Danmarks Tekniske Universitet

Danmarks Tekniske Universitet

Prøveeksamen / Trial exam:	Open 1-6 December 2021
Side 1 af 6 sider / Page 1 of 6 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: 42/100 Part 2: 38/100

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

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 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.

Question 2

The Poisson distribution is a discrete probability distribution with probability mass function

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

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.

Question 3

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 dcemv(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 `dcemv.c` for your implementation. The template is included in the ZIP file.

Question 4

Consider the system of equations

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

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 $s = 0$).

Part A

Implement a function that takes a and b as inputs and computes c and s such that a and b satisfy the system of equations.

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.

(Continues on the next page.)

Part B

Implement a function that applies the transformation

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

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 `-1`.

- Use the template `rot.c` for your implementation. The template is included in the ZIP file.