Rough for Block Tri diagonal system

February 22, 2021

[2]:
```
import numpy as np
```

[23]:
```
X = []
for i in range(4):
    X.append([[4,5],[2,3]])
```

[24]:
```
print(X[0])
```

```
[[4, 5], [2, 3]]
```

[42]:
```
a = np.array([ [[1,2],[3,4]], [[6,6],[7,8]] ])
```

[43]:
```
a
```

[43]:
```
array([[[1, 2],
        [3, 4]],

       [[6, 6],
        [7, 8]]])
```

[45]:
```
a = np.append(a, [[[5,6],[7,8]]], axis = 0)
```

[46]:
```
a
```

[46]:
```
array([[[1, 2],
        [3, 4]],

       [[6, 6],
        [7, 8]],

       [[5, 6],
        [7, 8]]])
```

[55]:
```
a = np.array([ [[1,2],[3,4]], [[6,6],[7,8]] ])
for i in range(3):
    a = np.append(a, [[[5+ i,6- i],[7+ i,8- i]]], axis = 0)
```

[56]:
```
a
```

[56]:
```
array([[[1, 2],
        [3, 4]],

       [[6, 6],
        [7, 8]],

       [[5, 6],
        [7, 8]],

       [[6, 5],
        [8, 7]],

       [[7, 4],
        [9, 6]]])
```

[61]:
```
def A(x):
    return 4
def B(x):
    return 1
def C(x):
    return -6
def D(x):
    return 1
```

[63]:
```
h = 0.2
x_0 = 0
x_n = 1
n = int((x_n- x_0)/h  + 0.5 )
```

[79]:
```
x = np.array([])
```

[80]:
```
for i in range(n+1):
    x = np.append(x, i*h)
```

[81]:
```
x
```

[81]:
```
array([0. , 0.2, 0.4, 0.6, 0.8, 1. ])
```

[76]:
```

```

[76]:
```
array([0. , 0.2, 0.4, 0.6, 0.8])
```

[ ]:
```

```

[87]:
```
A_star = np.array([ [[-1,-h/2],[0,((2 - A(x[0])*h)/(2*h*h) )]] ])
B_star = np.array([ [[ 1,-h/2],[C(x[0]),B(x[0] - (2)/(h*h))]] ])
C_star = np.array([ [[0 ,0],[0,((2 + A(x[0])*h)/(2*h*h) )]] ])
D_star = np.array([ [[0,D(x[0])]] ])
```

[89]:
```
print("A_star =" )
print(A_star)
print("B_star =" )
print(B_star)
print("C_star =" )
print(C_star)
print("D_star =" )
print(D_star)
```

```
A_star =
[[[-1.  -0.1]
  [ 0.  15. ]]]
B_star =
[[[ 1.  -0.1]
  [-6.   1. ]]]
C_star =
[[[ 0.  0.]
  [ 0. 35.]]]
D_star =
[[[0 1]]]
```

[116]:
```
A_star = np.array([ [[-1,-h/2],[0,((2 - A(x[0])*h)/(2*h*h) )]] ])
B_star = np.array([ [[ 1,-h/2],[C(x[0]),B(x[0]) - ((2)/(h*h))]] ])
C_star = np.array([ [[0 ,0],[0,((2 + A(x[0])*h)/(2*h*h) )]] ])
D_star = np.array([ [[0],[D(x[0])]] ])
print("A_star =" )
print(A_star)
print("B_star =" )
print(B_star)
print("C_star =" )
print(C_star)
print("D_star =" )
print(D_star)
for i in range(1,n+1):
    A_star = np.append(A_star,[ [[-1,-h/2],[0,((2 - A(x[i])*h)/(2*h*h) )]] ],␣
 ↪axis=0)
    B_star = np.append(B_star,[ [[ 1,-h/2],[C(x[0]),B(x[i]) - ((2)/(h*h))]] ],␣
 ↪axis=0)
    C_star = np.append(C_star,[ [[0 ,0],[0,((2 + A(x[i])*h)/(2*h*h) )]] ],␣
 ↪axis=0)
    D_star = np.append(D_star,[ [[0],[D(x[i])]] ], axis=0)
    print(i)
```

```
print("Now, after running the loop")
print("A_star =" )
print(A_star)
print("B_star =" )
print(B_star)
print("C_star =" )
print(C_star)
print("D_star =" )
print(D_star)
```

```
A_star =
[[[-1.  -0.1]
  [ 0.  15. ]]]
B_star =
[[[  1.   -0.1]
  [ -6.  -49. ]]]
C_star =
[[[ 0.  0.]
  [ 0. 35.]]]
D_star =
[[[0]
  [1]]]
1
2
3
4
5
Now, after running the loop
A_star =
[[[-1.  -0.1]
  [ 0.  15. ]]

 [[-1.  -0.1]
  [ 0.  15. ]]

 [[-1.  -0.1]
  [ 0.  15. ]]

 [[-1.  -0.1]
  [ 0.  15. ]]

 [[-1.  -0.1]
  [ 0.  15. ]]

 [[-1.  -0.1]
  [ 0.  15. ]]]
B_star =
```

```
[[[  1.   -0.1]
  [ -6.  -49. ]]

 [[  1.   -0.1]
  [ -6.  -49. ]]

 [[  1.   -0.1]
  [ -6.  -49. ]]

 [[  1.   -0.1]
  [ -6.  -49. ]]

 [[  1.   -0.1]
  [ -6.  -49. ]]

 [[  1.   -0.1]
  [ -6.  -49. ]]]
C_star =
[[[ 0.  0.]
  [ 0. 35.]]

 [[ 0.  0.]
  [ 0. 35.]]

 [[ 0.  0.]
  [ 0. 35.]]

 [[ 0.  0.]
  [ 0. 35.]]

 [[ 0.  0.]
  [ 0. 35.]]

 [[ 0.  0.]
  [ 0. 35.]]]
D_star =
[[[0]
  [1]]

 [[0]
  [1]]

 [[0]
  [1]]

 [[0]
  [1]]
```

```
[[0]
 [1]]

[[0]
 [1]]]
```

[96]:
```
np.transpose(A_star[0])
```

[96]:
```
array([[-1. , -0.1],
       [ 0. , 15. ]])
```

[97]:
```
np.transpose(A_star[0])
```

[97]:
```
array([[-1. ,  0. ],
       [-0.1, 15. ]])
```

[227]:
```
X_ans = np.array([[[0],[0]]],dtype = float)
```

[228]:
```
for i in range(n):
    X_ans = np.append(X_ans, [[[0],[0]]] ,axis = 0)
```

[229]:
```
X_ans
```

[229]:
```
array([[[0.],
        [0.]],

       [[0.],
        [0.]],

       [[0.],
        [0.]],

       [[0.],
        [0.]],

       [[0.],
        [0.]],

       [[0.],
        [0.]]])
```

[230]:
```
X_ans[n] = np.array([[[0],[1]]])
```

[231]:
```
X_ans
```

[231]:
```
array([[[0.],
        [0.]],
```

```
       [[0.],
        [0.]],

       [[0.],
        [0.]],

       [[0.],
        [0.]],

       [[0.],
        [0.]],

       [[0.],
        [1.]]])
```

[ ]:

[114]:
```
#X_ans = np.append(X_ans, [[[2],[3]]],axis = 0)
D_star[1] = D_star[1] - np.dot(A_star[1],X_ans[0])
```

[119]:
```
#X_ans
np.dot(np.linalg.inv(B_star[1]),D_star[1] )
```

[119]:
```
array([[-0.00201613],
       [-0.02016129]])
```

[140]:
```
cc_dash = np.dot(np.linalg.inv(B_star[1]),C_star[1] )
dd_dash = np.dot(np.linalg.inv(B_star[1]),D_star[1] )
```

[141]:
```
C_dash = np.array([cc_dash])
D_dash = np.array([dd_dash])
```

[142]:
```
C_dash = np.append(C_dash, [cc_dash], axis = 0)
D_dash = np.append(D_dash, [dd_dash], axis = 0)
```

[143]:
```
C_dash[1]
```

[143]:
```
array([[ 0.        , -0.07056452],
       [ 0.        , -0.70564516]])
```

[138]:
```
i = 2
b_dash = (B_star[i] - np.dot(A_star[i],C_dash[i-1]) )
b_dashinv = np.linalg.inv(b_dash)
cc_dash = np.dot(b_dashinv,C_star[i])
dd_dash = np.dot(b_dashinv , D_star[i] - np.dot(A_star[i],D_dash[i-1]))
```

[146]:

[146]:
```
array([[-0.01176435],
       [-0.0320662 ]])
```

[147]:
```
for i in range(2,n):
    b_dash = (B_star[i] - np.dot(A_star[i],C_dash[i-1]) )
    b_dashinv = np.linalg.inv(b_dash)
    cc_dash = np.dot(b_dashinv,C_star[i])
    dd_dash = np.dot(b_dashinv , D_star[i] - np.dot(A_star[i],D_dash[i-1]))
    C_dash = np.append(C_dash, [cc_dash], axis = 0)
    D_dash = np.append(D_dash, [dd_dash], axis = 0)
```

[148]:
```
C_dash
```

[148]:
```
array([[[ 0.        , -0.07056452],
        [ 0.        , -0.70564516]],

       [[ 0.        , -0.07056452],
        [ 0.        , -0.70564516]],

       [[ 0.        , -0.21171782],
        [ 0.        , -0.87802707]],

       [[ 0.        , -0.365797  ],
        [ 0.        , -0.91559   ]],

       [[ 0.        , -0.50523998],
        [ 0.        , -0.90649419]]])
```

[162]:
```
D_dash
```

[162]:
```
array([[[-0.00201613],
        [-0.02016129]],

       [[-0.00201613],
        [-0.02016129]],

       [[-0.01176435],
        [-0.0320662 ]],

       [[-0.02951053],
        [-0.03639253]],

       [[-0.05259415],
        [-0.0348868 ]]])
```

[163]:
```
b_dash = (B_star[n-1] - np.dot(A_star[n-1],C_dash[n-2]) )
b_dashinv = np.linalg.inv(b_dash)
```

[189]:
```
final_ans = np.dot(b_dashinv, D_star[n-1] - np.dot(C_star[n-1],X_ans[n]) - np.
 ↪dot(A_star[n-1], D_dash[n-2]))
```

[210]:
```
final_ans[0][0]
```

[210]:
```
0.4526458221053572
```

[223]:
```
final_ans
```

[223]:
```
array([[0.45264582],
       [0.87160739]])
```

[236]:
```

```

[237]:
```
X_ans[n-1] =  np.array(final_ans)
```

[238]:
```
X_ans
```

[238]:
```
array([[[0.        ],
        [0.        ]],

       [[0.        ],
        [0.        ]],

       [[0.        ],
        [0.        ]],

       [[0.        ],
        [0.        ]],

       [[0.45264582],
        [0.87160739]],

       [[0.        ],
        [1.        ]]])
```

[ ]:
```
D_dash[] - np.dot(C_dash[i], X_ans[i+1])
```

[239]:
```
for i in range(n-2, 0,-1):
    #D_dash[i] - np.dot(C_dash[i], X_ans[i+1])
    X_ans[i] = np.array(D_dash[i] - np.dot(C_dash[i], X_ans[i+1]))
    print("X["+str(i) +"] = ")
    print(X_ans[i])
```

```
X[3] =
[[0.28932083]
 [0.76164248]]
X[2] =
[[0.14948893]
 [0.63667652]]
X[1] =
[[0.04291064]
 [0.42910641]]
```

[ ]: