Computational Methods in Physics (PHY 365)

FA23

Dr. Muhammad Kamran

Department of Physics

COMSATS University Islamabad

Lab 2

Exercise: Matrix operations

- A1 = magic(4);

Exercise: Matrix operations

- A1 = magic(4);
- B1 = magic(6);

Exercise: Matrix operations

- A1 = magic(4);
- B1 = magic(6);
- C1 = rand(4,6);

Exercise: Matrix operations

- A1 = magic(4);
- B1 = magic(6);
- C1 = rand(4,6);
- D1 = [A1, B1];

Exercise: Matrix operations

- A1 = magic(4);
- B1 = magic(6);
- C1 = rand(4,6);
- D1 = [A1, B1];
- E1 = [A1; B1];

Exercise: Matrix operations

- F1 = [A1; C1];

Exercise: Matrix operations

- F1 = [A1; C1];
- G1 = [A1, C1];

Exercise: Matrix operations

- F1 = [A1; C1];
- G1 = [A1, C1];
- A1 (2) = 0;

Exercise: Matrix operations

- F1 = [A1; C1];
- G1 = [A1, C1];
- A1 (2) = 0;
- H1 = B1 (: , 3)

Exercise: Matrix operations

- F1 = [A1; C1];
- G1 = [A1, C1];
- A1 (2) = 0;
- H1 = B1 (: , 3)
- I1 = A1 (3 : 6, 3)

Exercise: Matrix operations

- a = [1 : 3]

  b = [5 : 7]

Exercise: Matrix operations

- a = [1 : 3]

  b = [5 : 7]

- a1 = dot(a , b)

Exercise: Matrix operations

- a = [1 : 3]

  b = [5 : 7]

- a1 = dot(a , b)

- a2 = dot(a , b')

Exercise: Matrix operations

- a = [1 : 3]

  b = [5 : 7]

- a1 = dot(a , b)
- a2 = dot(a , b')
- a3 = cross(a , b)

Exercise: Matrix operations

- a = [1 : 3]

  b = [5 : 7]

- a1 = dot(a , b)
- a2 = dot(a , b')
- a3 = cross(a , b)
- a4 = kron(a , b)

Exercise: Matrix operations

- a = [1 : 3]

  b = [5 : 7]

- a1 = dot(a , b)
- a2 = dot(a , b')
- a3 = cross(a , b)
- a4 = kron(a , b)
- a5 = length(a)

Exercise: Matrix operations

- a = [1 : 3]

  b = [5 : 7]

- a1 = dot(a , b)
- a2 = dot(a , b')
- a3 = cross(a , b)
- a4 = kron(a , b)
- a5 = length(a)
- a6 = max(a)

Exercise: Matrix operations

- a = [1 : 3]

  b = [5 : 7]

- a1 = dot(a , b)
- a2 = dot(a , b')
- a3 = cross(a , b)
- a4 = kron(a , b)
- a5 = length(a)
- a6 = max(a)
- a7 = min(b)

Exercise: Matrix operations

- A = [1,2,3,4 ; 5,6,7,8 ; 9,10,11,12]

Exercise: Matrix operations

- A = [1,2,3,4 ; 5,6,7,8 ; 9,10,11,12]

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

Exercise: Matrix operations

- A = [1,2,3,4 ; 5,6,7,8 ; 9,10,11,12]

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

- B = A(0)

Exercise: Matrix operations

- A = [1,2,3,4 ; 5,6,7,8 ; 9,10,11,12]

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

- B = A(0)
- C = A(1)

Exercise: Matrix operations

- A = [1,2,3,4 ; 5,6,7,8 ; 9,10,11,12]

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

- B = A(0)
- C = A(1)
- D = A(4)

Exercise: Matrix operations

- A = [1,2,3,4 ; 5,6,7,8 ; 9,10,11,12]

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

- B = A(0)
- C = A(1)
- D = A(4)
- E = A(end)

Exercise: Matrix operations

- F = size(A)

Exercise: Matrix operations

- F = size(A)
- G = length(A)

Exercise: Matrix operations

- F = size(A)
- G = length(A)
- H = A(2 , 3)

Exercise: Matrix operations

- F = size(A)
- G = length(A)
- H = A(2 , 3)
- A(2,2) = 20

Exercise: Matrix operations

- F = size(A)
- G = length(A)
- H = A(2 , 3)
- A(2,2) = 20
- I = A(: , 2)

Exercise: Matrix operations

- F = size(A)
- G = length(A)
- H = A(2 , 3)
- A(2,2) = 20
- I = A(: , 2)
- J = sum(A)

Exercise: Matrix operations

- F = size(A)
- G = length(A)
- H = A(2 , 3)
- A(2,2) = 20
- I = A(: , 2)
- J = sum(A)
- K = max(A)

Exercise: Matrix operations

- Create a 5×5 "magic square". Verify that the sum of the integers in each row, column and diagonal is equal.

Exercise: Matrix operations

- Create a 5×5 "magic square". Verify that the sum of the integers in each row, column and diagonal is equal.

```
A = magic(5);
```

Exercise: Matrix operations

- Create a 5×5 "magic square". Verify that the sum of the integers in each row, column and diagonal is equal.

```
A = magic(5);
A1 = sum(A)
```

Exercise: Matrix operations

- Create a 5×5 "magic square". Verify that the sum of the integers in each row, column and diagonal is equal.

```
A = magic(5);

A1 = sum(A)

A2 = sum(A,2)
```

Exercise: Matrix operations

- Create a 5×5 "magic square". Verify that the sum of the integers in each row, column and diagonal is equal.

```
A = magic(5);

A1 = sum(A)

A2 = sum(A,2)

A3 = trace(A)
```

Exercise: Matrix operations

- Create a 5×5 "magic square". Verify that the sum of the integers in each row, column and diagonal is equal.

```
A = magic(5);

A1 = sum(A)

A2 = sum(A,2)

A3 = trace(A)

A4 = trace(flip(A))
```

Exercise: Matrix operations

- Create a 5×5 "magic square". Verify that the sum of the integers in each row, column and diagonal is equal.

```
A = magic(5);

A1 = sum(A)

A2 = sum(A,2)

A3 = trace(A)

A4 = trace(flip(A))
```

- Extract the 3rd row of matrix A.

Exercise: Matrix operations

- Create a 5×5 "magic square". Verify that the sum of the integers in each row, column and diagonal is equal.

```
A = magic(5);

A1 = sum(A)

A2 = sum(A,2)

A3 = trace(A)

A4 = trace(flip(A))
```

- Extract the 3rd row of matrix A.

```
A5 = A(3 , :)
```

Exercise: Matrix operations

- Take transpose of the matrix A.

Exercise: Matrix operations

- Take transpose of the matrix A.

  C1 = A'

Exercise: Matrix operations

- Take transpose of the matrix A.

  C1 = A'

- Determine the highest integer in the matrix A.

Exercise: Matrix operations

- Take transpose of the matrix A.

  C1 = A'

- Determine the highest integer in the matrix A.

  C2 = max(max(A))

Exercise: Matrix operations

- Take transpose of the matrix A.

  C1 = A'

- Determine the highest integer in the matrix A.

  C2 = max(max(A))

- Sort the matrix A column-wise, in ascending order.

Exercise: Matrix operations

- Take transpose of the matrix A.

  C1 = A'

- Determine the highest integer in the matrix A.

  C2 = max(max(A))

- Sort the matrix A column-wise, in ascending order.

  C3 = sort(A)

Exercise: Matrix operations

- Take transpose of the matrix A.

  C1 = A'

- Determine the highest integer in the matrix A.

  C2 = max(max(A))

- Sort the matrix A column-wise, in ascending order.

  C3 = sort(A)

- Sort the matrix A row-wise, in descending order.

Exercise: Matrix operations

- Take transpose of the matrix A.

  C1 = A'

- Determine the highest integer in the matrix A.

  C2 = max(max(A))

- Sort the matrix A column-wise, in ascending order.

  C3 = sort(A)

- Sort the matrix A row-wise, in descending order.

  C4 = sort(A , 2 ,'descend')

Exercise: Matrix operations

- Take transpose of the matrix A.

  C1 = A'

- Determine the highest integer in the matrix A.

  C2 = max(max(A))

- Sort the matrix A column-wise, in ascending order.

  C3 = sort(A)

- Sort the matrix A row-wise, in descending order.

  C4 = sort(A , 2 ,'descend')

- Take determinant of the matrix A.

Exercise: Matrix operations

- Take transpose of the matrix A.

  C1 = A'

- Determine the highest integer in the matrix A.

  C2 = max(max(A))

- Sort the matrix A column-wise, in ascending order.

  C3 = sort(A)

- Sort the matrix A row-wise, in descending order.

  C4 = sort(A , 2 ,'descend')

- Take determinant of the matrix A.

  C5 = det(A)

Exercise: "for" loop

- for index = values
  statements
  end

Exercise: "for" loop

- for index = values
  statements
  end

- for i = 1 : 10
  end

Exercise: "for" loop

- for index = values
  statements
  end

- for i = 1 : 10
  end

- for a = 1 : 5
  rand
  end

Exercise: "for" loop

- for index = values
  statements
  end

- for i = 1 : 10
  end

- for a = 1 : 5
  rand
  end

- for a = 1 : 5
  rand
  pause(3)
  end

Exercise: "for" loop

- my_sum = 0;

  for i = 0 : 10

      my_sum = my_sum + 1;

  end

Exercise: "for" loop

- my_sum = 0;

  for i = 0 : 10

  my_sum = my_sum + 1;

  end

- for k = 0 : 10

  A(k + 1) = 1;

  end

Exercise: "for" loop

- my_sum = 0;

  for i = 0 : 10

  my_sum = my_sum + 1;

  end

- for k = 0 : 10

  A(k + 1) = 1;

  end

  my_sum_2 = sum(A);

Exercise: "if" loop

- if expression
  statements
  elseif expression
  statements
  else
  statements
  end

Exercise: "if" loop

- if expression
  statements
  elseif expression
  statements
  else
  statements
  end

- a = rand;
  if a < 0.25
  disp('1')
  elseif a > 0.25 && a < 0.5
  disp('2')
  elseif a > 0.5 && a < 0.75
  disp('3')
  else
  disp('4')
  end

Exercise: "while" loop

- while expression

  statements

  end

Exercise: "while" loop

- while expression

  statements

  end

- j = 0;

  while j < 5

  disp('1')

  j = j + 1;

  pause

  end

  disp('0')

Exercise: 2-D line plot

- Defining the x vector

```
x_i = 0;
x_f = 4 * pi;
x = linspace(x_i, x_f);
```

Exercise: 2-D line plot

- Defining the x vector

```
x_i = 0;
x_f = 4 * pi;
x = linspace(x_i, x_f);
```

- Calculating the sin and cos

```
sin_x = sin(x);
cos_x = cos(x);
```

Exercise: 2-D line plot

- Defining the x vector

```
x_i = 0;
x_f = 4 * pi;
x = linspace(x_i, x_f);
```

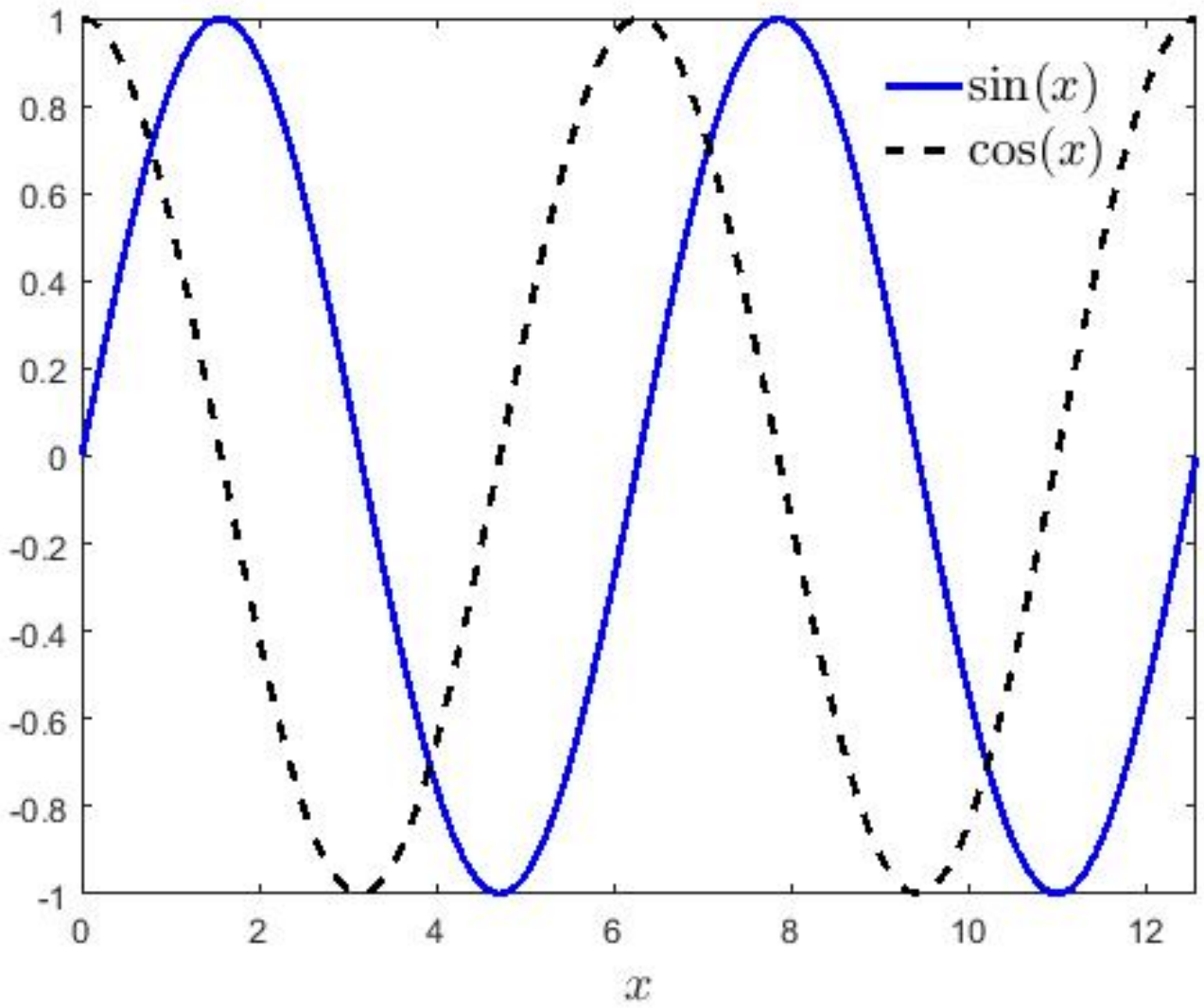
- Calculating the sin and cos

```
sin_x = sin(x);
cos_x = cos(x);
```

- Plotting

```
plot(x, sin_x, 'b', x, cos_x, 'k- -')
```

Exercise: 2-D line plot



Exercise: 2-D line plot

```
% This is my first MATLAB program, and it plots sin(x) and cos(x)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear variables

close all

clc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Defining the x vector

x_i = 0;

x_f = 4 * pi;

x = linspace(x_i,x_f);
```

Exercise: 2-D line plot

```
% Calculating sin and cos

sin_x = sin(x);

cos_x = cos(x);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plotting

plot(x,sin_x,'b',x,cos_x,'k--','LineWidth',2)

xlim([x_i,x_f])

xlabel('$$x$$','Interpreter','latex','fontsize',15)

legend({'$$\sin(x)$$','$$\cos(x)$$'},'Interpreter','latex','fontsize',15)

legend boxoff
```

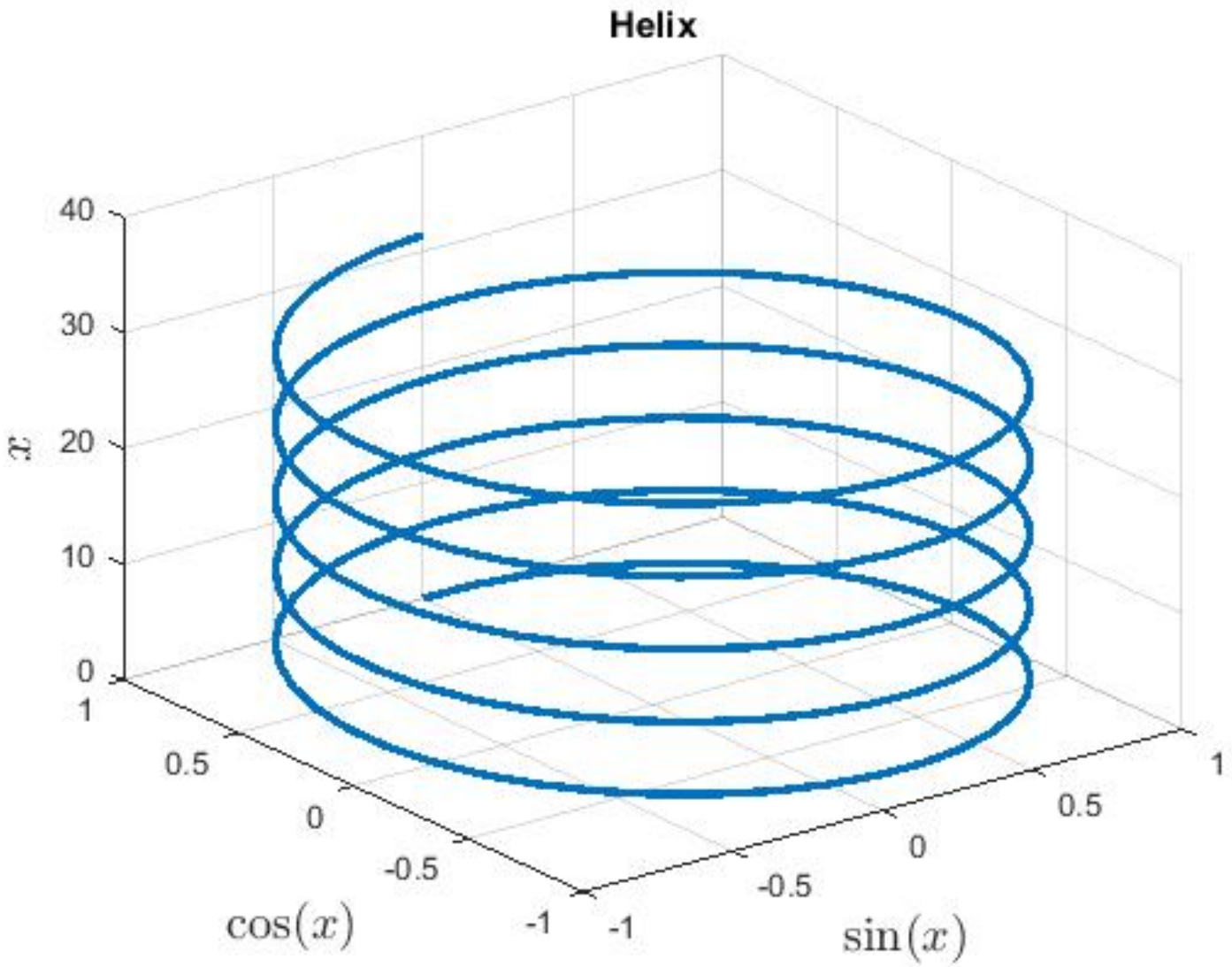
Exercise: 3-D line plot

```
% This program plots a helix using "plot3" function

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear variables

close all

clc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Defining the x vector

x = linspace(0,10 * pi,200);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculating sin and cos

sin_x = sin(x);

cos_x = cos(x);
```

Exercise: 3-D line plot

```
% Plotting

plot3(sin_x,cos_x,x,'linewidth',2.5)

xlabel('$$\sin(x)$$','Interpreter','latex','fontsize',15)

ylabel('$$\cos(x)$$','Interpreter','latex','fontsize',15)

zlabel('$$x$$','Interpreter','latex','fontsize',15)

grid on

title('Helix')
```

Exercise: 3-D line plot



Exercise: Surface plot

- Defining the x and y vectors

```
x_vec = linspace(-5, 5);
y_vec = linspace(-5, 5);
```

Exercise: Surface plot

- Defining the x and y vectors

  x_vec = linspace(−5, 5);

  y_vec = linspace(−5, 5);

- Meshgrid

  [X, Y] = meshgrid(x_vec, y_vec);

Exercise: Surface plot

- Defining the x and y vectors

```
x_vec = linspace(-5, 5);
y_vec = linspace(-5, 5);
```

- Meshgrid

```
[X, Y] = meshgrid(x_vec, y_vec);
```

- Defining the function

```
Z = Y .* sin(X) - X .* cos(Y);
```

Exercise: Surface plot

- Defining the x and y vectors

  x_vec = linspace(−5, 5);

  y_vec = linspace(−5, 5);

- Meshgrid

  [X, Y] = meshgrid(x_vec, y_vec);

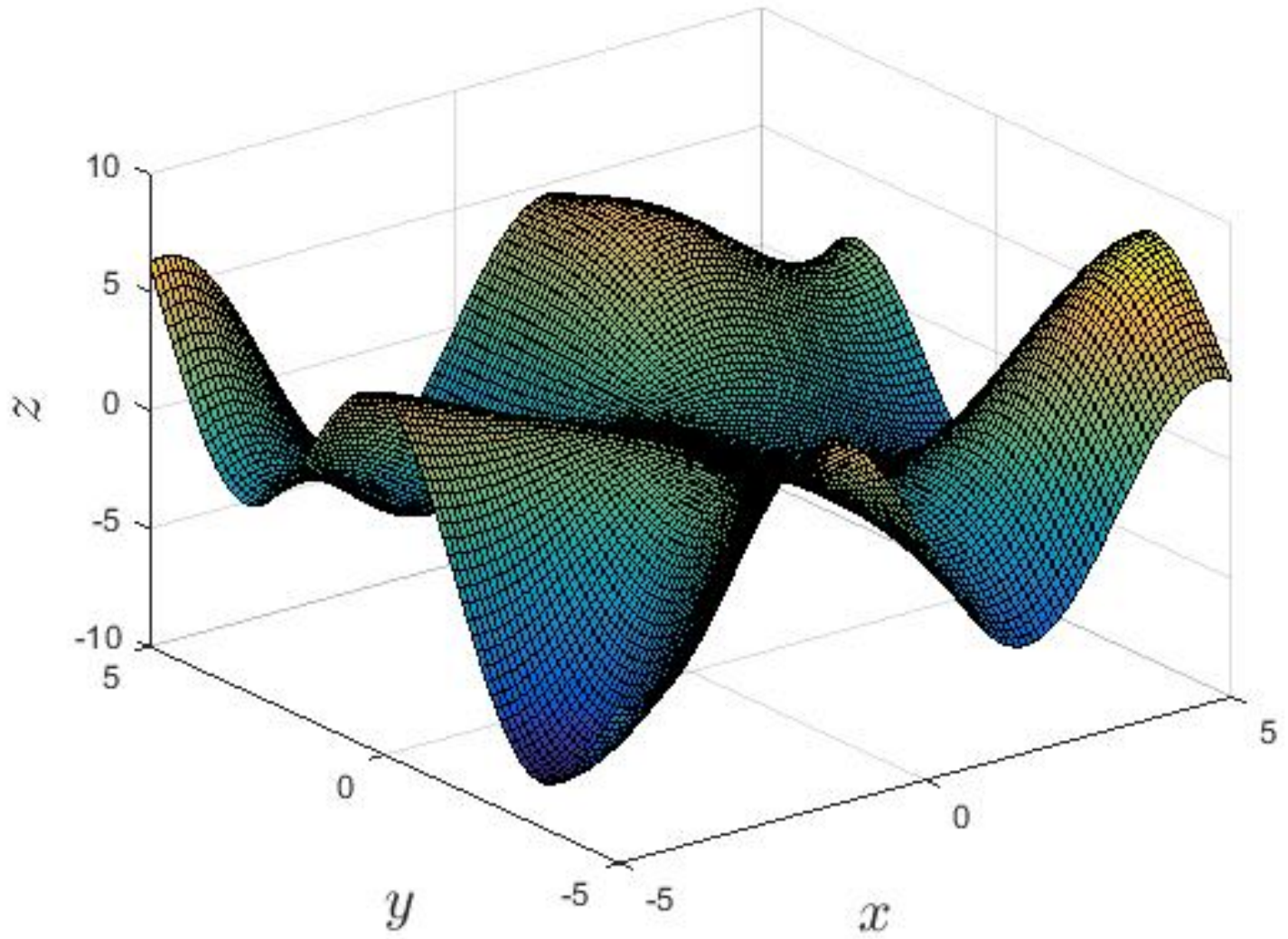
- Defining the function

  Z = Y .* sin(X) − X .* cos(Y);

- Plotting

  surf (X, Y, Z)

Exercise: Surface plot



Exercise: Contour plot

- Defining the x and y vectors

```
x_vec = linspace(0, 2 * pi);
y_vec = linspace(0, 4 * pi);
```

Exercise: Contour plot

- Defining the x and y vectors

```
x_vec = linspace(0, 2 * pi);
y_vec = linspace(0, 4 * pi);
```

- Meshgrid

```
[X, Y] = meshgrid(x_vec, y_vec);
```

Exercise: Contour plot

- Defining the x and y vectors

```
x_vec = linspace(0, 2 * pi);
y_vec = linspace(0, 4 * pi);
```

- Meshgrid

```
[X, Y] = meshgrid(x_vec, y_vec);
```

- Defining the function

```
Z = sin (X) + cos (Y);
```

Exercise: Contour plot

- Defining the x and y vectors

```
x_vec = linspace(0, 2 * pi);
y_vec = linspace(0, 4 * pi);
```

- Meshgrid

```
[X, Y] = meshgrid(x_vec, y_vec);
```

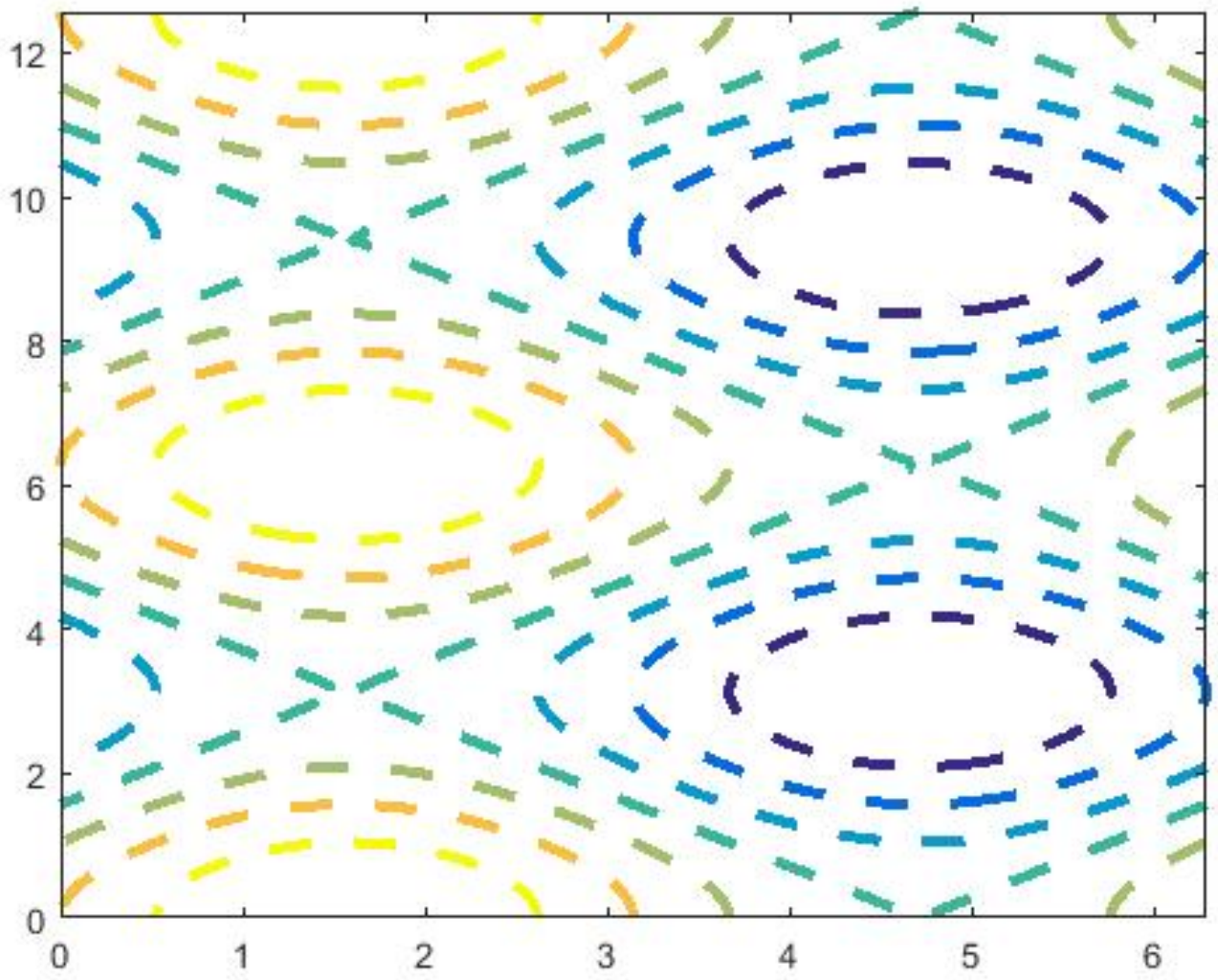
- Defining the function

```
Z = sin (X) + cos (Y);
```

- Plotting

```
contour (X, Y, Z)
```

Exercise: Contour plot



Exercise: Anonymous Functions

- Defining the dependent function

```
y = x .^ 2;
```

Exercise: Anonymous Functions

- Defining the dependent function

```
y = x .^ 2;
```

Error: Undefined function or variable 'x'

Exercise: Anonymous Functions

- Defining the dependent function

  y = x .^ 2;

  Error: Undefined function or variable 'x'

- Introduce anonymous function

  y = @ (x) x .^ 2;

Exercise: Anonymous Functions

- Defining the dependent function

```
y = x .^ 2;
```

Error: Undefined function or variable 'x'

- Introduce anonymous function

```
y = @ (x) x .^ 2;
y(3);
```

Exercise: Symbolic toolbox

- sym (1/3)

Exercise: Symbolic toolbox

- sym (1/3)
- sin (sym (pi))

Exercise: Symbolic toolbox

- sym (1/3)
- sin (sym (pi))
- t = 0.1;

  sym (t)

Exercise: Symbolic toolbox

- sym (1/3)
- sin (sym (pi))
- t = 0.1;

  sym (t)
- A = 1/sqrt(2)

  B = sym(A)

Exercise: Symbolic toolbox

- sym (1/3)
- sin (sym (pi))
- t = 0.1;

  sym (t)
- A = 1/sqrt(2)

  B = sym(A)
- A = sym ('a', [1, 20])

Exercise: Symbolic toolbox

- sym (1/3)
- sin (sym (pi))
- t = 0.1;

  sym (t)
- A = 1/sqrt(2)

  B = sym(A)
- A = sym ('a', [1, 20])
- phi = (1 + sqrt (sym (5)))/2

  f = phi^2 – phi – 1

Exercise: Symbolic toolbox

- syms x y

  $\log(x) + \exp(y)$

Exercise: Symbolic toolbox

- syms x y

  log(x) + exp(y)

- syms x

  f = sin(x)^2;

  diff(f)

  int(f)

Exercise: Symbolic toolbox

- syms x y

  log (x) + exp (y)

- syms x

  f = sin(x)^2;

  diff(f)

  int(f)

- syms f(x,y)

  f(x,y) = x^2 * y

  f(3,2)

  subs(f, x, 3)

Exercise: Symbolic toolbox

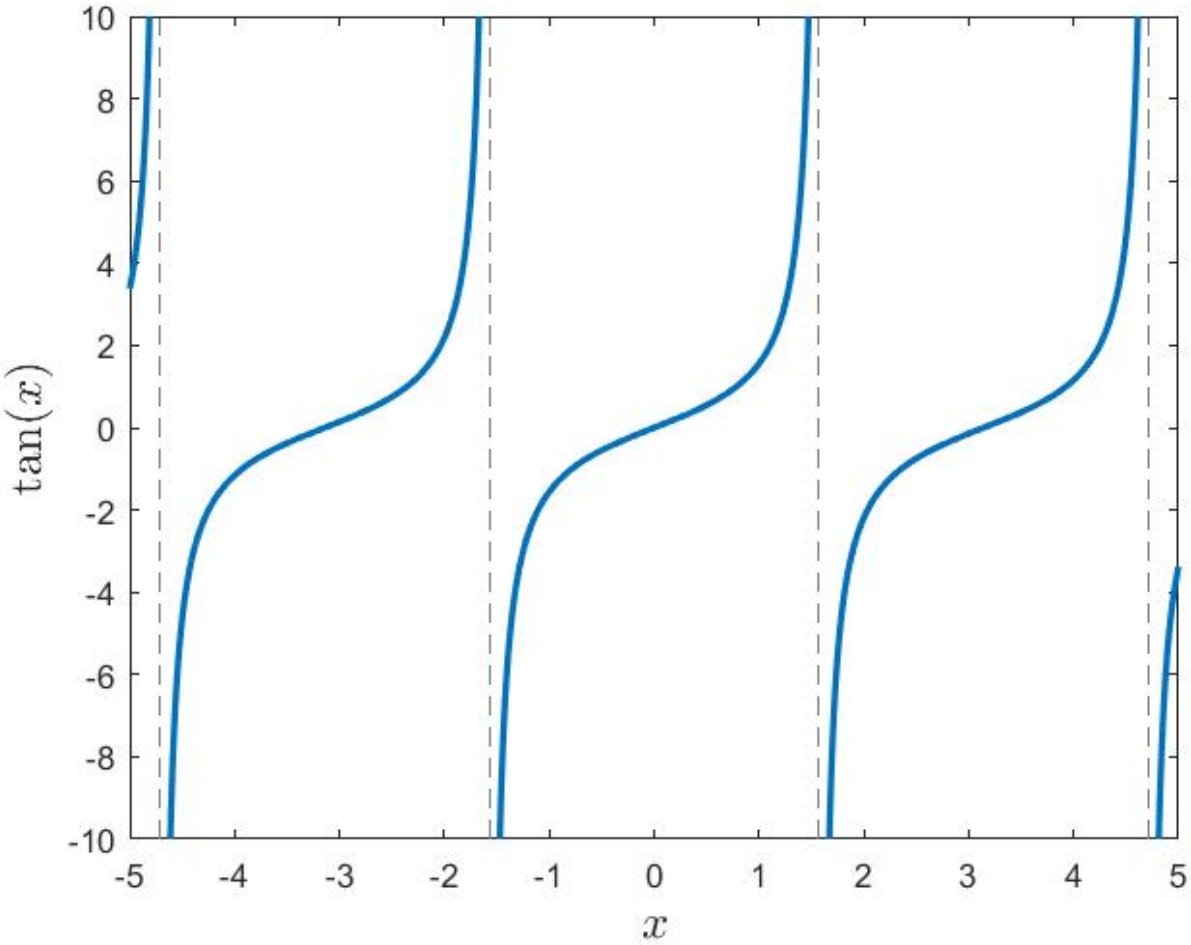
- syms x

  fplot(tan(x))

Exercise: Symbolic toolbox

- syms x

  fplot(tan(x))

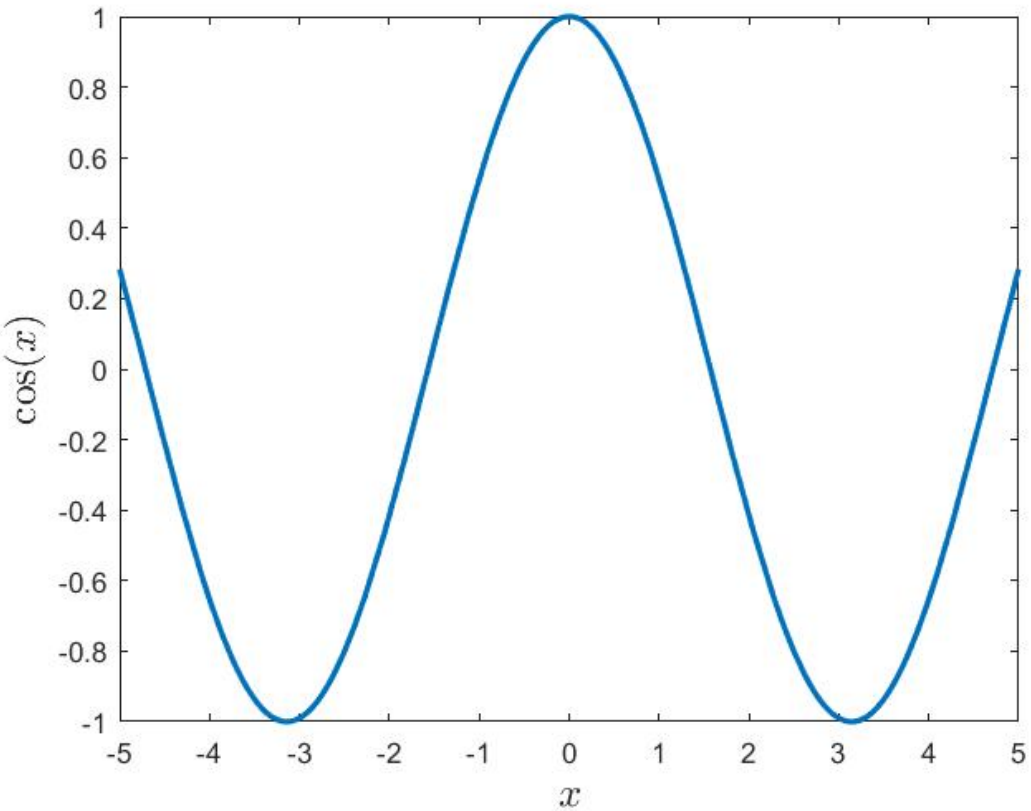


Exercise: Symbolic toolbox

- syms f(x)

  f(x) = cos(x);

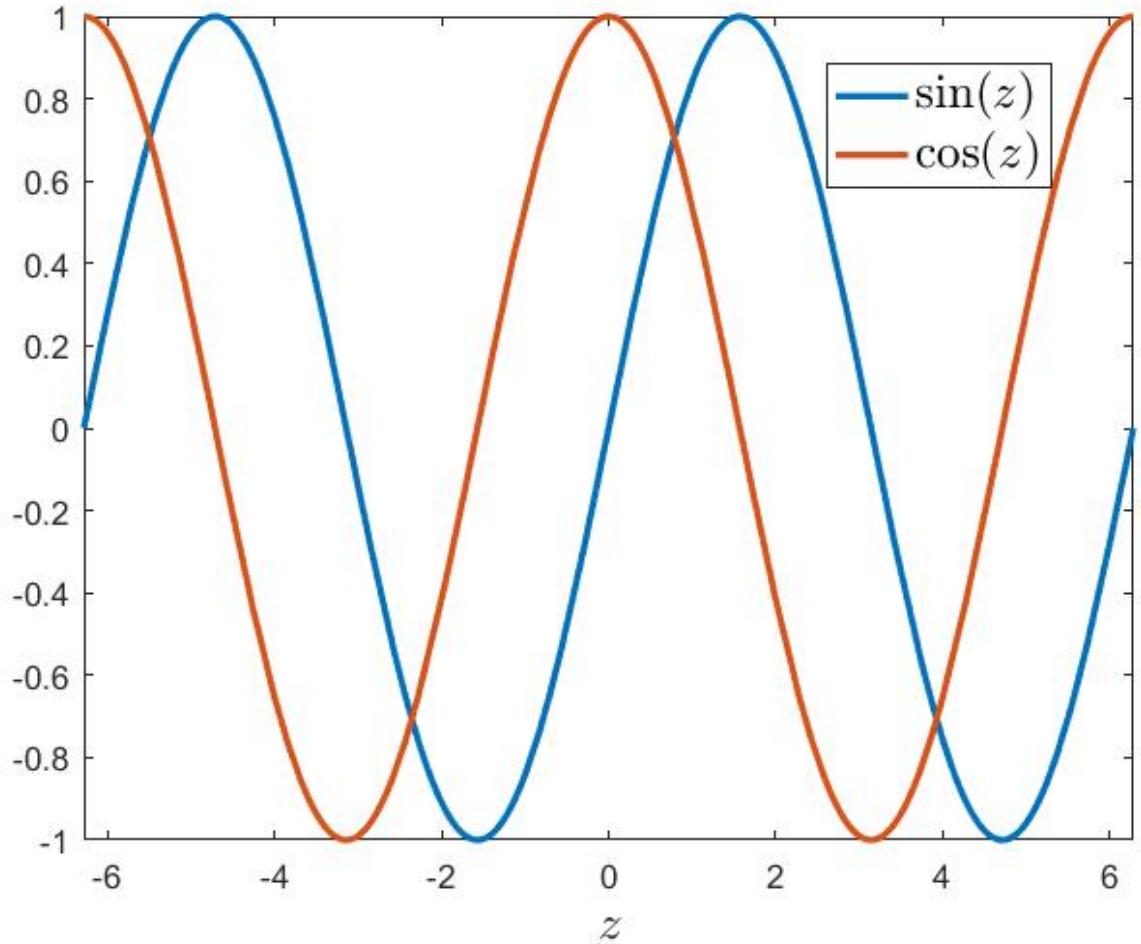
  fplot(f)



Exercise: Symbolic toolbox

- syms z

  fplot ([sin (z) , cos (z)], [−2 * pi, 2 * pi])



Exercise: Symbolic toolbox

- syms t

```
xt = exp(-t / 10) .* sin(5 * t);

yt = exp(-t / 10) .* cos(5 * t);

fplot3(xt, yt, t, [-10,10])
```

Exercise: Symbolic toolbox

- syms t

  xt = exp(−t / 10) .* sin(5 * t);

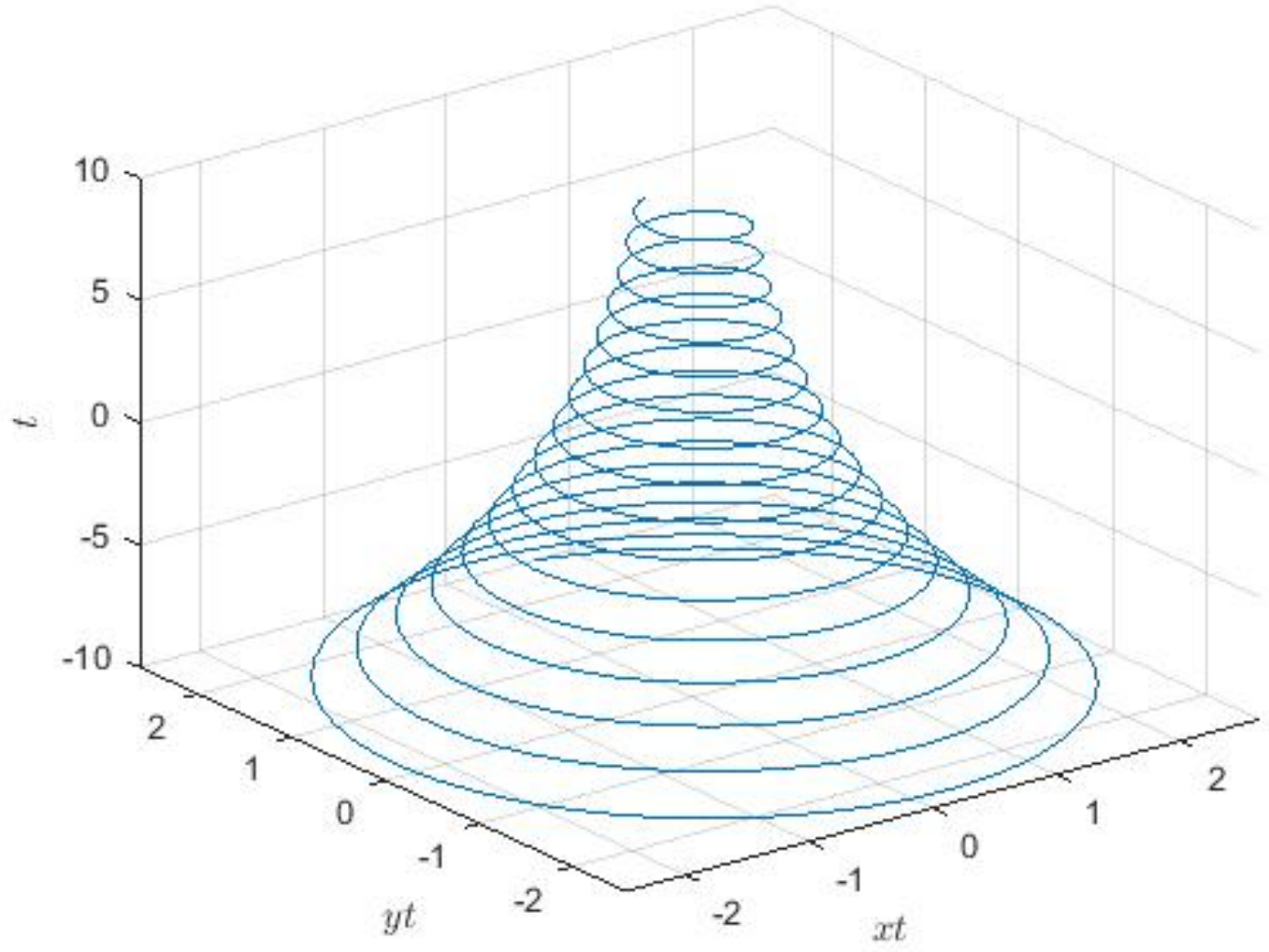
  yt = exp(−t / 10) .* cos(5 * t);

  fplot3(xt, yt, t, [−10, 10])

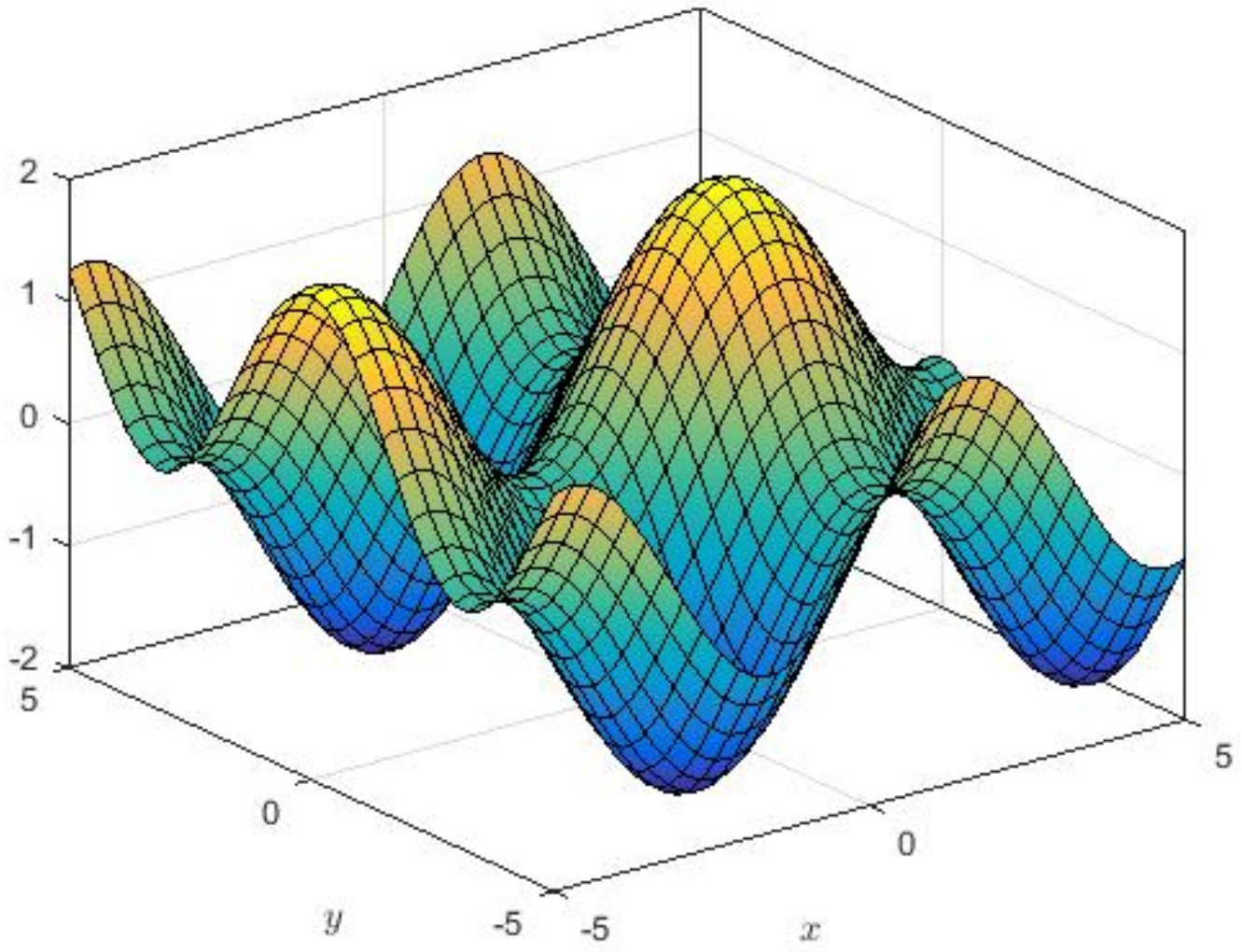
- syms x y

  fsurf(sin(x) + cos(y))

Exercise: Symbolic toolbox



Exercise: Symbolic toolbox



Exercise: User-defined Functions

Script file

- Defining a column vector of random integers

```
my_vec = randi (100, 50, 1);
```

Exercise: User-defined Functions

Script file

- Defining a column vector of random integers

```
my_vec = randi (100, 50, 1);
```

- Calling user-defined function file "my_fun"

```
[a, b, c] = my_fun (my_vec);
```

Exercise: User-defined Functions

Script file

- Defining a column vector of random integers

```
my_vec = randi (100, 50, 1);
```

- Calling user-defined function file "my_fun"

```
[a, b, c] = my_fun (my_vec);
```

- Displaying the results

```
fprintf ('Mean = %6.4f \n', a)
fprintf('Mode = %6.0f \n', b)
fprintf('Median = %6.4f \n', c)
```

Exercise: User-defined Functions

Function file

- Defining the function "my_fun"

```
function [my_mean, my_mode, my_median] =
my_fun (A)

my_mean = mean(A);

my_mode = mode(A);

my_median = median(A);
```

Exercise: User-defined Functions

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% This program takes a vector of random integers as input, and returns
% the mean, mode and median as output.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear variables

close all

clc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Defining a column vector of random integers

my_vec = randi(100,51,1);
```

Exercise: User-defined Functions

```
% Calling user-defined function file "my_fun"

[a,b,c] = my_fun(my_vec);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Displaying the results

fprintf('Mean = %6.4f\n',a)

fprintf('Mode = %6.0f\n',b)

fprintf('Median = %6.4f\n',c)
```

Exercise: User-defined Functions

```
% This is the function file for my_fun_main

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Defining the function

function [my_mean,my_mode,my_median] =  my_fun(A)

my_mean = mean(A);

my_mode = mode(A);

my_median = median(A);
```

Exercise: Data export/import

- Generating the data

```
A = rand(200,5);
```

Exercise: Data export/import

- Generating the data

  A = rand(200,5);

- Exporting the data

  filename = 'mytestdata.xlsx';

  xlswrite (filename, A)

Exercise: Data export/import

- Generating the data

  A = rand(200,5);

- Exporting the data

  filename = 'mytestdata.xlsx';

  xlswrite (filename, A)

- Refreshing the workspace

  clear variables

Exercise: Data export/import

- Generating the data

  A = rand(200,5);

- Exporting the data

  filename = 'mytestdata.xlsx';

  xlswrite (filename, A)

- Refreshing the workspace

  clear variables

- Importing the data

  B = xlsread ('mytestdata.xlsx')

  data_1 = B(: , 1);

  data_2 = B(: , 2);

Exercise: Data export/import

- Plotting the data

  plot(data_1, data_2, 'k.')

References

- https://www.mathworks.com/help/matlab/matlab_prog/loop-control-statements.html
- https://www.mathworks.com/help/matlab/ref/plot.html
- https://www.mathworks.com/help/matlab/ref/plot3.html
- https://www.mathworks.com/help/matlab/ref/surf.html
- https://www.mathworks.com/help/matlab/ref/contour.html
- https://www.mathworks.com/help/matlab/creating_plots/types-of-matlab-plots.html
- https://www.mathworks.com/products/symbolic.html
- https://www.mathworks.com/help/symbolic/examples.html?s_tid=CRUX_topnav

References

- https://www.mathworks.com/help/matlab/matlab_prog/anonymous-functions.html
- https://www.mathworks.com/help/matlab/ref/function.html
- https://www.mathworks.com/help/matlab/matlab_prog/create-functions-in-files.html
- https://www.mathworks.com/help/matlab/import_export/supported-file-formats-for-import-and-export.html