Model Predictive Control (H0E76A)

Exercise Session 6 | First-order optimization algorithms for MPC

leander.hemelhof@kuleuven.be
renzi.wang@kuleuven.be
brecht.evens@kuleuven.be

2024–2025

1 Problem description

In this session we will first-order optimization methods. These methods are well-studied and particularly suitable for real-time MPC purposes as they have relatively low memory footprints (they don't require solutions to linear systems or building/storing matrices) and are very easy to warm start.

1.1 Dynamics

In previous exercises, we already encountered the following linear model for a car driving in a straight line. Recall the following dynamics equation:

$$x_{k+1} = \underbrace{\begin{bmatrix} 1 & T_\mathrm{s} \\ 0 & 1 \end{bmatrix}}_{A} x_k + \underbrace{\begin{bmatrix} 0 \\ T_\mathrm{s} \end{bmatrix}}_{B} u_k$$

where we set $T_\mathrm{s} = 0.3$ s. As before, the state vector $x = [\,p\ v\,]^\top$ represents the position and the velocity of the vehicle and the input u is the acceleration.

1.2 Optimization problem

Consider the resulting optimal control problem:

$$\begin{aligned}
&\underset{x,u}{\text{minimize}}\ f(x,u), && && \text{(1a)}\\
&\text{s.t. } x_0 = \bar{x}_0, && && \text{(1b)}\\
&\quad x_{k+1} = Ax_k + Bu_k, && k = 0,\dots,N-1, && \text{(1c)}\\
&\quad u_k \in U, && k = 1,\dots,N-1. && \text{(1d)}
\end{aligned}$$

The objective function is our usual quadratic cost in terms of states and inputs, $f(x,u) = \frac{1}{2}x_N^\top P x_N + \sum_{k=0}^{N-1} \frac{1}{2}x_k^\top Q x_k + \frac{1}{2}u_k^\top R u_k$. Due to the input constraints (1d) (which are very common in practice for obvious reasons) this problem is not analytically solvable, in contrast to an equality-constrained QP (i.e., LQR). Hence, our need for numerical solvers.

1.3 Gradient projection

Let us start with the simplest first-order method to deal with the above problem: the gradient projection method. This method combines gradient descent steps on the objective function with projections onto the feasible set (here $U^N = \underbrace{U \times U \times \cdots \times U}_{N\text{times}}$). Before we can employ gradient projection though, we need to eliminate the system dynamics (1c) from the problem.

Exercise 1 | Explain why this is necessary for this particular algorithm.

Solution

A projection of the solution $(\bar{x}_N, \bar{x}_k, \bar{u}_k)_{k=0}^{N-1}$ onto the constraint set is of the form:

$$\begin{aligned} \underset{x_k, u_k}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \|\bar{x}_k - x_k\|^2 + \|\bar{u}_k - u_k\|^2 + \|\bar{x}_N - x_N\|^2 \\ \text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k, \quad k = 0, \dots, N-1 \\ & u_k \in U. \end{aligned}$$

Which is of the same form as the problem we started from. Thus, the projection step alone, would be as difficult as solving the original problem in the first place.

We solve this problem by substituting all the states recursively as functions of the inputs (and with the initial state $\bar{x}_0$ as a parameter). The remaining decision variables are the control inputs, and this formulation is called *single shooting*.

2 Exercises

Exercise 2 | Convert the problem to a single shooting formulation, i.e. eliminate the state variables as a function of the inputs and the initial state as a parameter. The result should be an objective function as a function of the inputs and initial state only, i.e., $f(u, \bar{x}_0) = \frac{1}{2} u^\top H u + g^\top u$. Use $Q = \text{diag}(10, 1)$, $R = 1$, $P = 5Q$, $N = 30$ and $\bar{x}_0 = \begin{bmatrix} -10 & 0 \end{bmatrix}^\top$.

Solution

Using (note that terms of the objective function in only $\bar{x}_0$ have been dropped be dropped as $\bar{x}_0$ is a parameter, not a decision variable)

$$\begin{aligned} f(x, u) = & \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix}^\top \overbrace{\begin{bmatrix} Q & 0 & 0 & \cdots & 0 \\ 0 & Q & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & Q & 0 \\ 0 & 0 & \cdots & 0 & P \end{bmatrix}}^{\bar{Q}} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix} \\ & + \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}^\top \underbrace{\begin{bmatrix} R & 0 & \cdots & 0 \\ 0 & R & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & R \end{bmatrix}}_{\bar{R}} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} \end{aligned}$$

and

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 0 & \cdots & 0 \\ B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix}}_{\bar{B}} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix} + \underbrace{\begin{bmatrix} I \\ A \\ \vdots \\ A^N \end{bmatrix}}_{\bar{A}} \bar{x}_0,$$

we find

$$f(u, \bar{x}_0) = \frac{1}{2} u^\top H u + g^\top u$$
$$H = (\bar{R} + \bar{B}^\top \bar{Q} \bar{B})$$
$$g = \bar{B}^\top \bar{Q}^\top \bar{A} \bar{x}_0.$$

Exercise 3 | Work out the gradient of the cost function and implement the gradient projection algorithm with a fixed step size, for [1].

- $U = [u_{\min}, u_{\max}] = [-10, 10]$;
- stepsize of $\gamma = \frac{1}{L_f}$ (with L_f the Lipschitz constant of ∇f);
- `max_it` = 200;
- Initial guess $u = 0$.

Plot the convergence rate, $\frac{f(u^k) - f(u^*)}{f(u^*)}$ as a function of iteration number k (use `semilogy`). **Hint.** You can use the final iteration u_T as an approximation for u^*. However, because of this approximation, it's best to plot only the errors of the first $T/2$ iterations, since the error in the last few iterations will most likely be in the order of magnitude of the approximation error $\|u^* - u_T\|$

Also take a look at the trajectory (using u^* perform the simulation loop) the car will take. What do you notice?

Solution

The gradient is $\nabla f(u) = Hu + g$. For the implementation, see the solution code. The car has not reached steady state at the end of the predicted horizon.

Exercise 4 | Accelerate the convergence using the Nesterov acceleration technique. Use $\alpha = \frac{\sqrt{\kappa_f} - 1}{\sqrt{\kappa_f} + 1}$, and compare the convergence with the regular projected gradient method. What are your observations?

Solution

See solution code. The Nesterov accelerated iteration is not a descent step! At some iterations, the cost actually increases. However, it is significantly faster than regular projected gradient.
Furthermore, the solution obtained with Nesterov acceleration manages to stabilize the system within the prediction horizon.

Exercise 5 | Implement a line search on the step size γ. Start with $\gamma = \frac{100}{L_f}$. Does the gradient projection step with linesearch perform better or worse than without linesearch? Why is that?

Solution

Line search performs worse in this case, because the step size converges to a value slightly smaller than $\frac{1}{L_f}$. This is due to the discrete nature of the update rule within the line search procedure. This can be tuned to jump with finer steps, but this comes at the cost of more inner iterations during line search. This introduces a trade-off between reducing the number of iterations and the time per iteration.

[1] See `Problem` in `given.config`

This trade-off needs to be made on a case-by-case basis. Thus, whenever information about the problem (such as the Lipschitz constant) is known, it will most likely pay off to use it within your solver.

Exercise 6 | Let's now add state constraints to the problem. Say that at every time step, the position should not exceed the origin, i.e.

$$\begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}p\\v\end{bmatrix}_k \leq 0, \quad k = 0, ..., N. \tag{2}$$

Implement AMA in the sequential approach to solve this state constrained OCP. Use maxiter = 2000. What is the violation on the constraints?

Solution

We have to construct the constraints as $Lu \in C$, where u is our concatenated decision variable. We are looking for a matrix L such that

$$w = (x_0, u_0, x_1, u_1, \ldots, x_N, u_N) = Lu$$

We have (from before)

$$x = \bar{B}u + \bar{A}\bar{x}_0.$$

Now, let us define,

$$F = \begin{bmatrix} F_0 & & & & \\ & F_1 & & & \\ & & \ddots & & \\ & & & F_{N-1} & \\ & & & & F_N \end{bmatrix}, G = \begin{bmatrix} G_0 & & & \\ & G_1 & & \\ & & \ddots & \\ & & & G_{N-1} \\ 0 & 0 & \cdots & 0 \end{bmatrix},$$

where

$$F_0 = F_1 = \cdots = F_{N-1} = \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}, \quad F_N = \begin{bmatrix}1 & 0\end{bmatrix},$$

$$G_0 = G_1 = \cdots = G_{N-1} = \begin{bmatrix}0\\1\end{bmatrix},$$

so that with

$$L = F\bar{B} + G,$$

we obtain that (taking into account the contribution of the initial state):

$$Lu = (p_0 - F_{1:}\bar{A}\bar{x}_0, u_0, \ldots, p_{N-1} - F_{1+2(N-1):}\bar{A}\bar{x}_0, u_{N-1}, p_N - F_{1+2(N-1)}\bar{A}\bar{x}_0),$$

where $p_t = \begin{bmatrix}1 & 0\end{bmatrix} x_t$ and $F_{i:}$ is the ith row of F.
The constraint set now contains copies of the same set (the same constraints are imposed at all time steps).

$$C = Y_0 \times Y_1 \times \cdots Y_{N-1} \times Y_N,$$

with

$$\begin{aligned} Y_k &= \{(w_1, w_2) : w_1 + F_{1+2k:}\bar{A}\bar{x}_0 \leq 0, u_{\min} \leq w_2 \leq u_{\max}\}, \quad k = 0, \cdots, N-1 \\ Y_N &= \{w : w + F_{1+2N:}\bar{A}\bar{x}_0 \leq 0\} \end{aligned}$$

In other words, every constraint set can be split into a one-sided inequality and a box, both of which have simple projections.