# MATLAB

**Natesan Srinivasan**

Department of Mathematics

Indian Institute of Technology Guwahati

# Topics

- What is MATLAB?
- Basic Characteristics
- Basic Operations
- Matrix Handling
- Graphics
- Tool Boxes

# What is MATLAB?

- MATLAB – MATrix LABoratory
- Consists of compiled C-code routines
- m-files – source code written in MATLAB
- Available for Windows, Linux, Macintosh, Sun Workstation
- Commercially available from 1984

# Calculator

- >> 2 – 5^2 + 4/3

  ans = -21.6667

- >> p = 2.5*5/8

  p = 1.5625

- >> Q = -34/6+65;

>>

# Output Format

- format – controls how the result of computations is displayed
- Default is format short
- >> format long

```
>> 2 – 5^2 + 4/3
   ans = -21.66666666666667
>> format short ( -21.6667)
```

# Output Format (Cont.)

| Command | Meaning |
|---|---|
| format short | fixed point with 4 decimals places |
| format long | fixed point with 14 decimal places |
| format short e | scientific notation with 4 decimals |
| format long e | scientific notation with 15 decimals |
| format hex | hexadecimal format |
| format rat | approximation by ratio of small integers |

# Elementary Computation

| | |
|---|---|
| General | ; = : , |
| Operators | + - * / |
| Functions | sin, cos, tan, asin, acos, atan, sqrt, exp, log, log10, round |
| Pre-defined constants | pi, eps, realmin, realmax, Inf, NaN |
| Complex Nos. | i, j, real, imag, abs, angle, conj |

# Elementary Functions

- >> sin(pi/4)

  ans = 0.7071

- >> log(2.2026e+04)

  ans = 10.0000

- >> log2(65536)

  ans = 16

- >> exp(10)

  ans = 2.2026e+04

- >> sqrt(2)  % comment

  ans = 1.4142

- >> x = 3; % no printout

  >> y = x^2 – x – 1

  y = 5

# Relational Operators

| Notation | Meaning |
|---|---|
| < | less than |
| <= | less than or equal |
| == | equal |
| >= | greater than or equal |
| > | greater than |
| ~= | not equal |

# Logical Operations

| Notation | Meaning |
|---|---|
| & | and |
| \| | or |
| ~ | not |
| xor | exclusive or |
| 1 | true |
| 0 | false |

# Variable

- MATLAB is **case sensitive**
- Begins with a letter and followed by letters, numbers, underscore
- Valid names: NetCost, Left2Pay, x3, X3, d2r4u
- Invalid names: Net-Cost, 2Pay, %x, @sign
- Must not be longer than 31 characters

# Complex Numbers

- Complex numbers are entered using the imaginary unit i or j = $(\sqrt{-1})$.
- >> z = 1 + i;
- >> i * z * sqrt(z)

ans = -1.5538 + 0.6436 i

- >> abs(z)

ans = 1.4142

- Other operations: **angle, real, imag, conj**

# Matrices

- Matrices are entered by separating elements either by a space or a comma and rows are separated by a semi-colon.
- >> A = [6 5 4; 2, 4, 3; -1 0 9]

```
A =
     6     5     4
     2     4     3
    -1     0     9
```

- >> A(1,3)

```
ans = 4
```

# Matrices (Cont.)

- >> A(1,:)

```
ans =
     6     5     4
>> A(:,2:3)
ans =
     5     4
     4     3
     0     9
```

- >> A([2 3],[2 1])

```
ans =
     4     2
     0    -1
```

- >> b = ones(3,1)

```
b =
     1
     1
     1
```

# Matrices (Cont.)

- >> C = diag([3 4])

```
C =
     3     0
     0     4
```

- >> diag(C)

```
ans  = 3  4
```

- >> C + eye(2)*C-zeros(2)

```
ans =
     6     0
     0     8
```

- >> b'*A

```
ans =
     7     9    16
```

- >> size(b)

```
ans = 3   1
```

- A*A

```
 ans = 42    50    75
       17    26    47
      -15    -5    77
```

# Matrices (Cont.)

- >> A.*A
- ans =

  36  25  16

  4  16  9

  1  0  81
- >> A.^2

ans =

  36  25  16

  4  16  9

  1  0  81

- >> D = randn(1,3)

D =

-0.4326  -1.6656 0.1253

- >> D./D

  ans =  1  1  1
- >> a = [1 5 3 9 8];

[asort, ind] = sort(a)

asot = 1  3  5  8  9

ind  = 1  3  2  5  4

>> sum(a)

ans = 26

# Matrices & Relational Operations

- >>[r,c]=find((A(:,2)>b)&(A(:,1)<=3*b))

[r,c] = [2,1]

>> xor([0 2 0 3], [0 0 1 4])

ans = 0 1 1 0

- >> A >= 5

ans =

1 1 0

0 0 0

0 0 1

- >> t = 0:0.1:10;
- >> size(t)

ans = 1 101

- >> ind = find(t ==5)

ind = 51

- >> y = sin(t);
- >> [ymax,ind] = max(y)

ymax = 0.9996

ind = 17

t(ind) = 1.6000

# Matrix Construction & Operations

| | |
|---|---|
| Operators | + - * ' .' .* .^ ./ |
| Building Matrix | ones, zeros, eye, diag, rand, randn |
| Elementary Matrix Functions | size, length, min, max, sum, sort, find, prod, cumsum, diff, cumprod, reshape |

# Polynomials

- Polynomials are represented as vectors
- p = x^3 + 2x^2 +3x +4
- >>p = [1 2 3 4];
- >>polyval(p,2)

ans = 26

- p1 = [1 1];
- >> pc = conv(p,p1)

pc = 1  3  5  7  4

- Roots of a polynomial will de determined by the command roots

- >> r = roots(pc)

r =  -0.1747 + 1.5469i
     -0.1747 - 1.5469i
     -1.6506
     -1.0000

# Matrix Algebra

```
>>A = [1 2 3; 4 5 6; 7 8 0]
A =     1     2     3
        4     5     6
        7     8     0
>> inv(A)
ans =
  -1.7778    0.8889   -0.1111
   1.5556   -0.7778    0.2222
  -0.1111    0.2222   -0.1111
>> det(A) =      27
>> rank(A) =  3
```

- >> eig(A)

```
ans =    12.1229
         -5.7345
         -0.3884
>> [V,D] = eig(A)
```

- V =

```
-0.7075    0.6582   -0.3884
-0.6400   -0.0931    0.8791
-0.2998   -0.7471   -0.2763
```

- D = 12.1229 0 0

```
0   -0.3884         0
0         0   -5.7345
```

# Matrix Algebra(Cont.)

- >> s = svd(A)

s = 13.2015
5.4388
0.3760

- >> norm(A) = 13.2015
- >> norm(A*V – V*D)

ans = 2.0400e-014

- >> cond(A) = 35.1059
- >> b = [1,1,1];
- >> x = inv(A)*b

x = -1.0000
1.0000
-0.0000

- >> x = A\b

x = -1.0000
1.0000
-0.0000

# Matrix Algebra(Cont.)

- >> B = A(1:2,1:2);
- >>B2 = sqrtm(B)

B2 = 0.5373 + 0.5373i 0.7339 - 0.1967i
1.4679 - 0.3933i 2.0052 + 0.1440i

- >> expm(A) = 1.0e+004 *

3.1591 3.9741 2.7487
7.4540 9.3775 6.4858
6.7431 8.4830 5.8672

# Matrix Algebra(Cont.)

- >> P = poly(A)

P = 1.0000 -6.0000 -72.0000 -27.0000

- >> A^3 - 6*A^2 - 72*A - 27*eye(3)

ans = 0 0 0
0 0 0
0 0 0

- >> r = roots(P)

r = 12.1229
-5.7345
-0.3884

- >> poly(r) = 1.0000 -6.0000 -72.0000 -27.0000

# Matrix Algebra & Polynomials

Operator: \

Functions: inv, eig, det, svd, rank, cond, rcond, rref, sqrtm, expm, conv, roots, poly, polyval, null

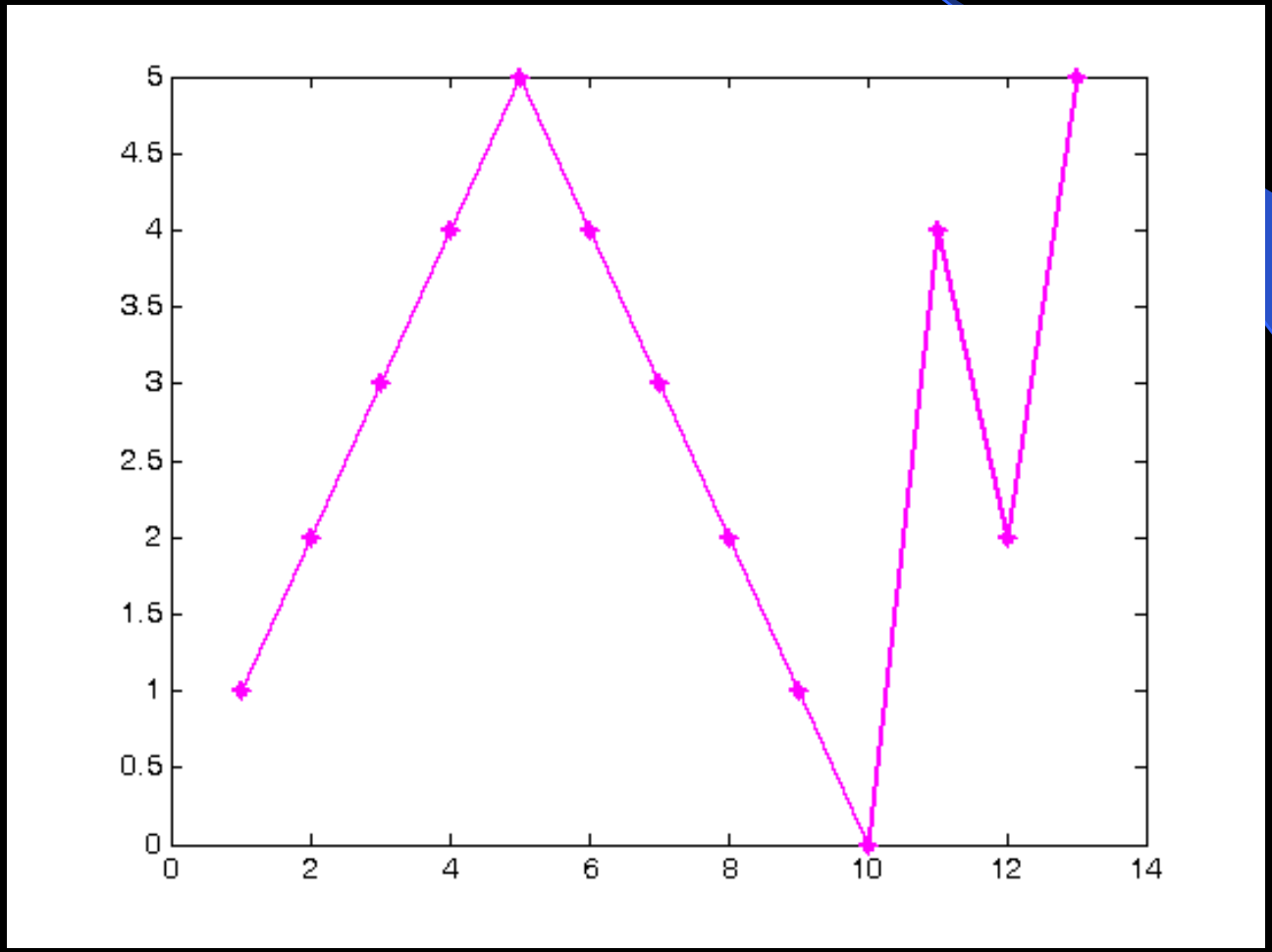
# Graphics

- **Functions**: plot, hist, stem, semilogx, semilogy, loglog, title, xlabel, ylabel, text, gtext, grid, axis, figure, subplot, hold, legend, ginput, zoom, print

# plot(y) & plot(x,y)

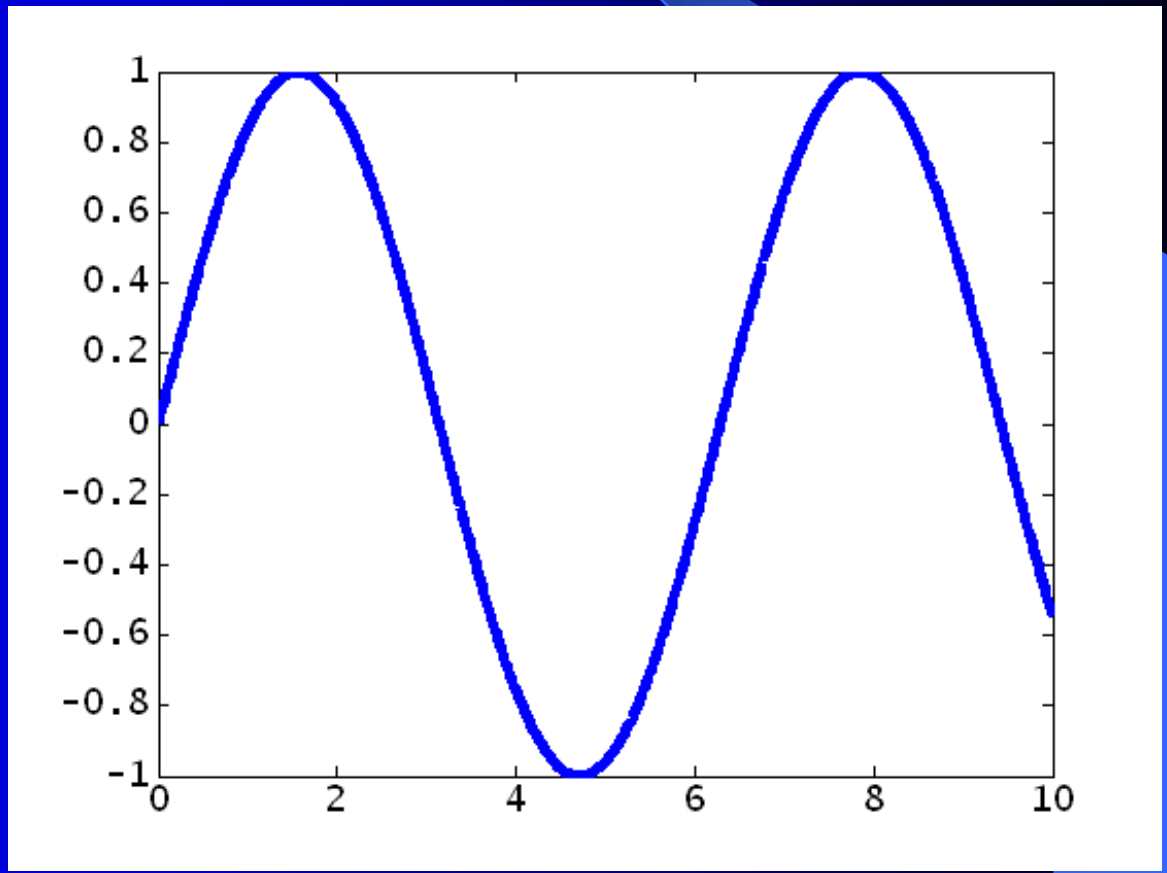
- plot(y) – plots the values in the vector x in a figure window
- plot(x,y) – plots x *vs y;* x in the horizontal axis, and y in the vertical axis
- *plot(y,'-*g') – plots x in green color with * in the prescribed places connecting by a line*

# plot(y,'-*m')
# y = [1:4 5:-1:0 4 2 5]



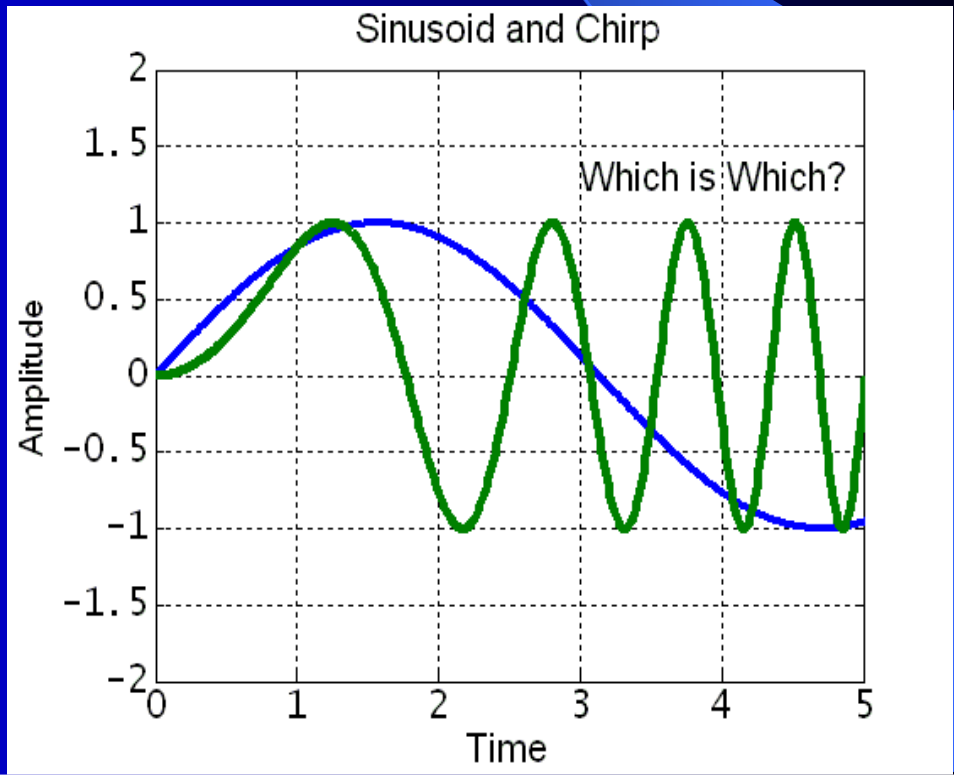
# Line Plot

- >> t = 0:0.01:10;
- plot(t, sin(t))



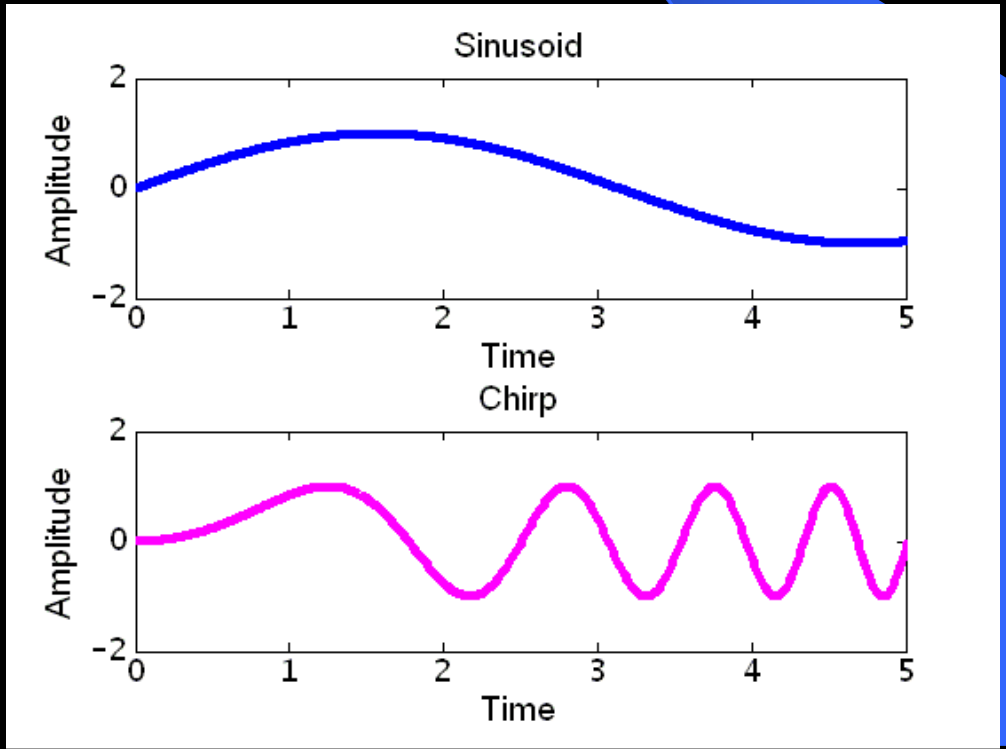
# Graphics

- >> y1 = sin(t); y2 = sin(t.^2); plot(t,[y1',y2'])
- >> title('Sinusoid and chirp'); xlabel('Time'); ylabel('Amplitude'); axis([0 5 –2 2]); grid; text(3,1.3, 'which is which?')



# Subplots

- >> subplot(211); plot(t,sin(t)); title('Sinusoid');
- >> xlabel('Time'); ylabel('Amplitude'); axis([0 5 –2 2])
- >> subplot(212); plot(t,sin(t.^2)); title('Chirp')
- >> xlabel('Time'); ylabel('Amplitude'); axis([0 5 –2 2])



# Advanced Graphics

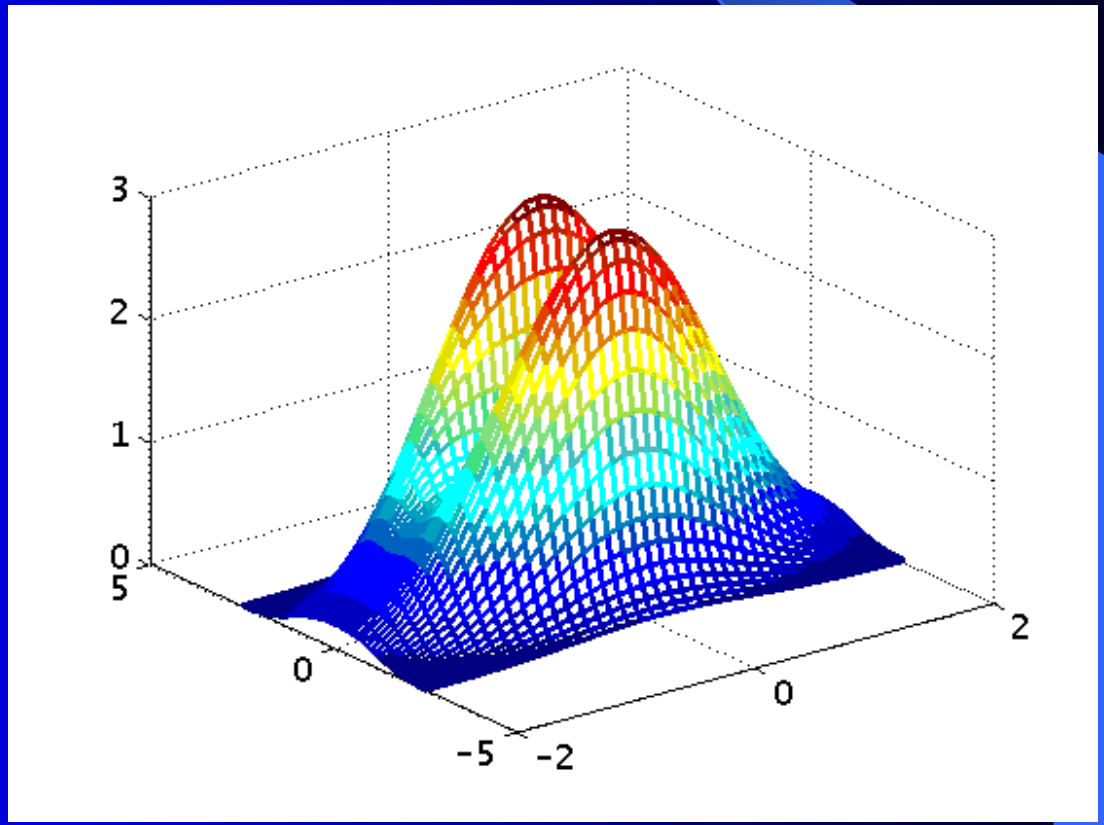
- **Functions**: mesh, meshgrid, view, rotate3d, surf, surfc, surfl, colormap, contour3, clabel, colorbar, propedit

- >> x = -2:0.1:2; y = -2.5:0.1:2.5;
- >> size(x), size(y)

ans = 1  41    & ans = 1  51

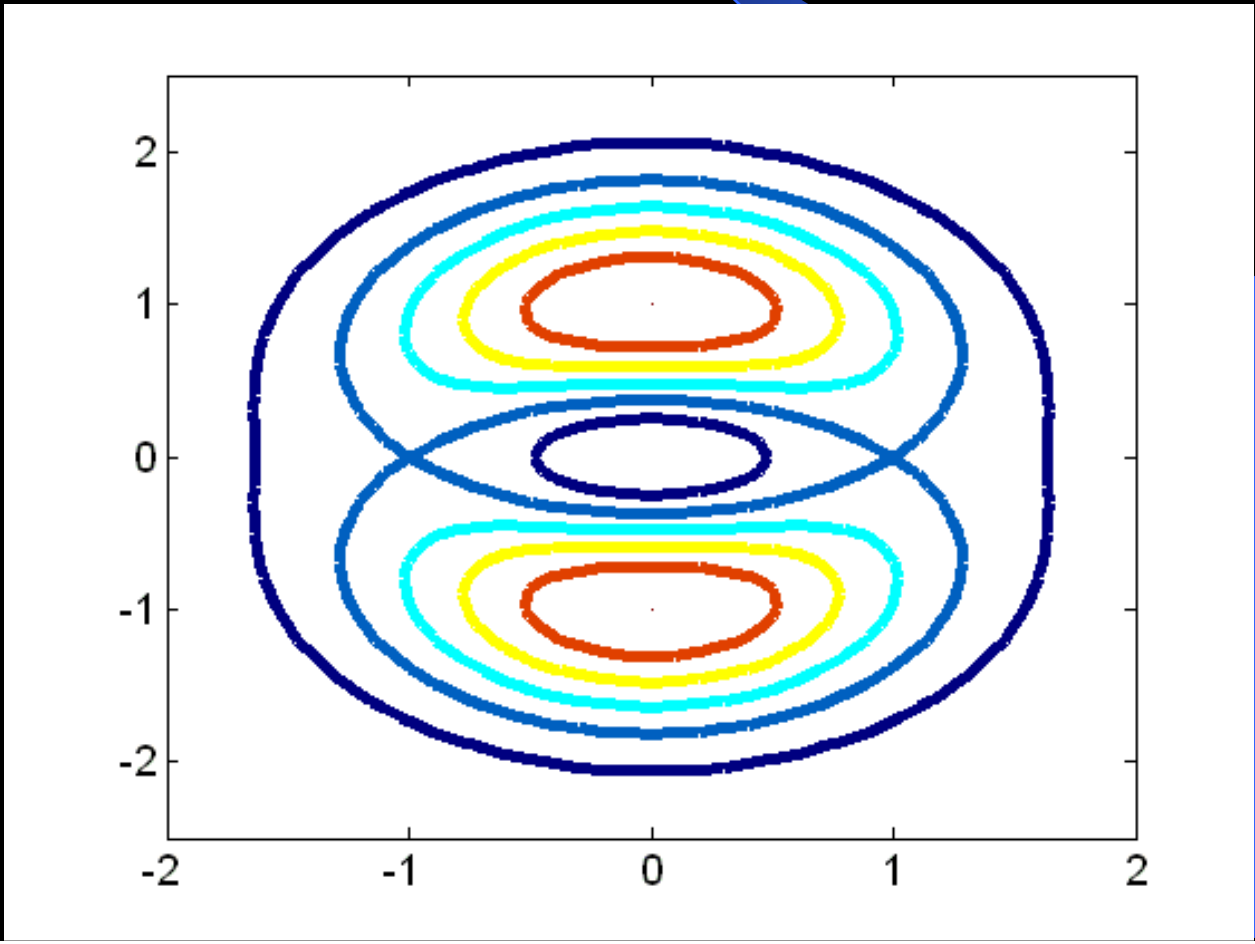
# Advanced Graphics(Cont.)

- >> [X, Y] = meshgrid(x,y);
- >> f = (X.^2 + 3*Y.^2).*exp(1-X.^2 – Y.^2);
- >> mesh(x,y,f)



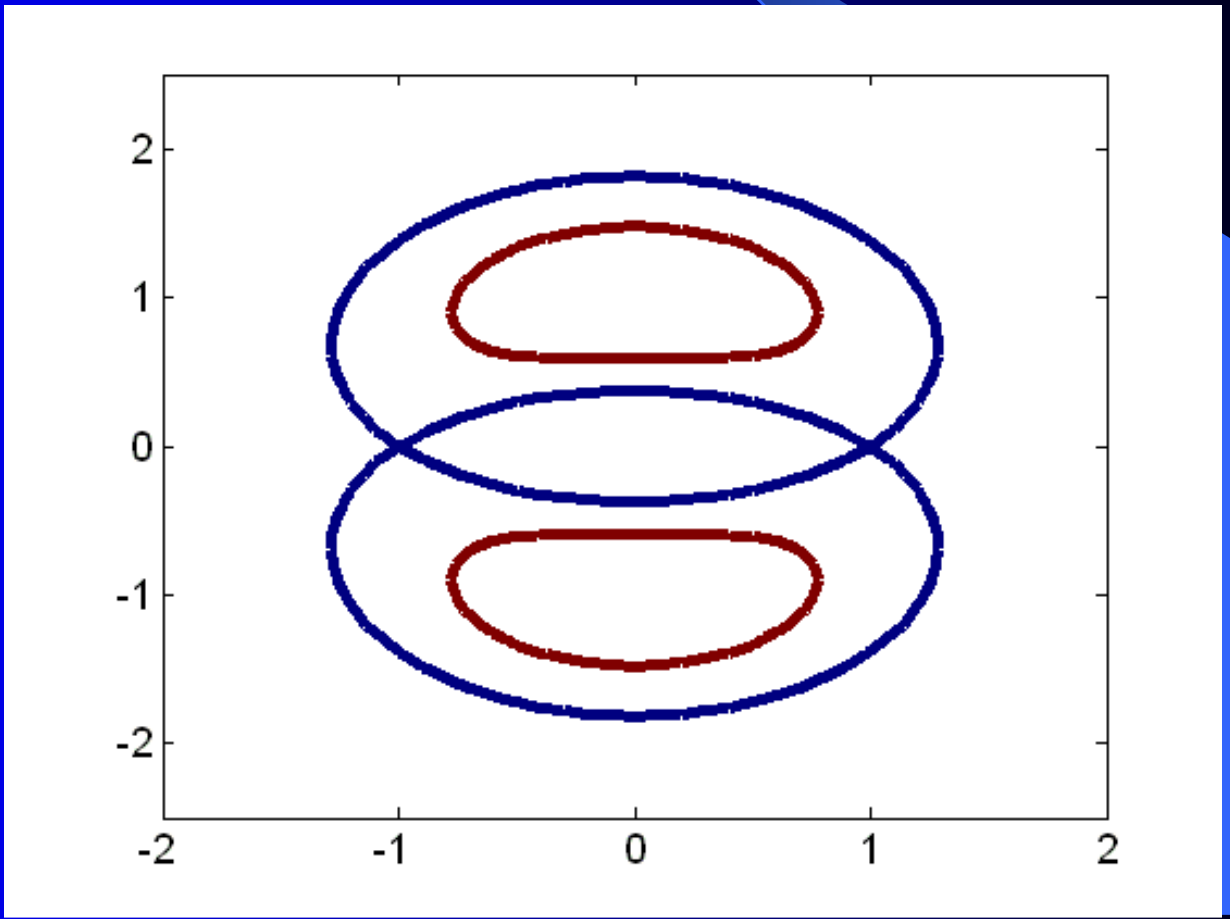
# Contour Plots

- >> contour(x,y,f)



# Contour Plots

- >> contour(x,y,f,[1 2]);

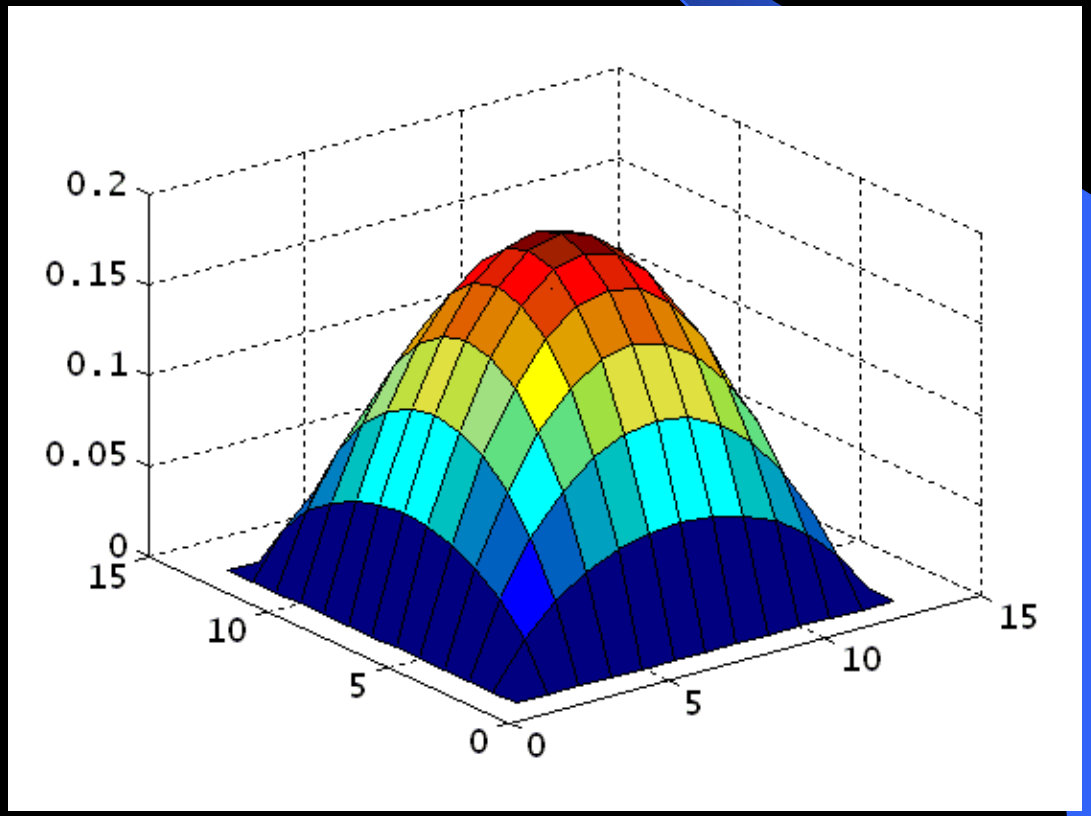


# Poisson Equation

- $-\Delta u = 1, \Omega = (0,1) \times (0,1)$

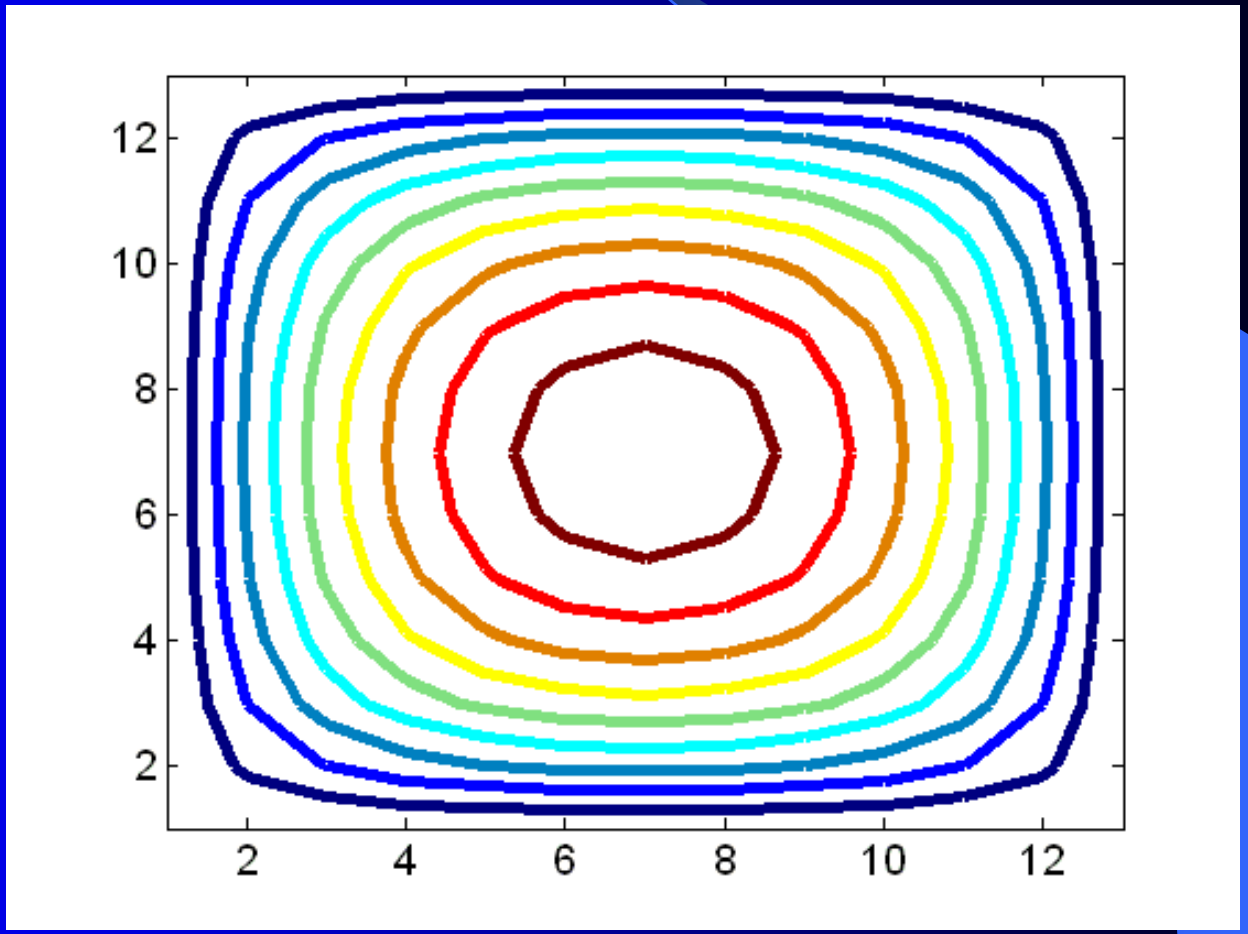
  $u = 0, \Gamma$

- >> surf(u)



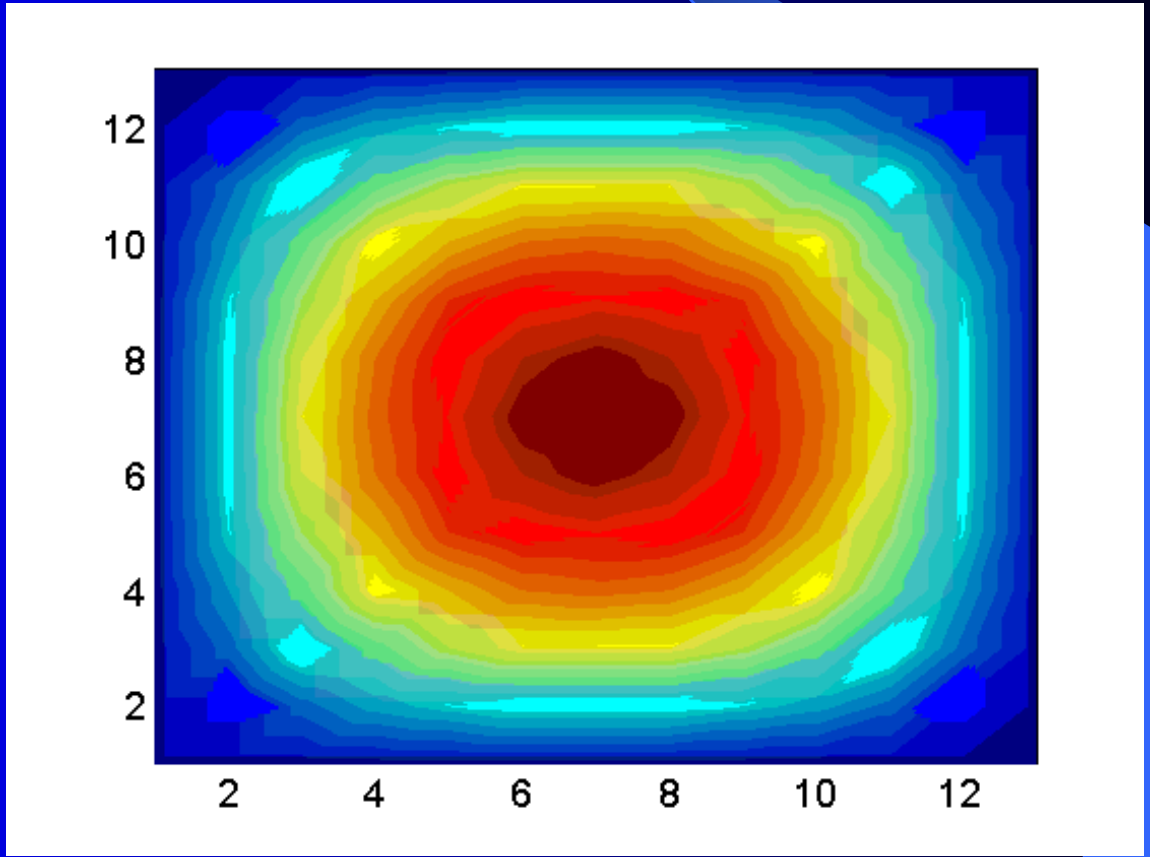
# Poisson Equation

- >> contour(u)



# Poisson Equation

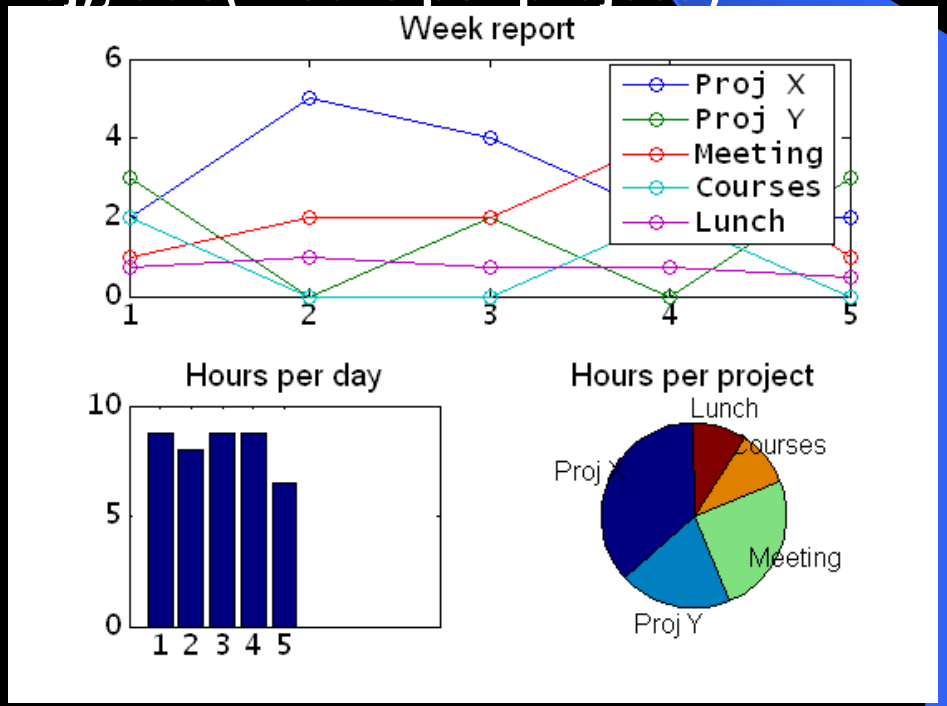
- >> pcolor(u); shading interp



# MATLAB Scripts

- projrep.m

```
proj = {‘Proj X’, ‘Proj Y’, ‘Meeting’, ‘Courses,’Lunch’};
subplot(2,1,1); plot(T,’-o’); legend(‘Week report’);
subplot(2,2,3); bar(sum(T’),’w’);title(‘Hours per Week’)
subplot(2,2,4); pie(sum(T),proj); title(‘Hours per project’)
```



# MATLAB Functions

- **Flow control**: if, else, end, for, while, switch, case, break
- **Functions**: break, input, %, keyword, dbstop, return, error, nargin, nargout
- **Variable context**: global, persistent
- **Timing**: clock, etime, cputime, tic, toc

# Script files & Functions

- Both are referred as m-files and need the file extension .m
- Basic difference between them is how they treat the variables
- Script file uses the global variables defined in the command lines of the workspace
- Running a script file is equivalent to executing a sequence of command lines
- m-file has local variables, input and output parameters need to be specified

# Differences between script & function m-files

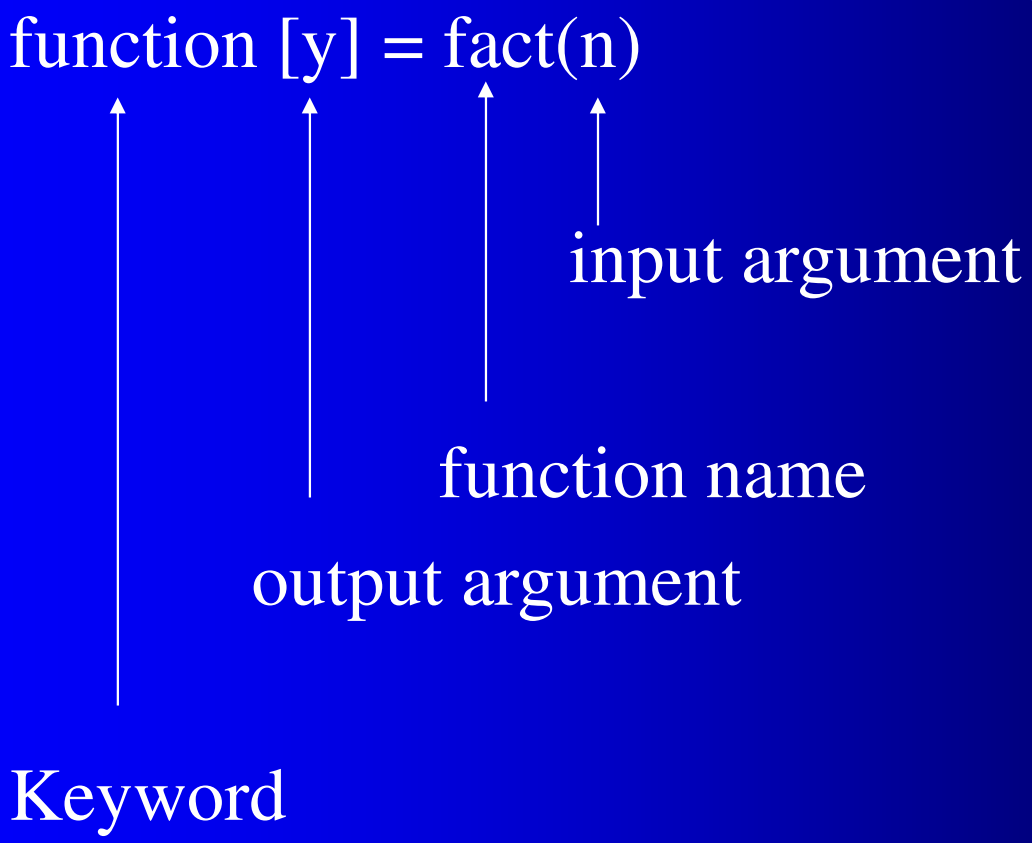
| Script m-files | function m-files |
| --- | --- |
| No input or ouput arguments | May take input and deliver output |
| Operates on the variables in the workspace, variables defined in the script file will be available after execution | Operates on local variables by default. May operate on workspace data if variables are declared as *global* and on previous local variables. May have sub-functions |
| Main use for automation of a series of command steps that needs to be reentered many times, *e.g.*, combination of plotting and visualization | Mainly used to extend the MATLAB system with new commands from your own application field |

# Function m-files

```
function [y] = fact(n)        % function definition
% Y = FACT(N)  Factorial n! = 1*2*…*n  (H1-line)
% FACT(N) computes the factorial    (Help text)
% of N, usually denoted N!  Only valid for integer,
% positive N

% The function body (this comment line is not part of the
% help text)
y = prod(1:n);
```

# Function m-files



# Controlling Program Flow

```
if  <logical condition>
   <statements>
elseif <logical conditions>
   <second case>
else
  <otherwise>
end
```

# if - else - end

```
function y = fact(n)
% Y = FACT(N) Factorial n!
if nargin < 1
  error('no input assigned')
elseif n < 0
  error('input not positive')
elseif abs(n – round(n)) > eps
  error('input must be integer')
end
y = prod(1:n);
```

# switch

```
switch <switch_expression>
  case <case_expression>
     <statements>
otherwise <case_expression>
     <statements>
end
```

otherwise statement is optional

# Statements - for & while

```
for k = <vector>
    <statements>
end
```

---

```
while <logical condition>
  <statements>
end
```

# Examples – for & while

```
% factlimit finds the largest integer n for which n! < realmax
for i = 1:10000
   if fact(i) == Inf
    n = i - 1
    return
   end
end
****************************************
n = 1;
while fact(n+1) < Inf
n = n + 1
end
```

# Functions of functions

- **Functions** : quad, quad1, fminbnd, fzero, eval, feval, inline, @
- quad or quad1 – General purpose numerical integration
- fzero – Finding roots of scalar nonlinear fns.
- ode45 – Solves ordinary differential equations
- feval – Evaluates a fn. specified by its name given in a character array
- eval – Evaluates a complete expression

# Sparse Matrices

- **Functions** : sparse, spy, full, nnz, find,
  speye, spones, spdiags, sprand,
  sprandn, issparse
- >> S = sparse([2 2 3 4 5],[1 4 5 2 2],[10 11 12 13 14])

(2,1) 10 (4,2) 13 (5,2) 14

(2,4) 11 (3,5) 12

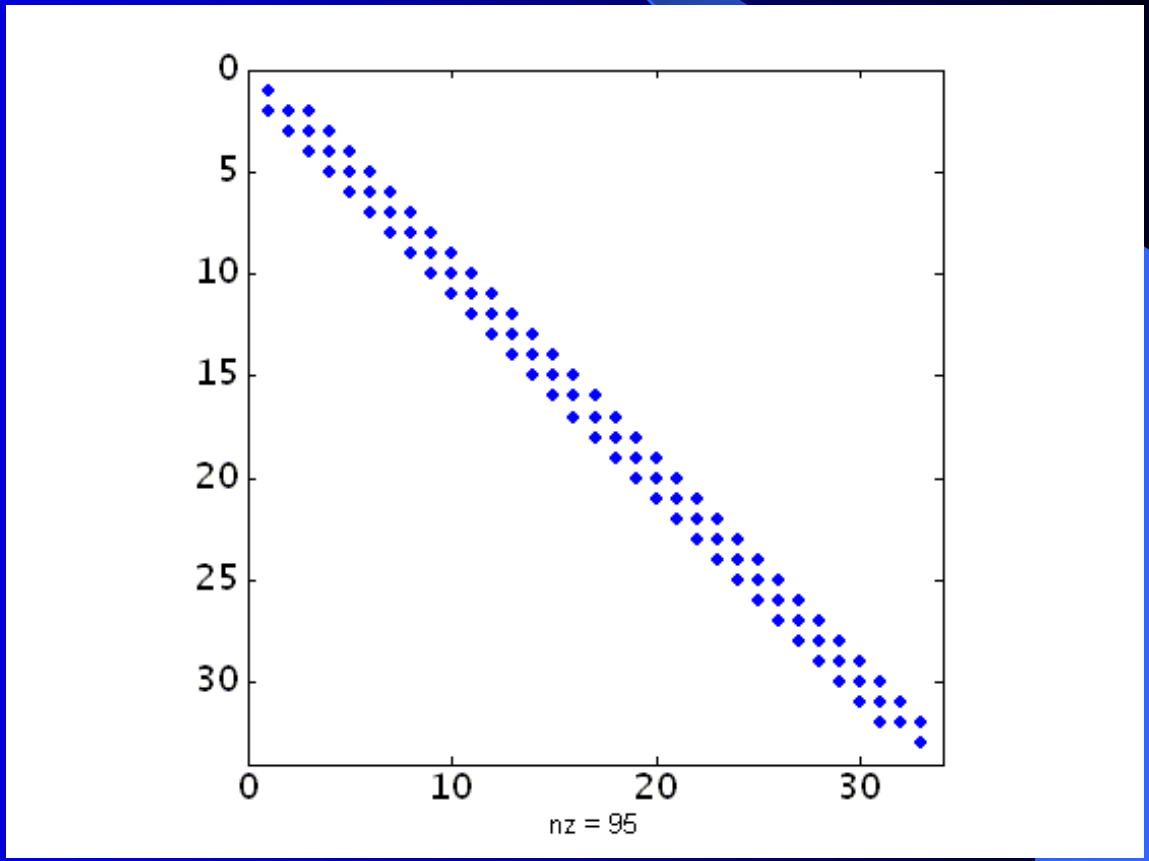
- >> SFull = full(S)

S 5x5 84 double array (sparse)

SFull 5x5 200 double array

# spy

- >> spy(A)  % matrix A – is the stiffness matrix of a two-point BVP.



# Toolboxes

- PDE Toolbox
- Signal Processing
- Optimization
- Statistics Toolbox
- Simulink
- Wavelet Toolbox

# References


- K.E. Chen, P. Giblin, and A. Irving, Mathematical explorations with Matlab, Cambridge University Press, Cambridge, 1999.
- D.M. Etter, D.C. Kuncicky, and D.~Hull, Introduction to Matlab 6, Pearson Education (Singapore) Pvt. Ltd., Delhi, India, 2004.
- F. Gustafsson and N. Bergman, Matlab for Engineers Explained, Springer-Verlag, London, 2003.

# ¿Questions? Comments!