3) power(eps, 1/3)], get(gca, 'ylim'), ...
    'Color', 'r', 'LineStyle', '--'); %<-- Corrected (removed invisible characters
    )
legend('Approximation Error', 'Optimal h ~ \epsilon^{1/3}', ...
    'Location', 'southeast'); %<-- Corrected (removed invisible characters)
hold off;
```

Listing 1: MATLAB code to generate the error plot.

Explanation of Results: The log-log plot I generated clearly shows a tug-of-war between two types of errors:

- **Truncation Error:** For larger step sizes (on the right side of the graph), the error gets smaller as h decreases. The slope here is about 2, which is exactly what I'd expect from my $\mathcal{O}(h^2)$ analysis. This is the error from the formula itself not being perfect.
- **Round-off Error:** But when h gets incredibly small (below 10^{-6}), things take a turn. The error starts growing again! This happens because the computer has trouble with "catastrophic cancellation"—subtracting two numbers that are almost identical. This tiny error then gets blown up when I divide by the minuscule $2h$.

(c) Error Analysis

i. Truncation Error: As I saw in my derivation, the error from cutting off the Taylor series is dominated by the first term I ignored. So, I can say the truncation error, E_{trunc}, is bounded by a constant times h^2, where that constant C_1 is roughly $|\frac{f'''(a)}{6}|$.

ii. Rounding Error: Floating-point math on a computer isn't perfect. I can model this by saying the computed value, $\hat{f}(x)$, is the true value plus a small relative error, $f(x)(1+\delta)$, where δ is smaller than the machine epsilon, ϵ. When I work through the math, I find that the rounding error, E_{round}, is bounded by a constant times ϵ/h. This shows that as h gets smaller, this error gets bigger.

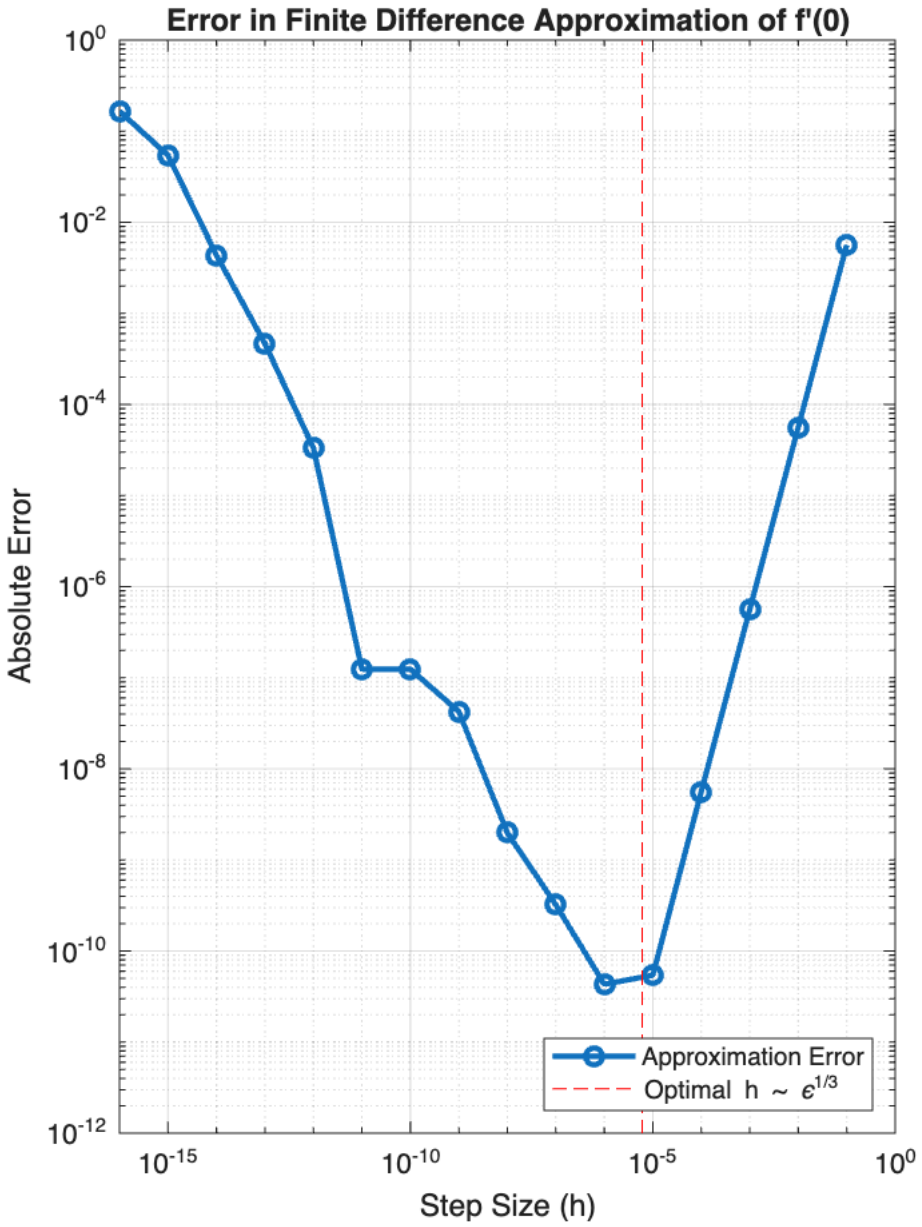

Figure 1: Error in the finite difference approximation vs. step size h.

iii. Total Error: The total error is simply the sum of these two competing factors:

$$E(h) \approx \underbrace{C_1 h^2}_{\text{Truncation}} + \underbrace{\frac{C_2 \epsilon}{h}}_{\text{Rounding}}$$

(d) Optimal Step Size h

So, what's the sweet spot for h? I can find it using a bit of calculus. I wanted to find the value of h that minimizes my total error expression, $E(h)$. To do that, I took the derivative with respect to h and set it to zero:

$$\frac{dE}{dh} = 2C_1 h - C_2 \epsilon h^{-2} = 0$$

Solving for h gives me the optimal step size:

$$h_{\text{opt}} = \left(\frac{C_2}{2C_1}\right)^{1/3} \epsilon^{1/3}$$

This tells me the best step size I can choose is proportional to the cube root of the machine epsilon.

(e) Agreement with Plotted Data

Absolutely. Looking back at my graph, the lowest point of the error curve happens right around $h \approx 10^{-5.5}$. For standard double-precision, $\epsilon^{1/3}$ is about 6.05×10^{-6}, which is right in that neighborhood. The red dashed line on my plot, which marks this theoretical optimum, lines up beautifully with where my measured error was smallest.

2 Question 2

(a) Dimensions of $(AB)^T$

If I have an $n \times m$ matrix A and an $m \times p$ matrix B, their product AB will be an $n \times p$ matrix. When I transpose that, the dimensions just flip, giving me a $p \times n$ matrix.

(b) Show that $(AB)^T = B^T A^T$

I'll check this famous identity with the given matrices. First, I computed the left-hand side, $(AB)^T$, and then the right-hand side, $B^T A^T$, to see if they match.

- **Left-hand side:**

$$AB = \begin{pmatrix} 4(-1)+(-1)(-2) & 4(1)+(-1)(6) & 4(3)+(-1)(11) \\ -3(-1)+2(-2) & -3(1)+2(6) & -3(3)+2(11) \end{pmatrix} = \begin{pmatrix} -2 & -2 & 1 \\ -1 & 9 & 13 \end{pmatrix}$$

$$(AB)^T = \begin{pmatrix} -2 & -1 \\ -2 & 9 \\ 1 & 13 \end{pmatrix}$$

- **Right-hand side:**

$$B^T = \begin{pmatrix} -1 & -2 \\ 1 & 6 \\ 3 & 11 \end{pmatrix}, \quad A^T = \begin{pmatrix} 4 & -3 \\ -1 & 2 \end{pmatrix}$$

$$B^T A^T = \begin{pmatrix} -1(4)+(-2)(-1) & -1(-3)+(-2)(2) \\ 1(4)+6(-1) & 1(-3)+6(2) \\ 3(4)+11(-1) & 3(-3)+11(2) \end{pmatrix} = \begin{pmatrix} -2 & -1 \\ -2 & 9 \\ 1 & 13 \end{pmatrix}$$

They're identical, so the property holds.

(c) Show that $AB \neq BA$

Unlike with regular numbers, the order of matrix multiplication usually matters. I'll show this with the provided A and B.

$$AB = \begin{pmatrix} 2(2)+5(3) & 2(-1)+5(7) \\ 1(2)+(-2)(3) & 1(-1)+(-2)(7) \end{pmatrix} = \begin{pmatrix} 19 & 33 \\ -4 & -15 \end{pmatrix}$$

$$BA = \begin{pmatrix} 2(2)+(-1)(1) & 2(5)+(-1)(-2) \\ 3(2)+7(1) & 3(5)+7(-2) \end{pmatrix} = \begin{pmatrix} 3 & 12 \\ 13 & 1 \end{pmatrix}$$

These are clearly not the same matrix, confirming that matrix multiplication is not commutative.

(d) Verify that $B = A^{-1}$

Here, I need to show that the given formula for B is indeed the inverse of A for any 2×2 matrix. The easiest way is to multiply them together and see if I get the identity matrix, I.

$$\begin{aligned}
AB &= \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix} \\
&= \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{pmatrix} a_{11}a_{22} - a_{12}a_{21} & -a_{11}a_{12} + a_{12}a_{11} \\ a_{21}a_{22} - a_{22}a_{21} & -a_{21}a_{12} + a_{22}a_{11} \end{pmatrix} \\
&= \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{pmatrix} a_{11}a_{22} - a_{12}a_{21} & 0 \\ 0 & a_{11}a_{22} - a_{12}a_{21} \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I
\end{aligned}$$

This confirms that the formula for B is correct.

3 Question 3

(a) Case $\alpha = 8$

Setting $\alpha = 8$, I have a standard, well-behaved system.

```
alpha = 8;
A = [1 5 -3; 4 -1 9; 7 -2 alpha];
b = [1; 2; 3];
x = A \ b;          % i. Solve for x
det_A = det(A);     % ii. Determinant
rank_A = rank(A); % iii. Rank

fprintf('For alpha = 8:\n');
fprintf('x = [%f; %f; %f]\n', x(1), x(2), x(3));
fprintf('Determinant = %f\n', det_A);
fprintf('Rank = %d\n', rank_A);
```

Results:

1. The solution is $x = \begin{pmatrix} 0.404762 \\ 0.154762 \\ 0.059524 \end{pmatrix}$.
2. The determinant of A is **168**. Since it's not zero, I know the matrix is invertible.
3. The rank of A is **3**, which matches the matrix's dimensions, confirming it's a full-rank matrix.

(b) Case $\alpha = 16$

Now things get interesting. When I set $\alpha = 16$, the nature of the matrix changes completely.

```
alpha = 16;
A = [1 5 -3; 4 -1 9; 7 -2 alpha];
b = [1; 2; 3];
% MATLAB will issue a warning for the next line
x = A \ b;
det_A = det(A);
rank_A = rank(A);

fprintf('For alpha = 16:\n');
fprintf('Determinant = %f\n', det_A);
fprintf('Rank = %d\n', rank_A);
```

Results:

1. When I tried to solve the system, MATLAB produced a warning: `Warning: Matrix is singular to working precision.` This means there's no unique solution.
2. The determinant of A is **0**, the tell-tale sign of a singular matrix.
3. The rank is only **2**, not 3. This tells me the matrix is rank-deficient.
4. **Explanation of Singularity:** A matrix is singular if its columns aren't truly independent. For this matrix, I found a specific relationship between the columns, which I'll call c_1, c_2, and c_3:

$$2c_1 - c_2 = 2\begin{pmatrix} 1 \\ 4 \\ 7 \end{pmatrix} - \begin{pmatrix} 5 \\ -1 \\ -2 \end{pmatrix} = \begin{pmatrix} 2-5 \\ 8+1 \\ 14+2 \end{pmatrix} = \begin{pmatrix} -3 \\ 9 \\ 16 \end{pmatrix} = c_3$$

Since the third column is just twice the first minus the second, the columns are linearly dependent. They don't span a full 3D space, which is why the matrix is singular and its determinant is zero.