

# Homework (B0B17MTB, BE0B17MTB)

## Problem Set 2

### 1 Assignment

Problem 2-A Consider matrix  $\mathbf{K} \in \mathbb{R}^{N \times 3}$ , containing  $\{x, y, z\}$  coordinates of  $N$  points. The matrix is generated as

```
c = 1:N;  
r = 1:3;  
K = toeplitz(c, r) - N/2;
```

where  $N$  can be chosen freely.

For a given matrix  $\mathbf{K}$ , calculate vector  $\mathbf{n} \in \mathbb{R}^{N \times 1}$  containing 1st norms (called also [Manhattan norm](#)) of all radius vectors pointing from the center of the coordinate system ( $x = 0, y = 0, z = 0$ ) to a corresponding point.

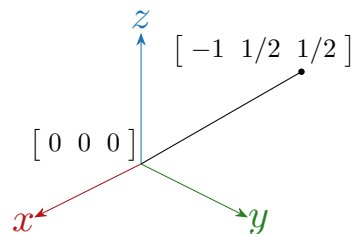


Figure A: Considering vector in this picture, the Manhattan norm is equal to 2.

Evaluate the ratio  $r$  between the biggest and the smallest value in the vector  $\mathbf{n}$ . (Check: for  $N = 6$ , we have  $r = \max\{\mathbf{n}\} / \min\{\mathbf{n}\} = 3$ .)

Implement the above-mentioned functionality into a function with the following header

```
function [n, r] = problem2A(N)
```

**(1 point)**

Problem 2-B Calculate vector  $\mathbf{d}$  containing distances between all consecutive prime numbers starting from 3 to  $N = 10^8$  (notice that they are all even numbers). Determine the longest distance  $d_{\max} = \max\{d\}$  and the associated prime numbers,  $p_1$  and  $p_2$ , for which it occurs. Finally, determine what distance value  $d_{\text{most}}$  occurs most often in the vector  $\mathbf{d}$  (so-called [the mode](#) of a sample  $\mathbf{d}$ ).

Implement the above-mentioned functionality into a function with the following header

```
function [dMax, p1, p2, dMost] = problem2B(N)
```

**(2 points)**

Problem 2-C Implement a function called `problem2C`, which evaluates Euclidean distances between two sets of points, finds a sphere with a center at the middle point between the two most distant points, and calculates its radius. Finally, verify if all points are inside this sphere.

Imagine two sets of points,  $\mathbf{p}_m \in \mathcal{P}$ ,  $m \in \{1, \dots, M\}$  and  $\mathbf{r}_n \in \mathcal{R}$ ,  $n \in \{1, \dots, N\}$ . Two matrices represent them,  $\mathbf{P} \in \mathbb{R}^{M \times 3}$  and  $\mathbf{R} \in \mathbb{R}^{N \times 3}$ , serving as the sole inputs

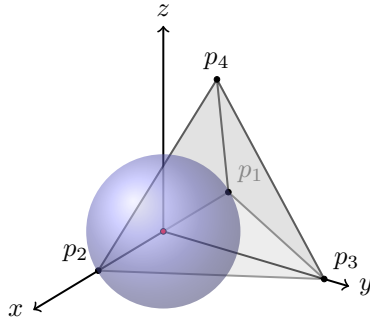


Figure C: An example of point set  $\mathbf{P} = \mathbf{R}$  forming a unitary tetrahedron. The distances between all  $m \neq n$  points is  $d_{mn} = 1$ . The radius of a sphere touching the most distant points is  $a = 1/2$  and its center non-unique, position  $\mathbf{c} = [0 \ 0 \ 0]$  shown here as red circle.

into the function. The function calculates Euclidean distance (2nd norm) between each pair of points, taken one by one from the sets  $\mathcal{P}$  and  $\mathcal{R}$ , as

$$d_{mn} = \|\mathbf{p}_m - \mathbf{r}_n\|, \quad \mathbf{D} = [d_{mn}] \in \mathbb{R}^{M \times N}. \quad (1)$$

The distance matrix  $\mathbf{D}$  is returned as the first output variable. Finally, the function evaluates the center  $\mathbf{c}$  of the sphere given as

$$\mathbf{c} = \frac{1}{2} (\mathbf{p}_{m_c} + \mathbf{r}_{n_c}) \quad (2)$$

with boundary points  $\mathbf{p}_{m_c}$  and  $\mathbf{r}_{n_c}$  found such that

$$m_c, n_c: \quad a = \frac{1}{2} \max_{m,n} \{\mathbf{D}\}, \quad (3)$$

*i.e.*, two points with the largest distance between them. Check at the end if all points from both sets are within this sphere and return `allPtsIn = true` if the answer is yes and `allPtsIn = false` if contrary is the case. To recap, the header of the function `problem2C` reads

```
function [D, a, c, allPtsIn] = problem2C(P, R)
```

For testing purposes, you may use an equilateral tetrahedron with unitary sides

$$\mathbf{P} = \mathbf{R} = \begin{bmatrix} -1/2 & 0 & 0 \\ 1/2 & 0 & 0 \\ 0 & \sqrt{3}/2 & 0 \\ 0 & \sqrt{3}/6 & \sqrt{2/3} \end{bmatrix} \quad (4)$$

with the results

$$\mathbf{D} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \quad (5)$$

$a = 1/2$ , and  $\mathbf{c} = [0 \ 0 \ 0]$ . Notice that the center point  $\mathbf{c}$  is, in general, not uniquely defined here; see Figure C. Any valid solution is therefore accepted.

A hint: Check out the function `find()`. You may use it with a syntax like

```
[iRow, iCol] = find(A, 1, 'first'); % the first non-zero entry of A is found
```

**(3 points)**

Problem 2-D Create a function called `problem2D` which can find all Pythagorean triplets up to the number  $N$  and calculates how many of these triplets there are. The header of the function reads

```
function [R, I] = problem2D(N)
```

where  $\mathbf{R}$  is the matrix of [Pythagorean triplets](#), described in details below,  $\mathbf{I}$  is the number of triplets found, and  $N$  is the input variable described below. The function should be reasonably fast, *i.e.*, to calculate all triplets up to  $n_I \leq N = 1000$  in terms of seconds. The output variable  $\mathbf{R}$  is a matrix  $\mathbf{R} \in \mathbb{Z}^{I \times 4}$  with the following structure

$$\mathbf{R} = \begin{bmatrix} n_1 & a_1 & b_1 & c_1 \\ \vdots & \vdots & \vdots & \vdots \\ n_i & a_i & b_i & c_i \\ \vdots & \vdots & \vdots & \vdots \\ n_I & a_I & b_I & c_I \end{bmatrix}, \quad (6)$$

where

$$n_i = a_i + b_i + c_i. \quad (7)$$

A Pythagorean triplet is a set of three natural numbers,  $a_i < b_i < c_i$ , for which,

$$c_i^2 = a_i^2 + b_i^2. \quad (8)$$

A well-known example of a Pythagorean triplet is  $a_1 = 3$ ,  $b_1 = 4$ , and  $c_1 = 5$  with  $n_1 = 12$ . As a sanity check, see the first two correct lines of the output variable  $\mathbf{R}$

$$\mathbf{R} = \begin{bmatrix} 12 & 3 & 4 & 5 \\ 24 & 6 & 8 & 10 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}. \quad (9)$$

To illustrate how the variable  $N$  is used: in case that  $N = 15$ , there is only one Pythagorean triplet for  $n_1 = 12$ , see (9), however, for  $N = 10$  there is no Pythagorean triplet at all. This problem is freely inspired by the [Project Euler, Problem 9](#).

**(4 points)**

## 2 Instructions

Complete all the assignments till

- April 2, 23:59. (Monday's group, B0B17MTB),
- April 4, 23:59 (Wednesday's group, BE0B17MTB).

Write your solutions into m-files called `problem2{A-D}.m` and upload them via the [BRUTE system](#). In the case of uploading more files, add them to a ZIP archive. You can use the Homework grader and validate the solution via `homework2.p` (right-click on `homework2.p` in Current Folder and choose Run, or press F9). You can start the grader as many times as you want. Once satisfied with your result, choose the option ("4: GENERATE SUBMISSION CODE"), and attach the generated zip archive to the BRUTE.

All the problems shall be solved by the students individually (notice the BRUTE system has a duplicity checker). Do not use functions from MATLAB Toolboxes.

Contact us at [matlab@fel.cvut.cz](mailto:matlab@fel.cvut.cz) with any questions. The team of teachers wishes you good luck in solving.