Introduction to LQ Model Predictive Control

Exact continuous-time to discrete-time conversion for LTI systems

$$\dot{x} = A_c x + B_c u \Rightarrow x_{k+1} = e^{A_c T_s} x_k + \int_{kT_s}^{(k+1)T_s} e^{A_c((k+1)T_s - \tau)} B_c u(\tau) d\tau$$

$$\Rightarrow x_{k+1} = e^{A_c T_s} x_k + \left(\int_{kT_s}^{(k+1)T_s} e^{A_c((k+1)T_s - \tau)} d\tau \right) B_c u_k$$

$$\Rightarrow x_{k+1} = e^{A_c T_s} x_k + A^{-1}(e^{A_c T_s} - I) B_c u_k$$

(if A_c is an invertible matrix, which is not always the case)

$$\Rightarrow x_{k+1} = A_d x_k + B_d u_k, \quad A_d = e^{A_c T_s}, \quad B_d = A_c^{-1}(e^{A_c T_s} - I) B_c$$

If A_c is not invertible, see D.S. Bernstein, Matrix Mathematics.

Equilibria and linearization

- Consider a nonlinear discrete-time system,

$$x_{k+1} = f(x_k, u_k)$$

- Suppose $u_k = u_e$ for all $k \geq 0$. Then $x_k = x_e$ is a corresponding to u_e equilibrium if $x_e = f(x_e, u_e)$. Multiple equilibria may exist, in general.

- Let $\delta x = x - x_e$, $\delta u = u - u_e$, where x_e, u_e are given and are not necessary an equilibrium pair. The linearized model at (x_e, u_e) approximates the dynamics of δx and has the form,

$$\delta x_{k+1} = \left[\frac{\partial f}{\partial x}(x_e, u_e)\right] \delta x_k + \left[\frac{\partial f}{\partial u}(x_e, u_e)\right] \delta u_k + [f(x_e, u_e) - x_e]$$

  If (x_e, u_e) is an equilibrium pair, then

$$\delta x_{k+1} = \left[\frac{\partial f}{\partial x}(x_e, u_e)\right] \delta x_k + \left[\frac{\partial f}{\partial u}(x_e, u_e)\right] \delta u_k$$

- The matrices

$$\left[\frac{\partial f}{\partial x}(x_e, u_e)\right], \quad \left[\frac{\partial f}{\partial u}(x_e, u_e)\right]$$

  are Jacobian matrices (matrices of partial derivatives).

Numerical approximation of derivatives (if derivatives cannot be computed analytically)

- Center difference approximation:

$$\frac{\partial f}{\partial x_i}(x_e, u_e) \approx \frac{f(x_e + he_i, u_e) - f(x_e - he_i, u_e)}{2h},$$

$$\frac{\partial f}{\partial u_i}(x_e, u_e) \approx \frac{f(x_e, u_e + he_i) - f(x_e, u_e - he_i)}{2h},$$

  where e_i is the ith unit vector

- Error is $O(h^2 + \epsilon_f/h)$ where ϵ_f is an upper bound on magnitude of "noise" in function value measurements.

- In practice: perturb x_e by $h|x_{ei}|e_i$ (or u by $h|u_{ei}|e_i$), assuming $x_{ei} \neq 0$ (or $u_{ei} \neq 0$) since different components may be scaled differently

LTI discrete-time systems without inputs

Consider a linear-time invariant autonomous system of the form,

$$x_{k+1} = Ax_k, \quad k \in Z^+, \quad x_k \in R^{n_x}.$$

Thus

$$x_1 = Ax_0, \quad x_2 = Ax_1 = A^2x_0, \text{ etc.}$$

Note that $x = 0$ is an equilibrium.

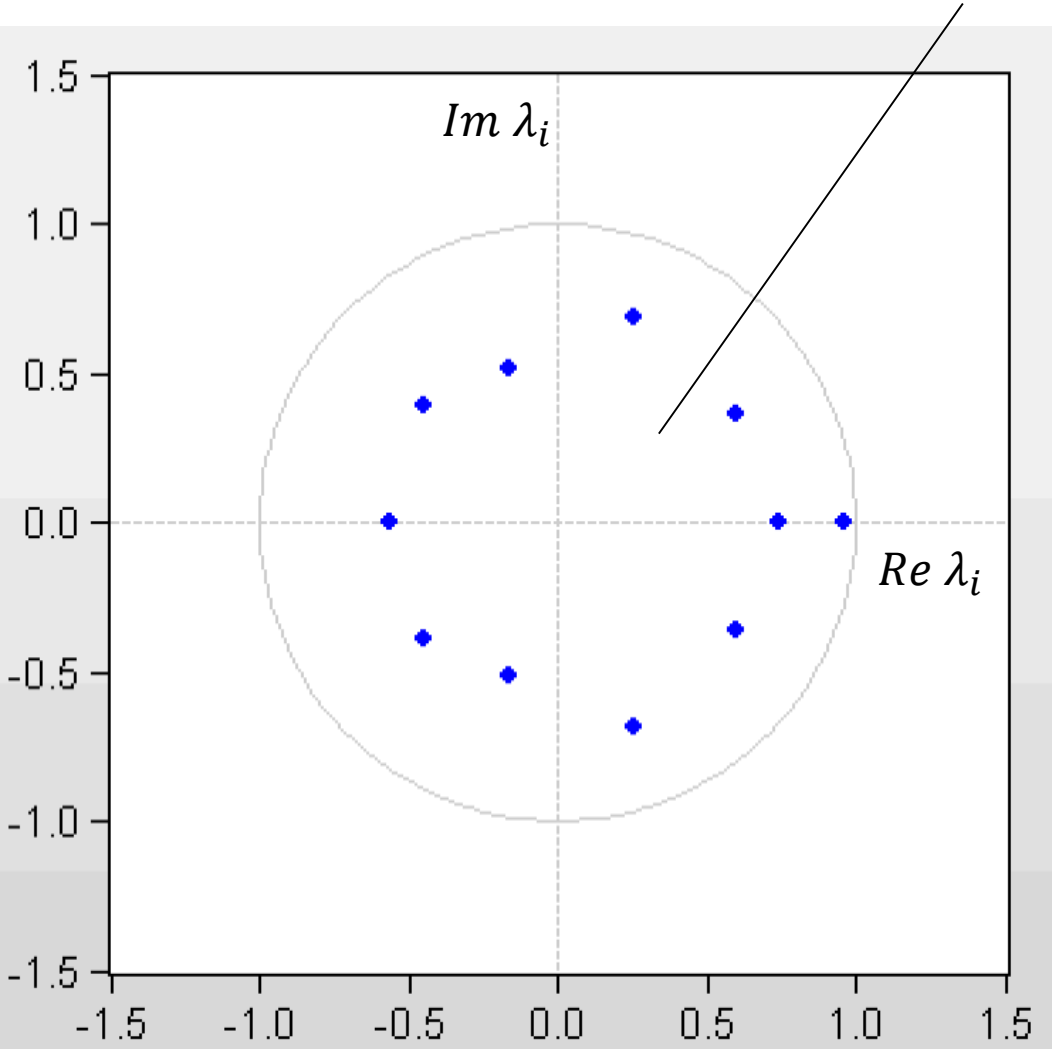
Let $spec(A) = \{\lambda \in \mathbb{C} : \quad det(\lambda I - A) = 0\}$.

Then, $spec(A) \subset D_s = \{\lambda \in \mathbb{C} : \quad |\lambda| < 1\}$ if and only if $x = 0$ is globally asymptotically stable (GAS).

The same definition of stability as in continuous time applies with continuous-time trajectories replaced by discrete-time trajectories.

Matrices with all eigenvalues inside unit disk of complex plane are called Schur's matrices.

All eigenvalues must be inside unit disk for asymptotic stability



Justification for the stability condition…

Asymptotic stability at the origin of $x_{k+1} = Ax_k$ means

- $\forall \epsilon > 0,\ \exists \delta > 0,$ such that $\|x_0\| < \delta \Rightarrow \|x_k\| < \epsilon,\ \forall k \in \mathbb{Z}_{\geq 0}$
- $x_k = A^k x_0 \to 0$ as $k \to \infty$

Suppose $q = \|A\| < 1$. Then

$$\|x_k\| \leq \|A^k x_0\| \leq \|A\| \|A^{k-1} x_0\| \leq \cdots \leq \|A\|^k \|x_0\| \leq q^k \|x_0\|$$

Thus $\|A\| < 1$ is sufficient for asymptotic stability (definition holds with $\epsilon = \delta$), where $\|\cdot\|$ can be any norm.

The necessary and sufficient condition in terms of the spectral radius, $\rho(A) = \max\{|\lambda|,\ \lambda \in \text{spec}(A)\}$, that $\rho(A) < 1$, follows from Gelfand's formula:

$$\lim_{k \to \infty} \frac{(\rho(A))^k}{\|A^k\|} = 1,$$

which holds for any norm $\|\cdot\|$. Hence $\|A^k x_0\| \leq \|A^k\| \|x_0\| \approx (\rho(A))^k \|x_0\|$ for large k.

LTI discrete-time systems with inputs

Consider now a linear-time invariant discrete-time system with input,

$$x_{k+1} = Ax_k + Bu_k, \quad k \in Z^+, \quad x_k \in R^{n_x}, \quad u_k \in R^{n_u}.$$

Thus

$$\begin{aligned} x_1 &= Ax_0 + Bu_0 \\ x_2 &= Ax_1 + Bu_1 \\ &= A^2x_0 + ABu_0 + Bu_1 \\ &\vdots \\ x_N &= A^Nx_0 + \sum_{k=0}^{N-1} A^{N-1-k}Bu_k \end{aligned}$$

The last expression is called the State Transition Formula for linear discrete-time systems.

Reachable states

What are states that can be reached by control from the origin?

Let $x_0 = 0$, $x_k \in \mathbb{R}^n$. Define

$$X_1 = \{x : \exists u_0 \text{ such that } Ax_0 + Bu_0 = x\} = \text{range}(B)$$
$$X_2 = \{x : \exists u_0,\ u_1 \text{ such that } A^2x_0 + ABu_0 + Bu_1 = x\} = range([B\ AB])$$
$$\vdots$$
$$X_n = range(C_n),\ C_n = [B\ AB\ \cdots\ A^{n-1}B]$$

Fact: $X_k \subseteq X_n$ for $k \geq n$ (Proof: use Cayley-Hamilton theorem)

Remark: Need n control moves $u_0,\ \cdots,\ u_{n-1}$ to be able to reach all states in X_n.

Controllability of linear discrete-time systems

Reachability from the origin $\Leftrightarrow X_n = \mathbb{R}^n$

$\Leftrightarrow$ Controllability to the origin (why?)

$\Leftrightarrow rank[B\ AB\ \cdots\ A^{n-1}B] = n$

$\Leftrightarrow rank[(sI - A)\ B] = n$ for all $s \in spec(A)$ (Hautus test)

$\Leftrightarrow \exists K$ such that eigenvalues of $(A + BK)$ can be arbitrarily placed

[Q&A] Can controllability be lost through sampling?

[**A**]: Yes. Consider

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u.$$

Then, the corresponding discrete-time model has the form,

$$x_{k+1} = \begin{bmatrix} \cos T_s & \sin T_s \\ -\sin T_s & \cos T_s \end{bmatrix} x_k + \begin{bmatrix} 1 - \cos T_s \\ \sin T_s \end{bmatrix} u_k$$

Consider sampling period, $T_s = m\pi$, $m \in \mathbb{Z}_{>0}$. Then,

$$x_{k+1} = \begin{bmatrix} (-1)^m & 0 \\ 0 & (-1)^m \end{bmatrix} x_k + \begin{bmatrix} 1 - (-1)^m \\ 0 \end{bmatrix} u_k$$

is not controllable (control cannot affect the second state).

[Q&A] Can controllability be lost through sampling?

Fact: Suppose for the continuous-time LTI model (A_c, B_c) is controllable, and whenever $Re[\lambda_i - \lambda_j] = 0$ for $\lambda_i, \lambda_j \in spec(A_c)$ it follows that $Im[\lambda_i - \lambda_j] \neq \frac{2\pi m}{T_s}$ for $m = 1, 2, \cdots$. Then for the corresponding discrete-time LTI model, (A_d, B_d) is controllable. This condition is also necessary in the single input case.

(Does this condition hold in the previous example?)

Observability of linear discrete-time systems

Observability is the ability to uniquely reconstruct $x_0 \in \mathbb{R}^n$ from output measurements, $y_k = Cx_k$. Let $u_k = 0$ and consider measurements,

$$\begin{aligned} y_0 &= Cx_0, \\ y_1 &= Cx_1 = CAx_0, \\ y_2 &= Cx_2 = CAx_1 = CA^2x_0, \\ &\vdots \\ y_{n-1} &= CA^{n-1}x_0, \end{aligned}$$

or

$$Y = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} x_0 = O_n x_0.$$

Observability of discrete-time systems

The vector x_0 can be uniquely reconstructed if the columns of the matrix O_n are independent, i.e., the rank of the observability matrix O_n is equal to n.

The condition rank $O_n = n$ can be demonstrated to be both necessary and sufficient for linear system observability (Why is it also necessary?).

While we assumed $u_k = 0$, the ability to reconstruct x_0 is unaffected by the presence of a known input, u_k (Why?)

Hautus test (equivalent necessary and sufficient condition for linear system observability):

$$rank(\begin{bmatrix} C \\ sI - A \end{bmatrix}) = n \text{ for all } s \in spec(A)$$

Finite horizon Linear Quadratic (LQ) control

Consider a finite-horizon optimal control problem (P-LQN):

$$\begin{aligned} \underset{u_0,u_1,\cdots,u_{N-1}}{\text{Minimize}} \quad & J_N = \sum_{k=0}^{N-1} x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k + x_N^{\mathrm{T}} P x_N \\ \text{subject to} \quad & x_{k+1} = A x_k + B u_k \\ & x_0 \text{ is given} \end{aligned}$$

Remark: If the discrete time instant $k = 0$ corresponds to the current time and x_0 corresponds to the current state, the solution to P-LQN can be used to define an LQ-MPC feedback law by the first move,

$$u_{MPC}(x_0) = u_0.$$

Such a control law does not account for state and control constraints for the moment but we will introduce them later.

Stack states and controls into vectors

Stack the states and controls into vectors $X \in \mathbb{R}^{N \cdot n_x}$ and $U \in \mathbb{R}^{N \cdot n_u}$,

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}, \quad U = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}.$$

$$(x \in \mathbb{R}^{n_x},\, u \in \mathbb{R}^{n_u})$$

Note we do not include x_0 in X (it is given) and u_N in U (has no effect on the problem).

Stacked states vector is expressed as an affine function of stacked controls vector

The state transition formula,

$$x_k = A^k x_0 + \sum_{i=0}^{k-1} A^{k-1-i} B u_i,$$

yields the following relation,

$$X = SU + Mx_0,$$

where

$$S = \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix}, \quad M = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}.$$

Cost function transformation

Let

$$\bar{Q} = \begin{bmatrix} Q & \cdots & & \cdots & 0 \\ 0 & Q & & & \\ \vdots & & \ddots & & \\ 0 & \cdots & & Q & \\ 0 & \cdots & & 0 & P \end{bmatrix}, \quad \bar{R} = \begin{bmatrix} R & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & R \end{bmatrix},$$

Then

$$\begin{aligned} J_N &= \sum_{k=0}^{N-1} x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k + x_N^{\mathrm{T}} P x_N \\ &= X^{\mathrm{T}} \bar{Q} X + U^{\mathrm{T}} \bar{R} U + x_0^{\mathrm{T}} Q x_0 \\ &= (SU + M x_0)^{\mathrm{T}} \bar{Q} (SU + M x_0) + U^{\mathrm{T}} \bar{R} U + x_0^{\mathrm{T}} Q x_0 \\ &= U^{\mathrm{T}} (S^{\mathrm{T}} \bar{Q} S + \bar{R}) U + 2 x_0^{\mathrm{T}} M^{\mathrm{T}} \bar{Q} S U + x_0^{\mathrm{T}} M^{\mathrm{T}} \bar{Q} M x_0 + x_0^{\mathrm{T}} Q x_0 \\ &= U^{\mathrm{T}} H U + 2 q^{\mathrm{T}} U + c \end{aligned}$$

where $H = S^{\mathrm{T}} \bar{Q} S + \bar{R}$, $q = S^{\mathrm{T}} \bar{Q} M x_0$, $c = x_0^{\mathrm{T}} (Q + M^{\mathrm{T}} \bar{Q} M) x_0$.

Reduction to minimizing a quadratic function

Thus we have demonstrated that P-LQN reduces to minimizing a quadratic function,

$$\underset{U \in \mathbb{R}^{N \times n_u}}{\text{Minimize}} \quad J_N = U^{\mathrm{T}} H U + 2 q^{\mathrm{T}} U$$

$$H = (S^{\mathrm{T}} \bar{Q} S + \bar{R}) = H^{\mathrm{T}}, \; q = S^{\mathrm{T}} \bar{Q} M x_0$$

Minimizing a quadratic function…

The necessary condition for $U = U^*$ to be a minimizer of J_N is that the gradient of J_N with respect to U should vanish at $U = U^*$.

Fact 1: $\nabla_U J_N = 2HU + 2q$

Assumption: The state weight matrices $Q = Q^{\mathrm{T}} \succeq 0$, $P = P^{\mathrm{T}} \succeq 0$ are symmetric positive semi-definite, the control weight matrix, $R = R^{\mathrm{T}} \succ 0$ is symmetric positive definite.

Fact 2: The matrix $H = H^{\mathrm{T}} \succ 0$ is a positive-definite symmetric matrix. Hence it is invertible.

Fact 3: If a (global) minimizer of J_N exists, it must be given by

$$U = U^* = -H^{-1}q = -(S^{\mathrm{T}}\bar{Q}S + \bar{R})^{-1}S^{\mathrm{T}}\bar{Q}Mx_0$$

Remark: Note that the existence of the global minimizer has not been established at this point. We are not done yet!

Definite matrices

A symmetric matrix $P = P^{\mathrm{T}} \in \mathbb{R}^{n \times n}$, with elements P_{ij}, is positive-definite if for any $x \in \mathbb{R}^n$, $x \neq 0$, the corresponding quadratic form is positive, i.e.,

$$x^{\mathrm{T}} P x = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i P_{ij} x_j > 0.$$

A symmetric matrix $P = P^{\mathrm{T}} \in \mathbb{R}^{n \times n}$, with elements P_{ij}, is positive semi-definite if for any $x \in \mathbb{R}^n$, the corresponding quadratic form is nonnegative,

$$x^{\mathrm{T}} P x = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i P_{ij} x_j \geq 0.$$

Definite matrices

Checking positive-definiteness of a symmetric matrix:

- If a matrix is symmetric (equal to its transpose), its eigenvalues are real.

- A matrix is positive-definite if and only if all its eigenvalues are positive. (Matlab: $sum(eig(P) <= 0) == 0$ or $all(eig(P) > 0) == true$)

- A matrix is positive semi-definite if and only if its eigenvalues are all non-negative (Matlab: $sum(eig(P) < 0) == 0$ or $all(eig(P) >= 0) == true$)

- Sylvester's criterion (useful for symbolic computations): Matrix is positive semi-definite if and only if all of its principal minors (determinants of square sub-matrices formed by deleted the same indexed rows and columns) are non-negative. A matrix is positive definite if and only if all its leading principal minors (determinants of square submatrices formed by deleting the last $n - k$ rows and columns, $k = 1, \cdots, n - 1$) are strictly positive.

Minimizing a quadratic function…

Note that

$$\begin{aligned}(U + H^{-1}q)^{\mathrm{T}} H (U + H^{-1}q) &= U^{\mathrm{T}} H U + q^{\mathrm{T}} H^{-1} H U \\ &\quad + q^{\mathrm{T}} H^{-1} H U + q^{\mathrm{T}} H^{-1} H H^{-1} q \\ &= U^{\mathrm{T}} H U + 2 q^{\mathrm{T}} U + q^{\mathrm{T}} H^{-1} q\end{aligned}$$

Thus

$$J_N(U) = (U + H^{-1}q)^{\mathrm{T}} H (U + H^{-1}q) - q^{\mathrm{T}} H^{-1} q.$$

Since $H \succ 0$, the first term is positive if $U \neq -H^{-1}q$ and the second term does not depend on U, it is clear from this expression that the global minimizer of $J_N(U)$ is at $U = U^* = -H^{-1}q$.

Remark: The process we went through is completing the squares.

MPC control law from P-LQN

- Note that the optimal control sequence, U^*, is a function of the initial state x_0, i.e., $U^* = U^*(x_0)$.
- LQ-MPC Feedback Law, when there are no state and no control constraints, is determined by the first move of the P-LQN problem. Thus

$$u_{MPC}(x_0) = \begin{bmatrix} \mathbb{I}_{n_u \times n_u} & 0 & \cdots & 0 \end{bmatrix} U^*(x_0)$$

$$= K_{0,N} x_0$$

$$K_{0,N} = -\begin{bmatrix} \mathbb{I}_{n_u \times n_u} & 0 & \cdots & 0 \end{bmatrix} (S^{\mathrm{T}} \bar{Q} S + \bar{R})^{-1} S^{\mathrm{T}} \bar{Q} M$$

- LQ-MPC Feedback Law is simply a constant gain feedback!

Finite horizon LQ control

Consider a finite-horizon optimal control problem (P-LQN):

$$\begin{aligned} \underset{u_0,u_1,\cdots,u_{N-1}}{\text{Minimize}} \quad & J_N = \sum_{k=0}^{N-1} x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k + x_N^{\mathrm{T}} P x_N \\ \text{subject to} \quad & x_{k+1} = A x_k + B u_k \\ & x_0 \text{ is given} \\ & (Q \succeq 0, \quad R \succ 0, \quad P \succeq 0) \end{aligned}$$

This problem can also be solved by the methods of optimal control theory (AEROSP 575) using either dynamic programming or Pontryagin Maximum Principle (PMP). The solution has the following form:

$$P_{N,N} = P$$

$$K_{k-1,N} = -(R + B^{\mathrm{T}} P_{k,N} B)^{-1} B^{\mathrm{T}} P_{k,N} A$$

$$P_{k-1,N} = Q + A^{\mathrm{T}} P_{k,N} A + A^{\mathrm{T}} P_{k,N} B K_{k-1,N}, \;\; k = N, N-1, \cdots, 1$$

$$u_k = K_{k,N} x_k, \quad k = 0 \cdots, N-1$$

Infinite horizon Linear Quadratic Regulator (LQR) control law

Properties: Let $Q = C^{\mathrm{T}}C$ (e.g., if $Q = N^T \Lambda N,\ \Lambda = diag[\lambda_i],\ \lambda_i \in spec(Q) \Rightarrow C = Q^{1/2} = N^T \Lambda^{1/2} N$). Under the assumptions of stabilizability of (A, B) and detectability of (A, C), for any fixed k, the sequence $P_{k,N}$ converges as $N \to \infty$ to P_∞ which is the (unique) positive semi-definite symmetric solution, $P_\infty \succeq 0$, of Discrete Algebraic Riccati Equation (DARE),

$$P_\infty = Q + A^{\mathrm{T}} P_\infty A + A^{\mathrm{T}} P_\infty B K_\infty, \quad K_\infty = -(R + B^T P_\infty B)^{-1} B^T P_\infty A$$

The control law $u_k = K_\infty x_k$, achieves closed-loop asymptotic stability, i.e., $spec(A + BK_\infty) \subset D_s$. The control sequence generated via $u_k = K_\infty x_k,\ x_{k+1} = Ax_k + Bu_k$ achieves the minimum value of the infinite horizon cost,

$$J_\infty = \sum_{k=0}^{\infty} x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k.$$

If (A, C) is not only detectable but also observable, then $P_\infty \succ 0$.

The optimal cost is given by $J_\infty^* = x_0^{\mathrm{T}} P_\infty x_0$, and the closed-loop system is represented by $x_{k+1} = (A + BK_\infty)x_k$.

Matlab dlqr.m command

```
>> help dlqr
 DLQR  Linear-quadratic regulator design for discrete-time systems.

   [K,S,E] = DLQR(A,B,Q,R,N)  calculates the optimal gain matrix K
   such that the state-feedback law  u[n] = -Kx[n]  minimizes the
   cost function

         J = Sum {x'Qx + u'Ru + 2*x'Nu}

   subject to the state dynamics   x[n+1] = Ax[n] + Bu[n].

   The matrix N is set to zero when omitted.  Also returned are the
   Riccati equation solution S and the closed-loop eigenvalues E:
                                          -1
    A'SA - S - (A'SB+N)(R+B'SB)   (B'SA+N') + Q = 0,   E = EIG(A-B*K).
```

Note: dlqr generates control law in a “feedback form” with minus in front

LQR weight tuning

- Weights in LQR become tuning knobs. Rather than adjusting controller gains directly (e.g., in PID tuning), in LQR you adjust weights that have more direct effect on response especially in multi-input systems. For this to be succesful, the model accuracy needs to be reasonable.

- Diagonal weights are often used

$$Q = \begin{bmatrix} q_1 & & 0 \\ & \ddots & \\ 0 & & q_n \end{bmatrix}, R = \begin{bmatrix} r_1 & & 0 \\ & \ddots & \\ 0 & & r_m \end{bmatrix}$$

States or control signals that should remain small have higher weights attached to them.

LQR weight tuning

- **Rule of thumb:** Choose q_i and r_j as the inverse of the square of the maximum magnitude value for the corresponding x_i or u_j. Then modify the elements to obtain a compromise among response time, damping and control effort by trial and error.

- Note that it is also possible to choose weights to emphasize only a subset of signals, e.g., let $y = Cx$, where (A, C) is observable. Then to penalize $y^T y$ and $ru^T u$, define $Q = C^T C$ and $R = rI$.

[Q&A] How are optimality and stability linked?

Stabilizability and LQR control

Consider an infinite horizon optimal control problem with

$$J = \sum_{k=0}^{\infty} x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k$$

If (A, B) stabilizable, one can find a feasible linear feedback law which "exponentially" stabilizes the origin. For discrete-time systems, this means that there exists a feedback gain, F, such that all closed-loop eigenvalues with the control law $u = Fx$ are inside the unit disk, i.e., $spec(A + BF) \subset D_s$.

The above cost J computed for a feasible control law $u_k = Fu_k$, where F is a stabilizing gain, is finite since all eigenvalues of the closed-loop system are inside the unit disk and $x_k \to 0$ and $u_k \to 0$ at at a rate faster than q^k for some $0 \leq q < 1$.

Since optimal control law cannot do worse than a feasible control law, the optimal cost, J^*, with $u = K_\infty x$ where K_∞ is the optimal LQ gain, must be finite.

Stabilizability and LQR control

Remark: Stabilizability is a weaker property than controllability, i.e., if (A, B) is controllable then (A, B) is stabilizable but the converse is not true.

Under the stronger assumption that (A, B) is controllable, there will exist a sequence of the form $\{u_0, u_1, \cdots, u_{N-1}, 0, 0, \cdots\}$ which will steer the state to zero in N steps, i.e., $x_N = 0$. Hence the infinite horizon cost will be finite since it is bounded by the cost value due to this feasible sequence steering the state to zero.

Observability and LQR control

Suppose $Q = C^{\mathrm{T}}C$.

Suppose (A, C) is observable. Consider the state cost computed on optimal trajectories,

$$\sum_{k=0}^{\infty} x_k^{\mathrm{T}} Q x_k = \sum_{k=0}^{\infty} x_k^{\mathrm{T}} C^{\mathrm{T}} C x_k = \sum_{k=0}^{\infty} ||Cx_k||^2 = \sum_{k=0}^{\infty} ||y_k||^2, \; y_k = Cx_k.$$

If the optimal cost, J^*, is bounded, the state cost is bounded, and $y_k \to 0$ as $k \to \infty$.

If C is full rank ($Q > 0$) then $x_k \to 0$ as $k \to \infty$.

Similarly, $R > 0$ and $\sum_{k=0}^{\infty} u_k^{\mathrm{T}} R u_k < \infty$ implies that $u_k \to 0$ as $k \to \infty$

Observability and LQR control

If C is not full rank but (C, A) is observable

$$\begin{bmatrix} y_k \\ y_{k+1} \\ \vdots \\ y_{k+n-1} \end{bmatrix} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix} x_k + \begin{bmatrix} 0 & 0 & \cdots & 0 \\ CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ CA^{n-1}B & \cdots & \cdots & CB \end{bmatrix} \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+n-2} \end{bmatrix} \to 0$$

Observability implies that this matrix is full column rank.
n is the dimension of x_k

This vector goes to 0 with k

$$y_k \to 0 \Rightarrow y_{k+1} \to 0, \cdots, y_{k+n-1} \to 0 \Rightarrow x_k \to 0 \text{ as } k \to \infty$$

Observability and LQR control

We conclude that if (A, B) is stabilizable, and (A, C) is observable, the solution to the infinite-horizon discrete-time LQR problem is always an asymptotically stabilizing feedback law.

If (A, B) is stabilizable, and (A, C) is not observable but detectable, the unstable modes are "visible" in the cost and the solution to the discrete-time LQ problem is also an asymptotically stabilizing feedback law.

Detectability is a weaker notion than observability and means that eigenvalues of the observer error equation, i.e., of $(A + LC)$ in $e_{k+1} = (A + LC)e_k$, can be placed inside the unit disk by an appropriate choice of the observer gain, L.

The unique stabilizing solution to DARE can be guaranteed to be positive-definite if (A, C) is observable, and it will be only positive semi-definite if (A, C) is detectable

MPC versus classical LQR control

Consider now a related finite-horizon optimal control problem, P-LQN', with the terminal penalty:

$$\begin{aligned} \underset{u_0, u_1, \cdots, u_{N-1}}{\text{Minimize}} \quad & J_N = \sum_{k=0}^{N-1} x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k + x_N^{\mathrm{T}} P_\infty x_N \\ \text{subject to} \quad & x_{k+1} = A x_k + B u_k \\ & x_0 \text{ is given} \end{aligned}$$

where $P_\infty \succ 0$ is the solution to DARE for P-LQ.

The Bellman's Principle of Optimality states that any final segment of an optimal trajectory is optimal with respect to its initial state.

Hence problems P-LQ and P-LQN' must have the same solution.

Thus with the selection of the terminal weighting matrix, P, as a solution to DARE for LQR, P_∞, MPC recovers the solution of LQR (independently of the horizon, N).

MPC versus classical LQR control

Consider now a finite-horizon optimal control problem without the terminal penalty:

$$\begin{aligned} \underset{u_0,u_1,\cdots,u_{N-1}}{\text{Minimize}} \quad & J_N = \sum_{k=0}^{N-1} x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k. \\ \text{subject to} \quad & x_{k+1} = A x_k + B u_k \\ & x_0 \text{ is given} \end{aligned}$$

Recall that MPC feedback law is defined by the first move and has the form of a constant gain feedback law, $u_{MPC}(x_0) = K_{0,N} x_0$. In general, $K_{0,N} \neq K_\infty$ and the gain is different from LQR gain.

If (A, B) is stabilizable, as you increase the horizon, $K_{0,N} \to K_\infty$ as $N \to \infty$. If (A, C) is also detectable, K_∞ is stabilizing (closed-loop is stable) and so is $K_{0,N}$ for a sufficiently long horizon, N.

LQ-MPC problem with constraints

Define now a Constrained LQ Model Predictive Control (P-LQMPC) problem,

$$
\begin{aligned}
\underset{u_0, u_1, \cdots, u_{N-1}}{\text{Minimize}} \quad & J_N = \sum_{k=0}^{N-1} x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k + x_N^{\mathrm{T}} P x_N \\
\text{subject to} \quad & x_{k+1} = A x_k + B u_k \\
& x_0 = \text{current state} \\
& x_{min} \leq x_k \leq x_{max}, \ k = 1, \cdots, N \\
& u_{min} \leq u_k \leq u_{max}, \ k = 0, \cdots, N-1
\end{aligned}
$$

The MPC feedback law is defined by the first optimal move, u_0^*, i.e, $u_{MPC}^*(x_0) = u_0^*$.

Remark 1: More general constraints that are affine in state and control and the case when constraint, control and prediction horizons are different ($N_c \neq N_u \neq N$) can be treated similarly.

Remark 2: Suppose $P = P_\infty$, where P_∞ is the solution to the corresponding DARE. If the constraints are inactive (hold as strict inequalities) for the optimal solution at x_0, the solutions to P-LQMPC and P-LQ coincide. This is typically the case near the origin, hence $u_{MPC}(x_0) = u_0^* = K_\infty x_0$ and local closed-loop stability is guaranteed.

Reduction to quadratic program

Stack the states and controls into vectors $X \in \mathbb{R}^{N \cdot n_x}$ and $U \in \mathbb{R}^{N \cdot n_u}$,

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}, \quad U = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}.$$

We will now demonstrate that P-LQMPC reduces to a quadratic programming problem, that is a problem of minimizing a quadratic function subject to linear constraints.

Stacked states vector is expressed as a linear function of stacked controls vector

The state transition formula,

$$x_k = A^k x_0 + \sum_{i=0}^{k-1} A^{k-1-i} B u_i,$$

yields the following relation,

$$X = SU + Mx_0,$$

where

$$S = \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix}, \quad M = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}.$$

Constraints on stacked up states vector

Let

$$X_{min} = \begin{bmatrix} x_{min} \\ x_{min} \\ \vdots \\ x_{min} \end{bmatrix}, \quad X_{max} = \begin{bmatrix} x_{max} \\ x_{max} \\ \vdots \\ x_{max} \end{bmatrix}.$$

Then, the state constraints have the form,

$$X_{min} \leq X = SU + Mx_0 \leq X_{max},$$

which can be written as

$$\begin{bmatrix} SU \\ -SU \end{bmatrix} \leq \begin{bmatrix} X_{max} - Mx_0 \\ -X_{min} + Mx_0 \end{bmatrix},$$

or

$$\begin{bmatrix} S \\ -S \end{bmatrix} U \leq \begin{bmatrix} X_{max} - Mx_0 \\ -X_{min} + Mx_0 \end{bmatrix}.$$

Augmenting control constraints

Let

$$U_{min} = \begin{bmatrix} u_{min} \\ u_{min} \\ \vdots \\ u_{min} \end{bmatrix}, \quad U_{max} = \begin{bmatrix} u_{max} \\ u_{max} \\ \vdots \\ u_{max} \end{bmatrix}.$$

Then, the constraints on the stacked-up controls vector have the form,

$$U_{min} \leq U \leq U_{max},$$

which can be written as

$$\begin{bmatrix} I \\ -I \end{bmatrix} U \leq \begin{bmatrix} U_{max} \\ -U_{min} \end{bmatrix}.$$

State and control constraints

The state and control constraints thus take the form,

$$\begin{bmatrix} S \\ -S \\ I \\ -I \end{bmatrix} U \leq \begin{bmatrix} X_{max} - Mx_0 \\ -X_{min} + Mx_0 \\ U_{max} \\ -U_{min} \end{bmatrix}.$$

This can be written as

$$GU \leq W + Tx_0,$$

$$G = \begin{bmatrix} S \\ -S \\ I \\ -I \end{bmatrix}, \quad W = \begin{bmatrix} X_{max} \\ -X_{min} \\ U_{max} \\ -U_{min} \end{bmatrix}, \quad T = \begin{bmatrix} -M \\ M \\ 0 \\ 0 \end{bmatrix}$$

Cost function transformation

Let

$$\bar{Q} = \begin{bmatrix} Q & \cdots & & \cdots & 0 \\ 0 & Q & & & \\ \vdots & & \ddots & & \\ 0 & \cdots & & Q & \\ 0 & \cdots & & 0 & P \end{bmatrix}, \quad \bar{R} = \begin{bmatrix} R & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & R \end{bmatrix},$$

Then

$$\begin{aligned} J_N &= \sum_{k=0}^{N-1} x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k + x_N^{\mathrm{T}} P x_N \\ &= X^{\mathrm{T}} \bar{Q} X + U^{\mathrm{T}} \bar{R} U + x_0^{\mathrm{T}} Q x_0 \\ &= (SU + M x_0)^{\mathrm{T}} \bar{Q} (SU + M x_0) + U^{\mathrm{T}} \bar{R} U + x_0^{\mathrm{T}} Q x_0 \\ &= U^{\mathrm{T}} (S^{\mathrm{T}} \bar{Q} S + \bar{R}) U + 2 x_0^{\mathrm{T}} M^{\mathrm{T}} \bar{Q} S U + x_0^{\mathrm{T}} M^{\mathrm{T}} \bar{Q} M x_0 + x_0^{\mathrm{T}} Q x_0 \\ &= U^{\mathrm{T}} H U + 2 q^{\mathrm{T}} U + c \end{aligned}$$

where $H = S^{\mathrm{T}} \bar{Q} S + \bar{R}$, $q = S^{\mathrm{T}} \bar{Q} M x_0$, $c = x_0^{\mathrm{T}} (Q + M^{\mathrm{T}} \bar{Q} M) x_0$.

Reduction to quadratic program

Thus we have demonstrated that P-LQMPC reduces to a quadratic programming problem

$$\begin{aligned} \underset{U \in \mathbb{R}^{N \times n_u}}{\text{Minimize}} \quad & J_N = U^{\mathrm{T}} H U + 2 q^{\mathrm{T}} U \\ \text{subject to} \quad & GU \leq W + T x_0. \end{aligned}$$

$$(q = S^{\mathrm{T}} \bar{Q} M x_0)$$

Since $H = (S^{\mathrm{T}} \bar{Q} S + \bar{R}) = H^{\mathrm{T}} > 0$, the cost is a strictly convex function of U, and the above optimization problem is a strictly convex QP with affine inequality constraints.

Note that the optimization problem is parameterized by the current state x_0.

Once the optimization problem is solved, and the solution $U^*(x_0)$ is obtained, the LQ-MPC feedback law is defined as

$$u_{MPC}(x_0) = \begin{bmatrix} \mathbb{I}_{n_u \times n_u} & 0 & \cdots & 0 \end{bmatrix} U^*(x_0)$$

QP solver is needed in the constrained case

What do you have to specify to a solver?

$$\begin{aligned} \underset{U \in \mathbb{R}^{N \times n_u}}{\text{Minimize}} \quad & J_N = \frac{1}{2} U^{\mathrm{T}} H U + q^{\mathrm{T}} U \\ \text{subject to} \quad & GU \leq \tilde{W} \quad (\tilde{W} = W + T x_0). \qquad (q = S^{\mathrm{T}} \bar{Q} M x_0) \end{aligned}$$

$$\mathbf{solveQP}(H, q, G, \tilde{W}, \mathbf{opts})$$

Common optional parameters (opts) are

- Bounds for the variables (min and max)
- Initial solution guess (otherwise the solver finds one)
- Maximum number of iterations (or max time)
- Termination condition (ϵ, $||J_N(U) - J_N(U^*)|| \leq \epsilon$)
- Constraint acceptance (ϵ, $GU - \tilde{W} \leq \mathbf{1}\epsilon$)

Tracking LQ-MPC design

Tracking LQ-MPC design

MPC controllers can be designed to perform set-point tracking, i.e., cause a given output y of the system,

$$x_{k+1} = Ax_k + Bu_k, \quad y_k = Cx_k,$$

follow a specified command (set-point) r. Specifically, if $r \equiv const$ the control design objective is to achieve convergence of the output to the command, i.e.,

$$y_k \to r \text{ as } k \to \infty \ .$$

Remark 1: The ability to achieve zero offset in the output versus the command in steady-state despite potential model mismatch or disturbances is known as zero offset or offset free tracking property. To achieve this $n_u \geq n_y$ (more inputs than outputs) is typically required.

Remark 2: Controllers well-designed for constant set-point tracking often perform well and are practically used even when command r is slowly varying.

Tracking LQ-MPC design

Remark 3: If $r \neq 0$, while $e_k \to 0$, we cannot in general expect that $x_k \to 0$ and $u_k \to 0$ as $k \to \infty$. Hence the cost function needs to be somehow changed to penalize the right quantities.

Reference tracking formulation 1

To achieve reference tracking, we could consider the following modified cost function,

$$J_N = \sum_{k=0}^{N-1} (x_k - x_{\text{ref}}(v))^{\text{T}} Q (x_k - x_{\text{ref}}(v)) + (u_k - u_{\text{ref}}(v))^{\text{T}} R (u_k - u_{\text{ref}}(v)),$$

where $x_{\text{ref}}(v)$ and $u_{\text{ref}}(v)$ are the equilibrium pair corresponding to the chosen value of output, $y = v$. We could simply set $v = r$ or we could add an extra integral control loop outside of MPC controller as

$$v_{k+1} = v_k + K_{\text{int}}(y_k - r)$$

where K_{int} is a gain matrix, to help ensure zero steady-state offset (if feasible under constraints) in presence of disturbances/uncertainties. From computational perspective this approach to achieving tracking has relatively low computational complexity (other approaches will require model augmentation), however, an outside integrator has to be used to achieve zero steady-state offset and hence care is needed to avoid integrator windup. The transient response may also not be the best as compared to other approaches but this is problem-dependent.

Reference tracking formulation 2

Another possibility is to consider the following cost function,

$$J_N = \sum_{k=0}^{N-1} e_k^{\mathrm{T}} Q_e e_k + \Delta u_k^{\mathrm{T}} R \Delta u_k,$$

where $\Delta u_k = u_k - u_{k-1}$, $e_k = Cx_k - r$, and $Q_e^{\mathrm{T}} = Q_e \in \mathbb{R}^{n_e \times n_e}$, $R^{\mathrm{T}} = R \in \mathbb{R}^{n_u \times n_u}$, $Q_e \succeq 0$, $R \succ 0$.

Reference tracking formulation 2

Consider the objective to ensure output command tracking, i.e.,

$$y_k = Cx_k \to r.$$

Define an augmented system with the extended state $z_k = [x_k^{\mathrm{T}}\ u_{k-1}^{\mathrm{T}}\ r_k^{\mathrm{T}}]^{\mathrm{T}}$ given by

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ u_k &= u_{k-1} + \Delta u_k \\ r_{k+1} &= r_k \end{aligned}$$

or

$$z_{k+1} = \begin{bmatrix} A & B & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} z_k + \begin{bmatrix} B \\ I \\ 0 \end{bmatrix} \Delta u_k$$

For the augmented system, we can define the tracking error as the output,

$$e_k = y_k - r_k = \begin{bmatrix} C & 0 & -I \end{bmatrix} z_k = Ez_k.$$

Reference tracking formulation 2

The MPC feedback law can be defined based on the following optimal control problem,

$$\underset{\Delta u_0, \Delta u_1, \cdots, \Delta u_{N-1}}{\text{Minimize}} \quad J_N = \sum_{k=0}^{N-1} e_k^{\mathrm{T}} Q_e e_k + \Delta u_k^{\mathrm{T}} R \Delta u_k$$

$$\text{subject to} \quad z_{k+1} = \begin{bmatrix} A & B & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} z_k + \begin{bmatrix} B \\ I \\ 0 \end{bmatrix} \Delta u_k$$

$$e_k = E z_k$$

$$z_0 = \begin{bmatrix} x_0^{\mathrm{T}} & u_{-1}^{\mathrm{T}} & r^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, \; x_0 = \text{current state}$$

$$r_k = r, \;\; k = 0, \cdots, N$$

$$x_{min} \leq x_k \leq x_{max}, \;\; k = 1, \cdots, N$$

$$u_{min} \leq u_k \leq u_{max}, \;\; k = 0, \cdots, N-1$$

(Note: $e_k^{\mathrm{T}} Q_e e_k = z_k^{\mathrm{T}} E^{\mathrm{T}} Q_e E z_k \Rightarrow Q = E^{\mathrm{T}} Q_e E$)

Reference tracking formulation 2

The MPC feedback law is defined by the first optimal move, Δu_0^*, i.e,

$$\Delta u_{MPC}^*(x_0, u_{-1}, r) = \Delta u_0^*.$$

Suppose x_k is the current state, u_{k-1} is the previous control and r_k is the current reference command during the operation. Then the control action is computed according to

$$u_k = u_{k-1} + \Delta u_{MPC}^*(x_k, u_{k-1}, r_k).$$

Remark: Note that we penalize the control increments, Δu_k, rather than actual controls, u_k, since to maintain an output at a nonzero commanded value in steady-state, a non-zero control is (typically) required and driving control to zero does not make sense.

Incremental (rate-based) model

Another LQ-MPC based tracking formulation exploits directly the incremental (rate-based) model and is able to guarantee zero steady state offset if constraints are inactive in presence of constant additive disturbances

Define rates,

$$\Delta x_k = x_{k+1} - x_k, \quad \Delta u_k = u_{k+1} - u_k$$

Then

$$\Delta x_{k+1} = x_{k+2} - x_{k+1} = Ax_{k+1} + Bu_{k+1} - Ax_k - Bu_k = A\Delta x_k + B\Delta u_k$$

$$e_{k+1} = Cx_{k+1} - r = C(x_{k+1} - x_k) + (Cx_k - r) = C\Delta x_k + e_k$$

$$x_{k+1} = x_k + \Delta x_k$$

$$u_{k+1} = u_k + \Delta u_k$$

Incremental (rate-based) model

$$\begin{cases} \Delta x_{k+1} = A\Delta x_k + B\Delta u_k \\ e_{k+1} = C\Delta x_k + e_k \\ x_{k+1} = x_k + \Delta x_k \\ u_{k+1} = u_k + \Delta u_k \end{cases}$$

Reference tracking formulation 3

$$\underset{\Delta u_0, \Delta u_1, \cdots, \Delta u_{N-1}}{\text{Minimize}} \quad J_N = \sum_{k=0}^{N-1} e_k^{\mathrm{T}} Q_e e_k + \Delta u_k^{\mathrm{T}} R \Delta u_k$$

$$\text{subject to} \quad \Delta x_{k+1} = A\Delta x_k + B\Delta u_k$$

$$\Delta x_0 = \text{given}$$

$$e_0 = Cx_0 - r = \text{current tracking error}$$

$$x_0 = \text{ current state}$$

$$u_0 = \text{ current control}$$

$$e_{k+1} = C\Delta x_k + e_k$$

$$x_{k+1} = x_k + \Delta x_k$$

$$u_{k+1} = u_k + \Delta u_k$$

$$x_{min} \leq x_k \leq x_{max}, \ \ k = 0, \cdots, N$$

$$u_{min} \leq u_k \leq u_{max}, \ \ k = 0, \cdots, N-1.$$

Reference tracking formulation 3

This tracking LQ-MPC problem can be re-written as the standard LQ-MPC problem for an extended system with a larger state vector,

$$x_k^{ext} = [\Delta x_k^{\mathrm{T}},\ e_k^{\mathrm{T}},\ x_k^{\mathrm{T}},\ u_k^{\mathrm{T}}]^{\mathrm{T}},$$

and the control input Δu_k. The extended state prediction model is given by

$$x_{k+1}^{ext} = \begin{bmatrix} A & 0 & 0 & 0 \\ C & \mathbb{I}_{n_e \times n_e} & 0 & 0 \\ \mathbb{I}_{n_x \times n_x} & 0 & \mathbb{I}_{n_x \times n_x} & 0 \\ 0 & 0 & 0 & \mathbb{I}_{n_u \times n_u} \end{bmatrix} x_k^{ext} + \begin{bmatrix} B \\ 0 \\ 0 \\ \mathbb{I}_{n_u \times n_u} \end{bmatrix} \Delta u_k$$

Remark: The states x_k, u_k in rate-based model are only included to enforce the constraints. If certain state or control channels are not constrained, the corresponding state or control variables can be removed from the rate-based model.

Reference tracking formulation 3

For the extended system, the state penalty matrix has the form,

$$Q^{ext} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & Q_e & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

The control penalty matrix is $R^{ext} = R$.

The cost becomes

$$J_N = \sum_{k=0}^{N-1} (x_k^{ext})^{\mathrm{T}} Q^{ext} x_k^{ext} + \Delta u_k^{\mathrm{T}} R \Delta u_k$$

Reference tracking formulation 3

- Adding a terminal penalty to the cost may also be advantageous to ensure closed-loop stability.

- Unfortunately, the extended system may not be controllable due to augmentation of constrained variables as states. Hence DARE/DLQR may not be directly used to generate the terminal penalty.

- A terminal penalty on Δx_N and e_N only may, however, be defined based on the solution to DLQR/DARE corresponding to dynamics and input matrices of the form

$$\begin{bmatrix} A & 0 \\ C & \mathbb{I}_{n_e \times n_e} \end{bmatrix}, \begin{bmatrix} B \\ 0 \end{bmatrix},$$

and *some* state and control weighting matrices selected by the designer, e.g.,

$$\begin{bmatrix} 0 & 0 \\ 0 & Q_e \end{bmatrix}, R,$$

to have the terminal cost consistent with the incremental cost. Let $\tilde{P}_\infty$ denote the solution to the corresponding DARE.

Reference tracking formulation 3

Then with the terminal penalty added, the cost function becomes

$$J_N = (x_N^{ext})^{\mathrm{T}} P_N x_N^{ext} + \sum_{k=0}^{N-1} (x_k^{ext})^{\mathrm{T}} Q^{ext} x_k^{ext} + \Delta u_k^{\mathrm{T}} R \Delta u_k$$

$$P_N = \begin{bmatrix} \tilde{P}_\infty & 0 \\ 0 & 0 \end{bmatrix}$$

Reference tracking formulation 3

The tracking LQ-MPC feedback law is defined by the first element of the optimal solution sequence that is

$$\Delta u_{MPC}(\Delta x_0,\ e_0,\ x_0,\ u_0) = \Delta u_0^*.$$

When implementing MPC, this feedback law is applied to the current values of the state increment, $\Delta x_t = x_t - x_{t-1}$, tracking error, $e_t = Cx_t - r$, state, x_t and control u_t, i.e., the controller has the form,

$$u_{t+1} = u_t + \Delta u_{MPC}(\Delta x_t, e_t, x_t, u_t).$$

$$\Delta x_t = x_t - x_{t-1},\ e_t = Cx_t - r$$

Note that this controller includes integral action at the output. The integral action helps ensure zero steady-state offset in tracking constant commands. It also ensures zero steady-state offset if constant unmeasured disturbances are acting on the right hand side.

Reference tracking formulation 3

Remark 1: If the problem is unconstrained (or constraints are inactive locally near the set-point), as $N \to \infty$, the MPC feedback law is expected to approach the LQR feedback law (assuming LQR solution exists) and be stabilizing for the model in the incremental form (with states $[\Delta x^{\mathrm{T}} \quad e^{\mathrm{T}}]^{\mathrm{T}}$ and control Δu), thereby guaranteeing $e_k \to 0$ as $k \to \infty$. This property of zero steady-state offset error, is also expected to hold under constant additive disturbances to the system dynamics.

Remark 2: Suppose the terminal penalty, selected based on the above procedure, is used. Then if the constraints are inactive near the set-point, the MPC feedback law coincides with LQR feedback law for the model in the incremental form for any N and hence $\tilde{e}_k \to 0$ as $k \to \infty$. This property of zero steady-state offset error, will also hold under constant additive disturbances to the system dynamics.

Reference tracking formulation 3

If the system is affected by a constant disturbance, $w_k = w$, and the model has the form,

$$x_{k+1} = Ax_k + Bu_k + B_w w_k,$$

the model in the incremental form

$$\Delta x_{k+1} = A\Delta x_k + B\Delta u_k + B_w \Delta w_k$$

$$e_{k+1} = e_k + C\Delta x_k$$

will remain the same as

$$\Delta w_k = w_k - w_{k-1} = 0$$

Consequently, the controller, at least in the unconstrained case, is expected to reject constant disturbances with zero steady-state offset ($y_k \to r$ as $k \to \infty$ irrespective of $w_k = w$).

Matlab software

Multi-parametric Toolbox 3 for Matlab

http://people.ee.ethz.ch/~mpt/3/

Hybrid MPC Toolbox for Matlab

http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox/

Double Integrator Example

The model is given by

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = u + w$$

The sampling period is $T_s = 0.1$ sec. The constraints are given by

$$-1 \leq x_1 \leq 1.1, \quad -0.2 \leq x_2 \leq 0.2, \quad -0.1 \leq u \leq 0.1.$$

The objective is to achieve offset-free tracking of feasible (under constraints) desired position commands, i.e.,

$$x_1(t) \to r(t)$$

if the command $r(t)$ is constant, and the unmeasured disturbance w is constant.

MPT3 Double Integrator Example

```
clear all
close all

% Continuous time model
    Ac = [0,1;0,0];
    Bc = [0;1];
    Cc = [1,0;0,1];
    Dc = [0;0];

% Discrete-time model
    Ts = 0.1;  % sampling period, sec
    sysd = c2d( ss(Ac, Bc, Cc, Dc), Ts );
    Ad = sysd.A;
    Bd = sysd.B;
    Cd = sysd.C;
    Dd = sysd.D;
```

MPT3 Double Integrator Example

```
% Augmented model for tracking: states (x, u, r) and control du
    Ax = [Ad           Bd    0*Ad(:,1)
          0*Ad(1,:)     1    0
          0*Ad(1,:)     0    1      ] ; % states are x, u, r

    Bx = [Bd;1;0] ;                     % delta u  = new control

    Cx = [1,  0,      0,      0;
          0,  1,      0,      0;
          0,  0,      1,      0;
          0,  0,      0,      1];

    Dx = [0;0;0;0];

    Ex = [Cd(1,:) 0 -1] ;  % error = position - set-point

    model = LTISystem('A',Ax,'B',Bx,'C',Cx,'D',Dx, 'Ts', Ts);
```

MPT3 Double Integrator Example

```
% Control constraints
    model.u.min= -1;
    model.u.max= -model.u.min;

% State constraints
    model.x.min=[-1, -0.2, -0.1, -1];
    model.x.max=[1.1, 0.2,  0.1,  1];

% Cost function
    model.u.penalty = QuadFunction(diag([0.1]))
    model.x.penalty = QuadFunction(Ex'*Ex)

% Horizon
    N=12;

% Generate MPC Controller
    ctrl = MPCController(model,N)
```

MPT3 Double Integrator Example

```
% Simulate closed loop response
    x0 = [0;0];
    u  = 0;
    r0 = 1;
    Nsim = 300;

    x = x0;
    data.X = zeros(length(x0),Nsim);
    data.U = zeros(size(Bd,2),Nsim);
    data.R = zeros(1,Nsim);
    data.W = zeros(1,Nsim);
```

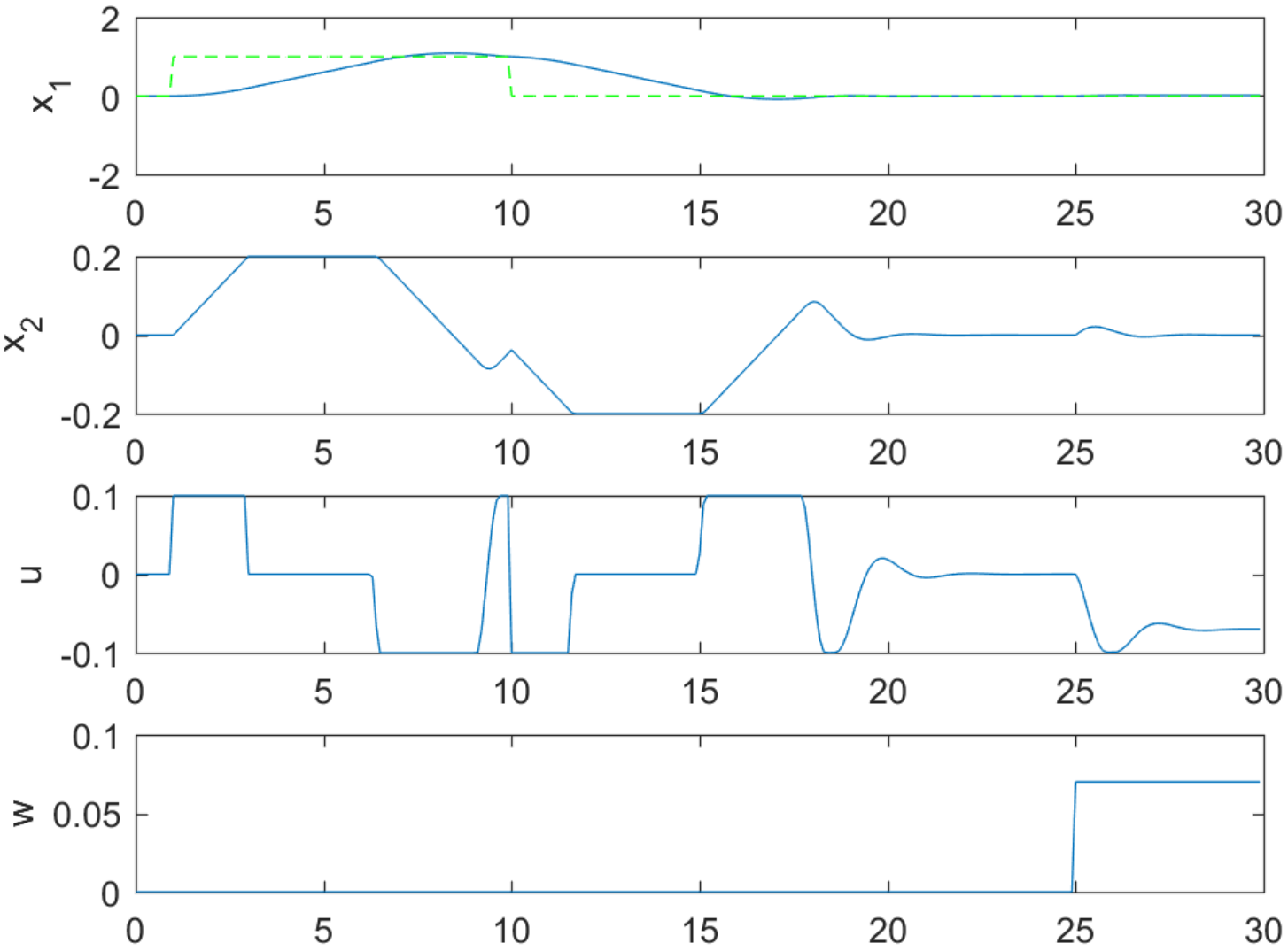
```
for i=1:Nsim,
        if (i-1)*Ts<1,
            r=0; w=0;
        elseif (i-1)*Ts<10,
            r = r0; w = 0.0;
        elseif (i-1)*Ts<25,
            r = 0;
        else,
            r = 0;
            w = 0.07;
        end;

        try,
            [du,feasible,openloop] = ctrl.evaluate([x(:); u; r]);
        catch,
            du = 0;
        end
        u = u + du;
        data.X(:,i)=x;
        data.U(:,i)=u;
        data.R(:,i)=r;
        data.W(:,i)=w;
        x = Ad*x + Bd*(u+w);
end;
```

MPT3 Double Integrator Example

```
% Plot the responses
    figure;
    subplot(411)
    plot(([1:Nsim]-1)*Ts, data.X(1,:), ([1:Nsim]-1)*Ts,…
                data.R, 'g--');
    ylabel('x_1')
    subplot(412)
    plot(([1:Nsim]-1)*Ts, data.X(2,:))
    ylabel('x_2')
    subplot(413)
    plot(([1:Nsim]-1)*Ts, data.U(1,:))
    ylabel('u')
    subplot(414)
    plot(([1:Nsim]-1)*Ts, data.W(1,:))
    ylabel('w')
```

MPT3 Double Integrator Example



Hybrid MPC Toolbox for Matlab

```
addpath(...
genpath('C:\Users\ilya\Desktop\MySoftware\hybtbx-win'),...
'-end')

clear variables
```

Hybrid MPC Toolbox for Matlab

```
% Define the linear discrete-time model from input to constrained
output
    Ts    = 0.1; % Sampling time
    model = c2d(ss([0 1;0 0],[0;1],[1,0;0,1],[0;0]),Ts,'zoh');

% Augmented model for tracking. States: (x,u,r). Control: du
    Ad = model.A;
    Bd = model.B;
    Cd = model.C;
    Ax = [Ad           Bd  0*Ad(:,1)
          0*Ad(1,:)    1    0
          0*Ad(1,:)    0    1     ] ; % states are x, u, r

    Bx = [Bd;1;0] ;                    % new control is delta u
    Cx = [1,  0,     0,     0;
          0,  1,     0,     0;
          0,  0,     1,     0;
          0,  0,     0,     1];

    Dx = [0;0;0;0];
    Ex = [Cd(1,:) 0 -1] ;        % error = position - set-point
    sysd   = ss(Ax, Bx, Cx, Dx, Ts) ;
```

Hybrid MPC Toolbox for Matlab

```
% Define cost, constraints and horizon
    clear cost constraints horizon
    Qy = 1;
    cost.Q = Ex'*Qy*Ex ;
    cost.R = 0.1 ;
    cost.P = cost.Q ;
    cost.K = Bx'*0 ;
    Smpc   = ss(Ax,Bx,Cx,Dx,Ts) ;
    cost.rho = 0.1*Inf; % set to Inf for hard constraints

    horizon.N   = 12; % output horizon   \sum_{k=0}^{Ny-1}
    horizon.Nu  = 12; % input horizon    u(0),...,u(Nu-1)
    horizon.Ncu = 11; % input constraints horizon k=0,...,Ncu
    horizon.Ncy = 11; % output constraints horizon k=0,...,Ncy

    constraints.umax =  Inf ;
    constraints.umin = -constraints.umax ;

    constraints.ymax =  [1.1 0.2 0.1 1.1] ; % x, u, r
    constraints.ymin = -constraints.ymax ;

MPCctrl=lincon(sysd,'reg',cost,horizon,constraints,'qpact',0) ;
```

Hybrid MPC Toolbox for Matlab

```
% Simulate closed-loop response
    x0 = [0;0];
    u  = 0;
    r0 = 1;
    Nsim = 300;

    x = x0;
    data.X = zeros(length(x0), Nsim);
    data.U = zeros(size(Bd,2), Nsim);
    data.R = zeros(1, Nsim);
    data.W = zeros(1, Nsim);
```

Hybrid MPC Toolbox for Matlab

```
for i=1:Nsim,
        if (i-1)*Ts<1,
            r=0; w=0;
        elseif (i-1)*Ts<10,
            r = r0; w = 0.0;
        elseif (i-1)*Ts<25,
            r = 0;
        else,
            r = 0;
            w = 0.07;
        end;
        try,
            du = eval(MPCctrl,([x(:);u;r]));
        catch,
            du = 0;
        end
        u = u + du;
        data.X(:,i) = x;
        data.U(:,i) = u;
        data.R(:,i) = r;
        data.W(:,i) = w;
        x = Ad*x + Bd*(u + w);
end;
```

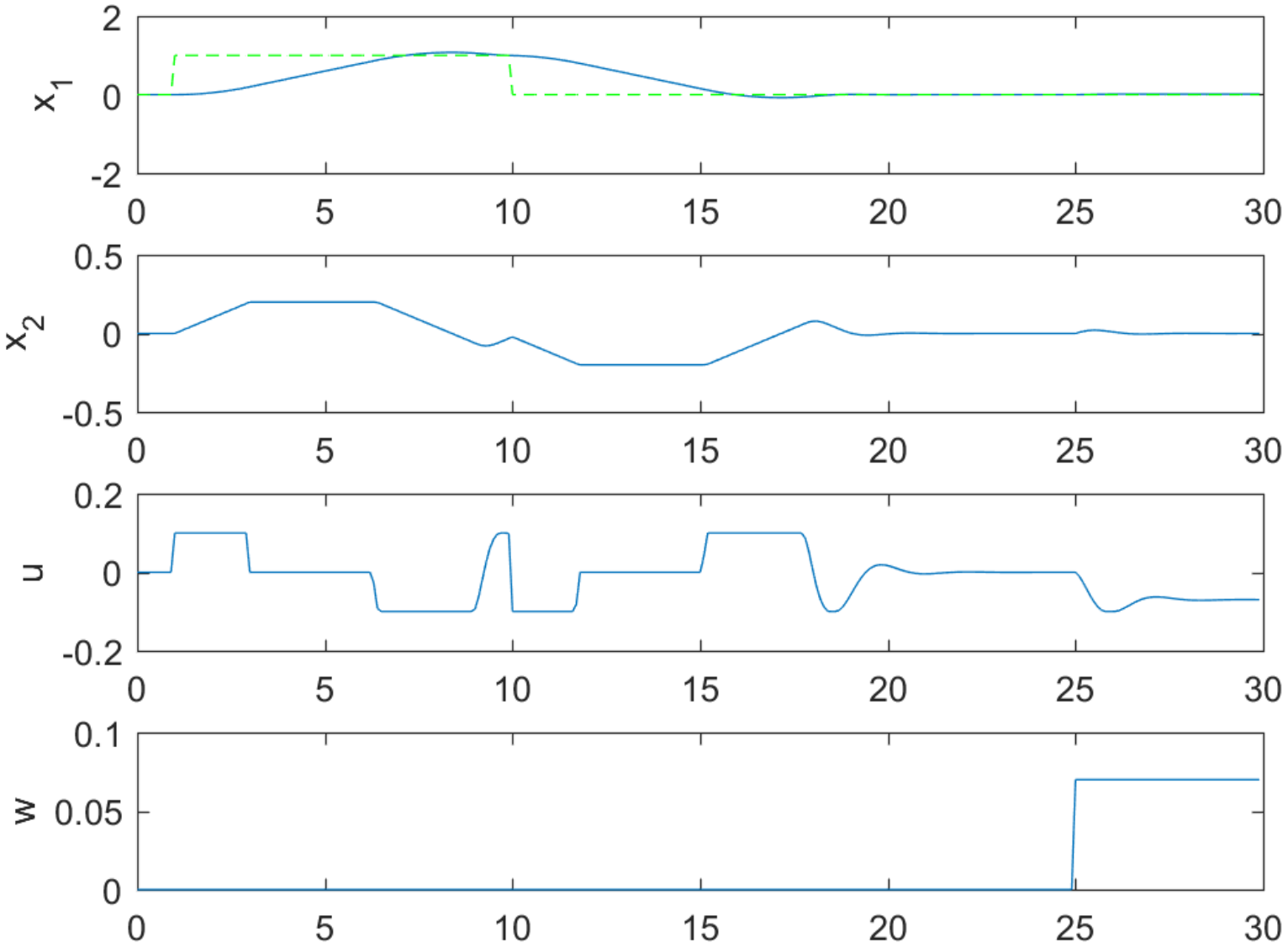
Hybrid MPC Toolbox for Matlab

```
% Plot the responses
    figure;
    subplot(411)
    plot(([1:Nsim]-1)*Ts, data.X(1,:), ([1:Nsim]-1)*Ts,…
              data.R, 'g--');
    ylabel('x_1')
    subplot(412)
    plot(([1:Nsim]-1)*Ts, data.X(2,:))
    ylabel('x_2')
    subplot(413)
    plot(([1:Nsim]-1)*Ts, data.U(1,:))
    ylabel('u')
    subplot(414)
    plot(([1:Nsim]-1)*Ts, data.W(1,:))
    ylabel('w')

rmpath(genpath('C:\Users\ilya\Desktop\MySoftware\hybtbx-win'))

return
```

Hybrid MPC Toolbox for Matlab



Basics of optimization theory

Global and local minima/minimizers

Consider a problem of finding a minimum value of a function $f(x)$ over a set $x \in D \subseteq \mathbb{R}^n$. We can state this problem as

$$f(x) \to \min_{x \in D}.$$

Definition: Suppose $x^* \in D \subset R^n$ and $f(x^*) \cdot \quad f(x)\ \forall x \in D$. Then x^* is a (global) minimizer of f on D and $f(x^*)$ is the minimum value of f on D.

Definition: Suppose $x^* \in D \subset R^n$ and $\exists \epsilon > 0$ such that $f(x^*) \cdot \quad f(x)$ $\forall x \in D \bigcap B(x^*, \epsilon)$. Then x^* is a (local) minimizer of f on D

Fact: x^* is a global minimizer $\Rightarrow$ x^* is a local minimizer.

Strict (unique) minimizers

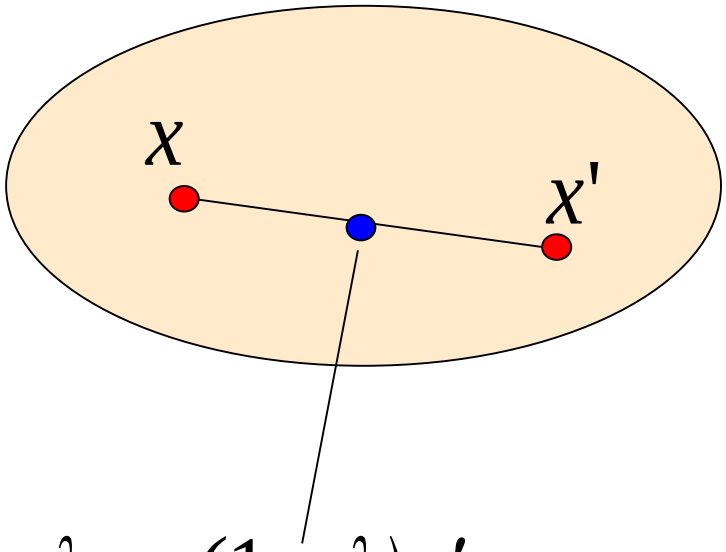
Definition: Suppose $x^* \in D \subset R^n$ and $f(x^*) < f(x)\ \forall x \in D, x \neq x^* \in D$. Then x^* is a strict (global) minimizer of f on D and $f(x^*)$ is the minimum value of f on D.

Definition: Suppose $x^* \in D \subset R^n$ and $\exists \epsilon > 0$ such that $f(x^*) < f(x)$ $\forall x \in D \bigcap B(x^*, \epsilon)$. Then x^* is a strict (local) minimizer of f on D

A strict local minimizer is also referred to as an isolated minimizer.

Fact: x^* is a strict global minimizer $\Rightarrow$ x^* is a strict local minimizer.

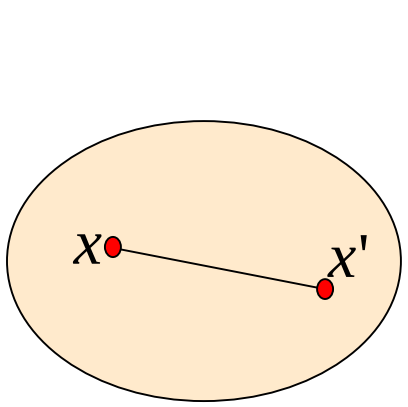
Line segment connecting points in a set



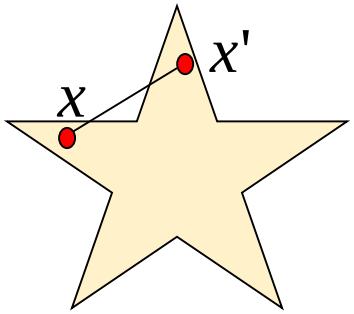
$$\lambda x + (1-\lambda)x' \qquad 0 \leq \lambda \leq 1$$

Convex sets

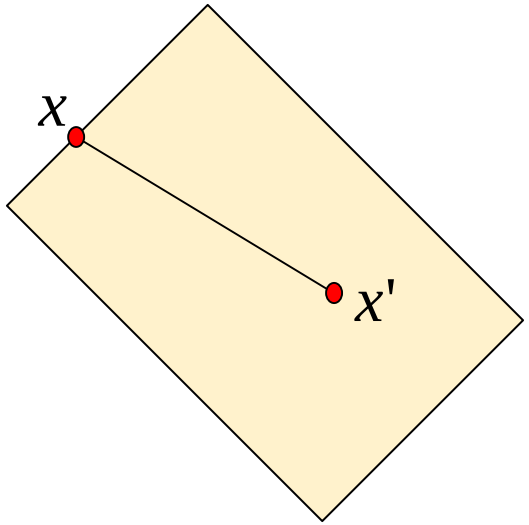
Definition (Convex Set):
A set $D \subset R^n$ is convex, if for any $x, x' \in D$ and any $\lambda \in [0,1]$, we have $\lambda x + (1-\lambda)x' \in D$.



convex



non-convex



convex

Convex sets

Fact (Intersection of Convex Sets):
If D, D' are convex sets then $D \bigcap D'$ is convex.

Fact (Union of Convex Sets):
If D, D' are convex sets then $D \bigcup D'$ may or may not be convex.

Fact: Projection of a convex set is convex.

Let $x = [x_1^{\mathrm{T}},\ x_2^{\mathrm{T}}]^{\mathrm{T}}$, $x_1 \in \mathbb{R}^{n_1}$, $x_2 \in \mathbb{R}^{n_2}$ and $D \subset \mathbb{R}^n$, $n = n_1 + n_2$. Then the projection of D on the subspace defined by the first n_1 coordinates is

$$\Pi_{x_1} D = \{x_1 :\ \exists x_2 \text{ such that } x = [x_1^{\mathrm{T}},\ x_2^{\mathrm{T}}]^{\mathrm{T}} \in D\}$$

Fact: Affine transformation of a convex set is convex.

Let $T(x) = Ax + b$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ be an affine transformation. Then,

$$T(D) = \{z \in \mathbb{R}^m :\ \exists x \in D \text{ such that } z = Ax + b\}$$

Special cases of convex sets

Hyperplanes

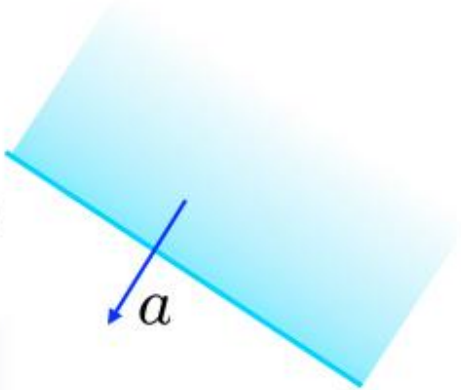
$a'x = b$ → scalar

E.G. : $5x_1 + 3x_2 = 6$

Halfspaces

$a'x \leq b$

E.G. : $5x_1 + 3x_2 \leq 6$

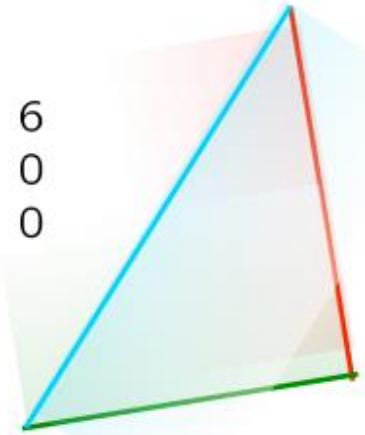


Polyhedra: the intersection of halfspaces
(i.e., a group of linear inequalities)

$Ax \leq b$

vector

E.G. : $\begin{cases} 5x_1 + 3x_2 \leq 6 \\ x_1 \leq 0 \\ x_2 \leq 0 \end{cases}$



A compact (closed+bounded) polyhedron is called polytope
A polytope is uniquely identified by its vertices
A polyhedron is uniquely identified by its vertices and rays

Convex functions

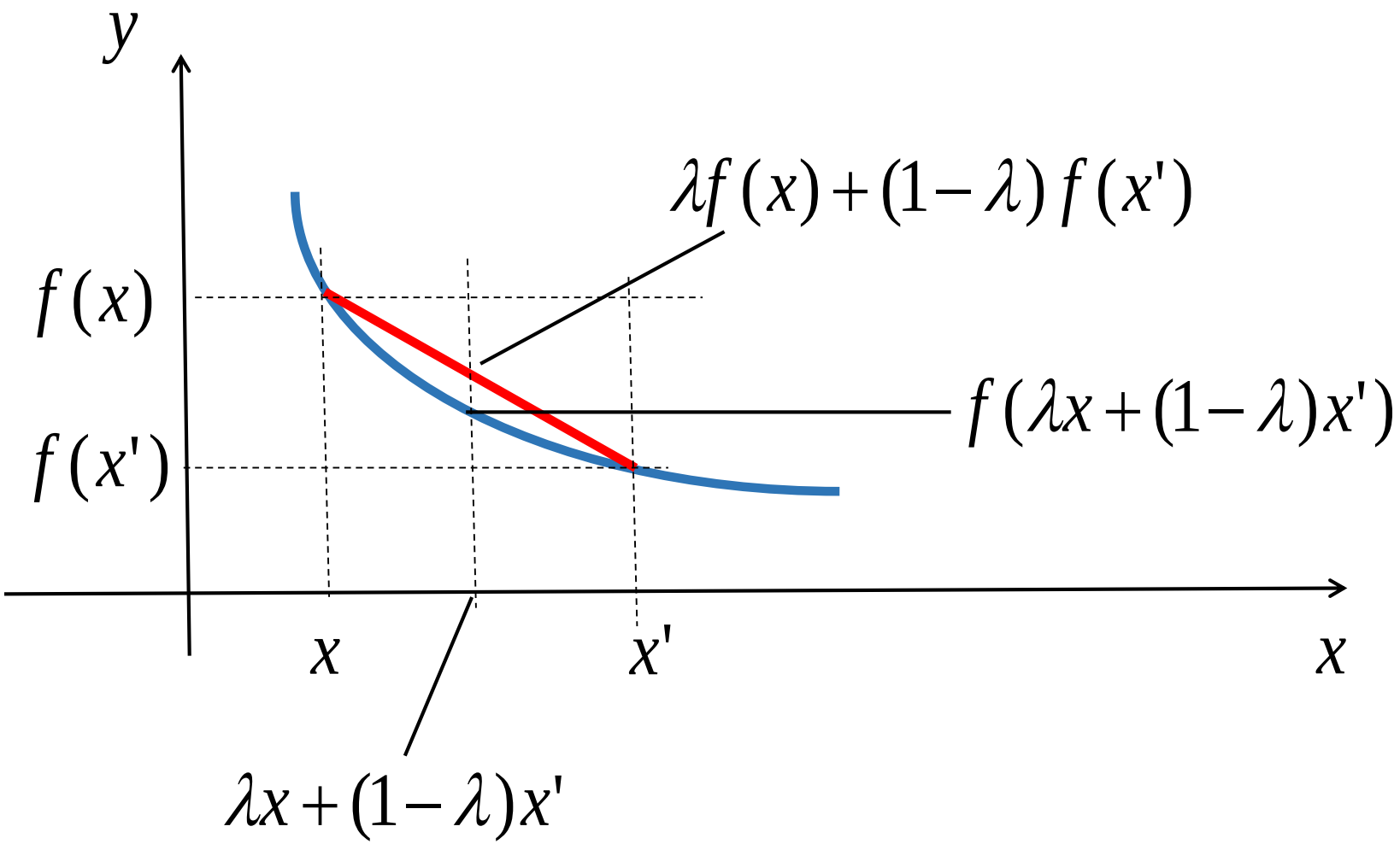
Definition (Convex Function):
A function f defined on a convex set D is called convex if, for any $x, x' \in D$ and all $\lambda \in [0, 1]$,

$$f(\lambda x + (1 - \lambda)x') \leq \lambda f(x) + (1 - \lambda) f(x')$$

Definition (Strictly Convex Function):
A function f defined on a convex set D is called strictly convex if, for any $x, x' \in D$ and all $\lambda \in (0, 1)$,

$$f(\lambda x + (1 - \lambda)x') < \lambda f(x) + (1 - \lambda) f(x')$$

Convex functions



Convex functions

Fact: If $f_1(x)$ and $f_2(x)$ are convex, then so are functions

$$\alpha_1 f_1(x) + \alpha_2 f_2(x), \quad \text{for } \alpha_1, \alpha_2 \geq 0$$

and

$$\max\{f_1(x), f_2(x)\}.$$

Proof: Exercise

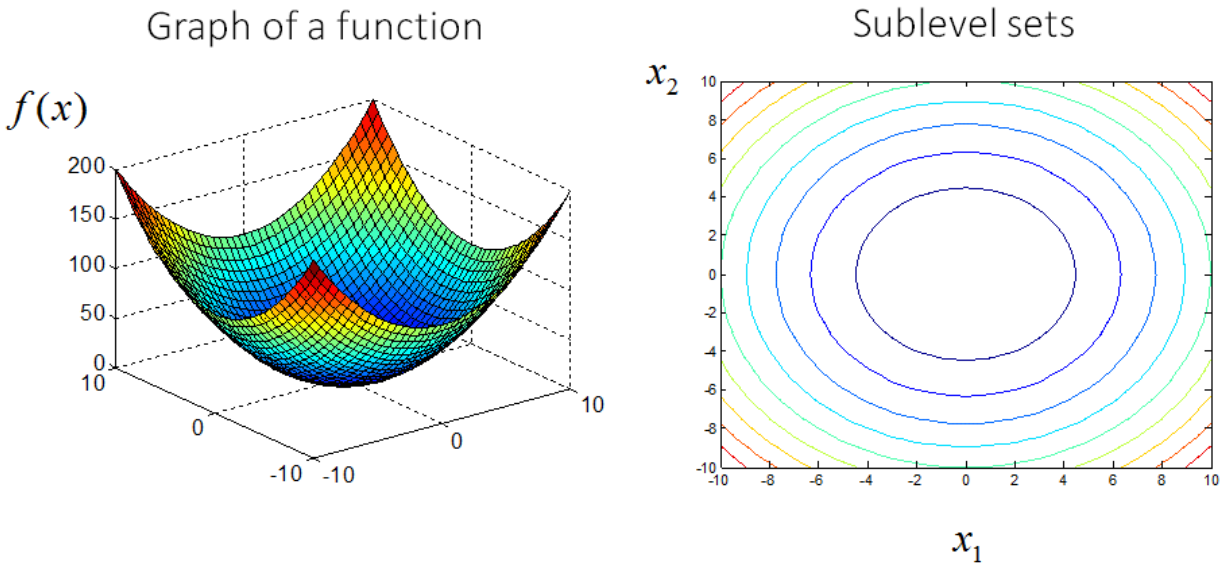
Sublevel sets of convex functions

Fact (Convexity of Sublevel Sets):

If f is a convex function on a convex set $D \subset R^n$ then for any $c \in R$, the sub-level set,

$$X = \{x : \ x \in D \text{ and } f(x) \leq c\},$$

is either empty or non-empty and convex.



Convex functions

Fact (Local Minimizer is a Global Minimizer, Strict Local Minimizer is a unique Global Minimizer):

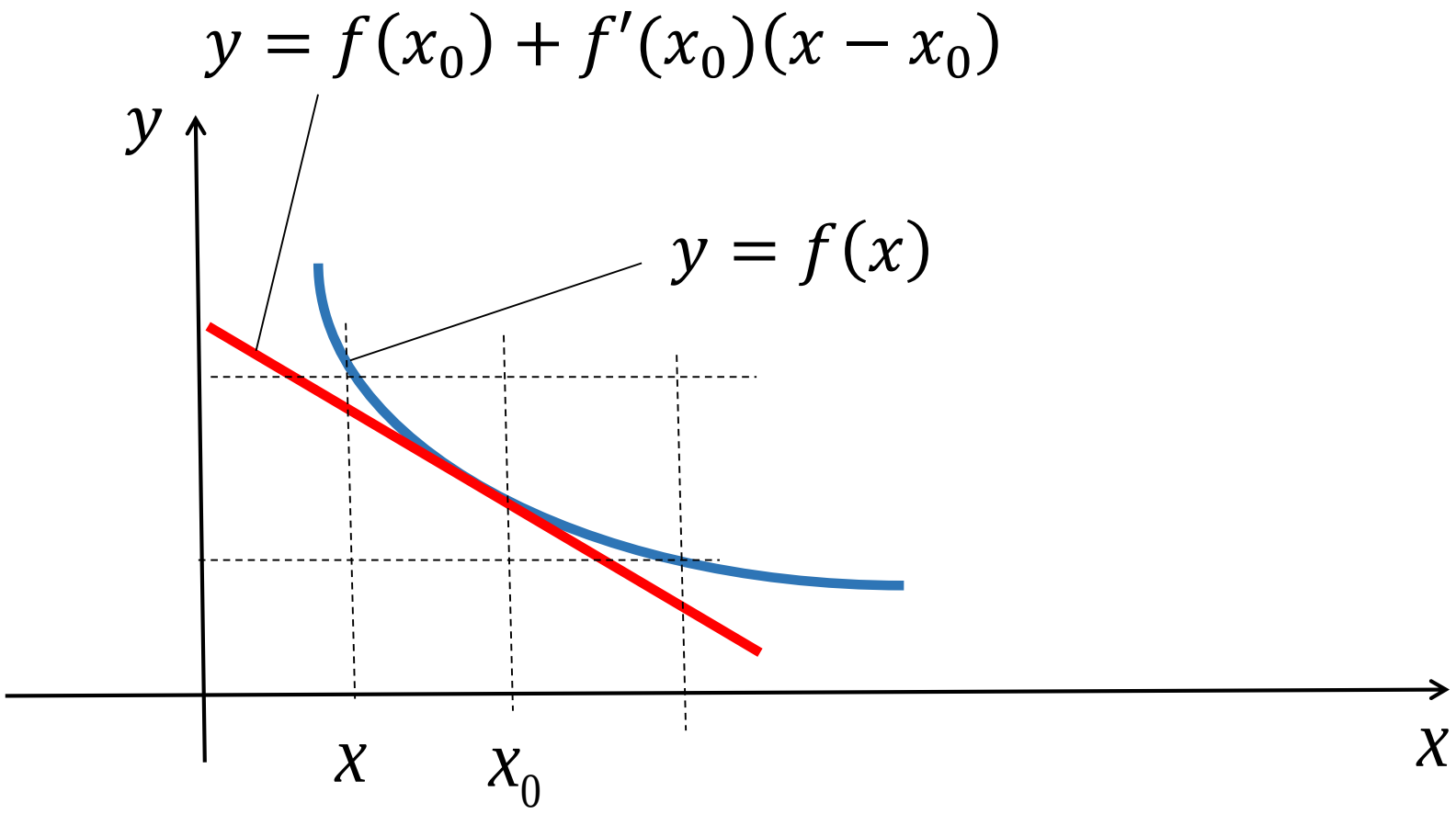
- If a convex function f has a local minimizer, x_0, on a convex set, $D \subset R^n$, then this local minimizer is a global minimizer
- If a convex function f has a strict local minimizer, x_0, on a convex set, $D \subset R^n$, then this local minimizer is the strict global minimizer.

Proof:

First x_0 must be a global minimizer since if $f(x_1) < f(x_0)$ for some $x_1 \in D$, then for $x = (1-\lambda)x_0 + \lambda x_1 \in D$, $f(x) \cdot \quad (1-\lambda)f(x_0) + \lambda f(x_1) = f(x_0) - \lambda(f(x_0) - f(x_1)) < f(x_0)$ for all $0 < \lambda < 1$ including λ arbitrary small; hence x_0 cannot be a strict local minimizer.

If $f(x_1) = f(x_0)$ for some $x_1 \in D$, it follows similarly that for $x = (1-\lambda)x_0 + \lambda x_1 \in D$, and λ arbitrary small, $f(x) \cdot \quad f(x_0)$, which contradicts x_0 being a strict local minimizer.

Convex differentiable functions



$$f(x) \geq f(x_0) + f'(x_0)(x - x_0)$$

Graph of the function lies above the tangent line

Convex differentiable functions

Let D be a non-empty open convex set in R^n and let $f: D \to R$ be differentiable on D. Then f is convex if and only if for any $\bar{x} \in D$,

$$f(x) \geq f(\bar{x}) + f'(\bar{x})(x - \bar{x}), \forall x \in D$$

(The graph of a convex differentiable function lies above a tangent line at any point)

Let D be a non-empty open convex set in R^n and let $f: D \to R$ be differentiable on D. Then f is strictly convex if and only if for any $\bar{x} \in D$,

$$f(x) > f(\bar{x}) + f'(\bar{x})(x - \bar{x}), \forall x \in D, x \neq \bar{x}$$

(The graph of a convex differentiable function lies strictly above a tangent line at any point)

Convexity of smooth functions

Fact (Convexity of Smooth Functions):
If f is twice continuously differentiable (C^2) over a convex open set $D \subset R^n$, then f is convex if and only if the Hessian (matrix of second partial derivatives of f) is nonnegative definite for all $x \in D$.

Moreover, f is strictly convex if the Hessian is positive definite for all $x \in D$ (but converse is not true, Hessian can be positive semi-definite but the function can be strictly convex. Example: $f(x) = x^4$, $f''(0) = 0$.)

$$H = f''(x) = \frac{\partial^2 f}{\partial x^2} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}(x) & \frac{\partial^2 f}{\partial x_2 \partial x_1}(x) & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_1}(x) \\ \frac{\partial^2 f}{\partial x_1 \partial x_2}(x) & \frac{\partial^2 f}{\partial^2 x_2}(x) & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_2}(x) \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n}(x) & \frac{\partial^2 f}{\partial x_2 \partial x_n}(x) & \cdots & \frac{\partial^2 f}{\partial^2 x_n}(x) \end{bmatrix}$$

Hessian matrix

$$f''(x_0) = \frac{\partial^2 f}{\partial x^2} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}(x_0) & \frac{\partial^2 f}{\partial x_2 \partial x_1}(x_0) & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_1}(x_0) \\ \frac{\partial^2 f}{\partial x_1 \partial x_2}(x_0) & \frac{\partial^2 f}{\partial^2 x_2}(x_0) & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_2}(x_0) \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n}(x_0) & \frac{\partial^2 f}{\partial x_2 \partial x_n}(x_0) & \cdots & \frac{\partial^2 f}{\partial^2 x_n}(x_0) \end{bmatrix}$$

If the second partial derivatives all exist and are continuous in a neighborhood of x_0, then the Hessian matrix is symmetric.

Necessary and sufficient conditions for a minimizer of a smooth convex function

Theorem: Let $D \subset R^n$ be a convex set, $f : D \to R$ be a convex function, $x^* \in intD$ ($intD$ = interior of D) and suppose that $f'(x^*)$ exists. Then x^* is a local minimizer of f if and only if $f'(x^*) = 0$. Furthermore, if f is strictly convex, x^* is a global minimizer of f if and only if $f'(x^*) = 0$.

(Note: $f'(x^*) = (\nabla_x f(x^*))^{\mathrm{T}}$ is a row vector of partial derivatives of f)

Proof: Follows from the observation that $f(x) \geq f(x^*) + f'(x^*)(x - x^*) = f(x^*)$ for $x \neq x^*$ if f is convex. Similarly, $f(x) > f(x^*) + f'(x^*)(x - x^*) = f(x^*)$ for all $x \neq x^*$ if f is strictly convex. The converse, that $f'(x^*) = 0$ at a minimizer, holds true even in the nonconvex case.

Constrained minimization problems

Consider the following constrained minimization problem:

Minimize an objective function subject to equality and inequality constraints

$$f(x) \to \min$$

subject to

$$g_i(x) = 0, i = 1, \cdots, m,$$

$$h_j(x) \leq 0, \ j = 1, \cdots, l.$$

$$x \in R^n$$

Assume all functions are continuously differentiable (C^1).

Karush-Kuhn-Tucker (KKT) conditions

Let

$$p = \begin{bmatrix} p_1 \\ \vdots \\ p_m \end{bmatrix} \in R^m \quad \lambda = \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_l \end{bmatrix} \in R^l,$$

be Lagrange multipliers, and introduce the Lagrangian

$$\begin{aligned} \mathcal{L}(x, p, \lambda) &= f(x) + p^{\mathrm{T}} g(x) + \lambda^{\mathrm{T}} h(x) \\ &= f(x) + \sum_{i=1}^{m} p_i g_i(x) + \sum_{j=1}^{l} \lambda_j h_j(x) \end{aligned}$$

KKT conditions

Suppose x^* is a local or a global minimizer. Under a constraint qualification condition, there exist p and λ such that the following necessary conditions hold

$$\nabla_x L(x^*, p, \lambda) = 0$$

$$\lambda \geq 0$$

$$\lambda^{\mathrm{T}} h(x^*) = 0$$

$$g(x^*) = 0$$

$$h(x^*) \leq 0$$

Remark: In general constrained optimization problems, a classical constraint qualification condition is Linear Independence Coonstraint Qualification (LICQ). The gradients of inequality constraints and active inequality constraints at x^* must be linearly independent. However, there are several others such as Slater's condition in convex problems (to be discussed).

Terminology

$$\nabla_x \mathcal{L}(x^*, p, \lambda) = \nabla f(x^*) + \sum_{i=1}^{m} p_i \nabla g_i(x^*) + \sum_{i=1}^{l} \lambda_i \nabla h_i(x^*) = 0$$

(stationarity condition)

$$h_i(x^*) \leq 0,\ i = 1, \cdots, l,\ g_j(x^*) = 0,\ j = 1, \cdots, m$$

(primal feasibility condition)

$$\lambda_i \geq 0,\ i = 1, \cdots, l$$

(dual feasibility condition)

$$\lambda_i = 0 \text{ if } h_i(x^*) < 0 \ \ (\leftrightarrow \lambda^{\mathrm{T}} h(x^*) = 0)$$

(complementary slackness condition)

The numbers $p_1, \cdots, p_m, \lambda_1, \cdots, \lambda_l$ are called Lagrange multipliers

Constraint qualification conditions

Linear Constraint Qualification Condition (LCQC) holds if the constraints h_i and g_j are affine.

Linear Independence Constraint Qualification Condition (LICQC) holds at x^* if the gradients of the equality constraints and active inequality constraints at x^*, $\{\nabla g_i(x^*),\ i = 1, \cdots, m,\ \nabla h_j(x^*),\ j \in \mathcal{A}(x^*)\}$ are linearly independent.

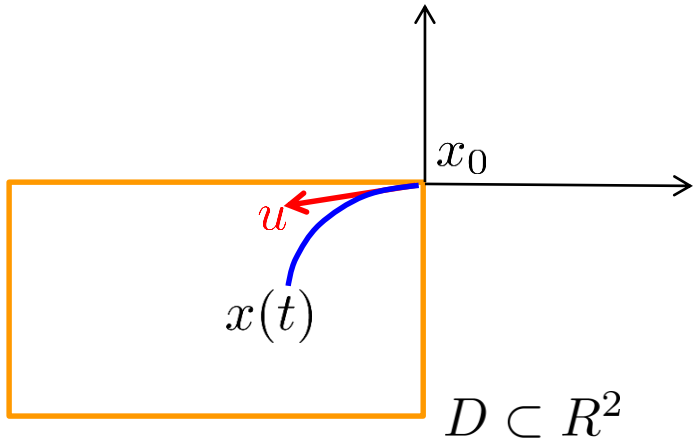
Mangasarian Fromovitz Constraint Qualification (MFCQC) holds at x^* if there exists a $v \in \mathbb{R}^n$ such that $(\nabla h_i(x^*))^{\mathrm{T}} v < 0$ for all $i \in \mathcal{A}(x^*)$, and $(\nabla g_j(x^*))^{\mathrm{T}} v = 0$ for all $j = 1, \cdots, m$. MF holds vacuously if $\mathcal{A}(x^*) = \emptyset$ and there are no equality constraints.

Slater's Constraint Qualification holds at x^* if $\exists \bar{x} \in R^n$ such that $h_i(\bar{x}) < 0$, $i \in \mathcal{A}(x^*)$, $g(\bar{x}) = 0$, where f, h_i, $i = 1, \cdots, l$ and g_i, $i = 1 \cdots, m$ are convex and C^1.

Tangent cone

Consider a constrained minimization problem, $f(x) \to \min$ subject to $x \in D$. A tangent cone is defined as

$$T(D, x_0) = \{u \in R^n : \ x(t) = x_0 + tu + o(t) \in D \text{ for all } t \geq 0 \text{ sufficiently small}\}$$



In other words, a tangent cone to D at x_0 is a set of all tangent vectors, $u = \dot{x}(0)$, to all smooth curves $x(t)$ with $x(0) = x_0$ such that $x(t) \in D$ for all $t \geq 0$ sufficiently small.

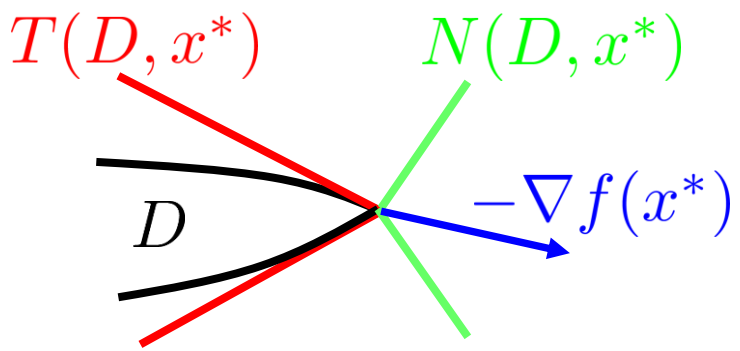
Normal cone and necessary conditions for the minimizer

Define the normal cone as the set of directions forming an angle of greater or equal to 90 deg with all the directions in the tangent cone, i.e.,

$$N(D, x_0) = \{v \in R^n : \ u^{\mathrm{T}} v \leq 0 \text{ for all } u \in T(D, x_0)\}.$$

Then, the necessary condition for the local minimizer that $f(x)$ must non decrease along curves that pass through x^* and stay in D implies $f'(x^*)u \geq 0$ for all $u \in T(D, x^*)$. This in turn can be restated as an equivalent necessary condition,

$$-\nabla f(x^*) = -(f'(x^*))^{\mathrm{T}} \in N(D, x^*).$$



Normal cone is spanned by normal to the constraints with positive coefficients

Suppose now that the set D is defined by l inequality constraints, that are continuously differentiable (C^1) functions, i.e.,

$$D = \{x \in R^n : \ h_i(x) \leq 0, i = 1, \cdots, l\},$$

where $h_i : \ R^n \to R$ and $h_i \in C^1$. Let $x^0 \in D$ be given and $\mathcal{A}(x_0) = \{i \in \{1, 2, \cdots, l\} : \ h_i(x_0) = 0\}$ be the active set, i.e., the set of constraints which hold as equalities rather than strict inequalities at x_0.

Suppose that the gradients of the active constraints at x_0, i.e., vectors $\nabla h_i(x_0)$, $i \in \mathcal{A}(x_0)$, are linearly independent. That is, the so called Linear Independence Constraint Qualification Condition (LICQC) holds.

Then:

$$T(D, x_0) = \{u \in R^n : \ h_i'(x_0)u \leq 0, \ \forall i \in \mathcal{A}(x_0)\}.$$

$$N(D, x_0) = \{v \in R^n : \ \exists \lambda_i \geq 0, \ i \in \mathcal{A}(x_0), \text{ such that } v = \sum_{i \in \mathcal{A}(x_0)} \lambda_i \nabla h_i(x_0)\}.$$

Finding minimizers using KKT conditions

* Assume a subset of inequality constraints that are active [correspond to h_i with $i \in \mathcal{A} \subset \{1, \cdots, l\}$ such that $h_i(x^*) = 0$] and inactive [correspond to h_i with $i \in \mathcal{I} \subset \{1, \cdots, l\}$ such that $h_i(x^*) < 0$]. Equality constraints are treated as active.

* Set $\lambda_i = 0$ for each $i \in \mathcal{I}$.

* Find all solutions to the nonlinear equations which follow from necessary conditions (these have same number of equations and unknowns):

$$\sum_{i=1}^{m} p_i \nabla g_i(x) + \sum_{i \in \mathcal{A}} \lambda_i \nabla h_i(x) = 0$$

$$g_i(x) = 0 \text{ for all } i = 1, \cdots, m$$

$$h_i(x) = 0 \text{ for all } i \in \mathcal{A}$$

Solve for x (n numbers), p_i, $i = 1, \cdots, m$ (m numbers) and λ_i, $i \in \mathcal{A}$ (as many numbers as the number of active constraints)

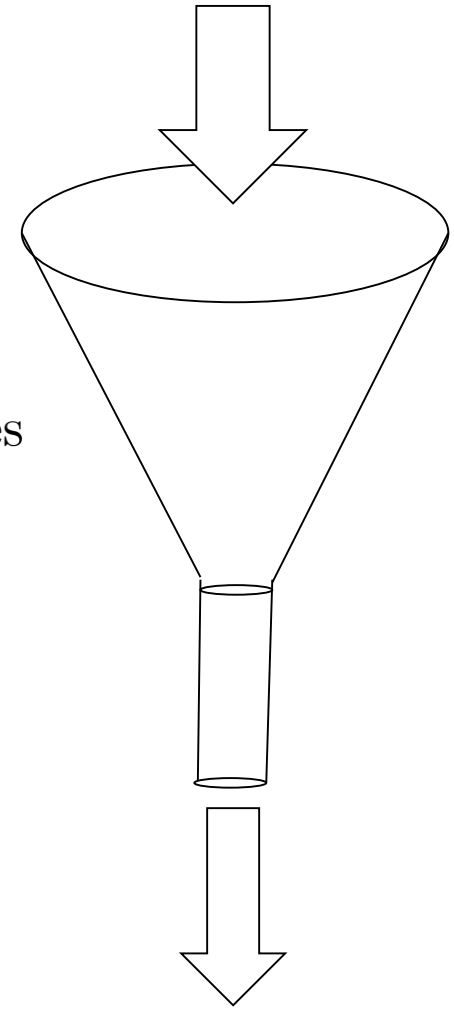
Notes on Application of KKT NC

- Since which constraints are active at a minimizer and which are not is not a priori known, one has to consider all possible combinations of active and inactive constraints and find solution candidates for each combination
- If for a solution candidate x^c, there are any multipliers with $\lambda_i < 0$ for some $i \in \mathcal{A}$ or if $h_i(x^c) > 0$ for any i, this solution candidate can be immediately rejected
- For some combications of active and inactive constraints, no feasible solution candidates may exist
- Once all solution candidates have been determined for all possible combinations of active and inactive constraints, the solution candidate which minimizes the objective function value is chosen.

Notes on application of KKT NC

Solution candidates (all ones that satisfy necessary conditions)

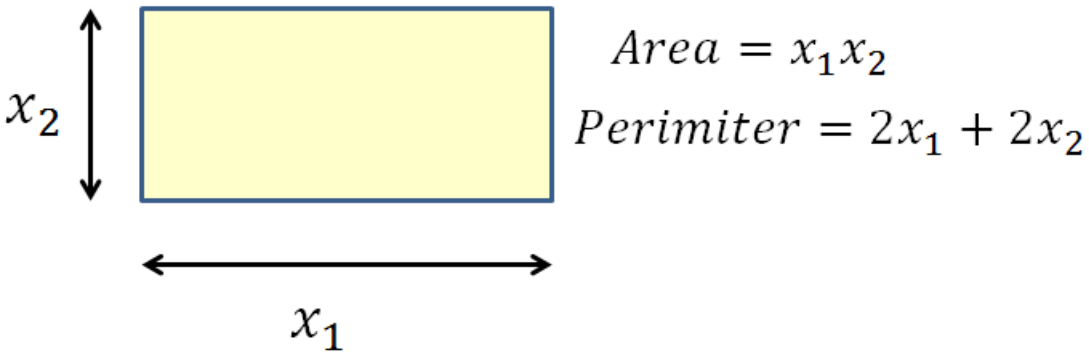
Check function values



The Solution

Example

Consider the problem of finding a rectangle that has the *maximum* area subject to the constraint that its perimeter does not exceed a specified value of $l > 0$.



Example (cont'd)

Reformulate the problem in the standard form as a minimization problem of a function

$$f(x) = -x_1 x_2 \to \min$$

subject to inequality constraints

$$h_1(x) = 2x_1 + 2x_2 - l \leq 0,$$

$$h_2(x) = -x_1 \leq 0,$$

$$h_3(x) = -x_2 \leq 0.$$

Example (cont'd)

Note that the second and third constraint cannot be active at the minimizer.

Indeed if either $x_1^* = 0$ or $x_2^* = 0$ at the minimizer, then $f(x^*) = 0$ while we know that for a feasible solution, $\tilde{x} = (\frac{l}{10}, \frac{l}{10})$, $f(\tilde{x}) = -\frac{l^2}{100} < 0$.

We only need to consider the active set possibilities as $\mathcal{A} = \emptyset$ and $\mathcal{A} = \{1\}$.

Example (cont'd)

For $\mathcal{A} = \emptyset$ there are no active constraints and their gradients (none) are linearly independent. For $\mathcal{A} = \{1\}$, the linear independence CQC trivially holds since

$$\nabla h_1(x) = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \neq 0.$$

Hence we can use the KKT conditions.

Example (cont'd)

Define the Lagrangian,

$$\mathcal{L} = -x_1 x_2 + \lambda_1(2x_1 + 2x_2 - l).$$

Note that we while we could also include $\lambda_2 h_2 + \lambda_3 h_3$ into the Lagrangian, we do not have to. Indeed, $\lambda_2 = 0$ and $\lambda_3 = 0$ since neither the second nor third constraint can be active at the minimizer.

Consider the stationarity condition,

$$\nabla_x \mathcal{L} = 0 \Rightarrow \begin{bmatrix} -x_2 + 2\lambda_1 \\ -x_1 + 2\lambda_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Clearly,

$$x_1 = 2\lambda_1, \quad x_2 = 2\lambda_1$$

Example (cont'd)

If $\mathcal{A} = \emptyset$, then $\lambda_1 = 0$ and $x_1 = x_2 = 0$. This satisfies the inequality constraints and is a solution candidate.

If $\mathcal{A} = \{1\}$, then $2x_1 + 2x_2 = l$ implying together with $x_1 = x_2$ that $x_1 = x_2 = \frac{l}{4}$ is a solution candidate.

The function value $f(x)$ is lower for $x_1 = x_2 = \frac{l}{4}$ than for $x_1 = x_2 = 0$. Since the solution can be shown to exist by a corollary from Weierstrass theorem (AEROSP 575), $x^* = (\frac{l}{4}, \frac{l}{4})$ is the solution.

Example (cont'd)

To check second order conditions, we compute the Hessian of the Lagrangian as

$$\mathcal{L}^{''}(x^*) = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}.$$

Note that $\mathcal{L}^{''}(x^*)$ is not a positive semi-definite matrix. The constraint $h_1(x) = 2x_1 + 2x_2 - l = 0$ is the only active constraint at x^*. Forming the Jacobian matrix of active constraints $J(x^*)$,

$$J(x^*) = h_1^{'}(x^*) = \begin{bmatrix} 2 & 2 \end{bmatrix}.$$

The columns of the matrix $N(x^*)$ are basis vectors of the nullspace of the matrix $J(x^*)$. We have,

$$J(x^*)\begin{bmatrix} \zeta_1 \\ \zeta_2 \end{bmatrix} = \begin{bmatrix} 2 & 2 \end{bmatrix}\begin{bmatrix} \zeta_1 \\ \zeta_2 \end{bmatrix} = 0 \Rightarrow 2\zeta_1 + 2\zeta_2 = 0 \Rightarrow \zeta_1 = -\zeta_2,$$

Setting $\zeta_1 = 1$, $\zeta_2 = -1$ we can choose,

$$N(x^*) = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Example (cont'd)

Now consider the condition that the Hessian of the Lagrangian must be positive semi-definite (in necessary conditions) or positive definite (in sufficient conditions) on all the directions in the tangent space spanned by active constraints. Checking this condition amounts to computing

$$N(x^*)^T \mathcal{L}''(x^*) N(x^*) = \begin{bmatrix} 1 & -1 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = 2 > 0$$

Note also that $\lambda_1 = x_1^*/2 > 0$. Thus both second order necessary conditions and second order sufficient conditions for x^* to be a (local) minimizer hold.

Example (cont'd)

Note that the necessary and sufficient conditions for convex functions cannot be used in this problem since the function f is not convex.

To see that it is not convex, one can compute its Hessian and confirm that its Hessian is not a positive semi-definite matrix.

First order sufficient conditions for convex problems

Theorem: Assume f is convex, $h_1, \cdots, h_l$ are convex, and g is affine (i.e., $g = Ax + b$). Also assume $f, h_1, \cdots, h_l$ are C^1 on R^n. Let $x^* \in D = \{x : \ g_i(x) = 0, i = 1, \cdots, m, h_i(x) \leq 0, i = 1, \cdots, l\}$ be such that

$$\nabla_x \mathcal{L} = \nabla f(x^*) + \sum_{i=1}^{m} p_i \nabla g_i(x^*) + \sum_{i=1}^{l} \lambda_i \nabla h_i(x^*) = 0$$

$$\sum_{i=1}^{l} \lambda_i h_i(x^*) = 0$$

for some numbers $p_1, \cdots, p_m, \lambda_1, \cdots, \lambda_l$ such that

$$\lambda_i \geq 0, \quad i = 1, \cdots, l.$$

Then x^* is a minimizer of f on D.

First order sufficient conditions for convex problems

Proof: Note that convexity of f, h_i, g_i and $-g_i$ and $q_i \geq 0$ imply

$$\bar{\mathcal{L}}(x) = f(x) + \sum_{i=1}^{m} p_i g_i(x) + \sum_{i=1}^{l} \lambda_i h_i(x)$$

is convex. Then $\nabla_x \bar{\mathcal{L}}(x^*) = 0$ implies that x^* is a minimizer of $\bar{\mathcal{L}}(x)$ on R^n and hence on

$$D = \{x : \ g_i(x) = 0, i = 1, \cdots, m, \ h_i(x) \leq 0, i = 1, \cdots, l\}.$$

But for $x \in D$, $g_i(x) = 0$ and $\lambda_i h_i(x) \leq 0$. Thus

$$\bar{\mathcal{L}}(x) = f(x) + \sum_{i=1}^{m} p_i g_i(x) + \sum_{i=1}^{l} \lambda_i h_i(x) \leq f(x) \text{ for all } x \in D.$$

First order sufficient conditions for convex problems

Proof (cont'd): On the other hand,

$$f(x^*) = f(x^*) + \sum_{i=1}^{l} \lambda_i h_i(x^*) + \sum_{i=1}^{m} p_i g_i(x^*) = \bar{\mathcal{L}}(x^*) \leq \bar{\mathcal{L}}(x) \leq f(x) \text{ for all } x \in D,$$

since x^* is a minimizer of $\bar{\mathcal{L}}$.

First order necessary conditions for convex problems

Suppose Slater's Constraint Qualification condition,

$$\exists \bar{x} \in R^n \text{ such that } h_i(\bar{x}) < 0,\ i \in \mathcal{A}(x^*),\ g(\bar{x}) = 0,$$

holds, where $\mathcal{A}(x^*)$ denotes the set of active inequality constraints at x^*, and f, h_i, $i = 1, \cdots, l$ and g_i, $i = 1 \cdots, m$ are convex and C^1.

Then KKT conditions are necessary for x^* to be a minimizer.

In case g is not only convex but also affine, KKT conditions are both necessary and sufficient for x^* to be a minimizer.

Second order necessary conditions

Assume linear independence constraint qualification condition holds, and f, g, h are twice continuously differentiable (C^2). Then at the minimizer x^* of $f(x)$ subject to $g(x) = 0$, $h(x) \leq 0$, and for the corresponding vectors of Lagrange multipliers p^*, λ^*, which satisfy the KKT conditions, the following matrix must be positive semi-definite:

$$N^{\mathrm{T}}(x^*)\mathcal{L}''(x^*, p^*, \lambda^*)N(x^*) \geq 0$$

where the Hessian matrix of the Lagrangian is given by

$$\mathcal{L}''(x^*, p^*, \lambda^*) = f''(x^*) + \sum_{i=1}^{m} p_i^* g_i''(x^*) + \sum_{j=1}^{l} \lambda_j^* h_j''(x^*)$$

The columns of the matrix $N(x^*)$ span the nullspace of the Jacobian matrix of active constraints at x^*, $J(x^*)$. Here $J^*(x)$ is a matrix whose rows are derivatives of active constraints at x^*. Active constraints include all equality constraints, $g_i(x) = 0$, $i = 1, \cdots, m$, and all active at x^* inequality constraints, $h_i(x) \leq 0$, $i \in \mathcal{A}(x^*)$.

In other words, the Hessian of the Lagrangian must be positive semi-definite on the tangent subspace of the active constraints at x^*

Second order sufficient conditions

Assume that f, g, h are twice continuously differentiable (C^2). Let x^* be a feasible and a regular point (linear independence CQC holds) with respect to the constraints of the problem, and let $J(x^*)$ denote the Jacobian matrix of active constraints (i.e., the matrix whose rows are derivatives of equality constraints and active inequality constraints) at x^* and $N(x^*)$ be a matrix whose columns span the null space of the matrix $J(x^*)$. Suppose $\exists$ real numbers $\lambda_1^*, \cdots, \lambda_l^*$, $p_1^*, \cdots, p_m^*$ such that for the Lagrangian $\mathcal{L}(x, p, \lambda) = f(x) + \sum_{i=1}^{m} p_i g_i(x) + \sum_{i=1}^{l} \lambda_i h_i(x)$ it follows that

$$\nabla_x \mathcal{L}(x^*, p^*, \lambda^*) = 0,$$

$$\lambda_i^* \geq 0, \ i = 0, \cdots, l,$$

$$\lambda_i^* > 0 \text{ if } h_i(x^*) = 0,$$

$$\lambda_i^* = 0 \text{ if } h_i(x^*) < 0,$$

(case $\lambda_i^* = 0$ for active constraints not allowed)

Suppose that the following matrix is positive-definite

$$N^{\mathrm{T}}(x^*) L''(x^*, p^*, \lambda^*) N(x^*) > 0$$

Then x^* is a strict local minimizer.

KKT conditions and QP problems

Consider a QP problem,

$$\begin{aligned} \underset{U\in\mathbb{R}^{N\cdot n_u}}{\text{Minimize}} \quad & J_N = \frac{1}{2}U^{\mathrm{T}}HU + q^{\mathrm{T}}U \qquad (q = S^{\mathrm{T}}\bar{Q}Mx_0) \\ \text{subject to} \quad & GU \leq W + Tx_0. \end{aligned}$$

Note that $H = H^{\mathrm{T}} > 0$ since $R > 0$ and hence $\bar{R} > 0$, so this QP problem is strictly convex.

In general constrained optimization problems, under suitable constraint qualification conditions, a constrained local minimizer must satisfy the so called Karush-Kuhn-Tucker (KKT) conditions, i.e., KKT conditions are necessary.

For QP problems with $H \geq 0$, KKT conditions are both necessary and sufficient.

If $H > 0$, the global constrained minimizer is unique.

KKT conditions

KKT conditions are based on introducing a vector of Lagrange multipliers, λ, of the same size as the number of constraints, and a Lagrangian of the form:

$$\mathcal{L} = \frac{1}{2}U^{\mathrm{T}}HU + q^{\mathrm{T}}U + \lambda^{\mathrm{T}}(GU - W - Tx_0)$$

For $U = U^*$ to be a minimizer it is necessary, under Slater's CQC, and sufficient that there there exists a Lagrange multipler, $\lambda = \lambda^*$, such that (U^*, λ^*) jointly satisfy the following KKT conditions.

Stationarity: $\nabla_U \mathcal{L} = 0 \Rightarrow HU + q + G^{\mathrm{T}}\lambda = 0$

Primal feasibility: $GU \leq W + Tx_0$

Dual feasibility: $\lambda \geq 0$

Complimentary slackness: $\lambda^{\mathrm{T}}(GU - W - Tx_0) = 0$

(Lagrange multipliers corresponding to inactive constraints are zero)

Solution to KKT conditions

Take some x_0^o and compute $U^*(x_0^o)$ by solving the QP numerically

Partition the constraints into active and inactive (at $U^*(x_0^o)$)

$$\tilde{G}U^*(x_0^o) = \tilde{W} + \tilde{T}x_0^o,$$

$$\hat{G}U^*(x_0^o) < \hat{W} + \hat{T}x_0^o.$$

Let the Lagrange multipliers be partitioned accordingly into $\tilde{\lambda}$ and $\hat{\lambda}$. Note that $\hat{\lambda} = 0$. Assume $\tilde{G}$ is full column rank.

Consider now x_0 near x_0^o and assume the set of active constraints and the set of inactive constraints remains the same.

By stationarity, $HU + q + G^{\mathrm{T}}\lambda = 0$ and $U^* = H^{-1}[-q - G^{\mathrm{T}}\lambda^*]$

By complimentary slackness, $G^{\mathrm{T}}\lambda^* = \tilde{G}^{\mathrm{T}}\tilde{\lambda}^*$ and

$$U^* = H^{-1}[-q - \tilde{G}^{\mathrm{T}}\tilde{\lambda}^*]$$

MPC feedback law is locally an affine function of the state

Plugging into the active constraints,

$$\tilde{G}H^{-1}[-q - \tilde{G}^{\mathrm{T}}\tilde{\lambda}^*] = \tilde{W} + \tilde{T}x_0.$$

Hence,

$$\tilde{\lambda}^* = -(\tilde{G}H^{-1}\tilde{G}^{\mathrm{T}})^{-1}[\tilde{G}H^{-1}q + \tilde{W} + \tilde{T}x_0],$$

and

$$U^*(x_0) = H^{-1}[-q + \tilde{G}^{\mathrm{T}}(\tilde{G}H^{-1}\tilde{G}^{\mathrm{T}})^{-1}[\tilde{G}H^{-1}q + \tilde{W} + \tilde{T}x_0]].$$

Since the last expression is linear in x_0, for x near x_0^o, $U^*(x_0)$ is an affine function of the state,

$$U^*(x_0) = Kx_0 + L,$$

for $K = H^{-1}\tilde{G}^{\mathrm{T}}(\tilde{G}H^{-1}\tilde{G}^{\mathrm{T}})^{-1}\tilde{T}$, $L = H^{-1}[-q+\tilde{G}^{\mathrm{T}}(\tilde{G}H^{-1}\tilde{G}^{\mathrm{T}})^{-1}[\tilde{G}H^{-1}q+\tilde{W}]]$.

The LQ-MPC feedback law is defined by the first move and is also an affine function, as

$$u_{MPC}(x_0) = \begin{bmatrix} \mathbb{I}_{n_u \times n_u} & 0 & \cdots & 0 \end{bmatrix} U^*(x_0)$$

Region of validity of a single affine function is a polyhedron

The MPC control law is an affine function of the state x_0 for x_0 near x_0^o as long as the activity status of the constraints does not change. The region in which the same affine expression for the control holds is called the critical region and is defined by primal and dual feasibility conditions holding, i.e.,

$$CR(x_0^o) = \{x_0 : \ \tilde{\lambda}^*(x_0) \geq 0, \ \hat{G}U^*(x_0) \leq \hat{W} + \hat{T}x_0\}.$$

where

$$\tilde{\lambda}^*(x_0) = -(\tilde{G}H^{-1}\tilde{G}^{\mathrm{T}})^{-1}[\tilde{G}H^{-1}q + \tilde{W} + \tilde{T}x_0],$$

and

$$U^*(x_0) = H^{-1}[-q + \tilde{G}^{\mathrm{T}}(\tilde{G}H^{-1}\tilde{G}^{\mathrm{T}})^{-1}[\tilde{G}H^{-1}q + \tilde{W} + \tilde{T}x_0]].$$

$$q = S^{\mathrm{T}}\bar{Q}Mx_0$$

The critical region is polyhedral, that is it is defined by a finite number of affine inequalities.

Naïve algorithm

Step 1: Select a set of active constraints.
Step 2: Partition constraints accordingly $(\tilde{G}, \hat{G})$ and form expressions for $\tilde{\lambda}^*(x_0)$, $U^*(x_0)$ by solving

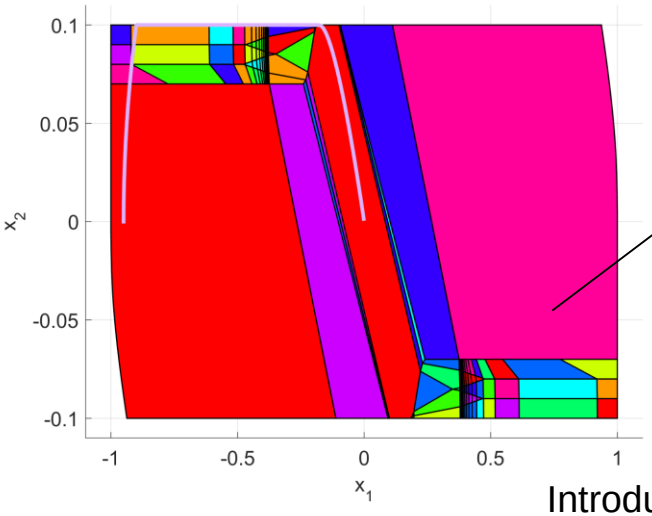
$$HU + q + \tilde{G}^{\mathrm{T}}\tilde{\lambda} = 0,$$

$$\tilde{G}U = \tilde{W} + \tilde{T}x_0.$$

Step 3: Define the region of validity of the solution based on primal and dual feasibility

$$\tilde{G}U - W - Tx_0 \leq 0.$$

$$\tilde{\lambda}(x_0) \leq 0$$



CR_i

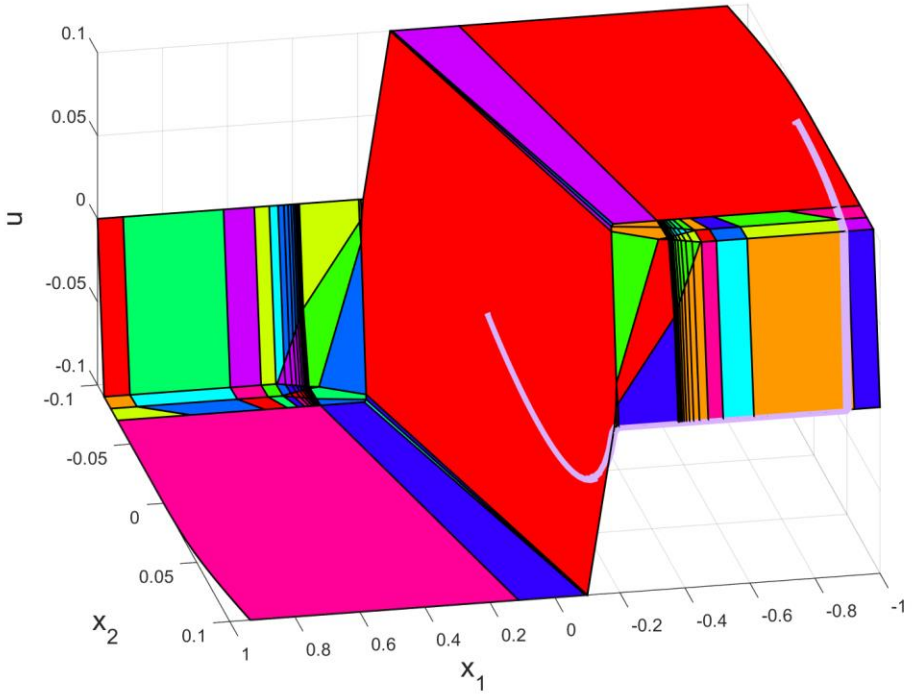
$$u = u_{MPC}(x) = K_i x + L_i \text{ if } x \in CR_i$$

$$CR_i = \{x: F_i x \leq f_i\}, i = 1, \cdots, N_r$$

Solution to QP and MPC feedback law are piecewise affine functions of the state

$$u = u_{MPC}(x) = K_i x + L_i \text{ if } x \in CR_i$$

$$CR_i = \{x : F_i x \leq f_i\}, i = 1, \cdots, N_r$$



Double integrator example - regulation (MPT3)

```
% Continuous time model
    Ac = [0,1;0,0];
    Bc = [0;1];
    Cc = [1,0;0,1];
    Dc = [0;0];

% Discrete-time model
    Ts = 0.1;  % sampling period, sec
    sysd = c2d( ss(Ac, Bc, Cc, Dc), Ts );
    Ad = sysd.A;
    Bd = sysd.B;
    Cd = sysd.C;
    Dd = sysd.D;

    model = LTISystem('A',Ad,'B',Bd,'C',Cd,'D',Dd,'Ts',Ts);
```

Double integrator example - regulation (MPT3)

```
% Control constraints
    model.u.min= -0.1;
    model.u.max= -model.u.min;
% State constraints
    model.x.min=[-1, -0.2];
    model.x.max=[1.1, 0.2];
% Cost function
    Q = diag([1,0.1]); R = diag([0.1]);
    model.x.penalty = QuadFunction( Q );
    model.u.penalty = QuadFunction( R );
    % add LQR terminal penalty
    model.x.with('terminalPenalty');
    model.x.terminalPenalty = model.LQRPenalty();
% Horizon
    N = 6;
% Generate MPC Controller
    ctrl = MPCController(model,N)
```

Double integrator example - regulation (MPT3)

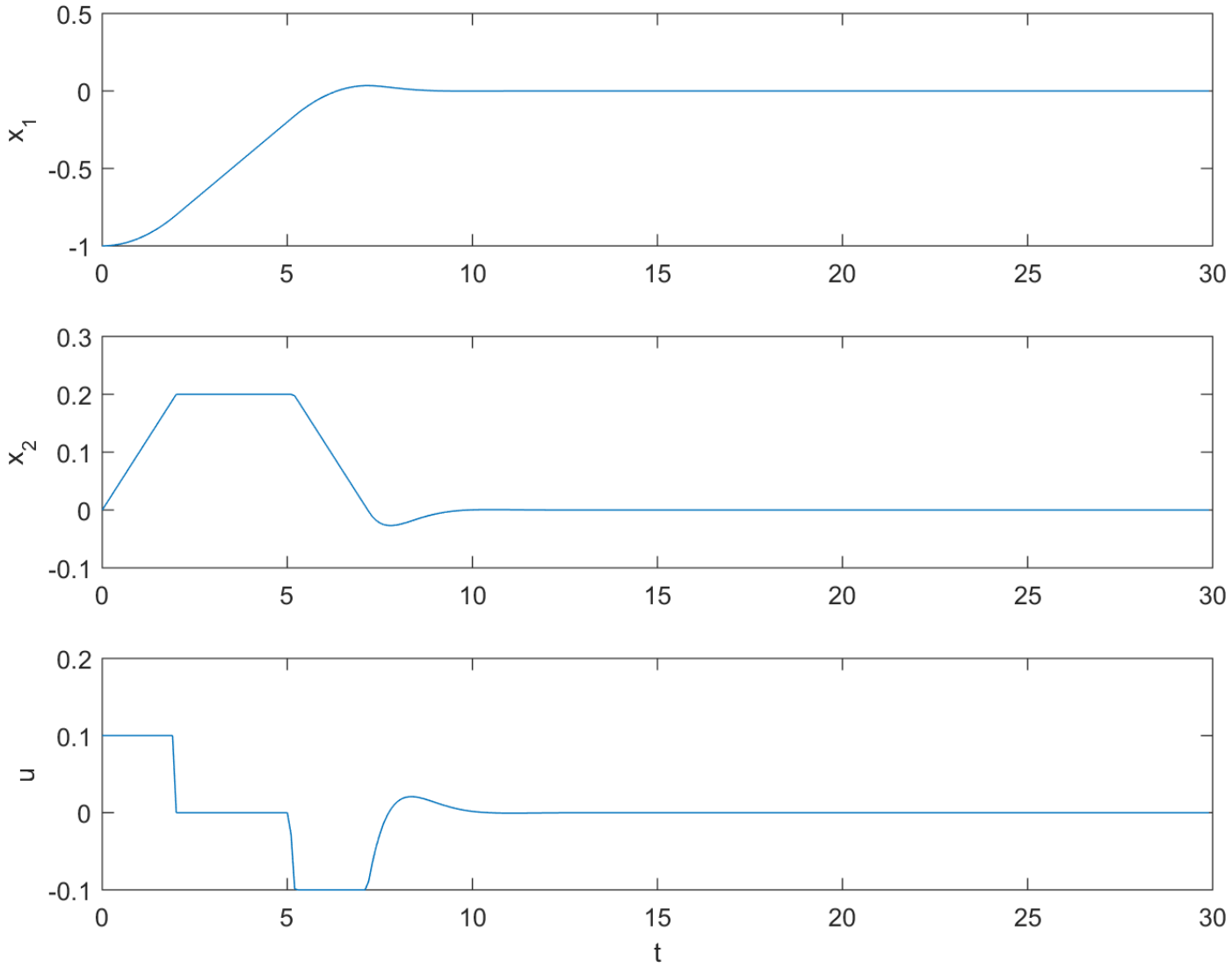
```
% Simulate closed loop response
    x0 = [-1;0];
    u  = 0;
    Nsim = 300;

    x = x0;
    data.X=zeros(length(x0),Nsim);
    data.U=zeros(size(Bd,2),Nsim);
    for i=1:Nsim,
        try,
            [u,feasible,openloop] = ctrl.evaluate([x(:)]);
        catch,
            u = 0;
        end
        data.X(:,i)=x;
        data.U(:,i)=u;
        x = Ad*x  + Bd*(u);
    end;
```

Double integrator example - regulation (MPT3)

```
% Plot the responses
    figure;
    subplot(311)
    plot(([1:Nsim]-1)*Ts, data.X(1,:));
    ylabel('x_1')
    subplot(312)
    plot(([1:Nsim]-1)*Ts, data.X(2,:))
    ylabel('x_2')
    subplot(313)
    plot(([1:Nsim]-1)*Ts, data.U(1,:))
    ylabel('u')
```

Double integrator example - regulation (MPT3)



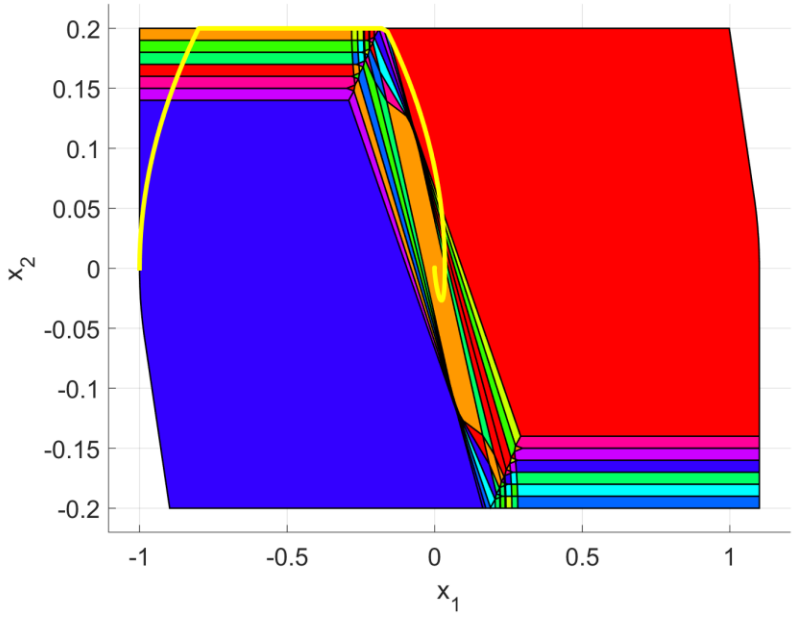
Double integrator example - regulation (MPT3)

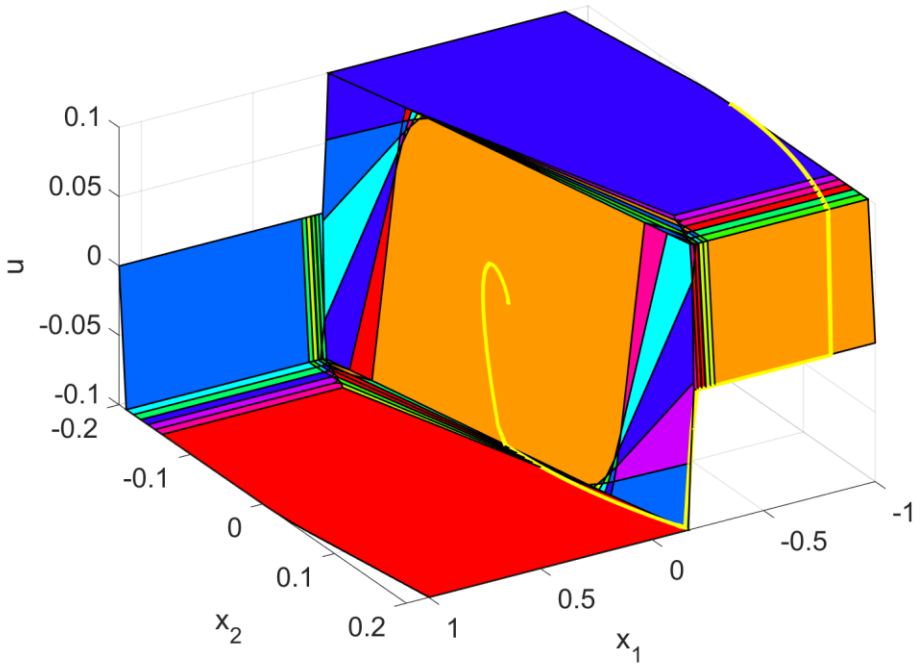
```
exp_ctrl    = ctrl.toExplicit();  % explicit controller

figure(11);
exp_ctrl.partition.plot()
hold on;
pl=plot(data.X(1,:),data.X(2,:),'b-')
set(pl,'linewidth',3); set(gca,'fontsize',16)
xlabel('x_1'); ylabel('x_2');

figure(12);
exp_ctrl.feedback.fplot()
hold on;
pl=plot3(data.X(1,:),data.X(2,:),data.U(1,:),'b-')
set(pl,'linewidth',3); set(gca,'fontsize',16)
xlabel('x_1'); ylabel('x_2'); zlabel('u');
```

Double integrator example – regulation (MPT3)





Double integrator example - regulation (MPT3)

```
F = {}; f = {}; K={}; L={};
for i = 1:exp_ctrl.optimizer.Num
    F{i} = exp_ctrl.optimizer.Set(i).A;
    f{i} = exp_ctrl.optimizer.Set(i).b;
    K{i} = exp_ctrl.optimizer.Set(i).Functions('primal').F;
    L{i} = exp_ctrl.optimizer.Set(i).Functions('primal').g;
    if F{i}*[0,0]' <= f{i}
      i0 = i;
      K0 = K{i};
      L0 = L{i};
    end;
end

[Klqr] = -dlqr(Ad, Bd, Q, R);

[K0(1,:), Klqr]
```

```
ans =

   -2.7623   -2.5075   -2.7623   -2.5075
```

C code generation

exp_ctrl.exportToC('myExplicitController', 'my_C_Directory');
produces myExplicitController.c and .mex file

```
reg = 0;
r = 0; notFound = True;

while(r<=Nr && notFound){

    r++;
    i=1;
    while(i<Nineq[r] && allSat){
        if(H[r][i][1]x[1]+…+H[r][i][n]x[n]> K[r][i]);
            allSat=FALSE;

            i++;}                    /* Region search */

    if(allSat){

        reg = r;
        notFound = False;}

}

for(i=1;i++;i<=n)                    /* Input computation */

    u[i]=F[reg][i][1]x[1]+…+F[reg][i][n]x[n]+ G[reg][i]
```

Code is fixed.
Only regions and gains data change.

Double integrator example - regulation (hybrid toolbox)

```
% Define cost, constraints and horizon
    clear cost constraints horizon
    cost.Q = diag([1, 0.1]);
    cost.R = 0.1;
    [Klqr,Plqr,E]=dlqr(Ad, Bd, cost.Q, cost.R);
    cost.P = Plqr;
    sysd   = ss(Ad,Bd,Cd,Dd,Ts) ;
    cost.rho = 0.1*Inf; % set to Inf for hard constraints

    horizon.N  =  6 ;
    horizon.Nu  = 6 ;
    horizon.Ncy = 5 ;
    horizon.Ncu = 5 ;
    constraints.umax =  0.1 ;
    constraints.umin = -constraints.umax ;
    constraints.ymax =  [1.1 0.2] ;
    constraints.ymin = -constraints.ymax ;
    MPCctrl=lincon(sysd, 'reg', cost, horizon, constraints,…
       'qpact',0) ;
```

Double integrator example - regulation (hybrid toolbox)

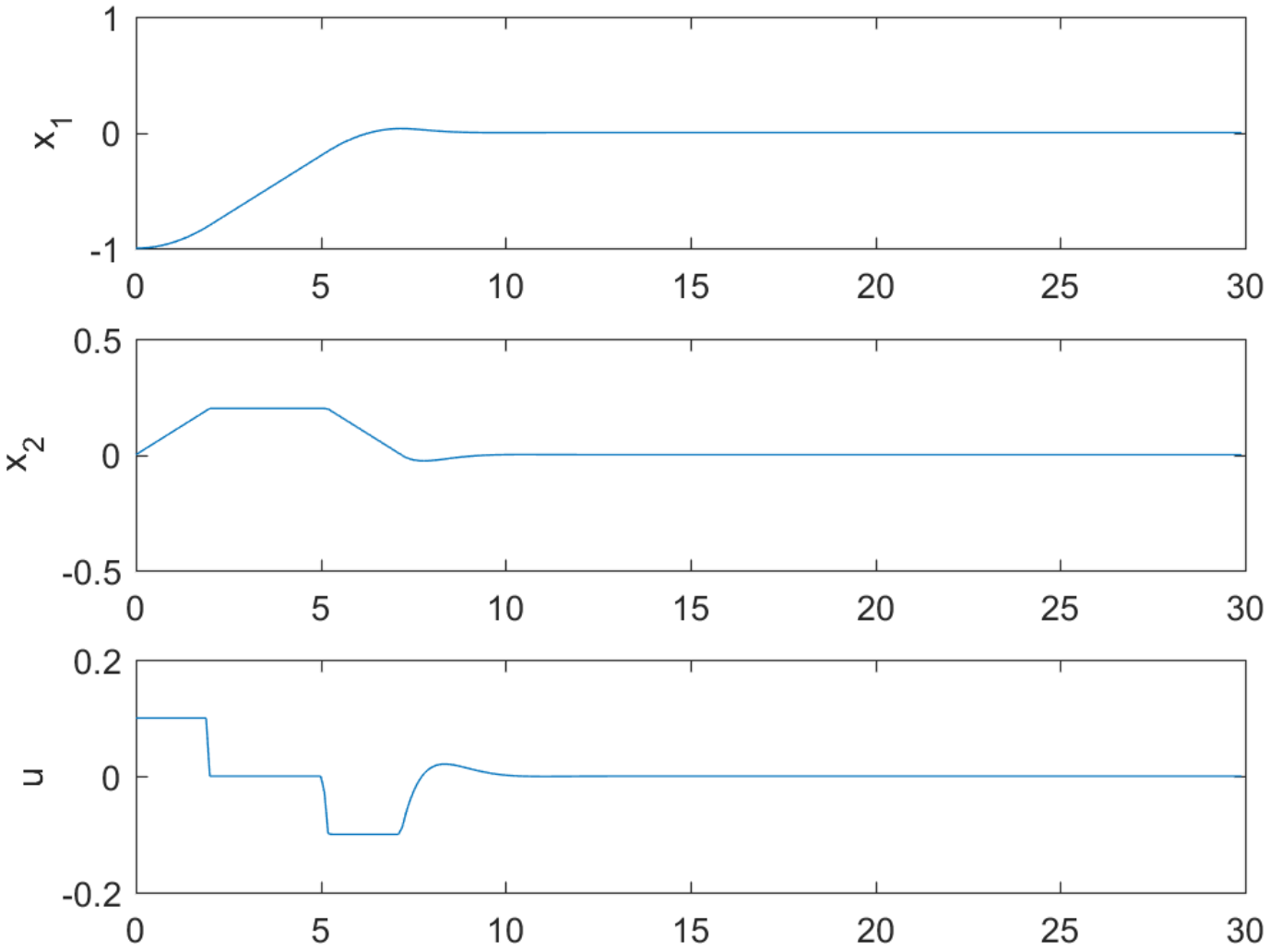
```
% Simulate closed-loop response
    x0 = [-1;0];
    Nsim = 300;

    x = x0;
    data.X=zeros(length(x0),Nsim);
    data.U=zeros(size(Bd,2),Nsim);
    for i=1:Nsim,
        try,
            u = eval(MPCctrl,([x(:)]));
        catch,
            u = 0;
        end
        u = u;
        data.X(:,i)=x;
        data.U(:,i)=u;
        x = Ad*x + Bd*(u);
    end;
```

Double integrator example - regulation (hybrid toolbox)

```
% Plot the responses
    figure;
    subplot(311)
    plot(([1:Nsim]-1)*Ts, data.X(1,:));
    ylabel('x_1')
    subplot(312)
    plot(([1:Nsim]-1)*Ts, data.X(2,:))
    ylabel('x_2')
    subplot(313)
    plot(([1:Nsim]-1)*Ts, data.U(1,:))
    ylabel('u')
    xlabel('t');
```

Double integrator example - regulation (hybrid toolbox)



Double integrator example - regulation (hybrid toolbox)

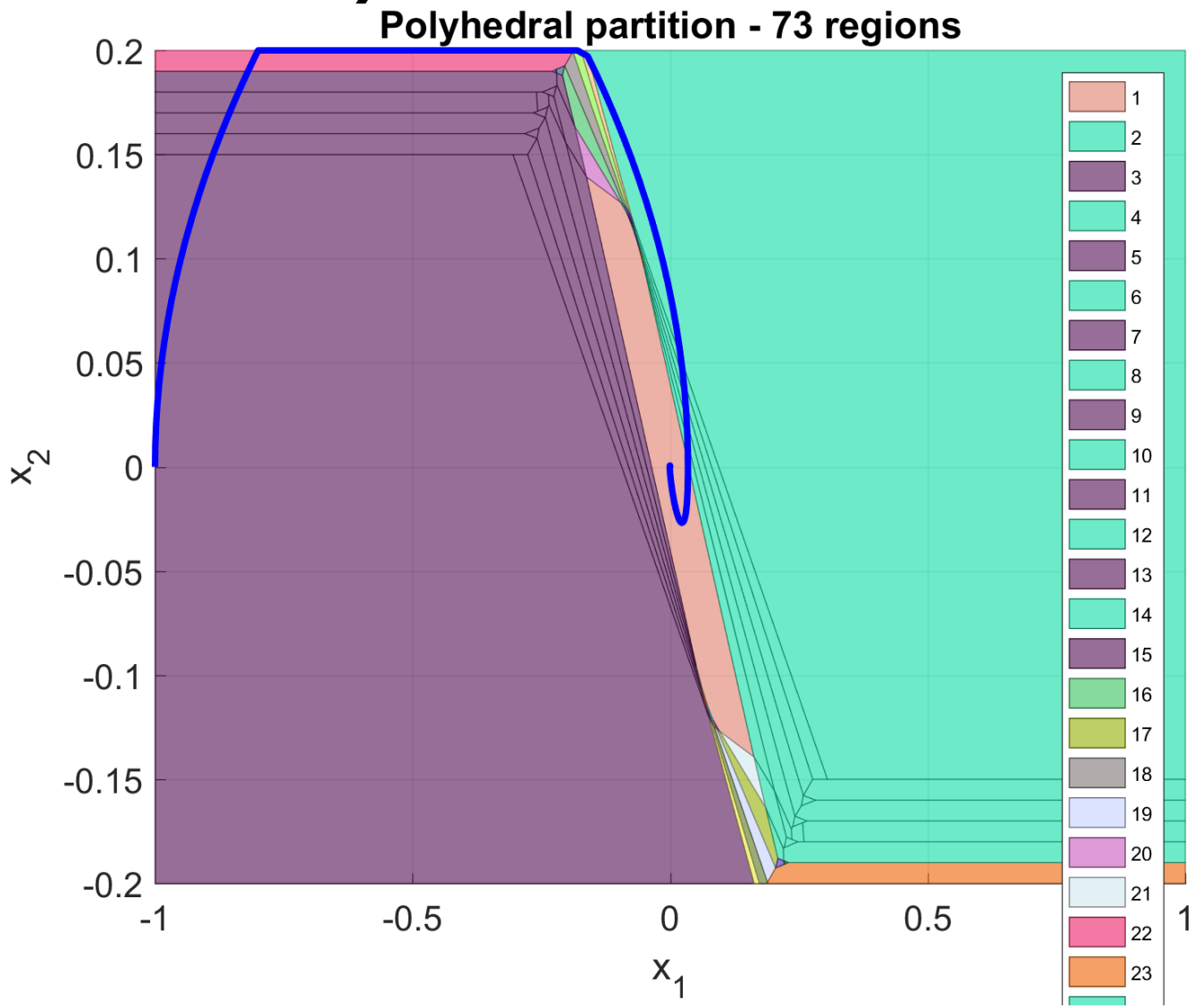
```
range = struct('xmin',[-1 -0.2],'xmax',[1 0.2]);

% Compute explicit version of the controller
MPCctrlexp = expcon(MPCctrl, range);

plot(MPCctrlexp)

hold on;
pl = plot(data.X(1,:), data.X(2,:), 'b-');
set(gca,'fontsize',16);
xlabel('x_1');ylabel('x_2');
set(pl,'linewidth',3);
axis([-1 1 -0.2 0.2]);
```

Double integrator example - regulation (hybrid toolbox)



Double integrator example - regulation (hybrid toolbox)

Generate C code

hwrite(MPCctrlexp)

Two files expcon.h and expcon.c are generated

Explicit MPC Discussion

- Explicit solution to MPC problem is pre-computed on a desktop PC, then stored for on-line use in ECU as a family of “if-then” rules, additions, multiplications and comparisons
- Optimization solver does not need to be embedded with ECU software
- Worst case chronometrics, ROM and RAM can be explicitly estimated
- Very fast if the number of regions is small
  - Number of regions is strongly correlated with control and constraint horizons
- Effective techniques for region storage / search are available
- Fixed point implementation is feasible
- Region “reduction” techniques are available to eliminate “never/rarely visited” regions (up to 100x reduction in regions in particular cases)
- Sufficient conditions for stability can be checked a posteriori

Handling infeasible constraints using slack variables

Handling infeasible constraints using slack variables

• State constraints can become infeasible during the system operation, i.e., no solution to MPC problem may exist that satisfies constraints.

• If state constraints are soft (slight violations are undesirable but do not lead to catastrophic failures), the system operation can be continued while the controller aggressively reduces constraint violation.

• In MPC the mechanism to handle potential infeasibility is through the slack variables.

• Mechanisms to ensure recursive feasibility will be discussed later in the course. However, in practical situations large disturbances or model mismatch can frequently cause infeasibility or constraint violation.

• Thus practical MPC controllers often use slack variables for state constraints while imposing tighter constraints than physical limits of the system.

Handling infeasible constraints using slack variables

Relax state constraints by introducing slack variables:

$$\begin{cases} x \leq x_{max} \\ -x \leq -x_{min} \end{cases} \Rightarrow \begin{cases} x \leq x_{max} + \epsilon \\ -x \leq -x_{min} + \epsilon \end{cases}$$

where ϵ is added as a manipulatable variable to the optimization problem.

Remark 1: Various modifications are possible, e.g., the slack variable can be a scalar, separate slacks can be used on the min and max side, only some constraints could be relaxed, etc.

Remark 2: The control constraints are typically not relaxed and treated as hard. Note that feasibility is trivially enforceable for the control constraints.

Modified LQ-MPC problem

$$
\begin{aligned}
\underset{u_0,u_1,\cdots,u_{N-1},\epsilon}{\text{Minimize}} \quad & J_N = \sum_{k=0}^{N-1} x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k + x_N^{\mathrm{T}} P x_N + \mu \epsilon^{\mathrm{T}} \epsilon \\
\text{subject to} \quad & x_{k+1} = Ax_k + Bu_k \\
& x_0 = \text{(given current state)} \\
& x_{min} - \epsilon \leq x_k \leq x_{max} + \epsilon, \;\; k = 0, \cdots, N \\
& u_{min} \leq u_k \leq u_{max}, \;\; k = 0, \cdots, N-1
\end{aligned}
$$

The MPC feedback law is defined by the first optimal move, u_0^*, i.e, $u_{MPC}^*(x_0) = u_0^*$.

The penalty weight $\mu > 0$ is large.

Remark: There are advantages to using nonsmooth penalties, $\mu||\epsilon||_\infty$ or $\mu||\epsilon||_1$ instead of $\mu\epsilon^{\mathrm{T}}\epsilon$. Specifically, for all μ sufficiently large, a constraint feasible solution will be generated if such a solution exist. This so called exact penalization approach.

Conversion to QP

Recall that to convert LQ-MPC problem to QP, we constructed the dependencies $X = SU + Mx_0$, $GU \leq W + Tx_0$, etc.

Replace U by U^s

$$U^s = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \\ \epsilon \end{bmatrix},$$

where ϵ is augmented to U.

Replace $\bar{R}$ by $\bar{R}^s$ which includes the penalty on the slack variable

$$\bar{R}^s = \begin{bmatrix} R & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & R & 0 \\ 0 & \cdots & \cdots & \mu \mathbb{I}_{n_x \times n_x} \end{bmatrix}.$$

Conversion to QP

Replace S by

$$S^s = \begin{bmatrix} S & 0_{n_X \times n_\epsilon} \end{bmatrix}$$

Replace G by

$$G^s = \begin{bmatrix} S & \begin{bmatrix} -\mathbb{I}_{n_x \times n_\epsilon} \\ \vdots \\ -\mathbb{I}_{n_x \times n_\epsilon} \end{bmatrix} \\ -S & \begin{bmatrix} -\mathbb{I}_{n_x \times n_\epsilon} \\ \vdots \\ -\mathbb{I}_{n_x \times n_\epsilon} \end{bmatrix} \\ \mathbb{I}_{n_U \times n_U} & 0_{n_U \times n_\epsilon} \\ -\mathbb{I}_{n_U \times n_U} & 0_{n_U \times n_\epsilon} \end{bmatrix}$$

Remark: Note we use the same number of slack variables as x's, so $n_x = n_\epsilon$.

Conversion to QP

Once the optimization problem is solved, and the solution $U^{s*}(x_0)$ is obtained, the LQ-MPC feedback law is defined as

$$u_{MPC}(x_0) = \left[\begin{array}{ccccc} \mathbb{I}_{n_u \times n_u} & 0 & \cdots & 0 & 0_{n_u \times n_\epsilon} \end{array} \right] U^{s*}(x_0)$$

Handling disturbance preview

Model with a disturbance preview

Suppose the model over the prediction horizon is given by

$$x_{k+1} = Ax_k + Bu_k + B_w w_k,$$

where the time instant $k = 0$ is the current time and w_k is a disturbance input. Suppose we know the future values of this input over some time window (this is called preview), i.e., $w_0,\ w_1,\ \cdots,\ w_{N_d}$ are known. We then assume that $w_k = w_{N_d}$ for $k > N_d$.

Auxiliary model

To incorporate such a disturbance preview in MPC design one approach involves augmenting an auxilliary model for the disturbance in the form of a shift register,

$$\tilde{w}_{k+1} = A_w \tilde{w}_k,$$

where

$$A_w = \begin{bmatrix} 0 & I & 0 & \cdots & 0 \\ 0 & 0 & I & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & I \\ 0 & 0 & 0 & 0 & I \end{bmatrix}, \quad \tilde{w}_0 = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{N_d} \end{bmatrix}.$$

Note that

$$\tilde{w}_1 = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{N_d} \\ w_{N_d} \end{bmatrix}, \; \tilde{w}_2 = \begin{bmatrix} w_2 \\ w_3 \\ \vdots \\ w_{N_d} \\ w_{N_d} \end{bmatrix}, \text{ etc.}$$

Augmented system and MPC design

We then change the model to

$$x_{k+1} = Ax_k + Bu_k + \tilde{B}_w \tilde{w}_k, \quad \tilde{B}_w = \begin{bmatrix} B_w & 0 & \cdots 0 \end{bmatrix}.$$

For the augmented system,

$$x_{k+1} = Ax_k + Bu_k + \tilde{B}_w \tilde{w}_k,$$

$$\tilde{w}_{k+1} = A_w \tilde{w}_k$$

the MPC feedback law will be defined by the first move u_0 of the suitable defined optimal control problem. Note that it will be a function of the disturbance preview,

$$u = u_{MPC}(x_0, \tilde{w}_0) = u_0^*.$$

Unmeasured but estimated disturbances

Suppose now that the disturbance, w, is not measured and its future values are unknown. The following approach is often used:

- Estimate the current value of the disturbance from output measurements. Frequently, this is done by assuming a state-space model for the disturbance and designing a Kalman filter to estimate it along with the system state.

- Assume that the disturbance is constant over the prediction horizon and set it equal to the current estimated value, i.e.,

$$w_0 = w_1 = \cdots = w_{N-1} = \hat{w}_0,$$

  where $\hat{w}_0$ is the current estimate of the disturbance.

- Thus the MPC feedback law will have the form,

$$u = u_{MPC}(x_0, \hat{w}_0)$$

Handling input delays

Model with an input delay

Suppose the model over the prediction horizon is given by

$$x_{k+1} = Ax_k + Bu_{k-\tau},$$

where $\tau \in \mathbb{Z}_{\geq 0}$ is the input delay, and the time instant $k = 0$ is the current time instant. The delay τ is assumed to be constant.

To incorporate the delay in MPC design one approach involves augmenting an auxilliary model for the delay in the form of,

$$\tilde{u}_{d,k+1} = A_{\text{del}}\tilde{u}_{d,k} + B_{del}u_k,$$

where

$$A_{\text{del}} = \begin{bmatrix} 0 & I & 0 & \cdots & 0 \\ 0 & 0 & I & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & I \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad B_{\text{del}} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ I \end{bmatrix}.$$

MPC design for augmented system

We then change the model to

$$x_{k+1} = Ax_k + \tilde{B}\tilde{u}_{d,k} \quad \tilde{B} = \begin{bmatrix} B & 0 & \cdots & 0 \end{bmatrix}.$$

For the augmented system,

$$x_{k+1} = Ax_k + \tilde{B}\tilde{u}_{d,k},$$

$$\tilde{u}_{d,k+1} = A_{\text{del}}u_{d,k} + B_{\text{del}}u_k$$

the MPC feedback law will be defined by the first move u_0 of the suitable defined optimal control problem. Note that it will be a function of the current state and past values of the input,

$$u = u_{MPC}(x_0, u_{d,0}) = u_0^*, \quad u_{d,0} = \begin{bmatrix} u_{-1} \\ u_{-2} \\ \vdots \\ u_{-\tau} \end{bmatrix}$$

Beyond quadratic - treating “linear stage” costs

Handling linear stage cost: Basic idea

Consider minimizing a function,

$$f(x,u) = \|Qx\|_p + \|Ru\|_p \to \min_{x,u}$$

where $p \in \{1, \infty\}$.

Note that

$$\|x\|_\infty = \max_{i=1,\cdots,n_x} |[x]_i|, \quad \|x\|_1 = \sum_{i=1}^{n_x} |[x]_i|,$$

$$Q \in \mathbb{R}^{n_x \times n_x}, \quad R \in \mathbb{R}^{n_u \times n_u}.$$

Handling linear stage cost: Basic idea

It is possible to re-write the above optimization problem in terms of minimizing a linear function subject to linear (affine) inequality constraints, specifically, as

$$c_x^{\mathrm{T}} \epsilon_x + c_u^{\mathrm{T}} \epsilon_u \to \min$$

subject to

$$\begin{bmatrix} Qx \\ -Qx \end{bmatrix} \leq W_x \epsilon_x, \quad \begin{bmatrix} Ru \\ -Ru \end{bmatrix} \leq W_u \epsilon_u,$$

where for $p = \infty$,

$$W_x = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}, \; c_x = 1, \; \epsilon_x \in \mathbb{R}, \; W_u = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}, \; c_u = 1, \; \epsilon_u \in \mathbb{R},$$

and for $p = 1$,

$$c_x = \begin{bmatrix} 1, \cdots, 1 \end{bmatrix}, W_x = \begin{bmatrix} I \\ -I \end{bmatrix}, \epsilon_x \in \mathbb{R}^{n_x}, \; c_u = \begin{bmatrix} 1, \cdots, 1 \end{bmatrix}, W_u = \begin{bmatrix} I \\ -I \end{bmatrix}, \epsilon_u \in \mathbb{R}^{n_u}.$$

Handling linear stage cost: Basic idea

Let $p \in \{1, \infty\}$. Consider an optimal control problem,

$$\begin{aligned} \underset{u_0, u_1, \cdots, u_{N-1}}{\text{Minimize}} \quad & J_N = \sum_{k=0}^{N-1} \|Qx_k\|_p + \|Ru_k\|_p + \|Px_N\|_p \\ \text{subject to} \quad & x_{k+1} = Ax_k + Bu_k \\ & x_0 = \text{current state} \\ & x_{min} \leq x_k \leq x_{max}, \ k = 1, \cdots, N \\ & u_{min} \leq u_k \leq u_{max}, \ k = 0, \cdots, N-1 \end{aligned}$$

By following the approach of transforming the norms into linear functions defined in terms of auxillliary variables, it can be shown that the above problem reduces to a Linear Programming (LP) problem.

The MPC law can be defined by the first move of the solution as in LQ-MPC case.

State estimation

Q&A: What about state estimation?

A related to LQ-MPC problem on the observer side is that of Moving Horizon Estimation (MHE).

In its simplest form, one is given measurements, $y_0, \cdots, y_N \in \mathbb{R}^{n_y}$, a model, $x_{k+1} = Ax_k,\ y_k = Cx_k,\ x_k \in \mathbb{R}^{n_x}$, which may not perfectly fit the measurements, and has some a priori estimate of the initial state, $\bar{x}_0$.

The objective is to estimate the state sequence $x_0, x_1, \cdots, x_N$ that best fits the measurements by minimizing the following cost function,

$$J = (x_0 - \bar{x}_0)^{\mathrm{T}}(Q_0)^{-1}(x_0 - \bar{x}_0) + \sum_{k=0}^{N-1}(x_{k+1} - Ax_k)^{\mathrm{T}}Q^{-1}(x_{k+1} - Ax_k)$$
$$+ \sum_{k=0}^{N}(y_k - Cx_k)^{\mathrm{T}}R^{-1}(y_k - Cx_k).$$

The weighting matrices are symmetric and $Q_0 \succ 0$, $Q \succ 0$ and $R \succ 0$. The minimization is performed with respect to the unknown state sequence $x_0, \cdots, x_N$.

Q&A: What about state estimation?

Note that no statistical information about the process noise and measurement noise is needed to formulate MHE (unlike Kalman filter). However, further analysis suggests that if statistical information is available, Q is best set to the process noise covariance matrix and R is best set to measurement noise covariance matrix.

To solve the MHE problem, we proceed in the same manner as in LQ-MPC.

Stack the states and outputs into larger vectors,

$$X = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_N \end{bmatrix}, \quad Y = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_N \end{bmatrix}.$$

Q&A: What about state estimation?

1. It is possible to write the cost J in the form,

   $$J = (Y - \bar{C}X)^{\mathrm{T}} \bar{R}^{-1} (Y - \bar{C}X) + (LX - MX)^{\mathrm{T}} \bar{Q}^{-1} (LX - MX) + (\Gamma X - \bar{x}_0)^{\mathrm{T}} Q_0^{-1} (\Gamma X - \bar{x}_0),$$

   for appropriately defined Y, $\bar{C}$, $\bar{R}$, $\bar{Q}$, M, L, Γ (how?).

2. One can derive an expression for the best estimate, $\hat{X}$, which minimizes J with respect to X, as a function of Y and $\bar{x}_0$ (how?). This is called batch estimation.

3. In the theory of Moving Horizon Estimation, the following solution is derived by the application of forward dynamic programming (see the book by Rawlings, Mayne and Diehl),

   $$\begin{aligned}
   &\hat{x}_k = \hat{x}_k^- + L_k(y_k - C\hat{x}_k^-) && \text{(a posteriori state estimate update)}\\
   &L_k = P_k^- C^{\mathrm{T}} (C P_k^- C^{\mathrm{T}} + R)^{-1} && \text{(gain)}\\
   &P_k = P_k^- - P_k^- C^{\mathrm{T}} (C P_k^- C^{\mathrm{T}} + R)^{-1} C P_k^- && \text{(a posteriori covariance matrix update)}\\
   &\hat{x}_{k+1}^- = A\hat{x}_k && \text{(a priori state estimate update)}\\
   &P_{k+1}^- = Q + A P_k A^{\mathrm{T}} && \text{(a priori covariance matrix update)}\\
   &(\hat{x}_0^-, P_0^-) = (\bar{x}_0, Q_0) && \text{(initialization)}
   \end{aligned}$$

4. What are potential advantages of this solution over the batch method, e.g., from implementation perspective?