Exercises for Advanced Topics in High Performance Scientific Computing
WS 15/16, Nr.: 0000003213, 02.10.011, Rechnerraum
Exercise Sheet 8 (until December 9, 2015)

Exercise 8* Use the finite element method with linear elements and the Jacobi method to solve the Laplace equation

$$\Delta u(x,y) = 0, \quad (x,y) \in \Omega \tag{1}$$

on the domain Ω with the boundary conditions

$$u(x,y,t) = \begin{cases} +1, & (x,y) \in \Gamma^+ \\ -1, & (x,y) \in \Gamma^- \\ 0, & (x,y) \in \Gamma^0 \end{cases} . \tag{2}$$

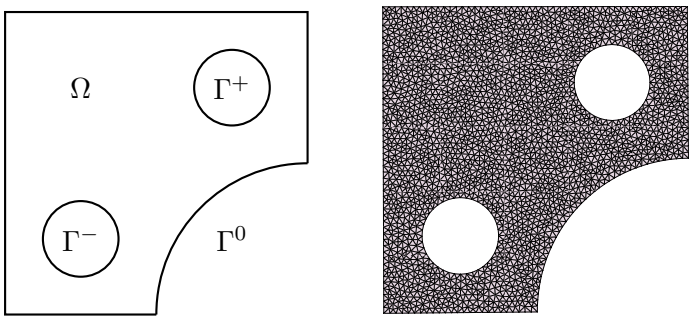

Figure 1: Domain Ω with boundaries Γ^+, Γ^-, and Γ^0.

Parallelize the finite element simulation by partitioning the triangle data in the `Elements.bin` file with respect to the partition vector $\pi \in \mathbb{N}_0^m$ stored as array of 32 bit integers in the file `Partition-x.bin` where `x` is replaced with the number of MPI processes P. The partition vector π stores for every triangle in the `Elements.bin` file the assigned MPI rank. Implement an efficient file I/O strategy for every process to read the assigned triangles. Test data: `/share/apps/domain.zip` on `mephisto.uni-graz.at` (Hint: Every process reads a big chunk of data from the file and redistributes the data with `MPI Alltoallv` to the other processes.)
After the element redistribution every process holds its set of triangles

$$T_p := \{(a_i, b_i, c_i) : \pi(i) = p \wedge 0 \leq i < m\} . \tag{3}$$

The set of global node numbers I_p present on process p is thus defined as the union of all global nodes appearing in T_p. Note that due to the finite element assembly process the row and column index sets I_p and J_p are the same. With the index set I_p the associated coordinates can be read from the `Coordinates.bin` file.
To enable efficient computations in the element assembly stage and in the Jacobi iteration it is necessary to introduce local indices for the set I_p. The global node numbers in T_p have then to be replaced with these local indices. After this substitution the sequential finite element assembly routine can be used to calculate the matrix A_p and the right hand side f_p.
Due to the fact that the node sets I_p and I_q on different processes p and q are now not disjoint the boundary node sets

$$B_p^q := I_p \cap I_q, \quad 0 \leq p, q < P \tag{4}$$

have to be calculated. The boundary node sets B_p^q then define the communication structure necessary for the sparse matrix-vector multiplication. Every process p has to send out data

values associated to the boundary node set B_p^q to process q and then has to accumulate the received data values. To communicate and then accumulate the boundary data for a vector $v_p \in \mathbb{R}^{I_p}$ the values

$$(v_p|_{B_p^0}, v_p|_{B_p^1}, \ldots, v_p|_{B_p^{P-1}}) \tag{5}$$

have to be send out to the other processes and the receive buffer

$$(v_0|_{B_0^p}, v_1|_{B_1^p}, \ldots, v_{P-1}|_{B_{P-1}^p}) \tag{6}$$

has to be accumulated, i.e. all values with the same global node index have to be added together. Denote the accumulated vector $\hat{v}_p \in \mathbb{R}^{I_p}$ to distinguish it from the original distributed vector $v_p \in \mathbb{R}^{I_p}$. Both vectors v_p and $\hat{v}_p$ are elements of the same vector space $\mathbb{R}^{I_p}$ only the values on the boundary nodes $B_p = \bigcup_{0 \leq q < P} B_p^q$ differ. The sparse matrix-vector multiplication requires the following pattern

$$v_p = A_p \hat{u}_p, \quad 0 \leq p < P\,. \tag{7}$$

to produce the correct result. Thus in every iteration of the Jacobi method an accumulation step has to take place before the matrix multiplication. For the Jacobi method the diagonal of the matrix $D_p := \mathrm{diag}(A_p)$ has to be first accumulated and then inverted.

$$u_p^{(0)} := 0 \tag{8}$$

$$u_p^{(k+1)} := u_p^{(k)} + \hat{D}_p^{-1}\left(f_p - A_p \hat{u}_p^{(k)}\right), \quad 0 \leq p < P, \quad k > 0 \tag{9}$$

* Place all source files of the exercises in a folder named `Exercise8` in your home directory on the `mephisto.uni-graz.at` cluster.