

Exercise for Course
Parallel High-Performance Computing
Dr. S. Lang

Return: 18. December 2014 at the beginning of the exercise or earlier

Task 17 MPI: Communication in the ring (5 points)

With this task we want to perform first steps with MPI. Implement a communication of 8 processes in the ring. Each process shall send its rank within a message once in the ring and terminate, if it again receives its rank within a message. Use synchronous sending resp. receiving and one of the techniques presented in the lecture, to avoid deadlocks, e.g. coloring of the edges. Each process shall in each send/receive step print its ranks and the just received message. Test your program in the pool and hand in an output of the communication sequence.

More details for using of MPI in the pool is on the Homepage. Helpful for the right syntax are the manpages about MPI (e.g. `man MPI_Comm_rank`).

Task 18 Parallel Computing of π with MPI (5 points)

From the identity $\pi = 4(\arctan 1)$ one gets by usage of the derivative of the arctan, $(\arctan x)' = 1/(1+x^2)$, a formula for calculating π :

$$\pi = \int_0^1 \frac{4}{1+x^2} dx.$$

By division of the interval into n equidistant partial pieces the integral can be evaluated with the midpoint rule. You can find a sequential program in the file `piSEQ.c` on the homepage. We want to parallelize it with MPI. The strategy is:

- process 0 reads the number of partial intervals and passes it to all other processes,
- The `for` loop over the partial intervals will be parallelised, each process calculates a local partial sum. The results will be collected by process 0 with a reduction operation `MPI_Reduce` and the partial sums are added.

First determine the convergence order of the midpoint rule with the sequential program. Establish a double-logarithmic plot with the integration error over the interval length h . The steepness of the line give the order. Now implement a parallel version. Compare the accuracy in the calculations (last digits) with the sequential solution and the exact value (short discussion).

Optional task

5 additional

The number of valid digit positions can be enhanced by using the *Gnu Multiprecision Arithmetic Library* (GMP, you can find at www.gmp.org, for the most Linux distributions there is a package). Thus π could be calculated e.g. up to 40., 60. or 80 positions. Of course the chosen method with quadratic convergence is much too slow for that, this means a up to 80. digit exact program would run nearly forever. Implement a version of the sequential or parallel program, that uses the GMP. If you want to gain high accuracy, you need to use a better method than the midpoint rule. Otherwise you can test which accuracy you can achieve with GMP and the midpoint rule in a meaningful computing time. You can get the reference value of π by internet.

Task 19 Simple Parallelisation of the Jacobi method with MPI (10 points)

For linear equation systems $A\mathbf{x} = \mathbf{b}$, $A \in \mathbb{R}^{n \times n}$, $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$, $n \in \mathbb{N}$, direct solution methods are mostly inefficient for large n . Therefore one often uses iterative methods like the *Jacobi method*. You get the

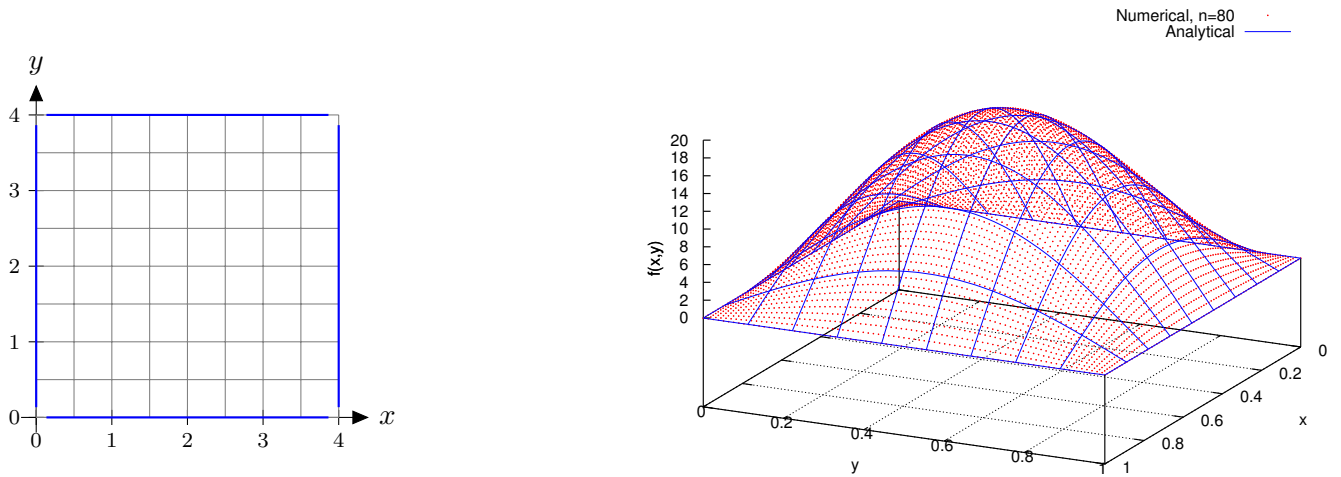


Abbildung 0.5: Left: area Ω with discretization, right: source $f(x, y)$.

Each processor gets in the Jacobi step m a copy of the previous solution $\mathbf{x}^{(m-1)}$, to calculate the new values x_i of a stripe. Then in each iteration each process has to communicate its new values of its partial domain to all other processes, e.g. using multi broadcast.

2. Initialize the initial solution u^h with 0.0. Set $r = 1.0$ and use a tolerance for $\epsilon = 10^{-4}$. Don't forget when assembling the matrix and the initialization of the vectors the treatment of the boundary points! Test with your code the convergence of the error $\|u - u^h\|_\infty$ between analytic u and numerical solution u^h at the grid points for $n = 4, 8, 16, 32$ and 64 . Establish a plot of the error over the grid width h . Can you determine the consistency order?
3. Measure the speedup in the pool against the sequential version ($P = 1$) for different problem sizes $n \leq 32$ and processor counts P .

Hints

- If you like you can implement another strategy, that you can find in a textbook on numerical solution methods for linear equation systems.
- If you have any difficulties you can make the task easier and choose a another matrix without physical application. For convergence of the Jacobi method the matrix A needs to be strictly diagonal dominant: It should apply for each line $\sum_{j=1; j \neq i}^n |a_{ij}| < |a_{ii}|, \forall i \in \{1, \dots, n\}$ (the absolute value of diagonal element of each line should be greater than the sum of absolute values of the off diagonal entries).