Introduction to MATLAB

Shaohao Chen
High performance computing @ Louisiana State University
Outline

- Overview
- Matrix and array
- Language fundamentals
- Programming
- Plotting and graphics
- Mathematics
- Exercises
Overview

- A high-level language and interactive environment for numerical computation, visualization, and programming.
- Analyze data.
- Develop algorithms.
- Create models and applications.
- More than a million engineers and scientists in industry and academia use Matlab.
- The language of technical computing.
- Commercial!
Key Features

- Interactive environment for iterative exploration, design, and problem solving.
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration, and solving ordinary differential equations.
- Built-in graphics for visualizing data and tools for creating custom plots.
- Development tools for improving code quality and maintainability and maximizing performance.
- Tools for building applications with custom graphical interfaces.
- Functions for integrating MATLAB based algorithms with external applications and languages such as C, Java and Microsoft Excel.
Matlab vs. C/C++/Fortran

- Matlab compared to C/C++ or Fortran
  - a higher level of programming language
  - similar grammar to C/C++ (but more simplified)
  - more tools and built-in math functions
  - reach a solution for a math or numerical problem faster
  - better graphical interface
  - slower computation speed for most numerical problems, not suitable for massive computation.
  - good for post analysis of data and visualization.
Matlab vs. Mathematica

- Matlab compared to Mathematica
  - both have very good built-in math functions
  - both have good graphical interface
  - better at numerical treatments
  - worse at analytical treatments
    (needs additional Symbolic Math toolbox)
  - more popular in engineering
  - less popular in science
Run MATLAB on HPC clusters

- **Log in and set up environments:**
  
  ssh -X username@mike.hpc.lsu.edu
  
  softenv -k matlab
  
  vi .soft
  
  Add one line in .soft: +matlab-r2012b
  
  resoft

- **Interactive section:**

  qsub -I -X -A hpc_train_2014 -l nodes=1:ppn=16,walltime=02:00:00
  
  matlab -nodesktop

- **Submit a batch job**

  See details at HPC@LSU website:
  
  http://www.hpc.lsu.edu/docs/guides/software.php?software=matlab
Desktop window

- matlab &
- Some Linux commands are available.
- cd
- ls
- !cp
- !rm
- !mv
- !vim
(pre exclamation)
Matrix and arrays

- Matlab = lab of matrix
- Most objects (e.g. data, text, color) in Matlab can be represented by a vector or a matrix.
- Convenient for linear algebra operations.

```plaintext
% use percent sign for comments,
% auto output without a semicolon

- **Scalar:** a = 5
- **Array (vector):** b = [1 2 3];    % row
  c = [4; 5; 6];    % column

- **Matrix:** A = [1 2 3; 4 5 6; 7 8 9]    % row-first convention

- **Vector operations:**
  - dot(b,c)    % dot product
  - cross(b,c)  % cross product
  - norm(b)     % norm
```
Matrix operations:

- \( A + a \)  \% a matrix plus a scalar
- \( A \times a \)  \% a matrix multiplies a scalar
- \( \sin(A) \)  \% sine of each element of a matrix
- \( \exp(A) \)  \% exponential of each element of a matrix
- \( A' \)  \% transpose of a matrix
- \( A + A \)  \% matrix addition
- \( A \times A \)  \% matrix multiplication
- \( A \times A \)  \% element-by-element multiplication.
- \( A / A \)  \% element-by-element division.
- \( A ^ 3 \)  \% element-by-element power
- \( A' \)  \% complex conjugate transpose of a matrix
- \( \text{inv}(A) \)  \% inverse of a matrix
- \( \text{det}(A) \)  \% determination of a matrix
- **Matrices indexing**
  - A(3,2) % the element of 3\textsuperscript{rd} line and 2\textsuperscript{nd} column
  - A(:,1) % the 1\textsuperscript{st} column
  - A(2,2:3) % through 2\textsuperscript{nd} to 3\textsuperscript{rd} elements of 2\textsuperscript{nd} line
  - sum(A(2,:)) % sum all elements of the 2\textsuperscript{nd} line
  - max(A(3,:)) % maximum element of the 3\textsuperscript{rd} line
  - find(isprime(A)) % find prime numbers among all elements

- **Basic matrices**
  - zeros(3,3) % All zeros
  - ones(3,3) % All ones
  - rand(3,3) % Uniformly distributed random elements, between 0 and 1.
  - randn(3,3) % Normally distributed random elements
  - I=eye(3) % Unit matrix
Language fundamentals

- **Variables**
  - No declaration required (more convenient than C or Fortran)
  - \( n = 25 \) % Integer
  - \( a = 6.2 \) % Real number

- **Data type**
  - \( \text{double}(n) \) % Convert to double precision
  - \( \text{single}(n) \) % Convert to single precision
  - \( \text{int8}(a), \text{int16}(a), \text{int32}(a), \text{int64}(a) \) % Convert to 8-bit, 16-bit, 32-bit, 64-bit signed integer
  - \( \text{eps} \) % Floating-point relative accuracy: 2.2204e-16 for double
Complex number

- $i$ \hspace{1cm} \% imaginary unit
- $j$ \hspace{1cm} \% imaginary unit
- $\sqrt{-1}$ \hspace{1cm} \% imaginary unit
- $x=3+4i$ \hspace{1cm} \% complex number
- $x=\text{complex}(a,b)$ \hspace{1cm} \% real part is $a$, imaginary part is $b$
- $\text{complex}(x)$ \hspace{1cm} \% convert to complex number
- $\text{real}(x)$ \hspace{1cm} \% real part of $x$
- $\text{imag}(x)$ \hspace{1cm} \% imaginary part of $x$
- $\text{angle}(x)$ \hspace{1cm} \% argument of $x$
- $\text{abs}(x)$ \hspace{1cm} \% amplitude of $x$
- $\text{conj}(x)$ \hspace{1cm} \% conjugate of $x$
- $\text{isreal}(x)$ \hspace{1cm} \% $x$ is real or not
Math expressions

- $a^3 + b^2 - 3c + d/6 + 9$
- `abs(x)` % absolute value
- `sin(x); cos(x); tan(x);` % triangle functions
- `asin(x); acos(x); atan(x); atan2(y,x);` % inverse triangle functions
- `sqrt(x)` % square root
- `exp(x)` % exponential
- `log(x)` % natural logarithm
- `log10(x)` % base-10 logarithm

Long statement

- $s = 1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + 1/7 \ldots$  // combine two lines
  - $- 1/8 + 1/9 - 1/10 + 1/11 - 1/12;$
Conditional Control — if, else

```matlab
a = randi(100, 1);
if a < 30
    fprintf('%i is smaller than 30. \n', a);
elseif a > 80
    fprintf('%d is larger than 80. \n', a);
else
    X=[num2str(a), ' is between 30 and 80.'];
    disp(X)
end
```
Conditional Control — switch

[dayNum, dayString] = weekday(date, 'long', 'en_US');
switch dayString
    case 'Monday'
        disp('Start of the work week')
    case 'Tuesday'
        disp('Day 2')
    case 'Wednesday'
        disp('Day 3')
    case 'Thursday'
        disp('Day 4')
    case 'Friday'
        disp('Last day of the work week')
    otherwise
        disp('Weekend!')
end
Loop Control — for

% sum of an array
s=0;
b=rand(100,1)
for i = 1:1:100
  s=s+b(i);  % not allow +=
end
s

% nested loop
m=50;
n=100;
for i = 1:1:m  % Stripe 1
  for j = 1:2:n  % Stripe 2
    H(i,j) = 1/(i+j);
    end
  end
end
Loop Control — while, break

% find a root of the cubic polynomial \( x^3 - 2x - 5 \) using Newton's method

\[
a = 0; \ fa = -\text{Inf}; \\
b = 3; \ fb = \text{Inf}; \\
\text{while} \ b-a > \text{eps*b} \\\n\quad \text{x} = (a+b)/2; \\
\quad \text{fx} = x^3-2*x-5; \\
\quad \text{if} \ \text{fx} == 0 \\
\quad \quad \text{break} \quad % \text{Already found the root, exit the loop}
\text{elseif} \ \text{sign(fx)} == \text{sign(fa)} \\
\quad \quad a = x; \ fa = fx;
\text{else} \\
\quad b = x; \ fb = fx;
\text{end} \\
\text{end}
\]

x
M-file and Script

- An m-file, or script file, is a simple text file where you can place MATLAB commands.
- Save your works
- Convenient for debugging
- Run directly. No explicit compilation and link.

- modify and run M-files:
  - !vim file_name.m
  - edit file_name.m
  - open file_name.m
  - run file_name.m
Matrix multiplication in Matlab

\[ n = 1000; \quad \text{% set matrix dimension} \]

\[ \text{for } i=1:1:n \quad \text{% initialize data} \]
\[ \text{for } j=1:1:n \]
\[ A(i,j)=i+j; \]
\[ B(i,j)=2*i-j; \]
\[ \text{end} \]
\[ \text{end} \]

\[ C=A*B \quad \text{% matrix multiplication & output} \]
Matrix multiplication in C

/* include head files if necessary */

int main()
{

int n = 1000; // set matrix dimension

/* declaration and allocation */
double *a = (double *) malloc ( n*n*sizeof(double) );
double *b = (double *) malloc ( n*n*sizeof(double) );
double *c = (double *) malloc ( n*n*sizeof(double) );

/* initialize data */
for (int i=0; i<n; i++)
for (int j=0; j<n; j++) {
    a[i][j]=i+j;
    b[i][j]=2.*i-j;
}

/* matrix multiplication */
for (int i=0; i<n; i++) // loop over output rows, i
    for (int j=0; j<n; j++) { // loop over output columns, j
        c[i][j] = 0; // initialize output result to zero
        for (k=0; k<n; k++) // loop over inner dimension, k
            c[i][j] += a[i][k] * b[k][j]; // perform sum
    }

/* output results */

} // end program

◊ use MKL or LAPACK/BLAS to simplify the C codes
Programming II : Functions

- Anonymous Functions
  
  \( f = @(\text{arglist}) \text{ expression} \)

  - One argument
    
    \[
    \text{my\_fun} = @(x) \ x.^2+\exp(x)+5; \\
    \text{my\_fun}(5)
    \]

  - Two arguments
    
    \[
    \text{my\_fun} = @(x,y) \ x.^3+6*\sqrt{y}; \\
    \text{my\_fun}(3,4)
    \]
Local Functions

function y = my_fun(x)

% In falling.m. File name should be the same as function name.
function height = falling(t)
    global GRAVITY
    height = 1/2*GRAVITY*t.^2;  % The height of a freely falling object
end

% In main.m. Should be in the same path of falling.m
global GRAVITY
GRAVITY = 32;
y = falling((0:.2:5))
Programming III : I/O with external files

- Input from a file
  - % text format
  - load(filename,'-ascii')
  - % Matlab format
  - load(filename,'-mat')

- Output to a file
  - % open a file
    - fileID = fopen(filename,permission)
  - % print data
    - fprintf(fileID,formatSpec,A1,...,An)

- An example
  - x = 0:.1:1;
  - A = [x; exp(x)];
  - fileID = fopen('exp.txt','w'); % open a writable file
  - fprintf(fileID,'%6.2f %12.8f\n',A); % print real data
  - fclose(fileID); % close the file
  - B = load('exp.txt','-ascii'); % load data from the file
Plot figures with command lines instead of mouse! Easy for repeating.

```matlab
plot
x = 0:pi/100:4*pi;
y = sin(x);
y2 = cos(x);
plot(x,y,'black',x,y2,'red--','linewidth',2)
xlabel('x')
ylabel('y')
axis([0 4*pi -1 1])
title('Plot of Sine and Cosine Functions','FontSize',12)
legend('sin(x)','cos(x)')
```
Plotting three-dimensional curves

- plot3

\[ t = 0:\pi/50:10*\pi; \quad \% \quad z \]
\[ st = \sin(t); \quad \% \quad x \]
\[ ct = \cos(t); \quad \% \quad y \]

figure
plot3(st,ct,t)
Contour Plot

- **Contour, pcolor**

  % Obtain data from evaluating peaks function
  [x,y,z] = peaks;

  % Create pseudocolor plot
  pcolor(x,y,z)

  % Smooth the colors
  shading interp

  % Hold the current graph
  hold on

  % Add the contour graph to the pcolor graph
  contour(x,y,z,15,'k')

  % Return to default
  hold off
Subfigures and layout

- **Subplot, mesh**

  ```matlab
  t = 0:pi/10:2*pi;
  % cylinder with a self-defined profile
  [X,Y,Z] = cylinder(4*cos(t));
  subplot(2,2,1);  % left-up
  mesh(X)
  subplot(2,2,2);  % right-up
  mesh(Y)
  subplot(2,2,3);  % left-down
  mesh(Z)
  subplot(2,2,4);  % right-down
  mesh(X,Y,Z)
  ```
Color Surface Plot

- surfc

[X,Y] = meshgrid(-8:.5:8);
R = sqrt(X.^2 + Y.^2) + eps;
Z = sin(R)./R;  \% sinc function
surf(X,Y,Z)
shading interp
colormap hsv  \% color map
colorbar  \% show color scaling
view([1 1 1])  \% view angle
diamond colormap

diamond color palette
Red [1 0 0]
Green [0 1 0]
Blue [0 0 1]
Black [0 0 0]
White [1 1 1]

% design your own color map

diamond view angles
From x axis: [1 0 0]
From y axis: [0 1 0]
From z axis: [0 0 1]
From diagonal line: [1 1 1]
Animation

◊ **gca, getframe, movie**

\[ Z = \text{peaks}; \quad \% \text{create random Gaussian peaks} \]

\[ \text{figure('Renderer','zbuffer');} \quad \% \text{rendering method used for screen and printing} \]

\[ \text{surf}(Z); \quad \% \text{plot the peaks} \]

\[ \text{axis tight manual;} \quad \% \text{set axis tight and freeze the scaling} \]

\[ \text{set(gca,'NextPlot','replaceChildren');} \quad \% \text{remove previous figures but keep the figure properties} \]

\[ \text{for } j = 1:20 \]

\[ \quad \text{surf}(\sin(2\pi j/20)*Z,Z) \quad \% \text{modify the shapes of the peaks at different time instants} \]

\[ \quad \text{F(j) = getframe;} \quad \% \text{capture the current figure as a movie frame} \]

\[ \text{end} \]

\[ \text{movie(F,10)} \quad \% \text{play the movie ten times} \]
Mathematics

- Numerical integration and differentiation
- Linear algebra equations
- Eigen problem
- Fast Fourier Transform (FFT)
- Mathematical special functions
Numerical integration

◇ One-dimensional integration

\[ q = \text{integral}(\text{fun}, \text{xmin}, \text{xmax}) \]

% approximates the integral of function fun from xmin to xmax using global adaptive quadrature and default error tolerances.

\[ \text{fun} = @(x) \exp(-x.^2).*\log(x).^2; \quad \% \text{define a function } f(x) = e^{-x^2} [\ln(x)]^2 \]

\[ p = \text{integral}(\text{fun}, 0, 0.5) \quad \% \text{proper integral} \]

\[ q = \text{integral}(\text{fun}, 0, \text{Inf}) \quad \% \text{improper integral} \]

\[ \text{fun} = @(x)\log(x); \quad \% \text{logarithm function} \]

\[ \text{format long} \quad \% \text{output long digits} \]

\[ q = \text{integral}(\text{fun}, 0, 1) \quad \% \text{integral with singularity at the lower limit} \]
Two- and three-dimensional integrations

\[ q = \text{integral2}(\text{fun}, \text{xmin}, \text{xmax}, \text{ymin}, \text{ymax}) \]

\[ q = \text{integral3}(\text{fun}, \text{xmin}, \text{xmax}, \text{ymin}, \text{ymax}, \text{zmin}, \text{zmax}) \]

\[
f(\theta, r) = \frac{r}{\sqrt{r \cos \theta + r \sin \theta} (1 + r \cos \theta + r \sin \theta)^2}
\]

\[
polarfun = @(\text{theta}, \text{r}) \text{fun}(\text{r}.*\cos(\text{theta}), \text{r}.*\sin(\text{theta})).*\text{r}; \quad \% \text{define a 2D function in polar coordinates}
\]

\[
rmax = @(\text{theta}) 1./\left(\sin(\text{theta}) + \cos(\text{theta})\right); \quad \% \text{set upper limit as a function of theta}
\]

\[
p = \text{integral2}(\text{polarfun}, 0, \text{pi}/2, 0, \text{rmax}) \quad \% \text{2D integral}
\]

\[
\text{fun} = @(\text{x}, \text{y}, \text{z}) \text{y}.*\sin(\text{x}) + \text{z}.*\cos(\text{x}); \quad \% \text{define a 3D function}
\]

\[
q = \text{integral3}(\text{fun}, 0, \text{pi}, 0, 1, -1, 1) \quad \% \text{3D integral with finite limits}
\]
Numerical differentiation

\[ Y = \text{diff}(X) \quad \% \text{ first derivative} \]
\[ Y = \text{diff}(X,n) \quad \% \text{ the } n^{\text{th}} \text{ derivative} \]

\[ h = 0.001; \quad \% \text{ step size} \]
\[ X = -\pi:h:pi; \quad \% \text{ domain} \]
\[ f = \sin(X); \quad \% \text{ range} \]
\[ Y = \text{diff}(f)/h; \quad \% \text{ first derivative} \]
\[ Z = \text{diff}(f,2)/h^2; \quad \% \text{ second derivative} \]

plot(X(:,1:length(Y)),Y,'r',X,f,'b', X(:,1:length(Z)),Z,'k')
Solve linear algebra equations

- Solves the linear system $A^*x = b$
  
  $x = A \backslash b$

  $x = \text{mldivide}(A,b)$

  $x = \text{linsolve}(A,b)$

  $x = \text{linsolve}(A,b,\text{opts})$

  % Using LU factorization when $A$ is square and QR factorization otherwise.

  % The number of rows of $A$ must equal the number of columns of $B$.  

  

  $A = \text{triu}(\text{rand}(5,5));$ % random 5*5 up-triangle matrix

  $b = \text{rand}(5,1);$ % random column array

  $\text{opts.UT} = \text{true};$ % up-triangle is ture

  $x = \text{linsolve}(A,b,\text{opts})$

  $x = A \backslash b$

  % The solution is straight forward, if $A$ is triangular.
Factorization

♦ LU factorization

\[ [L, U] = lu(A) \] expresses a matrix A as the product of two essentially triangular matrices, one of them a permutation of a lower triangular matrix and the other an upper triangular matrix. The factorization is often called the LU, or sometimes the LR, factorization.

\[
[L, U] = lu(A); \quad \% \text{obtain } L \text{ and } U \text{ matrices}
\]
\[
y = L \backslash b;
\]
\[
x = U \backslash y \quad \% \text{equivalent to } x = A \backslash b
\]

♦ QR factorization

\[ [Q, R] = qr(A) \] expresses a matrix A into a product \( A = QR \) of an orthogonal matrix \( Q \) and an upper triangular matrix \( R \).

\[
[Q, R] = qr(A); \quad \% \text{obtain } Q \text{ and } R \text{ matrices}
\]
\[
y = Q^\ast \times b;
\]
\[
x = R \backslash y \quad \% \text{equivalent to } x = A \backslash b
\]
Eigen problem

◊ Find eigen values and eigen vectors for a matrix ($A^\times x = \lambda^\times x$)

$[V,D] = \text{eig}(A)$  % returns a diagonal matrix containing the eigenvalues ($D$) and a matrix whose columns are the corresponding right eigenvectors ($V$).

◊ Solve eigen problem for symmetric and asymmetric matrix

$u = \text{triu}(\text{rand}(5,5))$;  % create a random 5*5 up-triangle matrix
$A = u+u^\prime$;  % create a symmetric matrix

$[P,L] = \text{eig}(A)$  % The eigen values and vectors of a symmetric matrix are real.
$P^\prime A P$  % check: should equal to $L$

$B = \text{rand}(5,5)$;  % create a random 5*5 matrix

$[V,D] = \text{eig}(B)$  % The eigen values and vectors could be complex.

$V^\prime B V$  % check: should equal to $D$
Fast Fourier Transform (FFT)

- FFT of an array
  
y = fft(x,n)
  
% FFT is an algorithm to compute the discrete Fourier transform (DFT) and its inverse.
% Fourier transform converts time (or space) to frequency (or momentum).

- Create a time series
  
Fs = 1000;       % Sampling frequency
T = 1/Fs;        % Sample time
L = 1000;        % Length of signal
t = (0:L-1)*T;   % Time vector

% Sum of a 50 Hz sinusoid and a 120 Hz sinusoid
x = 0.7*sin(2*pi*50*t) + sin(2*pi*120*t);
y = x + 2*randn(size(t));   % Sinusoids plus noise

plot(Fs*t(1:50),y(1:50))
title('Signal Corrupted with Zero-Mean Random Noise')
xlabel('time (milliseconds)')
Fourier analysis

% Next power of 2 from length of y
NFFT = 2^nextpow2(L);
Y = fft(y,NFFT)/L;
f = Fs/2*linspace(0,1,NFFT/2+1);

% Plot single-sided amplitude spectrum.
plot(f,2*abs(Y(1:NFFT/2+1)))
title('Single-Sided Amplitude Spectrum of y(t)')
xlabel('Frequency (Hz)')
ylabel('|Y(f)|')
Mathematical special functions

- Many special functions appear as solutions of differential equations or integrals of elementary functions.

- airy % Airy Functions
- besselh % Bessel function of third kind (Hankel function)
- besseli % Modified Bessel function of first kind
- besselj % Bessel function of first kind
- besselk % Modified Bessel function of second kind
- bessely % Bessel function of second kind
- beta % Beta function
- betainc % Incomplete beta function
- Betaincinv % Beta inverse cumulative distribution function
- betaln % Logarithm of beta function
- ellipj % Jacobi elliptic functions
- ellipke % Complete elliptic integrals of first and second kind
Mathematical special functions

- erf % Error function
- erfc % Complementary error function
- erfcinv % Inverse complementary error function
- erfcx % Scaled complementary error function
- erfinv % Inverse error function
- expint % Exponential integral
- gamma % Gamma function
- gammainc % Incomplete gamma function
- gammaincinv % Inverse incomplete gamma function
- gammaln % Logarithm of gamma function
- legendre % Associated Legendre functions
- psi % Psi (polygamma) function
Bessel Function

Solution of the differential equation

\[
\frac{d^2y}{dz^2} + \frac{dz}{d\zeta} + (\zeta^2 - \nu^2)y = 0,
\]

\[
J_\nu(z) = \left(\frac{z}{2}\right)^\nu \sum_{k=0}^{\infty} \frac{(-\frac{z^2}{4})^k}{k!\Gamma(\nu + k + 1)}
\]

X = 0:0.1:20;
J = zeros(5,201);
for i=0:4
    J(i+1,:) = besselj(i,X);
end
plot(X,J,'LineWidth',1.5)
axis([0 20 -.5 1])
grid on;
legend('J_0','J_1','J_2','J_3','J_4','Location','Best')
title('Bessel Functions of the First Kind for \nu = 0,1,2,3,4')
xlabel('X')
ylabel('J_\nu(X)')
Exercises

🔹 Exercise 1:

Create an n*n symmetric matrix A in two ways:
  i) use built-in functions up-triangle matrix (triu) and the matrix transpose;
  ii) use control flow (for, if, else, …).

🔹 Exercise 2:

Use the matrix A from exercise 2 and create a size-n vector b, then solve the linear algebra equation A*x=b with two methods:
  i) calculate the inverse of A.
  ii) calculate LU decomposition of A.

Compare the computation time and error of the two cases.
Exercise 3:

Consider the curve parameterized by the equations

\[ x(t) = \sin(2t), \quad y(t) = \cos(t), \quad z(t) = t, \]

where \( t \in [0, 3\pi] \).

Create a three-dimensional plot of this curve.

Compute the arc length of this curve. Tips: The arc length formula says the length of the curve is the integral of the norm of the derivatives of the parameterized equations:

\[
\int_{0}^{3\pi} \sqrt{4\cos(2t)^2 + \sin(t)^2 + 1} \, dt.
\]
References