1 Vector Manipulation

1.1 Basics

- MATLAB is often preferred because of the ease with which one handles vectors and matrices. Here are definitions of a 1x3 and a 2x3 matrix, respectively.

\[ a = [10 \ 20 \ 30]; \]
\[ b = [10 \ 20 \ 30; \ 100 \ 200 \ 300]; \]

Not placing a semicolon at the end of a line, will display the output of the line (otherwise, you will not see anything). Sometimes, the output might be huge, so you might want to use the semicolon.

- \( a(1) \) refers to the first element in \( a \), which is 10. Similarly, \( b(2,3) \) refers to the element at the second row, third column, in \( b \), which is 300.

We can also refer to a set of elements. \( a(2:3) \) is a 1x2 matrix consisting of the second and third elements of \( a \). \( b(1,1:3) \) refers to the first, second and third columns in \( b \), in the first row. So \( b(1,1:3) \) is the 1x3 matrix consisting of \([10 \ 20 \ 30]\).

- The colon operator can be used to specify a range of values. \( 1:10 \) generates all numbers from 1 to 10, in increments of 1. \( 0:.001:(2\pi) \) generates all numbers from 0 to \( 2\pi \), in increments of 0.001.

- \texttt{size(a)} returns the 1x2 matrix, where the first element is the number of rows in \( a \), and the second element the number of columns in \( a \). So \( b \) will be the 1x2 matrix consisting of the numbers \([2 \ 3]\), because \( b \) has 2 rows and 3 columns.

- \texttt{numel(b)} will instead produce the total number of elements in a matrix, over all rows and columns. For \( b \), this is 6.

- \texttt{reshape(b,m,n)} will produce a matrix with \( m \) rows and \( n \) columns, with the same elements as \( b \). The product of the number of rows and columns in \( b \) must be equal to \( m*\), for this to work.

For example, \texttt{reshape(b,1,6)} will convert the 2x3 matrix \( b \) into a 1x6 matrix (which can be thought of as a 6-dimensional vector). Since any vector can be thought of as a signal, and since we are dealing primarily with transmitting, processing and receiving signals, this is a useful operation that can convert any data into a (digital) signal.
1.2 Vector Math

- $b.^2$ will produce a 2x3 matrix, where each element is the square of the corresponding element in $b$. Similarly, $c.*d$ will produce a matrix with the same dimensions as $c$ and $d$, where each element is the product of the corresponding elements of $c$ and $d$. Clearly, $c$ and $d$ have to have the same dimensions for this to work.

- Other operations also usually operate on the entire matrix, one element at a time. $c+d$ will be the element-by-element sum of the matrices $c$ and $d$, and the result will be the same dimensions as $c$ (or $d$). $10*c$ and $c-1$ will multiply each element in $c$ by 10, and subtract 1 from each element in $c$, respectively.

1.3 Some Functions that Operate on Vectors

- $\sqrt{a}$ and $\text{abs}(a)$ will find the square root and absolute value for each element in $a$. The answer will have the same dimensions as $a$.

- Other common functions such as $\sin$, $\cos$, $\tan$, $\log$, $\log_{10}$ and so on behave in a similar manner.

- If $a$ is a vector (that is, a matrix with only 1 row or 1 column), $\text{sum}(a)$ and $\text{prod}(a)$ will find the sum and product of the elements in $a$, respectively. The answer will be a scalar.

- $\text{max}(a)$, $\text{min}(a)$ will return the maximum and minimum value in $a$, respectively (which is a scalar, if $a$ is a vector).

If $a$ is a matrix, then these commands will return a vector, where each element is the maximum (or minimum) value for each column of $a$. Hence, if we want to find the maximum value in a matrix, we should use $\text{max}(\text{max}(a))$, which will produce a scalar that is the largest element in the matrix $a$.

- $\text{find}(a>10, n)$, will find the indices of the first $n$ elements of $a$ that are greater than 10. We can use any condition instead of $a > 10$, such as $a==0$ (a is equal to zero), $a\neq0$ (a is not equal to zero), $\text{abs}(a) >= 1e-3$ (the absolute value of $a$ is greater than or equal to $0.001$), and so on.

For example, if $a = [2\ 4\ 6\ 8\ 10\ 12\ 14\ 16]$, and we would like to find the first element of $a$ that is bigger than 10, we would use $\text{find}(a>10, 1)$, which will produce the value 6 (because $a(6) = 12$ is the first number bigger than 10). If we instead directly wanted to access the first element that is bigger then 10, we would use $a(\text{find}(a>10, 1))$, which will produce the value 12.

If we wanted to access all elements of $a$, after the point when $a$ is bigger than 10, we would use $a(\text{find}(a>10, 1):\text{numel}(a))$, which will produce the result of $a(6:8)$, which is the vector $[12\ 14\ 16]$. This last operation is useful, when we have a signal with a lot of “silence” or noise at the beginning, and we would like to find the point at which this silence ends. If are recording from a microphone, and the first few seconds
of the recorded signal has no real data, but after that we have some recorded audio, we can say \( a(\text{find}(\text{abs}(a)>1e^{-3}, 1) : \text{find}(\text{abs}(a)>1e^{-3}, 1)+1000) \), which will look for the first element in the recorded signal \( a \) that is bigger than 0.001, and then take the next 1000 elements from that point.

### 1.4 Generating Standard Vectors

- \( \text{ones}(m,n) \) and \( \text{zeros}(m,n) \) will each generate an \( m \times n \) matrix of only the value 1, or 0, respectively. We can combine these commands to generate signals. For example, \([\text{zeros}(1, 100) \ \text{ones}(1, 100) \ \text{zeros}(1, 100)]\) will generate a 1x300 vector representing a “square pulse”.

- \( \text{linspace}(a, b, n) \) will divide the interval \([a,b]\) into \( n-1 \) equal pieces. So \( \text{linspace}(0, 1, 11) \) will be the vector \([0 \ 0.1 \ 0.2 \ 0.3 \ 0.4 \ 0.5 \ 0.6 \ 0.7 \ 0.8 \ 0.9 \ 1]\), and \( \text{linspace}(0, 1, 3) \) will produce the vector \([0 \ 0.5 \ 1]\).

### 2 Plotting

- \( \text{plot}(a) \) will simply plot a graph of the values \( a(1), a(2), \) and so on on the y-axis, and 1, 2, 3, and so on on the x-axis. \( \text{plot}(-10:10, a(1:21)) \) will plot the values \( a(1), a(2), \) and so on until \( a(21) \) on the y-axis, and the numbers -10, -9, -8, and so on until 10, on the x-axis. The number of elements on the x and y axes must be the same, for this to work.

- \( \text{plot}(-10:10, a(1:21), \text{‘r’}) \) will plot the same graph, but in red color. (Use \text{help plot} to find out what other colors and variations there are.) You can use File->Save As in MATLAB from the figure window to save your graph in your choice of file format.

- Use the \text{figure} command to create a new figure window. All future plot commands, will go to the last accessed window, so if you create this new window, all new plot commands will display on this window until you click on another figure.

- \text{hold on} will cause the figure to retain the current plot. Any future plots will be added onto the current plot, and will be displayed along with the current plot. If you don’t use this command, any plot command will erase the current plot in the figure window.

- You can use the zoom buttons to zoom in/out and view your plot. If you click on the arrow button, and double click on the plot, you will be able to adjust the axis scale, plot parameters, and other settings.

### 3 Images

- \( a=\text{imread(‘test.jpg’)}; \) will load the image file test.jpg stored in the Documents->MATLAB folder. You can use any .jpg file you like for this. The matrix
a will be the same dimensions of the image, but will contain red, green and blue values per pixel if this is a color image.

- Use \( b=\text{rgb2gray}(a) \); to convert the image loaded in a into a grayscale image, and store it in b. Now b will be a matrix of the same dimensions of the image, and each element will be a number between 0 and 255, where 0 represents black, 255 represents white, and all numbers in between represent various shades of gray. This form is convenient to work with.

- You can operate on this image as though it were an ordinary matrix. For instance, \( 2*b \) will be an image that is twice as “white”, because higher numbers represent white in the image. Recall that \( \text{max}(\text{max}(b)) \) will be the highest number in b, so \( \text{max}(\text{max}(b)) - b \) will create a “negative image”, where white will become black, and black will become white. Sometimes, images will invert to the negative while processing, so this is a useful operation.

- \( \text{imshow}(b) \) will display an image in the current figure window. \( \text{imshow}(b, []) \) will display the image, but will automatically adjust the brightness/contrast. This is useful, if the image has been received from a communication channel, and has been attenuated. For example, each pixel, instead of having values from 0 to 255, might only have values from 0 to 100, so \( \text{imshow}(b, []) \), on the other hand, will automatically adjust the display brightness to compensate, and display a much better image. For example, we can use \( \text{imshow}(-b, []) \) to view the negative image, without having to use \( \text{max}(\text{max}(b)) - b \).

- We can use the \texttt{reshape} command to convert an image (matrix) into a signal (vector), and vice-versa.

4 Audio

- \( \text{wavplay}(a, Fs) \) will play the waveform stored in the vector a, where Fs is a number representing the sample frequency (between 1 and 44100 Hz). If we connect the headphone output into any analog signal processing or modulating equipment, we are basically taking our digital signal a in MATLAB, and converting it into an analog signal available at the headphone output.

Since the maximum sample frequency is 44100 Hz, the maximum signal frequency we can produce is about 22 KHz. We will later use more sophisticated equipment that lets us use signals of much higher frequency.

- \( a=\text{wavrecord}(\text{numsamples}, Fs) \); will record \texttt{numsamples} samples at the rate of Fs samples per second, and store that in the vector a. a will have \texttt{numsamples} elements. For example, if we wanted to listen to 10 seconds of audio, at 4000 samples per second, we would use \( a = \text{wavrecord}(10*4000, 4000) \); to generate the vector a of 40000 elements. We can then \texttt{plot} this signal, \texttt{reshape} it to the dimensions of an image, use \texttt{wavplay} to play it back (at the same sample rate), and so on.
If you have 2-channel (stereo) microphones, use `a=wavrecord(numsamples, Fs, 1);` to record only from the first channel. You can select which microphone to record using the Windows audio control panel (usually found under playback or recording devices, after right clicking on the audio icon on the taskbar tray), and adjust the microphone volume. There is usually more than one mic: a line-in mic (which lets you feed in a signal through the mic input jack), and the integrated “open” microphone that records any audio (such as speech) in the vicinity of the laptop.