Winterterm 2012/2013

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

Dr. Jens Saak, Dipl.-Math. Martin Köhler
Website: http://www.mpi-magdeburg.mpg.de/mpcsc/lehre/2012_WS_SC/

Scientific Computing 1
11th Homework

Handout: 12/13/2012 **Return:** 12/20/2012

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. For example if you only need a single driver routine from it.

a.) Search on http://www.netlib.org/lapack/double/ for a driver which solves a general real doble precision linear system and download it with its dependencies.

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

c.) Write a small C program which uses this library to solve the linear system $Ax = b$ with

$$A = \begin{pmatrix} 1 & 7 & -2 \\ -4 & 2 & 0.3 \\ 3.5 & 0 & -8 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 6 \\ -1.7 \\ -4.5 \end{pmatrix}.$$

Hint: Subroutines from BLAS are not included as dependencies in the download. That means BLAS needs to be linked separately to the program.

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 Homework 10, Exercise 3):

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

Implement the following operations on such a matrix:

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:

```c
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

```c
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 framework from the last homework is a good starting point.

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

Overall Points: 20