Prof. Dr. Peter Bastian, Santiago Ospina De Los Ríos
IWR, Universität Heidelberg

Submission date: 2024-02-30

Exercise 1 *Isoefficiency Functions for the Jacobi Method*

In this exercise we will do an isoefficiency analysis of different implementations of the Jacobi iteration. Using different data decompositions we will get different scalability results.

1. In the first part of the exercise we will analyze the Jacobi iteration for a two dimensional problem using a one dimensional data decomposition and message passing.[2] Assume we have $N = \sqrt{N}^2$ lattice points with the indices (i, j), $i, j \in \{0, \ldots, \sqrt{N} - 1\}$. Split the inner lattice points along the y-axis into even chunks and add one layer of overlap as shown in figure 1.

   Calculate the isoefficiency function for a single iteration of the Jacobi method considering computations and passing along messages.

2. Let us now consider a two dimensional data decomposition. Instead of splitting just one of the index ranges we will split both leading to a data decomposition like in figure 2.

   Again, calculate the isoefficiency function for a single iteration of the Jacobi method considering computations and passing along messages.

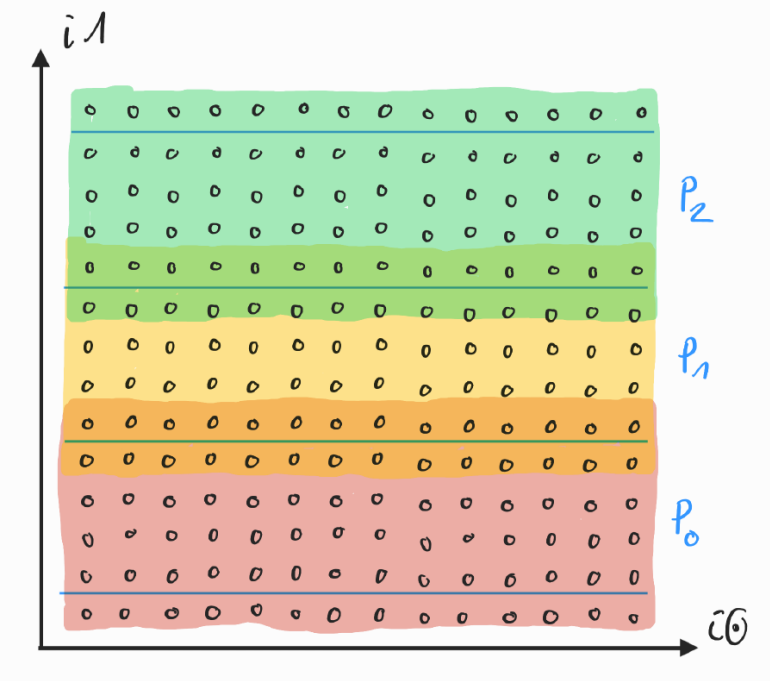

Figure 1: Show 1D data decomposition along the y-axis for $N = 14^2$ lattice points and three processes. The blue lines show the splitting of the inner points and the colors represent the points stored for each process.

[2] Yes, this is exactly what we did in one of the previous exercises.

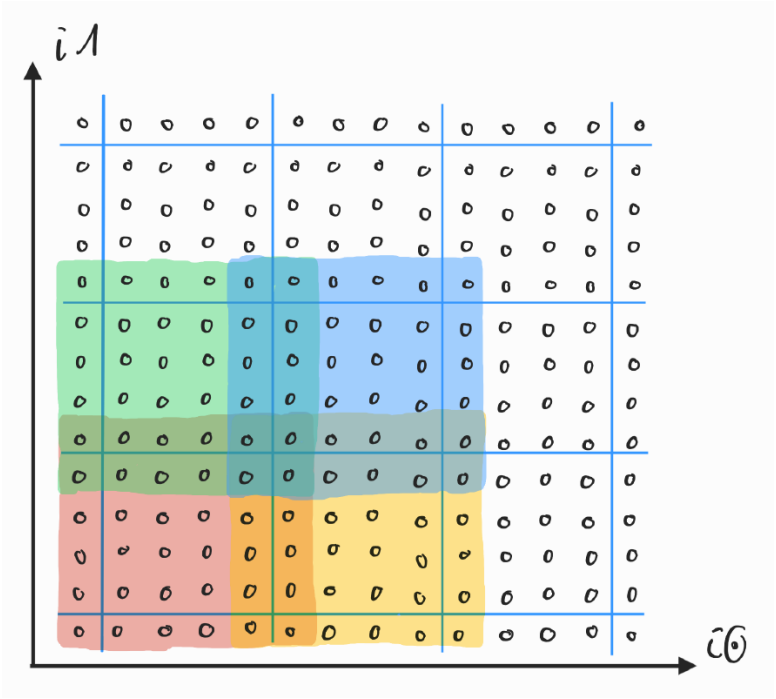

Figure 2: Show 2D data decomposition for $N = 14^2$ lattice points and nine processes. The picture only marks the points of four of the nine processes to keep the picture clear.

(2+2 Points)

Exercise 2 *Isoefficiency Functions for the Gauß-Seidel Method*

Now we want to do a similar analysis for the two dimensional Gauß-Seidel method with a one dimensional data decomposition. Choose a one dimensional data decomposition similar to the previous exercise but along the x-axis[3]. We look at the Gauß-Seidel iteration with lexicographic ordering without source term:

$$u_{i,j}^{l} = 0.25(u_{i-1,j}^{l} + u_{i,j-1}^{l} + u_{i+1,j}^{l-1} + u_{i,j+1}^{l-1})$$

where (i, j), $i, j \in \{0, \ldots, \sqrt{N} - 1\}$ specifies the lattice point and l the iteration number.

The big difference to the previous exercise is that some processes have to wait for certain data points to be computed and sent to them before they can start their computation, see figure 3. Make sure to consider this in your isoefficiency analysis.

1. Let us first look at a single iteration and compute the isoefficiency function.
2. Now consider doing k iterations in a row where each process starts the next iteration as soon as possible. This would for example be the case if we want to do a convergence check every k iterations.

   You may assume that all processes work at the same speed. This means: After finishing its work for iteration l a process can immediately start iteration $l + 1$ by receiving the first data point that was already updated by the previous process. Or in other words: Waiting for data points to be computed will only happen in the first iteration.

[3]For the isoefficiency analysis it is not important if we split along the x- or y-axis. If you want to implement it in an efficient way you have to consider which index is faster (x or y) and choose a decomposition that fits your algorithm.

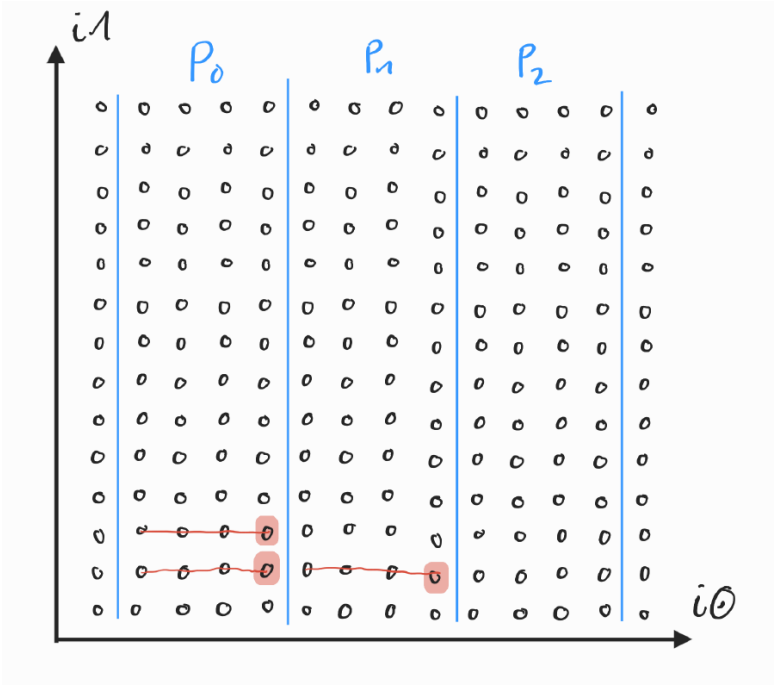

Figure 3: This figure shows the 1D data decomposition for the Gauß-Seidel iteration and for which points new values were already computed at a certain point in time. The first process has already done the first two rows, process P_1 has received one data point from P_0 and has done the computation for one row. Processor P_2 is currently receiving a data point from process P_1 but has not yet done any computations.

(2+2 Points)

Exercise 3 *Isoefficiency Functions for the Scalar Product*

Let us look at the computation of the scalar product using message passing and a blockwise data decomposition for both vectors.

1. Calculate the isoefficiency function using an all to one communication for the final reduction.
2. Calculate the isoefficiency function using recursive reduction (see lecture 2).

(1.5+1.5 Points)