Otto-von-Guericke-University Magdeburg
Max Planck Institute for Dynamics of Complex Technical Systems
Computational Methods in Systems and Control Theory

Winterterm 2016/2017

Dr. Jens Saak, Dipl.-Math. Martin Köhler
Website: http://www.mpi-magdeburg.mpg.de/csc/teaching/16ws/sc1/

Scientific Computing 1

8th Homework

Handout: 12/08/2016 **Return:** 12/16/2016

Exercise 1: **(12 Points)**

After discussing the memory related problems of the matrix-matrix product in one of the previous exercises, we now consider the implementation problems. In general the matrix-matrix product $C \leftarrow \alpha AB + \beta C$, where $\alpha, \beta \in \mathbb{R}$ and $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times p}$ and $C \in \mathbb{R}^{m \times p}$, is computed by the following algorithm:

1: **for** $i = 1, \ldots, m$ **do**
2:    **for** $j = 1, \ldots, p$ **do**
3:       $C_{ij} = \beta \cdot C_{ij}$
4:    **end for**
5: **end for**
6: **for** $i = 1, \ldots, m$ **do**
7:    **for** $j = 1, \ldots, p$ **do**
8:       **for** $k = 1, \ldots, n$ **do**
9:          $C_{ij} = C_{ij} + \alpha \cdot A_{ik} B_{kj}$
10:       **end for**
11:    **end for**
12: **end for**

The main loop (lines 6 to 12) can be arranged in six ways. The one above is the i-j-k variant but by swaping the loops one obtains the following additional reordered variants i-k-j, j-i-k, j-k-i, k-i-j, and k-j-i.
Implement all six variants as C functions with the following signature

```
void gemmXXX(int m, int p, int n, double alpha, double *A, int lda,
      double *B, int ldb, double beta, double *C, int ldc)
```

where `XXX` should be replaced, such that it indicates the loop order. The matrices A, B, and C are stored in column-major representation (Fortran-like) with the leading dimensions `lda`, `ldb`, and `ldc`.
Measure the runtime of all six variants for arbitrary matrices of different moderate sizes ($m, n, p \leq 500$). Compute the floprate from the size and the runtime. Compare the results and explain the differences.
Hint: The lecture's website provides a sample code to demontrate the runtime measurement in C programs.

Exercise 2: **(2 Points)**

Prove that

$$\kappa_2(A) = 1$$

holds for all unitary matrices $A \in \mathbb{C}^{n \times n}$. In which way does this influence the design of numerical algorithms?

Exercise 3: **(6 Points)**

Consider the linear system $Ax = b$ with

$$A = \begin{bmatrix} 2 & -1 & 1 \\ -1 & 10^{-10} & 10^{-10} \\ 1 & 10^{-10} & 10^{-10} \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 2(1+10^{-10}) \\ -10^{-10} \\ 10^{-10} \end{bmatrix}.$$

a.) Compute the solution x of the linear system.

b.) Show that $\kappa_\infty(A) = 2 \cdot 10^{10}$ holds.

c.) Let $D = \operatorname{diag}(10^{-5}, 10^5, 10^5)$. Check that $\kappa_\infty(DAD) \leq 5$ is true.

Exercise 4: (5 Points)

LAPACK is the key software package for high level numerical linear algebra. It provides Fortran subroutines to solve linear systems, eigenvalue problems and many more. The solution of linear systems is mostly done using the computational subroutines `DGETRF` and `DGETRS` or the driver `DGESV`.
Use the skeleton code provided on the lecture's website and solve the linear systems

$$Ax = b_1 \quad \text{and} \quad Ay = b_2$$

using a proper choice of the above LAPACK routines. The matrix $A \in \mathbb{R}^{487\times 487}$ and the right hand sides $b_1 \in \mathbb{R}^{487}$ and $b_2 \in \mathbb{R}^{487}$ are provided in the skeleton archive. Once, you have computed x and y write them to a file called `solution.dat` with the following style:

```
x_1 y_1
x_2 y_2
...
...
x_487 y_487
```

The `solution.dat` file is used together with `gnuplot` and the `display.plot` script at the end of the program to present the solution. What can you see?

Hints:

- The skeleton code already reads the matrix A and the right hand sides b_1 and b_2.
- If you do not use the virtual machine of the lecture please install `gnuplot` on your system before.
- If the plot does not work, you can use MATLAB as well. Therefore use

```
P = load('./solution.dat');
plot(P(:,1), P(:,2),'*')
```

- The skeleton code provides a Makefile which does all the compilation steps.

Overall Points: 25