# Notes on Domain Decomposition

Scott Aiton

October 3, 2017

## 1 Forming Schur Complement Matrix

If there is a $\gamma$ for each point on the interface, how to we determine the values?

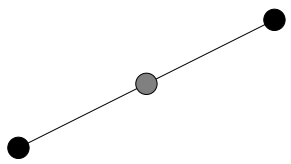


Figure 1: figure with l and r

We want the flux going out of ouf the $l$ cell to match the flux going into the $r$ cell. So we get the equation:

$$\gamma - l = r - \gamma \tag{1}$$

or:

$$2\gamma - (l + r) = 0 \tag{2}$$

Note that $l$ and $r$ are not constants, but are dependent on the value of $\gamma$. So, let's change them to the be functions of $\gamma$:

$$2\gamma - (L(\gamma) + R(\gamma)) = 0 \tag{3}$$

So now we have a function of $\gamma$:

$$F(\gamma) = 2\gamma - (L(\gamma) + R(\gamma)) \tag{4}$$

Therefore we have a function that we want to find the zero of this function:

$$F(\gamma) = 0 \tag{5}$$

If we have a single $\gamma$ value that we are solving for, this function becomes a linear equation of the form:

$$F(\gamma) = a\gamma - b \tag{6}$$

we can find the coefficients by:

$$b = -F(0) \tag{7}$$
$$a = F(1) + b \tag{8}$$

With these coefficients, we can solve for $\gamma$

$$F(\gamma) = 0 \tag{9}$$
$$a\gamma - b = 0 \tag{10}$$
$$a\gamma = b \tag{11}$$
$$\gamma = \frac{b}{a} \tag{12}$$

When there are multiple interface points, then we have a system of linear equations:

$$F(\gamma) = A\gamma - b \tag{13}$$

the $b$ vector is found by

$$b = -F(0) \tag{14}$$

each column of the matrix is found by

$$A(:, i) = F(e_i) + b \tag{15}$$

we can then determine the $\gamma$ vector by solving

$$A\gamma = b \tag{16}$$

# 2 Quick Formation of the Schur Complement Matrix

In this section, we can use the process described in equation (15) to derive an algorithm that allow us to quickly form the Schur compliment matrix.

**The matrix does not depent on the RHS** First, let's reconsider equations (14) and (15). If we are solving on a system where the rhs and lhs on each patch is zero, the $b$ vector for the Schur complement matrix will be 0, since 0 is the correct solution for the interfaces. So equation 15 turns into

$$A(:, i) = F_{zero}(e_i) \tag{17}$$

where $F_{zero}$ is the same as equation 4 but with the rhs on each domain replaced with 0. So now we use can use this to reduce the amount of work needed to form the Schur complement

matrix.

Let's consider what happens when we solve for $F_{zero}(e_i)$. If a single interface value is set to 1, only the two adjacent domains will have non-zero dirichlet boundary conditions, meaning that only the two adjacent domains will have non-zero solutions. This means that when we are solving for $F_{zero}(e_i)$, we can assume that solution on any domain that is not adjacent to the interface with the 1 is zero. In other words, we only have to do a solve on the two adjacent domains, rather than solving for all the domains.

**Indexing of $\gamma$ Values** If the individual values in the $\gamma$ vector are indexed in the following way:

- If the interface is on the east or west side of the patch, the $\gamma$ values will be indexed consecutively from the bottom up.
- If the interface is on the north or south side of the patch, the $\gamma$ values will be indexed consecutively from left to right.

Since the $\gamma$ values on each interface indexed consecutively, the Schur complement matrix takes on a block structure, where each block is $n \times n$.

**Sparsity** Consider the example grid given in Figure 2. When a single 1 is set on the interface $i_{\text{main}}$, only 7 interfaces will end up having non-zero values: the main interface, $i_{\text{main}}$, and the 6 auxiliary interfaces, $i_{\text{left north}}$, $i_{\text{left south}}$, $i_{\text{left west}}$, $i_{\text{right north}}$, $i_{\text{right east}}$, and $i_{\text{right south}}$.

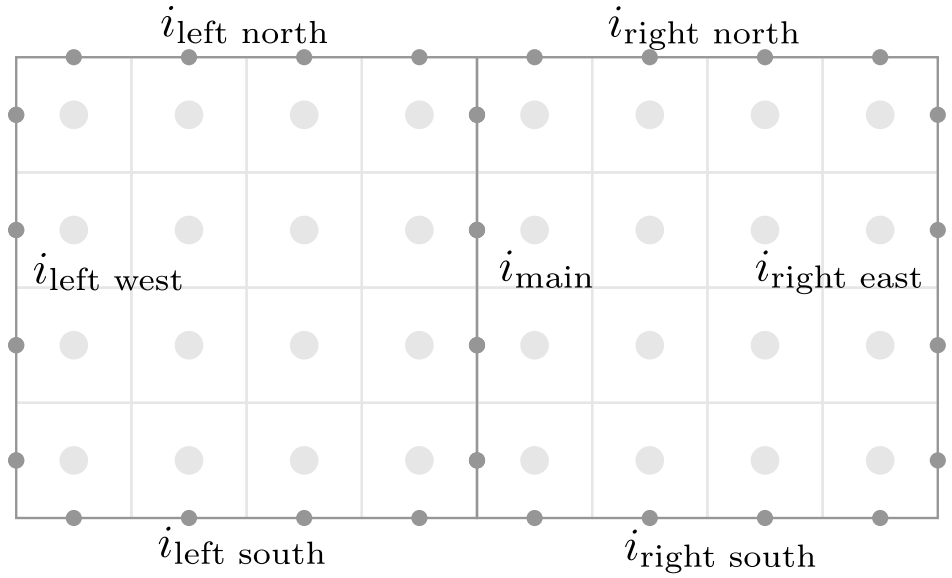


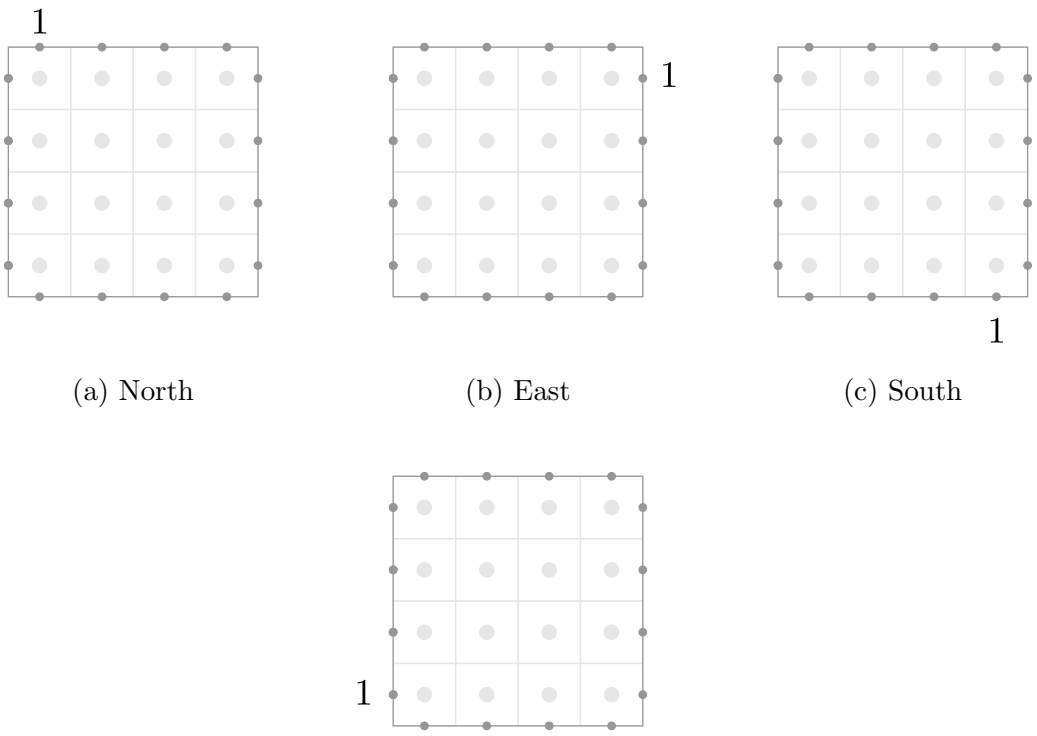

Figure 2: Two domain example

**Splitting up the work** If we look at equation 4, we can split up the work and process the two domains at different times.

$$F_{left}(\gamma) = \gamma - L(\gamma) \tag{18}$$

$$F_{right}(\gamma) = \gamma - R(\gamma) \tag{19}$$

So, when we process the left and right domains, we use equations 18 and 19, respectively, for the diagonal blocks ($i_{\text{main}}$). When we are forming the matrix, we will insert the diagonal block twice, summing the coefficients of one block into the other.




(a) North (b) East (c) South

(d) West

Figure 3: Rotation

**Rotation** Let the north side be our canonical side. Lets consider part a in figure 3 where the first $\gamma$ value on the north interface is set to 1. If rotate this domain clockwise by 90 degrees, that is equivalent to setting the last $\gamma$ value on the east interface to 1. If we rotate it again, it is equivalent to setting the last $\gamma$ value on the south interface to 1. And if we rotate once more, it would be equivalent to setting the first $\gamma$ value on the east interface.

This means that when we can get to coefficients for the blocks in the Schur complement matrix, by setting each $\gamma$ value along the north interface to 1, and then reuse these coefficients for the interfaces on the other sides.

**Algorithm**

```
1: procedure FormMatrix(Domains)
2:     tbp ← ∅                                   ▷ Set of interfaces to be processed
3:     for all Ω ∈ Domains do
4:         if Ω.northIndex ≠ null then
5:             iface()                           ▷ New iface object
6:             iface.side ← North
7:             iface.northIndex ← Ω.northIndex
8:             iface.eastIndex ← Ω.eastIndex
9:             iface.southIndex ← Ω.southIndex
10:            iface.westIndex ← Ω.westIndex
11:            tbp.insert(iface)
12:        end if
13:        if Ω.eastIndex ≠ null then
14:            iface()                           ▷ New iface object
15:            iface.side ← East
16:            iface.northIndex ← Ω.eastIndex
17:            iface.eastIndex ← Ω.southIndex
18:            iface.southIndex ← Ω.westIndex
19:            iface.westIndex ← Ω.northIndex
20:            tbp.insert(iface)
21:        end if
22:        if Ω.southIndex ≠ null then
23:            iface()                           ▷ New iface object
24:            iface.side ← South
25:            iface.northIndex ← Ω.southIndex
26:            iface.eastIndex ← Ω.westIndex
27:            iface.southIndex ← Ω.northIndex
28:            iface.westIndex ← Ω.eastIndex
29:            tbp.insert(iface)
30:        end if
31:        if Ω.westIndex ≠ null then
32:            iface()                           ▷ New iface object
33:            iface.side ← West
34:            iface.northIndex ← Ω.westIndex
35:            iface.eastIndex ← Ω.northIndex
36:            iface.southIndex ← Ω.eastIndex
37:            iface.westIndex ← Ω.southIndex
38:            tbp.insert(iface)
39:        end if
40:    end for
41:    Ω()
```

```
42:     Ω.rhs ← 0
43:     Ω.northBoundary ← 0
44:     Ω.eastBoundary ← 0
45:     Ω.southBoundary ← 0
46:     Ω.westBoundary ← 0
47:     NorthBlock(n ∗ n)                                  ▷ Allocate blocks of size n*n
48:     EastBlock(n ∗ n)
49:     SouthBlock(n ∗ n)
50:     WestBlock(n ∗ n)
51:     for i ← 1, n do
52:         Ω.northBoundary(i) ← 1
53:         Ω.solve()
54:         NorthBlock(:, i) ← Ω.northBoundary − Ω.lhs(:, n)
55:         EastBlock(:, i) ← −Ω.lhs(n, :)
56:         SouthBlock(:, i) ← −Ω.lhs(:, 1)
57:         WestBlock(:, i) ← −Ω.lhs(1, :)
58:     end for
59:     A()                                                ▷ Allocate Matrix
60:     for all iface ∈ tbp do                             ▷ Insert Blocks into Matrix
61:         reverseColumns ← False
62:         reverseRows ← False
63:         if iface.side = East or iface.side = South then
64:             reverseColumns ← True
65:         end if
66:
67:         j ← iface.northIndex
68:         A.insertBlock(NorthBlock, j, j, reverseColumns, reverseRows)
69:
70:         if Ω.southIndex ≠ null then
71:             i ← iface.southIndex
72:             A.insertBlock(SouthBlock, i, j, reverseColumns, reverseRows)
73:         end if
74:
75:         if iface.side = South or iface.side = West then
76:             reverseRows ← True
77:         end if
78:
79:         if Ω.eastIndex ≠ null then
80:             i ← iface.eastIndex
81:             A.insertBlock(EastBlock, i, j, reverseColumns, reverseRows)
82:         end if
83:
```

84: **if** $\Omega.westIndex \neq null$ **then**
85: $i \leftarrow iface.westIndex$
86: $A.insertBlock(WestBlock, i, j, reverseColumns, reverseRows)$
87: **end if**
88:
89: **end for**
90: **end procedure**

# 3 Handling Refinement

We want the flux going out of the coarse cell to match the fluxes going into the fine cells.

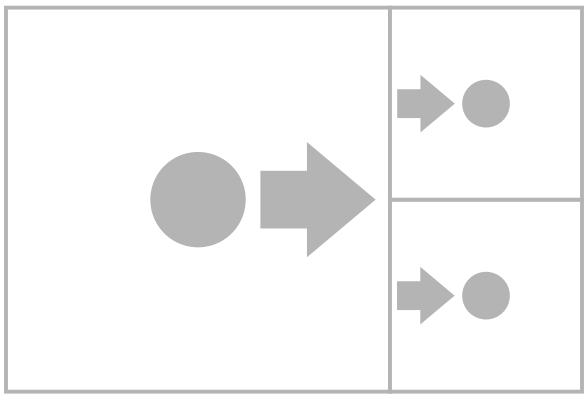


Figure 4: flux

We can represent this with the equation:

$$\Phi_c = \Phi_{f_1} + \Phi_{f_2} \tag{20}$$

### Coming up with a stencil

Lets say we want to find the ghost values for the coarse cell, the first fine cell, and the second fine cell. Labeled $g_c$,$g_{f_1}$,and $g_{f_2}$, respectively.

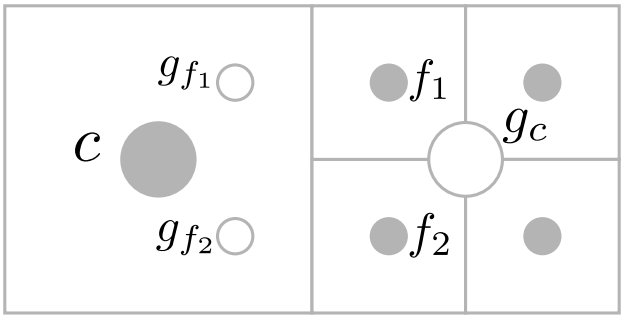



Figure 5: ghost points

We can enforce flux conservation by interpolating to the fine ghost points, and then using equation 20 to find the ghost point for the coarse cell.

The fluxes for each cell will be:

$$\Phi_c = g_c - c \tag{21}$$
$$\Phi_{f_1} = f_1 - g_{f_1} \tag{22}$$
$$\Phi_{f_2} = f_2 - g_{f_2} \tag{23}$$

We can then solve for the value of $g_c$:

$$\Phi_c = \Phi_{f_1} + \Phi_{f_2} \tag{24}$$
$$g_c - c = f_1 - g_{f_1} + f_2 - g_{f_2} \tag{25}$$
$$g_c = c + f_1 - g_{f_1} + f_2 - g_{f_2} \tag{26}$$

**Bilinear interpolation**

Bilinear interpolation for the fine ghost points works, but error is not continuous.

TODO: Explain this and show an example

**Quadratic interpolation**

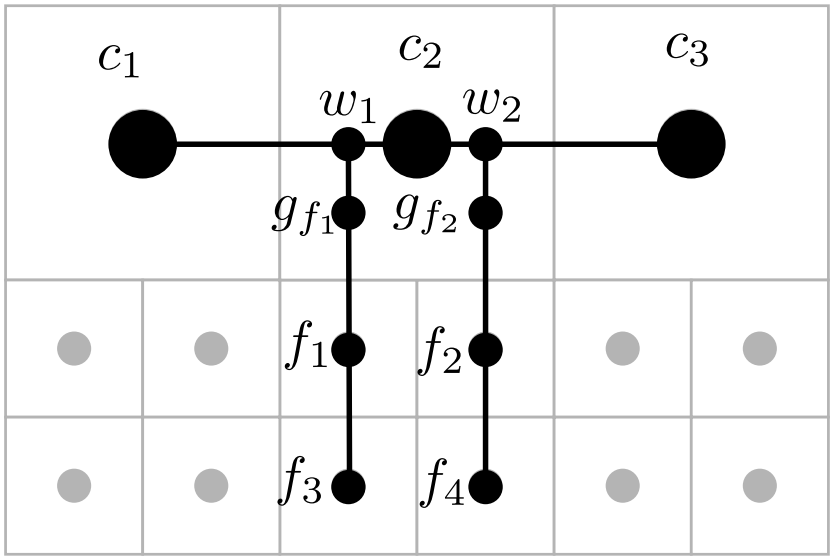



Figure 6: flux

To find the value of $g_{f_1}$, we first use quadratic interpolation with the points $c_1$, $c_2$, and $c_3$ to interpolate to $w_1$:

$$w_1 = \frac{5}{32}c_1 + \frac{15}{16}c_2 - \frac{3}{32}c_3$$

We then use quadratic interpolation with the points $w_1$, $f_1$, and $f_3$ to interpolate to $g_{f_1}$:

$$g_{f_1} = \frac{8}{15}w_1 + \frac{2}{3}f_1 - \frac{1}{5}f_3$$

Plug in the value for $w_2$, and we get the final equation for $g_{f_1}$:

$$g_{f_1} = \frac{1}{12}c_1 + \frac{1}{2}c_2 - \frac{1}{20}c_3 + \frac{2}{3}f_1 - \frac{1}{5}f_3$$

The equation for $g_{f_2}$ is similar:

$$g_{f_2} = -\frac{1}{20}c_1 + \frac{1}{2}c_2 + \frac{1}{12}c_3 + \frac{2}{3}f_2 - \frac{1}{5}f_4$$

Now that we have $g_{f_1}$ and $g_{f_2}$, we can use Eq. 26 to get the value of the ghost point for the coarse cell, $g_{c_2}$:

$$
\begin{aligned}
g_{c_2} &= c_2 + f_1 - g_{f_1} + f_2 - g_{f_2} \\
g_{c_2} &= c_2 + f_1 - \left( \frac{1}{12}c_1 + \frac{1}{2}c_2 - \frac{1}{20}c_3 + \frac{2}{3}f_1 - \frac{1}{5}f_3 \right) + f_2 - \left( -\frac{1}{20}c_1 + \frac{1}{2}c_2 + \frac{1}{12}c_3 + \frac{2}{3}f_2 - \frac{1}{5}f_4 \right) \\
g_{c_2} &= -\frac{1}{30}c_1 - \frac{1}{30}c_3 + \frac{1}{3}f_1 + \frac{1}{3}f_2 + \frac{1}{5}f_3 + \frac{1}{5}f_4
\end{aligned}
$$