Technische Universität München
Institut für Informatik
Prof. Dr. Thomas Huckle
Matous Sedlacek

WT 2010/2011

04.02.2011

Parallel Numerics

Exam

A handwritten sheet of paper (size A4, front and back page) may be used during the exam as mnemonic as well as the MPI operation reference distributed during the tutorials. No other material is allowed. The exam is to be solved within 90 minutes, and the answers are to be written in German or English. Please try to answer all parts of the questions precisely and briefly. For passing the exam you will need 17 out of 40 credits. The exam consists of six problems on three pages.

Problem 1: Flynn's Taxonomy (≈ 4 credits)
Describe the classifications of computer architectures defined by Michael J. Flynn in detail and give at least one example for each classification.

- *SISD: Single instruction stream, single data stream. Example: conventional serial computers, uniprocessor machine. (≈ 1 credit)*
- *SIMD: Single instruction stream can handle multiple data streams. All processors are synchronously executing the same instruction stream but on different sets of data. Example: array processor, SSE (Streaming SIMD Extensions) capable of processing multiple data elements in a single clock cycle. (≈ 1 credit)*
- *MISD: Machine is capable to process single data stream using multiple instruction streams. Example: uncommon architecture, fault tolerance (Space shuttle), multiple frequency filters operating on a single signal stream. (≈ 1 credit)*
- *MIMD: Multiple instruction streams on multiple data streams. Example: general purpose parallel computers. (≈ 1 credit)*

0.5 credits for each correct classification. 0.5 credits for each correct example.

Problem 2: MPI Send/Recv (≈ 7 credits)
Consider the following incomplete code fragment (arrays are correctly initialized with values):

```
20: int a[10], b[10], npes, myrank;
21: ...
22: MPI_Comm_size(MPI_COMM_WORLD, &npes);
23: MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
24: MPI_Send(a, 10, MPI_INT, (myrank+1)%npes,1,MPI_COMM_WORLD);
25: MPI_Recv(b, 10, MPI_INT, (myrank-1+npes)%npes,1,MPI_COMM_WORLD);
26: ...
```

a) What topology fits best to the given source code? (≈ **1 credit**)
1D Torus, or circle, or ring (≈ 1 credit). If someone gives only illustration, the credit is given, too.

b) Describe shortly what the given source code does when executed on more than 1 processor on a parallel computer? (≈ **1 credit**)
Process p_i sends a messages to p_{i+1} (modulo the number of processes `npes`*)*

*and process p_i receives a messages from process p_{i-1} (modulo **npes**). Neighboring processes exchange data between each other in the ring topology (≈ 1 credit).*

c) Describe whether the code will work correctly (is safe) if MPI uses a buffer for the send operation? ($\approx$ **1 credit**)
When `MPI_Send()` is implemented using buffering (MPI uses a buffer), the program will work correctly, since every call to `MPI_Send()` will get buffered, allowing the call of the `MPI_Recv()` to be performed ($\approx$ 1 credit), which will transfer the required data.

d) Assume MPI uses no buffer for the send operation but the program is executed on 2 processes only. Describe whether the code fragment is safe for this setting? ($\approx$ **1 credit**)
If `MPI_Send()` blocks until the matching receive has been issued, all processes will enter an infinite wait state($\approx$ 1 credit), waiting for the neighboring process to issue a `MPI_Recv()` operation. Even for 2 processes a deadlock occurs.

e) Rewrite the above code fragment such that it is safe. Use only blocking routines, i.e. `MPI_Send()` and `MPI_Recv()`. Hint: partition the processes into two groups. Modify the source code such that the program will work correctly in send mode with or without buffering ($\approx$ **3 credits**):
Above code fragment can be made safe by using two groups of processes. One consists of the odd-numbered processes and the other of the even-numbered processes. We rewrite the above example as follows:

```
20:  int a[10], b[10], npes, myrank;
21:  ...
22:  MPI_Comm_size(MPI_COMM_WORLD, &npes);
23:  MPI_Comm_rank(MPI_COMM_WOLRD, &myrank);
24:  if (myrank%2 == 1) {
25:      MPI_Send(a, 10, MPI_INT, (myrank+1)%npes, 1, MPI_COMM_WORLD);
26:      MPI_Recv(b, 10, MPI_INT, (myrank-1+npes)%npes, 1, MPI_COMM_WORLD);
27:  }
28:  else {
29:      MPI_Recv(b, 10, MPI_INT, (myrank-1+npes)%npes, 1, MPI_COMM_WORLD);
30:      MPI_Send(a, 10, MPI_INT, (myrank+1)%npes, 1, MPI_COMM_WORLD);
31:  }
32:  ...
```

($\approx$ 1 credit) for splitting into correct groups, ($\approx$ 1 credit) for `MPI_Send()` and `MPI_Recv()` in both branches, ($\approx$ 1 credit) for correct order of operations. For c) and d), if only short "yes" or "no" given, 0.5 credits are given.

Problem 3: Speedup (≈ 4 **credits**)

a) Describe in words Amdahl's law and Gustafson's law for the speedup. What are the differences between the two and what do they have in common with weak speedup and strong speedup? ($\approx$ **2 credits**)
Amdahl's law: constant problem size. Speedup of a program using multiple processes is limited by the sequential fraction of the program and does not depend on problem size ($\approx$ 0.5 credit). Solution time varies with the number of processors for a fixed total problem size (strong speedup) ($\approx$ 0.5 credit).
Gustafson's law: Parallel fraction grows with number of processes and is not bounded by sequential fraction of program ($\approx$ 0.5 credit). Problem is scaled

according to number of nodes available (weak speedup) (≈ 0.5 credit).
1 Credit for correct description of laws. If correspondance of strong and weak speedup is missing in the description (completely) than 1 credit is subtracted.

b) Give the formula for Amdahl's law for the speedup S and describe all variables. (≈ **1 credit**)
Amdahl's law for the speedup S with given sequential fraction f and number of processors p:

$$S = \frac{1}{f + \frac{(1-f)}{p}}.$$

c) Assume you have a program which has a parallel fraction of 80%. Give the maximum attainable speedup S, according to Amdahl, for an infinite number of processors, i.e. $p \to \infty$. (≈ **1 credit**)

$$\lim_{p\to\infty} S = \lim_{p\to\infty} \frac{1}{f + \frac{(1-f)}{p}} = \frac{1}{f} = \frac{1}{0.2} = 5.$$

Problem 4: Hockney/Golub Method (≈ 7 credits)

a) We consider tridiagonal systems of linear equations. Assume the system $Ax = d$ with

$$\begin{pmatrix} b_1 & c_1 & 0 \\ a_2 & b_2 & c_2 \\ 0 & a_3 & b_3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix}. \tag{1}$$

The system (1) can be solved by the Hockney/Golub method which combines the equations recursively to get a final system with a diagonal matrix. Therefore, a certain number of steps has to be executed. In the first combination step of the equations, the matrix A changes to matrix $A^{(1)}$. Write down the matrix $A^{(1)}$! (≈ **2 credits**)

$$A^{(1)} = \begin{pmatrix} b_1 - a_2\frac{c_1}{b_2} & 0 & -c_2\frac{c_1}{b_2} \\ 0 & b_2 - c_1\frac{a_2}{b_1} - a_3\frac{c_2}{b_3} & 0 \\ -a_2\frac{a_3}{b_2} & 0 & b_3 - c_2\frac{a_3}{b_2} \end{pmatrix}$$

1 credit for correct fill-in, 1 credit for correct values.

b) Explain how the Hockney/Golub method can be parallelized for (1) among $p = 3$ processors. Why and between which steps communication becomes necessary? It is sufficient to give an explanation in words of how each processor will compute which components. It is not explicitely requested to give all formulas by applying the Hockney/Golub method to (1). (≈ **3 credits**)

- $i \in \{1, 2, 3\}$
- *Parallelization is performed row-wise. Every p_i has $A^{(0)}$ in the beginning. For step $k = 1$: p_i computes $a_i^{(1)}$, $b_i^{(1)}$, $c_i^{(1)}$, $d_i^{(1)}$. (≈ 1 credit)*
- *For step $k = 2$: As for the computation of the values $a_i^{(2)}$, $b_i^{(2)}$, $c_i^{(2)}$, $d_i^{(2)}$ the values $a_j^{(1)}$, $b_j^{(1)}$, $c_j^{(1)}$, $d_j^{(1)}$ are necessary, $j \in \{i-2, i, i+2\}$, communication becomes necessary, i.e. for $j = i - 2$ communication with p_{i-2}*

necessary, for $j = i + 2$ communication with p_{i+2} necessary, if $i - 2 \geq 1$ and $i + 2 \leq 3$, respectively. Therefore, communication necessary between p_1 and p_3 as they need the values from the step before. (≈ 1 credit)

- *In iteration $k = 1$ every p_i is working / active. In iteration $k = 2$ only p_1 and p_3 are active to eliminate the subdiagonal elements.*
- *After computation of step $k = \lceil \log_2(3) \rceil = 2$, p_i will compute $x_i = \frac{d_i^{(2)}}{b_i^{(2)}}$. ($\approx 1$ credit)*

For a shorter and correct explanation in words the credits are given too.

c) Why is the Hockney/Golub method not applicable to the system $Bx = d$ with

$$B = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 1 & -1 \\ 0 & 1 & 1 \end{pmatrix} ?$$

For which class of matrices it is possible to use the Hockney/Golub method? ($\approx$ **2 credits**)

Hockney/Golub is not applicable to matrix B because for $k = 1,\ i = 1$ the diagonal element $b_1^{(1)} = b_1^{(0)} - a_2^{(0)} \frac{c_1^{(0)}}{b_2^{(0)}} = 0$ which is necessary, for example, for $k = 2,\ i = 3$ to compute $\alpha_3^{(1)} = -\frac{a_3^{(1)}}{b_1^{(1)}}$. Here a division by zero will occur (≈ 1 credit). If $A \in \mathbb{R}^{n \times n}$ is a H-matrix (≈ 1 credit) then Hockney/Golub is possible for every right-hand side d.

Problem 5: Sparse matrices **(≈ 7 credits)**

Given is the sparse matrix $C \in \mathbb{R}^{4 \times 4}$ with

$$C = \begin{pmatrix} -1 & 2 & -1 & 0 \\ 3 & 0 & -1 & 0 \\ 2 & 0 & 0 & 0 \\ 3 & 1 & 0 & 4 \end{pmatrix}.$$

a) Give the Jagged Diagonal storage format for the matrix C. ($\approx$ **2 credits**)

After reordering the matrix has the form

$$\tilde{C} = \begin{pmatrix} -1 & 2 & -1 & 0 \\ 3 & 1 & 0 & 4 \\ 3 & 0 & -1 & 0 \\ 2 & 0 & 0 & 0 \end{pmatrix}.$$

Jagged Diagonal storage format:

$$\begin{aligned} \texttt{AA} &= (-1\ 3\ 3\ 2 \mid 2\ 1\ -1 \mid\ -1\ 4) \\ \texttt{JDIAG} &= (1\ 1\ 1\ 1 \mid 2\ 2\ 3 \mid 3\ 4) \\ \texttt{IDIAG} &= (1\ 5\ 8\ 10) \end{aligned}$$

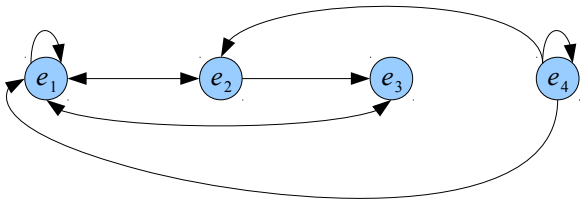
0.5 credits for permutation. 0.5 credits for value array `AA`*, 0.5 credits for* `JDIAG` *array, 0.5 credits for* `IDIAG` *array.*

b) Give the the Adjacency matrix and the graph of C, i.e. give $A(G(C))$ and $G(C)$. ($\approx$ **2 credits**)

The Adjacency matrix of the graph is

$$A(G(C)) = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}.$$

As $A(G(C))$ is nonsymmetric the graph $G(C)$ is directed: 0.5 credits for correct Adjacency matrix. 0.5 credits for directed graph. 1 credit for correct graph structure.



c) Consider the matrix-vector multiplication $f = E \cdot d$ with $E \in \mathbb{R}^{n \times n}$ and $f, d \in \mathbb{R}^n$. The matrix E is given in compressed sparse column format (CSC). Give a pseudocode to compute the vector f using the storage format of E, i.e. use the float array `AE` and the integer arrays `JE`, and `IE` in your pseudocode. Identify and discuss possible SAXPY and GAXPY operations in your code. ($\approx$ **3 credits**)

Pseudocode:

$f = \vec{0}$

```
for jE = 1,...,n
  for iE = JE(jE),...,JE(jE+1)-1
    f(IE(iE)) = f(IE(iE)) + AE(iE)·d(jE)
  endfor
endfor
```

Assuming indirect adressing and omitting of zero values is possible for SAXPY then inner for-loop is SAXPY as f is updated componentwise via columns of E. Outer for-loop is GAXPY as matrix-vector product is performed.

0.5 credits for every for-loop = 1 credit. 1 credit for correct summation of f elements. Therefore, 2 credits for correct pseudocode. 1 credit for correct identification and discussion of SAXPY and GAXPY operations. If the assumptions are not taken as valid for a SAXPY operation the credit is given too.

Problem 6: Graphs and colouring **(≈ 11 credits)**

Given is the symmetric update

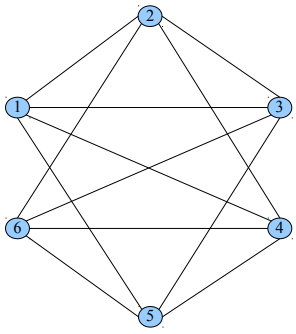
$$f(x) = \begin{pmatrix} f_1(x) \\ f_2(x) \\ f_3(x) \\ f_4(x) \\ f_5(x) \\ f_6(x) \end{pmatrix} = Ax = \frac{1}{5} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} x.$$

where

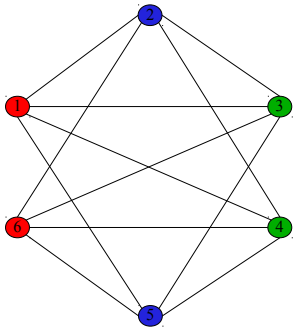
$$x^{(k+1)} = f(x^{(k)}) \tag{2}$$

reflects the dependency between the update $x^{(k+1)}$ and the old solution $x^{(k)}$ in an iterative algorithm. $A \in \mathbb{R}^{6\times 6}$ and $x \in \mathbb{R}^6$.

a) Give the corresponding graph of f which describes the data dependency in the update (2), i.e. give $G(f)$. (≈ **2 credits**)



b) Give and explain a minimum colouring of $G(f)$ such that the update can be performed in parallel. How many steps have to be performed in parallel for the update? (≈ **2 credits**)



There have to be performed 3 steps in parallel.

1 credit for correct graph. 1 credit for correct explanation.

c) Describe a renumbering of $G(f)$ according to the minimum coloring of $G(f)$ where the vertices of one color are numbered consecutively. Apply this permutation only to the first and second row and first and second column of A and

give the upper left 2×2-block of the permuted matrix $\tilde{A}$, i.e. give $\tilde{A}$(1:2,1:2). Explain how to read the parallelism out of the block $\tilde{A}$(1:2,1:2). ($\approx$ **3 credits**)
Applying the full ordering

$$\begin{aligned} 1 &\to 1 \\ 2 &\to 3 \\ 3 &\to 5 \\ 4 &\to 6 \\ 5 &\to 4 \\ 6 &\to 2 \end{aligned}$$

to rows and columns of A we receive

$$\tilde{A} = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} D & B & B \\ B & D & B \\ B & B & D \end{pmatrix}, \quad \textit{with} \quad D = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \textit{and} \quad B = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

Considering the diagonal blocks D (or just the upper left diagonal block for the 2 elements) it is possible to perform the update between the corresponding components of x in parallel. 2 credits for correct permutation. 1 credit for correct explanation how to read out the parallelism of $\tilde{A}$. The credits are given both for full consideration and for the 2×2-block as requested.

d) Give all nontrivial fixpoints of $x = Ax$ (besides $x = \vec{0}$). ($\approx$ **4 credits**)
Searching for kernel x in $(A - I)x = 0$:

$$\begin{pmatrix} 4 & -1 & -1 & -1 & -1 & 0 \\ -1 & 4 & -1 & -1 & 0 & -1 \\ -1 & -1 & 4 & 0 & -1 & -1 \\ -1 & -1 & 0 & 4 & -1 & -1 \\ -1 & 0 & -1 & -1 & 4 & -1 \\ 0 & -1 & -1 & -1 & -1 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = 0. \tag{3}$$

By II = 4II, III = 4III, IV = 4IV we get

$$\begin{pmatrix} 4 & -1 & -1 & -1 & -1 & 0 \\ -4 & 16 & -4 & -4 & 0 & -4 \\ -4 & -4 & 16 & 0 & -4 & -4 \\ -4 & -4 & 0 & 16 & -4 & -4 \\ -4 & 0 & -4 & -4 & 16 & -4 \\ 0 & -1 & -1 & -1 & -1 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = 0.$$

Adding row I to row II, III, IV, V and row VI to row II, III, IV, V we receive

$$\begin{pmatrix} 4 & -1 & -1 & -1 & -1 & 0 \\ 0 & 14 & -6 & -6 & -2 & 0 \\ 0 & -6 & 14 & -2 & -6 & 0 \\ 0 & -6 & -2 & 14 & -6 & 0 \\ 0 & -2 & -6 & -6 & 14 & 0 \\ 0 & -1 & -1 & -1 & -1 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = 0.$$

We consider the inner system divided by 2

$$\begin{pmatrix} 7 & -3 & -3 & -1 \\ -3 & 7 & -1 & -3 \\ -3 & -1 & 7 & -3 \\ -1 & -3 & -3 & 7 \end{pmatrix} \begin{pmatrix} x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = 0. \tag{4}$$

Using III= III-3IV, II = II - 3IV, and I = I + 7IV we obtain

$$\begin{pmatrix} 0 & -24 & -24 & 48 \\ 0 & 16 & 8 & -24 \\ 0 & 8 & 16 & -24 \\ -1 & -3 & -3 & 7 \end{pmatrix} \begin{pmatrix} x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = 0.$$

Dividing by 8 *and first row by* 2 *we consider the system*

$$\begin{pmatrix} -1 & -1 & 2 \\ 2 & 1 & -3 \\ 1 & 2 & -3 \end{pmatrix} \begin{pmatrix} x_3 \\ x_4 \\ x_5 \end{pmatrix} = 0. \tag{5}$$

Using II=II+2I and III = III + I we consider the system

$$\begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} x_4 \\ x_5 \end{pmatrix} = 0.$$

Using $x_4 = x_5$ *in (5) it follows* $-x_3 - x_4 + 2x_5$ *and thus* $x_3 = x_4 = x_5$. *Putting this inside (4) we obtain* $x_2 = x_3 = x_4 = x_5$ *and finally for (3) we get* $x_1 = x_2 = x_3 = x_4 = x_5 = x_6$. *Hence, all nontrivial fixpoints lie on the straight line*

$$x = \alpha \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \quad \alpha \in \mathbb{R}.$$