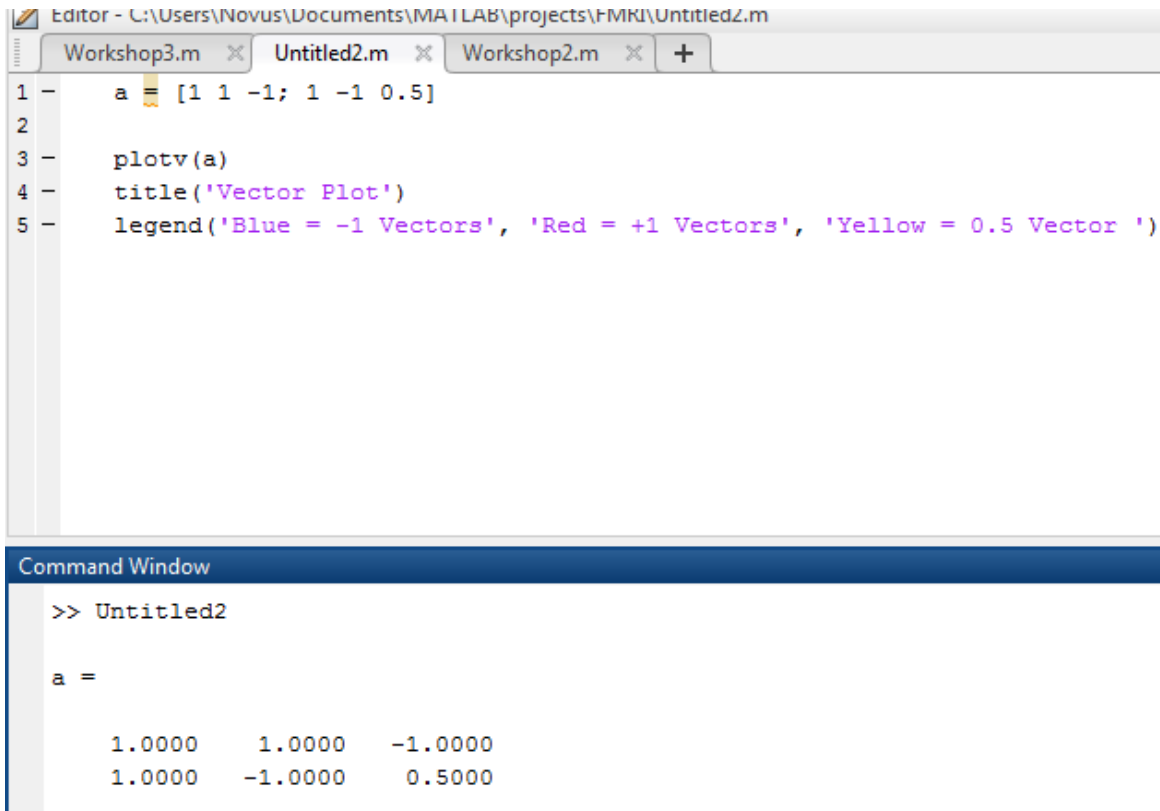


Computational Methods - Worksheet 2

Task 1 – Part A

The solution to part A is to generate a matrix and use the plotv function to graph the respective matrix vectors. Plotv takes two input arguments, and plots the column vectors of 1.) $M = R$ by Q matrix of Q column vectors with R elements, 2.) T = an optional Line plotting type. Plotv only plots the first 2 rows of a matrix if R is greater than 2. The below figure demonstrates plotv in use to solve task 1, part A:

Figure 1.1 Programmatic Form



The figure shows a MATLAB Editor window with the following code:

```
1 - a = [1 1 -1; 1 -1 0.5]
2
3 - plotv(a)
4 - title('Vector Plot')
5 - legend('Blue = -1 Vectors', 'Red = +1 Vectors', 'Yellow = 0.5 Vector ')
```

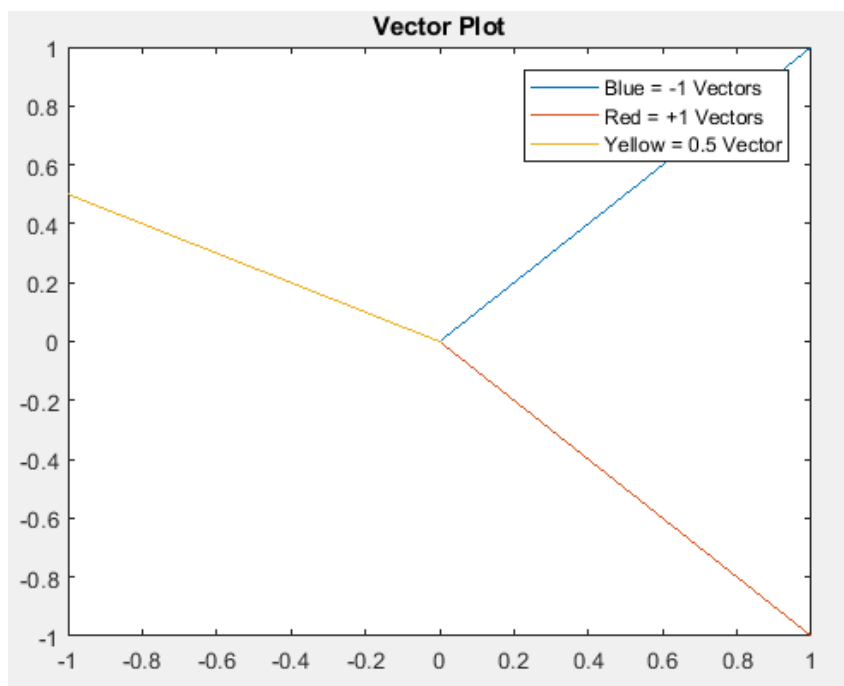
The Command Window shows the execution of the code:

```
>> Untitled2

a =

    1.0000    1.0000   -1.0000
    1.0000   -1.0000    0.5000
```

Figure 1.2 Programmatic results



The image to the left is the result of the plotv function of matrix a. As you can see it has plotted a line for each variation of vectors ranging from -1 , 0.5 and +1.

Task 1 – Part B

This section requires the plot generated in figure 1.2 to be rotated via a linear transformation of matrix a , with the overall goal being to rotate the vector lines on the graph by 90 degrees. The solution to this can be achieved for all vector points simultaneously by multiplying matrix a by the rotation matrix below:

Figure 2.1 – Programmatic form – Rotation

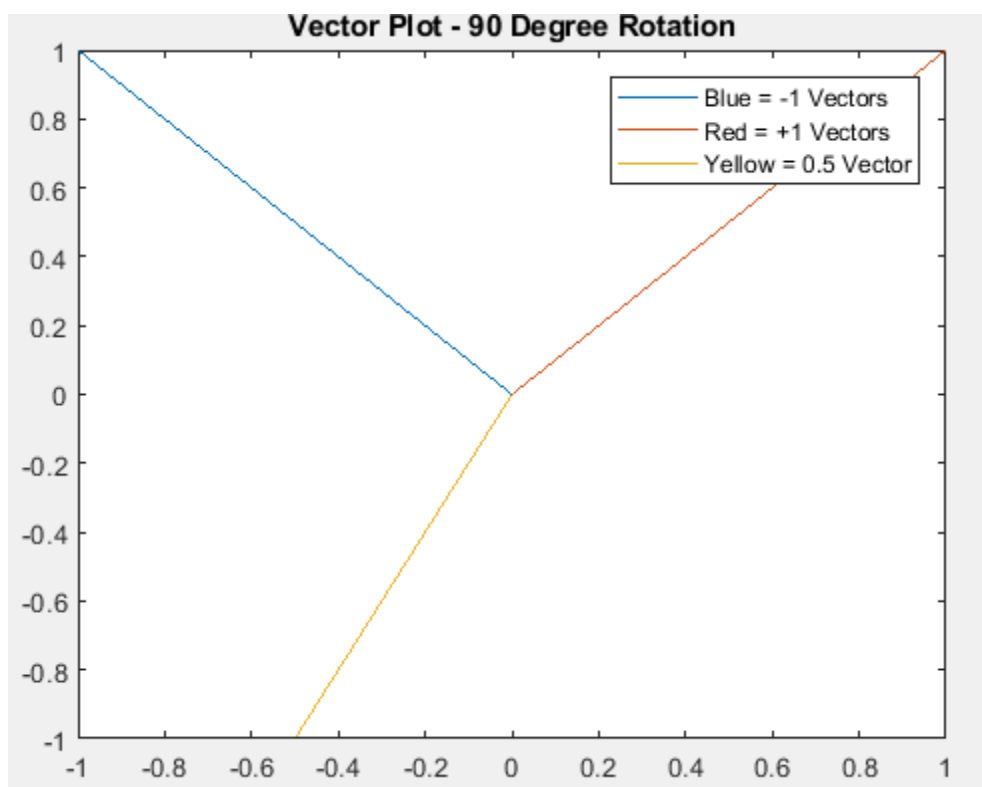
```

2
3 -   plotv(a)
4 -   title('Vector Plot')
5 -   legend('Blue = -1 Vectors', 'Red = +1 Vectors', 'Yellow = 0.5 Vector ')
6
7   %Rotation
8
9 -   R = [cosd(90), -sind(90); sind(90), cosd(90)];
10
11 -   P = R*a;
12   |
13 -   plotv(P)
14 -   title('Vector Plot - 90 Degree Rotation')
15 -   legend('Blue = -1 Vectors', 'Red = +1 Vectors', 'Yellow = 0.5 Vector ')

```

The above shows that variable R stores the rotation matrix. The `cosd()` and `sind()` functions have been utilised in contrast to the standard `cos()`/`sin()` functions to allow the rotation value to be represented in degrees rather than radians. The rotation matrix itself performs linear transformations counter clockwise from a the origin, being 0, of the coordinate system. The results of the 90 degree rotation can be seen in the plot below:

Figure 2.2 – Result of rotation transformation



Task 2 – Part A

Equation form of Task 2 part A:

$$x - y = 2$$

$$2x + 3y = -1$$

In order to solve the above linear System Gaussian elimination algorithm will be used to solve for the $Ax = b$. Gaussian elimination is performed using a sequence of row operations to modify the matrix coefficients, until it has been reduced to its row echelon form (left corner filled with zeros). At this stage the value of an unknown variable at the bottom of the matrix can be identified, allowing for this value to be substituted back up the matrix rows in order to determine the values of all other unknowns.

Step 1: Convert equations into an augmented matrix:

$$\left| \begin{array}{cc|c} 1 & -1 & 2 \\ 2 & 3 & -1 \end{array} \right|$$

The left-hand side is matrix A, each column can be thought to represent a different variable (1st column = x, column row = y) and the right-hand side is a column vector that holds the Constants. The aim is to make the bottom left coefficient of x = 0 (row 2), thus I will be able to determine the value of y. In order to do this, I perform the following operation:

$$-2(R1) + R2$$

Here the negative of the target number is multiplied by row 1 and then added to the results to row 2. In order to achieve this the following calculations are performed:

$$(-2 \times 1) + 2 = 0 \quad // \quad (-2 \times (-1)) + 3 = 5 \quad // \quad [(-2 \times 2) + (-1)] = -5 \quad //$$

The new values of row 2 are then represented in matrix form:

$$\left| \begin{array}{cc|c} 1 & -1 & 2 \\ 0 & 5 & -5 \end{array} \right|$$

It is now possible to deduce the value of y by doing $-5 / 5 = -1$. The value of y can now be substituted back into the original equation to allow for the determination of x.

MATLAB solution

```

Editor - C:\Users\Novus\Documents\MATLAB
Workshop3.m x Untitled2.m x V
1
2 - A = [1 -1; 2 3];
3 - B = [2 ; -1];
4
5 - linsolve(A,B)

Command Window
>> Untitled2

ans =

     1
    -1
  
```

As demonstrated the handwritten solution to the linear system is validated programmatically using MATLAB.

Handwritten - solution

$$x - (-1) = 2$$

$$x = 2 - 1$$

$$\underline{x = 1}$$

$$\underline{y = -1}$$

Task 2 - Part B

$$x + 2y + z = 2$$

$$x + y + 2z = 0$$

$$5x + y + z = 5$$

Part B Follows a similar process to part A however there is one more unknown.

Augmented Matrix

$$\left| \begin{array}{ccc|c} 1 & 2 & 1 & 2 \\ 1 & 1 & 2 & 0 \\ 5 & 1 & 1 & 5 \end{array} \right|$$

Perform Row operations to make x column for row 2 cancel to 0:

$$-1(R1) + R2$$

$$(-1 \times 1) + 1 = 0 // (-1 \times 2) + 1 = -1 // (-1 \times 1 + 2 = 1 // (-1 \times 2) + 0 = -2)$$

$$\left| \begin{array}{ccc|c} 1 & 2 & 1 & 2 \\ 0 & -1 & 1 & -2 \\ 5 & 1 & 1 & 5 \end{array} \right|$$

Perform Row operations to make x column for row 3 cancel to 0:

$$-5(R1) + R3$$

$$(-5 \times 1) + 5 = 0 // (-5 \times 2) + 1 = -9 // (-5 \times 1) + 1 = -4 // (-5 \times 2) + 5 = -5$$

$$\left| \begin{array}{ccc|c} 1 & 2 & 1 & 2 \\ 0 & -1 & 1 & -2 \\ 0 & -9 & -4 & -5 \end{array} \right|$$

Next row operations must be performed on row 3 element 2 in order to cancel the y term to 0, leaving just z:

$$-9(R2) + R3$$

$$(-9 \times (-1)) + (-9) = 0 // (-9 \times 1) + (-4) = -13 // (-9 \times (-2)) + (-5) = 13$$

$$\left| \begin{array}{ccc|c} 1 & 2 & 1 & 2 \\ 0 & -1 & 1 & -2 \\ 0 & 0 & -13 & 13 \end{array} \right|$$

Equations Thus far:

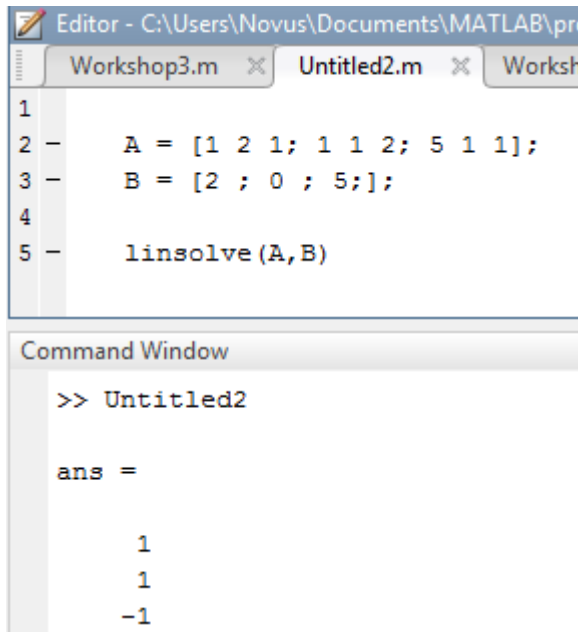
$$x + 2y + z = 2$$

$$-y + z = -2$$

$$-13z = 13$$

Solution: Identify the value of z then substitute the value into equation 2 to determine the value of y. Finally, substitute both z and y values into the 1st equation to ascertain the value of x, as follows:

Please see next page

MATLAB solution


```

Editor - C:\Users\Novus\Documents\MATLAB\pr
Workshop3.m  Untitled2.m  Worksh

1
2 -   A = [1 2 1; 1 1 2; 5 1 1];
3 -   B = [2 ; 0 ; 5;];
4
5 -   linsolve(A,B)

Command Window

>> Untitled2

ans =

     1
     1
    -1
  
```

Handwritten - solution

$$z = \frac{13}{-13}$$

$$\underline{z = -1}$$

$$-y - 1 = -2$$

$$-y = -2 + 1$$

$$-y = -1 \mid \cdot (-1)$$

$$\underline{y = 1}$$

As demonstrated the handwritten solution to the linear system is validated programmatically using MATLAB.

$$x + 2 - 1 = 2$$

$$x = 2 - 2 + 1$$

$$\underline{x = 1}$$

Task 3 – FMRI Analysis

Requirements: Noiseless FMRI = Find the mixture components (x) for each of the 20x20 voxels by solving its linear system ($y = A \cdot x$).

Functional MRI (FMRI) measures brain activity via a correlation between Blood oxygen flow and Neural activity. A neuron does not contain its own energy reserve, such as glucose stores, therefore a neurons energy input comes directly from oxygenated haemoglobin that circulate to the brain via the cardio-vascular system. In order to become increasingly active beyond its idle state a neuron must increase its oxygen input in order to have sufficient energy to exhibit dynamic inhibitory or excitatory behaviour, thus allowing for neural computation to be performed within and between networks of nuclei. The brain is mapped to a three dimensional voxel space allowing for different nuclei in specific brain regions to correspond to different voxel positions. Therefore FMRI can measure variations in Blood-oxygen-level dependent (BOLD) responses of nuclei and correlate this activity to neural activity, relative to the cognitive process being required for a given test condition. This BOLD activity is then mapped to its respective Voxel coordinates allowing one to determine which brain regions are facilitating a certain cognitive function.

Understanding & Conceptualising the Problem

This particular problem assumes that one can linearly add together the BOLD activity produced by each condition for a specific voxel. Therefore the activity of a single voxel at a specific time (t_{1-30}) can be defined by the following equation:

$$y = x_1 A_1 + x_2 A_2 + x_3 A_3 + x_4 A_4$$

Index:

$A_{1-4} = \text{Test conditions}$

$y = \text{Single Voxel}$

$x_{1-4} = \text{Voxel BOLD response per condition}$

Next, one can add a clearer mathematical representation of voxel activity by considering the dimension of time, which is across 30 different time stamps. The above equation can be extended too:

$$y_1(t_1) = x_1(t_1)A_1 + x_2(t_1)A_2 + x_3(t_1)A_3 + x_4(t_1)A_4$$

$$y_1(t_2) = x_1(t_2)A_1 + x_2(t_2)A_2 + x_3(t_2)A_3 + x_4(t_2)A_4$$



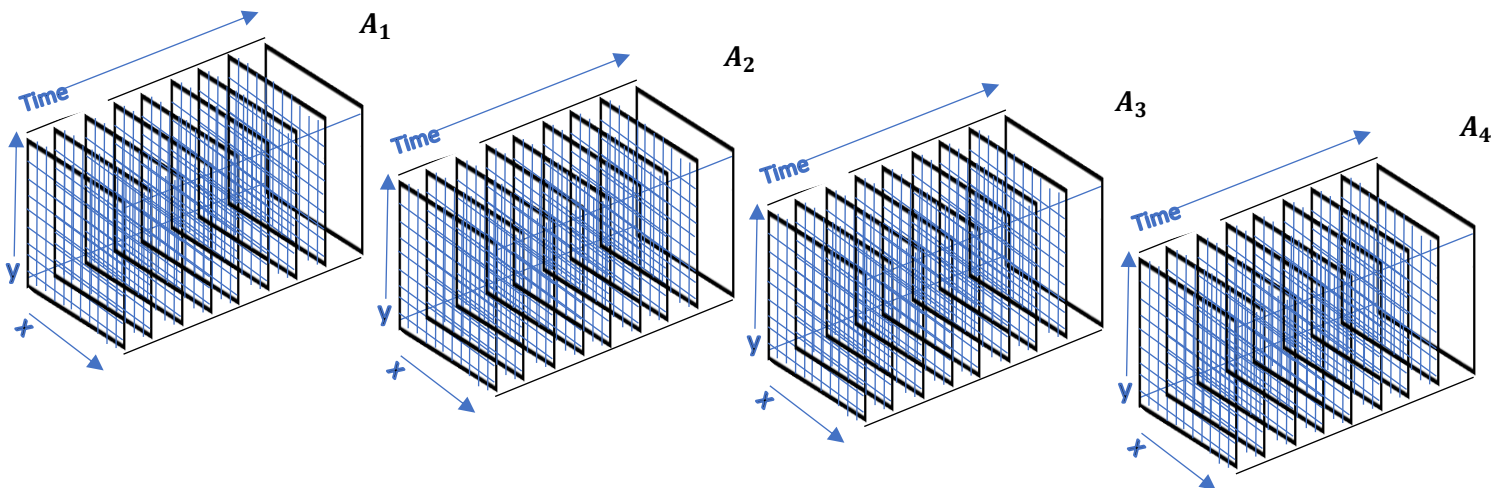
$$y_1(t_{30})$$

It becomes more apparent at this point that the above equations represents a rather large linear system, 30 equations, one for each time-stamp. The system can be solved by using Gaussian elimination to produce 4 different values of x , thus 4 different BOLD activity values for each condition, across the entire timeseries of $x_{1-4}A_{1-4}$.

Consequently, x values can be compared to determine whether there is relationship between the type of condition presented and the activity of BOLD responses in different areas of the brain. If this is so one can ascertain which neural nuclei process specific kinds of information. Finally since different regions of the brain constitute to different positions in voxel space one can investigate how multiple regions of the brain responded in terms of BOLD responses to different test conditions.

Task 3 essentially requires one to calculate the value of x_{1-4} for each condition across the entire range of the targeted voxel space, which is $20 \times 20 = 400$ voxels. One can conceptualise the task via the following diagram:

Figure 3.1 – Conceptual representation of Voxel space



X and y dimensions correspond to voxel space, each vector of $[x, y]$ corresponds to a different voxel and thus a different neural nuclei. Each condition is presented for a total of 30 timestamps thus the state of each voxel (x_{1-4}) may change as time passes, this can be represented as different slices of voxel space across time, for each condition presented. The task requires the construction a program that can navigate through this voxel space, extract each voxels time series to form the linear system first described above. From this a solution can be found that reveals the value of x_{1-4} for every voxel in voxel space.

Programmatic Implementation

Figure 4.1 – voxel space navigation and computation

```

for plane_x = 1:20
    for plane_y = 1:20
        results(plane_x, plane_y, :) = linsolve(A, squeeze(y(plane_x, plane_y, :)));
    end
end

```

Vector space in Matlab can be represented as a 20x20x30 matrix, thus I utilised a nested for loop. The first for loop iterates through the x plane, and the second for loop iterates through the y plane, thus x1, y1 will correspond to a specific Voxel. In the body of the nested for loop I call the results variable which stores a 20x20x4 matrix. Finally I use the linsolve function to programmatically calculate the solution to the linear system of the current voxels activity, across its entire time series(variable Y), relative to each test condition (Variable A), or as you recall:

$$y_1(t_1) = x_1(t_1)A_1 + x_2(t_1)A_2 + x_3(t_1)A_3 + x_4(t_1)A_4$$

$$y_1(t_2) = x_1(t_2)A_1 + x_2(t_2)A_2 + x_3(t_2)A_3 + x_4(t_2)A_4$$



$$y_1(t_{30})$$

The squeeze function removes singleton dimensions from a multi-dimensional input tensor. The linsolve function requires that the second argument be represented in two dimensions, thus I squeeze my three dimensional results onto a 2 dimensional “landscape” or matrix. Finally the results are stored in the results table 20x20x4 matrix which holds each voxels activity for each condition, as shown below:

Figure 4.2 – voxel space navigation and computation

```

%Store the solutions to the linear systems of each voxel for each of their
% time stamps for each condition (1-4)

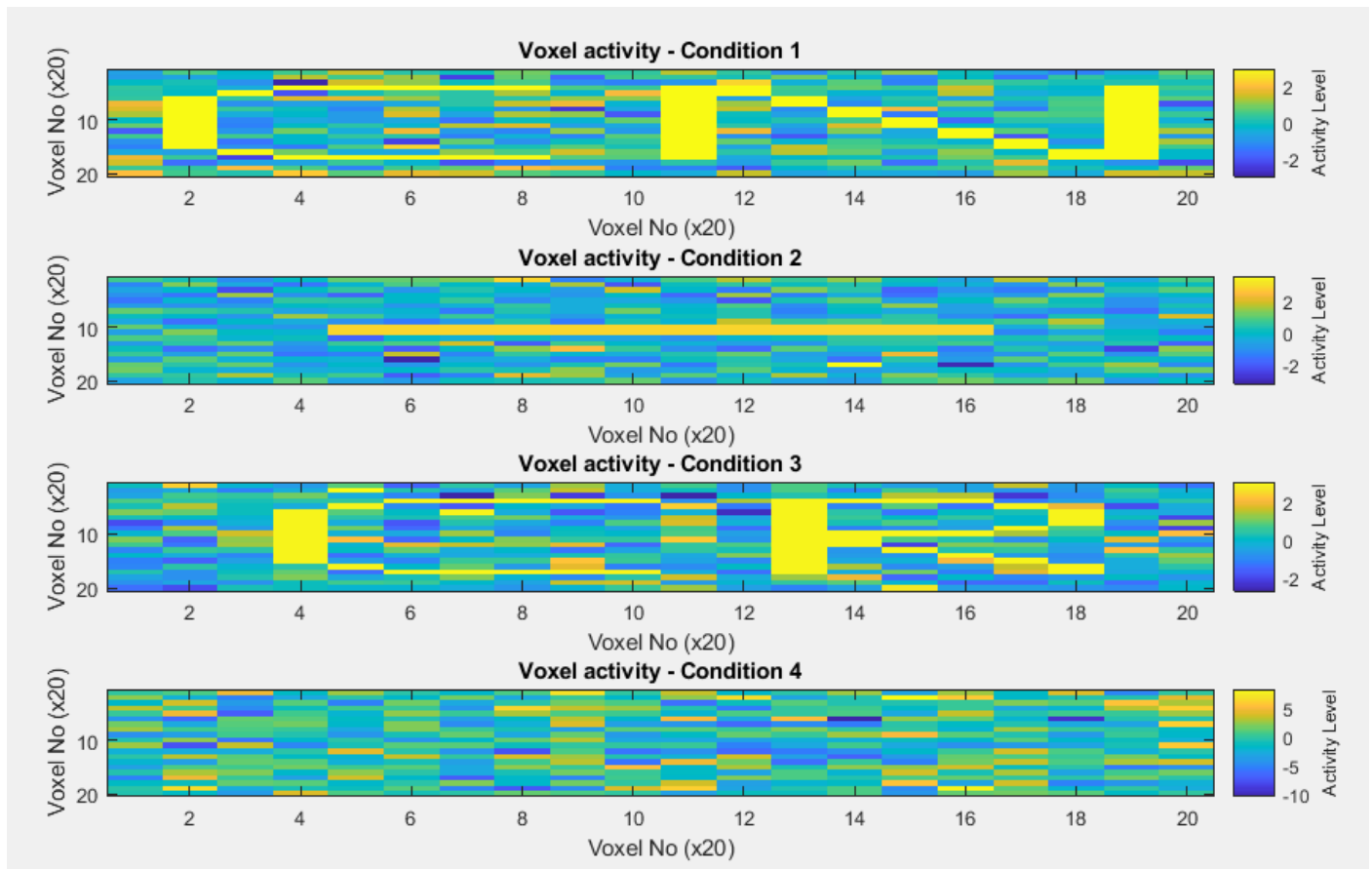
results = zeros(20, 20, 4);

```

Please see next page

When the code file is run the program solves each voxels linear system to produce its x_{1-4} value for each condition across each voxels entire timeseries, as shown below:

Figure 4.3 – Results Graph - voxel space navigation and computation



In conclusion of the execution of the program the results of voxel activity for condition are produced which is exactly what was expected based on the conceptual representation displayed in figure 3.1. The x and y labels represent voxel space, each coordinate will be a unique voxel. The colour bar metre on the right indicates the activity level (BOLD response) of each voxel, with yellow indicating high BOLD response levels and dark blue representing low BOLD response levels. From this graph we can infer a relationship between the conditions and BOLD responses in the brain. It is clear that the different test conditions produces different BOLD responses for each voxels. This concludes Task 3.

Thank-you for reading

