Control Engineering Practice 39 (2015) 56–66

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

A toolkit for nonlinear model predictive control using gradient projection and code generation

Jouko Kalmari [a,*], Juha Backman [b], Arto Visala [a]

[a] Aalto University, School of Electrical Engineering, Department of Electrical Engineering and Automation, P.O. Box 15500, 00076 Aalto, Finland
[b] MTT Agrifood Research Finland, Vakolantie 55, 03400 Vihti, Finland

ARTICLE INFO

Article history:
Received 4 August 2014
Accepted 8 January 2015
Available online 9 March 2015

Keywords:
Algorithms and software
Model predictive control
Gradient methods
Anti-sway control

ABSTRACT

Nonlinear model predictive control (NMPC) is a control strategy based on finding an optimal control trajectory that minimizes a given objective function. The optimization is recalculated at each control cycle and only the first control values are actually used. The dynamics of the system can be nonlinear and there can be constraints on states and controls. A new toolkit called VIATOC has been developed that can be used to automatically generate the code needed to implement NMPC. The generated code is self-contained ANSI C and the compiled program has a small footprint. In VIATOC, the gradient projection method is used to solve the nonlinear optimization problem. Barzilai–Borwein type step length selection for the gradient method has also been implemented. The performance of the controllers generated with the toolkit is compared with those solved with the ACADO toolkit and HQP. The performance of the optimization is compared with two different test cases with different numbers of controls and states. The first one is based on a model of a pendulum hanging freely on a movable platform. The second one is a more complex model of a chain of three masses connected by springs. Seven different prediction horizons between 10 and 100 steps are used. When the time to achieve a near optimum solution is measured, VIATOC is in most cases the fastest one when the length of the prediction horizon is shorter than 70 steps.

1. Introduction

Nonlinear model predictive control (NMPC) is a feedback control strategy based on nonlinear optimization. The optimization is usually recalculated at each time step for a new control. A prerequisite for NMPC is that there is a dynamic model of the system that is being controlled. As the name implies, the model consists of nonlinear differential equations. States and controls can have constraints that need to be taken into account. The optimization is based on minimizing an objective function that usually has quadratic cost terms for the differences in states and controls compared to a reference trajectory. An advantage of an NMPC is that it can predict the future state of the system and therefore find a near-optimal control path. One disadvantage is the required amount of computational power, which restricts the complexity of systems that can be controlled, the maximum control frequency, and the length of prediction horizons. Therefore, the efficiency of the optimization is usually of the utmost importance.

NMPCs have traditionally been used in the process industry to optimize the operation point of the process (Nagy, Mahn, Franke, & Allgöwer, 2007), while low-level unit controllers are used for setpoint tracking. Lately, NMPCs have also been utilized in path tracking for autonomous agricultural machines (Backman, Oksanen, & Visala, 2012), flood regulation of a river (Barjas Blanco et al., 2010), and overhead crane control (Schindele & Aschemann, 2011). Some of the applications have had relatively fast dynamics and require efficient algorithms for real-time operation.

In the general case, the optimization required by the NMPC cannot be solved analytically. Therefore numerical methods are used. There are two types of numerical iterative methods available: indirect and direct methods. With the indirect method, a root of the necessary optimization condition is searched. With the direct method, a sequence of control trajectories is constructed such that the objective function is minimized and typically the value of the objective function decreases at every step. There are advantages in both strategies, but the direct method is more popular because it is numerically more robust and easier to initialize (Betts, 2001).

Most toolkits that the authors are aware of utilize the so-called sequential quadratic programming (SQP) strategy. With the SQP

* Corresponding author. Tel.: +358 505680669.
E-mail addresses: jouko.kalmari@aalto.fi (J. Kalmari), juha.backman@mtt.fi (J. Backman), arto.visala@aalto.fi (A. Visala).

http://dx.doi.org/10.1016/j.conengprac.2015.01.002
0967-0661/

algorithm, the fundamental idea is to linearize the problem, solve it as a quadratic programming (QP) problem, and repeat the process until sufficient convergence is achieved. QP sub-problems can be solved iteratively using the Newton method. When the Lagrangian function is used, only the equality constraints are present. However, in NMPC usually there are also inequality constraints. One solution is to use an Active Set strategy where the active set of constraints changes. The interior point method is another way of solving the QP sub-problem of the SQP. However, the interior point method can also be used directly to solve the nonlinear optimization problem. IPOPT is a popular tool that is designed for solving optimization problems using the interior point method.

HQP (Huge Quadratic Programming) is a tool that uses an SQP algorithm to solve nonlinearly constrained optimization problems. Convex quadratic sub-problems are solved using a polynomial time interior-point method. The Lagrangian function of the problem is approximated quadratically by a sparse or dense Hessian matrix, which is updated numerically. The Jacobian matrices of the system equations and the cost function can be numerically approximated, or the user can provide exact Jacobians. For the continuous time optimization problem formulation, the Omuses front-end can be used (Franke & Arnold, 2008).

Another toolkit using SQP is ACADO, an open source toolkit designed for automatic control and dynamic optimization. The ACADO Toolkit is based on four key properties: open source, user friendliness, code extensibility, and self-contentedness (Houska, Ferreau, & Diehl, 2011a). The toolkit is freely available and is released under GNU LGPL. The user-friendliness and code extensibility have been achieved by utilizing the object-oriented capabilities of C++ and careful design of interfaces. External packages can be used, but ACADO is basically designed to be self-contained.

ACADO also has a code generation tool that is designed to export optimized C code (Houska, Ferreau, & Diehl, 2011). First, the user defines the NMPC problem using C++ code. The toolkit then exports a tailored Runge–Kutta method, required derivatives of the dynamic model, discretization, and condensing routines. The auto-generated code does not use any dynamic memory and is easily used on different hardware and operating systems. ACADO supports different QP solvers, e.g., CVXGEN (Mattingley & Boyd, 2009), qpOASES (Ferreau, 2012), and FORCES (Domahidi, 2012), which are used to solve the underlying quadratic programming problem. The ACADO code-generation tool is used in this study as the main performance benchmark due to many similarities compared to the newly developed toolkit.

CasADi is another open-source toolkit that is capable of solving optimization and optimal control problems. It supports different discretization methods, such as single and multiple shooting methods and collocation methods. However, currently the code generation support for integrators is still under development, and therefore CasADi was not used for performance comparison in this study.

GRAMPC is software designed for nonlinear model predictive control (Kapernick & Graichen, 2014). It uses a projected gradient method to solve the optimization. The step length in the direction of the gradient is selected either using a polynomial approximation of the cost function or a line search of Barzilai and Borwein (1988). Currently, GRAMPC only supports control constraints.

The objective of this research was to develop a lightweight NMPC toolkit based on an efficient implementation of the gradient projection method. In this paper, a new open source toolkit, VIATOC, will be presented. VIATOC is developed for generating C code from the user-defined NMPC problems, and it also contains integrators and optimization algorithms required for actually solving the optimal control trajectories. The methods and main ideas used and performance results with a comparison to other toolkits are detailed.

VIATOC has already been used for controlling a real hydraulic forestry crane. The objective was to control the position of the tip of the boom and simultaneously damp the undesired oscillations of a tool attached to it. The dynamic model consisted of 12 states and 4 controls, and the prediction horizon of 10 s was divided into 100 steps. The required 100 ms control frequency was achieved, even though the dynamic model was relatively complex and the prediction horizon was quite long. The results from the crane control tests are discussed in more detail by Kalmari, Backman, and Visala (2014).

The structure of this paper is the following. Section 2 presents the control problem formulation that the toolkit solves. In this section, the methods utilized to solve the problem are also explained. The usage of VIATOC and the structure of the software are presented in Section 3. Section 4 presents the test problem and how the performance is compared against ACADO Code Generation and HQP. Results of the comparisons are given and discussed in Section 5. Some concluding remarks are made in Section 6.

2. Numerical methods in VIATOC

The VIATOC toolkit is developed for solving NMPC problems based on the following optimal control problem:

$$\min_{\mathbf{x}(\cdot),\mathbf{u}(\cdot)} \int_0^{t_f} \left(\|\mathbf{x}(t)-\mathbf{x}_{ref}(t)\|^2_{\mathbf{Q}} + \|\mathbf{u}(t)-\mathbf{u}_{ref}(t)\|^2_{\mathbf{R}} \right) dt + \|\mathbf{x}(t_f)-\mathbf{x}_{ref}(t_f)\|^2_{\mathbf{P}}$$

$$\begin{aligned} \text{s.t. } & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t),\mathbf{u}(t)) \\ & \mathbf{x}(0) = \mathbf{x}_0 \\ & \mathbf{u}_{min} \le \mathbf{u}(t) \le \mathbf{u}_{max}, \quad t \in [0, t_f] \\ & \mathbf{x}_{min} \le \mathbf{x}(t) \le \mathbf{x}_{max}, \quad t \in [0, t_f]. \end{aligned} \tag{1}$$

The state vector of the system is $\mathbf{x}(t)$ and the system is controlled with $\mathbf{u}(t)$. The dynamics of the system are defined using a state-space model using the function $\mathbf{f}(\mathbf{x}(t),\mathbf{u}(t))$. Currently, the dynamics can consist of arithmetic operations, basic trigonometric, and other common mathematical functions. The initial condition of the system is given with $\mathbf{x}_0$. In NMPC, this initial value is usually the current measured or estimated state of the system.

The optimal solution minimizes the quadratic objective function of states and controls with the constraints taken into account. $\mathbf{x}_{ref}(t)$ and $\mathbf{u}_{ref}(t)$ are the state and control reference values, respectively. $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{P}$ are positive semidefinite weighting matrices of the error in state trajectory, error in control trajectory, and error in the final state values, respectively. $\mathbf{u}_{min}$, $\mathbf{u}_{max}$, $\mathbf{x}_{min}$, and $\mathbf{x}_{max}$ are constant upper and lower bounds for the controls and states.

The process of solving the optimal control problem is divided into two steps. In the first step, the state trajectory is integrated and the dynamics are linearized using the initial control trajectory. In the second step, the linearized optimization problem is solved. The solution of the second step is then used as the new initial solution and the steps are repeated as many times as the user requests. This process is basically the same as in SQP methods, where the original problem is solved iteratively as a series of QP problems. However, in our implementation, only a fixed number of iterations are done when solving the linearized problem in the second step. Fixing the number of iterations makes the time requirement for the optimization more predictable, which in the case of NMPC can be a more important factor than the exact optimality of the solution.