AMSC460: Homework 1

James Zhang[*]

September 3, 2024

0. **Review Questions**

(a) Scientific computing aims to use computers to use discrete or piecewise solutions to empirically approximate a continuous problem. Algorithms are employed to solve these problems efficiently, accurately, and reliably. on the other hand, the idea behind these algorithms is numerical analysis.

(b) If $|u|$ is large, then relative error is used and if $|u|$ is very small, then relative error is not a good measurement. For example, if $u = 1000$ and $v = 1001$, then relative error is better because relative error is 0.001 whereas the absolute error is 1 and looks worse, when in reality, this model is quite accurate. On the other hand, if $u = 0.00001$ and $v = 0.00003$, then the relative error is 2 which looks huge, but in reality, the model made a good prediction because the absolute error is only 0.00002.

(c) A major difference between the nature of discretization error and roundoff error is that *any* computation with a real number involves roundoff error. Furthermore, unlike roundoff errors, discretization errors have a smooth structure which can be exploited. In practice, in successful model approximations, discreitzation errors dominate roundoff errors in terms of magnitude.

(d) Roundoff errors are inevitable due to the way all computers store floating point numbers. If the accumulated roundoff error is smaller than the problem's required level of precision, then the predicted model output is okay; otherwise the algorithm needs to be redefined and re-executed such that the rounding errors do not compound throughout calculations. Usually, linear accumulation of errors is unpreventable, whereas exponential error accumulation cannot be tolerated.

(e) Accuracy is how close the model output is to the ground truth. Efficiency describes how long the algorithm and computation take to run. Efficiency is affected by the researcher's choices of computer language, algorithmic implementation, underlying hardware, and it can be affected by theoretical properties such as the rate of convergence of the algorithm. Robustness is if the solution algorithm works under various conditions. For example, returning the correct answer within tolerable error bounds, or returning the correct error in unsuitable conditions.

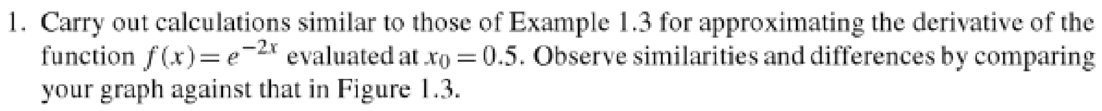
[*]Email: jzhang72@terpmail.umd.edu

(f) Note that any given polynomial can be rewritten as

$$P(x) = ((((a_n x + a_{n-1})x) + a_{n-2})x + \cdots + a_1)x + a_0$$

Each degree requires one addition and one multiplication, and so therefore there are $n + n = 2n$ operations, which is $\mathcal{O}(n)$ time complexity.

(g) Problem conditioning describes when a small change in the data produces a drastic change in the solution algorithm's output, whereas algorithm stability describes the solution's robustness to small pertubations and errors that occur during the data.



1. Carry out calculations similar to those of Example 1.3 for approximating the derivative of the function $f(x) = e^{-2x}$ evaluated at $x_0 = 0.5$. Observe similarities and differences by comparing your graph against that in Figure 1.3.

Solution. Using a Taylor Series to approximate this model yields

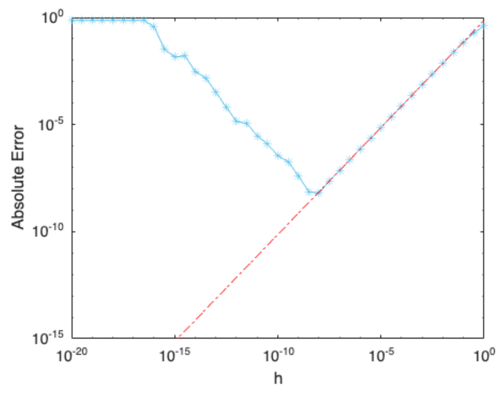
$$f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{h^2}{2}f''(x_0) + \cdots$$

Solving for $f'(x_0)$, we get

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0) - \frac{h^2}{2}f''(x_0)}{h} = \frac{f(x_0 + h) - f(x_0)}{h} - \frac{h}{2}f''(x_0)$$

Below is the code and the graph showing the combined effect of discretization and roundoff errors, and a dashed line showing the discretization error without roundoff error.

```
x0 = 0.5;
e = exp(1);
f0 = e^(-2 * x0);
fp = -2 * f0;
i = -20:0.5:0;
h = 10.^i;
err = abs(fp - (e.^(-2 * (x0 + h)) - f0)./h);
d_err = 2.*h * e^(-2 * x0);
loglog(h, err, '-*');
hold on;
loglog(h, d_err, 'r-.');
xlabel('h')
ylabel('Absolute Error')
```



This graph illustrates a lot of similarities to Figure 1.3 in the book. For example, the absolute error is relatively constant slightly under $10^0 = 1$ in both graphs. At $x = 10^{-14}$. both errors start decreasing almost linearly. However, at around $x = 10^{-8}$, the errors start increasing linearly and follow the red-dashed line, which means that the absolute error gets ver similar to the discretization error without roundoff error. □

2. Carry out derivation and calculations analogous to those in Example 1.2, using the expression

$$\frac{f(x_0+h)-f(x_0-h)}{2h}$$

for approximating the first derivative $f'(x_0)$. Show that the error is $\mathcal{O}(h^2)$. More precisely, the leading term of the error is $-\frac{h^2}{3}f'''(x_0)$ when $f'''(x_0) \neq 0$.

Solution. Note that using Taylor Series approximation, we immediately get both of the following

$$f(x_0+h) = f(x_0) + hf'(x_0) + \frac{h^2}{2}f''(x_0) + \frac{h^3}{3!}f'''(x_0) + \cdots \tag{1}$$

$$f(x_0-h) = f(x_0) - hf'(x_0) + \frac{h^2}{2}f''(x_0) - \frac{h^3}{3!}f'''(x_0) + \cdots \tag{2}$$

where for the $f(x_0 - h)$ expansion, the terms with h raised to an odd power are substracted. Now, when we subtract (2) from (1), observe that

$$f(x_0+h) - f(x_0-h) = 2hf'(x_0) + \frac{2h^3}{3!}f'''(x_0) + \frac{2h^5}{5!}f^{(5)}(x_0) + \cdots$$

Dividing every term by $2h$, we get

$$f'(x_0) = \frac{f(x_0+h)-f(x_0-h)}{2h} - \frac{h^2}{3!}f'''(x_0) - \frac{h^4}{5!}f^{(5)}(x_0) + \cdots \tag{3}$$

Thus, the error is

$$\text{Error} = f'(x_0) - \frac{f(x_0+h)-f(x_0-h)}{2h} = -\frac{h^2}{3!}f'''(x_0) - \cdots \tag{4}$$

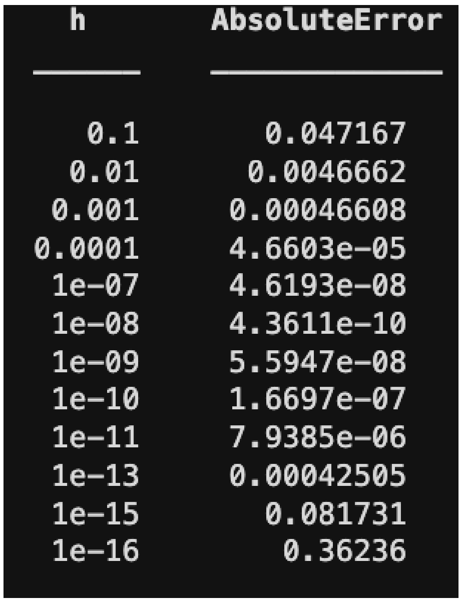
To show that the error is $\mathcal{O}(h^2)$, let $C = 100$ and let $q = 2$. As h decreases, the h^2 term will be the largest contributing term, and for all h small enough, we will have the following

$$|e| \leq 100h^2$$

$$\frac{h^2}{3!}f'''(x_0) + \frac{h^4}{5!}f^{(5)}(x_0) + \cdots + \frac{h^{2i}}{(2i+1)!}f^{(2i+1)}(x_0) \leq 100h^2, \text{ for } i \in \mathbb{N}$$

It's clear that when $f'''(x_0) \neq 0$, the leading term of the error is nonzero and it is $-\frac{h^2}{3!}f'''(x_0)$.

```
x0 = 1.2;
f0 = sin(x0);
fp = cos(x0);
h = [1e-1, 1e-2, 1e-3, 1e-4, 1e-7, 1e-8, 1e-9, 1e-10, 1e-11, 1e-13,
    1e-15, 1e-16];
err = abs(fp - (sin(x0 + h) - sin(x0)) ./ h);
loglog(h, err, '-*');
hold on;
xlabel('h')
ylabel('Absolute Error')
% Create a table
T = table(h', err', 'VariableNames', {'h', 'AbsoluteError'});
% Display the table
disp(T);
```



```
   h         AbsoluteError
 ______     _____________

    0.1        0.047167
   0.01       0.0046662
  0.001      0.00046608
 0.0001      4.6603e-05
  1e-07      4.6193e-08
  1e-08      4.3611e-10
  1e-09      5.5947e-08
  1e-10      1.6697e-07
  1e-11      7.9385e-06
  1e-13      0.00042505
  1e-15        0.081731
  1e-16         0.36236
```

□