MMAT 5320 Computational Mathematics

Eric Chung

Basic information

- Instructor: Eric Chung (Department of Mathematics)
- Office: LSB 205 (Lady Shaw Building)
- Email: tschung@math.cuhk.edu.hk

- TA: Xingguang Jin and Yucheng Liu (Department of Mathematics)
- Office: LSB 222C and LSB 222C
- Email: xgjin@math.cuhk.edu.hk, ycliu@math.cuhk.edu.hk
- Office hours: Xingguang Jin (TBC), Yucheng Liu (TBC)

- Classroom: YIA LT7
- Time: Every Thursday 6:30pm-9:00pm

Course information

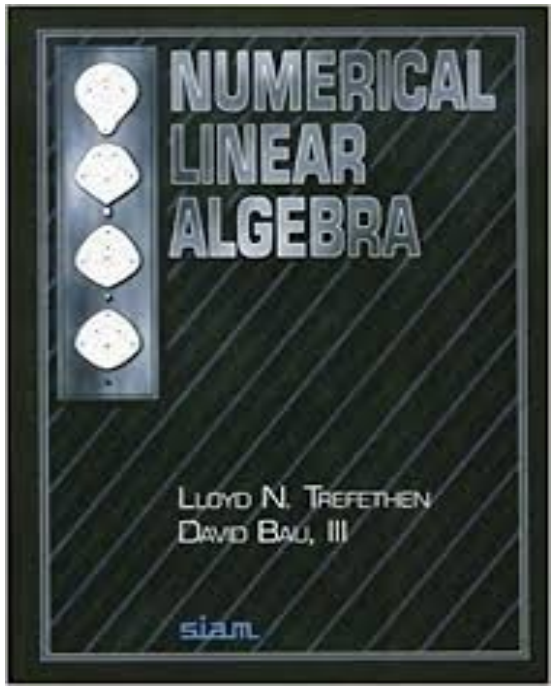
Goal: We cover some advanced mathematical techniques with applications to data-driven computations. We include both mathematical foundations and their applications.

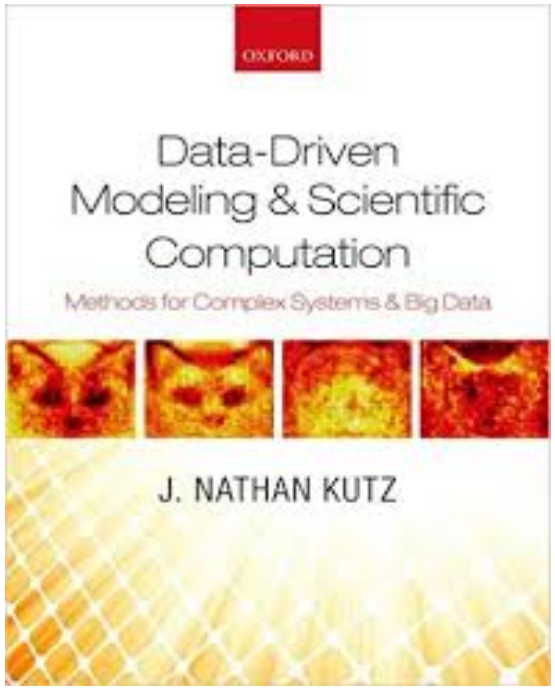
Topics:

- Review of MATLAB
- Review of linear algebra
- Singular value decomposition (SVD)
- Orthogonalization: QR factorization, Householder transform, least-squares problems
- Stability of computations
- Eigenvalue problems
- Iterative methods: power iteration, inverse iteration, QR iteration
- Computing SVD for large sparse matrices
- Principal component analysis (PCA)
- Independent component analysis (ICA)
- Basics of machine learning

Reference books

- Numerical Linear Algebra by Trefethen and Bau
- Data-driven Modeling & Scientific Computation by Kutz





Books:

- Numerical Linear Algebra by Trefethen and Bau
- Data-driven Modeling & Scientific Computation by Kutz

Topics:

- Review of MATLAB (C1-3,6)
- Review of linear algebra (L1-3)
- Singular value decomposition (SVD) (L4-5)
- Orthogonalization: QR factorization, Householder transform, least-squares problems (L6-11)
- Stability of computations (L12-14)
- Eigenvalue problems (L24-25)
- Iterative methods: power iteration, inverse iteration, QR iteration (L26-29)
- Computing SVD for large sparse matrices (L31)
- Principal component analysis (PCA) (C15)
- Independent component analysis (ICA) (C16)
- Basics of machine learning (C17)

Assessment scheme

The final grade depends on

- Homework (both theoretical and computational): 20%
- Mid-term (on Oct 16): 40%, 11:30am - 1:30pm
- Final exam (on Dec 4): 40%, 11:30am - 1:30pm. Topics after midterm.

Course webpage

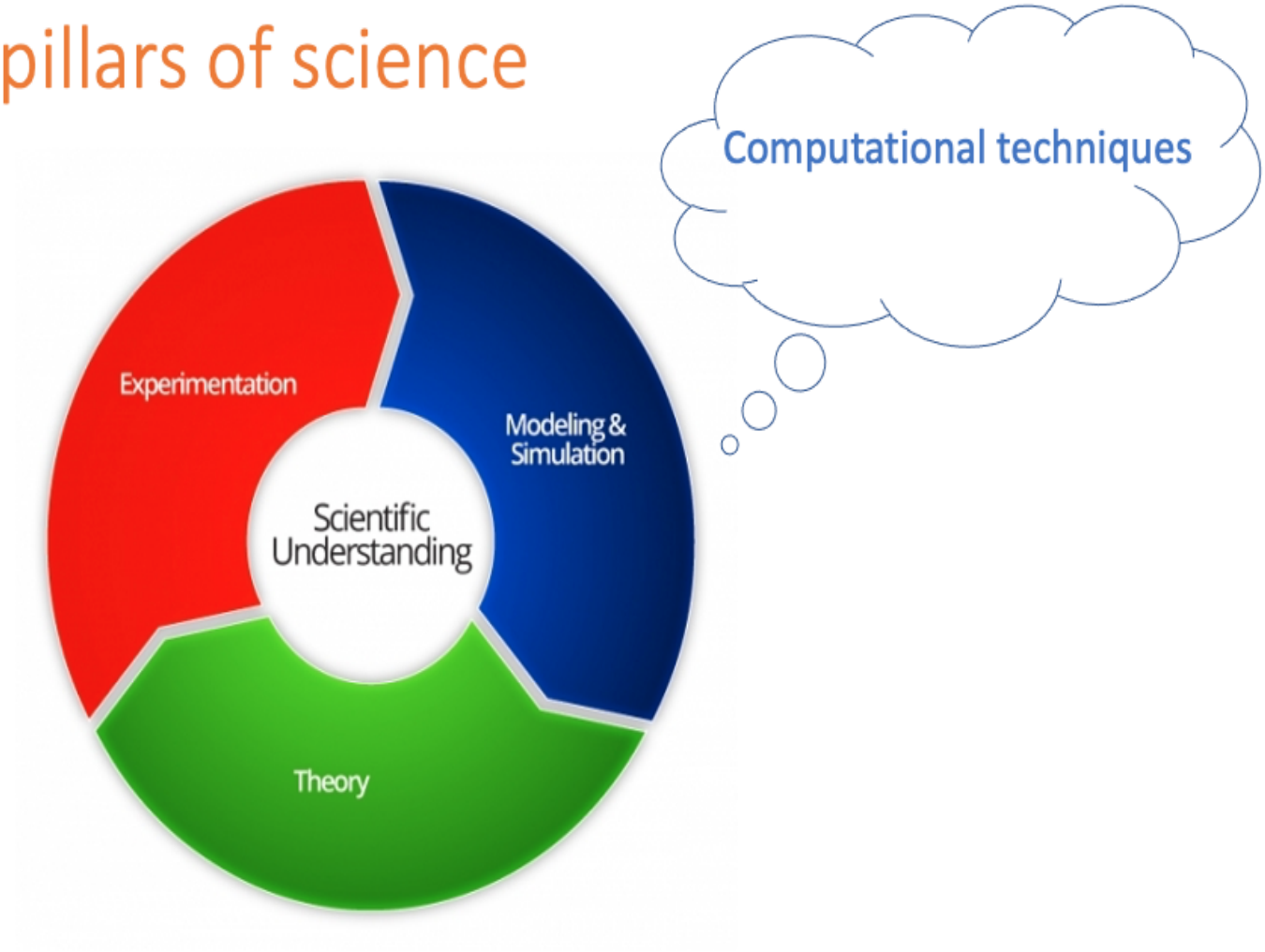
Address:

https://www.math.cuhk.edu.hk/course/2526/mmat5320

To download course material:

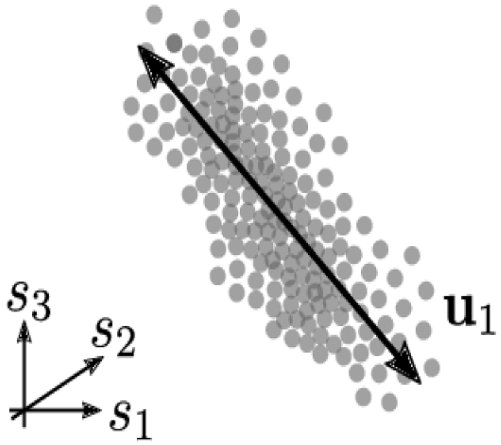
Username: mmat5320

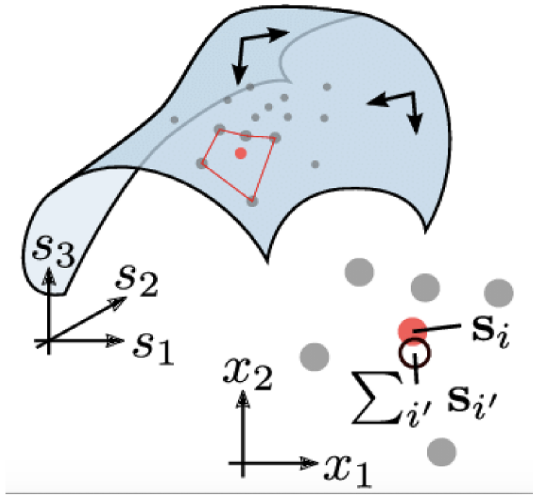
Password: mmat5320=2526


Three pillars of science
Experimentation
Modeling & Simulation
Scientific Understanding
Theory
Computational techniques

Dimensionality reduction

- Many complex systems exhibit dominant low-dimensional pattern
- Gives compact representation for modeling and control

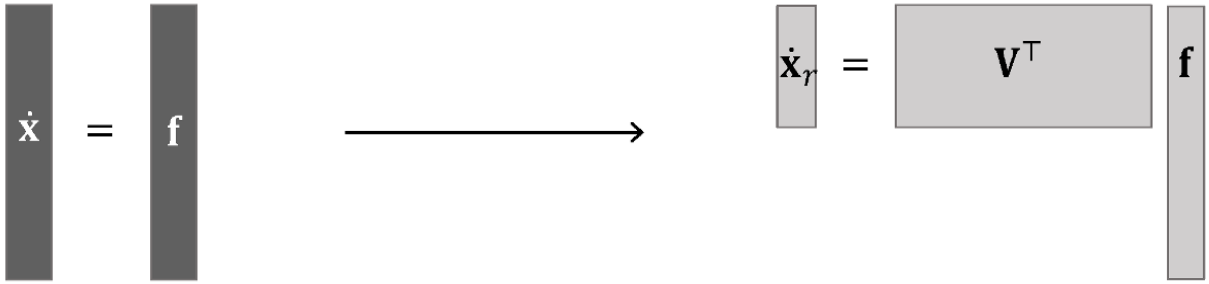




Data-driven coordinate transformation

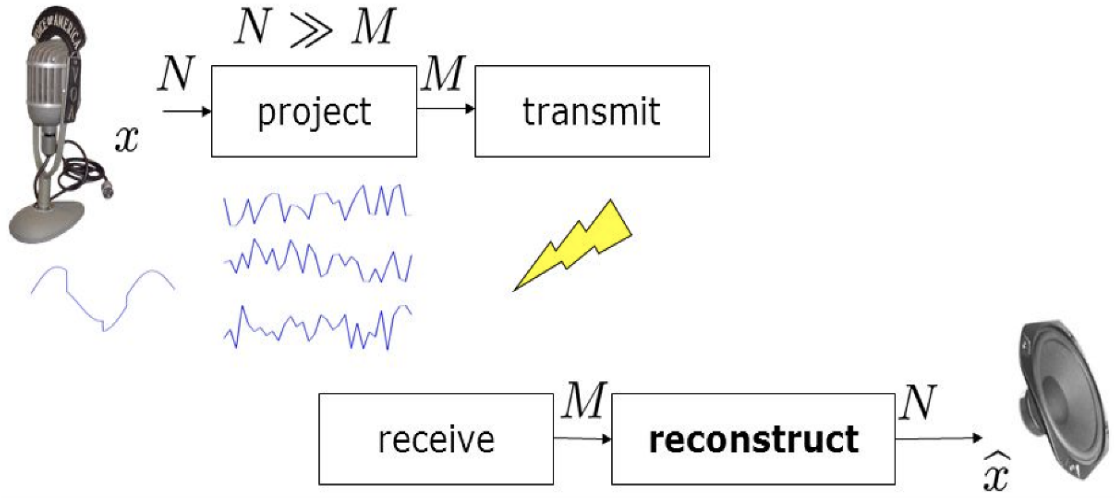
- Finding a coordinate system that can simplify the problem
- The basis is data-driven, and is tailored made
- It is a foundation for reduced order modeling

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) \qquad \dot{\mathbf{x}}_r = \mathbf{V}^\top \mathbf{f}(\mathbf{V}\mathbf{x}_r)$$



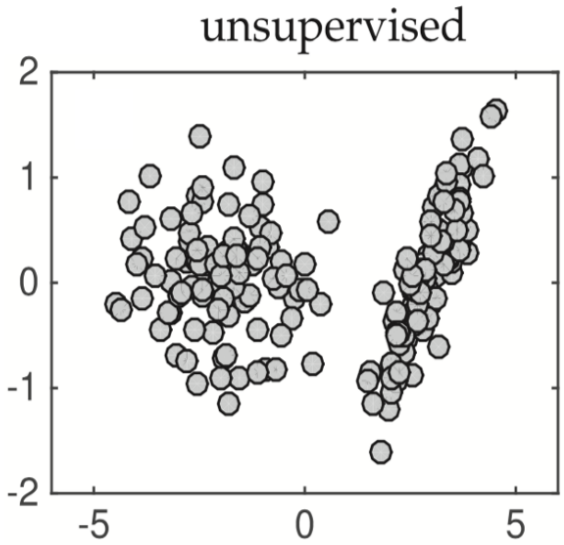
Compressed sensing

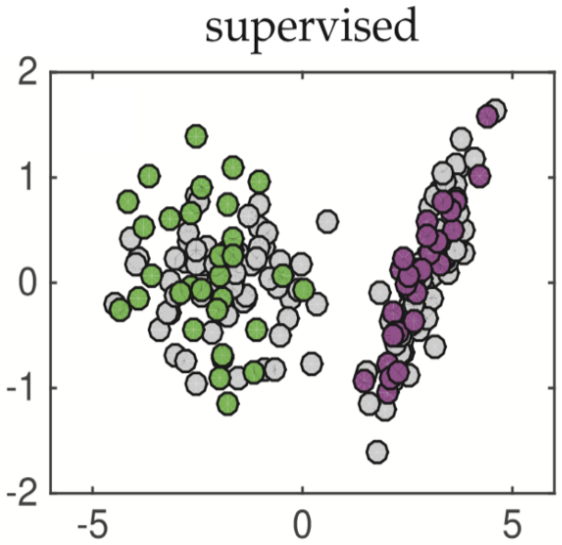
- Structures in data imply that the data admits a sparse representation in a suitable coordinate system
- Instead of collecting high dimensional measurement and then compressing, one can perform compressed measurement



Machine learning

- Supervised vs unsupervised learning
- Supervised learning uses labelled data to extract pattern
- Unsupervised learning finds pattern without using labelled data





Importance of computational math

- The use of computational mathematics is crucial in many data science applications
- Many applications involve the computations of dominant modes (eigenvectors)
- Due to large data size, fast algorithms are necessary
- Goal of this course: present basic computational math and related algorithms, discuss a few applications

MATLAB Introduction

MATLAB is a computing software, with strength in matrix and vector computations.

A row vector $x = (1\ 3\ 2)$ can be created as follows:

```
>>x=[1 3 2]
```

A column vector $\begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix}$ can be created as follows:

```
>>x=[1; 3; 2]
```

or

```
>>x=[1
     3
     2]
```

Creating vectors

- the following command creates the row vector $x = (0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10)$

```
>>x=0:1:10
```

- the following command creates the row vector $x = (0\ 2\ 4\ 6\ 8\ 10)$

```
>>x=0:2:10
```

- the following command creates the row vector $x = (0\ 0.2\ 0.4\ 0.6\ 0.8\ 1)$

```
>>x=0:0.2:1
```

- the following command creates the row vector $x = (0\ 1\ 2\ 3\ 4)$

```
>>x=0:4
```

Creating matrices

The following matrix

$$\begin{pmatrix} 1 & 3 & 2 \\ 5 & 6 & 7 \\ 8 & 3 & 1 \end{pmatrix}$$

can be created by

```
>>A=[1 3 2; 5 6 7; 8 3 1]
```

or

```
>>A=[1 3 2
    5 6 7
    8 3 1]
```

Accessing entries of matrices

Consider, for example, the matrix

$$\begin{pmatrix} 1 & 3 & 2 \\ 5 & 6 & 7 \\ 8 & 3 & 1 \end{pmatrix}$$

- the command gives $x = 7$

```
>>x=A(2,3)
```

- the command gives the second row $x = (5\ 6\ 7)$

```
>>x=A(2,:)
```

- the command gives the third column $x = \begin{pmatrix} 2 \\ 7 \\ 1 \end{pmatrix}$

```
>>x=A(:,3)
```

Consider the matrix

$$B = \begin{pmatrix} 1 & 7 & 9 & 2 \\ 2 & 3 & 3 & 4 \\ 5 & 0 & 2 & 6 \\ 6 & 1 & 5 & 5 \end{pmatrix}$$

- the command gives $x = \begin{pmatrix} 3 \\ 0 \end{pmatrix}$

```
>>x=B(2:3,2)
```

- the command gives $x = (1\ 5\ 5)$

```
>>x=B(4,2:end)
```

- the command gives $C = \begin{pmatrix} 7 & 9 & 2 \\ 3 & 3 & 4 \\ 0 & 2 & 6 \end{pmatrix}$

```
>>C=B(1:end-1,2:4)
```

Transpose and conjugate transpose

Consider the column vector $x = \begin{pmatrix} 3+2i \\ 1 \\ 8 \end{pmatrix}$ where $i = \sqrt{-1}$.

- the command gives $y = (3+2i \quad 1 \quad 8)$

```
>>y=x.'
```

(transpose)

- the command gives $y = (3-2i \quad 1 \quad 8)$

```
>>y=x'
```

(conjugate transpose)

Logical and component-wise operations

We first define a row vector by

```
x=[-1 2 3 5 -2];
```

Then the following commands

```
y1=(x>0);
y2=(x>0).*x;
y3=(x>0)*3;
```

give the following vectors respectively

$$\begin{aligned} y_1 &= (0 \quad 1 \quad 1 \quad 1 \quad 0) \\ y_2 &= (0 \quad 2 \quad 3 \quad 5 \quad 0) \\ y_3 &= (0 \quad 3 \quad 3 \quad 3 \quad 0) \end{aligned}$$

For statements

Some examples:

```
a=0
for j=1:5
    a=a+j
end
```

$a = 15$

```
a=0
for j=1:2:5
    a=a+j
end
```

$a = 9$

```
a=0
for j=[1 5 4]
    a=a+j
end
```

$a = 10$

If statements

General syntax:

```
if (logical statement)
      (expressions to execute)
elseif (logical statement)
      (expressions to execute)
elseif (logical statement)
      (expressions to execute)
else
      (expressions to execute)
end
```

Common logic expressions:

Logic	MATLAB expression
equal to	==
not equal	~=
greater than	>
less than	<
greater than or equal to	>=
less than or equal to	<=
AND	&
OR	|

Example: the Newton's method for solving $f(x) = 0$.
Choose an initial guess x_0, and construct the sequence

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Stop when $f(x_{n+1})$ is small enough.
As an example, take $f(x) = e^x - \tan(x)$. Then $f'(x) = e^x - \sec^2(x)$.

```
clear all  % clear all variables

x(1)=-4;  % initial guess

for j=1:1000  % j is the iteration variable

  x(j+1)=x(j)-(exp(x(j))-tan(x(j)))/(exp(x(j))-sec(x(j))^2);
  fc=exp(x(j+1))-tan(x(j+1));

   if abs(fc)<10^(-5)
       break   % quit the loop
   end

end
x(j+1)   % print value of root
fc   % print value of function
```

Remark: the above is an example MATLAB script, named newton.m

Choose initial guess $x_0 = -4$, and construct the sequence

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Stop when $f(x_{n+1}) < 10^{-5}$.

j	x(j)	fc
1	−4	1.1761
2	−3.4935	0.3976
3	−3.1335	0.0355
4	−3.0964	2.1946×10^{-6}

Functions

For example, the following

```
function y=myfun(x)
y=x.^3+sin(x);
```

defines the function $f(x) = x^3 + \sin(x)$.
To use, save the above as myfun.m, and the following command

```
>> y=myfun(1);
```

The above will give $y = f(1) = 1^3 + \sin(1) = 1.8415$.

Remark: in the above definition, x is the input argument, y is the output argument and myfun is the function name.

Multiple input and output arguments can be defined. For example

```
function [fA B]=funcA(x,A)

B=A^2 + cos(A);
fA=sin(B*x);
```

defines a function called funcA, which is

$$f(x) = \sin(Bx), \quad \text{where} \quad B = A^2 + \cos(A)$$

It has two input arguments and two output arguments.

One can use the function as follows:

```
[fA B]=funcA([3 4 5],2)
```

```
fA =

   -0.9704    0.9804   -0.8018

B =

    3.5839
```

Remark: one can use a vector as an input.

Plotting

To plot the function $y = \sin(x)$ for $-10 \leq x \leq 10$, we first define a set of points on the curve then use the command plot:

```
x1=-10:0.1:10;
y1=sin(x1);
```

```
plot(x1,y1)
```

Remark: MATLAB can generate points in non-uniform ways. For example

```
x2=[-5 -sqrt(3) pi];
y2=sin(x2);
```

One can use the command linspace to generate points. For example

```
x3=linspace(-10,10,64);
y3=x3.*sin(x3);
```

It generates 64 points uniformly from -10 to 10 on the x-axis, and the corresponding values of the function $y = x\sin(x)$.

Remark: We use ".$*$" for component-wise operations.

To plot multiple graphs on the same figure, one can use

```
figure(1)
plot(x1,y1), hold on
plot(x2,y2)
plot(x3,y3)
```

This creates Figure 1. The hold on command is needed to keep all subsequent graphs on the same figure, until hold off command is executed.

Alternatively, one can use

```
figure(2)
plot(x1,y1,x2,y2,x3,y3)
```

One can also give styles to the curves by, for example, using

```
figure(3)
plot(x1,y1,x2,y2,'g*',x3,y3,'mo:')
```

The second function is plotted with Green stars, and the third graph is plotted with Magenta dotted line with the actual data points given by the magenta hollow circles.

More styles:

Line style	Color	Symbol
- = solid	k = black	. = point
: = dotted	b = blue	o = circle
-. = dashed-dot	r = red	x = x-mark
– = dashed	c = cyan	+ = plus
(none) = no line	m = magenta	* = star
	y = yellow	s = square
	g = green	d = diamond
		v = triangle (down)
		∧ = triangle (up)
		< = triangle (left)
		> = triangle (right)
		p = pentagram
		h = hexagram

To give axis labels and figure title, use

```
xlabel('x values')
ylabel('y values')
title('Example Graph')
```

To give labels to each curve, use the legend command.

```
legend('Data set 1','Data set 2','Data set 3','Location','Best')
```

There are other choices of legend positions.

Save and load

To save workspace variables, use

```
save filename
```

To load, use

```
load filename
```

To save a specific variable, use

```
save x1.dat x1 -ASCII
```

To load, use

```
load x1.dat
```

Linear systems

MATLAB can be used to solve linear systems $Ax = b$. The command is

$$x = A\backslash b$$

Remarks:

- if A is nonsingular, then the command gives the unique solution
- if not, then the commands gives least-squares or minimum norm solution
- when use the command, A is not necessarily a square matrix

Eigenvalues and eigenvectors

Eigenvalue λ and the corresponding eigenvector $x \neq 0$ satisfy

$$Ax = \lambda x$$

To find eigenvalues and eigenvectors, use

```
[V,D]=eig(A)
```

The columns of V are eigenvectors. The matrix D is diagonal whose diagonal elements are eigenvalues.

For large matrices, it is slow to compute all eigenvalues and eigenvectors. One can use

```
[V,D]=eigs(A,K,'LM')
```

This command gives the first K largest eigenvalues and their corresponding eigenvectors.

Nonlinear systems

One can use the command fsolve to solve nonlinear systems.

For example, consider the system

$$2x_1 + x_2 + x_1^3 = 0, \qquad x_1 + x_1 x_2 + e^{x_1} = 0$$

We can use the following:

```
x=fsolve('system',[0 0])
```

```
function f=system(x)
f=[2*x(1)+x(2)+x(1)^3; x(1)+x(1)*x(2)+exp(x(1))];
```

Remark: the command fsolve has many options, see help.

Data fitting

Data stored in the file linefit.dat

```
1.7   3.8
2.1   4.3
2.5   4.7
2.9   4.8
3.3   5.5
3.7   6.1
4.2   6.3
4.9   7.1
5.3   7.1
6.0   8.2
6.7   6.9
7.0   5.3
```

To visualize the data, use

```
load linefit.dat
x=linefit(:,1);
y=linefit(:,2);
figure(1), plot(x,y,'o:')
```

Least-squares fitting: given a set of data point

$$(x_1, y_1),\ (x_2, y_2),\ \cdots,\ (x_n, y_n)$$

find a function $y = f(x)$ which minimizes

$$\sum_{k=1}^{n} |f(x_k) - y_k|^2$$

In practice, one needs to specify a function type for $f(x)$. For example, one can assume f is a polynomial given by

$$f(x) = p_n(x) := a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

Then, the problem becomes the determination of the coefficients $a_n, a_{n-1}, \cdots, a_1, a_0$.

Assume we fit the data to the linear function $p_1(x) = a_1 x + a_0$. Use

```
pcoeff=polyfit(x,y,1);
```

The output vector pcoeff gives the coefficients a_1 and a_0.

To evaluate and plot this line, use

```
xp=0:0.1:7;
yp=polyval(pcoeff,xp);
figure(2), plot(x,y,'O',xp,yp,'m')
```

To fit the data to a more general function, need to use fminsearch.

For example, we fit a set of data in the file gaussfit.dat to the function

$$f(x) = Ae^{-Bx^2}$$

use

```
coeff=fminsearch('gauss_fit',[1 1]);
```

```
function E=gauss_fit(x0)

load gaussfit.dat
x=gaussfit(:,1);
y=gaussfit(:,2);
E=sum(   ( x0(1)*exp(-x0(2)*x.^2)-y ).^2   )
```

```
xga=-3:0.1:3;
a=coeff(1); b=coeff(2)
yga=a*exp(-b*xga.^2);
figure(7), plot(x2,y2,'O',xga,yga,'m')
```

More on 2D plots

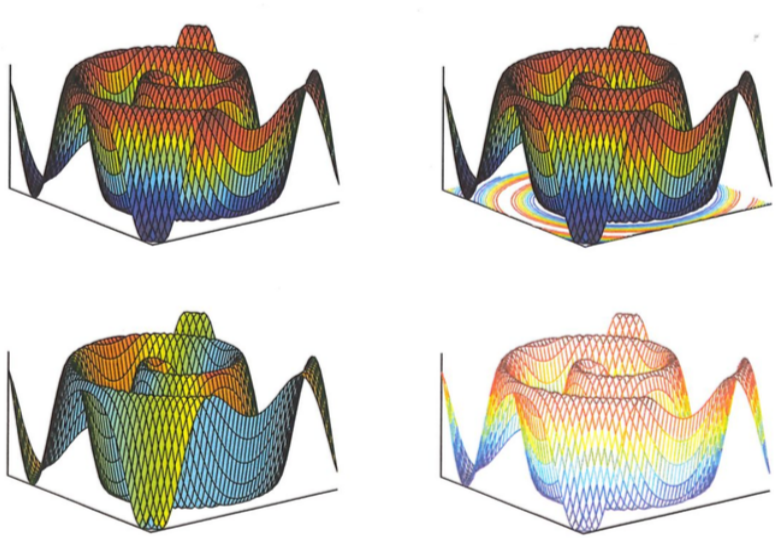
One can use MATLAB to plot the surface defined by a function $u(x, y)$. For example,

$$u(x, y) = \tanh\left(\sqrt{x^2 + y^2}\,\cos(m\angle(x + iy) - \sqrt{x^2 + y^2})\right)$$

```
L=10; x=linspace(-L,L,50); y=x;
[X,Y]=meshgrid(x,y);
m=1; % number of spirals
u=tanh(sqrt(X.^2+Y.^2)).*cos(m*angle(X+i*Y)-(sqrt(X.^2+Y.^2)));
```

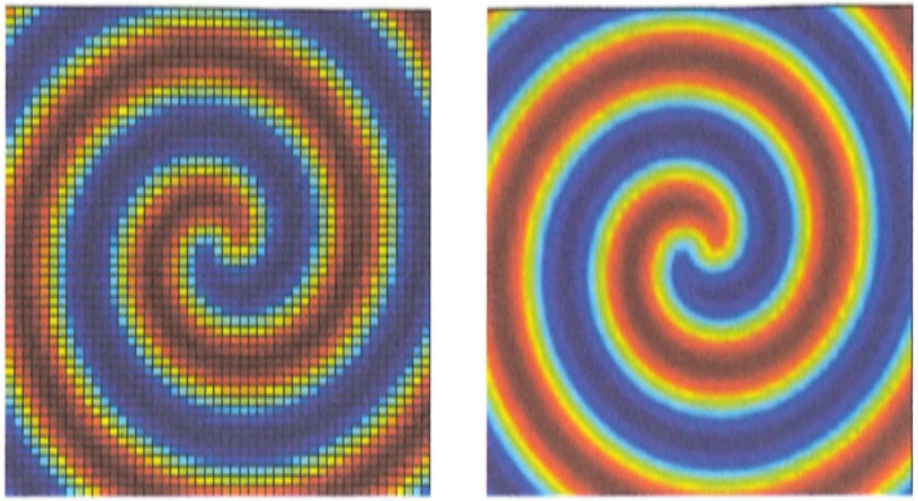
Four ways to plot: surf, surfc, surfl and mesh.

```
subplot(2,2,1), surf(x,y,u)
 set(gca,'Xtick',[],'Ytick',[],'Ztick',[])
subplot(2,2,2), surfc(x,y,u)  % with contour
 set(gca,'Xtick',[],'Ytick',[],'Ztick',[])
subplot(2,2,3), surfl(x,y,u)  % with lighting
 set(gca,'Xtick',[],'Ytick',[],'Ztick',[])
subplot(2,2,4), mesh(x,y,u)
 set(gca,'Xtick',[],'Ytick',[],'Ztick',[])
```



Projection of the surface using pcolor.

```
subplot(2,2,1), pcolor(x,y,u),
 set(gca,'Xtick',[],'Ytick',[],'Ztick',[])
subplot(2,2,2), pcolor(x,y,u), shading interp
 set(gca,'Xtick',[],'Ytick',[],'Ztick',[])
```



Different color schemes.

```
figure(8), surfl(x,y,u), shading interp
figure(9), surfl(x,y,u), shading interp, colormap(hot)
figure(10), surfl(x,y,u), shading interp, colormap(gray)
figure(11), surfl(x,y,u), shading interp, colormap(copper)
```