

Topic 3

Overview of Matlab and Octave

References

- [1] www.mathworks.com
- [2] *Getting Started*, Matlab Documentation
- [3] www.octave.org

School of Computer Science & IT, RMIT

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects.

Today, MATLAB engines incorporate the LAPACK and BLAS libraries, embedding the state of the art in software for matrix computation. MATLAB has evolved over a period of years with input from many users.

In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science.

In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis. MATLAB features a family of add-on application-specific solutions called toolboxes.

Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

What Is MATLAB?



<http://www.mathworks.com>

“MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation.”

Typical uses include

- Math and computation
- Algorithm development
- Data acquisition
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning.

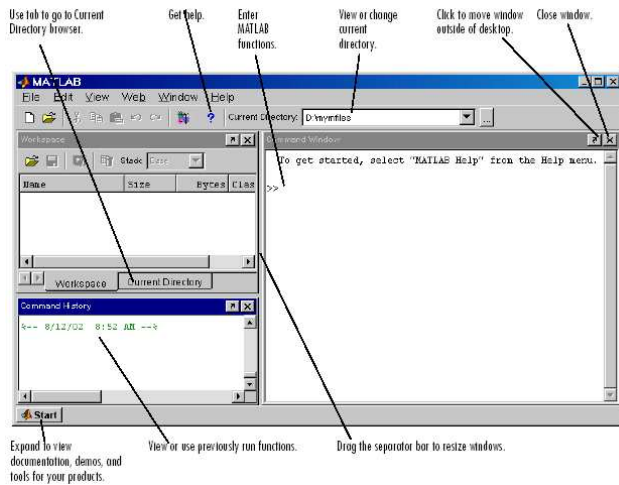
This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or Fortran.

The MATLAB System

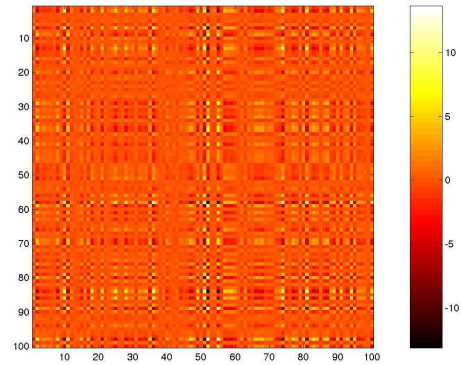
The MATLAB system consists of five main parts:

- Development Environment.
 - Tools to use MATLAB functions and files.
 - MATLAB desktop
 - Command Window and a command history,
 - An editor and debugger,
 - Browsers for viewing help, the workspace, files, and the search path.
- The MATLAB Mathematical Function Library.
 - Low and High level routines
- The MATLAB Language.
 - A high-level matrix/array language with control flow statements, functions, data structures, I/O, and object-oriented programming features.
- Extensive Graphics.
 - Low-level customisation and annotation
 - Build complete GUIs
- The MATLAB API.
 - C and Fortran programs can interact with MATLAB. Calling from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

The Matlab Desktop



This is a representation of the inverse of that matrix.



calculated in less than a second using the commands:

```
b = inv(a); % get Matrix inverse  
imagesc(b);colorbar; % also show legend as colorbar  
axis square;
```

While the numbers in the previous matrix were completely random, the elements in this matrix are anything BUT random. In fact, each element in this matrix **b** depends on every one of the ten thousand elements in the previous matrix **a**. But how do we know for sure if this is really the correct inverse for the original matrix?

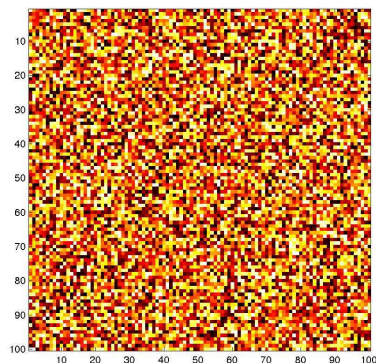
Example of use: Matrix Inverse

The best way to show off Matlab is through example.

This demo shows how to visualize matrices as images and uses this to illustrate matrix inversion.

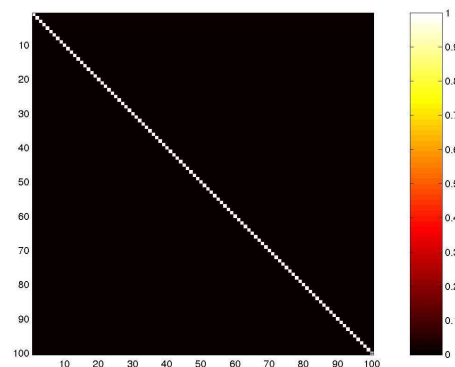
We create a matrix of random numbers and visualise the calculations on it as we go.

```
a = rand(N); % Create NxN 2D array of rands  
imagesc(a); % Display as image, autoscaling colormap  
colormap(hot); % Color = black-red-yellow-white  
axis square; % Make x and y axis scaling equal
```



Multiply the **a** and **b** together and see if the result is correct, because just as $3 \times (1/3) = 1$, so must $\mathbf{a} \times \mathbf{inv}(\mathbf{a}) = \mathbf{I}$, the identity matrix.

```
imagesc(a*b);colorbar;  
axis square;
```



This is the product of the matrix with its inverse: and sure enough, it has the distinctive look of the identity matrix, with a band of ones streaming down the main diagonal, surrounded by a sea of zeros.

By always keeping a feeling for the kinds of numbers and patterns you expect during calculation, the matlab tools can be used to confirm your calculations at every step of the way.

Solving an ODE in Matlab

We have previously looked at a second order DE for simple harmonic motion of the form

$$\ddot{x} = f(t, x, \dot{x})$$

which we can recast as

$$\dot{x} = u$$

$$\dot{u} = f(t, x, \dot{x}) = -\omega^2 x$$

Now, we saw how this leads to the equations

$$\frac{d^2x}{dt^2} - \omega^2 x = 0, \quad x = 0, \quad \frac{dx}{dt} = v_0, \quad \text{at } t = 0$$

and we know that the exact solution for this is

$$x = \frac{v_0}{\omega} \sin \omega t$$

Now let's solve this using Matlab's ODE functionality.

We first define the time span under consideration as

```
tspan = [0, 10*pi];
```

and $x_0 = 0$ at $t = 0$.

Octave



<http://www.octave.org>

GNU Octave is a high-level language, primarily intended for numerical computations. It provides a convenient command line interface for solving linear and nonlinear problems numerically, and for performing other numerical experiments using a language that is *mostly* compatible with Matlab. It may also be used as a batch-oriented language.

Octave has extensive tools for solving common numerical linear algebra problems, finding the roots of nonlinear equations, integrating ordinary functions, manipulating polynomials, and integrating ordinary differential and differential-algebraic equations. It is easily extensible and customisable via user-defined functions written in Octave's own language, or using dynamically loaded modules written in C++, C, Fortran, or other languages.

GNU Octave is also freely re-distributable software. You may redistribute it and/or modify it under the terms of the GNU General Public License (GPL) as published by the Free Software Foundation.

Octave was written by John W. Eaton and many others. Because Octave is free software you are encouraged to help make Octave more useful by writing and contributing additional functions for it, and by reporting any problems you may have.

We now need to define a function to be called by the solver. The function has two arguments: the first is the time span; and the second is an array of intermediate values – in our case, these are x and u .

```
function xdot = shm(t,xv)
xdot(1,1) = xv(2);
xdot(2,1) = -(omega^2)*xv(1);
```

We can then call the solver using the following code:

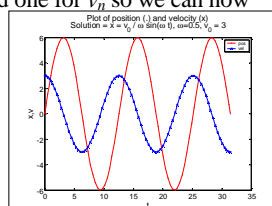
```
[t,x] = ode45(@shm, tspan, [x0;v0]);
```

where $\omega = 0.5$, $x_0 = 0$ is the initial position, and $v_0 = 3$ is the initial velocity. Note that we supplied the derivative 'behaviour' function, the time span, and the initial conditions. `ode45()` is a 5th order variable step size *Runge-Kutta* method. We could just as easily use `ode23()` for fewer function evaluations per iteration, or `ode23s()`/`ode45s()` which are optimised for *stiff* ODE's which converge slowly and need many evaluations to converge using `ode23()`/`ode45()`.

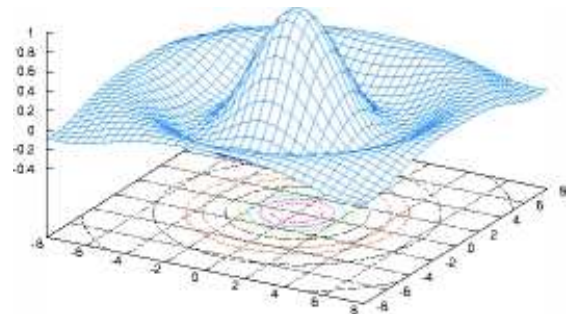
The result is supplied in two arrays t and x where t is a 1D array of time steps, and x is a 2D array with `length(t)` rows and two columns – one for x_n and one for v_n so we can now plot these two variables.

```
plot(t, x[:,1], 'x');
hold on;
plot(t, x[:,2], 'b');
title('.....');
xlabel('t'); ylabel('x,v');
```

It's that simple!



Octave Architecture Overview



Unlike Matlab, Octave uses Gnuplot for all of its graphics functions. While there are plotting commands within Octave that resemble those of Matlab, to do anything complex requires one to fall back to Gnuplot commands.

Creating a Matrix

To create a new matrix and store it in a variable so that it you can refer to it later, type the command

```
octave:1> a = [ 1, 1, 2; 3, 5, 8; 13, 21, 34 ]
a =
    1    1    2
    3    5    8
   13   21   34
```

Octave will respond by printing the matrix in neatly aligned columns. Ending a command with a semicolon tells Octave to not print the result of a command. For example

```
octave:2> b = rand (3, 2);
```

will create a 3 row, 2 column matrix with each element set to a random value between zero and one.

To display the value of any variable, simply type the name of the variable. For example, to display the value stored in the matrix b, type the command

```
octave:3> b
b =
  0.8026248  0.6931217
  0.5326290  0.0030976
  0.9793020  0.5955896
```