TTK4135 Optimization and Control
Solution to Final Exam — Spring 2014

Department of Engineering Cybernetics

1 Various topics (20 %)

Convexity

a (2 %) The feasible set exists (or is non-empty) as long as no constraints are inconsistent. (An example of two inconsistent constraints is $x_1 - 3x_2 \geq 2$, and $x_1 - 3x_2 \leq -1$.) When all constraints in (A.1) are linear the feasible set is a polytope, a convex, connected set with flat, polygonal faces.

b (4 %) Whenever (A.1) contains a nonlinear equality constraint function, the feasible set will be noncovex. For the feasible set to be convex, a line between *any two* points in the feasible set must remain inside the feasible set. Since the curve defined by a nonlinear function in not a line, it is not possible for the feasible set to be convex when there is a nonlinear equality constraint. This is illustrated in Figure 1.

c (4 %) Nonconvex optimization problems may have multiple local minima; this makes it difficult to find the global solution. Even if the global minimum is found, it is difficult to verify that that solution is the best one possible. Furthermore, nonconvex problems may be highly nonlinear, which increases the computational burden associated with finding gradients and evaluating functions.

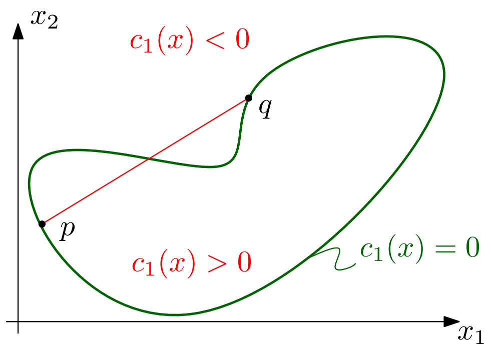

Figure 1: A nonconvex feasible area in the x_1-x_2 plane. The feasible area is the curve traced by $c_1(x)$ (in green). (The interior and the exterior of the curve are infeasible). The line drawn between p and x does remain inside the feasible area since it is not strictly on top of the curve.

Linear programming — LP

d (4 %) The Simplex algorithm tests a basic feasible point for optimality by checking whether the KKT conditions are satisfied. The KKT conditions are sufficient conditions for optimality in linear programs, which means that any basic feasible point that satisfies the KKT conditions is the solution to the linear program. Specifically, the Simplex algorithm tests for optimality by evaluating the condition

$$s_{\mathrm{N}} \geq 0 \tag{1}$$

The other KKT conditions are satisfied automatically at all points visited by the Simplex algorithm by design.

e (6 %) The equation set

$$Ax = b \tag{2}$$

where $A \in \mathbb{R}^{n \times n}$ can be solved for x by LU decomposition. This is computationally easy when the factors L and U exist; L is unit lower triangular and U is upper triangular. We start by writing

$$A = LU \tag{3}$$

which means that the original equation can be written

$$LUx = b \tag{4}$$

If we define a vector

$$y = Ux \tag{5}$$

we again rewrite the original equation $Ax = b$ as

$$Ly = b \tag{6}$$

We then first solve (6) for y and then (5) for x. It is trivial to solve (6) if we start at the top row and move down. The first row is simple $1y_1 = b_1$ since L is unit lower triangular. Since we now know y_1 it is easy to solve the second row $L_{2,1}y_1 + y_2 = b_2$ for y_2. Knowing y_2 makes it easy to solve the third row $L_{3,1}y_1 + L_{3,2}y_2 + y_3 = b_3$ for y_3. We continue like this until all elements of y are known. The procedure for solving $Ux = y$ for x is equally simple now that y is known. The main difference is that we now start with solving the bottom row $U_{n,n}x_n = y_n$ for x_n. With x_n now known we can easily solve row $n-1$, $U_{n-1,n-1}x_{n-1} + U_{n-1,n}x_n = y_{n-1}$, for x_{n-1}. We continue up through the equation set, solving for one element of x per row. Once the top row of $Ux = b$ is solved we know all elements of x and the equation $Ax = b$ is solved.

2 Nonlinear programming and SQP (45 %)

a (4 %) $\mathcal{E}$ is a set of indices; the indices in this set correspond to equation numbers for the equality constraints. Similarly, the index set $\mathcal{I}$ contains indices that correspond to equation numbers for inequality constraints. With the number of constraints given in the example we could write the constraint set like this:

$$c_1(x) = 0 \tag{7a}$$
$$c_2(x) = 0 \tag{7b}$$
$$c_3(x) \geq 0 \tag{7c}$$
$$c_4(x) \geq 0 \tag{7d}$$
$$c_5(x) \geq 0 \tag{7e}$$

With $\mathcal{E} = \{1, 2\}$ and $\mathcal{I} = \{3, 4, 5\}$ we can write it more compactly as

$$c_i(x) = 0 \qquad i \in \mathcal{E} \tag{8a}$$
$$c_i(x) \geq 0 \qquad i \in \mathcal{I} \tag{8b}$$

We can express the number of equality and inequality constraints compactly by writing $|\mathcal{E}| = 2$ and $|\mathcal{I}| = 3$.

b (4 %) The equality constraint $x_1^2 + 4x_2^2 = 4$ is an ellipse. We can define

$$c_1(x) = x_1^2 + 4x_2^2 - 4 = 0, \quad \mathcal{E} = \{1\} \tag{9}$$

The feasible set is the curve traced by the ellipse. The interior and the exterior of the ellipse are infeasible. Since this is a nonlinear equality constraint the feasible area is nonconvex. Figure 2 shows the ellipse; the feasible area is indicated with green while the infeasible interior ($c_1(x) < 0$) and exterior ($c_1(x) > 0$) are indicated with red.

c (4 %) The inequality constraint $x_1^2 + 4x_2^2 \geq 4$ describes an ellipse and its exterior. We can define

$$c_2(x) = x_1^2 + 4x_2^2 - 4 \geq 0, \quad \mathcal{I} = \{2\} \tag{10}$$

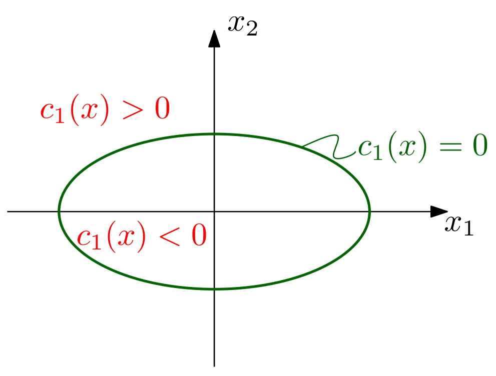

Figure 2: The feasible area of the equality constraint (9) from Problem 2 b, shown in green.

The feasible set is the ellipse and its exterior; the interior is infeasible. The feasible set is nonconvex since a line through two points in the feasible area may exit the feasible area by passing through the interior of the ellipse (which is infeasible). Figure 3 shows the ellipse; the feasible area ($c_1(x) = 0$ and $c_1(x) > 0$) is indicated with green while the infeasible interior ($c_1(x) < 0$) is indicated with red.

d (6 %) We have the constraints

$$c_1(x) = x_1^2 - 4 \geq 0 \tag{11a}$$

$$c_2(x) = -x_1 + 1 \geq 0 \tag{11b}$$

Linearizing the first constraint at $x_1 = 1$ gives

$$\begin{aligned} \nabla c_1(x_1)^\top p + c_1(x_1) &= 2x_1 p + x_1^2 - 4, \quad x_1 = 1 \\ &= 2p + 1 - 4 \\ &= 2p - 3 \geq 0 \end{aligned} \tag{12}$$

Similarly, the second constraint gives

$$\begin{aligned} \nabla c_2(x_1)^\top p + c_2(x_1) &= -p - x_1 + 1, \quad x_1 = 1 \\ &= -p + 1 - 1 \\ &= -p \geq 0 \end{aligned} \tag{13}$$

These linearized constraints mean that both $p \geq 3/2$ and $p \leq 0$ have to be satisfied simultaneously. This is clearly impossible. In other words, the constraints are inconsistent. Note it does not matter that $x_1 = 1$ is infeasible with respect to the original constraints; the fact that a point is infeasible is not a problem for the SQP algorithm and does not necessarily cause problems for the QP approximation. That is, the existence of a feasible set in the QP approximation is not determined by feasibility with respect to the constraints of the NLP.

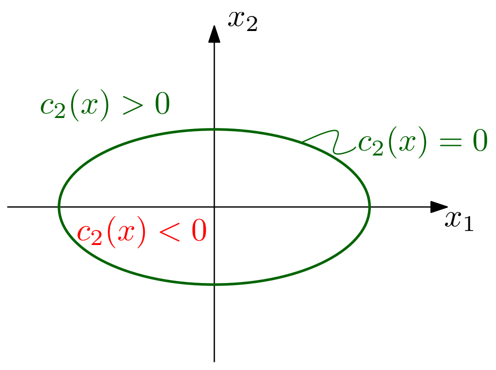

Figure 3: The feasible area of the inequality constraint (10) from Problem 2 c, shown in green.

e (7 %) An equality-constrained nonlinear programming problem can be written

$$\min_{x} \quad f(x) \tag{14a}$$

$$\text{s.t.} \quad c(x) = 0 \tag{14b}$$

where $c(x)$ is a vector of equality-constraint functions. The Lagrangean for the problem is

$$\mathcal{L}(x, \lambda) = f(x) - \lambda^\top c(x) \tag{15}$$

We define a matrix A of constraint gradients (the Jacobian of the constraints),

$$A^\top(x) = \begin{bmatrix} \nabla c_1(x), & \nabla c_2(x), & \dots, & \nabla c_m(x) \end{bmatrix} \tag{16}$$

where $\nabla c_i(x)$ is the ith component of the vector $c(x)$. The gradient with respect to x of the Lagrangean can then be written

$$\nabla_x \mathcal{L}(x, \lambda) = \nabla f(x) - A^\top(x)\lambda \tag{17}$$

The linearized equality constraints

$$\nabla c_i(x_k)^\top p + c_i(x_k) = 0, \quad i \in \mathcal{E} \tag{18}$$

can be written compactly as

$$A(x_k)p + c(x_k) = 0 \tag{19}$$

At iteration k, the objective is the second-order approximation of $\mathcal{L}(x_k, \lambda_k)$:

$$\mathcal{L}(x_k) + (\nabla_x \mathcal{L}(x_k, \lambda_k))^\top p + \frac{1}{2} p^\top \nabla_{xx} \mathcal{L}^\top(x_k, \lambda_k) p \tag{20}$$

With the Lagrangean (15) and its gradient (17) we can rewrite the objective as

$$f(x_k) - \lambda^\top c(x_k) + (\nabla f(x_k) - A^\top(x_k)\lambda)^\top p + \frac{1}{2} p^\top \nabla_{xx} \mathcal{L}^\top(x_k, \lambda_k) p \tag{21}$$

which after some rearranging becomes

$$f(x_k) + \nabla f^\top(x_k) p - \lambda^\top (A(x_k)p + c(x_k)) + \frac{1}{2} p^\top \nabla_{xx} \mathcal{L}^\top(x_k, \lambda_k) p \tag{22}$$

This can again be written

$$f(x_k) + \nabla f^\top(x_k) p + \frac{1}{2} p^\top \nabla_{xx} \mathcal{L}^\top(x_k, \lambda_k) p \tag{23}$$

as long as the linearized equality constraints (19) are satisfied. That is, the objective function (20) can be rewritten as (23) subject to $A(x_k)p + c(x_k) = 0$.

f (7 %) The parameter μ in the merit function is called a penalty parameter; it is used to penalize constraint violation in the line-search part of the SQP algorithm. The parameter decides the relative importance of constraint violation and reduction in the objective function value when the step length is chosen. The penalty parameter is usually smallest in the beginning so that long steps can be made towards the solution. As the algorithm gets closer to the solution the penalty parameter is increased in order to ensure constraint satisfaction closer to the the solution. Hence, the penalty parameter is usually increased from one iteration to the next.

g (6 %) A merit function $\phi(x;\mu)$ is exact if there exists a positive scalar μ^* such that for any $\mu > \mu^*$, any local solution to the nonlinear programming problem (A.1) is a local minimizer of the merit function. It can be shown that the merit function $\phi_1(x;\mu)$ given in the problem set is an exact merit function and that the threshold value for the penalty parameter is given by

$$\mu^* = \max\{|\lambda_i^*|,\ i \in \mathcal{E} \cup \mathcal{I}\} \tag{24}$$

where λ_i^* is the Lagrange multiplier for constraint i at a solution x^*.

h (7 %) The Maratos effect is the phenomenon where a merit function prevents rapid convergence because steps that make good progress toward a solution are rejected. The example illustrated in Figure 4 shows a step that gets very close to the solution, but that would be rejected by some merit functions since both the objective function value and constraint violation is increased. Since not all merit functions suffer from the Maratos effect, one can choose to use one of those merit functions on order to avoid the Maratos effect. Other techniques can also be used, including allowing the merit function to increase at certain iterations (known as a non-monotone strategy), and adding second-order correction steps to the original step so that constraint violation is reduced.

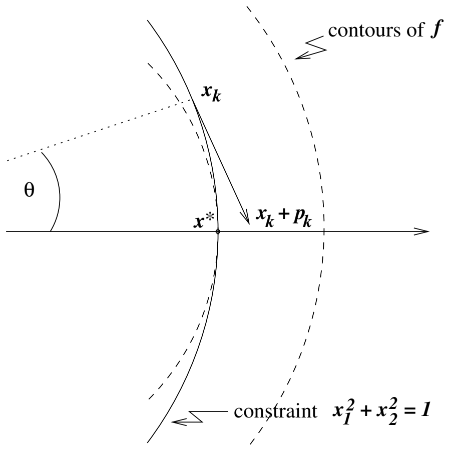

Figure 4: An illustration (taken from the textbook) of the Maratos effect from Problem 2 h.

3 MPC and dynamic optimization (35 %)

a (3 %) The optimization problem (A.9) is called an open-loop optimization problem since the predictions are made without any feedback law that determines the control as a function of the output. In other words, the loop from the state x to input u in open. Note that feedback control is achieved by solving the open-loop optimization problem at every sampling time instant with the latest state measurement as initial condition for x.

b (6 %) z is a 225-dimensional vector when $N = 15$, $n_x = 12$, and $n_u = 3$. In general, $n = N \cdot (n_x + n_u)$ and $z \in \mathbb{R}^n$. The number of variables n grows linearly with the length of the prediction horizon N.

c (4 %) The control moves Δu_t can be limited by adding a quadratic penalty term to the objective function (A.9a). This can be done by augmenting it with the term

$$\frac{1}{2}\sum_{t=0}^{N-1} \Delta u_t^\top R_{\Delta t} \Delta u_t \tag{25}$$

where $R_{\Delta t} \succeq 0$ is a penalty weight matrix.

d (6 %) With all linear terms in the objective set to zero and the weight matrices Q and R time invariant, the objective function (A.9a) becomes

$$\frac{1}{2}\sum_{t=0}^{N-1} x_{t+1}^\top Q x_{t+1} + u_t^\top R u_t \tag{26}$$

A first step in the tuning process is often scaling the problem so that all variables have similar ranges. If one decides not to scale, some care must be taken when choosing weights. For instance, if the state x_1 is typically between -0.2 and 0.2 and the state x_2 is typically in the range -50 to 50, they are weighted similarly if $0.2^2 q_1 = 50^2 q_2$, meaning $q_1 = (50/0.2)^2 q_2 = 62500 q_2$. The first state is weighted twice that of the second state if $q_1 = 2 \times 62500 q_2$. In general, states that are more important by some measure should have a higher weight relative to the less important ones. The same approach can be used to weight the different inputs against each other. The states must also be weighted against the control inputs, in which case it may be more difficult to decide on priorities among a number of states and control inputs. After deciding on a set of weights, simulation can be used to evaluate the choice. The weights should be adjusted until satisfactory simulation results are obtained for a wide range of conditions.

e (6 %) The estimator takes the measurement y and the control u as inputs and produces the state estimate $\hat{x}$ as output. From the figure we see that

$$\begin{aligned}\hat{x}_{t+1} &= A\hat{x}_t + Bu_t + K_F(y_t - \hat{y}_t) \\ &= A\hat{x}_t + Bu_t + K_F(y_t - C\hat{x}_t) \\ &= (A - K_F C)\hat{x}_t + Bu_t + K_F y_t \end{aligned} \tag{27}$$

The estimator can be tuned by the choice of K_F matrix which decided the eigenvalues of $A - K_F C$. It is important that the estimator be faster than the dominating dynamics in the MPC feedback loop since an estimator that lags behind the controller would supply "wrong" estimates.

f (10 %) A typical control hierarchy is shown in Figure 5.

The APC layer generates optimal setpoints for control u^{ref} and outputs y^{ref} or optimal reference trajectories u_t^{ref} and y_t^{ref}. These are sent to the regulator layer where less advanced controllers try to keep the process output close to the reference using a control action close to the setpoint. The APC layer receives measurements from the regulatory level and uses these to compute new optimal references.

The sampling frequency will in general increase downwards in the hierarchy. The regulator layer will in most applications operate at a high frequency while the APC layers may have a sampling frequency that can be orders of magnitude lower.

In practice the APC layer communicates with a realtime database rather than the DCS layer directly. This is a common way of implementing important features such as integrity quality control of the data used by the APC layer, where errors in the data set used for optimization can lead to issues like infeasabilty if the data is used in its raw form. Once the APC layer has completed its computational task it will normally write the result back to the realtime database rather than push it directly to the regulator layer. In this way, the DCS can read the data when necessary and subsequently adjust the appropriate setpoints. This design structure is shown in Figure 6

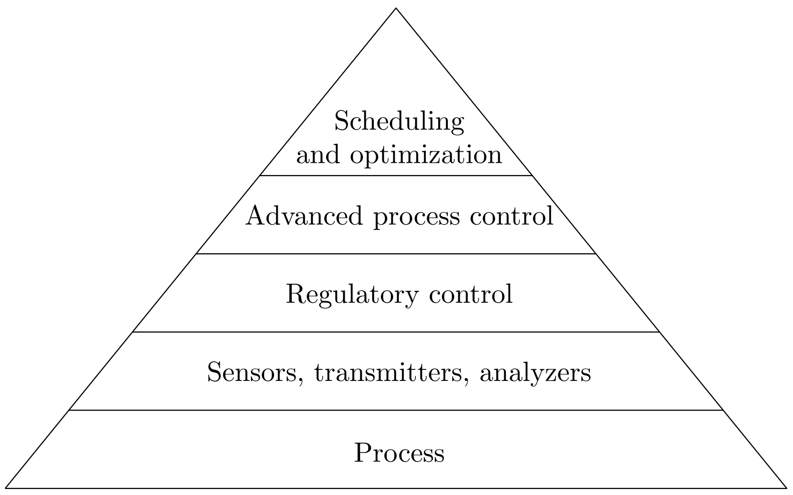

Figure 5: A typical control hierarchy from Problem 2 f.

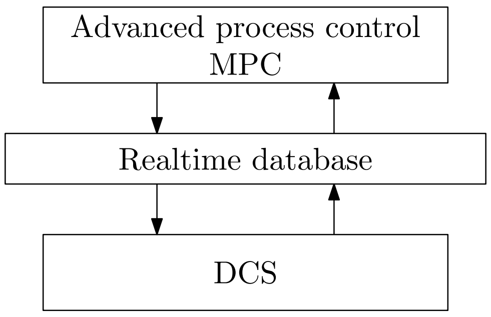

Figure 6: Illustration of APC layer implementation from Problem 2 f.

Appendix

Part 1 Optimization Problems and Optimality Conditions

A general formulation for constrained optimization problems is

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) && \text{(A.1a)} \\ \text{s.t.} \quad & c_i(x) = 0, \qquad i \in \mathcal{E} && \text{(A.1b)} \\ & c_i(x) \geq 0, \qquad i \in \mathcal{I} && \text{(A.1c)} \end{aligned}$$

where f and the functions c_i are all smooth, differentiable, real-valued functions on a subset of $\mathbb{R}^n$, and $\mathcal{E}$ and $\mathcal{I}$ are two finite sets of indices.

The Lagrangean function for the general problem (A.1) is

$$\mathcal{L}(x, \lambda) = f(x) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i c_i(x) \tag{A.2}$$

The KKT-conditions for (A.1) are given by:

$$\begin{aligned} \nabla_x \mathcal{L}(x^*, \lambda^*) &= 0 && \text{(A.3a)} \\ c_i(x^*) &= 0, \qquad i \in \mathcal{E} && \text{(A.3b)} \\ c_i(x^*) &\geq 0, \qquad i \in \mathcal{I} && \text{(A.3c)} \\ \lambda_i^* &\geq 0, \qquad i \in \mathcal{I} && \text{(A.3d)} \\ \lambda_i^* c_i(x^*) &= 0, \qquad i \in \mathcal{E} \cup \mathcal{I} && \text{(A.3e)} \end{aligned}$$

2nd order (sufficient) conditions for (A.1) are given by:

$$w \in \mathcal{C}(x^*, \lambda^*) \Leftrightarrow \begin{cases} \nabla c_i(x^*)^\top w = 0 & \text{for all } i \in \mathcal{E} \\ \nabla c_i(x^*)^\top w = 0 & \text{for all } i \in \mathcal{A}(x^*) \cap \mathcal{I} \text{ with } \lambda_i^* > 0 \\ \nabla c_i(x^*)^\top w \geq 0 & \text{for all } i \in \mathcal{A}(x^*) \cap \mathcal{I} \text{ with } \lambda_i^* = 0 \end{cases} \tag{A.4}$$

Theorem 1: (Second-Order Sufficient Conditions) *Suppose that for some feasible point $x^* \in \mathbb{R}^n$ there is a Lagrange multiplier vector λ^* such that the KKT conditions* (A.3) *are satisfied. Suppose also that*

$$w^\top \nabla^2_{xx} \mathcal{L}(x^*, \lambda^*) w > 0, \qquad \text{for all } w \in \mathcal{C}(x^*, \lambda^*),\ w \neq 0. \tag{A.5}$$

Then x^* is a strict local solution for (A.1).

LP problem in standard form:

$$\begin{aligned} \min_{x} \quad & f(x) = c^\top x && \text{(A.6a)} \\ \text{s.t.} \quad & Ax = b && \text{(A.6b)} \\ & x \geq 0 && \text{(A.6c)} \end{aligned}$$

where $A \in \mathbb{R}^{m \times n}$ and $\operatorname{rank} A = m$.

QP problem in standard form:

$$\begin{aligned} \min_{x} \quad & f(x) = \frac{1}{2} x^\top G x + x^\top c && \text{(A.7a)} \\ \text{s.t.} \quad & a_i^\top x = b_i, \qquad i \in \mathcal{E} && \text{(A.7b)} \\ & a_i^\top x \geq b_i, \qquad i \in \mathcal{I} && \text{(A.7c)} \end{aligned}$$

where G is a symmetric $n \times n$ matrix, $\mathcal{E}$ and $\mathcal{I}$ are finite sets of indices and c, x and $\{a_i\}, i \in \mathcal{E} \cup \mathcal{I}$, are vectors in $\mathbb{R}^n$. Alternatively, the equalities can be written $Ax = b$, $A \in \mathbb{R}^{m \times n}$.

Iterative method:

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k p_k && \text{(A.8a)} \\ & x_0 \text{ given} && \text{(A.8b)} \\ x_k, p_k &\in \mathbb{R}^n, \ \alpha_k \in \mathbb{R} && \text{(A.8c)} \end{aligned}$$

p_k is the search direction and α_k is the line search parameter.

Part 2 Optimal Control

A typical open-loop optimal control problem on the time horizon 0 to N is

$$\min_{z\in\mathbb{R}^n} f(z) = \sum_{t=0}^{N-1} \frac{1}{2}x_{t+1}^\top Q_{t+1}x_{t+1} + d_{xt+1}x_{t+1} + \frac{1}{2}u_t^\top R_t u_t + d_{ut}u_t \tag{A.9a}$$

subject to

$$\begin{aligned}
x_{t+1} &= A_t x_t + B_t u_t, & t &= 0,\dots,N-1 & \text{(A.9b)}\\
x_0 &= \text{given} & & & \text{(A.9c)}\\
x^{\text{low}} &\le x_t \le x^{\text{high}}, & t &= 1,\dots,N & \text{(A.9d)}\\
u^{\text{low}} &\le u_t \le u^{\text{high}}, & t &= 0,\dots,N-1 & \text{(A.9e)}\\
-\Delta u^{\text{high}} &\le \Delta u_t \le \Delta u^{\text{high}}, & t &= 0,\dots,N-1 & \text{(A.9f)}\\
Q_t &\succeq 0 & t &= 1,\dots,N & \text{(A.9g)}\\
R_t &\succeq 0 & t &= 0,\dots,N-1 & \text{(A.9h)}
\end{aligned}$$

where

$$\begin{aligned}
u_t &\in \mathbb{R}^{n_u} & \text{(A.9i)}\\
x_t &\in \mathbb{R}^{n_x} & \text{(A.9j)}\\
\Delta u_t &= u_t - u_{t-1} & \text{(A.9k)}\\
z^\top &= (x_1^\top,\dots,x_N^\top,u_0^\top,\dots,u_{N-1}^\top) & \text{(A.9l)}
\end{aligned}$$

The subscript t denotes discrete time sampling instants.

The optimization problem for linear quadratic control of discrete dynamic systems is given by

$$\min_{z\in\mathbb{R}^n} f(z) = \sum_{t=0}^{N-1} \frac{1}{2}x_{t+1}^\top Q_{t+1}x_{t+1} + \frac{1}{2}u_t^\top R_t u_t \tag{A.10a}$$

subject to

$$\begin{aligned}
x_{t+1} &= A_t x_t + B_t u_t & \text{(A.10b)}\\
x_0 &= \text{given} & \text{(A.10c)}
\end{aligned}$$

where

$$\begin{aligned}
u_t &\in \mathbb{R}^{n_u} & \text{(A.10d)}\\
x_t &\in \mathbb{R}^{n_x} & \text{(A.10e)}\\
z^\top &= (x_1^\top,\dots,x_N^\top,u_0^\top \dots,u_{N-1}^\top) & \text{(A.10f)}
\end{aligned}$$

Theorem 2: *The solution of (A.10) with $Q_t \succeq 0$ and $R_t \succ 0$ is given by*

$$u_t = -K_t x_t \tag{A.11a}$$

where the feedback gain matrix is derived by

$$K_t = R_t^{-1} B_t^\top P_{t+1}(I + B_t R_t^{-1} B_t^\top P_{t+1})^{-1} A_t, \qquad t = 0, \ldots, N-1 \tag{A.11b}$$

$$P_t = Q_t + A_t^\top P_{t+1}(I + B_t R_t^{-1} B_t^\top P_{t+1})^{-1} A_t, \qquad t = 0, \ldots, N-1 \tag{A.11c}$$

$$P_N = Q_N \tag{A.11d}$$

Part 3 Sequential quadratic programming (SQP)

Algorithm 18.3 (Line Search SQP Algorithm).

```
Choose parameters $\eta \in (0, 0.5)$, $\tau \in (0, 1)$, and an initial pair $(x_0, \lambda_0)$;
Evaluate $f_0, \nabla f_0, c_0, A_0$;
If a quasi-Newton approximation is used, choose an initial $n \times n$ symmetric
positive definite Hessian approximation $B_0$, otherwise compute $\nabla^2_{xx}\mathcal{L}_0$;
**repeat** until a convergence test is satisfied
        Compute $p_k$ by solving (18.11); let $\hat{\lambda}$ be the corresponding multiplier;
        Set $p_\lambda \leftarrow \hat{\lambda} - \lambda_k$;
        Choose $\mu_k$ to satisfy (18.36) with $\sigma = 1$;
        Set $\alpha_k \leftarrow 1$;
        **while** $\phi_1(x_k + \alpha_k p_k; \mu_k) > \phi_1(x_k; \mu_k) + \eta \alpha_k D_1(\phi(x_k; \mu_k) p_k)$
                Reset $\alpha_k \leftarrow \tau_\alpha \alpha_k$ for some $\tau_\alpha \in (0, \tau]$;
        **end (while)**
        Set $x_{k+1} \leftarrow x_k + \alpha_k p_k$ and $\lambda_{k+1} \leftarrow \lambda_k + \alpha_k p_\lambda$;
        Evaluate $f_{k+1}, \nabla f_{k+1}, c_{k+1}, A_{k+1}$, (and possibly $\nabla^2_{xx}\mathcal{L}_{k+1}$);
        If a quasi-Newton approximation is used, set
                $s_k \leftarrow \alpha_k p_k$ and $y_k \leftarrow \nabla_x \mathcal{L}(x_{k+1}, \lambda_{k+1}) - \nabla_x \mathcal{L}(x_k, \lambda_{k+1})$,
        and obtain $B_{k+1}$ by updating $B_k$ using a quasi-Newton formula;
**end (repeat)**
```