Controllable Sets, Invariant Sets, and MPC Feasibility and Stability

These notes serve the following purposes,

1. Clarify the properties and computation of N-step controllable sets, positive invariant and control invariant sets.
2. Show how invariant sets can be used to design persistently feasible and asymptotically stable MPC controllers.

We will use numerical examples with a second order unstable system

$$x(t+1) = Ax + Bu = \begin{bmatrix} 1.5 & 0 \\ 1 & -1.5 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t) \tag{1}$$

subject to the input and state constraints

$$u(t) \in \mathcal{U} = \{u \ : \ -5 \le u \le 5\}, \ \forall t \ge 0 \tag{2a}$$

$$x(t) \in \mathcal{X} = \left\{ x \ : \ \begin{bmatrix} -10 \\ -10 \end{bmatrix} \le x \le \begin{bmatrix} 10 \\ 10 \end{bmatrix} \right\}, \ \forall t \ge 0. \tag{2b}$$

---

1 Controllable Sets

Recall our definition of N-Step Controllable Set $\mathcal{K}_N(\mathcal{S})$.

For a given target set $\mathcal{S} \subseteq \mathcal{X}$, the N-step controllable set $\mathcal{K}_N(\mathcal{S})$ of the system (1) subject to the constraints (2) is defined recursively as:

$$\mathcal{K}_j(\mathcal{S}) \triangleq \textbf{Pre}(\mathcal{K}_{j-1}(\mathcal{S})) \cap \mathcal{X}, \ \ \mathcal{K}_0(\mathcal{S}) = \mathcal{S}, \quad j \in \{1, \ldots, N\} \tag{3}$$

All states x_0 of the system (1) belonging to the N-Step Controllable Set $\mathcal{K}_N(\mathcal{S})$ can be driven, by a suitable control sequence, to the target set $\mathcal{S}$ in N steps, while satisfying input and state constraints.

Note that the computation of the set does not provide the input sequence driving the system states to $\mathcal{S}$.

1.1 How to compute N-step controllable sets?

From the previous definition we have that a 1-step controllable set to the set $\mathcal{S}$ is computed as

$$\mathcal{K}_1 = \textbf{Pre}(\mathcal{S}) \cap \mathcal{X} \tag{4}$$

the 2-step as

$$\mathcal{K}_2 = \textbf{Pre}(\mathcal{K}_1) \cap \mathcal{X} \tag{5}$$

and so on. The operation $\textbf{Pre}(\mathcal{K}_1)$ was discussed last week for linear systems subject to linear constraints. We said that if

$$\mathcal{S} = \{x \mid Hx \le h\}, \quad \mathcal{U} = \{u \mid H_u u \le h_u\}, \tag{6}$$

The Pre set is

$$\mathrm{Pre}(\mathcal{S}) = \left\{ x \in \mathbb{R}^n \,|\, \exists u \in \mathbb{R} \text{ s.t. } \begin{bmatrix} HA & HB \\ 0 & H_u \end{bmatrix} \begin{pmatrix} x \\ u \end{pmatrix} \le \begin{bmatrix} h \\ h_u \end{bmatrix} \right\}$$

which is the projection onto the x-space (with dimension $\mathbb{R}^n$) of the polyhedron

$$\mathcal{XU} := \{ \begin{bmatrix} HA & HB \\ 0 & H_u \end{bmatrix} \begin{pmatrix} x \\ u \end{pmatrix} \le \begin{bmatrix} h \\ h_u \end{bmatrix} \}.$$

The version of the matlab code below works also for non-full dimensional set $\mathcal{S}$.

```
function PreS=Pre(A,B,S,U)
    nx=size(A,2);
    nu=size(B,2);
    XU_H=[S.H(:,1:nx)*A          S.H(:,1:nx)*B;
          zeros(size(U.H,1),nx) U.H(:,1:nu)];
    XU_h=[S.H(:,nx+1);U.H(:,nu+1)];
    XU=Polyhedron('H',[XU_H,XU_h],...
        'He',[S.He(:,1:nx)*A S.He(:,1:nx)*B S.He(:,nx+1)]);
    PreS=XU.projection(1:nx);
end
```

So we can compute N-step controllable sets as

```
% Unstable System
% x(k+1) = A*x(k) + B*u(k)
A=[1.5 0;1 -1.5];
B=[1;0];

% constraints on inputs and states
model.u.min = -5;
model.u.max = 5;
model.x.min = [-10;-10];
model.x.max = [ 10; 10];

% constraint sets represented as polyhedra
X = Polyhedron('lb',model.x.min,'ub',model.x.max);
U = Polyhedron('lb',model.u.min,'ub',model.u.max);

% target set
S=Polyhedron('He',[eye(2), zeros(2,1)]);
%S=X;

PreS=Pre(A,B,S,U);
for j=1:6
    K(j) = PreS.intersect(X);
    PreS=Pre(A,B,K(j),U);
end
```

```
% plot the target set
plot(S,'color','green');
hold on
% plot the controllable sets
for j=1:numel(K)
    plot(K(j),'alpha',0.1);
    pause
end
```

1.2 Evolution of N-step controllable sets

Execute the previous code for two different target sets. First use $\mathcal{S}$ equal to the origin

```
S=Polyhedron('He',[eye(2), zeros(2,1)]).
```

Then, consider $\mathcal{S}$ equal to the state constraints set $\mathcal{X}$

```
S=X.
```

Observe how $\mathcal{K}_j$ evolve in both cases and try to explain the observed behaviour.

1.3 N-step controllable sets and MPC initial feasible set

In the past lectures we studied Constrained Finite Time Optimal Control problem (CFTOC) of the form

$$\begin{aligned} J_0^*(x(0)) = \quad \min_{U_0} \quad & J_0(x(0), U_0) \\ \text{subj. to} \quad & x_{k+1} = Ax_k + Bu_k,\ k = 0, \ldots, N-1 \\ & x_k \in \mathcal{X},\ u_k \in \mathcal{U},\ k = 0, \ldots, N-1 \\ & x_N \in \mathcal{X}_f \\ & x_0 = x(0) \end{aligned} \tag{7}$$

and denoted with $\mathcal{X}_0$ the set of initial states $x(0)$ for which the optimal control problem (7) is feasible:

$$\begin{aligned} \mathcal{X}_0 = \quad & \{x_0 \in \mathbb{R}^n |\ \exists (u_0, \ldots, u_{N-1}) \text{ such that } x_k \in \mathcal{X},\ u_k \in \mathcal{U}, \\ & k = 0, \ldots, N-1,\ x_N \in \mathcal{X}_f,\ \text{where } x_{k+1} = Ax_k + Bu_k\} \end{aligned} \tag{8}$$

The set $\mathcal{X}_0$ is nothing but the N-step controllable set to $\mathcal{X}_f$ for system $x_{k+1} = Ax_k + Bu_k$ subject to constraints $x_k \in \mathcal{X},\ u_k \in \mathcal{U}$. The definition (8) is the "BATCH" version of the the recursive definition of the N-step controllable set $\mathcal{K}_N(\mathcal{X}_f)$.

You now know two approaches for computing $\mathcal{X}_0$.

- One approach is to transform the CFTOC problem into the QP

$$J_0^*(x(0)) = \min_{U_0} \quad [U_0'\ x(0)'] \left[\begin{smallmatrix} H & F' \\ F & Y \end{smallmatrix}\right] [U_0'\ x(0)']' \tag{9}$$

$$\text{subj. to} \quad G_0 U_0 \le w_0 + E_0 x(0) \tag{10}$$

  and obtain $\mathcal{X}_0$ as the projection of the polyhedron $[G_0 \quad E_0] \begin{bmatrix} U_0 \\ x(0) \end{bmatrix} \le w_0$ on the $x(0)$ space.

- The other option is to compute the $\mathcal{X}_0$ as the N-step controllable set, $\mathcal{X}_0 = \mathcal{K}_N(\mathcal{X}_f)$.

---

2 Invariant Sets

2.1 How to compute positive invariant sets

A set $\mathcal{O} \subseteq \mathcal{S}$ is said to be a positive invariant set for the autonomous system $x(t+1) = Ax(t)$ subject to the constraints $x(t) \in \mathcal{S}$, if

$$x(0) \in \mathcal{O} \quad \Rightarrow \quad x(t) \in \mathcal{O}, \quad \forall t \in \mathbb{N}^+$$

We introduced a simple algorithm for computing the Maximal Positive Invariant Set $\mathcal{O}_\infty$ (the largest positive invariant set in $\mathcal{S}$):

1. LET $\Omega_0 = \mathcal{S}$
2. LET $\Omega_{k+1} = \textbf{Pre}(A, \Omega_k) \cap \Omega_k$
3. IF $\Omega_{k+1} = \Omega_k$ THEN $\mathcal{O}_\infty \leftarrow \Omega_{k+1}$
4. ELSE GOTO 2

Below you find the Matlab implementation. Notice that the **Pre** operator refers to the definition for autonomous systems. In the next code it is implemented as the function "Pre_Aut".

```
function PreS=Pre_Aut(A,S)
    % works with polytope which are also not full dimensional
    nx=size(A,2);
    PreS=Polyhedron('H',[S.H(:,1:nx)*A S.H(:,nx+1)],...
        'He',[S.He(:,1:nx)*A S.He(:,nx+1)]);
end

function [Oinf,converged]=max_pos_inv(A,S)
    maxIterations=500;
    Omega_i = S; % initialization
    for i = 1:maxIterations
        % compute backward reachable set
        P = Pre_Aut(A,Omega_i);
        % intersect with the state constraints
        P = P.intersect(Omega_i).minHRep();
        if P==Omega_i
            Oinf=Omega_i;
            break
        else
            Omega_i = P;
        end
    end
    if i==maxIterations,
        converged=0;
```

```
    else
        converged=1;
    end
end


%% Maximal Invariant Set Computation
% first design a stabilizing control law:
K=place(A,B,[0.1 0.2]);
% closed loop system
Acl=A-B*K;
% remember to convert input constraints in state constraints
S=X.intersect(Polyhedron('H',[-U.H(:,1:nu)*K U.H(:,nu+1)]))
Oinf=max_pos_inv(Acl,S)
figure
plot(Oinf,'color','red','alpha',0.5)
```

Notice that the above Matlab code computes the set $\mathcal{O}_\infty$ for system (1) when it is controlled by $u = -Kx$, subject to the constraints (2).

Clearly the maximal positive invariant set $\mathcal{O}_\infty$ will be function of the feedback control law K. As commented in the code, DO NOT forget to convert input constraints in state constraints.

Try to change the control law K in the algorithm above and interpret the results. Try the controllers K=place(A,B,[0.7 0.8]) and K=place(A,B,[0.1 0.2]).

2.2 How to compute control invariant sets

A set $\mathcal{C} \subseteq \mathcal{X}$ is said to be a control invariant set if

$$x(t) \in \mathcal{C} \quad \Rightarrow \quad \exists u(t) \in \mathcal{U} \text{ such that } f(x(t), u(t)) \in \mathcal{C}, \quad \forall t \in \mathbb{N}^+$$

In class we introduced a simple algorithm for computing the Maximal Control Invariant Set $\mathcal{C}_\infty$ (the largest control invariant):

1. LET $\Omega_0 = \mathcal{X}$
2. LET $\Omega_{k+1} = \textbf{Pre}(A, B, \Omega_k) \cap \Omega_k$
3. IF $\Omega_{k+1} = \Omega_k$ THEN $\mathcal{C}_\infty \leftarrow \Omega_{k+1}$
4. ELSE GOTO 2

Below you find the Matlab implementation.

```
function [Cinf,converged]=max_cntr_inv(A,B,X,U)
    maxIterations=500;
    Omega0 = X; % initialization
    for i = 1:maxIterations
        % compute backward reachable set
        P = Pre(A,B,Omega0,U);
        % intersect with the state constraints
```

```
        P = P.intersect(Omega0).minHRep();
        if P==Omega0
            Cinf=Omega0;
            break
        else
            Omega0 = P;
        end
    end
    if i==maxIterations,
        converged=0;
    else
        converged=1;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Maximal Control Invariant Set Computation
Cinf=max_cntr_inv(A,B,X,U)
```

The maximal control invariant set $\mathcal{C}_\infty$ does not depend on a specific feedback control law. Try to compare $\mathcal{C}_\infty$ with the $\mathcal{O}_\infty$ sets compute before. Interpret the results.

---

3 Persistently Feasible and Stable MPC

Consider the problem of regulating to the origin the discrete-time linear time-invariant system

$$x(t+1) = Ax(t) + Bu(t), \tag{11}$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$ are the state and input vectors, respectively, subject to the constraints

$$x(t) \in \mathcal{X},\ u(t) \in \mathcal{U},\ \forall t \geq 0, \tag{12}$$

where the sets $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{U} \subseteq \mathbb{R}^m$ are polyhedra. MPC approaches such a constrained regulation problem in the following way. Assume that a full measurement or estimate of the state $x(t)$ is available at the current time t. Then the finite time optimal control problem

$$\begin{array}{rll} J_t^*(x(t)) = & \min_{U_{t\to t+N|t}} & p(x_{t+N|t}) + \sum_{k=0}^{N-1} q(x_{t+k|t}, u_{t+k|t}) \\ & \text{subj. to} & x_{t+k+1|t} = Ax_{t+k|t} + Bu_{t+k|t},\ k = 0, \ldots, N-1 \\ & & x_{t+k|t} \in \mathcal{X},\ u_{t+k|t} \in \mathcal{U},\ k = 0, \ldots, N-1 \\ & & x_{t+N|t} \in \mathcal{X}_f \\ & & x_{t|t} = x(t) \end{array} \tag{13}$$

is solved at time t, where $U_{t\to t+N|t} = \{u_{t|t}, \ldots, u_{t+N-1|t}\}$ and where $x_{t+k|t}$ denotes the state vector at time $t+k$ predicted at time t obtained by starting from the current state $x_{t|t} = x(t)$ and applying to the system model

$$x_{t+k+1|t} = Ax_{t+k|t} + Bu_{t+k|t}$$

the input sequence $u_{t|t}, \ldots, u_{t+k-1|t}$. Often the symbol $x_{t+k|t}$ is read as “the state x at time $t+k$ predicted at time t”. Similarly, $u_{t+k|t}$ is read as “the input u at time $t+k$ computed at time t”.

Let $U^*_{t\to t+N|t} = \{u^*_{t|t}, \ldots, u^*_{t+N-1|t}\}$ be the optimal solution of (13) at time t and $J^*_t(x(t))$ the corresponding value function. Then, the first element of $U^*_{t\to t+N|t}$ is applied to system (11)

$$u(t) = u^*_{t|t}(x(t)). \tag{14}$$

The optimization problem (13) is repeated at time $t+1$, based on the new state $x_{t+1|t+1} = x(t+1)$, yielding a *moving* or *receding horizon* control strategy.

Let $f_t : \mathbb{R}^n \to \mathbb{R}^m$ denote the *receding horizon* control law that associates the optimal input $u^*_{t|t}$ to the current state $x(t)$, $f_t(x(t)) = u^*_{t|t}(x(t))$. Then the closed-loop system obtained by controlling (11) with the RHC (13)-(14) is

$$x(k+1) = Ax(k) + Bf_k(x(k)) \triangleq f_{cl}(x(k), k),\ \ k \geq 0. \tag{15}$$

We also noticed that for linear time-invariant systems and time invariant cost and constraints, the control law (14)

$$u(t) = f_0(x(t)) = u^*_0(x(t)) \tag{16}$$

and closed-loop system (15)

$$x(k+1) = Ax(k) + Bf_0(x(k)) = f_{cl}(x(k)),\ \ k \geq 0 \tag{17}$$

are time-invariant.

3.1 MPC Main Theorem

In class we introduced the two concepts of

- Persistent feasibility.
- Asymptotic stability of the origin in a region called "domain of attraction".

We also proved the following fundamental theorem:

Theorem 1. *Assume that*

(A0) *The stage cost $q(x,u)$ and terminal cost $p(x)$ are continuous and positive definite functions.*

(A1) *The sets $\mathcal{X}$, $\mathcal{X}_f$ and $\mathcal{U}$ contain the origin in their interior and are closed.*

(A2) *$\mathcal{X}_f$ is control invariant, $\mathcal{X}_f \subseteq \mathcal{X}$.*

(A3) $$\min_{v\in\mathcal{U},\ Ax+Bv\in\mathcal{X}_f} (-p(x) + q(x,v) + p(Ax+Bv)) \leq 0,\ \forall x \in \mathcal{X}_f.$$

Then,

- *the closed-loop system (17) is persistently feasible in $\mathcal{X}_0$,*

- *the origin of the closed-loop system (17) is asymptotically stable with domain of attraction* $\mathcal{X}_0$.

Assumptions A0 and A1 are standard. We have shown that the simplest way to satisfy Assumptions A2 and A3 is to pick $\mathcal{X}_f = 0$ and any positive semi-definite terminal cost $p(x) = x'Px$. Next we are going to discuss other two options for selecting $\mathcal{X}_f$ and $p(x)$.

3.2 Using LQR

In this approach we design a stabilizing feedback controller for system (1) by using the infinite time, unconstrained linear quadratic regulator (LQR) F_∞ with the same weights Q and R of the desired MPC tuning.

```
[Finf,Pinf]=dlqr(A,B,Q,R);
```

Then we follow the approach of Section 2.1 to compute the maximal positive invariant set $\mathcal{O}_\infty$ for system (1) when controlled by the LQR controller $u = -F_\infty x$, subject to the constraints (2).

```
% closed loop system
Acl=A-B*Finf;
% remeber to convet input constraits in state constraints
S=X.intersect(Polyhedron('H',[-U.H(:,1:nu)*Finf U.H(:,nu+1)]))
Oinf=max_pos_inv(Acl,S)
```

Finally, in the CFTOC problem (13) solved by the MPC at each step we use $\mathcal{X}_f = \mathcal{O}_\infty$ and $p(x) = x'P_\infty x$ where P_∞ is LQR the infinite time cost. With this choice we satisfy assumption $A2$ (immediate to prove) and assumption $A3$ (proof next).

```
% x(k+1) = A*x(k) + B*u(k)
A=[1.5 0;1 -1.5];
B=[1;0];
nu=size(B,2);
nx=size(A,2);

% constraints on inputs and states
model.u.min = -5;
model.u.max = 5;
model.x.min = [-10;-10];
model.x.max = [ 10; 10];

% constraint sets represented as polyhedra
X = Polyhedron('lb',model.x.min,'ub',model.x.max);
U = Polyhedron('lb',model.u.min,'ub',model.u.max);

% stage cost x'Qx+u'Ru, MPC horizon is N
Q=eye(2);
R=1;
N = 3;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Maximal Invariant Set Computation
```

```
% first design a stabilizing control law via LQR
[Finf,Pinf]=dlqr(A,B,Q,R);
% closed loop system
Acl=A-B*Finf;
S=X.intersect(Polyhedron('H',[-U.H(:,1:nu)*Finf U.H(:,nu+1)]));
Oinf=max_pos_inv(Acl,S);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Compute the set of initial feasible states form MPC
Kc(N)=Oinf;
for j=N-1:-1:1
    Kc(j)=Pre(A,B,Kc(j+1),U);
end
X0=Pre(A,B,Kc(1),U);
plot(X0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% MPC control
x0=[-6;-3];
simsteps = 25;
xsim = zeros(2,simsteps+1);
usim = zeros(1,simsteps);
xsim(:,1) = x0;
options = sdpsettings('solver','quadprog');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Setup the CFTOC
yalmip('clear')
x = sdpvar(2,N+1);
u = sdpvar(1,N);
%set terminal constraint
constr = [Oinf.H(:,1:nx)*x(:,N+1)<=Oinf.H(:,nx+1)];
%set terminal cost
cost = x(:,N+1)'*Pinf*x(:,N+1);
for k = 1:N
    constr = [constr, x(:,k+1) == A*x(:,k) + B*u(:,k),...
              model.u.min <= u(:,k),u(:,k) <= model.u.max,...
              model.x.min <= x(:,k+1),x(:,k+1)<=model.x.max];
    cost = cost + x(:,k)'*Q*x(:,k) + u(:,k)'*R*u(:,k);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Run closed-loop with MPC
figure
for t = 1:simsteps
    optimize([constr, x(:,1) == xsim(:,t)],cost,options)
    xdata = double(x);
    udata = double(u);
    xsim(:,t+1) = xdata(:,2);
    usim(:,t) = udata(1);
    % Plot Open Loop
    %plot(x(1,:),x(2,:),'r--')
    %hold on
    %pause(0.1)
end
% Plot Closed Loop
plot(xsim(1,:),xsim(2,:),'bo-')
```

Why with this design Assumption A3 is satisfied?
Since the control law is fixed, Assumption A3 becomes

$$-x'P_\infty x + x'Qx + x'F_\infty' R F_\infty x + (Ax - BF_\infty x)' P_\infty (Ax - BF_\infty x).. \leq 0, \ \forall x \in \mathcal{X}_f$$

which can be rewritten as:

$$x'(-P_\infty + Q + F_\infty'(B'P_\infty B + R)F_\infty + A'P_\infty A - 2F_\infty' B' P_\infty A)x \leq 0, \ \forall x \in \mathcal{X}_f.$$

Recall from LQR (slide 3.34) that

$$F_\infty = (B'P_\infty B + R)^{-1} B' P_\infty A.$$

Therefore Assumption A3 becomes

$$\begin{aligned} &x'(-P_\infty + Q + A'P_\infty B(B'P_\infty B + R)^{-1}(B'P_\infty B + R)(B'P_\infty B + R)^{-1}B'P_\infty A + \\ &+A'P_\infty A - 2A'P_\infty B(B'P_\infty B + R)^{-1}B'P_\infty A)x \leq 0, \ \forall x \in \mathcal{X}_f, \end{aligned}$$

which can be simplified to

$$x'(-P_\infty + Q + A'P_\infty A - A'(P_\infty B(B'P_\infty B + R)^{-1}BP_\infty)A)x \leq 0, \ \forall x \in \mathcal{X}_f.$$

Note that the LQR cost P_∞ solves the Riccati Equation:

$$P_\infty = A'P_\infty A + Q - A'P_\infty B(B'P_\infty B + R)^{-1}B'P_\infty A.$$

Therefore we proved that

$$x'(-P_\infty + Q + A'P_\infty A - A'(P_\infty B(B'P_\infty B + R)^{-1}BP_\infty)A)x = 0, \ \forall x \in \mathcal{X}_f.$$

3.3 Using any stabilizing controller

In general, instead of F_∞ we can choose any controller F which stabilizes $A + BF$. For instance one could use a pole placement control design.

With $v = Fx$ the assumption (A3) in the main MPC Theorem becomes

$$-P + (Q + F'RF) + (A + BF)'P(A + BF) \leq 0. \tag{18}$$

It is satisfied as an equality if we choose P as a solution of the corresponding Lyapunov equation.

In Matlab:

```
% first design a stabilizing control law via pole placement
F=place(A,B,[0.1,0.2]);
% closed loop system
Acl=A-B*F;
S=X.intersect(Polyhedron('H',[-U.H(:,1:nu)*F U.H(:,nu+1)]));
% terminal Set
Oinf=max_pos_inv(Acl,S);
%terminal Cost
P=dlyap(Acl,Q+F'*R*F);
```

If the open loop system (11) is asymptotically stable, then we may even select $F = 0$. Note that depending on the choice of the controller the controlled invariant terminal region $\mathcal{X}_f$ changes.