4주차 과제 #6

2022145079 임혜린

본 과제는 Python, VS Code를 사용하였음을 밝힙니다.

공통 조건 (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 α는 1로 주어진다.

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

HW#6-1 (ϕ의 exact solution 구하기)

$\frac{\partial \phi}{\partial t} = \alpha\left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2}\right) + S(x,y)$ $\quad \alpha = 1,\ S(x,y) = 2(2-x^2-y^2)$ 이므로 $\frac{\partial \phi}{\partial t} = \underbrace{\left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2}\right)}_{①} + \underbrace{2(2-x^2-y^2)}_{②}$

① Assume $\phi(x,y,t) = F(x,y)\,G(t)$

then, $FG' = (F_{xx} + F_{yy})G \Rightarrow \frac{G'}{G} = \frac{F_{xx}+F_{yy}}{F}$. Assume that $-\lambda^2$. then $\frac{G'}{G} = -\lambda^2 \Rightarrow \ln G = -\lambda^2 t \Rightarrow G = e^{-\lambda^2 t}$

$\frac{F_{xx}+F_{yy}}{F} = -\lambda^2 \Rightarrow F_{xx} + F_{yy} + \lambda^2 F = 0$ Assume $F(x,y) = H(x)\,Q(y)$. then $H_{xx}Q + HQ_{yy} + \lambda^2 HQ = 0 \Rightarrow \frac{H_{xx}}{H} = -\frac{1}{Q}(\lambda^2 Q + Q_{yy})$. Assume that $-k^2$

then $\begin{cases} H_{xx} + k^2 H = 0 \Rightarrow H(x) = A\cos kx + B\sin kx. \text{ By boundary condition } H(-1) = H(1) = 0,\ k = \frac{m\pi}{2},\ A = 0,\ B = 1 \Rightarrow H(x) = \sin\frac{m\pi}{2}(x+1)\ (m = 1, 2, \cdots) \\ \lambda^2 Q + Q_{yy} - k^2 Q = 0 \quad Q_{yy} + (\lambda^2 - k^2)Q = 0 \Rightarrow Q = C\cos\sqrt{\lambda^2-k^2}\,y + D\sin\sqrt{\lambda^2-k^2}\,y. \end{cases}$

↳ 범위 [-1, 1]이므로 평행이동해서 [0,2]로

By Boundary condition, $\sqrt{\lambda^2-k^2} = \frac{n\pi}{2}$, $C = 0,\ D = 1 \Rightarrow Q(y) = \sin\frac{n\pi}{2}(y+1)\ (n = 1, 2 \ldots)$

$\sqrt{\lambda^2-k^2} = \frac{n\pi}{2}$, $k = \frac{m\pi}{2} \Rightarrow \lambda^2 = \frac{n^2\pi^2}{4} + \frac{m^2\pi^2}{4} \Rightarrow \lambda = \frac{\pi}{2}\sqrt{n^2+m^2}$

$F = HQ \Rightarrow F = \sum_{m=1}^{\infty} \sin\frac{m\pi}{2}(x+1) \sum_{n=1}^{\infty} \sin\frac{n\pi}{2}(y+1)$ $\qquad \therefore \phi = FG = \left(\sum_{m=1}^{\infty} \sin\frac{m\pi}{2}(x+1) \sum_{n=1}^{\infty} \sin\frac{n\pi}{2}(y+1)\right) e^{-\frac{\pi^2}{4}(m^2+n^2)t}$

② 정상상태에서 $0 = \left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2}\right) + 2(2-x^2-y^2) \Rightarrow \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 2x^2+2y^2-4$

By Boundary Condition, assume $\phi(x,y) = a(1-x^2)(1-y^2)$

$\frac{\partial^2 \phi}{\partial x^2} = -2a(1-y^2)$, $\frac{\partial^2 \phi}{\partial y^2} = -2a(1-x^2)$. $\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 2a(x^2+y^2-2) = 2x^2+2y^2-4$ $\quad \therefore a = 1$ $\quad \therefore \phi = (1-x^2)(1-y^2)$

$\therefore \phi = C_{mn}\left(\sum_{m=1}^{\infty} \sin\frac{m\pi}{2}(x+1) \sum_{n=1}^{\infty} \sin\frac{n\pi}{2}(y+1)\right) e^{-\frac{\pi^2}{4}(m^2+n^2)t} + (1-x^2)(1-y^2)$ $\quad$ By boundary condition $\phi(x,y,0) = 0$, $C_{mn} = \frac{-(1-x^2)(1-y^2)}{\sum_{m=1}^{\infty} \sin\frac{m\pi}{2}(x+1) \sum_{n=1}^{\infty} \sin\frac{n\pi}{2}(y+1)}$

$\phi = \iint_0^{\infty} C \sin\frac{k_x}{2}(x+1) \sin\frac{k_y}{2}(y+1)^{-(k_x^2+k_y^2)t}\, dk_x\, dk_y$

$C = -4\int_{-1}^{1}\int_{-1}^{1} (1-x^2)(1-y^2)\sin(k_x(x+1))\sin(k_y(y+1))\,dx\,dy$

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 Δt와 x, y 방향 격자수 (각각 N, M)에 변화를 주어 plot하시오.

Crank-Nicolson scheme은 시간 미분을 전 시간 단계와 다음 시간 단계의 중앙 평균으로 근사한다. Heat equation의 지배방정식에 이를 적용하면 아래와 같다.

$$\frac{\phi_{i,j}^{n+1} - \phi_{i,j}^{n}}{\Delta t} = \frac{\alpha}{2}\left[\nabla^2\phi_{i,j}^{n+1} + \nabla^2\phi_{i,j}^{n}\right] + \frac{1}{2}\left[S_{i,j}^{n+1} + S_{i,j}^{n}\right]$$

n은 time step, i,j는 spatial index이며 이때

$$\nabla^2\phi\big|_{i,j} \approx \frac{\phi_{i+1,j}^{n} - 2\phi_{i,j}^{n} + \phi_{i-1,j}^{n}}{\Delta x^2} + \frac{\phi_{i,j+1}^{n} - 2\phi_{i,j}^{n} + \phi_{i,j-1}^{n}}{\Delta y^2}$$

이다. $\Delta x=\Delta y$이므로 $\Delta x=\Delta y=h$로, $\beta=\alpha\Delta t/2h^2$으로 놓고 좌변을 unknown, 우변을 known으로 정리하면 아래와 같다.

$$\phi_{i,j}^{n+1} - \beta\Big[\,\phi_{i+1,j}^{n+1} + \phi_{i-1,j}^{n+1} + \phi_{i,j+1}^{n+1} + \phi_{i,j-1}^{n+1} - 4\phi_{i,j}^{n+1}\,\Big]$$

$$= \ \phi_{i,j}^{n} + \beta\Big[\,\phi_{i+1,j}^{n} + \phi_{i-1,j}^{n} + \phi_{i,j+1}^{n} + \phi_{i,j-1}^{n} - 4\phi_{i,j}^{n}\,\Big] + \frac{\Delta t}{2}(S_{i,j}^{n} + S_{i,j}^{n+1})$$

이때 operator Lx, Ly를 적용하면, 식을 아래와 같이 변형할 수 있다.

$$L_x(\Phi_{ij}) = \Phi_{i+1,j} - 2\Phi_{i,j} + \Phi_{i-1,j}$$
$$L_y(\Phi_{ij}) = \Phi_{i,j+1} - 2\Phi_{i,j} + \Phi_{i,j-1}$$

$$(I - \beta L_x - \beta L_y)\,\Phi_{i,j}^{n+1} = (I - \beta L_x - \beta L_y)\,\Phi_{i,j}^{n} + \frac{\Delta t}{2}\left(S_{i,j}^{n+1} + S_{i,j}^{n}\right)$$

$$(I - \beta L_x)(I - \beta L_y)\,\Phi_{i,j}^{n+1} - \beta^2 L_x L_y \Phi_{i,j}^{n+1}$$
$$= (I + \beta L_x)(I + \beta L_y)\,\Phi_{i,j}^{n} - \beta^2 L_x L_y \Phi_{i,j}^{n} + \frac{\Delta t}{2}\left(S_{i,j}^{n+1} + S_{i,j}^{n}\right)$$

β^2항은 n에 따라 크게 변화하지 않으므로 양 변의 β^2항이 상쇄 가능하다 가정하면 결과적으로 아래와 같은 식을 얻을 수 있다.

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

또한 상쇄한 β^2항이 Δt^2에 비례하므로 dt에 대한 L2norm error가 2차로 나올 것을 예상할 수 있다.

마지막으로 얻었던 식의 좌변의 I-βL_2를 제외한 부분을 ψ, 우변을 R이라고 하면 식은 (I-βL_x)ψ=R 이 된다. 이 형태에서 한 번 풀어서 ψ를 알아낸 후, (I-βL_y)ϕ=ψ를 이용하여 ϕ를 알아내는 방식으로 코드를 작성하였다. 즉,

$$(I - \beta L_x)\,\psi = R$$

이와 같이 변형할 수 있으므로 이것을 np.linalg.solve로 한 번 푼 후, 이렇게 찾은 ψ를 이용하여

$$\Phi_{i,j}^{n+1} = \psi\,\left[(I - \beta L_y)^T\right]^{-1}$$

이를 진행하여 ϕ를 구한 후, 이를 이용하여 다시 다음 t에서의 ϕ를 구해 나가는 방식이다.

```
alpha=1
n=41

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

def exact_pi(x, y):
    return (1-x**2)*(1-y**2)


x_hani=np.linspace(-1, 1, n)
x_list=x_hani[1:-1]
y_hani=np.linspace(-1, 1, n)
y_list=y_hani[1:-1]
X,Y=np.meshgrid(x_hani, y_hani)
h=x_hani[1]-x_hani[0]
dt=0.1
t=0
beta=alpha*dt/(2*h**2)
phi_exact = exact_pi(X, Y)

pi=np.zeros((n, n))
pi_list=[pi,]

I_bLx_main = (1-2*beta)*np.eye(n-2)
I_bLx_upper = (beta)*np.eye(n-2, k=1)
I_bLx_lower = (beta)*np.eye(n-2, k=-1)
I_bLx = I_bLx_lower + I_bLx_main + I_bLx_upper

I__bLx_main = (1+2*beta)*np.eye(n-2)
I__bLx_upper = (-1*beta)*np.eye(n-2, k=1)
I__bLx_lower = (-1*beta)*np.eye(n-2, k=-1)
I__bLx = I__bLx_lower + I__bLx_main + I__bLx_upper

error_list=[]
```

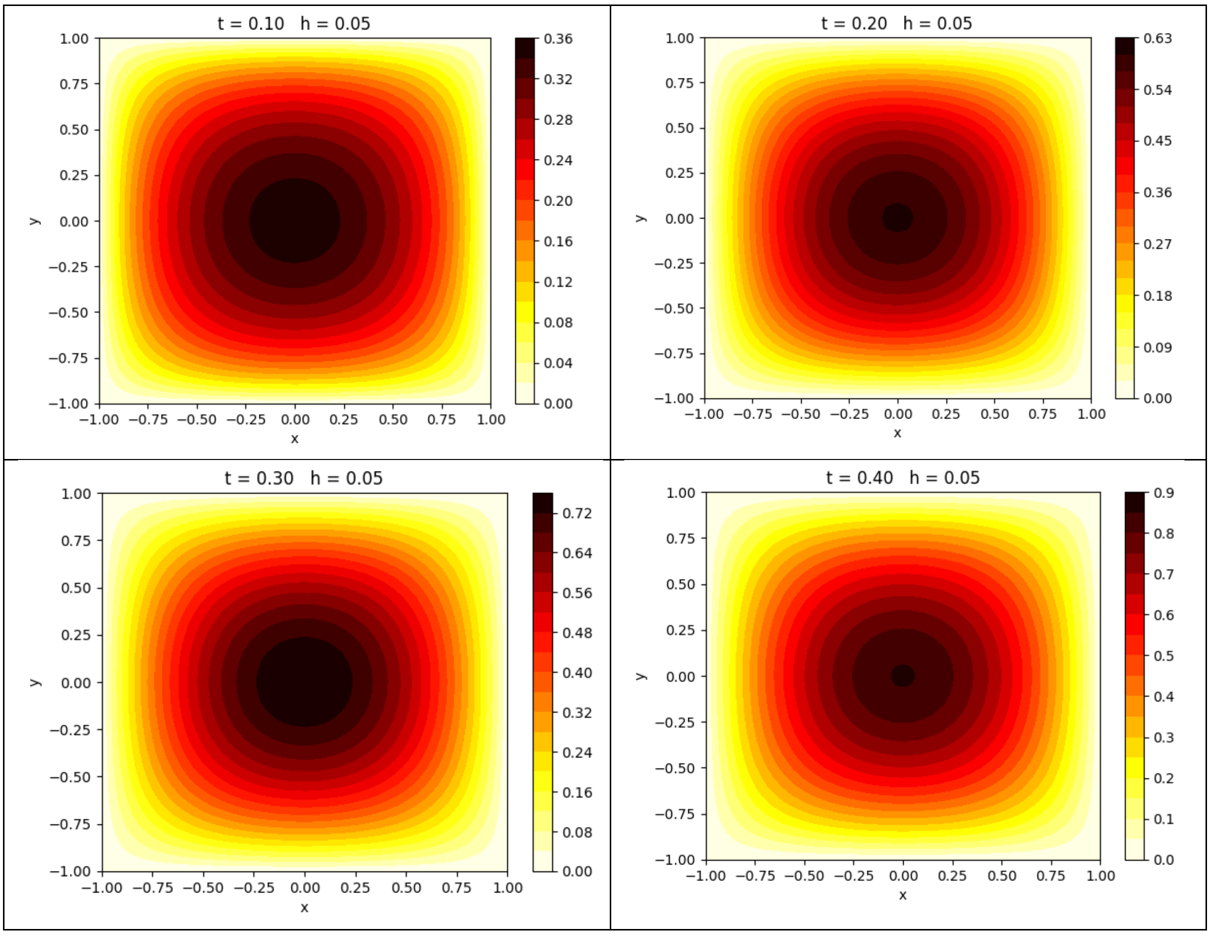
```
for j in range(100):
    t+=dt
    R=I_bLx @ (I_bLx @ pi_list[j][1:-1, 1:-1].T).T +S(X[1:-1,1:-1], Y[1:-1,1:-1])*dt
    psi=np.linalg.solve(I__bLx, R)
    pi_new = psi @ np.linalg.inv(I__bLx.T)
    pi_all=np.zeros((n,n))
    pi_all[1:-1, 1:-1]=pi_new
    pi_list.append(pi_all.copy())
```

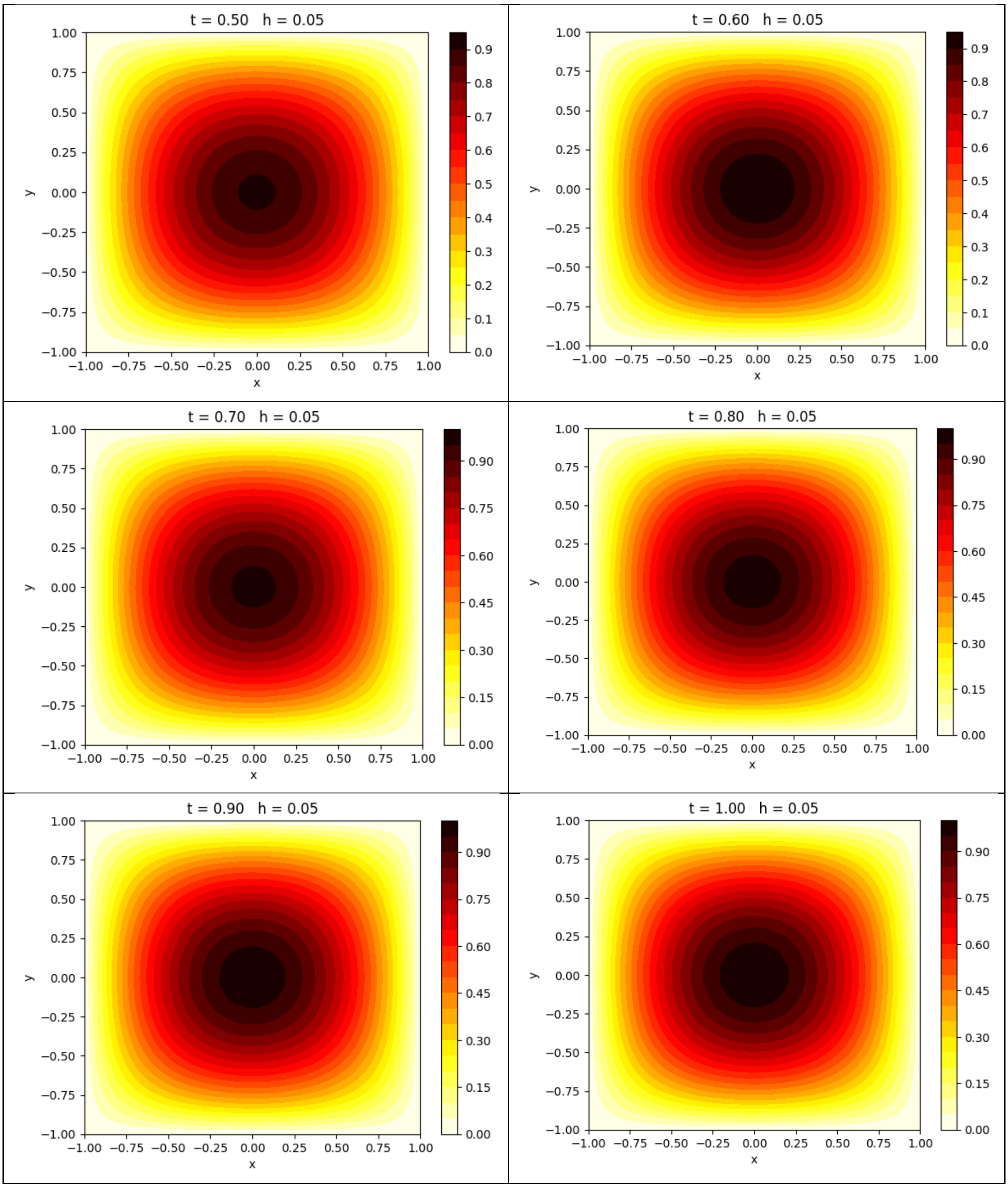
h=x, y 방향 격자수, h=Δx=Δy=2/n, I__bx=2차원 넘파이 배열로 나타낸 I-βL_x, I_bx=2차원 넘파이 배열로 나타낸 I+βL_x, pi_list=각 t에 대한 ϕ를 담은 2차원 넘파이 배열을 담은 리스트이다.

n=41, t=0.1로 설정하였다.

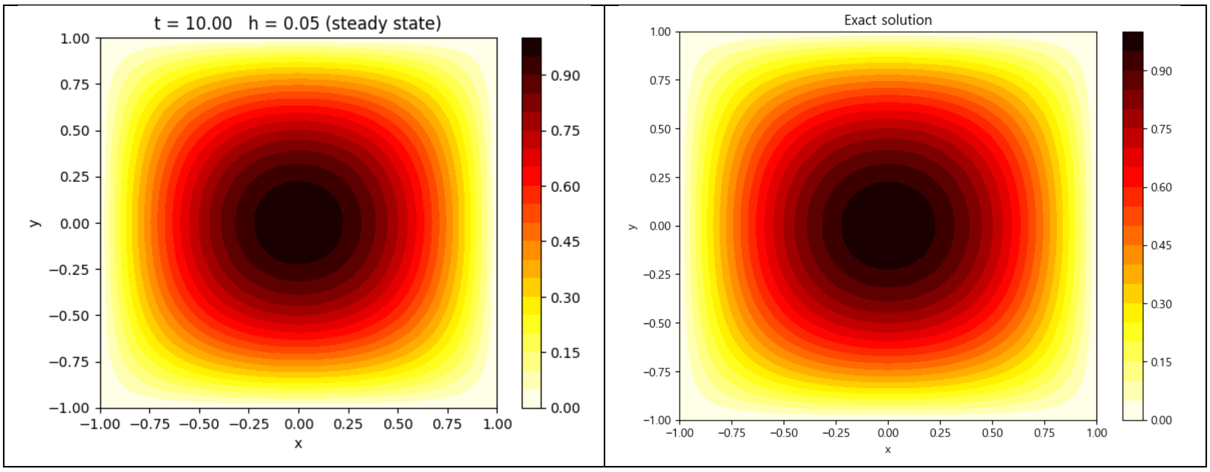
```
    if j in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
        plt.contourf(X, Y, pi_list[j+1], levels=20, cmap='hot_r')
        plt.colorbar()
        plt.xlabel('x')
        plt.ylabel('y')
        plt.title(f't = {t:.2f}    h = {h:.2f}')
        plt.show()
```

이중 초기 t에 대한 등고선을 그려보았다.



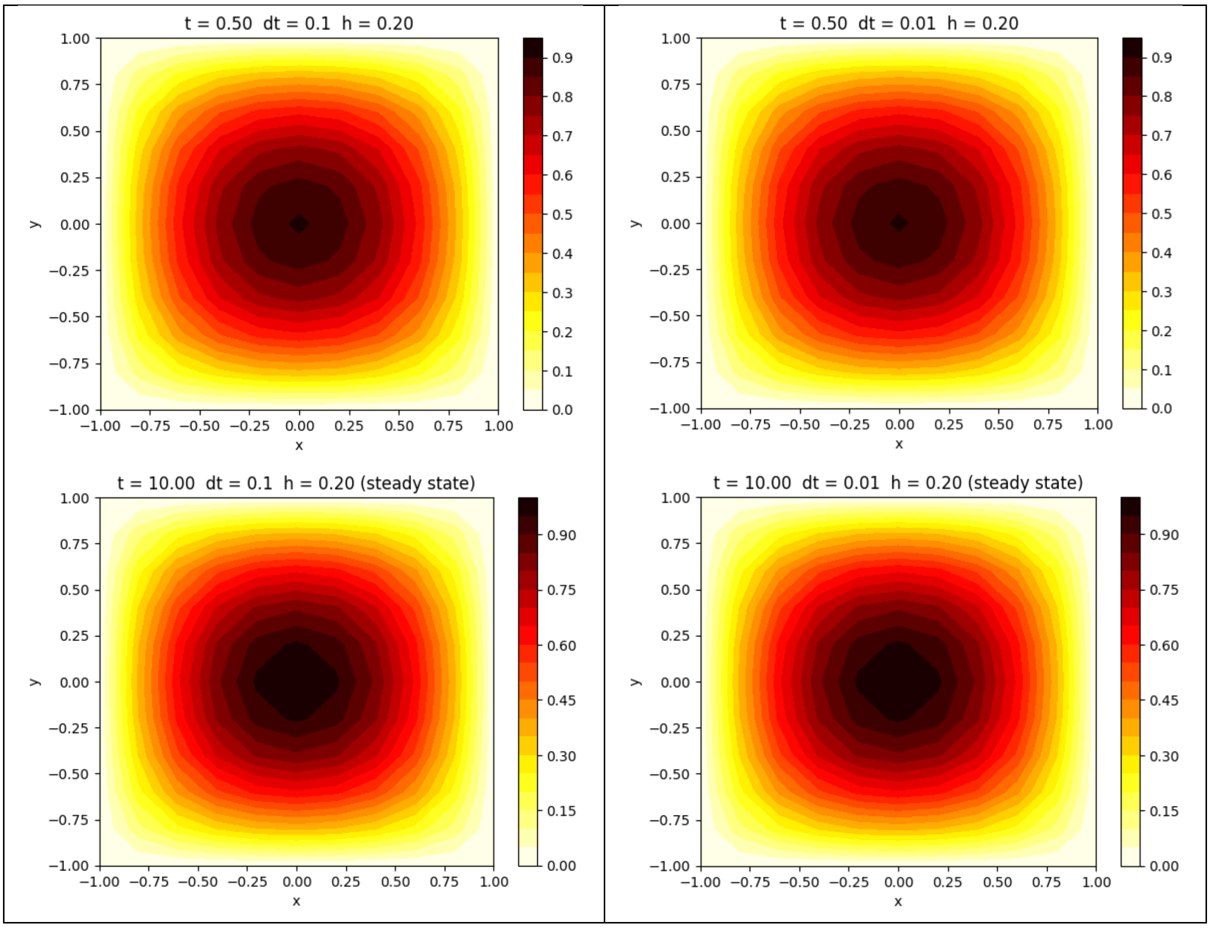


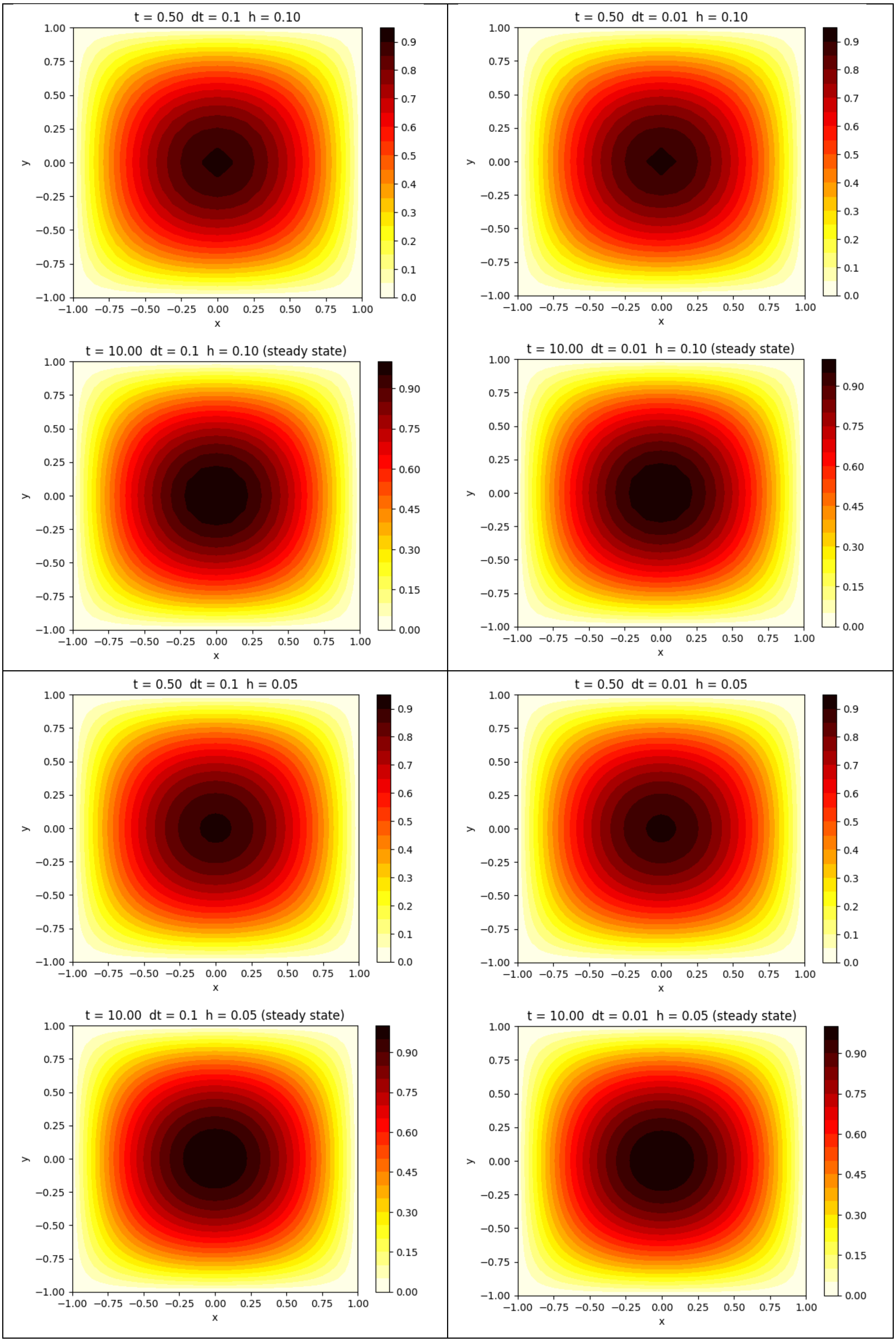
내부 온도가 진동하고 이 진동이 시간이 지날수록 느려진다는 것을 확인하였다. 다음으로 t=10.0일 때의 등고선과 exact solution의 등고선을 비교함으로써 두 등고선이 거의 일치함을 확인하였다.



다음으로 Δt와 h를 바꾸어가며 등고선을 비교하였다. Δt=0.1, 0.01, h=0.2, 0.1, 0.05이며 t가 0.5일 때와 10.0일 때, 총 12개의 등고선을 비교하였다.

그 결과 h가 감소, 즉 n이 증가할수록 등고선이 더욱 스무스하게 그려짐을 확인하였으며, Δt에 대해서는 육안으로 뚜렷한 차이를 찾지 못하였으나 Δt가 작을수록 작은 t에 대해 더 스무스한 결과를 얻게 되는 것으로 예상된다.




t = 0.50 dt = 0.1 h = 0.10
t = 0.50 dt = 0.01 h = 0.10
t = 10.00 dt = 0.1 h = 0.10 (steady state)
t = 10.00 dt = 0.01 h = 0.10 (steady state)
t = 0.50 dt = 0.1 h = 0.05
t = 0.50 dt = 0.01 h = 0.05
t = 10.00 dt = 0.1 h = 0.05 (steady state)
t = 10.00 dt = 0.01 h = 0.05 (steady state)
x
y

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

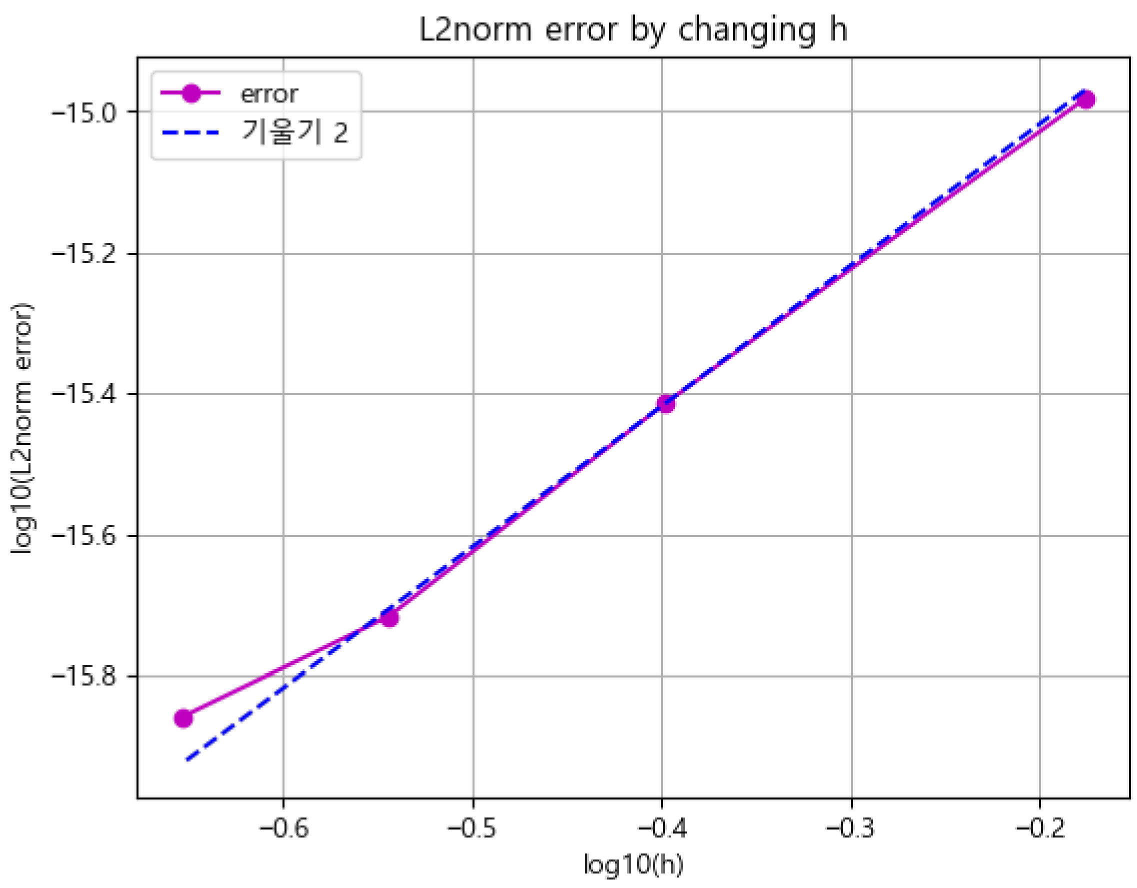
- h의 변화

$\Delta t=0.1$, $t=10.0$(steady state)를 고정해 놓은 뒤, ϕ와 (range(99)한 후 pi_list[-1]) steady state에서의 exact solution $(1-x^2)(1-y^2)$의 L2norm을 구하여 error을 구하였다. n=4, 6, 8, 10으로 격자 수의 변화를 주며 error을 구하며 구한 각 격자 수에서의 error을 error_list에 담아 error_list를 완성하였다.

log10(h)와 log10(error_list)의 그래프를 그려 그래프의 order of accuracy를 구하였다. 또한 기울기가 2인 점선 그래프를 그려 한 눈에 비교할 수 있도록 하였다.

```
error_list=[]
h_list=[]

def exact_pi(x, y):
    return (1-x**2)*(1-y**2)

for n in [4, 6, 8, 10]:
    # 4, 6, 8, 10
    x_hani=np.linspace(-1, 1, n)
    x_list=x_hani[1:-1]
    y_hani=np.linspace(-1, 1, n)
    y_list=y_hani[1:-1]
    X,Y=np.meshgrid(x_hani, y_hani)
    h=x_hani[1]-x_hani[0]
    dt=0.1
    t=0
    beta=alpha*dt/(2*h**2)
    phi_exact = exact_pi(X, Y)

    pi=np.zeros((n, n))
    pi_list=[pi,]

    I_bLx_main = (1-2*beta)*np.eye(n-2)
    I_bLx_upper = (beta)*np.eye(n-2, k=1)
    I_bLx_lower = (beta)*np.eye(n-2, k=-1)
    I_bLx = I_bLx_lower + I_bLx_main + I_bLx_upper

    I__bLx_main = (1+2*beta)*np.eye(n-2)
    I__bLx_upper = (-1*beta)*np.eye(n-2, k=1)
    I__bLx_lower = (-1*beta)*np.eye(n-2, k=-1)
    I__bLx = I__bLx_lower + I__bLx_main + I__bLx_upper

    h_list.append(h)

    for j in range(99):
        t+=dt
        R=I_bLx @ (I_bLx @ pi_list[j][1:-1, 1:-1].T).T +S(X[1:-1,1:-1], Y[1:-1,1:-1])*dt
        psi=np.linalg.solve(I__bLx, R)
        pi_new = psi @ np.linalg.inv(I__bLx.T)
        pi_all=np.zeros((n,n))
        pi_all[1:-1, 1:-1]=pi_new
        pi_list.append(pi_all.copy())
```

```
    error=np.linalg.norm(exact_pi(X, Y)-pi_list[-1], 2)*h
    error_list.append(error.copy())

plt.plot(np.log10(h_list), np.log10(error_list), marker='o', color='m', label='error')
plt.plot(np.log10(h_list), 2*(np.log10(h_list)-np.log10(h_list[1]))+np.log10(error_list[1
plt.xlabel('log10(h)')
plt.ylabel('log10(L2norm error)')
plt.title('L2norm error by changing h')
plt.grid()
plt.legend()
plt.show()

print(f'order of accuracy: {linregress(np.log10(h_list), np.log10(error_list)).slope}')
```



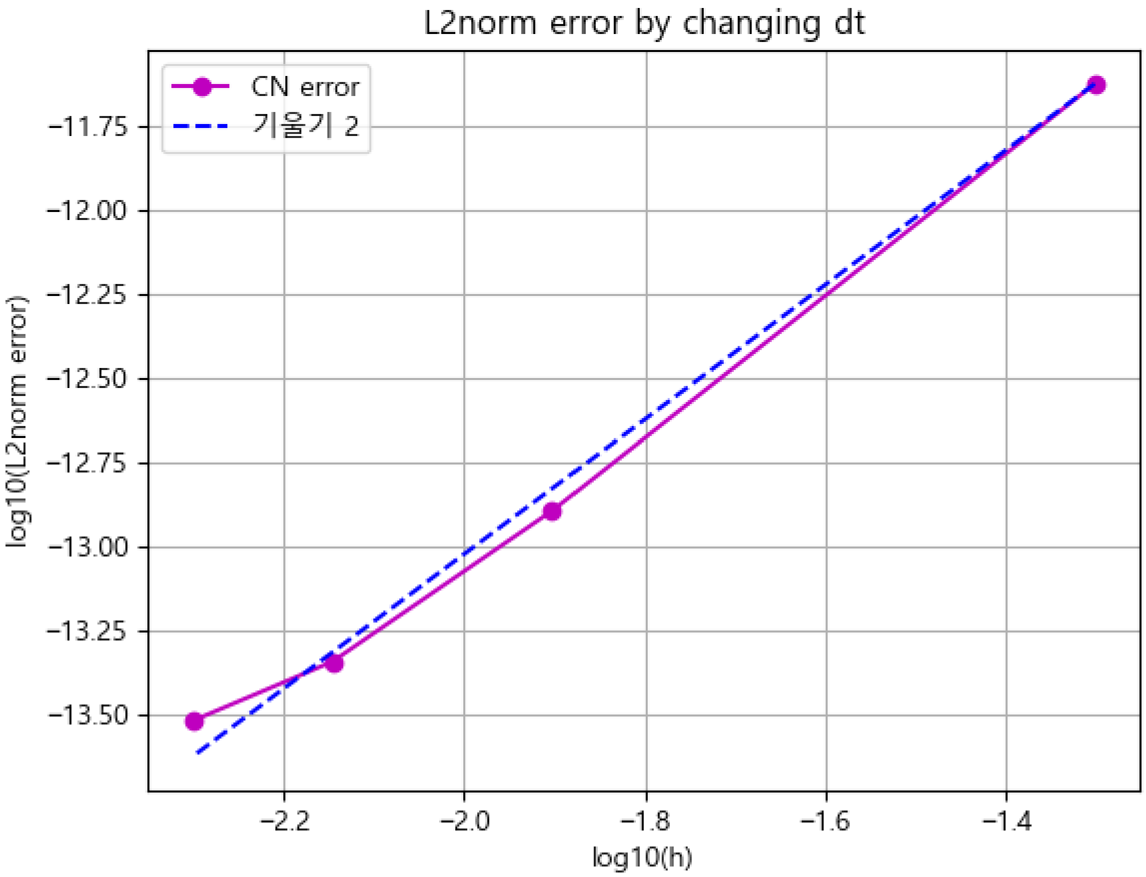

```
order of accuracy: 1.8737030036840558
```

그 결과 order of accuracy가 2에 가깝게 나오는 것을 확인하였다.

- Δt의 변화

비슷한 방법으로 Δt를 변화시키며 진행하였다. 사용된 Δt는 1/20, 1/80, 1/140, 1/200이다.

```python
error_list=[]
dt_list=[]


n=101
x_hani=np.linspace(-1, 1, n)
x_list=x_hani[1:-1]
y_hani=np.linspace(-1, 1, n)
y_list=y_hani[1:-1]
X,Y=np.meshgrid(x_hani, y_hani)
h=x_hani[1]-x_hani[0]
t=0
pi=np.zeros((n, n))


for k in [20, 80, 140, 200]:
    dt=1/k
    dt_list.append(dt)
    pi_list=[pi,]
    beta=alpha*dt/(2*h**2)
    phi_exact = exact_pi(X, Y)

    I_bLx_main = (1-2*beta)*np.eye(n-2)
    I_bLx_upper = (beta)*np.eye(n-2, k=1)
    I_bLx_lower = (beta)*np.eye(n-2, k=-1)
    I_bLx = I_bLx_lower + I_bLx_main + I_bLx_upper

    I__bLx_main = (1+2*beta)*np.eye(n-2)
    I__bLx_upper = (-1*beta)*np.eye(n-2, k=1)
    I__bLx_lower = (-1*beta)*np.eye(n-2, k=-1)
    I__bLx = I__bLx_lower + I__bLx_main + I__bLx_upper
    for j in range(10*k):
        t+=dt
        R=I_bLx @ (I_bLx @ pi_list[j][1:-1, 1:-1].T).T +S(X[1:-1,1:-1], Y[1:-1,1:-1])*dt
        psi=np.linalg.solve(I__bLx, R)
        pi_new = psi @ np.linalg.inv(I__bLx.T)
        pi_all=np.zeros((n,n))
        pi_all[1:-1, 1:-1]=pi_new
        pi_list.append(pi_all.copy())

    error=np.linalg.norm(exact_pi(X, Y)-pi_list[-1], 2)*h
    error_list.append(error.copy())

print(dt_list)
print(error_list)

plt.plot(np.log10(dt_list), np.log10(error_list), marker='o', color='m', label='CN error'
plt.plot(np.log10(dt_list), 2*(np.log10(dt_list)-np.log10(dt_list[0]))+np.log10(error_lis
plt.xlabel('log10(h)')
plt.ylabel('log10(L2norm error)')
plt.title('L2norm error by changing dt')
plt.grid()
plt.legend()
plt.show()

print(f'order of accuracy: {linregress(np.log10(dt_list), np.log10(error_list)).slope}')
```




```
order of accuracy: 1.939069822610387
```

이 또한 기울기가 2에 가깝게 나오는 것을 확인할 수 있었다.