# NumCSE Mock Exam, HS 2018

1. *Estimating point locations from distances* [14 pts.]

   Let $n \in \mathbb{N}$, $n > 2$ points be located on the real axis. The leftmost point is fixed at origin, whereas the other points are at unknown locations:

$$x_i \in \mathbb{R}, \quad \text{for } i = 1, 2, \dots, n,$$
$$x_{i+1} > x_i,$$
$$x_1 = 0.$$

   The distances $d_{i,j} := |x_i - x_j|$, $\forall i, j \in \{1, 2, \dots, n\}$, $i > j$ are measured and arranged in a vector

$$\mathbf{d} := [d_{2,1}, d_{3,1}, \dots, d_{n,1}, d_{3,2}, d_{4,2}, \dots, d_{n,n-1}]^\top \in \mathbb{R}^m,$$

   where $m = n(n-1)/2$. Assume that there are no measurement errors.

   Some templates are provided in `estimatePositions.cpp`, write your code in the template coresponding to the instructions in the tasks below.

   (a) To determine the unknown point locations using the distance measurements, formulate a linear least squares problem

$$\mathbf{z}^* = \underset{\mathbf{z} \in \mathbb{R}^{n-1}}{\arg\min} \|\mathbf{A}\mathbf{z} - \mathbf{b}\|. \tag{1}$$

   [2 pts.]

   (b) Write an EIGEN based C++ implementation

```
using namespace Eigen;

SparseMatrix<double> buildDistanceLSQMatrix(int n);
```

   which initializes the system matrix $\mathbf{A}$ from (1) in an efficient manner for large $n$. [2 pts.]

   (c) Give explicit formulas for the entries of the system matrix $\mathbf{M}$ of the *normal equations* corresponding to the system (1). [2 pts.]

   (d) Show that the system matrix $\mathbf{M}$ obtained in the previous step can be written as a rank-1 perturbation of a diagonal matrix. [2 pts.]

   (e) Write an EIGEN based C++ implementation

```
VectorXd estimatePointsPositions(const MatrixXd& D);
```

   which solves the linear least squares problem (1) using normal equations method. Here $\mathbf{D} \in \mathbb{R}^{n \times n}$

$$(\mathbf{D})_{i,j} = \begin{cases} d_{i,j} & \text{if } i > j, \\ 0 & \text{if } i = j, \\ -d_{i,j} & \text{if } i < j. \end{cases}$$

   Use the observations from the previous step. [5 pts.]

   (f) What is the asymptotic complexity of the function `estimatePointPositions` implemented in subproblem (e) for $n \to \infty$? [1 pt.]

2. *Solving an eigenvalue problem with Newton method* [14 pts.]

Given a symmetric positive-definite matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, solving

$$\begin{aligned} \mathbf{F}(\mathbf{z}) = \mathbf{0}, \quad \mathbf{z} = (\mathbf{x}, \lambda)^\top, \\ \text{for} \quad \mathbf{F}(\mathbf{z}) := \begin{pmatrix} \mathbf{A}\mathbf{x} - \lambda \mathbf{x} \\ 1 - \frac{1}{2}\|\mathbf{x}\|^2 \end{pmatrix}, \end{aligned} \tag{2}$$

is equivalent to finding an eigenvector $\mathbf{x}$ and associated eigenvalue $\lambda$ for $\mathbf{A}$.

Therefore, a possible numerical method for computing one eigenvalue/eigenvector of $\mathbf{A}$ is the application of Newton's method to find a zero of the vector-valued function $\mathbf{F} : \mathbb{R}^{n+1} \to \mathbb{R}^{n+1}$ defined in (6).

(a) Compute the Jacobian of $\mathbf{F}$ at $\begin{pmatrix} \mathbf{x} \\ \lambda \end{pmatrix} \in \mathbb{R}^{n+1}$. [2 pts.]

(b) Devise an iteration of the Newton method to solve $\mathbf{F}(\mathbf{z}) = \mathbf{0}$. [4 pts.]

(c) Write an EIGEN based C++ implementation for the Newton method devised in the previous step:

```
void eigNewton(const MatrixXd &A, double atol, int
    maxItr, VectorXd &z);
```

which, given the matrix `A`, tolerance `tol` and initial guess `z`, returns the solution in `z`. [8 pts.]

*Hint:* For the initial guess: choose $\mathbf{x}$, then evaluate $\lambda = \dfrac{\mathbf{x}^\top \mathbf{A}\mathbf{x}}{\mathbf{x}^\top \mathbf{x}}$.

*Hint:* Test your code with some small matrix $\mathbf{A}$.

3. *Gauss-Legendre quadrature rule* [14 pts.]

An $n$-point quadrature formula on $[a,b]$ provides an approximation of the value of an integral through a *weighted sum* of point values of the integrand:

$$\int_a^b f(x)\,\mathrm{d}t \;\approx\; Q_n(f) := \sum_{j=1}^{n} w_j^n\, f(c_j^n), \tag{3}$$

where $w_j^n$ are called quadrature weights $\in \mathbb{R}$ and $c_j^n$ quadrature nodes $\in [a,b]$.

The order of a quadrature rule $Q_n : C^0([a,b]) \to \mathbb{R}$ is defined as the maximal degree+1 of polynomials for which the quadrature rule is guaranteed to be exact. It can also be shown that the maximal order of an $n$-point quadrature rule is $2n$. So the natural question to ask is if such a family $Q_n$ of $n$-point quadrature formulas exist where $Q_n$ is of order $2n$. If yes, how do we find the nodes corresponding to it?

Let us assume that there exists a family of $n$-point quadrature formulas on $[-1,1]$ of order $2n$, i.e.

$$Q_n(f) := \sum_{j=1}^{n} w_j^n f(c_j^n) \approx \int_{-1}^{1} f(t)\,\mathrm{d}t\,, \quad w_j \in \mathbb{R}\,,\ n \in \mathbb{N}\,, \tag{4}$$

and the above approximation is exact for polynomials $\in \mathcal{P}_{2n-1}$.

Define the $n$-degree polynomial

$$\bar{P}_n(t) := (t - c_1^n) \cdot \dots \cdot (t - c_n^n)\,, \quad t \in \mathbb{R}\,.$$

If we are able to obtain $\bar{P}_n(t)$, we can compute its roots numerically to obtain the nodes for the quadrature formula.

(a) For every $q \in \mathcal{P}_{n-1}$, verify that $\bar{P}_n(t) \perp q$ in $L^2([-1,1])$ i.e.

$$\int_{-1}^{1} q(t)\bar{P}_n(t)\,\mathrm{d}t = 0. \tag{5}$$

[2 pts.]

(b) Switching to a monomial representation of $\bar{P}_n$

$$\bar{P}_n = t^n + \alpha_{n-1}t^{n-1} + \dots + \alpha_1 t + \alpha_0\,,$$

derive

$$\sum_{j=0}^{n-1} \alpha_j \int_{-1}^{1} t^\ell t^j\,\mathrm{d}t = -\int_{-1}^{1} t^\ell t^n\,\mathrm{d}t \qquad \forall\, \ell = 0 \dots, n-1. \tag{6}$$

[3 pts.]

*Hint:* Use (9) with the monomials $1, t, \dots, t^{n-1}$ and with $\bar{P}_n$ in its monomial representation.

(c) Find expressions for $\mathbf{A}$ and $\mathbf{b}$ such that the coefficients of the monomial expansion can be obtained by solving a linear system of equation $\mathbf{A}[\alpha_j]_{j=0}^{n-1} = \mathbf{b}$. [3 pts.]

(d) Show that $[\alpha_j]_{j=0}^{n-1}$ exists and is unique. [3 pts.]

*Hint:* verify that $\mathbf{A}$ is symmetric positive definite.

(e) Use a 5-point Gauss quadrature rule to compare the exact solution and the quadrature approximation of

$$\int_{-3}^{3} e^t\,\mathrm{d}t.$$

The polynomial obtained in (d) and the Legendre-polynomial $P_n$ differ by a constant factor. Thus, the Gauss quadrature nodes $(\widehat{c_j})_{j=1}^5$ are also the zeros of the 5-th Legendre polynomial $P_5$. Here, we provide the zeros of $P_5$ for simplicity, but they should ideally be obtained by a numerical method for obtaining roots (e.g Newton-Raphson method). Thus,

$$(\widehat{c_j})_{j=1}^5 = [-0.9061798459, -0.5384693101, 0, 0.5384693101, 0.9061798459]$$

Recall from Theorem 6.3.1 (found in Week 9 Tablet notes - pg. 9) that the corresponding quadrature weights $\widehat{w}_j$ are given by:

$$\widehat{w}_j = \int_{-1}^{1} L_{j-1}(t)\,\mathrm{d}t, \quad j = 1, \ldots, n, \tag{7}$$

where $L_j$, $j = 0, \ldots, n-1$, is the $j$-th Lagrange polynomial associated with the ordered node set $\{\widehat{c_1}, \ldots, \widehat{c_n}\}$. [3 pts.]