Mathematics III & Simulation

Prof. Dr. Michael Möckel

WS 2022/23

TH Aschaffenburg
university of applied sciences

Outline

Contents of the Course Maths III

Analytical Mathematics

- Fourier-Series
- Calculation of Fourier coefficients for periodic functions
- Ordinary differential equations (ODEs)
- Classifications of ODEs
- Analytical solutions

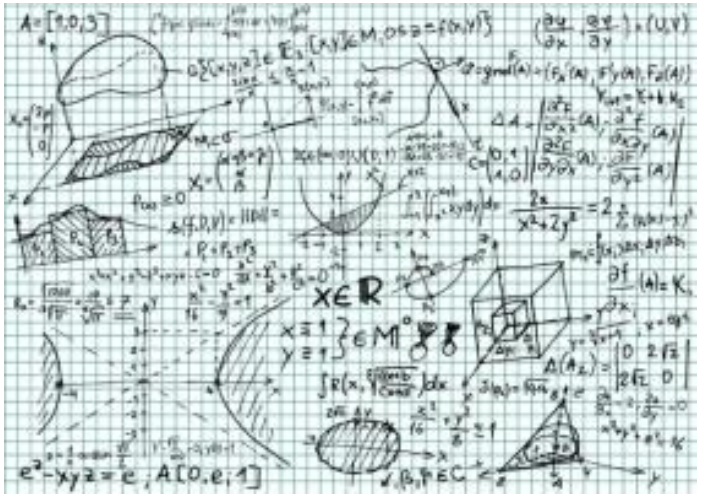
Numerical Maths and Simulation

- Solving nonlinear equations
- Nonlinear mappings (algorithms)
- Solving systems of coupled linear equations (Gauss algorithm)
- Interpolation and regression
- Numerical integration
- Numerical treatment of ODEs
- Simulink

Teaching of the Course Maths III

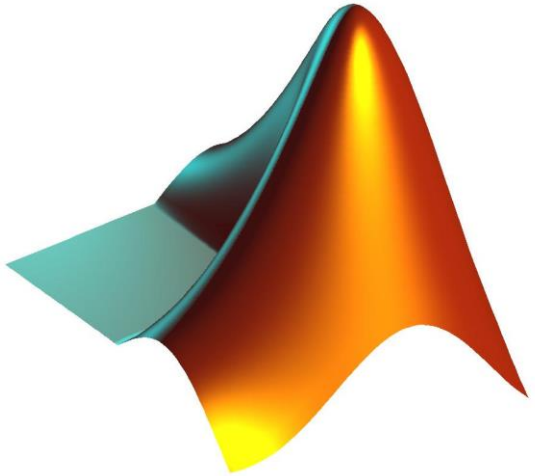
Analytical Mathematics

- Wednesday lunchtime therapy
- 40-230



Numerical Maths and Simulation

- Wednesday morning gymnastics
- Problem sets in MATLAB



Course resources on moodle

- Moodle website: https://moodle.th-ab.de/
- Moodle course: Fakultät IW - MT
- Course name: Mathematik III für MT (id 2907)
- Practical announcements will be published there and emails sent out to registered participants.
- Please register asap (this week!) Otherwise you won‘t be informed well.

Chapter 2

Introduction to MATLAB

Matlab compared with C

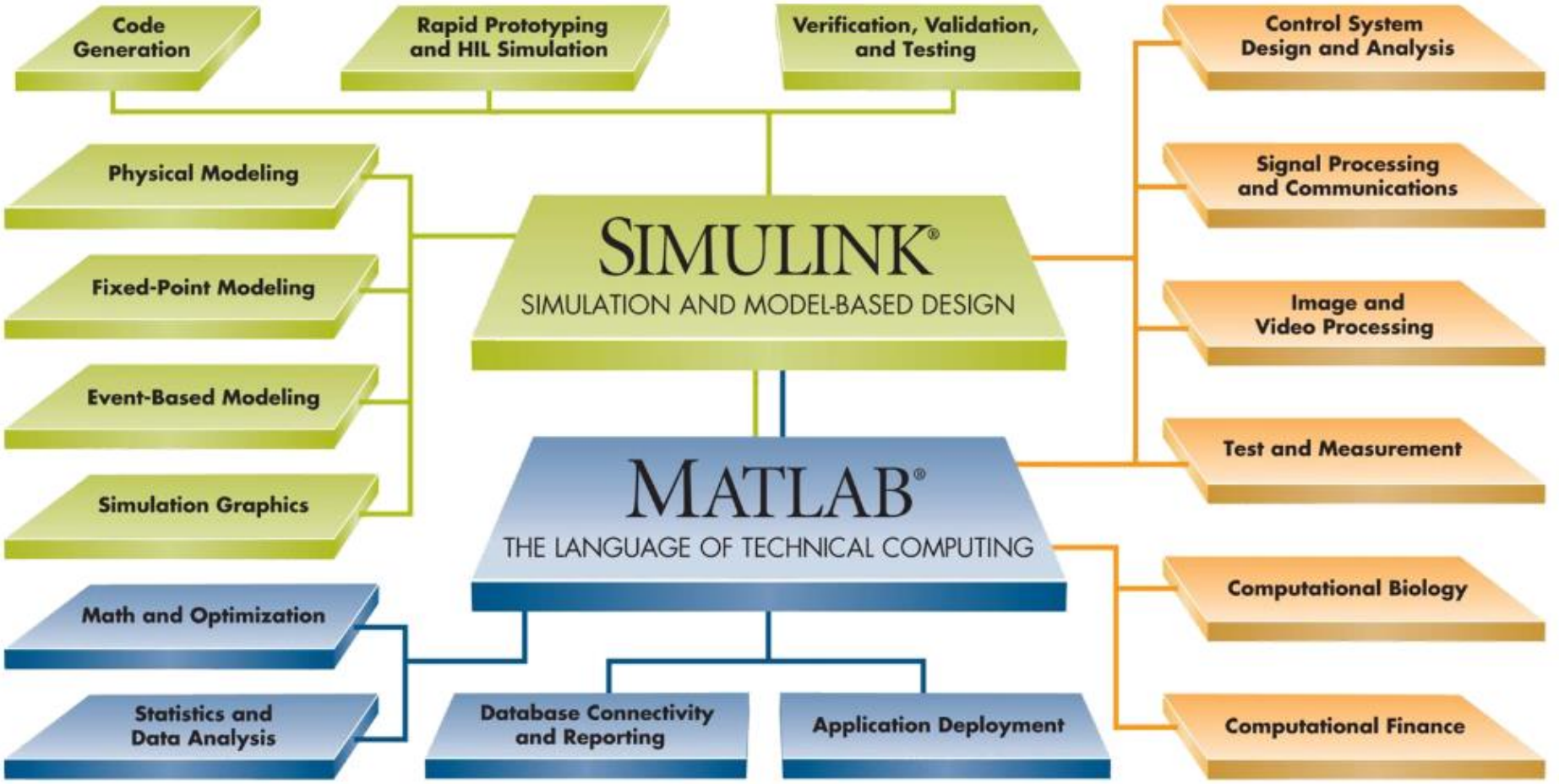
MATLAB

- Interpreted (cf python)
- Focus on simple implementation of linear algebra (easy)
- Clever concepts based on mathematical structures (e.g. *reshape* of „number arrays“)
- Very relaxed on type issues…
- Good for implementation of mathematical algorithms and physical / engineering problems
- Expensive (outside of h-ab)

C

- Compiled
- Efficient code (fast, memory ressources)
- But also more complex, more difficult to learn (pointers)
- Requires clean programming, e.g. strict declarations of types etc.
- Good for low level software development and large collaborative projects

MATLAB Product Family



Simple Commands and Constants

Character	Description
+	Addition
−	Subtraction
*	Multiplication (scalar and array)
/	Division (right)
^	Power or exponentiation
:	Colon; creates vectors with equally spaced elements
;	Semi-colon; suppresses display; ends row in array
,	Comma; separates array subscripts
...	Continuation of lines
%	Percent; denotes a comment; specifies output format
'	Single quote; creates string; specifies matrix transpose
=	Assignment operator
( )	Parentheses; encloses elements of arrays and input arguments
[ ]	Brackets; encloses matrix elements and output arguments

Variable	Description
`ans`	Value of last variable (answer)
`eps`	Floating-point relative accuracy
`i`	Imaginary unit of a complex number
`Inf`	Infinity (∞)
`eps`	Floating-point relative accuracy
`j`	Imaginary unit of a complex number
`NaN`	Not a number
`pi`	The number π (3.14159...)

MATLAB Plattform

Command	Description
`cd`	Change current directory
`clc`	Clear the Command Window
`clear (all)`	Removes all variables from the workspace
`clear x`	Remove x from the workspace
`copyfile`	Copy file or directory
`delete`	Delete files
`dir`	Display directory listing
`exist`	Check if variables or functions are defined
`help`	Display help for MATLAB functions
`lookfor`	Search for specified word in all help entries
`mkdir`	Make new directory
`movefile`	Move file or directory
`pwd`	Identify current directory
`rmdir`	Remove directory
`type`	Display contents of file
`what`	List MATLAB files in current directory
`which`	Locate functions and files
`who`	Display variables currently in the workspace
`whos`	Display information on variables in the workspace

Direct inputs on command line prompt (command window)

Command	Description	Example

`format short`	Fixed-point with 4 decimal digits for: $0.001 \le number \le 1000$ Otherwise display format `short e`.	`>> 290/7`
		`ans =`
		`41.4286`

`format long`	Fixed-point with 15 decimal digits for: $0.001 \le number \le 100$ Otherwise display format `long e`.	`>> 290/7`
		`ans =`
		`41.428571428571431`

`format short e`	Scientific notation with 4 decimal digits.	`>> 290/7`
		`ans =`
		`4.1429e+001`

`format long e`	Scientific notation with 15 decimal digits.	`>> 290/7`
		`ans =`
		`4.142857142857143e+001`

`format short g`	Best of 5-digit fixed or floating point.	`>> 290/7`
		`ans =`
		`41.429`

`format long g`	Best of 15-digit fixed or floating point.	`>> 290/7`
		`ans =`
		`41.4285714285714`

Format befehle

See A. Gilat, MATLAB An Introduction with Applications, WILEY

Rounding commands

Function	Description	Example

`round(x)`	Round to the nearest integer.	`>> round(17/5)`
		`ans =`
		`3`

`fix(x)`	Round toward zero.	`>> fix(13/5)`
		`ans =`
		`2`

`ceil(x)`	Round toward infinity.	`>> ceil(11/5)`
		`ans =`
		`3`

`floor(x)`	Round toward minus infinity.	`>> floor(-9/4)`
		`ans =`
		`-3`

`rem(x,y)`	Returns the remainder after x is divided by y.	`>> rem(13,5)`
		`ans =`
		`3`

Simple maths

Mathematical functions

Function	Description
`sqrt(x)`	Square root.
`nthroot(x,n)`	Real nth root of a real number x. (If x is negative n must be an odd integer.)
`exp(x)`	Exponential (e^x).
`abs(x)`	Absolute value.
`log(x)`	Natural logarithm. Base e logarithm (ln).
`log10(x)`	Base 10 logarithm.

These functions work pointwise!

Similar functions exist for matrix operations, e.g. exp of matrices done by expm, etc

Function	Description
`sign(x)`	Signum function. Returns 1 if $x > 0$, –1 if $x < 0$, and 0 if $x = 0$.

Function	Description
`factorial(x)`	The factorial function $x!$ (x must be a positive integer.)

Function	Description	Example

`mean(A)`	If `A` is a vector, returns the mean value of the elements of the vector.	`>> A=[5 9 2 4];`
		`>> mean(A)`
		`ans =`
		`5`

`C=max(A)`
 	
`[d,n]=max(A)`
	If `A` is a vector, `C` is the largest element in `A`. If `A` is a matrix, `C` is a row vector containing the largest element of each column of `A`.
	
	If `A` is a vector, `d` is the largest element in `A`, and `n` is the position of the element (the first if several have the max value).
		`>> A=[5 9 2 4 11 6 11 1];`
		`>> C=max(A)`
		`C =`
		`11`
		
		`>> [d,n]=max(A)`
		`d =`
		`11`
		`n =`
		`5`

`min(A)`
 	
`[d,n]=min(A)`
	The same as `max(A)`, but for the smallest element.
	
	The same as `[d,n]= max(A)`, but for the smallest element.
		`>> A=[5 9 2 4];`
		`>> min(A)`
		`ans =`
		`2`

`sum(A)`	If `A` is a vector, returns the sum of the elements of the vector.	`>> A=[5 9 2 4];`
		`>> sum(A)`
		`ans =`
		`20`

`sort(A)`	If `A` is a vector, arranges the elements of the vector in ascending order.	`>> A=[5 9 2 4];`
		`>> sort(A)`
		`ans =`
		`2 4 5 9`

Simple statistics with std(v)

These commands are defined as VECTOR functions!!!

If applied to a matrix it operates on all COLUMNS seperately

Similar prod(A) product of all elements of a vector!

Matrix routines

Function	Description	Example

`length(A)`	Returns the number of elements in the vector `A`.	`>> A=[5 9 2 4];`
		`>> length(A)`
		`ans =`
		`4`

`size(A)`	Returns a row vector `[m,n]`, where `m` and `n` are the size $m \times n$ of the array `A`.	`>> A=[6 1 4 0 12; 5 19 6 8 2]`
		`A =`
		`6 1 4 0 12`
		`5 19 6 8 2`
		`>> size(A)`
		`ans =`
		`2 5`

`reshape(A, m,n)`	Creates a `m` by `n` matrix from the elements of matrix `A`. The elements are taken column after column. Matrix `A` must have `m` times `n` elements.	`>> A=[5 1 6; 8 0 2]`
		`A =`
		`5 1 6`
		`8 0 2`
		`>> B = reshape(A,3,2)`
		`B =`
		`5 0`
		`8 6`
		`1 2`

Data handling

Save data to disk

- The save command: `save filename vars`
- Save variables in binary: assumes `.mat`.
  `save filename var1 var2 var3 ...`
  `save filename` will save all variables.
- Save numerical matrices in readable format:
  `save name.ext var1 var2 ... -ascii`
  will print var1, var2 etc. into a file, each row in a new line, in the format currently in effect.

Loading external data

- The load command: `load filename`
- Load binary: filename without extension, assumes `.mat`
  Example: `load myvars` will look for a file myvars.mat and will load all variables in it.
- Load ascii: filename with any extension ***except*** `.mat` will load ***one*** matrix.
  Example: `load nonmono.dat` will create a variable named `nonmono`.
- A = `xlsread('file-name')` will read the excel file file-name.

Problematic numbers

- Results of calculations: Inf, NaN
- Missing values: NaN
- `finite(A)` will return **true** for all elements which are not NaN or Inf.
- `isnan(A)` will return **true** for all elements which are NaN
- Outliers: exceptional values

Handling data with NaN

- Ignoring Inf and NaN values:
  `A(finite(A))`
- Ignoring NaN values:
  `A(~isnan(A))`
- Replacing NaN with another value:
  `A(isnan(A))=val`
- Removing lines having at least one NaN:
  `A(any(isnan(A),2),:)=[]`

Removing outliers

- Must have some criteria, e.g. known impossible values, standard deviation etc.
- Standard deviation: spread relative to the mean or to some fit.
- Relative to the mean of a 1D array V:

```
s = std(V);
V((V-mean(V)).^2 > s^2)=[];
```

- Removing lines from a matrix M:

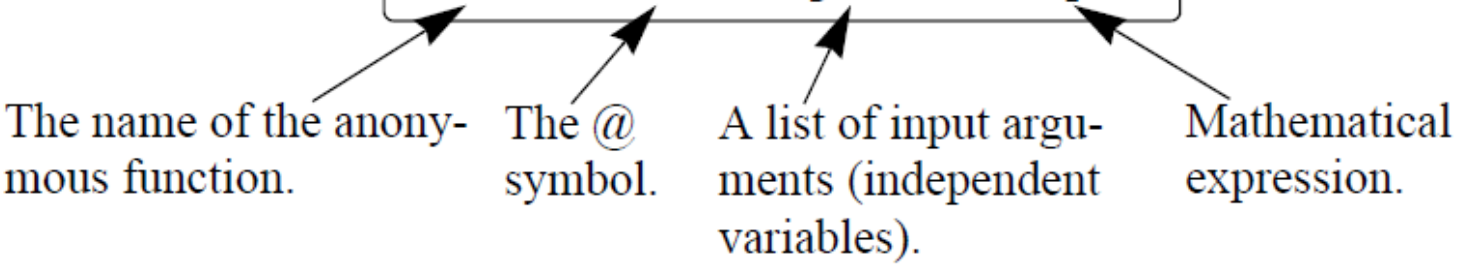
```
s = std(M(:));
M(std(M,2)>s,:)=[];
```

Function definitions in Matlab

Defining simple functions in Matlab

- **Anonymous function**

```
name = @ (arglist) expr
```



- **Inline function**

```
name = inline('math expression typed as a string')
```

```
name = inline('mathematical expression','arg1',
                                    'arg2','arg3')
```

Function definition in m-files

- Every function has **its own m-file**
- Can be created by clicking
- Defining first line of this file:

```
function [output arguments] = function_name(input arguments)
```

The word "function" must be the first word, and must be typed in lowercase letters.

A list of output arguments typed inside brackets.

The name of the function.

A list of input arguments typed inside parentheses.

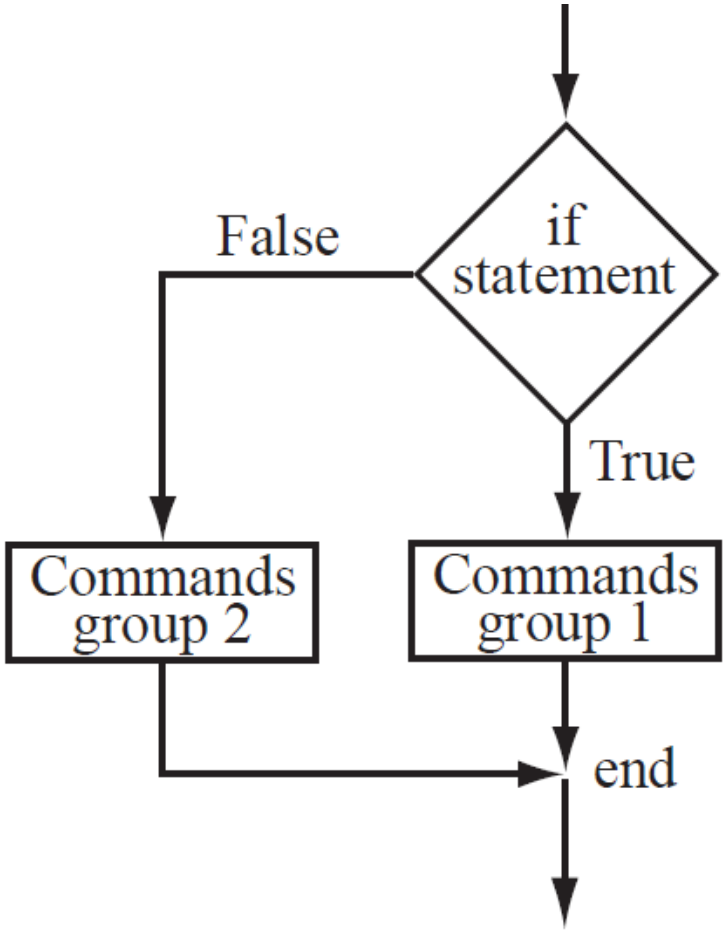
- Variables are local (i.e. not part of workspace) if not declared global

Programming in Matlab

Conditional statements with „switch / case“

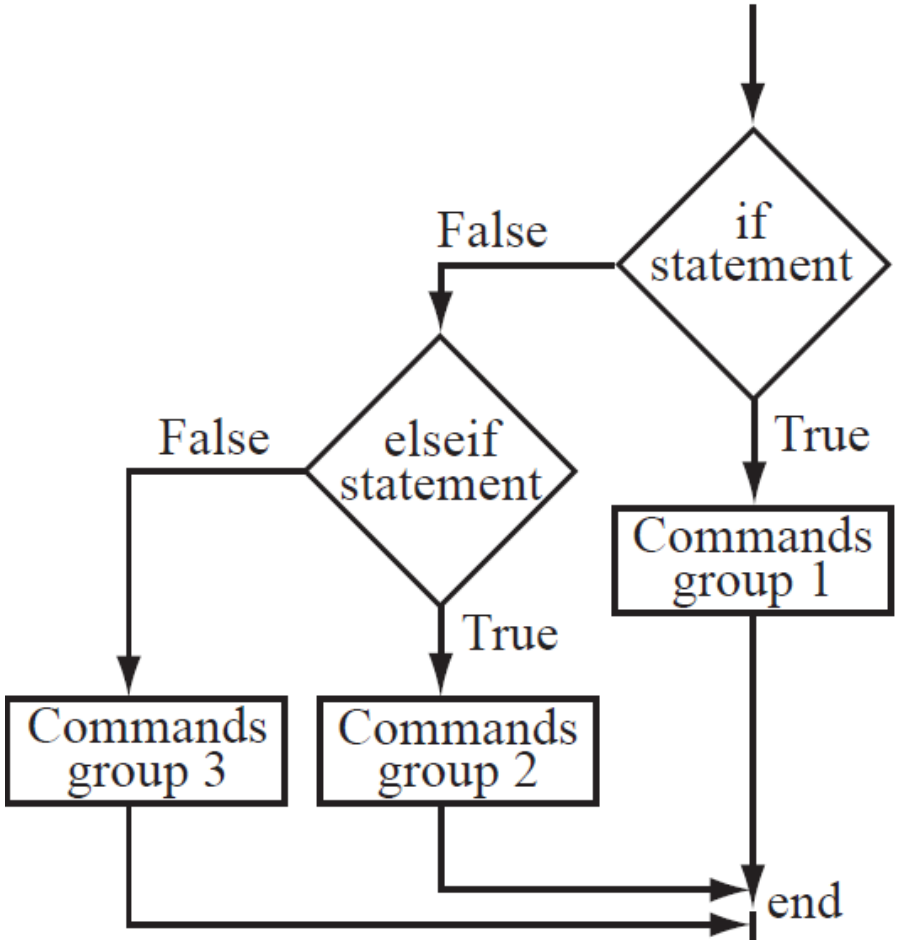
```
switch switch expression
    case value1
    .....
    .....          ] Group 1 of commands.
    case value2
    .....
    .....          ] Group 2 of commands.
    case value3
    .....
    .....          ] Group 3 of commands.
    otherwise
    .....
    .....          ] Group 4 of commands.
end
```

Conditional statements with „if/else“




```
if conditional expression
        .....
        .....        Group 1 of
        .....        MATLAB commands
else
        .....
        .....        Group 2 of
        .....        MATLAB commands
end
```

Conditional statements with „if/else“




```
if conditional expression
        .....
        .....        Group 1 of
        .....        MATLAB commands.
elseif conditional expression
        .....
        .....        Group 2 of
        .....        MATLAB commands.
else
        .....
        .....        Group 3 of
        .....        MATLAB commands
end
```

3D Plots

Several 2D plots in one

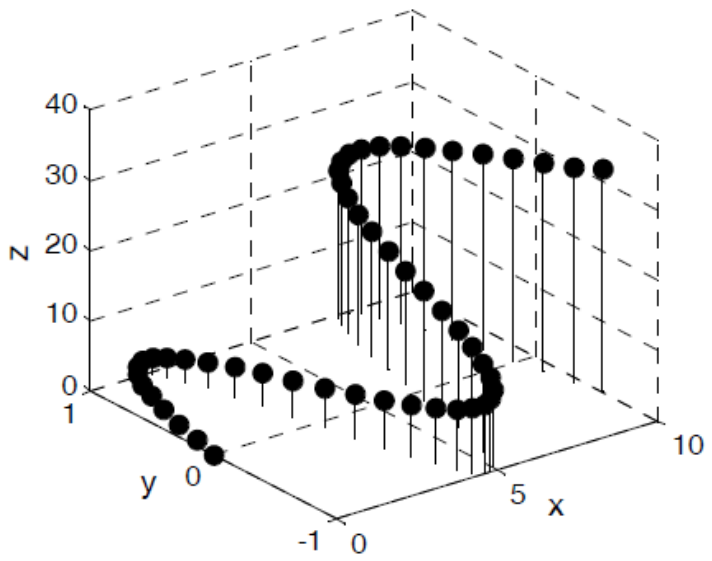
Let M=matrix, A=matrix, x=1D array

- `Plot(M)` - M columns are multiple y arrays, no x array
- `plot(x,M)` - M columns are multiple y arrays, one x array
- `plot(A,M)` - M columns are multiple y arrays, A columns are multiple x arrays.

Line in 3D space

3-D Stem Plot
(draws sequential points with markers and vertical lines from the x y plane)

Function format:
`stem3(X,Y,Z)`




```
t=0:0.2:10;
x=t;
y=sin(t);
z=t.^1.5;
stem3(x,y,z,'fill')
grid on
xlabel('x');
ylabel('y')
zlabel('z')
```

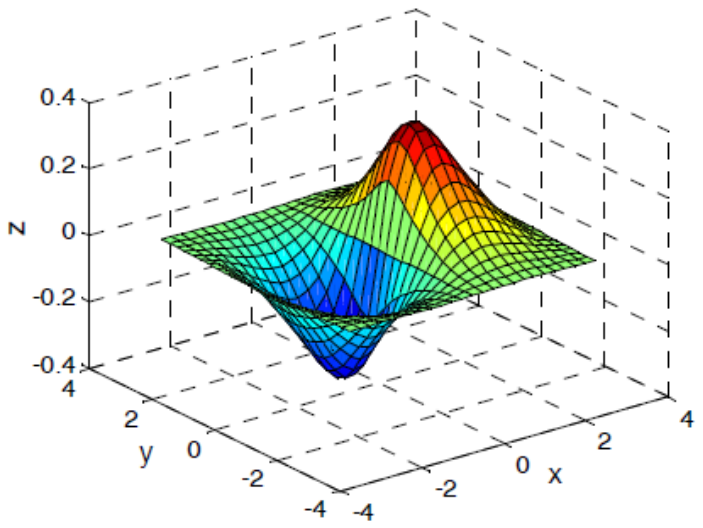
Without bars: scatter3(…)

Plotting functions of 2 variables in 3D

Create a set of input values `[X,Y] = meshgrid(x,y)`

Surface Plot

Function format:
`surf(X,Y,Z)`

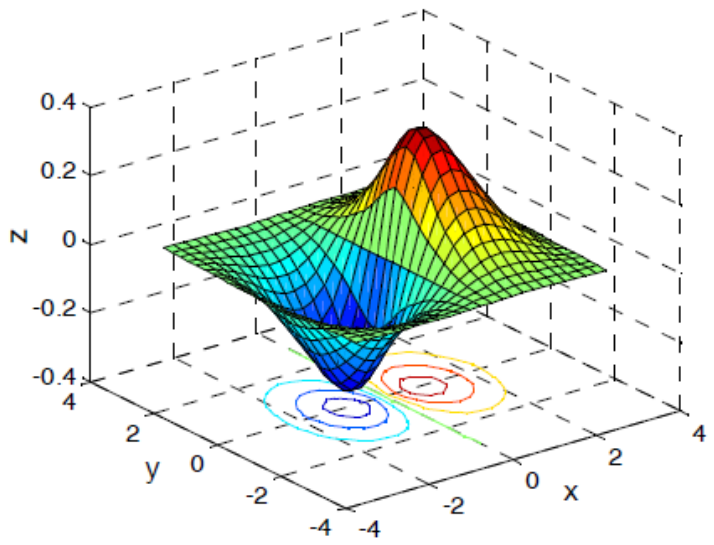



```
x=-3:0.25:3;
y=-3:0.25:3;
[X,Y]=meshgrid(x,y);
Z=1.8.^(-1.5*sqrt(X.^2+
Y.^2)).*cos(0.5*Y).*sin(X);
surf(X,Y,Z)
xlabel('x'); ylabel('y')
zlabel('z')
```

3D plots

Surface and Contour Plot (draws a contour plot beneath the surface)

Function format:
`surfc(X,Y,Z)`

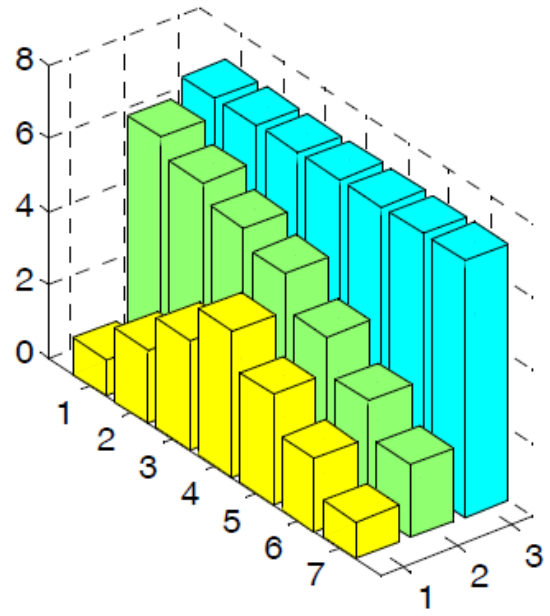



```
x=-3:0.25:3;
y=-3:0.25:3;
[X,Y]=meshgrid(x,y);
Z=1.8.^(-1.5*sqrt(X.^2+
Y.^2)).*cos(0.5*Y).*sin(X);
surfc(X,Y,Z)
xlabel('x'); ylabel('y')
zlabel('z')
```

Matrix values as bars

3-D Bar Plot

Function format:
`bar3(Y)`
Each element in `Y` is one bar. Columns are grouped together.




```
Y=[1 6.5 7; 2 6 7; 3
5.5 7; 4 5 7; 3 4 7;
2 3 7; 1 2 7];

bar3(Y)
```