Tehnici de optimizare
Laborator 3: Introducere in CVX, Control optimal

Acest laborator incepe cu o scurta prezentare a sistemului CVX, acesta preocupandu-se cu rezolvarea problemelor convexe. Dupa aceasta sectiune formulam o problema de control optimal pentru pendulul invers, in care am sarit peste pasul de modelare, nefiind obiectul cursului. Rezolvam problema cu doua metode invatate la curs si anume metoda barierei si cea a gradientului proiectat, iar solutiile obtinute sunt comparate cu cele gasite cu CVX si functiile oferite de Matlab. In cea de a treia sectiune, prezentam problema lantului suspendad pentru care am implementat gradientul proiectat si cel conditionat. In aceeasi maniera ca si in sectiunea precedenta, comparam cu solutiile produse de CVX si functiile Matlab.

1 Introducere CVX

CVX-ul este un sistem de modelare pentru construirea și rezolvarea problemelor convexe. Acesta este implementat in Matlab, transformand eficient Matlab-ul intr-un limbaj de modelare optimizat. CVX-ul accepta pe langa Matlab, doua *solutii gratuite*, **SeDuMi** [2] si **SDPT3** [3], acestea fiind incluse cu distribuția CVX. Alaturi de optiunile gratuite, CVX-ul accepta si doua *solvere comerciale*, **Gurobi** [4] si **MOSEK** [5].

1.1 Instalare

1. Se descarca pachetul standard CVX potrivit sistemului de operare utilizat de la urmatoarea adresa `http://cvxr.com/cvx/download/` [1]
2. Se dezarhiveaza intr-un director dorit (diferit de directorul toolbox al Matlab-ului)
3. Se porneste Matlab-ul;
4. Se comuta directorul curent in directorul unde am realizat dezarhivarea si se executa in consola comanda **cvx_setup**.

1.2 Elemente de baza

- Orice program CVX se scrie in interiorul unei functii Matlab.
- Programele CVX se delimiteaza de codul Matlab cu comenzile **cvx_begin** si **cvx_end**.

[1] La aceasta adresa gasiti si un ghid de instalare mai detaliat

- Valorile variabilelor create in portiunea de cod Matlab se pot folosi ca parametri in problemele de optimizare rezolvate cu CVX.
- Variabilele CVX se declara folosind comanda **variable** si specificarea dimensiunii variabilei
  - Exemple:
    Vectorul $x \in \mathbb{R}^n$: `variable x(n)`
    Matricea $A \in \mathbb{R}^{n \times m}$: `variable A(n,m)`
  - o varietate de optiuni aditionale pentru precizarea structurii matriceale sunt disponibile la adresa `http://cvxr.com/cvx/doc/basics.html`
    Exemplu: O matrice simetrica se declara
    `variable A(10,10) symmetric`
- La inceputul oricarui program CVX se definesc variabilele de decizie si dimensiunile acestora.
- Declararea *functiei obiectiv* a problemei de optimizare necesita precizarea tipului de problema (e.g. minimizare, maximizare) prin intermediul cuvintelor cheie **minimize** si **maximize**
  - Este necesar ca functia obiectiv sa fie convexa cand folosim minimize si concava cand folosim maximize. In caz contrar, pachetul CVX va furniza un mesaj de eroare corespunzator.
- *Constrangerile* suportate de modelele CVX sunt cele de *egalitate* (liniare) impuse prin operatorul == si de *inegalitate* impuse de operatorii <= si >=.
  - Pentru *constrangeri de tip box* (lant de inegalitati) este disponibila sintaxa $l <= x <= u$.

Pentru asimilarea facila a acestor reguli sa ne uitam la exemplele de implementare in CVX din urmatoarea sectiune.

1.3 Exemple

Metoda celor mai mici patrate constransa
Fie problema de optimizare cu constrangeri:

$$\min_{x \geq 1} \|Ax - b\|$$

. Luand o matrice A de dimensiune $m \times n$ (ex: $A = \text{randn}(m, n)$) si un vector b in Matlab, avem urmatorul cod CVX pentru a gasi solutia problemei:

```
m =5; n=10;
A = randn(m,n);
b = randn(m,1);
cvx_begin
variable x(n)
```

```
minimize(norm(A*x-b))
subject to
x >= 1
cvx_end
```

Observam in fereastra de comanda urmatoarele informatii odata ce am rulat codul de mai sus:

```
Calling SDPT3 4.0: 16 variables, 5 equality constraints
------------------------------------------------------------

 num. of constraints =  5
 dim. of socp   var  =  6,   num. of socp blk  =  1
 dim. of linear var  = 10
```

Remarcam ca suntem informatii ce solver a fost chemat. In cazul de mai sus a fost SDPT3. Acest lucru se poate schimba cu comanda **cvx_solver** urmat de numele solver-ului. Putem observa de asemenea ca numarul de constrangeri este egal cu m si anume 5, iar dimensiunea variabilei liniare cu n, respectiv 10.

```
it pstep dstep pinfeas dinfeas  gap      prim-obj      dual-obj    cputime
-------------------------------------------------------------------
 0|0.000|0.000|7.9e+00|1.6e+01|1.0e+03| 6.343522e+00  0.000000e+00| 0:0:00| chol  1  1
 1|1.000|0.236|4.0e-06|1.2e+01|9.2e+02| 7.245086e+01  9.552675e+00| 0:0:00| chol  1  1
```

Dupa aceasta sectiune sunt afisate informatii detaliate ale iteratiilor, cum ar fi: valoarea pasului la problema primal si duala, valoarea functiei obiectiv primala si duala, timpul CPU, etc De asemenea este afisat criteriul de oprire, si tot odata precizia cu care a fost calculata solutia. Acest parametru se poate schimba din setari sau adaugand comanda **cvc_precision high/medium/best** sau **low**. In functie de pozitionarea aceasteia, ea are efect global (daca este inainte de cvx_begin) sau local (intre cvx_begin si cvx_end)

```
  stop: max(relative gap, infeasibilities) < 1.49e-08
-------------------------------------------------------------------
 number of iterations   = 12
 primal objective value =  8.45423413e-09
 dual   objective value =  1.09498700e-09
 gap := trace(XZ)       = 1.42e-08
 relative gap           = 1.42e-08
 actual relative gap    = 7.36e-09
 rel. primal infeas (scaled problem)   = 5.24e-14
 rel. dual     "        "       "      = 4.89e-11
 rel. primal infeas (unscaled problem) = 0.00e+00
 rel. dual     "        "       "      = 0.00e+00
 norm(X), norm(y), norm(Z) = 1.3e+02, 2.8e-10, 1.0e+00
 norm(A), norm(b), norm(C) = 7.7e+00, 7.3e+00, 2.0e+00
 Total CPU time (secs)  = 0.59
 CPU time per iteration = 0.05
 termination code       =  0
 DIMACS: 7.4e-14  0.0e+00  4.9e-11  0.0e+00  7.4e-09  1.4e-08
-------------------------------------------------------------------
```

Detalierea iteratiilor, este urmata de un rezumat ce cuprinde: numarul total de iteratii (in cazul nostru au fost 12), valoarea optima a functiei obiectiv pentru problema primala si

duala, timpul CPU total in secunde, codul de terminare (0) si valoarea relativa a diferentei duale si asa mai departe.

```
------------------------------------------------------------
Status: Solved
Optimal value (cvx_optval): +8.45423e-09
```

In cele din urma starea problemei este afisata, in cazul de fata fiind rezolvata. Gasiti la adresa aceasta `http://cvxr.com/cvx/doc/solver.html#interpreting-the-results` interpretarea statusului in detaliu pentru restul starilor (de exemplu: Failed, Unbounded, etc). De asemenea se evidentiaza valoarea optima, aceasta regasindu-se si in rezumat.

In final daca nu se doreste afisarea acestor informatii, ele se pot suprima cu ajutorul cuvantului cheie **quiet**: `cvx_begin quiet ... cvx_end`

Stabilitatea sistemelor

Fie un sistem liniar dinamic $\dot{x} = Ax$. Dorim sa investigam daca el este stabil. Pentru aceasta trebuie sa verificam daca exista matricea X simetrica a.i.: $A^TX+XA \prec 0, \quad X \succ 0$

Cele 2 inegalitati stricte sunt omogene in X, deci problema poate fi formulata echivalent ca si:

$$A^TX + XA + I_n \preccurlyeq 0, \quad X \succcurlyeq I_n.$$

Deci obtinem o problema de programare semidefinita convexa [2] (*eng. semidefinite programming (SDP)*).

Amintim forma standard:

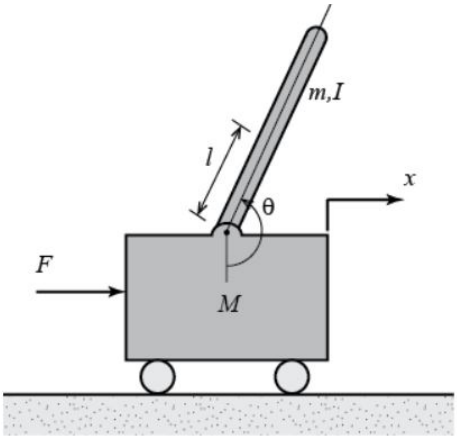
$$(SDP): \qquad \min_{x\in\mathbb{R}^n} \; c^Tx, \quad \text{s.l.:} \quad LMI(x) \preccurlyeq 0, \; Ax - b = 0,$$

Putem rezolva aceasta problema in CVX, dupa cum urmeaza:

```
% A=eigenvalues uniform logarithmic spaced [-10^-1;-10^1]
n = 10;A=diag(-logspace(-0.5,1,n));U=orth(randn(n,n));
A=U'*A*U;
cvx_begin sdp
variable X(n,n) symmetric %Obs: diagonal,...
minimize(trace(X)) %Obs: poate lipsi aceasta linie
A'*X + X*A + eye(n) <= 0,
X >= eye(n)
cvx_end
```

2 Control optimal

Sistemul din acest exemplu consta dintr-un pendul inversat montat pe un carucior motorizat. Popularitatea sistemului deriva in parte din faptul ca este instabila fara control, adica pendulul va cadea pur și simplu daca nu este mișcat caruciorul pentru a-l echilibra. In plus, dinamica sistemului este neliniară. Obiectivul sistemului de control este de a echilibra pendulul inversat prin aplicarea unei forțe pe caruciorul de care pendulul este



atasat. Un exemplu din lumea reala care se refera direct la acest sistem de pendul inversat este controlul atitudinii unei rachete rapel la decolare.

Notam:

- M - masa carucirului-ului = 0,5 kg
- m - masa pendulului = 0,2 kg
- b - coeficientul de frecare pentru carucior =0,1 N/m/sec
- l - lungimea până la centrul de masă al pendulului = 0,3 m
- I - momentul masic de inertie a pendulului = 0,006 kg m^2
- F - forta aplicata caruciorului
- x - coordonata de pozitie a caruciorului
- ϕ - unghiul pendulului

Utilizand legile lui Newton si liniariand ecuatiile dinamicii gasite obtinem urmatoarea reprezentare pe stare a sistemului:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-\left(I+ml^2\right)b}{I(M+m)+Mml^2} & \frac{m^2gl^2}{I(M+m)+Mml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mlb}{I(M+m)+Mml^2} & \frac{mgl(M+m)}{I(M+m)+Mml^2} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{I+ml^2}{I(M+m)+Mml^2} \\ 0 \\ ml \end{bmatrix} u \Leftrightarrow \dot{z}_{t+1} = Az_t + Bu_t$$

Pentru acest sistem formulam in continuare prolema de control optimal pe orizont finit N pentru urmarirea unei referinte date.

Pentru aceasta consideram costuri de etapa: $\ell_t^z(z_t) = 1/2 \left\| z_t - z_t^{ref} \right\|_{Qt}^2$, $\ell_t^u(u_t) = 1/2 \left\| u_t - u_t^{ref} \right\|_{R_t}^2$, unde $Q_t \succeq 0$ si $R_t \succ 0$. (Notatie: $\left\| x - x^{ref} \right\|_Q^2 = \left(x - x^{ref}\right)^T Q \left(x - x^{ref}\right)$)
De asemenea considerand dimensiunile caruciorului trebuie impusa o restictionata forta aplicate asupra acestuia. Astfel avem ca: $-3 \leq u \leq 3$

[2]probleme convexe unde multimea fezabila este descrisa de LMI-uri

Deci avem urmatoare formulare a problemei de control:

$$\begin{aligned}
&\min_{z_t,u_t} \sum_{t=1}^{N} \ell_t^z\left(z_t\right) + \sum_{t=0}^{N-1} \ell_t^u\left(u_t\right) \\
&\text{s.l.:} \quad z_{t+1} = A_t z_t + B_t u_t \quad \forall t = 0, \ldots, N-1, \quad z_0 \text{ dat.} \\
&\qquad -3 \leq u_t \leq 3
\end{aligned}$$

Observam ca variabila de decizie pentru problema de mai sus este:

$$x = \left[\begin{array}{cccccc} u_0^T & z_1^T & \ldots & u_{N-1}^T & z_N^T \end{array}\right]^T \in \mathbb{R}^{N(n_z+n_u)}$$

Pentru simplitate vom considera in cele ce urmeaza ca orizontul de predictie N= 3.

Putem sa restrangem variabila si sa o scriem doar in functie de intrari, utilizandu-ne de dinamica sistemului:

$$\begin{aligned}
z_1 &= Az_0 + Bu_0 \\
z_2 &= Az_1 + Bu_1 = A^2 z_0 + ABu_0 + Bu_1 \\
z_3 &= Az_2 + Bu_2 = A^3 z_0 + A^2 Bu_0 + ABu_1 + Bu_2
\end{aligned}$$

Eliminand starile obtinem ca variabila de decizie:

$$x = \left[\begin{array}{ccc} u_0^T & u_2^T & u_2^T \end{array}\right]^T$$

De asemenea notand $\bar{z} = [z_1^T \ z_2^T \ z_3^T]^T$ putem scrie ecuatia de mai sus in urmatoarea forma compacta

$$\bar{z} = \overline{AB}x + A_p z_0$$

unde:

$$\overline{AB} = \left[\begin{array}{ccc} B & 0 & 0 \\ AB & B & 0 \\ A^2B & AB & B \end{array}\right], A_p = \left[\begin{array}{c} A \\ A^2 \\ A^3 \end{array}\right]$$

Prin prisma celor de mai sus, putem rescrie functia cost pentru stare:

$$\begin{aligned}
\frac{1}{2}\sum_{j=1}^{3} \left\|z_j - z_j^{\text{ref}}\right\|_{Q_j}^2 &= \frac{1}{2}\left\|\bar{z} - \bar{z}^{\text{ref}}\right\|_{\bar{Q}}^2 = \frac{1}{2}\left\|\overline{AB}x + A_p z_0 - \bar{z}^{\text{ref}}\right\|_{\bar{Q}}^2 \\
&= \frac{1}{2}x^T \overline{AB}^T \bar{Q}\overline{AB}x + \left(z_0^T A_p^T \bar{Q}\overline{AB} - \left(\bar{z}^{\text{ref}}\right)^T \bar{Q}\overline{AB}\right) z
\end{aligned}$$

unde $\bar{Q} = \text{diag}\left(Q_1, \ldots, Q_2, Q_3\right)$ si $\bar{z}^{\text{ref}} = \left[z_1^{\text{ref}} \ldots z_3^{\text{ref}}\right]^T$

Procedand in aceasi maniera si pentru intrare, gasim:

$$\begin{aligned}
\sum_{j=0}^{2} \left\|u(j) - u^{\text{ref}}(j)\right\|_{R_j}^2 &= \left\|x - \bar{x}^{\text{ref}}\right\|_{\bar{R}}^2 \\
&= \frac{1}{2}x^T \bar{R}x - \left(\bar{x}^{\text{ref}}\right)^T \bar{R}x + \frac{1}{2}\left(\bar{x}^{\text{ref}}\right)^T \bar{R}x^{\text{ref}}
\end{aligned}$$

unde $\bar{R} = \text{diag}\left(R_0, \ldots, R_2\right)$ si $\bar{x}^{\text{ref}} = \left[x_0^{\text{ref}} \ \ldots u_2^{\text{ref}}\right]$

In cele de urma putem scrie functia obiectiv ca:$\frac{1}{2}x^THx+h^Tx$, unde $H = \bar{R}+\overline{AB}^T\bar{Q}\overline{AB}$ si $h = \overline{AB}^T\bar{Q}A_pz_0 - \overline{AB}^T\bar{Q}\bar{z}^{\text{ref}} - \bar{R}\bar{x}^{\text{ref}}$

Eliminand starile, am mai ramas doar cu constrangerea pe intrare: $-3\,ones(3,1) \leq x \leq 3\,ones(3,1)$

In concluzie avem de rezolvat urmatoarea problema QP:

$$\begin{aligned} &\min_x \frac{1}{2}x^THx + h^Tx \\ &\text{s.l. : } Cx \leq d \end{aligned}$$

A se observa ca reformularea problemei cu metoda eliminarea starilor obtinem o problema de optimizare care are doar constrangeri de inegalitate.

Amintim pentru rezolvarea acestei probleme doi algoritmi si vom compara rezultatele acestora cu ce ne va furniza cvx-ul si functia quadprog.

- **Metoda gradient proiectat**

  Fie: $\min_{x\in X} f(x)$

  Iteratia: $x_{k+1} = [x_k - \alpha_k \nabla f(x_k)]_X$

  - Pasul α_k se alege constant $1/L$ sau cu metoda ideala.
  - Multimea fezabila in cazul nostru este $X = -3\,ones(3,1) \leq x \leq 3\,ones(3,1)$. Astfel ca proiectia pe X in acest caz va avea formula:

    $$[y]_{[l,u]} = \min(u, \max(l, y))$$

  - Cum $f(x) = \frac{1}{2}x^THx - h^Tx \Rightarrow \nabla f(x) = Hx + h$

- **Metoda bariera**

  Fie: $\min\limits_{x\in\mathbb{R}^n} \quad f(x) \quad \text{s.l.} \quad g(x) \leq 0,\ Ax = b.$

  unde f, g sunt functii convexe. Formulam problema bariera:

  $$\min_{x\in\mathbb{R}^n} \quad f(x) - \tau\sum_{i=1}^{m}\log(-g_i(x)) \quad \text{s.l.} \quad Ax = b. \tag{1}$$

  Alegem x_0 strict fezabil, $\tau_0 > 0, \sigma < 1$ si $\epsilon > 0$.

  Atata timp cat $m\tau_k \geq \epsilon$ repeta:

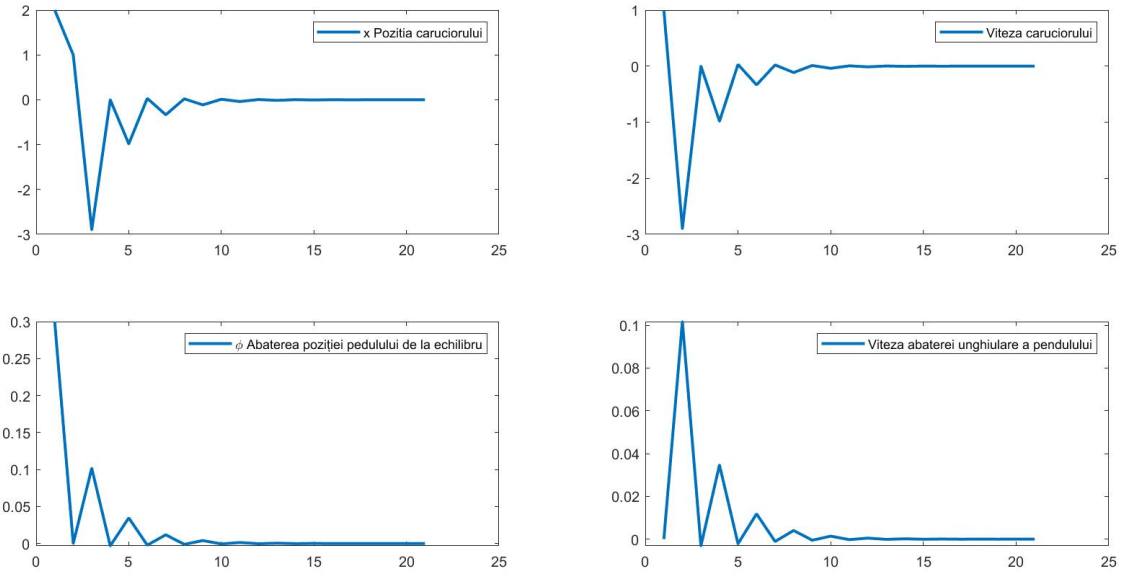
  1. Calculeaza
     $$x_{k+1} = x(\tau_k) \tag{2}$$
     pornind din punctul initial x_k ("warm start");
  2. Descreste parametrul $\tau_{k+1} = \sigma\tau_k$.

  $\Rightarrow$ Pentru a determina x_{k+1} din expresia 2 rezolvam (e.g. prin Metoda Newton) reformularea cu bariera logaritmica cu parametrul τ_k 1, pornind din punctul initial x_k;

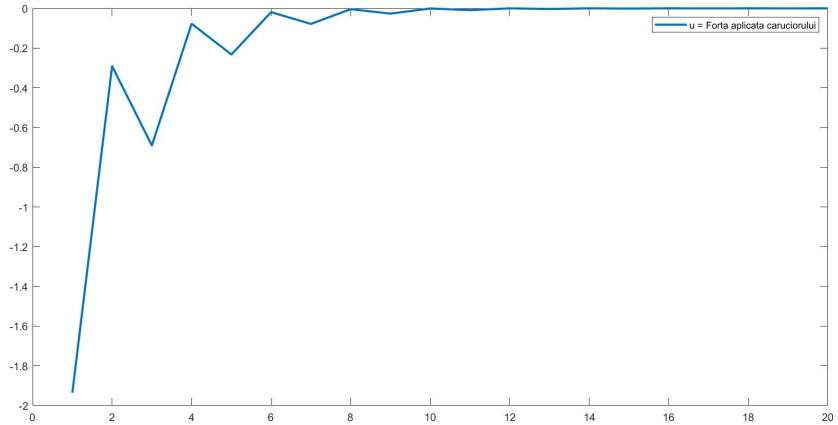
  $\Rightarrow$ Dupa k iteratii avem $f(x_k) - f^* \leq m\tau_0\sigma^k$.

Rezultate numerice

Luam pentru costul functiei: $Q = I_4$, dar 10 pe pozitiile (1,1) si (3,3) si $R = 1$. Alegem starea intiala: $z_0 = [2; 1; 0.3; 0]$, $\tau = 1$, $\sigma = 0.6$, si $\epsilon = 0.0001$.
Graficul variabilei de stare a sistemului



Graficul intrarii



3 Tema

Inlocuiti in template-ul atasat laboratorului, functia quadprog din matlab cu:

- (2p)Metoda bariera
- (2p)Metoda gradient proiectat
- (1p)Versiunea cvx

References

[1] http://web.cvxr.com/cvx/doc/index.html

[2] http://sedumi.ie.lehigh.edu/

[3] R.H. Tütüncü, K.C. Toh, and M.J. Todd. Solving semidefinite-quadratic-linear programs using SDPT3

[4] https://www.gurobi.com/

[5] https://www.mosek.com/

[6] Modelarea pundulului invers pe un carucior