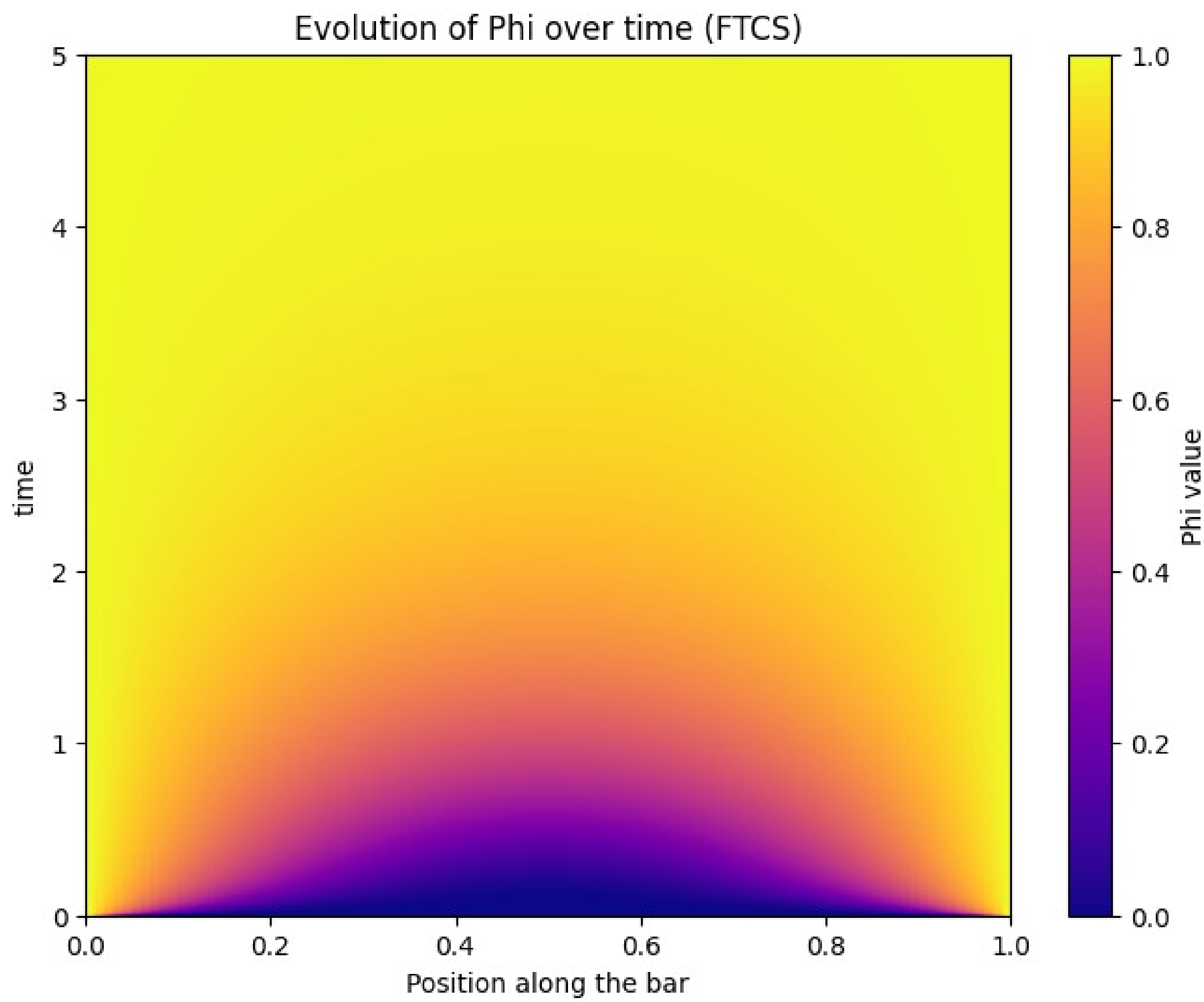
تکلیف سری دوم درس دینامیک سیالات محاسباتی پیشرفته محمد یوسفی 402126084
https://github.com/mohamadusefi

1. using FTCS method:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.sparse import diags


N = 100 # number of nodes
L = 1 # length of bar
alfa = 0.1 # diffusion number
dt = 0.0005 # time step
iteration = 10000
t = iteration * dt # time for computing exact solution

Phi_left = 1 # B.C. in x = 0
Phi_right = 1 # B.C. in x = 1
Phi_initial = 0 # initial condition

Phi = np.empty(N)
Phi.fill(Phi_initial)
Phi[0] = Phi_left
Phi[-1] = Phi_right
h = L / (N - 1)  # step size
x = np.linspace(0, L, N)
v = (alfa*dt)/h**2

# Define matrix A for FTCS method
diagonals = [(1-2*v) * np.ones(N), v*np.ones(N-1), v*np.ones(N-1)]
A = diags(diagonals, [0, -1, 1], format='csr')
A[0, 0] = 1
A[0, 1] = 0
A[N-1, N-2] = 0
A[N-1, N-1] = 1
results = np.zeros((iteration, N))

for i in range(iteration):
    Phi = A.dot(Phi)
    results[i] = Phi
print(Phi)

FTCS_solution = []
FTCS_solution = Phi.copy()

plt.figure(figsize=(8, 6))
plt.imshow(results, aspect='auto', cmap='plasma',
           origin='lower', extent=[0, L, 0, t])
plt.colorbar(label='Phi value')
```

```python
plt.xlabel('Position along the bar')
plt.ylabel('time')
plt.title('Evolution of Phi over time (FTCS)')
plt.show()
```

```
[1.         0.99970973 0.99941974 0.99913035 0.99884182 0.99855447
 0.99826857 0.99798441 0.99770228 0.99742247 0.99714525 0.9968709
 0.99659971 0.99633194 0.99606786 0.99580775 0.99555185 0.99530043
 0.99505375 0.99481205 0.99457556 0.99434455 0.99411922 0.99389982
 0.99368656 0.99347966 0.99327932 0.99308575 0.99289914 0.99271968
 0.99254755 0.99238293 0.99222598 0.99207685 0.9919357  0.99180267
 0.9916779  0.9915615  0.99145361 0.99135431 0.99126373 0.99118194
 0.99110903 0.99104507 0.99099013 0.99094426 0.99090751 0.99087991
 0.9908615  0.99085229 0.99085229 0.9908615  0.99087991 0.99090751
 0.99094426 0.99099013 0.99104507 0.99110903 0.99118194 0.99126373
 0.99135431 0.99145361 0.9915615  0.9916779  0.99180267 0.9919357
 0.99207685 0.99222598 0.99238293 0.99254755 0.99271968 0.99289914
 0.99308575 0.99327932 0.99347966 0.99368656 0.99389982 0.99411922
 0.99434455 0.99457556 0.99481205 0.99505375 0.99530043 0.99555185
 0.99580775 0.99606786 0.99633194 0.99659971 0.9968709  0.99714525
 0.99742247 0.99770228 0.99798441 0.99826857 0.99855447 0.99884182
 0.99913035 0.99941974 0.99970973 1.        ]
```



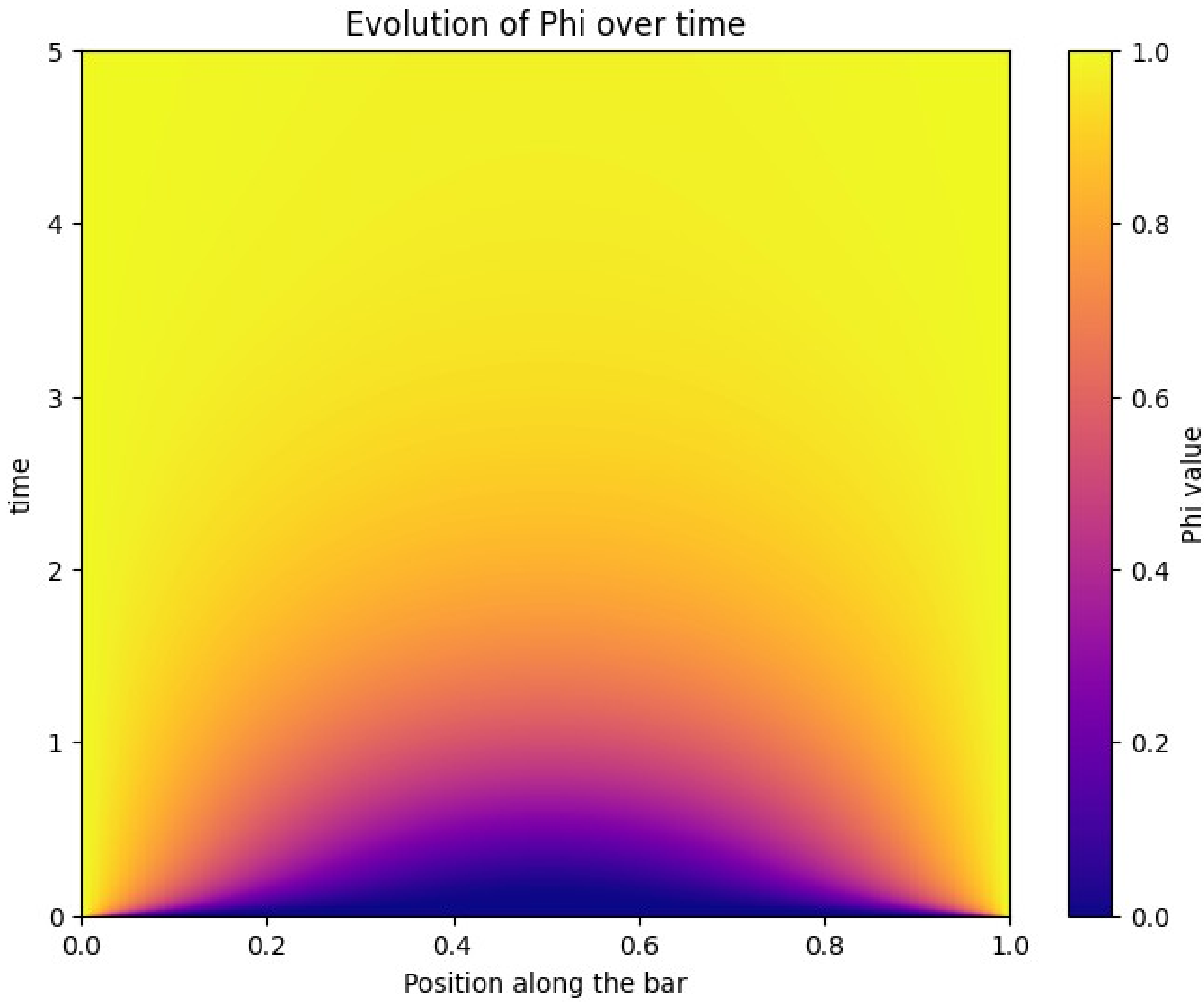
2. using BTCS method:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.sparse import diags
from scipy.sparse.linalg import cg

N = 100 # number of nodes
L = 1 # length of bar
alfa = 0.1 # diffusion number
dt = 0.0005 # time stepp
iteration = 10000
t = iteration * dt # time for computing exact solution
Phi_left = 1 # B.C. in x = 0
Phi_right = 1 # B.C. in x = 1
Phi_initial = 0 # initial condition

Phi = np.empty(N)
Phi.fill(Phi_initial)
```

```
Phi[0] = Phi_left
Phi[-1] = Phi_right
h = L / (N - 1)  # step size
x = np.linspace(0, L, N)
v = (alfa*dt)/h**2

# Define matrix B for BTCS method
diagonals = [(1+2*v) * np.ones(N), -v*np.ones(N-1), -v*np.ones(N-1)]
B = diags(diagonals, [0, -1, 1], format='csr')
B[0, 0] = 1
B[0, 1] = 0
B[N-1, N-2] = 0
B[N-1, N-1] = 1

for i in range(iteration):
    Phi, exit_code = cg(B, Phi)
    Phi[0] = Phi_left
    Phi[-1] = Phi_right
    results[i] = Phi
    if exit_code != 0:
        print(
            f"Conjugate Gradient did not converge at time step {i}, 
exit code: {exit_code}")
        break
print(Phi)

BTCS_solution = []
BTCS_solution = Phi.copy()

plt.figure(figsize=(8, 6))
plt.imshow(results, aspect='auto', cmap='plasma',
           origin='lower', extent=[0, L, 0, t])
plt.colorbar(label='Phi value')
plt.xlabel('Position along the bar')
plt.ylabel('time')
plt.title('Evolution of Phi over time')
plt.show()
```

```
[1.         0.99979268 0.99955909 0.99928806 0.99898205 0.99864372
 0.99827559 0.99788025 0.99746012 0.99701764 0.99655514 0.99607491
 0.99557919 0.99507013 0.99454987 0.99402045 0.99348388 0.9929421
 0.99239699 0.99185037 0.99130403 0.99075966 0.99021893 0.98968344
 0.98915471 0.98863423 0.98812344 0.98762368 0.98713628 0.98666248
 0.98620348 0.98576042 0.98533438 0.98492638 0.98453738 0.9841683
 0.98381998 0.98349323 0.98318878 0.9829073  0.98264943 0.98241573
 0.98220672 0.98202283 0.98186449 0.98173201 0.98162569 0.98154575
 0.98149236 0.98146564 0.98146564 0.98149236 0.98154575 0.98162569
 0.98173201 0.98186449 0.98202283 0.98220672 0.98241573 0.98264943
 0.9829073  0.98318878 0.98349323 0.98381998 0.9841683  0.98453738
 0.98492638 0.98533438 0.98576042 0.98620348 0.98666248 0.98713628
```

```
 0.98762368 0.98812344 0.98863423 0.98915471 0.98968344 0.99021893
 0.99075966 0.99130403 0.99185037 0.99239699 0.9929421  0.99348388
 0.99402045 0.99454987 0.99507013 0.99557919 0.99607491 0.99655514
 0.99701764 0.99746012 0.99788025 0.99827559 0.99864372 0.99898205
 0.99928806 0.99955909 0.99979268 1.        ]
```

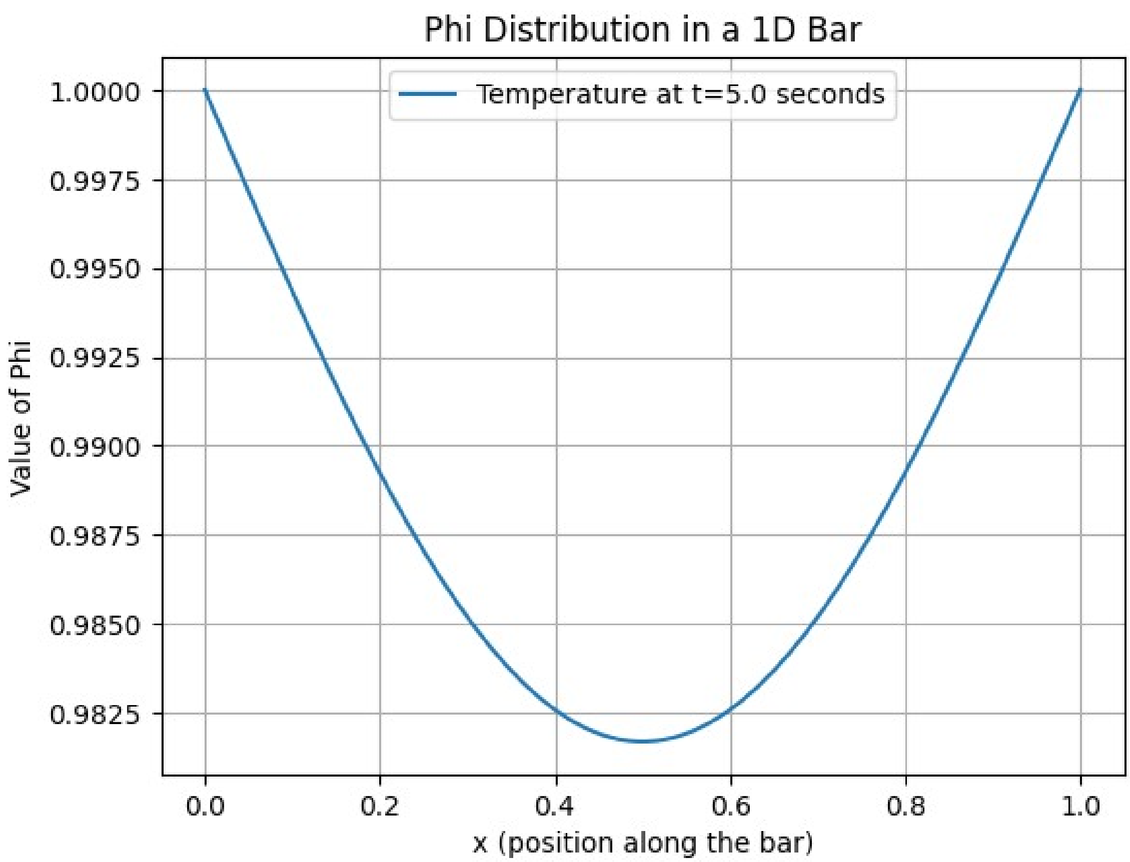


3. exact solution by using seperation of variables method :

```
import numpy as np
import matplotlib.pyplot as plt


N = 100 # number of nodes
L = 1 # length of bar
alfa = 0.1 # diffusion number
dt = 0.0005 # time stepp
iteration = 10000
t = iteration * dt # time for computing exact solution
n_terms = 100  # Number of terms in the exact solution series
```

```python
x = np.linspace(0, L, N)

def Phi_exact(x, t, n_terms, alfa):
    Phi = 1
    for n in range(1, n_terms + 1):
        B_n = 4 / (n * np.pi) * (1 - (-1)**n)
        Phi -= B_n * np.sin(n * np.pi * x) * \
            np.exp(-(n**2 * np.pi**2 * alfa) * t)
    return Phi
Phi = Phi_exact(x, t, n_terms, alfa)
print(Phi)
exact_solution = []
exact_solution = Phi.copy()

plt.plot(x, Phi, label=f'Temperature at t={t} seconds')
plt.xlabel('x (position along the bar)')
plt.ylabel('Value of Phi')
plt.title('Phi Distribution in a 1D Bar')
plt.grid(True)
plt.legend()
plt.show()
```

```
[1.         0.99941894 0.99883846 0.99825915 0.99768159 0.99710637
 0.99653406 0.99596524 0.99540048 0.99484036 0.99428543 0.99373625
 0.99319338 0.99265737 0.99212874 0.99160805 0.9910958  0.99059252
 0.99009871 0.98961488 0.9891415  0.98867905 0.988228   0.98778881
 0.98736191 0.98694773 0.9865467  0.98615922 0.98578567 0.98542643
 0.98508187 0.98475233 0.98443814 0.98413963 0.98385708 0.98359079
 0.98334102 0.98310802 0.98289203 0.98269327 0.98251194 0.98234821
 0.98220226 0.98207423 0.98196425 0.98187243 0.98179886 0.98174362
 0.98170676 0.98168832 0.98168832 0.98170676 0.98174362 0.98179886
 0.98187243 0.98196425 0.98207423 0.98220226 0.98234821 0.98251194
 0.98269327 0.98289203 0.98310802 0.98334102 0.98359079 0.98385708
 0.98413963 0.98443814 0.98475233 0.98508187 0.98542643 0.98578567
 0.98615922 0.9865467  0.98694773 0.98736191 0.98778881 0.988228
 0.98867905 0.9891415  0.98961488 0.99009871 0.99059252 0.9910958
 0.99160805 0.99212874 0.99265737 0.99319338 0.99373625 0.99428543
 0.99484036 0.99540048 0.99596524 0.99653406 0.99710637 0.99768159
 0.99825915 0.99883846 0.99941894 1.        ]
```




```
FTCS_error = np.linalg.norm(abs(exact_solution - FTCS_solution))
BTCS_error = np.linalg.norm(abs(exact_solution - BTCS_solution))
print(f"FTCS error compared to the exact solution is {FTCS_error},
BTCS error compared to the exact solution is {BTCS_error}")
```

```
FTCS error compared to the exact solution is 0.06448229787977347, BTCS
error compared to the exact solution is 0.015006610761898962
```

BTCS method is more accurate than FTCS in this problem