Homework (B0B17MTB, BE0B17MTB)

Problem Set 2

October 21, 2020

1 Assignment

Problem 2-A Implement a function called `problem2A`, which evaluates Euclidean distances between two sets of points, finds a sphere with centre at the middle-point between two most distant points and calculate its radius. Finally, verify if all points are inside this sphere.

Imagine two sets of points, $\boldsymbol{p}_m \in \mathcal{P}$, $m \in \{1, \ldots, M\}$ and $\boldsymbol{r}_n \in \mathcal{R}$, $n \in \{1, \ldots, N\}$. They are represented by two matrices, $\mathbf{P} \in \mathbb{R}^{M \times 3}$ and $\mathbf{R} \in \mathbb{R}^{N \times 3}$, serving as the sole inputs into the function. The function calculates Euclidean distance between each pair of points, taken one by one from the sets $\mathcal{P}$ and $\mathcal{R}$, as

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

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

$$\boldsymbol{c} = \frac{1}{2} \left(\boldsymbol{p}_{m_c} + \boldsymbol{r}_{n_c} \right) \tag{2}$$

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

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

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 `Problem2_A` reads

```
function [D, a, c, allPtsIn] = problem2A(P, R)
```

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

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

with the results

$$\mathbf{D} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \tag{5}$$

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

<u>A hint</u>: 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
```
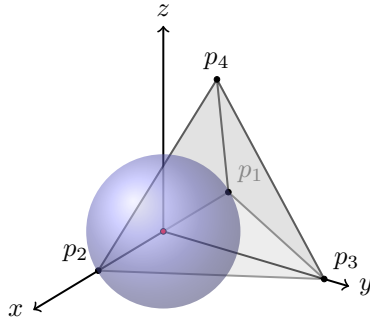
(**2 points**)

Figure A: 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 $\boldsymbol{c} = [0\ 0\ 0]$ shown here as red circle.

Problem 2-B Create a function called `problem2B` 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] = problem2B(N)
```

where `R` is the matrix of Pythagorean triplets, described in details below, `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

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

where

$$n_i = a_i + b_i + c_i. \tag{7}$$

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

$$c_i^2 = a_i^2 + b_i^2. \tag{8}$$

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 $\mathbf{R}$

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

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.

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**)

2   Instructions

Complete all the assignments till

- November 1st, 23:59 (Monday's group, B0B17MTB),

- October 27th, 23:59 (Wednesday's group, BE0B17MTB).

Write your solutions into m-files called `Problem2{A-B}.m` and upload them via the BRUTE system. In the case of uploading more files, add them into ZIP archive. Together with that, 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 you are 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 matlab@elmag.org with any questions.