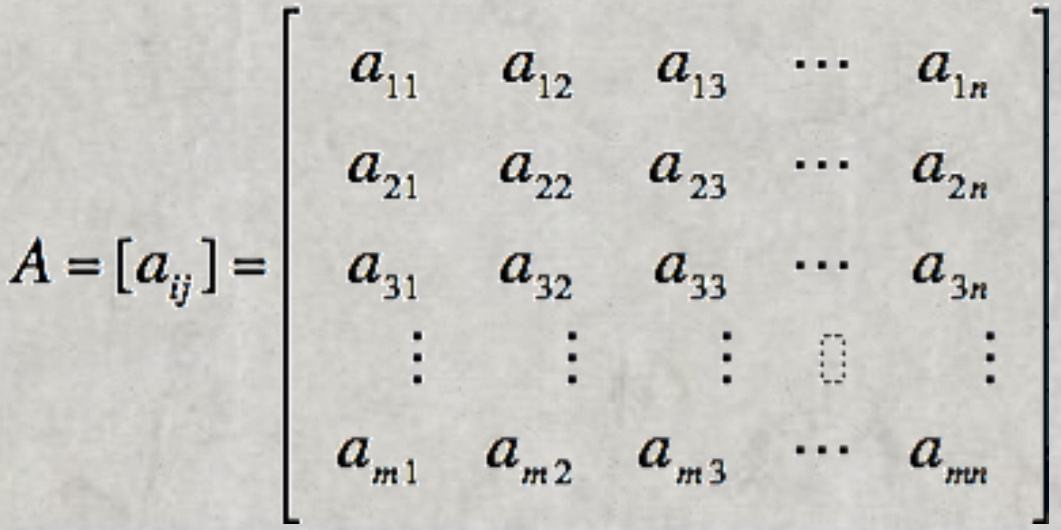


$$A = [a_{ij}] = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \square & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}$$

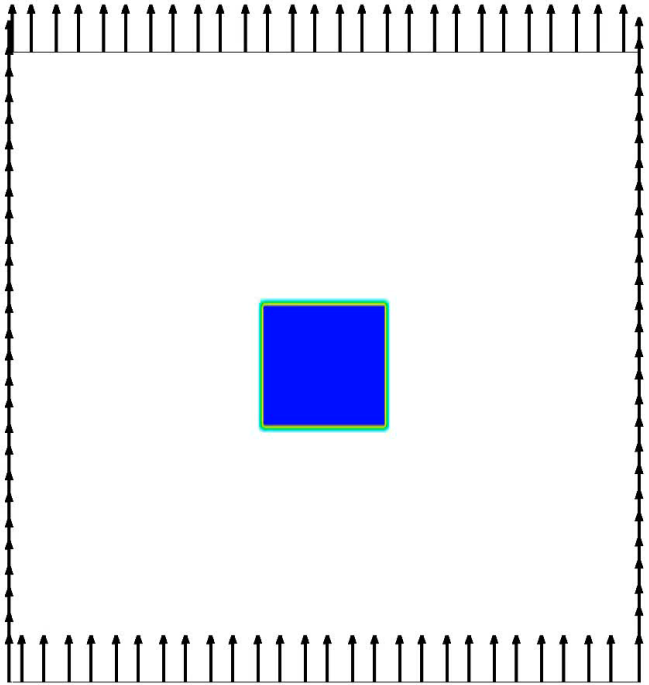
$$\begin{aligned} A^2XBA &= AB \\ A^{-2}A^2XBA &= A^{-2}AB \\ XBA &= A^{-1}B \\ XBAA^{-1} &= A^{-1}BA^{-1} \\ XB &= A^{-1}BA^{-1} \\ XBB^{-1} &= A^{-1}BA^{-1}B^{-1} \\ X &= A^{-1}BA^{-1}B^{-1} \end{aligned}$$

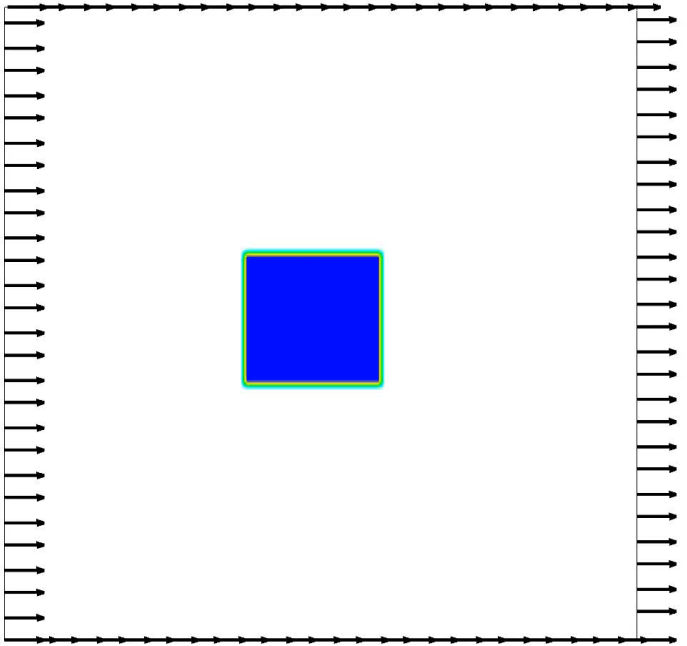
Stabilization of Gauss-Seidel Smoother

NAOFE, Dong-A university

Yeoun Joo KIM

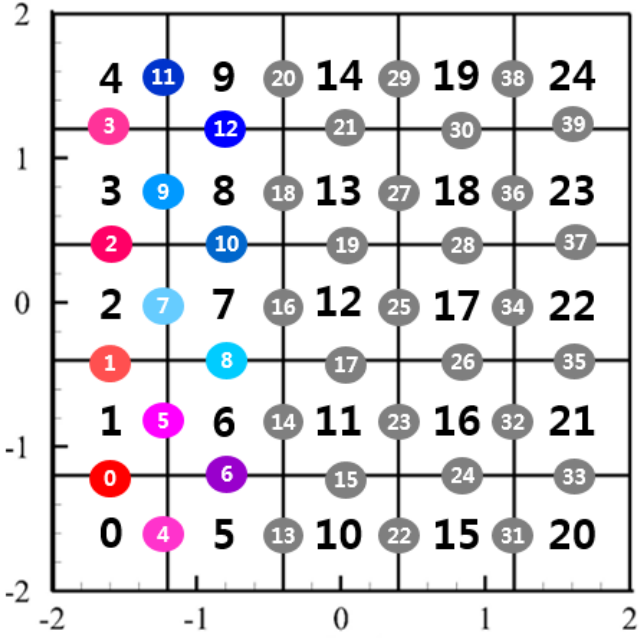
Motivation



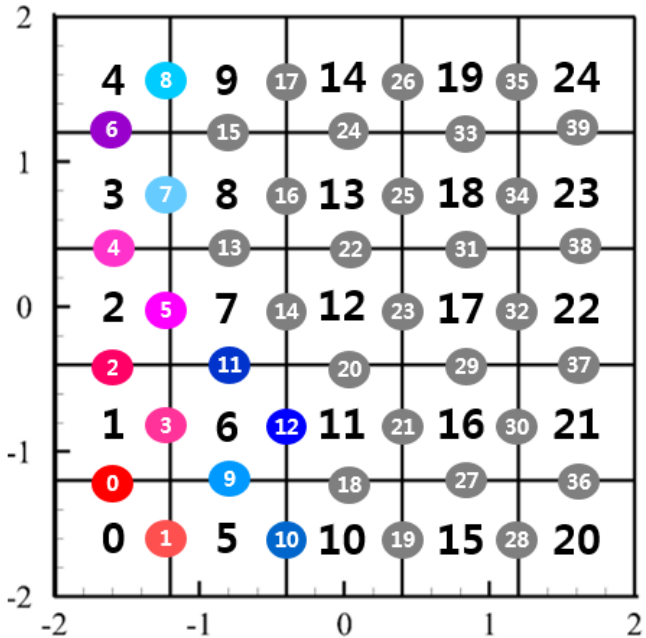


- x방향으로 흐르지 못하는 이유는?

Cases of Face-Ordering



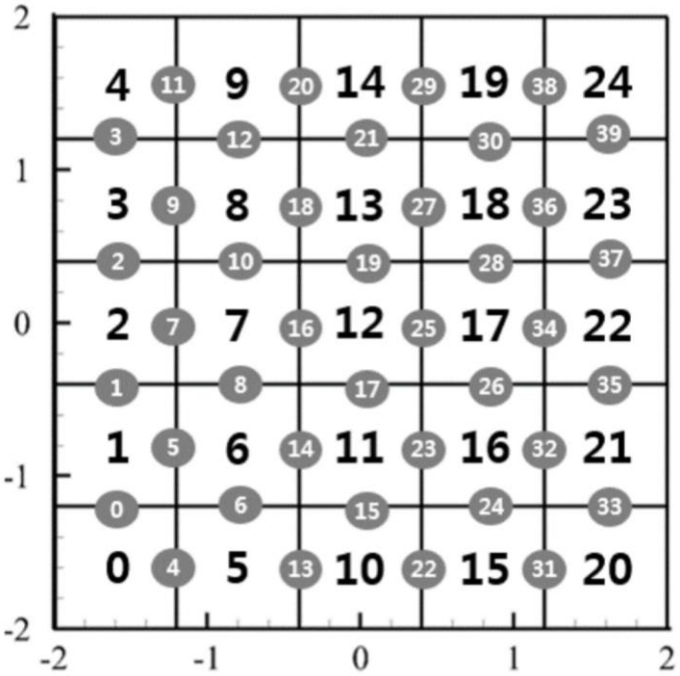
Grid 1



Grid 2

Different ordering of face index in two grid systems

GS Smoother : Grid 1




```
for (label celli=0; celli<nCells; celli++)
{
    // Start and end of this row
    fStart = fEnd;
    fEnd = ownStartPtr[celli + 1];

    // Get the accumulated neighbour side
    psii = bPrimePtr[celli];

    // Accumulate the owner product side
    for (label facei=fStart; facei<fEnd; facei++)
    {
        psii -= upperPtr[facei]*psiPtr[uPtr[facei]];
    }
```

$$\varphi_{i,j} = l_{i-1}\varphi_{i-1,j} + u_{i+1}\varphi_{i+1,j}$$ missed

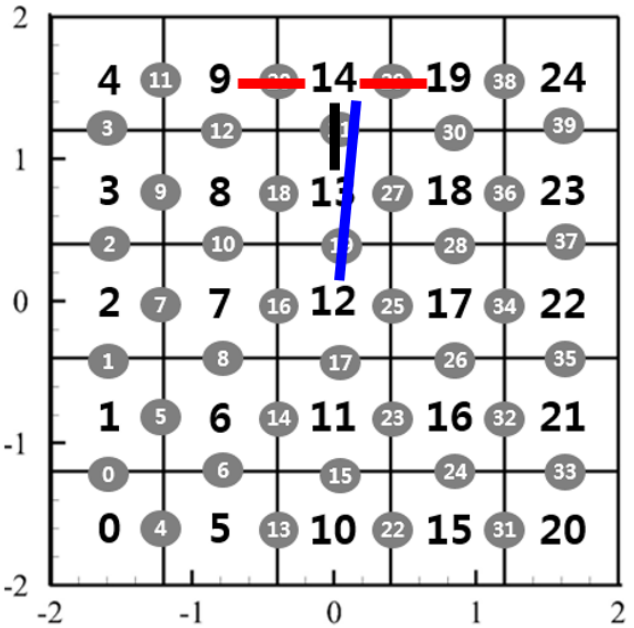
$$+ l_{j-1}\varphi_{i,j-1} + u_{j+1}\varphi_{i,j+1}$$

$$+ d_{k,l}\varphi_{k,l}$$ misplaced

$$\begin{cases} k \neq i-1, i+1 \\ l \neq j-1, j+1 \end{cases}$$

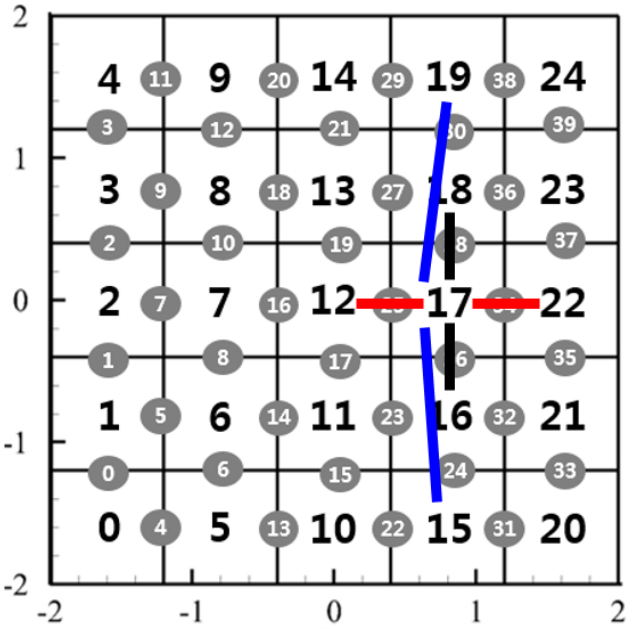
GS Smoother : Grid 1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24			
0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
3	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
4	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
5	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5
6	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6
7	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
8	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8
9	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9
10	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
11	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11
12	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0.997506	3.2	12
13	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	13
14	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0.0024876	0	14
15	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	15
16	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	16
17	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	17
18	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	18
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	19
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	0	0	0	0	20
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	0	0	0	21
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	0	0	22
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	0	23
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	24



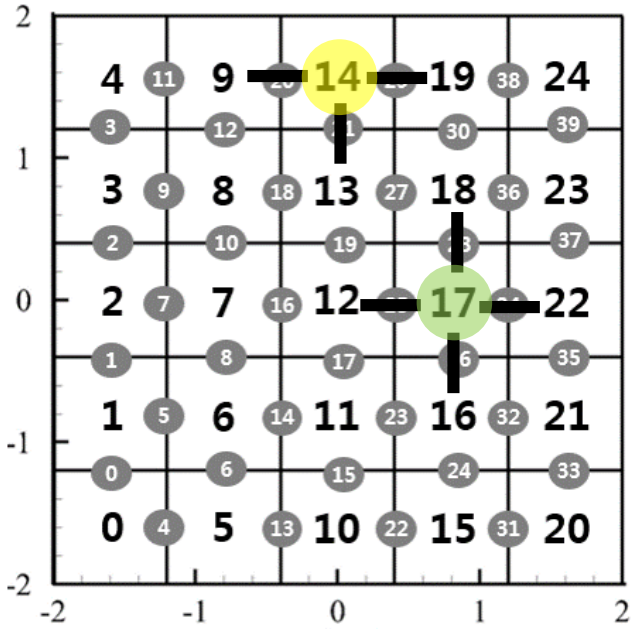
GS Smoother : Grid 1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24			
0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
3	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
4	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
5	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5
6	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6
7	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
8	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8
9	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9
10	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
11	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11
12	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0.997506	3.2	12
13	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	13
14	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0.0024876	0	14
15	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	15
16	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	16
17	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	17
18	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	18
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	19
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	0	0	0	0	20
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	0	0	0	21
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	0	0	22
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	0	23
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	24

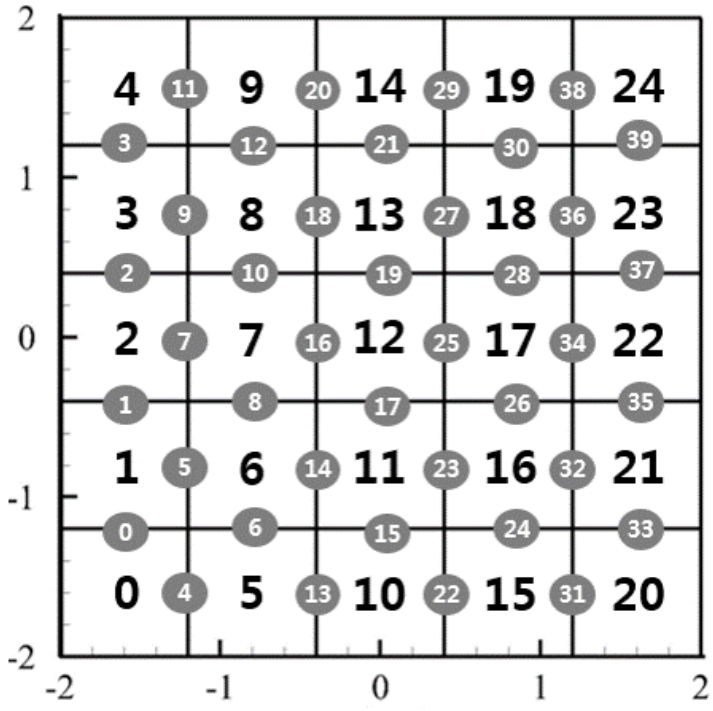


GS Smoother : Grid 1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24			
0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
3	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
4	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
5	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5
6	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6
7	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
8	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8
9	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9
10	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
11	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11
12	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0.9975062	3.2	12
13	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	13
14	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	14
15	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	15
16	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	16
17	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0.0024875	0	17
18	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	18
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	19
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	0	0	0	0	20
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	0	0	0	21
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	6.219E-06	0	22
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	0	23
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	24



DILU Smoother : Grid 1




```
for (label face=0; face<nFaces; face++)
{
    rAPtr[uPtr[face]] -= lowerPtr[face]*psiPtr[lPtr[face]];
    rAPtr[lPtr[face]] -= upperPtr[face]*psiPtr[uPtr[face]];
}

rA *= rD_;

label nFaces = matrix_.upper().size();
for (label face=0; face<nFaces; face++)
{
    label u = uPtr[face];
    rAPtr[u] -= rDPtr[u]*lowerPtr[face]*rAPtr[lPtr[face]];
}
```

```
label nFacesM1 = nFaces - 1;
for (label face=nFacesM1; face>=0; face--)
{
    label l = lPtr[face];
    rAPtr[l] -= rDPtr[l]*upperPtr[face]*rAPtr[uPtr[face]];
}

psi += rA;
```

$$Ax_i = B_i$$
$$A(x_i - x_{i-1}) = B_i - B_{i-1}$$
$$(D + L + U)(x_i - x_{i-1}) = B_i - B_{i-1}$$

$\|D(x_i - x_{i-1})\| \gg \|L(x_i x - x_{i-1})\|$
$\|D(x_i - x_{i-1})\| \gg \|U(x_i - x_{i-1})\|$

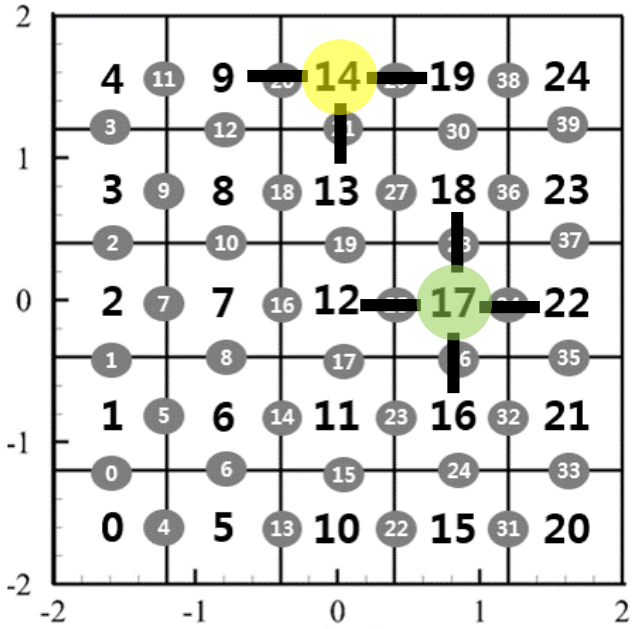
$$x_i - x_{i-1} \cong D^{-1}(B_i - B_{i-1})$$
$$x_i - x_{i-1} + D^{-1}L(x_i - x_{i-1}) + D^{-1}U(x_i - x_{i-1}) = D^{-1}(B_i - B_{i-1})$$
$$x_i \cong x_{i-1} + D^{-1}(B_i - B_{i-1}) - D^{-1}LD^{-1}(B_i - B_{i-1}) - D^{-1}UD^{-1}(B_i - B_{i-1})$$
$$rA = D^{-1}(B_i - B_{i-1})$$
$$x_i \cong x_{i-1} + rA - D^{-1}LrA - D^{-1}UrA$$

```
rAPtr[u] -= rDPtr[u]*lowerPtr[face]*rAPtr[lPtr[face]];
```

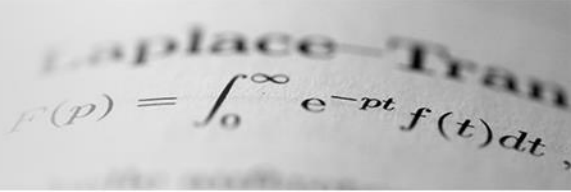
```
rAPtr[l] -= rDPtr[l]*upperPtr[face]*rAPtr[uPtr[face]];
```

DILU Smoother : Grid 1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24			
0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
3	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
4	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
5	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5
6	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6
7	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
8	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8
9	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9
10	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
11	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11
12	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0.9975062	3.2	12
13	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	13
14	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	14
15	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	15
16	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	16
17	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0.0024875	0	17
18	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	18
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	19
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	0	0	0	0	20
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	0	0	0	21
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	6.219E-06	0	22
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	0	23
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	24



Other Smoothers



DILUGaussSeidel solver

```
diluSmoother_.smooth(psi, source, cmpt, nSweeps);
gsSmoother_.smooth(psi, source, cmpt, nSweeps);
```

symGaussSeidel solver

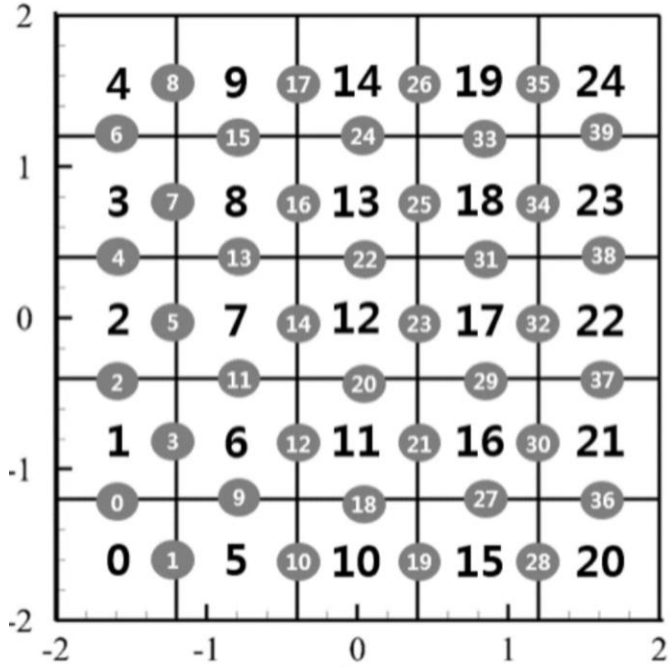
```
for (label celli=0; celli<nCells; celli++)
{
    // Start and end of this row
    fStart = fEnd;
    fEnd = ownStartPtr[celli + 1];

    // Get the accumulated neighbour side
    psii = bPrimePtr[celli];

    // Accumulate the owner product side
    for (label facei=fStart; facei<fEnd; facei++)
    {
        psii -= upperPtr[facei]*psiPtr[uPtr[facei]];
    }

    // Finish current psi
    psii /= diagPtr[celli];

    // Distribute the neighbour side using current psi
    for (label facei=fStart; facei<fEnd; facei++)
    {
        bPrimePtr[uPtr[facei]] -= lowerPtr[facei]*psii;
    }

    psiPtr[celli] = psii;
}

fStart = ownStartPtr[nCells];

for (label celli=nCells-1; celli>=0; celli--)
{
    // Start and end of this row
    fEnd = fStart;
    fStart = ownStartPtr[celli];

    // Get the accumulated neighbour side
    psii = bPrimePtr[celli];

    // Accumulate the owner product side
    for (label facei=fStart; facei<fEnd; facei++)
    {
        psii -= upperPtr[facei]*psiPtr[uPtr[facei]];
    }

    // Finish psi for this cell
    psii /= diagPtr[celli];

    // Distribute the neighbour side using psi for this cell
    for (label facei=fStart; facei<fEnd; facei++)
    {
        bPrimePtr[uPtr[facei]] -= lowerPtr[facei]*psii;
    }

    psiPtr[celli] = psii;
}
```

nonBlockingGaussSeidel solver

```
for (label celli=0; celli<blockStart; celli++)
{
    // Start and end of this row
    fStart = fEnd;
    fEnd = ownStartPtr[celli + 1];

    // Get the accumulated neighbour side
    curPsi = bPrimePtr[celli];

    // Accumulate the owner product side
    for (label curFace=fStart; curFace<fEnd; curFace++)
    {
        curPsi -= upperPtr[curFace]*psiPtr[uPtr[curFace]];
    }

    // Finish current psi
    curPsi /= diagPtr[celli];

    // Distribute the neighbour side using current psi
    for (label curFace=fStart; curFace<fEnd; curFace++)
    {
        bPrimePtr[uPtr[curFace]] -= lowerPtr[curFace]*curPsi;
    }

    psiPtr[celli] = curPsi;
}

for (label celli=blockStart; celli < nCells; celli++)
{
    // Start and end of this row
    fStart = fEnd;
    fEnd = ownStartPtr[celli + 1];

    // Get the accumulated neighbour side
    curPsi = bPrimePtr[celli];

    // Accumulate the owner product side
    for (label curFace=fStart; curFace<fEnd; curFace++)
    {
        curPsi -= upperPtr[curFace]*psiPtr[uPtr[curFace]];
    }

    // Finish current psi
    curPsi /= diagPtr[celli];

    // Distribute the neighbour side using current psi
    for (label curFace=fStart; curFace<fEnd; curFace++)
    {
        bPrimePtr[uPtr[curFace]] -= lowerPtr[curFace]*curPsi;
    }

    psiPtr[celli] = curPsi;
}
```

- Based on GaussSeidel solver

GS Smoother : Grid 2



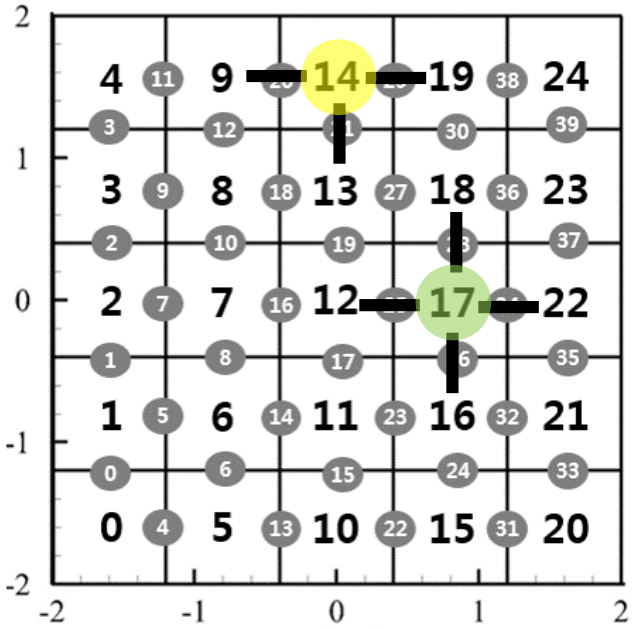

```
for (label celli=0; celli<nCells; celli++)
{
    // Start and end of this row
    fStart = fEnd;
    fEnd = ownStartPtr[celli + 1];

    // Get the accumulated neighbour side
    psii = bPrimePtr[celli];

    // Accumulate the owner product side
    for (label facei=fStart; facei<fEnd; facei++)
    {
        psii -= upperPtr[facei]*psiPtr[uPtr[facei]];
    }
```

$$\varphi_{i,j} = l_{i-1}\varphi_{i-1,j} + u_{i+1}\varphi_{i+1,j}$$

$$+ l_{j-1}\varphi_{i,j-1} + u_{j+1}\varphi_{i,j+1}$$

GS Smoother : Grid 2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24			
0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
3	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
4	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
5	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5
6	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6
7	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
8	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8
9	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9
10	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
11	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11
12	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0.9975062	3.2	12
13	0	0	0	0	0	0	0	0	-0.008	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13
14	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	14
15	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	15
16	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	16
17	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0.0024875	0	17
18	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	18
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	19
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	0	0	0	0	20
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	0	0	0	21
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	6.219E-06	0	22
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	0	23
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	24



Code Analysis of GS

```
scalar psii;
label fStart;
label fEnd = ownStartPtr[0];

for (label celli=0; celli<nCells; celli++)
{
    // Start and end of this row
    fStart = fEnd;
    fEnd = ownStartPtr[celli + 1];

    // Get the accumulated neighbour side
    psii = bPrimePtr[celli];

    // Accumulate the owner product side
    for (label facei=fStart; facei<fEnd; facei++)
    {
        psii -= upperPtr[facei]*psiPtr[uPtr[facei]];
    }

    // Finish psi for this cell
    psii /= diagPtr[celli];

    // Distribute the neighbour side using psi for this cell
    for (label facei=fStart; facei<fEnd; facei++)
    {
        bPrimePtr[uPtr[facei]] -= lowerPtr[facei]*psii;
    }

    psiPtr[celli] = psii;
}
```

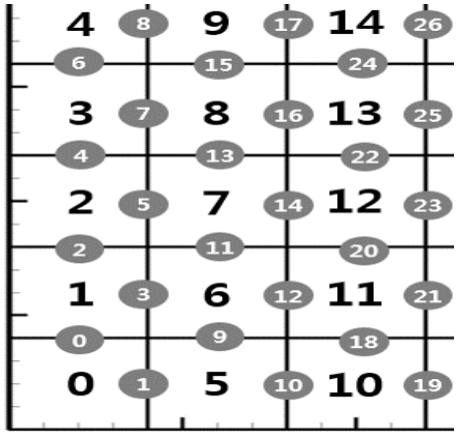
start owner face at each cell

start owner face at present cell

start owner face at next cell

psi at cell of owner faces

psi at cell of neighbour faces



cellIndex	faceIndex	
	fStart/ownStart	fEnd
0	0	2
1	2	4
2	4	6
3	6	8
4	8	9
5	9	11
6	11	13

Code Analysis of GS

```
scalar psii;
label fStart;
label fEnd = ownStartPtr[0];

for (label celli=0; celli<nCells; celli++)
{
    // Start and end of this row
    fStart = fEnd;
    fEnd = ownStartPtr[celli + 1];

    // Get the accumulated neighbour side
    psii = bPrimePtr[celli];

    // Accumulate the owner product side
    for (label facei=fStart; facei<fEnd; facei++)
    {
        psii -= upperPtr[facei]*psiPtr[uPtr[facei]];
    }

    // Finish psi for this cell
    psii /= diagPtr[celli];

    // Distribute the neighbour side using psi for this cell
    for (label facei=fStart; facei<fEnd; facei++)
    {
        bPrimePtr[uPtr[facei]] -= lowerPtr[facei]*psii;
    }

    psiPtr[celli] = psii;
}
```

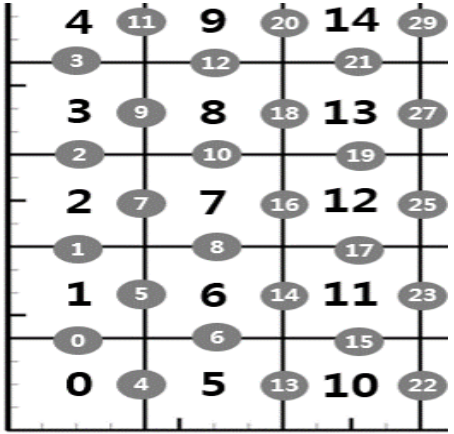
start owner face at each cell

start owner face at present cell

start owner face at next cell

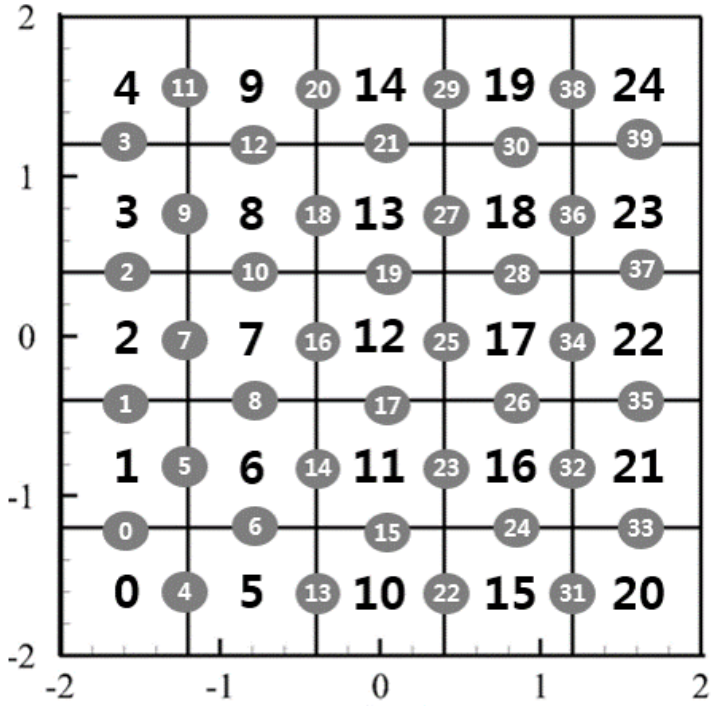
psi at cell of owner faces

psi at cell of neighbour faces



cellIndex	faceIndex	
	fStart/ownStart	fEnd
0	0	1
1	1	2
2	2	3
3	3	6
4	6	6
5	6	8
6	8	10

Stabilization of GS

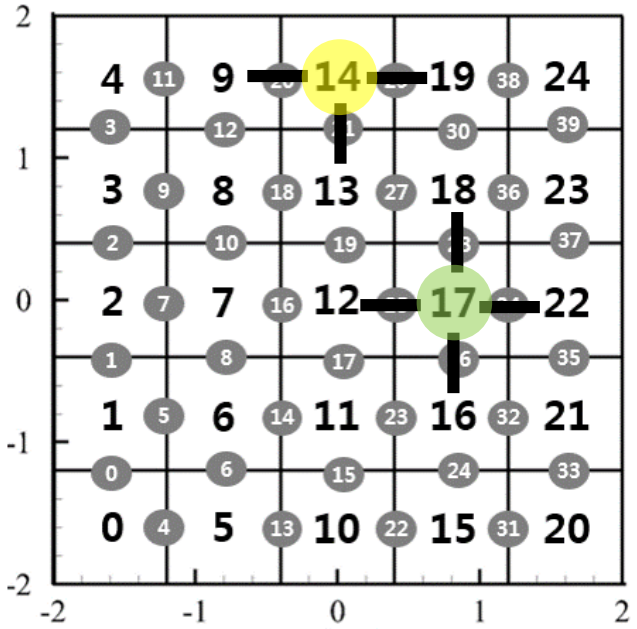



```
for(label celli=0;celli<nCells;celli++) psiPtr[celli] = bPrimePtr[celli]/diagPtr[celli];
for(label facei=0;facei<nFaces;facei++)
{
    psiPtr[lPtr[facei]] -= upperPtr[facei]*psiPtr[uPtr[facei]]/diagPtr[lPtr[facei]];
    psiPtr[uPtr[facei]] -= lowerPtr[facei]*psiPtr[lPtr[facei]]/diagPtr[uPtr[facei]];
}
```

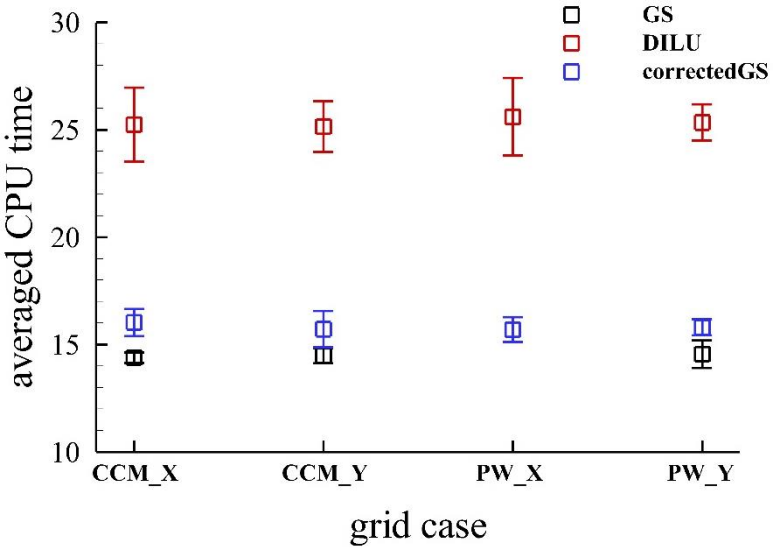
$$\varphi_{i,j} = (l_{i-1}\varphi_{i-1,j} + u_{i+1}\varphi_{i+1,j} + l_{j-1}\varphi_{i,j-1} + u_{j+1}\varphi_{i,j+1})/D_i$$

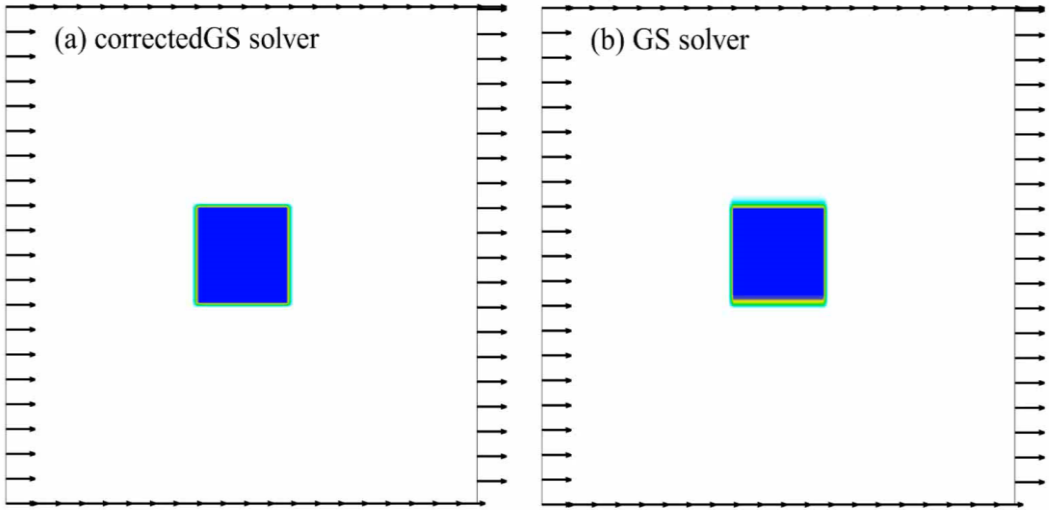
Stabilization of GS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24			
0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
3	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
4	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
5	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5
6	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6
7	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
8	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8
9	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9
10	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10
11	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11
12	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0.9975062	3.2	12
13	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	0	13
14	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	0	14
15	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	0	15
16	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	0	0	16
17	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0.0024875	0	17
18	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	0	18
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.208	0	0	0	0	0	0	0	19
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	0	0	0	0	20
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	0	0	0	21
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	6.219E-06	0	22
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	0	23
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.008	0	0	0	0	3.2	0	0	24



Performance of correctedGS





- GS solver
efficient but unstable depending on face-ordering
- DILU solver
stable but inefficient
- correctedGS solver
efficient and stable