Number Representations for Embedding Optimization Algorithms in Cyber-Physical Systems

Eric C. Kerrigan * **Juan L. Jerez** ** **Stefano Longo** **
George A. Constantinides **

* *Department of Electrical & Electronic Engineering and Department of Aeronautics, Imperial College London, Exhibition Road, London SW7 2AZ, UK (e-mail: e.kerrigan@ imperial.ac.uk).*
** *Department of Electrical and Electronic Engineering, Imperial College London, Exhibition Road, London SW7 2AZ, UK (e-mail: juan.jerez-fullana05,s.longo,g.constantinides@imperial.ac.uk)*

Abstract: We will give an overview of some recent work done at Imperial College London that aims to develop a deep understanding of the fundamental issues that are important in numerical representations when developing optimization algorithms for deployment in cyber-physical systems. We will show that existing formulations and algorithms for solving optimization problems result in very poor performance when low-precision or fixed-point arithmetic is used. We will also present some new systematic methods and theoretical results that we have developed for addressing some of these problems. These results allow for optimization algorithms to be implemented in embedded systems with significant reductions in cost, energy and computation time.

1. INTRODUCTION

In many cyber-physical applications, where one would like to implement optimal control and signal processing algorithms, one needs to use the latest measurements to update and solve a sequence of numerical optimization problems. Solving these optimization problems in a computationally efficient and numerically reliable fashion on an embedded computing system is a challenging task.

A major factor in deciding what to include on a microprocessor is the die area. Because of defects during manufacturing, not all manufactured dies are acceptable — a good rule of thumb for modern manufacturing processes is that the cost per die grows roughly with the *square* of the die area [Hennessy and Patterson, 2011, Chap. 1.6]. Furthermore, static and dynamic power consumption of a chip roughly grows at least linearly with area, ignoring the additional increase in power consumption due to increases in interconnection length. As a consequence, in order to reduce cost and energy requirements, it pays to minimize the area taken up by a microprocessor.

One of the key choices that an engineer has to make in order to control the area of a chip is the number representation that will be used in the arithmetic units. For fixed-point arithmetic with bit parallel two's complement, the area of multipliers typically scales quadratically and the area of adders scales linearly with the number of bits. For floating-point arithmetic, the area of multipliers and adders usually scale somewhere between linearly and quadratically with the number of bits, since the area of the shifter in the floating-point adder scales somewhere between linearly and quadratically with the number of bits. It is clear that significant savings in cost and power requirements are possible by appropriate choice of number representation and reducing the number of bits. Improvements in latency (time taken to complete a computation, or commonly referred to as the computational delay) and throughput (number of completed computations per unit time) are also possible by reducing the number of bits.

Nearly all CPUs in modern desktop PCs provide hardware support for the IEEE-754 standard for double-precision floating-point, which uses 52 bits for the mantissa (or significand), 11 bits for the exponent and one bit for the sign. However, most microprocessors in embedded systems do not offer any support for double-precision floating-point. Instead, they may only offer floating-point support for single precision (23 bits for the mantissa and 8 bits for the exponent) or half precision (10 bits for the mantissa and 5 bits for the exponent) or not even provide any support for floating-point at all. It is therefore possible that, because of a significant decrease in precision or different number representation, an optimization algorithm that gives reliable numerical results when implemented in the office desktop or laptop computer might give completely different results when implemented on an embedded system.

2. FIXED-POINT ARITHMETIC

Because energy consumption and cost are two major concerns in embedded systems, most microprocessors provide very good support for fixed-point arithmetic. This is because a fixed-point arithmetic unit takes up significantly less area on a chip than a floating-point unit with the same number of bits, therefore resulting in significant savings in manufacturing cost and energy requirements. Fixed-point

⋆ This summary is a shortened version of Kerrigan et al. [2012].

Table 1. Resources required and latency for a single floating- or fixed-point adder on a Xilinx Virtex-7 XT 1140 FPGA.

Number Representation	Registers Flip-flops (FFs)	Latency (clock cycles)
double float (52-bit mantissa)	1046	14
single float (23-bit mantissa)	557	11
53-bit fixed-point	53	1
24-bit fixed-point	24	1

units are also faster than floating-point units with the same number of bits, due to the fact that the radix points (binary points) of two numbers do not need to be aligned during addition or subtraction. Table 1 shows the resources required and latency of an addition on a modern Field Programmable Gate Array (FPGA), which is a popular class of embedded processors employed in many applications. As can be seen, it is possible to decrease the latency of an addition by roughly one order of magnitude, with an even bigger relative reduction in the resources required, by switching from floating-point to fixed-point.

In order to allow for the use of fixed-point arithmetic in optimization solvers, we will focus on the most computationally expensive and numerically critical part, namely the computation of the search direction. We will propose the use of the minimal residual (MINRES) method for solving the resulting set of linear equations and implementing the Lanczos algorithm using fixed-point arithmetic.

The main challenge for fixed-point over floating-point is to avoid overflow and develop a computationally tractable method to a priori determine tight bounds on the range of the variables. Current methods for automatically computing the range of variables cannot handle algorithms that are both nonlinear and recursive, which includes direct and iterative algorithms for solving systems of linear equations.

We will present a novel preconditioner that guarantees that the eigenvalues of the preconditioned matrix has eigenvalues inside the unit disk. This allows one to analytically derive tight bounds on all the variables in the Lanczos process, in order to determine a priori where to place the radix point (binary point) such that numerical errors due to overflow are avoided. We will show that fixed-point arithmetic makes more efficient use of the resources and reduces latency, while also being able to guarantee the same accuracy of the solution as with floating-point arithmetic.

3. LOW-PRECISION ARITHMETIC

Double- or single-precision floating-point representation may be unnecessarily precise for a given application, where precision would be better traded in for improving more important aspects, such as speed, cost and energy consumption. The problem with reducing the number of bits is that existing optimization algorithms may give unacceptable results when using a very low precision.

When implementing algorithms for solving optimization problems using low precision arithmetic, it is important to be careful with the formulation of the optimization problem so that important information is not lost prior to solving the problem. However, current methods for computing

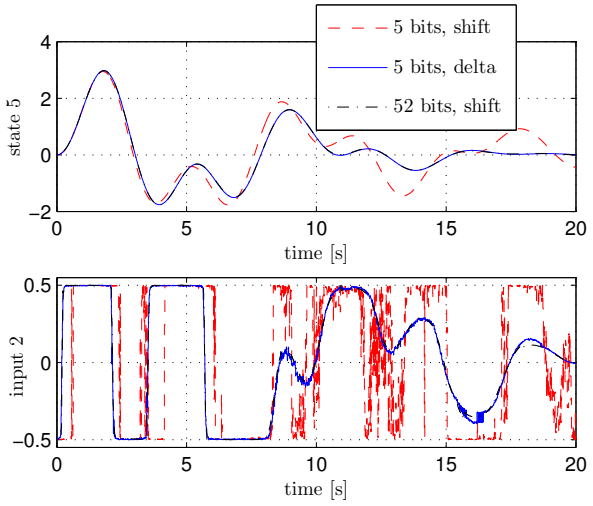

Fig. 1. Sample state and input trajectories of the closed-loop control of a benchmark spring-mass system with 3 masses, 2 inputs and sample period of 10 ms.

an equivalent discrete-time model of the continuous-time system are numerically sensitive to round-off error; an off-the-shelf optimization algorithm would not be able to detect and correct for any errors in the data, because information is lost prior to solving the optimization problem. Figure 1 shows that the closed-loop response of a 5-bit mantissa floating-point implementation of the model predictive control algorithm of Rao et al. [1998], which uses the shift form to obtain a discrete-time model, may give an unacceptable response compared to a double-precision (52-bit) implementation.

We will show how the delta operator approach of Middleton and Goodwin [1986] can be extended to develop a novel optimization problem formulation that is less susceptible to numerical errors compared to current methods. We will also outline a new optimization algorithm that pays attention to the order in which computations should be done in order to minimize the effect of numerical errors and avoid any increase in computational resources or latency. Figure 1 shows that a 5-bit mantissa floating-point implementation based on our methods produces trajectories that almost perfectly overlap with the ones from a double-precision shift implementation.

REFERENCES

J. J. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 5th edition, 2011.

E. C. Kerrigan, J. L. Jerez, S. Longo, and G. A. Constantinides. Number representation in predictive control. In *Proc. 4th IFAC Nonlinear Model Predictive Control Conference*, Noordwijkerhout, The Netherlands, August 2012. International Federation of Automatic Control.

R. H. Middleton and G. C. Goodwin. Improved finite word length characteristics in digital control using delta operators. *IEEE Trans. Automatic Control*, AC-31(11): 1015–1021, 1986.

C. V. Rao, S. J. Wright, and J. B. Rawlings. Application of interior-point methods to model predictive control. *J. Optimization Theory and Applications*, 99(3):723–757, 1998.