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

Dr. Jens Saak, Dipl.-Math. Martin Köhler
Website: http://www.mpi-magdeburg.mpg.de/2802004/sc1/

Winterterm 2014/2015

Scientific Computing 1
8th Homework

Handout: 12/17/2014 **Return:** 01/07/2015

Exercise 1: (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 2: (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 3: (10 Points)

The solution of triangular linear systems $Lx = b$ is a key operation for many high level linear algebra operations. The matrix $L \in \mathbb{R}^{n\times n}$ is stored in the `my_matrix_st` structure, which was presented in the lecture. A skeleton code which provides a read, a print, a random matrix function, and a Makefile is available from the lecture website. This skeleton also includes some example matrices L.
Implement the following functions inside the `main.c` file:

a.) The function

```
void L_solvev(struct my_matrix_st *L, double *b)
```

which takes a lower triangular matrix $L \in \mathbb{R}^{n\times n}$ and a right hand side $b \in \mathbb{R}^n$ as inputs and overwrites b with the solution of $Lx = b$. Implement the naive forward elimination scheme.

b.) The function

```
void L_solvev_trsv(struct my_matrix_st *L, double *b)
```

which overwrites b with the solution of $Lx = b$. Use the BLAS level 2 routine `DTRSV` instead of implementing the forward elimination scheme.

c.) The function

```
void L_solvem(struct my_matrix_st *L, struct my_matrix_st *B)
```

which solves $LX = B$ for multiple right hand sides, i.e. $B \in \mathbb{R}^{n \times p}$, using the triangular solve from a.).

d.) The function

```
void L_solvem_trsv(struct my_matrix_st *L, struct my_matrix_st *B)
```

which solves $LX = B$ for multiple right hand sides, i.e. $B \in \mathbb{R}^{n \times p}$, using the triangular solve from b.).

e.) The function

```
void L_solvem_trsm(struct my_matrix_st *L, struct my_matrix_st *B)
```

which solves $LX = B$ for multiple right hand sides, i.e. $B \in \mathbb{R}^{n \times p}$, using the BLAS level 3 triangular solve `DTRSM`.

Generate a random matrix $B \in \mathbb{R}^{n \times 100}$ and measure the time for one solve $LX = B$ with the different implementations. Use one of the demo matrices `demo100.mtx`, `demo2500.mtx` or `demo5625.mtx` from the skeleton archive as L. What can you recognize regarding the runtimes of the different implementations?

Hints:

- The runtime of a piece of code can be measured using the `wtime` function from the skeleton code:

```
double tic, toc;
tic = wtime();
... Your Code...
toc = wtime();
printf("The code took %lg seconds\n", toc-tic);
```

- If you do not use the virtual machine ensure that you use a optimized BLAS library like ATLAS, OpenBLAS, GotoBLAS, MKL, or Apple Accelerate.
- The skeleton code provides a Makefile which does all the compilation steps.

Exercise 4: (7 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