MPHY0020 Computing in Medicine - MATLAB Exercises 2

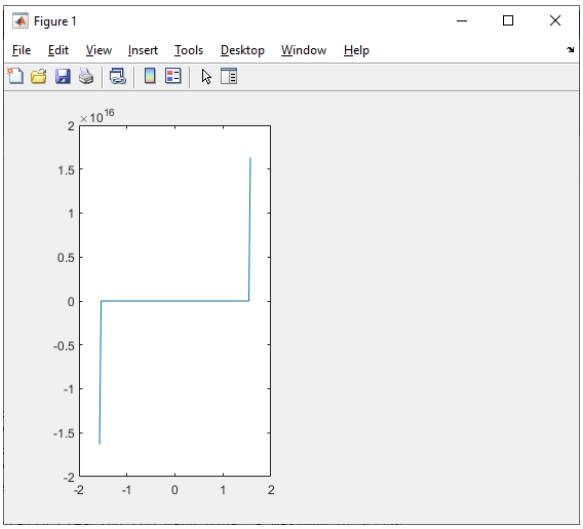
In this exercise you will practice displaying data in MATLAB, first using 1D data, then in 2D. You have seen several such examples in the lectures; please refer to the relevant slides and examples for inspiration.

Exercise 2.1 – Generating and displaying 1D data

In the first task, you will generate and then display 1D data – first of a number of simulated dice throws, followed by a the numerical implementation of a mathematical function.

1. In the first part, we will simulate repeated dice throws using a random number generator. Look up the MATLAB help on `rand` and `randi` to work out how to use either function to generate N random integer numbers between 1 and 6. Note that both functions by default assume that you ask for a 2D matrix filled with random numbers; carefully read the documentation to generate a 1D *vector* (of size [100x1]) instead.
   - Use either `rand` or `randi` to simulate 100 die throws, resulting in 100 random integers ranging between 1 and 6.
   - Display these values using `plot` – this should show a random variation of the die value between consecutive throws.
   - Next, generate a histogram of the die throws. Look up the documentation on the `histogram` function, and ensure the histogram contains exactly 6 bins centered around the numbers 1 through 6 – one for each possible face value of a die. The resulting plot should contain roughly the same number of counts in each bin, indicating that the die is "fair" and does not have a preferred side.
   - Finally, increase the number of simulated die throws to 2000 and again generate a histogram – hopefully, you will see that the distribution is much more even now!

2. In the second part, you will be plotting the tangent (tan) function. As you may know, this function is discontinuous in certain points, which can cause issues with plotting.
   - Start with defining the x axis: this should start at -π/2, end at +π/2, and contain 101 points. (Hint: you can use `linspace` for this, or alternatively use implicit vector generation of the form `-pi/2:step_size:pi/2`).
   - Next, compute `y = tan(x)` for all point in `x`.
   - Now have MATLAB generate a `figure`, and within this figure generate a `subplot` on the left of this window. Within this panel, use `plot` to display `y` as a function of `x`. This should result in a window that resembles the one below. Note how the vertical axis is dominated by the discontinuities at ±π/2; the actual shape of the tan function cannot be observed.

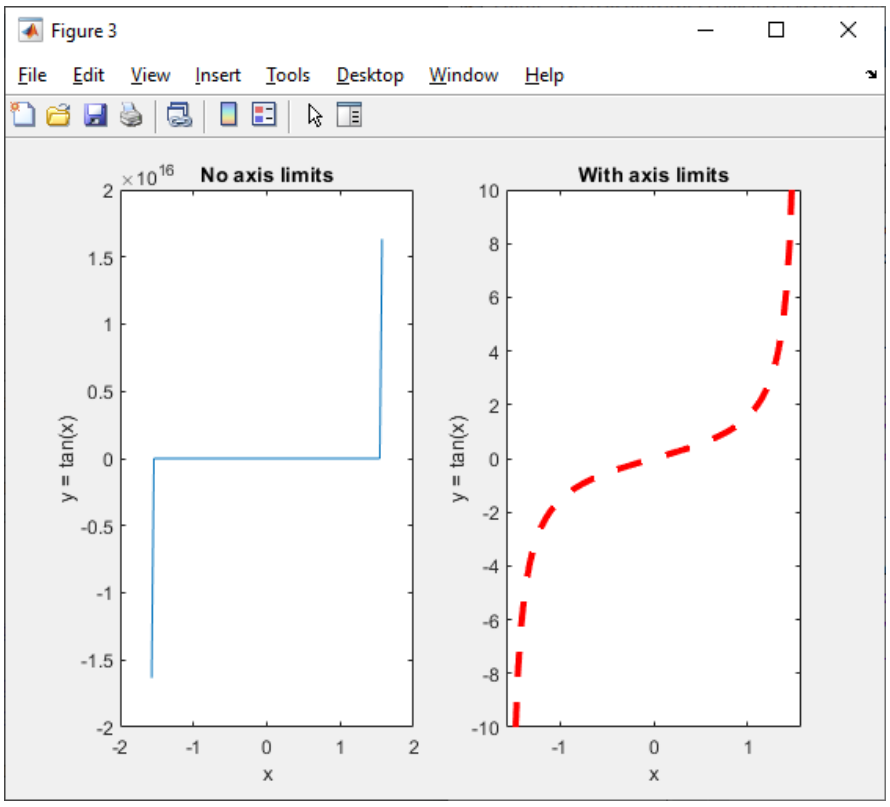


- To improve on the visualisation, instruct MATLAB to generate a second panel on the right side of the same window, and again display y as a function of x. This time, however, constrict the range of the horizontal and vertical axes to "sensible" numbers. To achieve this, use the `axis` function:

  ```
  axis([x_min x_max y_min y_max]);
  ```

  and set the limits `x_min` and `x_max` to match the range of `x`, `y_min = -10`, and `y_max = 10`.
- In addition, set the line colour of the right-hand-side plot to red, change the linewidth to 3 points, and set the line style to a dashed line – consult the documentation for `plot` to work out how.
- Finally, add meaningful axis labels and titles to the two panels using the `xlabel`, `ylabel` and `title` functions.

Your code should now generate a visualisation resembling the figure below.



Exercise 2.2 – Displaying 2D slices through a 3D CT scan

In this task, you will load a 3D CT scan, and display 2D slices through this volumetric data.

1. Start by downloading the file CT_vol.mat from Moodle, and load this into MATLAB using the `load` command. This file contains three variables:
   - `V` (a 3D array containing the CT scan data),
   - `img_dims` (containing the size of the 3D image in numbers of voxels in the x, y and z dimensions),
   - and `vox_dims` (containing the size in mm of each voxel in the three dimensions).

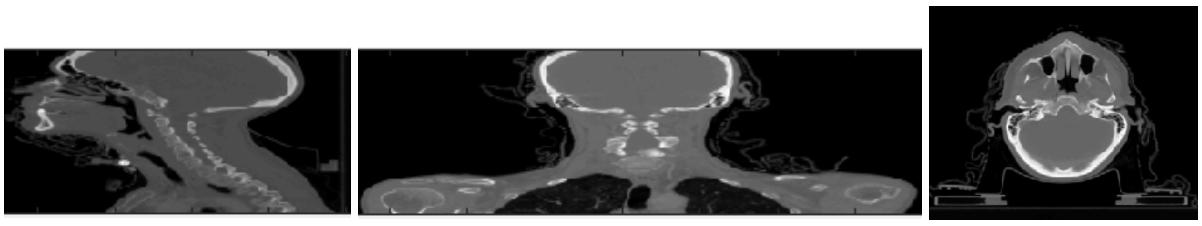
2. You can display slices from the volume `V` in directions orthogonal to the original slice directions using, for instance, the `imagesc` command (but other visualisation options exist). For instance, to display the 270th slice from the first dimension you would use:

```
imagesc(squeeze(V(270,:,:)));
axis image;
colormap(gray);
```

   In this code:
   - The `squeeze` function removes singular dimensions of an array (i.e. those dimensions of size 1). Compare the output of the following two commands:
     ```
     size(V(270,:,:))
     size(squeeze(V(270,:,:)))
     ```
     This is necessary as the `imagesc` function does not accept arrays with more than 2 dimensions as input. To prove this, try running
     ```
     imagesc(V(270,:,:));
     ```
     which should generate an error message.
   - The short-cut "`:`" is used to index all elements within that direction, and is equivalent to "`1:end`".

3. Next, try displaying slices from the other dimensions. Note that you do not need to use squeeze when displaying a slice from the 3rd dimension – make sure you understand why.

4. If you are used to viewing medical images using other software, you may notice that the displayed images may have different orientations than you would expect – This is due to MATLAB displaying the first dimension of an image on the y-axis (whereas most software will display it on the x-axis). The conventional orientation would appear like:



   This can be achieved by displaying the transpose of slice being displayed, i.e.
   ```
   imagesc(squeeze(V(270,:,:))');
   ```
   where the apostrophe signifies the transpose operation.

5. Depending on which method you use to visualise the slices, you might find that images displayed from the first two dimensions appear upside-down (this is the case for `imagesc`). This is because in MATLAB the numbering along the y-axis starts at the top of the image instead of the bottom. This can be corrected using the command:

```
axis xy;
```

Try this to confirm the images for the first two dimensions are now displayed in the correct orientation. **Note:** you do not need to run this command when displaying axial slices (from the 3rd dimensions) as the usual way to display these is with the y-axis numbering starting at the top.

6. You might also have noticed that the slices from the 1st and 2nd dimension looks a bit squashed (as they do in the images above). As we did not explicitly specify the voxel dimensions, MATLAB assumes the voxel size in each dimension is equal. However, the slices are thicker than the pixels in each slice as can be gleaned from the variable `vox_dims`:
    - Each voxel measures 1.12 mm by 1.12 mm in the X and Y directions;
    - Each slice is 3 mm thick

    There are two ways to ensure the images are displayed correctly, depending on whether you want the numbers along the axes ("axis tick labels") to correspond to voxels or mm:
    - numbers correspond to voxels:

```
imagesc(squeeze(V(270,:,:))');
axis xy;
daspect(1./[vox_dims(2) vox_dims(3) 1]);
```

    - numbers correspond to mm:

```
imagesc(vox_dims(2)*(0:img_dims(2)-1),...
vox_dims(3)*(0:img_dims(3)-1),squeeze(V(270,:,:))');
axis xy image;
```

    Here "`...`" just means the line of code carries on to the next line. Notice how the numbers along the image axes change when running the above commands. Use MATLAB help and the lecture notes to make sure you understand what the above commands actually do and how they affect the axis numbering.

7. Now write a function that can display any slice from any dimension of the 3D CT volume.
    - The function will not have any output variables; it will just take some inputs (you need to work out which inputs are needed) and then display the desired slice.
    - Display the images in greyscale and at the correct aspect ratio.
    - Make sure the slices are orientated correctly as shown above.
    - Scale the axes correctly depending on the orientation of the slice.
    - Allow the user to choose whether they want the axis numbered with mm or voxels. One of the inputs to the function should thus indicate whether mm or voxels should be used.

    Try displaying slices from different dimensions to confirm your code is working correctly.