COE 347: Homework 4

William Zhang, February 2024

Note: All code used in this homework can be found on my GitHub profile: https://github.com/williamzhang0306/COE347/tree/main/homework_4

Homework 4

1) Consider the second order finite difference approximation

$$u''(x_i)h^2 = u_{i+1} - 2u_i + u_{i-1} + Ch^4 u^{(4)}(x_i) + h.o.t$$

Find the value of C by using Taylor's expansion.

By Taylor expansion of $u(x_{i+1})$ and $u(x_{i-1})$ centered at x_i,

$$u_{i+1} = u(x_i+h) = u(x_i) + hu'(x_i) + \frac{h^2}{2!}u''(x_i) + \frac{h^3}{3!}u'''(x_i) + \frac{h^4}{4!}u^{(4)}(x_i) + h.o.t$$

$$u_{i-1} = u(x_i-h) = u(x_i) - hu'(x_i) + \frac{h^2}{2!}u''(x_i) - \frac{h^3}{3!}u'''(x_i) + \frac{h^4}{4!}u^{(4)}(x_i) + h.o.t$$

Now substitute,

$$u_{i+1} - 2u_i + u_{i-1} = u(x_i) + hu'(x_i) + \frac{h^2}{2!}u''(x_i) + \frac{h^3}{3!}u'''(x_i) + \frac{h^4}{4!}u^{(4)}(x_i) + h.o.t$$
$$- 2u(x_i)$$
$$+ u(x_i) - hu'(x_i) + \frac{h^2}{2!}u''(x_i) - \frac{h^3}{3!}u'''(x_i) + \frac{h^4}{4!}u^{4}(x_i) + h.o.t$$

$$u_{i+1} - 2u_i + u_{i-1} = 2\frac{h^2}{2!}u''(x_i) + 2\frac{h^4}{4!}u^4(x_i) + h.o.t$$

$$u_{i+1} - 2u_i + u_{i-1} = h^2u''(x_i) + \frac{1}{12}h^4u^{(4)}(x_i) + h.o.t$$

Rearrange terms,

$$u''(x_i)h^2 = u_{i+1} - 2u_i + u_{i-1} - \frac{1}{12}h^4u^{(4)}(x_i) + h.o.t$$

This equation matches the postulated form.

It follows that $C = -\frac{1}{12}$.

Problem 2

The Python implementation that uses the second order central difference approximation to solve the provided ODE can be found in the file `homework_functions.py`. The function is called `centered_diff()`.

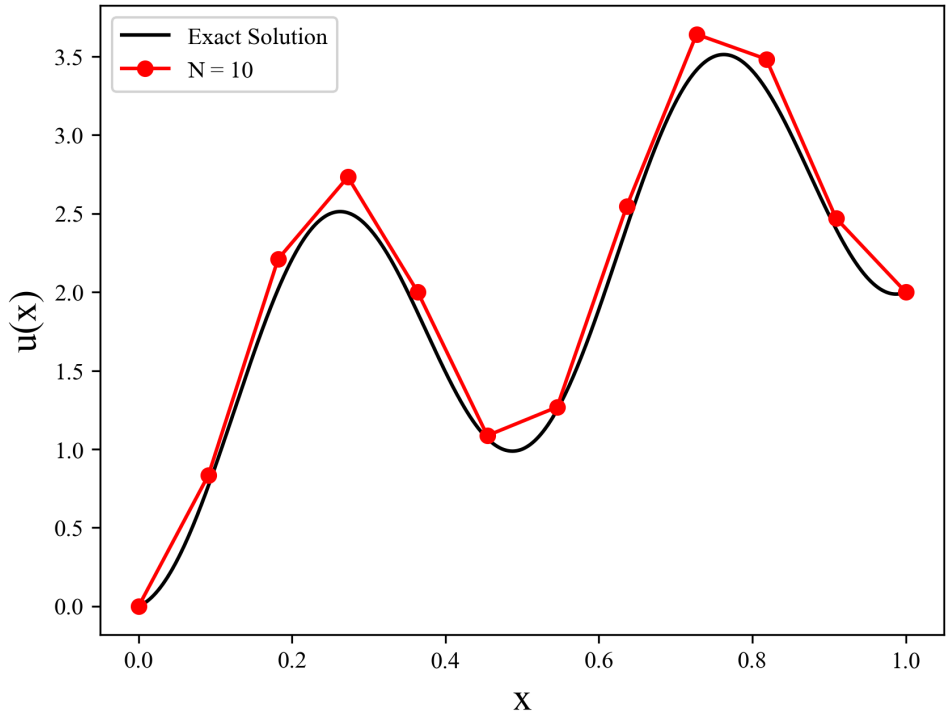

Figure 1: Numerical Solution for N = 10 and Exact Solution to ODE

Table 1: Numerical Solutions for N = 10

i	x_i	u_i	i	x_i	u_i
0	0.000	0.000	6	0.545	1.268
1	0.091	0.834	7	0.636	2.548
2	0.182	2.211	8	0.727	3.642
3	0.273	2.733	9	0.818	3.484
4	0.364	2.002	10	0.909	2.471
5	0.455	1.086	11	1.000	2.000

Problem 3

The Python implenetations for the calculations of E and e can be found in `homework_funcitons.py` and are called `error_1()` and `error_2()` respectively. For various values of N, the errors E and e were calculated. The the errors were plotted against $h = \frac{1}{N+1}$ on a log-log plot.

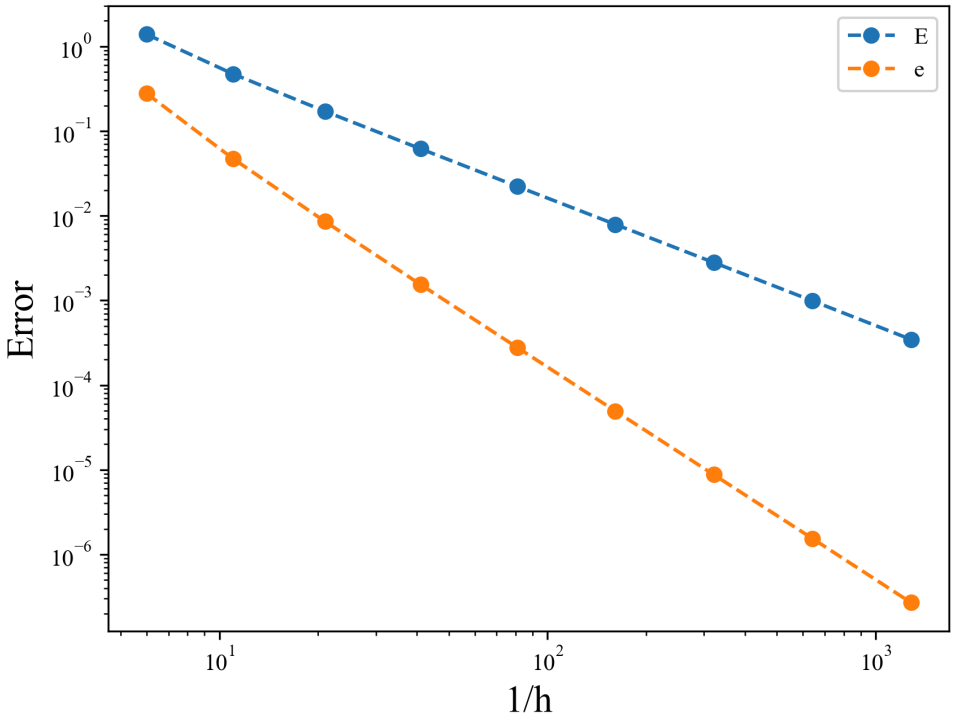

Figure 2: Errors (E and e) versus 1/h

Then a power law relation between h and the errors was fitted: $E = C_1 h^{\alpha_1}$, $e = C_2 h^{\alpha_2}$. It was found that for E, $\alpha = 1.531$ and for e, $\alpha = 2.5578$. The order of error e is approximately one order greater than E, which is what we expect since $e = \frac{1}{N} E \approx hE$ especially for values of $N \gg 1$. Additionally, while the centered difference formula is considered a second order approximation, neither of the calculated errors are appear to be second order. For example, in the calculation of E, a the sum of squared errors for N points is taken and then the square root of that value is returned. If the error $u_i - \widehat{u_i}$ is $O(h^2)$, then

$$E = (N \cdot O(h^2))^{1/2} = O(h^3)^{1/2} = O(h^{1.5})$$

Similarly,

$$e = \frac{1}{N}(N \cdot O(h^2))^{1/2} = h \cdot O(h^3)^{1/2} = O(h^{2.5})$$

4) Using Taylor's expansion method derive a one sided finite difference formula that approximates $u'(x)$ as

$$u'(x_i)h = a u_i + b u_{i+1} + c u_{i+2} + O(h^3)$$

by finding values of the coefficients a, b, c.

By Taylor expansion of u_i, u_{i+1}, u_{i+2} centered at x_i,

$u_i = u(x_i)$

$u_{i+1} = u(x_i) + h u'(x_i) + \frac{h^2}{2!} u''(x_i) + O(h^3)$

$u_{i+2} = u(x_i) + 2h u'(x_i) + 4\frac{h^2}{2!} u''(x_i) + O(h^3)$

Now substituting,

$u'(x_i)h = a u_i + b u_{i+1} + c u_{i+2}$

$u'(x_i)h = a u(x_i)$

$+ b u(x_i) + b u'(x_i)h + \frac{b}{2!} u''(x_i)h^2 + O(h^3)$

$+ c u(x_i) + 2c u'(x_i)h + 4c\frac{h^2}{2!} u''(x_i) + O(h^3)$

$u'(x_i)h = (a+b+c)u(x_i) + (b+2c)u'(x_i)h + (\frac{b}{2} + 2c)u''(x_i) + O(h^3)$

By the postulated form of $u'(x_i)h$,

$$\begin{cases} a+b+c = 0 \\ b+2c = 1 \\ \frac{1}{2}b+2c = 0 \end{cases}$$

After solving the system $a = -3/2$, $b = 2$, $c = -\frac{1}{2}$

Problem 5

The Python implementation that uses a second order one-sided finite difference approximation to $u'(0)$ and a second order central difference to numerically solve the ODE can be found in the file `homework_functions.py`. The function is called `problem_5_solver()`.

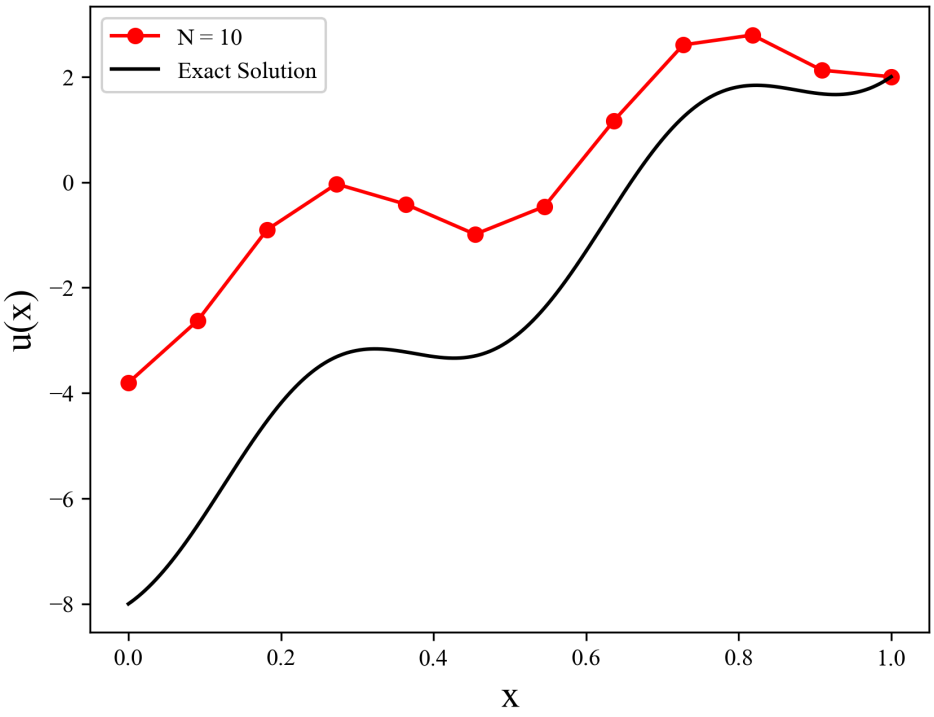

Figure 3: Numerical Solution for N = 10 and Exact Solution to ODE

Table 2: Numerical Solutions for N = 10

i	x_i	u_i	i	x_i	u_i
0	0.000	-3.804	6	0.545	-0.461
1	0.091	-2.625	7	0.636	1.165
2	0.182	-0.901	8	0.727	2.604
3	0.273	-0.034	9	0.818	2.792
4	0.364	-0.418	10	0.909	2.125
5	0.455	-0.989	11	1.000	2.000

Problem 6

A similar ODE solver to the one developed in problem 5 was created, but it uses a first order, one sided finite difference approximation of $u'(0)$:

$$u'(x_0)h = u(x_1) - u(x_0)$$

Its Python impletnation can be found in the file `homework_functions.py`. The function is called `problem_6_solver()`.

For various values of N, the errors E and e were calculated for both the first order and second order one sided approximation methods.

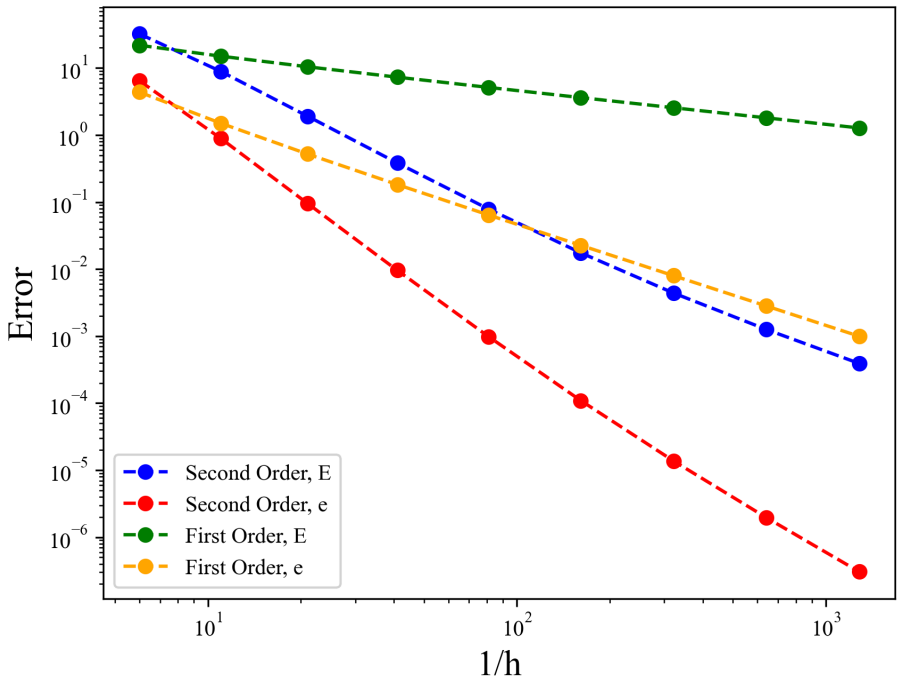

Figure 4: Errors versus 1/h For First Order and Second Order Methods

For each method and error type, a power law relation between the Error and h was first, $E = Ch^{\alpha}$.

Table 3: Reported α Values

Method Order, Error	α
First Order, E	0.5242
First Order, e	1.5515
Second Order, E	2.1508
Second Order, e	3.1781

For both types of error E and e, the second order approximation always has a higher error order compared to the first order approximation. Both methods use the same second order centered difference approximation solve for interior points, the only difference is the order of the approximation at the boundary. This shows the importance of using consistent error orders when combining different approximations, as the lower error order of at boundary propagted throughout the calculation for all other points.