Jackson Lyons
MTH 452 Numerical Analysis II
April 26, 2024

Final Project

Problem:

Solve the boundary value problem using Finite Difference method and Finite Element method with piecewise linear function, respectively. Compare the efficiency of the schemes.

$$-\frac{d}{dx}(xy') + 4y = 4x^2 - 8x + 1, \quad 0 \le x \le 1, \tag{1}$$

with boundary condition $y(0) = y(1) = 0$. The exact solution is $y = x^2 - x$.

Solution:

Finite Difference method:

First we have to rearrange the equation (1) to the form $y'' + p(x)y' + q(x)y = r(x)$

Rearranging (1) gives us $y'' + (\frac{1}{x})y' + (\frac{-4}{x})y = -4x + 8 - \frac{1}{x}$

So we have that $p(x) = \frac{1}{x}$, $q(x) = \frac{-4}{x}$, and $r(x) = -4x - \frac{1}{x} + 8$

Next, we setup the system $AU = b$ where A is a tridiagonal matrix with

subdiagonal: $v_i = 1 - \frac{h}{2}p(x_i)$
diagonal: $d_i = -2 + h^2 q(x_i)$
superdiagonal: $1 + \frac{h}{2}p(x_i)$

and $b = [u_a, h^2 r(x_i), ..., h^2 r(x_{N-1}), u_b]$

Python implementation:

```
# Define problem
N=10
a, b = 0, 1
u0, u1 = 0, 0
h = (b-a) / (N)
x = np.linspace(a, b, N+1)
```

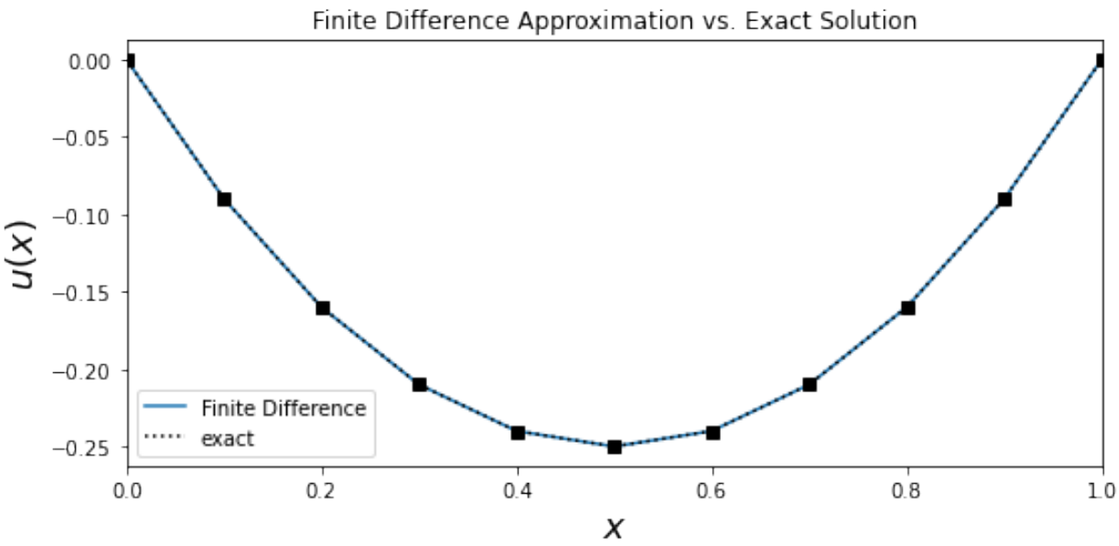
```
# Get A
A = np.zeros((N+1, N+1))
A[0, 0] = 1
A[N, N] = 1
for i in range(1, N):
    A[i, i-1] = 1-0.5*h*p(x[i]) # Subdiagonal
    A[i, i] = -2+h**2*q(x[i]) # Diagonal
    A[i, i+1] = 1+0.5*h*p(x[i]) # Superdiagonal

# Get b
b = np.zeros(N+1)
b[0] = u0
b[N-1] = u1
for i in range(1, N):
    b[i] = h**2*r(x[i])
```

Solve the system $AU = b$

```
U = la.solve(A, b)
```

Plotting the finite difference approximation vs the exact solution:



Finite Element method:

For the finite element method I imported a preexisting Python function [1]. This function applies the finite element method with a piecewise linear function to a boundary value problem. First, we need to determine the parameters to pass into the function. Our boundary value problem is of the form:

$$-\frac{d}{dx}(a(x)\frac{dy}{dx}) + c(x) * y(x) = f(x)$$

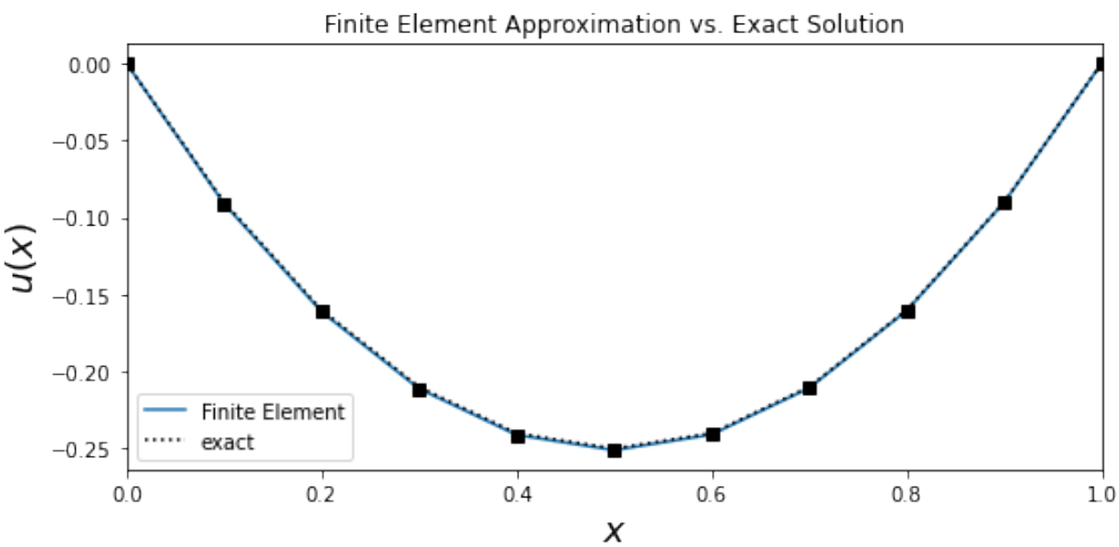
So from (1) we get that $a(x) = x$, $c(x) = 4$, and $f(x) = 4x^2 - 8x + 1$.

Python implementation

```
# define functions for a(x), c(x), f(x) for problem
def a00(x):
    return x
def c00(x):
    return 4
def f00(x):
    return 4*x**2 - 8*x +1
```

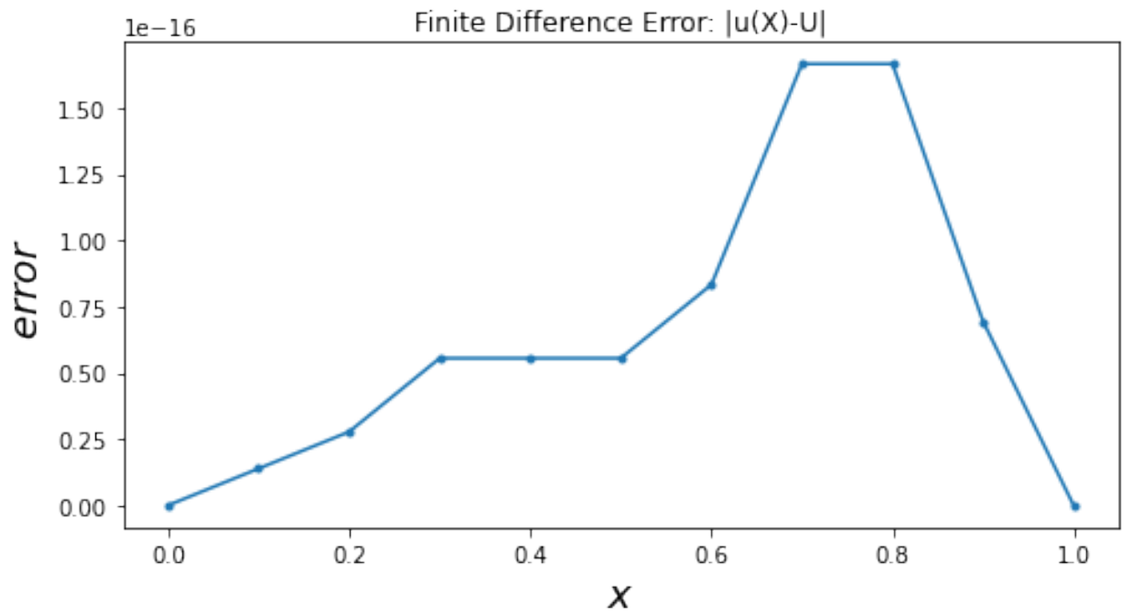
```
from fem1d_bvp_linear import fem1d_bvp_linear
n=11 # requires a different n from finite-difference method implementation
    (n+1)
U2 = fem1d_bvp_linear(n, a00, c00, f00, x)
```

Plotting the finite element approximation vs the exact solution:

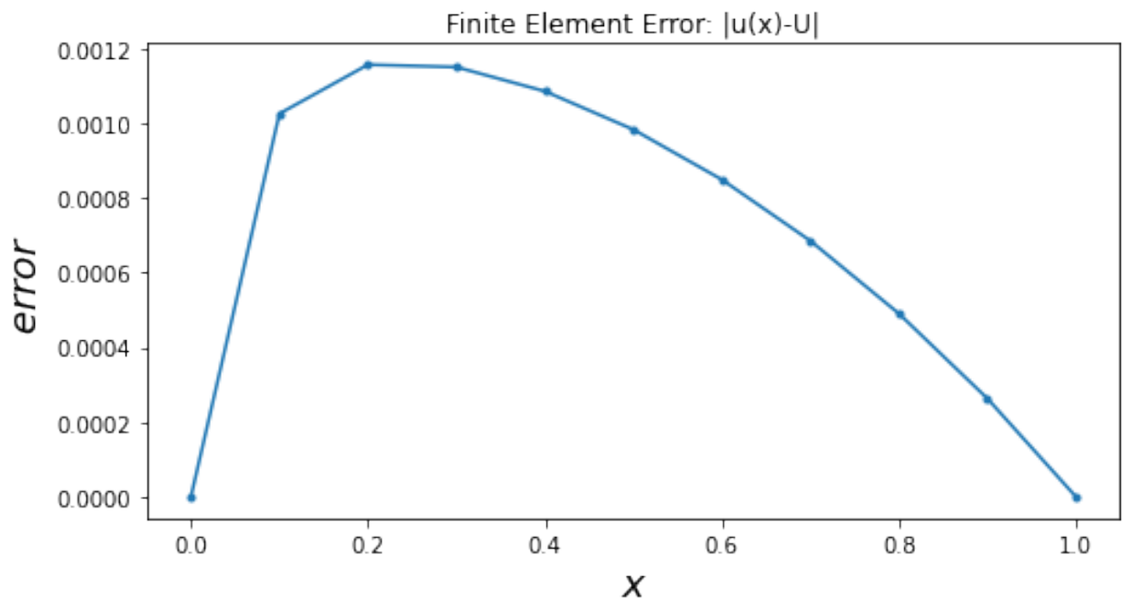


Comparison of FDM vs FEM:

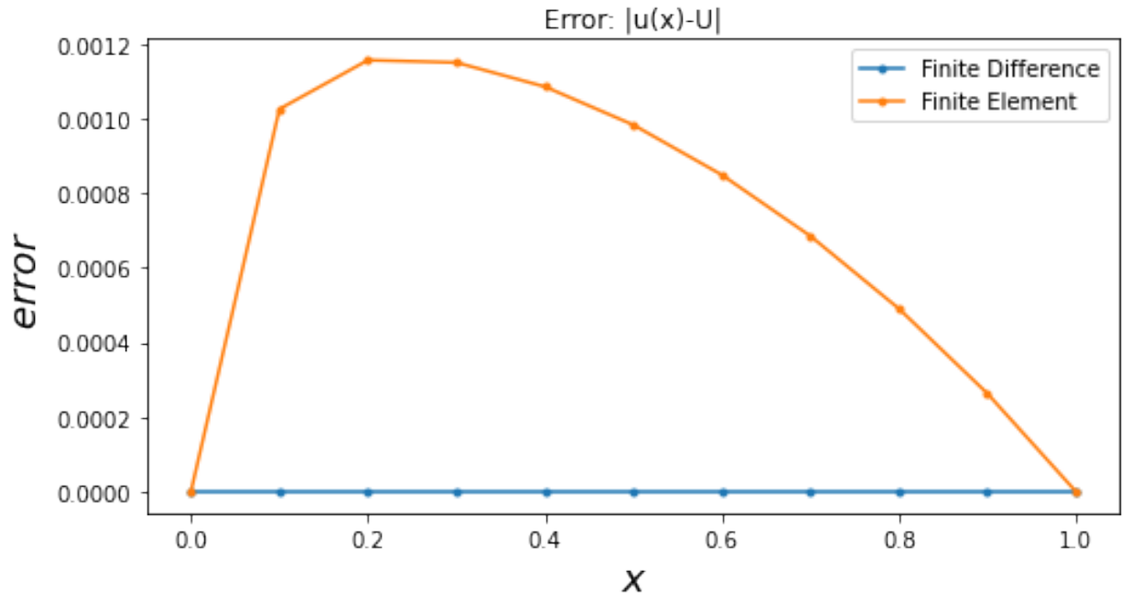
Plotting the error for the finite difference method vs the exact solution:



Plotting the error for the finite element method vs the exact solution:



Comparing the error of the two methods:



The methods have the following max errors:

Max error for FDM = 1.665e-16
Max error for FEM = 1.157e-03

Since the error for the finite difference method is substantially smaller than the error for the finite element method, we can conclude that the finite difference approximation is far more accurate for this specific problem.

References:

[1] Burkardt, J (2015) fem1d_bvp_linear [Source code]. https://people.math.sc.edu/Burkardt/py_src/fem1d_bvp_linear/fem1d_bvp_linear.html