Technische Universität München
Institut für Informatik
Prof. Dr. Thomas Huckle
Dipl.-Math. Benjamin Uekermann

WiSe 2016/17

Parallel Numerics

Exercise 11: Previous Exam Questions

1 Preconditioning & Iterative Solvers (From 2016)

Given is the system of linear equations $Ax = b$ with

$$A = \begin{pmatrix} 1 & 0 & 3 \\ 2 & 1 & 0 \\ 0 & -1 & -1 \end{pmatrix}, b = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}.$$

The exact solution of this system is $x = \left(\frac{10}{7}, -\frac{6}{7}, -\frac{1}{7}\right)^T$ and does not have to be calculated separately.

a) Apply the static SPAI algorithm to determine a part of preconditioner $\tilde{M}$ for matrix A with sparsity pattern $\mathcal{J}_1 = \{3\}$. **($\approx$ 3 credits)**

Hint: You can use the *normal equations* to solve a least squares problem.

$$\begin{aligned} \min_{\tilde{M}_1(\mathcal{J}_1)} ||A\tilde{M}_1 - e_1||_2^2 &= \min_{\tilde{M}_{31}} \left|\left| \begin{pmatrix} 1 & 0 & 3 \\ 2 & 1 & 0 \\ 0 & -1 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ \tilde{M}_{31} \end{pmatrix} - \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \right|\right|_2^2 \\ &= \min_{\tilde{M}_{31}} \left|\left| \begin{pmatrix} 3 \\ 0 \\ -1 \end{pmatrix} \tilde{M}_{31} - \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \right|\right|_2^2 \end{aligned}$$

By applying the Normal Equations, we solve the least squares problem:

$$(3,0,-1) \begin{pmatrix} 3 \\ 0 \\ -1 \end{pmatrix} \tilde{M}_{31} = (3,0,-1) \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \Rightarrow \tilde{M}_{31} = 0.3 \Rightarrow \tilde{M}_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.3 & 0 & 0 \end{pmatrix}$$

b) Combine your result of subtask a) with $\begin{pmatrix} 0 & 0.4 & 0 \\ 0 & 0 & -0.5 \\ * & 0 & 0 \end{pmatrix}$ to get an actual right-preconditioner M. Precondition $Ax = b$ with M and calculate two steps of the Jacobi iteration of the preconditioned system. Use $y^{(0)} = (0, 0, 0)^T$ as start vector. **(≈ 4 credits)**

(If you did not get any result in subtask a), then use $\tilde{M}_1 = (0, 0, 0.6)^T$.)

$$\hat{A} = \begin{pmatrix} 1 & 0 & 3 \\ 2 & 1 & 0 \\ 0 & -1 & -1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0.4 & 0 \\ 0 & 0 & -0.5 \\ 0.3 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0.9 & 0.4 & 0 \\ 0 & 0.8 & -0.5 \\ -0.3 & 0 & 0.5 \end{pmatrix}$$

$$\begin{aligned} r^{(0)} &= b - \hat{A}y^{(0)} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} - \begin{pmatrix} 0.9 & 0.4 & 0 \\ 0 & 0.8 & -0.5 \\ -0.3 & 0 & 0.5 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \\ y^{(1)} &= y^{(0)} + D^{-1}r^{(0)} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} \frac{10}{9} & 0 & 0 \\ 0 & \frac{5}{4} & 0 \\ 0 & 0 & 2 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{10}{9} \\ \frac{5}{2} \\ 2 \end{pmatrix} \\ r^{(1)} &= b - \hat{A}y^{(1)} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} - \begin{pmatrix} 0.9 & 0.4 & 0 \\ 0 & 0.8 & -0.5 \\ -0.3 & 0 & 0.5 \end{pmatrix} \cdot \begin{pmatrix} \frac{10}{9} \\ \frac{5}{2} \\ 2 \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \\ \frac{1}{3} \end{pmatrix} \\ y^{(2)} &= y^{(1)} + D^{-1}r^{(1)} = \begin{pmatrix} \frac{10}{9} \\ \frac{5}{2} \\ 2 \end{pmatrix} + \begin{pmatrix} \frac{10}{9} & 0 & 0 \\ 0 & \frac{5}{4} & 0 \\ 0 & 0 & 2 \end{pmatrix} \cdot \begin{pmatrix} -1 \\ 1 \\ \frac{1}{3} \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{15}{4} \\ \frac{8}{3} \end{pmatrix} \end{aligned}$$

c) Without preconditioning, Jacobi gives the following sequence of solutions $x^{(i)}$ for $Ax = b$ (again with a start vector $x^{(0)} = (0, 0, 0)^T$):

$$x^{(1)} = \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix}, x^{(2)} = \begin{pmatrix} 4 \\ 0 \\ -3 \end{pmatrix}, x^{(3)} = \begin{pmatrix} 10 \\ -6 \\ -1 \end{pmatrix}, \ldots, x^{(10)} = \begin{pmatrix} 94 \\ -618 \\ 185 \end{pmatrix}, \ldots$$

Compare this result with the actual solution x and your solution for the preconditioned system from subtask b). What's the reason for your observation? **(≈ 1 credit)**

(If you did not get any result in subtask b), then just compare the result from above with the actual solution x.)

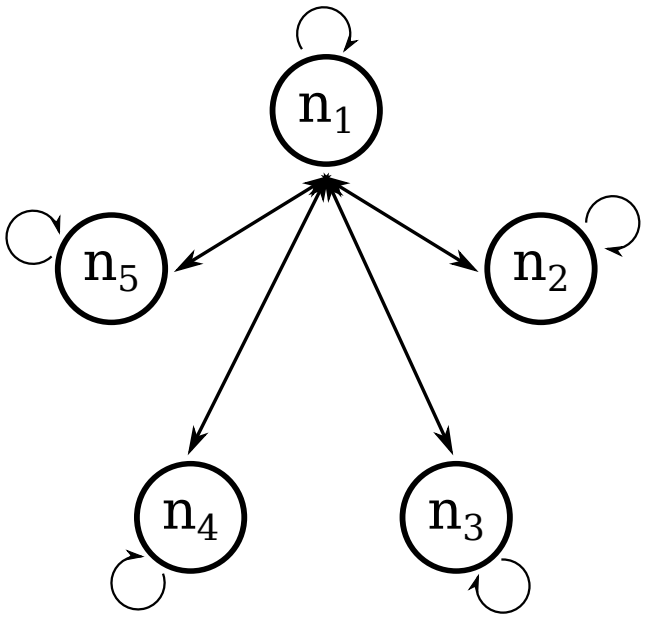
Obviously, the sequence $x^{(1)}, x^{(2)}, \ldots$ diverges while the sequence from subtask b) converges. The reason for this behavior is the usage of the relatively limited Jacobi method and the preconditioner M: M improves the properties of A (better condition number) so that even the relatively limited Jacobi method solves the system successfully.

2 Sparse Gauss Elimination (From 2015)

Given is the sparse symmetric 5×5 matrix

$$A = \begin{pmatrix} 4 & 1 & 2 & \frac{1}{2} & 2 \\ 1 & \frac{1}{2} & 0 & 0 & 0 \\ 2 & 0 & 3 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{5}{8} & 0 \\ 2 & 0 & 0 & 0 & 16 \end{pmatrix}.$$

a) Let A be an adjacency matrix. Draw the corresponding graph $G(A)$! **($\approx$ 1 credit)**



b) Use Gaussian elimination without any pivoting to eliminate the first column of A! What can be observed after this elimination regarding the remaining matrix elements in comparison to the original matrix A? **($\approx$ 2 credits)**

$$\begin{pmatrix} 4 & 1 & 2 & \frac{1}{2} & 2 \\ 1 & \frac{1}{2} & 0 & 0 & 0 \\ 2 & 0 & 3 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{5}{8} & 0 \\ 2 & 0 & 0 & 0 & 16 \end{pmatrix} \rightsquigarrow \begin{pmatrix} 4 & 1 & 2 & \frac{1}{2} & 2 \\ 0 & \frac{1}{4} & -\frac{1}{2} & -\frac{1}{8} & -\frac{1}{2} \\ 0 & -\frac{1}{2} & 2 & -\frac{1}{4} & -1 \\ 0 & -\frac{1}{8} & -\frac{1}{4} & \frac{9}{16} & -\frac{1}{4} \\ 0 & -\frac{1}{2} & -1 & -\frac{1}{4} & 15 \end{pmatrix}$$

There is massive fill in while eliminating the first column. After this elimination step, there are no zeros at all in the resulting matrix which means the matrix is not sparse any more.

c) The *multiple minimum degree reordering* is one way to do pivoting for sparse matrices. It works as follows:

- Define $r_m :=$ number of nonzero entries in row m.
- Choose pivot index i by $r_i = \min_m r_m$. If this holds for multiple i, then chose one of these i randomly.
- Do the pivot permutation and the elimination.

Use Gaussian elimination with *multiple minimum degree reordering* to eliminate the first column of A! What can be observed after this elimination regarding the remaining matrix elements in comparison to the result of subtask b)? **(≈ 3 credits)**

Hint: Since A is a symmetric matrix, rows **and** columns have to be interchanged when pivoting to keep the symmetric structure.

Since $r_2 = r_3 = r_4 = r_5 = 2$ all have the same minimum degree, one of these rows can be chosen randomly. We chose $i = 3$:

$$\begin{pmatrix} 4 & 1 & 2 & \frac{1}{2} & 2 \\ 1 & \frac{1}{2} & 0 & 0 & 0 \\ 2 & 0 & 3 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{5}{8} & 0 \\ 2 & 0 & 0 & 0 & 16 \end{pmatrix} \rightsquigarrow \begin{pmatrix} 3 & 0 & 2 & 0 & 0 \\ 0 & \frac{1}{2} & 1 & 0 & 0 \\ 2 & 1 & 4 & \frac{1}{2} & 2 \\ 0 & 0 & \frac{1}{2} & \frac{5}{8} & 0 \\ 0 & 0 & 2 & 0 & 16 \end{pmatrix} \rightsquigarrow \begin{pmatrix} 3 & 0 & 2 & 0 & 0 \\ 0 & \frac{1}{2} & 1 & 0 & 0 \\ 0 & 1 & -\frac{5}{6} & \frac{1}{2} & 2 \\ 0 & 0 & \frac{1}{2} & \frac{5}{8} & 0 \\ 0 & 0 & 2 & 0 & 16 \end{pmatrix}$$

The fill in is much smaller than without pivoting. With *multiple minimum degree reordering*, there are much more zeros in the transformed matrix then without (6 vs. 0) which also have a sparsity pattern.

d) Briefly describe how *multiple minimum degree reordering* can be used to parallelize Gaussian elimination for sparse matrices! **(≈ 1 credit)**

Assuming there are multiple i where $r_i = \min_m r_m$ holds, then the elimination can be done in parallel where every processing element is doing the elimination with a different i as pivot element. To do so, the corresponding nodes n_i in $G(A)$ must not be neighbors. This approach can also be applied when having different degrees but this can lead to load imbalances.

3 Parallel Jacobi (From 2013)

From the lecture and the tutorials you are familiar with the Jacobi stationary iterative method for solving a linear system $Ax = b$ where A is an $n \times n$ matrix, b the given right-hand side vector and x the solution vector to be computed.

Give a parallel implementation of Jacobi's method by answering the following questions. Assume that a block distribution of all vectors and a block-row distribution of A is available on all processing elements (PEs). The following variables used in the source code are place holders for:

`p`	number of processing elements (PEs)
`n`	dimension of the underlying problem
`my_rank`	rank ID of current PE
`max_iter`	maximum number of iterations to perform
`A_local`	local block-row distribution of A
`b_local`, `x_local`	local block distribution of b and x, respectively

a) Several data has to be available on all processing elements (PEs). Give the MPI/C-code to make the dimension of the underlying problem (variable `n`) and the maximum number of iterations (variable `max_iter`) available on all PEs. Provide only the requested source

code lines, i.e. no complete program is necessary.
Following source code lines are requested:

```
MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
MPI_Bcast(&max_iter, 1, MPI_INT, 0, MPI_COMM_WORLD);
```

b) Formulate the component-wise representation of the Jacobi method for $x_j^{(k+1)}$. Give additionally a **pseudocode** formulation to compute the complete vector $x^{(k+1)}$.
The component-wise representation:

$$x_j^{(k+1)} = \frac{1}{a_{jj}} \left(\sum_{i=1,\ i \neq j}^{n} (-a_{ji} x_i^{(k)}) + b_j \right)$$

Pseudocode for Jacobi with SAXPY for inner for-loop and GAXPY for outer for-loop:

$x^{(k+1)} = 0$
for $j = 1, ..., n$
  for $i = 1, ..., n$
    if $i \neq j$
      $x_i^{(k+1)} = x_i^{(k+1)} - x_j^{(k)} a_{ij}$
    end
  end
end
for $i = 1, ..., n$
  $x_i^{(k+1)} = \frac{x_i^{(k+1)} + b_i}{a_{ii}}$
end

c) Complete the following routine `void Parallel_Jacobi(...)` with **own source code at line 22**. The routine should compute a parallel version of the stationary Jacobi method. Assume that `n` is a multiple of `p` and that the macro `Swap(x,y)` in line **1** is available. Note that the iteration is stopped if `max_iter` is reached or $\|x_{\text{new}} - x_{\text{old}}\|_2 < 10^{-6}$.

```
1:#define Swap(x,y) {float* temp; temp = x; x = y; y = temp;}
2:
3:void Parallel_Jacobi(MATRIX_T A_local, float x_local[], float b_local[],
4:                     int n, int max_iter, int p, int my_rank) {
5:    int i_local, i_diag, j, iter_num;
6:    float x_temp1[MAX_DIM], x_temp2[MAX_DIM];
7:    float* x_old;
8:    float* x_new;
9:
10:   /* Initialize local temporary x */
11:   MPI_Allgather(b_local, n/p, MPI_FLOAT, x_temp1, n/p, MPI_FLOAT,
12:                 MPI_COMM_WORLD);
13:
14:   x_new = x_temp1;
15:   x_old = x_temp2;
```

```
16:    iter_num = 0;
17:    do {
18:       iter_num++;
19:       /* Interchange x_old and x_new */
20:       Swap(x_old, x_new);
21:
22:       /* *** insert code for parallel Jacobi here *** */
23:
24:    } while ((iter_num < max_iter) && (norm(x_new,x_old,n) >= 1e-6));
25:}
```

```
/* run through own local part of vector */
for (i_local = 0; i_local < n/p; i_local++){
   i_diag = i_local + my_rank * n/p;

   /* copy b to x */
   x_local[i_local] = b_local[i_local];

   /* first sum from 0:diag-1 */
   for (j = 0; j < i_diag; j++)
      x_local[i_local] = x_local[i_local] - A_local[i_local][j]*x_old[j];

   /* second sum from diag+1:n */
   for (j = i_diag+1; j < n; j++)
      x_local[i_local] = x_local[i_local] - A_local[i_local][j]*x_old[j];

   /* divide by diagonal element */
   x_local[i_local] = x_local[i_local] / A_local[i_local][i_diag];
}
/* collect new global vector x_new, must be available for norm */
MPI_Allgather(x_local, n/p, MPI_FLOAT, x_new, n/p, MPI_FLOAT, MPI_COMM_WORLD);
```