Operating Splitting Method(OSM)

Time splitting 또는 Fractional step 방법이라 불리우는 operator splitting의 기본 아이디어는 다음과 같다:

다음의 initial value equation이 있다고 가정하자.

$$\frac{\partial u}{\partial t} = Lu$$

여기서 L은 operator이다. 이것은 u에 대하여 m개의 선형결합으로 다시 쓸 수 있다고 가정하자.

$$Lu = L_1 u + L_2 u + \cdots + L_m u.$$

또한 m개의 operator 각각에 대하여 time step n부터 $n+1$까지 변수 u를 업데이트하는 scheme을 알고 있다고 가정하자. 즉,

$$\begin{aligned} u^{n+1} &= U_1(u^n, \Delta t), \\ u^{n+1} &= U_2(u^n, \Delta t), \\ &\cdots \\ u^{n+1} &= U_m(u^n, \Delta t). \end{aligned}$$

이제 operator splitting의 한 열은 다음 열에 의해 연쇄적으로 업데이트되면서 n에서 $n+1$을 얻게 될 것이다.

$$\begin{aligned} u^{n+\frac{1}{m}} &= U_1(u^n, \Delta t), \\ u^{n+\frac{2}{m}} &= U_2(u^{n+\frac{1}{m}}, \Delta t), \\ &\cdots \\ u^{n+1} &= U_m(u^{n+\frac{m-1}{m}}, \Delta t). \end{aligned}$$

2

OSM에 의한 열방정식의 풀이

2차원 열방정식에 대한 근사해를 OS방법을 이용하여 풀어보도록 하자.

2차원 열방정식은 다음과 같다.

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \qquad 0 < x < 1, \;\; 0 < y < 1, \qquad t > 0 \tag{1}$$

이 때 경계조건은 $u_{xx}(0,y,t) = u_{xx}(1,y,t) = u_{yy}(x,0,t) = u_{yy}(x,1,t) = 0 \; (t > 0)$이고, 초기조건은 $u(x,y,0) = \sin(\pi x)\sin(\pi y)$을 만족한다. 해석해는 $u(x,y,t) = \sin(\pi x)\sin(\pi y)e^{-2\pi^2 t}$이다.

먼저 편미분방정식(1)을 space step이 $h = \Delta x = \Delta y$인 uniform grid에서 편차분 방정식으로 근사시키자.

$$\frac{u_{ij}^{n+1} - u_{ij}^{n}}{\Delta t} = \frac{u_{i-1,j}^{n+1} - 2u_{ij}^{n+1} + u_{i+1,j}^{n+1}}{h^2} + \frac{u_{i,j-1}^{n+1} - 2u_{ij}^{n+1} + u_{i,j+1}^{n+1}}{h^2}. \tag{2}$$

이제 2D 열방정식(2)에 함축적(implicit) splitting scheme을 적용하여 정리하면 다음과 같다.

$$\frac{\tilde{u}_{ij} - u_{ij}^{n}}{\Delta t} = \frac{\tilde{u}_{i+1,j} - 2\tilde{u}_{ij} + \tilde{u}_{i-1,j}}{h^2}, \tag{3}$$

$$\frac{u_{ij}^{n+1} - \tilde{u}_{ij}}{\Delta t} = \frac{u_{i,j+1}^{n+1} - 2u_{ij}^{n+1} + u_{i,j-1}^{n+1}}{h^2}. \tag{4}$$

경계조건은 linear boundary condition을 이용하도록 하자.

Linear boundary condition은 다음과 같이 정의된다.

$$\frac{\partial^2 u}{\partial x^2}(0,y,t) = \frac{\partial^2 u}{\partial x^2}(1,y,t) = 0, \qquad \frac{\partial^2 u}{\partial y^2}(x,0,t) = \frac{\partial^2 u}{\partial y^2}(x,1,t) = 0, \qquad \forall t \in [0,T].$$

즉,

$$\frac{u_{0,j} - 2u_{1,j} + u_{2,j}}{\Delta x^2} = 0, \qquad \frac{u_{N_x-1,j} - 2u_{N_x,j} + u_{N_x+1,j}}{\Delta x^2} = 0,$$
$$\frac{u_{i,0} - 2u_{i,1} + u_{i,2}}{\Delta y^2} = 0, \qquad \frac{u_{i,N_y-1} - 2u_{i,N_y} + u_{i,N_y+1}}{\Delta y^2} = 0. \tag{5}$$

이제 식(3)에 위의 경계조건 (5)을 적용하기 위해

$$\tilde{u}_{0,j} = 2\tilde{u}_{1,j} - \tilde{u}_{2,j}, \qquad \tilde{u}_{N_x+1,j} = 2\tilde{u}_{N_x,j} - \tilde{u}_{N_x-1,j},$$
$$\tilde{u}_{i,0} = 2\tilde{u}_{i,1} - \tilde{u}_{i,2}, \qquad \tilde{u}_{i,N_y+1} = 2\tilde{u}_{i,N_y} - \tilde{u}_{i,N_y-1}$$

위의 식을 대입하면 다음과 같이 정리할 수 있다. (계산의 편의를 위해 $\alpha = \dfrac{\Delta t}{h^2}$라고 하자.)

$$\begin{aligned} &\tilde{u}_{1,j} = u^n_{1,j}, \qquad \tilde{u}_{Nx,j} = u^n_{Nx,j}, \\ &-\alpha\tilde{u}_{i+1,j} + (1+2\alpha)\tilde{u}_{ij} - \alpha\tilde{u}_{i-1,j} = u^n_{ij}, \\ &\qquad\qquad \text{for} \;\; i = 2, \cdots, N_x - 1, \;\; j = 1, \cdots, N_y \end{aligned} \tag{6}$$

마찬가지로, 식(4)에 위의 경계조건 (5)을 적용하기 위해

$$\begin{aligned} u^{n+1}_{0,j} &= 2u^{n+1}_{1,j} - u^{n+1}_{2,j}, \qquad u^{n+1}_{N_x+1,j} = 2u^{n+1}_{N_x,j} - u^{n+1}_{N_x-1,j}, \\ u^{n+1}_{i,0} &= 2u^{n+1}_{i,1} - u^{n+1}_{i,2}, \qquad u^{n+1}_{i,N_y+1} = 2u^{n+1}_{i,N_y} - u^{n+1}_{i,N_y-1} \end{aligned}$$

위의 식을 대입하면 다음과 같이 정리할 수 있다. (계산의 편의를 위해 $\alpha = \dfrac{\Delta t}{h^2}$라고 하자.)

$$\begin{aligned} &u^{n+1}_{i,1} = \tilde{u}_{i,1}, \qquad u^{n+1}_{i,Ny} = \tilde{u}_{i,Ny}, \\ &-\alpha u^{n+1}_{i,j+1} + (1+2\alpha)u^{n+1}_{ij} - \alpha u^{n+1}_{i,j-1} = \tilde{u}_{ij}, \\ &\qquad\qquad \text{for} \;\; i = 1, \cdots, N_x, \;\; j = 2, \cdots, N_y - 1 \end{aligned} \tag{7}$$

여기서 N_x와 N_y는 x와 y의 내부점의 개수이다.

1st step

먼저 식 (6)은 $x-$방향에서 함축적이므로 고정된 j에 대하여 다음과 같이 나타낼 수 있다.

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \dots & 0 \\ -r_x & 1+2r_x & -r_x & 0 & 0 & \dots & 0 \\ 0 & -r_x & 1+2r_x & -r_x & 0 & \dots & 0 \\ 0 & 0 & -r_x & 1+2r_x & -r_x & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & -r_x & 1+2r_x & -r_x \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{A}_x} \underbrace{\begin{bmatrix} \tilde{u}_{1,j} \\ \tilde{u}_{2,j} \\ \tilde{u}_{3,j} \\ \tilde{u}_{4,j} \\ \vdots \\ \tilde{u}_{Nx-1,j} \\ \tilde{u}_{Nx,j} \end{bmatrix}}_{\tilde{\mathbf{U}}} = \underbrace{\begin{bmatrix} u^n_{1,j} \\ u^n_{2,j} \\ u^n_{3,j} \\ u^n_{4,j} \\ \vdots \\ u^n_{Nx-1,j} \\ u^n_{Nx,j} \end{bmatrix}}_{\mathbf{b}_x}$$

즉,

$$\mathbf{A}_x\tilde{\mathbf{U}}(:,j) = \mathbf{b}_x \tag{8}$$

여기서 행렬 $\mathbf{A}_x$는 tridiagonal이며, $\mathbf{b}_x$는 N_x차 벡터이다. 따라서 식(8)은 tridiagonal system을 풀어서 벡터 $\tilde{\mathbf{U}}(:,j)$를 얻을 수 있다.

2nd step

이제 식 (7)은 위의 과정과 동일하게 풀면된다. 마찬가지로 식 (7)은 $y-$방향에서 함축적이므로 고정된 j에 대하여 다음과 같이 나타낼 수 있다.

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \dots & 0 \\ -r_y & 1+2r_y & -r_y & 0 & 0 & \dots & 0 \\ 0 & -r_y & 1+2r_y & -r_y & 0 & \dots & 0 \\ 0 & 0 & -r_y & 1+2r_y & -r_y & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & -r_y & 1+2r_y & -r_y \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{A}_y} \underbrace{\begin{bmatrix} u_{i,1}^{n+1} \\ u_{i,2}^{n+1} \\ u_{i,3}^{n+1} \\ u_{i,4}^{n+1} \\ \vdots \\ u_{i,Ny-1}^{n+1} \\ u_{i,Ny}^{n+1} \end{bmatrix}}_{\mathbf{U^{n+1}}} = \underbrace{\begin{bmatrix} \tilde{u}_{i,1} \\ \tilde{u}_{i,2} \\ \tilde{u}_{i,3} \\ \tilde{u}_{i,4} \\ \vdots \\ \tilde{u}_{i,Ny-1} \\ \tilde{u}_{i,Ny} \end{bmatrix}}_{\mathbf{b}_y}$$

즉,

$$\mathbf{A}_y \mathbf{U^{n+1}}(i,:) = \mathbf{b}_y \tag{9}$$

여기서 행렬 $\mathbf{A}_y$는 tridiagonal이며, $\mathbf{b}_y$는 N_y차 벡터이다. 따라서 식(9)은 tridiagonal system을 풀어서 벡터 $\mathbf{U^{n+1}}(i,:)$를 얻을 수 있다.

다음은 $Nx = Ny = 50$일 때, 시간에 따른 열방정식의 해를 나타낸 MATLAB 코드이다.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% OSM_heat2d.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear; clc; clf;

xleft = 0.0; xright = 1.0;
yleft = 0.0; yright = 1.0;
Nx = 50;  Ny = 50;
dx = (xright - xleft)/Nx;
%dy = (yright - yleft)/Ny;
h = dx;
T=0.1; Nt=100; dt=T/Nt;

alpha = dt/h^2;
```

```
13
14  % initalization
15  Ax(1:Nx, 1:Nx) = 0.0;
16  Ay(1:Ny, 1:Ny) = 0.0;
17  bx(1:Nx) = 0.0;
18  by(1:Ny) = 0.0;
19  u(1:Nx+2,1:Ny+2) = 0.0;
20  old_u = u;
21
22  % x and y points (cell centered grid)
23  for i = 1:Nx+2
24      x(i) = (i-1.5)*h;
25  end
26  for j = 1:Ny+2
27      y(j) = (j-1.5)*h;
28  end
29
30  %%% initial condition %%%
31  for i = 2:Nx+1
32      for j = 2:Ny+1
33          u(i,j) = sin(pi*x(i))*sin(pi*y(j));
34      end
35  end
36
37  % linear boundary condition
38  u(1,:) = 2.0*u(2,:) - u(3,:); u(Nx+2,:) = 2.0*u(Nx+1,:) - u(Nx,:);
39  u(:,1) = 2.0*u(:,2) - u(:,3); u(:,Ny+2) = 2.0*u(:,Ny+1) - u(:,Ny);
40
41  % draw mesh (initial u)
42  figure(1);
43  mesh(u); grid on; axis tight
44  xlabel('X','FontSize',14); ylabel('Y','FontSize',14);
45  zlabel('u(x,y,0)','FontSize',14);
46  title('2D Heat equation(initial condtion)','FontSize',16);
47  hold on
```

```
48
49  %%% update old_u %%%
50  old_u = u;
51
52  %%% analytic solution (T=0.1) %%%
53  exact_u = u;
54  for i = 1:Nx+2
55      for j = 1:Ny+2
56          exact_u(i,j) = sin(pi*x(i))*sin(pi*y(j))*exp(-2.0*pi^2*T);
57      end
58  end
59
60  % draw mesh (exact u)
61  figure(2);
62  mesh(exact_u); grid on; axis tight
63  xlabel('X','FontSize',14); ylabel('Y','FontSize',14);
64  zlabel('exact u(x,y,0.1)','FontSize',14);
65  title('2D Heat equation(exact solution)','FontSize',16);
66
67
68  %%% Operator Splitting Method %%%
69  for k = 1:Nt
70
71      %%% 1st step (x-direction) %%%
72
73      % make Ax matrix
74      for i = 1:Nx
75          Ax(i,i) = 1 + 2.0*alpha;
76      end
77      for i = 2:Nx
78          Ax(i-1,i) = -alpha;
79          Ax(i,i-1) = -alpha;
80      end
81      % applying linear boundary condition
82      Ax(1,1) = 1.0; Ax(Nx,Nx) = 1.0;
```

```
 83      Ax(1,2) = 0.0; Ax(Nx,Nx-1) = 0.0;
 84
 85      % make bx matrix
 86      for j = 2:Ny+1
 87          for i = 2:Nx+1
 88              bx(i-1) = old_u(i,j);
 89          end
 90          % Solve Ax*U=bx
 91          u(2:Nx+1,j) = inv(Ax)*bx';
 92      end
 93
 94      % linear boundary condition
 95      u(1,:) = 2.0*u(2,:) - u(3,:); u(Nx+2,:) = 2.0*u(Nx+1,:) - u(Nx,:);
 96      u(:,1) = 2.0*u(:,2) - u(:,3); u(:,Ny+2) = 2.0*u(:,Ny+1) - u(:,Ny);
 97
 98      %%% update old_u %%%
 99      old_u = u;
100
101
102      %%% 2nd step (y-direction) %%%
103      % make Ay matrix
104      for j = 1:Ny
105          Ay(j,j) = 1 + 2.0*alpha;
106      end
107      for j = 2:Ny
108          Ay(j-1,j) = -alpha;
109          Ay(j,j-1) = -alpha;
110      end
111      % applying linear boundary condition
112      Ay(1,1) = 1.0; Ay(Ny,Ny) = 1.0;
113      Ay(1,2) = 0.0; Ay(Ny,Ny-1) = 0.0;
114
115      % make by matrix
116      for i = 2:Nx+1
117          for j = 2:Ny+1
```

```
118             by(j-1) = old_u(i,j);
119         end
120         % Solve Ay*U=by
121         u(i,2:Ny+1) = inv(Ay)*by';
122     end
123
124     % linear boundary condition
125     u(1,:) = 2.0*u(2,:) - u(3,:); u(Nx+2,:) = 2.0*u(Nx+1,:) - u(Nx,:);
126     u(:,1) = 2.0*u(:,2) - u(:,3); u(:,Ny+2) = 2.0*u(:,Ny+1) - u(:,Ny);
127
128     %%% update old_u %%%
129     old_u = u;
130
131 end
132
133 % draw mesh (numerical u)
134 figure(3);
135 mesh(u); grid on; axis tight
136 xlabel('X','FontSize',14); ylabel('Y','FontSize',14);
137 zlabel('Numerical u(x,y,0.1)','FontSize',14);
138 title('2D Heat equation(numerical solution)','FontSize',16);
139
140 % draw mesh (error)
141 figure(4);
142 mesh( abs(u-exact_u) ); grid on; axis tight
143 xlabel('X','FontSize',14); ylabel('Y','FontSize',14);
144 zlabel('error','FontSize',14);
145 title('2D Heat equation(numerical-exact)','FontSize',16);
146
147 % maximum error
148 sprintf('Max_error = %18.16f', max(max(abs(u-exact_u))))
149 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```