Федеральное агентство связи

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

Кафедра ВС

Лабораторная работа №1

по курсу «Анализ данных»

«Работа в командном окне системы *MatLab*»

|  |
| --- |
| Выполнил:  студент гр. МГ-165  Терешков Р. В.  Проверил:  д.т.н., профессор  Рабинович Е. В. |

Новосибирск – 2017

**Цель работы**

В рамках данной лабораторной работы необходимо ознакомиться с основами работы в среде *MatLAB.* Изучить типы используемых данных, принципы работы с массивами, построения графиков, типовых конструкций языка, таких как функции, условия и циклы**.**

**Результаты выполнения заданий**

**Задание №1**

Найти и изобразить пересечение двух кривых: *y*1 *= x*2 *–* 5*x +* 3 и *y*2 *=* 7*x –* 4.

**Решение**

Для нахождения точек пересечения кривых *y*1и *y*2, воспользуемся функцией *fsolve*:

|  |
| --- |
| sx1 = fsolve('x.^2 - 5.\*x + 3 - 7.\*x + 4', 1)  sy1 = sx1^2 - 5\*sx1 + 3  sx2 = fsolve('x.^2 - 5.\*x + 3 - 7.\*x + 4', 12)  sy2 = sx2^2 - 5\*sx2 + 3 |

Полученные результаты:

|  |
| --- |
| >> task1  sx1 = 0.6148  sy1 = 0.3038  sx2 = 11.3852  sy2 = 75.6962 |

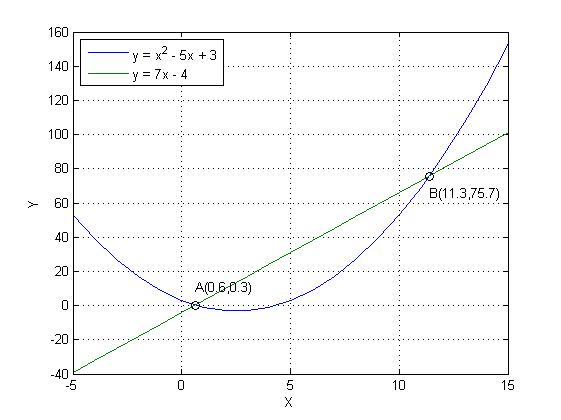
****

Рисунок 1 – График пересечения кривых

**Задание №2**

Найти и изобразить точку экстремума функции *y =* (*x+*2)(*x-*4).

**Решение**

Для нахождения экстремума функции *y*воспользуемся функцией *fminbnd*:

|  |
| --- |
| % find an extreme value of the function  [xmin, ymin] = fminbnd('(x + 2) .\* (x - 4)', -4, 6) |

Полученные результаты:

|  |
| --- |
| >> task2  xmin = 1  ymin = -9 |

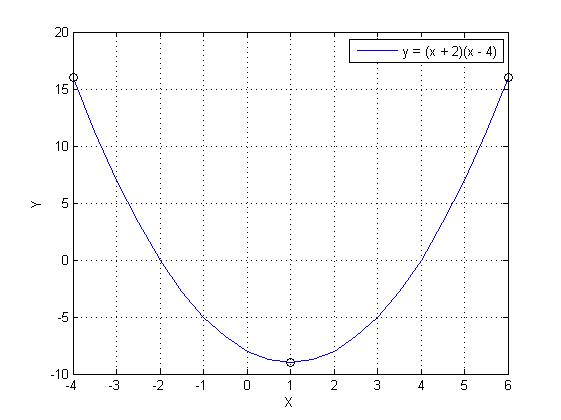
****

Рисунок 2 – Точка экстремума на отрезке [-4; 6]

**Задание №3**

Найти и изобразить координаты оснований перпендикуляров, опущенных из точки *B* на стороны *AC* и *A*1*С* треугольников *ABC* и *A*1*BC.* Координаты вершин: *A*(-1;-2), *A*1(2; 2), *B*(-1;4), *C*(5;4).

**Решение**

Для построения треугольников и нахождения перпендикуляров используется функция *makeTriangle*, листинг которой приведён в Приложении.

|  |
| --- |
| % ABC = [(-1,-2), (-1,4), (5,4)]  makeTriangle(-1, -2, -1, 4, 5, 4, 'tr1');  pause;  % A1BC = [(2,2), (-1,4), (5,4)]  makeTriangle(2, 2, -1, 4, 5, 4, 'tr2'); |

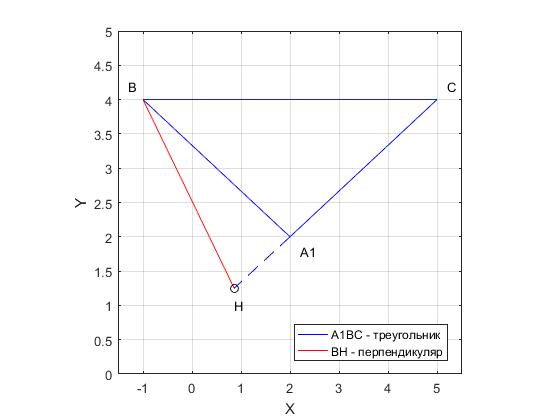
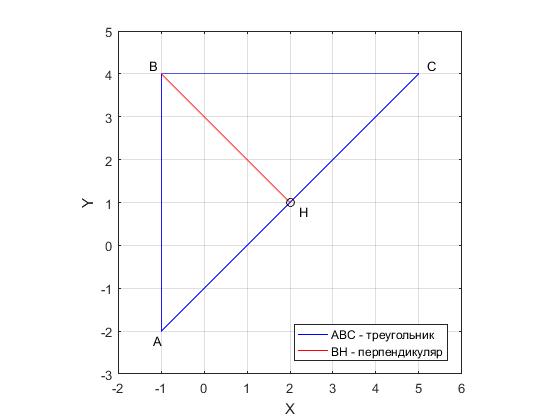


Рисунок 3 – Результат построения перпендикуляра для треугольников *ABC* и *A*1*BC*

**Задание №4**

Найти и изобразить основание перпендикуляра, опущенного из точки *A*(3;10) на окружность радиуса 4, с центром в начале координат.

**Решение**

Для построения окружности используется функция *makeCircle* (листинг приведён в Приложении). Чтобы найти точку пересечения перпендикуляра и окружности, используем формулу нахождения расстояния между двумя точками:

|  |
| --- |
| % center - (0,0), radius - 4  makeCircle(0, 0, 4);  …  x1 = 0; x2 = 3; x3 = 3;  y1 = 0; y2 = 0; y3 = 10;  % find the distance between A and O  d1 = sqrt(((x1 - x2)^2) + ((y1 - y2)^2));  d2 = sqrt(((x2 - x3)^2) + ((y2 - y3)^2));  d3 = sqrt((d1^2) + (d2^2));  d4 = 4;    % find the intersection point H  px = d4 \* d1 / d3;  py = d4 \* d2 / d3; |

Полученные результаты:

|  |
| --- |
| >> task4  px = 1.1494  py = 3.813 |

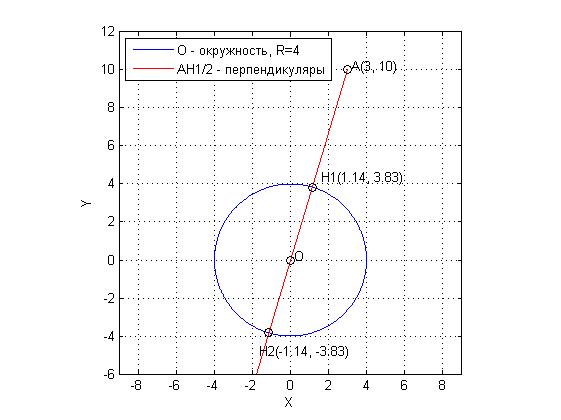


Рисунок 4 – Результат построения основания перпендикуляра к окружности

**Задание №5**

Найти и изобразить на графике функции *y =* (5*+x*)(5*-x*) точки, касательная в которых имеет угол наклона.

**Решение**

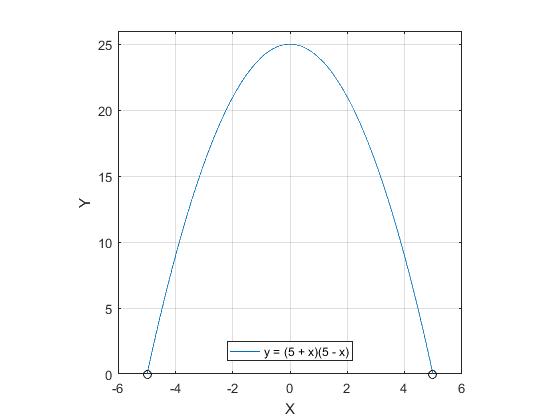


Рисунок 5 – Точки, касательная в которых не имеет угол наклона: (-5;0) и (5;0)

**Задание №6**

Написать программу (*m-файл*), которая строит график прозрачной сферы или квадратной пирамиды по выбору пользователя. Выбор фигуры и длины запрашиваются у пользователя в процессе работы функции:

для сферы – это идентификатор и радиус – *‘сфера’*, *R*.

для пирамиды – это идентификатор, высота, длина стороны основания – *‘пирамида’, h, l.*

**Решение**

Для построения сферы используется функция *sphere*. Функция *cylinder* позволяет построить основание пирамиды. Функция *surf* предназначена для построения трёхмерного графика:

|  |
| --- |
| % goes for a sphere  …  [X,Y,Z] = sphere(30);  X = X\*R;  Y = Y\*R;  Z = Z\*R;  surf(X, Y, Z, 'FaceAlpha', 0.1);  % goes for a pyramid  …  [X,Y,Z] = cylinder([l 0], 4);  Z = Z\*h;  surf(X, Y, Z, 'FaceAlpha', 0.1); |

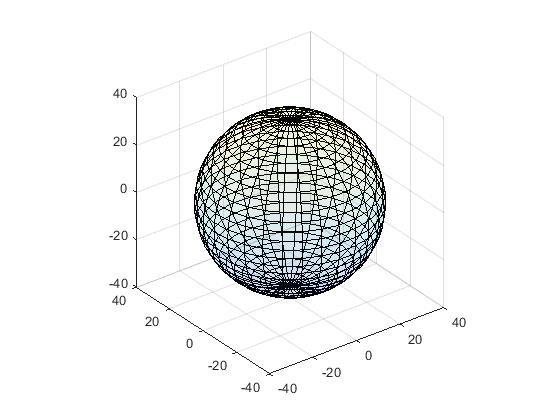


Рисунок 6.1 – Результат построения сферы с радиусом 30

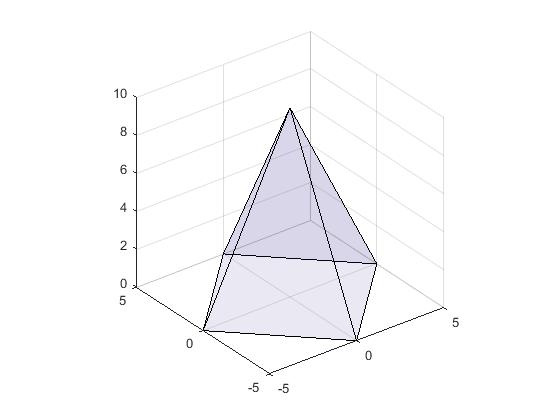


Рисунок 6.2 – Результат построения пирамиды высотой 10 и основанием 5

**Задание №7**

Написать программу (*m-файл*), которая строит график прозрачного куба или конуса. Выбор фигуры задается входными параметрами функции:

для куба – это идентификатор и длина ребра – *‘куб’, r.*

для конуса – это идентификатор, высота, радиус основания – *‘конус’, h, r.*

**Решение**

Для того чтобы построить куб, сперва зададим вершины (*vert*) для единичного куба. Затем определим, какие вершины нужно соединить для получения сторон (*face*) куба. Наконец соединим их при помощи функции *patch*. Конус строится по тому же принципу, что и пирамида:

|  |
| --- |
| % goes for a cube  …  % init vertices  vert = [-1 -1 -1; -1 1 -1; 1 1 -1; 1 -1 -1;  -1 -1 1; -1 1 1; 1 1 1; 1 -1 1];  % init faces  face = [1 2 3 4; 2 6 7 3; 4 3 7 8;  1 5 8 4; 1 2 6 5; 5 6 7 8];    % patch them  patch('Faces', face, 'Vertices', vert \* r, 'FaceAlpha', 0.1);  grid on;  axis('square');  view(3);  % this goes for a cone  …  [X,Y,Z] = cylinder([r 0]);  Z = Z\*h;  surf(X, Y, Z, 'FaceAlpha', 0.1) |

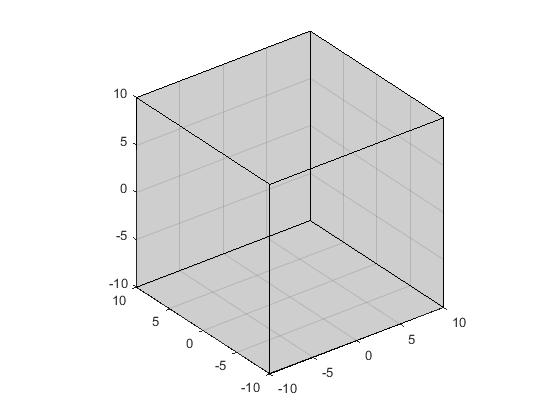


Рисунок 7.1 – Результат построения куба с длиной ребра *r*=10

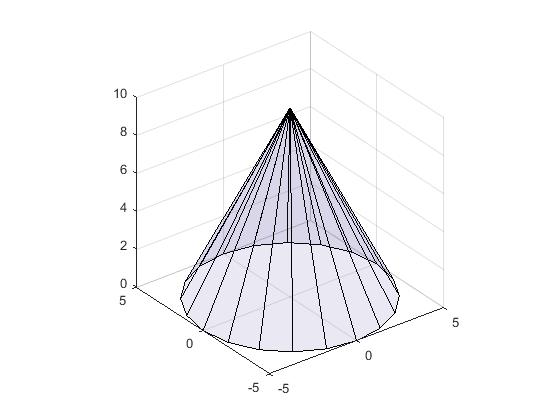


Рисунок 7.2 – Результат построения конуса высотой 10 и радиусом основания 5

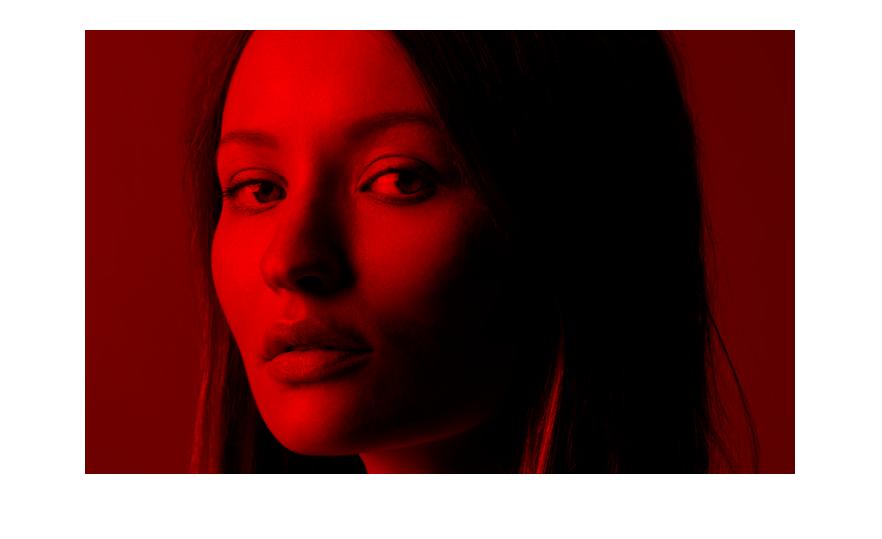
**Задание №8**

Написать программу (*m-файл*), которая раскрашивает черно-белый рисунок красным, зеленым или синим цветом. Цвет зависит от входного параметра: *id = 'красный'; id = 'зеленый'; id = 'синий'.*

**Решение**



Рисунок 8.1 – Исходное изображение



Рисунки 8.2-8.4 – Результат выполнения при id=’r’/’g’/’b’

**Задание №9**

Написать программу (*m-файл*), которая инвертирует цвета черно-белого изображения и поворачивает его на угол, кратный 90°. Угол поворота зависит от входного параметра: *id* = 90 – поворот на 90°; *id* = 180 – поворот на 180°; *id* = 270 – поворот на 270°; *id* = 0 – нет поворота.

**Решение**







Рисунок 9.1-9.4 – Результаты инвертирования цвета изображения и поворота его на все допустимые углы

**Выводы:**

В ходе выполнения данной лабораторной работы я ознакомился со средой разработки *MatLAB.* Изучил типы используемых данных, проделал работу с массивами, с построением графиков, с типовыми конструкциями языка, такими как функции, условия и циклы

**Приложение**

**Листинг 1**

% init x & ys

x = -5:1:15;

y1 = x.^2 - 5.\*x + 3;

y2 = 7.\*x - 4;

% solve the equations

sx1 = fsolve('x.^2 - 5.\*x + 3 - 7.\*x + 4', 1, optimset('Display', 'off'))

sy1 = sx1^2 - 5\*sx1 + 3

sx2 = fsolve('x.^2 - 5.\*x + 3 - 7.\*x + 4', 12, optimset('Display', 'off'))

sy2 = sx2^2 - 5\*sx2 + 3

% create graph with stuff

plot(x, y1, x, y2, sx1, sy1, 'ok', sx2, sy2, 'ok');

legend('y = x^2 - 5x + 3','y = 7x - 4', 'Location', 'northwest', 'Orientation', 'vertical');

xlabel('X');

ylabel('Y');

grid;

**Листинг 2**

% init x & y

x = -4:0.5:6;

y = (x + 2).\*(x - 4);

% find an extreme value of the function

[xmin, ymin] = fminbnd('(x + 2) .\* (x - 4)', -4, 6)

% create graph with stuff

plot(x, y, xmin, ymin, 'ok');

legend('y = (x + 2)(x - 4)', 'Location', 'northeast', 'Orientation', 'vertical');

xlabel('X');

ylabel('Y');

grid;

**Листинг 3**

% ABC = [(-1,-2), (-1,4), (5,4)]

makeTriangle(-1, -2, -1, 4, 5, 4, 'tr1');

pause;

% A1BC = [(2,2), (-1,4), (5,4)]

makeTriangle(2, 2, -1, 4, 5, 4, 'tr2');

% creates & display the triangle

function makeTriangle(a1, a2, b1, b2, c1, c2, name)

% init x and y

x = [a1 b1 c1 a1];

y = [a2 b2 c2 a2];

if name == 'tr1'

% create graph with stuff

subplot(2,0,1);

plot(x, y, 'b');

axis([a1-1 c1+1 a2-1 c2+1]);

axis('square');

hold on;

% draw the vertical

plot([-1 2], [4 1], 'r', 2, 1, 'ok')

text(2.2, 0.8, 'H');

legend('ABC - треугольник', 'BH - перпендикуляр', 'Location', 'southeast', 'Orientation', 'vertical');

text(a1-0.2, a2-0.2, 'A');

text(b1-0.3, b2+0.2, 'B');

text(c1+0.2, c2+0.2, 'C');

xlabel('X');

ylabel('Y');

grid on;

elseif name == 'tr2'

% create graph with stuff

subplot(2,0,2);

plot(x, y, 'b');

axis([b1-0.5 c1+0.5 a2-2 c2+1]);

axis('square');

hold on;

% draw the vertical

plot([-1 0.86], [4 1.25], 'r', 0.86, 1.25, 'ok')

text(0.86, 1, 'H');

% draw the imaginary line

plot([a1 0.86], [a2 1.25], '--b');

legend('A1BC - треугольник', 'BH - перпендикуляр', 'Location', 'southeast', 'Orientation', 'vertical');

text(a1+0.2, a2-0.2, 'A1');

text(b1-0.3, b2+0.2, 'B');

text(c1+0.2, c2+0.2, 'C');

xlabel('X');

ylabel('Y');

grid on;

else

disp('Nothing to display');

end

end

**Листинг 4**

% init A(3;10)

Ax = [-3 0 3];

Ay = [-10 0 10];

% center - (0,0), radius - 4

makeCircle(0, 0, 4);

% draw the vertical

plot(Ax, Ay, 'r');

text(3.2, 10.2, 'A');

x1 = 0; x2 = 3; x3 = 3;

y1 = 0; y2 = 0; y3 = 10;

% find the distance between A and O

d1 = sqrt(((x1 - x2)^2) + ((y1 - y2)^2));

d2 = sqrt(((x2 - x3)^2) + ((y2 - y3)^2));

d3 = sqrt((d1^2) + (d2^2));

d4 = 4;

% find the intersection point H

px = d4 \* d1 / d3

py = d4 \* d2 / d3

plot(px, py, 'ok');

text(px+0.5, py+0.5, 'H');

legend('O - окружность, R=4', 'AH - перпендикуляр', 'Location', 'northwest', 'Orientation', 'vertical');

xlabel('X');

ylabel('Y');

grid on;

% creates the circle

function makeCircle(x0, y0, r)

% phi is the angle

phi = 0:0.01:2\*pi;

% parametric equation

xp = r \* cos(phi);

yp = r \* sin(phi);

% draw graph

plot(x0 + xp, y0 + yp, 'b');

axis([-9 9, -6 12]);

axis('square');

text(x0+0.2, y0+0.2, 'O');

grid on;

hold on;

end

**Листинг 5**

%init x and y

x = -5:0.01:5;

y = (5 + x).\*(5 - x);

% draw graph

plot(x, y);

axis([-6 6 0 26]);

axis('square');

grid on;

hold on;

% mark points

plot(-5, 0, 'ok', 5, 0, 'ok');

legend('y = (5 + x)(5 - x)', 'Location', 'south', 'Orientation', 'vertical');

xlabel('X');

ylabel('Y');

grid on;

**Листинг 6**

% id - choice

id = input('Choose figure - sphere(s) or pyramid(p): ', 's');

% goes for a sphere

if strcmp(id, 's')

R = input('Input radius: ');

[X,Y,Z] = sphere(30);

X = X\*R;

Y = Y\*R;

Z = Z\*R;

surf(X, Y, Z, 'FaceAlpha', 0.1);

axis('square');

% and this one for a pyramid

elseif strcmp(id, 'p')

h = input('Input height: ');

l = input('Input length: ');

[X,Y,Z] = cylinder([l 0], 4);

Z = Z\*h;

surf(X, Y, Z, 'FaceAlpha', 0.1);

axis('square');

end

**Листинг 7**

% choose figure

id = input('Choose figure cube(q) or cone(c): ', 's');

% this goes for a cube

if strcmp(id, 'q')

r = input('Input r: ');

% init vertices

vert = [-1 -1 -1; -1 1 -1; 1 1 -1; 1 -1 -1;

-1 -1 1; -1 1 1; 1 1 1; 1 -1 1];

% init faces

face = [1 2 3 4; 2 6 7 3; 4 3 7 8;

1 5 8 4; 1 2 6 5; 5 6 7 8];

% patch them

patch('Faces', face, 'Vertices', vert \* r, 'FaceAlpha', 0.1);

grid on;

axis('square');

view(3);

% this goes for a cone

elseif strcmp(id, 'c')

h = input('Input height: ');

r = input('Input length: ');

[X,Y,Z] = cylinder([r 0]);

Z = Z\*h;

surf(X, Y, Z, 'FaceAlpha', 0.1)

axis('square')

end

**Листинг 8**

% read an image and put it into Img[] array

Img = imread('emily.jpg');

% convert an rgb image into the gray scale image

Img = rgb2gray(Img);

% create 3 copies of the current image (for each channel)

Img = repmat(double(Img)./255, [1 1 3]);

color = input('Input color (r/g/b): ', 's');

% zero those channels which we don't need to show

if (strcmp(color, 'r'))

Img(:,:,2) = 0;

Img(:,:,3) = 0;

elseif (strcmp(color, 'g'))

Img(:,:,1) = 0;

Img(:,:,3) = 0;

elseif (strcmp(color, 'b'))

Img(:,:,1) = 0;

Img(:,:,2) = 0;

end

% display an image

imshow(Img);

**Листинг 9**

% id is the angle

id = input('Input an angle (90, 180, 270, 0): ');

% read the image

Img = imread('emily.jpg');

% rotate the image

if (id == 90) || (id == 180) || (id == 270)

rImg = imcomplement(Img);

imshow(imrotate(rImg, id))

elseif (id == 0)

rImg = imcomplement(Img);

imshow(rImg)

end