Problem set 2a: M-file Programming

Name: Stanciu Iulia-Cristina

MATLAB release used: R2020a

Collaboration Info:

- https://ch.mathworks.com/help/matlab/ref/varargin.html
- https://ch.mathworks.com/help/matlab/ref/switch.html

Exercise 1. Generation of random points.

Explanation:

The function uses the following functions:

- nargin for the number of arguments passed.
- varargin for working with different types of arguments passed.
- switch – one for the number of arguments and one for the type of distribution (uniform or gaussian); If the number of arguments is different than 1, 2 or 4, the function gives an error. If the type of distribution is different from the mentioned, the program gives another error.
- If cases to verify the conditions for the coordinates interval and deviations.
- rand to generate random values for x and y when the distribution is uniform.
- randn to generate random values for x and y when the distribution is gaussian

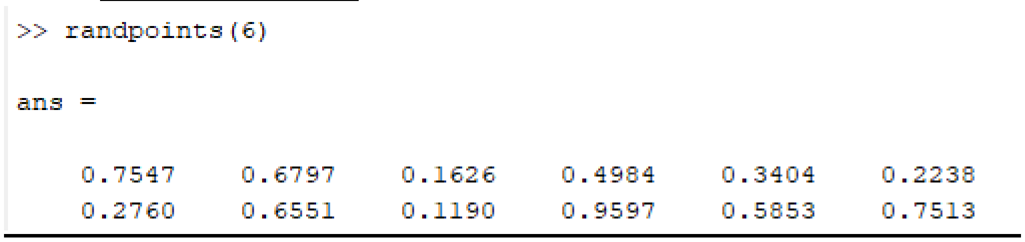
The output is P, a 2*N matrix containing the x and y coordinates of the points.

MATLAB Code:

```
function P = randpoints(N,varargin)

%disp ("Input arguments passed: " + nargin)
%celldisp (varargin)

switch nargin
    case 1
        P = rand(2,N);

    case 2
        switch varargin{1}
            case 'uniform'
                P = rand(2,N);
            case 'gaussian'
                P = randn(2,N);
            otherwise
                error('The points cannot be generated for this type of 
distribution.');
        end

    case 4
        switch varargin{1}
            case 'uniform'
                U = varargin{2};
                V = varargin{3};

                if(U(2)<U(1))
                    error('A1 must be less than B1.');
                elseif(V(2)<V(1))
                    error('A2 must be less than B2.');
                else
                    P(1,:) = rand(1,N)*(U(2)-U(1))+U(1);
                    P(2,:) = rand(1,N)*(V(2)-V(1))+V(1);
                end
            case 'gaussian'
                U = varargin{2};
                V = varargin{3};

                if(V(1)<0 || V(2)<0 )
                    error('Deviation cannot be negative.');
                else
                    P(1,:) = randn(1,N)*V(1)+U(1);
                    P(2,:) = randn(1,N)*V(2)+U(2);
                end
            otherwise
                error('The points cannot be generated for this type of
distribution.');
        end

    otherwise
        error('The number of parameters can not be supported.');
end

end
```

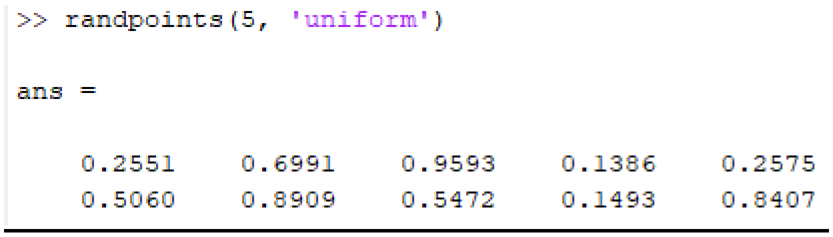
Results:

1. **P = randpoints(N);**



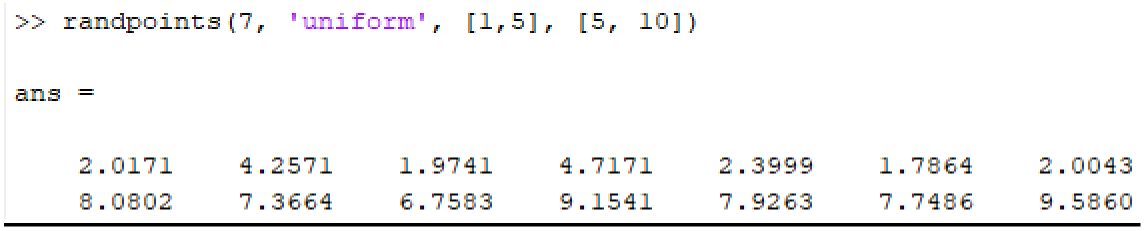
```
>> randpoints(6)

ans =

    0.7547    0.6797    0.1626    0.4984    0.3404    0.2238
    0.2760    0.6551    0.1190    0.9597    0.5853    0.7513
```

2. **P = randpoints(N,'uniform');**



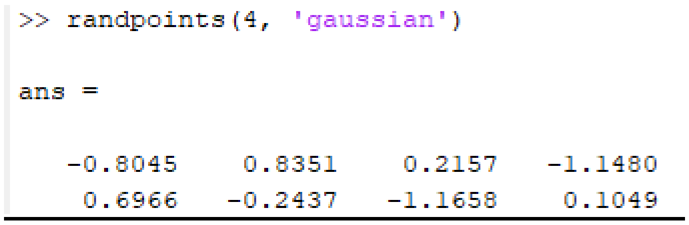
```
>> randpoints(5, 'uniform')

ans =

    0.2551    0.6991    0.9593    0.1386    0.2575
    0.5060    0.8909    0.5472    0.1493    0.8407
```

3. **P = randpoints(N,'uniform',[A1,B1],[A2,B2]);**



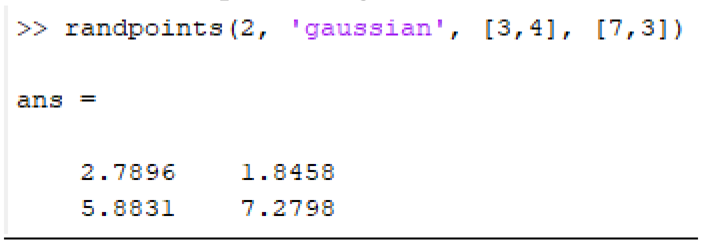
```
>> randpoints(7, 'uniform', [1,5], [5, 10])

ans =

    2.0171    4.2571    1.9741    4.7171    2.3999    1.7864    2.0043
    8.0802    7.3664    6.7583    9.1541    7.9263    7.7486    9.5860
```

4. **P = randpoints(N,’gaussian’);**



```
>> randpoints(4, 'gaussian')

ans =

   -0.8045    0.8351    0.2157   -1.1480
    0.6966   -0.2437   -1.1658    0.1049
```

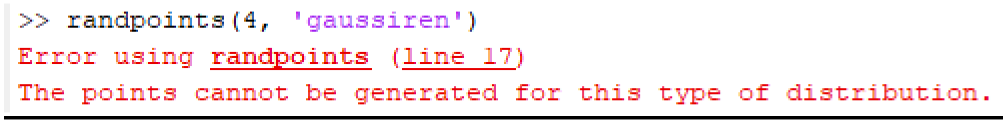
5. **P = randpoints(N,’gaussian’,[C1,C2],[Std1,Std2]);**



```
>> randpoints(2, 'gaussian', [3,4], [7,3])

ans =

    2.7896    1.8458
    5.8831    7.2798
```

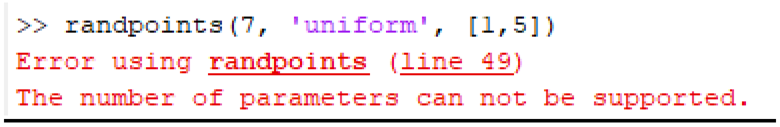
ERROR CASES:

6. **Wrong distribution name**



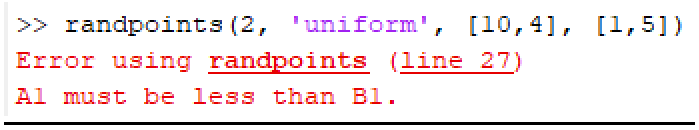
```
>> randpoints(4, 'gaussiren')
Error using randpoints (line 17)
The points cannot be generated for this type of distribution.
```

7. **Insufficient/unsupported number of arguments**

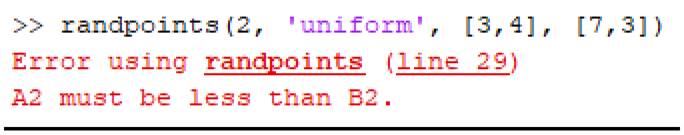


```
>> randpoints(7, 'uniform', [1,5])
Error using randpoints (line 49)
The number of parameters can not be supported.
```

8. **Wrong interval order**

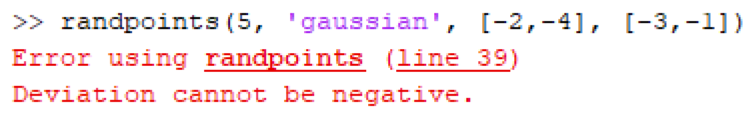


```
>> randpoints(2, 'uniform', [10,4], [1,5])
Error using randpoints (line 27)
A1 must be less than B1.
```



```
>> randpoints(2, 'uniform', [3,4], [7,3])
Error using randpoints (line 29)
A2 must be less than B2.
```

9. **Negative deviation**



```
>> randpoints(5, 'gaussian', [-2,-4], [-3,-1])
Error using randpoints (line 39)
Deviation cannot be negative.
```

Exercise 2. Convex Hull.

Explanation:

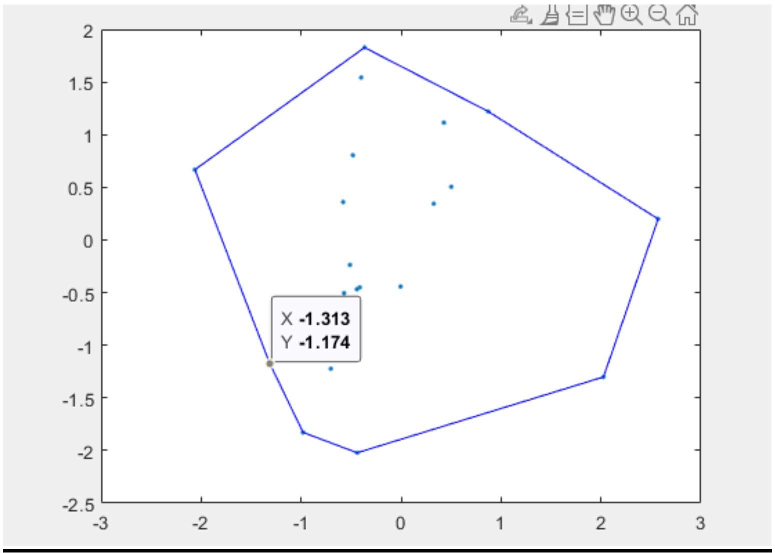
The code is mostly explained in the code comments. The function receives the x and y coordinates of several points. Firstly, I found the most extreme point to start drawing the convex polygon (maximum value for x and maximum value for y out of the points discovered with the previous condition). I used the function atan2d to calculate the angles in degrees and, as recommended, the angles are being stored in a vector. For the minimum value, we select and add the point to the output and then repeat the process. Sort was used for temporarily sorting the angle vector. The lines are drawn on top of the dot plot.

MATLAB Code:

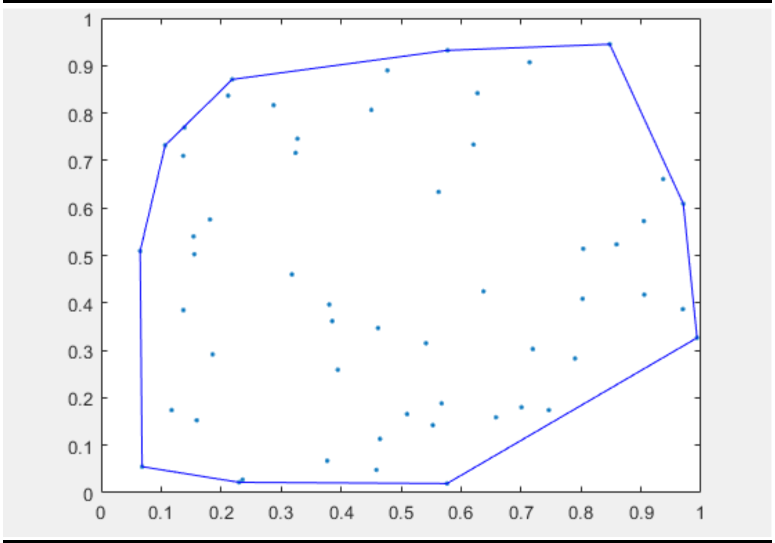
```
function H = convexhull(P)

plot(P(1,:),P(2,:),".") %plotting the points
hold on

X = P(1,:); %x coordinates
Y = P(2,:); %y coordinates

Imax = find(X==max(X)); %finding the indexes of the points with max value for
x coordonate
[~,J] = max(P(2,Imax)); %finding the index of the point with the maximum
value of y out of the ones with max value of x

FirstIndex = Imax(J); %index of the starting point P0(Xmax, Y0)
Index = FirstIndex;
H = Index; %initializing the convex hull


X0 = X(Index);
Y0 = Y(Index);

Xreference = X0;
Yreference = Y0+1;

Angle = atan2d(1,0);
Angle2 = atan2d(Y-Y0,X-X0);


ReferenceAngle = 0;
disp(ReferenceAngle)

DifferenceAngle = Angle2-Angle-ReferenceAngle;
DifferenceAngle = mod(DifferenceAngle,360);
DifferenceAngle(Index) = 360; %setting the angle between the vector of the
point and itself to 2*pi instead of 0, to be able to find the minimum angle
```

```matlab
DifferenceAngleMIN = min(DifferenceAngle);
Imin = find(DifferenceAngle==DifferenceAngleMIN); %finding all the points
with the minimum angle to the starting point

Xmin = P(1,Imin); %x coordinates of the points at the smallest angle
Ymin = P(2,Imin); %y coordinates of the points at the smallest angle
D2 = (Xmin-X0).^2+(Ymin-Y0).^2; %square distance from the points to the
starting point P0
[~,Perm] = sort(D2);

Index = Imin(Perm);
H = [H, Index]; %adding index to the convex hull

while(H(end)~= FirstIndex)

    X0 = X(Index);
    Y0 = Y(Index);
    Angle = atan2d(1,0);
    Angle2 = atan2d((Y-Y0),(X-X0));

    ReferenceAngle = DifferenceAngleMIN;

    DifferenceAngle = Angle2;
    DifferenceAngle = DifferenceAngle-Angle-ReferenceAngle;
    DifferenceAngle = mod(DifferenceAngle,360);
    DifferenceAngle(Index) = 360; %setting the angle between the vector of
the point and itself to 2*pi instead of 0, to be able to find the minimum
angle

    DifferenceAngleMIN = min(DifferenceAngle);

    Imin = find(DifferenceAngle==DifferenceAngleMIN); %finding all the points
with the minimum angle to the starting point

    Xmin = P(1,Imin); %x coordinate of the points at the smallest angle
    Ymin = P(2,Imin); %y coordinate of the points at the smallest angle
    D2 = (Xmin-X0).^2+(Ymin-Y0).^2; %square distance from the points to the
starting point P0
    [~,Perm] = sort(D2);

    Index = Imin(Perm);
    H = [H, Index]; %adding index to the convex hull
end

plot(P(1,H),P(2,H),'b');
end
```

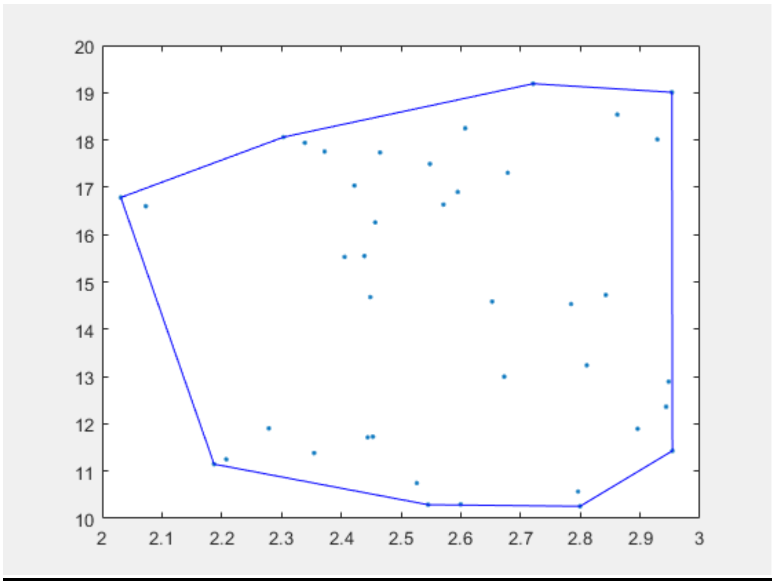
Results:



```
>> ConvexHull(randpoints(20,'gaussian'))
     0


ans =

     1    18    17     7     5    19     2    11     1
```



```
>> ConvexHull(randpoints(55))
     0


ans =

    47    10    23     8    38     9    42    17    44     7    47
```



```
>> ConvexHull(randpoints(40, 'uniform', [2,3], [10,20]))
     0


ans =

    18    36     7    13    23    24    40    16    18
```

Comments:

I didn't know how to solve this in order for the example to work.

```
Matrix dimensions must agree.

Error in convexhull (line 50)
    Angle2 = atan2d((Y-Y0),(X-X0));
```

In all the other examples using randpoints the program performed as expected.