INF560: VERSION 2 - SCHUR COMPLEMENT

1. Sub-structuring based parallelization of matrix forming . Parallelization of the forming of the matrix K must be based on the splitting of the matrix in various sub-blocks to be formed independently. The most natural way to split the matrix into blocks consists in splitting the computational domain into groups of elements, each group defining a subdomain. If the groups are distinct, the subdomains do not overlap. Since the stiffness matrix is built through elemental integration, integrating over each subdomain gives contributions to various sub-blocks of the global stiffness matrix.

For instance, a splitting into two groups of elements gives two disconnected open subsets, Ω_1 and Ω_2, with an interface Γ_3 as shown in Figure **??**. If the degrees of freedom of the subsets are reordered in such a way that subsets 1, 2 and 3, are respectively the sets of degrees of freedoms attached to inner nodes of Ω_1 and Ω_2 or to nodes located on Γ_3, then the global stiffness matrix has the following block structure:

$$K = \begin{pmatrix} K_{ii}^{(1)} & 0 & K_{ip}^{(1)} \\ 0 & K_{ii}^{(2)} & K_{ip}^{(2)} \\ K_{pi}^{(1)} & K_{pi}^{(2)} & K_{pp} \end{pmatrix} \tag{1.1}$$

Block K_{pi} of the lower part of the matrix is the transposed of block K_{ip} of the upper part of the matrix, and each diagonal block K_{ii} is symmetric positive definite if the global stiffness matrix K is.

With a local numbering of degrees of freedom in each subdomain according to increasing global number, local stiffness matrices of both subdomains Ω_1 and Ω_2 are:

$$K^{(1)} = \begin{pmatrix} K_{ii}^{(1)} & K_{ip}^{(1)} \\ K_{pi}^{(1)} & K_{pp}^{(1)} \end{pmatrix} \quad K^{(2)} = \begin{pmatrix} K_{ii}^{(2)} & K_{ip}^{(2)} \\ K_{pi}^{(2)} & K_{pp}^{(2)} \end{pmatrix} \tag{1.2}$$

The non zero entries in matrices $K_{pp}^{(1)}$ and $K_{pp}^{(2)}$ are the interaction coefficients between degrees of freedom attached to nodes located on the interface Γ_3,computed in Ω_1 and Ω_2. Block K_{pp} is the sum of these two blocks.
In the same way, local integration of right-hand side of equation (**??**) over each subdomain gives two contributed right-hand side vectors:

$$\begin{pmatrix} b_i^{(1)} \\ b_p^{(1)} \end{pmatrix} \quad \begin{pmatrix} b_i^{(2)} \\ b_p^{(2)} \end{pmatrix} \quad \text{with } b_p = b_p^{(1)} + b_p^{(2)} \tag{1.3}$$

So, the substructuring methodology gives a natural way to develop distributed parallel algorithms for forming and solving linear systems arising from finite element methods. Each subdomain will be allocated to an independent process that will handled only the subdomain data, mainly the mesh and physical problem description and the stiffness matrix. The source code of each process will be mostly identical to the standard sequential finite element program augmented with calls to message-passing routines to ensure the required exchange of data between the subdomains.

2. Parallel Block Factorization.

2.1. Sub-structuring based Condensation. A factorization of matrix (**??**) leads to:

$$\begin{pmatrix} K_{ii}^{(1)} & 0 & K_{ip}^{(1)} \\ 0 & K_{ii}^{(2)} & K_{ip}^{(2)} \\ K_{pi}^{(1)} & K_{pi}^{(2)} & K_{pp} \end{pmatrix} = \begin{pmatrix} L_{ii}^{(1)} & 0 & 0 \\ 0 & L_{ii}^{(2)} & 0 \\ L_{pi}^{(1)} & L_{pi}^{(2)} & L_{pp} \end{pmatrix} \begin{pmatrix} U_{ii}^{(1)} & 0 & U_{ip}^{(1)} \\ 0 & U_{ii}^{(2)} & U_{ip}^{(2)} \\ 0 & 0 & U_{pp} \end{pmatrix} \quad (2.1)$$

The blocks of the factorization can be derived by identification:

$$\begin{cases} K_{ii}^{(1)} = L_{ii}^{(1)} U_{ii}^{(1)} & \Rightarrow & \begin{cases} L_{pi}^{(1)} & = & K_{pi}^{(1)} {U_{ii}^{(1)}}^{-1} \\ U_{ip}^{(1)} & = & {L_{ii}^{(1)}}^{-1} K_{ip}^{(1)} \end{cases} \\ K_{ii}^{(2)} = L_{ii}^{(2)} U_{ii}^{(2)} & \Rightarrow & \begin{cases} L_{pi}^{(2)} & = & K_{pi}^{(2)} {U_{ii}^{(2)}}^{-1} \\ U_{ip}^{(2)} & = & {L_{ii}^{(2)}}^{-1} K_{ip}^{(2)} \end{cases} \end{cases} \quad (2.2)$$

$$\Rightarrow \quad L_{pp} U_{pp} = K_{pp} - K_{pi}^{(1)} {K_{ii}^{(1)}}^{-1} K_{ip}^{(1)} - K_{pi}^{(2)} {K_{pp}^{(2)}}^{-1} K_{ip}^{(2)} \quad (2.3)$$

******* Note $S = L_{pp}U_{pp}$ the Schur complement on interface defined in equation (2.3). ******* This Schur complement is clearly the sum of the two Schur complements of stiffness matrices of subdomains Ω_1 and Ω_2:

$$S = S^{(1)} + S^{(2)} \quad \begin{cases} S^{(1)} & = & K_{pp}^{(1)} - K_{pi} {K_{ii}^{(1)}}^{-1} K_{ip}^{(1)} \\ S^{(2)} & = & K_{pp}^{(2)} - K_{pi} {K_{ii}^{(2)}}^{-1} K_{ip}^{(2)} \end{cases} \quad (2.4)$$

Equations (2.3) and (2.4) show that the factorization of matrix K can be computed in three steps:

1. partial factorization of local matrices:

$$\begin{array}{rcl|rcl} K_{ii}^{(1)} & = & L_{ii}^{(1)} U_{ii}^{(1)} & K_{ii}^{(2)} & = & L_{ii}^{(2)} U_{ii}^{(2)} \\ L_{pi}^{(1)} & = & K_{pi}^{(1)} {U_{ii}^{(1)}}^{-1} & L_{pi}^{(2)} & = & K_{pi}^{(2)} {U_{ii}^{(2)}}^{-1} \\ U_{ip}^{(1)} & = & {L_{ii}^{(1)}}^{-1} K_{ip}^{(1)} & U_{ip}^{(2)} & = & {L_{ii}^{(2)}}^{-1} K_{ip}^{(2)} \\ S^{(1)} & = & K_{pp}^{(1)} - L_{pi}^{(1)} U_{ip}^{(1)} & S^{(2)} & = & K_{pp}^{(2)} - L_{pi}^{(2)} U_{ip}^{(2)} \end{array} \quad (2.5)$$

2. assembly of the Schur complement on interface:

$$S = S^{(1)} + S^{(2)} \quad (2.6)$$

3. factorization of the Schur complement on interface:

$$S = L_{pp} U_{pp} \quad (2.7)$$

The first step can be performed in parallel with one process per sub-domain. ******* The second one requires collecting data from all subdomain processes and cannot be parallelized using a subdomain based parallelization.
****

Once the matrix K has been factorized, the solution of system $Kx = b$ can be performed in parallel using a similar scheme in three steps:

1. local forward substitution:

$$\begin{array}{rcl|rcl} L_{ii}^{(1)} z_i^{(1)} & = & b_i^{(1)} & L_{ii}^{(2)} z_i^{(2)} & = & b_i^{(2)} \\ y_p^{(1)} & = & b_p^{(1)} - L_{pi}^{(1)} z_i^{(1)} & y_p^{(2)} & = & b_p^{(2)} - L_{pi} z_p^{(2)} \end{array} \tag{2.8}$$

2. assembly and solution of the condensed problem on interface:

$$\begin{array}{rcl} y_p & = & y_p^{(1)} + y_p^{(2)} \\ L_{pp} U_{pp} x_p & = & y_p \end{array} \tag{2.9}$$

3. local backward substitution:

$$\begin{array}{rcl|rcl} y_i^{(1)} & = & z_i^{(1)} - U_{ip}^{(1)} x_p & y_i^{(2)} & = & z_i^{(2)} - U_{ip} x_p \\ U_{ii}^{(1)} x_i^{(1)} & = & y_i^{(1)} & U_{ii}^{(2)} x_i^{(2)} & = & y_i^{(2)} \end{array} \tag{2.10}$$

REMARK 2.1. *S is a dense matrix.*

2.2. Generalization to more than two sub-domains. The partitioning of the global domain into several sub-domains can be performed with some software such as METIS, Jostle, Chaco, etc.

REMARK 2.2. *In the following notations we suppose that no cross point, i.e., point belongings to more two sub-domains, exist. If not the case, a weighted scalar product, as explained during the lectures, must be used.*

The generalization to multi-sub-domain splitting do not lead to particular difficulties. Each local Schur complement matrix is computed and then assembled with the other Schur complement matrices of the other subdomains. The assembled Schur complement can be expressed as:

$$S = \sum_s S^{(s)}$$

with

$$S^{(s)} = K_{pp}^{(s)} - K_{pi}^{(s)} [K_{ii}^{(s)}]^{-1} K_{ip}^{(s)}$$

the local Schur complement matrix in the sub-domain (s).

2.3. Implementation. The algorithm to compute the sub-structuring based condensation on the interface is presented Algorithm 1.

Algorithm 1: Algorithm:

Data: $K_{ii}^{(k)}$, $K_{pi}^{(k)}$, $K_{ip}^{(k)}$, $K_{pp}^{(k)}$
Result: Schur complement S
`// -- partial factorization of local matrices`
$K_{ii}^{(k)} \leftarrow L_{ii}^{(k)} U_{ii}^{(k)}$
$L_{pi}^{(k)} \leftarrow K_{pi}^{(k)} U_{ii}^{(k)^{-1}}$
$U_{ip}^{(k)} \leftarrow L_{ii}^{(k)^{-1}} K_{ip}^{(k)}$
$S^{(k)} \leftarrow K_{pp}^{(k)} - L_{pi}^{(k)} U_{ip}^{(k)}$
`// -- assembly of the condensed problem on interface`
for $s = 1 \rightarrow number\_of\_subdomains - 1$ **do**
  Send $S^{(k)}$ to all_subdomains
for $s = 1 \rightarrow number\_of\_subdomains - 1$ **do**
  Receive $S^{(s)}$ from all_subdomains
$S = \sum_{s=1 to all\_subdomains} S^{(s)}$
`// -- factorization of the condensed problem on interface`
$S \leftarrow L_{pp} U_{pp}$

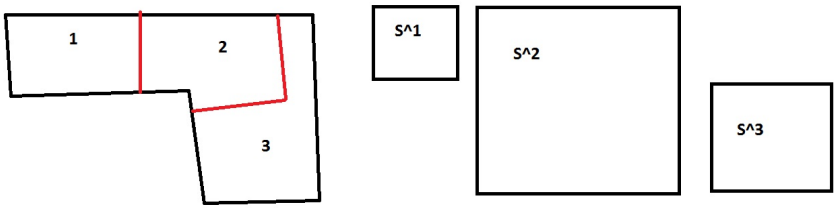

FIG. 2.1. *Left: Partitioning of the global domain. Right: Schur complement matrix* $S^{(1)}$ $S^{(2)}$ $S^{(3)}$

Once the sub-structuring based condensation on the interface obtained, the algorithm to solve the linear system is presented Algorithm 2.

Algorithm 2: Algorithm:

Data: $L_{ii}^{(k)}$, $b_i^{(k)}$, $L_{pi}^{(k)}$, $U_{ii}^{(k)}$, $U_{ip}^{(k)}$
`// -- local forward substitution`
$L_{ii}^{(k)} z_i^{(k)} = b_i^{(k)}$
$y_p^{(k)} = b_p^{(k)} - L_{pi}^{(k)} z_i^{(k)}$
`// -- assembly and solution of the condensed problem on interface`
for $s = 1 \rightarrow number\_of\_subdomains$ **do**
  Send $y^{(k)}$ to all_subdomains
for $s = 1 \rightarrow number\_of\_subdomains$ **do**
  Receive $y^{(s)}$ from all_subdomains
$y_p \leftarrow \sum_{s=1 to all\_subdomains} y_p^{(s)}$
$L_{pp} U_{pp} x_p = y_p$
`// -- local backward substitution`
$y_i^{(k)} \leftarrow z_i^{(k)} - U_{ip}^{(k)} x_p$
$U_{ii}^{(k)} x_i^{(k)} = y_i^{(k)}$

REMARK 2.3. *The complete Schur complement matrix is defined on the total interface, i.e., the union of all the interfaces between the subdomains. As a consequence, during the assembly, local-to-global correspondence array must be defined, such as the l2g introduced in the lectures.*

2.4. Properties of the Schur Complement Matrix. To analyze the convergence properties of the iterative method used to solve the condensed system **??**, some definition and properties are now announced.

DEFINITION 2.4. *A symmetric* $n \times n$ *real matrix* M *is said to be positive definite if the scalar* $z^\top M z$ *is positive for every non-zero column vector* z *of* n *real numbers. Here* $z^\top$ *denotes the transpose of* z.

LEMMA 2.5. *If the matrix* $K^{(s)}$ *is symmetric positive definite, then the Schur complement matrix* $S^{(s)} = K_{pp}^{(s)} - K_{pi}^{(s)} [K_{ii}^{(s)}]^{-1} K_{ip}^{(s)}$ *is symmetric positive definite.*

Proof. TO DO EXERCICE □

DEFINITION 2.6. *Let* A *be a* $n \times n$ *a real matrix. Then matrix* A *is also a Z-matrix if its off-diagonal entries are less than or equal to zero; that is, a Z-matrix* A *satisfies* $A = (a_{ij})$, $a_{ij} \leqslant 0$, *for* $i \neq j$, $1 \leq i, j \leq n$.

DEFINITION 2.7. *Let* A *be a* $n \times n$ *real Z-matrix. Then matrix* A *is also an M-matrix if one of the following conditions is satisfied: (i)* M *is non-singular and* $M^{-1} \geqslant 0$, *i.e., all the coefficients of the inverse are positive; (ii) all the eigenvalues*

of M have a real part strictly positive.

LEMMA 2.8. *If matrix $K^{(s)}$ is symmetric positive definite, then the Schur complement matrix $S^{(s)} = K_{pp}^{(s)} - K_{pi}^{(s)}[K_{ii}^{(s)}]^{-1}K_{ip}^{(s)}$ is a M-matrix.*

Proof. TO DO EXERCICE □

3. The Schur complement method .

3.1. Principle of the method. The Schur complement method can be presented as a mixture of the iterative and direct solution techniques based on substructuring in the case of a symmetric positive definite matrix. It consists in stopping the factorization after it has been computed for the inner sub-matrices within each subdomain and using an iterative method for solving the condensed interface problem associated with the Schur complement. As the Schur complement is also symmetric positive definite it can be solved by the conjugate gradient algorithm.
In the case of 2-subdomain splitting as in section 1, the condensed interface problem derives from the elimination of inner degrees of freedom in each subdomain:

$$K_{ii}x_i = b_i - K_{i3}x_3 \tag{3.1}$$

and can be written:

$$\begin{array}{c}[K_{33} - K_{31}K_{11}^{-1}K_{13} - K_{32}K_{22}^{-1}K_{23}]\ x_3 \quad = \\ b_3 - K_{31}K_{11}^{-1}b_1 - K_{32}K_{22}^{-1}b_2\end{array} \tag{3.2}$$

Since only products by the Schur complement matrix are required to solve the condensed interface problem, the Schur complement does not need to be actually assembled. Then, the method looks like an iterative solution of the global problem using exact solution of local problems. Such a solver is called a domain decomposition method.

3.2. Computation of the product by the Schur complement. As shown in equation (2.4), the Schur complement S is obtained by assembling the local Schur complements $S^{(1)}$ and $S^{(2)}$. So, computing the product by S can be performed by computing the products by $S^{(1)}$ and $S^{(2)}$ and assembling the resulting vectors.

Computing the matrix-vector product $S^{(s)}w_p$ can be performed in two steps:

1. solution of local problem:

$$K_{ii}^{(s)}w_i^{(s)} = -K_{ip}^{(s)}w_p \tag{3.3}$$

whose solution vector is defined on $\Omega^{(s)} \cup \Gamma_p$ by:

$$\begin{pmatrix} w_i^{(s)} \\ w_p \end{pmatrix} = \begin{pmatrix} -{K_{ii}^{(s)}}^{-1}K_{ip}^{(s)}w_p \\ w_p \end{pmatrix} \tag{3.4}$$

2. matrix-vector product:

$$\begin{pmatrix} K_{ii}^{(s)} & K_{ip}^{(s)} \\ K_{pi}^{(s)} & K_{pp}^{(s)} \end{pmatrix} \begin{pmatrix} w_i^{(s)} \\ w_p \end{pmatrix} = \begin{pmatrix} 0 \\ S^{(s)}w_p \end{pmatrix} \tag{3.5}$$

So, the matrix product by the local Schur complement $S^{(s)}$ can be performed just using the matrix of the subdomain and the factorization of sub-matrix $K_{ii}^{(s)}$ associated with

internal nodes and degrees of freedom of the subdomain. **In this method the Schur complement itself $S^{(s)}$ does not need to be formed.**

In the case of a multi-domain mesh splitting, the same technique applies: the computation of the product by the Schur complement can be performed by assembling on the interface the results of the products by the local Schur complements. The procedure for assembling the interface vectors is exactly the same as in the case of the complete matrix-vector product as in section **??**.

4. TODO PROJECT. As explained during the lectures on the black board:

1. Program the sub-structuring based condensation and its solution with forward-backward
2. Program the Primal Schur complement method