Convex Optimization and Fuel-Optimal Rendezvous

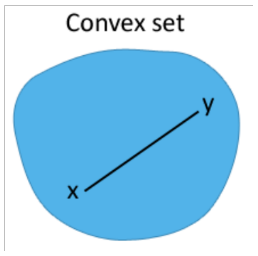
Govind Chari and Sam Noles

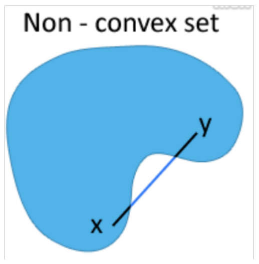
May 3, 2022

1 What/Why is Convex Optimization Important?

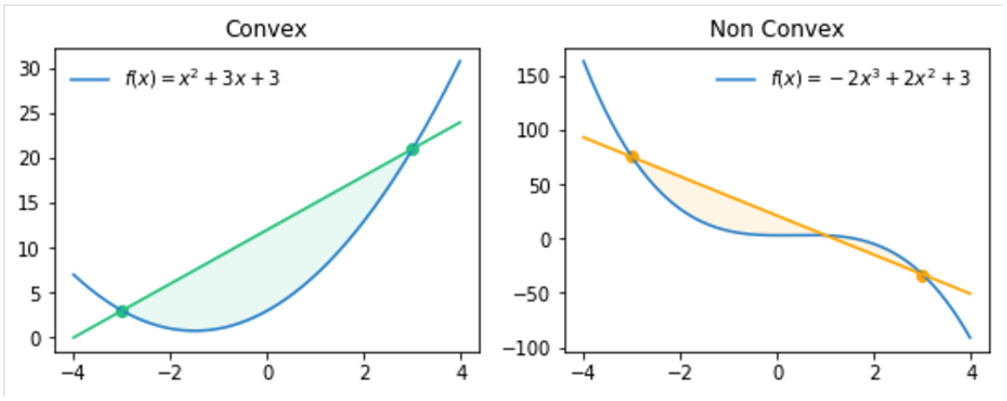
Convex optimization is a subclass of optimization where you are optimizing a convex function over a convex set.

A convex set is any set where if you pick any two points in the set and connect them with a line, the line lies fully within the set. Below is a figure of a convex and a nonconvex set.





A convex function is a function where if you pick any two points and draw a secant line, the line lies above the function. You can think of any convex function of any function that resembles a bowl, such as a parabola or a paraboloid. A convex and nonconvex function are shown below.



One important property of convex optimization problems is that a local minima is a global minima. This makes optimization methods extremely fast to converge (can be on the order of tenths of seconds). If a trajectory optimization problem can be formulated as a convex optimization problem, then the optimal trajectory can be solved for extremely quickly and the satellite/spacecraft can repeatedly re-optimize its trajectory while in flight. The primary appeal of these methods is that you can ensure that your solution meets all your problem constraints. This method is famously used in SpaceX's landing algorithm for their Falcon 9 booster [1].

2 Quadratic Programs

A quadratic program (QP) is a subclass of convex optimization problem when the objective function is a a convex quadratic function (for example a parabola or paraboloid), subject to linear equality and linear inequality constraints. The general form of this problem can be mathamatically written as the following:

$$\begin{array}{ll} \min\limits_{x} & \frac{1}{2}x^T Q x + q^T x \\ \text{s.t.} & Ax = b \\ & Gx \preceq h \end{array}$$

Where $\preceq$ indicates element-wise inequalities, and $Q > 0$.

QP's are the most common class of convex optimization problems that are used for trajectory generation. QP's are used by SpaceX to land their Falcon 9 booster [1], MIT in their Cheetah robot [4], and Boston Dynamics for their Atlas robot [5].

3 KKT Conditions

Before looking at how to solve quadratic programs, we must first understand the properties of optimal solutions to convex optimization problems.

For an unconstrained convex function $f : \mathbb{R}^n \to \mathbb{R}$, the necessary and sufficient condition for optimality is

$$\nabla f = 0$$

. For example consider $f(x) = x^2$. To find the minimum of this function we would set $\nabla f = 0$ and solve to get $x^* = 0$. However when we get introduce constraints, we must have different optimality conditions. For example consider minimizing $f(x) = x^2$ on the interval $x \in [1, 3]$. We can no longer set the derivative of $f(x)$ to 0 and solve for x^*. The optimality conditions for this constrained problem are called the KKT conditions. First we will introduce a slack variable, s in the inequality constraint to turn it into an equality constraint. Now we can write:

$$\begin{array}{lr} \min\limits_{x} & \frac{1}{2}x^T Q x + q^T x \\ \text{s.t.} & Ax = b \\ & Gx + s = h \\ & s \geq 0 \end{array}$$

We will now form the Lagrangian from multivariable calculus.

$$\mathcal{L}(x, s, y, z) = \frac{1}{2}x^T Q x + q^T x + y^T(Ax - b) + z^T(Gx + s - h)$$

where $s \geq 0$ and $z \geq 0$ and y and z are vectors of Lagrange multiplier corresponding to the equality and inequality constraints respectively. The intuition behind this is that we move the constraints into the objective function and penalize constraint violations. Now we can write the KKT conditions which are necessary and sufficient conditions for optimality for convex optimization problems. This means that if we find vectors x, s, y, and z that satisfies these KKT conditions, then x is our optimal solution and y and z are our Lagrange multipliers. The KKT conditions for our QP are as follows

$$\begin{gathered} \nabla_x \mathcal{L} = 0 \\ Ax - b = 0 \\ Gx + s - h = 0 \\ s \geq 0 \\ z \geq 0 \\ z_i s_i = 0 \end{gathered}$$

The intuition behind these conditions are as follows. The first condition is similar the the $f = 0$ condition for unconstrained problems, but just accounts for the existance of constraints. The second and third conditions just tell us that the optimal point must satisfy the equality and inequality constraints. The fourth and fifth constraints just tell us that s must be positive to be a slack variable, and all elements of z must be positive in order to penalize violations of the inequality constraint. The last constraint is a bit tougher to explain. The intuitive meaning behind the Lagrange multiplier z is that the values of z tell us how much more we could decrease the objective function if we relaxed the inequality constraint to $Gx - h \preceq 1$. Basically it tells us how much we can decrease our optimal solution by if we relax our constraints a bit. So this last condition says that either we have a nonzero Lagrange multiplier or our inequality constraint is active ($(Gx - h)_i = 0$).

The first condition is called stationarity, the second through fourth conditions are called primal feasibility, the fifth condition is called dual feasibility, and the last constraint is called complementary slackness.

4 Primal-Dual Interior Point Method (CVXGEN Solver) [3,6]

In this section we will see how to solve quadratic programs using the Primal-Dual Interior Point Method. This is the algorithm that the CVXGEN solver implements, which is the solver used by SpaceX for rocket landings [1].

We can rewrite the KKT conditions as the following nonlinear root-finding problem:

$$F(x, s, y, z) = \begin{bmatrix} Qx + q + A^T + G^T x \\ ZS\mathbf{1}^T \\ Gx + s - h \\ Ax - b \end{bmatrix} = 0$$

where $Z = diag(z)$, $S = diag(s)$, and $\mathbf{1}$ is a vector of all ones.

This nonlinear root-finding problem can be solved via Newton-Raphson iteration. The actual CVXGEN solver implements Newton's method with a number of additions such as centering steps, and corrector steps as well as a line-search for better convergence and robustness, but the core idea of the Primal-Dual Interior Point Method is that we write the KKT conditions as a nonlinear root-finding problem and use Newton-Raphson. We can set up the system below solve for the iteration update vector:

$$\begin{bmatrix} Q & 0 & G^T & A^T \\ 0 & Z & S & 0 \\ G & I & 0 & 0 \\ A & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta z \\ \Delta y \end{bmatrix} = \begin{bmatrix} -(Qx + q + A^T + G^T x) \\ -(ZS\mathbf{1}^T) \\ -(Gx + s - h) \\ -(Ax - b) \end{bmatrix}$$

We then iterate until our solution converges.

5 Application to Rendezvous

We can formula the fuel-optimal rendezvous problem as a convex optimization problem. We will first consider the fuel-optimal problem without state or input constraints, but then we will add in a thrust bound and an approach cone constraint.

When the fuel-optimal rendezvous problem is solved we will get a sequence of inputs (thrust vectors) that the spacecraft needs to apply in order to rendezvous with the ISS using minimum fuel. We will also get the state trajectory (position and velocity vs time) of the rendezvousing spacecraft.

5.1 Dynamics

We will consider the problem of a spacecraft rendezvousing with the ISS. The ISS is in a circular orbit and we will assume that the rendezvousing spacecraft is close enough to the ISS that the Clohessy-Wiltshire equations are valid. This system is a continuous-time linear time invariant system that can be written as

$$\dot{\boldsymbol{x}} = A_c \boldsymbol{x} + B_c \boldsymbol{u}$$

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & -n^2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}$$

5.2 Discretization

To use these dynamics in a numerical optimizer, we need to discretize the dynamics. We can discretize the A matrix exactly using the unforced solution of the Clohessy-Wiltshire equations and we can discretize the B matrix using an impulse model where a control input instintaneously changes the spacecraft's velocity. The discretized system can be written as

$$\boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k$$

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} 4-3cnt & 0 & 0 & (1/n)snt & (2/n)(1-cnt) & 0 \\ 6(snt-nt) & 1 & 0 & (2/n)(cnt-1) & (1/n)(4snt-3nt) & 0 \\ 0 & 0 & cnt & 0 & 0 & (1/n)snt \\ 3nsnt & 0 & 0 & cnt & 2snt & 0 \\ 6n(cnt-1) & 0 & 0 & -2snt & 4cnt-3 & 0 \\ 0 & 0 & -nsnt & 0 & 0 & cnt \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} +$$

$$\begin{bmatrix} (1/n)snt & (2/n)(1-cnt) & 0 \\ (2/n)(cnt-1) & (1/n)(4snt-3nt) & 0 \\ 0 & 0 & (1/n)snt \\ cnt & 2snt & 0 \\ -2snt & 4cnt-3 & 0 \\ 0 & 0 & cnt \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}$$

where $nt = n\Delta t$, $snt = sin(n\Delta t)$, $cnt = cos(n\Delta t)$, and $u_i = f_i\Delta t$. Notice how the B matrix is the last three columns of A. This is since we are using an impulse model for the discretization of the inputs.

5.3 Unconstrained Problem

Since the solution of the trajectory optimization problem is the control history (control input at each time-step) and the resulting state trajectory (state at each time-step) for optimal rendezvous we must specify how long we want the spacecraft to take to rendezvous. This time to rendezvous is referred to as the time-horizon, usually denoted as T.

For the optimal rendezvous problem without state and input constraints, the spacecraft has some initial position and velocity which we want to bring to 0 in T time-steps, which means that the spacecraft is at the ISS with zero relative velocity. Additionally this trajectory must minimize some metric of control effort. We call this the constrained problem since we do not have any constraints on the control input or the state of the spacecraft. In the next section we will put an upper limit on our control input and we will constrain the spacecraft to approach the ISS in some approach cone.

This can be mathematically written as

Problem 1

$$\begin{aligned} \min_{\boldsymbol{x}_0,\ldots,\boldsymbol{x}_T,\boldsymbol{u}_0,\ldots,\boldsymbol{u}_{T-1}} \quad & \textstyle\sum_{k=0}^{T-1} \|\boldsymbol{u}_k\|_2^2 \\ \text{s.t. } \boldsymbol{x}_{k+1} &= A\boldsymbol{x}_k + B\boldsymbol{u}_k \\ \boldsymbol{x}(0) &= \boldsymbol{x}_0 \\ \boldsymbol{x}(\mathrm{T}) &= 0 \end{aligned}$$

where $\|\boldsymbol{u}_k\|_2^2$ is the square of the L^2-norm of the control input at timestep k and $\boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k$ are the discrete dynamics that we derived in the previous section. In essence, this optimization problem seeks to minimize the sum of the squares of the control inputs subject to the dynamics and initial and terminal boundary conditions on the state.

This does not look like the QP which we introduced in section 2, but it can be cannonicalized into the form

Problem 2

$$\begin{aligned} \min_{x} \quad & \tfrac{1}{2}x^T Q x + q^T x \\ \text{s.t.} \quad & Ax = b \\ & Gx \preceq h \end{aligned}$$

if we define the state, cost, and constraint matrices as follows:

$$x = \begin{bmatrix} \boldsymbol{u}_0 & \boldsymbol{x}_1 & \boldsymbol{u}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{u}_{T-1} \end{bmatrix}^T$$

$$Q = \begin{bmatrix} I & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & I & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & I \end{bmatrix}$$

$$A = \begin{bmatrix} B & -I & 0 & 0 & \cdots & 0 \\ 0 & A & B & -I & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & A & B \end{bmatrix}$$

$$b = \begin{bmatrix} -A\boldsymbol{x}_0 & 0 & 0 & 0 & \cdots & \boldsymbol{x}_T \end{bmatrix}^T$$

For this problem we do not have any inequality constraints.

For this example, we showed how to canonicalize problem 1. In practice we can directly feed problem 1 into canonicalizers like CVX or CVXPY and it will cannonicalize problem 1 into problem 2 for us and then solve problem 2 with off-the-shelf quadratic program solvers.

5.4 Adding State and Input Constraints

Problem 1 was introduced to show how to cannonicalize a simple trajectory optimization problem into a quadratic program. However, to make our problem more realistic, we have to add constraints that are not supported by a quadratic program such as quadratic input constraints and conic constraints.

We know that we have some upper thrust limit that or spacecraft's thrusters can produce. This is our quadratic input constraints. Further we will constrain the spacecraft's state so that it must stay within a cone whose apex is at the ISS and open up in the along track (y) direction. This is a very standard constraint for rendezvous. We can write this problem as the following.

Problem 3

$$\begin{gathered} \min_{\boldsymbol{x}_0,\ldots,\boldsymbol{x}_T,\boldsymbol{u}_0,\ldots,\boldsymbol{u}_{T-1}} \quad \textstyle\sum_{k=0}^{T-1} \|\boldsymbol{u}_k\|_2^2 \\ \text{s.t. } \boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k \\ \boldsymbol{u}_k^T\boldsymbol{u}_k \le u_{max}^2 \\ \|S\boldsymbol{x}_k\| + \boldsymbol{c}^T\boldsymbol{x}_k \le 0 \\ \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ \boldsymbol{x}(\mathrm{T}) = 0 \end{gathered}$$

Where u_{max} is the maximum thrust the spacecraft can produce,

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad c = \begin{bmatrix} 0 & -tan(\gamma) & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

and γ is the half-angle of the approach cone Due to our thrust limit and approach cone constraints, this optimization problem is now a Second Order Cone Program (SOCP). To solve it, we inputted Problem 3 into CVXPY and called the ECOS solver.

5.5 L^1 vs L^2 Penalties

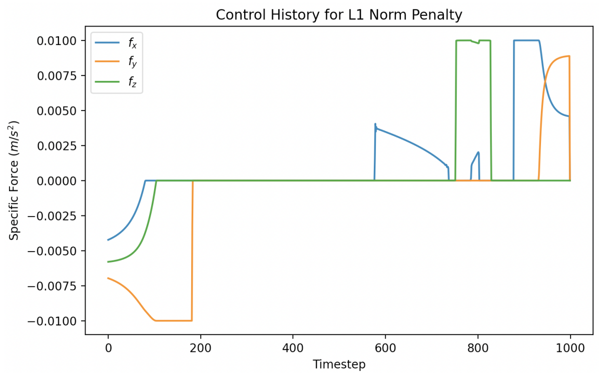
In the past few examples we used the sum of the squares of the L^2 norm to penalize our control input, however this isn't the most accurate metric of fuel consumption. In most chemical propulsion systems fuel consumption is linearly proportional to the thrust produced. If we assume that we have three orthogonal thruster, a more accurate metric for fuel consumption would be the sum of the L^1 norms of the control input. This problem is given below

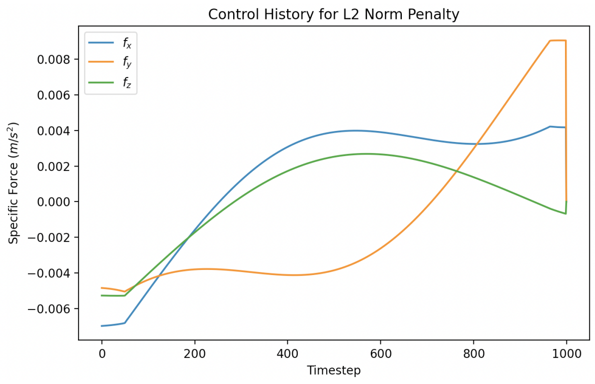
Problem 4

$$\begin{aligned} \min_{\boldsymbol{x}_0,\ldots,\boldsymbol{x}_T,\boldsymbol{u}_0,\ldots,\boldsymbol{u}_{T-1}} \quad & \textstyle\sum_{k=0}^{T-1} \|\boldsymbol{u}_k\|_1 \\ \text{s.t.} \ \boldsymbol{x}_{k+1} &= A\boldsymbol{x}_k + B\boldsymbol{u}_k \\ \boldsymbol{u}_k^T\boldsymbol{u}_k &\leq u_{max}^2 \\ \|S\boldsymbol{x}_k\| + \boldsymbol{c}^T\boldsymbol{x}_k &\leq 0 \\ \boldsymbol{x}(0) &= \boldsymbol{x}_0 \\ \boldsymbol{x}(\mathrm{T}) &= 0 \end{aligned}$$

By penalizing the sum of the L^1 norm, we encourage sparsity in our solutions. The thrusters will tend to be off most of the time, but when they are on, they apply a lot of thrust. On the other hand, penalizing the sum of the squares will encourage the thrusters to be on for the most part, but at a greatly reduced thrust, since large thrusts are penalized much heavier than in the L^1 case.

The code used to implement this optimal rendezvous can be found here





6 References

1. L. Blackmore, Autonomous precision landing of space rockets, The Bridge on Frontiers of Engineering, 4 (2016), pp. 15–20.

2. Stephen Boyd, L.V., Convex Optimization, 2009.

3. Vandenberghe L (2010) The cvxopt linear and quadratic cone program solvers. http://abel.ee.ucla.edu/cvxopt/documentation/coneprog.pdf, March 2010

4. Carlo, Jared Wensing, Patrick Katz, Benjamin Bledt, Gerardo Kim, Sangbae. (2018). Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control. 1-9. 10.1109/IROS.2018.8594448.

5. S. Feng, E. Whitman, X. Xinjilefu and C. G. Atkeson, "Optimization based full body control for the atlas robot," 2014 IEEE-RAS International Conference on Humanoid Robots, 2014, pp. 120-127, doi: 10.1109/HUMANOIDS.2014.7041347.

6. Mattingley, J., Boyd, S. CVXGEN: a code generator for embedded convex optimization. Optim Eng 13, 1–27 (2012). https://doi.org/10.1007/s11081-011-9176-9