Group Lasso Regularized Trajectory Optimization

Tarun Punnoose and Nathan Kau

We use the alternating direction method of multipliers (ADMM) to solve linear optimal control problems with group lasso regularization. We apply our solver to the spacecraft rendezvous problem, where sparsity in thruster firings is desired. We show that our ADMM-based solver better enforces control sparsity for the satellite while also being closer to bang-off-bang control. Additionally the ADMM-based solver is less computationally expensive than the CVX based version and can more readily be implemented on an embedded computer.

I. Introduction

Trajectory optimization is a powerful tool for controlling robots, spacecraft, or other vehicles along desired trajectories subject to various constraints. Specifically, trajectory optimization is formulated as the minimization of a cost function subject to the dynamics constraints and additional arbitrary state and control constraints. In this paper, we focus on a solver for systems with time-varying linear dynamics, convex control limits, and group lasso control regularization. The reason we focus on group lasso control regularization is that when each control input corresponds to a lasso group, the minimization of the cost function encourages control sparsity along the time dimension. When we add the control limit on top of the group lasso constraint, the controls tend to switch between zero and the maximum value. This property is particularly beneficial for applications in aerospace, where some types of thrusters can only be fully off or fully on.

We develop a solver based on the alternating direction method of multipliers to solve this particular control problem. It exploits the structure inherent in trajectory optimization problems, along with the closed form solutions for the group lasso regularization and control constraint. We find that the solver gives much higher solution accuracy than an off-the-shelf interior point solver like ECOS, and scales better when the problem size increases.

We apply our solver to the particular problem of satellite rendezvous, a problem where an active chase vehicle desires to catch up to a target vehicle according to the spacecraft dynamics, thruster constraints, and whatever fuel limitations it has. One example of this problem is the SpaceX Crew Dragon docking with the International Space Station. With certain assumptions that are generally good for these rendezvous problems, the dynamics can be linearized into what are known as the Clohessy-Wiltshire equations.

II. Prior Work

Le Cleac'h and Manchester previously developed an ADMM-based solver for trajectory optimization problems with an L1 control cost [1]. The L1 control cost encouraged sparsity along the time-dimension as well as along the control coordinate dimensions. The L1 cost on control also corresponds with the fuel or energy costs, and can also be used to construct minimum-time problems.

In contrast to the L1 control cost, we use the group lasso cost, which is a sum of L2 costs. Whereas the L1 cost induces coordinate-wise sparsity, the L2 cost on the control is coordinate-agnostic and imposes a spherically symmetric cost across the controls. This is better suited to some vehicles which have similar spherical symmetry in their thruster design. In particular, vehicles featuring one primary thruster may exert a force vector in any direction by re-orientating the spacecraft and then firing. For these types of spacecraft, the L1 norm would not be suitable since the control cost is spherically symmetric around the spacecraft, rather than particular per coordinate of control.

III. Problem Formulation

In this paper, we focus on time-varying linear dynamical systems with state x, control u, and initial state x_0. The objective function is composed of the a final cost l_N, a stage cost on the state l_k, and the group lasso cost on control u. The constraints to the optimization problem are the dynamics constraints on x and the constraint that control u lie in set C. The complete formulation is as follows:

$$\begin{aligned} \underset{X,U}{\text{minimize}} \quad & l_N(x_N) + \sum_{k=0}^{N-1} l_k(x_k) + \alpha \|u_k\|_2 \\ \text{subject to} \quad & x_{k+1} = A_k x_k + B_k u_k,\ k \in 0, 1, ..., N-1, \\ & u_k \in C,\ k \in 0, 1, ..., N-1 \end{aligned}$$

where the cost functions are assumed to quadratic functions of form:

$$\begin{aligned} l_N(x_N) &= \frac{1}{2} x_N^T Q_N x_N \\ l_k(x_k) &= \frac{1}{2} x_k^T Q_k x_k \end{aligned}$$

IV. Methodology

We used the alternative direction method of multipliers (ADMM) to solve the trajectory optimization problem. While it it possible to use an interior point method with an off-the-shelf solver like CVXPY to solve these types of optimization problems, we chose to use ADMM because it better exploits the structure of our problem to reach faster convergence. The performance of ADMM also scales better with as the number of optimization variables increases. The algorithm works by splitting the objective function and constraints into two or more parts, which are then iteratively solved as local subproblems as a function of a global parameter. This global parameter is updated at each iteration by the output of the local subproblems.

We split the prior trajectory optimization problem formulation into the following objective functions:

$$\begin{aligned} J_1(X, U) &= l_N(x_N) + \sum_{k=0}^{N-1} l_k(x_k) + \mathcal{I}_D(X, U) \\ J_2(U) &= \alpha \sum_{k=0}^{N-1} \|u_k\|_2 \\ J_3(U) &= \sum_{k=0}^{N-1} \mathcal{I}_C(u_k) \end{aligned}$$

where $\mathcal{I}_D$ represents an indicator function of the dynamics constraints and $\mathcal{I}_C$ represents and indicator function of the control constraints. We then use the ADMM group consensus algorithm in order to minimize this combined objective.

$$\begin{aligned} \underset{X,U}{\text{minimize}} \quad & J_1(X, U) + J_2(Y) + J_3(Z) \\ \text{subject to} \quad & U - W = 0, \\ & Y - W = 0, \\ & Z - W = 0, \end{aligned}$$

The augmented Lagrangian for this problem is then:

$$\begin{aligned} \mathcal{L}_\rho = J_1(X, U) + J_2(Y) + J_3(Z) + \sum_{k=0}^{N-1} \lambda_k^T (u_k - w_k) + \frac{\rho}{2} \|u_k - w_k\|_2^2 + \nu_k^T (y_k - w_k) + \frac{\rho}{2} \|y_k - w_k\|_2^2 \\ + \mu_k^T (z_k - w_k) + \frac{\rho}{2} \|z_k - w_k\|_2^2 \end{aligned}$$

Each iteration of ADMM is one pass of a Gauss-Siedel method minimizing the augmented Lagrangian term above. This leads to the following updates for each variable:

$$\begin{aligned}
(X^{(i+1)}, U^{(i+1)}) &= \underset{X,U}{\operatorname{argmin}}\ J_1(X) + \sum_{k=0}^{N-1} \frac{\rho}{2}||u_k - w_k^{(i)} + \frac{1}{\rho}\lambda_k^{(i)}||_2^2 \\
Y^{(i+1)} &= \underset{Y}{\operatorname{argmin}} \sum_{k=0}^{N-1} \alpha||y_k||_2 + \frac{\rho}{2}||y_k - w_k^{(i)} + \frac{1}{\rho}\nu_k^{(i)}||_2^2 \\
Z^{(i+1)} &= \underset{Z}{\operatorname{argmin}} \sum_{k=0}^{N-1} \mathcal{I}_C(z_k) + \frac{\rho}{2}||z_k - w_k^{(i)} + \frac{1}{\rho}\mu_k^{(i)}||_2^2 \\
W^{(i+1)} &= \frac{1}{3}(U^{(i+1)} + Y^{(i+1)} + Z^{(i+1)}) \\
\lambda_k^{(i+1)} &= \lambda_k^{(i)} + \rho(u_k^{(i+1)} - w_k^{(i+1)}),\ k \in \{0, ..., N-1\} \\
\nu_k^{(i+1)} &= \nu_k^{(i)} + \rho(y_k^{(i+1)} - w_k^{(i+1)}),\ k = 0, ..., N-1 \\
\mu_k^{(i+1)} &= \mu_k^{(i)} + \rho(z_k^{(i+1)} - w_k^{(i+1)}),\ k = 0, ..., N-1
\end{aligned}$$

Each of the updates for X, U, Y, Z can be performed in closed form. This results in the ADMM algorithm being a relatively simple and easy to implement algorithm on embedded hardware. Only a linear algebra library like Eigen would be necessary to implement this on real hardware.

A. LQR Update

The ADMM iterative update for (X, U) is the minimization of a quadratic cost subject to linear time varying dynamics constraints. This can be solved using a closed form dynamic programming solution, commonly known as LQR. By using this closed form LQR solution, the dynamics constraints are implicitly satisfied. The addition of the augmented Lagrangian term simply acts as a cost regularizer relative to a reference control $\bar{u}_k = w_k^{(i)} - \frac{1}{\rho}\lambda_k^{(i)}$. The LQR update equations can be summarized as:

$$\begin{aligned}
J_k(x) &= \frac{1}{2}x^T V_k x + v_k^T x \\
K_k &= -(R_k + B_k^T V_{k+1} B_k)^{-1} B_k^T V_{k+1} A_k \\
d_k &= (R_k + B_k^T V_{k+1} B_k)^{-1}(R_k(w_k - \frac{1}{\rho}\lambda_k) - B_k^T v_{k+1}) \\
V_k &= Q_k + K_k^T R_k K_k + (A_k + B_k K_k)^T V_{k+1}(A_k + B_k K_k) \\
v_k &= K_k^T R_k^T (d_k - \bar{u}) + (A_k + B_k K_k)^T V_{k+1} B_k d_k + (A_k + B_k K_k)^T v_{k+1} \\
u_k &= K_k x + d_k
\end{aligned}$$

Because we know that $V_N = Q_f$ and $v_N = 0$, we can iterate backwards using the above equations to find K_k and d_k throughout the trajectory. Then a forward pass through the linear dynamics equations from initial value x_0 can be performed through the dynamics equations:

$$\begin{aligned}
u_k &= K_k x_k + d_k \\
x_{k+1} &= A_k x_k + B_k u_k
\end{aligned}$$

Although a full derivation of the LQR update is outside the scope of this paper, there are many derivations can be found in the literature [2] [3]. By using the LQR update here, the structure of the dynamics problem is taken into account and can be solved much faster than through an interior point method.

B. L2 Regularization

The definition for a proximal function is:

$$\operatorname{prox}_{\psi f} = \underset{x}{\operatorname{argmin}}\ f(x) + \frac{1}{2\psi}||x - v||_2^2$$

Leveraging this definition of a proximal function, the iterative ADDM update for each y_k is:

$$\begin{aligned} y_k^{(i+1)} &= \underset{y}{\operatorname{argmin}}\ \alpha||y_k||_2 + \frac{\rho}{2}||y_k - w_k^{(i)} + \frac{1}{\rho}v_k||_2^2 \\ &= \operatorname{prox}_{\psi f}(w_k - \frac{1}{\rho}v_k),\ \text{where } \psi = \frac{\alpha}{\rho} \text{ and } f(x) = \|x\|_2 \\ &= \left(1 - \frac{\alpha}{\rho\|w_k - \frac{1}{\rho}v_k\|_2}\right)_+ \left(w_k - \frac{1}{\rho}v_k\right) \end{aligned}$$

By leveraging this closed form solution [4], the iterative update is very simple. By decoupling this update from the dynamics update, the structure of the L2 norm is automatically taken into account by the proximal function.

C. Control Constraints

The update for the control constraint variable Z can simply be written as a projection operator:

$$\begin{aligned} z_k^{(i+1)} &= \underset{z}{\operatorname{argmin}}\ \mathcal{I}_C(z_k) + \frac{\rho}{2}||z_k - w_k^{(i)} + \frac{1}{\rho}\mu_k||_2^2,\ k = 0, ..., N-1 \\ z_k^{(i+1)} &= \Pi_C(w_k^{(i)} - \frac{1}{\rho}\mu_k) \end{aligned}$$

In most cases the set of feasible controls C is a norm ball, so this projection is easily solved in closed form.

V. Results

We apply our ADMM-based trajectory optimization method to the problem of satellite rendezvous. Satellite rendezvous involves an actively-controlled chase vehicle approaching an uncontrolled target vehicle. One recent example is the SpaceX Crew Dragon approaching the International Space Station. When two spacecraft in circular orbits are relatively close to each other, the dynamics can be linearized and form the commonly known Clohessy-Wiltshire (CW) equations. In the CW equations, shown below, the state vector is composed of the position vector p and the velocity vector $\dot{p}$ of the chase vehicle relative to the target vehicle. The parameter n denotes the mean motion of the target vehicle, while the parameter m is the chaser vehicle's mass. The controls are u_1, u_2, and u_3, which correspond to the three components of the chaser vehicle's thrust vector.

$$x = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \dot{p}_1 \\ \dot{p}_2 \\ \dot{p}_3 \end{bmatrix} \quad \dot{x} = \begin{bmatrix} \dot{p}_1 \\ \dot{p}_2 \\ \dot{p}_3 \\ 3n^2 p_1 + 2np_2 + \frac{u_1}{m} \\ -2n\dot{p}_1 + \frac{u_2}{m} \\ -n^2 p_3 + \frac{u_3}{m} \end{bmatrix}$$

By applying our problem formulation to the satellite rendezvous problem, we can induce sparse controls that are either zero or at the control limit. This property is convenient, and even necessary, for compatibility with the vehicle's thrusters, which are designed to be fully off or fully on at any given point in time. The group lasso regularization also encourages the trajectory to minimize the fuel used to reach the target vehicle.

We compared the results of our ADMM-based solver to that of CVX running the ECOS solver. Our primary finding was that our ADMM solver generated sparse control commands while CVX did not generate sparsity, primarily because the ECOS backend failed to converge after more than one hundred iterations. In Figure 1, we see that the norm of the thruster commands generated by ADMM are much more sparse than those computed by CVX with the ECOS backend. Indeed, the ECOS-computed controls are never zero, indicating that the ECOS solver could not properly account for the non-smooth group lasso cost. In Figure 2, we see a detailed look at the individual components of the control generated by both our ADMM solver and by the ECOS solver. We see that the control generated by our ADMM solver takes a bang-off-bang approach to reach the target objective, that is, that the thruster first fires in one direction, the turns the thruster off, then at about 1000s, switches direction to fire in another direction. In comparison, we see that the

ECOS-computed components of the control are smoothly varying and have no sparsity. Finally, we can see in Figure 3 how for the ADMM-generated trajectory, the chase vehicle quickly reaches the target vehicle, but for the ECOS trajectory, the vehicle deviates further from the origin and only reaches the target position only at the end of the planning horizon.

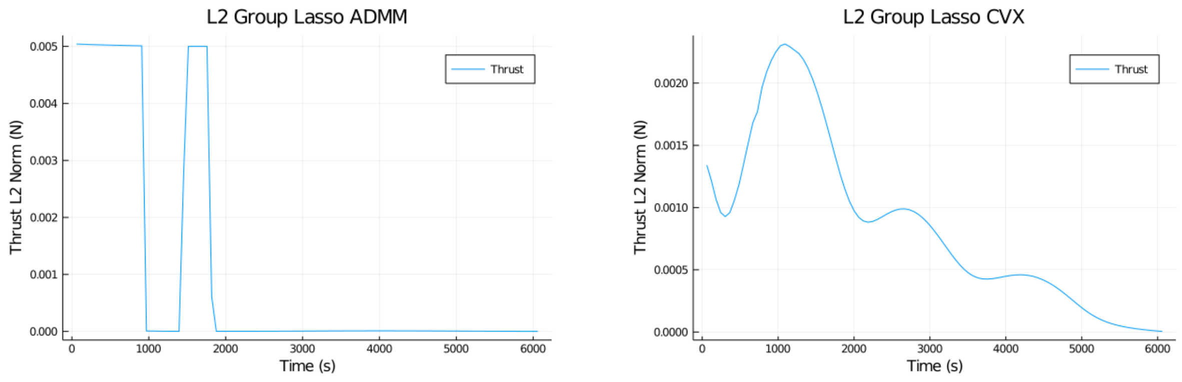

Fig. 1 Comparison between the sparsity of the thrust commands generated by our ADMM solver and those generated by CVX + ECOS. While our ADMM solver only used the thruster briefly in the beginning of the trajectory and then turned the thrusters off, the CVX + ECOS solution used the thrusters for the entire trajectory.

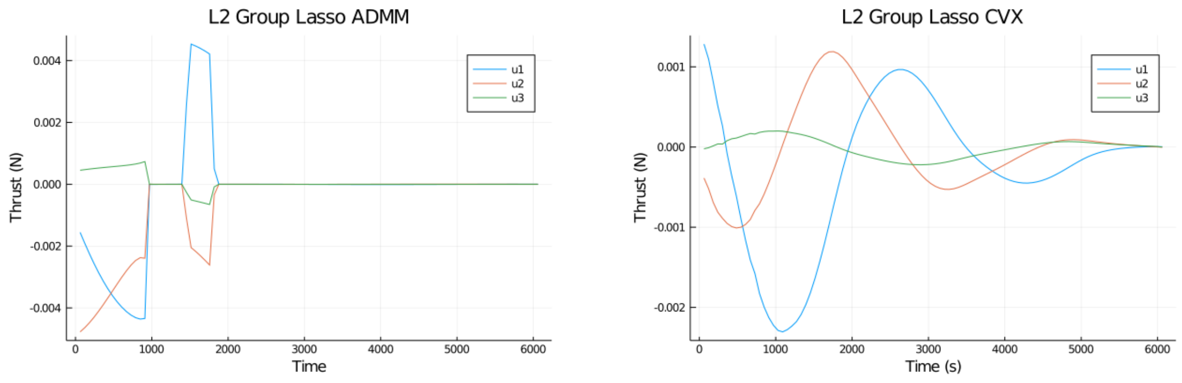

Fig. 2 Comparison between the components of the thruster commands generated by our ADMM solver and those generated by CVX + ECOS. The key feature of this comparison is that the ADMM solver fires thrust in one direction for the first 1000s, then turns off the thruster, and then reverses thrust for the next 500s to reach the final target vehicle. On the other hand, the CVX + ECOS solution uses thrust the entire time.

VI. Conclusion

We developed an ADMM-based solver for trajectory optimization problems with linear time-varying dynamics and group lasso regularization. The use of a group lasso regularizer for the controls allows for spherically symmetric regularization while still enforcing time wise sparsity. We apply our solver to a satellite rendezvous problem with the Clohessy-Wiltshire dynamics. We find that our implementation results better induces sparsity in the control than the Convex.jl implementation and scales better in problem size. Our ADMM implementation is also much lighter weight implementation which makes it more easily to implement on an embedded satellite.

All code for this project can be found at `https://github.com/tpunnoose/L2TrajectoryOptimization`.

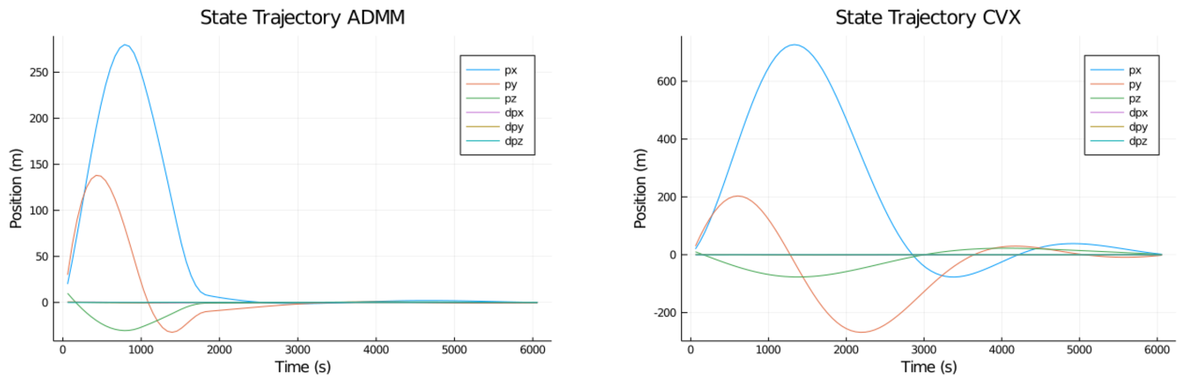

Fig. 3 Comparison between the state trajectories generated by CVX + ECOS and by our ADMM solver. The ADMM trajectory quickly reaches the target position, while in contrast, the CVX + ECOS solution takes a meandering approach to the final target.

References

[1] Le Cleac'h, S., and Manchester, Z., "Fast solution of optimal control problems with l1 cost," 2019.

[2] Li, W., and Todorov, E., "Iterative linear quadratic regulator design for nonlinear biological movement systems." *ICINCO (1)*, 2004, pp. 222–229.

[3] Howell, T. A., Jackson, B. E., and Manchester, Z., "ALTRO: A fast solver for constrained trajectory optimization," *2019 IEEE International Conference on Intelligent Robots and Systems, IEEE*, 2019.

[4] Parikh, N., and Boyd, S., "Proximal algorithms," *Foundations and Trends in optimization*, Vol. 1, No. 3, 2014, pp. 127–239.