QOCO Parsing

Govind M. Chari
Autonomous Controls Laboratory (ACL)
University of Washington, Seattle

May 23, 2024

1 QOCO Standard Form

QOCO solves second-order cone programs of the following form:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2}x^\top Px + c^\top x \\ \text{subject to} & Gx \preceq_{\mathcal{C}} h \\ & Ax = b \end{array} \tag{1}$$

with optimization variable $x \in \mathbb{R}^n$ and problem data $P = P^\top \succeq 0$, $c \in \mathbb{R}^n$, $G \in \mathbb{R}^{m \times n}$, $h \in \mathbb{R}^m$, $A \in \mathbb{R}^{p \times n}$, $b \in \mathbb{R}^p$, and $\preceq_{\mathcal{C}}$ is an inequality with respect to cone $\mathcal{C}$. Cone $\mathcal{C}$ is the Cartesian product of the non-negative orthant and second-order cones, which can be expressed as

$$\mathcal{C} = \mathbb{R}^l_+ \times \mathcal{Q}^{q_1}_1 \times \ldots \times \mathcal{Q}^{q_{\text{nsoc}}}_{\text{nsoc}} \tag{2}$$

where l is the dimension of the non-negative orthant, and $\mathcal{Q}^{q_i}_i$ is the i^{th} second-order cone with dimension q_i defined by

$$\mathcal{Q}^{q_i}_i = \left\{(t, x) \in \mathbb{R} \times \mathbb{R}^{q_i - 1} \mid \|x\|_2 \leq t\right\} \tag{3}$$

2 Optimal Control Problem Parsing

To solve any SOCP with QOCO we will first need to put in in the form given by Equation 1. In this section we will present an optimal control problem and show how to hand parse it into the cannonical form that QOCO accepts.

We will consider a variation of the three-degree of freedom powered-descent guidance problem. The state at time-step t is given by $z_t = (r_t, v_t)$, where $r_t \in \mathbb{R}^3$ is the position of the spacecraft and $v_t \in \mathbb{R}^3$ is the velocity of the spacecraft. The control at time t, $u_t \in \mathbb{R}^3$, is the thrust produced by the spacecraft's engines. We have initial and terminal conditions on the state as well as a box constraint on velocity given by 4e, maximum thrust constraint given by 4f and a pointing constraint on the thrust vector given by 4g.

$$
\begin{aligned}
\underset{z_t, u_t}{\text{minimize}} \quad & \frac{1}{2}\left(\sum_{t=1}^{N} z_t^\top Q_t z_t + \sum_{t=1}^{N-1} u_t^\top R_t u_t\right) && \text{(4a)}\\
\text{subject to} \quad & z_{t+1} = A_d z_t + B_d u_t + c \quad t \in [1, N-1] && \text{(4b)}\\
& z_1 = z_{\text{init}} && \text{(4c)}\\
& z_N = z_{\text{final}} && \text{(4d)}\\
& \|v_t\|_\infty \le v_{\max} \quad t \in [1, N] && \text{(4e)}\\
& \|u_t\|_2 \le u_{\max} \quad t \in [1, N-1] && \text{(4f)}\\
& \|u_t\|_2 \le \tan(\theta_{\max}) u_t^\top e \quad t \in [1, N-1] && \text{(4g)}
\end{aligned}
$$

where N is the time horizon, Q_t and R_t are the state and control penalty matrices respectively, x_{init} and x_{final} are the boundary conditions, and A_t^d, B_t^d and g_t are the dynamics matrices and feed-forward dynamics vector due to gravity respectively, and $e = [1\ 0\ 0]^\top$. For the double integrator with gravity, the dynamics are linear time invariant ($A_1^d = \ldots = A_{N-1}^d$ and same for B^d and g) but to keep this parsing example general, we represent the dynamics as a linear time variant system.

To parse Problem 4 into Problem 1, we must first introduce a slack variable, $\xi \in \mathbb{R}$, and an equality constraint to represent 4f as a second-order cone constraint, and we rewrite the infinity norm constraint on velocity as a system of inequalities.

$$
\begin{aligned}
\underset{z_t, u_t, \xi}{\text{minimize}} \quad & \frac{1}{2}\left(\sum_{t=1}^{N} z_t^\top Q_t z_t + \sum_{t=1}^{N-1} u_t^\top R_t u_t\right) && \text{(5a)}\\
\text{subject to} \quad & z_{t+1} = A_t^d z_t + B_t^d u_t + g \quad t \in [1, N-1] && \text{(5b)}\\
& z_1 = z_{\text{init}} && \text{(5c)}\\
& z_N = z_{\text{final}} && \text{(5d)}\\
& -v_{\max} 1_{3\times 1} \le v_t \le v_{\max} 1_{3\times 1} \quad t \in [1, N] && \text{(5e)}\\
& \|u_t\|_2 \le \xi \quad t \in [1, N-1] && \text{(5f)}\\
& \xi = u_{\max} && \text{(5g)}\\
& \|u_t\|_2 \le \tan(\theta_{\max}) u_t^\top e \quad t \in [1, N-1] && \text{(5h)}
\end{aligned}
$$

From here, we can finally parse Problem 5 into Problem 1 with the following dimensions and data in Problem 1.

$$
\begin{aligned}
n &= 9N - 2 && \text{(6a)}\\
p &= 6N + 7 && \text{(6b)}\\
m &= 0 && \text{(6c)}\\
l &= 6N && \text{(6d)}\\
\text{nsoc} &= 2N - 2 && \text{(6e)}\\
q &= (4_{N-1\times 1}, 3_{N-1\times 1}) && \text{(6f)}\\
\mathcal{C} &= \mathbb{R}_+^l \times \mathcal{Q}_1^4 \times \ldots \mathcal{Q}_{N-1}^4 \times \mathcal{Q}_1^3 \times \ldots \mathcal{Q}_{N-1}^3 && \text{(6g)}
\end{aligned}
$$

$$
x = (z_1, \ldots, z_N, u_1, \ldots, u_{N-1}, \xi) \tag{7}
$$

$$P = \begin{bmatrix} Q_1 & & & & & & \\ & \ddots & & & & & \\ & & Q_N & & & & \\ & & & R_1 & & & \\ & & & & \ddots & & \\ & & & & & R_{N-1} & \\ & & & & & & 0 \end{bmatrix} \tag{8}$$

$$c = 0_{n \times 1} \tag{9}$$

$$A = \begin{bmatrix} A_z^{\text{dyn}} & A_u^{\text{dyn}} & A_\xi^{\text{dyn}} \\ A_z^{\text{bc}} & A_u^{\text{bc}} & A_\xi^{\text{bc}} \\ A_z^{\text{slack}} & A_u^{\text{slack}} & A_\xi^{\text{slack}} \end{bmatrix} \tag{10}$$

$$A_z^{\text{dyn}} = \begin{bmatrix} A_1^d & -I_6 & & & & \\ & A_2^d & -I_6 & & & \\ & & & \ddots & & \\ & & & & A_{N-1}^d & -I_6 \end{bmatrix} \tag{11a}$$

$$A_u^{\text{dyn}} = \begin{bmatrix} B_1^d & & & \\ & B_2^d & & \\ & & \ddots & \\ & & & B_{N-1}^d \end{bmatrix} \tag{11b}$$

$$A_\xi^{\text{dyn}} = 0_{6(N-1) \times 1} \tag{11c}$$

$$A_z^{\text{bc}} = \begin{bmatrix} I_6 & 0_{6 \times 6(N-1)} \\ 0_{6 \times 6(N-1)} & I_6 \end{bmatrix} \tag{11d}$$

$$A_u^{\text{bc}} = 0_{12 \times 3(N-1)} \tag{11e}$$

$$A_\xi^{\text{bc}} = 0_{12 \times 1} \tag{11f}$$

$$A_z^{\text{slack}} = 0_{1 \times 6N} \tag{11g}$$

$$A_u^{\text{slack}} = 0_{1 \times 3(N-1)} \tag{11h}$$

$$A_\xi^{\text{slack}} = 1 \tag{11i}$$

$$\tag{11j}$$

$$b = (-g_1, \ldots, -g_{N-1}, z_{\text{init}}, z_{\text{final}}, u_{\text{max}}) \tag{12}$$

$$G = \begin{bmatrix} G_z^{\text{velocity}} & G_u^{\text{velocity}} & G_\xi^{\text{velocity}} \\ G_z^{\text{thrust}} & G_u^{\text{thrust}} & G_\xi^{\text{thrust}} \\ G_z^{\text{pointing}} & G_u^{\text{pointing}} & G_\xi^{\text{pointing}} \end{bmatrix} \tag{13}$$

$$G_z^{\text{velocity}} = \begin{bmatrix} 0_{3\times 3} & I_3 & & & & \\ & & 0_{3\times 3} & I_3 & & \\ & & & & \ddots & \\ & & & & 0_{3\times 3} & I_3 \\ 0_{3\times 3} & I_3 & & & & \\ & & 0_{3\times 3} & I_3 & & \\ & & & & \ddots & \\ & & & & 0_{3\times 3} & I_3 \end{bmatrix} \tag{14a}$$

$$G_u^{\text{velocity}} = 0_{6N\times 3(N-1)} \tag{14b}$$

$$G_\xi^{\text{velocity}} = 0_{6N\times 1} \tag{14c}$$

$$G_z^{\text{thrust}} = 0_{4(N-1)\times 6N} \tag{14d}$$

$$G_u^{\text{thrust}} = \begin{bmatrix} 0_{1\times 3} & & & \\ -I_3 & & & \\ & 0_{1\times 3} & & \\ & -I_3 & & \\ & & \ddots & \\ & & & 0_{1\times 3} \\ & & & -I_3 \end{bmatrix} \tag{14e}$$

$$G_\xi^{\text{thrust}} = (-1, 0_{3\times 1}, -1, 0_{3\times 1}, \dots, -1, 0_{3\times 1}) \tag{14f}$$

$$G_z^{\text{pointing}} = 0_{3(N-1)\times 6N} \tag{14g}$$

$$G_u^{\text{pointing}} = \begin{bmatrix} -\tan(\theta_{\max}) & & & & & & \\ & -I_2 & & & & & \\ & & -\tan(\theta_{\max}) & & & & \\ & & & -I_2 & & & \\ & & & & \ddots & & \\ & & & & & -\tan(\theta_{\max}) & \\ & & & & & & -I_2 \end{bmatrix} \tag{14h}$$

$$G_\xi^{\text{pointing}} = 0_{3(N-1)\times 1} \tag{14i}$$

$$h = (v_{\max} 1_{6N\times 1}, 0_{4(N-1)\times 1}, 0_{3(N-1)\times 1}) \tag{15}$$

3 Parsing Code

```python
from scipy import sparse
import numpy as np

N = 30  # Number of timesteps.
dt = 0.5  # Discretization interval.
g0 = 9.8  # Gravitational acceleration.
zi = np.array([100, 50, 50, -9, 5, -9])  # Initial condition.
zf = np.zeros(6)  # Terminal condition.
Q = 1.0 * sparse.eye(6)  # State cost matrix.
R = 5.0 * sparse.eye(3)  # Input cost matrix.
vmax = 10.0  # Max inf norm on velocity.
umax = 12.0  # Maximum thrust.
thmax = np.deg2rad(35.)  # Maximum Thrust pointing angle.

# Number of optimization variables.
n = 9 * N - 2

```

```python
# Number of affine equality constraints (rows of A).
p = 6 * N + 7

# Number of conic constraints (rows of G).
m = 13 * N - 7

# Dimension of non-negative orthant in cone C.
l = 6 * N

# Number of second order cones.
nsoc = 2*N - 2

# Dimension of each second order cone.
q = np.hstack((4 * np.ones(N - 1), 3 * np.ones(N - 1)))

# Parse cost function.
Qfull = sparse.kron(sparse.eye(N), Q)
Rfull = sparse.kron(sparse.eye(N - 1), R)
P = sparse.block_diag((Qfull, Rfull, 0.0*sparse.eye(1)))
c = np.zeros(n)

# Double integrator dynamics.
Ad = np.block([[np.eye(3), dt*np.eye(3)], [np.zeros((3, 3)), np.eye(3)]])
Bd = np.block([[0.5*dt**2*np.eye(3)], [dt*np.eye(3)]])
g = np.array([-0.5*g0*dt**2, 0, 0, -g0*dt, 0, 0])

# Parse dynamics constraint.
Azdyn = np.block([np.kron(np.eye(N-1), Ad), np.zeros((6*(N - 1), 6))]
                 ) - np.block([np.zeros((6*(N-1), 6)), np.eye(6*(N - 1))])
Audyn = np.kron(np.eye(N-1), Bd)
Axidyn = np.zeros((6*(N - 1), 1))

# Parse boundary conditions.
Azbc = np.block([[np.eye(6), np.zeros((6, 6*(N-1)))],
                [np.zeros((6, 6*(N-1))), np.eye(6)]])
Aubc = np.zeros((12, 3 * (N - 1)))
Axibc = np.zeros((12, 1))

# Parse slack variable.
Azslack = np.zeros((1, 6*N))
Auslack = np.zeros((1, 3*(N - 1)))
Axislack = np.array([1.0])

# Combine dynamics, boundary conditions, and slack equality into equality constraint
    matrix A, and vector b.
A = np.block([[Azdyn, Audyn, Axidyn], [Azbc, Aubc, Axibc],
                [Azslack, Auslack, Axislack]])
b = np.hstack((np.kron(np.ones(N - 1), -g), zi, zf, umax))

# Parse velocity constraint.
Gzvelocity = np.block([[np.kron(np.eye(N), np.block([np.zeros((3, 3)), np.eye(3)]))],
                        [np.kron(np.eye(N), np.block([np.zeros((3, 3)), -np.eye(3)]))
    ]])
Guvelocity = np.zeros((6 * N, 3*(N-1)))
Gxivelocity = np.zeros((6*N, 1))

# Parse thrust constraint.
Gzthrust = np.zeros((4*(N-1), 6*N))
Guthrust = np.kron(
    np.eye(N - 1), np.block([[np.zeros((1, 3))], [-np.eye(3)]]))
Gxithrust = np.kron(np.ones(N-1), np.array([-1, 0, 0, 0]))
Gxithrust = np.asmatrix(Gxithrust).T

# Parse pointing constraint.
Gzpointing = np.zeros((3*(N-1), 6*N))
block = -np.eye(3)
```

```python
block[0, 0] = -np.tan(thmax)
Gupointing = np.kron(np.eye(N-1), block)
Gxipointing = np.zeros((3*(N-1), 1))

# Combine velocity box constraint, thrust ball constraint, and thrust pointing constraint into G and h.
G = np.block([[Gzvelocity, Guvelocity, Gxivelocity], [
                Gzthrust, Guthrust, Gxithrust], [Gzpointing, Gupointing, Gxipointing]])
h = np.hstack((vmax * np.ones((6 * N)),
                np.zeros(4*(N-1)), np.zeros(3*(N-1))))
```

Listing 1: OCP Parsing in Python