

Technical University of Denmark

Written examination, December 10, 2019

Page 1 of 10 pages

Course title:	Mathematical Software Programming
---------------	-----------------------------------

Course number:	02635
----------------	-------

Aids allowed:	All aids allowed
---------------	------------------

Exam duration:	4 hours
----------------	---------

Weighting:	80/100
------------	--------

Final exam
Mathematical Software Programming

This exam contains a total of 16 questions: 12 multiple choice questions (questions 1–12) and 4 programming questions (questions 13–16). Your exam answers to the multiple choice questions must be submitted electronically as a **PDF document**, and your code must be attached (e.g., as a **ZIP file**).

1. (4 points) The unit round-off for double precision floating point numbers is $u = 2^{-53}$.
- (a) The unit round-off u can be represented as a double precision floating point number.
- A. True
B. False
- (b) The number $1 + u$ can be represented as a double precision floating point number.
- A. True
B. False
2. (4 points) Consider the real-valued function $f(x) = ax^2 + bx + c$ where $a, b, c, x \in \mathbb{R}$.
- (a) What is the relative condition number of f ?
- A. $c_f(x) = \frac{|ax^2+bx+c|}{|2ax+b|}$
B. $c_f(x) = \frac{|2ax+b|}{|ax^2+bx+c|}$
C. $c_f(x) = \frac{|2ax^2+bx|}{|ax^2+bx+c|}$
D. $c_f(x) = \frac{|ax^2+bx+c|}{|2ax^2+bx|}$
- (b) Suppose $a = 1$, $b = 4$, and $c = 4$. For what values of x is the problem of evaluating $f(x)$ ill-posed?
- A. When $x = 0$.
B. When $x = -1$.
C. When $x = -2$.
D. When $x \in \{-2, 0\}$.

3. (6 points) Let $f(x) = \log(1+x)$ where x is a positive real number.

(a) The relative condition number of f is given by

A. $c_f(x) = \left| \frac{1}{x \log(1+x)} \right|$

B. $c_f(x) = \left| \frac{1}{(1+x) \log(1+x)} \right|$

C. $c_f(x) = \left| \frac{1+x}{x \log(1+x)} \right|$

D. $c_f(x) = \left| \frac{x}{(1+x) \log(1+x)} \right|$

(b) What can be said about the relative condition number of f when $|x|$ is close to 0?

A. $c_f(x)$ is close to 0

B. $c_f(x)$ is close to 1

C. $c_f(x)$ is much larger than 1

(c) Consider the following implementation of the function $f(x) = \log(1+x)$:

```
#include <math.h>
double fun(double x) {
    return log(1.0+x);
}
```

This implementation may result in a large relative error when $|x|$ is close to 0. Why?

A. The condition number of $\log(x)$ is large when x is close to 1.

B. The sum $1+x$ is prone to catastrophic cancellation when $|x|$ is close to 0.

C. The sum $1+x$ may underflow when $|x|$ is close to 0.

4. (4 points) The Chebyshev polynomials of the first kind can be defined recursively as

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x), \quad k = 2, 3, \dots$$

- (a) What kind of recursion is this?

A. single recursion

B. multiple recursion

- (b) What is the complexity associated with the following implementation?

```
double T(double x, unsigned int k) {  
    if (k==0) return 1.0;  
    if (k==1) return x;  
    return 2*x*T(x,k-1)-T(x,k-2);  
}
```

A. $O(k)$ space and $O(k)$ time

B. $O(2^k)$ space and $O(k)$ time

C. $O(k)$ space and $O(2^k)$ time

D. $O(2^k)$ space and $O(2^k)$ time

5. (4 points) Suppose the variable **A** represents a two-dimensional array of size $m \times n$. Consider the following code segment:

```
for (int i=0; i<m; i++) {  
    for (int j=0; j<n; j++) {  
        y[i] += A[i][j]*x[j];  
    }  
}
```

- (a) References to **y[i]** are temporally local.

A. True

B. False

- (b) References to the array represented by **x** are spatially local.

A. True

B. False

6. (6 points) The theoretical improvement in speed of execution of a task executed on p processors can be expressed as

$$S(p) = \frac{T(1)}{T(p)} = \frac{fT(1) + (1-f)T(1)}{(f/p)T(1) + (1-f)T(1)}$$

where $T(p)$ is the execution time on p processors (real time) and f is the so-called parallel fraction of the task. For example, if 50% of a task can be parallelized, then $f = 0.5$.

Suppose that a specific task consists of two subtasks. The first subtask retrieves data from a file; it takes 20 seconds to execute on a given system, and it cannot be parallelized. The second subtask processes a large number of data records; it takes 60 seconds to execute sequentially. Each data record can be processed independently, so the second subtask is easily parallelized.

- (a) What is the parallel fraction of the task consisting of the two subtasks?

- A. $f = 1/4$
- B. $f = 1/3$
- C. $f = 3/4$
- D. $f = 4/5$

- (b) What is the theoretical speedup for $p = 9$ processors?

- A. 3
- B. 4
- C. 6
- D. 9

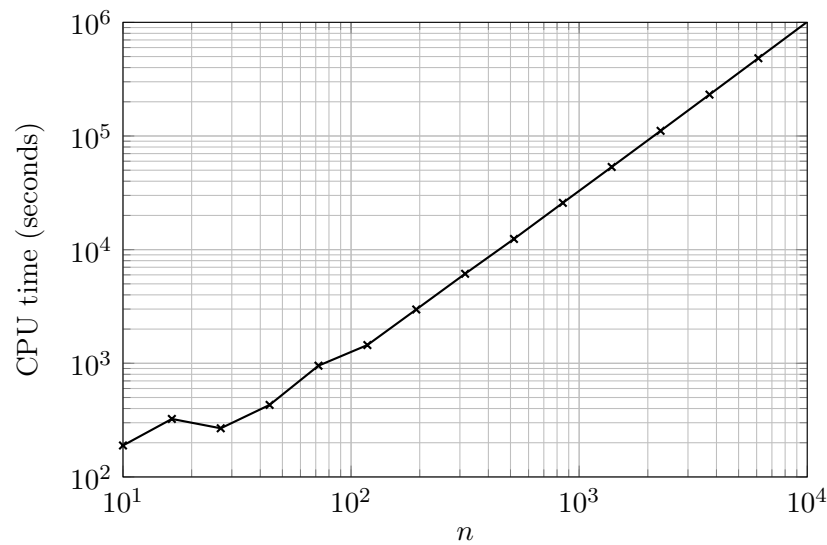
7. (2 points) Memory leaks are possible in C but not in C++.

- A. True
- B. False

8. (2 points) The operator `::` in C++ (e.g. `std::cout`) is referred to as the

- A. member access operator
- B. scope resolution operator
- C. namespace operator
- D. function operator

9. (4 points) The following plot shows the CPU time required by some algorithm to solve a certain problem as a function of its dimension n .



What is the time complexity of the algorithm?

- A. $O(n)$
 - B. $O(n^{3/2})$
 - C. $O(n^2)$
 - D. $O(2^n)$
10. (4 points) Consider the following function:

```
double * ones(size_t n) {  
    double *arr = malloc(n*sizeof(*arr));  
    if (arr != NULL) {  
        for (size_t k=0; k<n; k++)  
            arr[k] = 1.0;  
    }  
    return arr;  
}
```

- (a) What kind of memory allocation is used to allocate what `arr` points to?
- A. Automatic memory allocation
 - B. Dynamic memory allocation
- (b) What kind of memory allocation is used to allocate the pointer `arr`?
- A. Automatic memory allocation
 - B. Dynamic memory allocation

11. (4 points) Suppose the matrix

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \end{bmatrix}$$

is stored as a *one-dimensional, row-major* **double** array of length 12.

The elements of an array of double-precision floating-point numbers can be scaled using the CBLAS routine `cblas_dscal()` which has the following prototype:

```
void cblas_dscal(  
    const int n,           // number of elements  
    const double a,        // scalar a  
    double * x,            // pointer to first element of array  
    const int incx         // stride  
);
```

Now suppose `pA` is a pointer to A_{11} (i.e., `pA` is a **double ***).

(a) Which of the following calls to `cblas_dscal()` scales the elements A_{22} , A_{23} , and A_{24} by -1 ?

- A. `cblas_dscal(3, -1.0, pA+4, 1)`
- B. `cblas_dscal(3, -1.0, pA+4, 3)`
- C. `cblas_dscal(3, -1.0, pA+4, 4)`
- D. `cblas_dscal(3, -1.0, pA+5, 1)`
- E. `cblas_dscal(3, -1.0, pA+5, 3)`
- F. `cblas_dscal(3, -1.0, pA+5, 4)`

(b) Consider the following `cblas_dscal()` call:

```
cblas_dscal(4, 2.0, pA+1, 2);
```

Which elements of A are being scaled by a factor of 2?

- A. $A_{11}, A_{13}, A_{21}, A_{23}$
- B. $A_{12}, A_{14}, A_{22}, A_{24}$
- C. $A_{21}, A_{12}, A_{32}, A_{23}$
- D. $A_{11}, A_{31}, A_{22}, A_{13}$

12. (2 points) Cache memory is slower than main memory.

- A. True
- B. False

13. (8 points) Implement a function that performs the matrix-vector multiplication

$$y \leftarrow A^T x + y$$

where A is a sparse matrix of size $m \times n$ in triplet form, and $x \in \mathbb{R}^m$ and $y \in \mathbb{R}^n$ are vectors.

Requirements

- Your function must have the following prototype:

```
void sparse_triplet_mv_trans(  
    const struct sparse_triplet *A,  
    const double *x,  
    double *y);
```

You may assume that all inputs are valid.

- The sparse matrix A should be stored using the sparse triplet format represented by the following data structure:

```
/* Structure representing a sparse matrix in triplet form */  
struct sparse_triplet {  
    size_t m;    /* number of rows */  
    size_t n;    /* number of columns */  
    size_t nnz;  /* number of nonzeros */  
    size_t * I;  /* pointer to array with row indices */  
    size_t * J;  /* pointer to array with column indices */  
    double * V;  /* pointer to array with values */  
};
```

The header file `sparse_triplet.h` in the ZIP file attached to the exam defines the `sparse_triplet` data structure. You do not need to include this header file when you submit your solution.

- Use the template `sparse_triplet_mv_trans.c` for your implementation. The template is included in the ZIP file attached to the exam.

14. (8 points) The Poisson distribution is a discrete probability distribution with probability mass function

$$f(k; \lambda) = \frac{\lambda^k}{k!} e^{-\lambda}, \quad k \in \mathbb{N}_0,$$

where the parameter $\lambda > 0$ is the so-called rate. Implement a function that evaluates $f(k; \lambda)$.

Requirements

- Your function must have the following prototype:

```
double poisson_pmf(unsigned long k, double lambda);
```

- The function should return NAN if λ is not positive.
- Use the template `poisson_pmf.c` for your implementation. The template is included in the ZIP file attached to the exam.

15. (8 points) The centering matrix of order n is given by

$$C = I - \frac{1}{n} \mathbf{1} \mathbf{1}^T$$

where I is the $n \times n$ identity matrix and $\mathbf{1}$ is the vector of length n whose entries are all equal to 1. For example, the centering matrix of order 3 is the matrix

$$C = \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} \end{bmatrix}.$$

Implement a function that computes $x \leftarrow Cx$, i.e., the function should overwrite a vector $x \in \mathbb{R}^n$ by the matrix-vector product Cx .

Requirements

- The function must have the following prototype:

```
int dcmv(int n, double * x);
```

The input `n` is the order of the matrix C , and `x` is a pointer to the first elements of an array of length `n` that represent the vector x .

- The function should return the value -1 in case of an error or invalid input, and otherwise it should return the value 0.
- Use the template `dcmv.c` for your implementation. The template is included in the ZIP file attached to the exam.

16. (10 points) Consider the system of equations

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix} \quad (1)$$

where $r \geq 0$, $c = \cos(\theta)$, and $s = \sin(\theta)$ for some $\theta \in [0, 2\pi]$. It is easy to check that a, b must satisfy

$$\begin{bmatrix} a \\ b \end{bmatrix} = r \begin{bmatrix} c \\ -s \end{bmatrix},$$

and hence $r = \sqrt{a^2 + b^2}$. It follows that if $r > 0$, then $c = a/r$ and $s = -b/r$ (the angle θ can be chosen arbitrarily when $r = 0$, e.g., $c = 1$ and $s = 0$).

- (a) Implement a function that takes a and b as inputs and computes c and s such that a and b satisfies (1).

Requirements

- Your function must have the following prototype:

```
void rotg(double a, double b, double * c, double * s);
```

The inputs c and s point to the locations where c and s should be stored. You may assume that all inputs are valid.

- Use the template `rotg.c` for your implementation. The template is included in the ZIP file attached to the exam.

- (b) Implement a function that applies the transformation

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} \leftarrow \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}, \quad i = 1, \dots, n,$$

where $x = (x_1, \dots, x_n)$ and $y = (y_1, \dots, y_n)$ are vectors of length n , and $c = \cos(\theta)$ and $s = \sin(\theta)$ for some $\theta \in [0, 2\pi]$.

Requirements

- Your function must have the following prototype:

```
int rot(double c, double s, int n, double * x, double * y);
```

The input n is the length of the vectors x and y . The inputs x and y are pointers to the first element of the arrays that represent x and y , respectively.

- The function should return the value -1 in case of an invalid input, and otherwise it should return the value 0 .
- Use the template `rot.c` for your implementation. The template is included in the ZIP file attached to the exam.