Continuous-time systems that solve computational problems

P.-A. ABSIL[1,2]

[1] *Department of Mathematical Engineering, Université catholique de Louvain, 1348 Louvain-la-Neuve, Belgium. URL: http://www.inma.ucl.ac.be/∼absil/*
[2] *Peterhouse, University of Cambridge, Cambridge CB2 1RD, UK.*

The concept of using continuous-time dynamical systems (described by ordinary differential equations) in order to solve computational problems is discussed, with an emphasis on convergence analysis and design procedures. The continuous-time approach is illustrated on concrete examples related to the computation of eigenvalues and eigenvectors of matrices.

Key words: continuous-time systems, ordinary differential equations, continuous-time algorithms, gradient flows, Rayleigh quotient gradient flow, double-bracket flow, eigenvalue problem

1 INTRODUCTION

Many scientific computing algorithms (in various domains, such as weather prediction, structural analysis, or electrical network analysis) strongly rely on solving fundamental matrix computation problems (such as linear system solving, eigenvalue decomposition, singular value decomposition, matrix nearness problems, joint diagonalization of matrices...). These problems are often solved using iterative algorithms that assume the general form

$$x(k+1) = G(x(k)). \tag{1}$$

Equation (1) can be viewed as a *discrete-time dynamical system*, where the state x depends on the "time" k that takes integer values. A sequence of

points $\{x(k)\}_{k=0}^{\infty}$ satisfying (1) is called the (positive) solution trajectory of (1) based at $x(0)$. A simple example of discrete-time dynamical system is the power method,

$$x(k+1) = Ax(k)/\|Ax(k)\|; \tag{2}$$

under reasonable requirements on A and $x(0)$, it computes the dominant eigenvector of the matrix A, i.e., the solution trajectory converges to the dominant eigendirection of A as k goes to infinity. We refer to [27] for more information on this and other iterations for matrix computation problems.

There exist "natural" physical systems that can be described by a discrete-time dynamical system like (1). An example is the *Bouncing Ball* [29, §2.4], where a ball repeatedly impacts a sinusoidally vibrating table; this yields a two-dimensional system (because the state of the ball can be determined by the impact time and the velocity at impact) exhibiting rich dynamics (stable and unstable points of period one and higher, bifurcations, chaotic behavior). However, most physical dynamical systems are continuous-time systems, whose dynamics are governed by ordinary differential equations (ODE) or partial differential equations (PDE). Here we only consider ODEs, which assume the form

$$\dot{x}(t) = F(x(t)) \tag{3}$$

where $t \mapsto \frac{dx(t)}{dt} = \dot{x}(t)$ denotes the derivative of the signal $t \mapsto x(t)$. A curve $t \mapsto x(t)$ satisfying (3) is termed a solution trajectory of (3) based at $x(0)$. Examples of physical systems whose dynamics can be described in the form (3) include RLC electric circuits and rigid mechanical systems; details can be found in [29, 36, 32].

In the world of digital computers, the concept of using continuous-time systems to achieve computational tasks may seem questionable: in numerical analysis, computing a solution trajectory of (3) is often considered as a difficult and computationally demanding task in itself; see for example [35, 12] for the particular case of the double-bracket continuous-time system (14). Yet there is a vast literature on continuous-time algorithms, spanning several areas of computational science, including, but not limited to, linear programming [6, 7, 10, 25], continuous nonlinear optimization [24, 38], discrete optimization [33, 34, 54, 4], signal processing [5, 23, 18], model reduction [30, 56] and automatic control [30, 42, 28]. Applications in linear algebra, and especially in eigenvalue and singular value problems, are particularly abundant. Important advances in the area have come from the work on isospectral flows in the early 1980s. We refer to [30, 15, 19, 17, 50, 41, 9, 13, 43] and the

many references therein. There is a computability theory for continuous-time algorithms, initiated by the work of Shannon on the general-purpose analog computer [52]; see [48, 11, 45] and references therein. Finally, a theory of computational complexity for continuous-time algorithms has recently started developing [8].

In this paper, we first point out several explanations for the widespread interest for continuous-time algorithms; see Section 2. In Section 3, we give a more precise definition of continuous-time systems and explain how they can be utilized to produce solutions of computational problems. The concepts are illustrated on two practical examples. Section 4 provides an overview of mathematical techniques involved in studying and proving computational properties of continuous-time systems. In Section 5, we show on an example how a continuous-time system with specific computational properties can be obtained using a constructive approach. Conclusions are drawn in Section 6.

2 CONTINUOUS-TIME VS DISCRETE-TIME ALGORITHMS

In this section, we advance several reasons that contribute to explain the wide interest for continuous-time systems that solve computational problems. Continuous-time and discrete-time systems have crucial differences. For analysis purposes, a drawback of continuous-time systems is the issue of existence and uniqueness of solution trajectories: for the discrete-time system (1), uniqueness is guaranteed, and existence holds as long as x_k stays in the domain of definition of the function G; for the continuous-time system (3), it is more difficult to guarantee existence and uniqueness of solution trajectories, and indeed we show in Section 4 that things can go wrong in several ways when F is maliciously chosen. In spite of this, continuous-time systems are arguably easier to analyze than discrete-time systems. A fundamental reason, of topological nature, is that the solution trajectories of continuous-time systems are continuous curves in state space, while the solution trajectories of discrete-time systems are sequences of points, which are more difficult to "track down". This distinction has important topological consequences [29]. For example, while a continuous-time system needs at least three state variables to exhibit chaotic behavior, discrete-time systems of only one variable can be chaotic (see e.g. the *logistic map* $x(k+1) = ax(k)(1-x(k))$ [44]). The study of stable and unstable manifolds of equilibrium points is also easier for continuous-time systems [29].

This partly explains a long-standing interest in the numerical analysis community for continuous-time versions of iterative processes. The ideal case is

when the solution trajectory of the continuous-time system interpolates the solution trajectory of the discrete-time system. A simple example is the *power flow*

$$\dot{x} = Bx \tag{4}$$

whose solution trajectories are given by $x(t) = \exp(Bt)x(0)$. It is easy to see that if $x_D(k)$ is a solution trajectory of the power method (2) and $x_C(t)$ is a solution trajectory of the power flow (4) with $x_D(0) = x_C(0)$ and $A = \exp(B)$, then $x_C(k) = x_D(k)$ for all integer k. A celebrated example is the QR algorithm for eigenvalue computation (see, e.g., [55]) whose iterates were shown to be unit time samples of a particular Lax-pair equation [26, 53, 21, 46].

Oftentimes, however, this interpolation property does not hold and the continuous-time counterpart is merely "related" to the discrete-time system. The relation may be that the discrete-time system is the iteration obtained by applying a numerical integration scheme to the continuous-time system. For example, the Newton iteration $x(k+1) = x(k) - (f'(x(k)))^{-1} f(x(k))$, for finding a zero of the function f, may be regarded as one explicit Euler step with unit steplength applied to the continuous-time system $\dot{x} = -(f'(x))^{-1} f(x)$ [14]. The continuous-time system may turn out to have interesting properties, quite different from the discrete-time counterpart. For example, in [41], a continuous-time system related to the Rayleigh quotient iteration was shown to visit all eigenvectors of the given matrix in finite time. Such continuous-time systems may also be used to study the asymptotic behavior of their discrete counterpart, referring to the theory of Ljung [39] and Kushner and Clark [37]; see for example [47].

Another reason for considering continuous-time systems is that it is easier to enforce certain qualitative features on continuous-time systems than on discrete-time systems. In order to obtain a suitable discrete-time algorithm, it is thus often advantageous to first first produce a continuous-time algorithm, then attempt to discretize it correctly. An example is the computation of Lyapunov exponents via orthogonal flows; see [22] and references therein.

Continuous-time system are also of particular interest in tracking problems, when problem parameters change continuously over time. The task of the algorithm is here to follow a solution $y(t)$ of the problem $P(t)$ as time evolves. This suggests using a continuous-time approach to design algorithms for this problem; see, e.g., [19, 20].

Finally, we point out that the reservations mentioned in the introduction assume that the solution trajectories of continuous-time systems must be gen-

erated accurately using a digital computer. This is in general incorrect in two ways. First, it may not be necessary to compute the solution trajectory accurately; this holds in general for systems, such as (12), that converge to a solution of the computational problem for all or almost all initial conditions. Second, digital computers are not the only possibility for hardware implementation. There is a growing industry of analog special-purpose VLSI devices for signal processing and optimization problems, many of which implement neural network architectures [31].

3 CONTINUOUS-TIME ALGORITHMS: NOTATION AND DEFINITIONS

The remainder of the paper assumes from the reader a basic background in real analysis and linear algebra. We consider continuous-time systems defined by an ODE (3) where F is a function from $\mathbb{R}^n$ to $\mathbb{R}^n$, called a *vector field*. For simplicity, we only consider autonomous systems, that is, F does not depend explicitly on t. To emphasize that the solutions depend on the time t, on the initial conditions $x(0)$ and on the vector field F specifying the dynamics, we let $\phi_t^F(x_0)$ denote the solution trajectory of (3) equal to x_0 at $t = 0$; in other words,

$$\frac{d}{dt}\left(\phi_t^F(x_0)\right) = F\left(\phi_t^F(x_0)\right), \tag{5}$$

$$\phi_0^F(x_0) = x_0. \tag{6}$$

The superscript F will be omitted when the dynamics are clear from the context or irrelevant. The family of functions $\phi_t^F : \mathbb{R}^n \to \mathbb{R}^n$ is called the *flow* of the vector field F. (They form a family because there is one function ϕ_t^F for each t.) A point x^* where $F(x^*) = 0$ is called an *equilibrium point* of (3). The equilibrium point x^* is *stable* if, given a neighborhood V of x^*, there exists another neighborhood W of x^* such that all solution trajectories starting in W stay in V. Not surprisingly, x^* is *unstable* if it is not stable: there is a neighborhood V of x^* such that there are trajectories starting arbitrarily close to x^* that eventually leave V. A point x^* is attracting if it has a neighborhood from which all solution trajectories converge to x^*. An equilibrium point is *asymptotically stable* if it is both attracting and stable. Given a differentiable real-valued function f on $\mathbb{R}^n$, the steepest-descent system for f is

$$\dot{x}(t) = -\mathrm{grad}\, f(x(t)) \tag{7}$$

where $\mathrm{grad}\, f(x) = (\partial_1 f(x), \ldots, \partial_n f(x))^T$ is the gradient of f.

Consider the continuous-time system (3) and its induced flow ϕ_t^F. In general, the vector field F will depend on parameters. These parameters, along with the initial condition x_0, can be considered as the input of a continuous-time algorithm. (In the case of the example in Section 3.1, the input appears in F, while in Section 3.2, the input appears in the initial condition.) In most cases, the output we are interested in is the limit point of the solution trajectory,

$$\phi_\infty^F(x_0) := \lim_{t\to\infty} \phi_t^F(x_0). \tag{8}$$

This raises several questions of existence and uniqueness nature that are addressed in Section 4. Note that in practice, the algorithm is terminated in finite time: the computation is halted when some stopping criterion is satisfied. (The stopping criterion may be based on the norm of $F(x(t))$, or on any information indicating that $x(t)$ has reached a sufficiently good approximation of a desired solution.)

3.1 Example: the Rayleigh quotient gradient flow

Let A be an n-by-n real symmetric matrix. A nonzero vector v is called an eigenvector of A if there exists a scalar λ, called eigenvalue, such that

$$Av = \lambda v. \tag{9}$$

Notice that the scale of v is irrelevant in (9): if v is an eigenvector associated to λ, then αv is also an eigenvector associated to λ for every $\alpha \neq 0$. It is a well-known result of linear algebra (see, e.g., [49]) that the symmetric matrix A admits n real orthonormal eigenvectors $v_1, \ldots, v_n$, i.e.,

$$v_i^T v_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j, \end{cases} \tag{10}$$

with associated real eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$. Even then, the "sign" of the v_i's is undetermined: $(\pm v_1, \ldots, \pm v_n)$ forms an orthonormal basis of eigenvector for all choices of signs. The computation of all or a few eigenvectors of a matrix is a fundamental problem of linear algebra, with numerous applications in science and engineering, such as the study of mechanical vibrations and electronic structure computations; see [51] for details.

The *Rayleigh quotient* of A is the smooth real-valued function f_A defined on $\mathbb{R}^n$ without its origin by

$$f_A(x) := \frac{x^T A x}{x^T x}. \tag{11}$$

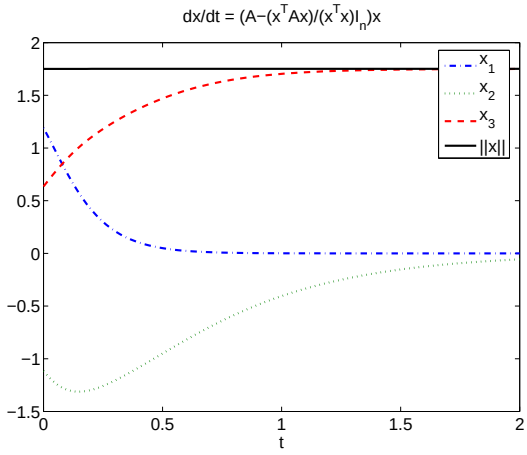

FIGURE 1
Convergence of flow (12).

We have $\operatorname{grad} f_A(x) = (A - f_A(x)I_n)x$, where I_n denotes the n-by-n identity matrix. The steepest-ascent system for f_A is thus

$$\dot{x} = (A - f_A(x)I_n)x, \tag{12}$$

also known as the continuous-time power algorithm [19]. Figure 1 shows the evolution of the three ($n = 3$) components of $x(t)$ for a (randomly chosen) initial condition $x(0) \approx (1.2, -1.1, 0.6)^T$, with

$$A := \begin{bmatrix} -5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix}. \tag{13}$$

Two remarkable properties can be observed. (i) The norm of $x(t)$ is constant. (ii) The solution $x(t)$ converges to a vector collinear with $(0, 0, 1)^T$, that is, it converges to the eigendirection corresponding to the largest (in algebraic value) eigenvalue of A. In Section 4, we will show that the first property holds always and the second property holds for almost all initial conditions. The ODE (12) thus defines a continuous-time algorithm for dominant eigenvector computation.

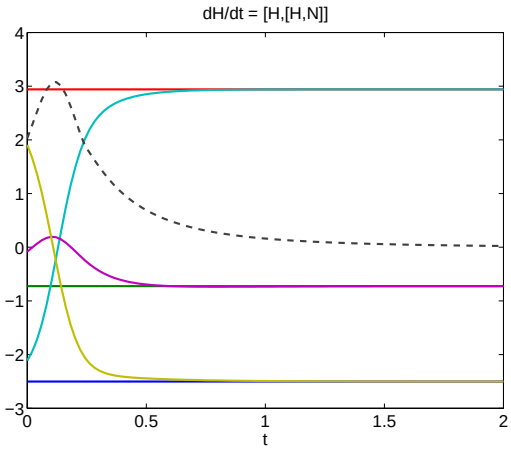

FIGURE 2
Convergence of flow (14). The straight lines show the evolution of the three eigenvalues of $H(t)$. The curved lines show the evolution of the three diagonal elements of $H(t)$. The dashed line show the evolution of the norm of the off-diagonal part of $H(t)$, which is seen to converge to zero.

3.2 Example: the double bracket flow

Consider the dynamical system

$$\dot{H} = [H, [H, N]], \tag{14}$$

where $H(t)$ is an n-by-n matrix, N is a diagonal matrix chosen as $N = \mathrm{diag}(n, n-1, \ldots, 1)$, and $[A, B] := AB - BA$ denotes the commutator. This continuous-time system was introduced by Brockett [10] and Chu and Driessel [16]. Simulation results for this flow are presented in Figure 2 in the case $n = 3$. It shows that the eigenvalues of $H(t)$ are constant and that $H(t)$ converges towards a diagonal matrix. Since the eigenvalues of a diagonal matrix are equal to its diagonal elements, it follows that the flow of (14) computes the eigenvalues of the initial matrix $H(0)$. It is shown in [10] that the double bracket flow can also be used to sort a list of numbers and to solve linear programming problems. For a convergence analysis of (14), we refer to [30] or [19, §4.2].

4 ANALYSIS OF CONTINUOUS-TIME ALGORITHMS

In this section, existence and uniqueness issues are discussed for the solution trajectories of continuous-time systems, and convergence theorems are presented. A convergence analysis of the continuous-time system (12) is presented as an illustration. We refer to [30, 19] for further reading on the topic.

We first discuss and illustrate existence and uniqueness issues. The solution of (3) may not be unique; for example, the system

$$\dot{x} = 3x^{2/3} \tag{15}$$

with initial condition $x(0) = 0$ admits both $x(t) = 0$ and $x(t) = t^3$ as solutions. In the other extreme, there may be no solution at all; for example, the system defined by

$$\dot{x} = \begin{cases} 1 & \text{if } x < 0, \\ -1 & \text{if } x \geq 0, \end{cases} \tag{16}$$

with initial condition $x(0) = 0$, has no solution. (Indeed, the vector field is such that the solution must decrease, but it cannot since the vector field below zero points to positive values.) Moreover, even if the solution exists and is unique for some interval of time, it may be defined only on a finite domain of time; for example, the system

$$\dot{x} = 1 + x^2 \tag{17}$$

with initial condition $x(0) = 0$ has solution $x(t) = \tan(t)$, which cannot be extended beyond the interval $-\frac{\pi}{2} < t < \frac{\pi}{2}$. Finally, as for discrete-time systems, the solution $x(t)$ may reach a region of $\mathbb{R}^n$ where the vector field F is not defined.

Nevertheless, existence and uniqueness of solutions can be guaranteed under rather mild conditions (see existence and uniqueness theorems in [29, 36, 32]). Notably, it is proven that if the vector field F is continuously differentiable about a point x_0, then there exists a unique solution trajectory of (3) based at x_0 on some nonvanishing time interval. If moreover the solution is known to stay in a compact set where F is differentiable, then it exists and is unique for all time.

Now, even if $\phi_t^F(x_0)$ is well defined for all t, it does not automatically follow that the asymptotic solution $\phi_\infty^F(x_0)$, as defined in (8), does exist. First of all, the solution may escape to infinity. Barring this situation, the solution trajectory $t \mapsto \phi_t^F(x_0)$ may have more than one limit point [29, 32]. (By definition, a point y is a *limit point* of $t \mapsto \phi_t(x_0)$ if there exists a

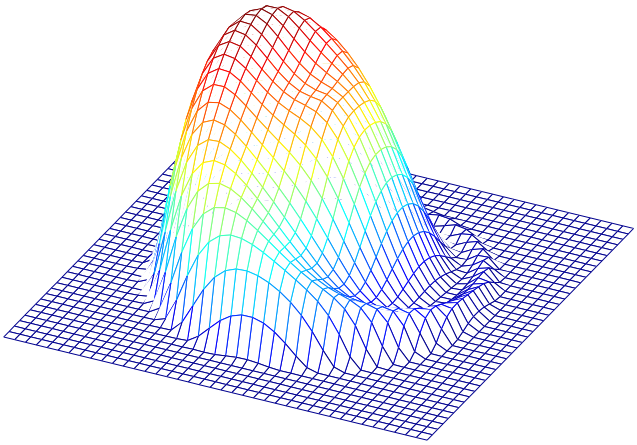

FIGURE 3
A plot of the 'Mexican Hat' function.

sequence $\{t_i\}$ such that $\lim_{i\to\infty} \phi_{t_i}(x_0) = y$, and the set of all limit points is called the *limit set* of $t \mapsto \phi_t(x_0)$.) To see how a solution trajectory can possibly converge to a set of equilibria, even for steepest-descent flows, it is interesting to consider the example known as the Mexican hat. Think about a real-valued function f on $\mathbb{R}^2$, whose graph looks like a hat with a groove spiraling towards a circle centered on the graph; see Figure 3. (The circle is not easy to locate in Figure 3 because the size of the groove quickly decreases to zero.) Consider the steepest-descent system

$$\dot{x} = -\mathrm{grad}\, f(x). \tag{18}$$

Intuitively, it is easy to imagine that a solution trajectory of (18), starting close to the top of the hat, will be trapped in the groove and thus spiral ad infinitum towards the rim; this was proven in [3] by exhibiting a function f admitting a closed-form steepest-descent trajectory whose limit set is the whole unit circle.

It has been long known [40] that such a situation cannot happen when the function f is real analytic. Interestingly, it is only recently [3] that this result has been adapted to the case of discrete-time descent systems, and the development remains less streamlined than in the continuous-time case; in particular, the steepest-descent concept, which only makes sense in continuous-time, had to be replaced by less intuitive sufficient-descent conditions on the discrete-time system. This is thus an illustration that continuous-time systems tend to be easier to analyze than discrete-time systems.

An important tool in the convergence analysis of descent-type continuous-time systems is LaSalle's invariance principle. Consider the dynamical system (3) with F continuously differentiable, and let f be a continuously differentiable real-valued function on $\mathbb{R}^n$ such that $\dot{f}(x) \leq 0$ for all x in a compact set Ω, where $\dot{f} := \frac{d}{dt} f(\phi_t(x))|_{t=0}$. If $t \mapsto \phi_t(x_0)$ is a trajectory of (3) that remains in Ω for all $t > 0$, then the limit set of the trajectory $t \mapsto \phi_t(x_0)$ is contained in the largest invariant set inside $S := \{x \in \Omega : \dot{f}(x) = 0\}$. In particular, if the largest invariant set inside S reduces to one point, then this point is asymptotically stable. Applied to the steepest-descent flow (18), this result yields the following statement.

Proposition 1 *If $t \mapsto x(t)$ is a bounded solution trajectory of* (18)*, where f is a continuously differentiable function, then the limit set of $t \mapsto x(t)$ is a connected component of the set of stationary points of f.*

We also have

Proposition 2 *The stationary points of f are the equilibrium points of* (18)*. If x^* is an isolated stationary point and a local minimum of f, then x^* an asymptotically stable equilibrium point.*

Note that the condition that x^* be isolated *as a stationary point* is often overlooked [2]. More convergence results, for gradient flows and continuous-time flows in general, can be found in [29, 36, 32].

Using the above theoretical results, we can analyze the convergence of the solution trajectories of the Rayleigh quotient gradient flow (12); a similar development can be found in [30]. The fact that $\|x(t)\|^2 := x^T(t)x(t)$ is constant follows from $\frac{d}{dt}(x^T x) = \dot{x}^T x + x^T \dot{x} = 2x^T(A - f_A(x)I)x = 0$. Therefore, we can restrict the analysis to the set $\Omega := \{x : \|x\| = \|x(0)\|\}$, which is a compact set, and the solution trajectories of (12) thus converge to a connected component of the set of stationary points of f_A. An analysis of f_A shows that its stationary points are the eigenvectors of A. Assuming that the eigenvalues of A are simple, the eigenvectors in S are isolated, thus each

solution trajectory converges to an eigenvector. Finally, among the eigenvectors, only $\pm\|x(0)\|v_1$ is a local (and global) minimizer of $f_A|_S$, thus these two points are the only attractors. In conclusion, the solution trajectories converge to $\pm\|x(0)\|v_1$, unless they start on the stable manifold of another eigenvector; those manifolds form a set of measure zero. This concludes the convergence analysis of (12).

5 DESIGN OF CONTINUOUS-TIME ALGORITHMS

A usual design procedure for continuous-time algorithms is to construct a function f whose minimizers correspond to solutions of the computational problem, and to rely on a gradient-descent approach (7) in order to compute a minimizer. It is also possible to exploit symmetries of the problem in order to reduce the computational task to a combination of problems in lower-dimensional spaces. The following continuous-time algorithm for principal component analysis, proposed in [1], is an illustration.

Let A be a symmetric matrix, with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$ and orthonormal eigenvectors $v_1, \ldots, v_n$ as defined above. Assume further that $\lambda_1 > \cdots > \lambda_{p+1}$. Recall that the *column space* (or *range*, or *image*) of a matrix is the linear subspace spanned by its columns. We want to construct a continuous-time system

$$\dot{Y} = F(Y), \qquad Y(t) \in \mathbb{R}^{n\times p}, \tag{19}$$

such that $Y(t)$ converges to $[\pm v_1|\ldots|\pm v_p]$. We will achieve this goal if the solution trajectory combines the following three properties: (i) the column space of $Y(t)$ converges to the column space of $V := [v_1|\ldots|v_p]$; (ii) $Y(t)$ converges to the set of orthonormal matrices; (iii) $Y^T(t)AY(t)$ converges to the set of diagonal matrices with entries sorted in decreasing order. With a view towards satisfying these three properties, consider the following dynamics

$$\dot{Y} = W_Y + Y(Y^TY)^{-1}S_Y + Y(Y^TY)^{-1}\Omega_Y, \tag{20}$$

where W_Y is orthogonal to Y (that is, $Y^TW_Y = 0$), S_Y is symmetric, and Ω_Y is skew-symmetric ($\Omega_Y^T = -\Omega_Y$). (The subscript "Y" denotes the dependency on Y.) A crucial observation is that the three terms in (20) have specific actions related to the three properties stated above: it is possible to show that only the term W_Y affects the column space of Y; only the term $Y(Y^TY)^{-1}S_Y$ modifies the *shape* of Y (the two other terms correspond to rigid motions); finally, the term $Y(Y^TY)^{-1}\Omega_Y$ corresponds to rigid motions

that do not modify the column space of Y. The idea is thus to choose W_Y such that property (i) is satisfied, choose S_Y such that property (ii) is satisfied, and choose Ω_Y to rotate Y in such a way that (iii) is satisfied. For example, with the choices $W_Y := AY - Y(Y^TY)^{-1}Y^TAY$, $S_Y := (I_p - Y^TY)$ and $\Omega_Y := [Y^TAY, N]$, it can be shown that the solution trajectories of (20) satisfy (ii) for all initial conditions and (i) and (iii) for almost all initial conditions. Such a flow is called a *principal component flow* since, assuming positive-definiteness of A, it computes $v_1, \ldots, v_p$ which are the principal components of A.

6 CONCLUSION

Numerous continuous-time systems that solve computational problems have been proposed and analyzed in the scientific literature. The goal of this paper has been to discuss several justifications for this widespread interest and to give a short illustrated introduction to techniques used in the design and analysis of continuous-time algorithms. The recent wide availability of special-purpose analog VLSI devices calls for further advances in the understanding of continuous-time algorithms, particularly in terms of computational complexity, error analysis and systematic design procedures.

ACKNOWLEDGEMENT

The author wishes to thank Arieh Iserles and the anonymous reviewers for several valuable suggestions.

This work was supported by Microsoft Research through a Research Fellowship at Peterhouse, University of Cambridge.

REFERENCES

[1] P.-A. Absil. (2004). Continuous-time flows on quotient spaces for principal component analysis. In *Proceedings of the 16th International Symposium on Mathematical Theory of Networks and Systems (MTNS2004)*.

[2] P.-A. Absil and K. Kurdyka. (2005). On the stable equilibrium points of gradient systems. Systems & Control Letters, in press, http://dx.doi.org/10.1016/j.sysconle.2006.01.002.

[3] P.-A. Absil, R. Mahony, and B. Andrews. (2005). Convergence of the iterates of descent methods for analytic cost functions. *SIAM J. Optim.*, 6(2):531–547.

[4] P.-A. Absil and R. Sepulchre. (2004). Continuous dynamical systems that realize discrete optimization on the hypercube. *Systems Control Lett.*, 52(3-4):297–304.

[5] Shun-ichi Amari and Andrzej Cichocki. (1998). Adaptive blind signal processing—neural network approaches. *Proc. IEEE*, 86(10):2026–2048.

[6] D. A. Bayer and J. C. Lagarias. (1989). The nonlinear geometry of linear programming. I. Affine and projective scaling trajectories. *Transactions of the American Mathematical Society*, 314(2):499–526.

[7] D. A. Bayer and J. C. Lagarias. (1989). The nonlinear geometry of linear programming. II. Legendre transform coordinates and central trajectories. *Transactions of the American Mathematical Society*, 314(2):527–581.

[8] Asa Ben-Hur, Hava T. Siegelmann, and Shmuel Fishman. (2002). A theory of complexity for continuous time systems. *J. Complexity*, 18(1):51–86.

[9] Anthony M. Bloch and Arieh Iserles. (2004). On the optimality of double-bracket flows. *Int. J. Math. Math. Sci.*, 2004(61-64):3301–3319.

[10] R. W. Brockett. (1991). Dynamical systems that sort lists, diagonalize matrices, and solve linear programming problems. *Linear algebra appl.*, 146:79–91.

[11] Manuel Lameiras Campagnolo. (2004). Continuous-time computation with restricted integration capabilities. *Theoret. Comput. Sci.*, 317(1-3):147–165.

[12] Fernando Casas. (2004). Numerical integration methods for the double-bracket flow. *J. Comput. Appl. Math.*, 166(2):477–495.

[13] M. Chu, N. Del Buono, L. Lopez, and T. Politi. (2005). On the low-rank approximation of data on the unit sphere. *SIAM J. Matrix Anal. Appl.*, 27(1):46–60.

[14] M. T. Chu. (1988). On the continuous realization of iterative processes. *SIAM Review*, 30(3):375–387.

[15] Moody T. Chu. (1994). A list of matrix flows with applications. In *Hamiltonian and gradient flows, algorithms and control*, volume 3 of *Fields Inst. Commun.*, pages 87–97. Amer. Math. Soc., Providence, RI.

[16] Moody T. Chu and Kenneth R. Driessel. (1991). Constructing symmetric nonnegative matrices with prescribed eigenvalues by differential equations. *SIAM J. Math. Anal.*, 22(5):1372–1387.

[17] Moody T. Chu and Gene H. Golub. (2002). Structured inverse eigenvalue problems. *Acta Numer.*, 11:1–71.

[18] Andrzej Cichocki and Pando Georgiev. (2003). Blind source separation algorithms with matrix constraints. *IEICE Trans. Fundamentals*, E86-A(1):1–9.

[19] J. Dehaene. (1995). *Continuous-time matrix algorithms, systolic algorithms and adaptive neural networks*. PhD thesis, Katholieke Universiteit Leuven, Faculteit Toegepaste Wetenschappen, Departement elektrotechniek-ESAT, Kard. Mercierlaan 94, 3001 Leuven, Belgium. ftp://ftp.esat.kuleuven.ac.be/pub/SISTA/dehaene/phd/.

[20] J. Dehaene, M. Moonen, and J. Vandewalle. (1999). Analysis of a class of continuous-time algorithms for principal component analysis and subspace tracking. *IEEE Trans. Circuits Systems I Fund. Theory Appl.*, 46(3):364–372.

[21] P. Deift, T. Nanda, and C. Tomei. (1983). Ordinary differential equations and the symmetric eigenvalue problem. *SIAM J. Numer. Anal.*, 20(1):1–22.

[22] Luca Dieci and Erik S. Van Vleck. (2002). Lyapunov spectral intervals: theory and computation. *SIAM J. Numer. Anal.*, 40(2):516–542.

[23] Scott C. Douglas. (2000). Self-stabilized gradient algorithms for blind source separation with orthogonality constraints. *IEEE Trans. Neural Networks*, 11(6):1490–1497.

[24] L. Faybusovich. (1991). Dynamical systems which solve optimization problems with linear constraints. *IMA J. Math. Control Inform.*, 8(2):135–149.

[25] L. Faybusovich. (1991). Hamiltonian structure of dynamical systems which solve linear programming problems. *Physica D*, 53:217–232.

[26] H. Flashka. (1974). The Toda latice, II. Existence of integrals. *Physical Review B*, 9(4):1924–1925.

[27] G. H. Golub and C. F. Van Loan. (1996). *Matrix Computations, third edition*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press.

[28] F. Grognard and R. Sepulchre. (2001). Global stability of a continuous-time flow which computes time-optimal switchings. In *Proceedings of the 16th IEEE Conference on Decision and Control*, pages 3826–3831.

[29] John Guckenheimer and Philip Holmes. (1983). *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields*, volume 42 of *Applied Mathematical Sciences*. Springer-Verlag, New York.

[30] U. Helmke and J. B. Moore. (1994). *Optimization and Dynamical Systems*. Springer.

[31] John Hertz, Anders Krogh, and Richard G. Palmer. (1991). *Introduction to the theory of neural computation*. Santa Fe Institute Studies in the Sciences of Complexity. Lecture Notes, I. Addison-Wesley Publishing Company Advanced Book Program, Redwood City, CA. With forewords by Jack Cowan and Christof Koch.

[32] Morris W. Hirsch, Stephen Smale, and Robert L. Devaney. (2004). *Differential equations, dynamical systems, and an introduction to chaos*, volume 60 of *Pure and Applied Mathematics (Amsterdam)*. Elsevier/Academic Press, Amsterdam, second edition.

[33] J. J. Hopfield. (1984). Neurons with graded response have collective computational cababilities like those of two-state neurons. *Proceedings of the National Academy of Sciences USA*, 81:3088–3092.

[34] J. J. Hopfield and D. W. Tank. (1985). 'Neural' computation of decision optimization problems. *Biological Cybernetics*, 52:141–152.

[35] Arieh Iserles. (2002). On the discretization of double-bracket flows. *Found. Comput. Math.*, 2(3):305–329.

[36] H. K. Khalil. (1996). *Nonlinear systems, second edition.* Prentice Hall.

[37] H. J. Kushner and D. S. Clark. (1978). *Stochastic approximation methods for constrained and unconstrained systems*, volume 26 of *Applied Mathematical Sciences*. Springer-Verlag.

[38] Xue-Bin Liang and Jun Wang. (2000). A recurrent neural network for nonlinear optimization with a continuously differentiable objective function and bound constraints. *IEEE Trans. Neural Networks*, 11(6):1251–1262.

[39] L. Ljung. (1977). Analysis of recursive stochastic algorithms. *IEEE Trans. Automatic Control*, 22(4):551–575.

[40] Stanisław Łojasiewicz. (1984). Sur les trajectoires du gradient d'une fonction analytique. In *Seminari di Geometria 1982-1983*, pages 115–117, Università di Bologna, Istituto di Geometria, Dipartimento di Matematica.

[41] R. Mahony and P.-A. Absil. (2003). The continuous-time Rayleigh quotient flow on the sphere. *Linear Algebra Appl.*, 368C:343–357.

[42] R. E. Mahony and U. Helmke. (1998). System assignment and pole placement for symmetric realisations. *J. Math. Systems Estim. Control*, 8(3):321–352.

[43] Jonathan H. Manton, Uwe Helmke, and Iven M. Y. Mareels. (2005). A dual purpose principal and minor component flow. *Systems Control Lett.*, 54(8):759–769.

[44] R. H. May. (1976). Simple mathematical models with very complicated dynamics. *Nature*, 261:459–467.

[45] Jerzy Mycka and José Félix Costa. (2005). The computational power of continuous dynamic systems. In *Machines, Computations, and Universality*, volume 3354 of *Lecture Notes in Computer Science*, pages 164–175. Springer-Verlag.

[46] T. Nanda. (1985). Differential equations and the QR algorithm. *SIAM J. Numer. Anal.*, 22(2):310–321.

[47] E. Oja and J. Karhunen. (1985). On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *J. Math. Anal. Appl.*, 106(1):69–84.

[48] P. Orponen. (1997). A survey of continuous-time computation theory. In D.-Z. Du and K.-I. Ko, editors, *Advances in Algorithms, Languages, and Complexity*, pages 209–224. Kluwer, Dordrecht.

[49] B. N. Parlett. (1998). *The Symmetric Eigenvalue Problem*, volume 20 of *Classics in applied mathematics*. SIAM, Philadelphia.

[50] Maria Przybylska. (2003). Isospectral-like flows and eigenvalue problem. *Future Generation Computer Systems*, 19:1165–1175.

[51] Y. Saad. (1992). *Numerical methods for large eigenvalue problems.* Manchester University Press, Manchester.

[52] Claude E. Shannon. (1941). Mathematical theory of the differential analyzer. *J. Math. Phys. Mass. Inst. Tech.*, 20:337–354.

[53] W. W. Symes. (1982). The QR algorithm and scattering for the finite nonperiodic Toda lattice. *Physica D*, 4(2):275–280.

[54] M. Vidyasagar. (1995). Minimum-seeking properties of analog neural networks with multilinear objective functions. *IEEE Trans. Automatic Control*, 40(8):1359–1375.

[55] D. S. Watkins. (1982). Understanding the QR algorithm. *SIAM Review*, 24(4):427–440.

[56] Wei-Yong Yan and James Lam. (1999). An approximate approach to H^2 optimal model reduction. *IEEE Trans. Automat. Control*, 44(7):1341–1358.