Investigation on Different Optimization Solvers for Quadratic Programming and Implementation on MPC

Xiaozhu Fang [a], Heejin Kim [a], Zhu Wang [a]

[a] *Department of Mechanical Engineering, University of Michigan, 2350 Hayward, Ann Arbor, MI 48109*

Abstract

In this paper, six different algorithms to solve LQ-MPC problem are compared in computational aspect, namely, Fast Interior Point Method(FIP), Proximally Stabilized FB Method (FBstab), Alternating Direction Method of Multipliers(ADMM), Gradient Projection Method (GPD), and Accelerated Gradient Projection Method (GPAD). For each method, we outline the algorithm and state any convergence or time complexity properties. Particularly for FIP, we implement several fast computation methods cited from Boyd[1] which exploit the special structure of MPC. Then, we apply them to a tracking control problem of a satellite system with input constraints. Results show that FIP is the most competitive among the six in terms of computation, but with the sacrifice of controller performance.

1 Introduction

Convex Optimization recently has attracted lots of interest nowadays because of its application on Model Predictive Control (MPC) and Support Vector Machine (SVM). The standard form of Convex Optimization problem is following.

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, i = 1, \cdots, m \\ & h_i(x) = 0, i = 1, \cdots, p \end{aligned} \tag{1}$$

$f(x)$ should be convex to get the unique solution. Besides, the function g_i are also convex functions while h_i are affine. In such cases, the intersection of constraints results in a convex set. The approach to solve Convex Optimization is very sophisticated.
Quadratic Programming (QP) is a method used to solve a special convex problem, which gives a simpler form as follows:

$$\begin{aligned} \min_{z} \quad & \frac{1}{2} z^T H z + g^T z \\ \text{subject to} \quad & Pz \leq h \\ & Cz = b \end{aligned} \tag{2}$$

H is a real symmetric matrix, which is also positive definitive. P and C are both matrices that incorporate all the constraints.
If Eqn. 10 was free of constraints, the analytical solution could be directly given by the matrix's inverse. Alternatively, the numerical Gradient Descent (GD) is applicable if the inverse operation is intractable. Furthermore, the higher order methods such as the Newton method and Runge–Kutta methods can be used to accelerate the convergence. The only thing needs to pay attention is the step size, whose magnitude affects the convergence. In other words, Lipschitz continuity of the gradient should be guaranteed so that the Hessian of the objective function, H, is used. The optimal step size of classical GD is

$$\alpha_* = \frac{2}{\sigma_1(H) + \sigma_N(H)} \tag{3}$$

where σ_1 and σ_N represent the maximum and minimum singular values. Based on that, the various step size methods are proposed. In practice, however, singular value is hard to calculate when the matrix is large. Then the practical step size is

$$\alpha = \frac{2}{\| H \|_F}, \text{or } \frac{2}{\| H \|_1} \tag{4}$$

Furthermore, α does not need to be a constant. There are two extensions to improve that. One is the precondition to consider the gradient in each direction. Another is the various step size, that is, the step size varies with time, adjusting to each step. Nesterov's Accelerated Gradient Descent is typical method[2], and we implement it in

⋆ This is a project report in AEROSP 740 Model Predictive Control instructed by Professor Ilya Kolmanovsky in University of Michigan.

Email addresses: fangxz@umich.edu (Xiaozhu Fang), heejink@umich.edu (Heejin Kim), wangzhu@umich.edu (Zhu Wang).

Implementation. Another example is called Optimized Gradient Method (OGM)[3]. This is the general problem existing in convex optimization.
If the constraint is only the equality, the problem can be modified by Lagrange multiplies, which introduces more variables. However, the approaches to solve that makes no difference.
If the constraints include the inequality, the problem turns to be completely different. In QP, its strong duality provides another representation of the problem.

$$\begin{aligned} &\min_x \max_{\lambda \geq 0, v} \quad f(x) + \sum_i \lambda_i g_i(x) + \sum_i v_i h_i(x) \\ =\ &\max_{\lambda \geq 0, v} \min_x \quad f(x) + \sum_i \lambda_i g_i(x) + \sum_i v_i h_i(x) \end{aligned}$$

The primal form denotes the original problem while the dual form denotes the augmented ones. Although the dual form has more variables, its nonlinear constraints become much more desirable: it gets rid of the ill-condition matrix P.
For both forms, Karush–Kuhn–Tucker condition is the basic approach to find the solution: traverse all the combination of constraints to find the active ones, which is related to the active set method. However, this method requires too much computation. Interior point method uses much trickier method to solve that. Furthermore, more advanced methods such as Non-Negative Least Square are developed to enhance the performance.
In this paper, we will investigate several typical methods. They include:

- the primal-dual interior point(PDIP),
- the Fast Interior Point Method(FIP),
- the Proximally Stabilized FB Method (FBstab),
- Alternating Direction Method of Multipliers(ADMM)
- Gradient Projection Method (GPD)
- Accelerated Gradient Projection Method (GPAD).

IP, FBstab, and ADMM belong to indirect methods, which introduce the additional penalty or regularization to approaching the exact result. The methods such as GPD, GPAD as well as active sets belong to direct methods.

2 Methods

2.1 Interior Point and Faster Interior Point

The original Quadratic Programming Eqn. 2 under both constraint of equations and inequations can be transformed into following Eqn. 5[1].

$$\min_z \quad z^T H z + g^T z + \kappa\phi(z) \tag{5}$$
$$\text{subject to} \quad Cz = b \tag{6}$$

where $\phi(z) = \sum_{i=1}^{k} -\log(h_i - p_i^T z)$; and $p_1^T, \ldots, p_k^T$ are rows of P. $\phi(z)$ has meaning when $h_i \geqslant p_i z$ holds for all $i = 1, \ldots, k$. We define $\phi(z) = \infty$ when $z < 0$ to avoid taking logarithm of negative value.
Inequality constraints are not shown in the Eqn. 5, but this information of constraint $Pz \leqslant h$ is implicitly held by addition barrier function $\kappa\phi(z)$.

2.1.1 Primal-dual Method

To solve Eqn. 5, we use primal-dual method for find the optimal point[4]. Dual variable ν is associated with constraint $Cz = b$. The residual of two variables are:

$$r_d = 2Hz + g + \kappa P^T d + C^T \nu \to 0 \tag{7}$$
$$r_p = Cz - b \to 0 \tag{8}$$

where $d_i = \frac{1}{h_i - p_i^T z}$, and p_i^T denotes the ith row of P.
Using infeasible start Newton method[1], we determine the steps of Δz and $\Delta\nu$ by:

$$\begin{bmatrix} 2H + \kappa P^T \mathbf{diag}(d)^2 P & \Delta z \\ C & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta\nu \end{bmatrix} = - \begin{bmatrix} r_d \\ r_p \end{bmatrix} \tag{9}$$

And then, update z, ν by Δz and $\Delta\nu$ to find the optimal result.
If updated z is not feasible when $Pz \leqslant h$ doesn't hold, Δz and $\Delta\nu$ will be reduced until the new z becomes in feasible set. Taking inverse of matrix in Eqn. 9 will take $O(T^3(m+n)^3)$, where T is the horizon length, m is number of input and n is number of state variables.

2.1.2 Faster Method using Schur Complement

Step of Δz and $\Delta\nu$ are calculated by doing inverse of matrix in Eqn.9, which takes lots of time. A faster way is to do Schur complement[1]:

(1) $Y = C\Phi^{-1}C^T$ and $\beta = -r_p + C\Phi^{-1}r_d$
(2) Calculate $\Delta\nu$ by $Y\Delta\nu = -\beta$
(3) Calculate Δz by $\Phi\Delta z = r_d - C^T\Delta\nu$

where Φ is a blocked diagonal matrix with square matrix of dimension $n \times n$, $m \times m$ appears every the other element. We can take inverse of the each block on the diagonal instead of take inverse of the entire matrix. It will take $O(T(n^3 + m^3))$
In addition, $Y = C\Phi^{-1}C^T$ is of Cholesky factorization, and Y block tridiagonal with block size of $n \times n$. Block tridiagonal matrix has closed form to calcualte the inverse, which cost $O(Tn^3)$ time[5].
In addition to improve the performance in computation time, we keep κ which may scarify control performance[1].

2.2 Alternating Direction Method of Multipliers

Alternating direction method of multipliers is actually designed for more general cases but also available in QP problem. It decomposes QP problem into two blocks and solve them alternatively. It takes the advantages of dual form and multipliers. The convergence of two blocks ADMM was proven by Boyd[4].Take typical form QP without equality constraint as the example.

$$\min_{z} \quad \frac{1}{2}z^T Hz + g^T z + \mathcal{L}_+(x) \tag{10}$$
$$\text{subject to} \quad Pz - h + x = 0 \tag{11}$$

where $\mathcal{L}_+(x)$ is the indicator that is infinite large when it is negative. Then we assent the equality into the objective function by Lagrange form.

$$\min_{x,z,u} \quad \frac{1}{2}z^T Hz + g^T z + \mathcal{L}_+(x) + \frac{\rho}{2} \parallel Pz - h + x + u \parallel_2^2 \tag{12}$$

ADMM iteration alternatively update these three variables, converging to the final result. The detailed updating equation of ADMMM are following.

$$z^{k+1} = -(H + \rho P^T P)^{-1}[g + \rho P^T (x^k + u^k - h)] \tag{13}$$
$$x^{K+1} = \max\{0, -Pz^{k+1} - u^k + h\} \tag{14}$$
$$u^{k+1} = u^k + Pz^{k+1} - h + x^{k+1} \tag{15}$$

Besides, the over-relaxation ADMM was also addressed to improve the convergence[6].

2.3 FBstab

Recently recovering Nonnegative Least Squares (NNLS) by QP has attracted significant attention[7]. Given the standard formula of QP, Bemprad transformed H by the Cholesky factorization, $H = \mathcal{L}'\mathbb{L}$. Then he defined

$$M = P\mathcal{L}^{-1}, \quad d = h + PH^{-1}g \tag{16}$$

Then the NNLS problem is formulated

$$\min_{y} \quad \frac{1}{2} \parallel \begin{bmatrix} -M' \\ -d' \end{bmatrix} y - \begin{bmatrix} 0 \\ \gamma \end{bmatrix} \parallel_2^2 \tag{17}$$
$$\text{subject to} \quad y \geq 0 \tag{18}$$

where γ is a positive scalar. It is set to 1 at fist but can be adjusted for a better numerical result.
Let y^* be the optimal result in Eqn.17. We can retrieve the original QP solution by

$$z^* = -H^{-1}(c + \frac{1}{\gamma + d'y^*}Py^*) \tag{19}$$

The solution is feasible as long as the minimum of the objective function in Eqn.17 is nonzero.
The proximally stabilized Fischer-Burmeister method (FBstab) was an extension on NNLS methods. It adds dual form for one thing and applies regularized and smoothed Fischer–Burmeister (FBRS) method for another. Since dual form has already been explained in Introduction, we talk more about FBRS here.
FBRS can be regarded as another kind of proximal function to improve the convergence of Newton's method. The smoothed Fischer–Burmeister function is

$$\phi(a, b)_\epsilon = a + b - \sqrt{a^2 + b^2 + \epsilon} \tag{20}$$

when $\epsilon = 0$, it becomes the regular Fischer–Burmeister function. It has the properties

$$\phi_\epsilon(a, b) = 0 \iff a \geq 0, b \geq 0, \sqrt{2ab} = \epsilon \tag{21}$$

In consideration of QP problem, we have

$$F_\epsilon(z) = \begin{bmatrix} \nabla_z L(z, v) \\ \phi_\epsilon(v, y) \end{bmatrix} \tag{22}$$

$$z_{k+1} = z_k - t_k \nabla_z F_{\epsilon_k}(x_k) F_{\epsilon_k}(z_k) \tag{23}$$

where t_k is the step length for convergence. Note $\nabla_z F_{\epsilon_k}(x_k)$ is likely to be ill-conditioned so that we need regularization term added to $\phi_\epsilon(v, y)$. The mathematical details of Fbstab can refer to the original papers[8][9].

2.4 Accelerated Dual Gradient Projection Method (GPAD)

The last two algorithms used for LQ-MPC are called Accelerated Dual Gradient Projection Method (GPAD) and Generalized Dual Gradient Projection Method (GPD), the former being originated from the latter. At each time step, GPD updates the state iteratively using step size λ_k:

$$z^{k+1} = z^k - \lambda_k \nabla f(z^k) \tag{24}$$

It is widely used in constrained QP because it guarantees convergence to optimum under certain constant step size conditions:

$$\lambda_k \leq \frac{1}{L} \tag{25}$$

where $L > 0$ is an upper bound of maximum eigenvalue of Hessian matrix.
GPD spends $O(\log \frac{1}{\epsilon})$ iterations to obtain ϵ-near optimum solution for strictly convex problems; however, it requires $O(\frac{1}{\epsilon})$ iterations for those that are not strictly convex. Therefore, Nesterov[2] has proposed GPAD as

an alternate version of GPD by taking into account previous gradient direction in each iteration, thereby creating momentum while converging:

$$\begin{aligned} \bar{z}^{k+1} &= z^k - \lambda_k \nabla f(z^k) \\ z^{k+1} &= (1-\gamma_k)\bar{z}^{k+1} + \gamma_k \bar{z}^k \end{aligned} \tag{26}$$

Here, the closer to 1 γ_k is, the more aggressive the momentum will become. In this paper, γ_k was iteratively chosen such that:

$$\begin{aligned} \gamma_k &= \frac{1-t_k}{t_{k+1}} \\ t_{k+1} &= \frac{1}{2}(1+\sqrt{1+4t_k^2}) \end{aligned} \tag{27}$$

3 Implementation

3.1 Problem formulation

Since the focus of this paper is the optimizer, we use the satellite system model from the homework problem. The state space representation of the model shows as follows.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \frac{1}{\cos(\theta)} \begin{bmatrix} \cos(\theta) & \sin(\phi)\sin(\theta) & \cos(\phi)\sin(\theta) \\ 0 & \cos(\phi)\cos(\theta) & -\sin(\phi)\cos(\theta) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}$$

$$\begin{bmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{\omega}_3 \end{bmatrix} = \begin{bmatrix} ((J_2-J_3)\omega_2\omega_3/J_1 + M_1/J_1 \\ ((J_3-J_1)\omega_3\omega_1/J_2 + M_2/J_2 \\ ((J_1-J_2)\omega_1\omega_2/J_3 + M_3/J_3 \end{bmatrix}$$

where ϕ, θ and ψ represents the spacecraft's roll, pitch and yaw angles; ω_1,ω_2 and ω_3 are spacecraft angular velocity vector in the body fixed frame; J_1,J_2 and J_3 are principal moments of inertia. M_1, M_2 and M_3 are the control moments, which are the input of the state space model. In our case, we give $J_1 = 120$,$J_2 = 100$ and $J_3 = 80$. We introduce the simple constraints on input, that is,

$$-0.1 \leqslant M_i \leqslant 0.1, \quad i = 1,2,3 \tag{28}$$

We linearize the stat space at the origin to simplify the problem. The state space equations becomes those.

$$\begin{bmatrix} \delta\dot{\phi} \\ \delta\dot{\theta} \\ \delta\dot{\psi} \\ \delta\dot{\omega}_1 \\ \delta\dot{\omega}_2 \\ \delta\dot{\omega}_3 \end{bmatrix} = \begin{bmatrix} 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&1 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \end{bmatrix} \begin{bmatrix} \delta\phi \\ \delta\theta \\ \delta\psi \\ \delta\omega_1 \\ \delta\omega_2 \\ \delta\omega_3 \end{bmatrix} + \begin{bmatrix} 0&0&0 \\ 0&0&0 \\ 0&0&0 \\ \frac{1}{J_1}&0&0 \\ 0&\frac{1}{J_2}&0 \\ 0&0&\frac{1}{J_3} \end{bmatrix} \begin{bmatrix} \delta M_1 \\ \delta M_2 \\ \delta M_3 \end{bmatrix} \tag{29}$$

$$Q = \begin{bmatrix} 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&0.01&0&0 \\ 0&0&0&0&0.01&0 \\ 0&0&0&0&0&0.01 \end{bmatrix}, \; R = \begin{bmatrix} 0.01&0&0 \\ 0&0.01&0 \\ 0&0&0.01 \end{bmatrix} \tag{30}$$

Initial state is

$$X_0 = [-0.4, -0.8, 1.2, -0.02, -0.02, 0.02]^T \tag{31}$$

We need to clarify the terminology to avoid the confusion. In the following section,

> **Time step**: for each time step, MPC runs one time and get the result. We choose the first input and run the simulation. In our case, time step is [0:2:100]
> **Horizon**: horizon indicates the prediction steps for each optimization, which affects the dimension of H and number of constraints. We vary horizon in $[10 : 2 : 100]$.
> **Iteration**: Iteration n means the steps for each optimization method to reach the converges. In fact, its definition is ambiguous because some methods such as FBstab and inner-loop iteration. Hence, we abandon this term in our comparison.

The objective of this implementation is to compare the computational time under two criteria below:

- Computation time required for each time step $t = 0, 2, ..., 200$ $[s]$ at horizon length $T = 20$ $[s]$
- Computation time required for each horizon length $T = 10, 12, 14, ..., 100$ $[s]$ at initial time step $t = 0$ $[s]$

The first criteria reflects the influence of inital state on influence and the second criteria reflects the influence of the problem's dimension.

3.2 Setup

In order to set a fair comparison, we keep optimization parameters or strategies as same as possible for all six methods. According to our practices, some trivial changes will significantly affects the result. For instance, ADMM code set optimal step size $\alpha = \frac{2}{\sigma_1^2+\sigma_n^2}$ while GPD choose the simple $\alpha = \frac{1}{||A||_2^2}$. Theoretically it makes little difference, but in practice, the function svd() would cost lots of computational time. We list the all the conditions we notice for a better comparison.

- **Termination criteria**: step tolerance = 10^{-8}, i.e., terminate when:

$$\frac{||x^{k+1} - x^k||}{1 + ||x^k||} \leq 10^{-8} \tag{32}$$

Since we have dual variables, x here is the stacked vector includes primal variables and dual variables.

- **Step size**: we use step size $\lambda_k = \frac{1}{L}$, where L is the approximate maximum singular value, found by 'max(svd(P))' command in MATLAB.
- **Warm start**: we use the warm start in all approaches so that the later time step would be accelerated. However, the first time step can be regarded as cold start because we begin with zero.

For the codes, we refer to the ADMM method from github.com/danielrherber/admm-qp and FBstab method form the ADMM method from github.com/dliao mcp/fbstab-matlab. PDIP is just from quadprog function. As for GPD, GPAD and FIP, we written them by ourselves. However, FIP we written cannot compete with the commercial code `quadprog` because of the unprofessional coding styles. To be fair, we also written a home-made DPIP only used to compare with FIP. We show these result at the last of this section.

3.3 Result: Control law of MPC

Although we use different optimization methods, the result of MPC is similar. Fig.1 shows the result given by FBstab including the output and input curves.It is also the same as the result we calculated by MPC3 or Hybrid MPC in homework 2. All the methods did the same result of MPC so we don't put all the figures.

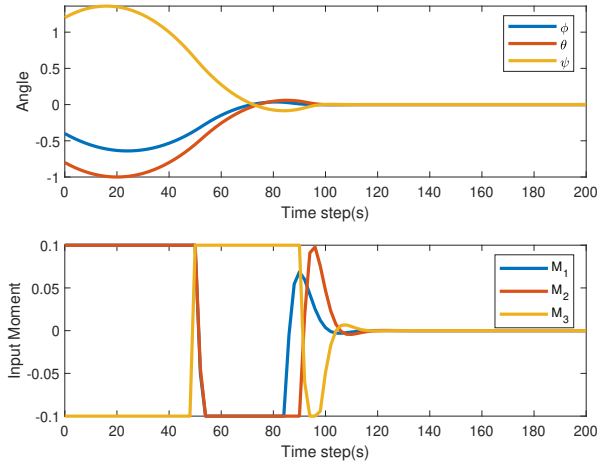

Fig. 1. Dynamcis and control law of MPC calculated by FBstab method.

3.4 Result: Computational time

The first comparison is regarding the time steps. As we know, optimization methods highly depend on the initial state x_0. If the initial state is far from the optimal point, it costs more time to reach the convergence. In this case, the computational time rises up.
For the first time step in each method, we start from original point, that is, it is cold start. For the following steps, the dual variables in last step is inherited. Hence, it is warm start, resulting in less computational time. The result of computational time over time steps shows in Fig.2 Fig.2 illustrates the features of each method.

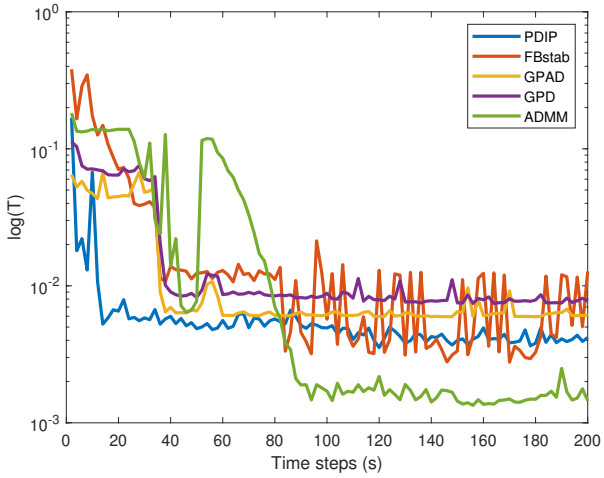

Fig. 2. Computational time (taken log) versus time steps of MPC for different methods.

As we expected, the computational time dramatically descent at the beginning due to the warm start and keep at a specific level. PDIP is the best one that outperforms others. The most obvious advantage of DPIP is its significantly decreasing after the first step. Afterwards, other methods are accelerateda and stablized to a certian value with more or less ossicilation. As we can see, ADMM in Fig.2 is not as smooth as other methods. After 100s, we can see the dynmacis totally decay to zero in Fig.1. The comparision after 100s does not make sense.
All curves decrease over time but such descending phenomenon is not only due to the warm start but also to the initialization of the program. The first search to the function costs more time due to the mechanism of computers.
The performance of QP solvers are not only affected by the initial point, but also the horizon. The influence of horizon shows in Fig.3 and Fig.4. The result shows that PDIP outperforms all other methods, both in cold start and warm start. FBstab has the medium performance. As for ADMM, the time is little for the small horizon but the time dramitically rises up over horizon. This drawback of ADMM does not show in Fig.2.
What is interesting is the time of GPAD and GPD significantly slide when horizon is 50. We cannot give a convincing explaining for this weird phenomenon, but we guess MATLAB automatically use transfer H to sparse matrix when the dimension is high. However, we observe GPAD and GPD are still increasing, so these

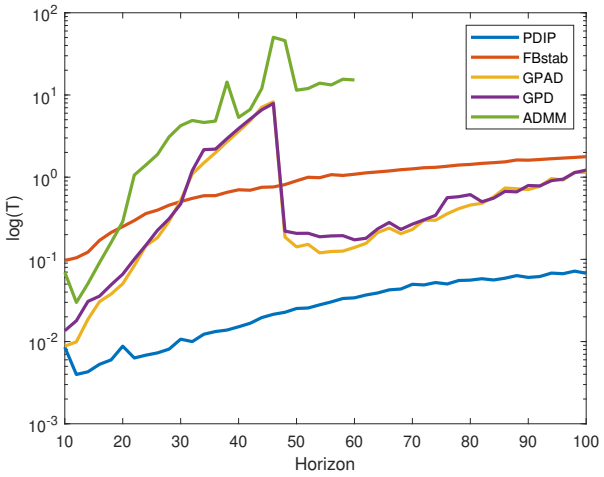

Fig. 3. Computational time(taken log) versus Horizon for the first time step(cold start).

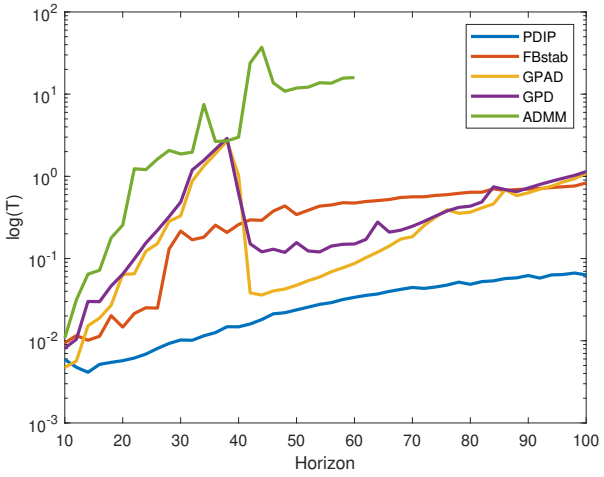

Fig. 4. Computational time(taken log) versus Horizon for the fifth time step(warm start).

methods are not suitable for high dimension problem. The matrix H in this case is well-conditioning, but ill-conditioning matrix is supposed to affect the result. We tried the ill-condition but it actually makes no difference.

3.5 Result: Compare FIP and PDIP

Using faster interior point, we can have following input to the satellite to be stable:

The performance of Faster Interior Point is comparable with the standard PDIP. Because we keep constant κ for barrier function, optimizer cannot find the exact best input when it is close to constraints boundary.
Because of the memory management, variable allocation and other unknown coding techniques, our program of Faster Interior Point is not comparable with the `quadprog.m` in MATLAB. We are comparing the speed of program using PDIP and FIP coded by us. The time for computing 20 iterations in each sampling loop of different horizons is shown in Table 1.

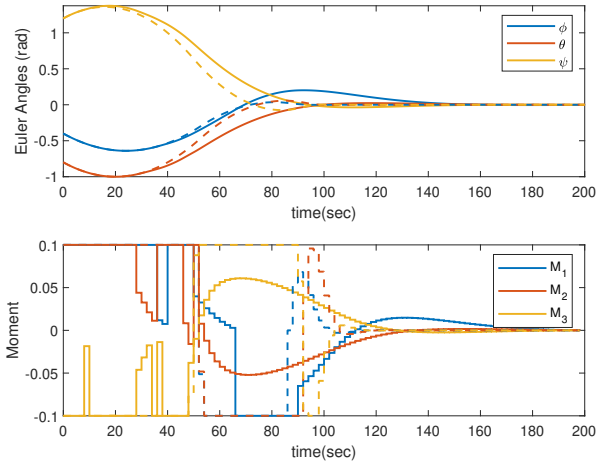

Fig. 5. Euler angles and input moment of satellite using faster interior point (solid line) and `quadprog.m` (dash line)

Horizon T	Faster IP (s)	PDIP (s)	`quadprog.m` (s)
15	0.6550	0.8224	1.1708
20	1.0192	1.3944	1.5191
30	3.0008	4.2143	2.6659
40	4.8353	8.4644	4.2998
50	8.8622	12.7402	6.4753
70	21.6592	26.8253	14.2384

Table 1
Computing time for different interior point methods: Faster Interior Point coded by us, original Interior Point coded by us, and `quadprog.m` of interior point by MATLAB running 20-step optimization for 50 iterations.

3.6 Result: Compare GPD and GPAD

GPD and GPAD were able to stabilize the system for horizon $T = 20$. As seen from Fig 6, GPD and GPAD were very similar in terms of time domain behavior; both achieved similar convergence time. However, GPAD saved more computation time than GPD by using more aggressive convergence of its dual variable at each time step.
In Fig 4, we can see that there is an instantaneous drop in computation around $T = 38$. We suspect that since computation time depends on not only the problem dimension but also the structure of the problem and the algorithm, this may happen. We can also see some minor dips in other methods as well.

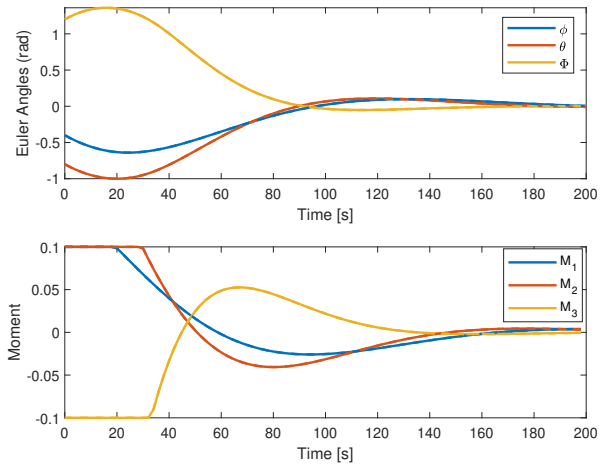

Fig. 6. Euler angles and input moment of satellite using GPD (solid line) and GPAD (dash line), $T = 20$

4 Conclusions and Discussion

For Faster Interior Point, it is faster than original Interior Point Method because of calculating inverse of much smaller matrix or band matrix. However, we still cannot reach the computation speed compared with MATLAB internal function `quadprog.m` due to reasons like memory management, variable allocation, etc. At the same time, Faster Interior Point Method sacrifices controller performance because we set a constant κ and the calculated control input cannot reach the very limit of constraints.

Compared with other methods like PDIP, GPAD, GPD, ADMM, the FBStab is comparable with these methods when horizon is small and it becomes faster than ADMM, GPD and GPAD when horizon is large (when horizon $>$ 100 in our system). However, PDIP (using `quadprog.m`) is the fastest for both large horizon and small horizon. As for the same horizon (like horizon= 20 in our system), the computation time of different methods in each step is similar, while PDIP has the shortest time when constraints are active ($t < 80sec$). When time is long enough ($t > 100sec$) and system has no active constraints, the computation time for ADMM in each step is the shortest.

References

[1] Yang Wang and Stephen Boyd. Fast model predictive control using online optimization. *IFAC Proceedings Volumes*, 41(2):6974–6979, 2008.

[2] Yurii Nesterov. A method of solving a convex programming problem with convergence rate o(1/k2). *Doklady Mathematics*, 27(2):372–376, 1983.

[3] Donghwan Kim and Jeffrey A Fessler. Adaptive restart of the optimized gradient method for convex optimization. *Journal of Optimization Theory and Applications*, 178(1):240–263, 2018.

[4] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[5] Emrah Kılıç and Pantelimon Stanica. The inverse of banded matrices. *Journal of Computational and Applied Mathematics*, 237(1):126–135, 2013.

[6] Euhanna Ghadimi, André Teixeira, Iman Shames, and Mikael Johansson. Optimal parameter selection for the alternating direction method of multipliers (admm): quadratic problems. *IEEE Transactions on Automatic Control*, 60(3):644–658, 2014.

[7] Alberto Bemporad. A quadratic programming algorithm based on nonnegative least squares with applications to embedded model predictive control. *IEEE Transactions on Automatic Control*, 61(4):1111–1116, 2015.

[8] Dominic Liao-McPherson, Mike Huang, and Ilya Kolmanovsky. A regularized and smoothed fischer-burmeister method for quadratic programming with applications to model predictive control. *IEEE Transactions on Automatic Control*, 2018.

[9] Dominic Liao-McPherson and Ilya Kolmanovsky. Fbstab: A proximally stabilized semismooth algorithm for convex quadratic programming. *arXiv preprint arXiv:1901.04046*, 2019.