Introduction to Mathematical Software 5th Exercise Sheet

Department of Mathematics
PD Dr. Ulf Lorenz
Christian Brandenburg

winter term 2009/2010
30/11/2009

Important Notice

Before starting this exercise sheet, you should at least have completed exercises **2.1**, **2.2**, **3.1**, **3.2**, and **4.1**.

Exercise 5.1 Decimal Expansion of Rational Numbers

Let two integers a, b be given with $b > 0$. Write a program that prints the rational number $\frac{a}{b}$ to the screen in *decimal expansion*.

Browse the internet for a suitable algorithm to accomplish this task.

The output might be finite or periodic and should be of the form `0.75` for $a = 3$, $b = 4$ in the finite case and `0.1p6` for $a = 1$, $b = 6$ in the periodic case, where the `p` denotes the beginning of the period.

Exercise 5.2 Matrix-Vector Multiplication

Write a function that computes matrix-vector products. Your program should dynamically allocate the memory needed for the matrices and vectors.

You have two possibilities to represent matrices:

- a two-dimensional array of type `double**` (i.e. an array of pointers to arrays of type `double`) with the first index being the row index and the second index being the column index. Your function for computing the matrix vector product $A \cdot x = y$ should be of the form
  `void MatMult(double** A, double* x, double* y, int rows, int cols);`
  where `m` denotes the number of rows of A (and y) and `n` denotes the number of columns of A (and rows of x).

- a one-dimensional array of type `double*` that contains all matrix rows one after the other in a single long array. The different rows can then be accessed with offsets. Your function for computing the matrix vector product $A \cdot x = y$ should be of the form
  `void MatMult(double* A, double* x, double* y, int rows, int cols);`
  with the variables defined as above.

You can proceed in the following way:

- Ask the user for the matrix dimensions (number of rows, columns).
- dynamically allocate the required memory (you need to `#include <stdlib.h>`).
- fill the matrix A and the vector x with random values or ask the user for the data. The statement `(double)rand()/(double)RAND_MAX` produces "random" numbers between 0 and 1.
- compute the matrix-vector product
- free the memory that you have allocated.
- you might want to add functions like
  `void print_matrix(double **A, unsigned int rows, unsigned int cols);`
  (or `void print_matrix(double *A, unsigned int rows, unsigned int cols);`)
  and

```
void print_vector(double *x, unsigned int rows);
```

to visualize your results.