Control Engineering Practice 78 (2018) 105–115

Contents lists available at ScienceDirect

Control Engineering Practice

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

Nonlinear predictive control on a heterogeneous computing platform

Bulat Khusainov [a], Eric Kerrigan [a,b,*], Andrea Suardi [a], George Constantinides [a]

[a] Department of Electrical & Electronic Engineering, Imperial College London, London, SW7 2AZ, UK
[b] Department of Aeronautics, Imperial College London, London, SW7 2AZ, UK

ARTICLE INFO

Keywords:
Predictive control
Hardware–software co-design
Scheduling
FPGA
Optimization-based control

ABSTRACT

We propose an implementation of an interior-point-based nonlinear predictive controller on a heterogeneous processor. The workload can be split between a general-purpose CPU and a field-programmable gate array to trade off the contradicting design objectives of control performance and computational resource usage. A new way of exploiting the structure of the KKT matrix yields significant memory savings. We report an 18x memory saving, compared to existing approaches, and a 6x speedup over a software implementation with an ARM Cortex-A9 processor. We also introduce a new release of Protoip, which abstracts low-level details of heterogeneous programming and allows processor-in-the-loop verification.

1. Introduction

Explicit performance optimization, systematic constraint handling and the inherent capability of dealing with nonlinearities are the main features that explain the success of Model Predictive Control (MPC) in recent decades (Rawlings & Mayne, 2009). In the MPC framework a dynamic optimization problem is solved at each sampling instant, which might restrict the application scope to systems with slow dynamics and/or render expensive implementations limited to high-performance computers.

Conventionally these challenges were addressed on the algorithmic and software levels by developing new optimization problem formulations and generating hardware-efficient code (Houska, Ferreau, & Diehl, 2011). However, in addition to improvements on the software side, recent developments in reconfigurable computing allowed acceleration of predictive control algorithms on Field-Programmable Gate Arrays (FPGAs), which resulted in low-cost and resource-efficient realizations of custom quadratic programming (QP) solvers for MPC. Consider the following implementations:

- Jerez et al. (2014) and Peyrl, Zanarini, Besselmann, Liu, and Boéchat (2014) propose several implementations of first order-based MPC on FPGAs. The main computational kernel for this class of algorithms is matrix–vector multiplication, which has a huge parallelization potential. These papers differ in the way this operation is implemented: the former implementation is based on an adder tree, while the latter relies on multiply-accumulate units.
- Vouzis, Bleris, Arnold, and Kothare (2009) presents an implementation of linear MPC on a system-on-a-chip that consists of a CPU and an FPGA. Eliminating the states from the decision variables and incorporating inequality constraints in the cost function leads to an unconstrained nonlinear optimization problem, which is solved using Newton's method. For Newton's method the authors propose evaluating derivatives on the CPU and accelerating solving linear systems on the FPGA.
- Jerez, Ling, Constantinides, and Kerrigan (2012) describes a two-stage architecture for interior point-based predictive control. Interior point algorithms have a similar computational pattern as Newton's method. However, the considered architecture is tailored for solving quadratic programming problems (in contrast to general nonlinear programming problems), which simplifies evaluation of the derivatives. The linear system solver is based on an iterative algorithm, hence the computational logic is reused efficiently.

Other examples of QP-based MPC implementations and/or architectures can be found in Hartley et al. (2014), Knagge, Wills, Mills, and Ninness (2009) and Ling, Yue, and Maciejowski (2006).

Extending a hardware acceleration approach from linear to nonlinear model predictive control (NMPC), which can be considered as the next logical step, requires mapping numerical integration algorithms on hardware, which is not a trivial task, since dynamic models are problem-dependent. As a result, instead of using systematic dynamic optimization, existing FPGA implementations of NMPC either rely on

* Corresponding author at: Department of Electrical & Electronic Engineering, Imperial College London, London, SW7 2AZ, UK.
E-mail address: e.kerrigan@imperial.ac.uk (E. Kerrigan).

https://doi.org/10.1016/j.conengprac.2018.06.016
Received 24 October 2017; Received in revised form 18 June 2018; Accepted 25 June 2018
Available online 3 July 2018
0967-0661/

stochastic optimization (Xu, Chen, Gong, & Mei, 2016) or approximate the offline solution with machine learning techniques (Ayala et al., 2016). These approaches cannot guarantee scalability or closed-loop performance.

Accelerating deterministic algorithms on hardware might be achieved by employing heterogeneous computing platforms that involve both a general-purpose processor with a fixed architecture and FPGA logic. For example, Peyrl, Ferreau, and Kouzoupis (2015) present a heterogeneous implementation of a multiple-shooting based NMPC algorithm. The authors propose implementing integration in software while accelerating a fast gradient-based quadratic programming solver on an FPGA. The reported speedup of the heterogeneous implementation over a software realization is 1.6x and further improvement is limited, since integration and optimization algorithms have comparable computational complexity. This is a consequence of Amdahl's law (Amdahl, 1967), which states that an algorithm's speedup is limited by the part of the workload that cannot benefit from acceleration.

We present a new heterogeneous implementation of a nonlinear interior-point algorithm for predictive control that was first introduced in Khusainov, Kerrigan, Suardi, and Constantinides (2017). The main features of the proposed implementation are:

- A method for scheduling sparse matrix–vector multiplication within an iterative linear system solvers to enable significant improvements in terms of computation time vs resource usage. For the example considered, an 18x memory saving compared to existing approaches and a 6x speedup over a software implementation are reported.
- Flexible splitting of the algorithmic workload between software and hardware for trading off the computational resource usage against performance.

In addition to the initial results presented in Khusainov et al. (2017), this paper presents the following extensions:

- The whole family of implicit and explicit Runge–Kutta methods are supported for ordinary differential equations integration. In contrast, the initial implementation was limited to the Euler method only.
- The optimal control objective is generalized from a quadratic function to nonlinear least squares.
- The proposed controller is experimentally validated in the loop with a gantry crane high-fidelity Simscape (The Mathworks, Inc, 2015) model. In Khusainov et al. (2017) the controller was only validated in the loop with a nominal model.

Another contribution of the paper is a new release of the Protoip software tool (Suardi, Constantinides, & Kerrigan, 2015). Protoip allows quick prototyping and processor-in-the-loop verification of optimization algorithms on a Xilinx Zynq system-on-a-chip (SoC), which contains an ARM processor and FPGA fabric. In contrast to the previous releases, which were focused on pure FPGA implementations, the new version of Protoip allows the incorporation of both an ARM processor and FPGA. Protoip can be used both for quick testing of the proposed implementation from the MATLAB environment and for design and verification of other heterogeneous implementations.

The remainder of the paper is organized as follows: Section 2 describes the considered optimal control problem formulation; existing NMPC algorithms, with a focus on suitability for hardware implementation, are reviewed in Section 3; in Section 4 a heterogeneous computer-based implementation of NMPC is presented, followed by experimental setup description (Section 5) and experimental results (Section 6). Note, that Protoip is a part of the experimental setup and hence presented in the corresponding section. Section 7 concludes the paper.

2. Optimal control problem formulation

We consider nonlinear time-invariant systems that can be described as an ordinary differential equation (ODE) of the form

$$\dot{x}(t) = f_c(x(t), u(t)), \tag{1}$$

where $f_c : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$. We consider the nonlinear optimal control problem (OCP) with initial state $\hat{x}$ and prediction horizon T:

$$\min_{x,u,s} \quad \int_0^T \|h(x(t), u(t), s(t))\|_2^2 dt + \|h_T(x(T), s_T)\|_2^2 \tag{2a}$$

subject to:

$$x(0) = \hat{x}, \tag{2b}$$
$$\dot{x}(t) = f_c(x(t), u(t)), \qquad \forall t \in [0, T] \tag{2c}$$
$$q(x(t), u(t), s(t)) = 0, \qquad \forall t \in [0, T] \tag{2d}$$
$$q_T(x(T), s_T) = 0, \tag{2e}$$
$$x_l \le x(t) \le x_u, \qquad \forall t \in [0, T] \tag{2f}$$
$$u_l \le u(t) \le u_u, \qquad \forall t \in [0, T] \tag{2g}$$
$$s_l \le s(t) \le s_u, \qquad \forall t \in [0, T] \tag{2h}$$
$$x_{Tl} \le x(T) \le x_{Tu}, \tag{2i}$$
$$s_{Tl} \le s_T \le s_{Tu}, \tag{2j}$$

where $h : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^{n_S} \to \mathbb{R}^{n_h}$ and $h_T : \mathbb{R}^n \times \mathbb{R}^{n_{ST}} \to \mathbb{R}^{n_{hT}}$. Slack trajectory s and slack variable s_T are introduced alongside with (2d) and (2e), which is a common technique that allows for the handling of general nonlinear inequality constraints (Wächter & Biegler, 2006). Note that in the general case $s(T) \neq s_T$. The presented formulation can be generalized for time-varying reference tracking, which, as will be shown later, requires only changing the software part of the algorithm.

3. Nonlinear predictive control algorithms

Direct solution of the continuous-time optimal control problem (2) involves two main stages: integration, i.e. solving the ordinary differential equation (ODE), and optimization. Implementing integration on an FPGA is not desirable because of the following reasons:

- The ODE (2c) may involve mathematical expressions (e.g. sine and square root) that, in contrast to standard addition and multiplication operations, require significant amounts of computational resources (Fig. 1) and might be unsuitable for pipelining. For examples on implementation of nonlinear operators on FPGAs refer to de Dinechin, Joldes, Pasca, and Revy (2010) and Detrey and de Dinechin (2007).
- A vector function f_c is composed of scalar functions that might have different underlying mathematical operations and different evaluation complexities, i.e. f_c might have *irregular* structure. Irregularity potentially limits reusing computational logic and speedup by parallelization.

Optimization algorithms, on the other hand, can benefit from hardware acceleration due to (i) their iterative nature, which is beneficial for reusing computational logic, and (ii) the fact that underlying linear algebra algorithms can be efficiently mapped onto hardware (Jerez et al., 2012).

Taking the above into account we consider two classes of algorithms for solving (2): shooting-based and direct transcription algorithms (Betts, 2010). The common feature of shooting methods is decoupling the ODE and optimization solvers. Accelerating only the latter does not result in significant improvements, due to Amdahl's law, since the workloads of the two operations are comparable. In contrast, direct transcription algorithms transform (2) directly to a discrete OCP by