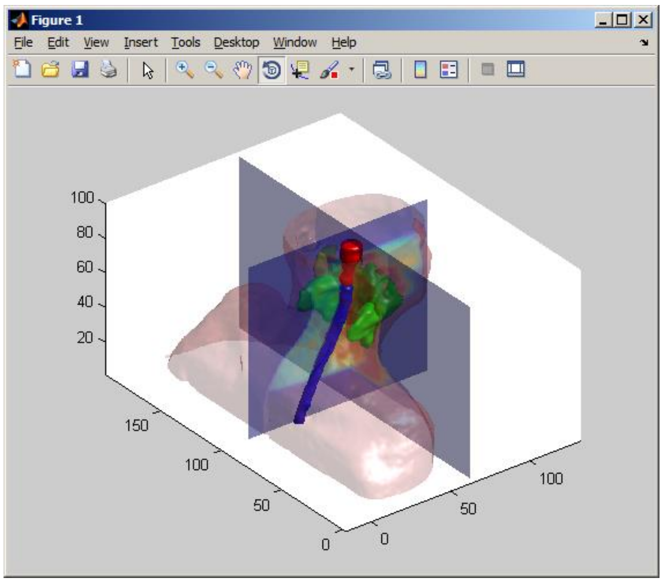
MPHY0020 Computing in Medicine - MATLAB Exercises 3

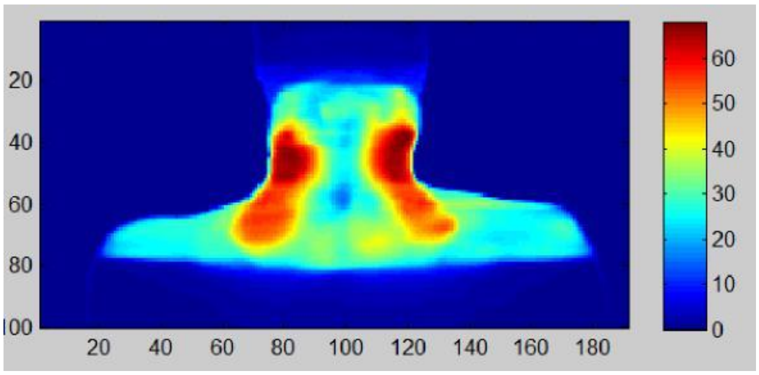
These exercises will use data from planning Radiotherapy. First, you will produce a 3D visualisation of the different anatomical structures and dose distribution. Second, you will write functions using non-vectorised and vectorised code that can be used to calculate Dose-Volume-Histograms.

When planning Radiotherapy (RT) treatments, the clinician will delineate the tumour and organs at risk (those that should not receive too much dose to avoid side effects) on a CT scan.



The above figure shows renderings of the delineated tumour and organs at risk (as well as the external body contour). The tumour is shown in green, the brainstem in red, the spinal canal in blue, and the right parotid gland in yellow.

When assessing an RT treatment plan, a dose calculation is performed to estimate the amount of dose delivered to each voxel in the CT scan. This estimated dose can then be used to assess the dose delivered to the organs at risk. These calculations have already been performed for you, and the image below shows the dose distribution for a slice in the middle of the CT volume.



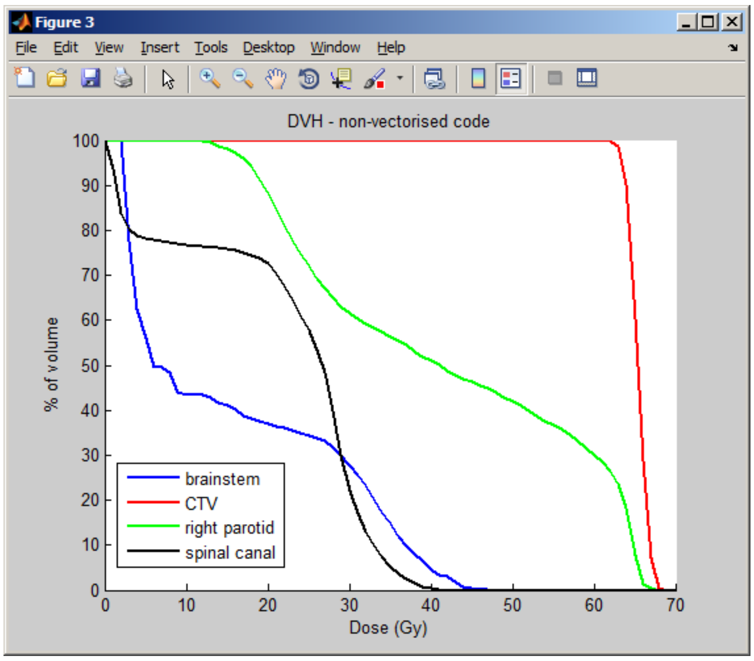
Exercise 3.1 – 3D visualization

The goal of this exercise is to visualise, in 3D, the shape of the brainstem, spinal canal, right parotid gland, and the treatment volume.

1. Load the data file RTdata.mat into MATLAB (which is available from Moodle). This file contains four 3D logical arrays, which correspond to binary images of some of the structures delineated when planning radiotherapy. In this case, the CTV (clinical target volume), right parotid gland, brainstem and spinal canal are included, as well as a 3D double array containing the predicted dose distribution.
   Note - the data provided does not include the external skin contour shown in the visualisation above and in the lecture.
2. Use the `patch` and `isosurface` commands to render visualisations of the CTV structure.
   - For the first visualisation you should use the unprocessed binary image (i.e. do not apply any smoothing).
     - Set the `edgecolor` to none and the `facecolor` to red.
     - Set the view so it is suitable for 3D visualisations, create a light object, and set the lighting algorithm to `gouraud`.
   - Now generate a 2nd visualisation in the same figure (but in a different `subplot`) – this time you should smooth the binary image before displaying it, but everything else should be the same as the first visualization. Note the difference the smoothing makes.
   - Now generate a 3rd visualisation in the same figure (the figure should now display three panels) where in addition you apply the `isonormals` command to the patch object to further enhance the visualization. Ensure the volume data used for `isonormals` is the smoothed volume and not the original binary volume. Again note the difference the `isonormals` command makes.
3. Next, you should produce a nice looking visualization of all the structures included in RTdata.mat, as well as the predicted dose distribution. Using the example you have seen in the MATLAB lectures, ensure that you:
   - Smooth the data prior to visualization
   - Display the different structures using different colours:
     - the brainstem as a blue patch object
     - the spinal canal as a red patch object
     - the right parotid as a green patch object
     - the CTV as a magenta patch object
   - Set the CTV patch object partially transparent by setting its `facealpha` property (a value of 0 corresponds to completely transparent, a value of 1 corresponds to completely opaque).
   - Use the `isonormals` function to produce nicer visualisations
   - Display a slice through the dose at z = 55 using the `slice` function. This function produces a patch object with one face for each voxel that intersects with the slice. By default each face will be displayed with edges drawn around every voxel in the slice, and the slice will appear black in colour unless you zoom in on it very closely. To avoid this, ensure you set the slice's properties so that the edges are not displayed.
   - Also make the slice partially transparent.
   - Set the correct aspect ratio, axis labels, view angle, and add lighting.
4. Save all the commands used to make your visualisations in a script and submit it as your solution to this task.

Exercise 3.2 – Dose Volume Histograms – code vectorisation

To evaluate the efficacy and safety of a radiotherapy plan, it is useful to calculate Dose Volume Histograms (DVHs). These are cumulative histograms that show the percentage of the tumour (or organ at risk) that will receive a dose exceeding a certain value, and can be useful for assessing how good the plan is. For a good treatment plan, a high dose is delivered to all of the tumour, and a low dose to most of the organs at risk.



The above image shows an example DVH. If you pick a dose value, e.g. 20 Gy, you can see that just under 40% of the brainstem received 20 Gy or more, whereas about 75% of the spinal canal, 95% of the right parotid, and 100% of the CTV received 20 Gy or more.

The goal of this exercise is to compute DVHs for the four structures contained in the data set RTdata.mat and display the outputs, resulting in the plot shown above. To achieve this, you will write three functions, of varying levels of vectorization, with the following inputs and outputs:

```
function dvh = calcDVHSlow(dose,structure,dose_vals)
% dvh = calcDVHSlow(dose,structure,dose_vals)
%
% This function calculates the dose volume histogram (DVH) for a given
% structure and dose distribution
%
% INPUTS:
%   dose:        a 3D array containing the dose at each voxel [Gy]
%   structure:   a logical 3D array indicating which voxels are inside
%                the structure
%   dose_vals:   a vector containing the dose values that the DVH will be
%                calculated for [Gy]
%
% OUTPUTS:
%   dvh:         a column vector the same length as dose_vals, giving the
%                percentage of voxels in structure that recieved the
%                corresponding dose value or greater
%
% for example, if dose_vals was [20 40] the function will return a vector
% dvh containing 2 elements, where the first element gives the percentage
% of voxels within the structure that received a dose greater than or equal
% to 20, and the second element gives the percentage of voxels within the
% structure that received a dose greater than or equal to 40.
```

1. Your first function, that does not use any vectorisation, should:
   - Loop over all voxels in the 3D arrays
   - Test if the current voxel lies within the `structure`. If so, it should:
     - Loop through all of the values in `dose_vals`
     - Test if the dose at this voxel is greater than or equal to the current value from `dose_vals`.
     - If so you, should increment the value of `dvh` corresponding to the current value from `dose_vals`.
   - Each value in `dvh` should now contain a count of how many voxels have a dose value equal to or greater than the corresponding value in `dose_vals`. Convert these counts into percentages by dividing by the total number of voxels within the `structure`.
2. The second function should repeat this computation, but should use vectorisation to remove the loops over all the voxels. It should:
   - Use logical indexing to extract only those values of `dose` that fall within the `structure`.
   - Loop through `dose_vals` and compare all extracted dose values against the current value (hint: the element-wise operator "`>=`" works on scalars as well as arrays!)
3. The third function should use element-wise operations to also remove the loop over `dose_vals`, which can be achieved using an expression of the form

   ```
   dvh_counts = DOSE_IN_STRUCT >= DOSE_VALS;
   ```

   Note: To achieve this, you need to ensure the two arrays being compared are the same size and shape. Use `meshgrid` or `ndgrid` to produce matrices containing *all* of the values from `dose_vals` and *all* of the `dose` values inside the `structure`.
4. Use your functions to calculate the DVH for each of the four provided structures. The DVHs should be computed for doses ranging from 0 Gy to 70 Gy (in increments of 1 Gy). Plot your results on a graph as above; add the axis labels, legend, title, and line colors, and set the linewidth of each plot to 2.
   - Your result should look exactly the same as the graph above. In particular, the first value of the DVH for all structures should be 100% (as all voxels have received a dose of 0 Gy or more), and the last value should be 0% (as no voxels received 70 Gy or more).
   - Check that all of your functions result in the same DVHs (hint: look up `isequal`).
   - Measure how long your functions take to compute the four DVHs using the `tic` and `toc` commands (look at MATLAB help how to do this).
   - Which function computes the results the fastest? Can you explain why?
5. Save all the commands used to produce your graph and time your functions into a single script, and submit this script (together with the .m files for your three functions) as your solution to this exercise.