1 Help and documentation

Help for a specific function may be viewed by typing

```matlab
>> help <function name>
>> doc <function name>
```

in the Matlab command window.

2 Creating variables

Some examples of creating arrays and vectors follow

```matlab
>> x = 1:10; % the x vector consists of the integers from 1 to 10
>> y = 0:0.1:0.9; % the y vector has values ranging from 0 to 0.9 in steps of 0.1
>> A = eye(3); % create a 3x3 identity matrix
>> B = ones(3,2); % a 3x2 matrix of ones
>> whos
```

The “%” denotes comments - note that what follows the percent is ignored. The “whos” command displays the variables currently available.

Other useful functions:

- `zeros`
- `rand`
- `randn`
- `linspace`
- `logspace`

3 Arithmetic

Some examples of creating arrays and vectors follow

```matlab
>> a=x+y; % add the vector x and y
>> b=x*y; % matrix multiply the vectors x and y
>> c=x.*y; % element-wise multiply x and y
>> d=B'; % transpose B
>> help +; % show help for arithmetic operations
```
4 Indexing

Accessing values in arrays and vectors

```matlab
>> a=x(2);          % get the second value in the vector x
>> b=A(2,1)         % the the value in the second row and first column
                    % of A
>> c=A(:,1);        % get the first column of A
>> i=1:3;d=x(i);    % make a new vector consisting of the first through
                    % third elements of x
>> inds=find(y>0.3); % get the indices of values in y whose values are
                    % greater than 0.3
>> e=y(inds);       % get those values
```

Other useful functions:

- nonzero
- ind2sub
- sub2ind

5 Dimensions

Dealing with multi-dimensional data

```matlab
>> A = rand([4 4 4 4]); % create a 4D array: think - [x y z t]
>> a1= A(:,:,,:,1);    % get the first time volume of A
>> a1_3= a1(3,:,:,:); % get the third slice of a1
>> A13 = A(3,:,:,1);   % get the same data from A directly.
                    % note: A13 is 1x4x4
>> A13 = squeeze(A13); % get rid of the singleton dimension
                    % now A13 is 4x4
>> [x,y,z,t]=ndgrid(1:4,1:4,...
                   1:4, 1:4); % these variables can be used for
>> B = log(t.*(x.^2 + ...
      y.^2 + z.^2)); % function evaluation
>> mesh(squeeze(B(:,1,1,:))); % visualize a subset of B
```

Other useful functions:

- shiftdim
- flipdim
- fliplr
- permute
- ipermute
- sort
- cat
- squeeze
- interp3
- interpn
6 Signal/Image Processing

Matlab’s signal and imaging processing toolboxes feature some very useful functions.

```
>> A=zeros(256); A(36:63,36:64)=1;  % create a simple image
>> h = ones(5,5) / 25;              % make a simple linear filter
>> I2 = imfilter(A,h); imagesc(I2); % filter the image
>> B = imrotate(A,25,'bilinear');  % rotate the image
>> imagesc(B);
>> F = fft2(B);                     % compute the fourier transform
>> Fm=log(abs(F));imagesc(Fm)        % compute its log-magnitude
```

![Convolution and Cross-Correlation](conv2 xcorr2)

![Image Cropping and Resizing](imcrop imresize)

![Making and Applying Transformations](maketform tformfwd tforminv)

![Image Histogram and Correlation](imhist corr2 mean2)

7 Visualization

We’ve already seen a simple example in the previous section. The following are more techniques.

```
>> [i,j,k] = ndgrid(-10:0.1:10,...
    -10:0.1:10,-10:0.1:10);
>> A=exp(-(i.*i)/5 - (j.*j)/7 - (k.*k)/9);  % create a simple image
>> imagesc(A(:,:,50));                     % make a picture
>> axis image;                             % ensure axes are scaled appropriately
>> contour(A(:,:,50),[0.8 0.4 0.2]);       % a 2d contour plot
>> h = slice(A,[],[40 60],[49]);           % slices in 3d
>> set(h,'EdgeColor','none');              % make easier to see
>> p=patch(isosurface(A,0.5),...              % a 3d surface
    'FaceColor','red');
>> set(p,'EdgeColor','none');              % make pretty
>> camlight; lighting phong;
```