ამოცანა 1

ა) $y(t) = v_0t - 5t^2$. $y'(t)=v_0 - 10t$ გავუტოლოთ ნულს. $v_0=10t \Rightarrow t = v_0/10 \Rightarrow ymax = v_0^2/10 - v_0^2/20 = v_0^2/20$. თუმცა, თუ თავიდან ქვემოთ გაუშვეს, ანუ $v_0<0$, მაშინ მაქსიმალური სიმაღლე 0 იქნება.

ბ)

(6.i) $\left(e^{\frac{-(x-x_0)^2-(y-y_0)^2}{2\sigma^2}}\right)' = e^{\frac{-(x-x_0)^2-(y-y_0)^2}{2\sigma^2}} \cdot$

$\cdot \frac{-2(x-x_0)}{2\sigma^2}$ გვაქვს 0-ს, $x=x_0$

(6.ii) $\left(e^{\frac{-(y-y_0)^2}{2\sigma^2}}\right)' = e^{\frac{-(y-y_0)^2}{2\sigma^2}} \cdot \frac{-2(y-y_0)}{2\sigma^2}$

(6.iv) ნიშნავს 0-ის გარშემოა $y=y_0$ და ეს ასევე ნიშნავს $(x_0;y_0)$-ის ე.წ. მნიშვნელობას და $e^0=1$

(6.i და 6.ii ფაქტიურად მიუთითებენ. წარმოებულები 6.i-ში x-ის მიმართ და $y=const$, 6.ii-ში პირიქით.)

ამოცანა 2

კოდი:

```
%reads in the image, converts it to grayscale, and converts the intensities
%from uint8 integers to doubles. (Brightness must be in 'double' format for
%computations, or else MATLAB will do integer math, which we don't want.)
dark = double(rgb2gray(imread('u2dark.png')));
%%%%%% Your part (a) code here: calculate statistics
maximum_value = max(max(dark));
minimum_value = min(min(dark));
average_value = mean(mean(dark));
%%%%%% Your part (b) code here: apply offset and scaling
fixedimg = (dark - minimum_value)*(256 / (maximum_value-minimum_value));
%displays the image
imshow(uint8(fixedimg));
%%%%%% Your part (c) code here: apply the formula to increase contrast,
% and display the image
contrasted = uint8(2*(fixedimg - 128)+128);
imshow(contrasted);
```

მინიმუმი – 25, მაქსიმუმი -153, საშუალო = 76.9622

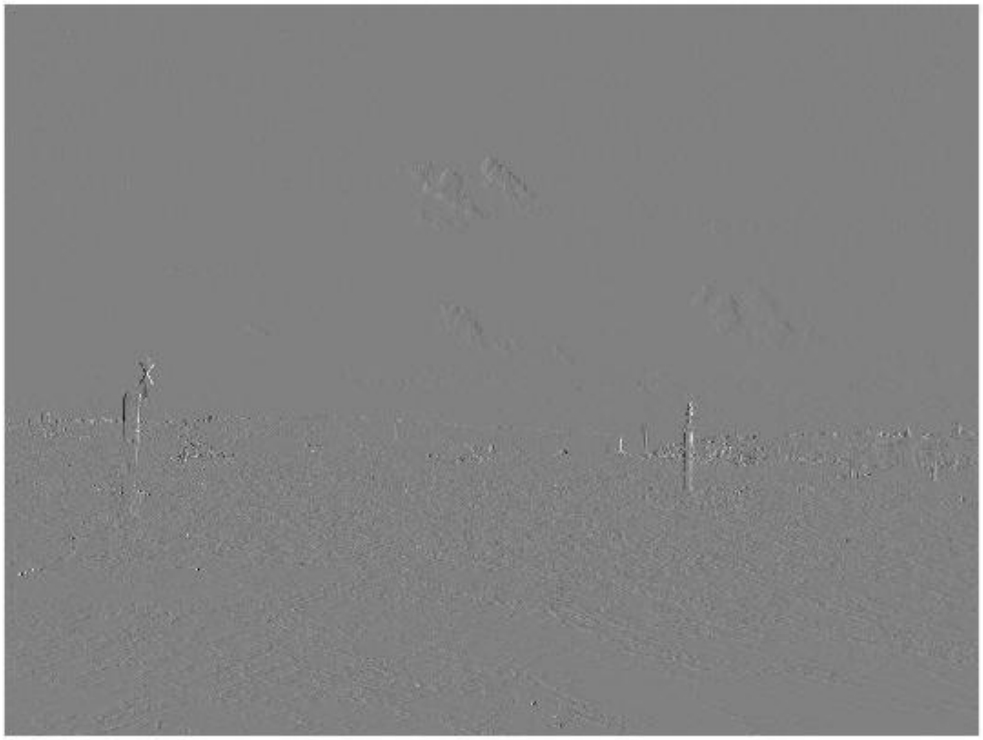
3)

ამოცანა 3

კოდი

```
% This function calls the functions below and displays their results.
% You don't need to edit it.
function edgedetector()
   img = double(rgb2gray(imread('buoys.jpg')));
   edges = DetectVerticalEdges(img);
   blurred_edges = BoxBlur(edges);

   figure('Name','Original Image')
   imshow(img, []);
   figure('Name','Edges')
   imshow(edges, []);
   figure('Name','Blurred Edges')
   imshow(blurred_edges, []);
end
% Returns a matrix containing the horizontal component of the gradient at every
% image location.
function edges = DetectVerticalEdges(img)
   % MATLAB images use matrix indices, so the order is (y,x), and the +y
   % direction is downward.
   width = size(img, 2);
   height = size(img, 1);
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
   %%%%%%%%%% Your part (a) code here. You can accomplish part (a) with
   %%%%%%%%%% nested "for" loops, but an easier way is to use matrix
   %%%%%%%%%% indexing to make a matrix of the "left" pixels and a matrix
   %%%%%%%%%% of the "right" pixels, and subtract the two matrices.
   %%%%%%%%%% REMEMBER: left/right position is the SECOND index in MATLAB.
   right = img(1:height,2:width);
   left = img(1:height,1:width-1);
   edges = right - left;
   %%%%%%%%%% End of your part (a) code.
end
% Applies a box blur to the input image and returns the result.
function blurred = BoxBlur(img)
   img = double(img);
   height = size(img, 1);
   width = size(img, 2);
   n=5; % width of the blur
```

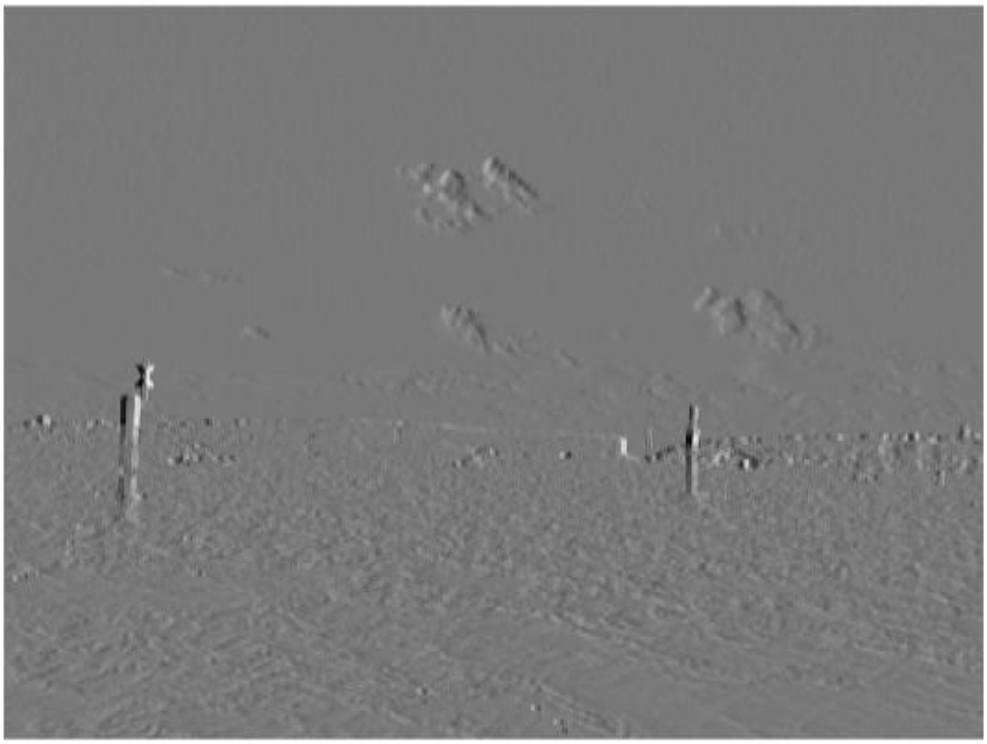
```
    blurred = zeros(height-(n-1),width-(n-1));
    % Loop through each pixel location in the result
    for y=1:height-(n-1)
        for x=1:width-(n-1)
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
            %%%%%%% Your part (b) code here. Calculate blurred(y,x).
            block_sum = 0;
            for i=0:n-1
                for j=0:n-1
                    block_sum = block_sum + img(y+i,x+j);
                end
            end
            blurred(y,x) = block_sum;
            %%%%%%% End of part (b) code
        end
    end

    % Usually we'll divide at the end so that we don't make the image
    % brighter:
    blurred = blurred / n^2;
end
```
ა)



პატარა edge-ები გამოიწვია ტალღებმა, რადგან სხვაობა თვალით გარჩევადიც მარტივად იყო.

ბ)

boxblurr-მა უკეთესად გამოაჩინა, მცირე noise გააქრო და სურათი უკეთესია.



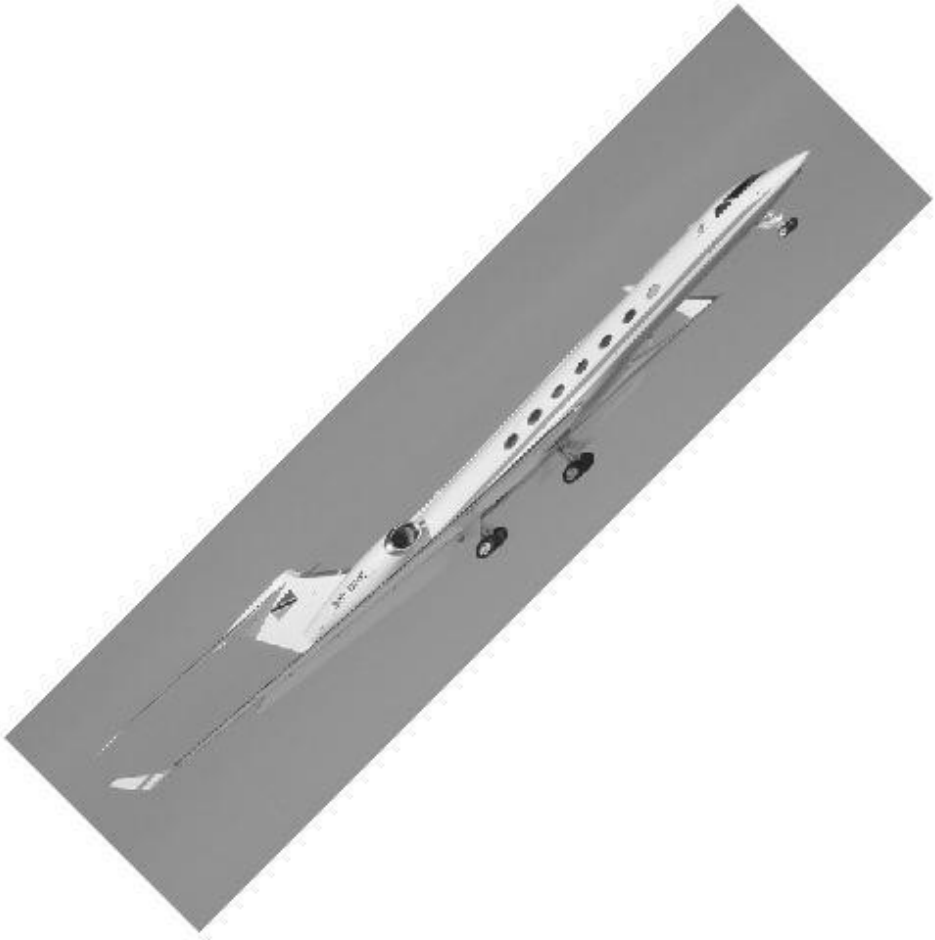
ამოცანა 4

ა) მოტრიალების მატრიცის ორჯერ გამოყენებით, უბრალოდ ორჯერ მოტრიალდება ყველაფერი 45 გრადუსით, ანუ ყველაფერი 90 გრადუსით მოტრიალდება საათის ისრის საწინააღმდეგო მიმართულებით.

ბ) scaling-ისთვის მატრიცა იქნება M = [1,0; 0, 0.5] აქედან $X_2 = AMX_0$ M გაწელავს, A კი შემდეგ მოაბრუნებს.

გ) ბ-ში დასათვლელი მატრიცა MATLAB-ით ეს გამოვიდა : [0.7071 , -0.3536;0.7071 , 0.3536]

ეს სურათი მივიღე.



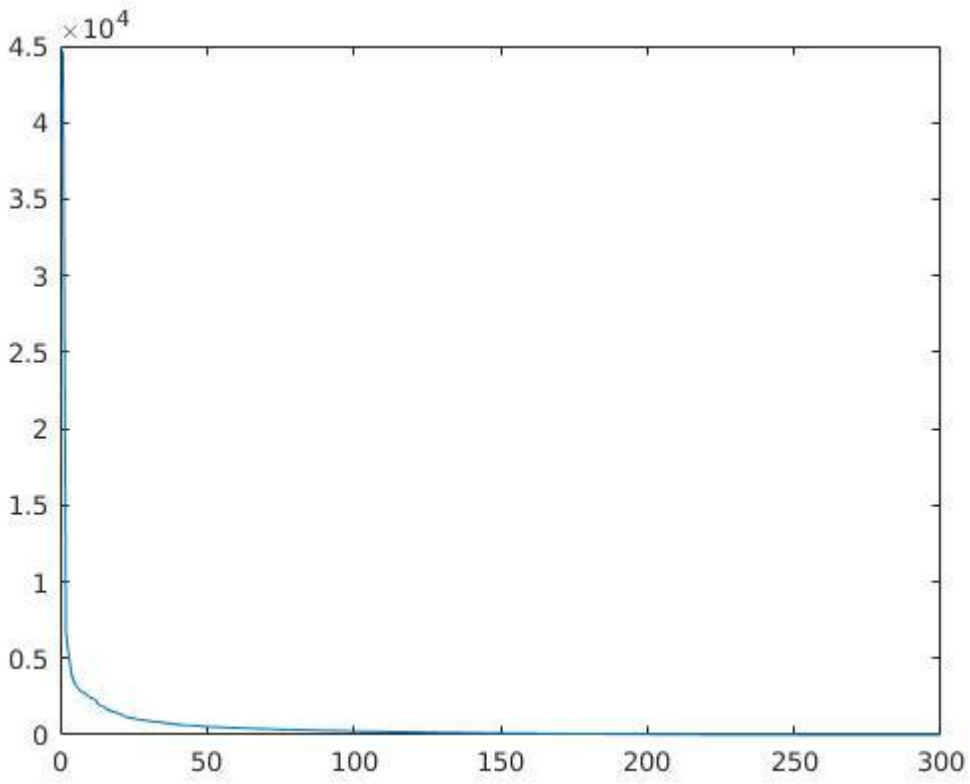
დ) T = [1,0,2; 0,1,0; 0,0,1] x-ს დაუმატებს 2ს. იმისთვის რომ 90 გრადუსით დავატრიალოთ საათის ისრის საწინააღმდეგო მხარეს, გვინდა შემდეგი მატრიცა R = [0.7071 , -0.7071, 0; 0.7071 , 0.7071, 0; 0, 0, 1] (A გადავამრავლე თავის თავზე და შემდეგ ჰომოგენურ ფორმაში ჩავწერე).

TRP = [6.9497; 4.9497; 1.0000]

RTP = [6.3639; 6.3639; 1.0000] ცხადია ერთი და იგივე არაა, რადგან გადადება და დატრიალება არაა დატრიალება და გადადება. არაკუმუტატიური ოპერაციებია.

ე)RYA=B გავამრავლოთ ორივე მხარე R^{-1} ზე მარცხნიდან და A^{-1} ზე მარჯვნიდან. რადგან R rotation მატრიცაა, მას აქვს შებრუნებული (მისი ტრანსპონირებულია) ანუ მივიღებთ IYI = $R^{-1}BA^{-1}$ სადაც I identity matrix არის. ანუ Y = $R^{-1}BA^{-1}$.

ამოცანა 5



ა)ესაა plot.

Top 10 MATLAB-ის მიხედვით

[4.467159114221457e+04;6.622972152560838e+03;5.165774038308063e+03;3.874659896642687e+03;3.328367213707513e+03;3.068852842410373e+03;2.852643613529795e+03;2.773350693374228e+03;2.651003116733963e+03;2.455272624751885e+03]

კოდი -

```
dark = double(rgb2gray(imread('flower.bmp')));
top_ten = svds(dark,10);
all_singulars = svd(dark);
rankings = 1:1:size(all_singulars,1);
plot(rankings, all_singulars);
```

ბ)

```
[U,S,V] = svd(dark);
tmp_matrix = U*S*V';
imshow(uint8(tmp_matrix));
```

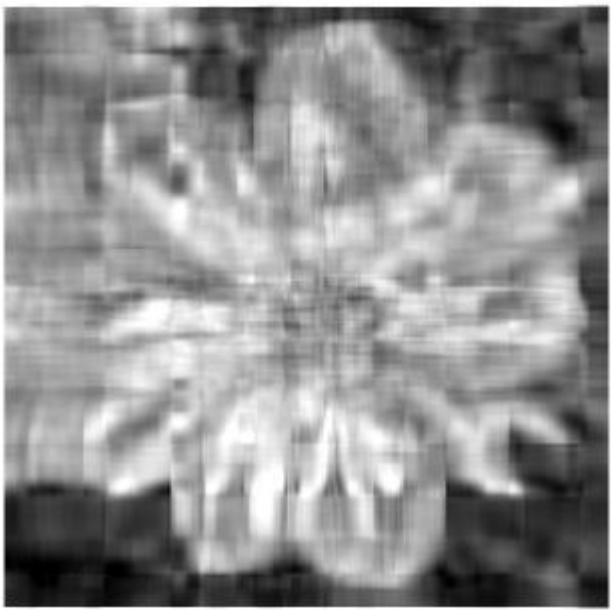
ამ კოდით დავრწმუნდი რომ აღდგენა შესაძლებელია.

```
[U,S,V] = svd(dark);
k = 10;
```

```
U_k = U(:,1:k);
V_K = V(:,1:k);
U_tmp = U;
V_tmp = V;
S_tmp = zeros(size(S));
U_tmp(:,1:k) = U_k;
V_tmp(:,1:k) = V_K;
S_tmp(1:k,1:k) = S(1:k,1:k);
tmp_matrix = U_tmp * S_tmp * V_tmp';
imshow(uint8(tmp_matrix));
```

ამ კოდით ვნახულობთ ახალ სურათებს,მცირე მეხსიერების გამოყენებით.

K=10



k=50

k=100

ცხადია რაც მეტია K მით უკეთესად ინახება სურათი. მეტი ინფორმაცია = უკეთეს სურათს

გ) სურათი 300X300 არის. K=200-სთვის ვხარჯავთ 200*300+200+200*300>300*300. ანუ არ ვზოგავთ ამით მეხსიერებას.

დ)

5

(0) SVD-ით, $A = \sum_{i}^{r} u_i \cdot \sigma_i \cdot v_i^T$

სადაც u_i და v_i შესაბამისად სვეტებია შესაბამისად SVD დაშლის მატრიცებიდან, ხოლო σ_i კი სინგულარული value შესაბამისად.

ცნობილია, რომ საუკეთესო k რანგის მატრიცა ასეთი ფორმით ჩაიწერება

$$P = \sum_{i=k+1}^{r} u_i \sigma_i v_i^T$$

აქედან, ვიპოვოთ ისეთი ρ-ც, რომ შესრულდეს

$$\sum_{i=k+1}^{r} u_i \sigma_i v_i^T \leq \rho \sum_{i=k+1}^{r} \sigma_i$$

$u_i v_i^T$ მატრიცებს ერთეულოვანი ნორმა აქვთ, ხოლო σ_i-ს ნორმა

ამიტომ ეს უტოლობა ნებისმიერი ρ-სთვის სრულდება) კერძოდ. $\rho = 1$ დაიკმაყოფილებს ამ პირობას.

ე) დ-ს ფორმულის მიხედვით , უბრალოდ დიდი ალბათობით ტერნარული ძებნა k-ზე გაამართლებს, სავარაუდოდ არ იქნება მაგაზე რთული.

09:00 $\|A\|_2 = \max_x \sqrt{(Ax)^T (Ax)}$ შევიტანოთ $A = U\Sigma V^T$

10:00 და ვიმუშაოთ გამოსახულებაზე

$$\max_x \sqrt{(U\Sigma V^T x)^T \cdot U\Sigma V^T x} =$$

11:00

$$= \max_x \sqrt{x^T V \Sigma^T U^T U \Sigma V^T x} \quad U^T U = I \text{ ის}$$

12:00

$$= \max_x \sqrt{x^T V \Sigma^T \Sigma V^T x} \quad x^T V = X^T V^{++}$$ და

13:00

$$V^T x = \left((x^T V)^T\right)^T$$

14:00

$X^T V^{+T} \equiv U$-ა, ამიტომ ფორმულა, რომელიც

15:00 pdf-ში იწერება. მივიღებთ

16:00 $\max_U \sqrt{U^T \Sigma^T \Sigma U}$ ფორმულებს.

17:00 Σ და ორთოგონალური და შესაბამისად $\Sigma^T\Sigma$-ც დიაგო-

18:00 ნალური იქნება $\sigma_1^2; \sigma_2^2 \dots \sigma_n^2$-ით დალაგებული ნიშნავს.

19:00 U ისეთნაირ ორთოგონალურ, რომელიც მატრიცას

ამოგვიყრის σ^2 ვექტორებს ანუ ნიშნავს Σ-ს, ანუ

$$\|A\|_2 = \sigma_{max} \quad \text{ა.ე.დ.}$$

20:00

ამოცანა 6

a) ვიყენებთ SVD-ს.

$$A^+A = V\Sigma^+U^+U\Sigma V^+ = V\Sigma^+\Sigma V^+$$

$\Sigma^+\Sigma$ ასრე გამოიყურება σ_1^2, σ_2^2 σ_n^2-ით გამოსახული. $\Sigma^+\Sigma = Z$

ვიცით V rotation matrix ასრე,

$V^+ = V^{-1}$ და ტრანსპონირების თვისებიდან გამომდინარე

ანუ $\text{trace}(VZV^+) = \text{trace}(Z) =$

$= \sigma_1^2 + \sigma_2^2 + \dots + \sigma_n^2$ ა.დ.გ.