BIT Numerical Mathematics (2021) 61:119–139
https://doi.org/10.1007/s10543-020-00819-y

Explicit stabilised gradient descent for faster strongly convex optimisation

Armin Eftekhari[1] · Bart Vandereycken[2] · Gilles Vilmart[2] · Konstantinos C. Zygalakis[3]

Received: 28 October 2019 / Accepted: 10 June 2020 / Published online: 4 July 2020

Abstract

This paper introduces the Runge–Kutta Chebyshev descent method (RKCD) for strongly convex optimisation problems. This new algorithm is based on explicit stabilised integrators for stiff differential equations, a powerful class of numerical schemes that avoid the severe step size restriction faced by standard explicit integrators. For optimising quadratic and strongly convex functions, this paper proves that RKCD nearly achieves the optimal convergence rate of the conjugate gradient algorithm, and the suboptimality of RKCD diminishes as the condition number of the quadratic function worsens. It is established that this optimal rate is obtained also for a partitioned variant of RKCD applied to perturbations of quadratic functions. In addition, numerical experiments on general strongly convex problems show that RKCD outperforms Nesterov's accelerated gradient descent.

Keywords Runge–Kutta methods · Strongly convex optimization · Accelerated gradient descent

Communicated by Lothar Reichel.

✉ Armin Eftekhari
armin.eftekhari@umu.se

Bart Vandereycken
bart.vandereycken@unige.ch

Gilles Vilmart
gilles.vilmart@unige.ch

Konstantinos C. Zygalakis
k.zygalakis@ed.ac.uk

1 Department of Mathematics and Mathematical Statistics, Umeå University, 901 87 Umeå, Sweden

2 Section of Mathematics, University of Geneva, CP 64, 1211 Genève 4, Switzerland

3 School of Mathematics, University of Edinburgh, Edinburgh EH9 3FD, UK

Mathematics Subject Classification 90C25 · 65L20

1 Introduction

Optimisation is at the heart of many applied mathematical and statistical problems, while its beauty lies in the simplicity of describing the problem in question. In this work, given a function $f : \mathbb{R}^d \to \mathbb{R}$, we are interested in finding a minimiser $x_* \in \mathbb{R}^d$ of the problem

$$\min_{x \in \mathbb{R}^d} f(x). \tag{1.1}$$

We make the common assumption throughout that $f \in \mathscr{F}_{\ell,L}$, namely, the set of ℓ-strongly convex differentiable functions that have L-Lipschitz continuous derivative [13]. Corresponding to f is its *gradient flow*, defined as

$$\frac{dx}{dt} = -\nabla f(x), \quad x(0) = x_0 \in \mathbb{R}^d, \tag{1.2}$$

where x_0 is its initialisation. It is easy to see that traversing the gradient flow always reduces the value of f. Indeed, for any positive h, it holds that

$$f(x(h)) - f(x_0) = -\int_0^h \|\nabla f(x(t))\|_2^2 \, dt \le 0. \tag{1.3}$$

By discretising the gradient flow in (1.2), we can design various optimisation algorithms for (1.1). For example, by substituting in (1.2) the approximation

$$\frac{dx}{dt}(t_n) \approx \frac{x(t_n + h) - x(t_n)}{h}, \tag{1.4}$$

we obtain the gradient descent (GD) method as the iteration

$$x_{n+1} = x_n - h\nabla f(x_n), \quad n = 0, 1, 2, \ldots \tag{1.5}$$

Here, x_n is the numerical approximation of $x(t_n)$ for all n and $h > 0$ is the step size [13]. For this discretisation to remain *stable*, that is, for x_n in (1.5) to remain close to the exact gradient flow $x(t_n)$ and, consequently, for the value of f to reduce in every iteration, the step size h must not be too large.

Indeed, a well-known shortcoming of GD is that we must take $h \le 2/L$ to ensure stability, otherwise f might increase from one iteration to the next [13]. One can consider a different discretization of (1.2), by for example substituting in (1.2) the approximation (1.4) at $x(t_n + h)$ instead of $x(t_n)$. We then arrive at the update

$$x_{n+1} = x_n - h\nabla f(x_{n+1}), \tag{1.6}$$

which is known as the *implicit Euler method* in numerical analysis [10] because, as the name suggests, it involves solving (1.6) for x_{n+1}. It is not difficult to see that, unlike GD, there is no size restriction on the step size h for the implicit Euler method to decay, a property related to its algebraic stability [9]. Moreover, it is easy to verify that x_{n+1} in (1.6) is also the unique minimiser of the problem

$$\min_{x\in\mathbb{R}^d} hf(x) + \frac{1}{2}\|x - x_n\|_2^2; \tag{1.7}$$

the map from x_n to x_{n+1} is known in the optimisation literature as the *proximal map* of the function hf [14]. Unfortunately, even if ∇f is known explicitly, solving (1.6) for x_{n+1} or equivalently computing the proximal map is often just as hard as solving (1.1), with a few notable exceptions [14]. This setback severely limits the applicability of the proximal algorithm in (1.6) for solving problem (1.1).

Contributions With this motivation, we propose the *Runge–Kutta Chebyshev descent* (RKCD) method for solving problem (1.1). RKCD offers the best of both worlds, namely the computational tractability of GD (explicit Euler method) and the stability of the proximal algorithm (implicit Euler method). Inspired by [16], RKCD uses *explicit stabilised methods* [1,5,18] to discretise the gradient flow (1.2).

For the numerical integration of stiff problems, explicit stabilised methods provide a computationally efficient alternative to the implicit Euler method for stiff differential equations, where standard integrators face a severe step size restriction, in particular for spatial discretisations of high-dimensional diffusion PDEs; see the review [2]. Every iteration of RKCD consists of s *internal stages*, where each stage performs a simple GD-like update. Unlike GD however, RKCD does *not* decay monotonically along its internal stages, which allows it to take longer steps and travel faster along the gradient flow. After s internal stages, RKCD ensures that its new iterate is stable, namely, the value of f indeed decreases after each iteration of RKCD.

Recently, there has been a revived interest about the design and the interpretation of optimization methods as discretizations of ODEs [16]. In particular, discrete gradient methods were used in [8] for the integration of (1.2) and shown to have similar properties to the gradient descent for (strongly) convex objective functions. In addition, the work in [24], considers numerical discretizations of a rescaled version of the gradient flow (1.2) and shows that acceleration can be achieved when extra smoothness assumptions are imposed to the objective function f. Furthermore, in [19,23] an alternative second-order differential equation to the gradient flow was introduced containing a momentum term. Similarly, to the spirit of this work, a number of different numerical discretizations including Runge–Kutta methods were used for the integration of this second-order equation and shown to behave in an accelerated manner [6,17,25,27]. Our method, on the other hand, can achieve similar acceleration by a direct integration of the gradient flow (1.2) and does not need to include such a momentum term explicitly.

The rest of this paper is organised as follows. Section 2 formally introduces RKCD, which is summarised in Algorithm 1 and accessible without reading the rest of this paper. In Sect. 3, we then quantify the performance of RKCD for solving strongly convex quadratic programs, while in Sect. 4 we introduce and study theoretically a composite variant of RKCD applied to perturbations of quadratic functions. Then in Sect. 5, we empirically compare RKCD to other first-order optimisation algorithms and conclude that RKCD improves over the state of the art in practice. This paper concludes with an overview of the remaining theoretical challenges.

2 Explicit stabilised gradient descent

Let us start with the simple scalar problem where $f(x) = \frac{1}{2}\lambda x^2$, that is,

$$\min_{x\in\mathbb{R}} \tfrac{1}{2}\lambda x^2, \qquad \lambda > 0, \tag{2.1}$$

and consider the corresponding gradient flow

$$\frac{dx}{dt} = -\lambda x, \qquad x(0) = x_0 \in \mathbb{R}, \tag{2.2}$$

also known as the *Dahlquist test equation* [10]. It is obvious from (2.2) that $\lim_{t\to\infty} x(t) = 0$ and any sensible optimisation algorithm should provide iterates $\{x_n\}_n$ with a similar property, that is,

$$\lim_{n\to\infty} x_n = 0. \tag{2.3}$$

For GD, which corresponds to the explicit Euler disctretisation of (2.2), it easily follows from (1.5) that

$$x_{n+1} = R_{gd}(z)x_n, \quad R_{gd}(z) = 1 + z, \quad z = -\lambda h, \tag{2.4}$$

where R_{gd} is the *stability polynomial* of GD. Hence, (2.3) holds if $z \in \mathscr{D}_{gd}$, where the *stability domain* $\mathscr{D}_{gd}$ of GD is defined as

$$\mathscr{D}_{gd} = \{z \in \mathbb{C}\,;\ |R_{gd}(z)| < 1\}. \tag{2.5}$$

That is, (2.3) holds if $h \in (0, 2/\lambda)$, which imposes a severe limit on the time step h when λ is large. Beyond this limit, namely, for larger step sizes, the iterates of GD might not necessarily reduce the value of f or, put differently, the explicit Euler method might no longer be a faithful discretisation of the gradient flow.

At the other extreme, for the proximal algorithm which corresponds to the implicit Euler discretisation of the gradient flow in (2.2), it follows from (1.6) that

$$x_{n+1} = R_{pa}(z)x_n, \quad R_{pa}(z) = \frac{1}{1-z}, \quad z = -\lambda h, \tag{2.6}$$

with the stability domain

$$\mathscr{D}_{pa} = \left\{z \in \mathscr{C}\,; |R_{pa}(z)| < 1\right\}.$$

Therefore, (2.3) holds for *any* positive step size h. This property is known as A-stability of a numerical method [10]. Unfortunately, the proximal algorithm (implicit Euler method) is often computationally intractable, particularly in higher dimensions.

In numerical analysis, explicit stabilised methods for discretising the gradient flow offer the best of both worlds, as they are not only explicit and thus computationally tractable, but they also share some favourable stability properties of the implicit method. Our main contribution in this work is adapting these methods for optimisation, as detailed next.

For discretising the gradient flow (2.2), the key idea behind explicit stabilised methods is to relax the requirement that every step of explicit Euler method should remain stable, namely, faithful to the gradient flow. This relaxation in turn allows the explicit stabilised method to take longer steps and traverse the gradient flow faster. To be specific, applying any given explicit *Runge–Kutta method* with s stages (i.e., s evaluations of ∇f) per step to (2.2) yields a recursion of the form

$$x_{n+1} = R_s(z)x_n, \quad R_s(z) = 1 + z + a_2 z^2 + \cdots + a_s z^s, \tag{2.7}$$

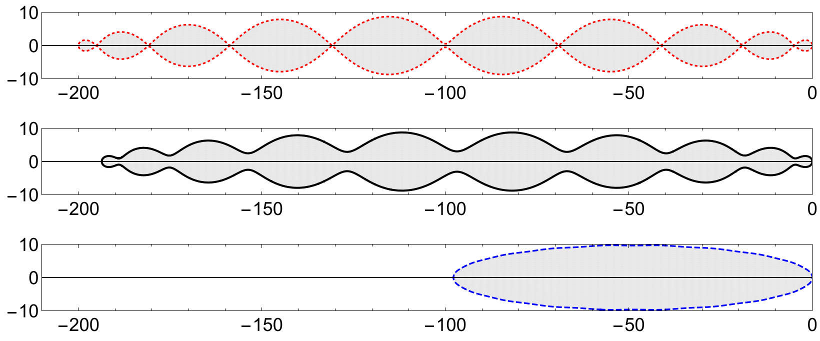
with the corresponding stability domain $\mathscr{D}_s = \{z \in \mathbb{C}\,;\, |R_s(z)| < 1\}$. We wish to choose $\{a_j\}_{j=2}^s$ to maximise the step size h while ensuring that $z = -h\lambda$ still remains in the stability domain $\mathscr{D}_s$, namely, for the update of the explicit stabilised method to remain stable. More formally, we wish to solve

$$\max_{a_2,\ldots,a_s} L_s \quad \text{subject to} \quad |R_s(z)| \le 1, \quad \forall z \in [-L_s, 0]. \tag{2.8}$$

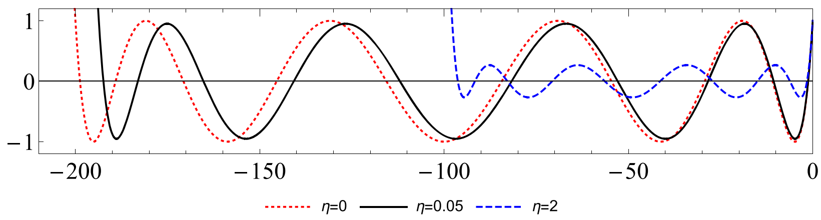
As shown in [10] (see also [2]), the solution to (2.8) is $L_s = 2s^2$ and, after substituting the optimal values for $\{a_2\}_{j=1}^s$ in (2.7), we find that the unique corresponding $R_s(z)$ is the shifted Chebyshev polynomial $R_s(z) = T_s(1+z/s^2)$ where $T_s(\cos\theta) = \cos(s\theta)$ is the Chebyshev polynomial of the first kind with degree s. In Fig. 1, $R_s(z)$ is depicted as $\eta = 0$ in red. It is clear from panel (b) that $R_s(z)$ equi-oscillates between -1 and 1 on $z \in [-L_s, 0]$, which is a typical property of minimax polynomials. As a consequence, after every s internal stages, the new iterate of the explicit stabilised method remains stable and faithful to (2.2), while travelling the most along the gradient flow.

Numerical stability is still an issue for the explicit stabilised method outlined above, particularly for the values of $z = -\lambda h$ for which $|R_s(z)| = 1$. As seen on the top of Fig. 1a, even the slightest numerical imperfection due to round-off will land us outside of the stability domain $\mathscr{D}_s$, which might make the algorithm unstable. In addition, for such values of z where $|R_s(z)| = 1$, the new iterate x_{n+1} is not necessarily any closer to the minimiser, here the origin. As a solution, it is common practice (see, e.g., [10]) to tighten the stability requirement to $|R_s(z)| \le \alpha_s(\eta) < 1$ for every $z \in [-L_{s,\eta}, -\delta_\eta]$. A popular choice is to introduce a positive dampening parameter η so that the stability function satisfies

$$R_s(z) = \frac{T_s(\omega_0 + \omega_1 z)}{T_s(\omega_0)}, \quad \omega_0 = 1 + \frac{\eta}{s^2}, \quad \omega_1 = \frac{T_s(\omega_0)}{T_s'(\omega_0)}. \tag{2.9}$$



(a) Complex stability domains $\mathscr{D}_s$: level set of $|R_s(z)| \leq 1$ for complex z.



(b) Graph of the stability function $R_s(z)$ as function of real z.

Fig. 1 Stability domains and stability functions of the Chebyshev method with $s = 10$ stages and different damping values $\eta = 0, 0.05$, and 2

Now, $R_s(z)$ oscillates between $-\alpha_s(\eta)$ and $\alpha_s(\eta)$ for every $z \in [-L_{s,\eta}, -\delta_\eta]$, where

$$\alpha_s(\eta) = \frac{1}{T_s(\omega_0)} < 1, \qquad L_{s,\eta} = \frac{1 + \omega_0}{\omega_1}. \tag{2.10}$$

In fact, $L_{s,\eta} \simeq (2 - \frac{4}{3}\eta)s^2$ is close to the optimal stability domain size $L_{s,0} = 2s^2$ for small η; see [2]. It also follows from (2.7) that

$$|x_{n+1}| \leq \alpha_s(\eta)|x_n|,$$

namely, the new iterate of the explicit stabilised method is indeed closer to the minimiser. In addition, as we can see in Fig. 1, introducing damping also ensures that a strip around the negative real axis is included in the complex stability domain $\mathscr{D}_s$, which grows in the imaginary direction as the damping parameter η increases. We also point out that, while a small damping $\eta = 0.05$ is usually sufficient for standard stiff problems, the benefit of large damping η was first exploited in [4] in the context of stiff stochastic differential equations and later improved in [3] using second kind Chebyshev polynomials.

It is relatively straightforward to generalise the ideas from above to integrate general multivariable gradient flows, thus not only the scalar test function (2.2). For the algorithmic implementation to be numerically stable, however, one should not evaluate the Chebyshev polynomials $T_s(z)$ naively but instead use their well-known three-term recurrence $T_{j+1}(x) = 2xT_j(x) - T_{j-1}(x)$, for $j = 1, 2, \dots$ (with $T_0(x) = 1$) in the implementation of the methods [20]. We dub this algorithm the Runge–Kutta Chebyshev descent method (RKCD), summarised in Algorithm 1.

```
Algorithm 1 Runge–Kutta Chebyshev descent method (RKCD) for solving (1.1)
Input: The gradient $\nabla f$ of a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$. Damping $\eta$ (e.g., $\eta = 1.17$). Lower and upper bounds $\ell$ and $L$ for eigenvalues of $\nabla^2 f$. Initialisation $x_0 \in \mathbb{R}^d$.
Body: Until convergence, repeat:
  – Compute $h$ and $s$ using (3.6) and
        $\omega_0 = 1 + \frac{\eta}{s^2}, \quad \omega_1 = \frac{T_s(\omega_0)}{T_s'(\omega_0)},$    (2.11)
    with $T_s$ is the Chebyshev polynomial of the first kind with degree $s$.
  – Set $x_n^0 = x_n$ and $x_n^1 = x_n^0 - h\mu_1 \nabla f(x_n)$ with $\mu_1 = \omega_1/\omega_0$
  – For $j \in \{2, \dots, s\}$, repeat:
        $x_n^j = -\mu_j h \nabla f(x_n^{j-1}) + \nu_j x_n^{j-1} - (\nu_j - 1)x_n^{j-2},$
        $\mu_j = \frac{2\omega_1 T_{j-1}(\omega_0)}{T_j(\omega_0)}, \quad \nu_j = \frac{2\omega_0 T_{j-1}(\omega_0)}{T_j(\omega_0)}.$
  – Set $x_{n+1} = x_n^s$.
Output: Estimate $\widehat{x} = x_{n+1}$ of a minimiser of (1.1).
```

3 Strongly convex quadratic

Consider the problem (1.1) with

$$f(x) = \tfrac{1}{2}x^T Ax - b^T x, \tag{3.1}$$

where $A \in \mathbb{R}^{d\times d}$ is a positive definite and symmetric matrix, and $b \in \mathbb{R}^d$. As the next proposition shows, the convergence of any Runge–Kutta method (such as GD, proximal algorithm) depends on the eigenvalues of A; the proof can be found in the appendix.

Proposition 1 *For solving problem* (1.1) *with f as in* (3.1)*, consider an optimisation algorithm with stability function R (for example, $R = R_{gd}$ for GD in* (2.4)*) and step size h. Let $\{x_n\}_{n\geq 0}$ be the iterates of this algorithm. Also assume that $0 < \ell = \lambda_1 \leq \cdots \leq \lambda_d = L$, where $\{\lambda_i\}_i$ are the eigenvalues of A. Then, for every iteration $n \geq 0$, it holds*

$$f(x_{n+1}) - f(x_*) \leq \max_{1\leq i\leq d} R^2(-h\lambda_i)\,(f(x_n) - f(x_*)) \tag{3.2}$$

with x_* the minimizer of f.

Let us next apply Proposition 1 to both GD and RKCD.

Gradient descent Recalling (2.4), it is not difficult to verify that

$$\max_{1\le i\le d} R_{gd}^2(-\lambda_i h) = \begin{cases} R_{gd}^2(-\ell h), & \text{if} \quad 0 < h \le \frac{2}{\ell+L}, \\ R_{gd}^2(-Lh), & \text{if} \quad h \ge \frac{2}{\ell+L}. \end{cases} \tag{3.3}$$

It then follows from (3.3) that we must take $h \in (0, 2/L)$ for GD to be stable, namely, for GD to reduce the value of f in every iteration. In addition, one obtains the best possible decay rate by choosing $h = 2/(\ell + L)$. More specifically, we have that

$$\min_h \max_{1\le i\le d} R_{gd}^2(-\lambda_i h) = \left(\frac{\kappa-1}{\kappa+1}\right)^2,$$

where $\kappa = L/\ell$ is the *condition number* of the matrix A. That is, the best convergence rate for GD is predicted by Proposition 1 as

$$f(x_{n+1}) - f(x^*) \le \left(\frac{\kappa-1}{\kappa+1}\right)^2 (f(x_n) - f(x^*)), \tag{3.4}$$

achieved with the step size $h = 2/(\ell + L)$. Remarkably, the same conclusion holds for any function in $\mathscr{F}_{\ell,L}$, as discussed in [12].

Explicit stabilised gradient descent In the case of Algorithm 1, there are three different parameters that need to be chosen, namely, the step size h, the number of internal stages s, and the damping factor η. For a fixed positive η, let us next select h and s so that the numerator of $R_s(z)$ in (2.9), namely $|T_s(\omega_0 + \omega_1 z)|$, is bounded by one. Equivalently, we take h, s such that

$$-1 \le \omega_0 - \omega_1 Lh \le \omega_0 - \omega_1 \ell h \le 1. \tag{3.5}$$

For an efficient algorithm, we choose the smallest step size h and the smallest number s of internal stages such that (3.5) holds. More specifically, (3.5) dictates that $\kappa \le (1+\omega_0)/(-1+\omega_0) = 1 + 2s^2/\eta$; see (2.11). This in turn determines the parameters s and h as

$$s = \left\lceil \sqrt{(\kappa-1)\eta/2} \right\rceil, \qquad h = \frac{\omega_0 - 1}{\omega_1 \ell}, \tag{3.6}$$

with $\kappa = L/\ell$. Under (3.5) and using the definitions in (2.9,2.10), we find that

$$|R_{rkcd}(-\lambda_i h)| \le \alpha_s(\eta)$$

for every $1 \leq i \leq d$. Then, an immediate consequence of Proposition 1 is

$$f(x_{n+1}) - f(x_*) \leq \alpha_s(\eta)^2 (f(x_n) - f(x_*)). \tag{3.7}$$

Given that every iteration of RKCD consists of s internal stages—hence, with the same cost as s GD steps—it is natural to define the *effective* convergence rate of RKCD as

$$c_{rkcd}(\kappa) = \alpha_s(\eta)^{2/s}. \tag{3.8}$$

The following result, proved in the appendix, evaluates the effective convergence rate of RKCD, as given by (3.8), in the limit of $\eta \to \infty$.

Proposition 2 *With the choice of step size h and number s of internal stages in* (3.6)*, the effective convergence rate $c_{rkcd}(\kappa)$ of RKCD for solving problem* (1.1) *with f given in* (3.1) *satisfies*

$$\begin{aligned} \lim_{\eta\to\infty} c_{rkcd}(\kappa) &= c_{opt}(\kappa) + O(\kappa^{-3/2}), \\ c_{opt}(\kappa) &= \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^2. \end{aligned} \tag{3.9}$$

Above, $c_{opt}(\kappa)$ is the optimal convergence rate of a first-order algorithm, which is achieved by the conjugate gradient (CG) algorithm for quadratic f; see [13]. Put differently, Proposition 2 states that RKCD nearly achieves the optimal convergence rate in the limit of $\eta \to \infty$.[1] Moreover, it is perhaps remarkable that the performance of RKCD relative to the conjugate gradient *improves* as the condition number of f worsens, namely, as κ increases. The non-asymptotic behaviour of RKCD is numerically investigated in Fig. 2, corroborating Proposition 2. As illustrated in Sect. 5, Algorithm 1 can also be used effectively for optimization problems with non-quadratic $f \in \mathscr{F}_{\ell,L}$.[2]

4 Perturbation of a quadratic objective function

Proposition 2 shows us that, for strongly convex quadratic problems, one can recover the optimal convergence rate of the conjugate gradient for large values of the damping parameter η. Here we discuss a modification of Algorithm 1 to a specific class of non-linear problems for which we can prove the same convergence rate as in the quadratic case. More precisely, we consider

$$f(x) = \tfrac{1}{2}x^T Ax + g(x), \tag{4.1}$$

where A is a positive definite matrix for which we know (bounds on) the smallest and largest eigenvalue, and g is a β-smooth convex function [13]. Inspired by [26], we

[1] In practice, as η becomes larger the number of stages s grows; see (3.6). Therefore, the largest possible η is dictated by the computational budget in terms of gradient evaluations.

[2] We remark that the parameters h and s should then also be chosen using (3.6), where $\kappa = L/\ell$.

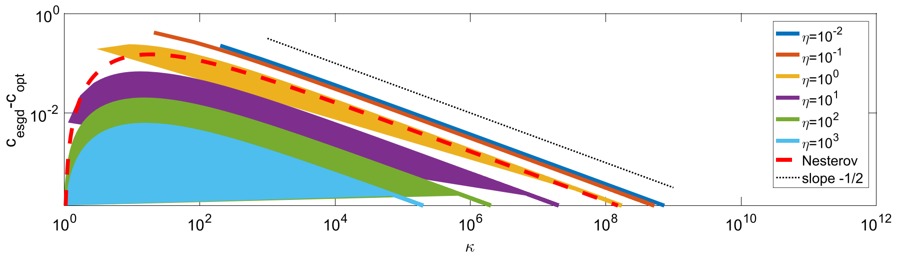


Fig. 2 This figure considers the non-asymptotic scenario from Prop. 2 and plots, as a function of the condition number κ, the difference $c_{rkcd}(\kappa) - c_{opt}(\kappa)$ of the (effective) convergence rate of RKCD compared to the optimal one, for many fixed values of the damping parameter η. In this plot, we observe that $c_{rkcd}(\kappa) - c_{opt}(\kappa)$ decays as $C_\eta/\sqrt{\kappa}$ for $\kappa \to \infty$ and for fixed η, see the slope $-1/2$ in the plot. Numerical evaluations suggest that the constant $C_\eta = O(1/\sqrt{\eta})$ becomes arbitrarily small as η grows, which corroborates Proposition 2. For comparison, we also include the result $c_{agd}(\kappa) - c_{opt}(\kappa)$ for the optimal rate of the Nestrov's accelerated gradient descent (AGD) given by $c_{agd} = (1-2/\sqrt{3\kappa+1})^2$ [12]. Comparing the two algorithms in Sect. 5 we find that RKCD outperforms AGD, namely $c_{rkcd} \le c_{agd}$ for $\eta \ge \eta_0$, where $\eta_0 \simeq 1.17$ is a moderate size constant

consider the modification[3] of Algorithm 1 specifically designed for the minimization of composite functions of the form (4.1). We call this method the partitioned Runge–Kutta Chebyshev descent method (PRKCD) and show in Proposition 3 (proved in the appendix) that it matches the rate given by the analysis of quadratic problems. PRKCD is derived as a variant of the RKCD method applied to the target function (4.1), where the nonquadratic term is replaced by the linearization $g(x) \simeq g(x_n) + (x - x_n)^T \nabla g(x_n)$, as described in Algorithm 2. In practice, PRKCD is implemented almost identically to the RKCD method, except that the gradient terms ∇g in $\nabla f(x) = Ax + \nabla g(x)$ are evaluated at x_n in all the intermediate steps $j = 1, \dots, s$ described in Algorithm 1. Hence, PRKCD is a numerically stable implementation of the update

$$\begin{aligned} x_{n+1} &= R_s(-Ah)x_n - B_s(-Ah)\nabla g(x_n), \\ B_s(z) &= \frac{1 - R_s(z)}{z}, \end{aligned} \tag{4.2}$$

where $R_s(z)$ is the stability function given by (2.9). It is worth noting that this method has only one evaluation of ∇g per step of the algorithm, which can be advantageous if the evaluations of ∇g are costly.

Proposition 3 *Let f be given by (4.1) where $A \in \mathbb{R}^{d\times d}$ is a positive definite matrix with largest and smallest eigenvalues L and $\ell > 0$, respectively, and condition number $\kappa = L/\ell$. Let $\gamma > 0$ and g be a β-smooth convex function for which $0 < \beta < C(\eta)\gamma\ell$ where $C(\eta) = \omega_1\alpha_s(\eta)/(\omega_0 - 1)$ and the parameters of PRKCD are chosen according to conditions (3.6). Then the iterates of PRKCD satisfy*

$$f(x_{n+1}) - f(x_*) \le (1+\gamma)^2\alpha_s^2(\eta)(f(x_n) - f(x_*)) \tag{4.4}$$

[3] In the case where $g(x) = 0$ this algorithm coincides with Algorithm 1 for $\nabla f(x) = Ax$.

Algorithm 2 Partitioned Runge–Kutta Chebyshev descent method (PRKCD) for minimizing (4.1)

Input: The gradient ∇g of a differentiable function $g : \mathbb{R}^d \to \mathbb{R}$, and the matrix $A \in \mathbb{R}^{d\times d}$. Damping η (e.g., $\eta = 1.17$). Lower and upper bounds ℓ and L for eigenvalues of A. Initialisation $x_0 \in \mathbb{R}^d$.

Apply Algorithm 1 to the function

$$f(x) = \tfrac{1}{2}x^T Ax + g(x_n) + (x - x_n)^T \nabla g(x_n), \tag{4.3}$$

with gradient $\nabla f(x) = Ax + \nabla g(x_n)$.

with x_* the minimizer of f.

Roughly speaking, Proposition 3 states that the effective convergence rate of PRKCD matches that of RKCD in (3.7) up to a factor of $(1 + \gamma)^2$, as long as

$$(1 + \gamma)\alpha_s(\eta) < 1 \tag{4.5}$$

to guarantee the convergence of Algorithm 2. Since $\lim_{s\to\infty}(1 + \gamma)^{2/s} = 1$, the effective rate in (4.4) is equivalent to $c_{rkcd}(\kappa)$ in the limit of a large condition number κ. Indeed, the numerical evidence in Fig. 2 suggests that PRKCD remains efficient for moderate values of the damping parameter η. In particular, for $\eta = \eta_0 = 1.17$, we have that $c_{rkcd} \simeq c_{agd} = (1 - 2/\sqrt{3\kappa + 1})^2$ and $C(\eta_0) \simeq 0.59$ when $\kappa \gg 1$. Using (3.6),(3.8) yields that $\alpha_s(\eta_0)$ is close to $\lim_{\kappa\to\infty} c_{agd}^{s/2} \simeq 0.413$, and the convergence condition (4.5) holds for $\gamma < 1/\alpha_s(\eta_0) - 1$ and $\beta/\ell < C(\eta_0)\gamma \simeq 0.83$ for $\kappa \gg 1$, where β is the smoothness parameter of g and ℓ the smallest eigenvalue of A.

Remark 4.1 For the case of a stiff objective function g_1 perturbed by a nonstiff and possibly costly nonquadratic function g_2, the PRKCD method can be generalized as follows to minimize the function

$$f(x) = g_1(x) + g_2(x). \tag{4.6}$$

Similar to Algorithm 2, one can simply apply Algorithm 1 to the modified objective function where $g_2(x)$ is replaced by $g_2(x_n) + (x - x_n)^T \nabla g_2(x_n)$ in (4.6). Note that the corresponding method coincides with Algorithm 2 in the case of a quadratic function $g_1(x) = \frac{1}{2}x^T Ax$ perturbed by $g_2(x) = g(x)$.

5 Numerical examples

In this section we illustrate the performance of of RKCD (Algorithm 1) and PRKCD (Algorithm 2) for solving problem (1.1) for a number of different test problems. In all the test problems, we will compare our method with optimally-tuned gradient descent (GD), given by

$$x_{k+1} = x_k - \frac{2}{\ell + L}\nabla f(x_k),$$

as well as the accelerated gradient descent (AGD), given in [13] as

$$x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k),$$
$$y_{k+1} = x_{k+1} + \frac{\sqrt{L}-\sqrt{\ell}}{\sqrt{L}+\sqrt{\ell}}(x_{k+1} - x_k).$$

Two of our examples are for quadratic stiff problems, either pure or perturbed by a non-stiff term, while the other test problems are for stiff non-quadratic problems. This is only to verify the theoretical analyses of Propositions 1–3. We do not advocate RKCD and PRKCD for such problems since exponential integrators with (rational) Krylov subspace techniques might be more appropriate then; see, e.g., [7,11]. However, the experiments below will show that our methods also perform well for minimizing non-quadratic strongly convex functions, which our theoretical results do not cover.

Strongly convex quadratic programming We consider the quadratic function $f(x) = \frac{1}{2}x^T Ax - x^T b$, with $A \in \mathbb{R}^{n\times n}$ a random matrix drawn from the Wishart distribution, namely, $A \sim \frac{1}{m}W_n(I_n, m)$ with $n = 4800$, $m = 5000$, and the entries of $b \in \mathbb{R}^n$ are independently drawn from the standard Gaussian distribution $\mathcal{N}(0, 1)$. For this matrix, with high probability, we have the following estimates from [22] for its largest and smallest eigenvalues

$$L = \left(1 + \sqrt{\frac{n}{m}}\right)^2, \quad \ell = \left(1 - \sqrt{\frac{n}{m}}\right)^2$$

giving $\kappa \approx 10^4$. Since this problem is quadratic we will also solve it using the conjugate gradient method (CG).

We plot $f(x_k) - f(x_*)$ for the RKCD, GD, AGD and CG in Fig. 3. As predicted by Proposition 2, as the damping parameter η for RKCD increases, the behaviour of RKCD approaches that of CG. Furthermore, even for the modest damping parameter of $\eta \approx 1.17$, RKCD is comparable to AGD.

Regularised logistic regression We now study our first example not covered by our theoretical results. Consider the objective function

$$f(x) = \sum_{i=1}^{m} \log(1 + \exp(-y_i \xi_i^T x)) + \frac{\tau}{2}\|x\|_2^2,$$

where $\Xi = \left[\xi_1 \cdots \xi_m\right]^T \in \mathbb{R}^{m\times d}$ is the design matrix with the columns $\{\xi_i\}_i$, and $y \in \{-1, 1\}^d$. Note that $f \in \mathscr{F}_{\ell,L}$ with $\ell = \tau$ and $L = \tau + \|\Xi\|_2^2/4$. As in [15], we used the Madelon UCI dataset with $d = 500$, $m = 2000$, and $\tau = 10^2$. This results in a very poorly conditioned problem with $\kappa = L/l \approx 10^9$.

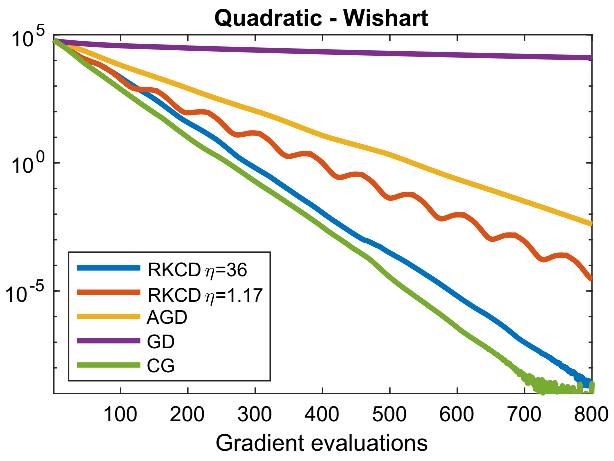


Fig. 3 Error in function value $f(x_k) - f(x_*)$ for the strongly convex quadratic programming problem

We plot $f(x_k) - f(x_*)$ for the RKCD, GD, and AGD in Fig. 4. We again observe that, as the damping parameter increases, the convergence rate of RKCD improves, thus requiring half of the number of evaluations needed for the RKCD to achieve the same level of accuracy of 10^{-5}, compared to AGD. Furthermore, for this problem, RKCD is comparable to AGD even for the modest damping parameter of $\eta \approx 1.17$.

Regression with (smoothed) elastic net regularisation We now study a regression problem with (smoothed) elastic net regularisation. In particular, in the case the objective function is of the form $f(x) = \frac{1}{2}\|Ax - b\|_2^2 + \lambda L_\tau(\|x\|_1) + \frac{\ell}{2}\|x\|_2^2$, where $L_\tau(t)$ is the standard Huber loss function with parameter τ to smooth $|t|$; see [28]. We used $A \in \mathbb{R}^{m\times d}$ a random matrix drawn from the Gaussian distribution scaled by $1/\sqrt{d}$, $d = 3000$, $m = 900$, $\lambda = 0.2$, $\tau = 10^{-3}$, $l = 10^{-2}$. The objective function is in $\mathscr{F}_{\ell,L}$ with $L \approx (1 + \sqrt{m/d})^2 + \lambda/\tau + \ell$. The condition number is $\kappa \approx 10^4$.

We plot $f(x_k) - f(x_*)$ for the RKCD, GD, and AGD in Fig. 5. The behaviour of all the different methods is very similar to the one for the regularised logistic regression.

Nonlinear elliptic PDE We consider the one-dimensional integro-differential PDE

$$\frac{\partial}{\partial x}\left(\exp\big(-\alpha u(x)\big)\frac{\partial u(x)}{\partial x}\right) = \int_0^1 \frac{u^4(s)}{(1+|x-s|)^2}dx, \quad u(0) = 1,\ u(1) = 0 \tag{5.1}$$

on the interval $0 \le x \le 1$, where α is a fixed parameter. We first consider the semilinear case $\alpha = 0$, where the left-hand side of (5.1) reduces to the Laplacian $\frac{\partial^2 u(x)}{\partial x^2}$, which yields a model describing the stationary variant of a temperature profile of air near the ground [21]. Using a finite difference approximation $u(i\Delta x) \simeq U_i$ for $i = 1, \ldots d$

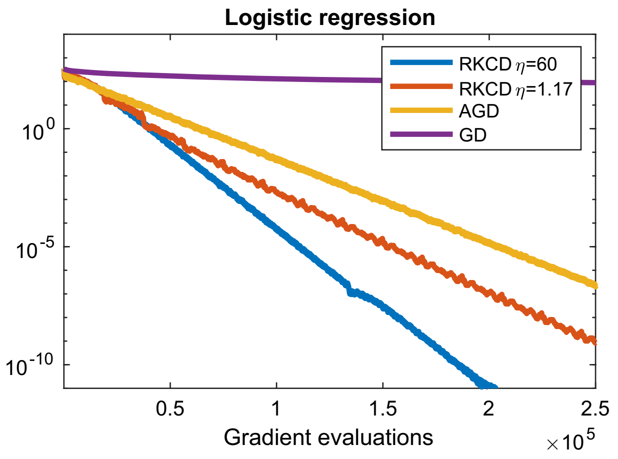


Fig. 4 Error in function value $f(x_k) - f(x_*)$ for the regularised logistic regresion problem

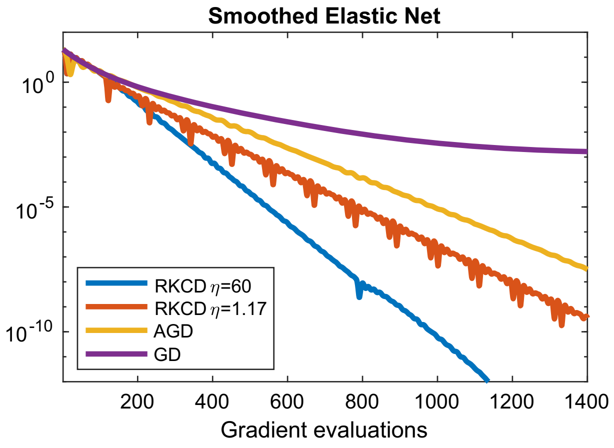
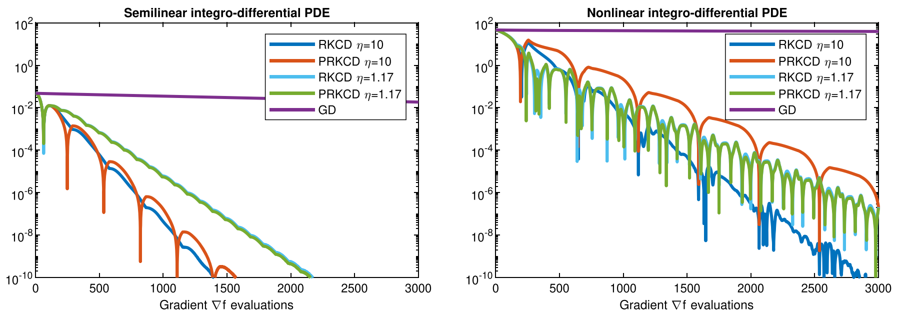


Fig. 5 Error in function value $f(x_k) - f(x_*)$ for the regression with (smoothed) elastic net regularisation

on a spatial mesh with size $\Delta x = 1/(d+1)$, and using the trapezoidal quadrature for the integral, we obtain a problem of the form (4.1) where A is the usual tridiagonal discrete Laplace matrix of size $d \times d$ with condition number $\kappa \simeq 4\pi^{-2}d^2$ (using $\lambda_{\max} \sim 4\Delta x^{-2}$ and $\lambda_{\min} \to \pi^2$ as $\Delta x \to 0$), and the entries of the gradient vector $\nabla g(U) \in \mathbb{R}^d$ are given by

$$\frac{\partial g(U)}{\partial U_i} = \frac{\Delta x}{2(1+i\Delta x)^2} + \sum_{j=1}^{d} \frac{\Delta x U_j^4}{(1+\Delta x|i-j|)^2}.$$

We observe that, when the dimension d is large, calculating $\nabla g(x)$ can become very expensive as one needs to calculate a sum over all j's, which makes PRKCD a particularly attractive option here. We plot $f(x_k) - f(x_*)$ for the RKCD, PRKCD and GD



(a) Semilinear PDE ($\alpha = 0$) **(b)** Semilinear PDE ($\alpha = 1$)

Fig. 6 Error in function value $f(x_k) - f(x_*)$ for the nonlinear PDE problem (5.1)

in Fig. 6 for $d = 200$. We use the initialization corresponding to the simple function $u(x) = 1 - x$ which satisfies the boundary conditions in (5.1).

In the semilinear case ($\alpha = 0$), we see in Fig. 6 (left) that the PRKCD performs similarly to RKCD for the sets of parameters $\eta = 10, s = 288$ and $\eta = 1.17, s = 99$, where s denotes the number of evaluations per step of $\nabla f(x) = Ax + \nabla g(x)$ for RKCD, and respectively the number of matrix–vector products Ax_n^{j-1} for PRKCD (recall that PRKCD needs a single evaluation of the gradient ∇g per step), which corroborates Proposition 3. For the case of a nonlinear diffusion (with $\alpha = 1$ in (5.1)), we consider the natural generalization of the PRKCD method described in Remark 4.1. For the discrete nonlinear diffusion term evaluated at point $x_i = i\Delta t$, we use the natural finite difference formula

$$\frac{\partial}{\partial x}\left(\exp\big(-\alpha u(x_i)\big)\frac{\partial u(x_i)}{\partial x}\right) \simeq e^{-\alpha U_i}\frac{U_{i+1} - U_i}{\Delta x^2} - e^{-\alpha U_{i-1}}\frac{U_i - U_{i-1}}{\Delta x^2},$$

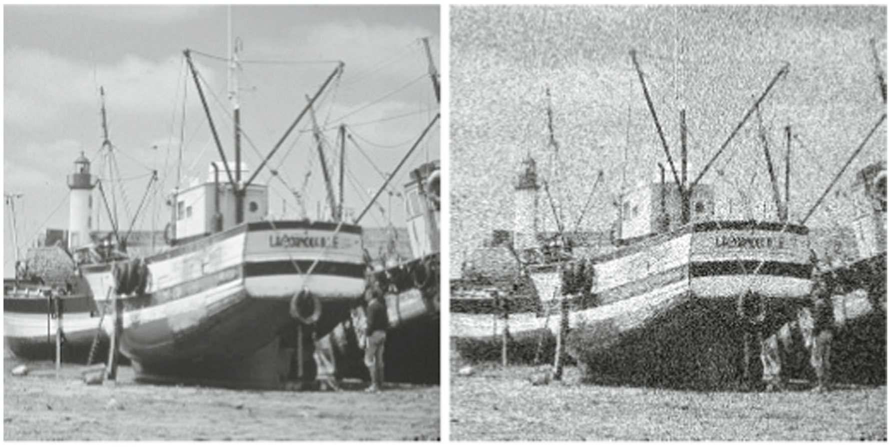
and consider in the algorithms the bounds $\ell = \pi^2 e^{-\alpha}$ and $L = 4\Delta x^{-2}$ for the spectrum of $\nabla^2 f(x)$. In Fig. 6 (right) we observe again excellent performances of the RKCD and PRKCD methods, although such a nonquadratic problem associated to the nonlinear PDE (5.1) is not covered by Proposition 3.

Smoothed total variation denoising Here we consider the problem of image denoising using a smoothed total variation regularisation. In particular the objective function is of the form

$$f(x) = \frac{1}{2}||x - y||^2 + \lambda J_\epsilon(x)$$

where y is the noisy image[4] and $J_\epsilon(x)$ is is a smoothed total variation of the image.

[4] The original image and the noisy version can be seen in Fig. 7.



(a) Original image **(b)** Noisy image

Fig. 7 Images used for the smoothed total variation denoising

In particular,

$$J_\epsilon(x) = \sum_i ||(Gx)_i||_\epsilon$$

where $(Gx)_i \in \mathbb{R}^2$ is an approximation of the gradient of x at pixel i and for $u \in \mathbb{R}^2$ and

$$||u||_\epsilon = \sqrt{\epsilon^2 + ||u||^2}$$

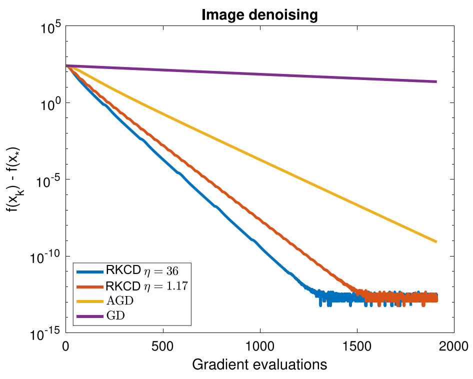
a smoothing of the L^2 norm in $\mathbb{R}^2$.

The objective function is in $\mathscr{F}_{\ell,L}$ with $\ell = 2$ and $L = 2\left(1 + \frac{4\lambda}{\epsilon}\right)$. We choose $\lambda = 6 \cdot 10^{-2}$, $\epsilon = 10^{-4}$ and hence the conditioning number is $\kappa \approx 2.4 \times 10^4$. We plot $f(x_k) - f(x_*)$ for the RKCD, GD and AGD in Fig. 8a as well as the denoised image in Fig. 8b.

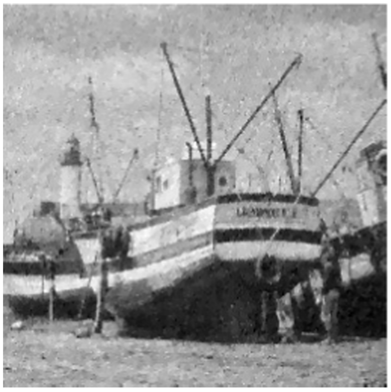
6 Conclusion

In this paper, using ideas from numerical analysis of ODEs, we introduced a new class of optimisation algorithms for strongly convex functions based on explicit stabilised methods. With some care, these new methods RKCD and PRKCD are as easy to implement as SD but require in addition a lower bound ℓ on the smallest eigenvalue. They were shown to match the optimal convergence rates of first order methods for certain subclasses of $\mathscr{F}_{\ell,L}$.

Our numerical experiments illustrate that this might be the case for all functions in $\mathscr{F}_{\ell,L}$, and proving this is the subject of our current research efforts. In addition, there is a number of different interesting research avenues for this class of methods, including



(a) Error in function values.



(b) Denoised image

Fig. 8 Total variation image denoising example

adjusting them to convex optimisation problems with $\ell = 0$, as well as for adaptively choosing the time-step h using local information to optimise their performance further. Furthermore, their adaptation to stochastic optimisation problems, where one replaces the full gradient of the function f by a noisy but cheaper version of it, is another interesting but challenging direction we aim to investigate further.

Acknowledgements Open access funding provided by Umea University. We thank the referees for their constructive remarks when reviewing this paper. This work was partially supported by The Alan Turing Institute by a small Grant award SG020. The research of K.C.Z was partially supported by the Alan Turing Institute under the EPSRC Grant EP/N510129/1. The research of B.V. and G.V. was partially supported by Grant 200020_178752 of the SNSF (Swiss National Science Foundation). The research of G.V. was partially supported by Grant 200020_184614 of the SNSF.

A Proof of the main results

In this section we will discuss the proofs of the main results in the paper

Proof of Proposition 1 We start our proof by noticing that the gradient flow (1.2) in the case of the quadratic function (3.1) becomes

$$\frac{dx}{dt} = -Ax + b.$$

In addition since A is positive definite and symmetric, there exists an orthogonal matrix V such that

$$A = VDV^{-1}, \quad D = \mathrm{diag}(\lambda_1, \ldots, \lambda_d), \quad \lambda_1 \le \cdots \le \lambda_d.$$

If we now make the change of variables $y = V^{-1}x - D^{-1}V^{-1}b$, we obtain the following equation

$$\frac{dy}{dt} = -Dy. \tag{A.1}$$

Hence, in this coordinate system each coordinate is independent of the other, while the objective function can be written as

$$f(y) - f(y^{(*)}) = \frac{1}{2}\sum_{i=1}^{d} \lambda_i y_i^2, \quad y^{(*)} = 0.$$

We can now write Eq. (A.1) in the vector form as

$$\frac{dy_i}{dt} = -\lambda_i y_i,$$

and hence each coordinate satisfies independently the simple quadratic (3.1) and the application of Runge–Kutta method with stability function $R(z)$ gives $y^{(n+1)} = R(-h\lambda_i)y^{(n)}$ (where $y^{(n)}$ is the nth iterate, that is, $y^{(n)} = V^{-1}x_n$). Hence

$$\begin{aligned} f(y^{(n+1)}) - f(y^{(*)}) &= \sum_{i=1}^{d} \lambda_i \left[R(-h\lambda_i)y_i^{(n)}\right]^2 \\ &\le \max_{1\le i\le d} R^2(-h\lambda_i)\,(f(y^{(n)}) - f(y^{(*)})), \end{aligned}$$

which completes the proof. □

Proof of Proposition 2 Using (2.10) and properties of Chebyshev polynomials we have

$$\alpha_s(\eta) = \left[\cosh\left(s \,\mathrm{arcosh}\left(1 + \frac{\eta}{s^2}\right)\right)\right]^{-1}.$$

Using (3.6) and the estimate $\cosh(sx)^{-2/s} \to e^{-2x}$ for $s \to \infty$, we deduce

$$\begin{aligned} \lim_{\eta\to\infty} c_{rkcd}(\kappa) &= e^{-2\,\mathrm{arcosh}\left(1+\frac{2}{\kappa}\right)} \\ &= \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^2 + O(\kappa^{-3/2}). \end{aligned}$$

□

Proof of Proposition 3 Our starting point is Eq. (4.2). In particular, we will show that if we choose our parameters suitably, then this scheme converges with the rate predicted by the theorem, and we will then show how Algorithm 2 corresponds to an implementation of this scheme.

We start the proof by noting that since f in (4.1) is strongly convex there exists a unique minimizer x_* satisfying

$$Ax_* + \nabla g(x_*) = 0. \tag{A.2}$$

Thus

$$\begin{aligned} x_{n+1} - x_* &= R_s(-Ah)x_n - hB_s(-Ah)\nabla g(x_n) - x_* \\ &= R_s(-Ah)(x_n - x_*) \\ &\quad - hB_s(-Ah)(\nabla g(x_n) - \nabla g(x_*)) \\ &\quad + (R_s(-Ah) - I + hB_s(-Ah)A)x_* \end{aligned}$$

where in the above identity we have used (A.2) multiplied on the left by $B_s(-Ah)$. Now, by definition of the matrix B we have $R_s(-Ah) - I + hB_s(-Ah)A = 0$, and we obtain

$$\begin{aligned} x_{n+1} - x_* &= R_s(-Ah)(x_n - x_*) \\ &\quad - hB_s(-Ah)(\nabla g(x_n) - \nabla g(x_*)) \end{aligned}$$

and hence

$$||x_{n+1} - x_*|| \leq (||R_s(-Ah)|| + h||B_s(-Ah)||\beta)\, ||x_n - x_*||.$$

We now know from the analysis in the main text that if h, s are chosen according to (3.6), then

$$||R(-Ah)|| \leq \alpha_s(\eta)$$

and using the fact that $||B_s(-Ah)|| \leq 1$, which is a consequence of Lemma A.1 below, we see that

$$||x_{n+1} - x_*|| \leq \left[\alpha_s(\eta) + \beta\left(\frac{\omega_0 - 1}{\omega_1 \ell}\right)\right] ||x_n - x_*||$$

and hence

$$||x_{n+1} - x_*|| \leq (1 + \gamma)\alpha_s(\eta)||x_n - x_*||$$

for

$$0 < \beta < \beta_* = \gamma \mu C(\eta)$$

where we define

$$C(\eta) = \frac{\omega_1 \alpha_s(\eta)}{\omega_0 - 1}.$$

We have thus proved Proposition 3 for a numerical scheme of the form (4.2). It remains to prove that Algorithm 2 is a scheme equivalent to (4.2). Indeed, the following identity can be proved by induction on $j = 2, \dots, s$, using $T_j(x) = 2xT_{j-1}(x) - T_{j-2}(x)$,

$$x_n^j = R_{s,j}(-hA)x_n - B_{s,j}(-hA)\nabla g(x_n)$$

where $R_{s,j}(x) = T_j(\omega_0 + \omega_1 x)/T_j(\omega_0)$, $B_{s,j}(x) = (I - R_{s,j}(x))/x$, and we obtain the result (4.2) by taking $j = s$. □

Lemma A.1 *Consider the polynomial* $B(\xi)$ *defined in* (4.2). *Then,* $\max_{\xi \in [-L_{s,\eta}, 0]} |B(\xi)| = 1$.

Proof By the Lagrange theorem, for ξ in the interval $(-L_{s,\eta}, 0)$, there exists z in the same interval such that $-B(\xi) = (R_s(z) - 1)/z = R_s'(z)$, and using the definition of $R_s(z)$ in (2.9), we have for $x = \omega_0 + \omega_1 z$,

$$R_s'(z) = \frac{T_s'(x)}{T_s'(\omega_0)}.$$

Recalling $x \in [-1, \omega_0]$, for the first case where $x \in [1, \omega_0]$, we use the fact that $T_s(x)$ and all its derivatives are increasing functions of $x \geq 1$ (this is because by the Rolle theorem all the roots of the Chebyshev polynomials $T_s(x)$ and their derivatives are in the interval $[-1, 1]$). Hence $|T_s'(x)| \leq |T_s'(\omega_0)|$, which yields $|R_s'(z)| \leq 1$. For the second case where $x \in [-1, 1]$, we have $x = \cos(\theta)$ for some θ, and we obtain

$$s^{-2}|T_s'(x)| = s^{-2}|T_s'(\cos(\theta))| = \frac{|\sin(s\theta)|}{s|\sin\theta|} \leq 1,$$

where the latter bound can be obtain by an elementary study of the function $\sin(s\theta)/(s \sin\theta)$, while $s^{-2}T_s'(\omega_0) \geq s^{-2}T_s'(1) = 1$ is an increasing function of η, hence $|R_s'(z)| \leq 1$, and this concludes the proof of Lemma A.1. □

References

1. Abdulle, A.: Fourth order Chebyshev methods with recurrence relation. SIAM J. Sci. Comput. **23**(6), 2041–2054 (2002)
2. Abdulle, A.: Explicit stabilized Runge–Kutta methods. In: Engquist, B. (ed.) Encyclopedia of Applied and Computational Mathematics, pp. 460–468. Springer, Berlin (2015)
3. Abdulle, A., Almuslimani, I., Vilmart, G.: Optimal explicit stabilized integrator of weak order 1 for stiff and ergodic stochastic differential equations. SIAM/ASA J. Uncertain. Quantif. **6**(2), 937–964 (2018)
4. Abdulle, A., Cirilli, S.: S-ROCK: Chebyshev methods for stiff stochastic differential equations. SIAM J. Sci. Comput. **30**(2), 997–1014 (2008)

5. Abdulle, A., Medovikov, A.: Second order Chebyshev methods based on orthogonal polynomials. Numer. Math. **90**(1), 1–18 (2001)
6. Betancourt, M., Jordan, M.I., Wilson, A.C.: On symplectic optimization (2018). arXiv:1802.03653
7. Druskin, V., Lieberman, C., Zaslavsky, M.: On adaptive choice of shifts in rational Krylov subspace reduction of evolutionary problems. SIAM J. Sci. Comput. **32**(5), 2485–2496 (2010)
8. Ehrhardt, M.J., Riis, E.S., Ringholm, T., Schönlieb, C.-B.: A geometric integration approach to smooth optimisation: foundations of the discrete gradient method (2018). arXiv:1805.06444
9. Hairer, E., Lubich, C.: Energy-diminishing integration of gradient systems. IMA J. Numer. Anal. **34**(2), 452–461 (2014)
10. Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems. Springer, Berlin (1996)
11. Hochbruck, M., Ostermann, A.: Exponential integrators. Acta Numer. **19**, 209–286 (2010)
12. Lessard, L., Recht, B., Packard, A.: Analysis and design of optimization algorithms via integral quadratic constraints. SIAM J. Optim. **26**(1), 57–95 (2016)
13. Nesterov, Y.: Introductory Lectures on Convex Optimization: A Basic Course, 1st edn. Springer, Berlin (2014)
14. Parikh, N., Boyd, S.: Proximal algorithms. Found. Trends Optim. **1**(3), 127–239 (2014)
15. Scieur, D., d'Aspremont, A., Bach, F.: Regularized nonlinear acceleration. In: NIPS'16 Proceedings of the 30th International Conference on Neural Information Processing Systems, pp. 712–720 (2016)
16. Scieur, D., Roulet, V., Bach, F.R., d'Aspremont, A.: Integration methods and optimization algorithms. Adv. Neural Inf. Process. Syst. **30**, 1109–1118 (2017)
17. Shi, B., Du, S.S., Su, W., Jordan, M.I.: Acceleration via symplectic discretization of high-resolution differential equations. In: Advances in Neural Information Processing Systems, pp. 5745–5753 (2019)
18. Sommeijer, B.P., Shampine, L.F., Verwer, J.G.: RKC: an explicit solver for parabolic PDEs. J. Comput. Appl. Math. **88**, 316–326 (1998)
19. Su, W., Boyd, S., Candès, E.J.: A differential equation for modeling Nesterov's accelerated gradient method: theory and insights. J. Mach. Learn. Res. **17**(153), 1–43 (2016)
20. van der Houwen, P.J., Sommeijer, B.P.: On the internal stability of explicit, m-stage Runge–Kutta methods for large m-values. Z. Angew. Math. Mech. **60**(10), 479–485 (1980)
21. Vasudeva, A.S., Verwer, J.G.: Solving parabolic integro-differential equations by an explicit integration method. J. Comput. Appl. Math. **39**(1), 121–132 (1992)
22. Vershynin, R.: Introduction to the non-asymptotic analysis of random matrices (2010). arXiv preprint arXiv:1011.3027
23. Wibisono, A., Wilson, A.C., Jordan, M.I.: A variational perspective on accelerated methods in optimization. Proc. Natl. Acad. Sci. **113**(47), E7351–E7358 (2016)
24. Wilson, A.C., Mackey, L., Wibisono, A.: Accelerating rescaled gradient descent: fast optimization of smooth functions. Adv. Neural Inf. Process. Syst. **32**, 13555–13565 (2019)
25. Wilson, A.C., Recht, B., Jordan, M.I.: A Lyapunov analysis of momentum methods in optimization (2016). arXiv:1611.02635
26. Zbinden, C.J.: Partitioned Runge–Kutta–Chebyshev methods for diffusion–advection–reaction problems. SIAM J. Sci. Comput. **33**(4), 1707–1725 (2011)
27. Zhang, J., Mokhtari, A., Sra, S., Jadbabaie, A.: Direct Runge–Kutta discretization achieves acceleration. In: Advances in Neural Information Processing Systems, pp. 3900–3909 (2018)
28. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. J. R. Stat. Soc. Ser. B (Stat. Methodol.) **67**(2), 301–320 (2005)