Encyclopedia of Systems and Control
DOI 10.1007/978-1-4471-5102-9_9-1

Optimization Algorithms for Model Predictive Control

Moritz Diehl* [a,b]
[a]Department of Microsystems Engineering (IMTEK), University of Freiburg, Freiburg, Germany
[b]ESAT-STADIUS/OPTEC, KU Leuven, Leuven-Heverlee, Belgium

Abstract

This entry reviews optimization algorithms for both linear and nonlinear model predictive control (MPC). Linear MPC typically leads to specially structured convex quadratic programs (QP) that can be solved by structure exploiting active set, interior point, or gradient methods. Nonlinear MPC leads to specially structured nonlinear programs (NLP) that can be solved by sequential quadratic programming (SQP) or nonlinear interior point methods.

Introduction

Model predictive control (MPC) needs to solve at each sampling instant an optimal control problem with the current system state $\bar{x}_0$ as initial value. MPC optimization is almost exclusively based on the so-called *direct approach* which first discretizes the continuous time system to obtain a discrete time optimal control problem (OCP). This OCP has as optimization variables a state trajectory $X = [x_0^\top, \ldots, x_N^\top]^\top$ with $x_i \in \mathbb{R}^{n_x}$ for $i = 0, \ldots, N$ and a control trajectory $U = [u_0^\top, \ldots, u_{N-1}^\top]^\top$ with $u_i \in \mathbb{R}^{n_u}$ for $i = 0, \ldots, N-1$. For simplicity of presentation, we restrict ourselves to the time-independent case, and the OCP we treat in this article is stated as follows:

$$\begin{aligned}
\underset{X,\,U}{\text{minimize}} \quad & \sum_{i=0}^{N-1} L(x_i, u_i) \;+\; E\,(x_N) && \text{(1a)}\\
\text{subject to} \quad & x_0 - \bar{x}_0 = 0, && \text{(1b)}\\
& x_{i+1} - f(x_i, u_i) = 0, \quad i = 0, \ldots, N-1, && \text{(1c)}\\
& h(x_i, u_i) \le 0, \quad i = 0, \ldots, N-1, && \text{(1d)}\\
& r\,(x_N) \le 0. && \text{(1e)}
\end{aligned}$$

The MPC objective is stated in Eq. (1a), the system dynamics enter via Eq. (1c), while path and terminal constraints enter via Eqs. (1d) and (1e). All functions are assumed to be differentiable and to have appropriate dimensions ($h(x,u) \in \mathbb{R}^{n_h}$ and $r(x) \in \mathbb{R}^{n_r}$). Note that $\bar{x}_0 \in \mathbb{R}^{n_x}$ is not an optimization variable, but a parameter upon which the OCP depends via the initial value constraint in Eq. (1b). The optimal solution trajectories depend only on this value and can thus be denoted by $X^*(\bar{x}_0)$ and $U^*(\bar{x}_0)$. Obtaining them, in particular the first control value $u_0^*(\bar{x}_0)$, as fast and reliably as possible for each new value of $\bar{x}_0$ is the aim of all MPC optimization algorithms. The most important dividing line is between convex and non-convex optimal control problems

*E-mail: moritz.diehl@imtek.uni-freiburg.de

DOI 10.1007/978-1-4471-5102-9_9-1

(OCP). If the OCP is convex, algorithms exist that find a global solution reliably and in computable time. If the OCP is not convex, one usually needs to be satisfied with approximations of locally optimal solutions. The OCP (1) is convex if the objective (1a) and all components of the inequality constraint functions (1d) and (1e) are convex and if the equality constraints (1c) are linear.

We typically speak of *linear MPC* when the OCP to be solved is convex, and otherwise of *nonlinear MPC*.

General Algorithmic Features for MPC Optimization

In MPC we would dream to have the solution to a new optimal control problem instantly, which is impossible due to computational delays. Several ideas can help us to deal with this issue.

Off-line Precomputations and Code Generation

As consecutive MPC problems are similar and differ only in the value $\bar{x}_0$, many computations can be done once and for all before the MPC controller execution starts. Careful preprocessing and code optimization for the model routines is essential, and many tools automatically generate custom solvers in low-level languages. The generated code has fixed matrix and vector dimensions, has no online memory allocations, and contains a minimal number of if-then-else statements to ensure a smooth computational flow.

Delay Compensation by Prediction

When we know how long our computations for solving an MPC problem will take, it is a good idea *not* to address a problem starting at the current state but to simulate at which state the system will be when we will have solved the problem. This can be done using the MPC system model and the open-loop control inputs that we will apply in the meantime. This feature is used in many practical MPC schemes with non-negligible computation time.

Division into Preparation and Feedback Phase

A third ingredient of several MPC algorithms is to divide the computations in each sampling time into a preparation phase and a feedback phase. The more CPU intensive *preparation phase* is performed with a predicted state $\bar{x}_0$, before the most current state estimate, say $\bar{x}_0'$, is available. Once $\bar{x}_0'$ is available, the *feedback phase* delivers quickly an *approximate* solution to the optimization problem for $\bar{x}_0'$.

Warmstarting and Shift

An obvious way to transfer solution information from one solved MPC problem to the next one uses the existing optimal solution as an initial guess to start the iterative solution procedure of the next problem. We can either directly use the existing solution without modification for warmstarting or we can first shift it in order to account for the advancement of time, which is particularly advantageous for systems with time-varying dynamics or objectives.

Encyclopedia of Systems and Control
DOI 10.1007/978-1-4471-5102-9_9-1

Iterating While the Problem Changes

A last important ingredient of some MPC algorithms is the idea to work on the optimization problem while it changes, i.e., to never iterate the optimization procedure to convergence for an MPC problem getting older and older during the iterations but to rather work with the most current information in each new iteration.

Convex Optimization for Linear MPC

Linear MPC is based on a linear system model of the form $x_{i+1} = Ax_i + Bu_i$ and convex objective and constraint functions in (1a), (1d), and (1e). The most widespread linear MPC setting uses a convex quadratic objective function and affine constraints and solves the following quadratic program (QP):

$$\begin{aligned}
\underset{X,\,U}{\text{minimize}} \quad & \frac{1}{2}\sum_{i=0}^{N-1}\begin{bmatrix}x_i\\u_i\end{bmatrix}^\top\begin{bmatrix}Q & S\\S^\top & R\end{bmatrix}\begin{bmatrix}x_i\\u_i\end{bmatrix} + \frac{1}{2}x_N^\top P x_N && \text{(2a)}\\
\text{subject to} \quad & x_0 - \bar{x}_0 = 0, && \text{(2b)}\\
& x_{i+1} - Ax_i - Bu_i = 0, \quad i = 0,\dots,N-1, && \text{(2c)}\\
& b + Cx_i + Du_i \le 0, \quad i = 0,\dots,N-1, && \text{(2d)}\\
& c + Fx_N \le 0. && \text{(2e)}
\end{aligned}$$

Here, b, c are vectors and Q, S, R, P, C, D, F matrices, and matrices $\begin{bmatrix}Q & S\\S^\top & R\end{bmatrix}$ and P are symmetric and positive semi-definite to ensure the QP is convex.

Sparsity Exploitation

The QP (2) has a specific sparsity structure that can be exploited in different ways. One way is to reduce the variable space by a procedure called *condensing* and then to solve a smaller-scale QP instead of (2). Another way is to use a *banded matrix factorization*.

Condensing

The constraints (2b) and (2c) can be used to eliminate the state trajectory X. This yields an equivalent but smaller-scale QP of the following form:

$$\begin{aligned}
\underset{U\,\in\,\mathbb{R}^{Nn_u}}{\text{minimize}} \quad & \frac{1}{2}\begin{bmatrix}U\\\bar{x}_0\end{bmatrix}^\top\begin{bmatrix}H & G\\G^\top & J\end{bmatrix}\begin{bmatrix}U\\\bar{x}_0\end{bmatrix} && \text{(3a)}\\
\text{subject to} \quad & d + K\bar{x}_0 + MU \le 0. && \text{(3b)}
\end{aligned}$$

DOI 10.1007/978-1-4471-5102-9_9-1

The number of inequality constraints is the same as in the original QP (2) and given by $m = Nn_h + n_r$. Note that in the simplest case without inequalities ($m = 0$), the solution $U^*(\bar{x}_0)$ of the condensed QP can be obtained by setting the gradient of the objective to zero, i.e., by solving $HU^*(\bar{x}_0) + G\bar{x}_0 = 0$. The factorization of a dense matrix H with dimension $Nn_u \times Nn_u$ needs $O(N^3 n_u^3)$ arithmetic operations, i.e., the computational cost of condensing-based algorithms typically grows cubically with the horizon length N.

Banded Matrix Factorization

An alternative way to deal with the sparsity is best sketched at hand of a sparse convex QP (2) without inequality constraints (2d) and (2e). We define the vector of Lagrange multipliers $Y = [y_0^\top, \ldots, y_N^\top]^\top$ and the Lagrangian function by

$$\begin{aligned}\mathcal{L}(X,U,Y) &= y_0^\top(x_0 - \bar{x}_0) + \frac{1}{2}x_N^\top P x_N \\ &+ \frac{1}{2}\sum_{i=0}^{N-1}\begin{bmatrix}x_i\\u_i\end{bmatrix}^\top\begin{bmatrix}Q & S\\S^\top & R\end{bmatrix}\begin{bmatrix}x_i\\u_i\end{bmatrix} + y_{i+1}^\top(x_{i+1} - Ax_i + Bu_i).\end{aligned} \tag{4}$$

If we reorder all unknowns that enter the Lagrangian and summarize them in the vector

$$W = [y_0^\top, x_0^\top, u_0^\top, y_1^\top, x_1^\top, u_1^\top, \ldots, y_N^\top, x_N^\top]^\top$$

the optimal solution $W^*(\bar{x}_0)$ is uniquely characterized by the first-order optimality condition

$$\nabla_W \mathcal{L}(W^*) = 0.$$

Due to the linear quadratic dependence of $\mathcal{L}$ on W, this is a block-banded linear equation in the unknown W^*:

$$\begin{bmatrix}
0 & I & & & & & & \\
I & Q & S & -A^\top & & & & \\
 & S^\top & R & -B^\top & & & & \\
 & -A & -B & 0 & I & & & \\
 & & & I & Q & \cdot & \cdot & \\
 & & & & \cdot & \cdot & \cdot & \\
 & & & & \cdot & \cdot & 0 & I \\
 & & & & & & I & P
\end{bmatrix} W^* = \begin{bmatrix}\bar{x}_0\\0\\0\\0\\\cdot\\\cdot\\\cdot\\0\end{bmatrix}.$$

Because the above matrix has nonzero elements only on a band of width $2n_x + n_u$ around the diagonal, it can be factorized with a computational cost of order $O(N(2n_x + n_u)^3)$.

DOI 10.1007/978-1-4471-5102-9_9-1

Treatment of Inequalities

An important observation is that both the uncondensed QP (2) and the equivalent condensed QP (3) typically fall into the class of strictly convex parametric quadratic programs: the solution $U^*(\bar{x}_0)$, $X^*(\bar{x}_0)$ is unique and depends piecewise affinely and continuously on the parameter $\bar{x}_0$. Each affine piece of the solution map corresponds to one *active set* and is valid on one polyhedral *critical region* in parameter space. This observation forms the basis of *explicit MPC algorithms* which precompute the map $u_0^*(\bar{x}_0)$, but it can also be exploited in online algorithms for quadratic programming, which are the focus of this section.

We sketch the different ways of how to treat inequalities only for the condensed QP (3), but they can equally be applied in sparse QP algorithms that directly address (2). The optimal solution $U^*(\bar{x}_0)$ for a strictly convex QP (3) is – together with the corresponding vector of Lagrange multipliers, or dual variables $\lambda^*(\bar{x}_0) \in \mathbb{R}^m$ – uniquely characterized by the so-called Karush-Kuhn-Tucker (KKT) conditions, which we omit for brevity. There are three big families of solution algorithms for inequality-constrained QPs that differ in the way they treat inequalities: *active set methods*, *interior point methods*, and *gradient projection methods*.

Active Set Methods

The optimal solution of the QP is characterized by its *active set*, i.e., the set of inequality constraints (3b) that are satisfied with equality at this point. If one would know the active set for a given problem instance $\bar{x}_0$, it would be easy to find the solution. Active set methods work with guesses of the active set which they iteratively refine. In each iteration, they solve one linear system corresponding to a given guess of the active set. If the KKT conditions are satisfied, the optimal solution is found; if they are not, another division into active and inactive constraints needs to be tried. A crucial observation is that an existing factorization of the linear system can be reused to a large extent when only one constraint is added or removed from the guess of the active set. Many different active set strategies exist, three of which we mention: *Primal active set methods* first find a feasible point and then add or remove active constraints, always keeping the primal variables U feasible. Due to the difficulty of finding a feasible point first, they are difficult to warmstart in the context of MPC optimization. *Dual active set strategies* always keep the dual variables λ positive. They can easily be warmstarted in the context of MPC. *Parametric* or *online active set strategies* ensure that all iterates stay primal and dual feasible and go on a straight line in parameter space from a solved QP problem to the current one, only updating the active set when crossing the boundary between critical regions, as implemented in the online QP solver qpOASES.

Active set methods are very competitive in practice, but their worst case complexity is not polynomial. They are often used together with the condensed QP formulation, for which each active set change is relatively cheap. This is particularly advantageous if an existing factorization can be kept between subsequent MPC optimization problems.

Interior Point Methods

Another approach is to replace the KKT conditions by a smooth approximation that uses a small positive parameter $\tau > 0$:

$$H U^* + G\bar{x}_0 + M^\top \lambda^* = 0, \tag{5a}$$

DOI 10.1007/978-1-4471-5102-9_9-1

$$(d + K\bar{x}_0 + MU^*)_i \, \lambda_i^* + \tau = 0, \; i = 1, \ldots, m. \tag{5b}$$

These conditions form a smooth nonlinear system of equations that uniquely determines a primal dual solution $U^*(\bar{x}_0, \tau)$ and $\lambda^*(\bar{x}_0, \tau)$ in the interior of the feasible set. They are not equivalent to the KKT conditions, but for $\tau \to 0$, their solution tends to the exact QP solution. An interior point algorithm solves the system (5a) and (5b) by Newton's method. Simultaneously, the path parameter τ, that was initially set to a large value, is iteratively reduced, making the nonlinear set of equations a closer approximation of the original KKT system. In each Newton iteration, a linear system needs to be factored and solved, which constitutes the major computational cost of an interior point algorithm. For the condensed QP (3) with dense matrices H, M, the cost per Newton iteration is of order $O(N^3)$. But the interior point algorithm can also be applied to the uncondensed sparse QP (2), in which case each iteration has a runtime of order $O(N)$. In practice, for both cases, 10–30 Newton iterations usually suffice to obtain very accurate solutions. As an interior point method needs always to start with a high value of τ and then reduces it during the iterations, warmstarting is of minor benefit. There exist efficient code generation tools that export convex interior point solvers as plain C-code such as CVXGEN and FORCES.

Gradient Projection Methods

Gradient projection methods do not need to factorize any matrix but only evaluate the gradient of the objective function $HU^{[k]} + G\bar{x}_0$ in each iteration. They can only be implemented efficiently if the feasible set is a simple set in the sense that a projection $\mathcal{P}(U)$ on this set is very cheap to compute, as, e.g., for upper and lower bounds on the variables U, and if we know an upper bound $L_H > 0$ on the eigenvalues of the Hessian H. The simple gradient projection algorithm starts with an initialization $U^{[0]}$ and proceeds as follows:

$$U^{[k+1]} = \mathcal{P}\left(U^{[k]} - \frac{1}{L_H}(HU^{[k]} + G\bar{x}_0)\right).$$

An improved version of the gradient projection algorithm is called the *optimal* or *fast gradient method* and has probably the best possible iteration complexity of all gradient type methods. All variants of gradient projection algorithms are easy to warmstart. Though they are not as versatile as active set or interior point methods, they have short code sizes and can offer advantages on embedded computational hardware, such as the code generated by the tool FIORDOS.

Optimization Algorithms for Nonlinear MPC

When the dynamic system $x_{i+1} = f(x_i, u_i)$ is not affine, the optimal control problem (1) is nonconvex, and we speak of a nonlinear MPC (NMPC) problem. NMPC optimization algorithms only aim at finding a locally optimal solution of this problem, and they usually do it in a Newton-type framework. For ease of notation, we summarize problem (1) in the form of a general nonlinear programming problem (NLP):

DOI 10.1007/978-1-4471-5102-9_9-1

$$\underset{X,\,U}{\text{minimize}} \quad \Phi(X,U) \tag{6a}$$

$$\text{subject to} \quad G_{\text{eq}}(X,U,\bar{x}_0) = 0, \tag{6b}$$

$$G_{\text{ineq}}(X,U) \leq 0. \tag{6c}$$

Let us first discuss a fundamental choice that regards the problem formulation and number of optimization variables.

Simultaneous vs. Sequential Formulation

When an optimization algorithm addresses problem (6) iteratively, it works intermediately with nonphysical, infeasible trajectories that violate the system constraints (6b). Only at the optimal solution the constraint residual is brought to zero and a physical simulation is achieved. We speak of a *simultaneous approach* to optimal control because the algorithm solves the simulation and the optimization problems simultaneously. Variants of this approach are *direct discretization* or *direct multiple shooting*.

On the other hand, the equality constraints (6b) could easily be eliminated by a nonlinear forward simulation for given initial value $\bar{x}_0$ and control trajectory U, similar to condensing in linear MPC. Such a forward simulation generates a state trajectory $X_{\text{sim}}(\bar{x}_0, U)$ such that for the given value of $\bar{x}_0, U$ the equality constraints (6b) are automatically satisfied: $G_{\text{eq}}(X_{\text{sim}}(\bar{x}_0, U), U, \bar{x}_0) = 0$. Inserting this map into the NLP (6) allows us to formulate an equivalent optimization problem with a reduced variable space:

$$\underset{U}{\text{minimize}} \quad \Phi(X_{\text{sim}}(\bar{x}_0, U), U) \tag{7a}$$

$$\text{subject to} \quad G_{\text{ineq}}(X_{\text{sim}}(\bar{x}_0, U), U) \leq 0. \tag{7b}$$

When solving this reduced problem with an iterative optimization algorithm, we sequentially simulate and optimize the system, and we speak of the *sequential approach* to optimal control.

The sequential approach has a lower dimensional variable space and is thus easier to use with a black-box NLP solver. On the other hand, the simultaneous approach leads to a sparse NLP and is better able to deal with unstable nonlinear systems. In the remainder of this section, we thus only discuss the specific structure of the simultaneous approach.

Newton-Type Optimization

In order to simplify notation further, we summarize and reorder all optimization variables U and X in a vector $V = [x_0^\top, u_0^\top, \ldots, x_{N-1}^\top, u_{N-1}^\top, x_N^\top]^\top$ and use the same problem function names as in (6) also with the new argument V.

As in the section on linear MPC, we can introduce multipliers Y for the equalities and λ for the inequalities and define the Lagrangian

DOI 10.1007/978-1-4471-5102-9_9-1

$$\mathcal{L}(V, Y, \lambda) = \Phi(V) + Y^\top G_{\text{eq}}(V, \bar{x}_0) + \lambda^\top G_{\text{ineq}}(V). \tag{8}$$

All Newton-type optimization methods try to find a point satisfying the KKT conditions by using successive linearizations of the problem functions and Lagrangian. For this aim, starting with an initial guess $(V^{[0]}, Y^{[0]}, \lambda^{[0]})$, they generate sequences of primal-dual iterates $(V^{[k]}, Y^{[k]}, \lambda^{[k]})$.

An observation that is crucial for the efficiency of all NMPC algorithms is that the Hessian of the Lagrangian is at a current iterate given by a matrix of the form

$$\nabla^2_V \mathcal{L}(\cdot) = \begin{bmatrix} Q_0^{[k]} & S_0^{[k]} & & & & & \\ S_0^{[k],\top} & R_0^{[k]} & & & & & \\ & & Q_1^{[k]} & S_1^{[k]} & & & \\ & & S_1^{[k],\top} & R_1^{[k]} & & & \\ & & & & \ddots & & \\ & & & & & & P^{[k]} \end{bmatrix}.$$

This block sparse matrix structure makes it possible to use in each Newton-type iteration the same sparsity-exploiting linear algebra techniques as outlined in section "Convex Optimization for Linear MPC" for the linear MPC problem.

Major differences exist on how to treat the inequality constraints, and the two big families of Newton-type optimization methods are *sequential quadratic programming (SQP)* methods and *nonlinear interior point (NIP)* methods.

Sequential Quadratic Programming (SQP)

A first variant to iteratively solve the KKT system is to linearize all nonlinear functions at the current iterate $(V^{[k]}, Y^{[k]}, \lambda^{[k]})$ and to find a new solution guess from the solution of a quadratic program (QP):

$$\begin{aligned} \underset{V}{\text{minimize}} \quad & \Phi_{\text{quad}}(V; V^{[k]}, Y^{[k]}, \lambda^{[k]}) && \text{(9a)} \\ \text{subject to} \quad & G_{\text{eq,lin}}(V, \bar{x}_0; V^{[k]}) = 0, && \text{(9b)} \\ & G_{\text{ineq,lin}}(V; V^{[k]}) \le 0. && \text{(9c)} \end{aligned}$$

Here, the subindex "lin" in the constraints $G_{\cdot,\text{lin}}$ expresses that a first-order Taylor expansion at $V^{[k]}$ is used, while the QP objective is given by $\Phi_{\text{quad}}(V; V^{[k]}, Y^{[k]}, \lambda^{[k]}) = \Phi_{\text{lin}}(V; V^{[k]}) + \frac{1}{2}(V - V^{[k]})^\top \nabla^2_V \mathcal{L}(\cdot)(V - V^{[k]})$. Note that the QP has the same sparsity structure as the QP (2) resulting from linear MPC, with the only difference that all matrices are now time varying over the MPC horizon. In the case that the Hessian matrix is positive semi-definite, this QP is convex so that global solutions can be found reliably with any of the methods from section "Convex Optimization for Linear MPC." The solution of the QP along with the corresponding constraint multipliers gives the next SQP iterate $(V^{[k+1]}, Y^{[k+1]}, \lambda^{[k+1]})$. Apart from the presented "exact Hessian" SQP variant, which has quadratic convergence speed, several other SQP variants exist, which make use of other

DOI 10.1007/978-1-4471-5102-9_9-1

Hessian approximations. A particularly useful Hessian approximation for NMPC is possible if the original objective function $\Phi(V)$ is convex quadratic, and the resulting SQP variant is called the *generalized Gauss-Newton* method. In this case, one can just use the original objective as cost function in the QP (9a), resulting in convex QP subproblems and (often fast) linear convergence speed.

Nonlinear Interior Point (NIP) Method

In contrast to SQP methods, an alternative way to address the solution of the KKT system is to replace the last nonsmooth KKT conditions by a smooth nonlinear approximation, with $\tau > 0$:

$$\nabla_V \mathcal{L}(V^*, Y^*, \lambda^*) = 0 \tag{10a}$$

$$G_{\text{eq}}(V^*, \bar{x}_0) = 0 \tag{10b}$$

$$G_{\text{ineq},i}(V^*)\ \lambda_i^* + \tau = 0, \quad i = 1, \ldots, m. \tag{10c}$$

We summarize all variables in a vector $W = [V^\top, Y^\top, \lambda^\top]^\top$ and summarize the above set of equations as

$$G_{\text{NIP}}(W, \bar{x}_0, \tau) = 0. \tag{11}$$

The resulting root finding problem is then solved with Newton's method, for a descending sequence of path parameters $\tau^{[k]}$. The NIP method proceeds thus exactly as in an interior point method for convex problems, with the only difference that it has to re-linearize all problem functions in each iteration. An excellent software implementation of the NIP method is given in the form of the code IPOPT.

Continuation Methods and Tangential Predictors

In nonlinear MPC, a sequence of OCPs with different initial values $\bar{x}_0^{[0]}, \bar{x}_0^{[1]}, \bar{x}_0^{[2]}, \ldots$ is solved. For the transition from one problem to the next, it is beneficial to take into account the fact that the optimal solution $W^*(\bar{x}_0)$ depends almost everywhere differentiably on $\bar{x}_0$. The concept of a continuation method is most easily explained in the context of an NIP method with fixed path parameter $\bar{\tau} > 0$. In this case, the solution $W^*(\bar{x}_0, \bar{\tau})$ of the smooth root finding problem $G_{\text{NIP}}(W^*(\bar{x}_0, \bar{\tau}), \bar{x}_0, \bar{\tau}) = 0$ from Eq. (11) is smooth with respect to $\bar{x}_0$. This smoothness can be exploited by making use of a *tangential predictor* in the transition from one value of $\bar{x}_0$ to another. Unfortunately, the interior point solution manifold is strongly nonlinear at points where the active set changes, and the tangential predictor is not a good approximation when we linearize at such points.

Generalized Tangential Predictor and Real-Time Iterations

In fact, the true NLP solution is not determined by a smooth root finding problem (10a)–(10c) but by the (nonsmooth) KKT conditions. The solution manifold has smooth parts when the active set does not change, but non-differentiable points occur whenever the active set changes. We can deal

DOI 10.1007/978-1-4471-5102-9_9-1

with this fact naturally in an SQP framework by solving one QP of form (9) in order to generate a tangential predictor that is also valid in the presence of active set changes. In the extreme case that only one such QP is solved per sampling time, we speak of a *real-time iteration (RTI)* algorithm. The computations in each iteration can be subdivided into two phases, the *preparation phase*, in which the derivatives are computed and the QP is condensed, and the *feedback phase*, which only starts once $\bar{x}_0^{[k+1]}$ becomes available and in which only a condensed QP of form (3) is solved, minimizing the feedback delay. This NMPC algorithm can be generated as plain C-code, e.g., by the tool ACADO. Another class of real-time NMPC algorithms based on a continuation method can be generated by the tool AutoGenU.

Cross-References

- Explicit Model-Predictive Control
- Model-Predictive Control in Practice
- Numerical Methods for Nonlinear Optimal Control Problems

Recommended Reading

Many of the algorithmic ideas presented in this article can be used in different combinations than those presented, and several other ideas had to be omitted for the sake of brevity. Some more details can be found in the following two overview articles on MPC optimization: Binder et al. (2001) and Diehl et al. (2009). The general field of numerical optimal control is treated in Bryson and Ho (1975), Betts (2010), and the even broader field of numerical optimization is covered in the excellent textbooks (Fletcher 1987; Wright 1997; Nesterov 2004; Gill et al. 1999; Nocedal and Wright 2006; Biegler 2010). General purpose open-source software for MPC and NMPC is described in the following papers: FORCES (Domahidi et al. 2012), CVXGEN (Mattingley and Boyd 2009), qpOASES (Ferreau et al. 2008), FiOrdOs (Richter et al. 2011), AutoGenU (Ohtsuka and Kodama 2002), ACADO (Houska et al. 2011), and IPOPT (Wächter and Biegler 2006).

Bibliography

Betts JT (2010) Practical methods for optimal control and estimation using nonlinear programming, 2nd edn. SIAM, Philadelphia

Biegler LT (2010) Nonlinear programming. SIAM, Philadelphia

Binder T, Blank L, Bock HG, Bulirsch R, Dahmen W, Diehl M, Kronseder T, Marquardt W, Schlöder JP, Stryk OV (2001) Introduction to model based optimization of chemical processes on moving horizons. In: Grötschel M, Krumke SO, Rambau J (eds) Online optimization of large scale systems: state of the art. Springer, Berlin, pp 295–340

Bryson AE, Ho Y-C (1975) Applied optimal control. Wiley, New York

Diehl M, Ferreau HJ, Haverbeke N (2009) Efficient numerical methods for nonlinear MPC and moving horizon estimation. In: Nonlinear model predictive control. Lecture notes in control and information sciences, vol 384. Springer, Berlin, pp 391–417

DOI 10.1007/978-1-4471-5102-9_9-1

Domahidi A, Zgraggen A, Zeilinger MN, Morari M, Jones CN (2012) Efficient interior point methods for multistage problems arising in receding horizon control. In: IEEE conference on decision and control (CDC), Maui, Dec 2012, pp 668–674

Ferreau HJ, Bock HG, Diehl M (2008) An online active set strategy to overcome the limitations of explicit MPC. Int J Robust Nonlinear Control 18(8):816–830

Fletcher R (1987) Practical methods of optimization, 2nd edn. Wiley, Chichester

Gill PE, Murray W, Wright MH (1999) Practical optimization. Academic, London

Houska B, Ferreau HJ, Diehl M (2011) An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. Automatica 47(10):2279–2285

Mattingley J, Boyd S (2009) Automatic code generation for real-time convex optimization. In: Convex optimization in signal processing and communications. Cambridge University Press, New York, pp 1–43

Nesterov Y (2004) Introductory lectures on convex optimization: a basic course. Applied optimization, vol 87. Kluwer, Boston

Nocedal J, Wright SJ (2006) Numerical optimization. Springer series in operations research and financial engineering, 2nd edn. Springer, New York

Ohtsuka T, Kodama A (2002) Automatic code generation system for nonlinear receding horizon control. Trans Soc Instrum Control Eng 38(7):617–623

Richter S, Morari M, Jones CN (2011) Towards computational complexity certification for constrained MPC based on Lagrange relaxation and the fast gradient method. In: 50th IEEE conference on decision and control and European control conference (CDC-ECC), Orlando, Dec 2011, pp 5223–5229

Wächter A, Biegler LT (2006) On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Math Program 106(1):25–57

Wright SJ (1997) Primal-dual interior-point methods. SIAM, Philadelphia