# WEEK 4

2020136006 정성원

**Heat equations**

Source term을 포함하는 2차원 Heat equation이 다음과 같이 주어진다.

$$\frac{\partial \phi}{\partial t} = \alpha\left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2}\right) + S(x,y), \qquad -1 \le x \le 1, -1 \le y \le 1.$$

균일한 초기 및 경계조건은 $\phi(x,y,0) = 0, \phi(\pm 1,y,t) = 0, \phi(x,\pm 1,t) = 0$이며, 본 문제에서 열전도율 $\alpha$는 1로 주어진다.

1. Source term, $S(x,y) = 2(2 - x^2 - y^2)$일 때, $\phi$의 대한 exact solution을 구하시오.

시간에 대한 미분 term 이 있으면 계산이 어려워지므로 시간에 대한 미분 term 을 0 으로 바꾸고 계산한다. $\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = -2(2 - x^2 - y^2)$를 계산하기 위해서는 $\phi = a(x^2 - 1)(y^2 - 1)$로 가정할 수 있다. 해당 값을 식에 넣어서 계산을 진행하면 a=1 라는 것을 알 수 있었다. $\phi = (x^2 - 1)(y^2 - 1)$

2. uniform 격자계에서 시간에 대하여 Crank-Nicolson method를, 공간에 대하여 2차 central difference scheme을 사용하여 정상상태 (steady state)에 도달하도록 방정식을 푸시오. Exact solution과 수치해석한 정상상태의 solution을 시간간격 $\Delta t$와 $x, y$ 방향 격자수 (각각 $N, M$)에 변화를 주어 plot하시오.

```
def S(x,y):
    s = 2*(2-x**2-y**2)
    return s
```

로 S 를 정의한다.

```
n = 21
alpha = 1 # thermal diffusivity
x_list = np.linspace(-1, 1, n) # x grid points
y_list = np.linspace(-1, 1, n) # y grid points
X, Y = np.meshgrid(x_list, y_list) # create a meshgrid
h = x_list[1] - x_list[0] # grid spacing
dt = 0.1 # time step
t = 0 # initial time
beta = alpha * dt / (h**2) / 2
```

초기 값을 넣는다.

```python
pi = np.zeros((n, n)) # initialize pi
I = np.eye(n-2) # identity matrix
```

$\phi$행렬을 정의하고 I 행렬을 정의한다.

```python
L = np.zeros((n-2, n-2)) # initialize pi
for i in range(n-2):
    L[i, i] = -2

    if i == 0:
        L[i, i+1] = 1
    elif i == n-3:
        L[i, i-1] = 1
    else:
        L[i, i-1] = 1
        L[i, i+1] = 1
A = (I - beta * L)
R = S(X[1:-1,1:-1], Y[1:-1,1:-1]) + (I + beta * L) @ ((I + beta * L) @ pi[1:-1, 1:-1].T).T

psi = np.linalg.solve(A, R)  # solve the linear system
pi_new = psi @ np.linalg.inv((I - beta * L).T)
pi_full = np.zeros((n, n))  # initialize full pi
pi_full[1:-1, 1:-1] = pi_new
```

$$(I-\beta L_x-\beta L_y)\phi_{ij}^{n+1} = (I-\beta L_x-\beta L_y)\phi_{ij}^{n} + \frac{1}{2}(S^{n+1}+S^{n})\Delta t$$

$$(I-\beta L_x)(I-\beta L_y)\phi_{ij}^{n+1} \overset{-?}{+} \beta^2 L_x L_y \phi_{ij}^{n+1}$$

$$= (I+\beta L_x)(I+\beta L_y)\phi_{ij}^{n} + \frac{1}{2}(S^{n+1}+S^{n})\underset{\checkmark}{\Delta t} + \beta^2 L_x L_y \phi_{ij}^{n}$$

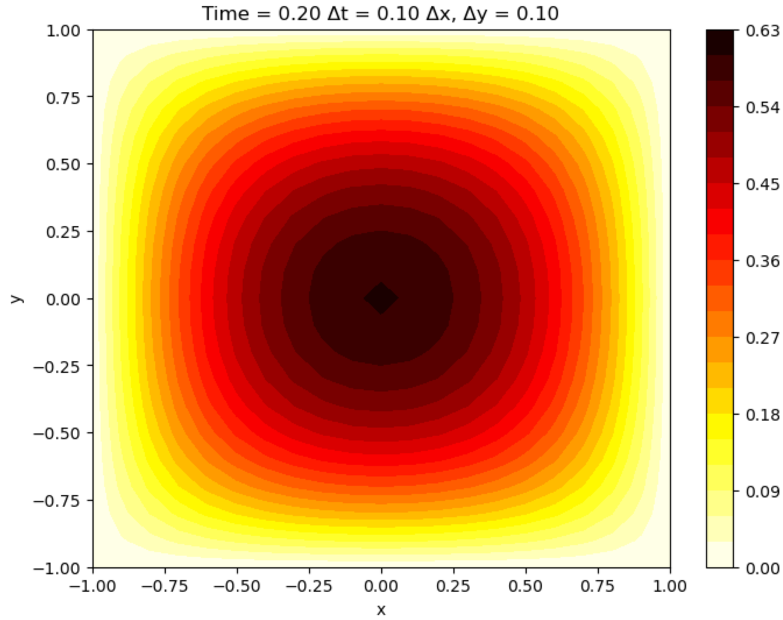
$$(I-\beta L_x)\underbrace{(I-\beta L_y)\phi_{ij}^{n+1}}_{\psi} = \underbrace{(I+\beta L_x)(I+\beta L_y)\phi_{ij}^{n} + \frac{1}{2}(S^{n+1}+S^{n})\Delta t}_{R} - \underbrace{\beta^2 L_x L_y(\phi_{ij}^{n+1}-\phi_{ij}^{n})}_{e \sim O(\Delta t^4)}$$

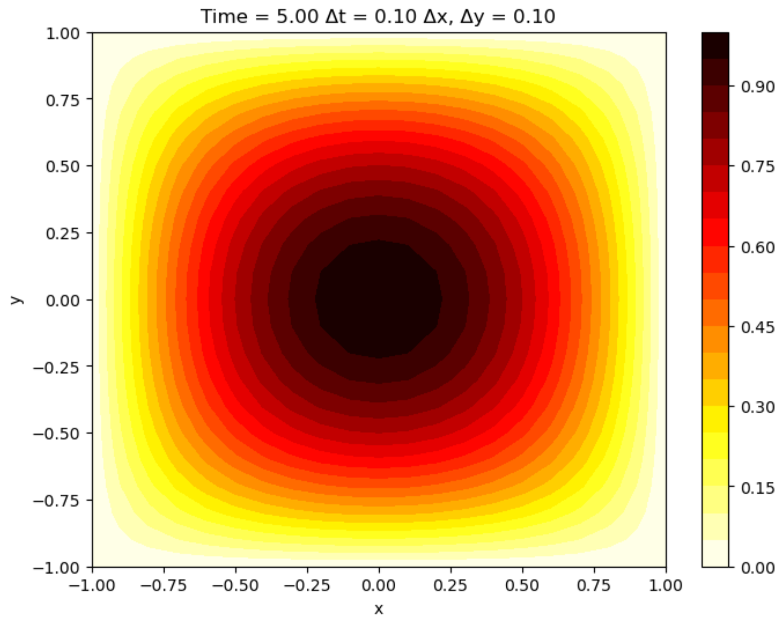
$$L_y = \begin{pmatrix} -2 & 1 & 0 & \cdots \\ 1 & -2 & 1 & \cdots \\ \cdots & 1 & -2 & 1 \ \cdots \\ & & 1 & -2 \ 1 \\ & & & 1 \ -2 \end{pmatrix}\begin{pmatrix} \phi_{11} & \phi_{12} & \cdots \\ \phi_{21} \\ \phi_{31} \end{pmatrix}$$

$$\underbrace{(I-\beta L_x)}_{A}\underbrace{\psi_{ij}}_{x} = \underbrace{R_{ij}}_{b} \qquad \beta L_x = \begin{pmatrix} -2\beta & \beta & \cdots \\ \beta & -2\beta & \cdots \\ \cdots & & -2\beta \end{pmatrix} \quad i_y = 1, \dots, n_y - 1$$

$$= \begin{bmatrix} 1+2\beta & -\beta & \cdots & 0 \\ -\beta & 1+2\beta & -\beta & \cdots \\ \vdots & & \ddots & \\ & & -\beta \ \ 1+2\beta & -\beta \\ 0 & & -\beta & 1+2\beta \end{bmatrix}\underbrace{\begin{bmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_{n_x-1} \end{bmatrix}}_{(n-2)\times(n-2)} = \underbrace{\begin{bmatrix} R_1 + \beta\psi_0 \\ R_2 \\ \vdots \\ R_{n_x-1} + \beta\psi_{n_x} \end{bmatrix}}_{(n-2)\times(n-2)}$$

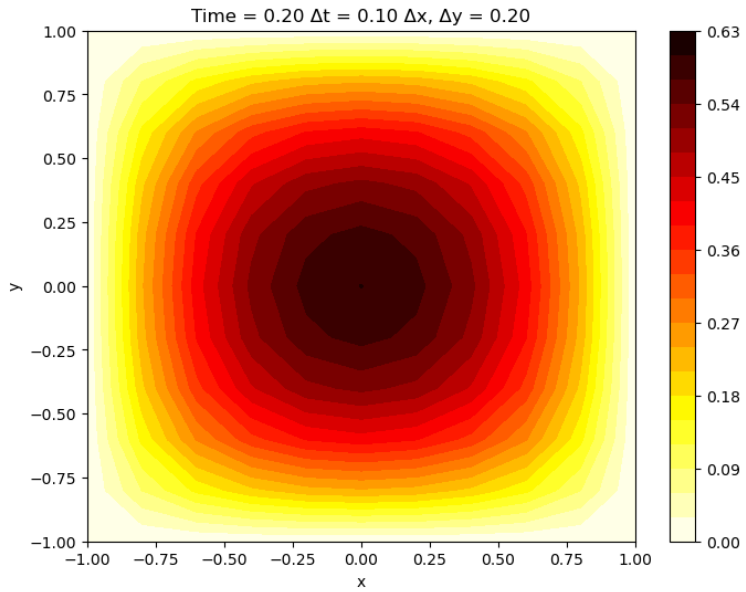
삼각 대각 행렬을 만들기 위해서 L 을 정의한다. 위와 같은 공식을 만족시키기 위해서 알와 같은 psi 행렬을 정의하고 이를 풀고 pi_full 에 pi 값을 정의한다.

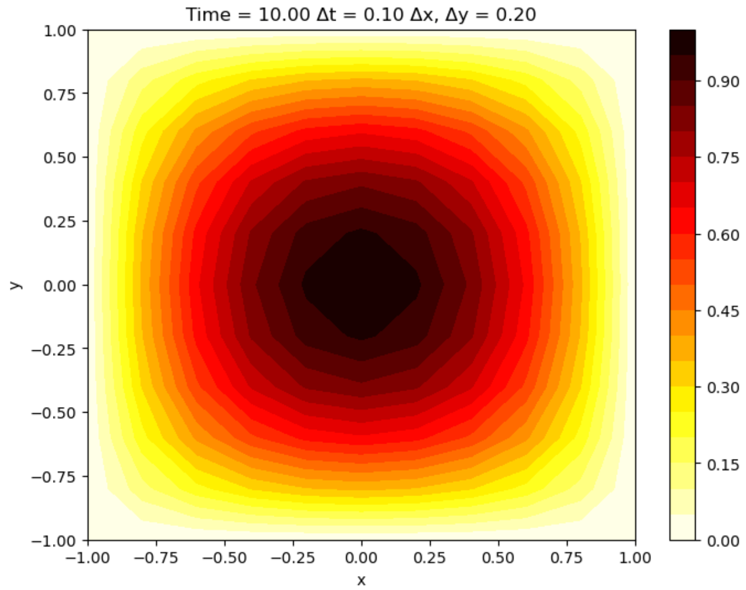



시각화하면 다음과 같다.

Dx,dy 를 바꾸면 다음과 같다.

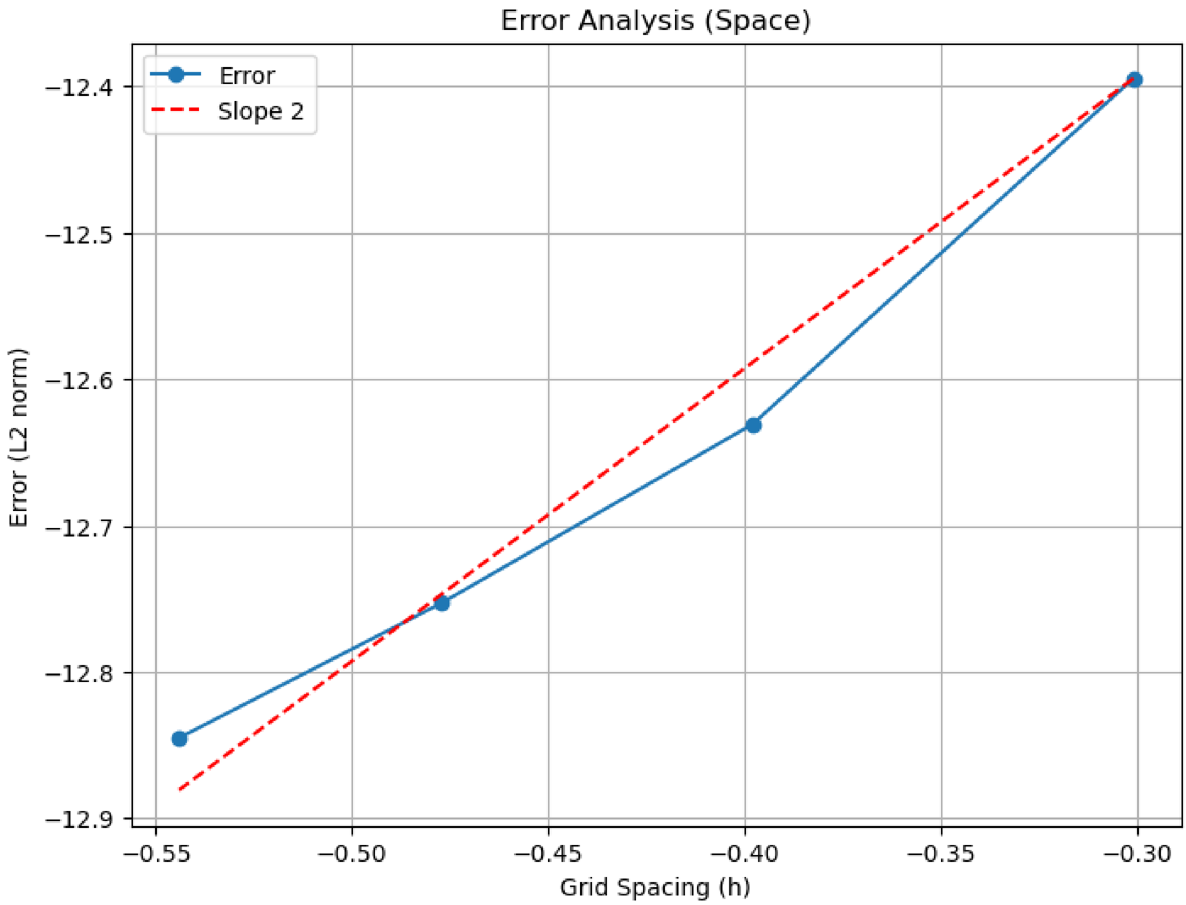

Time = 0.20 Δt = 0.10 Δx, Δy = 0.20
y
x
0.63
0.54
0.45
0.36
0.27
0.18
0.09
0.00


Time = 10.00 Δt = 0.10 Δx, Δy = 0.20
y
x
0.90
0.75
0.60
0.45
0.30
0.15
0.00

3. 수치해석 결과의 order of accuracy를 시간과 공간에 대하여 분석하시오.

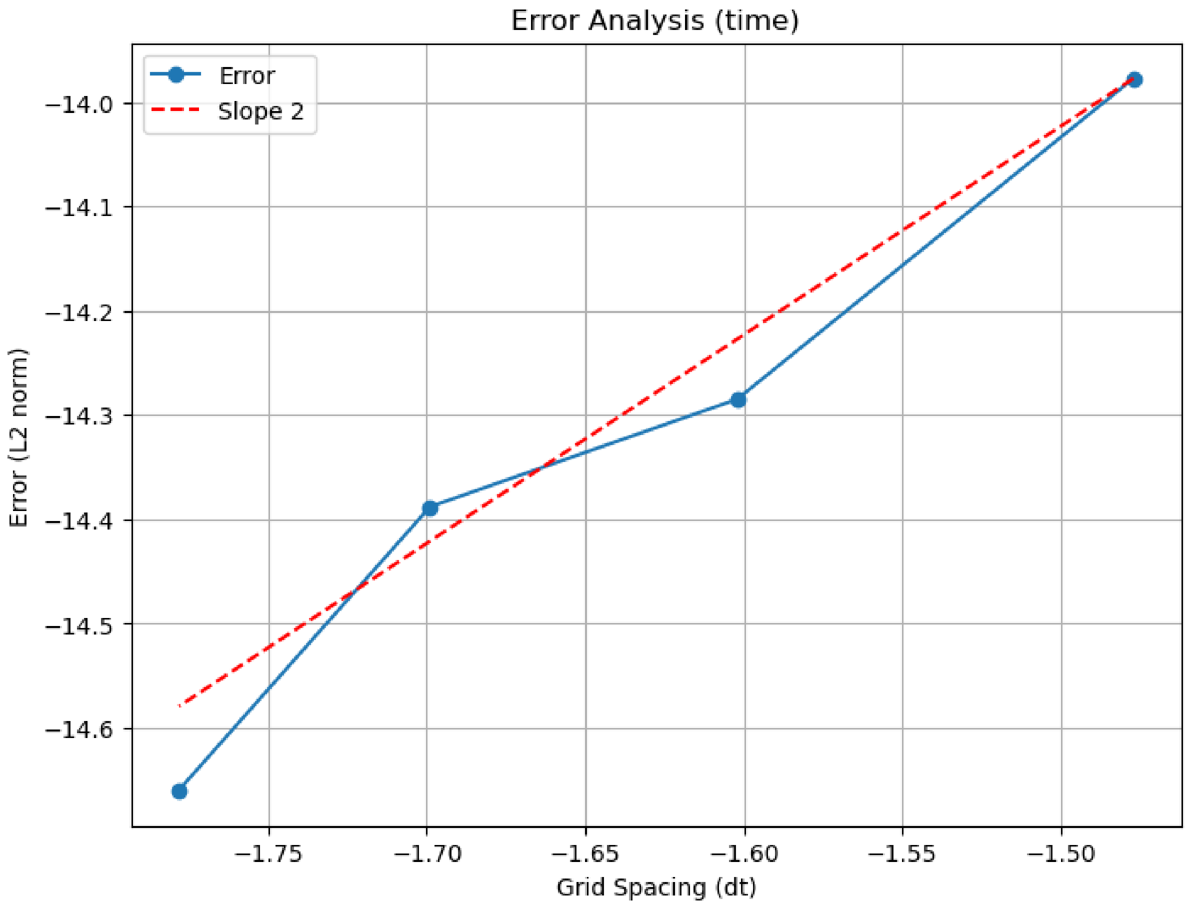
```python
# Space
h_list = []
error_list = []
for n in tqdm(range(5,9,1)):
    alpha = 1 # thermal diffusivity
    x_list = np.linspace(-1, 1, n) # x grid points
    y_list = np.linspace(-1, 1, n) # y grid points
    X, Y = np.meshgrid(x_list, y_list) # create a meshgrid
    h = x_list[1] - x_list[0] # grid spacing
    dt = 0.0061 # time step
    t = 0 # initial time
    beta = alpha * dt / (h**2) / 2

    pi = np.zeros((n, n)) # initialize pi
    I = np.eye(n-2) # identity matrix
    L = np.zeros((n-2, n-2)) # initialize pi

    pi_list = [pi,]

    for i in range(n-2):
        L[i, i] = -2

        if i == 0:
            L[i, i+1] = 1
        elif i == n-3:
            L[i, i-1] = 1
        else:
            L[i, i-1] = 1
            L[i, i+1] = 1

    A = (I - beta * L)

    for j in range(1000):
        t = t + dt

        R = S(X[1:-1,1:-1], Y[1:-1,1:-1])*dt + (I + beta * L) @ ((I + beta * L) @ pi_list[j]
        psi = np.linalg.solve(A, R)  # solve the linear system
        pi_new = np.dot(psi, np.linalg.inv(A.T))
        pi_full = np.zeros((n, n))  # initialize full pi
        pi_full[1:-1, 1:-1] = pi_new
        pi_list.append(pi_full)
    error_list.append(np.linalg.norm(pi_list[-1] - pi_exact(X, Y), 2)*h)
    h_list.append(h)
    time.sleep(0.02)
```

위에서 구한 값을 똑같이 넣고 dx 와 dy 를 변화 시켜서 error_list 에 값을 넣어서 norm2 를 계산한다. 이를 시각화 하면 다음과 같다.




Order 는 2 차인 것을 확인할 수 있다.

위와 똑같이 dt 를 바꿔서 계산하여 나타내면 아래와 같은 그래프가 나오는 것을 확인할 수 있다.




공간과 마찬가지로 기울기가 2 인 것을 확인할 수 있다.