4F3 – Predictive Control

Lecture 4

Predictive Control with Constraints

Dr Eric Kerrigan

Outline Of Course

- Introduction to predictive control
- Digital state space control theory
- Unconstrained predictive control
- Predictive control with constraints
- Set-point tracking and offset-free control
- Stability and feasibility in predictive control – Dr Jan Maciejowski
- Case study by industrial speaker – Dr Paul Austin

Outline Of Lecture 4

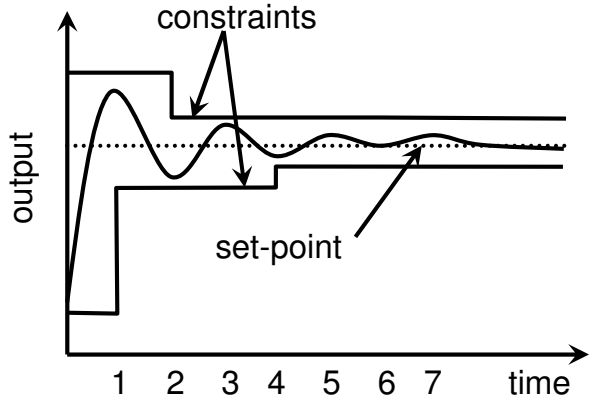
- Constraints
  - Performance or safety constraints
  - Physical limitations
- Finite horizon version of the constrained LQR problem
- Derivation of constraint matrices
- Solution via Quadratic Programming

Reminder Of Notation

- Given a vector/matrix v (for example, x, u, y, z):
- $v(k)$ or v is the ***actual/measured*** value of $v(\cdot)$ at time instant k
- $\hat{v}(i|k)$ is an ***estimate*** of the value of $v(\cdot)$ at time instant i, given measurements up to time k
- v_s is shorthand for the ***prediction/estimate*** of the value of $v(\cdot)$ at time instant $k+s$, given measurements up to time k, i.e. $v_s = \hat{v}(k+s|k)$
- $v_0 = v(k)$ or $\hat{v}(k|k)$ depending on whether or not a measurement of $v(k)$ is available at time instant k
- $v_{\{i\}}$ (and *not* v_i) is the i^{th} ***row/component*** of v

Constraints On System Variables

- In practice, it is nearly always desirable to constrain the system variables due to:
  - Physical limitations, e.g. actuator limits or size of a reservoir
  - Safety considerations, e.g. critical temperatures
  - Performance specifications, e.g. limit overshoot



$$
\begin{aligned}
1 \le y_1 \le 4 \\
1 \le y_2 \le 3 \\
1 \le y_3 \le 3 \\
1.5 \le y_4 \le 3 \\
1.5 \le y_5 \le 3 \\
1.5 \le y_6 \le 3 \\
\vdots
\end{aligned}
$$

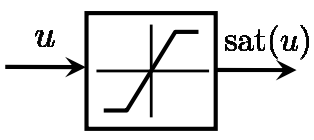
Systems With Input Saturation

- One of the strengths of predictive control is that it can be used to control systems with **input saturation** (a common nonlinearity):

$$
\begin{aligned}
x(k+1) &= Ax(k) + B\,\mathrm{sat}(u(k)) \\
y(k) &= Cx(k) \\
z(k) &= Hx(k)
\end{aligned}
$$

where the i^{th} component ($i = 1,\ldots,m$) of the vector $\mathrm{sat}(u)$ is

$$
\mathrm{sat}(u)_{\{i\}} := \begin{cases} \underline{u}_{\{i\}} & \text{if } u_{\{i\}} < \underline{u}_{\{i\}} \\ u_{\{i\}} & \text{if } \underline{u}_{\{i\}} \le u_{\{i\}} \le \overline{u}_{\{i\}} \\ \overline{u}_{\{i\}} & \text{if } u_{\{i\}} > \overline{u}_{\{i\}} \end{cases}
$$



- Recall that $v_{\{i\}}$ is the i^{th} component (row) of the column vector v

Constrained Linear Quadratic Regulator Problem

- For the given current state $x=x_0$, compute the **infinite** input sequence $u_0,u_1,...$, that minimises the cost:

$$\sum_{s=0}^{\infty} x_s^T Q x_s + u_s^T R u_s$$

  while guaranteeing that the **constraints** (on the inputs, states, outputs, etc.) are satisfied at all time instants
- ***Unconstrained*** LQR solution published in late 1950s/early 1960s
- ***Constrained*** LQR problem has an infinite number of decision variables *and* an infinite number of constraints!
- ***Constrained*** LQR problem was impossible to solve, but:
  - Predictive control solutions published in 1987, 1996 and 1998
  - The analytical/explicit solution was published in 2002 – solution is a ***piecewise linear*** (i.e. nonlinear) control law. Computational requirements are still too high for "large" systems

Finite Horizon Optimal Control Problem With Constraints

- For the given current state x, compute a ***finite*** input sequence $u_0,u_1,...,u_{N-1}$, which minimises the cost:

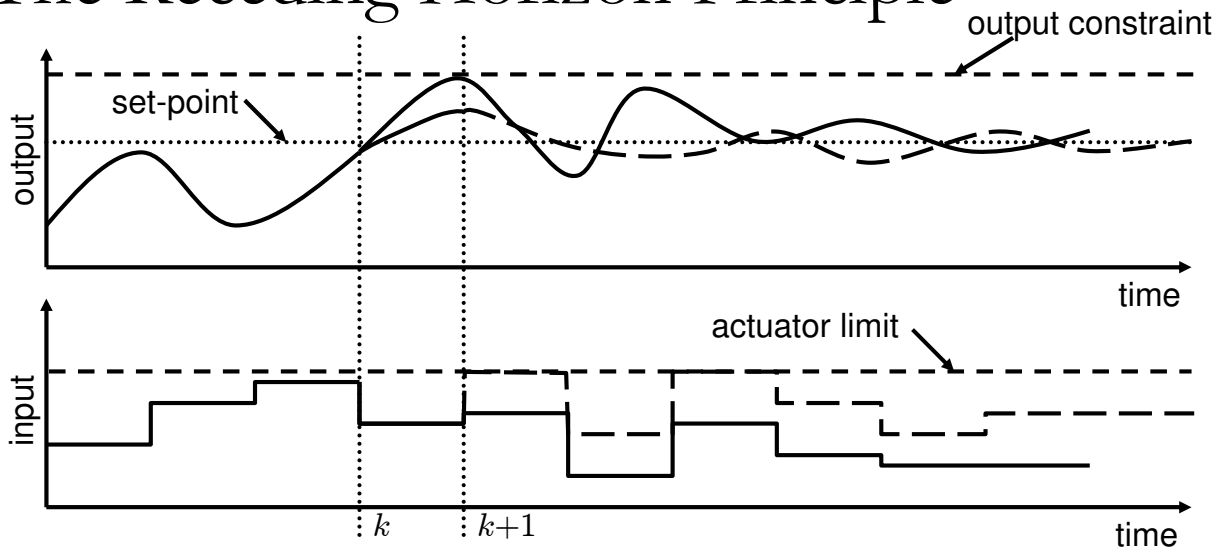
$$V(x, u_0, \ldots, u_{N-1}) := x_N^T P x_N + \sum_{s=0}^{N-1} x_s^T Q x_s + u_s^T R u_s$$

  where

$$x_0 = x$$
$$x_{s+1} = Ax_s + Bu_s \qquad s = 0,1,\ldots,N-1$$

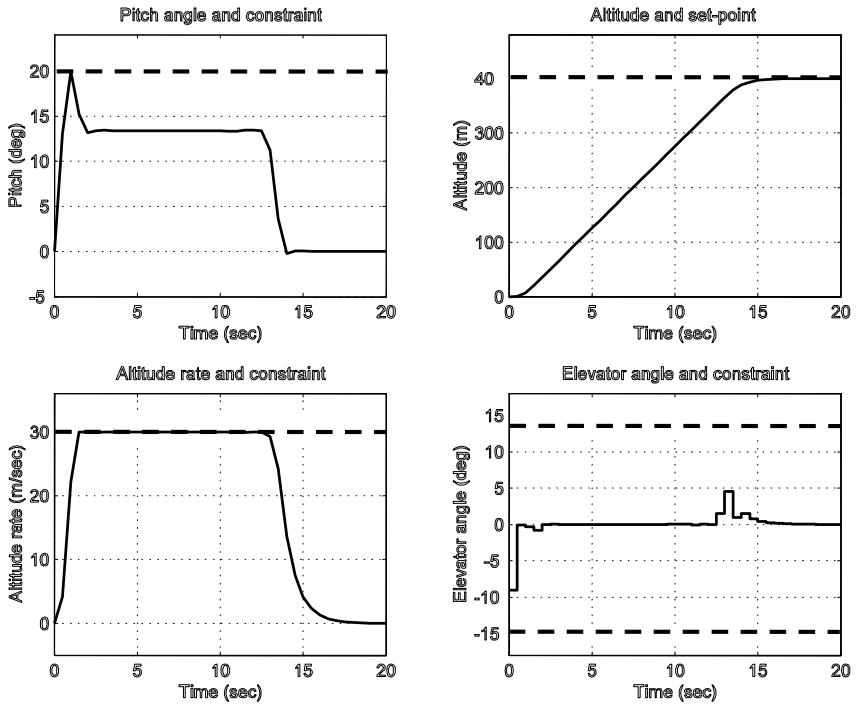
  while guaranteeing that the ***constraints*** (input limits, safety and performance constraints, etc.) are satisfied over the horizon $s = 0,1,\ldots,N$
- Recall that each state $x_s \in \mathbb{R}^n$ and input $u_s \in \mathbb{R}^m$

The Receding Horizon Principle



1. Obtain a measurement/estimate of the ***current*** output/state
2. Compute a ***finite*** optimal input sequence that satisfies the ***constraints***
3. Implement only the ***first*** part of the input sequence
4. Repeat the measurement and optimisation in order to counter the effect of any disturbances or model mismatch, i.e. go to step 1

Example of Constrained RHC: Cessna



Prediction Matrices

- Recall that we defined the **stacked vectors** of inputs $U \in \mathbb{R}^{Nm}$ and states $X \in \mathbb{R}^{Nn}$:

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

- We also derived the **prediction matrices** Φ and Γ such that we can write

$$X = \Phi x + \Gamma U$$

Prediction Matrices

- In Lecture 3, we recursively substituted the expression for x_s into $x_{s+1} = Ax_s + Bu_s$ and collected terms to get:

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} A \\ A^2 \\ \vdots \\ A^N \end{pmatrix} x_0 + \begin{pmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{pmatrix}$$

- Since $x_0 = x$, the **prediction matrices** Φ and Γ in the equation $X = \Phi x + \Gamma U$ are given by:

$$\Phi := \begin{pmatrix} A \\ A^2 \\ \vdots \\ A^N \end{pmatrix}, \qquad \Gamma := \begin{pmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{pmatrix}$$

Constraints: A Simple Numerical Example

- We have $m = 1$, $n = 1$, $N = 2$ and the constraints:

$$u_s \leq 1, \;\; s = 0, 1$$

$$x_s \geq -2, \;\; s = 1, 2$$

- Rewrite as:

$$\begin{pmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \end{pmatrix} \leq \begin{pmatrix} 2 \\ 2 \\ 1 \\ 1 \end{pmatrix}$$

Constraints: A Simple Numerical Example

- Let the model be given by

$$x_{s+1} = 2x_s + 3u_s$$

- In other words,

$$x_1 = 2x_0 + 3u_0$$

$$x_2 = 2x_1 + 3u_1 = 4x_0 + 6u_0 + 3u_1$$

- The constraints can then be we rewritten as:

$$\begin{pmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 2x_0 + 3u_0 \\ 4x_0 + 6u_0 + 3u_1 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \end{pmatrix} \leq \begin{pmatrix} 2 \\ 2 \\ 1 \\ 1 \end{pmatrix}$$

Constraints: A Simple Numerical Example

- Taking

$$\begin{pmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 2x_0 + 3u_0 \\ 4x_0 + 6u_0 + 3u_1 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \end{pmatrix} \leq \begin{pmatrix} 2 \\ 2 \\ 1 \\ 1 \end{pmatrix}$$

collecting terms and rearranging we get

$$\begin{pmatrix} -3 & 0 \\ -6 & -3 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \end{pmatrix} \leq \begin{pmatrix} 2 \\ 2 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 2 \\ 4 \\ 0 \\ 0 \end{pmatrix} x_0$$

Writing Constraints In Terms Of x And U

- We would like to generalise the procedure used in the example for writing constraints in terms of x and U only
- The basic recipe is:
  - Given linear inequality constraints on u_s, x_s, y_s, z_s
  - **Write** the constraints in the form:

$$M_s x_s + E_s u_s \leq b_s, \;\; s = 0, \ldots, N-1$$

$$M_N x_N \leq b_N$$

  - **Stack** the constraints to get it into the form:

$$\mathcal{D}x + \mathcal{M}X + \mathcal{E}U \leq c$$

  - **Substitute** $X = \Phi x + \Gamma U$ and **rearrange** to get the inequalities into the form:

$$\mathcal{J}U \leq c + Wx$$

Incorporating Constraints On Predicted Variables

- Write constraints on x_s and u_s as a set of **linear inequalities**:

$$M_s x_s + E_s u_s \leq b_s$$

  - M_s and E_s are *matrices* and b_s is a *vector*
  - $M_s = 0 \Rightarrow$ constraints only on inputs/manipulated variables (MVs)
  - $E_s = 0 \Rightarrow$ constraints only on states
  - $M_s = C$ and $E_s = 0 \Rightarrow$ constraints only on outputs
  - $M_s = H$ and $E_s = 0 \Rightarrow$ constraints only on controlled variables
  - Any other combination is possible
  - Note that $E_N := 0$ (there is no input u_N in prediction)
- To simplify issues, assume that the constraints are constant over the horizon $s = 0,1,\ldots,N-1$, i.e.

$$E_s = E,\ M_s = M \text{ and } b_s = b \text{ for } s = 0,1,\ldots,N-1$$

How To Get Constraints Into The Form $M_s x_s + E_s u_s \leq b_s$

- Suppose we are given input and output constraints:

$$u_{\text{low}} \leq u_s \leq u_{\text{high}}, \quad s = 0, \ldots, N-1$$
$$y_{\text{low}} \leq y_s \leq y_{\text{high}}, \quad s = 0, \ldots, N$$

  where u_{low} and y_{low} are vectors of lower bounds; u_{high} and y_{high} are vectors of upper bounds
- The above constraints are equivalent to:

$$-u_s \leq -u_{\text{low}}, \quad u_s \leq u_{\text{high}}, \quad s = 0, \ldots, N-1$$
$$-y_s \leq -y_{\text{low}}, \quad y_s \leq y_{\text{high}}, \quad s = 0, \ldots, N$$

How To Get Constraints Into The Form $M_s x_s + E_s u_s \leq b_s$

- Recall that the constraints are equivalent to:

$$-u_s \leq -u_{\text{low}}, \;\; u_s \leq u_{\text{high}}, \;\; s = 0, \ldots, N-1$$

$$-y_s \leq -y_{\text{low}}, \;\; y_s \leq y_{\text{high}}, \;\; s = 0, \ldots, N$$

- Rewrite constraints over horizon $s = 0,\ldots,N–1$ in stacked vector form:

$$\begin{pmatrix} -u_s \\ u_s \\ -y_s \\ y_s \end{pmatrix} \leq \begin{pmatrix} -u_{\text{low}} \\ u_{\text{high}} \\ -y_{\text{low}} \\ y_{\text{high}} \end{pmatrix}, \; s = 0, \ldots, N-1$$

How To Get Constraints Into The Form $M_s x_s + E_s u_s \leq b_s$

- We now take the vector constraints

$$\begin{pmatrix} -u_s \\ u_s \\ -y_s \\ y_s \end{pmatrix} \leq \begin{pmatrix} -u_{\text{low}} \\ u_{\text{high}} \\ -y_{\text{low}} \\ y_{\text{high}} \end{pmatrix}, \; s = 0, \ldots, N-1$$

and, recalling that $y_s = Cx_s$, we rewrite them as:

$$\begin{pmatrix} 0_{m\times 1} \\ 0_{m\times 1} \\ -Cx_s \\ Cx_s \end{pmatrix} + \begin{pmatrix} -u_s \\ u_s \\ 0_{p\times 1} \\ 0_{p\times 1} \end{pmatrix} \leq \begin{pmatrix} -u_{\text{low}} \\ u_{\text{high}} \\ -y_{\text{low}} \\ y_{\text{high}} \end{pmatrix}, \; s = 0, \ldots, N-1$$

How To Get Constraints Into The Form $M_s x_s + E_s u_s \leq b_s$

- The constraints

$$\begin{pmatrix} 0_{m\times 1} \\ 0_{m\times 1} \\ -Cx_s \\ Cx_s \end{pmatrix} + \begin{pmatrix} -u_s \\ u_s \\ 0_{p\times 1} \\ 0_{p\times 1} \end{pmatrix} \leq \begin{pmatrix} -u_{\text{low}} \\ u_{\text{high}} \\ -y_{\text{low}} \\ y_{\text{high}} \end{pmatrix}, \quad s = 0, \ldots, N-1$$

are equivalent to

$$\begin{pmatrix} 0_{m\times n} \\ 0_{m\times n} \\ -C \\ C \end{pmatrix} x_s + \begin{pmatrix} -I_m \\ I_m \\ 0_{p\times m} \\ 0_{p\times m} \end{pmatrix} u_s \leq \begin{pmatrix} -u_{\text{low}} \\ u_{\text{high}} \\ -y_{\text{low}} \\ y_{\text{high}} \end{pmatrix}, \quad s = 0, \ldots, N-1$$

How To Get Constraints Into The Form $M_s x_s + E_s u_s \leq b_s$

- Clearly,

$$\begin{pmatrix} 0_{m\times n} \\ 0_{m\times n} \\ -C \\ C \end{pmatrix} x_s + \begin{pmatrix} -I_m \\ I_m \\ 0_{p\times m} \\ 0_{p\times m} \end{pmatrix} u_s \leq \begin{pmatrix} -u_{\text{low}} \\ u_{\text{high}} \\ -y_{\text{low}} \\ y_{\text{high}} \end{pmatrix}, \quad s = 0, \ldots, N-1$$

is equivalent to $M_s x_s + E_s u_s \leq b_s$ for all values $s = 0,\ldots,N-1$ if we let

$$M_s := \begin{pmatrix} 0_{m\times n} \\ 0_{m\times n} \\ -C \\ C \end{pmatrix}, \quad E_s := \begin{pmatrix} -I_m \\ I_m \\ 0_{p\times m} \\ 0_{p\times m} \end{pmatrix}, \quad b_s := \begin{pmatrix} -u_{\text{low}} \\ u_{\text{high}} \\ -y_{\text{low}} \\ y_{\text{high}} \end{pmatrix}$$

The Terminal Constraint $M_N x_N \leq b_N$

- Remember that we had a terminal constraint on the output at $s = N$:

$$y_{\text{low}} \leq y_N \leq y_{\text{high}}$$

- Since there is no input u_N in the prediction, we have $E_N := 0$, hence

$$M_N := \begin{pmatrix} -C \\ C \end{pmatrix}, \; b_N := \begin{pmatrix} -y_{\text{low}} \\ y_{\text{high}} \end{pmatrix}$$

gives the ***terminal state constraint***

$$M_N x_N \leq b_N$$

We're Not Done Yet

- We now have the constraints:

$$M_s x_s + E_s u_s \leq b_s, \;\; s = 0, \ldots, N-1$$

$$M_N x_N \leq b_N$$

- Using same procedure one can derive M_s, E_s and b_s from an arbitrary set of linear inequality constraints on u_s, x_s, y_s, z_s
- Next, rewrite the above in terms of the current state x and the input sequence

$$U := \begin{pmatrix} u_0^T & \cdots & u_{N-1}^T \end{pmatrix}^T$$

Writing Constraints In Terms Of x and U

- The constraints can be written as:

$$\begin{pmatrix} M_0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} x_0 + \begin{pmatrix} 0 & \cdots & 0 \\ M_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & M_N \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix} + \begin{pmatrix} E_0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & E_{N-1} \\ 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} u_0 \\ \vdots \\ u_{N-1} \end{pmatrix} \leq \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_N \end{pmatrix}$$

- By appropriately defining $\mathcal{D}$, $\mathcal{M}$, $\mathcal{E}$ and c from the above (remember $x = x_0$), write the constraints in the form:

$$\mathcal{D}x + \mathcal{M}X + \mathcal{E}U \leq c$$

Writing Constraints In Terms Of x and U

- By substituting $X = \Phi x + \Gamma U$ into

$$\mathcal{D}x + \mathcal{M}X + \mathcal{E}U \leq c$$

and collecting terms, the constraints can be written in the form

$$JU \leq c + Wx$$

where

$$J := \mathcal{M}\Gamma + \mathcal{E}$$

and

$$W := -\mathcal{D} - \mathcal{M}\Phi$$

Writing Constraints In Terms Of x and U

- Summarising, the basic recipe is:
  - Given linear inequality constraints on u_s, x_s, y_s, z_s
  - **Write** the constraints in the form:

$$M_s x_s + E_s u_s \leq b_s, \;\; s = 0, \ldots, N-1$$

$$M_N x_N \leq b_N$$

  - **Stack** the constraints to get it into the form:

$$\mathcal{D}x + \mathcal{M}X + \mathcal{E}U \leq c$$

  - **Substitute** $X = \Phi x + \Gamma U$ and **rearrange** to get the inequalities into the form:

$$JU \leq c + Wx$$

Alternative Formulations

- As mentioned before, many other variations exist on the formulation presented in this course
- One common formulation is to add constraints on predicted ***changes*** in the input $\Delta u_s := u_s - u_{s-1}$, in addition to having constraints on u_s
  - The constraints on U are dependent on $x(k)$ *and* $u(k-1)$
- Another common formulation is to have a larger horizon for prediction than for control
  - Additional input constraints are then added after the end of the control horizon, e.g. $u_s = Kx_s$ or $\Delta u_s = 0$ for all $s \geq N$

Computing Optimal Input Sequence

- In Lecture 3 we showed that **cost function** can be written as

$$V(x,U) = (1/2)U^TGU + U^TFx + x^T(Q + \Phi^T\Omega\Phi)x$$

- The last term is **not** dependent on U
- We therefore only consider minimising the function

$$\tilde{V}(x,U) := (1/2)U^TGU + U^TFx$$

subject to the **constraints**

$$JU \leq c + Wx$$

- We still denote the **minimiser** as $U^*(x)$

Solution Via Quadratic Programming

- Find the solution to the **optimisation problem**:

$$U^*(x) := \arg\min_U (1/2)U^TGU + U^TFx$$

where U also has to satisfy the **constraints**

$$JU \leq c + Wx$$

- Compare this to finding the solution to a well-known class of optimisation problems, called ***Quadratic Programs (QPs)***:

$$\min_U (1/2)U^TGU + f^TU$$

where U also has to satisfy constraints

$$JU \leq d$$

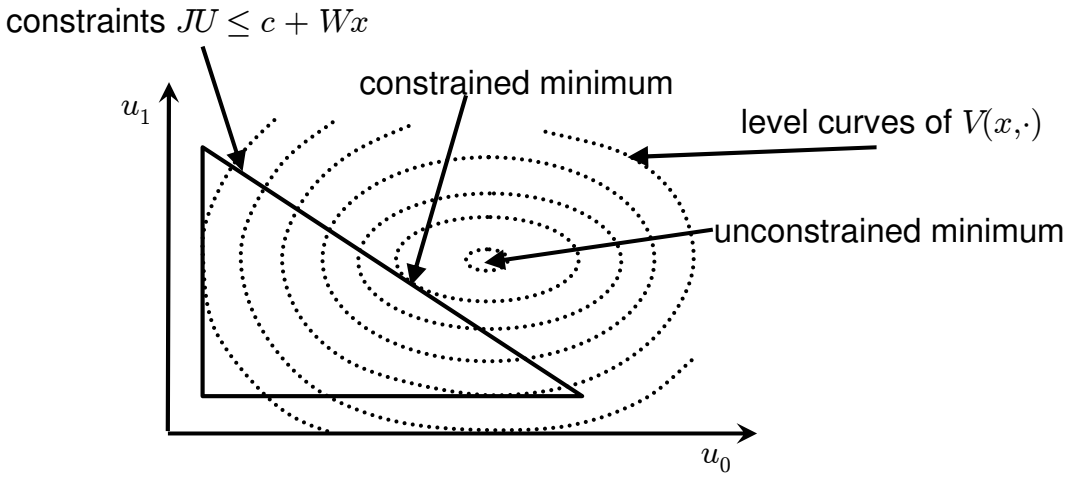
- Clearly, since $f^TU = U^Tf$, the two problems are equivalent if

$$f := Fx, \quad d := c + Wx$$

Solution Via Quadratic Programming

- Expressions for f and d are **functions of the current state** x
- **Analytical expression** of the solution to a **QP** (as a function of f and d) is
  - ***nonlinear*** and
  - difficult to compute
- However, sophisticated numerical algorithms:
  - Compute the solution efficiently for ***a given*** f and d if $G > 0$
  - Recall from Lecture 3 that $G > 0$ if $P \geq 0$, $Q \geq 0$ and $R > 0$
- See Sections 3.2 and 3.3 in JMM for an excellent introduction on how solutions to QPs are computed
  - Active set methods
  - Interior point methods
    - In MATLAB, make function call **quadprog(G,F*x,J,c+W*x)**

Nature Of Solution To A QP



The optimal solution to the constrained problem is often on a subset of the constraints (called the set of ***active constraints***)

Implementing The RHC Law

- Recall that the RHC input is the **first input** in the optimal input sequence:

$$u_0^*(x) = \begin{pmatrix} I_m & 0_{m\times(N-1)m} \end{pmatrix} U^*(x)$$

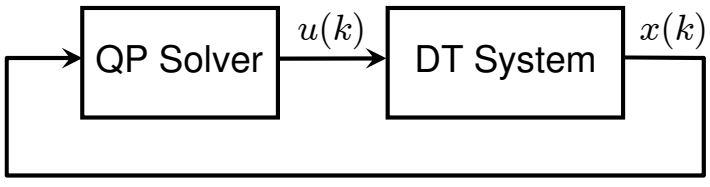
- Since $U^*(\cdot)$ is **no longer linear**, the RHC control law is also no longer linear
- The RHC control law is a ***nonlinear*** map $\kappa_{\mathrm{RHC}}: \mathbb{R}^n \to \mathbb{R}^m$ and $u = \kappa_{\mathrm{RHC}}(x)$, where

$$\kappa_{\mathrm{RHC}}(x) := \begin{pmatrix} I_m & 0_{m\times(N-1)m} \end{pmatrix} U^*(x)$$

Implementing The RHC Law

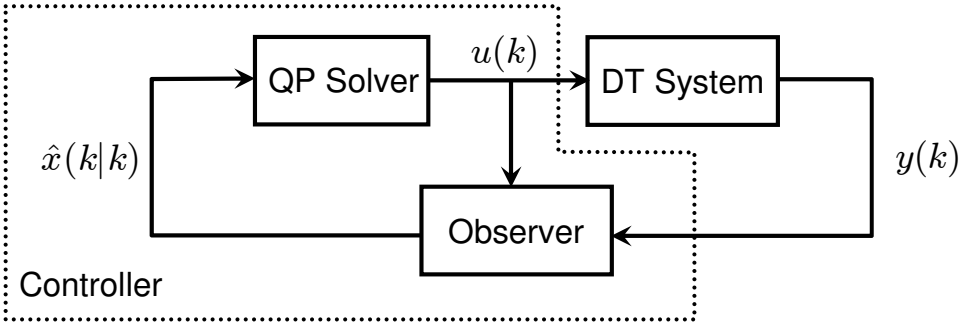
At each time instant k,

- Take a measurement of the **current state** x
- **Solve the QP** defined on page 30 to get $U^*(x)$ (e.g. using MATLAB function **quadprog**)
- Extract **first input** $\kappa_{\mathrm{RHC}}(x) := \begin{pmatrix} I_m & 0_{m\times(N-1)m} \end{pmatrix} U^*(x)$
- Implement RHC input $u = \kappa_{\mathrm{RHC}}(x)$



Output Feedback Predictive Control

- $C \neq I \Rightarrow$ **output feedback**
- Use an **observer** or **Kalman filter** to provide an estimate of the state $\hat{x}(k|k)$
- Computation and implementation of RHC law is the same, except with $x(k)$ replaced with $\hat{x}(k|k)$



Summary

- Many systems are **constrained**
  - Physical limitations, e.g. input constraints
  - Safety and performance constraints
- We posed a **finite horizon** version of the **infinite horizon** *constrained* LQR problem
- Given **linear inequality constraints** on x_s, u_s, y_s and z_s, one can rewrite them in the form $JU \leq c + Wx$
- Use the same cost $V(\cdot)$ as in the unconstrained problem, but this time **include the constraints** in the problem formulation
- Given a measurement of the **current state** x, find the optimal input sequence $U^*(x)$ by solving a **Quadratic Program (QP)**
- As in the unconstrained problem, only implement the **first part** of the optimal input sequence