JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 40, No. 2, February 2017

Toward an Algorithmic Control Theory

Panagiotis Tsiotras*
Georgia Institute of Technology, Atlanta, Georgia 30332-0150
and
Mehran Mesbahi†
University of Washington, Seattle, Washington 98195-2400

DOI: 10.2514/1.G002754

WHAT does it mean to compute? For the ancient Greeks of the Pythagorean school, a "legitimate" computation required only a ruler and a compass; for the early 9th century Persian mathematician al-Khwarizmi, it was "completion" and "balancing" of equations. In the absence of infinitesimal calculus (a theory that would not become available until the late 17th century), and influenced by the Greek and Persian schools of thought,‡ most problems considered tractable/solvable for centuries involved geometry and algebra. The advent of computers§ and associated algorithms have changed all that, expanding what it means for a problem to be "solvable". Today, we do not give a second thought on whether answers provided by a computer are legitimate, even when "computing" is embedded in the operation of engineered systems. But should we?

As casual users of personal computers and calculators in our daily life, we have become accustomed to adopting the numbers spit out by these devices as the indisputable truth. We embrace them as the answer to our query, without second guessing, although we all know very well that this is a fallacy. We routinely use radicals and trigonometric expressions in our formulas but we tend to forget that we never compute them exactly. Computed values of $\sqrt{2}$ or sin 56 deg, for instance, are rational approximation surrogates of these quantities, which we take *ipso facto* to be their "true values" for all practical purposes — or not.¶ This leap of faith of what an "answer" to an algebraic equation should be is firmly grounded on the certainty and precision by which this answer is provided: the calculated value is close to its true value with an arbitrary degree of accuracy. In addition, this value can be computed deterministically, after a finite number of (known a priori) floating-point operations.

Let us now consider the task of computing a simple control action, namely, the design of a controller to generate such an action via state feedback. Feedback control design, by and large, focuses on finding a mapping $\varphi: X \to U$ that assigns to each plant state $x \in X$ a control action $u \in U$ so that the controlled plant under the action $u = \varphi(x)$ has the desired performance. In many cases, the mapping φ is required in "closed form", often as the result of a laborious offline process. The sole task of the onboard computer is to reliably *implement* this precomputed mapping. There is a price to pay, however, for restricting the search of state-feedback strategies to such "nice" closed-form mappings; to begin with, such a mapping may not even exist. The French mathematician Évariste Galois (1811–1832), for instance, in his celebrated 1846 paper "Oeuvres Mathématiques" (published posthumously after finally Liouville reviewed his manuscript and declared it to be correct) showed that no polynomial of degree five or higher has a solution that can be computed in analytic form, as a function of the polynomial coefficients.** Of course, nowadays no one considers the evaluation of the roots of a polynomial (of any order!) to be an issue. Computers have thus redefined what problems are considered solvable.

By expanding what we may consider an acceptable feedback control strategy, we therefore open the stage for a larger class of feedback mappings that are, perhaps, the result of an iterative process (i.e., an algorithm). As long as these mappings can be computed with predetermined precision after a finite number of iterations in a timely manner (very much like the calculation of $\sqrt{2}$ or sin 56 deg on a handheld calculator), these are as good as closed-form algebraic expressions from X to U.

To stress this point further, consider for a moment the case of LQR control design. If linear quadratic theory had not been introduced by R. Kalman in 1950s, but rather long before computers could efficiently solve matrix Riccati equations, it is doubtful whether the impact of the beautiful LQR/LQG theory would have been so profound. In all likelihood, it would have remained exactly that: a beautiful *theory.* Certainly, the linear quadratic problem would not have been considered solvable (for engineering purposes), at least not more so than a fifth-order polynomial would have been considered solvable in the late 19th century. Fortunately, this is not the case. The simultaneous development of reliable numerical algorithms (and computer technology) that could efficiently solve matrix Riccati equations elevated†† the theory to one of the most useful control engineering tools ever. Nonetheless, we argue that there is a lot of room for improvement.

Over the years, computers and computer algorithms have redefined and extended the meaning of "solvability" in engineering. Let us not forget that computational methods, parallelization of numerical algorithms, and new computer architectures and hardware (GPUs, multicore CPUs, FPGAs) have revolutionized many engineering disciplines in the past 40 years. Many traditional fields, such as fluid mechanics and structures, have embraced the use of computers and numerical methods many years ago. The use of computational fluid dynamics, finite elements, etc., has had an enormous impact on solving real-world problems. By comparison, control theory, by and large, has been somewhat reluctant to incorporate purely numerical methods for *deriving* (as opposed to just implementing) feedback controllers on the fly. Granted, what distinguishes control from most other disciplines is that, although computations for solving fluid and structure problems can be performed offline, (feedback) control, on the other hand, requires online computation of the respective control actions. Going from solving offline the equations for the controller gains to the online

Received 14 January 2017; revision received 15 January 2017; accepted for publication 16 January 2017; published online 6 February 2017.

*Professor, Daniel Guggenheim School of Aerospace Engineering; tsiotras@gatech.edu. Fellow AIAA.

†Professor, William M. Boeing Department of Aeronautics and Astronautics; mesbahi@aa.washington.edu. Senior Member AIAA.

‡And by a fortuitous coincidence, the Guest Editorial duo of this Special Issue reflects both cultural heritage.

§And not just digital ones. The Abacus used since Sumerian and Babylonian times and still used today in many places around the world is such a computer.

¶High-confidence avionics software avoids the calculation of elusive radicals such as $\sqrt{2}$ in their code for precisely this reason. We thank E. Feron for pointing this out.

**More precisely, the roots cannot be expressed in terms of radicals.

††A pure mathematician might say "demoted".

computation of the associated control actions (including the controller gains) brings about many challenges and necessitates a new thinking in terms of guaranteed stability and robustness, safety margins, and validation and verification. It should be stressed, however, that computational methods for online control design do not delegate rigorous mathematical analysis to the backseat. Quite the opposite; if online computations are to result in reliable control actions, significant effort has to be invested upfront, for the developing of the right type of algorithm to achieve that. Aerospace applications, in particular, leave no room for blind "trial-and-error" numerical techniques without guarantees of stability and performance.

The basic premise of this Special Issue is that a paradigm shift is needed (and is indeed happening) for control design. We argue that the time is ripe for control practitioners to incorporate more extensively *online* computer algorithms to solve real-world problems (many of which defy nice closed-form solutions) by generating control actions (with guarantees!) from real-world data, while, at the same time, ensuring acceptable performance. Control theory combined hand-in-hand with computer/computational science and engineering have recently redefined how control actions are computed from sensed data and have opened up the possibility of generating control actions on the fly for highly complex nonlinear systems. This change in philosophy will have an enormous impact on the way control system design will affect the aerospace (and not only) field in the future.

In theoretical computer science, theory of computation is a well-defined subfield that studies how efficiently problems can be solved using a mathematical abstraction of a computer, such as a Turing machine, using an algorithm. Computation is thus inexplicably connected with the notion of an algorithm, that is, a (typically iterative) sequence of steps that lead to the solution of a mathematical problem. In that sense, the topic of this special issue ("Computational Guidance and Control") could may very well have been "Algorithmic Guidance and Control" or "Algorithmic Control for Aerospace Applications". Whatever the terminology, computational (aka algorithmic) control is a novel way of thinking about online controller design, via the extensive use of algorithms and computer programs, that goes beyond the current practice of using computers only for offline controller design, followed by online controller implementation.

The striking difference of what from now on we will succinctly refer to as algorithmic control (AC) with prior practice is the absence of a control law (that is, a mathematical "recipe") that, before even implementation, has decided (once and for all!) the appropriate resulting control action for each sensory input. This is the case for either classical control (e.g., PID, lead–lag) or modern (e.g., $\mathrm{LQR}/\mathrm{LQG}/H_\infty$) control design methodologies. On the contrary, AC departs from this paradigm and delegates the control action, including the "control law" design itself, to an online algorithm during execution. The feedback control law is only implicitly materialized by the corresponding control action, and only after the data have been collected and operated upon, perhaps following some iterative process (i.e., an algorithm).

We should hasten to point out that this is not an entirely new idea. Indeed, several widely used control methodologies follow a similar philosophy and would therefore fall under the AC label, most notably model predictive control (MPC) [1] (and its surrogate, receding horizon control, RHC), as well as similar optimization-based approaches based on convex optimization solvers [2,3]. According to the MPC/RHC paradigm, for instance, the control actions are generated, at each time step t_k, via an iterative process: $u_k(t_k) = \lim_{j\to\infty} u_{k,j}$. The control sequence implemented is thus a sequence of the form:

$$u_1(t_1), u_2(t_2), \ldots, u_k(t_k), \ldots \tag{1}$$

Because each $u_k(t_k)$ is generated implicitly by this iterative process, no closed-form state-feedback expression $u(x(t))$ (i.e., control law recipe) exists such that

$$u(x(t_k)) = u_k(t_k), \qquad k = 1, 2, \ldots \tag{2}$$

The same is true for the static and dynamic output-feedback cases. Computational complexity analysis of certain classes of control synthesis problems has also been examined in the literature in the past [4,5], with particular attention to NP-hardness of certain control problems.

Similarly, convex solvers generate control actions on-the-fly by solving a convex optimization problem [2] without following the recipe of an explicit control law formula. This has been made possible recently due to powerful interior point machinery for efficiently[‡‡] solving such convex problems, with strong guarantees on convergence to the unique, global minimizer/maximizer. Hence, nowadays, for all practical purposes, convex problems are considered solvable. The fact that we do not have a closed-form solution is inconsequential and irrelevant as long as we have an algorithm to compute the control action, at each time, with convergence and precision guarantees. Feedback controllers based on the online iterative solution of convex optimization problems have recently been used with great success in challenging engineering control tasks [6].

In both previous examples, the resulting control action is the outcome of an optimization process. Is then AC different from optimization-based control design? Yes and no. It certainly encompasses optimization-based control design (as many of the articles in this Special Issue show) but, at least in our opinion, goes way beyond that.

First, optimization-based control design can be used only when the answer to the underlying optimization problem can be done online, under prespecified deadlines and strict convergence guarantees. Interestingly, the distinction between optimization and control is largely semantic and (alas!) implementation-dependent. If one has the capability of solving optimization problems fast enough on the fly to close the loop, then one has (in principle) a feedback control law. This is what MPC does. Thus, the element of time (which, mind you, is completely absent during the control algorithm development) becomes crucial for the purposes of (feedback) control. Not surprisingly then, the same algorithm can be viewed as solving an optimization or a control problem, based solely on the capabilities of the available hardware. With the continued advent of faster and more capable computer hardware architectures, the boundary between optimization and control will become even more blurred. However, when optimization is embedded in the implementation of feedback control, the classical problems of control such as robustness to model uncertainty, time delays, and process and measurement noise become of paramount importance, particularly for high-performance aerospace systems. Such topics have received relatively less attention in optimization and control.

Second, feasibility (rather than optimality) is often merely what is needed for many engineering problems. Ironically, optimization is one (the most common?) approach that we know of to obtain feasible solutions; optimality is just the serendipitous byproduct. But optimization may not be the only way to obtain feasible/implementable solutions. Evaluating a control action as a result of a weighted average of outcomes of past experiments may be far from optimal, but depending on the application, it may be an acceptable alternative because it could provide fast (re)action to complex, dynamically evolving scenarios. Machine learning uses this idea, and control design using statistical/machine-learning techniques has only recently been considered as a serious alternative to classical approaches by control theoreticians and control practitioners alike.

Third, AC often tries to incorporate discrete and continuous dynamics in the same feedback loop, thus encompassing the very rich and rapidly developing area of hybrid control systems [7]. Although computer science deals with the discrete world (over finite alphabets), control theory by and large focuses on the continuous world. The seamless integration of the two, which AC hinges upon,

[‡‡]In polynomial time; that is, a Turing machine is guaranteed to terminate within a number of steps that is a polynomial function of the size of the problem.

promises to provide new tools to solve difficult problems that would not have been possible to address using methodologies and theories originating in either one of these two worlds.

Finally, AC encompasses many topics that go beyond simple control design, which all impact system performance. For example, deep theoretical questions include computability/decidability issues [8] as well as the validation and verification of such control algorithms [9] for which classical stability analyses may be inadequate (what are the gain and phase margins for a mode-switching controller based on a finite-state automaton?). Also, online control design computations require resources (hardware, energy). How does one account for limited onboard resources? The first immediate question that comes to mind when trying to extend current control theory to account for computational limitations is how to measure computational resources. In other words, what is the "price of computation" for the purpose of control? There is not an easy answer to this question (flops, memory storage and access, information-theoretic bounds, convergence rate, sensitivity to ill-conditioning, among others, might all be appropriate for encoding the price of computation on control performance), and trying to concretize an appropriate universal metric not only could be a hopeless endeavor, but it would also obscure the essence of the problem at hand. For example, physicists and information theorists have argued that thermodynamic principles provide a powerful paradigm for measuring computational resources in physical systems in terms of information processing costs. Feynman [10] elegantly explains how ideas from statistical thermodynamics (i.e., entropy) can be used to measure the price of computation. Very little is currently known, however, about how such methodologies can be used to design controllers that are "computation and complexity aware" [11]. Needless to say, there is a fertile ground for open problems in the AC field begging for further investigation.

This Special Issue is a reflection of our conviction that recent advances in control theory and computer science provide a fertile opportunity to revisit AC in the context of aerospace guidance and control. The selected 23 papers out of the total of 62 submissions reflect some of the current trends in our community that, in our view, best represent novel intersections of aerospace guidance and control and computation. The accepted papers in this Special Issue can be roughly categorized with respect to their computational backbone (e.g., convex optimization, real-time optimal control, MPC, sampling-based, logic-based, graph-search based, using learning or parallel computation, and semi-analytic methods, among others). Yet they could also be categorized by their adopted parameterization and application-driven implementation and requirements. Adopting the first point of view, the paper by Açıkmeşe et al. and the paper by Scharf et al. aim to explore the adoption of convex optimization and custom solvers based on interior point algorithms for powered descent guidance. In the same vein, the paper by Nicotra et al. examines the adoption of convex optimization for fast reference governors. Real-time nonlinear optimal control provides the backbone of the contributions by Lu et al., Adurthi et al., Sagliano et al., Ferranti and Keviczky, De La Torre et al., Zidek and Kolmanovsky, and finally by Liu et al. All these papers examine topics ranging from entry guidance, onboard trajectory generation, predictive flight control, trajectory optimization for autonomous suspended load operations, computational optimal control with disturbance, and guidance for constrained impact. Approximate dynamic programming in the context of nonlinear adaptive control is discussed in the paper by Zhou et al., whereas MPC-based approaches with embedded convex optimization solvers are the subject of the contributions by Lee and Mesbahi, Lee et al., and Petersen et al., with applications in constrained landing, underactuated spacecraft, and constrained attitude control with reaction wheels, respectively. In the same vein, heuristic-guided numerical reachability analyses in a receding horizon setting is the focus of the contribution by Surovik and Scheeres for small-body missions. The extension of MPC in the context of model predictive path integral control is explored in the paper by Williams et al., which also examines parallel implementations; nonlinear model predictive control is the focus of the paper by Joos et al. for unmanned aerial vehicle path planning. Sampling-based approaches and computational guidance with inherently discrete, graph-based, and temporal-logic character are represented in the works by Tanygin for constrained attitude pathfinding, by Verma and Mettler for examining a learning framework for autonomous guidance, by Cowlagi and Zhang for aircraft route guidance, by Shaviv and Oshman for estimation-guided guidance, and by Starek et al. for efficient spacecraft planning. Finally, a semi-analytic approach is examined in the paper by Gerlach and Doman for ballistic airdrops.

In closing, an algorithmic point of view that advocates merging control theory with computer/computational science and engineering in a unified framework to address future challenging aerospace-related problems is the main objective of this Special Issue. It is the hope of the two Guest Editors that this Special Issue raises the awareness of the community on the available techniques (as well as the plethora of remaining problems) in this area and starts a healthy debate about the pros and cons of the numerous online computational approaches to aerospace guidance and control. If, by reading this Special Issue, the average *Journal of Guidance, Control, and Dynamics* reader is motivated to investigate further and learn more about this exciting new area, we would be content that our goal has been achieved.

References

[1] Borelli, F., Bemporad, A., and Morari, M., *Predictive Control for Linear and Hybrid Systems*, Cambridge, 2015.

[2] http://cvxr.com/cvx [retrieved 10 Dec. 2016].

[3] Boyd, S., and Vandenberghe, L., *Convex Optimization*, Cambridge Univ. Press, Cambridge, England, U.K., 2004.

[4] Blondel, V. D., and Tsitsiklis, J. N., "A Survey of Computational Complexity Results in Systems and Control," *Automatica*, Vol. 36, No. 9, 2000, pp. 1249–1274.
doi:10.1016/S0005-1098(00)00050-9

[5] Papadimitriou, C. H., and Tsitsiklis, J. N., "Intractable Problems in Control Theory," *SIAM Journal on Control and Optimization*, Vol. 24, No. 4, 1986, pp. 639–654.
doi:10.1137/0324038

[6] Blackmore, L., "Autonomous Precision Landing of Space Rockets," *The Bridge*, Vol. 4, No. 46, 2016, pp. 15–20.

[7] Goebel, R., Sanfelice, R. G., and Teel, A. R., *Hybrid Dynamical Systems: Modeling, Stability, and Robustness*, Princeton Univ. Press, Princeton, NJ, 2012.

[8] Lewis, H. R., and Papadimitriou, C. H., *Elements of the Theory of Computation*, 2nd ed., Prentice–Hall, Upper Saddle River, NJ, 1997.

[9] Roozbehani, M., Feron, E., and Megrestki, A., *Modeling, Optimization and Computation for Software Verification*, Springer, Berlin, 2005, pp. 606–622.

[10] Feynman, R. P., *Feynman Lectures on Computation*, Addison Wesley Longman, Reading, MA, 1998.

[11] Brockett, R., "Minimum Attention Control," *Proceedings of the 36th IEEE Conference on Decision and Control*, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 1997, pp. 2628–2632.