Tehnici de Optimizare

Seminar 1

1 Noţiuni introductive

1.1 Continuitate şi Diferenţiabilitate

O funcţie $f : \mathbb{R}^n \mapsto \mathbb{R}$ este *diferenţiabilă* în x dacă există $a := \nabla f(x) \in \mathbb{R}$ astfel încât:

$$f(x + d) = f(x) + a^T d + o(d),$$

unde $\lim_{t\to 0} \frac{o(t)}{\|t\|} = 0$ si $o(0) = 0$. Cu alte cuvinte f este diferentiabila in x daca admite o aproximare liniara de ordin I $\tilde{f}(x; d)$ in x:

$$f(x + d) - \underbrace{f(x) - \nabla f(x)^T d}_{\tilde{f}(x;d)} = o(d).$$

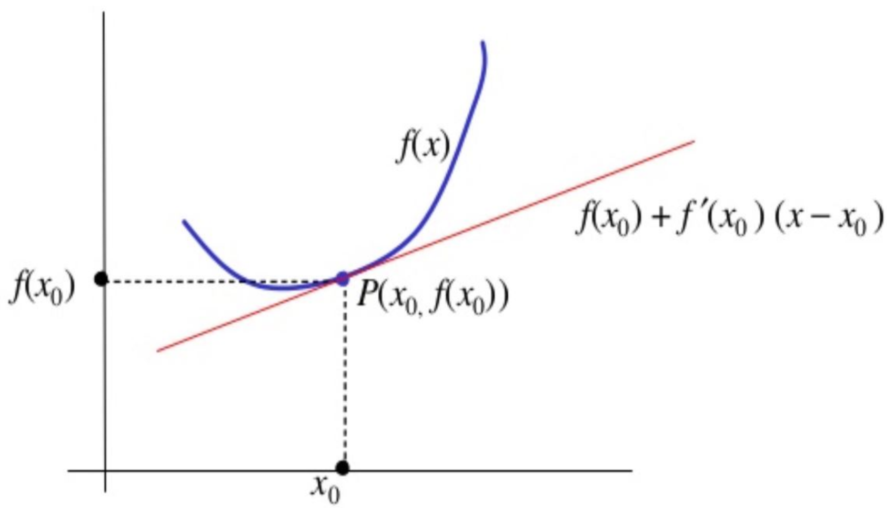


Figure 1: slideshare.net

De exemplu, considerati: $f : \mathbb{R}^n \to \mathbb{R}, A \in \mathbb{R}^{n\times n}, A = A^T$

$$f(x) = 2x_1^2 - 4x_2^2 + 3x_1x_2 = \frac{1}{2}[x_1 \quad x_2] \begin{bmatrix} 4 & 3 \\ 3 & -8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\nabla f(x) = \begin{bmatrix} 4x_1 + 3x_2 \\ -8x_2 + 3x_1 \end{bmatrix} = Ax \qquad \nabla^2 f(x) = \begin{bmatrix} 4 & 3 \\ 3 & -8 \end{bmatrix}$$

- Matricea **HESSIANA** A este constanta in raport cu variabila x (in cazul functiilor patratice)

$$f(x) = \frac{1}{2}x^T Ax - b^T x,$$

unde $A \in \mathbb{R}^{n\times n}$ simetrică. Atunci verifică definiţia de mai sus?

$$\begin{aligned}
f(x+d) &= \frac{1}{2}(x+d)^T A(x+d) - b^T(x+d)\\
&= \frac{1}{2}x^T A(x+d) + \frac{1}{2}d^T A(x+d) - b^T x - b^T d\\
&= \frac{1}{2}x^T Ax + \frac{1}{2}x^T Ad + \frac{1}{2}d^T Ax + \frac{1}{2}d^T Ad - b^T x - b^T d\\
&= f(x) + d^T Ax - d^T b + \frac{1}{2}d^T Ad\\
&= f(x) + d^T(Ax - b) + \underbrace{\frac{1}{2}d^T Ad}_{o(d)}
\end{aligned}$$

$$|d^T Ad| \leq \|A\|\|d\|^2$$

$$0 = \lim_{\|d\|\to 0} \frac{|d^T Ad|}{\|d\|} \leq \lim_{\|d\|\to 0} \|A\|\|d\|_2 = 0$$

Daca da, ce expresie are $\nabla f(x)$? $\nabla f(x) = Ax - b$

O funcţie $f : \mathbb{R}^n \mapsto \mathbb{R}$ este continua Lipschitz dacă există $L < \infty$ astfel încât:

$$|f(x) - f(y)| \leq L\|x - y\|, \qquad x, y \in \mathrm{dom} f.$$

Pe scurt, variatiile functiei f (in raport cu argumentul) sunt marginite de constanta L. Ca sa aratam proprietati de continuitate, este utila *inegalitatea triunghiului.*

Exemple:

- $f(x) = |x|$
- $f(x) = |c^T x|$
- $f(x) = 2x^2$?

O funcţie diferenţiabilă $f : \mathbb{R}^n \mapsto \mathbb{R}$ are gradient continuu Lipschitz dacă există $L < \infty$ astfel încât:

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \qquad x, y \in \mathrm{dom} f.$$

Pe scurt, variatiile primei derivate ale functiei f sunt marginite de constanta L. Se poate arata ca definitia de mai sus este echivalenta cu :

$$|f(x) - f(y) - \nabla f(y)^T(x - y)| \leq \frac{L}{2}\|x - y\|^2, \qquad x, y \in \mathrm{dom} f.$$

Graficul functiei f poate fi aproximat superior/inferior cu o functie patratica simpla convexa/concava. In general, e utila proprietatea:

$$\|Ax\| \leq \|A\|\|x\|$$

Exemple:

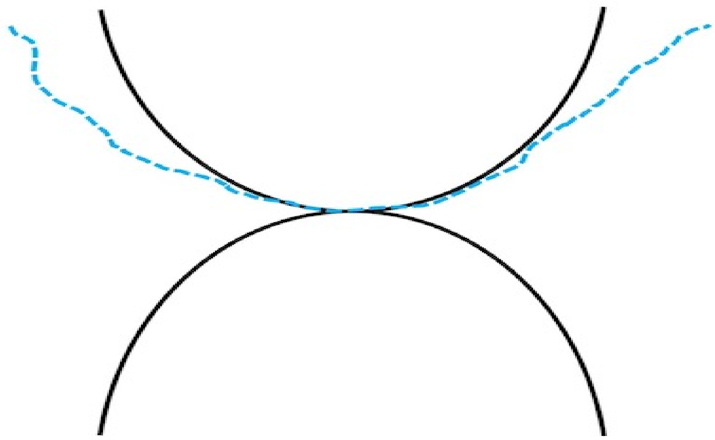


Figure 2: csinva.github.io

- $f(x) = |x|?, f'(x) = \text{sign}(x)\ [x \neq 0]$; in 0, f nu este diferentiabila.
- $f(x) = \frac{1}{2}x^T Hx, \nabla f(x) = Hx \Rightarrow \|Hx - Hy\| \leq \|H\|\|x-y\| = L\|x-y\|; \|H\|_2 = \max_i |\lambda_i(H)|$
- $f(x) = x^3; f'(x) = 3x^2; |f'(x) - f'(y)| = 3|x^2 - y^2| = \underbrace{3|x+y|}_{\to\infty (pentru\ x,y\to\infty)} |x-y|$
- $f(x) = \frac{1}{2}(c^T x)^2$

Exercitii. Calculaţi prima derivată şi a doua derivată a următoarelor funcţii:

(i) $f(x) = ax; f'(x) = a; f''(x) = 0$

(ii) $f(x) = a^T x + b; \nabla f(x) = a; \nabla^2 f(x) = 0_{n\times n}$

(iii) $f(x) = ax^2 + bx + c; \quad f(x_1, x_2) = x_1^2 + x_2^2 - 2x_1x_2 = \frac{1}{2}[x_1 \ \ x_2] \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

(iv) $f(x) = \|x\|_2^2 = x^T x = \sum_{i=1}^{n} x_i^2; \nabla f(x) = 2x; \nabla^2 f(x) = 2I_n$

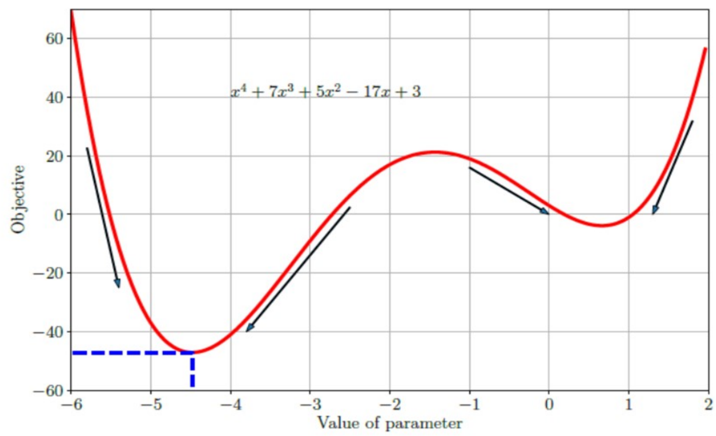
(v) $f(x) = (a^T x)^2 = a^T x \cdot a^T x = x^T(aa^T)x; \nabla^2 f(x) = aa^T \in \mathbb{R}^{n\times n}$

(vi) $f(x) = \|Ax - b\|_2^2$

1.2 Puncte de extrem

Scopul programarii matematice este *gasirea punctelor de extrem* ale functiei obiectiv, care respecta constrangerile problemei.

- punct de minim (local)
- punct de maxim (local)
- punct şa



In general, punctele de extrem relevante sunt greu de obtinut daca functia nu are proprietati de ”netezime” (diferentiabilitate) sau convexitate. Exista relatii analitice care ne ajuta sa le calculam?

O funcţie $f : \mathbb{R}^n \mapsto (-\infty, \infty]$ se numeste *convexă* dacă:

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y) \qquad \forall \alpha \in [0,1], x, y \in \text{dom} f.$$

Punctele de extrem ale unei functii convexe sunt puncte de minim GLOBAL.

Exemple:
1. $f(x) = ax + b; f(\alpha x + (1-\alpha)y) = a[\alpha x + (1-\alpha)y] + b = \alpha(ax+b) + (1-\alpha)(ay+b); f(x) = a^T x + b$

2. $f(x) = |x|$, cu extensie la $f(x) = \|x\|$

3. $f(x) = \frac{x^2}{2}$ cu extensie la $f(x) = \frac{1}{2}\|x\|^2$

4. $f(x) = \frac{1}{2}x^T H x$?

2 Introducere în pachetul de optimizare Scipy.optimize

SciPy *optimize* conţine algoritmi pentru minimizarea (sau maximizarea) locală şi globală a funcţiilor, eventual supuse la constrângeri. Pachetul utilizează implementări ale solver-elor pentru probleme neliniare, programare liniară, probleme de regresie (Cele-Mai-Mici-Pătrate) neliniare şi constrânse, calcul de rădăcini şi aproximări de curbe.

Funcţia principală de minimizare a unei funcţii scalare de una sau mai multe variabile este *minimize*.

```
scipy.optimize.minimize(fun, x0, args=(), method=None, jac=None, hess=None, hessp=None,
bounds=None, constraints=(), tol=None, callback=None, options=None)
```

Principalii parametri :

- **fun**: Funţia obiectiv de minimizat.

  ```
  fun(x, *args) -> float
  ```

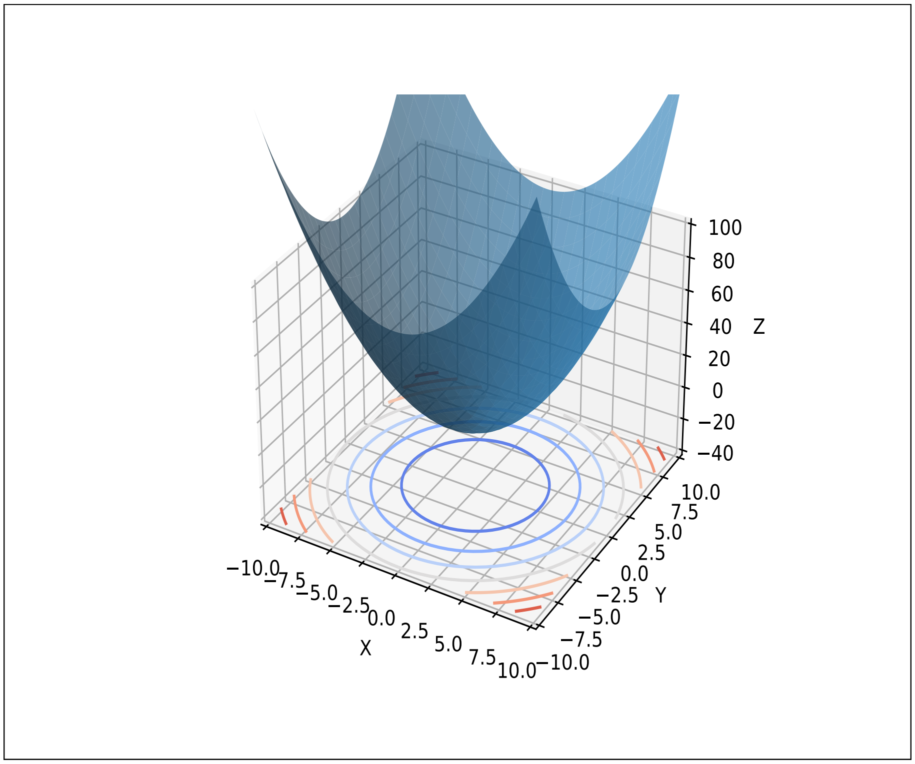
unde x este un array $1-D$ cu forma $(n,)$ şi *args* este o colecţie de parametri fixaţi (constante).

- **x0**: Punct iniţial. Tablou de elemente reale de dimensiune $(n,)$, unde n este numărul de variabile.
- **args**: Argumente transmise funcţiei obiectiv si derivatelor ei.
- **method**: Algoritmul de optimizare. Exemple: 'Nelder-Mead', 'Powell', 'CG' etc.
- **jac**: Funcţie de definire a Jacobianului
- **hess**: Funcţie de definire a matricei Hessiene
- .....

Pentru a figura graficele functiilor obiectiv si traiectoriile algoritmilor folosim pachetul *matplotlib*. Vezi fisierul *plot_functions.py*.

In general, funcţiile pe care analizăm nu pot fi figurate grafic datorita numărului de variabile, înă pentru a obţine o înţelegere intuitivă a peisajului geometric a acesteia se utilizează mulţimile izonivel (sau subnivel). **Definiţie.** Mulţimea izonivel a funcţiei $f : \mathbb{R}^n \mapsto \mathbb{R}$ este:

$$L_f(\alpha) = \{x \in \mathbb{R}^n : f(x) = \alpha\}$$



3 Probleme de minimizare neconstrânsă

Problemele de optimizare neconstrânsă presupun minimizarea (sau maximizarea) unei anumite funcţii obiectiv (criteriu) prin intermediul unui set de variabile de decizie, fără ca acestea să respecte anumite restricţii (constrângeri). Se consideră funcţia (în general, neliniară) $f : \mathbb{R}^n \to \mathbb{R}$, căreia i se asociază problema de optimizare:

$$\min_{x \in \mathbb{R}^n} \quad f(x)$$

unde x reprezintă vectorul variabilelor de decizie. Problema de optimizare se consideră rezolvată dacă s-a obţinut un vector x^* pentru care valoarea funcţiei f în x^* este minimă, adică $f(x^*) \leq f(x)$ pentru oricare $x \in \mathbb{R}^n$. Mulţimea în care se efectuează căutarea se numeşte mulţime fezabilă, care în cazul de faţă este dată de întreg spaţiul Euclidian $\mathbb{R}^n$.

Pentru problemele fără constrângeri, functia *minimize* oferă:

- algoritmi de ordin zero "black-box" (e.g. metode Nedler-Mead), care utilizează doar evaluari ale funcţiei obiectiv pentru a găsi minimum-ul
- algoritmi de ordin întâi, care utilizează, in plus fata de cei de ordin zero, prima derivata a functiei obiectiv (e.g. metoda Gradientilor Conjugati, metoda BFGS)
- algoritmi de ordin doi, care utilizează, in plus fata de cei de ordin I, şi a doua derivată a funcţiei obiectiv (e.g. metoda Newton)

Exemplu. Consideram functia patratica:

$$f(x) = 2x_1^2 + 2x_2^2 + 3x_1x_2$$

Deoarece minimul se atinge in $(0, 0)$, soluţia reală a problemei de optimizare este $x^* = (0, 0)$.

```
x0 = (5,5)
A = np.ones((2,2)) + np.eye(2)
fun = lambda x : x.T@A@x + x[0]*x[1]
res = minimize(fun,x0,method='Nelder-Mead')
    print(res.x)
```

Algoritmul simplex Nelder-Mead converge la punctul suboptimal $\bar{x} = [-1.56554047e-05 \qquad 2.32754122e-05]$. Distantele reziduale $f(\bar{x}) - f(x^*)$ sau $\|\bar{x} - x^*\|$ reprezintă masuri de optimalitate (calitate) ale punctului la care algoritmul s-a oprit.

O altă variantă de program care nu foloseşte expresii lambda este redată mai jos:

```
def func(x,H,q):

   z = 0.5*x.T@H@x + q.T@x

return z

def unconstrained_fexample():

    x0 = np.ones((2,1))
    A = np.ones((2,2))
    b = np.ones((2,1))
    res =minimize(func,x0,args=(A,b),method='BFGS')
    print(res.x)
```

Exerciţiu. Fie problema de optimizare neconstrânsă (Rosenbrock):

$$\min_{x\in\mathbb{R}^2} \; 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

- Aplicaţi functia *minimize* pentru a rezolva problema. Variaţi punctul iniţial şi algoritmul de rezolvare.
- Creaţi o figură care indică graficul funcţiei obiectiv şi punctul de optim găsit

Indicatii: Folosiţi implementările existente ataşate laboratorului! Punctul de minim este $(1, 1)$.

4 Probleme de optimizare cu constrângeri

Problemele de optimizare cu constrangeri presupun minimizarea (sau maximizarea) unei anumite funcţii obiectiv (criteriu), impunand restricţii (constrângeri) asupra variabileor de decizie. Se consideră funcţia (în general, neliniară) $f : \mathbb{R}^n \to \mathbb{R}$, căreia i se asociază problema de optimizare:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ \text{s.t. } & g(x) \le 0, \;\; h(x) = 0, \\ & Ax = b, \;\; Cx \le d. \end{aligned}$$

Multimea fezabila in care se face cautarea punctului de optim x^* este definita de $X = \{x \in \mathbb{R}^n : \; g(x) \le 0, \;\; h(x) = 0, Ax = b, \;\; Cx \le d\}$.

Exemplu. Fie problema de optimizare:

$$\begin{aligned} \min_{x\in\mathbb{R}^2} \; & (x_1 - 3)^2 + (x_2 - 2)^2 \\ \text{s.t. } \; & \|x\|_1 \le 1.5 \end{aligned}$$

Trasaţi grafic mulţimile izonivel ale funcţiei obiectiv şi mulţimea fezabilă.

```
x, y = np.mgrid[-2.03:4.2:.04, -1.6:3.2:.04]
x = x.T
y = y.T

plt.figure(1, figsize=(3, 2.5))
plt.clf()
plt.axes([0, 0, 1, 1])

contours = plt.contour(np.sqrt((x - 3)**2 + (y - 2)**2),
                       extent=[-2.03, 4.2, -1.6, 3.2],
                       cmap=plt.cm.gnuplot)
plt.clabel(contours, inline=1, fmt='%1.1f', fontsize=14)
plt.plot([-1.5,    0,  1.5,    0, -1.5],
         [   0,  1.5,    0, -1.5,    0], 'k', linewidth=2)
plt.fill_between([ -1.5,    0,  1.5],
                 [    0, -1.5,    0],
                 [    0,  1.5,    0],color='.8')
plt.axvline(0, color='k')
plt.axhline(0, color='k')

plt.text(-.9, 2.8, '$x_2$', size=20)
```

```
plt.text(3.6, -.6, '$x_1$', size=20)
plt.axis('tight')
plt.axis('off')

# And now plot the optimization path
accumulator = list()

def f(x):
       # Store the list of function calls
       accumulator.append(x)
       return np.sqrt((x[0] - 3)**2 + (x[1] - 2)**2)


def constraint(x):
       return np.atleast_1d(1.5 - np.sum(np.abs(x)))

optimize.minimize(f, np.array([0, 0]), method="SLSQP",
                  constraints={"fun": constraint, "type": "ineq"})

accumulated = np.array(accumulator)
plt.plot(accumulated[:, 0], accumulated[:, 1], color='g')



plt.show()
```

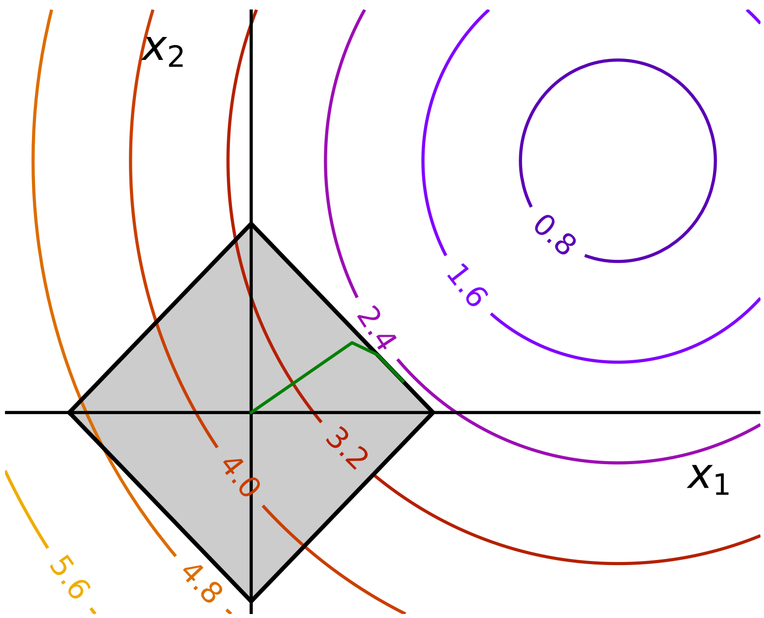


Figure 3: Mulţimile izonivel ale funcţiei $(x_1 - 3)^2 + (x_2 - 2)^2$, reprezentarea mulţimii fezabile şi traiectoria algoritmului Sequential Least-Squares Programming

Exerciţiu. Fie problema de optimizare constrânsă:

$$\begin{aligned}\min_{x\in\mathbb{R}^2} \quad & 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \\ \text{s.t. } & \|x\|_2^2 \leq r\end{aligned}$$

- Aplicaţi functia *minimize* pentru a rezolva problema pentru $r \in \{1, 2, 3\}$
- Creati o figura care indica graficul functiei obiectiv si punctul de optim gasit pentru cele 3 cazuri

Indicatii: Folosiţi implementările existente ataşate laboratorului!

4.1 Programare Liniară

Problemele de Programare Liniara (LP) au forma "standard":

$$\begin{aligned}\min_{x\in\mathbb{R}^n} \quad & c^T x \qquad\qquad (LP) \\ \text{s.t. } & Ax = b, \\ & x \geq 0\end{aligned}$$

unde observăm o funcţie obiectiv liniară şi constrângeri liniare de egalitate şi inegalitate.

Exerciţiu. Pentru $n = 2$, consideraţi un vector $c \in \mathbb{R}^2$ la alegere. Trasaţi multimile izonivel ale funţiei obiectiv folosind funcţia *contour* din exemplele precedente.

4.2 Programare Pătratică

Problemele de Programare Pătratică (QP) sunt problemele cu cost pătratic şi constrângeri liniare de forma:

$$\begin{aligned}\min_{x\in\mathbb{R}^n} \quad & \frac{1}{2}x^T H x + q^T x \qquad\qquad (QP) \\ \text{s.t. } & Ax = b, \\ & Cx \leq d\end{aligned}$$

Exerciţiu. Consideraţi matricile Hessiene:

$$H^1 = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}, \qquad H^2 = \begin{bmatrix}10 & 0\\0 & 10^{-5}\end{bmatrix}, \qquad H^3 = \begin{bmatrix}3 & 2\\2 & 3\end{bmatrix}$$

şi un vector $q \in \mathbb{R}^2$ la alegere. Trasaţi multimile izonivel ale funţiei obiectiv folosind funcţia *contour* din exemplele precedente.