Image Deblurring

Toea Mariana
Grupa: 324AA

1. Descrierea aplicației

Imaginea blurată este o problemă comună în procesarea imaginilor.

Scopul este recuperarea unei imagini originale pornind de la o versiune degradată (blurată): efectul de blur fiind cauzat adesea de mișcarea camerei, defocalizare sau turbulențe atmosferice(în imagistica la distanță)

Acest lucru se poate modela ca o problemă de optimizare cu constrângeri.

Relevanță practică:

•Restaurarea fotografiilor vechi

•Îmbunătățirea imaginilor medicale

•Reconstrucție în camere low-light sau mișcare

•Aplicații militare și satelitare

Exemple imagini deblurate

2. Formularea matematică a problemei

Problema se reduce la rezolvarea:

$$\min_{x \in \mathbb{R}^{mn}} \|Dx - y\|_2^2 \quad := \text{f(x)} \rightarrow \text{funcția obiectiv(cost)}$$

s.l: $0 \leq x \leq 1$

s.l: $0 \leq x_i \leq 1$, i = 1, ..., mn

Variabila de decizie: $x \in \mathbb{R}^{mn}$

Constrângeri: $0 \leq x_i \leq 1$, i = 1, ..., mn (nr. de constrângeri: 2*mn)

, unde mn = m * n, m = nr. de linii din imaginea blurată(forma matriceală) și n = nr. de coloane, x – imaginea necunoscută(vectorizată), y – imaginea blurată(vectorizată) și D – matricea de blurare(matrice rară de mediere, de dimensiune mn x mn)

2. Formularea matematică a problemei

Funcția obiectiv se poate scrie ca:

$$f(x) = (Dx - y)^T(Dx - y) = (x^T D^T - y^T)(Dx - Y) = x^T D^T Dx - 2y^T Dx + y^T y$$

Problema abordată este o **problemă QP**(quadratic programming)

Demonstrație:

Formulă generală : $\min_x \frac{1}{2} x^T Q x + q^T x + cst$, s.l.: Ax=b, Cx≤d, x$\epsilon$[lb, ub]

Rescriu funcția obiectiv astfel: $f(x) = \frac{1}{2} x^T (2D^T D)x - (2D^T y)^T x + y^T y \rightarrow problemă\ QP$

2. Formularea matematică a problemei

Problema abordată este un **QP convex.**

Demonstrație:

gradientul: $\nabla f(x) = 2D^T(Dx - y) = Qx + q$

hessiana: $\nabla^2 f(x) = 2D^T D = Q$

Ca funcția obiectiv să fie convexă → $\nabla^2 f(x) \geq 0$

1. Simetria lui Q = $2D^T D, Q^T = (2D^T D)^T = 2D^T D \rightarrow Q = Q^T \rightarrow Q\ este\ simetrică$

2. Q = $2D^T D$ este pozitiv semidefinită, aplic definiția

2. Formularea matematică a problemei

$z^T Qz \geq 0 \; \forall \; z \epsilon \mathbb{R}^{mn}, Q = 2D^T D$

$z^T Qz = 2 \; z^T D^T Dz = 2 \; (Dz)^T Dz = 2\|Dz\|_2^2 \geq 0 \; \forall \; z \epsilon \mathbb{R}^{mn} \; \rightarrow$
$Q \; este \; pozitiv \; semidefinita (și \; simetrică) \rightarrow \nabla^2 f(x) \geq 0 \; \rightarrow funcția \; obiectiv \; este \; convexă (1)$

Constrângerile sunt liniare(de tip box) (2)

Din (1) și (2) → problema abordată este o problemă QP convexă

2. Formularea matematică a problemei

Matricea de blur, D (matrice rară, simetrică, de dimensiune mn x mn), este o matrice de mediere cu wrap around, producând un alt vector(imaginea blurată vectorizată) în care fiecare valoare este o medie a valorilor(pixelilor) vecine(vecini).

Pentru simplitate, voi da un exemplu pentru o imagine cu mn = 5 pixeli, blur = 1 (media pentru un vecin la stânga și un vecin la dreapta).

$$D = \begin{matrix} \frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 & 0 & \frac{1}{3} & \frac{1}{3} \end{matrix}$$

3. Rezultate obținute

- Pentru rezolvarea problemei am implementat Metoda Gradient Proiectat și Metoda Gradient Condițional(Frank-Wolfe) și am comparat rezultatele obținute cu cele 2 metode alese cu rezultatele returnate de funcții din MatLab(fmincon) și CVX.

3. Rezultate obținute

Algoritmul GP

Date de intrare : $\boldsymbol{x}_0\,(k=0)$ punctul initial, pasul $\alpha_k > 0$

1. Atata timp cat `criteriu`$(x_k) \geq \epsilon$:

   1.1 $\boldsymbol{x}_{k+1} = [\boldsymbol{x}_k - \alpha_k \nabla f\,(\boldsymbol{x}_k)]_\Omega$

   1.2 $k = k + 1$

2.Returneaza x_{k+1}

- **Metoda gradient proiectat**

Punctul inițial la pasul k = 0, x_0 - vectorul plin de zero-uri

Pasul $\alpha_k = \frac{c}{k+1}$, pas descrescător, asigură convergența, balans între viteză și precizie, c - constantă

Proiecția pe $\Omega = [0,1]^{mn}$ (rol de a impune constrângerile)

3. Rezultate obținute

Atât pentru MGP cât și pentru MGC, mulțimea pe care proiectez Ω trebuie să fie o mulțime simplă, convexă și compactă (închisă și marginită).

$$\Omega = [0,1]^{mn}$$

1. Este o mulțime simplă – proiecția pe ea se face ușor

$$P_{\Omega}(z)_i = \min(\max(z_i, 0), 1)$$

2. Este o mulțime compactă

- Este închisă(include marginile 0 și 1)

- Este marginită(fiecare componentă este între 0 și 1)

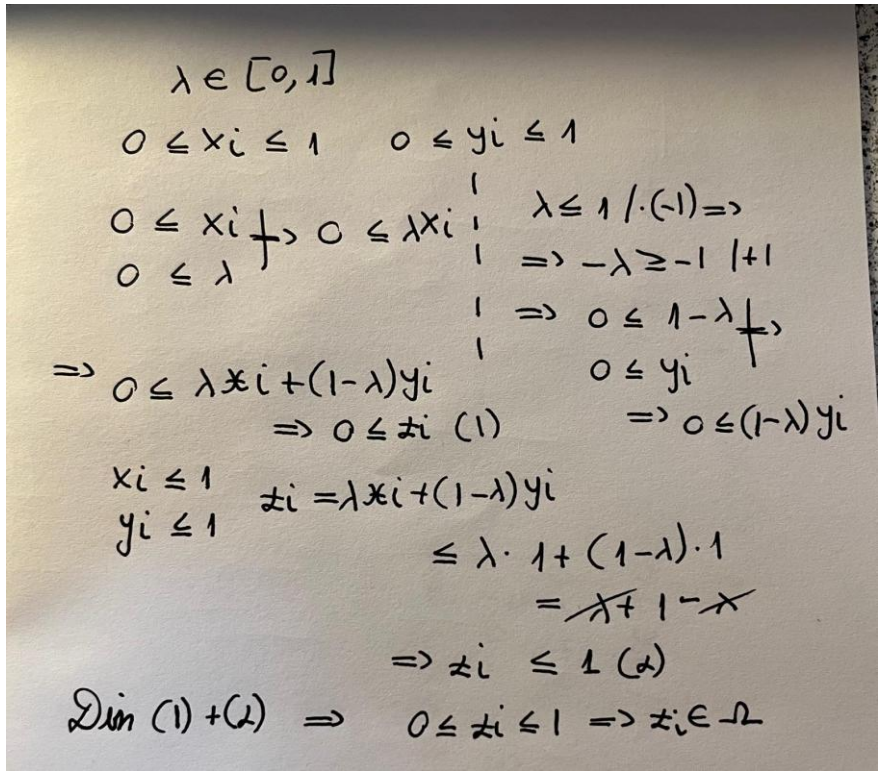
În $\mathbb{R}^{mn}$ orice mulțime închisă și marginită este compactă (conform teoremei Heine-Borel)

3. Rezultate obținute

3. Este o mulțime convexă

Fie x, y $\epsilon\ \Omega$, $0\ \leq\ x_i\ \leq 1, 0\ \leq\ y_i\ \leq 1$, $i = 1, \ldots\ldots, mn$, λ∈[0,1]

$z_i =\ \lambda x_i + (1-\lambda) y_i$ *trebuie să aparțină lui* Ω



3. Rezultate obținute

Algoritmul GC

Date de intrare : $\boldsymbol{x}_0\,(k=0)$ punctul initial, pasul $\alpha_k > 0$

1. Atata timp cat $\texttt{criteriu}(x_k) \geq \epsilon$:

   1.1 $s_k = \arg\min_{s\in\Omega} \nabla^T f(x_k)(s - x_k)$

   1.2 $x_{k+1} = x_k + \alpha(s_k - x_k)$

   1.3 $k = k + 1$

2.Returneaza x_{k+1}

- Metoda gradient condițional

Punctul inițial la pasul k = 0, x_0 - vectorul plin de zero-uri

Pasul $\alpha_k = \frac{2}{k+2}$, pas adaptiv, asigură convergența -» *Jaggi, M. (2013)* – „Revisiting Frank-Wolfe”

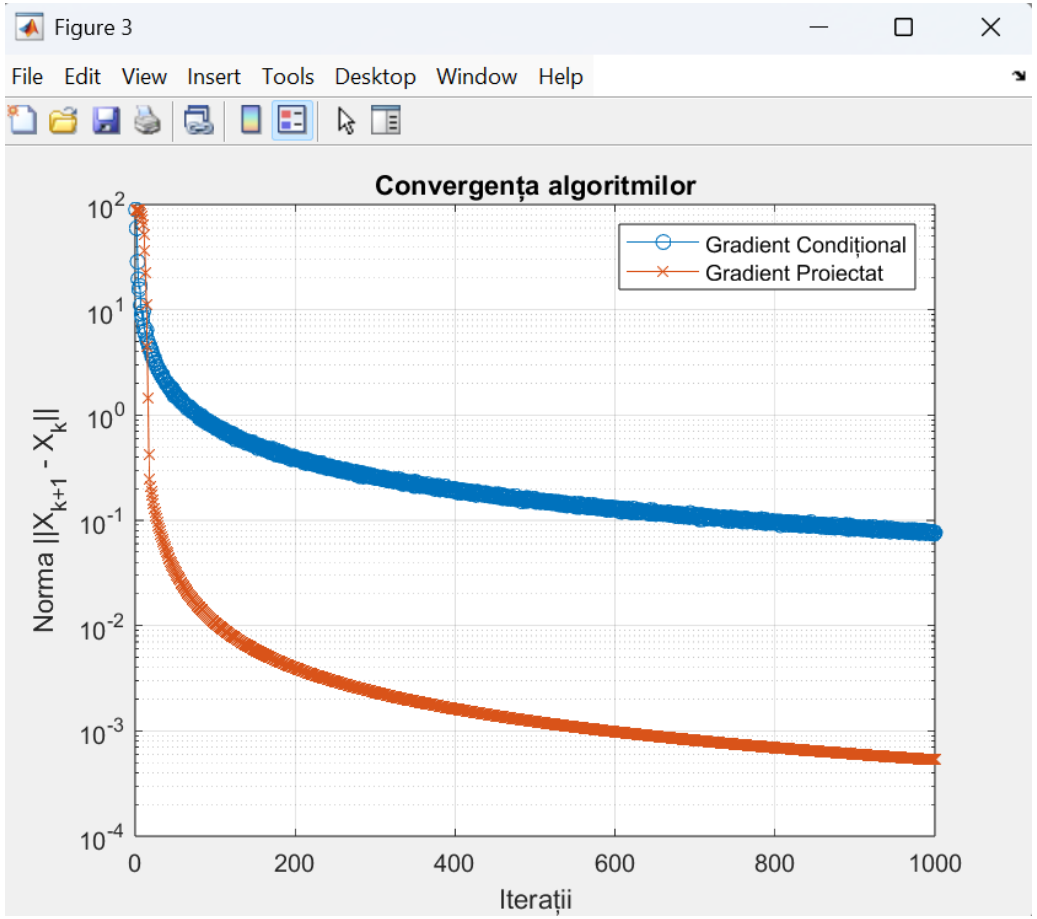
3. Rezultate obținute

Atât pentru MGC cât și pentru MGP criteriul de oprire este:

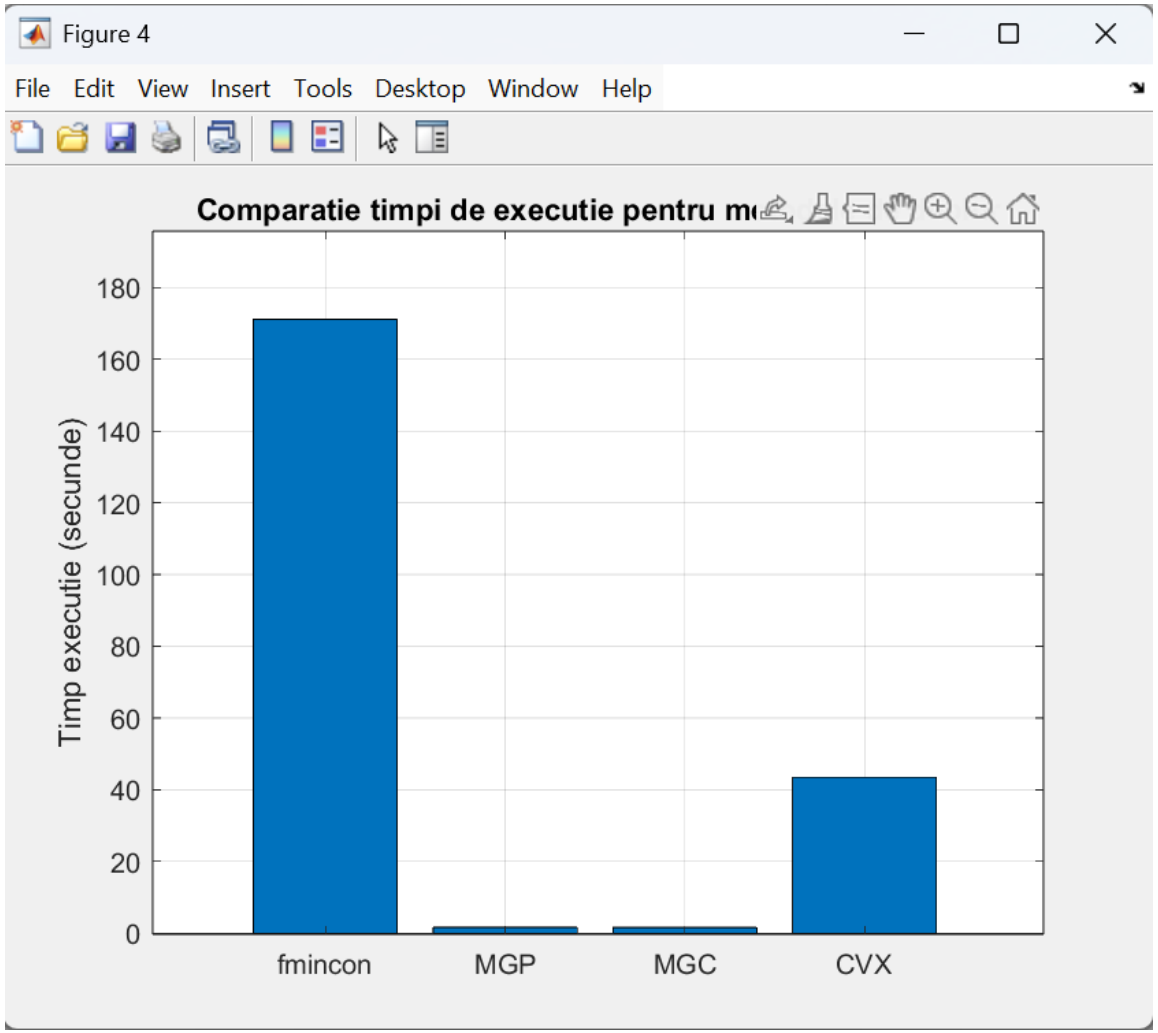
$$\|x_{k+1} - x_k\|_\infty \leq \varepsilon$$

3. Rezultate obținute

Convergența algoritmilor(Metoda gradient proiectat și Metoda gradient condițional sau Frank-Wolfe)

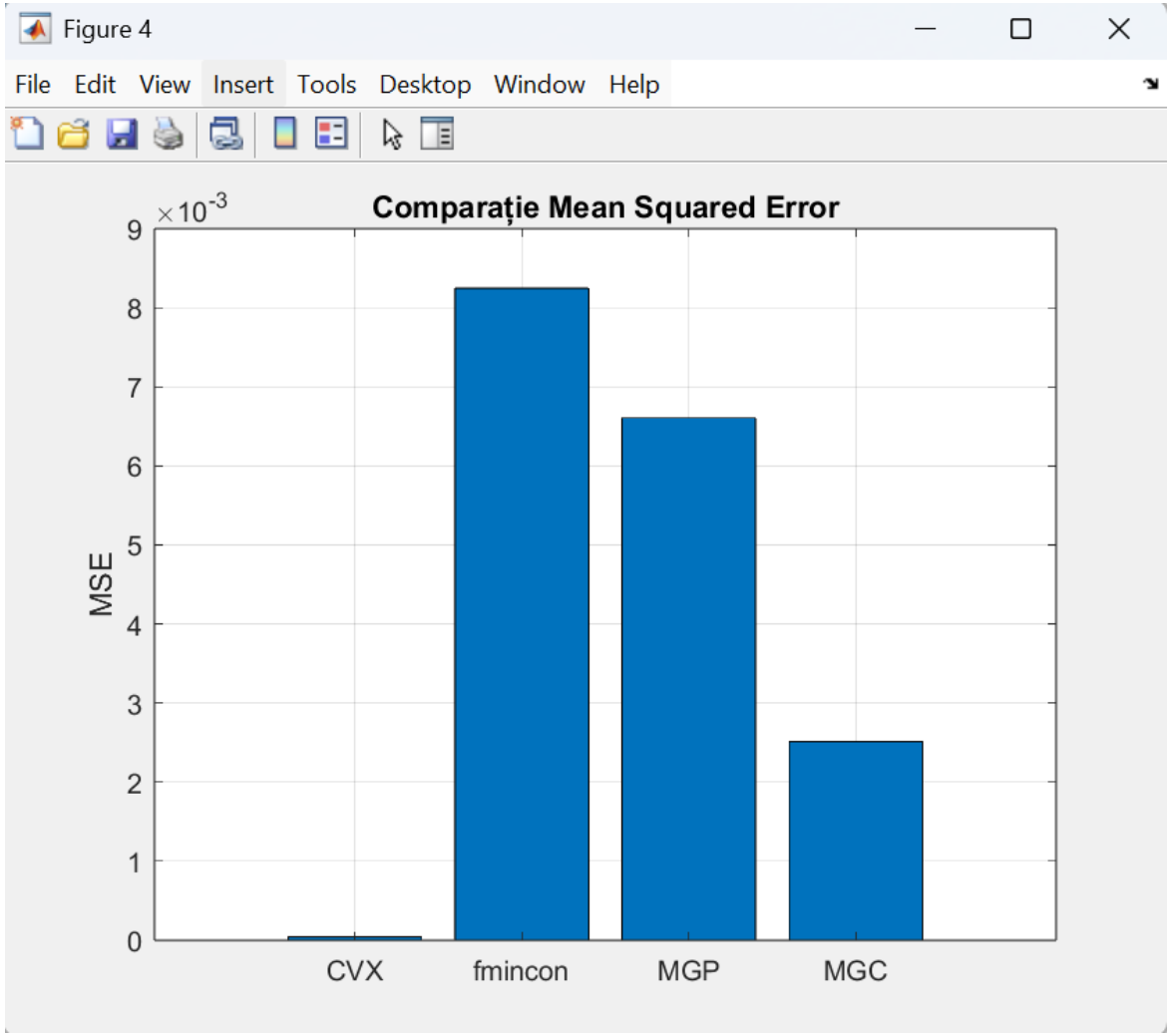


Comparație - timp de execuție



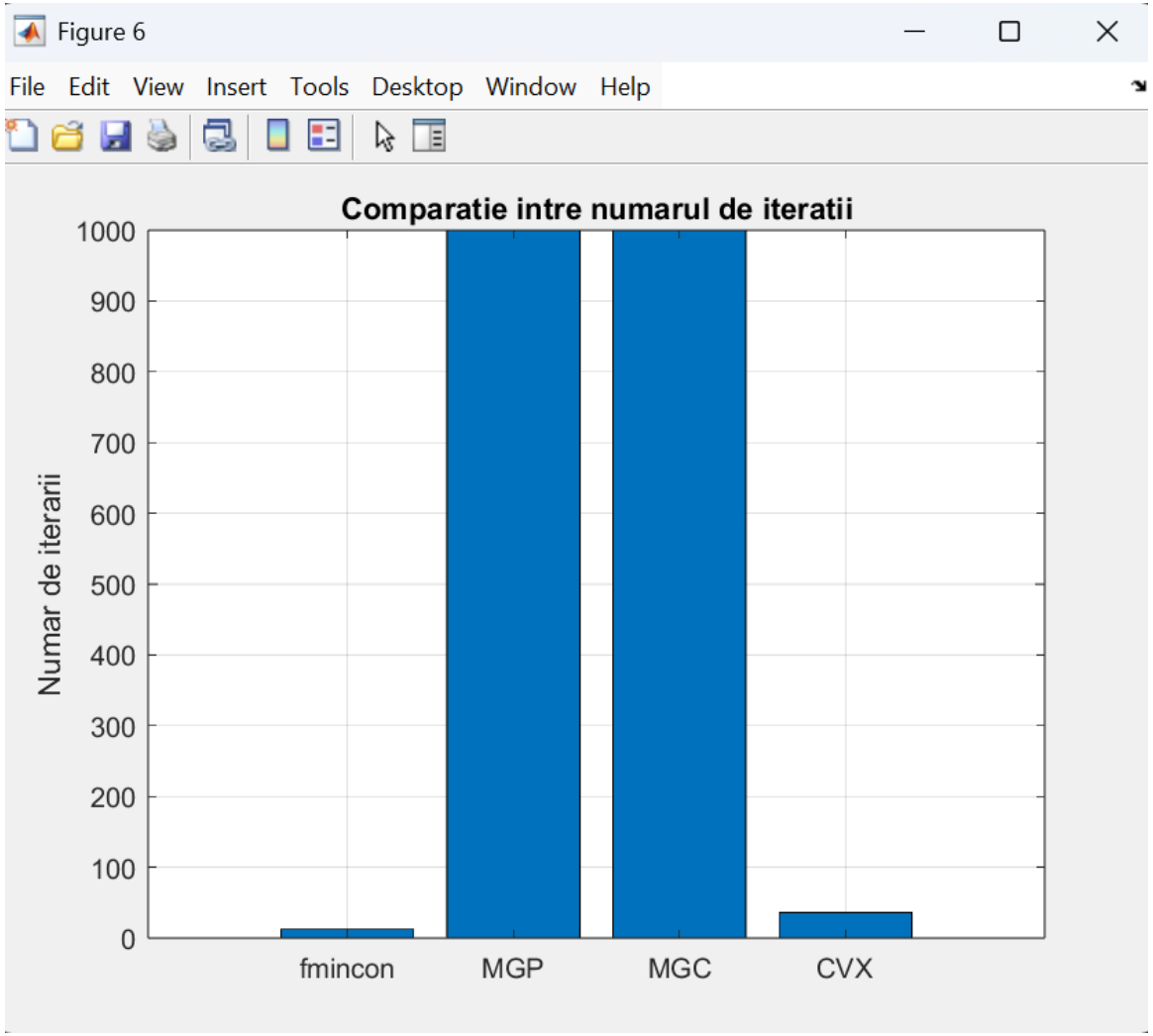
3. Rezultate obținute

Comparație – cea mai apropiată de adevăr metodă



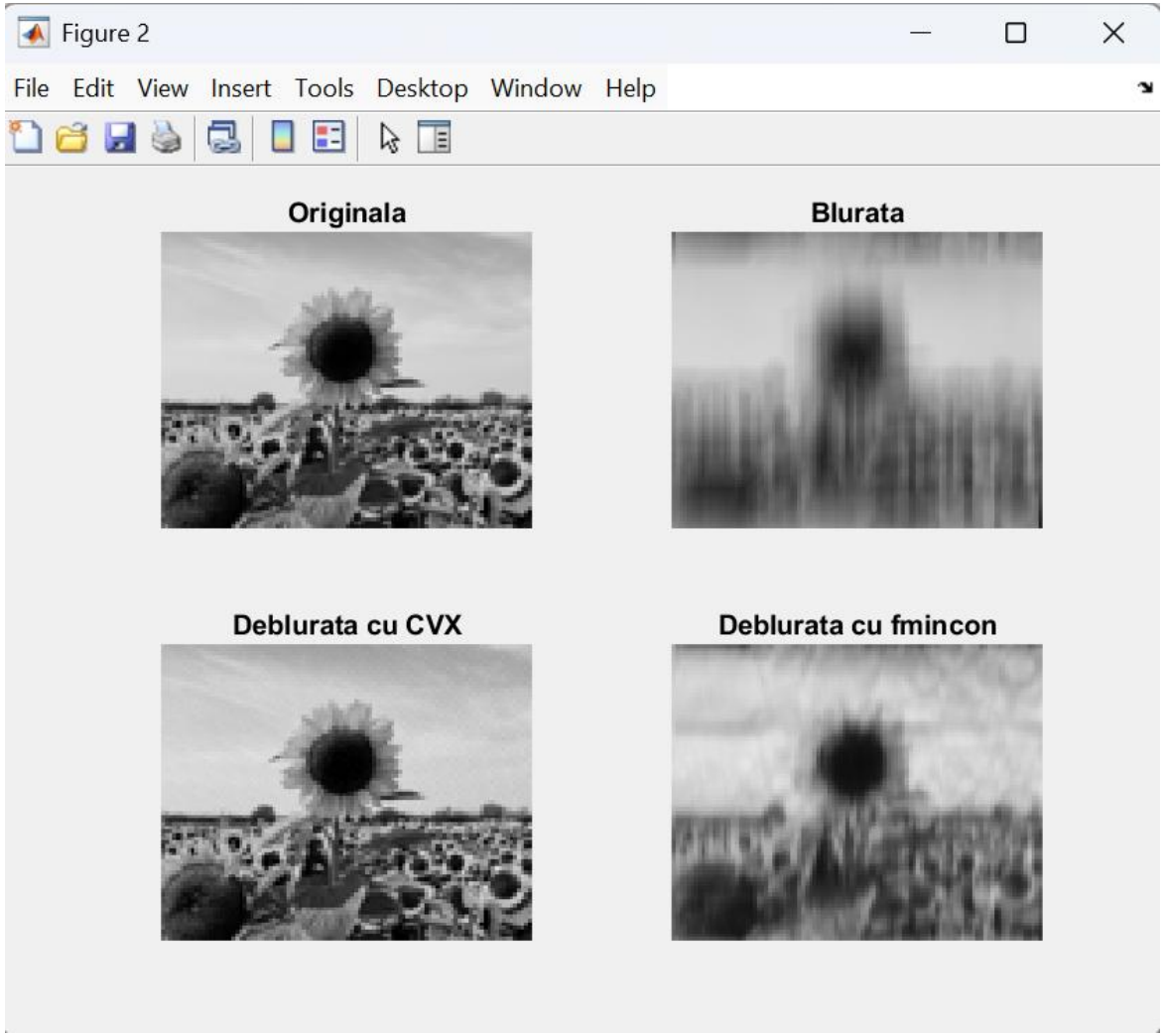
3. Rezultate obținute

Comparație – număr de iterații



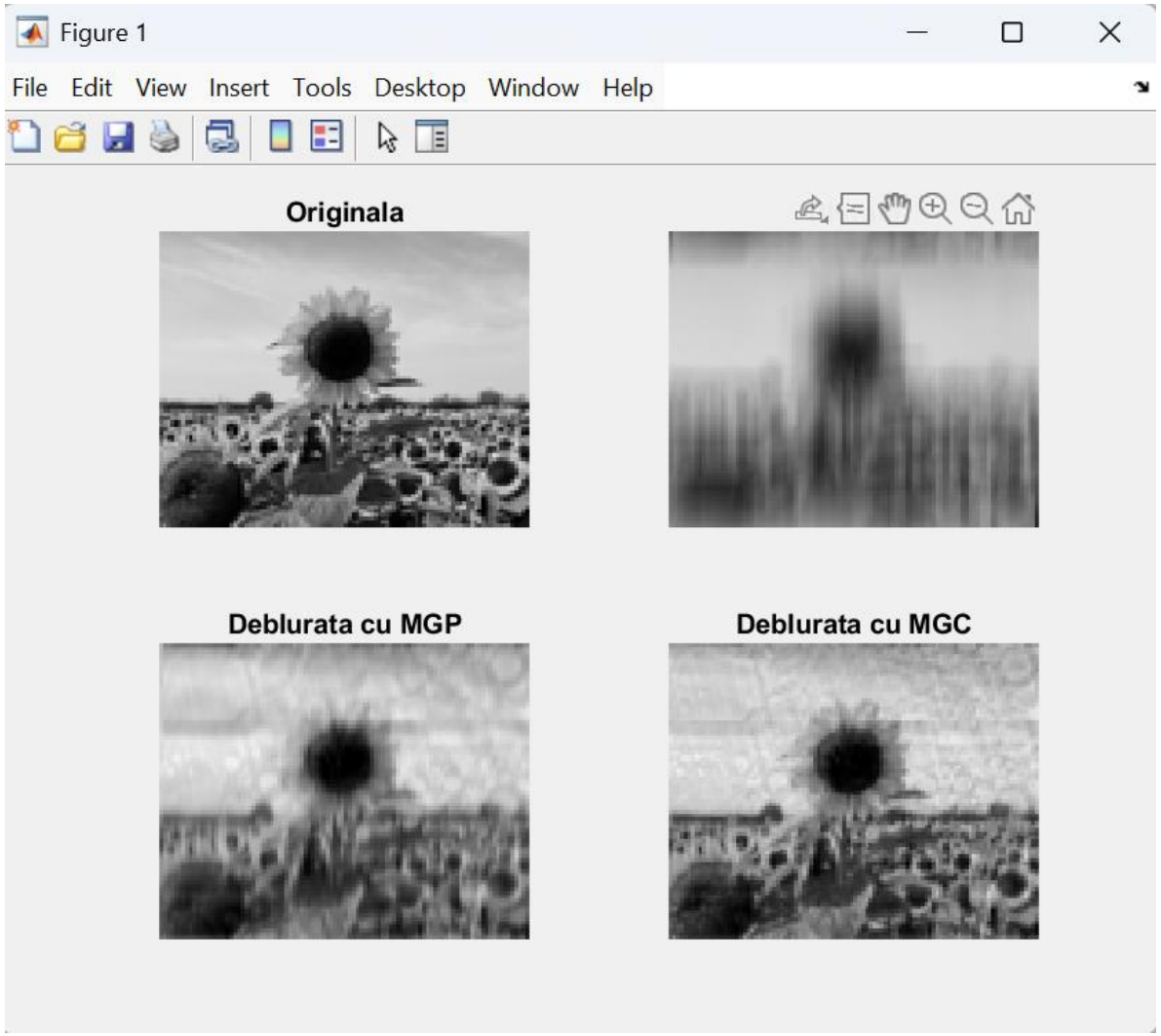
3. Rezultate obținute

Imaginile obținute



3. Rezultate obținute

Imaginile obținute



3. Rezultate obținute

Anexă – main.m

Editor - C:\Users\maria\OneDrive\Documents\Facultate\Anul 2\Semestrul2\Optimizari\Colocviu\main.m

main.m | gradient_proiectat.m | gradient_conditional.m | create_blur_matrix.m

```
clc; clear; close all;

% imagine - matrice - culori alb-negru, 0<=pixel<=1
Image = im2double(rgb2gray(imread('original.jpg')));

% redimensionare imagine, sa aiba maxim 100 pe linie/coloana
max_size = 100;
[m, n] = size(Image);
scale = max_size / max(m, n);
Image_resized = imresize(Image, scale); % imagine originala - matrice
[m, n] = size(Image_resized);

% numarul de linii din vectorii coloana
mn = m * n;

% imaginea - matrice de pixeli(valori)
x_true = Image_resized(:); % transform imaginea matrice in imagine vector, imaginea originala

% valoarea blur-ului, valoare mai mare blur mai mare, valoare mai mica blur mai mic
blur = 10;

% creez matricea de blur(matrice de mediere)
D = create_blur_matrix(mn, blur);

y = D * x_true; % creez imaginea blurata - vector
Y = reshape(y, m, n); % transform imaginea blurata vector in imagine blurata matrice

% figure;
% subplot(1,2,1); imshow(Image_resized); title('Originala');
% subplot(1,2,2); imshow(Y); title('Blurata');

% Image_resized - imaginea originala sub forma de matrice
```

Editor - C:\Users\maria\OneDrive\Documents\Facultate\Anul 2\Semestrul2\Optimizari\Colocviu\main.m

main.m | gradient_proiectat.m | gradient_conditional.m | create_blur_matrix.m

```
% Image_resized - imaginea originala sub forma de matrice
% x_true - imaginea originala sub forma de vector
% Y - imaginea blurata sub forma de matrice
% y - imaginea blurata sub forma de vector


% ---------------------------------------------------------------------
%                       CVX
% Rezolvarea problemei de optimizare folosind CVX
tic;
cvx_begin
    variable x(mn)    % variabila de decizie
    minimize (square_pos(norm(D*x-y)));
    subject to
        0 <= x <= 1     % constrangerile
cvx_end
t_cvx = toc;

X_cvx = reshape(x, m, n);  % transform imaginea obtinuta din vector in matrice
% figure;
% subplot(1,3,1); imshow(Image_resized); title('Originala');
% subplot(1,3,2); imshow(Y); title('Blurata');
% subplot(1,3,3); imshow(X_cvx); title('Deblurata cu CVX');

% ---------------------------------------------------------------------
%                   Cu fct. MatLab: fmincon

% Fct obiectiv: f(x) = ||Dx - y||^2
fun = @(x) norm(D * x - y)^2;

% Dimensiune var.
```

Editor - C:\Users\maria\OneDrive\Documents\Facultate\Anul 2\Semestrul2\Optimizari\Colocviu\main.m

main.m | gradient_proiectat.m | gradient_conditional.m | create_blur_matrix.m

```
% Dimensiune var.
mn = length(y);

% Cstr.: 0 <= x <= 1
lb = zeros(mn,1);
ub = ones(mn,1);

% Punct initial
x0 = 0.5 * zeros(mn,1);

% Optiuni (pentru a urmari progresul si a seta limita de iteratii)
options = optimoptions('fmincon', 'Display', 'iter', ...
    'Algorithm', 'interior-point', ...
    'MaxIterations', 1000, ...
    'MaxFunctionEvaluations', 1e5, ...
    'OptimalityTolerance', 1e-6);

% Rezolvare
tic;
[x_fmincon, fval_fmincon, exitflag, output] = fmincon(fun, x0, [], [], [], [], lb, ub, [], options);
t_fmincon = toc;

% Reconstruire imagine
X_fmincon = reshape(x_fmincon, m, n);


% ------------------------------------------------------------------------
%                   METODA GRADIENT PROIECTAT

max_iter = 1000;
c = 10;
epsilon = 1e-4;
```

Editor - C:\Users\maria\OneDrive\Documents\Facultate\Anul 2\Semestrul2\Optimizari\Colocviu\main.m

main.m | gradient_proiectat.m | gradient_conditional.m | create_blur_matrix.m

```
c = 10;
epsilon = 1e-4;
tic;
[x_gp, iter_vec, diff_vec] = gradient_proiectat(D, y, max_iter, c, epsilon);
t_gp = toc;
X_gp = reshape(x_gp, m, n);


% figure;
% subplot(2,2,1); imshow(Image_resized); title('Originala');
% subplot(2,2,2); imshow(Y); title('Blurata');
% subplot(2,2,3); imshow(X_cvx); title('Deblurata cu CVX');
% subplot(2,2,4); imshow(X_gp); title('Deblurata cu MGP');
%
% figure;
% semilogy(iter_vec, diff_vec, '-o');
% xlabel('Iteratii');
% ylabel('Norma ||X_{k+1} - X_k||');
% title('Criteriu de oprire pentru Gradient Proiectat');
% grid on;

% ------------------------------------------------------------------------
%                   METODA GRADIENT CONDITIONAL (FRANK-WOLFE)


max_iter1 = 1000;
epsilon1 = 1e-4;
tic;
[x_gc, iter_gc, diff_gc] = gradient_conditional(D, y, max_iter1, epsilon1);
t_gc = toc;
X_gc = reshape(x_gc, m, n);
```

Editor - C:\Users\maria\OneDrive\Documents\Facultate\Anul 2\Semestrul2\Optimizari\Colocviu\main.m

main.m | gradient_proiectat.m | gradient_conditional.m | create_blur_matrix.m

```
124
125 % cu metodele de optimizare alese de mine
126 figure;
127 subplot(2,2,1); imshow(Image_resized); title('Originala');
128 subplot(2,2,2); imshow(Y); title('Blurata');
129 subplot(2,2,3); imshow(X_gp); title('Deblurata cu MGP');
130 subplot(2,2,4); imshow(X_gc); title('Deblurata cu MGC');
131
132 % cu cvx si functii matlab
133 figure;
134 subplot(2,2,1); imshow(Image_resized); title('Originala');
135 subplot(2,2,2); imshow(Y); title('Blurata');
136 subplot(2,2,3); imshow(X_cvx); title('Deblurata cu CVX');
137 subplot(2,2,4); imshow(X_fmincon); title('Deblurata cu fmincon');
138
139 % grafice pentru convergenta algoritmilor MGP si MGC
140 figure;
141 semilogy(iter_gc, diff_gc, '-o'); hold on;
142 semilogy(iter_vec, diff_vec, '-x');
143 legend('Gradient Conditional', 'Gradient Proiectat');
144 xlabel('Iteratii');
145 ylabel('Norma ||X_{k+1} - X_k||');
146 title('Convergenta algoritmilor');
147 grid on;
148
149 %      Comparare timpi de executie intre metode
150
151 % Vector cu timpi (in secunde) pentru toate metodele
152 times = [t_fmincon, t_gp, t_gc, t_cvx];
153
154 % Etichete metode
155 labels = {'fmincon', 'MGP', 'MGC', 'CVX'};
```

Editor - C:\Users\maria\OneDrive\Documents\Facultate\Anul 2\Semestrul2\Optimizari\Colocviu\main.m

main.m | gradient_proiectat.m | gradient_conditional.m | create_blur_matrix.m

```
155 labels = {'fmincon', 'MGP', 'MGC', 'CVX'};
156
157 % Grafic bara
158 figure;
159 bar(times);
160 set(gca, 'xticklabel', labels);
161 ylabel('Timp executie (secunde)');
162 title('Comparatie timpi de executie pentru metodele de deblur');
163 grid on;
164
165 % Comparatie intre acuratetea rezultatelor
166 mse_cvx = immse(X_cvx, Image_resized);
167 mse_fmincon = immse(X_fmincon, Image_resized);
168 mse_mgp = immse(X_gp, Image_resized);
169 mse_mgc = immse(X_gc, Image_resized);
170
171 % Grafic MSE
172 mse_vals = [mse_cvx, mse_fmincon, mse_mgp, mse_mgc];
173 labels = {'CVX', 'fmincon', 'MGP', 'MGC'};
174
175 figure;
176 bar(mse_vals);
177 set(gca, 'xticklabel', labels);
178 ylabel('MSE');
179 title('Comparatie Mean Squared Error');
180 grid on;
181
182
183 % comparatie intre numarul de iteratii
184 num_iter_gp = length(iter_vec);
185 num_iter_gc = length(iter_gc);
186 num_iter_fmincon = output.iterations;
```

Editor - C:\Users\maria\OneDrive\Documents\Facultate\Anul 2\Semestrul2\Optimizari\Colocviu\main.m

main.m | gradient_proiectat.m | gradient_conditional.m | create_blur_matrix.m

```
175     figure;
176     bar(mse_vals);
177     set(gca, 'xticklabel', labels);
178     ylabel('MSE');
179     title('Comparatie Mean Squared Error');
180     grid on;
181
182
183     % comparatie intre numarul de iteratii
184     num_iter_gp = length(iter_vec);
185     num_iter_gc = length(iter_gc);
186     num_iter_fmincon = output.iterations;
187     num_iter_cvx = 36;
188
189     num_iter = [num_iter_fmincon, num_iter_gp, num_iter_gc, num_iter_cvx];
190     labels = {'fmincon', 'MGP', 'MGC', 'CVX'};
191
192     figure;
193     bar(num_iter);
194     set(gca, 'xticklabel', labels);
195     ylabel('Numar de iterarii');
196     title('Comparatie intre numarul de iteratii');
197     grid on;
```

Anexă – gradient_proiectat.m

Editor - C:\Users\maria\OneDrive\Documents\Facultate\Anul 2\Semestrul2\Optimizari\Colocviu\gradient_proiectat.m

main.m | gradient_proiectat.m | gradient_conditional.m | create_blur_matrix.m

```
function [x, iter_vec, diff_vec] = gradient_proiectat(D, y, max_iter, c, epsilon)

    mn = length(y);
    x = zeros(mn, 1);  % punctul initial
    iter = 0;
    diff = Inf;

    % vectorii pentru plot ulterior
    iter_vec = [];
    diff_vec = [];

    while iter < max_iter && diff > epsilon
        x_prev = x;
        grad = 2 * D' * (D * x - y);
        alpha = c / (iter + 1);
        x = x - alpha * grad;
        x = min(max(x, 0), 1); % proiectie pe [0,1]

        % Diferenta intre iteratii (norma euclidiana)
        diff = norm(x - x_prev);

        iter = iter + 1;
        iter_vec(end+1) = iter;
        diff_vec(end+1) = diff;
    end

end
```

Anexă – gradient_conditional.m

Editor - C:\Users\maria\OneDrive\Documents\Facultate\Anul 2\Semestrul2\Optimizari\Colocviu\gradient_conditional.m

main.m | gradient_proiectat.m | gradient_conditional.m | create_blur_matrix.m

```
function [x, iter_vec, diff_vec] = gradient_conditional(D, y, max_iter, epsilon)

    mn = length(y);
    x = zeros(mn, 1); % punctul initial
    iter = 0;
    diff = Inf;

    % vectorii pentru plot ulterior
    iter_vec = [];
    diff_vec = [];

    while iter < max_iter && diff > epsilon
        grad = 2 * D' * (D * x - y);

        % Minimizez prod. scalar grad'*s sub constrangerea s apartine lui
        % [0,1]^n
        % s(i) = 0 daca grad(i) > 0, s(i) = 1 daca grad(i) < 0
        s = double(grad < 0);

        % Pas adaptiv: gamma = 2 / (k + 2)
        gamma = 2 / (iter + 2); % iter + 1 + 1, incep de la 0

        x_new = x + gamma * (s-x);

        % Calcul criteriu de oprire
        diff = norm(x_new - x);

        iter = iter + 1;
        iter_vec(end+1) = iter;
        diff_vec(end+1) = diff;

        x = x_new;
    end

end
```

Anexă –create_blur_matrix.m

Editor - C:\Users\maria\OneDrive\Documents\Facultate\Anul 2\Semestrul2\Optimizari\Colocviu\create_blur_matrix.m

main.m | gradient_proiectat.m | gradient_conditional.m | create_blur_matrix.m

```
1  function D = create_blur_matrix(mn, blur)
2      mindex = 1:mn;
3      nindex = 1:mn;
4      for i = 1:blur
5  %                               vecini
6  %                             stg     dr
7          mindex = [mindex, i+1:mn, 1:mn-i]; % pentru liniile din D
8          nindex = [nindex, 1:mn-i, i+1:mn]; % pentru coloanele din D
9      end
10     D = sparse(mindex, nindex, 1 / (2*blur + 1), mn, mn); % media vec. stg + element curent +
11                                                           % + vec. dr
12 end
13
14 % blur - cati vecini la stanga si la dreapta consider pentru mediere
```

Bibliografie

- Curs Optimizări
- Laborator Optimizări (Laboratorul 4)
- https://www.mathworks.com/help/optim/ug/deblur-problem-based.html
- Conditional Gradient (Frank-Wolfe) Method – Ryan Tibshirani Convex Optimization 10-725/36-725
- *Jaggi, M. (2013)* – „Revisiting Frank-Wolfe” - https://proceedings.mlr.press/v28/jaggi13.html