Systems & Control Letters 203 (2025) 106146

Contents lists available at ScienceDirect

Systems & Control Letters

journal homepage: www.elsevier.com/locate/sysconle

Analysis of gradient descent algorithms: Discrete to continuous domains and circuit equivalents

He Hao [a], Daniel Silvestre [a,c,d,*], Carlos Silvestre [a,b]

[a] Institute for Systems and Robotics, Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal
[b] Department of Electrical and Computer Department, Faculty of Science and Technology, University of Macau, Macao Special Administrative Region of China
[c] School of Science and Technology, NOVA University of Lisbon (FCT/UNL), 2829-516 Caparica, Caparica, Portugal
[d] UNINOVA – CTS and LASI, Campus de Caparica, 2829-516, Caparica, Portugal

ARTICLE INFO

Keywords:
Optimization-based controllers
Iterative solvers

ABSTRACT

In recent years, there have been several advances in iterative optimization algorithms seen as closed-loop control systems in discrete-time that solve unconstrained optimization problems. In this paper, we extend these advances to the continuous setting and leverage circuit equivalence to present a possible implementation of such controllers. Next, we address constrained Quadratic Programming (QP) challenges within a primal–dual framework that appears in many controller definitions. By drawing parallels between second-order ODEs and circuit dynamics, our study bridges theoretical optimization with practical electrical analogues that can be designed and implemented for problems in systems engineering, robotics, and autonomous vehicles that could benefit from the low power and latency of these optimization-based controllers.

1. Introduction

The Control community has devoted a significant research effort into the field of autonomous systems from high-altitude drones to underwater vehicles, for instance in [1,2] for drone technologies, [3] for communications, along with [4,5] for underwater exploration or [6,7] for search in unknown environments. These technologies have been applied to environmental surveillance, disaster management, and unmanned logistics, as highlighted in [8–12].

This paper focuses on the idea of leveraging the application of gradient descent methods either for trajectory planning or controller design. As a consequence, this paper first investigates continuous-time gradient descent methods by introducing ODE models and establishing expressions for parameter selection. Then, we introduce electrical circuits analogues to the differential equations that govern these gradient descent methods in continuous-time domain.

In the literature, there has been substantial attention to gradient descent methods in discrete time, such as the Heavy-Ball [13], Nesterov's accelerated gradient [14], and the Triple-Momentum method [15]. The paper in [15] provides a comprehensive analysis of poles and zeros in transition matrices that allows an optimal selection of parameters to minimize worst-case convergence rate, while the authors in [16] employ eigenvalue analysis to examine convergence rates and stability. In the literature, there has also been some interest in continuous-time methods such as [17], where continuous-time models of Nesterov's method are derived that result in time-varying parameters. The work in [18] presented high-resolution ODEs with higher-order correction terms for gradient methods, which differs from our methods with constant parameters that enables presenting theoretical results for parameter selection.

An interesting feature of continuous-time methods is the possibility to be represented by an equivalent circuit. The advantage of using circuits over digital processors is the faster response times and more energy efficient computations. This shift from digital computation to circuit analogs is analyzed in studies like [19], which highlights the low power requirements and intrinsic parallelism of analog computers. Recent work shows that domain-specific analog accelerators can outperform digital systems, offering improved optimization speed and reduced energy consumption in [20,21], while hybrid analog/digital frameworks further enhance computational efficiency and adaptability [22,23]. Emphasizing the transition from digital calculation to circuit analogs, research such as [24–27], and [28] demonstrates the growing trend in this area. Further, [29,30] discuss equivalent circuits in unconstrained optimization, presenting methodologies for designing algorithms robust to hyperparameters and achieving fast convergence rates. Our research diverges from these approaches by proposing equivalent circuits directly derived from op-amp-based circuits. The main contributions of this paper can be summarized as follows:

* Corresponding author.
E-mail addresses: he.hao@tecnico.ulisboa.pt (H. Hao), dsilvestre@isr.tecnico.ulisboa.pt (D. Silvestre), csilvestre@umac.mo (C. Silvestre).

https://doi.org/10.1016/j.sysconle.2025.106146
Received 16 September 2024; Received in revised form 4 April 2025; Accepted 12 May 2025
Available online 4 June 2025

- Continuous-time versions of various gradient methods focusing on their parameter optimization to obtain optimal convergence rates;
- Development of equivalent circuits such that the optimization can be carried out using low power and fast speeds;

These contributions lead to two main advantages:

- We can provide optimal parameters for continuous models much like the ones already presented for discrete-time in the literature;
- QP-based CBF-CLF controllers can be implemented using circuits and thus have the controllers working directly in continuous time.

We would like to remark that the last point is of particular interest, considering that the QPs are based on continuous conditions for the CLFs and CBFs, but due to normally being implemented in a digital embedded system, they must be run with a sufficiently small sampling time. The safety guarantee through forward set invariance is given by performing a Lyapunov analysis on the continuous closed-loop system. Therefore, it makes sense that the controller can be implemented directly in continuous time.

2. Gradient methods in continuous time

Having in mind the design of continuous controllers based on CLF-CBF QPs, let us start by addressing the problem of minimizing a quadratic function without considering constraints. Let us consider $x \in \mathbb{R}^n$ as our optimization variable and a matrix Q and vector e that translate how each entry is weighed, leading to the following problem formulation:

Problem 1. Consider a set of agent states collectively represented by the state vector x. The objective is to optimize a quadratic cost function of the states by minimizing the objective function:

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad g(x) = \frac{1}{2}x^\intercal Q x + \mathrm{e}^\intercal x \tag{1}$$

where $Q \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix to ensure the existence and uniqueness of the solution to the optimization problem, and $\mathrm{e} \in \mathbb{R}^n$ accounts for a linear term in the cost.

2.1. Review of discrete-time gradient descent methods

Before tackling Problem 1 in a continuous time setting, let us review discrete time methods that will help build the intuition to the continuous counterparts. The gradient descent method is given by the iteration:

$$x_{k+1} = x_k - \beta\nabla g(x_k). \tag{2}$$

The gradient descent method might suffer from slow convergence, which can be improved by alternatives like the Heavy-Ball method, which is inspired by the concept of physical ball moving along g with momentum, taking the form:

$$x_{k+1} = (1+\alpha)x_k - \alpha x_{k-1} - \beta\nabla g(x_k). \tag{3}$$

Further building on the idea of momentum, Nesterov's Accelerated Gradient introduces a predictive step, which leads to:

$$\begin{aligned} x_{k+1} &= y_k - \beta\nabla g(y_k) \\ y_k &= (1+\alpha)x_k - \alpha x_{k-1} \end{aligned} \tag{4}$$

while a generalization name Triple-Momentum method was presented in [15]:

$$\begin{aligned} x_{k+1} &= (1+\alpha)x_k - \alpha x_{k-1} - \beta\nabla g(y_k) \\ y_k &= (1+\gamma)x_k - \gamma x_{k-1} \\ o_k &= (1+\sigma)x_k - \sigma x_{k-1} \end{aligned} \tag{5}$$

We recover the main result of the literature regarding their optimal parameters to ensure optimal convergence rates for L-smooth and m-strong convex g functions. Worst-case convergence rate λ corresponds to $\|x(k) - x^\star\|_2 \le \lambda^k \|x(0) - x^\star\|_2$, regardless of the initial conditions $x(0)$.

Theorem 2 ([15,16]). *In addressing Problem 1, gradient descent, heavy-ball, Nesterov's accelerated gradient, and triple momentum, achieve optimal convergence rates when their parameters are chosen as follows:*

Method	*Parameters* $(\alpha, \beta, \gamma, \sigma)$
Gradient descent	$(0, \frac{2}{L+m}, 0, 0)$
Heavy-ball	$(\frac{(\sqrt{L}-\sqrt{m})^2}{(\sqrt{L}+\sqrt{m})^2}, \frac{4}{(\sqrt{L}+\sqrt{m})^2}, 0, 0)$
Nesterov's method	$(\frac{\sqrt{L}-\sqrt{m}}{\sqrt{L}+\sqrt{m}}, \frac{1}{L}, 0, 0)$
Triple momentum	$(\frac{1+\rho}{L}, \frac{\rho^2}{2-\rho}, \frac{\rho^2}{(1+\rho)(2-\rho)}, \frac{\rho^2}{1-\rho^2})$

where $\rho = 1 - 1/\sqrt{\kappa}$ *and* $\kappa = L/m$. *In this context,* $L > 0$ *and* $m > 0$ *represent constants indicating that the function* f *is L-smooth and m-strongly convex, respectively.*

2.2. Continuous-time counterparts of gradient methods

In this section, we start by noting that Nesterov's method can be rewritten to exhibit an implicit temporal increment

$$x_{k+1} - x_k = \alpha(x_k - x_{k-1}) - \beta\nabla g(y_k) \tag{6}$$

where the discrete differences $x_{k+1} - x_k$ and $x_k - x_{k-1}$ represent finite approximations in differential calculus.

Let us introduce a scaling factor, $\sqrt{s}$, to adapt the discrete parameters α and β into their continuous-time equivalents $\alpha_c = \frac{1-\alpha}{\sqrt{s}}$ and $\beta_c = \frac{\beta}{s}$. Consequently, (6) is reformulated as follows:

$$x_{k+1} = x_k + (1 - \alpha_c\sqrt{s})(x_k - x_{k-1}) - \beta_c s\nabla g(y_k) \tag{7}$$

where $y_k = x_k + (1 - \alpha_c\sqrt{s})(x_k - x_{k-1})$. The discrete-time update rule is then transformed into

$$\frac{x_{k+1} - x_k}{\sqrt{s}} = (1 - \alpha_c\sqrt{s})\frac{x_k - x_{k-1}}{\sqrt{s}} - \beta_c\sqrt{s}\nabla g(y_k), \tag{8}$$

assuming the discrete sampling time k, scaled to the continuous-time variable t via the factor $\sqrt{s}$ as $k = \frac{t}{\sqrt{s}}$. Approximating finite differences by first and second-order derivatives, we obtain

$$\frac{x_{k+1} - x_k}{\sqrt{s}} = \dot{x}(t) + \frac{1}{2}\ddot{x}(t)\sqrt{s} + o(\sqrt{s}) \tag{9a}$$

$$\frac{x_k - x_{k-1}}{\sqrt{s}} = \dot{x}(t) - \frac{1}{2}\ddot{x}(t)\sqrt{s} + o(\sqrt{s}) \tag{9b}$$

$$\sqrt{s}\nabla g(y_k) = \sqrt{s}\nabla g(x(t)) + o(\sqrt{s}) \tag{9c}$$

which means that (8) is approximately

$$\begin{aligned} \dot{x}(t) + \frac{1}{2}\ddot{x}(t)\sqrt{s} + o(\sqrt{s}) = (1 - \alpha_c\sqrt{s})\left(\dot{x}(t) - \frac{1}{2}\ddot{x}(t)\sqrt{s} + o(\sqrt{s})\right) \\ - \beta_c\sqrt{s}\nabla g(x(t)) + o(\sqrt{s}). \end{aligned} \tag{10}$$

By grouping terms according to the order of their derivatives, we can further simplify into

$$\begin{aligned} &\alpha_c\sqrt{s}\dot{x}(t) + (\sqrt{s} + \frac{\alpha_c}{2}s)\ddot{x}(t) \\ &+ \beta_c\sqrt{s}\nabla g(x(t)) + (1 + \alpha_c\sqrt{s} + \beta_c\sqrt{s})o(\sqrt{s}) = 0 \end{aligned} \tag{11}$$

and neglect higher-order infinitesimal terms $o(\sqrt{s})$ to obtain

$$\alpha_c\sqrt{s}\dot{x}(t) + (\sqrt{s} + \frac{\alpha_c}{2}s)\ddot{x}(t) + \beta_c\sqrt{s}\nabla g(x(t)) = 0 \tag{12}$$

Further simplification is achieved by dividing the entire equation by $\sqrt{s}$:

$$\alpha_c \dot{x}(t) + \ddot{x}(t) + \frac{\alpha_c}{2}\sqrt{s}\ddot{x}(t) + \beta_c \nabla g(x(t)) = 0 \tag{13}$$

As $\sqrt{s}$ approaches zero, we can disregard the term $\frac{\alpha_c}{2}\sqrt{s}\ddot{x}(t)$ and get the final expression:

$$\ddot{x}(t) = -\alpha_c \dot{x}(t) - \beta_c \nabla g(x(t)) \tag{14}$$

Building upon the previous differential equation and its connection with discrete-time gradient methods, we can propose a general linear system to represent the whole family of optimization methods as follows:

$$\begin{aligned} \ddot{x} &= -\alpha\dot{x} - \beta\nabla g(y) \\ y &= x + \gamma\dot{x} \\ o &= x + \sigma\dot{x} \end{aligned} \tag{15}$$

where parameters α, β, γ, and σ are to be chosen as to get the fastest convergence rate in worst-case. We have named the methods as: Nesterov method (2-parameter), the Triple-Momentum method without output equation (3-parameter), and the continuous Triple-Momentum method (4-parameter). This nomenclature is deliberately chosen to highlight their structural similarities from discrete methods, facilitating a coherent understanding of their application.

To shorten the notation, we write the methods

- Continuous Heavy-Ball: $\ddot{x} = -\alpha\dot{x} - \beta\nabla g(x)$.
- Continuous Nesterov: $\begin{aligned} \ddot{x} &= -\alpha\dot{x} - \beta\nabla g(y) \\ y &= x + \alpha\dot{x} \end{aligned}$
- Continuous Triple-Momentum: $\begin{aligned} \ddot{x} &= -\alpha\dot{x} - \beta\nabla g(y) \\ y &= x + \gamma\dot{x} \\ o &= x + \sigma\dot{x} \end{aligned}$

as a linear system in state–space using matrices

- Continuous Heavy-Ball: $\left[\begin{array}{cc|c} -\alpha I_n & 0_n & -\beta I_n \\ I_n & 0_n & 0_n \\ \hline 0_n & I_n & 0_n \end{array}\right]$
- Continuous Nesterov: $\left[\begin{array}{cc|c} -\alpha I_n & 0_n & -\beta I_n \\ I_n & 0_n & 0_n \\ \hline \alpha I_n & I_n & 0_n \end{array}\right]$
- Continuous Triple-Momentum: $\left[\begin{array}{cc|c} -\alpha I_n & 0_n & -\beta I_n \\ I_n & 0_n & 0_n \\ \hline \gamma I_n & I_n & 0_n \end{array}\right]$

where $u = \nabla g(y)$ and the Continuous Triple-Momentum method has a tuned output $o = x + \sigma\dot{x}$.

2.2.1. Optimized parameter selection for continuous gradient methods

In this section, we present a result regarding the optimal selection of parameters to obtain the fastest worst-case convergence rate.

Theorem 3. *Continuous gradient methods achieve optimal convergence rates when their parameters are chosen as follows:*

Method	*Parameters* $(\alpha, \beta, \gamma, \sigma)$
Heavy-ball	$(\sqrt{4\beta m}, \infty, 0, 0)$
Nesterov	$(\frac{2\sqrt{\beta L}}{1+\beta L}, \infty, 0, 0)$
Triple momentum	$(2\sqrt{\beta L m}\frac{\sqrt{L}-\sqrt{m}}{L-m}, \infty, \frac{2}{\sqrt{\beta}}\frac{\sqrt{L}-\sqrt{m}}{L-m}, \frac{8}{(\alpha+\beta\gamma m)^2})$

Proof. Let us start by computing the gradient of g

$$\nabla g(x) = Qx + e \tag{16}$$

In equilibrium conditions, we have $\dot{x}^\star = 0$ and $u^\star = 0$, which makes

$$\dot{x}^\star = 0 \iff Ax^\star + Bu^\star = 0 \implies Ax^\star = 0 \tag{17}$$

and additionally

$$u^\star = 0 \iff \nabla g(y(t)) = 0 \iff QCx^\star + e = 0. \tag{18}$$

The error $\tilde{x} = x - x^\star$ has dynamics

$$\begin{aligned} \dot{\tilde{x}} &= \dot{x} - \dot{x}^\star \\ &= Ax + Bu \\ &= Ax + B(QCx + e) \\ &= Ax - Ax^\star + BQCx - BQCx^\star \\ &= (A + BQC)(x - x^\star) = (A + BQC)\tilde{x} \end{aligned} \tag{19}$$

where we used (17) to allow adding $-Ax^\star$ and (18) to replace the value for e.

From (19), convergence to zero error corresponds to checking the eigenvalues of $T := A + BQC$. Utilizing the eigenvalue decomposition $Q = U\Lambda U^\intercal$, where U is orthogonal and Λ is diagonal, allows to diagnolize the matrix and study

$$T_i = A_1 + B_1\lambda_i C_1, \quad i = 1, \dots, n \tag{20}$$

where each T_i corresponds to each mode of the overall error dynamics corresponding to each eigenvalue λ_i of Q A_1, B_1 and C_1 are the matrices for the scalar case.

Continuous Heavy-Ball: For this method, the matrix T_i results in

$$T_i = A_1 + B_1\lambda_i C_1 = \begin{bmatrix} -\alpha & -\beta\lambda_i \\ 1 & 0 \end{bmatrix} \tag{21}$$

with the characteristic polynomial being

$$\nu^2 + \alpha\nu + \beta\lambda_i = 0. \tag{22}$$

The roots of this polynomial are:

$$\nu = \frac{-\alpha \pm \sqrt{\Delta}}{2} \tag{23}$$

where $\Delta = \alpha^2 - 4\beta\lambda_i$. To avoid oscillations, we need $\Delta = 0$, leading to $\alpha = \sqrt{4\beta m}$. However, notice that we can make the real part arbitrarily negative by making $\beta \to \infty$.

Continuous Nesterov:

$$T_i = A_1 + B_1\lambda_1 C_1 = \begin{bmatrix} -\alpha - \beta\alpha\lambda_i & -\beta\lambda_i \\ 1 & 0 \end{bmatrix} \tag{24}$$

The characteristic polynomial of the matrix T_i is thus expressed as:

$$\nu^2 + (\alpha + \beta\alpha\lambda_i)\nu + \beta\lambda_i = 0 \tag{25}$$

with roots:

$$\nu = \frac{-\alpha - \beta\alpha\lambda_i \pm \sqrt{\Delta}}{2} \tag{26}$$

where $\Delta = (\alpha + \beta\alpha\lambda_i)^2 - 4\beta\lambda_i$. Therefore, to avoid oscillations it must satisfy

$$(\alpha + \beta\alpha\lambda_i)^2 - 4\beta\lambda_i = 0 \iff \alpha = \frac{2\sqrt{\beta L}}{1 + \beta L} \tag{27}$$

with the real part of the eigenvalues being arbitrarily negative is we make $\beta \to \infty$.

Continuous Triple-Momentum Doing the same steps, we arrive at T_i being

$$T_i = A_1 + B_1\lambda_1 C_1 = \begin{bmatrix} -\alpha - \beta\gamma\lambda_i & -\beta\lambda_i \\ 1 & 0 \end{bmatrix} \tag{28}$$

with a characteristic polynomial

$$\nu^2 + (\alpha + \beta\gamma\lambda_i)\nu + \beta\lambda_i = 0 \tag{29}$$

having roots

$$\nu = \frac{-\alpha - \beta\gamma\lambda_i \pm \sqrt{\Delta}}{2} \tag{30}$$

where $\Delta = (\alpha + \beta\gamma\lambda_i)^2 - 4\beta\lambda_i$.

In order to get real solutions, $\Delta = 0$, which results in

$$\alpha = 2\sqrt{\beta L m} \cdot \frac{\sqrt{L} - \sqrt{m}}{L - m}, \quad \gamma = \frac{2}{\sqrt{\beta}} \cdot \frac{\sqrt{L} - \sqrt{m}}{L - m} \tag{31}$$

which also indicates optimizing for the fastest convergence involves selecting $\beta \to \infty$.

In order to find the best value for σ, let us write the output error dynamics

$$\begin{aligned} \frac{d}{dt}(o - o^\star) &= \frac{d}{dt}(o - x^\star) \\ &= (\sigma(A + BQC) + I)(x - x^\star) \\ &= (\sigma T + I)(x - x^\star) \\ &= \tilde{T}(x - x^\star) \end{aligned} \tag{32}$$

Therefore, we want to place the pole of the inverse of the transfer function to the output with the slowest pole ρ given by

$$\rho = \max\left(-\frac{\alpha + \beta\gamma\lambda_i \pm \sqrt{\Delta}}{2}\right), \text{for } i = 1, \dots, n, \tag{33}$$

which after replacing with the relationship between α and γ, simplifies to

$$\rho = -\frac{\alpha + \beta\gamma m}{2}. \tag{34}$$

Applying the Laplace transform to the output function, $o = x + \sigma\dot{x}$, this yields:

$$O(s) = X(s) + \sigma s X(s) \tag{35}$$

which leads us to the transfer function:

$$\frac{O(s)}{X(s)} = 1 + \sigma s \tag{36}$$

and subsequently, its inverse:

$$\frac{X(s)}{O(s)} = \frac{1}{1 + \sigma s} \tag{37}$$

with a critical pole at $s = -\frac{1}{\sigma}$. Setting the pole equal to $\int \rho = -\frac{\rho^2}{2}$ gets us

$$-\frac{1}{\sigma} = -\frac{\rho^2}{2} = -\frac{1}{2}\left(-\frac{\alpha + \beta\gamma m}{2}\right)^2 \tag{38}$$

resulting in :

$$\sigma = \frac{8}{(\alpha + \beta\gamma m)^2}. \tag{39}$$

With these parameters established, the next section aims to provide a bridge to a direct continuous implementation by providing an equivalent electrical circuit that could be used to solve the unconstrained optimization problems with low power consumption.

3. Equivalent circuits for continuous gradient descent methods

In this section, we design circuits that translate the differential equations found for the continuous-time gradient descent methods. Although, increasing β always speeds up the convergence, in practice, we must choose a finite β to balance accuracy and convergence speed. Leveraging the correspondence between second-order ordinary differential equations (ODEs) and op-amp-based circuits, we introduce circuit implementations that embody these continuous approaches.

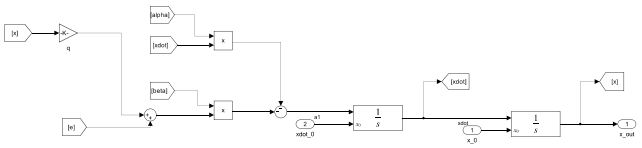
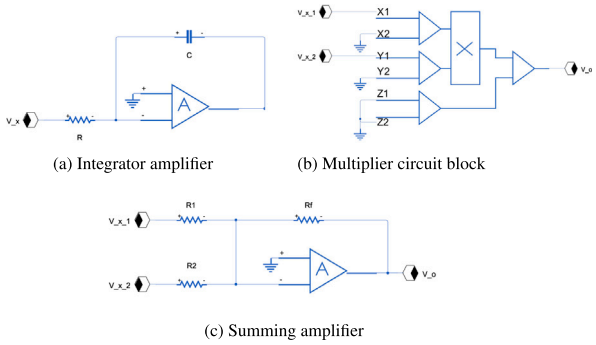


Fig. 1. Block diagram of the continuous Heavy-Ball method applied to the scalar case.



(a) Integrator amplifier (b) Multiplier circuit block

(c) Summing amplifier

Fig. 2. Illustration of circuit components to implement basic blocks.

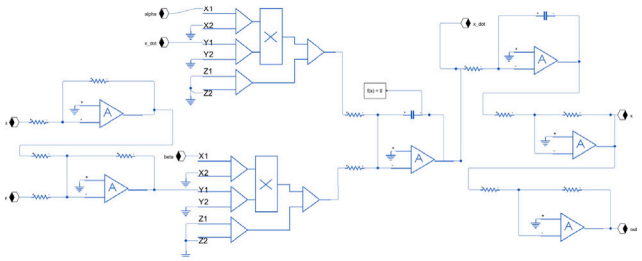


Fig. 3. Circuit implementation of the continuous Heavy-Ball for the scalar case.

3.1. Continuous heavy-ball

Let us start by considering the continuous Heavy-Ball method applied to a scalar function

$$\begin{aligned} \ddot{x} &= -\alpha\dot{x} - \beta\nabla g(x) \\ &= -\alpha\dot{x} - \beta q x - \beta e. \end{aligned} \tag{40}$$

To construct the circuit using operational amplifiers, we break down the equation into components suitable for electronic representation, focusing on integration, multiplication, and summation. Op-amp-based integrators model the integration of $\dot{x}$ and $\ddot{x}$, while analog multipliers handle the products of $\alpha\dot{x}$ and $\beta(qx + e)$. Summing amplifiers then aggregate these signals to reconstruct the equation $-\alpha\dot{x} - \beta(qx + e)$, which is diagrammatically summarized in Fig. 1 using logic blocks.

Integrators use op-amps with capacitors in their feedback loops to achieve integration, while multiplication is implemented using multiplier circuit blocks based on Gilbert cells or alternative CMOS designs. A comprehensive description of each circuit function can be found in [31]. The practical implementations of these circuits are shown in Fig. 2.

Combining the building blocks from Fig. 2 to construct the full Heavy-Ball method results in the circuit depicted in Fig. 3. The multidimensional case involves designing a circuit for each scalar case that uses variables from adjacent scalar sub-circuits and combining them with multiplications with constants and additions to implement each ith block

$$\ddot{x}_i = -\alpha\dot{x}_i - \beta\sum_{j=1}^{n} q_{ij}x_j - \beta e_i \quad \text{for} \quad i = 1, 2, \dots, n \tag{41}$$

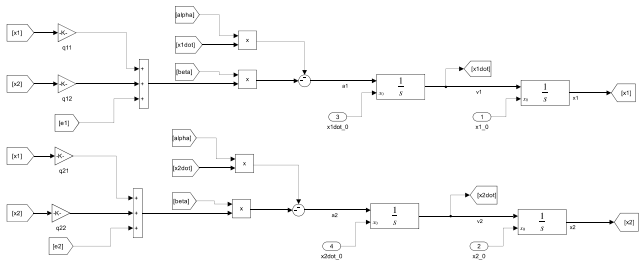


Fig. 4. Block diagram for the continuous Heavy-Ball method for a function g with 2 variables.

where x_i and e_i represent the ith elements of vectors x and e respectively, and q_{ij} is the (i, j)th element of the matrix Q. This setup requires n sub-circuits, each mirroring the basic configuration shown in Fig. 1, with additional interconnections to accommodate the matrix structure of the system. The logic block for the case of 2 dimensions is shown in Fig. 4.

3.2. Continuous methods with additional momentum

Adding additional momentum terms requires a more complex circuit representation. Unlike the continuous Heavy-Ball approach, these methods model the gradient as a first-order proportional relationship, specifically $\nabla g(y) = \nabla g(x + \gamma\dot{x})$, and include a tuned output $o = x + \sigma\dot{x}$.

We will focus on the general case of the continuous methods, which encompasses the foundational principles of related models like the continuous Nesterov method. In contrast to strictly replicating the ODEs detailed in (15), which would require constructing additional subloops to capture the first-order gradient dynamics, we adopt a more practical approach. We directly modify the value of $\dot{x}$, which allows us to integrate additional momentum effects without extensive changes to the existing circuit configuration:

$$\begin{aligned}\ddot{x} &= -\alpha\dot{x} - \beta\nabla g(y)\\ &= -\alpha\dot{x} - \beta q x - \beta q\gamma\dot{x} - \beta e\\ &= -(\alpha + \beta q\gamma)\dot{x} - \beta q x - \beta e\end{aligned} \tag{42}$$

This formula represents how the additional component $\beta q\gamma\dot{x}$ is merged into the existing parameters of the circuit. We accomplish this by adjusting the voltage at the integrator's input using the existing multiplier and summing op-amp. The tuned output $o = x + \sigma\dot{x}$ is not directly involved in the feedback loop of the circuit and can be added independently.

3.3. Practical implementation and parameter tuning

As we translate these theoretical models into practical circuit implementations, some component selection is required to maintain accuracy:

- **Resistors**: Set to 10^3 Ω to minimize electrical noise, thus enhancing the circuit signal accuracy.
- **Capacitors**: Adjusted to 10^{-3} F to ensure a product of resistance and capacitance equal to 1, stabilizing voltage levels.
- **Feedback Loop Resistors**: Calibrated to $q_{i,j} \times 10^3$ to accurately simulate the matrix operations and dynamics essential to the system.

Voltage scaling. Adjusting the parameter β significantly affects the circuit voltage levels, requiring careful scaling to prevent component saturation and ensure the circuit functions properly.

- **Op-Amp Voltage Limits**: According to [32], operational amplifiers should operate with input voltages below 100 V to prevent saturation.
- **Implementation Strategy**: Initial settings, including multiplier gains, are scaled down by $1/\beta$ to maintain voltages within safe limits.
- **Post-Adjustment Corrections**: After obtaining the output results, the voltages are adjusted by multiplying by β to achieve accurate values that correspond closely with theoretical expectations.

Optimizing circuit accuracy. Accurate low-voltage operation is critical, especially when dealing with sensitive components. Since voltage levels can be scaled down to $1/\beta$ in previous adjustments to ensure proper circuit functionality, it is important to carefully select the value of β.

- **Op-Amp Sensitivity**: Standard op-amps typically process signals as low as 1 μV, with performance degradation below 100 nV unless specifically designed to handle such low voltages.
- **Beta Selection**: Capping β at 10^4 is necessary because higher values would further reduce the voltage, potentially below the operational amplifier capabilities, thus affecting the precision of the system. Originally capable of precision around 10^{-7}, the scaling by $1/\beta$ proportionally reduces this to 10^{-3}, considering the voltage reduction.
- **System Settling Time** (T_s): A carefully selected β ensures faster convergence while maintaining the accuracy of the system within a 10^{-3} tolerance.

4. Constrained optimization problem using equivalent circuits

Quadratic Programming (QP) problems are fundamental in control systems and often include inherent constraints critical to their formulation. Building on the continuous-time gradient descent methods and their circuit implementations developed thus far, we now address the challenge of incorporating constraints into the optimization framework. We revisit Problem 1 to integrate these limitations, thereby enhancing its relevance and applicability to practical scenarios:

Problem 4 (*Problem 1 with Constraints*)**.** Consider a set of agents, each characterized by a state vector x. The goal is to optimize the operational states of these agents by minimizing the objective function:

$$\begin{aligned}\min_x \quad & \tfrac{1}{2}x^\intercal Q x + \mathrm{e}^\intercal x\\ \text{s.t.} \quad & Sx = b\end{aligned} \tag{43}$$

where $Q \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix, $\mathrm{e} \in \mathbb{R}^n$ is a vector and $S \in \mathbb{R}^{n\times n}$ is the matrix defining equality constraints with corresponding vectors $b \in \mathbb{R}^n$.

4.1. Circuit design for primal–dual gradient descent method

Given that the circuits discussed in the previous section are tailored for unconstrained problems, an effective strategy to address our current problem involves reformulating it as an unconstrained problem. This can be achieved through the augmented Lagrangian method which integrates Lagrange multipliers directly into the objective function, effectively transforming the constraints into part of the optimization objective. The augmented Lagrangian for our problem is formulated as:

$$L(x, \lambda) = \frac{1}{2}x^\intercal Q x + x^\intercal \mathrm{e} + \lambda^\intercal(Sx - b) \tag{44}$$

where $\lambda \in \mathbb{R}^m$ is a vector of Lagrange multipliers, corresponding to m constraints. Leveraging strong duality, the Karush–Kuhn–Tucker (KKT) conditions for optimality are expressed as:

$$\begin{aligned}\frac{\partial L}{\partial x} &= 0 \Leftrightarrow 0 = Qx + S^\intercal\lambda + \mathrm{e},\\ \frac{\partial L}{\partial \lambda} &= 0 \Leftrightarrow 0 = Sx - b.\end{aligned} \tag{45}$$

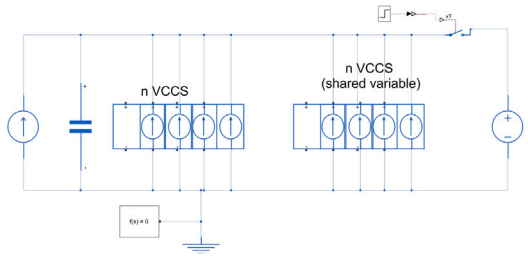


Fig. 5. Partial circuit model for primal–dual optimization problem.

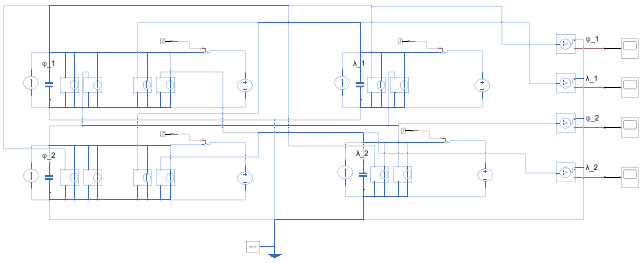


Fig. 6. Circuit representation for 2-D primal–dual gradient descent optimization.

Solving these linear equations yields the unique global optimizer (x^*, λ^*):

$$\begin{aligned} x^* &= -Q^{-1}(S^\intercal \lambda^* + \mathrm{e}), \\ \lambda^* &= -(SQ^{-1}S^\intercal)^{-1}(b + SQ^{-1}\mathrm{e}). \end{aligned} \tag{46}$$

While the augmented Lagrangian method provides an exact solution for the constrained optimization problem, its practical implementation requires centralized knowledge of the matrices S, Q, and vectors e, b. Any changes or temporal variations in these parameters would necessitate recalculations of the optimizer. In the context of the primal–dual framework, this scenario is characterized by the following differential equations:

$$T_x \dot{x} = -\frac{\partial}{\partial x} L(x, \lambda), \quad T_\lambda \dot{\lambda} = \frac{\partial}{\partial \lambda} L(x, \lambda) \tag{47}$$

where T_x and T_λ are positive definite diagonal matrices representing time constants. The dynamics of the optimization are then expressed by:

$$\begin{aligned} T_x \dot{x} &= -T_x \frac{\partial L}{\partial x} = -Qx - S^\intercal \lambda - \mathrm{e}, \\ T_\lambda \dot{\lambda} &= T_\lambda \frac{\partial L}{\partial \lambda} = Sx - b, \end{aligned} \tag{48}$$

For a scalar instance, the primal problem is represented as $t_x \dot{x} = -qx - s\lambda - e$. To adapt this to a multi-dimensional context, we extend the formulation to encompass each ith dimension in the primal problem. In this expanded view, the equation for each dimension is expressed as $t_{x_i} \dot{x}_i = -\sum_{j=1}^n Q_{i,j} x_j - \sum_{j=1}^n S_{j,i} \lambda_j - e_i$.

In parallel, the dual problem for each dimension i is defined as $t_{\lambda_i} \dot{\lambda}_i = \sum_{j=1}^n S_{i,j} x_j - b_i$. Combining both equations, it is required to implement

$$\begin{aligned} \text{Primal: } \dot{x}_i &= -\sum_{j=1}^n \frac{Q_{i,j} x_j}{t_{x_i}} - \sum_{j=1}^n \frac{S_{j,i} \lambda_j}{t_{x_i}} - \frac{e_i}{t_{x_i}}, \\ \text{Dual: } \dot{\lambda}_i &= \sum_{j=1}^n \frac{S_{i,j} x_j}{t_{\lambda_i}} - \frac{b_i}{t_{\lambda_i}}. \end{aligned} \tag{49}$$

The sub-circuit model for this setup can be simplified to the representation $\dot{x} = c_1 x + c_2$. Given the higher costs of op-amps compared to passive components, we have adapted the methods from [29] to optimize cost-effectiveness in our approach. This model is realized using several key components: 1. A capacitor is utilized to represent the first-order derivative term; 2. Voltage-Controlled Current Sources (VCCSs) are employed to create variables proportional to the coefficients in the equation; 3. A constant term is integrated into the circuit using a current source.

To accurately set the initial condition of the system, we incorporate a voltage source coupled with a switch. This setup is activated at $t = 0$ to establish the initial voltage across the capacitor, corresponding to the initial value of the variable in question. The circuit configuration that encapsulates this design is illustrated in Fig. 5.

Applying Kirchhoff's Current Law (KCL) to the node at the top of our proposed circuit configuration, we derive the following equation:

$$C \frac{dv}{dt} = \sum K_1 v + \sum K_2 v + I_s. \tag{50}$$

This equation effectively captures the structure of each equation in (49). By dividing by the capacitance C on both sides, we obtain:

$$\frac{dv}{dt} = \sum \frac{K_1}{C} v + \sum \frac{K_2}{C} v + \frac{I_s}{C}. \tag{51}$$

The terms involving K_1 and K_2 in conjunction with the capacitor C create a dynamic that mirrors the primal and dual equations of the optimization problem. In Fig. 6, we illustrate a two-dimensional representation of this circuit arrangement, designed specifically for applications based on the primal–dual gradient descent method. This figure provides a visual guide to understanding how the theoretical aspects of the optimization problem are translated into practical circuit configurations.

5. Simulation results

In our simulation, we focus on three key aspects: Firstly, we analyze the convergence speeds of both discrete and continuous gradient descent methods, particularly emphasizing trajectory analysis in a two-dimensional space and comparing achieved versus desired steady states. Secondly, we delve into the effects of the parameter β on the convergence rate in different continuous-time gradient descent methods. Finally, we assess error estimations in equivalent circuits, juxtaposing these with outcomes from ordinary differential equations (ODE) functions, to underscore the intricate interplay between circuit parameters and system behavior.

5.1. Convergence speed for gradient descent methods

In this subsection, we start by exploring the convergence rates of different gradient descent methodologies. In discrete methods using the code in [33], factors such as the initial guess and the methods way of functioning influence the convergence speed. To illustrate this, we consider a dynamic system characterized by a high bias rate in a random direction and a 6-dimensional configuration. Fig. 7 depicts the error evaluation, represented by $\epsilon = |x - x^*|$, across various methods in a complex, randomly selected dynamic system.

The Triple-Momentum without considering the tuned output o is named as Triple Variables. Our simulations reveal that the triple momentum method exhibits enhanced convergence rates, aligning with the findings of [15]. Nonetheless, this method still requires a significant number of steps for convergence. In contrast, given the same $Q \in \mathbb{R}^{6\times6}$ and $e \in \mathbb{R}^6$, continuous-time gradient methods demonstrate a rapid and stable convergence. As shown in Fig. 8, the continuous Heavy-Ball method and the continuous Triple-Momentum method can attain an accuracy level of 10^{-6} within 1 s for $\beta = 1000$. This performance corresponds to a value of around 10^{-2} for the capacitors and inductors in both the continuous Heavy-Ball method and the continuous Triple-Momentum method—a level of accuracy that the fastest discrete-time method fails to reach within 100 steps.

Many continuous-time approaches, such as the $\frac{3}{t}$-damped flow $\ddot{X}(t) + \frac{3}{t}\dot{X}(t) + \nabla f(X(t)) = 0$ introduced in [17], provide useful

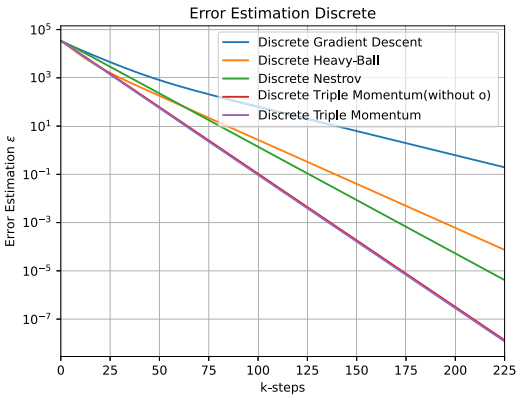


Fig. 7. Comparison of error estimations in different discrete-time gradient descent methodologies under a complex random dynamic system.

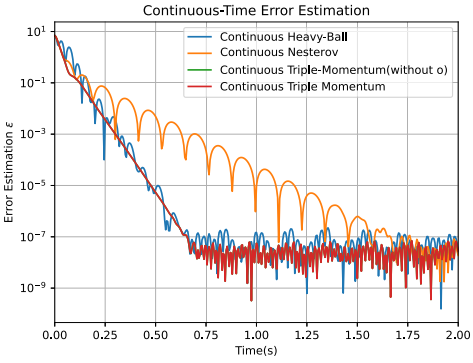
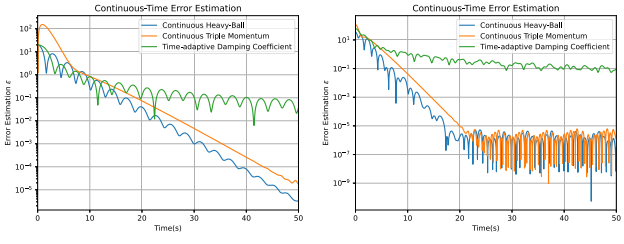


Fig. 8. Comparison of error estimations in different continuous-time gradient descent methodologies under a complex random dynamic system.



(a) Error evolution for a function in 2 dimensions.

(b) Error evolution for a function in 6 dimensions.

Fig. 9. Comparison of the $\frac{3}{t}$-damped ODE and our continuous-time gradient descent methods at $\beta = 1$, demonstrating differences in convergence behavior across problems of varying dimension.

analogies to discrete-time acceleration and offer Nesterov-like asymptotic rates. However, in practice, their performance can be limited by forcing a specific time-dependent damping ratio. In contrast, our proposed continuous-time frameworks employ adjustable parameters to achieve both stable and fast convergence. This flexibility also enhances practical implementation (e.g., with constant parameters in circuits). As shown in Fig. 9, at $\beta = 1$ our parameterized methods, including a continuous Heavy-Ball method inspired by Nesterov's acceleration, outperform time-dependent continuous schemes, especially as dimensionality increases.

One envisioned application in this paper is the possibility to use gradient methods to generate trajectories for dynamical systems. In Figs. 10 and 11 we provide the trajectories that could be used for autonomous vehicles to be tracked by a low-level controller. As it can

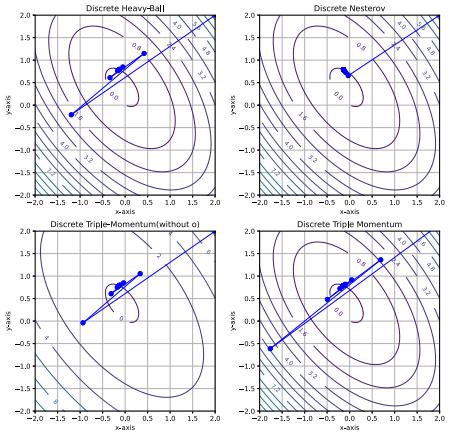


Fig. 10. Trajectory for discrete-time gradient descent methods.

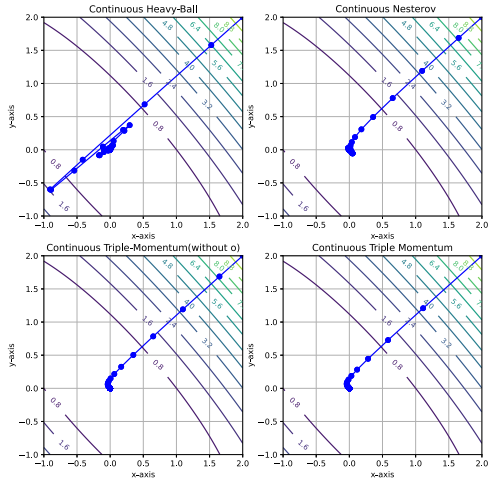


Fig. 11. Trajectory for in a continuous-time gradient descent methods.

be observed, the use of discrete-time versions of the gradient methods does not return smooth trajectories, which would preclude its use.

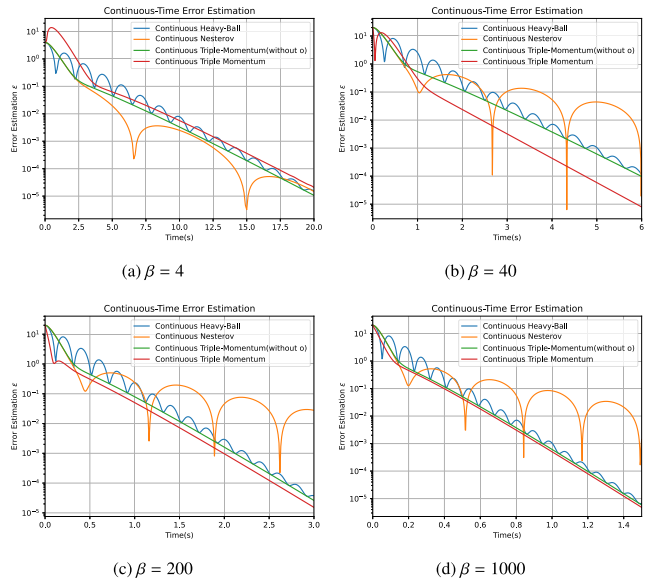
5.2. Convergence analysis of continuous-time gradient descent methods

The choice of the β parameter for the continuous gradient methods is of prime importance. The convergence speed in a 2-dimensional randomly generated function, as shown in Fig. 12, varies with different selections of β. We observe that for $\beta \geq 200$, the system quickly reaches a steady-state with an error below 10^{-6}, indicating rapid convergence. This observation suggests that the primary constraint on convergence speed in practice is the physical limitation, specifically, the minimum achievable values for capacitors and inductors. Moreover, as the values of these components decrease, the resistance inherent in the capacitors and even the circuit lines introduces significant noise to the outcomes.

5.3. Equivalent circuits of continuous-time gradient descent methods

This subsection presents the simulation results for equivalent circuit representations in continuous-time optimization methods. Fig. 13 shows the error, with the results being comparable to those obtained using the Python ode solver—akin to MATLAB's ode45. This comparison uses the same duration as depicted in Fig. 12, with identical parameters and initial value x_0.

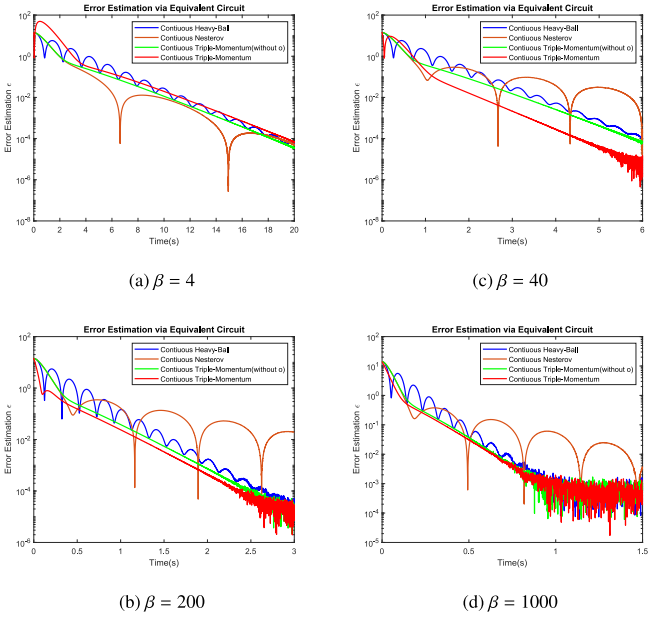
Following this, Fig. 14 presents a comparative analysis that underscores the alignment between Python simulation results and circuit



(a) $\beta = 4$ (b) $\beta = 40$

(c) $\beta = 200$ (d) $\beta = 1000$

Fig. 12. Error $\epsilon = |x - x^*|$ with different chosen of β.



(a) $\beta = 4$ (c) $\beta = 40$

(b) $\beta = 200$ (d) $\beta = 1000$

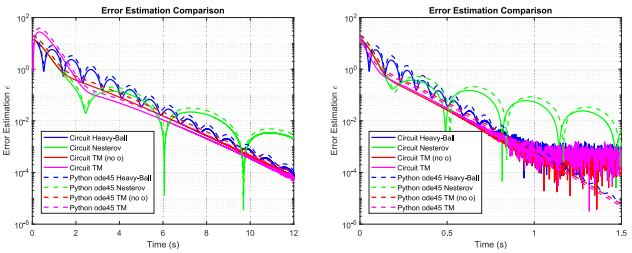
Fig. 13. Error evolution $\epsilon = |x - x^*|$ for the circuit equivalent with different choices of β.

simulations for continuous-time methods. This figure clearly displays two sets of results: dashed lines from the Python solver using ordinary differential equations (ODEs), and solid lines representing practical outcomes from equivalent circuit simulations as shown in Fig. 13.

Analyzing the outcomes presented in these figures, we observe a primary distinction: When $\beta \geq 40$, after the error estimation falls below 10^{-5}, the convergence patterns show oscillations in the circuit results. Similarly, for $\beta \geq 1000$, oscillations appear after the error falls below 10^{-3}.

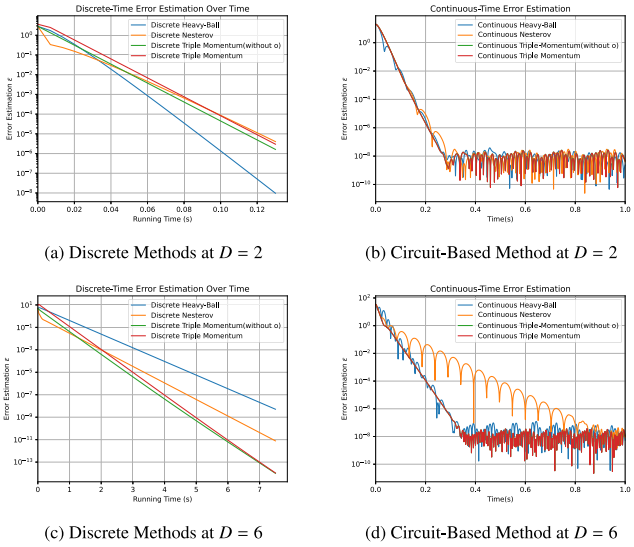
This behavior stems from scaling the circuit's setup voltages to $1/\beta$ of their original values to maintain voltages within acceptable limits, as discussed in Section 3.3. Opting for a larger β enhances the convergence rate, but this setup carefully balances accuracy with the required settling time for the system.

To further highlight the advantages of our circuit approach, we compare it against conventional discrete gradient descent methods. In



(a) $\beta = 10$ (b) $\beta = 1000$

Fig. 14. Error comparison between circuit equivalent and python ode45 with various selections of β.



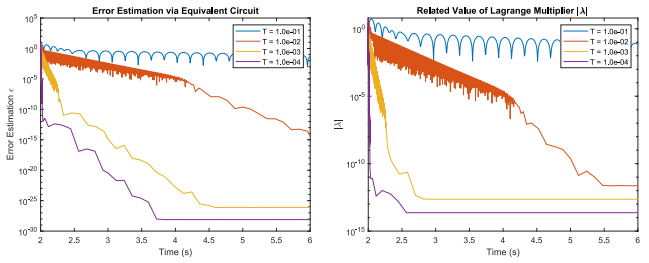
(a) Discrete Methods at $D = 2$ (b) Circuit-Based Method at $D = 2$

(c) Discrete Methods at $D = 6$ (d) Circuit-Based Method at $D = 6$

Fig. 15. Comparison of running times for discrete gradient methods and their equivalent continuous-time circuit implementations ($\beta = 4000$).

Fig. 15, the "running time" for the circuit-based method represents the theoretical performance of an actual analog circuit, as approximated by our MATLAB simulation. In contrast, the digital methods' timing is derived from CPU/GPU-based computations. For instance, on a computer with an AMD Ryzen 9 5900HS CPU and 16 GB RAM, each iteration of a discrete gradient method takes approximately 0.0065 s when $D = 2$ and increases to about 0.03 s at $D = 6$. In comparison, our circuit-based approach converges within about 0.5 s, regardless of dimensionality. In terms of power consumption, for the 2D case, the discrete gradient method on the computer consumes approximately 9.1 J, while the continuous gradient method on the analog circuit consumes only 0.1 J; similarly, for the 6D case, the computer method consumes 490 J, compared to 0.3 J for the analog circuit. These results point towards a direction of future research in conducting experiment with real circuits implemented, for instance with Field-Programmable Gate Arrays (FPGAs).

To analyze the constrained version of the problem, we examine the primal–dual problem and its equivalent circuit representation. The findings, depicted in Fig. 16, are contingent on the chosen time constants. Specifically, the selection of positive definite diagonal matrices for time constants, referred to as T, significantly impacts the circuit's convergence rate. Under randomly generated system parameters and constraints, certain choices of T yield suboptimal performance. For instance, when T is on the order of 10^{-1}, the system's behavior is notably poor. Conversely, with T smaller than 10^{-3}, the system rapidly converges. This behavior underscores the potential advantages of using equivalent circuits to address related optimization problems, despite



(a) Error Estimation (b) Magnitude of λ

Fig. 16. Error estimation via equivalent circuit between the current state and the desired state $\epsilon = |x - x^*|$ with different chosen of η.

the inherent challenges posed by simple 2-D dynamic systems. In Fig. 16, the value chosen for T affects the convergence behavior, illustrating the nuanced relationship between circuit parameters and system dynamics.

6. Conclusion

In this paper, we have proposed continuous-time counterparts to discrete-time gradient descent methods and presented optimal parameters to obtain the fastest worst-case convergence rate. Through numerical simulations, we show that the proposed methods have better accuracy and convergence properties than previously known continuous gradient methods expressions with time-varying constants. We have also provided circuit implementations for the theoretical differential equations with the objective of solving the optimization problems using analog computing. Lastly, we adapt these methods to primal dual algorithms in order to be able to solve constrained optimization programs.

CRediT authorship contribution statement

He Hao: Writing – original draft, Software, Methodology, Investigation, Formal analysis. **Daniel Silvestre:** Writing – review & editing, Supervision, Methodology, Investigation, Funding acquisition, Conceptualization. **Carlos Silvestre:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Daniel Silvestre reports financial support was provided by Foundation for Science and Technology. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The work from Daniel Silvestre is supported in part by the Portuguese Foundation for Science and Technology through the project CTS/00066. The work from Carlos Silvestre is supported in part by the Macao Science and Technology Development Fund under Grant FDCT/0192/2023/RIA3, and in part by the University of Macau, Macao, China, Project MYRG2022-00205-FST.

Data availability

No data was used for the research described in the article.

References

[1] T. Baca, D. Hert, G. Loianno, M. Saska, V. Kumar, Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of unmanned aerial vehicles, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2018, pp. 6753–6760, http://dx.doi.org/10.1109/IROS.2018.8594266.

[2] Q. Yuan, X. Li, Distributed model predictive formation control for a group of UAVs with spatial kinematics and unidirectional data transmissions, IEEE Trans. Netw. Sci. Eng. 10 (6) (2023) 3209–3222, http://dx.doi.org/10.1109/TNSE.2023.3252724.

[3] D. Silvestre, J. Hespanha, C. Silvestre, Fast desynchronization algorithms for decentralized medium access control based on iterative linear equation solvers, IEEE Trans. Autom. Control 67 (11) (2022) 6219–6226, http://dx.doi.org/10.1109/TAC.2021.3130888.

[4] R.M. Saback, A.G.S. Conceicao, T.L.M. Santos, J. Albiez, M. Reis, Nonlinear model predictive control applied to an autonomous underwater vehicle, IEEE J. Ocean. Eng. 45 (3) (2020) 799–812, http://dx.doi.org/10.1109/JOE.2019.2919860.

[5] M.J. Er, H. Gong, Y. Liu, T. Liu, Intelligent trajectory tracking and formation control of underactuated autonomous underwater vehicles: A critical review, IEEE Trans. Syst. Man, Cybern.: Syst. (2023) 1–13, http://dx.doi.org/10.1109/TSMC.2023.3312268.

[6] R. Ribeiro, D. Silvestre, C. Silvestre, Decentralized control for multi-agent missions based on flocking rules, in: J.A. Gon¸calves, M. Braz-César, J.a.P. Coelho (Eds.), CONTROLO 2020, Springer International Publishing, Cham, 2021, pp. 445–454.

[7] R. Ribeiro, D. Silvestre, C. Silvestre, A rendezvous algorithm for multi-agent systems in disconnected network topologies, in: 2020 28th Mediterranean Conference on Control and Automation, MED, 2020, pp. 592–597, http://dx.doi.org/10.1109/MED48518.2020.9183093.

[8] R. Thomazella, J.E. Castanho, F. Dotto, O.R. Júnior, G. Rosa, A. Marana, J. Papa, Environmental monitoring using drone images and convolutional neural networks, in: IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium, 2018, pp. 8941–8944, http://dx.doi.org/10.1109/IGARSS.2018.8518581.

[9] Y. Wu, S. Wu, X. Hu, Cooperative path planning of & and UGVs for a persistent surveillance task in urban environments, IEEE Internet Things J. 8 (6) (2021) 4906–4919, http://dx.doi.org/10.1109/JIOT.2020.3030240.

[10] G. Rohi, O. Ejofodomi, G. Ofualagba, Autonomous monitoring, analysis, and countering of air pollution using environmental drones, Heliyon 6 (1) (2020) e03252, http://dx.doi.org/10.1016/j.heliyon.2020.e03252.

[11] C.J. Shin, S.E. Seo, Y. Nam, K.H. Kim, L. Kim, J. Kim, E. Ryu, J.Y. Hwang, G.-J. Kim, M.-W. Jung, S.H. Lee, O.S. Kwon, Real-time monitoring of cyanobacterial harmful algal blooms by graphene field-effect transistor, Chem. Eng. J. 459 (2023) 141419, http://dx.doi.org/10.1016/j.cej.2023.141419.

[12] H.-C. Chang, Y.-L. Hsu, C.-Y. Hsiao, Y.-F. Chen, Design and implementation of an intelligent autonomous surveillance system for indoor environments, IEEE Sensors J. 21 (15) (2021) 17335–17349, http://dx.doi.org/10.1109/JSEN.2021.3081831.

[13] B. Polyak, Some methods of speeding up the convergence of iteration methods, USSR Comput. Math. Math. Phys. 4 (5) (1964) 1–17, http://dx.doi.org/10.1016/0041-5553(64)90137-5.

[14] Y. Nesterov, Introductory Lectures on Convex Optimization: A Basic Course, first ed., Springer Publishing Company, Incorporated, 2014.

[15] B. Van Scoy, R.A. Freeman, K.M. Lynch, The fastest known globally convergent first-order method for minimizing strongly convex functions, IEEE Control. Syst. Lett. 2 (1) (2018) 49–54, http://dx.doi.org/10.1109/LCSYS.2017.2722406.

[16] D. Silvestre, J. Hespanha, C. Silvestre, Desynchronization for decentralized medium access control based on Gauss-seidel iterations, in: 2019 American Control Conference, ACC, 2019, pp. 4049–4054, http://dx.doi.org/10.23919/ACC.2019.8814471.

[17] W.J. Su, S.P. Boyd, E.J. Candès, A differential equation for modeling nesterov's accelerated gradient method: Theory and insights, J. Mach. Learn. Res. 17 (2014) 153:1–153:43.

[18] B. Shi, S.S. Du, M.I. Jordan, W.J. Su, Understanding the acceleration phenomenon via high-resolution differential equations, Math. Program. 195 (1) (2022) 79–148, http://dx.doi.org/10.1007/s10107-021-01681-8.

[19] S. Köppel, B. Ulmann, L. Heimann, D. Killat, Using analog computers in today's largest computational challenges, Adv. Radio Sci. (2021).

[20] B. Ulmann, Analog and Hybrid Computer Programming, De Gruyter Oldenbourg, Berlin, Boston, 2023, http://dx.doi.org/10.1515/9783110787733.

[21] S. Mandal, J. Liang, H. Malavipathirana, N. Udayanga, H. Silva, S.I. Hariharan, A. Madanayake, Integrated analog computers as domain-specific accelerators: A tutorial review, in: 2024 IEEE 67th International Midwest Symposium on Circuits and Systems, MWSCAS, 2024, pp. 875–881, http://dx.doi.org/10.1109/MWSCAS60917.2024.10658915.

[22] N. Guo, Y. Huang, T. Mai, S. Patil, C. Cao, M. Seok, S. Sethumadhavan, Y. Tsividis, Energy-efficient hybrid analog/digital approximate computation in continuous time, IEEE J. Solid-State Circuits 51 (7) (2016) 1514–1524, http://dx.doi.org/10.1109/JSSC.2016.2543729.

[23] G. Cowan, R. Melville, Y. Tsividis, A VLSI analog computer/digital computer accelerator, IEEE J. Solid-State Circuits 41 (1) (2006) 42–53, http://dx.doi.org/10.1109/JSSC.2005.858618.

[24] J.-H. Li, Genetic algorithm PID control of second-order analog circuit system, in: 2022 International Conference on System Science and Engineering, ICSSE, 2022, pp. 022–025, http://dx.doi.org/10.1109/ICSSE55923.2022.9948240.

[25] R.M. Levenson, A.A. Adegbege, Analog circuit for real-time optimization of constrained control, in: 2016 American Control Conference, ACC, 2016, pp. 6947–6952, http://dx.doi.org/10.1109/ACC.2016.7526767.

[26] B.E. Varghese, A. Swarup, Novel op-amp based hardware design of analog PID circuit to control the altitude of quadcopter, in: 2022 2nd Asian Conference on Innovation in Technology, ASIANCON, 2022, pp. 1–8, http://dx.doi.org/10.1109/ASIANCON55314.2022.9908807.

[27] X. Jin, W. Che, Z. Wu, H. Wang, Analog control circuit designs for a class of continuous-time adaptive fault-tolerant control systems, IEEE Trans. Cybern. 52 (2020) 4209–4220.

[28] G. Pappas, V. Alimisis, C. Dimas, P.P. Sotiriadis, Analogue realization of a fully tunable fractional-order PID controller for a DC motor, in: 2020 32nd International Conference on Microelectronics, ICM, 2020, pp. 1–4, http://dx.doi.org/10.1109/ICM50269.2020.9331798.

[29] A. Agarwal, C. Fiscko, S. Kar, L. Pileggi, B. Sinopoli, An equivalent circuit workflow for unconstrained optimization, 2023, arXiv:2305.14061.

[30] A. Agarwal, L. Pileggi, An equivalent circuit approach to distributed optimization, 2023, arXiv:2305.14607.

[31] J. Araújo, L. Oliveira, B. Guerreiro, F. Silva, CMOS analog simulators of dynamical systems, IEEE Access PP (2024) http://dx.doi.org/10.1109/ACCESS.2024.3391511, 1–1.

[32] P. Horowitz, W. Hill, The Art of Electronics, third ed., Cambridge University Press, USA, 2015.

[33] D. Silvestre, OPTool—An optimization toolbox for iterative algorithms, SoftwareX 11 (2020) 100371, http://dx.doi.org/10.1016/j.softx.2019.100371.