Prof. Dr. Stefan Funken
Prof. Dr. Karsten Urban
M.Sc. Lewin Ernst

High Performance Computing II
Institute for Numerical Mathematics
Summer Term 2022

Parallel FEM

Lets consider the following poisson problem with neumann and dirichlet boundary conditions

$$\begin{aligned} -\Delta u &= f, && \text{in } \Omega \\ u &= u_D, && \text{on } \Gamma_D \subset \partial\Omega \\ \frac{\partial u}{\partial n} &= g, && \text{on } \Gamma_N := \partial\Omega \setminus \Gamma_D. \end{aligned}$$

As we know from the lecture and previous exercise we can reformulate this into its weak form and seek for a solution in a finite dimensional space, such that we end up with a discrete problem. Find coefficients $\mathbf{u}_1, \dots, \mathbf{u}_N$ with

$$\sum_{i=1}^{N} \mathbf{u}_i \int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j dx = \int_\Omega f\varphi_j dx + \int_{\Gamma_N} g\varphi_j ds - \sum_{i=N+1}^{N+M} \mathbf{u}_{D,i} \int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j dx, \quad \forall j = 1, \dots, N. \tag{1}$$

Eventually, we receive a linear system of equations

$$\mathbf{A}\mathbf{u} = \mathbf{b} \tag{2}$$

with the stiffness matrix

$$\mathbf{A} = (a_{i,j})_{i,j=1,\dots,N}, \quad a_{i,j} = \int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j dx \tag{3}$$

and the right hand side

$$\mathbf{b} = (b_j)_{j=1,\dots,N}, \quad b_j = \int_\Omega f\varphi_j dx + \int_{\Gamma_N} g\varphi_j ds - \sum_{i=N+1}^{N+M} \mathbf{u}_{D,i} \int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j dx \tag{4}$$

Now we aim at solving (2) in a parallel manner with `MPI` on Pacioli. Due to the fact that the entries of (3) and (4) are actually integrals over just a few elements T of the triangulation $\mathcal{T}_h$, we decompose the domain Ω in order to decompose the matrix and the right hand side.

To keep things simple, we choose as a domain $\Omega := (0,1)^2$ the unit square in $\mathbb{R}^2$. Consider a special partitioning of Ω with a local and global node numbering.

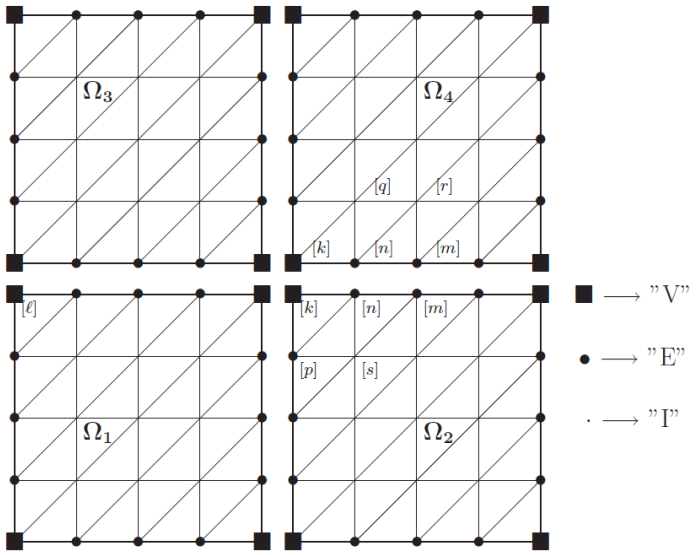

Figure 1: Non-overlapping subdomains (see [3], p.80ff)

Regarding figure 1, we denote by **I** the interior nodes, **E** the nodes in interior of subdomains-edges and **V** the crosspoints. Now, we may define P subdomains $\bar{\Omega}_i$ and assign the entries of the matrix and vectors to the P processes $\mathbb{P}_i$ w.r.t. the node numbering. With the boolean matrix P_i, we can map a global vector, denoted by $\underline{u}$ to a local vector $\underline{u}_i$ of subdomain $\bar{\Omega}_i$. In general, we distinguish between two types of vectors:

- Accumulated vector (type 1): $\bar{u}$ is stored in process $\mathbb{P}_i$ such that $\bar{u}_i = P_i \bar{u}$, i.e. each process $\mathbb{P}_i$ owns the full values of its vector components.
- Distributed vector (type 2): $\underline{r}$ is stored in process $\mathbb{P}_i$, such that $\underline{r} = \sum_{i=1}^{P} P_i^T \underline{r}_i$.

With this knowledge we are ready to implement parallel classical solvers, e.g. ω-Jacobi- or PCG-method.

Tasks

1. To further enlarge your knowledge read [1] chapter 5 and 6 or [3] chapter 5 and 6. Actually both are nearly identical, the latter is more detailed.
2. With the new information try to understand what the difference between serial FEM and parallel FEM is. This means answering the questions: What steps are necessary to calculate a solution of (1) with the serial FEM (Triangulation, Integrals,...) ? What data structures are needed? What does the pseudo code look like in this case? Which steps are different within the parallel FEM? In which way are they different? Write down an abstract pseudo code for the parallel FEM.
3. For the parallel FEM you will need a `mesh_split` function, a `local_mesh` struct and a `scatter_mesh` function.
4. After the mesh has been scattered one can assemble the distributed stiffness matrix and the right hand side on each process $\mathbb{P}_i$.
5. You will need a type 2 to type 1 vector conversion and a matrix vector product.

6. Implement at least two of the following parallel classical schemes, which can be found either in the material or in [3]:

   a) Implement the (parallel) CG Method with respect to the decomposed domain.

   b) Implement the (parallel) ω-Jacobi Method with respect to the decomposed domain..

   c) Implement the (parallel) Gauss-Seidel-ω-Jacobi Method with respect to the decomposed domain.

   d) Implement the (parallel) block-Gauss-Seidel Method with respect to the decomposed domain ([3] Alg.6.11). Therefore you have to adjust your triangulation.

   e) Implement a (parallel) PCG Method (Gauss-Seidel precond.) with respect to the decomposed domain.

7. Test the methods by solving the Poisson problem for different degrees of freedom.

8. Evaluate the scalability of your code, i.e. plot for different number of N (DOF) the time over the number of processes.

References

[1] Craig C Douglas, Gundolf Haase, and Ulrich Langer. *A tutorial on elliptic PDE solvers and their parallelization*. SIAM, 2003.

[2] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI, third edition: Portable Parallel Programming with the Message-Passing Interface*. Scientific and Engineering Computation. MIT Press, 2014. ISBN: 9780262527392. URL: `https://books.google.de/books?id=3I9kBQAAQBAJ`.

[3] Gundolf Haase. *Parallelisierung numerischer Algorithmen für partielle Differentialgleichungen*. Teubner, 1999.

[4] M Quin. "parallel programming in C with MPI and OpenMP". In: *Tata McGraw Hills edition* (2000).