EE364B Project Report: Embedded MPC

Philippe Weingertner
pweinger@stanford.edu

1 Introduction

We study methods for accelerating Model Predictive Control (MPC) solutions to make them suitable for real-time robotics applications. For embedded devices, reliability, determinism, and real-time performances are of paramount importance [1]. Embedded MPC solutions correspond to a growing commercial market with companies like odys, and embotech. We can exploit structure in the MPC problem [2, 3, 4] to get fast MPC solutions.

Our project features a summary of existing methods, an efficient implementation of Accelerated Dual Gradient Project algorithm (DGPA), an improvement over an existing DGPA algorithm, and a benchmark with off-the-shell interior point solvers on a constrained and ill-conditioned tracking MPC problem.

2 Problem Statement

We formulate a convex-embedded MPC tracking problem as follows:

$$\begin{array}{ll} \min\limits_{x,u} & \frac{1}{2}\sum\limits_{k=0}^{N-1}\left(\left(x_k - x_k^{\text{ref}}\right)^T Q\left(x_k - x_k^{\text{ref}}\right) + u_k^T R u_k\right) + \frac{1}{2}\left(x_N - x_N^{\text{ref}}\right)^T Q_N\left(x_N - x_N^{\text{ref}}\right) \\ \text{s.t.} & x_0 = x_{\text{init}} \\ & x_{k+1} = Ax_k + Bu_k \quad k = 0, \dots, N-1 \\ & y_{\min} \leq Cx_k \leq y_{\max} \quad k = 0, \dots, N \\ & u_{\min} \leq u_k \leq u_{\max} \quad k = 0, \dots, N-1 \end{array}$$

A, B correspond to linearized dynamics, $x_k \in \mathbb{R}^n, u_k \in \mathbb{R}^m, Q, Q_N \in S_{++}^n, R \in S_{++}^m, A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}$.

We rewrite the problem in the following form

$$\begin{array}{ll} \underset{z}{\text{minimize}} & \frac{1}{2} z^T H z \\ s.t. & A_e z = b_e \\ & A_i z \leq b_i \end{array}$$

With

$$z = \left(x_0 - x_0^{\text{ref}}, u_0, \dots, x_{N-1} - x_{N-1}^{\text{ref}}, u_{N-1}, x_N - x_N^{\text{ref}}\right)$$

$$H = \begin{bmatrix} W & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & 0 & W & 0 \\ 0 & 0 & 0 & Q_N \end{bmatrix}, W = \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix}, A_e = \begin{bmatrix} I & 0 & 0 & \cdots & \cdots & 0 \\ -A & -B & I & 0 & \cdots & 0 \\ 0 & -A & -B & I & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & -A & -B & I \end{bmatrix}, b_e = \begin{bmatrix} x_{\text{init}} - x_0^{\text{ref}} \\ Ax_0^{\text{ref}} - x_1^{\text{ref}} \\ Ax_1^{\text{ref}} - x_2^{\text{ref}} \\ \vdots \\ Ax_{N-1}^{\text{ref}} - x_N^{\text{ref}} \end{bmatrix}$$

$$\tilde{A}_i = \begin{bmatrix} C & & & & & \\ & I_m & & & & \\ & & \ddots & & & \\ & & & C & & \\ & & & & I_m & \\ & & & & & C \end{bmatrix}, b_{i_1} = \begin{bmatrix} y_{\max} - Cx_0^{\text{ref}} \\ u_{\max} \\ \vdots \\ y_{\max} - Cx_{N-1}^{\text{ref}} \\ u_{\max} \\ y_{\max} - Cx_N^{\text{ref}} \end{bmatrix}, b_{i_2} = \begin{bmatrix} -y_{\min} + Cx_0^{\text{ref}} \\ -u_{\min} \\ \vdots \\ -y_{\min} + Cx_{N-1}^{\text{ref}} \\ -u_{\min} \\ -y_{\min} + Cx_N^{\text{ref}} \end{bmatrix}, A_i = \begin{bmatrix} \tilde{A}_i \\ -\tilde{A}_i \end{bmatrix}, b_i = \begin{bmatrix} b_{i_1} \\ b_{i_2} \end{bmatrix}$$

The two problems are equivalent, from the z vector we then retrieve x and u vectors.

3 Dual Projected Gradient Approach

Among the algorithms used to solve MPC problems, active set [5] and interior point methods [2] are very popular. Their convergence rate is known in theory, but they are over-conservative. Fast gradient methods have a convergence speed much closer to the theoretical value [6, 7]. It may be advantageous for embedded solutions where predictability and determinism are essential. These methods rely on straightforward arithmetic operations, and the core iterative loop reduces to a few matrix-vector multiplications. With the growing adoption of GPUs, even on embedded platforms, this may also become an advantage. Off-the-shell interior point solvers are not optimized to benefit from GPU capabilities. In any case, active set, interior point, ADMM [8, 9], and fast gradient methods [10] are candidates for fast embedded MPC applications, and each may have its sweet spot. In this project, we study fast gradient methods. We consider a dual approach as the projection related to the inequalities constraints is efficiently handled with a simple projection on the positive orthant. We provide a step-by-step derivation of DGPA algorithms, review variations, and propose two extensions. The proposed algorithm does not depend on hyperparameters tuning, which is also an advantage.

The Lagrangian of the problem is

$$L(z, \lambda) = \frac{1}{2} z^T H z + \lambda^T (A_i z - b_i)$$

with $\lambda \geq 0$. The dual function is

$$g(\lambda) = \inf_{z | A_e z = b_e} \frac{1}{2} z^T H z + \lambda^T (A_i z - b_i)$$

The inner problem we have to solve, given some λ, is:

$$\begin{array}{ll} \underset{z}{\text{minimize}} & \frac{1}{2} z^T H z + \lambda^T (A_i z - b_i) \\ s.t. & A_e z = b_e \end{array}$$

This is a Quadratic Problem (QP) with linear equality constraints. We solve it by applying the KKT conditions. The Lagrangian of the sub problem is

$$L_{inner}(z, \nu) = \frac{1}{2} z^T H z + \lambda^T (A_i z - b_i) + \nu^T (A_e z - b_e)$$

The KKT conditions lead to

$$\nabla_z L_{inner}(z, \nu) = 0, \nabla_\nu L_{inner}(z, \nu) = 0$$

We solve the KKT system $\begin{bmatrix} H & A_e^T \\ A_e & 0 \end{bmatrix} \begin{bmatrix} z \\ \nu \end{bmatrix} = \begin{bmatrix} -A_i^T \lambda \\ b_e \end{bmatrix}$ by elimination.

H is a diagonal matrix. We get an analytical solution:

$$z^*(\lambda) = -H^{-1} A_e^T \left(-H_A^{-1} \left(b_e + A_e H^{-1} A_i^T \lambda \right) \right) - H^{-1} A_i^T \lambda$$

We can precompute H_A^{-1} by Cholesky factorization in $\mathcal{O}\left(\left((N+1)\, n \right)^3 \right)$ as $H_A \in \mathbb{R}^{(N+1)n \times (N+1)n}$. In the formula of $z^*(\lambda)$, all terms not dependent on λ, can also be precomputed and cached. Thus we get:

$$z^*(\lambda) = H^{-1}A_e^T H_A^{-1} b_e + \left(H^{-1}A_e^T H_A^{-1} A_e H^{-1} A_i^T - H^{-1}A_i^T\right)\lambda$$

Which is implemented as:

$$\begin{gathered} z^*(\lambda) = Z_0 + Z_1\lambda \\ Z_0 = H^{-1}A_e^T H_A^{-1} b_e, Z_1 = H^{-1}A_e^T H_A^{-1} A_e H^{-1} A_i^T - H^{-1}A_i^T \end{gathered} \tag{1}$$

Solving the inner sub problem is very fast: a matrix multiplication followed by a matrix addition.

For the dual function

$$g(\lambda) = \inf_{z|A_e z = b_e} \frac{1}{2} z^T H z + \lambda^T (A_i z - b_i)$$

We get

$$g(\lambda) = \frac{1}{2} z^*(\lambda)^T H z^*(\lambda) + \lambda^T (A_i z^*(\lambda) - b_i) \tag{2}$$

We maximize $g(\lambda)$ subject to $\lambda \geq 0$. Thus the update rule is:

$$\lambda_{k+1} \leftarrow \left(\lambda_k + \frac{1}{L}(A_i z^*(\lambda_k) - b_i)\right)_+ \tag{3}$$

L is a Lipschitz constant or Lipschitz matrix we will derive next.

We summarize a first version of the algorithm, a dual projected gradient as:

Algorithm 1 Dual Projected Gradient

1: Initialize λ_0
2: Set $H_A = A_e H^{-1} A_e^T$
3: Compute H_A^{-1}
4: Set $Z_0 = H^{-1}A_e^T H_A^{-1} b_e$
5: Set $Z_1 = H^{-1}A_e^T H_A^{-1} A_e H^{-1} A_i^T - H^{-1}A_i^T$
6: **for** $k = 0$ to $K_{\max}$ **do**
7: $\quad z_k = Z_0 + Z_1\lambda_k$
8: $\quad \lambda_{k+1} = \left(\lambda_k + \frac{1}{L}(A_i z_k - b_i)\right)_+$
9: **end for**

The convergence rate of this algorithm is $\mathcal{O}\left(\frac{1}{k}\right)$, as demonstrated in Nesterov [7] section 2.1.5.

A method from Nesterov in 1983 [6] enables a much faster $\mathcal{O}\left(1/k^2\right)$ convergence rate. These methods are thoroughly treated in Nesterov [6] section 2.2. The main results are concisely summarized in Bertsekas [11] section 6.10.2: Gradient Projection with Extrapolation.The method has the form:

$$\begin{array}{ll} y_k = x_k + \beta_k (x_k - x_{k-1}) & \text{extrapolation step} \\ x_{k+1} = P_X (y_k - \alpha \nabla f(y_k)) & \text{gradient projection step} \end{array}$$

with $\beta_k \in (0,1)$. With the proper choice of β_k, the method has iteration complexity $\mathcal{O}\left(1/k^2\right)$. β_k can be chosen such that

$$\beta_k = \frac{\theta_k (1 - \theta_{k-1})}{\theta_{k-1}}$$

where the sequence $\{\theta_k\}$ satisfies

$$\theta_0 = \theta_1 \in (0, 1]$$

and

$$\frac{1 - \theta_{k+1}}{\theta_{k+1}^2} \leq \frac{1}{\theta_k^2}, \theta_k \leq \frac{2}{k+2}$$

Tseng [12] proposes to use

$$\theta_{k+1} = \frac{\sqrt{\theta_k^4 + 4\theta_k^2} - \theta_k^2}{2}$$

These formula are applied in Patrinos and Bemporad [10].

Algorithm 2 Accelerated Dual Projected Gradient with L set as in Patrinos and Bemporad [10]

Initialize λ_0
Initialize $\theta_0 = \theta_{-1} = 1$
Set $H_A = A_e H^{-1} A_e^T$
Compute H_A^{-1}
Set $Z_0 = H^{-1} A_e^T H_A^{-1} b_e$
Set $Z_1 = H^{-1} A_e^T H_A^{-1} A_e H^{-1} A_i^T - H^{-1} A_i^T$
Set $L = \left\| A_i H^{-1} A_i^T \right\|_2$
for $k = 0$ to $K_{\max}$ **do**
    $\beta_k = \frac{\theta_k (1 - \theta_{k-1})}{\theta_{k-1}}$
    $\omega_k = \lambda_k + \beta_k (\lambda_k - \lambda_{k-1})$
    $z_k = Z_0 + Z_1 \omega_k$
    $\lambda_{k+1} = \left(\omega_k + \frac{1}{L} (A_i z_k - b_i)\right)_+$
    $\theta_{k+1} = \frac{\sqrt{\theta_k^4 + 4\theta_k^2} - \theta_k^2}{2}$
end for

These algorithms have a convergence rate of $\mathcal{O}\left(1/k^2\right)$ assuming the gradient step size is $\alpha = 1/L$ where L is a Lipschitz constant such that

$$\|\nabla f(x) - \nabla f(y)\| \leq \|x - y\|, \forall x, y \in X$$

with $f : \mathbb{R}^n \to \mathbb{R}$ a differentiable convex function defined over a closed convex set X.

One of the key question is how to derive L. In Giselsson [13] the notions of strong convexity and Lipschitz continuity of the gradient of convex functions are generalized to account for different curvatures in different directions. For differentiable and convex functions $f : \mathbb{R}^n \to \mathbb{R}$ with a Lipschitz continuous gradient with constant L, the author generalizes the notion of Lipschitz continuity

$$f(x_1) \leq f(x_2) + \nabla f(x_2)^T (x_1 - x_2) + \frac{L}{2} \|x_1 - x_2\|_2^2$$

to

$$f(x_1) \leq f(x_2) + \nabla f(x_2)^T (x_1 - x_2) + \frac{1}{2} \|x_1 - x_2\|_L^2$$

where

$$\|x\|_L = \sqrt{x^T L x} \text{ with } L \in \mathbb{S}_{++}^n$$

and the notion of strong convexity

$$f(x_1) \geq f(x_2) + \nabla f(x_2)^T (x_1 - x_2) + \frac{\sigma}{2} \|x_1 - x_2\|_2^2$$

to

$$f(x_1) \geq f(x_2) + \nabla f(x_2)^T (x_1 - x_2) + \frac{1}{2} \|x_1 - x_2\|_H^2$$

These two inequalities provide lower and upper quadratic bounds of the function f. By setting

$$\mathbf{L} = L\mathbf{I} \text{ and } \mathbf{H} = \sigma\mathbf{I}$$

we retrieve the original definitions. We replace the constants L and σ by matrices $\mathbf{L}$ and $\mathbf{H}$, which enable gradient updates with different step sizes in different directions. This should provide faster convergence. In Giselsson [13] section 6, an improved fast dual gradient method for solving the problem

$$\text{minimize } \frac{1}{2} y^T H y \text{ subject to } Ay = b, By \leq d$$

is provided. The author introduces dual variables μ for $By \leq d$ which are updated iteratively by

$$\mu_k = \text{prox}_g^{\mathbf{L}_\mu} \left(v^k + \mathbf{L}_\mu^{-\mathbf{1}} \left(By^k - d\right)\right)$$

where $g = I_{\mathcal{Y}}$ is an indicator function on the set $\mathcal{Y} = \{y \mid By \leq d\}$. By restricting $\mathbf{L}_\mu$ to be diagonal, this update takes the form:

$$\mu^k = \max\left(0, v^k + \mathbf{L}_\mu^{-\mathbf{1}}\left(By^k - d\right)\right) \tag{4}$$

In the paper different methods to precompute the matrix $\mathbf{L}_\mu$ are presented. One option is to choose $\boldsymbol{L_\mu}$ by solving a semi-definite program:

$$\text{minimize tr } \mathbf{L}_\mu \text{subject to } \mathbf{L}_\mu \succeq BH^{-1}B^T, \mathbf{L}_\mu \text{ diagonal}$$

We propose a method related to the above family of Improved fast dual gradient methods, which are accelerated first-order methods. We further exploit the structure of our problem and introduce an explicit second-order iteration step to update the dual variables. Considering the dual function in 2 and substituting with 1, we get:

$$g(\lambda) = \frac{1}{2}(Z_0 + Z_1\lambda)^T H(Z_0 + Z_1\lambda) + \lambda^T(A_i(Z_0 + Z_1\lambda) - b_i)$$

We recognize a quadratic function in λ with Hessian

$$H_g = \frac{1}{2}Z_1^T H Z_1 + A_i Z_1$$

We change the first-order gradient update rule of the dual variable in 3 by a second-order Newton step update

$$\lambda_{k+1} \leftarrow \left(\lambda_k + H_g^{-1}\nabla g(\lambda_k)\right)_+$$

where

$$H_g^{-1} = \left(\frac{1}{2}Z_1^T H Z_1 + A_i Z_1\right)^{-1}$$

can be precomputed. The Newton update can be related to 4 where H_g^{-1} is used in place of $\mathbf{L}_\mu^{-\mathbf{1}}$. Unfortunately this method did not work in our experiments. The condition number of H_g is above 10^{33} and we can't apply a Cholesky factorization. As a consequence, we worked out an approximation of either H_g^{-1} or $\mathbf{L}_\mu^{-\mathbf{1}}$ and came up with the following approximation:

$$\mathbf{L}_\mu^{-\mathbf{1}} = \text{diag}\left(A_i H^{-1} A_i^T\right)^{-1}$$

This approximation is related to the SDP problem

$$\text{minimize tr } \mathbf{L} \text{ subject to } \mathbf{L} \succeq A_i H^{-1} A_i^T$$

with L diagonal which is one of the SDP method proposed in [13] to compute the Lipschitz matrix. It works very well in practice. Ultimately, we propose the following algorithm:

Algorithm 3 Accelerated++ Dual Projected Gradient

Initialize λ_0
Initialize $\theta_0 = \theta_{-1} = 1$
Set $H_A = A_e H^{-1} A_e^T$
Compute H_A^{-1}
Set $Z_0 = H^{-1} A_e^T H_A^{-1} b_e$
Set $Z_1 = H^{-1} A_e^T H_A^{-1} A_e H^{-1} A_i^T - H^{-1} A_i^T$
Compute $\mathbf{L}^{-1} = \text{diag}\left(A_i H^{-1} A_i^T\right)^{-1}$
for $k = 0$ to $K_{\max}$ **do**
    $\beta_k = \frac{\theta_k(1-\theta_{k-1})}{\theta_{k-1}}$
    $\omega_k = \lambda_k + \beta_k(\lambda_k - \lambda_{k-1})$
    $z_k = Z_0 + Z_1\omega_k$
    $\lambda_{k+1} = \left(\omega_k + \mathbf{L}^{-\mathbf{1}}(A_i z_k - b_i)\right)_+$
    $\theta_{k+1} = \frac{\sqrt{\theta_k^4 + 4\theta_k^2} - \theta_k^2}{2}$
end for

4 Experiments and Results

We implement the code in Julia. We rely on SparseArrays.jl to take advantage of A_i, A_e sparsity. While we are interested in applying this algorithm to tracking problems for Autonomous Underwater Vehicles (AUV), the proposed algorithm is evaluated on a AFTI-16 aircraft model as in Giselsson [13] and Patrinos and Bemporad [10]. We use a linearized version, at 3000 feet altitude and 0.6 mach velocity, of the F16 aircraft longitudinal dynamics. The problem is setup with

$$x \in \mathbb{R}^4, u \in \mathbb{R}^2, y \in \mathbb{R}^2$$

$$A = \begin{bmatrix} 0.999 & -3.008 & -0.113 & -1.608 \\ 0 & 0.986 & 0.048 & 0 \\ 0 & 2.083 & 1.009 & 0 \\ 0 & 0.053 & 0.05 & 1 \end{bmatrix}, B = \begin{bmatrix} -0.08 & -0.635 \\ -0.029 & -0.014 \\ -0.968 & -0.092 \\ -0.022 & -0.002 \end{bmatrix}, C = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Q = Q_N = \text{diag}\left(\begin{bmatrix} 10^{-4} & 10^{2} & 10^{-3} & 10^{2} \end{bmatrix}\right), R = \text{diag}\left(\begin{bmatrix} 10^{-2} & 10^{-2} \end{bmatrix}\right)$$

$$|u| \leq (25, 25) \text{ and } |y| \leq (0.5, 100)$$

leading to cond $(H) = 10^6$. The objective is to drive the pitch angle y_2 from 0° to 10° and then back to 0°. The control command is given by u_1 an elevator angle and u_2 a flaperon angle. The y_1 output is the angle of attack, and y_2 the pitch angle. The horizon of the MPC problem is given by $N \in [10, 120]$ with a time step $T_s = 0.05s$. We generate a tracking trajectory such that the requested pitch angle y_2 is 10° for $t \in \left[0, \frac{N}{2} T_s\right]$ and 0° afterward, and the requested angle of attack y_1 is 0°. This tracking trajectory is not feasible due to the constraint $|y_2| \leq 0.5$. The constraint on the angle of attack limits the rate of how fast the pitch angle can be changed. We set x_{init} to zero. This problem is ill-conditioned and has open-loop unstable poles. If we solve the problem with a Linear Quadratic Regulator (LQR) method, ignoring the actuators and angle of attack constraints, and then applying actuators clipping, the system becomes unstable. We have to solve the fully constrained optimization problem.

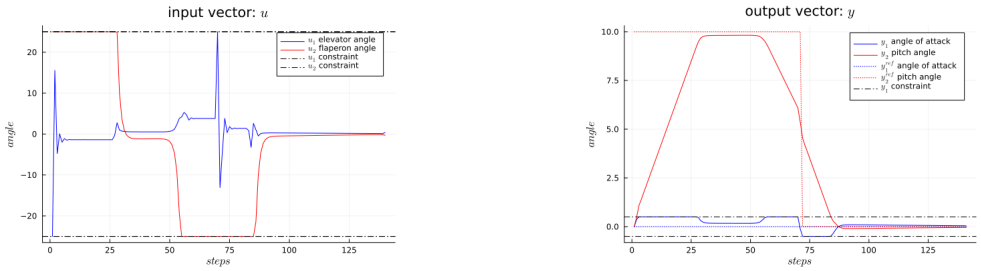

Figure 1: MPC solution for $N = 120$

We benchmark results with off-the-shell solvers: ECOS an interior point solver designed specifically for embedded applications, and SCS. In the figure 2, GPD Ld does not benefit from Nesterov acceleration but computes a step size matrix

$$\mathbf{L^{-1}} = \text{diag}\left(A_i H^{-1} A_i^T\right)^{-1}$$

GPAD Bemporad from [10] uses a fixed step size

$$\alpha = \left\| A_i H^{-1} A_i^T \right\|.$$

GPAD Giselsson from [13] computes $\mathbf{L^{-1}}$ as the solution of a SDP problem:

$$\text{minimize tr } \mathbf{L} \text{ subject to } \mathbf{L} \succeq A_i H^{-1} A_i^T$$

with $\mathbf{L}$ diagonal. In our case, $A_i H^{-1} A_i^T$ is a singular matrix while diag $\left(A_i H^{-1} A_i^T\right)$ is not. In the proposed GPAD algorithm, we simply approximate the solution of the SDP problem with

$$\mathbf{L^{-1}} = \text{diag}\left(A_i H^{-1} A_i^T\right)^{-1}$$

We obtain the same solutions in the same number of iterations as GPAD Giselsson method without precomputing the solution of an SDP problem. In MPC tracking problems, the A_i matrix typically changes every time we set up the MPC problem as we re-linearize nonlinear dynamics models. Thus, in this case, we get a significant speedup.

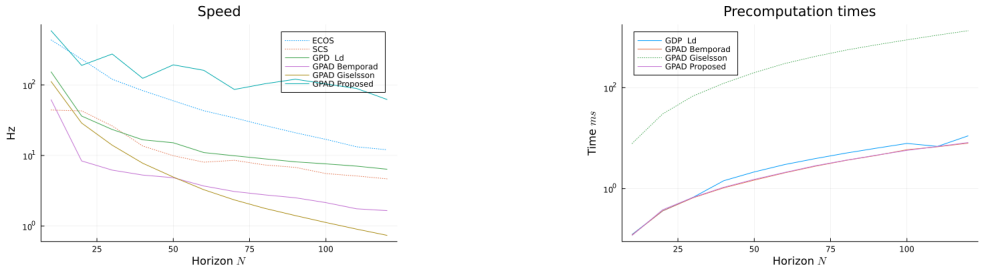

Figure 2: Solver runtimes

For all GPAD solvers, the termination criterion is conditioned on a maximum constraint violation below 10^{-5} or a maximum number of iterations. The proposed GPAD is faster than ECOS except for $N = 20$. The higher the complexity, the more significant the difference. For $N = 100$, GPAD* is six times faster while spending a significant % of time, 50%, in the precomputations. There may still be room for improvement. The solution plots returned by ECOS and the proposed GPAD solvers are the same. The delta in terms of optimal values of $8e^{-3}$ is insignificant from a practical solution point of view.

Table 1: Comparison with ECOS interior point solver

N_{horizon}	10	20	40	60	80	100	120
$(n_{\text{vars}}, m_{\text{constraints}})$	$(64, 128)$	$(124, 248)$	$(244, 488)$	$(364, 728)$	$(484, 968)$	$(604, 1208)$	$(724, 1448)$
ECOS ms	2.3 ms	4.3 ms	12 ms	23 ms	38 ms	59 ms	82 ms
GPAD* ms	1.7 ms	4.9 ms	8 ms	7 ms	10 ms	10 ms	17 ms
GPAD* precomp	6%	7%	12%	38%	38%	50%	50%
GPAD* iters	262	479	441	181	204	120	204
$f_{\text{opt}}^{\text{gpad}^*} - f_{\text{opt}}^{\text{ecos}}$	$1e^{-7}$	$1e^{-7}$	$4.5e^{-5}$	$1.4e^{-3}$	$2e^{-3}$	$8e^{-3}$	$8e^{-3}$

5 Discussion and future work

In the above experiments, the interior point ECOS solver is not tuned for this specific problem. For a fair comparison with interior point methods, we should also study customized interior point methods. Concerning GPAD methods, there may still be room for improvement. We could consider using an augmented Lagrangian and further optimize the precomputations that become dominant for higher complexity problems. However, for the more complex problems, we expect ADMM [8, 9] to be even more promising. We also proposed a variation of the DGPA algorithm with a Newton step instead of the gradient update step. Unfortunately, this method did not work in practice as the Hessian matrix H_g of interest had a terrible condition number of 10^{33}. We could revisit this idea combined with a preconditioning step [14].

6 Conclusion

We derived an Accelerated Dual Projected Gradient implementation running at 588 Hz for a limited time horizon of $N = 10$ and 100 Hz for $N = 100$. The proposed algorithm is up to six times faster than the off-the-shell interior point solver ECOS. We exploited the problem structure using sparse matrices, solved KKT by block elimination, performed precomputations, and used acceleration methods. We update the gradient with different step sizes in different directions avoiding hyper-parameter tuning thanks to a simple formula when previous publications relied on solving an SDP problem.

References

[1] Carlo Alberto Pascucci, Alberto Bemporad, Samir Bennani, and Max Rotunno. Iaa-aas-dycoss 2-14-1407 embedded mpc for space applications. In 2014.

[2] Yang Wang and Stephen Boyd. Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology*, 18(2):267–278, 2010. DOI: 10.1109/TCST.2009.2017934.

[3] Juan L. Jerez, Paul J. Goulart, Stefan Richter, George A. Constantinides, Eric C. Kerrigan, and Manfred Morari. Embedded online optimization for model predictive control at megahertz rates. *IEEE Transactions on Automatic Control*, 59(12):3238–3251, 2014. DOI: 10.1109/TAC.2014.2351991.

[4] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, USA, 2004. ISBN: 0521833787.

[5] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock, and Moritz Diehl. Qpoases: a parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6:327–363, 2014.

[6] Yurii Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. *Proceedings of the USSR Academy of Sciences*, 269:543–547, 1983.

[7] Yurii Nesterov. *Lectures on Convex Optimization*. Springer Publishing Company, Incorporated, 2nd edition, 2018. ISBN: 3319915770.

[8] Felix Rey, Peter Hokayem, and John Lygeros. Admm for exploiting structure in mpc problems. *IEEE Transactions on Automatic Control*, 66(5):2076–2086, 2021. DOI: 10.1109/TAC.2020.3022492.

[9] Felix Rey, Peter Hokayem, and John Lygeros. Admm for exploiting structure in mpc problems. *IEEE Transactions on Automatic Control*, 66(5):2076–2086, 2021. DOI: 10.1109/TAC.2020.3022492.

[10] Panagiotis Patrinos and Alberto Bemporad. An accelerated dual gradient-projection algorithm for embedded linear model predictive control. *IEEE Transactions on Automatic Control*, 59(1):18–33, 2014. DOI: 10.1109/TAC.2013.2275667.

[11] Dimitri P. Bertsekas. Convex optimization theory. In 2009.

[12] P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. *submitted to SIAM Journal on Optimization*, January 2008.

[13] Pontus Giselsson. Improved fast dual gradient methods for embedded model predictive control. *IFAC Proceedings Volumes*, 47:2303–2309, 2014.

[14] Pontus Giselsson and Stephen Boyd. Preconditioning in fast dual gradient methods. In *53rd IEEE Conference on Decision and Control*, pages 5040–5045, 2014. DOI: 10.1109/CDC.2014.7040176.