J. Nonlinear Var. Anal. 10 (2026), No. 3, pp. 661-686
Available online at http://jnva.biemdas.com
https://doi.org/10.23952/jnva.10.2026.3.08

ON THE MATHEMATICS OF THE NATURAL PHYSICS OF OPTIMIZATION

I. M. ROSS

Department of Mechanical and Aerospace Engineering, Naval Postgraduate School, Monterey, CA, USA

Dedicated to Yurii Nesterov on the occasion of his 70th birthday

Abstract. A number of optimization algorithms have been inspired by the physics of Newtonian motion. Here, we ask the question: do algorithms themselves obey some "natural laws of motion," and can they be derived by an application of these laws? We explore this question by positing the theory that optimization algorithms may be considered as some manifestation of hidden algorithm primitives that obey certain universal non-Newtonian dynamics. This natural physics of optimization is developed by equating the terminal transversality conditions of an optimal control problem to the generalized Karush/John-Kuhn-Tucker conditions of an optimization problem. Through this equivalence formulation, the data functions of a given constrained optimization problem generate a natural vector field that permeates an entire hidden space with information on the optimality conditions. An "action-at-a-distance" operation via a Pontryagin-type minimum principle produces a local action to deliver a globalized result by way of a Hamilton-Jacobi inequality. An inverse-optimal algorithm is generated by performing control jumps that dissipate quantized "energy" defined by a search Lyapunov function. Illustrative applications of the proposed theory show that a large number of algorithms can be generated and explained in terms of the new mathematical physics of optimization.

Keywords. Hamilton-Jacobi inequality; Inverse optimality; Karush/John-Kuhn-Tucker conditions; Partial guidability; Proximal aiming; Transversality conditions.

1. Introduction

For the limited purposes of introducing a key idea, we use the notation $x_f \in \mathbb{R}^n$, $n \in \mathbb{N}_+$ to be a continuous variable in the constrained optimization problem given abstractly by

$$(O_A) \begin{cases} \text{minimize} & g_0(x_f) \\ \text{subject to} & x_f \in C, \end{cases} \tag{1.1}$$

where $C \subseteq \mathbb{R}^n$ is a continuous set and $g_0 : x_f \mapsto \mathbb{R}$ is a Lipschitz-continuous function. The first-order necessary condition for Problem (O_A) is given by the (generalized) Karush/John-Kuhn-Tucker condition [13, 14, 43, 67]:

$$0 \in \nu_0 \, \partial g_0(x_f) + N_C(x_f), \tag{1.2}$$

where $\nu_0 \in \mathbb{R}_+$ is a Fritz John cost multiplier [1, 38, 49], $\partial g_0(x_f)$ is the (limiting/Mordukhovich) subdifferential [14, 43, 67] of g_0 at x_f, and $N_C(x_f)$ is the (limiting/Mordukhovich) normal cone

E-mail address: imross@nps.edu.

Received 23 September 2025; Accepted 2 November 2025; Published online 1 April 2026.

[14, 43, 67] to C at x_f. Next, we consider an optimal control problem given by

$$(\widetilde{O_B}) \begin{cases} \text{minimize} & g_0(x(t_f)) \\ \text{subject to} & \dot{x}(t) = \widetilde{f}(x(t), \widetilde{u}(t)) \\ & x(t_f) \in C \\ & x(t_0) = x^0, \end{cases} \tag{1.3}$$

where $t \in \mathbb{R}$ is an independent "time" variable, $x(t) \in \mathbb{R}^n$ is a state variable, $\widetilde{u}(t) \in \mathbb{R}^l$ is a control variable, $\widetilde{f} : \mathbb{R}^n \times \mathbb{R}^l \to \mathbb{R}^n$ is a controllable vector field that satisfies the standard hypothesis [14], and $x^0 \in \mathbb{R}^n$ is a given initial point at time $t = t_0 \leq t_f$. The unknowns in Problem $(\widetilde{O_B})$ are the state-control function pair, $t \mapsto (x(t), \widetilde{u}(t))$ and $t_f < \infty$. The terminal transversality condition for Problem $(\widetilde{O_B})$ is given by [14, 67], $\lambda(t_f) \in \widetilde{\nu}_0 \, \partial g_0(x(t_f)) + N_C(x(t_f))$, where $\lambda(t_f) \in \mathbb{R}^n$ is the final-time value of the adjoint covector and $\widetilde{\nu}_0 \in \mathbb{R}_+$ is the cost multiplier associated with (1.3). Suppose that there exists a Problem (O_B) whose extremal solution is such that

$$\lambda(t_f) = 0. \tag{1.4}$$

Then it is obvious that the following statements are true:

(1) The final-time value $x(t_f)$ of an extremal state trajectory to Problem (O_B) is a candidate solution to Problem (O_A);
(2) An extremal state trajectory $t \mapsto x(t)$ to Problem (O_B) is a continuous-time convergent "algorithm" for Problem (O_A) with x^0 as a starting point (i.e., x^0 is a guess of a solution to Problem (O_A)); and,
(3) An extremal control trajectory $[t_0, t_f] \ni t \mapsto \widetilde{u}(t)$ to Problem (O_B) generates a continuous-time convergent algorithm for Problem (O_A) via a solution to the initial value problem (IVP) given by,

$$\dot{x} = \widetilde{f}(x, \widetilde{u}(t)), \quad x(t_0) = x^0. \tag{1.5}$$

Definition 1.1 (*Hidden Algorithm Primitive for Problem* (O_A))**.** Given a point $x^0 \in \mathbb{R}^n$, let $[t_0, t_f] \ni t \mapsto x(t)$ be a solution to Problem (O_B). Then the point-to-set map,

$$x^0 \mapsto \left\{ x^0 = x(t_0), [t_0, t_f] \ni t \mapsto x(t) \right\}$$

is called a hidden algorithm primitive for Problem (O_A).

Definition 1.1 assumes the existence of Problem (O_B). The existence of this problem is given by the Transversality Mapping Principle [56, 57, 58].

Theorem 1.1 (Transversality Mapping Principle [56, 57, 58])**.** *There exists a Problem* (O_B) *whose terminal transversality condition is given by* (1.4).

Remark 1.1. By definition, a hidden algorithm primitive is convergent to a candidate optimal solution to Problem (O_A).

It is apparent that a discretization of a hidden algorithm primitive produces an algorithm. However, *it is extremely important to note at the very outset that we do not propose to produce algorithms by generating differential equations and discretizing them afterwards*; rather, it will be apparent as we develop the ideas that the eventual formulas for generating algorithms do not directly depend upon producing hidden algorithm primitives. Yet, the concept of a hidden algorithm primitive will be pervasive to the logical process of understanding the natural mathematical physics of optimization articulated in terms of a Hamilton-Jacobi theory. Once these equations are developed, we will combine the ideas of inverse optimality [26, 30] and proximal aiming [15, 16] to produce algorithms without ever generating hidden algorithm primitives.

2. A Preliminary Discussion of the Proposed Ideas

As a stand-alone concept, the transversality mapping principle inverts conventional wisdom. That is, it seems to suggest that optimal control theory can be used to solve an optimization problem instead of the other way around as is conventionally presumed [1, 38, 49]. Indeed, in the absence of additional information, this inversion seems ill-advised. All of these undesirable motivational features are acknowledged in [56] wherein the initial ideas were first formulated. However, as noted at the end of Section 1, we will never generate an extremal state trajectory $t \mapsto x(t)$ for Problem (O_B); yet, its idea will be pervasive and critical to the production of a practical algorithm for solving Problem (O_A). This is why a hidden algorithm primitive is indeed hidden.

It is critically important to note at this juncture that the central feature of the ideas introduced in Section 1 is *not* to take the limits of existing algorithms to generate ordinary differential equations (ODEs) and/or to modify the resulting ODEs to improve algorithmic efficiency. This is a subject of a large body of work [2, 3, 4, 5, 19, 20, 21, 24, 25, 29, 31, 36, 39, 40, 44, 53, 63, 64, 65, 69, 70, 72, 73] with many interesting side stories. For instance, in 1958, Gavurin [29] produced the continuous Newton-flow equation by taking the limit of Newton's method and analyzing the resulting ODE; see also Smale [63]. Over a hundred years earlier, Cauchy (according to Lemaréchal [39]) performed the opposite: he proposed the gradient method based on the continuous gradient-flow equation [19]. Recent renewed interest in using ODEs as models for optimization algorithms can be broken down into two categories: one that is focused on solving nonlinear programming problems [4, 5, 24, 36, 44, 72, 73] and the other that is largely concentrated on "accelerated" optimization algorithms for unconstrained or convex optimization problems [20, 21, 25, 31, 64, 65, 69, 70]. In many ways, the ongoing spike in interest in using ODEs to explain or improve accelerated optimization algorithms can be considered as an evolution of the early ideas [2, 3, 19, 29, 39, 53, 63] that lay somewhat dormant prior to the machine learning revolution[6, 9]. It is apparent from this brief review of the rich literature that the concepts introduced in Section 1 are sharply different from these prior works in almost all aspects. To drive home this point, we explicitly identify some of these fundamental differences as follows:

(1) The key idea of equating the first-order necessary conditions of an optimization problem to the transversality condition of an optimal control problem was first initiated in [56] and represents a marked departure from all prior works. This paper is an advancement of the theory introduced in [56].
(2) The proposed theory *never* employs taking limits of existing algorithm, as, for example, in [63, 64, 65]; rather, it is based on an independent production of algorithms without using any a priori knowledge of existing ones. This is part of the reason behind Definition 1.1.
(3) As presented in Section 1, sheer intellectual curiosity pertaining to (1.4) is the primary motivation rather than a need to produce a new theory for optimization. The fact that this

intellectual curiosity leads to a new and natural "physics" of optimization (as developed in the remainder of this paper) is an a posteriori result.

(4) It is evident that the ideas presented in Section 1 are not limited to unconstrained optimization or convex optimization or first- or second-order methods [20, 21, 25, 31, 64, 65, 69, 70]. It will be apparent later that (1.5) in the ensuing theory is merely a projection of a larger set of controllable differential equations. Hence, there is also no presumption of an order in terms of the order of an ODE.

(5) As shown in [57], the proposed theory explains the "mystery" surrounding Nesterov's accelerated gradient algorithm [50]. This explanation is not achieved by taking the limits of Nesterov's original algorithm [50] to generate ODEs [64, 65]; rather, Nesterov's algorithm is shown to be a particular consequence of (1.4) and an unconstrained version of Problem (O_B). In other words, Nesterov's algorithm is *derived* in [57] and not assumed. This derivation is based on seeking $x(\cdot) \in W^{2,\infty}([t_0,t_f],\mathbb{R}^n)$ as an engineering technique [59] to reduce the total variation of a hidden algorithm primitive. See also [60] and Section 7.1 of this paper.

(6) It will be apparent later that the control vector $\widetilde{u}$ of Problem (O_B) is a proxy for the search vector of an algorithm to solve Problem (O_A). Suppose this control vector is generated by a feedback law $\widetilde{u} = \widetilde{K}(x,t)$. Any given feedback control changes the ODE in (1.5) to a different one given by $\dot{x} = \widetilde{f}(x,\widetilde{K}(x,t))$. This idea of systematically generating ODEs is well-known in control theory [14, 62, 67] and forms the basis for algorithm generation presented in [5, 40]. These prior ideas of using control theory for optimization do not incorporate (1.4); rather, they are based on drawing analogies between control theory and optimization. Furthermore, as implied earlier, the production of algorithms as proposed in this paper will never require the simulation of $\dot{x} = \widetilde{f}(x,\widetilde{K}(x,t))$; hence, there is no discretization and propagation of an ODE.

(7) It is useful to note at this juncture that any ODE given by $\dot{x} = f(x,t)$ may be viewed as a control ODE $\dot{x} = u$ with feedback law $u = f(x,t)$.

(8) It will be apparent later that a search direction for a candidate inverse optimal algorithm is completely determined by two factors:
   (a) a selection of a state-dependent constraint set $\mathbb{U}$ for the control vector, and
   (b) a choice of a search Lyapunov function (SLF) that satisfies a Hamilton-Jacobi inequality.

   Although Lyapunov functions are widely used in optimization, the notion of an SLF is unique to this paper, albeit a generalization of the concept of a control Lyapunov function used in [56, 57, 58]. Furthermore, as shown in [56], a selection of $\mathbb{U}$ has the effect of metricizing the search space. See also Section 6 of this paper.

(9) The same ideas (outlined in Section 1) applies to both constrained and unconstrained optimization problems. In fact, one of the key strengths of the proposed theory is its universal management of constraints.

A theory that explains the collection of the preceding ideas and more is the subject of this paper.

3. A Word About Notation

There is little doubt almost all readers will be dissatisfied with the notation used in this paper. Starting with the very first equation, it is obvious that we did not use the standard notation x for the optimization variable in (1.1). Doing so would require us to use a nonstandard notation for the state variable x in (1.3). Needless to say, we have chosen in Section 1 to use the conventional optimal-control symbol x for the state variable at the price of abandoning convention in optimization. Because the same symbol across diverse disciplines have different conventional interpretations, we note upfront that the notation used in this paper, while customary in some fields, may be atypical in another. For example, because x and $f(x)$ are one of the most overused symbols, we have chosen to abandon them in favor of q for the state variable, a customary notation for generalized coordinates in physics. Similarly, we use the symbol L for the generalized Lagrangian in optimization and not in the sense of a Lagrange cost in optimal control. In the same spirit, we do not use ∇_q for the gradient operator as is customary in optimization. Instead, we use ∂_q for the gradient and ∂_q^2 for the Hessian while retaining ∂ (without a subscript) to indicate the limiting/Mordukhovich subdifferential (which we use sparingly). Other aspects of our notation are explained during their first occurrence.

4. A Hamilton-Jacobi Theory for Optimal Hidden Algorithm Primitives

As remarked in Section 2, Cauchy's inspiration for creating the gradient algorithm was the physics of a flow (motion). We briefly note that even though a gradient algorithm may be "obvious" today, its invention did not come about until more than a hundred years after Newton's method [19, 39]. Yet another concept borrowed from physics is Polyak's momentum method [53] that spawned Nesterov's accelerated gradient algorithm [50]. In other words, it is not uncommon to design optimization algorithms using the natural physics of motion. Rather than use Nature's laws of motion, here we aim to seek the more fundamental physics of optimization itself using the ideas introduced in Section 1. Because basic physics is governed by equations rather than abstract sets, we shall now limit the remainder of this paper to discussions wherein the constraint set C in (1.1) is parameterized by continuously differentiable functions. See Section 7.2 for some comments on nonsmooth optimization. In addition, per the remarks of Section 3, it will be apparent shortly that it is notationally convenient to depart from the convention of using $x \in \mathbb{R}^n$ for the unknown variable. Hence, for the remainder of this paper, we consider the optimization variable to be part of a larger set of generalized coordinates $q \in \mathbb{R}^N$, $N > n$, and relabel x to be $q_1 \in \mathbb{R}^n$. As a result, (1.1) is parameterized and reformulated as,

$$(N)\begin{cases} \underset{q_1 \in \mathbb{R}^n}{\text{minimize}} & g_0(q_1) \\ \text{subject to} & g^L \le g(q_1) \le g^U, \end{cases} \tag{4.1}$$

where $g(q_1) := (g_1(q_1), \ldots, g_m(q_1))$ is a constraint function with $m \in \mathbb{N}$ components whose values are restricted to lie between some specified lower and upper bounds given by $g^L \in \mathbb{R}^m$ and $g^U \in \mathbb{R}^m$ respectively. If any component of g^L is equal to g^U, then the problem formulation contains an equality constraint. Consequently, Problem (N) defines a general problem formulation that permits both equality and inequality constraints.

4.1. **The Natural Dynamics of Hidden Algorithm Primitives.** Using Definition 1.1, a hidden algorithm primitive for (4.1) can be articulated as a continuous-time evolution of $t \mapsto q_1(t)$ from an initial state $q_1(t_0)$ (i.e., a starting point of an algorithm) to a final state $q_1(t_f)$ where the latter is a candidate optimal point (i.e., a point that satisfies the algebraic version of (1.2) for Problem (N)).

Theorem 4.1 (Natural Dynamics of Optimization)**.** *Let $q := (q_0, q_1, q_2, q_3, q_4, q_5) \in \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}$ and $u := (u_0, u_1, u_2) \in \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m$. Define $L(q_0, q_1, q_2) := q_0 g_0(q_1) + \langle q_2, g(q_1)\rangle$ to be the generalized Lagrangian function* [1, 13, 38] *associated with the data functions of* (4.1) *that are assumed to be twice continuously differentiable. Then all hidden algorithm primitives for Problem (N) are governed by the dynamical equation, $\dot{q}(t) = F(q(t), u(t))$, where $F(\cdot,\cdot)$ is given explicitly by the right-hand-side of the following differential equation:*

$$(D)\begin{cases} \dot{q}_0(t) = u_0(t) \\ \dot{q}_1(t) = u_1(t) \\ \dot{q}_2(t) = u_2(t) \\ \dot{q}_3(t) = -\left[\partial^2_{q_1} L(q_0(t), q_1(t), q_2(t))\right] u_1(t) - \partial_{q_1} L(u_0(t), q_1(t), u_2(t)) \\ \dot{q}_4(t) = \left[\partial_{q_1} g(q_1(t))\right] u_1(t) \\ \dot{q}_5(t) = \langle \partial_{q_1} g_0(q_1(t)), u_1(t)\rangle \end{cases} \tag{4.2}$$

Proof. The proof of this theorem follows from [56] wherein a version of (4.2) is derived from first principles based on optimal control theory. Here we provide a different and shorter version of a proof but one that relies on optimization theory.

Differentiating $t \mapsto q_1(t)$ with respect to t and setting its derivative to $u_1(t)$ produces

$$\dot{q}_1(t) := u_1(t). \tag{4.3}$$

Doing the same for the cost and constraint multipliers (i.e., considering q_0 and q_2 to be functions of t and differentiating) yields

$$\dot{q}_0(t) := u_0(t), \tag{4.4}$$

$$\dot{q}_2(t) := u_2(t). \tag{4.5}$$

Define

$$\widetilde{q}_3(t) := -\partial_{q_1} L(q_0(t), q_1(t), q_2(t)). \tag{4.6}$$

Differentiating (4.6) with respect to t and using (4.3)–(4.5), we obtain

$$\begin{aligned} \dot{\widetilde{q}}_3(t) &= -\left[\partial^2_{q_1} L(q_0(t), q_1(t), q_2(t))\right] u_1(t) - \partial_{q_1} g_0(q_1(t)) u_0(t) - \left[\partial_{q_1} g(q_1(t))\right]^T u_2(t) \\ &= -\left[\partial^2_{q_1} L(q_0(t), q_1(t), q_2(t))\right] u_1(t) - \partial_{q_1} L(u_0(t), q_1(t), u_2(t)). \end{aligned}$$

Hence $\widetilde{q}_3(t)$ and $q_3(t)$ differ by a constant of motion. Similarly, by defining

$$\widetilde{q}_4(t) := g(q_1(t)) \tag{4.7}$$

and differentiating (4.7) with respect to t, we obtain $\dot{\widetilde{q}}_4(t) = \left[\partial_{q_1} g(q_1(t))\right] u_1(t)$. As a result, $\widetilde{q}_4(t)$ and $q_4(t)$ differ only by a constant of motion. Finally, we set

$$\widetilde{q}_5(t) := g_0(q_1(t)). \tag{4.8}$$

Differentiating (4.8) with respect to t generates $\dot{\widetilde{q}}_5(t) = \langle \partial_{q_1} g_0(q_1(t)), u_1(t)\rangle$, thereby implying that $\widetilde{q}_5(t)$ and $q_5(t)$ differ by a constant of motion. Thus, the dynamical system given by (4.2) describes an evolution of $t \mapsto q(t)$ when subject to a control action $t \mapsto u(t)$ where $q(t) =$

$(q_0(t), q_1(t), q_2(t), q_3(t), q_4(t), q_5(t))$ represents, up to a constant of motion, the instantaneous values of the (Fritz John) cost multiplier q_0, the optimization variable q_1, the constraint (Karush-Kuhn-Tucker) multiplier q_2, the (negative of the) gradient of the generalized Lagrangian q_3, the constraint function q_4, and the cost function q_5. Similarly, the controls $u(t) = (u_0(t), u_1(t), u_2(t))$ represent the instantaneous values of the rates of change in the cost multiplier, the optimization variable and the constraint multiplier respectively. Thus, the dynamical system (D) represents the evolution of all of the primal and dual variables and functions associated with Problem (N) and its first-order necessary conditions.

The first-order necessary condition for Problem (N), stated in terms of the final-time conditions of the dynamical system (D), is given by [1, 38],

$$q(t_f) \in T := \left\{q:\ q_0 \geq 0,\ (q_0, q_2) \neq (0,0),\ q_3 = 0,\ g^L \leq q_4 \leq g^U,\ \ q_2 \dagger q_4\right\} \tag{4.9}$$

where $\dagger$ denotes the complementarity conditions defined by [1, 38, 59],

$$q_2 \dagger q_4 \ \Leftrightarrow\ q_2^i \begin{cases} \leq 0 & \text{if} \quad q_4^i = g_i^L, \\ = 0 & \text{if} \quad g_i^L < q_4^i < g_i^U, \\ \geq 0 & \text{if} \quad q_4^i = g_i^U, \\ \textit{unrestricted} & \text{if} \quad g_i^L = g_i^U, \end{cases}$$

and the superscript $i = 1, \ldots, m$ on q_2 and q_4 denotes their i^{th} component. These necessary conditions are the Karush/John-Kuhn-Tucker conditions[1, 13, 38, 49]. Because these necessary conditions only involve the variables q_0, q_2, q_3, and q_4, it follows that all hidden algorithm primitives satisfy the dynamical equations given by (4.2). $\square$

Remark 4.1. The dynamical equations given in (4.2) are contained among the various equations presented in [56] but without the statement of Theorem 4.1. Hence, Theorem 4.1 can also be proved using the first-principles approach of [56]. However, because this first-principles' approach is lengthy, a shorter proof is presented here but one that relies on the theorems of Karush, John, Kuhn and Tucker [1, 38, 49].

Remark 4.2. It is shown in [56] that the Pontryagin Hamiltonian for Problem (O_B) is given by $\langle (q_3(t) + \partial_{q_1} L(q_0(t), q_1(t), q_2(t))),\ u_1(t)\rangle$ and that the minimized Hamiltonian vanishes along an extremal. As a result we get (4.6) with $\widetilde{q}_3(t) = q_3(t)$ for $u_1(t) \neq 0$. In the current proof of Theorem 4.1 we have simply chosen to define $\widetilde{q}_3(t)$ according to (4.6). If $\widetilde{q}_3(t)$ and hence $q_3(t)$ is defined without the negative sign in (4.6), Theorem 4.1 holds but with a corresponding change in the sign of the q_3-dynamics in (4.2). Choosing to define $q_3(t)$ with the negative sign simply maintains the consistency of Theorem 4.1 with the first-principles' results of [56]. Furthermore, the natural definition of $q_3(t)$ with the negative sign generates an asymmetric saddle point matrix when the resulting ideas are applied to address a constrained optimization problem [56]. See also Section 6. The asymmetric saddle point matrix has (under appropriate conditions [7, 8, 41]) better numerical properties than its symmetric counterpart.

In view of Theorem 4.1, we now provide a more precise definition of a hidden algorithm primitive for Problem (N) that is consistent with the use of the symbol $q \in \mathbb{R}^{2(1+n+m)}$.

Definition 4.1 (*Hidden Algorithm Primitive for Problem* (N))**.** Let $\dot{q}(t) = F(q(t), u(t))$ be defined by (4.2). Suppose there exists a (measurable) control function $t \mapsto u(t) \in \mathbb{R}^{1+n+m}$ that drives a given a point $q(t_0) = q^{00} \in \mathbb{R}^{2(1+n+m)}$ to $q(t_f) \in T$, where T is given by (4.9). Then a (Carathéodory) solution $[t_0, t_f] \ni t \mapsto q(t)$ to the IVP,

$$\dot{q} = F(q, u(t)), \quad q(t_0) = q^{00} \tag{4.10}$$

is called a hidden algorithm primitive for Problem (N).

4.2. A Hidden-Algorithm-Primitive Formulation of Optimal Optimization.

Let $\mathbb{U}(q,t) \subset \mathbb{R}^{1+n+m}$ be a compact set of allowable values of u that jointly depends on q and t. Let $t \mapsto q(t)$ be a hidden algorithm primitive generated by a control function $t \mapsto u(t) \in \mathbb{U}(q(t),t)$ acting on $\dot{q} = F(q,u)$ from some given point $q(t_0) = q^{00}$. Suppose the optimality of the resulting hidden algorithm primitive $q(\cdot)$ is determined by a cost functional defined by [59, 67],

$$J[q(\cdot), u(\cdot), t_f] := \ell(q(t_f), t_f) + \int_{t_0}^{t_f} \mathscr{L}(q(t), u(t), t)\, dt,$$

where $\ell : (q,t) \mapsto \mathbb{R}$ is a given differentiable endpoint (or Mayer) cost function and $\mathscr{L} : (q,u,t) \mapsto \mathbb{R}$ is a given differentiable running (or Lagrange) cost function. Then the optimal control problem that defines an optimal hidden algorithm primitive for Problem (N) is given by

$$(M) \left\{ \begin{array}{ll} \underset{[q(\cdot),u(\cdot),t_f]}{\text{minimize}} & J[q(\cdot), u(\cdot), t_f], \\ \text{subject to} & \dot{q}(t) = F(q(t), u(t)), \\ & u(t) \in \mathbb{U}(q(t), t), \\ & q(t_0) = q^{00}, \\ & q(t_f) \in T. \end{array} \right.$$

Remark 4.3. At this stage, it is not particularly important to define a precise functional form for ℓ and/or $\mathscr{L}$ other than the implication that they are chosen by the problem designer.

The following are standard definitions in optimal control theory [13, 59, 67].

Definition 4.2. (Pontryagin Hamiltonian) The Pontryagin Hamiltonian $H(q, \lambda, u, t)$ for Problem (M) is defined by [13, 59, 67],

$$H(q, \lambda, u, t) := \mathscr{L}(q, u, t) + \langle \lambda, F(q, u) \rangle, \tag{4.11}$$

where $\lambda \in \mathbb{R}^{2(1+n+m)}$ is a Pontryagin adjoint covector.

Definition 4.3 (Lower Hamiltonian)**.** The lower Hamiltonian $\mathscr{H}(q, \lambda, t)$ for Problem (M) is defined by [13, 59]

$$\mathscr{H}(q, \lambda, t) := \min_{u \in \mathbb{U}(q,t)} H(q, \lambda, u, t) \tag{4.12}$$

Definition 4.4. (Hamilton-Jacobi Equation) The Hamilton-Jacobi equation for Problem (M) is the partial differential equation given by [14, 67]

$$\mathscr{H}\left(q, \partial_q V(q,t), t\right) + \partial_t V(q,t) = 0 \tag{4.13}$$

where the function $V : (q,t) \mapsto \mathbb{R}$ satisfies the boundary condition, $V(q(t_f), t_f) = \ell(q(t_f), t_f)$ over all values of $q(t_f) \in T$.

Theorem 4.2 (Optimal Optimization)**.** *Let $t \mapsto q^\flat(t)$ be a hidden algorithm primitive for Problem (N) generated by the action of a control function $t \mapsto u^\flat(t)$ from the point $q(t_0) = q^{00}$. Suppose there exists a continuously differentiable function $\varphi : (q,t) \mapsto \mathbb{R}$ that satisfies*

$$\mathscr{H}\left(q, \partial_q \varphi(q,t), t\right) + \partial_t \varphi(q,t) \geq 0 \qquad \forall\, q, \forall\, t \in [t_0, t_f], \tag{4.14a}$$

$$\varphi(q(t_f), t_f) = \ell(q(t_f), t_f) \qquad \forall q(t_f) \in T, \tag{4.14b}$$

$$\varphi(q(t_0), t_0) = J[q^\flat(\cdot), u^\flat(\cdot), t_f^\flat]. \tag{4.14c}$$

Then $q^\flat(\cdot)$ is an optimal hidden algorithm primitive with optimal value given by $\varphi(q^{00}, t_0)$.

A proof of this theorem follows directly from the well-known verification theorem in optimal control [14, 67]. From Theorem 4.2, it follows that a constructive approach to obtain an optimal control and hence an optimal hidden algorithm primitive is to solve the Hamilton-Jacobi equation and use the resulting control function to solve the IVP given by (4.10) as follows:

$$u(q,t) = \arg \min_{u \in \mathbb{U}(q,t)} H\left(q, \partial_q V(q,t), u, t\right) \tag{4.15a}$$

$$\dot{q} = F(q, u(q,t)), \quad q(t_0) = q^{00} \tag{4.15b}$$

However, as explained in Section 1, our objective was not to construct an optimal hidden algorithm primitive in an explicit form; rather, our motivation was to seek out the fundamental mathematical physics of optimization. In this context, Theorem 4.2 and (4.15) accomplish this objective by establishing the fact that the natural physics of optimal optimization (via an optimal hidden algorithm primitive) is described by a Hamilton-Jacobi equation.

4.3. **Some Remarks On the Natural Physics of Optimization.** Although we have not yet fully connected a hidden algorithm primitive to an actual algorithm, some remarks on the developments of the results up to this point are in order. We begin by noting that Theorem 4.1 creates a natural vector field $F(\cdot,u)$ over the space $2(1+n+m)$ using the $(1+m)$ data functions of Problem (N) and its n optimization variables. This vector field permeates a "lifted" space $\mathbb{R}^{2(1+n+m)}$ and carries with it the information about the optimality of Problem (N). A hidden algorithm primitive is a trajectory in $\mathbb{R}^{2(1+n+m)}$. A standard guess to a solution of Problem (N) [38, 49] is defined by $q_0(t_0) = 1$, $q_1(t_0) = q_1^{00}$, and $q_2(t_0) = q_2^{00}$, where q_1^{00} and q_2^{00} are guesses of the optimization variable and the constraint multiplier respectively. Consequently, an initial point $q^{00} \in \mathbb{R}^{2(1+n+m)}$ for Problem (M) and hence (4.15b) may be generated according to the following construction:

$$\begin{aligned} &q_0(t_0) = 1, \quad q_1(t_0) = q_1^{00}, \quad q_2(t_0) = q_2^{00}, \\ &\qquad q_3(t_0) = -\partial_{q_1} L\left(1, q_1^{00}, q_2^{00}\right), \quad q_4(t_0) = g\left(q_1^{00}\right), \quad q_5(t_0) = g_0(q_1^{00}). \end{aligned} \tag{4.16}$$

Any feasible control function $u(\cdot)$ drives the given initial point q^{00} to a candidate optimal point $q(t_f) \in T$ by sensing the instantaneous values of the derivatives of the data functions of Problem (N) via the vector field $F(\cdot,u)$. The minimization of the Pontryagin Hamiltonian is a local action that generates an instantaneous value of the control action predicated on the knowledge of the transversality condition. Thus, a continuous concatenation of the local control action delivers the global result of optimal optimization wherein the optimality criterion is specified by the functional $J[q(\cdot), u(\cdot), t_f]$. A main theoretical challenge at this stage is to develop meaningful measures of optimality for optimal optimization. For example, one could conceive the total variation of $q(\cdot)$ as a valid measure of optimality for a hidden algorithm primitive. Thus, a potentially viable path to advance the preceding ideas is to design desirable optimality criteria for optimal optimization. In the next section we embark on a different path toward optimality by way of inverse optimality [26, 30]. It will be apparent shortly that this approach generates a number of near-term practical results.

5. A Theory for Generating Inverse Optimal Algorithms

From Section 4, it follows that the Hamilton-Jacobi framework constitutes a theoretical foundation for optimal hidden algorithm primitives. To migrate this idea to a practical algorithmic framework, we jointly employ the notion of inverse optimal control [26, 30] and proximal aiming [14, 15]. It will be apparent shortly that these two ideas obviate the need to solve the Hamilton-Jacobi equation or directly produce a hidden algorithm primitive although both constructs are essential to the development of a theory for generating inverse-optimal algorithms.

5.1. **Inverse Optimal Hidden Algorithm Primitives.**

In inverse optimal control, one flips the roles of the functional $J[q(\cdot),u(\cdot),t_f]$ and the value function $V(q,t)$ in (4.13). That is, instead of solving for $V(q,t)$ for a given functional $J[q(\cdot),u(\cdot),t_f]$, one selects $V(q,t)$ a priori and determines $J[q(\cdot),u(\cdot),t_f]$ afterwards. This role reversal is made meaningful by the notion that a selection of $V(q,t)$ parameterizes a family of control functions $u(\cdot)$ that indirectly minimizes some cost functional $J[q(\cdot),u(\cdot),t_f]$ [26, 32]. Furthermore, because the task of a control in (4.2) is to always drive any given point $q(t_0) = q^{00}$ to $q(t_f) \in T$, a hidden algorithm primitive can be reframed in terms of stabilizability (with respect to T) of the dynamical system (D). Under this concept, $V(q,t)$ takes the role of a control Lyapunov function [14, 26, 62]; i.e., a generalization of the classical Lyapunov function. Thus the link between stabilizability and optimality is that a control Lyapunov function is a meaningful value function [26].

5.1.1. *Partial Stabilizability and Guidability.*

The stabilizability requirements for (D) are substantially weaker than those frequently encountered in control theory. For instance, we are only interested in partial stabilizability [18, 32, 35, 68, 74] because the target set T does not directly involve q_1 or q_5. In view of this particular requirement, we split (D) into two dynamical systems,

$$(A)\begin{cases} \dot{q}_0(t) = u_0(t), \\ \dot{q}_2(t) = u_2(t), \\ \dot{q}_3(t) = -\left[\partial^2_{q_1} L(q_0(t),q_1(t),q_2(t))\right] u_1(t) - \partial_{q_1} L(u_0(t),q_1(t),u_2(t)), \\ \dot{q}_4(t) = \left[\partial_{q_1} g(q_1(t))\right] u_1(t), \end{cases} \tag{5.1a}$$

$$(B)\begin{cases} \dot{q}_1(t) = u_1(t), \\ \dot{q}_5(t) = \langle \partial_{q_1} g_0(q_1(t)), u_1(t)\rangle, \end{cases} \tag{5.1b}$$

where only (A) needs to be stabilizable with respect to T while the trajectories of (B) must be merely bounded. In other words, the optimization variable q_1 must be bounded and the cost function q_5 must be bounded from below, a standard assumption in optimization [1, 38, 49]. Furthermore, the weak requirements of partial stabilizability of (D) are even weaker for optimization than typical cases considered in control theory [32, 68, 74]. This is because, for the purposes of optimization, we do not seek partial stabilizability for disturbance rejection or robust design or other common criterion used in control theory [16, 26, 62]. Hence, we do not require $u(\cdot)$ to be smooth or be explicitly defined in terms of a feedback law, $u = K(q,t)$. In principle, an open-loop control suffices. All of these weak requirements imply Clarke's notion of guidability [16] suffices in terms of generating hidden algorithm primitives. That is, from the point of view of solving Problem (N) via the dynamical system (D), we only need to partially guide the state q to T rather than keep it close to T when the initial condition is already near the target. The latter is the stability issue while the former is the (partial) guidability problem [16].

5.1.2. *Search Lyapunov Functions.* In view of all of the weak requirements discussed in the preceding paragraph, we define a search Lyapunov function (SLF) $S : (q,t) \mapsto \mathbb{R}_+$ as a control Lyapunov function associated with the partial guidability to T of the dynamical system (D) (i.e, guidability of the dynamical system (A) with respect to the set T). Hence, following the results for Lyapunov functions presented in [14, 16, 18, 32, 74], we define an SLF as follows:

Definition 5.1. A continuous function $S : (q,t) \to \mathbb{R}_+$ is called an SLF for the pair $\big((D), T\big)$ if there exists a continuous rate function $R : (q,t) \to \mathbb{R}_+$ and a class $\mathscr{K}$ function [18, 68] $\beta : \mathbb{R}_+ \to \mathbb{R}_+$ with respect to T such that the following conditions of positive semidefiniteness, growth and infinitesimal decrease hold:

$$S(q,t) = 0 \iff q \in T, \quad R(q,t) = 0 \iff q \in T \tag{5.2a}$$

$$S(q,t) \geq \beta(\|q_a\|) \quad \forall\, q \notin T,\ \forall\, t \geq t_0 \tag{5.2b}$$

$$\exists u \in \mathbb{U}(q,t) \text{ such that } DS(q,t;F(q,u)) \leq -R(q,t) \quad \forall\, q \notin T,\ \forall\, t \geq t_0, \tag{5.2c}$$

where $q_a := (q_0, q_2, q_3, q_4)$, $\beta(\cdot)$ is a continuously increasing unbounded function satisfying the condition $\beta(\|q_a\|) = 0$ for all $q_a \in T$ and $DS(q,t;F(q,u))$ is a generalized directional derivative of $S(q,t)$ along the direction $\big(F(q,t), 1\big)$.

For a continuously differentiable SLF, $DS(q,t;F(q,u))$ is given by,

$$DS(q,t;F(q,u)) := \langle \partial_q S(q,t), F(q,u) \rangle + \partial_t S(q,t). \tag{5.3}$$

If the SLF is not continuously differentiable, then $DS(q,t;F(q,u))$ in (5.2c) is an appropriately generalized directional derivative based on the regularity of the SLF [14, 17, 52]. Of these generalized directional derivatives, the Dini derivate [14, 52] turns out to be one of the most useful. See Section 7.2 for further discussions.

Remark 5.1. Not all requirements of an SLF indicated in (5.2) are needed in all situations. For instance, for a continuously differentiable SLF, the rate constraint in (5.2c) is not necessary [17, 18]. However, if the Bhat-Bernstein rate constraint [10, 32, 54] is imposed on a continuously differentiable SLF, then convergence can be achieved in finite time, which may be particularly useful for optimization [60]. On the other hand, additional requirements on the SLF must be imposed for other types of partial- stability/guidability requirements such as partial global asymptotic guidability, partial uniform Lyapunov stability etc. [17, 18, 32, 35, 52, 68, 74]. We deliberately avoid discussing the myriad notions of partial- stability/guidability and the corresponding requirements they impose on the SLF in order to focus on using the most basic set of ideas that are necessary for the exclusive purposes of optimization.

Substituting (5.3) in (5.2c) (and, hence, disregarding the rate constraint) we get the other side of the Hamilton-Jacobi inequality (Cf. (4.14a)),

$$\partial_t S(q,t) + \min_{u \in \mathbb{U}(q,t)} \langle \partial_q S(q,t), F(q,u) \rangle < 0 \quad \forall\, q \notin T. \tag{5.4}$$

Comparing (5.4) with (4.11), (4.12) and (4.13) the inverse-optimality argument follows from the existence of some running cost $\mathscr{L}(q(t),u(t),t) > 0$ that is minimized by the choice of $S(q,t)$ [26, 32].

5.1.3. *Process for Generating an Inverse-Optimal Hidden Algorithm Primitive.* Collecting all of the key ideas, the generation of an inverse-optimal hidden algorithm primitive can be summarized as follows:

(1) Given an optimization Problem (N) generate the dynamical system (D) (and hence (A) and (B)). This step is straightforward and only requires the problem data functions and their derivatives.
(2) Select a search Lyapunov function $S : (q,t) \mapsto \mathbb{R}_+$.
(3) Choose $\mathbb{U}(q,t)$, a compact set of allowable values for u.
(4) Solve (5.4) for $u(q,t)$.
(5) Solve the IVP given by (4.15b).

Remark 5.2. The selection of $S(q,t)$ and $\mathbb{U}(q,t)$ in Steps 2 and 3 may be interdependent because of the requirement stipulated by (5.2c).

5.2. **Inverse Optimal Algorithm Generators.** From Section 5.1, it follows that the notion of inverse optimality allows us to escape the problem of solving the Hamilton-Jacobi equation for optimal optimization (via a hidden algorithm primitive). We now use a procedure inspired by proximal aiming [14, 15] and practical stabilizability [11, 16, 45] to escape the problem of generating a hidden algorithm primitive as a means to produce an algorithm for solving Problem (N). We briefly note that this escape hatch still requires the notion of a hidden algorithm primitive.

5.2.1. *Practical Guidability and Polygonal Arcs.* From the point of view of convergence of an optimization algorithm, it is not necessary for $q(t_f)$ to solve a given problem exactly. That is, it suffices for $q(t_f)$ to be within an arbitrarily specified ε-ball around T. In fact, in most practical cases implemented on a digital computer, the value of $\varepsilon > 0$ cannot be smaller than $\sqrt{\varepsilon_M}$, where ε_M is the machine precision. Regardless, the concept of $q(t)$ approaching T within an $\varepsilon > 0$ value is known in control theory as "practical" stabilization [11, 17, 45, 52]. Obviously, this weaker concept of stabilizability provides the right fit for an optimization algorithm versus other stronger notions such as continuous asymptotic controllability.

One approach used in control theory for practical guidability is to produce a feedback control $u = K(q,t)$ in the sample-and-hold sense [17, 52]. That is, a control is sampled at time t_k and held a constant value $K(q_k,t_k)$, $q_k = q(t_k)$ up to time $t_{k+1} > t_k$. In between the sample points, $[t_{k+1},t_k] \ni t \mapsto q(t)$ is a solution to the differential equation $\dot{q}(t) = F(q(t), K(q_k,t_k))$ with initial condition $q(t_k) = q_k$. A direct application of this control-theoretic idea for the purposes of optimization would produce an inverse-optimal hidden algorithm primitive in the sample-and-hold sense. However, *from an algorithmic perspective, the value of $q(t)$ in between the sample points is totally irrelevant*. In fact, even the precise location of the points, $q(t_k)$, are also irrelevant (except at $t = t_f$). This is simply because the goal of a convergent algorithm is to merely arrive at a point in T. Hence, we are not interested in a point-wise accurate, discrete-time solution to $\dot{q} = F(q, K(q,t))$ [2, 3]. Because of this unique requirement of an optimization algorithm, conventional discretization methods for ODEs [33] are not necessarily relevant to the production of efficient algorithms. The only requirement for a convergent algorithm generator is to produce a sequence $q(t_0), q(t_1), \ldots$ that accumulates near T within an ε tolerance with no burden to track any solution of the flow resulting from $\dot{q} = F(q, K(q,t))$. Hence, if we were to somehow jump from $q(t_k)$ to $q(t_{k+1})$, $k = 0, 1, \ldots$, the resulting polygonal arc would directly produce an (inverse-optimal) algorithm without ever passing through the process of generating a hidden algorithm primitive. We achieve these controlled jumps by large discrete decrements of the SLF.

5.2.2. *Discrete Decrements of an SLF.* An inverse-optimal algorithm uses an SLF to produce a sequence of strictly decreasing numbers,

$$S(q(t_0), t_0) > S(q(t_1), t_1) > \cdots S(q(t_k), t_k) > S(q(t_{k+1}), t_{k+1}) \cdots \tag{5.5}$$

so that (Cf. (5.2))

$$\lim_{k \to \infty} S(q(t_k), t_k) = 0. \tag{5.6}$$

Hence, the limit of the sequence $q(t_0), q(t_1), \cdots$ resulting from (5.5) is (Cf. (5.2a)) a solution to Problem (N):

$$\lim_{k \to \infty} q(t_k) := q_\infty \in T. \tag{5.7}$$

Lemma 5.1. *Let $S : (q,t) \to \mathbb{R}_+$ be a continuously differentiable SLF. Let $u(t_k) = u$ be a solution to the problem,*

$$(M_u) \begin{cases} \underset{u}{\textit{minimize}} & DS(q(t_k), t_k; F(q(t_k), u)) \\ \textit{subject to} & u \in \mathbb{U}(q(t_k), t_k) \end{cases}$$

at any point $(q(t_k), t_k)$. Assume $q(t_k) \notin T$. Then there exists a sequence t_k, $k = 0, 1, \ldots$, $t_{k+1} > t_k$ and a polygonal arc given by

$$q(t_{k+1}) = q(t_k) + F(q(t_k), u(t_k))(t_{k+1} - t_k) \tag{5.8}$$

such that (5.7) holds.

Proof. By definition of an SLF, at any $t = t_k$, there exists a sufficiently small time interval $\delta t > 0$ such that

$$S(q(t + \delta t), t + \delta t) - S(q(t), t) = DS\big(q(t), t; F(q(t), u(t_k))\big)\, \delta t < 0 \tag{5.9}$$

Setting $t + \delta t$ to t_{k+1} in (5.9), we obtain

$$S(q(t_{k+1}), t_{k+1}) < S(q(t_k), t_k) \quad k = 0, 1, \ldots \tag{5.10}$$

Similarly, from Theorem 4.1, it follows that there exists a sufficiently small interval $\delta t > 0$ around $t = t_k$ such that

$$q(t + \delta t) - q(t) = F(q(t), u(t_k))\delta t \tag{5.11}$$

Setting $t + \delta t$ to t_{k+1} in (5.11), we get (5.8). Since $u(t_k)$ decrements the SLF at each t_k, we obtain (5.6) from the property of an SLF. Hence, (5.7) follows. □

Lemma 5.2. *Let the assumptions of Lemma 5.1 hold. Let $u(k)$ be a solution to Problem (M_u). Let $h_k = h$ be a solution to the following problem:*

$$
(M_h) \begin{cases} \underset{(h>0,\ q(k+1))}{\textit{minimize}} & S(q(k+1), t_k + h), \\ \textit{subject to} & q_0(k+1) - q_0(k) - h\, u_0(k) = 0, \\ & q_1(k+1) - q_1(k) - h\, u_1(k) = 0, \\ & q_2(k+1) - q_2(k) - h\, u_2(k) = 0, \\ & q_3(k+1) + \partial_{q_1} L(q_0(k+1), q_1(k+1), q_2(k+1)) = 0, \\ & q_4(k+1) - g(q_1(k+1)) = 0, \\ & q_5(k+1) - g_0(q_1(k+1)) = 0. \end{cases}
$$

Then $S(q(k+1), t_k + h_k)$ is maximally decremented in the direction $DS\big(q(t), t; F(q(t), u(k))\big)$.

Proof. From the proof of Theorem 4.1, it is obvious that the dynamical system (D) has the following three integrals of motion:

$$
\begin{aligned} q_3(t) + \partial_{q_1} L(q_0(t), q_1(t), q_2(t)) &= 0, \\ q_4(t) - g(q_1(t)) &= 0, \\ q_5(t) - g_0(q_1(t)) &= 0. \end{aligned} \tag{5.12}
$$

Hence, all evolutions of $\dot{q} = F(q,u)$ lie on the hypersurface defined by (5.12). Setting $t = t_k + h$ in (5.12), it follows that any point $q(k+1)$ that satisfies the equation,

$$
\begin{aligned} q_3(k+1) + \partial_{q_1} L(q_0(k+1), q_1(k+1), q_2(k+1)) &= 0, \\ q_4(k+1) - g(q_1(k+1)) &= 0, \\ q_5(k+1) - g_0(q_1(k+1)) &= 0, \end{aligned}
$$

is on the hypersurface defined by (5.12). These hypersurface constraints constitute the last three constraints that define Problem (M_h). Define

$$
\begin{aligned} q_0(k+1) &:= q_0(k) + h\, u_0(k) \\ q_1(k+1) &:= q_1(k) + h\, u_1(k) \\ q_2(k+1) &:= q_2(k) + h\, u_2(k) \end{aligned} \tag{5.13}
$$

Then, for sufficiently small values of $h > 0$ in (5.13), $q(k+1)$ satisfies the conditions for Lemma 5.1 to hold. As a result, we have $S(q(k+1), t_k + h) < S(q(k), t_k),\ k = 0, 1, \ldots$ for a sufficiently small value of $h > 0$ in Problem (M_h). Furthermore, for any $h > 0$ we have $S(q(k+1), t_k + h) \geq 0$ (by definition). Hence, a solution to Problem (M_h) produces a value of $h = h_k$ that generates the maximum possible decrement of $S(q(k+1), t_k + h_k)$ in the direction $DS\big(q(t), t; F(q(t), u(k))\big)$. □

Remark 5.3. From Lemma 5.2, it follows that the sequence $q(0), q(1), \ldots$ is a polygonal arc that does not necessarily track a hidden algorithm primitive. Furthermore, it is apparent that h generated by solving Problem (M_h) may not satisfy any of the established consistency conditions for discretizations of differential equations [33].

Remark 5.4. At first glance, Problem (M_h) appears similar to an exact line search algorithm [1, 38, 49]; however, there are three sharp differences between the two:

(1) Problem (M_h) is a constrained optimization problem;

(2) The objective function in Problem (M_h) is an SLF; and
(3) The global minimum of an SLF is zero (Cf. (5.2a)).

5.2.3. *An Inverse Optimal Algorithm Generator.* An inverse optimal algorithm essentially comprises two steps. In the first step, Problem (M_u) is solved to produce $u = u(k)$. This value of $u(k)$ is used in second step to generate $h = h_k$ and thus the next iterate $q(k+1)$.

Theorem 5.1. *Let $u(k)$ and h_k be solutions to Problems (M_u) and (M_h) respectively. Let $q(0)$ be given by $q(t_0)$ defined in (4.16). Assume $q(0) \notin T$. Let $q(k+1)$ be given by the iterative map,*

$$\begin{aligned} q_0(k+1) &\leftarrow q_0(k) + h_k\, u_0(k), \\ q_1(k+1) &\leftarrow q_1(k) + h_k\, u_1(k), \\ q_2(k+1) &\leftarrow q_2(k) + h_k\, u_2(k), \\ q_3(k+1) &\leftarrow -\partial_{q_1} L(q_0(k+1), q_1(k+1), q_2(k+1)), \\ q_4(k+1) &\leftarrow g(q_1(k+1)), \\ q_5(k+1) &\leftarrow g_0(q_1(k+1)), \end{aligned} \tag{5.14}$$

for $k = 0, 1, \ldots$. Then $\lim_{k\to\infty} q(k)$ converges to a solution of Problem (N).

Proof. The proof of this theorem follows directly from Lemmas 5.1 and 5.2. □

5.2.4. *A Practical Inverse Optimal Algorithm Generator.* The objective function in Problem (M_u) is linear in u. Hence, the ease (or difficulty) in solving Problem (M_u) is primarily dictated by the choice of $\mathbb{U}(q,t)$. For example, if $\mathbb{U}(q,t)$ is convex for all (q,t) then Problem (M_u) is a convex optimization problem. Hence a practical inverse optimal algorithm is predicated on the fact that Problem (M_u) is an easier problem to solve than Problem (N). Because $q(k+1)$ can be computed explicitly from $q(k)$, Problem (M_h) may be considered to be a univariate minimization problem. Furthermore, because the global minimum of an SLF is zero, one possible choice for a guess of $h = h^0$ to initiate a solution to Problem (M_h) is given by [60],

$$h^0 = \frac{S(q(k), t_k)}{-DS(q(k), t_k; F(q(k), u(k))}. \tag{5.15}$$

For much the same reason as why practical optimization algorithms do not solve the exact line search problem, a practical inverse optimal algorithm may solve Problem (M_h) approximately. In this case, (5.15) may also be used as a starting point to solve Problem (M_h) approximately and backtrack when necessary [60]. Furthermore, because the three data points, $S(q(k), t_k)$, $DS(q(k), t_k; F(q(k), u(k))$ and $S(q(k+1), t_k + h^0)$ are available, backtracking may also be performed via minimizing Hermite interpolations of $t \mapsto S(q(t), t)$. In other words, similar to a practical line-search algorithm, one may choose to solve Problem (M_h) that is consistent with the idea that $S(q(k+1), t_k + h)$ has decreased sufficiently. Note also that an SLF does not necessarily prevent the cost function $g_0(q_1(k+1))$ from increasing in value; hence, Problem (M_h) naturally allows "hill climbing" with respect to the cost function (provided the value of the SLF decreases at the next iterate).

Collecting all of the preceding ideas, the key steps of a practical inverse optimal algorithm generator can be summarized as follows:

Step 0 (a) Using the data functions of a given Problem (N), generate its dynamical system (D) (Cf. Theorem 4.1).

(b) Select an SLF $S(q,t)$ together with a compact control set $\mathbb{U}(q,t)$ such that Problem (M_u) is easy to solve (i.e., easier than Problem (N)).

(c) Choose a small value for $\varepsilon > 0$. Set $k = 0$. Use (4.16) to evaluate $q(k) = q(t_0)$.

Step 1 If $S(q(k),t_k) \le \varepsilon$ stop and exit. Else, go to Step 2.

Step 2 Solve the (easy) Problem (M_u) to produce $u(k)$.

Step 3 Using $u(k)$ solve Problem (M_h) approximately to produce h_k.

Step 4 Advance to the next iterate using (5.14). Set $k \leftarrow k+1$ and go to Step 1.

Remark 5.5. The stopping criterion in Step 1 of the inverse optimal algorithm uses the fact that the SLF vanishes at a candidate optimal point (Cf. (5.2a)). From this perspective, an SLF may also be used as a measure of "distance" to optimality. Obviously, these ideas are totally different from the concept of Bregman divergence and the ensuing results developed in [21, 69, 70].

Remark 5.6. From the totality of steps outlined in its production, it is clear that a hidden algorithm primitive is never generated although its concept was crucial to the development of a practical inverse optimal algorithm.

In summary, a practical inverse optimal algorithm generator uses a blend of old and new ideas with the transversality mapping principle providing the spark for an investigation of optimization via optimal control theory.

5.3. **Two Basic Minimum Principles for Generating Algorithms.** To limit the scope of the remainder of this paper, we now consider a narrower class of SLFs that do not depend on all of q and t. These types of SLFs can be designed by recalling that only the variables of the dynamical system (A) need to be guided to T. In view of this observation, we simplify the notation for (A) and rewrite it as $\dot{q}_a = f(q_0,q_1,q_2,u)$, where $q_a := (q_0,q_2,q_3,q_4)$ as before and $f(\cdot)$ is the right-hand-side of (5.1a). We now define an SLF that depends only on q_a:

$$\begin{aligned}
&S(q_a) = 0 \iff q_a \in T,\\
&S(q_a) > 0 \quad \forall\, q_a \notin T,\\
&\exists u \in \mathbb{U}(q,t) \text{ such that } \langle \partial_{q_a} S(q_a), f(q_0,q_1,q_2,u)\rangle < 0 \quad \forall\, q_a \notin T, \quad \forall\, t \ge t_0.
\end{aligned}$$

Under this assumption, Problem (M_u) simplifies to solving the linear-cost minimization problem,

$$(P)\begin{cases} \underset{u\in\mathbb{R}^{1+n+m}}{\text{minimize}} & \pounds_f S := \langle \partial_{q_a} S(q_a), f(q_0,q_1,q_2,u)\rangle,\\ \text{subject to} & u \in \mathbb{U}(q,t),\end{cases} \tag{5.16}$$

where the symbol $\pounds_f S$ is used to denote the fact that $\langle \partial_{q_a} S(q_a), f(q_0,q_1,q_2,u)\rangle$ is the Lie derivative of S along the vector field f.

As noted earlier, in certain cases, it might be more useful to specify a minimum rate of descent for $\pounds_f S$. Let $R : (q,t) \mapsto \mathbb{R}_+$ be such a function that satisfies $R(q,t) = 0 \Leftrightarrow q_a \in T$. This implies we require u to satisfy the constraint,

$$\pounds_f S + R(q,t) \le 0. \tag{5.17}$$

Let $Q : (u,q,t) \mapsto \mathbb{R}$ be some objective function such that a solution to the following problem exists

$$(P^*)\begin{cases} \underset{u\in\mathbb{R}^{1+n+m}}{\text{minimize}} & Q(u,q,t),\\ \text{subject to} & \pounds_f S + R(q,t) \le 0.\end{cases} \tag{5.18}$$

Then Problem (P^*) may be considered as an alternative to Problem (P) to determine u [60]. Note that the constraint in Problem (P^*) is linear and merely one-dimensional.

In the next section, we use the minimum principles given by (P) and (P^*) to generate algorithms for an illustrative class of problems.

6. Illustrative Examples

Consider the fundamental problem of solving an equality-constrained optimization problem given by

$$(N_0)\begin{cases}\underset{q_1\in\mathbb{R}^n}{\text{minimize}} & g_0(q_1)\\ \text{subject to} & g(q_1)=0,\end{cases}$$

where $g(q_1) := (g_1(q_1),\ldots,g_m(q_1))$ as before. The standard Lagrangian for this problem is given by $L(q_1,q_2) := g_0(q_1)+\langle q_2, g(q_1)\rangle$. Suppose we set $q_0(t)=1$ for all t with $u_0(t)=0\ \forall t$. Then the target set $T=T_0$ for Problem (N_0) simplifies to (Cf. (4.9))

$$T_0 := \{q:\ \ q_3=0,\ \ q_4=0\}. \tag{6.1}$$

Because there are no final-time conditions on q_2, the q_2-dynamics belong to the (B) system (Cf. (5.1)); hence, we split the dynamical system (D) (for Problem (N_0)) according to

$$(A_0)\begin{cases}\dot q_3(t) = -\left[\partial^2_{q_1}L(q_1(t),q_2(t))\right]u_1(t)-\left[\partial_{q_1}g(q_1(t))\right]^T u_2(t),\\ \dot q_4(t)=\left[\partial_{q_1}g(q_1(t))\right]u_1(t),\end{cases} \tag{6.2a}$$

$$(B_0)\begin{cases}\dot q_1(t)=u_1(t),\\ \dot q_2(t)=u_2(t),\\ \dot q_5(t)=\langle\partial_{q_1}g_0(q_1(t)),u_1(t)\rangle,\end{cases} \tag{6.2b}$$

and define $q_a := (q_3,q_4)$ with $f(\cdot)$ given by the right-hand-side of (6.2a). From Section 5.3, it follows that a simple and tentative choice of an SLF for the partial guidability of the pair (A_0,B_0) is the quadratic given by,

$$S(q_3,q_4) := \frac{1}{2}\langle q_3,q_3\rangle+\frac{1}{2}\langle q_4,q_4\rangle. \tag{6.3}$$

Application of the minimum principle (P) given by (5.16) generates the linear-cost minimization problem,

$$(P_0)\begin{cases}\underset{(u_1,u_2)\in\mathbb{R}^n\times\mathbb{R}^m}{\text{minimize}} & -\langle q_3,\left[\partial^2_{q_1}L(q_1,q_2)\right]u_1+\left[\partial_{q_1}g(q_1)\right]^T u_2\rangle+\langle q_4,\left[\partial_{q_1}g(q_1)\right]u_1\rangle\\ \text{subject to} & (u_1,u_2)\in\mathbb{U}(q,t).\end{cases}$$

Next, we consider the family of quadratic sets for $\mathbb{U}(q,t)$ given by

$$\mathbb{U}(q,t)=\left\{(u_1,u_2):\ [u_1,u_2]^T W(q,t)\begin{bmatrix}u_1\\u_2\end{bmatrix}\le\Delta(q,t)\right\}, \tag{6.4}$$

where $W(q,t)$ is an $(n+m)\times(n+m)$ symmetric positive definite matrix that metricizes the search (control) space and $\Delta(q,t)>0$ is some finite number that may depend on q and/or t.

Remark 6.1. The quantity $\Delta(q,t)$ in (6.4) is similar to a trust region parameter if $W(q,t)$ is taken to be the identity matrix. Nonetheless, it is still different from a trust region because the optimization problem (P_0) has no solution if u is strictly inside $\mathbb{U}(q,t)$.

Solving Problem (P_0) with $\mathbb{U}(q,t)$ given by (6.4) yields,

$$W(q,t)\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \sigma \begin{bmatrix} \partial^2_{q_1} L(q_1,q_2) & -\left[\partial_{q_1} g(q_1)\right]^T \\ \partial_{q_1} g(q_1) & 0 \end{bmatrix} \begin{bmatrix} q_3 \\ q_4 \end{bmatrix}, \qquad \sigma > 0 \tag{6.5}$$

where $(1/\sigma)$ is a multiplier associated with the control constraint. Deferring a discussion on the selection of $W(q,t)$, a particular class of inverse optimal algorithms for solving Problem (N_0) can be defined as follows:

(1) Using (4.16) evaluate the SLF given by (6.3). If this value of the SLF is within a specified $\varepsilon > 0$ tolerance, stop and exit. Else, set the counter $k = 0$ and continue.
(2) Solve Problem (P_0) to produce $u(k) = (u_1, u_2)$. (Cf. (6.5).)
(3) Solve Problem (M_h) (approximately) to generate h_k. (See also (5.15).)
(4) Evaluate (6.3) at $q(k+1)$ using (5.14). If $S(q_3(k+1), q_4(k+1)) \leq \varepsilon$ stop and exit. Otherwise set $k \leftarrow k+1$ and go to Step 2.

It is apparent from this procedure that a differential equation is not produced and solved to generate an inverse-optimal algorithm.

6.1. **Choosing** $W(q,t)$**, Part I.** The block 2×2 matrix on the right-hand side of (6.5) is the asymmetric KKT matrix [7, 41, 49]. This equation can be easily transformed to produce the symmetric KKT matrix by changing the sign of the q_3 coordinate,

$$\bar{q}_3 := -q_3, \quad \bar{q} := (q_1, q_2, \bar{q}_3, q_4, q_5). \tag{6.6}$$

Using (6.6), we have the transformed version of (6.5) as

$$W(\bar{q},t)\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = -\sigma \begin{bmatrix} \partial^2_{q_1} L(q_1,q_2) & \left[\partial_{q_1} g(q_1)\right]^T \\ \partial_{q_1} g(q_1) & 0 \end{bmatrix} \begin{bmatrix} \bar{q}_3 \\ q_4 \end{bmatrix}, \qquad \sigma > 0. \tag{6.7}$$

Proposition 6.1 ([56])**.** *Let $K_S(q_1,q_2)$ denote the block 2×2 symmetric matrix on the right-hand side of (6.7). Assume this saddle-point matrix is nonsingular along the iterates of q. Suppose we choose $W(\bar{q},t)$ to be the square of $K_S(q_1,q_2)$,*

$$W(\bar{q},t) := K_S^2(q_1,q_2). \tag{6.8}$$

Then the resulting inverse-optimal algorithm for solving Problem (N_0) is equivalent to a sequential quadratic programming (SQP) method with (6.7) constituting a (damped) Newton method for solving the QP subproblem.

Proof. Substituting (6.8) into (6.7), we have

$$\begin{bmatrix} \partial^2_{q_1} L(q_1,q_2) & \left[\partial_{q_1} g(q_1)\right]^T \\ \partial_{q_1} g(q_1) & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = -\sigma \begin{bmatrix} \bar{q}_3 \\ q_4 \end{bmatrix}, \qquad \sigma > 0. \tag{6.9}$$

Applying Newton's method to the necessary conditions of Problem (N_0), we obtain

$$\begin{bmatrix} \partial^2_{q_1} L(q_1,q_2) & \left[\partial_{q_1} g(q_1)\right]^T \\ \partial_{q_1} g(q_1) & 0 \end{bmatrix} \begin{bmatrix} q_1^* - q_1 \\ q_2^* - q_2 \end{bmatrix} = -\begin{bmatrix} \bar{q}_3 \\ q_4 \end{bmatrix}, \tag{6.10}$$

where (q_1^*, q_2^*) is the value of (q_1,q_2) at the next iterate. Obviously, (6.9) is identical to (6.10) under the interpretation $(u_1,u_2)/\sigma = (q_1^* - q_1, q_2^* - q_2)$. □

6.1.1. *SQP as an Inverse Optimal Algorithm.* According to Proposition 6.1, an SQP algorithm may be interpreted as an inverse optimal algorithm for solving Problem (N_0) under a choice of a quadratic SLF given by (6.3) and a Riemannian metric stipulated by (6.8). This "SQP metric" is over the tangent space defined by the (q_1, q_2) coordinates and is given by the square of the KKT matrix. Note however that we did not seek to derive an SQP algorithm. It is simply a happenstance of selecting a quadratic SLF and the Riemannian metric given by (6.8). In other words, an inverse optimal algorithm is completely defined once an SLF $S(q,t)$ and a control space $\mathbb{U}(q,t)$ are specified; however, the precise form of the resulting inverse optimal algorithm are not known until the consequences of the choice of the pair $\big(S(q,t), \mathbb{U}(q,t)\big)$ are analyzed. In this context, Proposition 6.1 is an "a posteriori derivation" of an SQP algorithm using the theory of inverse optimality.

6.1.2. *SQP Versus an Inverse Optimal Algorithm.* Proposition 6.1 interprets the SQP algorithm in terms of metric spaces. The concept of using metric spaces for optimization was first proposed by Davidon in 1959 [23]. Note, however, that Davidon's "variable metric" is based on just the (inverse of the) Hessian and not the square of the KKT matrix. A few other technical differences between an inverse optimal algorithm and a classical SQP algorithm are as follows:

(1) In an inverse optimal algorithm, one solves the subproblem (P_0). The cost function in this subproblem is linear while the constraint is quadratic. In contrast, a classical SQP is a quadratic cost problem with a linear constraint.
(2) A direct solve of Problem (P_0) involves choosing $\Delta(q,t) > 0$. If Problem (P_0) is solved via (6.9) then $\Delta(q,t)$ need not be chosen a priori. Instead, the parameter $\sigma > 0$ may be absorbed in the constraints of Problem (M_h) as part of the control jump parameter $h = h_k$ to generate the next iterate. Because there are no guarantees that the product $\alpha_k := h_k\sigma$ will be equal to unity, it follows that Proposition 6.1 does not imply convergence for a unit Newton step.
(3) Proposition 6.1 does indeed imply convergence if the next iterate is chosen as part of the inverse optimality process of dissipating the SLF given by (6.3). This dissipation is quantized by the process of solving Problem (M_h).

6.2. **Choosing** $W(q,t)$**, Part II.** Let $K_A(q_1,q_2)$ denote the block 2×2 asymmetric matrix on the right-hand side of (6.5). A somewhat surprising aspect of this asymmetric matrix is that it can be positive semidefinite even though its symmetric counterpart (i.e., $K_S(q_1,q_2)$) is indefinite [7, 8, 41].

Proposition 6.2. *Let $K_A(q_1,q_2)$ denote the block 2×2 asymmetric matrix on the right-hand side of (6.5). Assume $K_A(q_1,q_2)$ is positive stable. Suppose we choose $W(q,t)$ in (6.5) according to,*

$$W(q,t) := K_A(q_1,q_2) \tag{6.11}$$

and set $\sigma = 1$. Then (6.5) reduces to the Arrow-Hurwicz-Uzawa flow corresponding to a differential-equation method for solving Problem (N_0).

Proof. Substituting (6.11) into (6.5) and setting $\sigma = 1$, we obtain

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} q_3 \\ q_4 \end{bmatrix}. \tag{6.12}$$

Substituting (6.2b) and (5.12) into (6.12), we see that

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} -\partial_{q_1} L(q_1, q_2) \\ g(q_1) \end{bmatrix} = \begin{bmatrix} -\partial_{q_1} L(q_1, q_2) \\ \partial_{q_2} L(q_1, q_2) \end{bmatrix}$$

which is the Arrow-Hurwicz-Uzawa equation [27, 34, 38, 42]. □

Although Proposition 6.2 has derived the Arrow-Hurwicz-Uzawa flow in the spirit of Proposition 6.1, note that $K_A(q_1, q_2)$ serves as a pseudometric and not a metric (as erroneously assumed in [56]). As a result, $\mathbb{U}(q,t)$ as defined in (6.4) is not compact and hence σ in (6.5) may be unbounded. The statement of Proposition 6.2 is based on setting $\sigma = 1$ (or, equivalently, selecting σ to be a bounded number) in order to match the Arrow-Hurwicz-Uzawa equation. In other words, the convergence of the Arrow-Hurwicz-Uzawa flow is not guaranteed by Proposition 6.2. Because the Hamilton-Jacobi formalism (upon which Proposition 6.2 is based) is a sufficient but not a necessary condition, it is possible to prove convergence without the aid of an SLF by considering (6.12) as a control ansatz in the fundamental dynamical equations $\dot{q} = F(q, u)$. In this alternative process, we substitute the control ansatz in (6.2a) and obtain

$$\begin{bmatrix} \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = -\begin{bmatrix} \partial^2_{q_1} L(q_1, q_2) & [\partial_{q_1} g(q_1)]^T \\ -\partial_{q_1} g(q_1) & 0 \end{bmatrix} \begin{bmatrix} q_3 \\ q_4 \end{bmatrix} = -K_A^T(q_1, q_2) \begin{bmatrix} q_3 \\ q_4 \end{bmatrix} \tag{6.13}$$

Because $K_A(q_1, q_2)$ is positive stable (by assumption) (6.13) converges to its equilibrium point [33], namely a candidate optimal point for Problem (N_0) defined by the set T_0 given by (6.1). Hence, we have the following result:

Proposition 6.3. *Let the assumptions of Proposition 6.2 hold. Then the Arrow-Hurwicz-Uzawa flow converges to its equilibrium point.*

In recognizing that (6.13) is based on the more fundamental physics of optimization as articulated by Theorem 4.1, we get the following result for linear programming problems.

Theorem 6.1. *The Arrow-Hurwicz-Uzawa algorithm is not convergent for linear programming problems.*

Proof. For linear programming problems, the Hessian of the Lagrangian $\partial^2_{q_1} L(q_1, q_2) = 0$. Hence, the symmetric part of $K_A(q_1, q_2)$, namely $(1/2)(K_A + K_A^T)$, is also the zero matrix. As a result, $K_A^T(q_1, q_2)$ is not positive stable and hence (6.13) does not converge to its equilibrium point. □

Theorem 6.1 was also proved in He et al. [34] via traditional optimization theory. He et al. also demonstrate the divergence of the Arrow-Hurwicz-Uzawa algorithm for certain nonlinear optimization problems. Their results on the divergence of the Arrow-Hurwicz-Uzawa algorithm follow readily from (6.13) and the fact that $K_A(q_1, q_2)$ is not positive stable.

A sufficient condition for $K_A(q_1, q_2)$ to be positive stable is given by the following theorem [8]:

Theorem 6.2 (Benzi-Simoncini)**.** *Assume $\partial^2_{q_1} L(q_1, q_2)$ is positive definite and $\partial_{q_1} g(q_1)$ has full rank. Define $S_H := [\partial_{q_1} g(q_1)][\partial^2_{q_1} L(q_1, q_2)]^{-1}[\partial_{q_1} g(q_1)]^T$ and let $\lambda_{\min}$ denote the smallest eigenvalue of $\partial^2_{q_1} L(q_1, q_2)$. If $\lambda_{\min} \geq 4\|S_H\|_2$ then all eigenvalues of $K_A(q_1, q_2)$ are real and positive.*

As noted by Benzi and Simoncini [8], Theorem 6.2 is a sufficient but not a necessary condition.

7. Additional Examples with Theoretical Generalizations

Section 6 provides new precision to the examples discussed in [56]. Additional examples are provided in [57] and [58] with extensions of the theory along different fronts. In the following we briefly comment on some of these results with further insights.

7.1. **An (Inverse) Optimal Control Theory for Accelerated Optimization.** In recent years, there have been a number of attempts to "explain" Nesterov's celebrated accelerated optimization method [50]. As noted in Section 2, barring [57], all of these explanations begin with Nesterov's algorithm. The question posed in [57] is whether accelerated optimization algorithms can be "derived" without using any a priori information of Nesterov's equations. An explanation was alluded to in Section 4 by suggesting that one possible measure of optimality for optimal optimization was the total variation of a hidden algorithm primitive. Consider applying this "principle" to the state trajectory $t \mapsto q(t)$. A standard assumption in optimal control theory [14, 67] is the regularity condition, $q(\cdot) \in W^{1,1}\big([t_0,t_f],\mathbb{R}^{2(1+n+m)}\big)$. A simple engineering approach [59] to reduce the total variation of a state variable is to seek $q(\cdot) \in W^{2,\infty}\big([t_0,t_f],\mathbb{R}^{2(1+n+m)}\big)$. For $q_1(\cdot)$, the functional constraint $q_1(\cdot) \in W^{2,\infty}\big([t_0,t_f],\mathbb{R}^{n}\big)$ can be directly imposed in Theorem 4.1 by setting,

$$\dot{q}_1(t) := v_1(t), \quad \dot{v}_1(t) := u_1(t) \tag{7.1}$$

so that a bounded $u_1(t)$ steers the "acceleration" $\dot{v}_1(t)$. Linking inverse-optimality to partial guidability per Section 5.1 generates a final-time equilibrium condition for the v_1 variable given by $v_1(t_f) = 0$. Carrying out the remainder of the steps developed in Section 5 generates an inverse optimal algorithm [57] which turns out to be equivalent to Nesterov's accelerated gradient method. Note, however, that this process of generating an accelerated gradient method is not based on seeking Nesterov's method; rather, it is a natural consequence of producing equations for hidden algorithm primitives with reduced total variations based on the "intuition" suggested by (7.1). Furthermore, in much the same way as Propositions 6.1 and 6.2 were not based on deriving an SQP and the Arrow-Hurwicz-Uzawa algorithms respectively, the results presented in [57] are not based on deriving Nesterov's method; rather, it is shown that Nesterov's method is a consequence of (7.1), inverse optimality and a quadratic SLF. Indeed, there is no a priori assumption of Nesterov's equations in this process and hence the results of [57] may be viewed as an a posteriori "derivation" of Nesterov's accelerated gradient method (with caveats similar to those that accompany Propositions 6.1 and 6.2).

7.2. **Nonsmooth SLFs and Nonsmooth Optimization.** In [58], nonsmooth SLFs are used to derive several variants of coordinate descent algorithms. This "derivation" is achieved in the same spirit as Propositions 6.1 and 6.2 by using a collection of nonsmooth max functions for the SLFs. When a weak Lyapunov function (such as an SLF) is nonsmooth, several variants of the infinitesimal decrease condition (Cf. (5.2c)) based on different types of subgradients have been proposed in the controls literature[16, 17, 52]. In [58], coordinate descent algorithms for unconstrained optimization are generated based on the following infinitesimal decrease condition,

$$\max_{\zeta \in \partial S(q_a)} \min_{u \in \mathbb{U}(q,t)} \langle \zeta, f(q_0,q_1,q_2,u)\rangle + R(q,t) \le 0 \quad \text{for } q_a \notin T \tag{7.2}$$

where $R(q,t)$ is the rate function as defined before (Cf. (5.17)). As a new illustration of this process, consider an unconstrained optimization problem (i.e., $m = 0$ in Problem (N)). Suppose we choose an SLF given by the ℓ_1-norm of q_3 so that $S(q) := \|q_3\|_1$. Suppose further that we choose a control space similar to (6.4) with $W(q,t) := \partial^2_{q_1} g_0(q_1)$. Then, following the same procedure as in [58], it is straightforward to show that the resulting inverse-optimal algorithm is a sign gradient descent given by,

$$q_1(k+1) = q_1(k) - \alpha_k \text{ sign}\left[\partial_{q_1} g_0(q_1(k))\right], \quad \alpha_k > 0 \tag{7.3}$$

Note once again that the derivation of (7.3) does not involve producing a differential equation or a hidden algorithm primitive. Furthermore, it follows from the results of Section 5.2 that α_k in (7.3) must be chosen to decrement the (nonsmooth) SLF given by $S(q) = \|q_3\|_1$.

The sign gradient descent is an invention of recent origin [12, 48] in that it was motivated by the needs of distributed computing for large n in machine learning applications. That is, by transmitting only the sign of the gradient across distributed processing units, the communication bottleneck is alleviated for a faster learning process [12, 55]. Thus, the sign gradient descent algorithm was invented out a practical necessity and not via the theory of inverse optimality. Nonetheless, it can now be explained in terms of the new physics of optimization as an algorithm that is generated by dissipating the ℓ_1-norm of the gradient of the objective function under a Riemannian metric given by the Hessian (of the objective function).

The infinitesimal decrease condition given by (7.2) can be further weakened using a Dini derivate of the SLF [17, 52]. In this case, the max operation in (7.2) can be replaced by the minimum [17]:

$$\min_{\zeta \in \partial S(q_a)} \min_{u \in \mathbb{U}(q,t)} \langle \zeta, f(q_0, q_1, q_2, u) \rangle + R(q,t) \leq 0 \quad \text{for } q_a \notin T \tag{7.4}$$

It is straightforward to show that replacing (7.2) by (7.4) in the results of [58] yields generalized coordinate descent algorithms that more closely follow the Gauss-Southwell rule [51, 71], where $\zeta \in \partial S(q_a)$ is chosen to maximally decrement the (nonsmooth) max SLFs.

Finally, we briefly expand on the comment alluded to Section 5.2.4 with regards to choosing $\mathbb{U}(q,t)$. Suppose we choose $\mathbb{U}(q,t)$ to be any compact convex set (for all (q,t)). Then Problem (M_u) is a convex optimization problem with a linear objective function. Hence it follows from Theorem 5.1 that this family of inverse-optimal algorithms solves Problem (N) via a sequence of convex optimization problems. Because Problem (M_u) is not generated from Problem (N) by either convexification or linearization as in the case of current sequential convex programming methods [22, 37, 47], the resulting new algorithms constitute a different breed of sequential convex optimization methods based on inverse optimality.

From the preceding discussions and the results of Section 6, it is clear that a vast number of algorithms can be derived or invented using the Hamilton-Jacobi theory of inverse optimality. The basic tool for the derivation/invention is simply choosing the pair $(S(q,t), \mathbb{U}(q,t))$.

An open question at the present time is the proper way to generalize Theorem 4.1 for nonsmooth data functions. It is apparent that this generalization would require the use of second-order subdifferential calculus [43, 46, 61]. As noted in Rockafellar and Wets [61], page 579, "The technical and conceptual challenges are formidable, however." We do not pretend it to be otherwise. Further discussions on this topic are well beyond the scope of this paper.

8. Conclusions and Outlook

The results of the present paper along with those presented in [56, 57, 58] show the following:

(1) When the data functions of a continuous optimization problem are smooth (twice differentiable) Theorem 4.1 describes the mathematical physics of hidden algorithm primitives for many number of constrained and unconstrained optimization algorithms.
(2) Hidden algorithm primitives for accelerated optimization algorithms, including Nesterov's accelerated gradient algorithm, also satisfy the dynamical equations given by Theorem 4.1 with an additional smoothness constraint imposed on the evolution of $t \mapsto q(t)$. This smoothness condition has the effect of reducing the total variation of $q(\cdot)$ thereby producing "acceleration."
(3) The generation of algorithms according to the proposed theory of inverse optimality never requires the production and propagation of differential equations. In fact, there are no assurances that such an algorithm even tracks a solution to a differential equation.
(4) In much the same way as nonsmooth Lyapunov functions for continuously differentiable dynamics are useful in control theory, nonsmooth SLFs are particularly useful for optimization because inverse-optimal algorithms are produced in terms of controlled jumps where each jump obeys an instantaneous version of a Hamilton-Jacobi inequality.
(5) Unlike traditional optimization theory wherein convergence must be proved for a proposed algorithm, the inverse-optimal framework comes pre-equipped with convergence per Theorem 5.1. The essence of this theorem is that each controlled jump must decrement the SLF upon which the control was derived. The resulting polygonal arc (i.e., the algorithm) converges to the zero of the SLF, which, by definition, satisfies the necessary conditions of optimality (within some $\varepsilon > 0$ tolerance).
(6) The specific details of a particular inverse-optimal algorithm that is defined by the selection of an SLF and the control space are unknown until the results of these choices are further analyzed. For example, it us unknown a priori that using a quadratic SLF and a particular Riemannian metric generates an SQP method.

The notion that optimization algorithms can be derived from a small set of universal equations sounds preposterous. Nonetheless, once the idea of a hidden algorithm primitive is postulated, the result is less surprising after the (control) dynamical equations are derived and optimal control theory is invoked. What is indeed surprising is that it is eventually not necessary to ever generate a differential equation to produce an algorithm. At the same time, the continuous-time concept of a hidden algorithm primitive is crucial to an understanding of the natural physics of optimization.

Because of the close connection between the Hamilton-Jacobi and Schrödinger equations [28, 66], it may be possible to transform (4.13) to quantum mechanical equations using the wave ansatz and a Madelung-Bohm-type quantum potential. If this transformation is possible, then the resulting Schrödinger equation becomes the fundamental equation for optimal algorithm primitives. Consequently, such a Schrödinger-equation-based algorithm can be directly implemented on a quantum computer for solving Problem (N). In other words, it may be more efficient to implement a Schrödinger-equation-based algorithm on a quantum machine than one based on simulating Hamilton-Jacobi equations. If such a transformation is possible, the main use of the resulting quantum solver will likely be future machine learning applications that are expected to solve massively large-scale optimization problems that require computational throughput beyond the limits of classical computing devices.

REFERENCES

[1] M. S. Bazaraa, H. D. Sherali, C. M. Shetty, Nonlinear Programming: Theory and Algorithms, Wiley-Interscience, New York, 2006.

[2] P. T. Boggs, The solution of nonlinear system of equations by A-stable integration techniques, SIAM J. Numer. Anal. 8 (1971), 767–785.

[3] A. A. Brown, M. C. Bartholomew-Biggs, Some effective methods for unconstrained optimization based on the solution of systems of ordinary differential equations, J. Optim. Theory Appl. 62 (1989), 211–224.

[4] A. A. Brown, M. C. Bartholomew-Biggs, ODE versus SQP methods for constrained optimization, J. Optim. Theory Appl. 62 (1989) 371–386.

[5] A. Bhaya, E. Kaszkurewicz, Control Perspectives on Numerical Algorithms and Matrix Problems, Advances in Design and Control, SIAM, Philadelphia, PA, 2006.

[6] Y. Bengio, Y. LeCun, G. Hinton, Deep learning, Nature, 521 (2015), 436–444.

[7] M. Benzi, G. H. Golub, J. Liesen, Numerical solution of saddle point problems, Acta Numerica 14 (2005), 1–137.

[8] M. Benzi, V. Simoncini, On the eigenvalues of a class of saddle point matrices, Numer. Math. 103 (2006), 173–196.

[9] L. Bottou, F. E. Curtis, J. Nocedal, Optimization methods for large-scale machine learning, SIAM Rev. 60/2 (2018), 223–311.

[10] S. P. Bhat, D. S. Bernstein, Finite-time stability of continuous autonomous systems, SIAM J. Control Optim., 38/3 (2000) 751–766.

[11] S. R. Bernfeld, V. Lakshmikantham, Practical stability and Lyapunov functions, Tôhoku Math. J. 32 (1980), 607–613.

[12] J. Bernstein, Y.X. Wang, K. Azizzadenesheli, A. Anandkumar, Compression by the signs: distributed learning is a two-way street, 6th International Conference on Learning Representations, pp. 1–6, 2018.

[13] F. H. Clarke, Optimization and Nonsmooth Analysis, SIAM, Philadelphia, 1990.

[14] F. H. Clarke, Functional Analysis, Calculus of Variations and Optimal Control, Springer-Verlag, London, 2013.

[15] F. H. Clarke, Yu. S. Ledyaev, A. I. Subbotin, Universal feedback control via proximal aiming in problems of control under disturbances and differential games, Univ. de Montréal, Report CRM 2386, 1994.

[16] F. H. Clarke, Lyapunov functions and feedback in nonlinear control, In: M.S. de Queiroz, M. Malisoff, P. Wolenski (eds) Optimal control, stabilization and nonsmooth analysis. Lecture Notes in Control and Information Science, vol. 301, pp. 267–282, Springer, Berlin, Heidelberg, 2004.

[17] F. Clarke, Discontinuous feedback and nonlinear systems, Proc. IFAC conference on nonlinear control (NOLCOS), pp. 1–29, Bologna ,2010.

[18] V. Chellabonia, W. M. Haddad, A unification between partial stability and stability theory for time-varying systems, IEEE Control Systems Magazine, 22 (2002), 66–75.

[19] H. B. Curry, The method of steepest descent for non-linear minimization problems, J. Quart. Appl. Math. 2 (1944), 258–261.

[20] J. Diakonikolas, L. Orecchia, The approximate duality gap technique: A unified theory of first-order methods, SIAM J. Optim. 29 (2019), 660-689.

[21] J. Diakonikolas, M. I. Jordan, Generalized momentum-based methods: A Hamiltonian perspective, SIAM J. Optim. 31 (2021), 915–944.

[22] Q. T. Dinh, M. Diehl, Local convergence of sequential convex programming for nonconvex optimization, Recent Advances in Optimization and its Applications in Engineering, Springer, Berlin, Heidelberg, 2010.

[23] W. C. Davidon, Variable metric method for minimization, SIAM J. Optim. (1991), 1–17.

[24] Yu.G. Evtushenko, V.G. Zhadan, Stable barrier-projection and barrier-Newton methods in nonlinear programming, Optim. Methods Softw. 3 (1994), 237–256.

[25] M. Even, R. Berthier, F. Bach, N. Flammarion, H. Hendrikx, P. Gaillard, L. Massoulié, A. Taylor, A continuized view on Nesterov acceleration for stochastic gradient descent and randomized gossip, Proc. NeurIPS 2149 (2021), 28054–28066.

[26] R. A. Freeman, P. V. Kokotovic, Inverse optimality in robost stabilization, SIAM J. Control Optim. 34 (1996), 1365–1391.
[27] D. Feijer, F. Paganini, Stability of primal-dual gradient dynamics and applications to network optimization, Automatica 46 (2010), 1974–1981.
[28] J. H. Field, Derivation of the Schrödinger equation from the Hamilton-Jacobi equation in Feynman's path integral formulation of quantum mechanics, Eur. J. Phys. 32 (2011), 63–87.
[29] M. K. Gavurin, Nonlinear functional equations and continuous analogues of iteration methods, Izv. Vyssh. Uchebn. Zaved. Mat. 5 (1958) 18–31.
[30] S. T. Glad, Robustness of nonlinear state feedback – A survey, Automatica, 23 (1987), 425–445.
[31] F. Guilherme, M. I. Jordan, R. Vidal, On dissipative symplectic integration with applications to gradient-based optimization, J. Statist. Mech.: Theory and Experiment, 2021 (2021), 043402.
[32] W. M. Haddad, A. L'Afflitto, Finite-time partial stability and stabilization, and optimal feedback control, J. Franklin Institute, 352 (2015), 2329–2357.
[33] E. Hairer, S. P. Nørsett, G. Wanner, Solving Ordinary Differential Equations I: Nonstiff Problems, Springer-Verlag, 1993.
[34] B. He, S. Xu, X. Yuan, On convergence of the Arrow-Hurwicz method for saddle point problems, J. Math. Imaging Vision 64 (2022), 662–671.
[35] C. Jammazi, Continuous and discontinuous homogeneous feedbacks finite-time partially stabilizing controllable multichained systems, SIAM J. Control Optim. 52 (2014), 520–544.
[36] I. Karafyllis, M. Krstic, Global dynamical solvers for nonlinear programming problems, SIAM J. Control Optim. 55 (2017), 1302–1331.
[37] M. Kheirandishfard, F. Zohrizadeh, S. R. Alimo, F. Kamangar, R. Madani, Sequential convex programming revisited, Proc. 60th IEEE CDC, pp. 3137–3142, 2021.
[38] D. G. Luenberger, Y. Ye, Linear and Nonlinear Programming, Springer, 2008.
[39] C. Lemaréchal, Cauchy and the gradient method, Documenta Mathematica, Extra Volume: ISMP. (2012), 251–254.
[40] L. Lessard, B. Recht, A. Packard, Analysis and design of optimization algorithms via integral quadratic constraints, SIAM J. Optim. 26 (2016), 57–95.
[41] J. Liesen, B. N. Parlett, On nonsymmetric saddle point matrices that allow conjugate gradient iterations, Numerische Math. 108 (2008) 605–624.
[42] Y. Liu, C. Lageman, B. D.O. Anderson, G. Shi, An Arrow-Hurwicz-Uzawa type flow as least squares solver for network linear equations, Automatic, 100 (2019), 187–193.
[43] B. S. Mordukhovich, Variational Analysis and Generalized Differentiation, I: Basic Theory, Grundlehren Math. Wiss. 330, Springer, Berlin, 2006.
[44] D. M. Murray, S. J. Yakowitz, The application of optimal control methodology to nonlinear programming problems. Math. Program. 21 (1981), 331–347.
[45] A. A. Martynyuk, On practical stability and optimal stabilization of controlled motion, Banach Center Publications 14 (1985), 383–400.
[46] B. S. Mordukhovich, R. T. Rockafellar, Second-order subdifferential calculus with applications to tilt stability in optimization, SIAM J. Optim. 22 (2012), 953–986.
[47] F. Messerer, K. Baumgärtner, M. Diehl, Survey of sequential convex programming and generalized Gauss-Newton methods, ESAIM: Proc. 71 (2021), 64–88.
[48] E. Moulay, V. Léchappé, F. Plestan, Properties of the sign gradient descent algorithms, Info. Sci. 492 (2019), 29–39.
[49] J. Nocedal, S. Wright, Numerical Optimization, Springer, 2006.
[50] Yu. E. Nesterov, A method of solving a convex programming problem with convergence rate $\mathscr{O}(1/k^2)$, Soviet Math. Dokl. 27 (1983), 371–376.
[51] J. Nutini, M. Schmidt, I. H. Laradji, M. Friedlander, H. Koepke, Coordinate descent converges faster with the Gauss-Southwell rule than random selection, Proc. 32nd Inter. Conf. Machine Learning, vol. 37, pp. 1632–1641, Lille, 2015.
[52] P. Osinenko, P. Schmidt, S. Streif, Nonsmooth stabilization and its computational aspects, IFAC PapersOnLine, 53 (2020), 6370–6377.

[53] B. T. Polyak, Some methods of speeding up the convergence of iteration methods, USSR Computational Math. and Math. Phys., 4/5 (1964) 1–17.

[54] A. Polyakov, Discontinuous Lyapunov functions for nonasymptotic stability analysis, Proc. 19th World Congress, IFAC, pp., 5455–5460, Cape Town, 2014.

[55] M. Pandey, M. Fernandez, F. Gentile, F. et al., The transformational role of GPU computing and deep learning in drug discovery, Nat. Mach. Intell. 4 (2022), 211–221.

[56] I. M. Ross, An optimal control theory for nonlinear optimization, J. Comput. Appl. Math. 354 (2019), 39–51.

[57] I. M. Ross, Generating Nesterov's accelerated gradient algorithm by using optimal control theory for optimization, J. Comput. Appl. Math. 423 (2023), 114968.

[58] I. M. Ross, Derivation of coordinate descent algorithms from optimal control theory, Oper. Res. Forum 4 (2023), 31.

[59] I. M. Ross, A Primer on Pontryagin's Principle in Optimal Control, Collegiate Publishers, San Francisco, 2015.

[60] I. M. Ross, An optimal control theory for accelerated optimization, doi: 10.48550/arxiv. 1902.09004, https://arxiv.org/abs/1902.09004.

[61] R. T. Rockafellar, R. J.-B. Wets, Variational Analysis, Grundlehren Math. Wiss. 317, Springer, Berlin, 2009.

[62] E. D. Sontag, Mathematical Control Theory: Deterministic Finite Dimensional Systems, Springer, 1998.

[63] S. Smale, A convergent process of price adjustment and global Newton methods, J. Math. Econom. 3 (1976), 107–120.

[64] B. Shi, S. S. Du, M. I. Jordan, W. J. Su, Understanding the acceleration phenomenon via high-resolution differential equations, Math. Progam. 195 (2022), 79–148.

[65] W. Su, S. Boyd, E. J. Candes, A differential equation for modeling Nesterov's accelerated gradient method: theory and insights, J. Mach. Learn. Res. 17 (2016), 1–43.

[66] W. P. Schleich, D. M. Greenberger, D. H. Kobe, M. O. Scully, Schrödinger equation revisited, Proc. Natl. Acad. Sci. U.S.A. 110 (2013), 5374–5379.

[67] R. B. Vinter, Optimal Control, Birkhäuser, Boston, 2000.

[68] V. I. Vorotnikov, Partial stability, stabilization and control: some recent results, 15th IFAC World Congress, Barcelona, 2002.

[69] A. Wibisono, A. C. Wilson, M. I. Jordan, A variational perspective on accelerated methods in optimization, Proc. Nat. Acad. Sci. USA 113 (2016), E7351-E7358.

[70] A. C. Wilson, B. Recht, M. I. Jordan, A Lyapunov analysis of accelerated methods in optimization, J. Mach. Learn. Res. 22 (2021), 1–34.

[71] J. Wolfson-Pou, E. Chow, Distributed Southwell: An iterative method with low communication costs. 2017, Proc. SC17, pp. 12-17, Denver, 2017.

[72] H. Yamashita, A differential equation approach to nonlinear programming, Math. Program. 18 (1980), 155–168.

[73] L. Zhou, Y. Wu, L. Zhang, G. Zhang, Convergence analysis of a differential equation approach for solving nonlinear programming problems, Appl. Math. Comput. 184 (2007), 789–797.

[74] A. L. Zuev, Stabilization of non-autonomous systems with respect to a part of the variables by means of controlled Lyapunov functions, J. Auto. Info. Sci. 32 (2000), 18–25.