IDNM 680: HOMEWORK 1

JAMES DELLA-GIUSTINA

Task 1

A finite difference is a mathematical expression of the form $f(x-b)-f(x-a)$. If a finite difference is divided by $b-a$, one gets a **difference quotient**. The approximation of derivatives by finite differences plays a central role in finite difference methods for the numerical solution of differential equations, especially boundary value problems. Three basic types are commonly used: **forward**, **backward**, and **central** finite differences. In this task, derive the expressions for the forward, backward, and central finite difference methods.

For the sake of notation, let $f_i = f(x_i)$. The following are Taylor expansions centered around x_i at $x = x_i, x_{i+1}, x_{i+2}, x_{i-1}, x_{i-2}$ are:

$$
\begin{aligned}
f_i &= f_i \\
f_{i+1} &= f_i + f_i'h + \frac{f_i''h^2}{2} + \frac{f_i'''h^3}{6} + \frac{f_i^{(4)}h^4}{24} + O(f^{(5)}) \\
f_{i+2} &= f_i + 2f_i'h + \frac{f_i''4h^2}{2} + \frac{f_i'''8h^3}{6} + \frac{f_i^{(4)}16h^4}{24} + O(f^{(5)}) \\
f_{i-1} &= f_i - f_i'h + \frac{f_i''h^2}{2} - \frac{f_i'''h^3}{6} + \frac{f_i^{(4)}h^4}{24} + O(f^{(5)}) \\
f_{i-2} &= f_i - 2f_i'h + \frac{f_i''4h^2}{2} - \frac{f_i'''8h^3}{6} + \frac{f_i^{(4)}16h^4}{24} + O(f^{(5)})
\end{aligned}
$$

A key observation is that each $f_{i\pm j}$, for $j \in \mathbb{Z}$, is a linear combination of derivatives of f_i with real coefficients. Therefore, when considering the approximations of multiple function values, we obtain a system of equations that may be represented by $Ax = b$. Note that for approximating k different function values, we must truncate our approximations at the derivative of $k-1$ for the coefficient matrix A to be invertible. With these properties in mind, we can therefore multiply on the left by A^{-1} to obtain an approximation of the jth derivative as a linear combination of $f_{i\pm j}$ with rational coefficients.

For a forward difference formula of the first derivative, consider the Taylor expansions of f_i, f_{i+1} and f_{i+2}, truncated at the second derivative, and represented by the equation $Ax = b$.

$$
\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} f_i \\ f_i'h \end{bmatrix} = \begin{bmatrix} f_i \\ f_{i+1} \end{bmatrix}
$$

Finding the inverse coefficient matrix and multiplying on the left results in:

$$
\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} f_i \\ f_{i+1} \end{bmatrix} = \begin{bmatrix} f_i \\ f_i'h \end{bmatrix}
$$

$$
\begin{aligned}
\therefore f_i' &\approx \frac{f_{i+1} - f_i}{h} \\
f_{i+1} &\approx f_i'h + f_i
\end{aligned}
$$

Therefore, the forward difference formula for the first derivative is $f_i' \approx \frac{f_{i+1}-f_i}{h}$. Similarly, for a backward finite difference formula, use the Taylor expansions of f_i and f_{i-1}.

Date: Spring 2023.

$$\begin{bmatrix} 1 & 0 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} f_i \\ f_i'h \end{bmatrix} = \begin{bmatrix} f_i \\ f_{i-1} \end{bmatrix}$$

Then:

$$\begin{bmatrix} 1 & 0 \\ 1 & -1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} f_i \\ f_i'h \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & -1 \end{bmatrix}^{-1} \begin{bmatrix} f_i \\ f_{i-1} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} f_i \\ f_{i-1} \end{bmatrix} = \begin{bmatrix} f_i \\ f_i'h \end{bmatrix}$$

$$\therefore f_i' \approx \frac{f_i - f_{i-1}}{h}$$
$$f_{i-1} \approx f_i - f_i'h$$

For a central finite difference formula, set up a coefficient matrix A for $x = f_i, f_{i+1}, f_{i-1}$. The reason for approximating up to the second derivative will be apparent for Task 2.

$$\begin{bmatrix} 1 & 1 & 1/2 \\ 1 & 0 & 0 \\ 1 & -1 & 1/2 \end{bmatrix} \begin{bmatrix} f_i \\ hf_i' \\ 2h^2 f'' \end{bmatrix} = \begin{bmatrix} f_{i+1} \\ f_i \\ f_{i-1} \end{bmatrix}$$

Then:

$$\begin{bmatrix} 1 & 1 & 1/2 \\ 1 & 0 & 0 \\ 1 & -1 & 1/2 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1/2 \\ 1 & 0 & 0 \\ 1 & -1 & 1/2 \end{bmatrix} \begin{bmatrix} f_i \\ hf_i' \\ 2h^2 f'' \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1/2 \\ 1 & 0 & 0 \\ 1 & -1 & 1/2 \end{bmatrix}^{-1} \begin{bmatrix} f_{i+1} \\ f_{i-1} \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 0 \\ 1/2 & 0 & -1/2 \\ 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} f_{i+1} \\ f_i \\ f_{i-1} \end{bmatrix} = \begin{bmatrix} f_i \\ hf_i' \\ 2h^2 f'' \end{bmatrix}$$

$$\therefore f_i' \approx \frac{f_{i+1} - f_{i-1}}{2h}$$
$$f_{i+1} \approx f_{i-1} + 2hf_i'$$

Task 2

Simple harmonic oscillator (SHO) is a model system that is frequently encountered in physics and engineering. The system can be described using a second-order differential equation with the form:

$$x''(t) = -\omega^2 x, \tag{1}$$

here ω is the angular frequency of the oscillation. Note that here x is a function of t, different from the above (where x is the independent variable). In this task, use the central finite difference method to discretize the second-order differential equation and find the recursive relationship.

In the above derivation of the central difference method, we found that:

$$f_i'' \approx \frac{f_{i+1} + f_{i-1} - 2f_i}{h^2}$$
$$\text{Since } f_i'' \approx -\omega_0^2 f_i$$
$$f_i(2-h^2) \approx f_{i+1} + f_{i-1}$$
$$f_i \approx \frac{f_{i+1} + f_{i-1}}{2-h^2}$$

But since we are going to program this in a loop, we cannot recursively calculate values for $i+1$ for the value of i. Therefore:

$$f_{i+1} \approx f_i(2-h^2) - f_{i-1}$$

The analytic solution of Equation (1) is $x(t) = c_1\cos(\omega_0 t) + c_2\sin(\omega_0 t)$ for $\omega_0 = \sqrt{\frac{k}{m}}$ by considering the auxiliary equation $m^2 + 1 = 0$. Recovering constant coefficients c_1 and c_2 from the boundary conditions.

For Case (1):

$$x(0) = c_1 \implies c_1 = 0$$

and since $x'(t) = c_2\omega_0\cos(\omega_0 t)$, then by the boundary condition $x'(0) = 1$ we get $c_2 = 1$. Therefore:

$$x(t) = \sin(\omega_0 t) \quad \text{and} \quad x'(t) = \cos(\omega_0 t)$$

For Case(2):

$$x(0) = c_1 \implies c_1 = 1$$

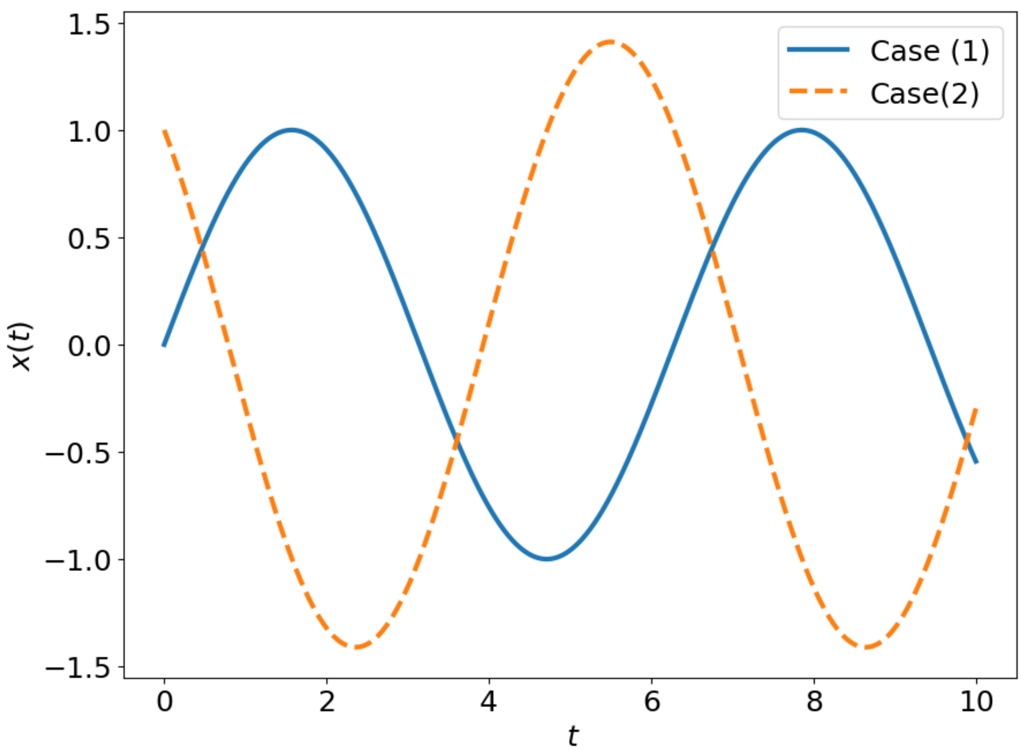
Then $x(t) = \cos(\omega_0 t) + c_2\sin(\omega_0 t)$ and since $x'(0) = -1$, then $c_2 = -1$ and therefore $x(t) = \cos(\omega_0 t) - \sin(\omega_0 t)$.

With the derived formula, write a python code to solve it with initial conditions (1) $x(0) = 0$, $x'(0) = 1$; (2) $x(0) = 1$,$x'(0) = -1$ and assuming $\omega = 1$.

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
params = {'font.size' : 18,
          'figure.figsize':(10, 7.5),
          'lines.linewidth': 3.,
          'lines.markersize':18,}
matplotlib.rcParams.update(params)

N = 1001
t_min, t_max = 0, 10
t = np.linspace(t_min,t_max,N)
h=(t_max-t_min)/(N-1)
x=np.zeros(N)
y=np.zeros(N)
mesh = t[::]
analytic1 = np.sin(mesh) # The following are analytic solutions to verify our
    approximations
analytic2 = np.cos(mesh) - np.sin(mesh)
x[0]=0 # initial conditions
y[0]=1
for i in range(0,len(t)-1):
    if i == 0: # We need to use the finite forward difference of the first derivative
         for the initial value
        x[i+1] = h
```

```
        y[i+1] = -h + 1
    else:
        x[i+1] = x[i]*(2-h**2) - x[i-1] # Central finite difference
        y[i+1] = y[i]*(2-h**2) - y[i-1]
fig = plt.figure()
plt.plot(t,x,'-', label = 'Case (1)')
#plt.plot(t,analytic1, label = 'Analytic Case (1)')
plt.plot(t,y,'--', label = 'Case(2)')
#plt.plot(t,analytic2, label = 'Analytic Case (2)')
plt.xlabel("$t$")
plt.ylabel("$x(t)$")
plt.legend()
```




```
error1 = max(abs(analytic1-x))
error2 = max(abs(analytic2-y))
print("The error for Case (1) is: " ,error1, "and the error for Case (2) is: " ,
    error2)
```

```
The error for Case (1) is:  4.257721103412271e-05 and the error for Case (2) is:
↪  0.004981013785120503
```

Task 3

Using Taylor expansions to derive an expression for discretizing the 3rd and 4th order derivatives.

First, set up a matrix of coefficients for $f_i, f_{i\pm1} \;\; f_{i\pm2}$.

$$\begin{bmatrix} 1 & -2 & 2 & -\frac{4}{3} & \frac{2}{3} \\ 1 & -1 & \frac{1}{2} & -\frac{1}{6} & \frac{1}{24} \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & \frac{1}{2} & \frac{1}{6} & \frac{1}{24} \\ 1 & 2 & 2 & \frac{4}{3} & \frac{2}{3} \end{bmatrix} \begin{bmatrix} f_i \\ f_i' \; h \\ f_i'' \; h^2 \\ f_i''' \; h^3 \\ f_i^{(4)} \; h^4 \end{bmatrix} = \begin{bmatrix} f_{i-2} \\ f_{i-1} \\ f_i \\ f_{i+1} \\ f_{i+2} \end{bmatrix}$$

Finding the inverse of the coefficient matrix and multiplying on the left, we obtain:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ \frac{1}{12} & \frac{-2}{3} & 0 & \frac{2}{3} & \frac{-1}{12} \\ \frac{-1}{12} & \frac{4}{3} & \frac{-5}{2} & \frac{4}{3} & \frac{-1}{12} \\ \frac{-1}{2} & 1 & 0 & -1 & \frac{1}{2} \\ 1 & -4 & 6 & -4 & 1 \end{bmatrix} \begin{bmatrix} f_{i-2} \\ f_{i-1} \\ f_i \\ f_{i+1} \\ f_{i+2} \end{bmatrix} = \begin{bmatrix} f_i \\ f_i' \; h \\ f_i'' \; h^2 \\ f_i''' \; h^3 \\ f_i^{(4)} \; h^4 \end{bmatrix}$$

Therefore $f_i''' = \dfrac{f_{i-1} + f_{i+1} - 1/2 f_{i-2} + 1/2 f_{i+2}}{h^3}$ and $f_i^{(4)} = \dfrac{f_{i-1} + 4f_{i-1} + 6f_i - 4f_{i+1} + f_{i+1}}{h^4}$.

Task 4

Using the above knowledge, write a python code to numerically solve $x'' = -kx^3$. Assume $k = 1$. Illustrate how the discretization spacing h affects your result and how the finite difference schemes (foward, backward, and central) could possibly affect the accuracy.

We choose the boundary conditions $x(0) = 1$ and $x'(0) = -1 \implies f_0 = 1$ and $f_0' = -1$. From Task 3, we know that for a central finite difference:

$$f_i'' \approx \frac{f_{i+1} - 2f_i + 2f_{i-1}}{h^2}$$

Since $f'' = -f_i^3$, then substituting:

$$-f_i^3 h^2 \approx f_{i+1} - 2f_i + f_{i-1} \tag{2}$$

$$f_{i+1} = f_i(2 - f_i^2 h^2) + f_{i-1} \tag{3}$$

```
    # Centered Finite Difference
N = 1001
t_min, t_max = 0, 10
t = np.linspace(t_min,t_max,N)
h = (t_max-t_min)/(N-1)
h2 = 1.1*(t_max-t_min)/(N-1)
h3 = .9*(t_max-t_min)/(N-1)
y0 = np.zeros(N)
y1 = np.zeros(N)
y2 = np.zeros(N)
y3 = np.zeros(N)
y0[0] = 1
y1[0] = 1
y2[0] = 1
```

```
y3[0] = 1
for i in range(0,len(t)-1):
    if i == 0: # We need to use the finite forward difference of the first derivative
         for the initial value
        y0[i+1] = 1-h_min
        y1[i+1] = 1-h
        y2[i+1] = 1-h2
        y3[i+1] = 1-h3
    else:
        y1[i+1] = y1[i]*(2-y1[i]**2*h**2)-y1[i-1]
        y2[i+1] = y2[i]*(2-y2[i]**2*h2**2)-y2[i-1]
        y3[i+1] = y3[i]*(2-y3[i]**2*h3**2)-y3[i-1]
fig = plt.figure()
plt.plot(t,y1,'--', label = '$h$')
plt.plot(t,y2,'-', label = 'Smaller $h$')
plt.plot(t,y3,'--', label = 'Larger $h$')
plt.xlabel(" $t$ ")
plt.ylabel("$x(t)$")
plt.legend()
```

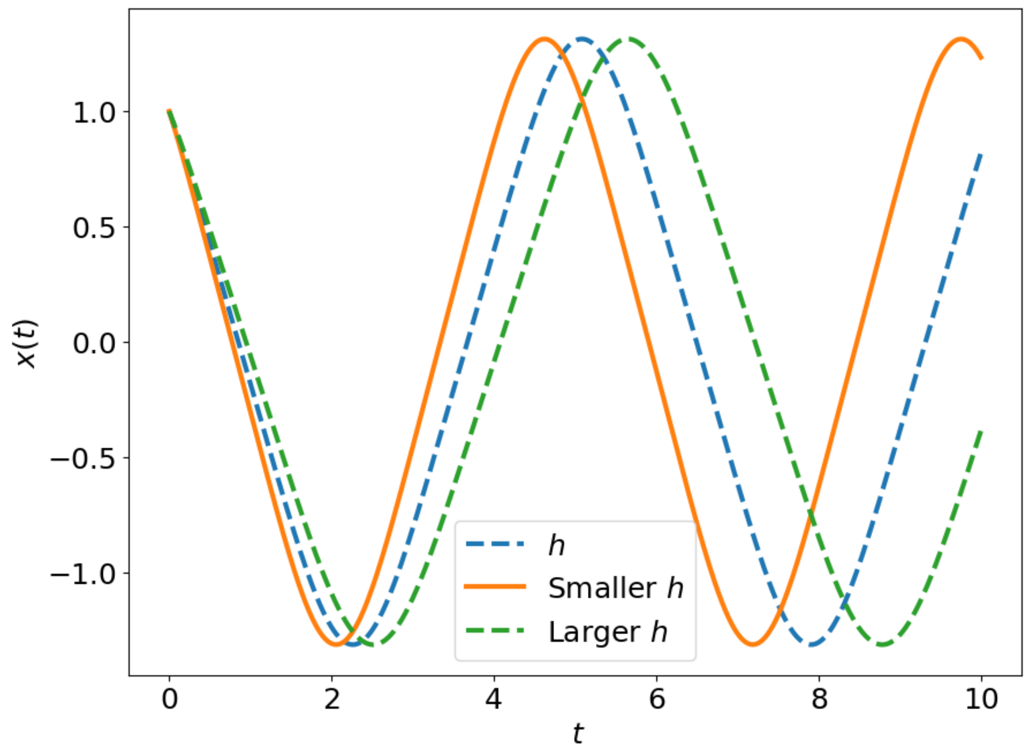


FIGURE 1. Central Difference Method

Now, we wish to obtain formulas for the second derivative approximations of both the forward and backward differences. For the forward difference, use f_i, f_{i+1}, and f_{i+2} to obtain the following:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1/2 \\ 1 & 2 & 2 \end{bmatrix} \begin{bmatrix} f_i \\ f_i' h \\ f_i'' h^2 \end{bmatrix} = \begin{bmatrix} f_i \\ f_{i+1} \\ f_{i+2} \end{bmatrix}$$

Then:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1/2 \\ 1 & 2 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1/2 \\ 1 & 2 & 2 \end{bmatrix} \begin{bmatrix} f_i \\ f_i'h \\ f_i''h^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1/2 \\ 1 & 2 & 2 \end{bmatrix}^{-1} \begin{bmatrix} f_i \\ f_{i+1} \\ f_{i+2} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ -3/2 & 2 & -1/2 \\ 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} f_i \\ f_{i+1} \\ f_{i+2} \end{bmatrix} = \begin{bmatrix} f_i \\ f_i'h \\ f_i''h^2 \end{bmatrix}$$

$$\therefore f_i'' \approx \frac{f_i - 2f_{i+1} + f_{i+2}}{h^2}$$

$$f_{i+2} \approx f_i''h^2 + 2f_{i+1} - f_i$$

Since $f_i'' = -f_i^3 \implies f_{i+2} \approx -f_i(h^2 f_i^2 + 1) + 2f_{i+1}$.

```
# Forward finite difference, notice that in the derived formula we approximate f_{i
   +2},
# but in our code we shift each index by -1
N = 1001
t_min, t_max = 0, 10
t = np.linspace(t_min,t_max,N)
h = (t_max-t_min)/(N-1)
h2 = 1.1*(t_max-t_min)/(N-1)
h3 = .9*(t_max-t_min)/(N-1)
mesh = t[::]
x1 = np.zeros(N)
x2 = np.zeros(N)
x3 = np.zeros(N)
xfound = np.zeros(N)
xfounddt=np.zeros(N)
x1[0]=1
x2[0]=1
x3[0]=1
for i in range(0,len(t)-1):
    if i == 0: # # We need to use the finite forward difference of the first
       derivative
                          #for the initial value 2 values
        x1[i+1] = 1-h
        x2[i+1] = 1-h2
        x3[i+1] = 1-h3
    else:
        x1[i+1] = -x1[i-1]**3 * h**2 - x1[i-1] + 2*x1[i]
        x2[i+1] = -x2[i-1]**3 * h2**2 - x2[i-1] + 2*x2[i]
        x3[i+1] = -x3[i-1]**3 * h3**2 - x3[i-1] + 2*x3[i]
fig = plt.figure()
plt.plot(t,x1,'--', label = '$h$')
plt.plot(t,x2,'-', label = 'Smaller $h$')
plt.plot(t,x3,'--', label = 'Larger $h$')
plt.xlabel(" $t$ ")
plt.ylabel("$x(t)$")
```

```
plt.legend()
```

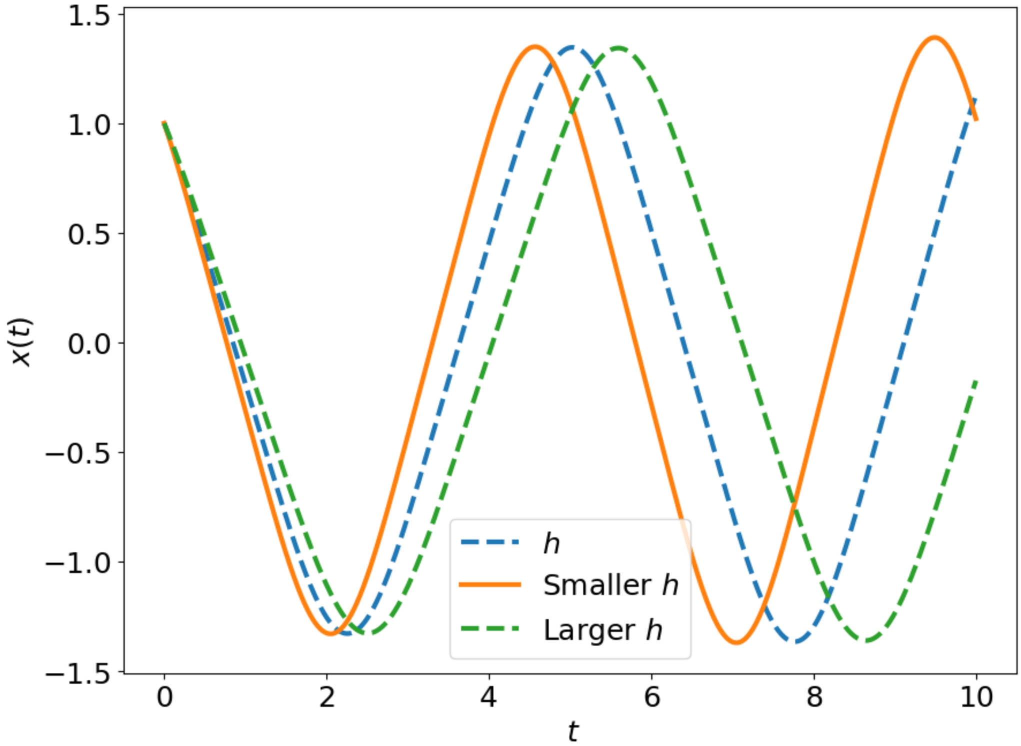


FIGURE 2. Forward Difference Method

For the backward finite difference, instead use f_i, f_{i-1} and f_{i-2}:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 1/2 \\ 1 & -2 & 2 \end{bmatrix} \begin{bmatrix} f_i \\ f_i'h \\ f_i''h^2 \end{bmatrix} = \begin{bmatrix} f_i \\ f_{i-1} \\ f_{i-2} \end{bmatrix}$$

Then:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 1/2 \\ 1 & -2 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 1/2 \\ 1 & -2 & 2 \end{bmatrix} \begin{bmatrix} f_i \\ f_i'h \\ f_i''h^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 1/2 \\ 1 & -2 & 2 \end{bmatrix}^{-1} \begin{bmatrix} f_i \\ f_{i-1} \\ f_{i-2} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 3/2 & -2 & 1/2 \\ 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} f_i \\ f_{i-1} \\ f_{i-2} \end{bmatrix} = \begin{bmatrix} f_i \\ f_i'h \\ f_i''h^2 \end{bmatrix}$$

$$\therefore f_i'' \approx \frac{f_i - 2f_{i-1} + f_{i-2}}{h^2}$$

$$f_{i-2} \approx f_i''h^2 + 2f_{i-1} - f_i$$

Since $f'' = -f_i^3$, then $f_{i-2} \approx 2f_{i-1} - f_i(1 + h^2 f_i^2)$.

I was unable to implement this method correctly, since f_i had a cubic term. Therefore, using functions defined by chatGPT, I implemented the following code:

```python
N = 1001
t_min, t_max = 0, 10
h = (t_max-t_min)/(N-1)
h2 = 1.1*(t_max-t_min)/(N-1)
h3 = .9*(t_max-t_min)/(N-1)
t = np.linspace(t_min, t_max, N)
z1 = np.zeros(N)
z1_prime = np.zeros(N)
z2 = np.zeros(N)
z2_prime = np.zeros(N)
z3 = np.zeros(N)
z3_prime = np.zeros(N)
z1[0] = 1
z1_prime[0] = -1
z2[0] = 1
z2_prime[0] = -1
z3[0] = 1
z3_prime[0] = -1

for i in range(1, N):
    z1_prime[i] = z1_prime[i - 1] - h*z1[i - 1]**3
    z1[i] = z1[i - 1] + h*z1_prime[i]
    z2_prime[i] = z2_prime[i - 1] - h*z2[i - 1]**3
    z2[i] = z2[i - 1] + h*z2_prime[i]
    z3_prime[i] = z3_prime[i - 1] - h*z3[i - 1]**3
    z3[i] = z3[i - 1] + h*z3_prime[i]

plt.plot(t,z1,'--', label = '$h$')
plt.plot(t,z2,'-', label = 'Smaller $h$')
plt.plot(t,z3,'--', label = 'Larger $h$')
plt.xlabel('$t$')
plt.ylabel('$x(t)$')

plt.legend()
plt.show()
```

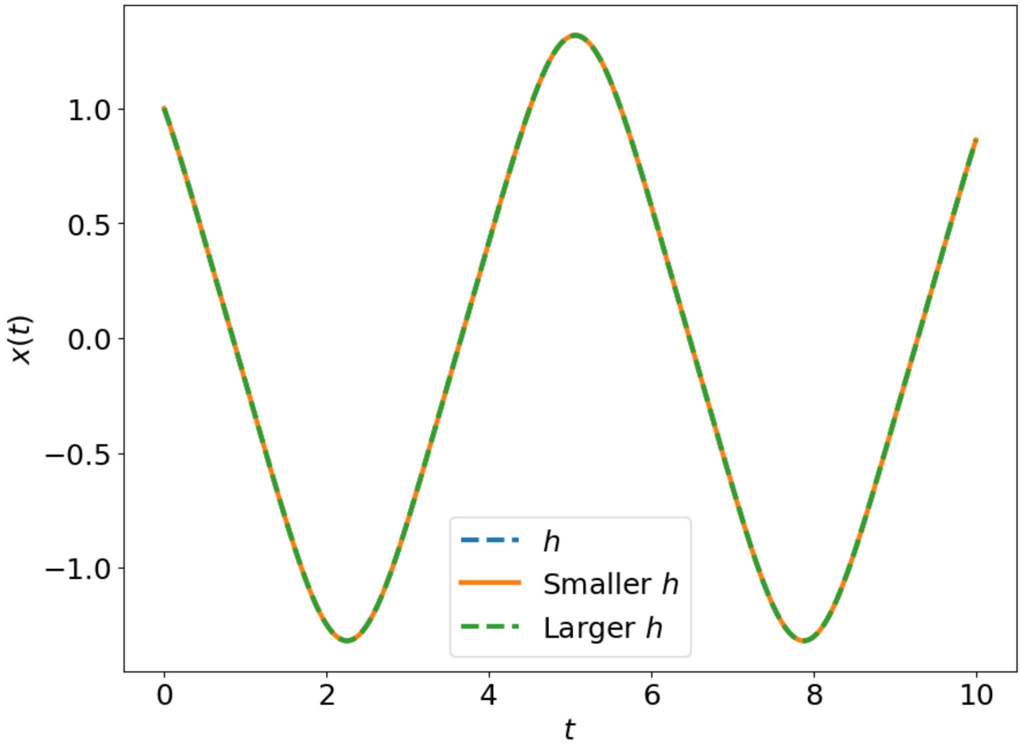


Figure 3. Backward Difference Method

We note here that different values of h all agree on the initial function values, but that the period of $x(t)$ increases as t grows larger. However, the backward finite difference method provided by chatGPT shows convergence for any choice of small h. This leads me to believe that some part of the implementation is wrong.