

## Matlab tips

### 1. Basic Matlab commands

```
x = randn(20,1); % A vector with 20 N(0,1) random numbers.
whos; % List current variables, with size information.
[n m d] = size(x); % Get size of x.
x(end-4:end); % Get the five last elements in x.
```

### 2. Multiple quadratic forms.

We want to calculate  $b_i = \mathbf{x}_i^T \mathbf{A} \mathbf{y}_i$ , for a large number of vectors  $\mathbf{x}_i$  and  $\mathbf{y}_i$ , stored as matrices (x and y) with one vector per column (i.e.  $\mathbf{x}(:, i) = \mathbf{x}_i$ ).

Start by calculating  $\mathbf{z} = \mathbf{A} * \mathbf{y}$ . Then each  $b_i$  is given by  $b(i) = \mathbf{x}(:, i)' * \mathbf{z}(:, i)$ . This can be written as  $b(i) = \text{sum}(\mathbf{x}(:, i) .* \mathbf{z}(:, i), 1)$ , where  $.*$  is element-wise multiplication, and the entire vector of  $b_i$ 's, is given by  $\mathbf{b} = \text{sum}(\mathbf{x} .* \mathbf{z}, 1)$ , eliminating the need for a for-loop.

### 3. Covariance matrix estimation for vector-valued pixels.

```
x=double(lanread('rio.lan')); % x: m-n-d
mnd=size(x); % mnd=[m,n,d]
mn=mnd(1)*mnd(2); % The total number of pixels.
y=colstack(x); % y: mn-d
mean_y=mean(y,1); % mean_y: 1-d
y=y-repmat(mean_y,[mn,1]); % Mean subtracted.
Sigma=(y'*y)/mn; % Covariance estimate, d-d.
```

### 4. Some plot commands.

#### (a) Functions of two variables.

```
x=linspace(-5,5,25); % 25 linearly spaced values from -1 to 5.
y=logspace(-1,1,25); % 25 log spaced values from 10^-1 to 10^1.
[X,Y]=meshgrid(x,y); % Create a 2D grid.
f=exp(-1./Y.*(2+X.^2))./Y; % A function of two variables.
surf(X,Y,f); % Plot the function.
```

#### (b) Images, Scatterplots and histograms.

```
x=double(lanread('rio.lan'))/255; % x: m-n-d
y=colstack(x); % y: mn-d

imagesc(x(:,:,1)) % Plot with scaled image intensities.
image(rgbimage(x(:,:,1,2,3)))) % True-colour image.
hist(y(:,1),256) % Histogram with 256 "bins".
hist2(y(:,1:2),100); % 2D histogram, channel 1 vs 2.
hist2(y(:,[1 3]),100); % 2D histogram, channel 1 vs 3.
```

```

plot(y(:,1),y(:,2),'.') % Scatter plot, channel 1 vs 2.
I=1:50:size(y,1); % Index vector.
plot(y(I,1),y(I,2),'.') % Scatter plot with fewer data points.
plot3(y(I,1),y(I,2),y(I,3),'.') % 3D scatter plot.

```

## (c) Surfaces and colouring.

```

z=peaks(100); % A sample surface.
surfl(z) % Surface 3D-plot with "realistic" illumination.
surf(z) % Surface 3D-plot.
shading flat % Don't show grid lines.
shading interp % Interpolate colours.
colorbar % Adds colorbar relating colours to z-value.
caxis([0 10]) % Changes range of colouring.

colormap(gray(256)) % Set gray colormap with 256 levels.
colormap(jet(256)) % Reset to default colormap.
help graph3d % Functions for three dimensional graphs,
% includes a list of colormaps.

```

## 5. Visually appealing plots of LANDSAT data.

```

x=double(lanread('rio.lan'))/255;
image(rgbimage(x(:,:, [1,2,3]))) % BGR mapped to BGR
image(rgbimage(x(:,:, [2,3,4]))) % Blueshifted data, mapping
% G->B, R->G, NearIR->R. Nice.
image(rgbimage(x(:,:, [3,4,5]))) % Even more blueshifted data, mapping
% R->B, NearIR->G, IR->R. Nicer.

```

The visual spectrum image (i.e. the first plot above) is visually dark. A visual improvement can be obtained by removing the highest and lowest values, and rescaling:

```

% Find the 2.5 and 97.5 percent quantiles for each component:
xc = sort(colstack(x));
xc = xc(round(size(xc,1)*[0.025,0.975]),:);
% Rescale each component:
y = zeros(size(x));
for k=1:size(x,3)
    y(:,:,k) = (x(:,:,k)-xc(1,k))/(xc(2,k)-xc(1,k));
end
% Truncate the data:
y = max(0,min(1,y));

```

The new image data is usually more visually appealing than the original, both for the visual spectrum and for blueshifted images.

## 6. Some course specific Matlab functions.

- General files
  - fmsn20path** Set path to fmsn20-files.
- Classification
  - kmeans** Classify data using the K-means algorithm.
  - manualclass** Mark image pixels as belonging to different classes.
  - normmix\_classify** Classify data in a Gaussian mixture model.
  - normmix\_em** Estimate parameters in a Gaussian mixture model.
  - normmix\_kmeans** Use a single K-means to get a random initial estimate.
  - normmix\_posterior** Helper function for normmix\_em.
- Fields
  - covest\_ls** Estimates a Matérn covariance with the Least Squares method.
  - covest\_ml** Estimates a Matérn covariance with the Maximum Likelihood method.
  - covest\_nonparametric** Non-parametric covariance estimator.
  - distance\_matrix** Calculates the distance matrix for a set of locations.
  - matern\_covariance** Calculates Matérn covariances.
- Graphics
  - globe\_plot** Plot a field defined on a spherical grid.
  - hist2** Calculate and display 2D-histograms.
  - landsatimage** Make an RGB image matrix from LANDSAT data.
  - rgbimage** Make an RGB image from several weight images.
  - trisphere2anglegrid** Map a spherical field to a flat projection.
- Markov random fields
  - calc\_gmrf\_props** Helper function for calculating pointwise GMRF properties.
  - gmrf\_mcmc\_skeleton** Simulate GMRF parameters with MCMC, and perform calculations
  - gmrf\_negloglike\_skeleton** Calculate the GMRF data likelihood.
  - gmrf\_param\_hessian** Calculates the hessian for a negated log-likelihood
  - gmrf\_param\_map** Finds the MAP estimate in a simple field model.
  - gmrfprec** Constructs a precision matrix for a GMRF on a regular grid
  - igmrfprec** Constructs a precision matrix for a 1:st or 2:nd order IGMRF
  - igmrfprec\_sphere** Constructs a precision matrix for spherical IGMRF
  - matern\_prec\_matrices** Calculate matrices need to build Matérn precisions
  - mrf\_gaussian\_est** Weighted estimate of Gaussian parameters
  - mrf\_gaussian\_post** Posterior alpha-parameters in a MRF with Gaussian data
  - mrf\_icm** Estimate the MAP field from an MRF model
  - mrf\_ple** Pseudo-likelihood estimation of MRF parameters
  - mrf\_sim** Simulate a samples for a MRF
  - reorder\_trigraph** Reorder nodes in a triangular graph for nice Cholesky

- Miscellany
  - colstack** Column stack multidimensional images.
  - icolstack** Revert column stacking of multidimensional images.
  - defstruct** Fill struct with default values.
  - fillholes** Fill NaN-holes in an image.
  - helmert** Construct a Helmert (sub-)matrix.
  - indicshape** Compute an indicator image for a shape.
  - lanread** Read Landsat data file type .lan
  - noise01** Modify binary image with noise.
  - pca** Perform Principal Component transformation of data.
  - polyimage** Computes an aliased indicator image for a polygon.
  - reshape** Compute the reshapes of landmark objects.
  - reparameterise** Reparameterises a shape to obtain equidistant landmarks.
  - rotmat** Generate a rotation matrix.
  - simplespline** Compute a spline interpolation of a sequence of landmarks.
  - sphereHarmonics** Creates spherical harmonic functions.
  - triSphere** Triangulates a sphere.
  - vec** Vectorise a landmark matrix.
  - ivec** The inverse of vec, anti-vectorise landmarks.
- Shape analysis
  - gmrf\_snake** Estimate a closed shape using a GMRF-snake model.
  - mark** Mark landmarks in an image.
  - procrustes\_align** Perform Procrustes alignment of landmark data.
  - procrustes\_dist** Compute Procrustes distances.
  - procrustes\_mean** Estimate the Procrustes mean.
  - shape\_tangent\_inv** Compute pre-shapes from shape tangent coordinates.
  - shape\_tangent** Compute vectorised shape space tangent coordinates.
  - snake\_neg\_loglike** Calculate negative log-likelihood for a snake.
- Warping
  - tps\_prep** Precompute data for use in tps\_pull and tps\_push
  - tps\_pull** Computes a pull warp
  - tps\_push** Computes a push warp
  - Older low level routines
  - tps\_warp0** Deform an image using a TPS warp.
  - tps\_warp0\_prep** Precompute data for use in tps\_warp0
  - tps\_warp1** Deform an image using an inverse TPS warp.
  - tps\_warp1\_prep** Precompute data for use in tps\_warp1