IMA Journal of Numerical Analysis (2016) Page 1 of **??**
doi:10.1093/imanum/drn000

Numerical Methods for Gradient Dynamics

MOODY T. CHU†
Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205.

AND

LI-ZHI LIAO‡
Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong.

[Received on 26 July 2016]

Gradient adaption is ubiquitous in nature and has been employed over a wide range of applications. It is often the case that each application has been given consideration separately for its best efficiency. Deriving effective numerical methods for general gradient dynamics should be of practical importance as they may serve as a common computational platform for gradient flows across diverse fields. This work investigates the arithmetic foundations of several possible algorithms with the aim at developing a unified paradigm in this regard. While standard, general-purpose numerical integrators are readily applicable, they are sophisticatedly designed for the purpose of closely tracking exact integral curves, which is slow, expensive, and unnecessary for gradient flows. In contrast, this work discusses some nontraditional ways of discretization that exploit the gradient structure, namely, they allow the possibility to relax the precision, hence the speedup; the ability to guide the path, hence the robustness; and the contractibility to correct the error, hence the precision. The high-order transition methods, capable of transforming themselves into fast converging Newton-like methods, can be combined with suitable starting procedure in the initial phase to further improve both speed and precision.

Keywords: gradient flows, dissipative vector fields, contractive methods, inexact Newton methods, transient methods, hybrid methods

1. Introduction

Given a differentiable scalar function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, the basic fact that its gradient vector $\nabla f(\mathbf{x})$ points in the steepest ascent direction with the maximum rate of change $\|\nabla f(\mathbf{x})\|_2$ for the function value at the point $\mathbf{x}$ is well known. Generalizing to a differentiable functional f over a Hilbert space, the gradient should be interpreted as the Riesz representation of the Fréchet derivative of the functional f, whose true denotation depends on, of course, the underlying inner product. It is quite natural to expect that a system adapts itself continuously in the gradient direction. Simply put, a dynamical system in the form

$$\frac{d\mathbf{x}}{dt} = -\nabla f(\mathbf{x}), \quad \mathbf{x}(0) = \mathbf{x}_0, \tag{1.1}$$

where f is a second-order differentiable functional over an appropriately defined Hilbert space H, is referred to as a (negative) gradient flow.

†Email: chu@math.ncsu.edu. This research was supported in part by the National Science Foundation under grant DMS-1316779.

‡Email: liliao@hkbu.edu.hk. This research was supported in part by the Hong Kong General Research Fund and FRG of Hong Kong Baptist University.

© The author 2016. Published by Oxford University Press on behalf of the Institute of Mathematics and its Applications. All rights reserved.

Gradient adaption plays a significant role in nature and in applications. Thermal conduction along the negative temperature gradient of the isothermal surfaces and osmosis down the concentration gradient across the cell membrane typify the gradient adaption. Mathematical models for the phase separation of materials in iron alloys (Chen, 2002; Penzler *et al.*, 2012), the segmentation or edge detection in image processing or computer vision (Michailovich *et al.*, 2007; Strzodka *et al.*, 2004), the surface evolution in differential geometry (Bobenko & Schröder, 2005; Droske & Rumpf, 2004), the flow of an ideal gas in porous medium (Vázquez, 2007), and the ground state in quantum systems (Bao & Du, 2004) all employ this notion of gradient flow. Numerous other systems that evolve in time can also be interpreted in this way. See, for example, applications in the game theory (Flåm, 2002), financial markets (Friedman & Abraham, 2009), mechanism design (Shi *et al.*, 2009), and the general surveys discussed by Hale (1988) and Hirsch & Smale (1974). An interesting story about the history of gradient flow, starting with Courant, can be found in the thesis by Behrman (1998).

1.1 *Examples of gradient flows*

We give a few specific examples below to motivate the importance of gradient flows. Consider a heat equation of the form

$$\frac{\partial u}{\partial t} = \nabla^2 u + p(u),$$

for $u = u(\mathbf{v},t) \in L^2(U)$ satisfying the boundary conditions $u|_{\partial U} = 0$. Then the right-hand side is precisely the negative gradient of the functional

$$f(u) := \int_U \left(\frac{1}{2}\|\nabla u\|^2 - P(u) \right) d\mathbf{v}, \tag{1.2}$$

where P is such that $p = P'$ and is assumed to guarantee the existence of a minimizer u for f. Indeed, a surprisingly large number of well-known diffusive partial differential equations have the intrinsic structure of a gradient flow with respect to some appropriately chosen energies and dissipative mechanisms. See Peletier (2012) for an extensive list of additional examples.

It is further advocated in the two books by Tsypkin (1971, 1973) that gradient dynamical systems also "cover many iterative formulas of numerical analysis". To give an additional attestation, we mention the Toda lattice

$$\frac{dX}{dt} = [X, \Pi_0(X)] \tag{1.3}$$

for symmetric tridiagonal matrices $X(t) \in \mathbb{R}^{n\times n}$, where $[\cdot,\cdot]$ stands for the Lie bracket, $\Pi_0(X) := X^- - X^{-\top}$, and X^- is the strictly lower triangular portion of X. On one hand, it is known that $X(t)$ is a Hamiltonian flow maintaining the spectrum of $X(0)$ for all t. On the other hand, it is also known that the vector field in (1.3) is precisely the (projected) gradient for the minimization of the objective function (Bloch *et al.*, 1992)

$$f(Q) := \frac{1}{2}\|Q^\top X_0 Q - N\|_F^2 \tag{1.4}$$

subject to the constraint $Q \in \mathcal{O}(n)$, where $\mathcal{O}(n)$ is the orthogonal group and $N = \mathrm{diag}\{n, n-1, \ldots, 2, 1\}$. That is, $X(t)$ is a gradient flow moving to reduce the off-diagonal entries of $X(t) = Q(t)^\top X_0 Q(t)$ while aligning the diagonal entries in a specified descending order. Most interestingly, the sequence $\{X(k)\}$

$\phi(\mathbf{x},\mathbf{r})$	$\frac{dV}{dt}$	$\frac{d\mathbf{x}}{dt}$
$\mathbf{g}'(\mathbf{x})^\top\mathbf{r}$	$-\|\mathbf{g}'(\mathbf{x})^\top\mathbf{r}\|_2^2$	$-\mathbf{g}'(\mathbf{x})^\top\mathbf{g}(\mathbf{x})$
$\mathbf{g}'(\mathbf{x})^{-1}\mathbf{r}$	$-\|\mathbf{r}\|_2^2$	$-\mathbf{g}'(\mathbf{x})^{-1}\mathbf{g}(\mathbf{x})$
$\mathbf{g}'(\mathbf{x})^{-1}\mathrm{sgn}(\mathbf{r})$	$-\|\mathbf{r}\|_1$	$-\mathbf{g}'(\mathbf{x})^{-1}\mathrm{sgn}(\mathbf{g}(\mathbf{x}))$
$\mathrm{sgn}(\mathbf{g}'(\mathbf{x})^\top\mathbf{r})$	$-\|\mathbf{g}'(\mathbf{x})^\top\mathbf{r}\|_1$	$-\mathrm{sgn}(\mathbf{g}'(\mathbf{x})^\top\mathbf{g}(\mathbf{r}))$
$\mathbf{g}'(\mathbf{x})^\top\mathrm{sgn}(\mathbf{r})$	$-\|\mathbf{g}'(\mathbf{x})^\top\mathrm{sgn}(\mathbf{r})\|_2^2$	$-\mathbf{g}'(\mathbf{x})^\top\mathrm{sgn}(\mathbf{g}(\mathbf{x}))$

Table 1. Control strategies and the associated dynamical systems

obtained by evaluating the solution flow $X(t)$ at integer times corresponds to exactly the sequence generated by the *QR* algorithm for eigenvalue computation (Deift *et al.*, 1983; Symes, 8182), justifying that even the all-important *QR* algorithm is a gradient adaption[1].

Along a similar thought, many crucial numerical algorithms can also be interpreted via systems and control theory point of view. Consider the basic model

$$\frac{d\mathbf{x}(t)}{dt} = \phi(\mathbf{x},\mathbf{r}), \tag{1.5}$$

where the state variable $\mathbf{x}(t)$ is controlled by some properly selected function $\phi(\mathbf{x},\mathbf{r})$ which depends on a certain output feedback $\mathbf{r}$. Different choices of ϕ can be used to manipulate the flow $\mathbf{x}(t)$ and, hence, lead to various algorithms. It is convenient that the choice of the control strategy ϕ depends on optimizing a certain cost function $V(\mathbf{x}(t))$ which, in turn, plays the role as a Lyapunov function for the dynamical system. In this case, $\phi(\mathbf{x},\mathbf{r})$ is not necessarily the gradient of $V(\mathbf{x})$, but still points to a descent direction. Take as an example the classical problem of finding zero(s) of a given differentiable function

$$\mathbf{g}:\mathbb{R}^n\to\mathbb{R}^n$$

with the residue function

$$\mathbf{r}(t) = -\mathbf{g}(\mathbf{x}(t)) \tag{1.6}$$

as the feedback. Following Bhaya & Kaszkurewicz (2006) and Chu (2008), we summarize a few possible choices for the control $\phi(\mathbf{x},\mathbf{r})$ in Table 1.

It is not difficult to verify that $V(\mathbf{x}) = \frac{1}{2}\|\mathbf{g}(\mathbf{x})\|_2^2$ and $V(\mathbf{x}) = \|\mathbf{g}(\mathbf{x})\|_1$ can be used, respectively, as the cost functions in the first four cases and in the last case. Also listed in Table 1 are the derivatives of the associated cost functions. The first case corresponds exactly to the gradient flow (1.1) with respect

[1]In addition to being a gradient flow, however, it must be noted the solution $X(t)$ to (1.3) preserves eigenvalues. For an ODE system with the additional constraint that its solution must stay on a specific manifold, e.g., the set of isospectral matrices in the Toda lattice, special types of ODE solvers should be employed. This is the realm of differential algebraic equations and, in particular, geometric integration. See, for example, Calvo *et al.* (1997), Iserles (2002), Iserles *et al.* (2000), Iwasaki & Nakamura (2002), Iwasaki & Nakamura (2004), Munthe-Kaas (1998), and especially the book by Hairer *et al.* (2006). For projected gradient flows, see our first work in Kelley *et al.* (2007). This notion of projected gradient flows on manifolds is such an important problem that it deserves further attention. To stay focused as a general overview, this paper does not address gradient dynamics constrained on manifolds.

to $V(\mathbf{x})$. The second case is the well-known continuous Newton method (Hirsch & Smale, 1979; Smale, 1977) which is not the gradient flow with respect to the Euclidean norm. However, because

$$\langle \mathbf{g}'(\mathbf{x})^{-1}\mathbf{r}, \mathbf{g}'(\mathbf{x})^{\top}\mathbf{r}\rangle = \langle \mathbf{r}, \mathbf{r}\rangle \geqslant 0,$$

the Newton direction forms an acute angle with the steepest descent direction. So the continuous Newton flow is still a descent flow. Assuming that $\mathbf{g}'(\mathbf{x})$ remains nonsingular, then the flow stops only when $\mathbf{r}(\mathbf{x}) = 0$. Similar decent properties hold for other cases, as is seen in the second column about $\frac{dV}{dt}$ in the table. A descent flow such as (1.5) for a suitable cost function $V(\mathbf{x})$ therefor generalizes the notion of the gradient flow. Note that the vector fields $\phi(\mathbf{x}, \mathbf{r})$ are only piecewise continuous in the last three cases. A discretization of such a differential system must take into account the change of sign when it happens.

1.2 *Goals in gradient integrator*

Given the fact that gradient flows are prevalent in diverse areas of applications, it is of practical importance to develop an effective gradient integrator. This area has a lengthy history already (Behrman, 1998; Hirsch & Smale, 1974; Stuart & Humphries, 1996; Tsypkin, 1971). As is typical the case, while the common element of the larger number of applications is gradient dynamics, each application probably deserves and has been given consideration separately for the best efficiency and effectiveness. One must ask, however, among the various algorithms what the essential ingredients must be to make an algorithm effective. At this point, there are two major views about a desirable gradient integrator. The difference is at exploiting two related but different properties of the gradient flow, which leads to two related but different treatments.

The first property of a gradient flow is that the potential $f(\mathbf{x}(t))$ decreases monotonically along the solution trajectory $\mathbf{x}(t)$, i.e.,

$$f(\mathbf{x}(t)) \leqslant f(\mathbf{x}(s)) \quad \text{for } t > s,$$

with strict inequality except at stationary points of f. Correspondingly, we would like a numerical method to preserve the monotonicity at least locally, i.e., if the method starts from $\mathbf{x}_n$ exactly with no past errors, then the next step produced by the numerical method would ensure that

$$f(\mathbf{x}_{n+1}) \leqslant f(\mathbf{x}_n).$$

Gradient integrators for this purpose include algebraically stable Runge-Kutta methods (Hairer & Wanner, 1996; Humphries & Stuart, 1994), discrete-gradient methods (McLachlan *et al.*, 1999; Norton *et al.*, 2015; Norton & Quispel, 2014), and average vector field collocation methods (Hairer, 2010; McLachlan *et al.*, 1999). More specifically, within the class of algebraically stable Runge-Kutta methods, the Radau IIA methods under a mild step size restriction enjoy good energy reduction and strong damping property for stiff gradient systems, but the Gauss methods do not have the damping property at all. On the other hand, the discrete-gradient methods and the average vector field methods can diminish the energy without any step size restriction, but also have no damping property for stiff gradient systems. Details can be found in a recent survey article by Hairer & Lubich (2014).

The emphasis in this paper is on exploiting the second property of a gradient flow — the contractivity. The reason why this study is important is because it might help achieve the ultimate goal of a gradient flow (1.1) (or a general descent flow (1.5)), namely, finding its limit point

$$\mathbf{x}^* = \lim_{t\to\infty} \mathbf{x}(t), \tag{1.7}$$

more effectively. At first glance, it seems that we could simply tackle the first-order optimality condition $\nabla f(\mathbf{x}) = 0$ (or the nonlinear system $\mathbf{g}(\mathbf{x}) = 0$) directly by using some general-purpose Newton-like iterative methods. Such an approach, however, fails to take advantage of the gradient property of ∇f. First, the limit points of $\mathbf{x}(t)$ might not be isolated or unique (Absil *et al.*, 2005). By iteration alone, it may locate a point far away from a specific solution $\mathbf{x}^*$ we are after. Second, an iteration method usually suffers from the inherent limitation of local convergence. On the other hand, it also seems that employing existing numerical integrators to carefully trace the trajectory $\mathbf{x}(t)$ could be another way of finding $\mathbf{x}^*$. As reliable as this approach might be, nonetheless, it usually spends expensive computation at the transient state, which would be a waste if only the limit point $\mathbf{x}^*$ is needed. An ideal gradient integrator should follow the true trajectory approximately without striving for precision and should be able to eventually rise to fast convergence to the limit point with high precision. We think that the property of contractivity provides us with several possible avenues for this pursuit. More rigorous technical details will be presented in a subsequent discussion.

For the ease of discussion, we limit our consideration to the ODE setting of gradient dynamics. For PDEs of gradient nature, a suitable discretization of the space variable, such as an application by the method of lines, can often reduce the problem to an ODE system in the form (1.1), which will not be elaborated in this paper. Thus, this paper is organized as follows. In Section 2 we bring forth the essential notion of contractivity which has been studied extensively in the literature. The gradient dynamics is contractive. It is known that a *G*-stable linear multi-step method or an algebraically stable Runge-Kutta method generates contractive numerical solutions. We argue that for gradient dynamics, any A_0-stable method is sufficient for maintaining contractivity. We exploit this property by proposing some new iterative schemes in Section 3. Being of higher orders, some of these methods have the dual capabilities of initially tracking the true trajectory more precisely by using larger step sizes and ultimately transforming themselves into fast converging algorithms. We explain why others do not have these characteristics. Some experimental results are reported in Section 4, which eventually suggests a 2-stage hybrid method in a spirit similar to those proposed by Brown & Saad (1990) and Cai *et al.* (1998).

2. Contractivity

For gradient flows, the monotonic decreasing of the f values along one single solution trajectory $\mathbf{x}(t)$ is obvious, but it does not tell how $\mathbf{x}(t)$ is influenced at the presence of perturbations. Relative to neighboring trajectories, perhaps the most conspicuous property of the gradient dynamics is the contractivity which we describe below.

2.1 *Contractive vector fields*

A vector field[2]

$$\mathbf{y}' = \mathbf{f}(t, \mathbf{y}) \tag{2.1}$$

with $\mathbf{f} : D \subset [a,b] \times \mathbb{R}^n \to \mathbb{R}^n$ is said to satisfy a one-sided Lipschitz condition[3] if there exists a scalar function $v(t)$ such that the inequality

$$\langle \mathbf{f}(t, \mathbf{y}_1) - \mathbf{f}(t, \mathbf{y}_2), \mathbf{y}_1 - \mathbf{y}_2 \rangle \leqslant v(t) \|\mathbf{y}_1 - \mathbf{y}_2\|_2^2 \tag{2.2}$$

[2]To avoid confusion we shall reserve the boldfaced $\mathbf{x}$ for the state variable in a gradient flow and $\mathbf{x}_k$ for the related iterates, while we shall use $\mathbf{y}$ as the dependent variable of a general ODE system and $\mathbf{y}_n$ for the discrete approximation at $\mathbf{y}(t_n)$.

[3]A conventional Lipschitz condition necessarily implies a one-sided Lipschitz condition, but not the converse.

holds for all $\mathbf{y}_1, \mathbf{y}_2$ in the set $\{\mathbf{y} \in \mathbb{R}^n | (t, \mathbf{y}) \in D\}$. The one-sided Lipschitz function $\nu(t)$ in (2.2) can be negative. Suppose that $\mathbf{y}(t)$ and $\mathbf{z}(t)$ are two solutions of (2.1). Then it can be argued that (Lambert, 1991, Section 7.3) (see also Hairer & Wanner (1996, Lemma 12.1))

$$\|\mathbf{y}(t_2) - \mathbf{z}(t_2)\|_2 \leqslant e^{\int_{t_1}^{t_2} \nu(\xi) d\xi} \|\mathbf{y}(t_1) - \mathbf{z}(t_1)\|_2, \quad a \leqslant t_1 \leqslant t_2 \leqslant b. \tag{2.3}$$

A contraction happens (segmentally) over the interval $[a, b]$ when $e^{\int_{t_1}^{t_2} \nu(\xi) d\xi} < 1$ for $a \leqslant t_1 \leqslant t_2 \leqslant b$. For our application, we shall limit ourselves throughout this paper to the autonomous case where

$$\mathbf{f}(t, \mathbf{y}) = \mathbf{f}(\mathbf{y}) := -\nabla f(\mathbf{y}). \tag{2.4}$$

We may make the following claim.

LEMMA 2.1 Suppose that $\mathbf{x}^*$ is isolated and that $\nabla^2 f(\mathbf{x}^*)$ is positive definite[4]. Then there is a closed ball $\mathscr{B}$ centered at $\mathbf{x}^*$ such that the gradient flow is contractive in the sense

$$\|\mathbf{y}(t_2) - \mathbf{z}(t_2)\|_2 \leqslant e^{-(t_2 - t_1)\lambda_{\mathscr{B}}} \|\mathbf{y}(t_1) - \mathbf{z}(t_1)\|_2, \quad a \leqslant t_1 \leqslant t_2 \leqslant b \tag{2.5}$$

between any two gradient flows $\mathbf{y}(t)$ and $\mathbf{z}(t)$ with starting points in $\mathscr{B}$.

Proof. By continuity, there is a closed ball $\mathscr{B}$ centered at $\mathbf{x}^*$ in which the Hessian $\nabla^2 f(\mathbf{x})$ maintains positive definiteness in $\mathscr{B}$. Reducing the radius of $\mathscr{B}$ if necessary, a positive minimal value

$$\lambda_{\mathscr{B}} := \min_{\mathbf{y} \in \mathscr{B}} \lambda^{[1]}(\nabla^2 f(\mathbf{y})) \tag{2.6}$$

exists, where $\lambda^{[1]}(M)$ means the smallest eigenvalue of a symmetric matrix M. By the mean-valued theorem for vector-valued functions, we know

$$\langle -\nabla f(\mathbf{y}_1) + \nabla f(\mathbf{y}_2), \mathbf{y}_1 - \mathbf{y}_2 \rangle = \left\langle -\left(\int_0^1 \nabla^2 f(\mathbf{y}_1 + s(\mathbf{y}_2 - \mathbf{y}_1)) ds \right) (\mathbf{y}_1 - \mathbf{y}_2), \mathbf{y}_1 - \mathbf{y}_2 \right\rangle. \tag{2.7}$$

For any $\mathbf{y}_1, \mathbf{y}_2 \in \mathscr{B}$, let $[\mathbf{y}_1, \mathbf{y}_2]$ denote the segment connecting $\mathbf{y}_1$ to $\mathbf{y}_2$. The line integral

$$A_{[\mathbf{y}_1, \mathbf{y}_2]} := \int_0^1 \nabla^2 f(\mathbf{y}_1 + s(\mathbf{y}_2 - \mathbf{y}_1)) ds$$

remains positive definite and satisfies the additivity property

$$A_{[\mathbf{y}_1, \mathbf{y}_2]} = A_{[\mathbf{y}_1, \mathbf{z}]} + A_{[\mathbf{z}, \mathbf{y}_2]}$$

for any $\mathbf{z}$ in the segment $[\mathbf{y}_1, \mathbf{y}_2]$. Therefore, by the Weyl inequality (Horn & Johnson, 1990, Theorem 4.3.1), we see that

$$\lambda^{[1]}(A_{[\mathbf{y}_1, \mathbf{z}]}) + \lambda^{[1]}(A_{[\mathbf{z}, \mathbf{y}_2]}) \leqslant \lambda^{[1]}(A_{[\mathbf{y}_1, \mathbf{y}_2]}). \tag{2.8}$$

Because $A_{[\mathbf{z}, \mathbf{y}_2]}$ is positive definite, it follows that

$$\lambda^{[1]}(A_{[\mathbf{y}_1, \mathbf{z}]}) \leqslant \lambda^{[1]}(A_{[\mathbf{y}_1, \mathbf{y}_2]})$$

[4]In the case that $\nabla^2 f(\mathbf{x}^*)$ is only positive semi-definite, the eigenvectors corresponding to the zero eigenvalue form a center manifold. A similar argument can be made by using the center manifold theory (Carr, 1981; Wang *et al.*, 2015). So as to focus on the main ideas, we assume the generic case of positive definiteness in this paper.

for all $\mathbf{z} \in [\mathbf{y}_1, \mathbf{y}_2]$. In particular,

$$\lambda^{[1]}(\nabla^2 f(\mathbf{y}_1)) = \lim_{\mathbf{z}\to\mathbf{y}_1} \lambda^{[1]}(A_{[\mathbf{y}_1,\mathbf{z}]}) \leqslant \lambda^{[1]}(A_{[\mathbf{y}_1,\mathbf{y}_2]}). \tag{2.9}$$

Applying the Courant-Fischer theorem to (2.7) and using (2.9), we may choose $v(t) \equiv -\lambda_{\mathscr{B}}$. By (2.3), we conclude that flows within this neighborhood contract toward each other when t advances. □

The fact that the vector field of gradients is contractive is of practical importance because, if one solution inadvertently jumps on to a neighboring trajectory, the contraction will bring the paths back to the same equilibrium point. This property gives us the leeway to follow the gradient trajectory loosely while keeping the equilibrium in sight.

2.2 *Contractive numerical methods*

When discretizing a contractive vector field for numerical calculation, we certainly wish that the property of contraction is preserved by the underlying numerical method. Toward this end, some sufficient conditions have been developed, known as the nonlinear stability theory, and some methods satisfying these conditions have been identified. Readers are referred to the book by Hairer & Wanner (1996, Sections IV.12, V.6, and V.9) and references contained therein for the in-depth discussion and history of development. A quick introduction and survey of some main results can be found in the books by Dekker & Verwer (1984) and Lambert (1991). Without repeating the details, we simply mention that a *G*-stable linear multi-step method (Hairer & Wanner, 1996, Definition 9.1) and an algebraically stable Runge-Kutta method (Hairer & Wanner, 1996, Definition 12.5) generate contractive numerical solutions. These elegant results are motivated by the desire of maintaining contractivity for the numerical solutions of general contractive differential systems. For our application, we want to tackle the gradient flows only. We do not need the full potency of those stability conditions.

One property that stands out for gradient flows is that the Hessian of any sufficiently smooth objective function $f(\mathbf{x})$ is always symmetric and, hence, has only real eigenvalues. For this specific class of differential systems, no complex eigenvalues will come into play. So any numerical ODE method whose region of absolute stability contains the negative x-axis, that is, A_0-stability, and maintains contractivity is sufficient for following a gradient flow. However, in what follows, we do not use any ODE integrator as is given! We do not adjust the step sizes in accordance with the conventional way where the main concern is to maintain the precision and the stability while tracking the entire trajectory! Though rooted in numerical ODE notions, we derive methods that should be regarded as new iterative schemes per se. It is this kind non-traditional combination that give us the edge of effectiveness.

We demonstrate our point by considering the following approach. Given the current approximate $\mathbf{y}_k$ to $\mathbf{y}(t_k)$ and a step size h_k, if the next step is to be determined by minimizing the total energy

$$\frac{1}{2h_k}\|\mathbf{y}_{k+1} - \mathbf{y}_k\|_2^2 + f(\mathbf{y}_{k+1}),$$

then $\mathbf{y}_{k+1}$ is given by the implicit Euler method (Hairer & Lubich, 2014; Stuart & Humphries, 1996). Instead of obtaining $\mathbf{y}_{k+1}$, we perform only one Newton iteration, immediately accept the outcome, and continue to the next step, yielding the iterative scheme

$$\mathbf{x}_{k+1} = \phi(\mathbf{x}_k; \varepsilon_k), \tag{2.10}$$

where $\phi : \mathbb{R}^n \to \mathbb{R}^n$ is defined by

$$\phi(\mathbf{x}; \varepsilon) := \mathbf{x} - \left(\frac{1}{\varepsilon} I_n + \nabla^2 f(\mathbf{x})\right)^{-1} \nabla f(\mathbf{x}) \tag{2.11}$$

and ε is regarded as a parameter. In the meantime, the step size ε_k is varied according to the so-called "switched evolution relaxation (SER)" strategy (van Leer & Mulder, 1985)

$$\varepsilon_{k+1} = \varepsilon_k \frac{\|\nabla f(\mathbf{x}_k)\|_2}{\|\nabla f(\mathbf{x}_{k+1})\|_2}, \tag{2.12}$$

which has the characteristics of being relatively small in the initial phase and becoming large when $\nabla f(\mathbf{x}_{k+1})$ gets smaller in the terminal phase of the iteration. This scheme, known as the pseudo-transient iteration, is a special implicit upwind method used in the PDE community (Mulder & van Leer, 1985) for computing steady-state solutions. At the terminal phase it can be argued that $\varepsilon_{k+1} \geqslant 2\varepsilon_k$ (Kelley & Keyes, 1998, Lemma 2.1). So, the method eventually transitions itself into a fast converging Newton iteration.

The scheme fits in the frame of an inexact Newton method (Dembo *et al.*, 1982) because, from (2.10), it is easy to see that

$$\|\nabla^2 f(\mathbf{x}_k)\Delta\mathbf{x}_k + \nabla f(\mathbf{x}_k)\|_2 \leqslant \eta_k \|\nabla f(\mathbf{x}_k)\|_2, \tag{2.13}$$

where

$$\eta_k := \| \left(I_n + \varepsilon_k \nabla^2 f(\mathbf{x}_k)\right)^{-1} \|_2. \tag{2.14}$$

Assume the generic condition that $\nabla^2 f(\mathbf{x}^*)$ is positive definite and denote its smallest eigenvalue by $\mu^{[1]}$. Then by continuity, if $\mathbf{x}_k$ is sufficiently close to $\mathbf{x}^*$, we estimate from (2.14) that

$$\eta_k \approx \frac{1}{1+\varepsilon_k \mu^{[1]}}. \tag{2.15}$$

In this case, it is further known that

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\|_2 = O(\eta_k \|\mathbf{x}_k - \mathbf{x}^*\|_2 + \|\mathbf{x}_k - \mathbf{x}^*\|_2^2). \tag{2.16}$$

Therefore $\mathbf{x}_k$ converges at least superlinearly and gradually behaves like the regular Newton iterates when ε_k goes to infinity (Dembo *et al.*, 1982).

We claim that (2.10) is also a contractive method. We divide the argument into two parts. The action of the Fréchet derivative of ϕ on a vector $\mathbf{d}$ at a general point $\mathbf{x}$ is given by

$$\begin{aligned} \phi'(\mathbf{x};\varepsilon).\mathbf{d} &= \left(I_n - \left(\frac{(\nabla^2 f(\mathbf{x}))^{-1}}{\varepsilon} + I_n\right)^{-1}\right)\mathbf{d} \\ &\quad -\varepsilon^2 \left(\left(I_n + \varepsilon\nabla^2 f(\mathbf{x})\right)^{-1} \left(\nabla^3 f(\mathbf{x}).\mathbf{d}\right) \left(I_n + \varepsilon\nabla^2 f(\mathbf{x})\right)^{-1}\right) \nabla f(\mathbf{x}), \end{aligned} \tag{2.17}$$

where $\nabla^3 f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ is the Fréchet derivative of $\nabla^2 f(\mathbf{x})$ and is a tensor-to-vector multiplication. At the initial stage, ε is small, we may thus ignore the effect of the second order term and conclude that

$$\|\phi(\mathbf{x}+\mathbf{d}) - \phi(\mathbf{x}\| \leqslant c\|\mathbf{d}\| \tag{2.18}$$

for some $c < 1$ at any $\mathbf{x}$ where $\nabla^2 f(\mathbf{x})$ is positive definite. When ε is large, such an argument does not work. However, the method will use large step sizes only at the later stage when the iterates $\mathbf{x}_k$ are close to $\mathbf{x}^*$. The Jacobian at the equilibrium point $\mathbf{x}^*$ satisfies

$$\left\| \frac{d\phi(\mathbf{x}^*;\varepsilon)}{d\mathbf{x}} \right\|_2 = \frac{1}{1+\varepsilon\mu^{[1]}} < 1. \tag{2.19}$$

It follows that if $\{\mathbf{y}_k\}$ and $\{\mathbf{z}_k\}$ are two sequences generated by (2.10) using the same sequence $\{\varepsilon_k\}$ of step sizes and if $\mathbf{y}_k, \mathbf{z}_k$ are sufficiently close to $\mathbf{x}^*$, then by (2.19) we see that

$$\|\mathbf{y}_{k+1} - \mathbf{z}_{k+1}\|_2 = \|\phi(\mathbf{y}_k; \varepsilon_k) - \phi(\mathbf{z}_k; \varepsilon_k)\|_2 \leqslant \nu_k \|\mathbf{y}_k - \mathbf{z}_k\|_2 \tag{2.20}$$

for some $\nu_k < 1$. At this stage, ε_k is expected to be large.

Different from the energy-diminishing approach discussed by Hairer & Lubich (2014), the idea of the pseudo-transient method is to start the scheme with a small ε_k to let the iteration process mimic an explicit, time-accurate integration. This allows to maintain reasonable closeness to the gradient trajectory. Once $\mathbf{x}_k$ begins to sense the attraction of the final solution, indicated by diminishing values of $\nabla f(\mathbf{x}_k)$, we take large ε_k to mimic the Newton iteration which allows to attain fast convergence to $\mathbf{x}^*$. Our interest in this paper is to search for some other gradient integrators of similar nature and test their workability.

3. High-order contractive methods

In this section, we propose three classes of schemes. We argue that, when properly augmented, they have similar convergence properties as the pseudo-transient method.

3.1 *Backward differentiation formulas*

One important feature of the classical p-step backward differentiation formulas (BDF), $p = 1, \ldots, 6$,

$$\mathbf{y}_{k+1} = \sum_{j=0}^{p-1} \alpha_j \mathbf{y}_{k-j} + h\beta \mathbf{f}(\mathbf{y}_{k+1}), \tag{3.1}$$

is that they contain the negative real axis in their regions of absolute stability. Applied to the gradient flows, this means that the choice of any step sizes will not exhibit instability. On the other hand, it is generally expected that high-order methods can integrate faster because they allow larger step sizes while maintaining the desirable precision. These attributes together seem favorable for following a gradient flow, especially in the initial stage — while staying relatively close to the gradient trajectory for guidance, use larger step sizes so as to move quickly into the final stage for finding the stable equilibrium.

We do not employ the well implemented, high-order, BDF-based stiff solvers for gradient flows. Instead, we propose to use these schemes in the same spirit as the pseudo-transient iteration for gradient flows. That is, starting from $\mathbf{x}_k$ with step size ε_k, we advance to $\mathbf{x}_{k+1}$ by taking only one Newton iteration as

$$\mathbf{x}_{k+1} := \mathbf{x}_k + \left(\frac{1}{\varepsilon_k \beta} I_n + \nabla^2 f(\mathbf{x}_k)\right)^{-1} \left(\frac{1}{\varepsilon_k \beta}\left(\sum_{j=0}^{p-1} \alpha_j \mathbf{x}_{k-j} - \mathbf{x}_k\right) - \nabla f(\mathbf{x}_k)\right). \tag{3.2}$$

Such an iteration is no more expensive than that in (2.10), except for a few extra memories of past values and vector arithmetic operations. The following result shows that, similar to the pseudo-transient method, the more sophisticated scheme has the desirable property of contractivity.

LEMMA 3.1 The multi-step iterative scheme (3.2) applied to the gradient flow is a contraction.

Proof. We shall consider only the contractivity nearby the equilibrium point $\mathbf{x}^*$. For small step size, the argument is essentially the same as that in the preceding section by dropping the higher order term in the Fréchet derivative of the following function.

Through the vectorization

$$X_k := \left[\mathbf{x}_k^\top, \ldots, \mathbf{x}_{k-p+1}^\top\right]^\top \in \mathbb{R}^{pn}, \tag{3.3}$$

we may regard the iteration (3.2) as a fixed-point iteration

$$X_{k+1} = \psi(X_k; \varepsilon_k), \tag{3.4}$$

with

$$\psi(X;\varepsilon) := \begin{bmatrix} \mathbf{x}^{[0]} + \left(\frac{1}{\varepsilon\beta} I_n + \nabla^2 f(\mathbf{x}^{[0]})\right)^{-1} \left(\frac{1}{\varepsilon\beta}\left(\sum_{j=0}^{p-1} \alpha_j \mathbf{x}^{[j]} - \mathbf{x}^{[0]}\right) - \nabla f(\mathbf{x}^{[0]})\right) \\ \mathbf{x}^{[0]} \\ \vdots \\ \mathbf{x}^{[p-2]} \end{bmatrix} \tag{3.5}$$

where X is partitioned in blocks as $X = \left[\mathbf{x}^{[0]\top}, \ldots, \mathbf{x}^{[p-1]\top}\right]^\top$ with $\mathbf{x}^{[j]} \in \mathbb{R}^n$. Then the Jacobian of ψ at $X^* = [\mathbf{x}^{*\top}, \ldots, \mathbf{x}^{*\top}]^\top$ is given by

$$\frac{d\psi(X^*)}{dX} = \begin{bmatrix} \Omega(\varepsilon)\frac{\alpha_0}{\varepsilon\beta} & \Omega(\varepsilon)\frac{\alpha_1}{\varepsilon\beta} & & \ldots & & \Omega(\varepsilon)\frac{\alpha_{p-1}}{\varepsilon\beta} \\ I_n & 0 & 0 & \ldots & & 0 \\ 0 & I_n & 0 & & & 0 \\ & & & \ddots & & \\ \vdots & & \vdots & & & \vdots \\ 0 & 0 & 0 & \ldots & I_n & 0 \end{bmatrix}, \tag{3.6}$$

where for convenience we introduce the abbreviation

$$\Omega(\varepsilon) := \left(\frac{1}{\varepsilon\beta} I_n + \nabla^2 f(\mathbf{x}^*)\right)^{-1}. \tag{3.7}$$

The Jacobian (3.6) appears as the structure of a block companion matrix. So eigenvalues of (3.6) are precisely those of the matrix polynomial

$$p(\lambda) \quad := \quad I_n\lambda^p - \Omega(\varepsilon)\frac{\alpha_0}{\varepsilon\beta}\lambda^{p-1} - \Omega(\varepsilon)\frac{\alpha_1}{\varepsilon\beta}\lambda^{p-2} - \ldots - \Omega(\varepsilon)\frac{\alpha_{p-1}}{\varepsilon\beta}. \tag{3.8}$$

Let the eigenvalues of $\nabla^2 f(\mathbf{x}^*)$ be denoted by $\mu^{[1]} \leqslant \mu^{[2]} \leqslant \ldots \leqslant \mu^{[n-1]} \leqslant \mu^{[n]}$. It suffices to consider the roots of the polynomials

$$q(\lambda; \mu^{[i]}) := \lambda^p - \frac{\alpha_0}{(1+\varepsilon\beta\mu^{[i]})}\lambda^{p-1} - \ldots - \frac{\alpha_{p-1}}{(1+\varepsilon\beta\mu^{[i]})}, \quad i = 1, \ldots, n. \tag{3.9}$$

If the step size ε is large enough such that

$$\sum_{j=0}^{p-1} |\alpha_j| < 1 + \varepsilon\beta\mu^{[1]}, \tag{3.10}$$

then, using the Rouché theorem, we see that all roots are bounded within the unit disk. □

What is important is the following lemma asserting that the scheme (3.2) can also be regarded as an inexact Newton method.

LEMMA 3.2 Define $\zeta_0 = 1$ and

$$\zeta_j := 1 - \sum_{i=0}^{j-1} \alpha_i, \quad j = 1, \ldots p-1. \tag{3.11}$$

Then the iteration (3.2) is equivalent to

$$\nabla^2 f(\mathbf{x}_k)\Delta\mathbf{x}_k + \nabla f(\mathbf{x}_k) = -\frac{1}{\varepsilon_k}\left(\sum_{j=0}^{p-1} \frac{\zeta_j}{\beta}\Delta\mathbf{x}_{k-j}\right), \tag{3.12}$$

whereas the summation on the right side of (3.12) is an average of the time series $\Delta\mathbf{x}_k, \ldots, \Delta\mathbf{x}_{k-p+1}$.

Proof. For each fixed p, the coefficients of a BDF in the form (3.1) satisfy necessarily the basic algebraic relationships (Hairer & Wanner, 1996; Lambert, 1991)

$$\sum_{j=0}^{p-1} \alpha_j = 1, \tag{3.13}$$

$$\sum_{j=0}^{p-1} (j+1)\alpha_j = \beta, \tag{3.14}$$

Using (3.13), we can rewrite the summation on the right side of (3.2) recursively as

$$\mathbf{x}_k - \sum_{j=0}^{p-1} \alpha_j \mathbf{x}_{k-j} = \sum_{j=0}^{p-1} \alpha_j (\mathbf{x}_k - \mathbf{x}_{k-j}) = \sum_{j=1}^{p-1} \left(1 - \sum_{i=0}^{j-1} \alpha_i\right) \Delta\mathbf{x}_{k-j}.$$

An rearrangement of terms in (3.2) leads to (3.12). To see the averaging effect, the identity

$$\sum_{j=0}^{p-1} \zeta_j = \beta \tag{3.15}$$

is simply a rearrangement of the summation in (3.14). □

It should be noted that some of the weights $\frac{\zeta_j}{\beta}$ in (3.12) might be negative, but they sum to be 1. The average nature

$$\Delta\overline{\mathbf{x}}_k := \sum_{j=0}^{p-1} \frac{\zeta_j}{\beta}\Delta\mathbf{x}_{k-j} \tag{3.16}$$

serves as a good estimate of the convergence in the following sense which generalizes that in (2.14) for $p > 1$.

LEMMA 3.3 Suppose that the SER strategy (2.12) is used. Then the sequence $\{\Delta\overline{\mathbf{x}}_k\}$ determines the rate of convergence. In particular, if $\lim_{k\to\infty} \Delta\overline{\mathbf{x}}_k = 0$, then $\mathbf{x}_k$ converges to a stationary point $\mathbf{x}^*$ superlinearly.

Proof. By Lemma 3.2, the residual of the (Newton) iteration is given by $\mathbf{r}_k := -\frac{1}{\varepsilon_k}\Delta\overline{\mathbf{x}}_k$. By the theory of the inexact Newton method, the rate of convergence depends on the forcing sequence (Dembo *et al.*, 1982, Corollary 3.5)

$$\eta_k := \frac{\|\mathbf{r}_k\|_2}{\|\nabla f(\mathbf{x}_k)\|_2} = \frac{1}{c}\Delta\overline{\mathbf{x}}_k, \tag{3.17}$$

where $c := \varepsilon_0 \|\nabla f(\mathbf{x}_0)\|_2$ is a constant determined by the first step size ε_0. □

By now, most of properties known for the pseudo-transient method have been generalized to the class of BDFs. Of particular usefulness is that a higher order BDF has the advantages of allowing larger steps in the initial stage when tracing a gradient trajectory and, when $\mathbf{x}_k$ is close enough to x^*, it transitions itself to a faster Newton method.

3.2 *Nordsieck methods*

An important point must be further clarified. The BDFs are used only to motivate the relationship (3.2) which is regarded as a stand-alone iterative scheme for the sequence $\{\mathbf{x}_k\}$. However, the BDFs we refer to are all based on the assumption that a fixed step size h is used for all values $\{\mathbf{y}_{k+1}, \mathbf{y}_k, \ldots\}$, whereas in our iteration the step sizes are changed frequently. Can the multi-step scheme (3.2) be used as is? There is no such a concern for the pseudo-transient method (2.10) because it is a one-step method. Would it be more appropriate to first rewrite the multi-step BDFs in the one-step Nordsieck form (Hairer & Wanner, 1996; Lambert, 1991) before employing the SER strategy? This section is an attempt to address this question.

For $p = 1, \ldots, 6$, define the $n \times (p+1)$ matrix

$$Z_k := \left[\mathbf{y}_k, h\mathbf{y}'_k, \frac{h^2}{2}\mathbf{y}''_k, \ldots, \frac{h^p}{p!}\mathbf{y}_k^{(p)} \right], \tag{3.18}$$

where $\mathbf{y}_k^{(j)}$ is an approximation to $\mathbf{y}^{(j)}(\mathbf{x}_k)$. Then the p-step BDF method is equivalent to (Hairer *et al.*, 1993, Section III.6)

$$Z_{k+1} = Z_k P^\top + \left(h\mathbf{f}(\mathbf{y}_{k+1}) - Z_k[0, 1, 2, \ldots, p]^\top \right) \ell^\top, \tag{3.19}$$

where P is the $(p+1) \times (p+1)$ Pascal matrix

$$P := \begin{bmatrix} 1 & 1 & 1 & 1 & \ldots & 1 \\ 0 & 1 & 2 & 3 & & p \\ 0 & 0 & 1 & 3 & & \\ \vdots & & & & & \\ 0 & 0 & 0 & & \ldots & 1 \end{bmatrix},$$

$\ell = [\beta, 1, \ell_2, \ldots, \ell_p]^\top$ is a specific $(p+1) \times 1$ vector whose other entries can be found in (Hairer *et al.*, 1993, Section III.6, Table 6.2). The first column in the equation (3.19) is a nonlinear system for $\mathbf{y}_{k+1}$, i.e.,

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \sum_{i=1}^{p} \frac{h^i}{i!}\mathbf{y}_k^{(i)} + \beta \left(h\mathbf{f}(\mathbf{y}_{k+1}) - \sum_{i=1}^{p} \frac{h^i}{(i-1)!}\mathbf{y}_k^{(i)} \right). \tag{3.20}$$

Once $\mathbf{y}_{k+1}$ is obtained, the other columns are explicitly determined.

Without repeating the details, for example, the 2-step BDF method

$$\mathbf{y}_{k+1} = \frac{4}{3}\mathbf{y}_k - \frac{1}{3}\mathbf{y}_{k-1} + \frac{2h}{3}\mathbf{f}(\mathbf{y}_{k+1}) \tag{3.21}$$

is equivalent to the scheme

$$\mathbf{y}_{k+1} \quad = \quad \mathbf{y}_k + \frac{h}{3}\mathbf{y}'_k - \frac{h^2}{6}\mathbf{y}''_k + \frac{2h}{3}\mathbf{f}(\mathbf{y}_{k+1}). \tag{3.22}$$

The difference between (3.21) and (3.22) is that the former requires the past value $\mathbf{y}_{k-1}$ with a fixed step size h, while the latter depends solely on the current value $\mathbf{y}_k$ and its derivatives and, hence, h can be arbitrary. Applying (3.22) to the gradient flow by taking one Newton correction with step size ε_k yields the scheme

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \left(\frac{3}{2\varepsilon_k} I_n + \nabla^2 f(\mathbf{x}_k)\right)^{-1} \left(\frac{3}{2\varepsilon_k}\left(\varepsilon_k \mathbf{x}_k' - \frac{1}{3}\frac{\varepsilon_k^2}{2}\mathbf{x}_k''\right)\right). \tag{3.23}$$

If the step size is changed to ε_{k+1}, then we are ready for the next iteration by taking

$$\varepsilon_{k+1}\mathbf{x}_{k+1}' := -\varepsilon_{k+1}\nabla f(\mathbf{x}_{k+1}), \tag{3.24}$$

$$\frac{\varepsilon_{k+1}^2}{2}\mathbf{x}_{k+1}'' = -\frac{1}{3}\left(\frac{\varepsilon_{k+1}}{\varepsilon_k}\right)^2 \left(\varepsilon_k \mathbf{x}_k' - \frac{\varepsilon_k^2}{2}\mathbf{x}_k'' + \varepsilon_k \nabla f(\mathbf{x}_{k+1})\right). \tag{3.25}$$

This Nordsieck-based scheme is not equivalent to the BDF-based scheme.

LEMMA 3.4 If $\mathbf{x}_0$ is sufficiently close to $\mathbf{x}^*$, then the iterates generated by (3.23) converge to $\mathbf{x}^*$.

Proof. It is necessary to consider the local behavior of $\mathbf{x}_k, \mathbf{x}_k', \mathbf{x}_k''$ simultaneously. Define the column vector $W_k := [\mathbf{x}_k^\top, \mathbf{x}_k'^\top, \mathbf{x}_k''^\top]^\top$. Then the Nordsieck-based method for $p = 2$ can be written as a fixed-point iteration

$$W_{k+1} = \vartheta(W_k; \varepsilon_k), \tag{3.26}$$

where

$$\vartheta(W;\varepsilon) =: = \begin{bmatrix} \mathbf{w}^{[0]} + \left(I_n + \frac{2\varepsilon}{3}\nabla^2 f\left(\mathbf{w}^{[0]}\right)\right)^{-1}\left(\varepsilon \mathbf{w}^{[1]} - \frac{\varepsilon^2}{6}\mathbf{w}^{[2]}\right) \\ -\nabla f\left(\mathbf{w}^{[0]} + \left(I_n + \frac{2\varepsilon}{3}\nabla^2 f\left(\mathbf{w}^{[0]}\right)\right)^{-1}\left(\varepsilon \mathbf{w}^{[1]} - \frac{\varepsilon^2}{6}\mathbf{w}^{[2]}\right)\right) \\ \frac{2}{3\varepsilon}\left(-\nabla f\left(\mathbf{w}^{[0]} + \left(I_n + \frac{2\varepsilon}{3}\nabla^2 f\left(\mathbf{w}^{[0]}\right)\right)^{-1}\left(\varepsilon \mathbf{w}^{[1]} - \frac{\varepsilon^2}{6}\mathbf{w}^{[2]}\right)\right) - \mathbf{w}^{[1]} + \frac{\varepsilon}{2}\mathbf{w}^{[2]}\right) \end{bmatrix},$$

and W is partitioned in blocks as $W = \left[\mathbf{w}^{[0]^\top}, \mathbf{w}^{[1]^\top}, \mathbf{w}^{[2]}\right]^\top$ with $\mathbf{w}^{(j)} \in \mathbb{R}^n$. It is easy to check that the fixed point of (3.26) must be of the form $(\mathbf{x}^*, \mathbf{0}, \mathbf{0})$. The question is whether it is reachable. The spectrum of the Jacobian matrix of ϑ at the equilibrium point $(\mathbf{x}^*, \mathbf{0}, \mathbf{0})$ consists of eigenvalues of each of the matrices

$$\begin{bmatrix} 1 & \frac{\varepsilon}{1+\frac{2}{3}\varepsilon\mu^{[i]}} & -\frac{\varepsilon^2}{6\left(1+\frac{2}{3}\varepsilon\mu^{[i]}\right)} \\ -\mu^{[i]} & -\frac{\varepsilon\mu^{[i]}}{1+\frac{2}{3}\varepsilon\mu^{[i]}} & \frac{\varepsilon^2\mu^{[i]}}{6\left(1+\frac{2}{3}\varepsilon\mu^{[i]}\right)} \\ -\frac{2}{3\varepsilon}\mu^{[i]} & -\frac{2}{3\varepsilon}\left(\frac{\varepsilon\mu^{[i]}}{\left(1+\frac{2}{3}\varepsilon\mu^{[i]}\right)}+1\right) & \frac{2}{3\varepsilon}\left(\frac{\varepsilon^2\mu^{[i]}}{6\left(1+\frac{2}{3}\varepsilon\mu^{[i]}\right)}+\frac{\varepsilon}{2}\right) \end{bmatrix}, \quad i = 1, \ldots n, \tag{3.27}$$

which are

$$0, \frac{2 \pm \sqrt{1 - 2\varepsilon\mu^{[i]}}}{3 + 2\varepsilon\mu^{[i]}}, \quad i = 1, \ldots n, \tag{3.28}$$

respectively. For each $i = 1, \ldots n$, all eigenvalues are in the unit disk for any step sizes ε and converge to zero when ε goes to infinity. Hence, the local convergence by the iteration (3.23) is expected. □

The above argument can be extended to the general $p = 1, \ldots, 6$. The following lemma shows that we may still write the scheme derived from the Nordsieck-based methods as an inexact Newton iteration.

LEMMA 3.5 For a fixed $p = 1, \ldots, 6$, let $\mathbf{x}_{k+1}$ denote the result of one Newton step of the nonlinear equation (3.20) applied to the gradient flow with step size ε_k and starting value $\mathbf{x}_k$. Then

$$\nabla^2 f(\mathbf{x}_k)\Delta\mathbf{x}_k + \nabla f(\mathbf{x}_k) = \nabla f(\widehat{\mathbf{y}}_{k+1}) + \frac{1}{\beta\varepsilon_k}(\widehat{\mathbf{y}}_{k+1} - \mathbf{x}_{k+1}), \tag{3.29}$$

where $\widehat{\mathbf{y}}_{k+1}$, depending on fixed $\mathbf{x}_k$ and ε_k, is the exact solution to (3.20).

Proof. One Newton step applied to (3.20) leads to

$$\left(I_n + \beta\varepsilon_k\nabla^2 f(\mathbf{x}_k)\right)\Delta\mathbf{x}_k = \sum_{i=1}^{p}\frac{h^i}{i!}\mathbf{x}_k^{(i)} - \beta\sum_{i=1}^{p}\frac{h^i}{(i-1)!}\mathbf{x}_k^{(i)} - \beta\varepsilon_k\nabla f(\mathbf{x}_k).$$

A rearrangement shows that

$$\beta\varepsilon_k\left(\nabla^2 f(\mathbf{x}_k)\Delta\mathbf{x}_k + \nabla f(\mathbf{x}_k)\right) \quad = \quad (\widehat{\mathbf{y}}_{k+1} - \mathbf{x}_k + \beta\varepsilon_k\nabla f(\widehat{\mathbf{y}}_{k+1})) - \Delta\mathbf{x}_k,$$

whereas the right hand side follows from (3.20). □

If the SER strategy is used in this one-step Nordsieck setting, then the forcing sequence becomes

$$\eta_k = \frac{1}{\beta c}\|\beta\varepsilon_k\nabla f(\widehat{\mathbf{y}}_{k+1}) + (\widehat{\mathbf{y}}_{k+1} - \mathbf{x}_{k+1})\|_2. \tag{3.30}$$

An intuitive way to see why η_k would be small is that if $\{\mathbf{x}_k\}$ ever converges, regardless of the step size taken as we have argued in Lemma 3.4, then the sequence $\{\widehat{\mathbf{y}}_k\}$ should behave like

$$\widehat{\mathbf{y}}_{k+1} \approx \mathbf{x}^* - \beta\varepsilon_k\nabla f(\widehat{\mathbf{y}}_{k+1}) \approx \mathbf{x}_{k+1} - \beta\varepsilon_k\nabla f(\widehat{\mathbf{y}}_{k+1}). \tag{3.31}$$

As such, there is a cancelation in (3.30). In our numerical experiment, however, we find that the convergence rate of the Nordsieck methods such as (3.23)-(3.25) is no better than that observed in (3.12) in general. Perhaps a different step size selection strategy can make a difference. Several plausible alternatives have been proposed by Eisenstat & Walker (1996).

3.3 *Numerical differentiation formulas*

A further modification of the BDF's is the so called numerical differentiation formulas (NDF)

$$\mathbf{y}_{k+1} = \sum_{j=0}^{p-1}\alpha_j\mathbf{y}_{k-j} + h\beta\mathbf{f}(\mathbf{y}_{k+1}) + \beta\kappa\gamma_p\left(\mathbf{y}_{k+1} - \widetilde{\mathbf{y}}_{k+1}\right), \tag{3.32}$$

where

$$\widetilde{\mathbf{y}}_{k+1} := \sum_{m=0}^{p}\nabla^m\mathbf{y}_k \tag{3.33}$$

is a known quantity from previous steps via the back difference operator $\nabla \mathbf{y}_k := \mathbf{y}_k - \mathbf{y}_{k-1}$, $\gamma_p := \sum_{j=1}^{p} \frac{1}{j}$, and κ is chosen to make the NDF's more accurate than the BDF's at the cost of less stability (Shampine & Reichelt, 1997). It is reported, for instance, that the first-order formula NDF1

$$\mathbf{y}_{k+1} - \mathbf{y}_k - \kappa(\mathbf{y}_{k+1} - 2\mathbf{y}_k + \mathbf{y}_{k-1}) = h\mathbf{f}(\mathbf{y}_{k+1}) \tag{3.34}$$

with $\kappa = -0.1850$ can achieve the same accuracy as BDF1 with a step size about 26% bigger (Shampine & Reichelt, 1997, Table 1). This class of methods has been implemented in the solver `ode15s` in the MATLAB ODE suite.

Similar to (3.2), one Newton iteration of (3.32) with step size ε_k applied to the gradient flow leads to

$$\mathbf{x}_{k+1} := \mathbf{x}_k - \left(\frac{1-\kappa\gamma_p\beta}{\varepsilon_k\beta} I_n + \nabla^2 f(\mathbf{x}_k) \right)^{-1} \left(\nabla f(\mathbf{x}_k) - \frac{1}{\varepsilon_k\beta} \left(\sum_{j=0}^{p-1} \alpha_j \mathbf{x}_{k-j} - \mathbf{x}_k + \kappa\gamma_p\beta \left(\mathbf{x}_k - \widetilde{\mathbf{x}}_{k+1} \right) \right) \right). \tag{3.35}$$

We claim that the iterative scheme (3.35) is still an inexact Newton method with averaging.

LEMMA 3.6 Denote $\eta_0 := -1$ and

$$\eta_j := \sum_{m=j}^{p} (-1)^{j-1} \binom{m-1}{j-1}, \quad j = 1, \dots p. \tag{3.36}$$

Then (3.35) is equivalent to

$$\nabla^2 f(\mathbf{x}_k)\Delta\mathbf{x}_k + \nabla f(\mathbf{x}_k) = -\frac{1}{\varepsilon_k} \left(\sum_{j=0}^{p-1} \left(\frac{\zeta_j}{\beta} + \kappa\gamma_p\eta_j \right) \Delta\mathbf{x}_{k-j} + \kappa\gamma_p\eta_p\Delta\mathbf{x}_{k-p} \right). \tag{3.37}$$

In particular, the summation on the right side of (3.37) is an average of the time series $\Delta\mathbf{x}_k, \dots, \Delta\mathbf{x}_{k-p}$.

Proof. Recall the identity that

$$\nabla^m \mathbf{y}_k = \sum_{j=0}^{m} (-1)^j \binom{m}{j} \mathbf{y}_{k-j}.$$

We thus can write the intermediate vector $\widetilde{\mathbf{x}}_{k+1}$ as

$$\begin{aligned} \widetilde{\mathbf{x}}_{k+1} &= \mathbf{x}_k + \sum_{m=1}^{p} \nabla^m \mathbf{x}_k = \mathbf{x}_k + \sum_{m=1}^{p} \nabla^{m-1} \Delta\mathbf{x}_{k-1} = \mathbf{x}_k + \sum_{m=1}^{p} \sum_{j=0}^{m-1} (-1)^j \binom{m-1}{j} \Delta\mathbf{x}_{k-1-j} \\ &= \mathbf{x}_k + \sum_{j=0}^{p-1} \sum_{m=j+1}^{p} (-1)^j \binom{m-1}{j} \Delta\mathbf{x}_{k-1-j} = \mathbf{x}_k + \sum_{j=1}^{p} \eta_j \Delta\mathbf{x}_{k-j}. \end{aligned}$$

Upon substituting $\widetilde{\mathbf{x}}_{k+1}$ into (3.35) and rearranging terms, we obtain (3.37). Note that by definition of η_j we have

$$\sum_{j=0}^{p} \eta_j = 0. \tag{3.38}$$

Together with (3.15), we see that the coefficients of the time series $\Delta\mathbf{x}_k, \dots, \Delta\mathbf{x}_{k-p}$ sum to the unity. So the right side of (3.35) can be regarded as an average. □

Note that (3.37) involves one extra memory than (3.12). Since the iteration (3.35) can be regarded as an inexact Newton method, an argument similar to that in the preceding section can establish that the NRF-based iteration (3.12) is also contractive.

3.4 *Linear implicit formulas*

Rosenbrock methods are one-step method of the form

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\sum_{i=1}^{s} b_i \mathbf{k}_i, \tag{3.39}$$

where each $\mathbf{k}_i$, $i = 1, 2, \ldots, s$, is obtained consecutively from solving

$$W\mathbf{k}_i = \mathbf{f}\left(\mathbf{y}_n + h\sum_{j=1}^{i-1} a_{ij}\mathbf{k}_j\right) + hJ\sum_{j=1}^{i-1} d_{ij}\mathbf{k}_j, \tag{3.40}$$

with

$$\begin{aligned} W &:= I - hdJ, \\ J &:= \frac{\partial \mathbf{f}(\mathbf{y}_n)}{\partial y}. \end{aligned}$$

These methods are said to be linearly implicit because, unlike the implicit Runge-Kutta methods, each $\mathbf{k}_i$ is obtained by solving systems of linear equations with the same coefficient matrix W. Of course, the coefficients are supposed to satisfy a certain specific equations in order to achieve a desired order (Steihaug & Wolfbrandt, 1979, Table 1). For instance, for an s-stage formula to have order two, it is required that

$$\sum_{i=1}^{s} b_i = 1, \tag{3.41}$$

$$\sum_{i=2}^{s} b_i \left(\sum_{j=1}^{i-1} a_{ij}\right) = \frac{1}{2}, \tag{3.42}$$

$$\sum_{i=2}^{s} b_i \left(\sum_{j=1}^{i-1} d_{ij}\right) + d = 0. \tag{3.43}$$

The pair

$$\begin{cases} W\mathbf{k}_1 &= \mathbf{f}(\mathbf{y}_n) \\ W\mathbf{k}_2 &= \mathbf{f}\left(\mathbf{y}_n + \frac{2}{3}h\mathbf{k}_1\right) - \frac{4}{3}hJ\mathbf{k}_1 \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + \frac{h}{4}(\mathbf{k}_1 + 3\mathbf{k}_2), \end{cases} \tag{3.44}$$

with $d = 1$, therefore is a 2-stage, A-stable, and second order formula (Steihaug & Wolfbrandt, 1979, Section 4). A modification to the triplet (Shampine & Reichelt, 1997, Section 3.1)

$$\begin{cases} W\mathbf{k}_1 &= \mathbf{f}(\mathbf{y}_n) \\ W\mathbf{k}_2 &= \mathbf{f}\left(\mathbf{y}_n + \frac{1}{2}h\mathbf{k}_1\right) - hdJ\mathbf{k}_1 \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + h\mathbf{k}_2 \\ W\mathbf{k}_3 &= \mathbf{f}(\mathbf{y}_n + h\mathbf{k}_2) + (e-2)hdJ\mathbf{k}_1 - ehdJ\mathbf{k}_2, \end{cases} \tag{3.45}$$

with $d = \frac{1}{2+\sqrt{2}}$ and $e = 6+\sqrt{2}$, has been implemented in the solver ode23s in the MATLAB ODE suite. This scheme has the advantage of FSAL (first same as last) and L-stability, but is still a second order method because $\mathbf{k}_3$ is used only for the error estimator and is not involved in advancing to $\mathbf{y}_{n+1}$. It is said that ode23s is effective at crude tolerances, which might be to our advantage since we only ask to stay nearby the gradient trajectory.

Different from other formulas, when applied to our gradient flow, the linear implicit formulas looks like but does not always assume the inexact Newton form. One step of Rosenbrock iteration (3.39) is equivalent to the equation

$$\frac{\partial \mathbf{f}(\mathbf{x_k})}{\partial \mathbf{y}} \Delta \mathbf{x}_k + \sum_{i=1}^{s} \frac{b_i}{d} \mathbf{f}\left(\mathbf{x}_k + \sum_{j=1}^{i-1} a_{ij} \Delta \mathbf{x}_k^{[j]} \right) = \frac{1}{hd} \Delta \mathbf{x}_k - \sum_{i=1}^{s} \frac{b_i}{d} \frac{\partial \mathbf{f}(\mathbf{x_k})}{\partial \mathbf{y}} \left(\sum_{j=1}^{i-1} d_{ij} \Delta \mathbf{x}_k^{[j]} \right)$$

for $\Delta \mathbf{x}_k$, where intermediate quantities

$$\Delta \mathbf{x}_k^{[j]} := h\mathbf{k}_j, \quad j = 1, \ldots, s$$

are known values in the process of calculation. Linearizing the second term yields an approximate system

$$\frac{\partial \mathbf{f}(\mathbf{x_k})}{\partial \mathbf{y}} \Delta \mathbf{x}_k + \frac{1}{d} \mathbf{f}(\mathbf{x}_k) = \frac{1}{hd} \Delta \mathbf{x}_k - \sum_{j=1}^{s-1} \frac{\omega_j}{d} \frac{\partial \mathbf{f}(\mathbf{x_k})}{\partial \mathbf{y}} \Delta \mathbf{x}_k^{[j]}, \tag{3.46}$$

where by using (3.42) and (3.43), the coefficients

$$\omega_j \quad := \quad \sum_{i=j+1}^{s} b_i (a_{ij} + d_{ij}), \quad j = 1, \ldots, s-1$$

satisfy

$$\sum_{j=1}^{s-1} \omega_j \quad = \quad \frac{1}{2} - d.$$

Note that even (3.46) is not in the inexact Newton form. The special case (3.44), however, is somewhat exceptional. It can be expressed as

$$\frac{\partial \mathbf{f}(\mathbf{x_k})}{\partial \mathbf{y}} \Delta \mathbf{x}_k + \frac{1}{4} \mathbf{f}(\mathbf{x}_k) + \frac{3}{4} \mathbf{f}\left(\mathbf{x}_k + \frac{2}{3} \Delta \mathbf{x}_k^{[1]} \right) = \frac{1}{h} \Delta \mathbf{x}_k + \frac{\partial \mathbf{f}(\mathbf{x_k})}{\partial \mathbf{y}} \Delta \mathbf{x}_k^{[1]}, \tag{3.47}$$

where $\Delta \mathbf{x}_k^{[1]}$, satisfying

$$\frac{\partial \mathbf{f}(\mathbf{x_k})}{\partial \mathbf{y}} \Delta \mathbf{x}_k^{[1]} + \mathbf{f}(\mathbf{x}_k) = \frac{1}{h} \Delta \mathbf{x}_k^{[1]},$$

is itself a one-step inexact Newton (2.10) discussed earlier. Overall, the system (3.47) can be considered as approximately

$$\frac{\partial \mathbf{f}(\mathbf{x_k})}{\partial \mathbf{y}} \Delta \mathbf{x}_k + \mathbf{f}(\mathbf{x}_k) = \frac{1}{h} \Delta \mathbf{x}_k + \frac{1}{2} \frac{\partial \mathbf{f}(\mathbf{x_k})}{\partial \mathbf{y}} \Delta \mathbf{x}_k^{[1]},$$

so the inexact Newton iteration is quite obvious in this case. On the other hand, for the *L*-stability, the choice for d is limited to $1 \pm \frac{1}{\sqrt{2}}$ in modified scheme (3.45) for the ode23s. In this case, the one-step Rosenbrock iteration may be cast as an approximation of

$$\frac{\partial \mathbf{f}(\mathbf{x_k})}{\partial \mathbf{y}} \Delta \mathbf{x}_k + \frac{1}{d}\mathbf{f}(\mathbf{x}_k) = \frac{1}{dh}\Delta \mathbf{x}_k + (1 - \frac{1}{2d})\frac{\partial \mathbf{f}(\mathbf{x_k})}{\partial \mathbf{y}} \Delta \mathbf{x}_k^{[1]}, \tag{3.48}$$

where obviously the weight scheme does not add up to one. The Rosenbrock-based iteration does not have the ability of transforming itself into a fast Newton method, but its L-stability and high efficiency for crude tolerance, together with a different step size selection strategy (see (4.2)), still make it a reasonable algorithm for gradient flows.

4. Numerical experiment

The purpose of numerical experiments reported in this section is merely to demonstrate the concepts discussed in the proceeding sections. Critical issues such as adapting the best step size or optimizing the implementation for each individual method have not been explored in this experiment. Rather, we employ the same SER strategy for all methods with the aim at comparing step sizes variation and checking whether the methods themselves will transform at the end stage. To set a point of reference, we use the solver ode23s in MATLAB as the base for evaluating the performance.

4.1 *Test problems*

There are quite a few test functions for unconstrained optimization. We report test results for the following five problems. One is from theoretical physics, three are from real-world applications, and one is from the benchmark collection by Moré *et al.* (1981).

Problem 1. The Müller-Brown potential energy surface (Müller & Brown, 1979):

$$\begin{aligned} f(x_1,x_2) &= -200e^{-(x_1-1)^2-10x_2^2} - 100e^{-x_1^2-10(x_2-0.5)^2} \\ &\quad -170e^{-6.5(x_1+0.5)^2+11(x_1+0.5)(x_2-1.5)-6.5(x_2-1.5)^2} + 15e^{0.7(x_1+1)^2+0.6(x_1+1)(x_2-1)+0.7(x_2-1)^2}; \\ \mathbf{x}_0 &= [-1.5,-0.5]. \end{aligned}$$

The remaining four test problems are of the nonlinear least squares form

$$f(\mathbf{x}) = \frac{1}{2}\|\boldsymbol{\phi}(\mathbf{x})\|_2^2,$$

where the ultimate goal is to find roots for certain application-specific functions $\boldsymbol{\phi}(\mathbf{x})$ defined below (Andrei, 2004).

Problem 2. Powell badly scaled function (Moré *et al.*, 1981):

$$\begin{aligned} \phi_1(\mathbf{x}) &= 10^4 x_1 x_2 - 1; \\ \phi_2(\mathbf{x}) &= e^{-x_1} + e^{-x_2} - 1.0001; \\ \mathbf{x}_0 &= [0,1]. \end{aligned}$$

Problem 3. Chemical equilibrium system (Meintjes & Morgan, 1990):

$$\begin{aligned}
\phi_1(\mathbf{x}) &= x_1x_2 + x_1 - 3x_5;\\
\phi_2(\mathbf{x}) &= 2x_1x_2 + x_1 + 3r_{10}x_2^2 + x_2x_3^2 + r_7x_2x_3 + r_9x_2x_4 + r_8x_2 - rx_5;\\
\phi_3(\mathbf{x}) &= 2x_2x_3^2 + r_7x_2x_3 + 2r_5x_3^2 + r_6x_3 - 8x_5;\\
\phi_4(\mathbf{x}) &= r_9x_2x_4 + 2x_4^2 - 4rx_5;\\
\phi_5(\mathbf{x}) &= x_1x_2 + x_1 + r_{10}x_2^2 + x_2x_3^2 + r_7x_2x_3 + r_9x_2x_4 + r_8x_2 + r_5x_3^2 + r_6x_3 + x_4^2 - 1;\\
\mathbf{x}_0 &= [1.0000, 0, 10.5000, 5.5000, 0.0500];
\end{aligned}$$

with the parameters

$$\begin{array}{lll}
r_5 = 0.193, & r_6 = 4.10622\times10^{-4}, & r_7 = 5.45177\times10^{-4}, \quad r_8 = 4.4975\times10^{-7},\\
r_9 = 3.40735\times10^{-5}, & r_{10} = 9.615\times10^{-7}, & r = 10.
\end{array}$$

Problem 4. Stead-state reaction rate equation (Shacham, 1986):

$$\begin{aligned}
\phi_1(\mathbf{x}) &= 1 - x_1 - k_1x_1x_6 + r_1x_4;\\
\phi_2(\mathbf{x}) &= 1 - x_2 - k_2x_2x_6 + r_2x_5;\\
\phi_3(\mathbf{x}) &= -x_3 + 2k_3x_4x_6;\\
\phi_4(\mathbf{x}) &= k_1x_1x_6 - r_1x_4 - k_3x_4x_5;\\
\phi_5(\mathbf{x}) &= 1.5(k_2x_2x_6 - r_2x_5) - k3x_4x_5;\\
\phi_6(\mathbf{x}) &= 1 - x_4 - x_5 - x_6;\\
\mathbf{x}_0 &= [1.0900, 1.0500, 0.0500, 0.9900, 0.0500, 0];
\end{aligned}$$

with the parameters

$$k_1 = 31.24, \quad k_2 = 0.272, \quad k_3 = 303.03, \quad r_1 = 2.062, \quad r_2 = 0.02.$$

Problem 5. Circuit design application (Ratschek & Rokne, 1993):

$$\begin{aligned}
\phi_k(\mathbf{x}) &= (1 - x_1x_2)x_3\left(e^{x_5(g_{1k} - g_{3k}x_710^{-3} - g_{5k}x_810^{-3})} - 1\right) - g_{5k} + g_{4k}x_2;\\
\phi_{k+4}(\mathbf{x}) &= (1 - x_1x_2)x_4\left(e^{x_6(g_{1k} - g_{2k} - g_{3k}x_710^{-3} - g_{4k}x_910^{-3})} - 1\right) - g_{5k}x_1 + g_{4k}, \quad k = 1,\ldots,4;\\
\phi_9(\mathbf{x}) &= x_1x_3 - x_2x_4;\\
\mathbf{x}_0 &= [0.75, 0.45, 0.9, 1.77, 8.5, 7.5, 5.5, 1.25, 1.88];
\end{aligned}$$

with

$$g = \begin{bmatrix}
0.4850 & 0.7520 & 0.8690 & 0.9820\\
0.3690 & 1.2540 & 0.7030 & 1.4550\\
5.2095 & 10.0677 & 22.9274 & 20.2153\\
23.3037 & 101.7790 & 111.4610 & 191.2670\\
28.5132 & 111.8467 & 134.3884 & 211.4823
\end{bmatrix}.$$

4.2 *Results from low order ODE solvers*

In this section we report statistics by first testing three low order ODE solvers on the above five problems. These methods are the ode23s which is order 2 and the BDF and NDF in ode15s with MaxOrder = 2. We do not preset the interval length of integration, but let the process terminate automatically when one of the following events occurs:

$$\|\nabla f(\mathbf{x}_k)\| < \text{TerTol} \quad \text{or} \quad \frac{\|\nabla f(\mathbf{x}_k)\|}{\|\nabla f(\mathbf{x}_0)\|} < \text{TerTol} \quad \text{or} \quad \|\mathbf{x}_k - \mathbf{x}_{k-3}\| < \text{TerTol}, \tag{4.1}$$

where the termination tolerance TerTol is set at 10^{-9}. The first criterion is that the current gradient is small enough. The second criterion is that, relative to the starting gradient, the current gradient has been improved enough. The third criterion is that the iterates with a relay of 3 steps do not change significantly. We are interested in low precision with the hope that the contractivity of the gradient dynamics will keep the convergence in bay, so we set the local tolerance AbsTol=RelTol in Matlab to be at either 10^{-2} or 10^{-3}. For these ODE solver testing, we use the in-built step size selection strategy which estimates the local error e_i in the entry y_i at each step and chooses step sizes to ascertain that the criterion

$$|e_i| \leqslant |y_i|\text{RelTol} + \text{AbsTol} \tag{4.2}$$

is satisfied. The main difference between this strategy and the SER strategy is that the latter does not care about local feedback. To compare with the overhead of closely following the trajectory, we also include statistics for using the full strength of ode15s with low tolerance 10^{-12}. Such a low tolerance certainly is not advantageous to ode23s, but the results clearly show why gradient dynamics should be treated differently from conventional flows.

Through this test, we learn of the following facts.

1. For high precision integration, the length of integration needed serves as an indicator of how long the true gradient trajectory takes to reach its equilibrium point.

2. For high tolerance integration, since the trajectory is not followed closely, it is expected that the integration will take longer time interval to meet the stopping criteria.

3. It is obvious from Tables 2 to 6 that in all tests, except for the badly scaled Problem 2, the ode23s with high tolerance requires far less time steps than BDF and NDF to meet the same stopping criteria[5]. In the next subsection we therefore compare the transient methods with only the ode23s.

4. It is worth noting that, although the integration lengths taken by these ODE solvers for Problems 2 to 4 are fairly long, the time steps required are only in hundreds, implying that in average the step size selection strategy (4.2) is still capable of adopting very large step sizes while satisfying the specified local tolerance condition.

[5]However, it is mentioned by Shampine & Reichelt (1997, Page 7) that "the current version of ode23s forms a new Jacobian at every step for several reasons. A formula of order 2 is most appropriate at crude tolerances. At such tolerances solution components often change significantly in the course of a single step, so it is often appropriate to form a new Jacobian."

method (tolerance)	integration length	time steps	failed steps	function evals	partial evals	LU decomp	linear solvers
ode23s (10^{-2})	0.09	21	3	71	21	24	72
BDF (10^{-2})	0.17	51	10	107	8	23	104
NDF (10^{-2})	0.14	43	5	93	6	16	90
ode23s (10^{-3})	0.08	35	4	115	35	39	117
BDF (10^{-3})	0.15	78	6	159	4	25	156
NDF (10^{-3})	0.17	69	6	134	4	20	131
BDF (10^{-12})	0.06	1441	14	2330	2	158	2328
NDF (10^{-12})	0.06	1428	12	372	2	140	2370

Table 2. Results by ODE for Problem 1. All methods converge to $[-0.5582, 1.4417]$, an approximate minimum of f.

method (tolerance)	integration length	time steps	failed steps	function evals	partial evals	LU decomp	linear solvers
ode23s (10^{-2})	9.20e+08	249	23	808	249	272	816
BDF (10^{-2})	1.69e+06	142	97	439	59	141	437
NDF (10^{-2})	3.45e+06	157	112	497	65	157	477
ode23s (10^{-3})	5.50e+09	159	10	501	159	169	507
BDF (10^{-3})	5.89e+05	157	58	395	43	100	393
NDF (10^{-3})	6.86e+06	210	118	603	75	174	601
BDF (10^{-12})	4.94e+04	1651	40	3179	24	264	3177
NDF (10^{-12})	4.94e+04	1646	42	3201	24	264	3199

Table 3. Results by ODE for Problem 2. All methods converge to $[1.0982e\text{–}05, 9.1061]$, an approximate root of $\boldsymbol{\phi}(\mathbf{x})$.

method (tolerance)	integration length	time steps	failed steps	function evals	partial evals	LU decomp	linear solvers
ode23s (10^{-2})	4.01e+09	253	8	777	253	261	783
BDF (10^{-2})	1.00e+10	526	366	1613	184	517	1608
NDF (10^{-2})	1.00e+08	462	342	1410	153	484	1460
ode23s (10^{-3})	6.57e+08	219	1	661	219	220	660
BDF (10^{-3})	4.70e+10	587	270	1499	167	418	1496
NDF (10^{-3})	1.33e+08	444	70	1474	444	514	1542
BDF (10^{-12})	1.01e+05	5133	140	11454	24	792	11452
NDF (10^{-12})	1.01e+05	4769	95	10517	24	484	10515

Table 4. Results by ODE for Problem 3. NDF (10^{-2}) converges to a local minimum of $\|\boldsymbol{\phi}(\mathbf{x})\|^2$. All other methods converge to $[0.0034, 31.3265, 0.0684, 0.8595, 0.0370]$, an approximate root of $\boldsymbol{\phi}(\mathbf{x})$.

4.3 *Results from high order BDF-based pseudo-transient methods*

In this section we report statistics by testing the BDF-based pseudo-transient methods (3.2) on the same five problems. Because in such a scheme all the other computational overheads are proportional to the time steps, it suffices to tabulate only the numbers of steps taken by each method. What is also important for the multi-step pseudo-transient methods is the starting values which affect which equilibrium point the iterates will converge to. There are several possible ways to generate the starting values. We conduct two experiments using different startup strategies.

The first experiment generates, from the initial value, the needed starting values via explicit Euler steps with initial step size $\varepsilon_0 = 10^{-2}$. We then vary the subsequent ε_k in accordance with the SER strategy and obtain results in Table 7. Recall that the SER strategy implies the constraint that

method (tolerance)	integration length	time steps	failed steps	function evals	partial evals	LU decomp	linear solvers
ode23s (10^{-2})	3.28e+05	66	0	200	66	66	198
BDF (10^{-2})	3.78e+06	159	86	445	61	134	443
NDF (10^{-2})	3.82e+06	168	103	494	68	151	492
ode23s (10^{-3})	1.78e+05	103	0	311	103	103	309
BDF (10^{-3})	1.08e+05	218	85	551	66	138	549
NDF (10^{-3})	5.88e+06	218	99	581	73	153	580
BDF (10^{-12})	5.39e+04	3378	282	8231	246	755	8229
NDF (10^{-12})	5.39e+05	2574	264	6091	247	622	6089

Table 5. Results by ODE for Problem 4. All methods converge to $[0.9743, 0.9828, 36.4266, 0.9357, 0.0001, 0.0642]$, an approximate root of $\boldsymbol{\phi}(\mathbf{x})$.

method (tolerance)	integration length	time steps	failed steps	function evals	partial evals	LU decomp	linear solvers
ode23s (10^{-2})	3.59e+03	86	14	288	86	100	300
BDF (10^{-2})	4.65e+04	139	53	361	32	92	358
NDF (10^{-2})	1.43e+05	105	42	272	31	72	270
ode23s (10^{-3})	2.43e+03	82	8	264	82	90	270
BDF (10^{-3})	1.81e+04	195	49	444	32	99	441
NDF (10^{-3})	6.85e+04	163	44	381	32	83	379
BDF (10^{-12})	6.01e+02	3358	56	7088	20	427	7086
NDF (10^{-12})	6.01e+02	3067	40	6170	21	395	6168

Table 6. Results by ODE for Problem 5. All converge to $[0.9000, 0.4500, 1.0000, 2.0001, 8.0000, 7.9997, 5.0000, 1.0000, -2.0001]$, an approximate root of $\boldsymbol{\phi}(\mathbf{x})$.

$\varepsilon_k \|\nabla f(\mathbf{x}_k)\|_2 = c$. Through this experiment, we make the following general observations:

1. Upon comparison with the time steps taken by the ode23s, the BDF-based pseudo-transient methods requires fewer iterations for convergence.

2. There are two cases in Table 7, marked by BK, where the iteration breaks down because of near singularity. This might have something to do with the step size selection. More understanding is needed here.

3. The use of higher-order methods does reduce the numbers of required iterations in some cases, but the overall effect is not immediately obvious. This might have something to do with the facts that we have not fine tuned the implementation and that the SER strategy, easy and proven for the order-1 pseudo-transient method, might not be as effective for higher-order methods. This topic deserves further research and experiment.

4. While the stopping criteria (4.1) are met, in contrast to those found by the ODE methods in the preceding subsection, the equilibria found by the pseudo-transient iterations in Table 7 generally are local minima of $\|\boldsymbol{\phi}(\mathbf{x})\|$, but are not roots of $\boldsymbol{\phi}(\mathbf{x})$.

We also make one special observation on the convergence behavior of (3.2). Taking the chemical combustion equilibrium system in Problem 3 as an illustration, plotted in Figure 1 are the history of gradient reduction and the variation of step sizes based on the SER strategy. The drawings clearly manifest the quadratic convergence of the gradient and the exponential growth of ε_k at the final stage

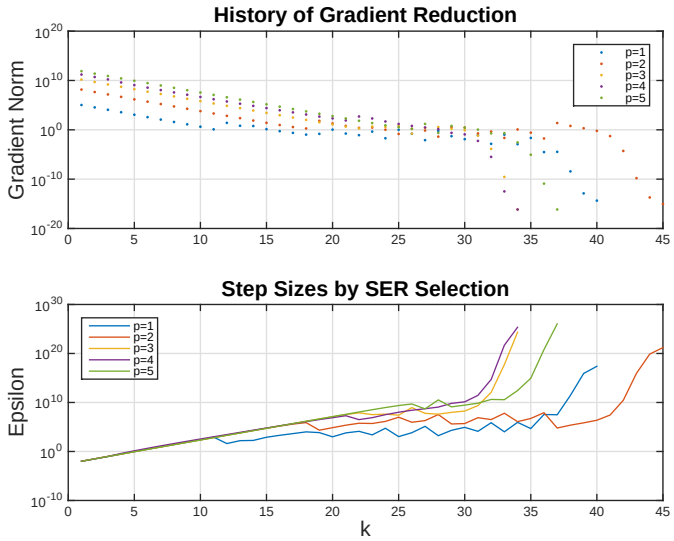


FIG. 1. Step size variations by the SER strategy and gradient reduction for Problem 3 by the BDF-based scheme (3.2).

of iteration, even though we are using merely the SER strategy as a rough estimator. At present, except for the order-1 pseudo-transient method (Kelley & Keyes, 1998), we have yet to establish a rigorous mathematical proof for the higher-order methods, but the same convergence behavior has been observed numerically for all cases.

	order 1	order 2	order 3	order 4	order 5
Problem 1	75	9	43	108	214
Problem 2	55	28	43	25	31
Problem 3	30	33	37	49	48
Problem 4	48	27	41	46	49
Problem 5	66	36	BK	BK	1290

Table 7. Results by BDF-based pseudo-transient methods: starting values generated by the explicit Euler method with step size 10^{-2}.

The second experiment generates the starting values by integrating the gradient dynamics (1.1) by means of ode23s (or any available ODE integrators) with the local tolerance 10^{-2} over a small interval, say $[0, 0.01]$, whence select needed starting points. This amounts to a short segment of the pseudo-transient phase. Summarized in Table 8 are the numbers of iterations needed by the same class of methods to meet the same stopping criteria (4.1). At first glance, it is disappointing that the pseudo-transient methods perform worse than using the ode23s alone. This exceeding overhead can be explained. Partially it suggests that, in comparison with (4.2), the SER strategy under this starting procedure is unable to boost the step sizes fast enough. There might be room to improve either the starting procedure or the SER strategy. Mainly this slow start is due to the fact the gradients of these few test problems, including the seemingly simpler Problem 1, have very large magnitudes at the initial points. Therefore, starting values subject to the local tolerance control by the standard ODE strategy (4.2) over a short time span tend to cluster together, which stagnates the flow and gives no advantage to multi-step methods.

	order 1	order 2	order 3	order 4	order 5
Problem 1	111	6	8	9	44
Problem 2	84379	NC	NC	NC	NC
Problem 3	211	1846	1846	1846	1847
Problem 4	1073	1335	1335	1335	1335
Problem 5	92	139	139	139	139

Table 8. Results by BDF-based pseudo-transient methods: starting values generated by ode23s over $[0, 0.01]$. No convergence is reached for Problem 2.

One possible remedy to circumvent the stiffness in the transient phase is a hybrid approach where a numerical ODE method with favorable damping property (see the discussion in Hairer & Lubich (2014)) is employed initially with low precision to navigate nearby the true trajectory and then the iterative method is turned on for faster and high-precision convergence. Some empirical evidence with the ode23s to support this idea is given in Table 9. At present the starting interval is chosen ad hoc so that the time steps taken by ode23s for the initialization will not overwhelm those seen in Tables 2 to 7 and yet give starting values close enough to the equilibrium points for fast convergence when switching to iterative schemes. The performance, judged by the total time steps needed for meeting (4.1), in Table 9 by such a combined strategy is superior than that in Table 7 by the pseudo-transient methods alone and those in Tables 2 to 6 by the ode23s alone.

	starting interval	ode23s steps	order 1	order 2	order 3	order 4	order 5
Problem 1	[0,2.e-02]	13	3	4	6	7	7
Problem 2	[0,1.e+06]	58	25	126	129	128	129
Problem 3	[0,1.e+06]	91	43	385	389	389	154
Problem 4	[0,1.e+03]	37	37	163	162	162	74
Problem 5	[0,5.e+01]	56	12	23	23	22	23

Table 9. Results by BDF-based pseudo-transient methods: starting values generated by ode23s over extended intervals.

5. Conclusion

Gradient dynamics occurs and appears in a wide range of disciplines. This work aims specifically at exploring several high-order but promising techniques which require few steps in tracking the gradient trajectory initially and eventually transform themselves into faster and high precision schemes at the refinement phase. These methods are nonstandard in that they are ODE-based but do not follow the usual numerical ODE observances such as precision and stability. Rather, they stand along as iterative methods, follow the gradient trajectory roughly, but locate the stationary points effectively. This paper discusses some mathematical foundations of these methods. In particular, we show the innate transition capability and the property of contractivity. Preliminary experiments, even by using a rather rudimentary SER step size selection strategy, seem to suggest the potential of these methods.

Many questions remain open. One of the main challenges is the proper phrase transition. Another is the proper strategies for step size control. The theoretical results and the numerical experiments in this paper employ exact Hessian information. In actual implementation, especially for large scale PDE problems, perhaps a hybrid Newton-Krylov type iteration should be used and is sufficient (Brown & Saad, 1990; Cai *et al.*, 1998). Also important is the modification of the general gradient dynamics techniques for structured problems where the preservation of structures is critical for gaining efficacy. One such

an application with significant importance is a proper projected gradient method for the Toda lattice (1.3) that would exploit the gradient property for fast transition and contractivity as well as preserve the spectrum for eigenvalue computation.

REFERENCES

ABSIL, P.-A., MAHONY, R. & ANDREWS, B. (2005) Convergence of the iterates of descent methods for analytic cost functions. *SIAM J. Optim.*, **16**, 531–547 (electronic).

ANDREI, N. (2004) Gradient flow method for nonlinear least squares minimization. *Manuscript*. Research Institute for Informatics. http://camo.ici.ro/neculai/nleast.pdf.

BAO, W. & DU, Q. (2004) Computing the ground state solution of bose–einstein condensates by a normalized gradient flow. *SIAM Journal on Scientific Computing*, **25**, 1674–1697.

BEHRMAN, W. (1998) An efficient gradient flow method for unconstrained optimization. *Ph.D. thesis*, Stanford University.

BHAYA, A. & KASZKUREWICZ, E. (2006) *Control perspectives on numerical algorithms and matrix problems*. Advances in Design and Control, vol. 10. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), pp. xxvi+272.

BLOCH, A. M., BROCKETT, R. W. & RATIU, T. S. (1992) Completely integrable gradient flows. *Comm. Math. Phys.*, **147**, 57–74.

BOBENKO, A. I. & SCHRÖDER, P. (2005) Discrete willmore flow. *Proceedings of the Third Eurographics Symposium on Geometry Processing*. SGP '05. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association.

BROWN, P. N. & SAAD, Y. (1990) Hybrid Krylov methods for nonlinear systems of equations. *SIAM J. Sci. Statist. Comput.*, **11**, 450–481.

CAI, X.-C., GROPP, W. D., KEYES, D. E., MELVIN, R. G. & YOUNG, D. P. (1998) Parallel Newton-Krylov-Schwarz algorithms for the transonic full potential equation. *SIAM J. Sci. Comput.*, **19**, 246–265 (electronic). Special issue on iterative methods (Copper Mountain, CO, 1996).

CALVO, M. P., ISERLES, A. & ZANNA, A. (1997) Numerical solution of isospectral flows. *Math. Comp.*, **66**, 1461–1486.

CARR, J. (1981) *Applications of centre manifold theory*. Applied Mathematical Sciences, vol. 35. Springer-Verlag, New York-Berlin, pp. vi+142.

CHEN, L.-Q. (2002) Phase-field models for microstructure evolution. *Annual review of materials research*, **32**, 113–140.

CHU, M. T. (2008) Linear algebra algorithms as dynamical systems. *Acta Numer.*, **17**, 1–86.

DEIFT, P., NANDA, T. & TOMEI, C. (1983) Ordinary differential equations and the symmetric eigenvalue problem. *SIAM J. Numer. Anal.*, **20**, 1–22.

DEKKER, K. & VERWER, J. G. (1984) *Stability of Runge-Kutta methods for stiff nonlinear differential equations*. CWI Monographs, vol. 2. North-Holland Publishing Co., Amsterdam, pp. ix+307.

DEMBO, R. S., EISENSTAT, S. C. & STEIHAUG, T. (1982) Inexact Newton methods. *SIAM J. Numer. Anal.*, **19**, 400–408.

DROSKE, M. & RUMPF, M. (2004) A level set formulation for willmore flow. *Interfaces and free boundaries*, **6**, 361–378.

EISENSTAT, S. C. & WALKER, H. F. (1996) Choosing the forcing terms in an inexact Newton method. *SIAM J. Sci. Comput.*, **17**, 16–32. Special issue on iterative methods in numerical linear algebra (Breckenridge, CO, 1994).

FLÅM, S. D. (2002) Equilibrium, evolutionary stability and gradient dynamics. *Int. Game Theory Rev.*, **4**, 357–370.

FRIEDMAN, D. & ABRAHAM, R. (2009) Bubbles and crashes: gradient dynamics in financial markets. *J. Econom. Dynam. Control*, **33**, 922–937.

HAIRER, E., NØRSETT, S. P. & WANNER, G. (1993) *Solving ordinary differential equations. I*. Springer Series in Computational Mathematics, vol. 8, second edn. Berlin: Springer-Verlag, pp. xvi+528. Nonstiff problems.

HAIRER, E., LUBICH, C. & WANNER, G. (2006) *Geometric numerical integration.* Springer Series in Computational Mathematics, vol. 31, second edn. Berlin: Springer-Verlag, pp. xviii+644. Structure-preserving algorithms for ordinary differential equations.

HAIRER, E. (2010) Energy-preserving variant of collocation methods. *J. Numer. Anal. Ind. Appl. Math.*, **5**, 73–84.

HAIRER, E. & LUBICH, C. (2014) Energy-diminishing integration of gradient systems. *IMA J. Numer. Anal.*, **34**, 452–461.

HAIRER, E. & WANNER, G. (1996) *Solving ordinary differential equations. II*. Springer Series in Computational Mathematics, vol. 14, second edn. Berlin: Springer-Verlag, pp. xvi+614. Stiff and differential-algebraic problems.

HALE, J. K. (1988) *Asymptotic behavior off dissipative systems*. Mathematical Surveys and Monographs, vol. 25. American Mathematical Society, Providence, RI, pp. x+198.

HIRSCH, M. W. & SMALE, S. (1974) *Differential equations, dynamical systems, and linear algebra*. Academic Press [A subsidiary of Harcourt Brace Jovanovich, Publishers], New York-London, pp. xi+358. Pure and Applied Mathematics, Vol. 60.

HIRSCH, M. W. & SMALE, S. (1979) On algorithms for solving $f(x) = 0$. *Comm. Pure Appl. Math.*, **32**, 281–313.

HORN, R. A. & JOHNSON, C. R. (1990) *Matrix analysis*. Cambridge: Cambridge University Press, pp. xiv+561. Corrected reprint of the 1985 original.

HUMPHRIES, A. R. & STUART, A. M. (1994) Runge-Kutta methods for dissipative and gradient dynamical systems. *SIAM J. Numer. Anal.*, **31**, 1452–1485.

ISERLES, A., MUNTHE-KAAS, H. Z., NØRSETT, S. P. & ZANNA, A. (2000) Lie-group methods. *Acta numerica, 2000*. Acta Numer., vol. 9. Cambridge: Cambridge Univ. Press, pp. 215–365.

ISERLES, A. (2002) On the discretization of double-bracket flows. *Found. Comput. Math.*, **2**, 305–329.

IWASAKI, M. & NAKAMURA, Y. (2002) On the convergence of a solution of the discrete Lotka-Volterra system. *Inverse Problems*, **18**, 1569–1578.

IWASAKI, M. & NAKAMURA, Y. (2004) An application of the discrete Lotka-Volterra system with variable step-size to singular value computation. *Inverse Problems*, **20**, 553–563.

KELLEY, C. T., LIAO, L.-Z., QI, L., CHU, M. T., REESE, J. P. & WINTON, C. (2007) Projected pseudo-transient continuation. *preprint*. North Carolina State University.

KELLEY, C. T. & KEYES, D. E. (1998) Convergence analysis of pseudo-transient continuation. *SIAM J. Numer. Anal.*, **35**, 508–523 (electronic).

LAMBERT, J. D. (1991) *Numerical methods for ordinary differential systems*. John Wiley & Sons, Ltd., Chichester, pp. x+293. The initial value problem.

MCLACHLAN, R. I., QUISPEL, G. R. W. & ROBIDOUX, N. (1999) Geometric integration using discrete gradients. *R. Soc. Lond. Philos. Trans. Ser. A Math. Phys. Eng. Sci.*, **357**, 1021–1045.

MEINTJES, K. & MORGAN, A. P. (1990) Chemical-equilibrium systems as numerical test problems. *ACM Trans. Math. Soft*, **16**, 143–151.

MICHAILOVICH, O., RATHI, Y. & TANNENBAUM, A. (2007) Image segmentation using active contours driven by the bhattacharyya gradient flow. *IEEE Transactions on Image Processing*, **16**, 2787–2801.

MORÉ, J. J., GARBOW, B. S. & HILLSTROM, K. E. (1981) Testing unconstrained optimization software. *ACM Trans. Math. Software*, **7**, 17–41.

MULDER, W. A. & VAN LEER, B. (1985) Experiments with implicit upwind methods for the Euler equations. *J. Comput. Phys.*, **59**, 232–246.

MÜLLER, K. & BROWN, L. D. (1979) Location of saddle points and minimum energy paths by a constrained simplex optimization procedure. *Theoret. Chim. Acta*, **53**, 75–93.

MUNTHE-KAAS, H. (1998) Runge-Kutta methods on Lie groups. *BIT*, **38**, 92–111.

NORTON, R. A., MCLAREN, D. I., QUISPEL, G. R. W., STERN, A. & ZANNA, A. (2015) Projection methods and

discrete gradient methods for preserving first integrals of ODEs. *Discrete Contin. Dyn. Syst.*, **35**, 2079–2098.

NORTON, R. A. & QUISPEL, G. R. W. (2014) Discrete gradient methods for preserving a first integral of an ordinary differential equation. *Discrete Contin. Dyn. Syst.*, **34**, 1147–1170.

PELETIER, M. A. (2012) Energies, gradient flows, and large deviations: a modelling point of view. *Lecture notes*. Technische Universiteit Eindhoven. http://www.win.tue.nl/ mpeletie/Onderwijs/Pisa2011/PeletierLectureNotesPisa2011.pdf.

PENZLER, P., RUMPF, M. & WIRTH, B. (2012) A phase-field model for compliance shape optimization in nonlinear elasticity. *ESAIM: Control, Optimisation and Calculus of Variations*, **18**, 229–258.

RATSCHEK, H. & ROKNE, J. (1993) Experiments using interval analysis for solving a circuit design problem. *J. Global Optim.*, **3**, 501–518.

SHACHAM, M. (1986) Numerical solution of constrained nonlinear algebraic equations. *Internat. J. Numer. Methods Engrg.*, **23**, 1455–1481.

SHAMPINE, L. F. & REICHELT, M. W. (1997) The MATLAB ODE suite. *SIAM J. Sci. Comput.*, **18**, 1–22.

SHI, K., XIAO, T. & SHENG, Z. (2009) Incentive mechanism design problem based on gradient dynamics. *Kybernetes*, **38**, 481–488.

SMALE, S. (1977) Convergent process of price adjustment and global Newton methods. *Frontiers of quantitative economics, Vol. IIIA (Invited papers, Econometric Soc., Third World Congress, Toronto, Ont., 1975)*. Amsterdam: North-Holland Publishing Co., pp. 191–205. Contributions to Economic Analysis, Vol. 105.

STEIHAUG, T. & WOLFBRANDT, A. (1979) An attempt to avoid exact Jacobian and nonlinear equations in the numerical solution of stiff differential equations. *Math. Comp.*, **33**, 521–534.

STRZODKA, R., DROSKE, M. & RUMPF, M. (2004) Image registration by a regularized gradient flow. a streaming implementation in dx9 graphics hardware. *Computing*, **73**, 373–389.

STUART, A. M. & HUMPHRIES, A. R. (1996) *Dynamical systems and numerical analysis*. Cambridge Monographs on Applied and Computational Mathematics, vol. 2. Cambridge: Cambridge University Press, pp. xxii+685.

SYMES, W. W. (1981/82) The *QR* algorithm and scattering for the finite nonperiodic Toda lattice. *Phys. D*, **4**, 275–280.

TSYPKIN, Y. Z. (1971) *Adaptation and learning in automatic systems*. New York: Academic Press, pp. xix+291. Translated from the Russian by Z. J. Nikolic, Mathematics in Science and Engineering, Vol. 73.

TSYPKIN, Y. Z. (1973) *Foundations of the theory of learning systems*. Academic Press [A subsidiary of Harcourt Brace Jovanovich, Publishers], New York-London, pp. xiii+205. Translated from the Russian by Z. J. Nikolic, Mathematics in Science and Engineering, Vol. 101.

VAN LEER, B. & MULDER, W. A. (1985) Relaxation methods for hyperbolic conservation laws. *Numerical methods for the Euler equations of fluid dynamics (Rocquencourt, 1983)*. SIAM, Philadelphia, PA, pp. 312–333.

VÁZQUEZ, J. L. (2007) *The porous medium equation: mathematical theory*. Oxford University Press.

WANG, L., CHU, M. T. & BO, Y. (2015) A computational framework of gradient flows for general linear matrix equations. *Numer. Algorithms*, **68**, 121–141.