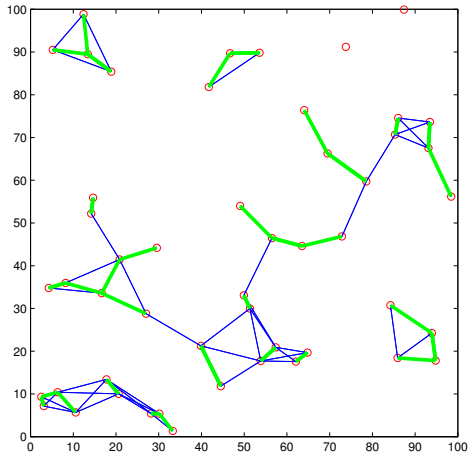
Problem set 1b: MATLAB Fundamentals and Graphics

Handed out: Friday, September 23, 2016
Due: 11:55pm, Thursday, October 6, 2016

You must hand in two files with the same naming convention as in the previous problem set: `your_name_E1b.pdf` and `your_name_E1b.zip`, the second one containing all the MATLAB code you used to solve the exercises.

Exercise 5. Network connectivity.

Consider a wireless network whose nodes are randomly deployed in a given square area. Those nodes are able to set up a radiolink to the neighboring nodes as long as the Euclidean distance between them does not exceed a certain threshold. The purpose of this exercise is to do a graphical depiction of the network connectivity, showing each node with a red circle, plotting a blue line among those nodes for which the connection is feasible and plotting a thicker green line in the link that would have smallest distance for each connected node. The following image shows an example of the plot we want to obtain for $N = 50$ and $Dmax = 15$ (see parameters definition below). Please note that no **`for`** loops are required to solve this exercise.



(1) Generate randomly the (x, y) coordinates of a set of N nodes in the square $[0, 100] \times [0, 100]$ (**`rand`**). Store these coordinates in two vectors. Do a plot that indicates the location of the nodes in the square as red circles (**`plot`**).

(2) Generate a matrix that stores in the element (i, j) the Euclidean distance between the i-th and the j-th node, computed as $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ (**`sqrt`**, **`meshgrid`** or **`repmat`**, operator "**`.^`**").

(3) Identify the node pairs for which the distance is below a certain threshold $Dmax$ (**`find`**[1]). Add to the previous plot (**`hold on`**) a blue line for each pair using commands **`plot`** or

[1]Note that the **`find`** command can index the elements of a $N \times N$ matrix either as a number from 1 to N^2 or as a pair of numbers (row and column) from 1 to N, depending on how it is called. The mapping of one format to the other one can be done with **`ind2sub`** and **`sub2ind`**.

line (in that case use **set** to choose line color). Notice that both commands can draw all the segments at once (without using any loop). For this, you must provide two matrices M_X, M_Y , with two rows each one. Each column of M_X and the corresponding column of M_Y must contain the coordinates of the endpoints of a segment.

(4) For each node identify its closest neighbor and if the distance between them is smaller than the threshold $Dmax$ then add to the drawing a thick green line between them (**plot** or **line**, **min**, **find**, **<**). Set the desired line thickness by changing the parameter "**LineWidth**" with the function **set**. A simple way to avoid the trivial solution that a node is at distance zero from itself is to set to infinity (**Inf**) the diagonal entries of the distance matrix before searching for the minimum. There are many ways to do that, one of them would be to add to the original matrix a diagonal matrix whose entries are infinite (**diag**, **ones**).

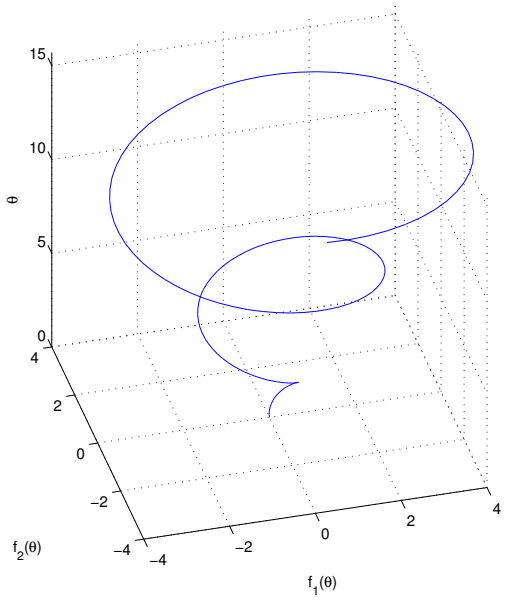
Exercise 6. Plots of two-dimensional functions.

Let's continue with the Archimedes spiral that appeared in Exercise 3 (problem set E1a):

$$\begin{cases} f_1(\theta) = \frac{\theta}{\pi}\sin\theta \\ f_2(\theta) = \frac{\theta}{\pi}\cos\theta \end{cases}$$

We want to depict it now as a three dimensional plot.

(1) Depict the spiral as shown next. That is, generate a plot where the vertical axis corresponds to the value of θ and the two dimensions in the horizontal plane correspond to $f_1(\theta)$ and $f_2(\theta)$ for values of θ in the range of 0 to 5π (operator '**:**', **sin**, **cos**, **plot3**). Add labels to the three axis[2] (**xlabel**, **ylabel**, **zlabel**) and play with the **view** command and the 'Rotate 3D' icon in the figure menu to get the desired perspective. Set also the axis values and the axis ratio to the appropriate values (**grid on**, **axis**, **axis equal**).



[2]The subindex can be introduced in a text by means of the underscore sign "_" and the greek character θ with the string "\theta", e.g. **xlabel('f_1(\theta)')** will show in the x-axis '$f_1(\theta)$'. The conventions are the same as in Latex.

(2) The compression problem[3] consists on the compact representation of the input data employing as few parameters as possible. For example, a 2:1 compression ratio means that amount of data is reduced by half. In the case of lossy compression it is accepted that the compressed data does not allow for a complete recovery of the original data but only of an approximation of it, so the most relevant information is preserved but some details are lost.

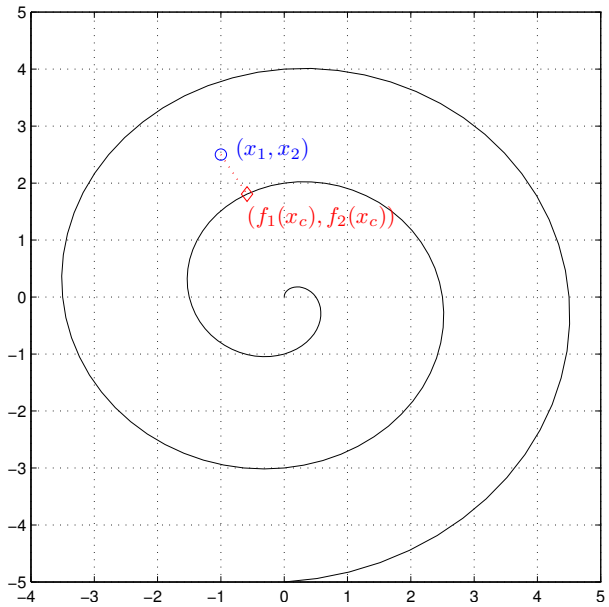
One of the applications of the Archimedes spiral is a very simple memoryless lossy compression scheme with a 2:1 compression ratio. The idea is to represent any point in the plane (x_1, x_2) by means of the value of θ that corresponds to the spiral value that is closest to that plane point in terms of Euclidean distance, denoted as x_c. Thus, x_c is the compressed representation of the pair (x_1, x_2) and is obtained as[4]

$$x_c = \underset{\theta}{\operatorname{argmin}} \left[(x_1 - f_1(\theta))^2 + (x_2 - f_2(\theta))^2 \right]$$

When the original signal wants to be recovered from its compressed version it is then estimated as

$$(\hat{x_1}, \hat{x_2}) = (f_1(x_c), f_2(x_c))$$

The following figure depicts an example of this mapping: for $(x_1, x_2) = (-1, 2.5)$ the closest value is obtained for approximately $x_c = 5.969$ and is $(f_1(5.969), f_2(5.969)) = (-0.587, 1.807)$, so these would be the values of the reconstructed signal.



In this exercise we want to do a graphical depiction of the mapping from (x_1, x_2) to x_c by doing a 3D plot that has in the z-axis the compressed value x_c that corresponds to each point in the (x_1, x_2) plane.

First, start by generating a grid of equally spaced points in the square $[-4, 4] \times [-4, 4]$ (using operator '**:**' and function **meshgrid** or **repmat**), storing them into two matrices X1 and X2 (with the x_1-coordinates and the x_2-coordinates of the grid points, respectively). A spacing of 0.005 will produce good graphic results, but use a larger spacing if needed in

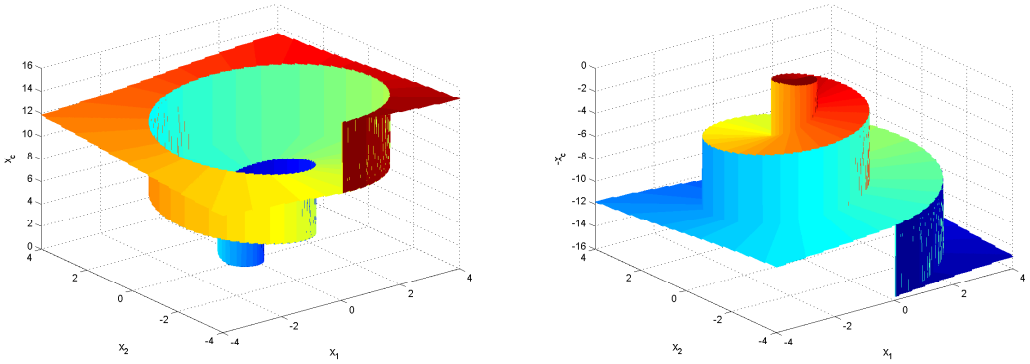
[3] See https://en.wikipedia.org/wiki/Data_compression.

[4] The mathematical operator "argmin" chooses the argument that minimizes the value of the function. In general $\operatorname{argmin}_x f(x) := \{x \mid \forall y : f(y) \geq f(x)\}$.

order to reduce the computational load. Please double check that the matrices are such that the x_1 values change with the row index and the x_2 values change with the column index in both matrices.

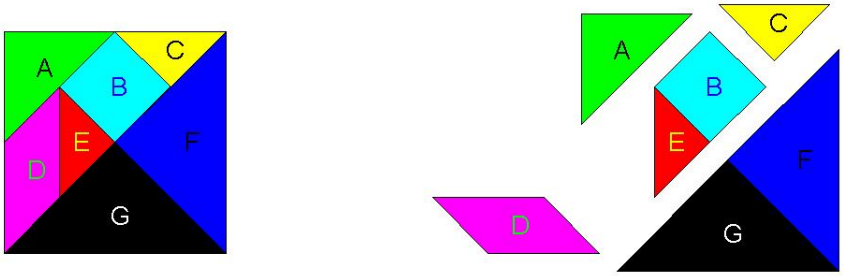
Next, For every node in the grid given by X1, X2 , find the closest spiral point by computing the distance of points in the grid to the spiral points and choosing the minimum (use two **`for`** loops, operator "**`.^`**" and command **`min`**). Store the value of θ that corresponds to this closest point in matrix Xc that has the same size as X1 and X2.

Finally, depict the mapping from (x_1, x_2) to x_c in a 3D plot using command **`mesh`**. The outcome should look like this (the plots for x_c and $-x_c$ are depicted for better understanding of the result).



Exercise 7. Polygons and colours.

Generate the plot shown in the figure. Use functions **`fill`**, **`text`** and **`axis`**.



Use **`set`** in conjunction with **`text`** to set the **`'Fontsize'`** and **`'Color'`** attributes of the text labels. To do that, create vectors X_i, Y_i for each piece in the puzzle to form the figure in the left. Then introduce some displacements (by just adding constants) or rotations (by exchanging X_i, Y_i and changing the sign of one of the two) to every piece to create your own disposition (as the one on the right). You also need to compute the coordinates of each numeric label, and apply the same geometric transformations to them.