Danmarks Tekniske Universitet

Danmarks Tekniske Universitet

Skriftlig prøve/dato: / Written examination date:	9. December 2025
Side 1 af 4 sider / Page 1 of 4 pages	
Kursus navn: / Course title:	Mathematical Software Programming
Kursus nummer: / Course number:	02635
Hjælpemidler: / Aids allowed:	All aids allowed.
Varighed: / Exam duration:	4 hours
Vægtning: / Weighting:	Part 1: 35/75 Part 2: 40/75

Please note that the exam consists of two parts: part 1 is a set of multiple-choice questions, and part 2 is a set of programming questions.

This is part 2 of the exam. Use the templates in the ZIP file for your answers.

Question 1 (10 points)

Consider the real-valued function

$$f(x) = \ln(1 - \exp(-x)), \quad x > 0.$$

Implement a function that takes x as input and returns $f(x)$.

The function must have the following prototype:

```
double log1mexp(double x);
```

- The function should return `NAN` if `x` is negative or `NAN`.
- The function should return `-INF` if `x` is zero.

Use the template `exam_e25_q1.c` for your implementation.

Question 2 (10 points)

Consider the polynomial

$$p(x) = \sum_{k=0}^{n} c_k P_k(x), \quad x \in [-1,1],$$

where $(c_0, c_1, \ldots, c_n)$ are $n+1$ real-valued coefficients and $P_k(x)$ is the kth Legendre polynomial, which may be defined recursively as

$$P_0(x) = 1, \qquad P_1(x) = x, \qquad (k+1)P_{k+1}(x) = (2k+1)xP_k(x) - kP_{k-1}(x), \;\; k \geq 1.$$

Write a function that evaluates $p(x)$ given x and the coefficients $(c_0, c_1, \ldots, c_n)$. The function must have the following prototype:

```
double legendreseries(double x, const double *c, int n);
```

The input `c` is a pointer to an array of length $n+1$ containing the coefficients $(c_0, c_1, \ldots, c_n)$, and the input `n` is the degree of the polynomial $p(x)$.

The function should return `NAN` if `c` is `NULL` or if $x \notin [-1,1]$ or if `n` is negative.

Use the template `exam_e25_q2.c` for your implementation.

Question 3 (20 points)

Given vectors $a = (a_1, \dots, a_n)$ and $b = (b_1, \dots, b_n)$ of length n, we define a matrix

$$M = \mathrm{diag}(a) + bb^T$$

where $\mathrm{diag}(a)$ is the diagonal matrix with the entries of a on its main diagonal (and zeros elsewhere) and bb^T denotes an outer product, i.e.,

$$\mathrm{diag}(a) = \begin{bmatrix} a_1 & & & \\ & a_2 & & \\ & & \ddots & \\ & & & a_n \end{bmatrix}, \qquad bb^T = \begin{bmatrix} b_1 b_1 & b_1 b_2 & \cdots & b_1 b_n \\ b_2 b_1 & b_2 b_2 & \cdots & b_2 b_n \\ \vdots & \vdots & \ddots & \vdots \\ b_n b_1 & b_n b_2 & \cdots & b_n b_n \end{bmatrix}.$$

Part A

Implement a function that takes vectors a, b, and x and computes $x \leftarrow Mx$, i.e., it should overwrite the vector x by Mx.

The function must have the following prototype:

```
int dsr1mv(int n, const double * a, const double * b, double * x);
```

- The inputs `a`, `b`, and `x` are pointers to the first entry of arrays of length `n`, representing a, b, and x, respectively.
- The function should return `-1` if one or more input arugements are invalid (e.g., if `n` is negative or if a pointer is `NULL`), and otherwise the function should return `0`.

Use the template `exam_e25_q3a.c` for your implementation.

Part B

We will now require that $a_i > 0$ for $i \in \{1, \dots, n\}$, which implies that M is nonsingular, and its inverse can be expressed as

$$M^{-1} = \mathrm{diag}(a)^{-1} - \mathrm{diag}(a)^{-1} b (1 + b^T \mathrm{diag}(a) b)^{-1} b^T \mathrm{diag}(a)^{-1}.$$

Implement a function that takes vectors a, b, and x and computes $x \leftarrow M^{-1}x$, i.e., it should overwrite the vector x by $M^{-1}x$.

The function must have the following prototype:

```
int dsr1sv(int n, const double * a, const double * b, double * x);
```

- The inputs `a`, `b`, and `x` are pointers to the first entry of arrays of length `n`, representing a, b, and x, respectively.
- The function should return `-1` if one or more input arugements are invalid (e.g., if `n` is negative, if a pointer is `NULL`, or if a does not have positive entries), and otherwise the function should return `0`.

Use the template `exam_e25_q3b.c` for your implementation.