Technical University of Denmark

Technical University of Denmark

Written examination, December 11, 2018

Page 1 of 11 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 14 questions: 11 multiple choice questions (questions 1–11) and 3 programming questions (questions 12–14). Your exam answers to the multiple choice questions must be submitted electronically as a **PDF document**, and you code must be attached (e.g., as a **ZIP file**).

---

1. (4 points) Properties of floating-point arithmetic in finite precision.

   (a) The commutative property of multiplication holds for floating-point arithmetic, *i.e.*,

   $$ab = ba$$

   where a, b, and c are floating-point numbers with a given precision.

   A. True
   B. False

   (b) The distributive property of addition holds for floating-point arithmetic, *i.e.*,

   $$(a + b) + c = a + (b + c)$$

   where a, b, and c are floating-point numbers with a given precision.

   A. True
   B. False

2. (4 points) A finite-precision flointing-point number in base b can be expressed as

   $$x = s \cdot (d_0.d_1 \cdots d_{p-1})_b \cdot b^E$$

   where s represents the sign, $d_0, \ldots, d_{p-1} \in \{0, \ldots, b-1\}$ are the p digits, and $E \in \{E_{\min}, E_{\max}\}$ is the exponent.

   Consider the following two floating-point formats:

   - binary64 (the data type `double` in C): $p = 53$, $b = 2$, $E \in \{-1022, \ldots, 1023\}$
   - decimal64: $p = 16$, $b = 10$, $E \in \{-383, \ldots, 384\}$

   Both formats require 8 bytes (64 bits) to store a number. Which of the two formats has the *smallest* machine epsilon?

   A. binary64
   B. decimal64

3. (4 points) Condition number.

   (a) What is the condition number of $f(x) = \exp(x) - 1$?

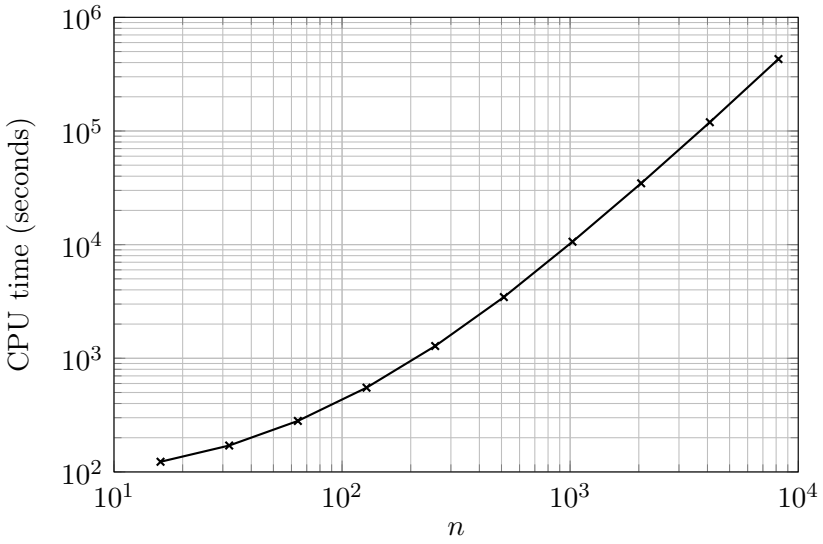
   A. $1/|1 - \exp(x)|$
   B. $1/|1 - \exp(-x)|$
   C. $|x|/|1 - \exp(x)|$
   D. $|x|/|1 - \exp(-x)|$

   (b) When does the following implementation of $f(x) = \exp(x) - 1$ suffer from catastrophic cancellation?

```
double fval(double x) {
    return exp(x)-1.0;
}
```

   A. When $|x|$ is large.
   B. When x is close to -1.
   C. When x is close to 0.
   D. When x is close to 1.

4. (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. $O(n^2)$
C. $O(n^3)$
D. $O(2^n)$

5. (4 points) Suppose `pA` is a `double **` that points to a two-dimensional, dynamically allocated array of size $m \times n$.

   (a) What is the type of `pA[0]`?

   A. `double`
   B. `void *`
   C. `double *`
   D. `double **`

   (b) With zero-based indexing, which of the following expressions represents the value of the jth element of row i?

   A. `&pA[i][j]`
   B. `(*pA+i)+j`
   C. `*(pA+i)+j`
   D. `*(*(pA+i)+j)`

6. (6 points) Consider the following C++ snippet:

```
std::vector<double> v;
for (int i = 1; i <= 5; i++)
    v.push_back(2.0*i);
std::cout << "Size : " << v.size() << std::endl;
std::cout << "Capacity : " << v.capacity();
```

   (a) What is the variable `v`?

   A. a class
   B. a class template
   C. a method
   D. an object

   (b) What is `std::vector`?

   A. a class
   B. a class template
   C. a method
   D. an object

   (c) What is `std::vector::push_back`?

   A. a class
   B. a class template
   C. a method
   D. an object

7. (4 points) To represent an array of n double-precision floating-point numbers, a programmer defined the following data structure:

```
typedef struct darray {
    unsigned long n;   // length of array
    double * v;        // pointer to first element of array
} darray_t;
```

The programmer also implemented a function `malloc_darray()` for dynamically allocating a `darray_t` data structure:

```
#include <stdlib.h>
darray_t * malloc_darray(unsigned long n) {
    darray_t arr;
    arr.n = n;
    arr.v = malloc(n*sizeof(double));
    if (arr.v == NULL)
        return NULL;
    else
        return &arr;
}
```

There is a problem with this implementation. What is the problem?

A. The implementation leaks memory.

B. The implementation allocates too much memory.

C. The implementation returns a pointer to a local variable.

D. The implementation may lead to catastrophic cancellation.

8. (6 points) A real, univariate polynomial $p(x)$ of degree n with roots $a_1, \ldots, a_n$ can be expressed in factored form as

$$p(x) = \prod_{i=1}^{n} (x - a_i).$$

It is easy to verify that if we define $p_0(x) = 1$ and

$$p_k(x) = (x - a_k)p_{k-1}(x), \quad k \geq 1,$$

then $p(x) = p_n(x)$. In other words, $p(x)$ can be computed recursively given the n roots of the polynomial.

(a) Consider the following implementation of a function that evaluates $p(x)$ using recursion:

```
/* Evaluates polynomial in factored form */
double fp_eval(unsigned int n, double *a, double x)
{
    if (n>=1)
        return (x-a[n-1])*fp_eval(n-1,a,x);
    else
        return 1.0;
}
```

The input `a` points to the first element of an array with the n coefficients $a_1, \ldots, a_n$. What is the time complexity associated with the function `fp_eval()`?

A. $O(n)$
B. $O(n^2)$
C. $O(n^3)$
D. $O(2^n)$

What is the space complexity associated with the function `fp_eval()`?

A. $O(n)$
B. $O(n^2)$
C. $O(n^3)$
D. $O(2^n)$

(b) The derivative of the polynomial $p(x)$ can be expressed as

$$p'(x) = p'_n(x) = p_{n-1}(x) + (x - a_n)p'_{n-1}(x)$$

which follows by applying the product rule to the recursive definition of $p_n(x)$. Consider the following implementation of a function that evaluates the derivative of $p(x)$ using recursion:

```
/* Evaluates derivative of polynomial in factored form */
double fp_deriv_eval(unsigned int n, double *a, double x)
{
    double fval, dval;
    if (n>=1) {
        fval = fp_eval(n-1,a,x);
        dval = fp_deriv_eval(n-1,a,x);
        return fval + (x-a[n-1])*dval;
    }
    else
        return 0.0;
}
```

What is the time complexity associated with the function `fp_deriv_eval()`?

A. $O(n)$
B. $O(n^2)$
C. $O(n^3)$
D. $O(2^n)$

What is the space complexity associated with the function `fp_deriv_eval()`?

A. $O(n)$
B. $O(n^2)$
C. $O(n^3)$
D. $O(2^n)$

9. (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 $f = 0.5$.

Consider the following code snippet that is a task from a program that (i) loads data from a file into a two-dimensional, dynamically allocated row-major array represented by a pointer `X`, and (ii) performs some data processing row by row:

```
/* Subtask (i): Load data from file */
int n=0;
double **X=NULL;
FILE * fp = open("measurements.dat","r");
if (fp == NULL) exit(EXIT_FAILURE);
read_array2d(fp, &n, &X);
if (X == NULL) exit(EXIT_FAILURE);

/* Subtask (ii): process rows */
#pragma omp parallel for
for (int k=0;k<n;k++) {
    process_row(X[k]);
}
```

(a) Suppose that the first subtask (loading data from a file) is inherently serial and takes 12 seconds for a given data set. Furthermore, suppose that the second subtask (processing each row of the array) can be parallelized and takes 20 seconds when executed serially. What is the parallel fraction of the task consisting of the two subtasks?

A. 0.375
B. 0.60
C. 0.625
D. 0.80

(b) What is approximately the theoretical speedup for $p = 4$ processors?

A. 1.39
B. 1.81
C. 1.88
D. 2.5

10. (6 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*, *column-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. $A_{11}, A_{31}, A_{22}, A_{13}$

11. (4 points) The determinant of a 2×2 matrix A can be expressed as

$$|A| = \left| \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \right| = A_{11}A_{22} - A_{12}A_{21}.$$

The matrix A is singular if and only if $|A| = 0$.

Suppose that A is given by

$$A = \begin{bmatrix} 2.0 & -0.2 \\ 9.9 & 1.0 \end{bmatrix}.$$

What is approximately the worst-case relative error if we compute the determinant of this matrix in floating-point arithmetic with unit round-off u using the expression $A_{11}A_{22} - A_{12}A_{21}$?

A. u
B. $2u$
C. $10u$
D. $20u$

12. (8 points) The so-called *softplus* function is defined as

$$f(x) = \log(1 + \exp(x)).$$

Implement a function that evaluates the derivative of the softplus function.

Your function must have the following prototype:

```
double softplus_deriv(double x);
```

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

13. (8 points) Write a function that computes the geometric mean of n non-negative numbers $x = (x_1, \ldots, x_n)$, i.e.,

$$f(x) = \left(\prod_{i=1}^{n} x_i \right)^{1/n}.$$

Your function must have the following prototype:

```
double geo_mean(unsigned int n, const double * x);
```

The input `x` points to the first element of an array with the n numbers $x_1, \ldots, x_n$.

Requirements

- The function should return `NAN` if any of the numbers $x_1, \ldots, x_n$ is negative.
- Use the template `geo_mean.c` for your implementation. The template is included in the ZIP file attached to the exam.

14. (12 points) Let J be a diagonal matrix of size $n \times n$ defined as

$$J = \begin{bmatrix} 1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{bmatrix}$$

for $n \geq 2$. In other words, J may be obtained from the identity matrix of order $n \geq 2$ by changing the sign of its last $n-1$ diagonal element. Given a column vector $u \in \mathbb{R}^n$ for which $u^T Ju = u_1^2 - \sum_{i=2}^n u_i^2 \neq 0$, we may define the matrix

$$T = J - 2\frac{uu^T}{u^T Ju}$$

which is a so-called *hyperbolic Householder* matrix. The requirement that $u^T Ju \neq 0$ ensures that the matrix T is well-defined and nonsingular. The inverse of T can then be expressed as

$$T^{-1} = JTJ = J - 2\frac{Juu^T J}{u^T Ju}.$$

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

Requirements

- The function must have the following prototype:

```
int dhhmv(int n, const double * u, double * x);
```

  The input `n` is the order of the matrix T, and `u` and `x` are pointers to the first elements of two arrays of length `n` that represent the vectors u and x, respectively.
- The function should return the value `-1` in case of an error, and otherwise it should return the value `0`.
- Use the template `dhhmv.c` for your implementation. The template is included in the ZIP file attached to the exam.

(b) Write a function that computes $x \leftarrow T^{-1}x$, i.e., the function should overwrite a vector $x \in \mathbb{R}^n$ by the matrix-vector product $T^{-1}x$.

Requirements

- The function must have the following prototype:

```
int dhhsv(int n, const double * u, double * x);
```

  The input `n` is the order of the matrix T, and `u` and `x` are pointers to the first elements of two arrays of length `n` that represent the vectors u and x, respectively.
- The function should return the value `-1` in case of an error, and otherwise it should return the value `0`.
- Use the template `dhhsv.c` in the ZIP file attached to the exam for your implementation.