Chandler Justice - A02313187

MATH5620: Homework #2
Due: February 2, 2024 @ 23:59

I have solved the criteria for this assignment with the following code:

```
import math
from typing import Callable
from scipy.sparse import diags
import numpy as np



class FiniteDifference:
    def __init__(self, left_endpoint: float, right_endpoint: float, num_points: int, fx: Callable) -> None:
        # initialize data

        self.function_values = []
        self.interval_points = []
        self.h = 0
        self.approx_solutions = []
        self.matrix = np.array


        self.left_endpoint = left_endpoint
        self.interval_points.append(left_endpoint)
        self.right_endpoint = right_endpoint
        self.interval_points.append(right_endpoint)

        self.num_points = num_points

        self.mesh_interval = self.determine_mesh_interval()
        self.create_interval_points()

        self.fx = fx

        self.h = 1 / (1 + self.mesh_interval)

    def determine_mesh_interval(self):
        return (self.right_endpoint - self.left_endpoint) / (self.num_points - 1)

    def create_interval_points(self):

        curr_val = self.left_endpoint + self.mesh_interval
        for i in range(1, self.num_points - 1):
            self.interval_points.insert(i, curr_val)
            curr_val += self.mesh_interval

    def construct_matrix(self):
        k = [np.ones(self.num_points-1),-2*np.ones(self.num_points),np.ones(self.num_points-1)]
        offset = [-1,0,1]
        self.matrix = diags(k,offset).toarray()

    def evaluate_function(self, x):
```

```
        return self.fx(x)

    def create_approximate_solutions(self):
        self.function_values.append(self.left_endpoint)

        # use 2nd order finite difference approx
        for i in range(1,self.num_points - 1):
            second_difference = (1 / self.h**2) * (self.fx(self.interval_points[i - 1])
\ - (2 * self.fx(self.interval_points[i])) + self.fx(self.interval_points[i + 1]))
            self.function_values.append(second_difference)
        self.function_values.append(self.right_endpoint)

    def solve_linear_system(self):
        func_vals = np.array(self.function_values).transpose()
        self.approx_solutions = np.linalg.solve(self.matrix, func_vals)


    def solve(self):
        self.construct_matrix()
        self.create_approximate_solutions()
        self.solve_linear_system()


def test_finite_solver():
    mesh_widths = [4,8,16]
    for mesh in mesh_widths:
        solver = FiniteDifference(0, 1, mesh, lambda a: math.sin(a))
        solver.solve()
        print(f'approx solutions for mesh size {mesh}:')
        print(solver.approx_solutions)


def main():
    test_finite_solver()

main()
```

Running this program I get the following output

```
approx solutions for mesh size 4:
[-0.11317098 -0.22634196 -0.40354777 -0.70177388]
approx solutions for mesh size 8:
[-0.08034362 -0.16068724 -0.24481941 -0.33645149 -0.43914205 -0.55622434
 -0.69073843 -0.84536921]
approx solutions for mesh size 16:
[-0.04633856 -0.09267712 -0.13935242 -0.18669972 -0.23505128 -0.28473489
 -0.33607244 -0.38937848 -0.44495878 -0.50310903 -0.56411351 -0.62824379
 -0.69575758 -0.76689755 -0.84189025 -0.92094513]
```