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
9th Homework

Handout: 01/07/2015 **Return:** 01/14/2015

Exercise 1: (6 Points)

a.) Let $A \in \mathbb{R}^{n\times n}$ be a matrix and $A_{ik} \in \mathbb{R}^{(n-1)\times(n-1)}$ be the submatrix which is created by removing the i-th row and the k-th column of A. We can compute the determinant of the matrix by the recursion formula

$$\det(A) = \sum_{i=1}^{n} (-1)^{i+k} a_{ik} \det(A_{ik}).$$

Derive a recursion formula for the number of necessary floating point operations to compute $\det(A)$. Approximate the computation time for the determinant of a 100×100 matrix on a current CPU. Assume that the CPU has a peak performance of 150 GFlops/s ($150 \cdot 10^9$ floating point operations per second).

b.) Show that the determinant can be computed using the LU decomposition as $\det(A) = \prod_{i=1}^{n} u_{ii}$ Compare the computational effort with the one from a.).

Exercise 2: (6 Points)

In many cases it is not necessary to compile and link the whole LAPACK library to a program, e.g., if one only needs a single driver routine from it.

a.) Search on http://www.netlib.org/lapack/double/ for a driver which computes the eigenvalues and eigenvectors of a general matrix and download it together with its dependencies.

b.) Write a `Makefile` which creates a small static library called `liblapack_pocket.a` containing the eigenproblem solver and its dependencies.

c.) Write a small C program which uses this library to compute all eigenvalues and eigenvectors of

$$A = \begin{pmatrix} -3 & 8 & -2 & 1 \\ 6 & -4 & 1 & 5 \\ 2 & 5 & -8 & 8 \\ 1 & 5 & -8 & -7 \end{pmatrix}$$

Hint: Subroutines from BLAS are not automatically included as dependencies in the download. That means either BLAS needs to be linked to the program, or the corresponding files need to be downloaded separately.

Exercise 3: **(8 Points)**

Consider a matrix stored in the column major format. In order to combine all information related to such a matrix we use the following structure (like in the previous Homework):

```
struct my_matrix_st {
  int cols;
  int rows;
  int LD;
  double * values;
  char structure;
};
```

Implement the following operations on top of this structure without using BLAS or LAPACK:

a.) Write a C function which scales a part of a matrix column by a scalar s. In MATLAB®-notation this can be expressed as: `A(i:j,col)= s*A(i:j,col)`. The function should have the following header:

```
void scale_col(double s,struct my_matrix_st A, int i, int j, int col);
```

b.) Consider the rank-1 update of the lower right block of a matrix $A \in \mathbb{R}^{n \times n}$:

$$A(i:n, i:n) = A(i:n, i:n) + s \cdot vw^T,$$

with $s \in \mathbb{R}$ and $v, w \in \mathbb{R}^{(n-i+1)}$. Implement the C function

```
void r1_update(struct my_matrix_st A, int i, double s, double *v, int
                    incv, double *w, int incw);
```

which performs this update. The arguments `incv` and `incw` are the strides for accessing the vectors v and w. For example: The k-th element of the vector v is stored in `v[k*incv]`.

c.) Demonstrate both functions in a `main` program.

Hint 1: The skeleton frameworks from the last homework sheet are good starting points.

Hint 2: Take care of the leading dimension of the matrix A to get a flexible implementation.

Overall Points: 20