EL2310 – Scientific Programming

Lecture 2: Matlab as a Tool

Ramviyas Parasuraman (ramviyas@kth.se)

Royal Institute of Technology – KTH

Overview

Lecture 2: Matlab as a Tool

- Wrap Up
- Matrices (continued)
- Linear Algebra
- Plotting & Visualization
- Tasks for Home

Last time

- To get help: `help, lookfor, doc`
- To load/save/clear variables: `save, load, clear`
- To “write” a diary: `diary`

Last time

- Initialize a vector: `A = [1 2 3];`
- Initialize a matrix: `B = [1 2 3; 4 5 6; 7 8 9];`
- Simple operations on scalars, matrices and vectors

  `C = A * B; D = B'; E = B.*D (element-wise);`
- Access values of vectors and matrices

```
A(0);
A(end);
B(2,2); B(4);
B(2,:);
B([1 3],:);
```

Loading data

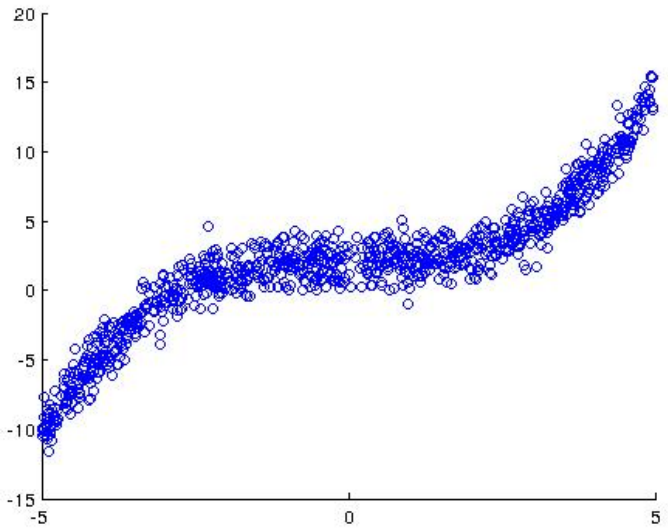
- We saw how you can load saved variables with
  `load <filename>`
- You can easily load data directly into MATLAB if the data is matrix-like, i.e. same number of columns for each row
- To load a file “filename.txt” do
  `load('filename.txt')`
- This will put the loaded matrix into a variable filename (the name of the file).
- Can also do
  `d = load('filename.txt');`

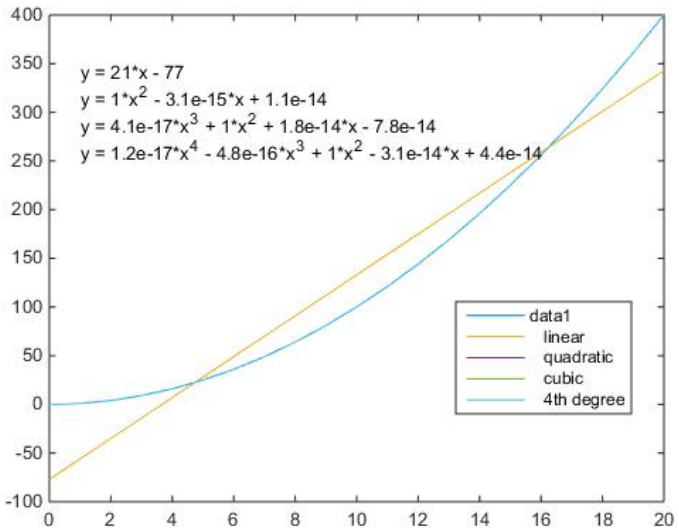
Elementary matrices

- Predefined elementary matrices. `help elmat`
- Examples:
  Identity matrix: `I = eye(n);`
  Zero-matrix: `Z = zeros(n,m);`
  One-matrix: `O = ones(n,m);`
  If the second dimension is omitted, creates a square matrix.
- Reshape matrices with `reshape(X,M,N)`
- Array info: `size(A), length(A), numel(A), isempty(A),...`

Some tools to deal with real data:

Fast fitting and statistics in the Tools menu of Figure

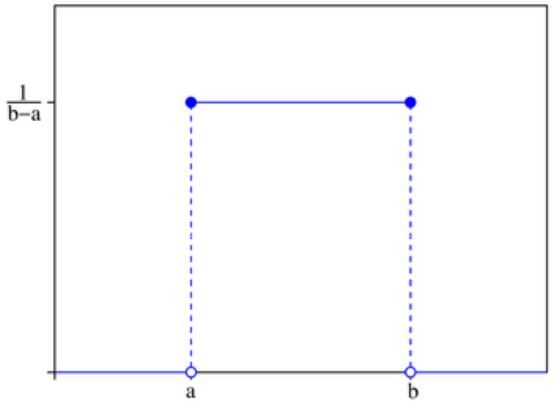




Solving linear systems

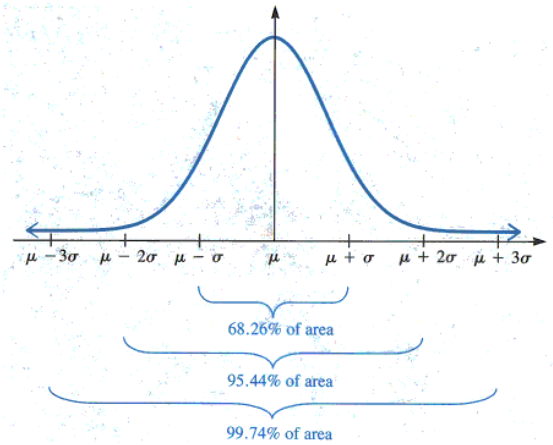
- You can solve a matrix equation $AX = B$ using $X = A\backslash B$.
  If B is invertible, this is the same as $X = A^{-1}B$, otherwise the solution is a solution in the least squares sense.

Random matrices



- Can easily create random matrices in $[0, 1]$
- Uniform distribution
  `rand(n,m)`
- Normal distribution
  `randn(n,m)`

- How to generate 1000 values
  from a normal distribution
  with mean 1 and standard deviation 2?



Sequences

- Enumerate `v = [1 3 7];`
- Colon notation. Ex: `v = 1:9;` `v = 1:2:9;`
- More general linearly spaced vectors
  - `v = linspace(start_value, end_value, N);`
  - Do not have to calculate the step yourself
- Logarithmically spaced vector
  - `v = logspace(start_exp, end_exp, N);`
  - Calculates 10 to the power of these values.
  - logspace(x_1, x_2, N) = $10^{\text{linspace}(x_1, x_2, N)}$

Creating matrices from vectors

- Diagonal matrices can be created with `diag(<vector>)` argument
- You can shift the vector up and down from the diagonal
  `diag(<vector>, k)`
  where $k > 0$ means shifting up and $k < 0$ mean shifting down
- You can also create diagonal block matrices with
  `blkdiag(M1, M2, ...)`
- `A=[1 2 3;4 5 6;7 8 9]; diag(A)` ?

Manipulating matrices

- Get lower triangular part
  `tril(A)`
- Get upper triangular part
  `triu(A)`
- Flip a matrix upside down
  `flipud(A)`
- Flip a matrix left/right
  `fliplr(A)`
- Rotate matrix 90° anti-clock
  `rot90(A)`
- rotate clockwise?

```
>> A=[1 2 3;4 5 6;7 8 9];
>> tril(A)

ans =

     1     0     0
     4     5     0
     7     8     9

>> triu(A)

ans =

     1     2     3
     0     5     6
     0     0     9
```

Finding elements

- You can find non-zero elements
  `[ind] = find(A)`
  returns the linear index (single index)
- Can get the subscripts by providing two output arguments
  `[ii, jj] = find(A)`
- Can replace test for non-zero with a logic expression such as
  `[ii, jj] = find(A>3)`
- Note that `A>3` is a matrix of the same dimension as A and with 1-elements for each element in A that is $>$ 3 and 0 for the rest

Linear algebra (some examples)

- Easy to calculate basic linear algebra
- Inverse: `inv(A)`
- Determinant: `det(A)`
- Rank: `rank(A)`
- Trace: `trace(A)`

Linear algebra: Eigenvalues

- Finding eigenvalues
  `eig(A)`
- Getting eigenvalue and vectors
  `[V,D] = eig(A)`
  `V` full matrix contains the eigen vectors (columns) and `D` is a diagonal matrix with the eigenvalues on the diagonal
  Fulfills $AV = VD$

Linear algebra: Singular value decomposition (SVD)

- Calculating svd is simple
  `[U,S,V] = svd(A)`
- Fulfills $A = U * S * V^T$
- `s = svd(A)` gives the singular values

Square root matrix

- Square root matrix fulfills $A = XX$
- Calulated with
  `X = sqrtm(A);`
- Remember: Element wise multiplication with `.*`

```
>> A = [1 2;3 4]

A =

     1     2
     3     4

>> As = sqrtm(A)

As =

   0.5537 + 0.4644i   0.8070 - 0.2124i
   1.2104 - 0.3186i   1.7641 + 0.1458i

>> As*As

ans =

   1.0000 + 0.0000i   2.0000
   3.0000 + 0.0000i   4.0000

>> As.*As

ans =

   0.0909 + 0.5143i   0.6061 - 0.3428i
   1.3636 - 0.7714i   3.0909 + 0.5143i
```

More operations

- Easy to calculate mean, standard deviation, etc.
- Applies to a vector or columns of a matrix
- Mean value: `mean(v)`
- Standard deviation: `std(v)`
- Min value : `min(v)` (also `min(A,2)`)
- Max value : `max(v)` (also `max(A,2)`)
- Sum : `sum(v)`
- Difference : `diff(v)`
- Cumulative sum: `cumsum(v)`
- Covariance: `cov(X)`

More operations cont.

- Useful tip: Convert a matrix to column vector `A(:)`
  What's `min(A)` and `min(A(:))` if `A` is a matrix?
- Additional parameter specifies dimension:

```
mean(A, 1)
min(A, [], 1)
```
Why `[]`?
```
max(A, [], 2)
sum(A, 1)
```

Plotting data

- Plotting data with
  `plot(x,y)`
- With one argument the x-axis will be the vector index and the y-axis the value of the input vector
- Can specify color and type of line/points, e.g.
  `plot(x,y,'r.')` to get a red dot for every data point
- For more information do
  `help plot`
- Example: Plot $\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$, i.e. a normal distribution with standard deviation σ and mean value μ.

Titles, labels, legends, etc

- Label the axes with
```
xlabel('text on the x-axis')
ylabel('text on the y-aixs')
```
- and give a title with
```
title('Some nice title')
```
- You can change the font size by adding extra arguments
```
xlabel('text on the x-axis', 'FontSize',20)
```

Handles and set/get

- Calls to graphics functions return a “handle”
- Can use this handle to set/get properties
- `h = title('Some nice title');`
- List properties with
  `get(h);`
- Set property with
  `set(h, 'FontSize', 20);`
- Get current handle:
  `gcf` - figure
  `gca` - axes

Plotting continued

- You can plot more than one thing at a time:
  `plot(x1, y1, x2, y2)`
- Each pair assigned it own color automatically
- You can manually specify color/marker for each:
  `plot(x1, y1, 'r', x2, y2, 'b')`
- Every plot call will clear the figure
- Use `hold on` and `hold off` to stop from clearing

```
hold on
plot(x1,y1)
plot(x2,y2)
hold off
```

More plotting

- You can provide labels for your data with `legend`

```
plot(x1, y1, x2, y2)
legend('data set 1', 'data set 2')
```

- You can specify which figure window number: `figure(n)`
- You can clear a figure (the current one) with `clf`
- Can get grid with `grid`
- Can plot with one or both axis in logarithmic scale

```
semilogx(x,y)
semilogy(x,y)
loglog(x,y)
```

Creating histograms

- Displaying histograms:
  `hist(v, b)`
  where `v` is vector with data and `b` is number of bins.
- If you want the histogram data use:
  `[n,x] = hist(v,b)`
  where `n` are frequency counts and `x` are bin locations.
- You can plot histogram data with `bar(x,n)`

Exercise 1

- Load data from "gyrosignal.txt"
  `http://www.csc.kth.se/~ramviyas/gyrosignal.txt`
- Collected from a gyro while standing still
- Format: Each row contains `time` and `gyrosignal`
- The time is in seconds
- The gyrosignal is in rad/s (maybe biased)

- Task:
  1. Plot as it is
  2. Remove any bias and then plot over the previous plot
  3. Integrate the signal to verify that the angle is zero at the last time step.

Modifying the axis

- MATLAB will automatically choose the axis range for you,
- but in some cases this is not what you want.
- Set using: `axis([x_min x_max y_min y_max])`
- Get current axis settings with: `a = axis;`
- Same x/y unit (aspect) size `axis equal`
- Square figure with `axis square`
- Fit to figure `axis normal`
- You can “turn off” the axis with `axis off`

Saving/printing a figure

- You often want to save a figure
- This can be done from the figure menu or with `print` command.

- To create an eps file, select desired figure and do
  `print -deps <filename>` (black/white)
  `print -depsc <filename>` (color)
- For png: `print -dpng <filename>`
- For print options do `help print`
- If you want high quality prints for your thesis/publications, check out matlab2tikz

Getting input from a figure

- You can get information (coordinates) by clicking inside figure
- Use command

  `xy = ginput`

  (pressing ENTER terminates the command)

  or

  `xy = ginput(n)`

  (if you know beforehand how many data points)

`drawnow` and `pause`

- To force a figure to display its content now (flush event queue), use
  `drawnow`
- To pause execution and wait for ENTER in command window, use `pause`
- You can pause for *n* seconds with
  `pause(n)` (e.g. `pause(0.1)` to pause 0.1s)

Subplots

- Easy to put many plot in the same figure with `subplot(n,m,k)`
- Sets up for n by m plots in a figure and prepares to add plot k
- Example

```
subplot(2,1,1), plot(x1,y1)
subplot(2,1,2), plot(x2,y2)
```

3D plots

- Several functions to plot in 3D
- `plot3(x,y,z)`
- `mesh(X,Y,Z)`
- `surf(X,Y,Z)`
- `contour(X,Y,Z)`
- `mesh`, `surf` and `contour` plot the matrix Z against the values of X and Y.
- You can create values for X and Y with
  `[X,Y] = meshgrid(x,y);`
  where x and y are vectors and X and Y are matrices
- See also `colormap`

Exercise 2

- Display the function $z = 1 - x^2 + y^2$
- Use the interval $x, y \in [-1, 1]$

Data Analysis

- Let's generate, plot and analyse data with Matlab

Exercises to try

- Generate a vector of normally distributed random samples
- Compute the mean and standard deviation from the samples
- Plot the histogram of the samples (use `hist`)
- Generate two sequences of random samples and compute covariance

Exercises to try

- Generate a ”data set” using $x = 5 - 10 * rand(1, 1000)$ $y = 2 + 3 * x + randn(1, 1000)$.
- Save the result in a file *data.mat*.

Exercises to try

- Assume someone hands you the data generated in the previous task without information about how it was generated.
- Load and plot the (x, y) data to understand it (try *scatter*).
- Assume that you don't know how the data was actually generated. Try to fit line to the data (x, y) using just the data samples.
- Plot your line approximation

Exercises to try

- Create a ”data set” using
  x = 5-10*rand(1, 1000)
  y = 2+0.1*x.^3+randn(1, 1000).
- Plot the (x, y) data to understand it.
- Read about regression methods online and check useful matlab commands.
- Can you fit a non-linear function to the data?
- Quantify the error in your approximation compared to a simple line fit to this data?

Next time

- Functions and scripts in detail
- Basic programming

Group presentations

1. Principal component analysis (PCA): 8th Sep, 20 mins
2. Image processing in Matlab, sample implementation
3. Regression and classification, example implementation in Matlab
4. Deep learning concepts and sample implementation
5. Matlab and C/C++ integration - MEX files
6. Optimization of C/C++ code
7. Data compression algorithms in C/C++
8. Multithreading concepts in C/C++
9. GPU programming!

Matlab Project

- Will announce tomorrow through Bilda
- Please email me if you are not yet in Bilda.
- Deadline: 12th Sep (Mon)
- Help session: 7th Sep (Lab session), 1-3pm, Teknikringen 14, room 304.