Design of second order neural networks as dynamical control systems that aim to minimize nonconvex scalar functions

Fernando A. Pazos, Amit Bhaya and Eugenius Kaszkurewicz

August 12, 2011

Abstract

This article presents an unified way to design neural networks characterized as second order ordinary differential equations (ODE), that aim to find the global minimum of nonconvex scalar functions. These neural networks, alternative referred to as continuous time algorithms, are interpreted as dynamical closed loop control systems. The design is based on the control Liapunov function (CLF) method. For nonconvex scalar functions, the goal of these algorithms is to produce trajectories, starting from an arbitrarily chosen initial guess, that do not get stuck in local minima, thereby increasing the chances of converging to the global minimum.

1 Introduction

Neural networks are typically analog circuit implementations of dynamical systems represented by ordinary differential equations (ODE). There has been a revival of interest in analog or ordinary differential equations based methods for optimization, dating from the seminal paper of Karmarkar and the subsequent boom in interior point and trajectory following methods (see, for example, [1, 2, 3] and the references therein). In particular, some old proposals for ODE-based methods have been revived recently and an objective of this paper is to demonstrate that these methods, as well as some new methods, proposed herein, arise in a natural manner by taking a control viewpoint.

Another connection of the present paper with the field of neural networks has its origin in the well known backpropagation with momentum (BPM) method used to solve the nonconvex problem of choosing the weights of a feedforward artificial neural network. The paper [4] showed that the BPM method, which is essentially steepest descent with momentum, for quadratic functions is a version of the conjugate gradient (CG) method. In addition, the BPM method was also shown to be a discretization of a continuous-time or ODE method for minimizing nonconvex functions, known as the *heavy ball with friction* (HBF) method, proposed by Polyak [5] and later studied in detail in [6, 7, 8]. A continuous-time version of the CG method was also proposed in [4], and it will be generalized and studied in detail herein.

Methods to solve different optimization problems arise in a unified and natural manner by taking a control viewpoint. This control perspective was presented in [3, 9, 10]. In these references, the authors formulate the problem of

finding a zero of a vector function as that of designing a closed-loop system, with a plant defined as the objective function, and with a static controller. The overall closed loop system is modeled by a first-order ODE whose trajectories converge to the desired zeros of the function. The choice of the feedback controller is based on the control Liapunov function (CLF) method (detailed in the present context in the book [3]), well known in control theory. This control perspective offers powerful tools for the design and analysis of systems to solve optimization problems.

Following this CLF methodology, this paper presents several second order algorithms, adequate for implementation as neural networks, to minimize nonconvex scalar functions, where the words second order refer to the fact that the closed-loop dynamics is described by a second order differential equation, and, in the control perspective, this means that the controller is now dynamic instead of static.

In order to proceed, consider a scalar function $\phi(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$, continuous and with continuous partial derivatives, such that for all $\mathbf{x} \in \mathrm{dom}(\phi(\mathbf{x}))$, $\phi(\mathbf{x}) \geq \phi(\mathbf{x}^*) = \phi_{\min}$, i.e. the global minimum $\mathbf{x}^*$ is assumed to exist and be finite, and $\phi(\mathbf{x})$ tends to infinity when $\|\mathbf{x}\|$ tends to infinity in any direction. It is desired to find $\mathbf{x}^*$, by following an ODE trajectory that starts from some given initial point $\mathbf{x}_0$.

The main motivation to use second order algorithms, instead of first order ones, is as follows. It is well known that, for nonconvex functions, trajectories $\mathbf{x}(t)$ starting at an initial point $\mathbf{x}(0) = \mathbf{x}_0 \in \mathrm{dom}(\phi(\mathbf{x}))$ generated by first order gradient-type algorithms, converge to any point where the gradient of the function is zero, even though this point may be a local minimum of the function; using second order algorithms, the hope is that, with an adequate choice of parameters (gains), trajectories are able to converge to the global minimum of the function $\mathbf{x}^*$. These ODEs are suitable for implementation as neural networks.

Preliminary and partial versions of this work were first presented in [11] and [12]. In [11], several second order ODE's were interpreted and designed as closed-loop control system using CLFs. The ODE's studied in [11] were the *heavy ball with friction* (HBF) method, the *dynamical inertial Newton-like system* (DIN), proposed by Alvarez et. al. [13], and a continuous-time version of the *conjugate gradient* (CG) method that was first proposed for quadratic functions in [4] and generalized in [11]. In addition, other second order systems to minimize nonconvex scalar functions were also designed using the CLF method, by choosing of different candidate Liapunov functions and with adequate choices of the plant and the controller of the systems. In [12], a comparative study of some generalizations of the HBF method proposed therein are outperformed on a suite of standard nonconvex problems used in the literature on global optimization by the continuous version of the CG algorithm.

All these methods and new ones can all be designed easily by the application of the CLF method, including methods inspired by mechanical or other physical analogies, but also going beyond these. Of course, this does not free the designer from the necessity of carrying out numerical experiments to check the behavior of the nets under the usual non-ideal circumstances of practical implementations.

2 Preliminaries: Motivation and CLF-based design

In optimization theory, it is well known that continuous-time trajectories generated by gradient descent methods are able to converge to a local minimum of a scalar objective function when it is continuous and has continuous partial derivatives. Moreover, when this objective function is convex, the minimum is unique and global.

In order to arrive at a natural motivation for the approach proposed in this paper, first consider a continuous version of the *steepest descent* (SD) algorithm, written as:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{u} && (1) \\ \mathbf{u} &= -\nabla\phi(\mathbf{x}(t)) && (2) \end{aligned}$$

where $\mathbf{u} \in \mathbb{R}^n$ and the gradient of ϕ is written $\nabla\phi : \mathbb{R}^n \to \mathbb{R}^n : \mathbf{x} \mapsto \nabla\phi(\mathbf{x})$. The reason that (1) and (2) are written in this fashion is twofold: first, it is clear that its trajectories will converge to local minima satisfying the necessary condition $\nabla\phi(\mathbf{x}(t)) = 0$ and second, in order to escape from confinement to a local minimum, a more sophisticated choice of $\mathbf{u}$ (which is being thought of as the control) is needed in order to drive $\mathbf{x}$ to the global minimum $\mathbf{x}^*$ ($\phi(\cdot)$ is being thought of as defining the output of the controlled object or plant). Thus, one possibility is that $\mathbf{u}$ must itself be subject to some dynamics and, if this dynamics is chosen to be first-order, it immediately implies that the overall dynamics will be a second-order ODE in $\mathbf{x}$. In this manner, one is led naturally to the introduction of second-order ODEs and the remaining question is how to design $\mathbf{u}$.

Assume that $\mathbf{u}$ has first-order dynamics, i.e., (2) is replaced by:

$$\dot{\mathbf{u}}(t) = \psi(\mathbf{x}(t), \dot{\mathbf{x}}(t)) \tag{3}$$

Then, a natural CLF arises from the attempt to solve the so called regulation problem, in which the reference signal is the global minimum value of the function $\phi_{\min}$, and the output of the plant, $\phi(\mathbf{x})$, is to be driven to this value. This is to be done by choice of the RHS $\psi(\mathbf{x}, \dot{\mathbf{x}})$ of the $\mathbf{u}$-dynamics (3), also ensuring that $\mathbf{u}$ goes to zero, so that the $\mathbf{x}$-dynamics (1) converges to the desired global equilibrium. In other words, a candidate control Liapunov function (CLF) can be written as follows:

$$V(\mathbf{x}, \mathbf{u}) = \phi(\mathbf{x}) - \phi_{\min} + \frac{\mathbf{u}^T\mathbf{u}}{2} \geq 0 \tag{4}$$

which is greater than zero for all $\mathbf{x} \neq \mathbf{x}^*$ or $\mathbf{u} \neq \mathbf{0}$, $V(\mathbf{x}^*, \mathbf{0}) = 0$, and $V(\mathbf{x}, \mathbf{u}) \to \infty$ as $\|\mathbf{x}\| \to \infty$ or $\|\mathbf{u}\| \to \infty$.

The time derivative of (4) is:

$$\dot{V}(\mathbf{x}, \mathbf{u}) = \nabla\phi(\mathbf{x})^T\dot{\mathbf{x}} + \mathbf{u}^T\dot{\mathbf{u}} \tag{5}$$

Equations (1), (3), (4) and (5) are the starting point for the CLF design of new second-order ODEs by choosing $\dot{\mathbf{x}}$ and $\dot{\mathbf{u}}$ in (5) (respectively, the plant dynamics and the controller dynamics in control jargon) so that $\dot{V}(\mathbf{x}, \mathbf{u}) \leq 0$,

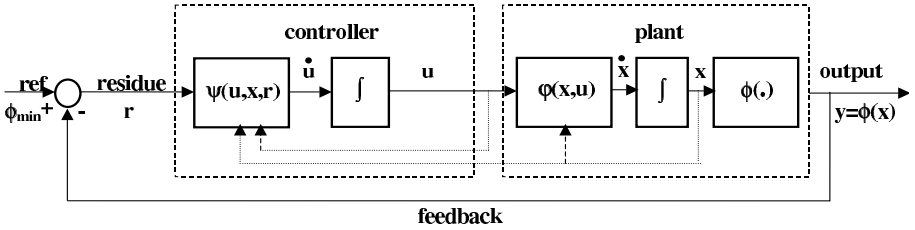

Figure 1: Block diagram of the second order closed loop control system

ensuring stability by the Liapunov's direct method (see [14, c. 3]). The name CLF is justified by the fact that $\dot{V}$ is made negative by choice of the control or **u**-dynamics, given some choice of the plant or **x**-dynamics.

Figure 1 shows a block diagram of the overall control system.

An important observation is that, although the global minimum value of the function ($\phi_{\min}$) occurs in the definition of the candidate CLF (4), it disappears in its derivative (5), which is the crucial equation for the design. In other words, knowledge of $\phi_{\min}$ is not required for the CLF design.

2.1 Related works

Second-order algorithms to minimize nonscalar functions are not very abundant in the literature and this section briefly describes some of the existing ones. The best known algorithm of the class of so called second-order algorithms is the *"heavy ball with friction"* (HBF), already referred to in the introduction. It was first introduced in [5] (also see [6],[13],[7],[8]) and can be represented by the following second-order differential equation:

$$\ddot{\mathbf{x}} + \gamma\dot{\mathbf{x}} + \nabla\phi(\mathbf{x}) = \mathbf{0} \tag{6}$$

where γ is a positive scalar parameter.

In a mechanical analogy of (6), the HBF ODE represents a dissipative system, the parameter γ being interpreted as viscous friction. As pointed out in [7], the damping term $\gamma\dot{\mathbf{x}}(t)$ confers optimizing properties on (6), but it is isotropic and ignores the geometry of ϕ. The second derivative term $\ddot{\mathbf{x}}(t)$, which induces inertial effects, is a singular perturbation or regularization of the classical *Newton ODE*, which may be written as follows:

$$\nabla^2\phi(\mathbf{x}(t))\dot{\mathbf{x}}(t) + \nabla\phi(\mathbf{x}(t)) = \mathbf{0} \tag{7}$$

where $\nabla^2\phi(\mathbf{x})$ denotes the Hessian matrix of the scalar function $\phi(\mathbf{x})$.

Note that, in this mechanical analogy made by Polyak, the candidate Liapunov function (4) can be interpreted as the mechanical energy of a ball of unit mass rolling on a surface represented by the function $\phi(\mathbf{x})$ with a unitary gravitational constant. The first term in (4) represents the potential energy, while the second term in (4) represents the kinetic energy, where the velocity $\mathbf{u} = \dot{\mathbf{x}}$ is defined as in (1). Hence, if the constant γ in equation (6) is chosen as zero, there is no friction term, and the mechanical energy remains constant (hence $\dot{V}(\mathbf{x}, \mathbf{u}) = 0$ in (5)). With a positive friction parameter γ, the mechanical energy decreases with the time (hence $\dot{V}(\mathbf{x}, \mathbf{u}) < 0$ in (5)). The proof will be presented further in the following section.

Alvarez *et al.* ([13]), present a modification of the HBF algorithm, termed *"dynamical inertial Newton-like system"* (DIN). The modification consists of the addition of a spatial damping term, in addition to the existing temporal one:

$$\ddot{\mathbf{x}} + a\dot{\mathbf{x}} + b\nabla^2\phi(\mathbf{x})\dot{\mathbf{x}} + \nabla\phi(\mathbf{x}) = \mathbf{0} \tag{8}$$

where a and b are positive scalar parameters. The authors note that HBF is a better version of the Newton algorithm, and when the function $\phi(\mathbf{x})$ is convex, trajectories converge quickly to the global minimum. However, the trajectories generated by HBF can present spatial oscillations, which are damped by the term $b\nabla^2\phi(\mathbf{x})\dot{\mathbf{x}}$ in the DIN algorithm, even when the Hessian is degenerate ([13]). Hence, trajectories generated by this algorithm could have better behavior in the sense of converging faster to the minimum of the function. In a similar direction, DIN can be viewed as a second-order inertial relaxation of a Hessian-type preconditioned gradient method for the local minimization of an objective function (see [15]).

Cabot ([8]), proposes the following second order dynamical system:

$$\ddot{\mathbf{x}} + \gamma\dot{\mathbf{x}} + \nabla\phi(\mathbf{x}) + \epsilon(t)\nabla U(\mathbf{x}) = \mathbf{0} \tag{9}$$

where γ is a positive scalar parameter, $\epsilon(t)$ is a positive scalar function that tends to zero when t tends to infinity, and $U(\mathbf{x})$ is a positive scalar cost function.

The author demonstrates that if $\phi(\mathbf{x})$ and $U(\mathbf{x})$ are convex, then the trajectories generated by (9) converge to a minimum of U in the set $\operatorname{argmin}(\phi)$, assuming that this set is non empty. The application of this algorithm is the following: if $\phi(\mathbf{x})$ is convex and the set $\operatorname{argmin}(\phi)$ is convex, the problem becomes a particular case of a convex optimization problem, where the feasible set can be defined as $\chi = \{\operatorname{argmin}(\phi(\mathbf{x}))\}$, and which can be described as:

$$\begin{aligned} \min \quad & U(\mathbf{x}) \\ \text{s.t} \quad & \mathbf{x} \in \operatorname{argmin}(\phi(\mathbf{x})) \end{aligned}$$

The Heavy Ball idea, without friction, was taken up by Snyman and Fatti [16], who reevaluated the method in [17]. Since they used inertial frictionless (gradient) descent, they proposed random initialization from multiple starting points, as well as a heuristic technique to modify the trajectories in a manner that ensures, in the case of multiple local minima, a higher probability of convergence to a lower local minimum than would have been achieved had conventional gradient local search methods been used. Shimizu et al. [18] used the HBF ODE, but proposed an additional "attraction-repulsion" spring-like term to modify trajectories and introduce chaotic dynamics to favor convergence to the global minimum.

Other methods that are based on the continuous-time steepest descent and use other local minima escape strategies such as "subenergy tunneling" and "terminal repellers" [19, 20, 21] are less directly related to the HBF method.

Other continuous-time global search methods for solving unconstrained nonlinear optimization problems, not necessarily based on second order ODE's, can be found in [22, c. 3], [23], [24], [25], [26], [27] and [28].

A complete review about recent advances in global optimization can be found in [29].

The conjugate gradient (CG) method is a well known discrete algorithm applied to unconstrained optimization. It is commonly used to minimize scalar

convex quadratic functions. In the control perspective, the CG method can be viewed as an acceleration of the *steepest descent* method, which can be thought of as the standard feedback control system with a proportional controller.

The acceleration is achieved by using a discrete version of a classical control strategy for faster "closed-loop" response (i.e., acceleration of convergence to the equilibrium): this strategy is known as derivative action in the control (see [4] and references therein).

Along this line of thought, a natural approach to continuous time version of the CG method was first presented in [4, p. 69] and is restated below for convenience:

$$\dot{\mathbf{r}} = -\alpha(\mathbf{x},\mathbf{u})\nabla^2\phi(\mathbf{x})\mathbf{u} \tag{10}$$

$$\dot{\mathbf{u}} = \mathbf{r} - \beta(\mathbf{x},\mathbf{u})\mathbf{u} \tag{11}$$

where $\mathbf{r} := -\nabla\phi(\mathbf{x})$, $\nabla^2\phi(\mathbf{x}) \in \mathbb{R}^{n\times n}$ is the Hessian matrix and $\alpha, \beta : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ are scalar-valued functions of the state vector $[\mathbf{x}^T\ \mathbf{u}^T]^T$. In [4], the Hessian matrix is denoted as A because it is applied to minimize the convex scalar function $\phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{b}^T\mathbf{x}$, where $A = A^T \in \mathbb{R}^{n\times n}$ is symmetric positive definite, $\mathbf{b} \in \mathbb{R}^n$, and the vector $\mathbf{u}$ is denoted as $\mathbf{p}$, but here it has been changed to unify nomenclature. Since $\dot{\mathbf{r}} = -\nabla^2\phi(\mathbf{x})\dot{\mathbf{x}} = -\alpha(\mathbf{x},\mathbf{u})\nabla^2\phi(\mathbf{x})\mathbf{u}$, equations (10) and (11) can be written in $\mathbf{x}$-coordinates as:

$$\begin{aligned} \dot{\mathbf{x}} &= \alpha(\mathbf{x},\mathbf{u})\mathbf{u} \\ \dot{\mathbf{u}} &= -\nabla\phi(\mathbf{x}) - \beta(\mathbf{x},\mathbf{u})\mathbf{u} \end{aligned} \tag{12}$$

and these will be considered to be the defining equations of the *CG net*, so named to recall its origins as a continuous version of the conjugate gradient algorithm. It should be emphasized that, in the continuous time case (12), the letters CG are just a convenient mnemonic to recall the origin of the equation and not intended to draw attention to any conjugacy concepts. Two observations are important here: (i) the first order vector CG ODE (12) is a generalization of the second order HBF ODE (6), which is obtained by choosing $\alpha(\mathbf{x},\mathbf{u}) = 1$, $\beta(\mathbf{x},\mathbf{u}) = \gamma > 0$, $\forall\mathbf{x}$; (ii) the first order vector CG ODE (12) cannot in general be reduced to a second order ODE in $\mathbf{x}$ (see Remark 3.1).

In the backpropagation context, $\mathbf{x}$ is the weight vector, usually denoted $\mathbf{w}$, and the potential energy function ϕ is the error function usually denoted $E(\mathbf{w})$ (as in [30]). In fact, with these changes of notation, it is clear that the HBF equation (6) is exactly the equation that has been proposed in [30] as the continuous analog of BPM, using a similar physical model (point mass moving in a viscous medium with friction under the influence of a conservative force field and with Newtonian dynamics). Thus, the continuous version of BPM is the HBF ODE and may be regarded either as a regularization of the steepest descent ODE or the classical Newton ODE.

2.2 Analysis of the convergence of a second order algorithm to the global minimum of a scalar objective function

In this subsection, the goal is to show that the trajectories generated by a second order algorithm similar to the HBF (6) always can converge to the global

minimum of a scalar objective function $\phi(x) : \mathbb{R} \to \mathbb{R}$, making an analogy with the potential and the kinetic energies mentioned above.

A unitary mass ball rolling on a surface represented by the function $\phi(\mathbf{x})$, and affected by a weight force produced by a unitary gravitational constant, has a variation of kinetic energy $\frac{1}{2}\Delta\|\dot{\mathbf{x}}\|^2$, a variation of potential energy which can be expressed as $\Delta\phi(\mathbf{x})$, and, when the movement is affected by a viscous friction, there exists a non-conservative force whose work can be expressed as $-\gamma\|\Delta\mathbf{x}\|$, where γ is a friction parameter. Hence, by the conservation of energy:

$$\frac{1}{2}\Delta\|\dot{\mathbf{x}}\|^2 + \gamma\|\Delta\mathbf{x}\| + \Delta\phi(\mathbf{x}) = 0 \tag{13}$$

Therefore:

$$\begin{aligned}
\Rightarrow \quad & \frac{1}{2}\partial\|\dot{\mathbf{x}}\|^2 + \gamma\|\partial\mathbf{x}\| + \partial\phi(\mathbf{x}) = 0 \\
\Rightarrow \quad & \frac{1}{2}\frac{\partial\dot{\mathbf{x}}^T\dot{\mathbf{x}}}{\partial t} + \gamma\|\frac{\partial\mathbf{x}}{\partial t}\| + \frac{\partial\phi(\mathbf{x})}{\partial t} = 0 \\
\Rightarrow \quad & \ddot{\mathbf{x}}^T\dot{\mathbf{x}} + \gamma\frac{\dot{\mathbf{x}}^T\dot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|} + \nabla^T\phi(\mathbf{x})\dot{\mathbf{x}} = 0 \\
\Rightarrow \quad & \ddot{\mathbf{x}} + \gamma\frac{\dot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|} + \nabla\phi(\mathbf{x}) = \mathbf{0}
\end{aligned} \tag{14}$$

Without lost of generality, we assume the scalar case where $\phi(x) : \mathbb{R} \to \mathbb{R}$, such that $\phi(x)$ is continuous and with a continuous derivative. Of course, $\phi(x)$ can be non-convex. We also assume that there exists $x^* \in \mathbb{R}$ such that for all $x \neq x^*$, $\phi(x) > \phi(x^*)$, that is, the minimum is unique and finite. In the following lemma, we prove that for all initial point $x_0 \in \mathbb{R}$, there exist $\gamma > 0$ and $\dot{x}_0 \in \mathbb{R}$ such that trajectories generated by (14) converge to x^*.

Lemma 2.1 *For all $\phi(x) : \mathbb{R} \to \mathbb{R} \mid \phi(x) \in C_1$, for all $x_0 \in \mathbb{R}$: there exist $\dot{x}_0$ and $\gamma > 0$ such that trajectories generated by (14) converge to x^*.*

The hypotheses are:

a) There exists $x^* \in \mathbb{R}$ such that for all $x \neq x^*$: $\phi(x^*) < \phi(x)$.

b) For all $x_0 \in \mathbb{R}$ there exists $x_1 \in [x_0, x^*)$ such that for all $x \in [x_0, x^*)$, $\phi(x_1) \geq \phi(x)$.[1]

c) There exists $\delta > 0$ and $x_2 \in \mathbb{R}$ such that $x^* \in (x_1, x_2)$ and for all $x \in \mathcal{B}(x^*, \delta)$: $\phi(x_2) > \phi(x)$.

Figure 2 shows possible points $\mathbf{x}_0$, $\mathbf{x}_1$, $\mathbf{x}^*$ and $\mathbf{x}_2$.

To prove this lemma, note that it is enough to prove that for all $x_0 \in \mathbb{R}$, there exist $\gamma > 0$ and $\dot{x}_0 \in \mathbb{R}$ such that $\|\dot{x}^*\| \geq 0$ and there does not exist $\dot{x}_2 \in \mathbb{R}$ (so $\|\dot{x}_2\|^2 < 0$).

Case 1) $x_1 = x_0$

In this case, it is enough to prove that for all $\dot{x}_1$, there exist $\gamma > 0$ such that $\|\dot{x}^*\|^2 \geq 0$ and $\|\dot{x}_2\|^2 < 0$.

From (13):

[1] Note that x_1 can be equal to x_0.

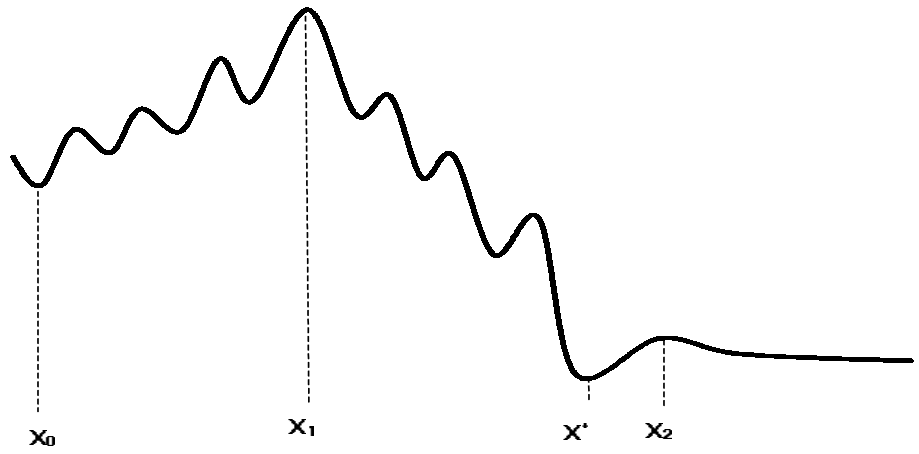

Figure 2: Representation of the different points of the scalar objective function

$$\begin{aligned} & \|\dot{x}_2\|^2 - \|\dot{x}_1\|^2 + 2[\phi(x_2) - \phi(x_1)] = -2\gamma\|x_2 - x_1\| \\ \Rightarrow \quad & \|\dot{x}_2\|^2 = \|\dot{x}_1\|^2 - 2[\phi(x_2) - \phi(x_1)] - 2\gamma\|x_2 - x_1\| < 0 \\ \Rightarrow \quad & \gamma > \frac{\|\dot{x}_1\|^2 - 2[\phi(x_2) - \phi(x_1)]}{2\|x_2 - x_1\|} \end{aligned} \tag{15}$$

and

$$\begin{aligned} & \|\dot{x}^*\|^2 - \|\dot{x}_1\|^2 + 2[\phi(x^*) - \phi(x_1)] = -2\gamma\|x^* - x_1\| \\ \Rightarrow \quad & \|\dot{x}^*\|^2 = \|\dot{x}_1\|^2 + 2[\phi(x_1) - \phi(x^*)] - 2\gamma\|x^* - x_1\| \geq 0 \\ \Rightarrow \quad & \gamma \leq \frac{\|\dot{x}_1\|^2 + 2[\phi(x_1) - \phi(x^*)]}{2\|x_1 - x^*\|} \end{aligned} \tag{16}$$

From (15) and (16):

$$\frac{\|\dot{x}_1\|^2 - 2[\phi(x_2) - \phi(x_1)]}{2\|x_2 - x_1\|} < \gamma \leq \frac{\|\dot{x}_1\|^2 + 2[\phi(x_1) - \phi(x^*)]}{2\|x_1 - x^*\|} \tag{17}$$

Note that, by the hypotheses b) and c), the last term in (17) is always greater than the first one, hence, always there exists $\gamma > 0$ that satisfies (17). Note also that the first term can be negative, and in this case any non-negative value of the parameter γ satisfies the first inequality.

Case 2) $x_1 \in (x_0, x^*)$

In this case (in addition with the case 1), it is enough to prove that for all x_0 and $\gamma > 0$, there exists $\dot{x}_0$ such that $\dot{x}_1 > \epsilon$ for some positive constant ϵ arbitrarily small.

From (13):

$$\begin{aligned} & \|\dot{x}_1\|^2 - \|\dot{x}_0\|^2 + 2[\phi(x_1) - \phi(x_0)] + 2\gamma\|x_1 - x_0\| = 0 \\ \Rightarrow \quad & \|\dot{x}_1\|^2 = \|\dot{x}_0\|^2 - 2[\phi(x_1) - \phi(x_0)] - 2\gamma\|x_1 - x_0\| > \epsilon^2 \\ \Rightarrow \quad & \|\dot{x}_0\| > [\epsilon^2 + 2[\phi(x_1) - \phi(x_0)] + 2\gamma\|x_1 - x_0\|]^{\frac{1}{2}} \end{aligned} \tag{18}$$

From (17) and (18), for all x_0, there exist $\gamma > 0$ and $\dot{x}_0$ such that $\|\dot{x}_1\| > \epsilon$ for some ϵ arbitrarily small, $\|\dot{x}^*\|^2 \geq 0$ and there does not exist $\|\dot{x}_2\| \in \mathbb{R}$. Therefore, trajectories generated by (14) converge to x^*.

In the sequel, new algorithms as well as the HBF, DIN and CG algorithms will be derived in a unified manner using the CLF method and compared.

3 Continuous time second order algorithms

In order to carry out the design of second order algorithms, we first describe the corresponding control system illustrated in Figure 1. The reference variable is the lower bound for minimum value of the objective function, denoted ϕ_{min}. This lower bound is compared with the current value of the function, generating a residual or error function $r(t)$. The residue is the input to a controller which generates the input, $\mathbf{u}(t)$ to the plant, which, in turn, has output equal to its state $\mathbf{x}(t)$. With an appropriate choice of controller and plant, the state variable describes a trajectory from an initial point to the global minimum of the objective function, i.e. the residue is minimized.

The update laws of the control variable $\mathbf{u}$ and the state variable $\mathbf{x}$ are designed using control Liapunov functions.

Choosing (4) as the Liapunov function candidate, equation (5) is its corresponding time derivative. In the sequence, several choices of plant ($\mathbf{x}$-dynamic) and controller ($\mathbf{u}$-dynamic) arise in different second order algorithms.

1) First choice

Plant $\quad \dot{\mathbf{x}} = \mathbf{u}$.

Substituting in (5) yields: $\dot{V} = \mathbf{u}^T(\dot{\mathbf{u}} + \nabla\phi(\mathbf{x}))$.

Controller $\quad \dot{\mathbf{u}} = -\kappa\,\mathrm{sgn}(\mathbf{u}) - \nabla\phi(\mathbf{x})$

where $\kappa > 0$.

In the sequel, an analysis of the stability of the continuous-time system formed by these choices of plant and controller and the convergence of their trajectories will be realized.

Substituting the equations of the plant and controller in (5):

$$\dot{V} = -\kappa\|\mathbf{u}\|_1 \leq 0 \tag{19}$$

Note that $\dot{V}$ is continuous but it is not continuously differentiable; it fails to be differentiable at $u_i = 0$ for all $i \in \{1, \cdots, n\}$. However, $\mathbf{u} = \mathbf{0}$, $\nabla\phi(\mathbf{x}) = \mathbf{0}$ is an equilibrium point of (19).

By the controller equation, which has the function sgn, this is a nonsmooth system. Thus, commonly used tools to ensure stability of nonlinear dynamical systems, e.g. Barbalat's lemma, cannot be applied here.

The overall system can be written as the following Filippov set valued map:

$$\dot{\mathbf{w}} = \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{u} \\ -\nabla\phi(\mathbf{x}) - \kappa\,\mathrm{sgn}(\mathbf{u}) \end{bmatrix} \in \mathcal{F}(\mathbf{w}(t)) \tag{20}$$

where $\mathbf{w}(t) := [\mathbf{x}(t)^T\ \mathbf{u}(t)^T]^T \in \mathbb{R}^{2n}$ and $\mathcal{F}(\mathbf{w}) : \mathbb{R}^{2n} \to \mathcal{B}(\mathbb{R}^{2n})$ is a set valued map which can be defined as

$$\mathcal{F}(\mathbf{w}) = \begin{cases} \left\{ \begin{bmatrix} \mathbf{u} \\ -\nabla\phi(\mathbf{x}) - \kappa\,\mathrm{sgn}(\mathbf{u}) \end{bmatrix} \right\} & \text{if } u_i \neq 0\, \forall i \in \{1, \cdots, n\} \\ \begin{bmatrix} \mathbf{u} \\ -\nabla_1\phi(\mathbf{x}) - \kappa\,\mathrm{sgn}(u_1) \\ \cdot \\ -\nabla_i\phi(\mathbf{x}) - \kappa\,\mathrm{co}\{-1\,,\,1\} \\ \cdot \\ -\nabla_n\phi(\mathbf{x}) - \kappa\,\mathrm{sgn}(u_n) \end{bmatrix} & \text{if } \exists i \in \{1, \cdots, n\} \,|\, u_i = 0 \end{cases} \tag{21}$$

If there exists $i \in \{1, \cdots, n\}$ such that $u_i = 0$, then, the $n+i^{\text{th}}$ component of the set valued map is $\mathcal{F}_{n+i}(\mathbf{w}) = -\nabla_i\phi(\mathbf{x}) - \kappa\,\mathrm{co}\{-1\,,\,1\}$, where co denotes the convex hull between the scalar values, which in this case is $\mathrm{co}\{-1\,,\,1\} = [-1\,,\,1]$, and $\nabla_i\phi(\mathbf{x})$ denotes the i^{th} component of the vector $\nabla\phi(\mathbf{x})$.

Defining $\mathbf{w}^* := [\mathbf{x}_*^T\ \mathbf{0}^T]^T$, being $\mathbf{x}_*$ any zero gradient point, i.e. any point such that $\nabla\phi(\mathbf{x}_*) = \mathbf{0}$, we observe that $\mathbf{0} \in \mathcal{F}(\mathbf{w}^*)$, hence $\mathbf{w}^*$ is an equilibrium point of (20).

Note that:

i) $\mathcal{F} : \mathbb{R}^{2n} \to \mathcal{B}(\mathbb{R}^{2n})$ is a piecewise continuous time-invariant set valued map, it is locally bounded and take nonempty, compact and convex values. For all $\mathbf{w} \in \mathbb{R}^{2n}$, the set valued map $\mathcal{F}(\mathbf{w}(t))$ is upper semi-continuous and measurable.

The Lie derivative of $V(\mathbf{w})$, defined as in (4), along the trajectories defined by $\dot{\mathbf{w}}$ is defined as (see [31, p. 63]):

$$\dot{V} \in \bar{\mathcal{L}}_{\mathcal{F}} V(\mathbf{w}) = \left\{ \frac{\partial V}{\partial \mathbf{w}}^T \mathbf{v} \,|\, \mathbf{v} \in \mathcal{F}(\mathbf{w}) \right\} = \frac{\partial V}{\partial \mathbf{x}}^T \dot{\mathbf{x}} + \frac{\partial V}{\partial \mathbf{u}}^T \dot{\mathbf{u}} = -\kappa \mathbf{u}^T \mathrm{sgn}(\mathbf{u})$$

which implies

$$\begin{aligned} \bar{\mathcal{L}}_{\mathcal{F}} V &= -\kappa \left[\textstyle\sum_{\forall i | u_i \neq 0} |u_i| + \sum_{\forall j | u_j = 0} u_j \mathrm{co}\{-1, 1\} \right] \\ &= -\kappa \textstyle\sum_{\forall i | u_i \neq 0} |u_i| < 0 \quad \forall \mathbf{u} \neq \mathbf{0} \end{aligned}$$

Hence, $V(\mathbf{x}(t), \mathbf{u}(t))$ is nonincreasing and $\mathbf{u}(t)$ and $\phi(\mathbf{x}(t))$ are limited.

The following conditions are satisfied:

ii) $V(\mathbf{x}, \mathbf{u})$ is locally Lipschitz and regular on $\mathbb{R}^{2n}$.

iii) $V(\mathbf{x}, \mathbf{u}) = 0$ for all $\mathbf{x} = \mathbf{x}^*$, $\mathbf{u} = \mathbf{0}$ and $V(\mathbf{x}, \mathbf{u}) > 0$ for all $\mathbf{x} \neq \mathbf{x}^*$ or $\mathbf{u} \neq \mathbf{0}$.

iv) $\max \bar{\mathcal{L}}_{\mathcal{F}} V(\mathbf{w}(t)) \leq 0$ for all $\mathbf{w} \neq \mathbf{w}^*$. $\bar{\mathcal{L}}_{\mathcal{F}} V(\mathbf{w}^*) = 0$.

Then, by i), ii), iii), and iv), $\mathbf{w}^*$ is a strongly stable equilibrium of (20) (see [31, theorem 1]).

Note that $\{\mathbf{w}^*\}$ is an invariant set for the system (20). For condition iv), all the trajectories $\mathbf{w}(t) : [0\,,\,\infty) \to \mathbb{R}^{2n}$ converge to the largest weakly invariant set contained in

$$\{\mathbf{w} \in \mathbb{R}^{2n} \,|\, 0 \in \bar{\mathcal{L}}_{\mathcal{F}} V(\mathbf{w})\}$$

and this set consists in all the points $\mathbf{w}^*$ (see [31, theorem 2]).

Hence, $\dot{V}$ tends to zero, and therefore $\dot{\mathbf{u}}$ and $\mathbf{u}$ tend to zero too. By the controller equation, $\nabla\phi(\mathbf{x})$ tends to zero too.

This ensures that the trajectory tends to a point where the gradient $\nabla\phi(\mathbf{x})$ becomes zero, which is merely a necessary condition for a global minimum, but far from being sufficient. In fact, as in the case of the HBF mechanical analogy mentioned above, a suitable choice of the parameter κ of the nonlinear damping term may make it possible for trajectories to traverse basins of attraction of local minima and converge to the global minimum, although this is not guaranteed.

Observe that the controller above is of the variable structure type since it contains a signum function sgn, and the typical behavior of the trajectories in these systems is the sliding mode. Thus, the control variable $\mathbf{u}(t)$ describes a sliding mode trajectory, not the output variable $\mathbf{x}(t)$.

Given the choices of the plant and the controller, the resulting second order ODE neural network model is:

$$\ddot{\mathbf{x}} + \kappa\text{sgn}(\dot{\mathbf{x}}) + \nabla\phi(\mathbf{x}) = \mathbf{0} \tag{22}$$

and will be denoted MI1 (minimization algorithm 1).

2) Second choice

Plant $\quad \dot{\mathbf{x}} = \mathbf{u}$.

Substituting in (5): $\dot{V} = \mathbf{u}^T(\dot{\mathbf{u}} + \nabla\phi(\mathbf{x}))$.

Controller $\quad \dot{\mathbf{u}} = -\kappa(\nabla^2\phi(\mathbf{x}))^2\mathbf{u} - \nabla\phi(\mathbf{x})$

where $\kappa > 0$ and $\nabla^2\phi(\mathbf{x})$ is the Hessian matrix of the function. Substituting in (5):

$$\dot{V} = -\kappa\mathbf{u}^T(\nabla^2\phi(\mathbf{x}))^2\mathbf{u} \leq 0 \tag{23}$$

which implies that $\mathbf{u}$ and $\phi(\mathbf{x})$ are limited. Since $\dot{V}$ is uniformly continuous, by Barbalat's lemma (see [14, p. 123]) $\dot{V}$ tends to zero when $t \to \infty$. Note that this may implies $\mathbf{u} = \mathbf{0}$, or a point where $\nabla^2\phi(\mathbf{x}) = 0$, or $\mathbf{u} \in \mathcal{N}(\nabla^2\phi(\mathbf{x}))$, the null-space of the Hessian matrix. Thus convergence is only local.

Similarly to the case MI1, this algorithm generates trajectories that can stop at any zero gradient point, not necessarily at the global minimum, and the possibility to pass through undesired local minima depends on the choice of the parameter κ.

By the equations of the plant and the controller, the resulting second order ODE neural network model is:

$$\ddot{\mathbf{x}} + \kappa(\nabla^2\phi(\mathbf{x}))^2\dot{\mathbf{x}} + \nabla\phi(\mathbf{x}) = \mathbf{0} \tag{24}$$

and will be denoted MI2.

3) Third choice

Plant $\quad \dot{\mathbf{x}} = \nabla^{-2}\phi(\mathbf{x})\mathbf{u}$.

where $\nabla^{-2}\phi(\mathbf{x})$ denotes the inverse of the Hessian matrix. Substituting in (5): $\dot{V} = \mathbf{u}^T(\dot{\mathbf{u}} + \nabla^{-2}\phi(\mathbf{x})\nabla\phi(\mathbf{x}))$.

Controller $\quad \dot{\mathbf{u}} = -\kappa\mathbf{u} - \nabla^{-2}\phi(\mathbf{x})\nabla\phi(\mathbf{x})$

where $\kappa > 0$. Substituting in (5):

$$\dot{V} = -\kappa\mathbf{u}^T\mathbf{u} \tag{25}$$

Since $\dot{V}$ is uniformly continuous, by Barbalat's lemma, $\dot{V}$ tends to zero when t tends to infinity, and hence $\mathbf{u}$ and $\dot{\mathbf{u}}$ both tend to zero too, implying, from the controller equation, that $\nabla^{-2}\phi(\mathbf{x})\nabla\phi(\mathbf{x})$ tends to zero too. Note that this implies that the trajectories can stop not only at a zero gradient point, but also at a point such that $\nabla\phi(\mathbf{x}) \in \mathcal{N}(\nabla^{-2}\phi(\mathbf{x}))$, the null-space of the inverse of the Hessian matrix, and thus convergence is only local. Moreover, since the inverse of the Hessian matrix occurs in the definition of both plant and controller, this algorithm is not defined at the points where the Hessian is singular, that is, where $\det(\nabla^2\phi(\mathbf{x})) = 0$.

From the choices of the plant and the controller, the resulting second order ODE neural network model is:

$$\ddot{\nabla}\phi(\mathbf{x}) + \kappa\dot{\nabla}\phi(\mathbf{x}) + \nabla^{-2}\phi(\mathbf{x})\nabla\phi(\mathbf{x}) = \mathbf{0} \tag{26}$$

This algorithm will be denoted MI3.

4) Fourth choice

Plant $\quad \dot{\mathbf{x}} = \nabla^2\phi(\mathbf{x})\mathbf{u}$.

Substituting in (5): $\dot{V} = \mathbf{u}^T(\dot{\mathbf{u}} + \nabla^2\phi(\mathbf{x})\nabla\phi(\mathbf{x}))$.

Controller $\quad \dot{\mathbf{u}} = -\kappa\mathbf{u} - \nabla^2\phi(\mathbf{x})\nabla\phi(\mathbf{x})$

where $\kappa > 0$. Substituting in (5):

$$\dot{V} = -\kappa\mathbf{u}^T\mathbf{u} \tag{27}$$

An analysis similar to those made above, allows us to conclude that the trajectories can stop not only at zero gradient points, but also at points $\mathbf{x}$ such that $\nabla\phi(\mathbf{x}) \in \mathcal{N}(\nabla^2\phi(\mathbf{x}))$, the null-space of the Hessian matrix. Thus convergence is only local.

The resulting second order ODE neural network model is:

$$\ddot{\mathbf{x}} + (\kappa I - \dot{\nabla}^2\phi(\mathbf{x})\nabla^{-2}\phi(\mathbf{x}))\dot{\mathbf{x}} + (\nabla^2\phi(\mathbf{x}))^2\nabla\phi(\mathbf{x}) = \mathbf{0} \tag{28}$$

This algorithm will be denoted as MI4.

5) Fifth choice

It is possible to design other algorithms by changing the candidate Liapunov function. For example, choosing

$$V(\mathbf{x}, \mathbf{u}) = (a + b)(\phi(\mathbf{x}) - \phi_{\min}) + \frac{\|\nabla\phi(\mathbf{x}) + \mathbf{u}\|^2}{2} \geq 0 \tag{29}$$

where a and b are scalar parameters such that $a + b > 0$.

Note that this candidate Liapunov function is equal to zero only at a point $\mathbf{x} = \mathbf{x}^*$ such that $\phi(\mathbf{x}^*) = \phi_{\min}$, point where $\nabla\phi(\mathbf{x}^*) = \mathbf{0}$, and for $\mathbf{u} = \mathbf{0}$.

The time derivative of (29) is

$$\dot{V} = (a + b)\nabla^T\phi(\mathbf{x})\dot{\mathbf{x}} + (\nabla\phi(\mathbf{x}) + \mathbf{u})^T(\nabla^2\phi(\mathbf{x})\dot{\mathbf{x}} + \dot{\mathbf{u}}) \tag{30}$$

The plant and the controller are chosen as follows:

Plant $\quad \dot{\mathbf{x}} = \nabla^{-2}\phi(\mathbf{x})\mathbf{u}$

where $\nabla^{-2}\phi(\mathbf{x})$ denotes the inverse of the Hessian matrix. Substituting in (30):
$\dot{V} = (a + b)\nabla^T\phi(\mathbf{x})\nabla^{-2}\phi(\mathbf{x})\mathbf{u} + (\nabla\phi(\mathbf{x}) + \mathbf{u})^T(\mathbf{u} + \dot{\mathbf{u}})$

Controller $\dot{\mathbf{u}} = -\gamma\nabla\phi(\mathbf{x}) - (1-\gamma)\mathbf{u} - \nabla^{-2}\phi(\mathbf{x})(a\mathbf{u} + b\nabla\phi(\mathbf{x}))$
where $\gamma \in \mathbb{R}$. Substituting in (30):

$$\begin{aligned}
\dot{V} &= (a+b)\nabla^T\phi(\mathbf{x})\nabla^{-2}\phi(\mathbf{x})\mathbf{u} + \\
&\quad (\nabla\phi(\mathbf{x}) + \mathbf{u})^T(\mathbf{u} - \gamma\nabla\phi(\mathbf{x}) - (1-\gamma)\mathbf{u} - \nabla^{-2}\phi(\mathbf{x})(a\mathbf{u} + b\nabla\phi(\mathbf{x}))) \\
&= (a+b)\nabla^T\phi(\mathbf{x})\nabla^{-2}\phi(\mathbf{x})\mathbf{u} + \nabla^T\phi(\mathbf{x})\mathbf{u} - \gamma\nabla^T\phi(\mathbf{x})\nabla\phi(\mathbf{x}) \\
&\quad + \mathbf{u}^T\mathbf{u} - \gamma\mathbf{u}^T\nabla\phi(\mathbf{x}) - (1-\gamma)\nabla^T\phi(\mathbf{x})\mathbf{u} - (1-\gamma)\mathbf{u}^T\mathbf{u} \\
&\quad - a\nabla^T\phi(\mathbf{x})\nabla^{-2}\phi(\mathbf{x})\mathbf{u} - b\nabla^T\phi(\mathbf{x})\nabla^{-2}\phi(\mathbf{x})\nabla\phi(\mathbf{x}) \\
&\quad - a\mathbf{u}^T\nabla^{-2}\phi(\mathbf{x})\mathbf{u} - b\mathbf{u}^T\nabla^{-2}\phi(\mathbf{x})\nabla\phi(\mathbf{x}) \\
&= -\nabla^T\phi(\mathbf{x})(\gamma I + b\nabla^{-2}\phi(\mathbf{x}))\nabla\phi(\mathbf{x}) - \mathbf{u}^T(-\gamma I + a\nabla^{-2}\phi(\mathbf{x}))\mathbf{u}
\end{aligned}$$

where the fact that the Hessian matrix of a scalar function is symmetric has been used.

Note that if a, b and γ were chosen in such a way that $\gamma I + b\nabla^{-2}\phi(\mathbf{x}) \succ 0$ and $-\gamma I + a\nabla^{-2}\phi(\mathbf{x}) \succ 0$ in a region, then $\dot{V} \leq 0$ and since this is uniformly continuous, by Barbalat's lemma it tends to zero inside this region, thus convergence is only local. If $\dot{V}$ tends to zero, then $\mathbf{u}$, $\dot{\mathbf{u}}$ and $\nabla\phi(\mathbf{x})$ tend to zero too, but the trajectories can stop at any zero gradient point.

This algorithm can also be represented as a second order ODE:

$$\ddot{\nabla}\phi(\mathbf{x}) + (I - \gamma I + a\nabla^{-2}\phi(\mathbf{x}))\dot{\nabla}\phi(\mathbf{x}) + (\gamma I + b\nabla^{-2}\phi(\mathbf{x}))\nabla\phi(\mathbf{x}) = \mathbf{0} \qquad (31)$$

This algorithm will be denoted as MI5.

6) DIN

DIN algorithm (8), can also be viewed as a dynamical closed loop control system, with the following choices for the plant and the controller:

Plant $\dot{\mathbf{x}} = \mathbf{u}$.
Controller $\dot{\mathbf{u}} = -\nabla\phi(\mathbf{x}) - a\mathbf{u} - b\nabla^2\phi(\mathbf{x})\mathbf{u}$.

To make an analysis of the trajectories generated by this algorithm, we choose a candidate Liapunov function similar to the energy function used in [13]:

$$V(\mathbf{x}, \mathbf{u}) = (ab+1)(\phi(\mathbf{x}) - \phi_{\min}) + \frac{\|\mathbf{u} + b\nabla\phi(\mathbf{x})\|^2}{2} \geq 0 \qquad (32)$$

Note that (32) is equal to zero only at $\mathbf{x} = \mathbf{x}^*$ and $\mathbf{u} = \mathbf{0}$. The time derivative of (32), applying the plant and the controller equations, that is, along the trajectories generated by the closed loop system, is:

$$\begin{aligned}
\dot{V} &= (ab+1)\nabla^T\phi(\mathbf{x})\mathbf{u} + (\mathbf{u} + b\nabla\phi(\mathbf{x}))^T \\
&\quad (-\nabla\phi(\mathbf{x}) - a\mathbf{u} - b\nabla^2\phi(\mathbf{x})\mathbf{u} + b\nabla^2\phi(\mathbf{x})\mathbf{u}) \\
&= ab\nabla^T\phi(\mathbf{x})\mathbf{u} + \nabla^T\phi(\mathbf{x})\mathbf{u} - \mathbf{u}^T\nabla\phi(\mathbf{x}) - a\|\mathbf{u}\|^2 \\
&\quad - b\|\nabla\phi(\mathbf{x})\|^2 - ab\nabla^T\phi(\mathbf{x})\mathbf{u} \\
&= -a\|\mathbf{u}\|^2 - b\|\nabla\phi(\mathbf{x})\|^2
\end{aligned}$$

By Barbalat's lemma, since $\dot{V}$ is uniformly continuous, $\dot{V}$ tends to zero and therefore $\mathbf{u}$ and $\nabla\phi(\mathbf{x})$ also tend to zero, but trajectories can stop at any zero gradient point, depending on the choice of the parameters a and b to be able to bypass local minima.

Making the change of variables proposed in the plant and the controller choices, that is, $\dot{\mathbf{x}} = \mathbf{u}$ and $\ddot{\mathbf{x}} = \dot{\mathbf{u}} = -\nabla\phi(\mathbf{x}) - a\dot{\mathbf{x}} - b\nabla^2\phi(\mathbf{x})\dot{\mathbf{x}}$, we easily arrive

to (8).

7) HBF

The algorithm HBF (6), can also be represented as a dynamical closed loop control system. Choosing (4) as candidate Liapunov function, and with the following choices for plant and controller:

Plant $\quad \dot{\mathbf{x}} = \mathbf{u}$
and substituting in (5): $\dot{V} = \mathbf{u}^T(\nabla\phi(\mathbf{x}) + \dot{\mathbf{u}})$.
Controller $\quad \dot{\mathbf{u}} = -\gamma\mathbf{u} - \nabla\phi(\mathbf{x})$
where $\gamma > 0$. Substituting in (5):

$$\dot{V} = -\gamma\mathbf{u}^T\mathbf{u} \leq 0$$

Making the same analysis as above, by Barbalat's lemma $\mathbf{u}$ and $\dot{\mathbf{u}}$ tend to zero, and from the controller equation, we conclude that the trajectories can stop at any point $\mathbf{x}$ such that $\nabla\phi(\mathbf{x}) = \mathbf{0}$. Trajectories could pass through local minima and converge to the global minimum by suitable choice of the parameter γ.

Making the change of variables realized in the plant definition $\dot{\mathbf{x}} = \mathbf{u}$, and applying the controller equation $\dot{\mathbf{u}} = \ddot{\mathbf{x}} = -\nabla\phi(\mathbf{x}) - \gamma\dot{\mathbf{x}}$, we easily arrive at (6).

Here, we extend the definition of the HBF algorithm (6) to the following second order ODE:

$$\ddot{\mathbf{x}} + \gamma(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} + \mu(\mathbf{x}, \dot{\mathbf{x}})\nabla\phi(\mathbf{x}) = \mathbf{0} \tag{33}$$

where $\gamma(\mathbf{x}, \dot{\mathbf{x}})$ and $\mu(\mathbf{x}, \dot{\mathbf{x}}) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ are scalar-valued functions of the state variables $\mathbf{x}$ and $\dot{\mathbf{x}}$. Algorithm (33) will be denoted as *generalized heavy ball with friction* (gHBF). Note that the choice $\mu = 1$ leads to the HBF ODE (6).

The system (33) can also be interpreted as a closed loop control system, by the following choices of plant and controller.

Plant $\dot{\mathbf{x}} = \mathbf{u}$.
Controller $\dot{\mathbf{u}} = -\gamma(\mathbf{x}, \mathbf{u})\mathbf{u} - \mu(\mathbf{x}, \mathbf{u})\nabla\phi(\mathbf{x})$.

Choosing (4) as candidate Liapunov function, and substituting the equations of the plant and the controller in (5), yields:

$$\dot{V} = (1 - \mu(\mathbf{x}, \mathbf{u}))\nabla^T\phi(\mathbf{x})\mathbf{u} - \gamma(\mathbf{x}, \mathbf{u})\mathbf{u}^T\mathbf{u} \tag{34}$$

The choices of different expressions for $\mu(\mathbf{x}, \mathbf{u})$ and $\gamma(\mathbf{x}, \mathbf{u})$ that make $\dot{V}$ negative in (34) lead to different algorithms. In the sequel, some choices of these functions will be presented.

7a) First choice
$\gamma > 0$ as a constant value.

$$\mu(\mathbf{x}, \mathbf{u}) = \begin{cases} 1 - \gamma\frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi(\mathbf{x})\mathbf{u}} + a & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} > \epsilon \\ 1 - \gamma\frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi(\mathbf{x})\mathbf{u}} - b & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} < -\epsilon \\ 1 & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| \leq \epsilon \end{cases} \tag{35}$$

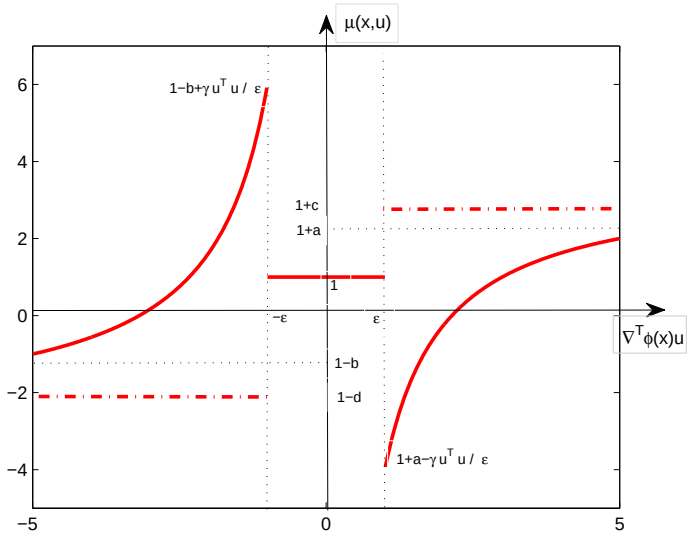

Figure 3: Possible values of the parameter $\mu(\mathbf{x},\mathbf{u})$ versus $\nabla^T\phi(\mathbf{x})\mathbf{u}$

where the parameters $a > 0$, $b > 0$, and the parameter ϵ is a positive scalar value small enough. The parameters a and b can be variables. For example, the choices

$$\begin{array}{ll} a = \gamma\frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi(\mathbf{x})\mathbf{u}} + c, \quad c > 0 & \text{leads to the parameter } \mu = 1 + c \text{ if } \nabla^T\phi(\mathbf{x})\mathbf{u} > \epsilon \\ b = -\gamma\frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi(\mathbf{x})\mathbf{u}} + d, \quad d > 0 & \text{leads to the parameter } \mu = 1 - d \text{ if } \nabla^T\phi(\mathbf{x})\mathbf{u} < -\epsilon \end{array}$$

Note that for $|\nabla^T\phi(\mathbf{x})\mathbf{u}| \leq \epsilon$, the algorithm (33) becomes equal to (6).

Figure 3 shows possible values of the parameter $\mu(\mathbf{x},\mathbf{u})$ versus $\nabla^T\phi(\mathbf{x})\mathbf{u}$.

With the parameter $\mu(\mathbf{x},\mathbf{u})$ chosen as in (35), the system (33) becomes nonsmooth, and it can be represented by the following Filippov set valued map:

$$\dot{\mathbf{w}} = \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{u} \\ -\gamma\mathbf{u} - \mu(\mathbf{x},\mathbf{u})\nabla\phi(\mathbf{x}) \end{bmatrix} \in \mathcal{F}(\mathbf{w}(t)) \tag{36}$$

where $\mathbf{w}(t) := [\mathbf{x}(t)^T\ \mathbf{u}(t)^T]^T$ and $\mathcal{F}(\mathbf{w})$ is:

$$\mathcal{F}(\mathbf{w}) = \begin{cases} [\mathbf{u}^T \ -\gamma\mathbf{u}^T - \left(1 - \gamma\frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi(\mathbf{x})\mathbf{u}} + a\right)\nabla^T\phi(\mathbf{x})]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} > \epsilon \\ [\mathbf{u}^T \ -\gamma\mathbf{u}^T - \left(1 - \gamma\frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi(\mathbf{x})\mathbf{u}} - b\right)\nabla^T\phi(\mathbf{x})]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} < -\epsilon \\ [\mathbf{u}^T \ -\gamma\mathbf{u}^T - \nabla^T\phi(\mathbf{x})]^T & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| < \epsilon \\ [\mathbf{u}^T \ -\gamma\mathbf{u}^T - \text{co}\left\{1\,,\, 1 - \gamma\frac{\mathbf{u}^T\mathbf{u}}{\epsilon} + a\right\}\nabla^T\phi(\mathbf{x})]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = \epsilon \\ [\mathbf{u}^T \ -\gamma\mathbf{u}^T - \text{co}\left\{1\,,\, 1 + \gamma\frac{\mathbf{u}^T\mathbf{u}}{\epsilon} - b\right\}\nabla^T\phi(\mathbf{x})]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = -\epsilon \end{cases}$$

Defining $\mathbf{w}^* := [\mathbf{x}_*^T\ \mathbf{0}^T]^T$, being $\mathbf{x}_*$ any zero gradient point, $\mathbf{0} \in \mathcal{F}(\mathbf{w}^*)$ and hence $\mathbf{w}^*$ is an equilibrium point of (36).

Note that:

i) $\mathcal{F} : \mathbb{R}^{2n} \to \mathcal{B}(\mathbb{R}^{2n})$ is a piecewise continuous set valued map, it is locally bounded and take nonempty, compact and convex values. For all $\mathbf{w} \in \mathbb{R}^{2n}$, the set valued map $\mathcal{F}(\mathbf{w}(t))$ is upper semi-continuous and measurable.

The Lie derivative of the candidate Liapunov function (4), $V(\mathbf{w})$ along the trajectories defined by $\dot{\mathbf{w}}$ in (36) is defined as (see [31, p. 63]):

$$\begin{aligned}\dot{V} &= (1-\mu(\mathbf{x},\mathbf{u}))\nabla^T\phi(\mathbf{x})\mathbf{u} - \gamma\mathbf{u}^T\mathbf{u} \in \bar{\mathcal{L}}_{\mathcal{F}}V(\mathbf{w}) = \left\{ \tfrac{\partial V}{\partial \mathbf{w}}^T \mathbf{v} \,|\, \mathbf{v} \in \mathcal{F}(\mathbf{w}) \right\} \\ &= \begin{cases} -a\nabla^T\phi(\mathbf{x})\mathbf{u} < 0 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} > \epsilon \\ b\nabla^T\phi(\mathbf{x})\mathbf{u} < 0 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} < -\epsilon \\ -\gamma\mathbf{u}^T\mathbf{u} \le 0 & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| < \epsilon \\ \text{co}\{-\gamma\mathbf{u}^T\mathbf{u}\,,\, -a\epsilon\} < 0 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = \epsilon \\ \text{co}\{-\gamma\mathbf{u}^T\mathbf{u}\,,\, -b\epsilon\} < 0 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = -\epsilon \end{cases}\end{aligned}$$

Hence $\dot{V}(\mathbf{w}) < 0$ for all $\mathbf{u} \neq \mathbf{0}$. Hence $\mathbf{u}$ and $\phi(\mathbf{x})$ are limited.

The following conditions are satisfied:

ii) $V(\mathbf{x},\mathbf{u})$ is locally Lipschitz and regular on $\mathbb{R}^{2n}$.

iii) $V(\mathbf{x},\mathbf{u}) = 0$ for all $\mathbf{x} = \mathbf{x}^*$, $\mathbf{u} = \mathbf{0}$ and $V(\mathbf{x},\mathbf{u}) > 0$ for all $\mathbf{x} \neq \mathbf{x}^*$ or $\mathbf{u} \neq \mathbf{0}$.

iv) $\max \bar{\mathcal{L}}_{\mathcal{F}}V(\mathbf{w}(t)) \le 0$ for all $\mathbf{w} \neq \mathbf{w}^*$, $\bar{\mathcal{L}}_{\mathcal{F}}V(\mathbf{w}^*) = 0$.

Then, by i), ii), iii), and iv), $\mathbf{w}^*$ is a strongly stable equilibrium of (36) (see [31, theorem 1]) and the trajectories generated by (36) converge to the largest invariant set $\{\mathbf{w}^*\} = \{[\mathbf{x}^T\ \mathbf{0}^T]^T \mid \nabla\phi(\mathbf{x}) = \mathbf{0}\}$ (see [31, theorem 2]).

Hence, $\dot{V}$ tends to zero, and therefore $\dot{\mathbf{u}}$ and $\mathbf{u}$ tend to zero too. By the controller equation, $\nabla\phi(\mathbf{x})$ tends to zero too. Here again, depends on the choice of the parameters γ, a and b that trajectories achieve pass through local minima to converge to the global minimum of the function.

This algorithm will be denoted as HBF1.

7b) Second choice

$\gamma > 0$ as a constant value.

$\mu(\mathbf{x},\mathbf{u}) = 1 + \kappa\,\text{sgn}(\nabla^T\phi(\mathbf{x})\mathbf{u})$

where $\kappa > 0$. Here again, this is a nonsmooth system, which can be represented by the following set valued map:

$$\dot{\mathbf{w}} = \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{u} \\ -\gamma\mathbf{u} - (1 + \kappa\,\text{sgn}(\nabla^T\phi(\mathbf{x})\mathbf{u}))\nabla\phi(\mathbf{x}) \end{bmatrix} \in \mathcal{F}(\mathbf{w}(t)) \qquad (37)$$

where $\mathbf{w}(t) := [\mathbf{x}(t)^T\ \mathbf{u}(t)^T]^T$ and the set valued map $\mathcal{F}(\mathbf{w})$ can be represented as:

$$\mathcal{F}(\mathbf{w}) = \begin{cases} [\mathbf{u}^T & -\gamma\mathbf{u}^T - (1+\kappa)\nabla^T\phi(\mathbf{x})]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} > 0 \\ [\mathbf{u}^T & -\gamma\mathbf{u}^T - (1-\kappa)\nabla^T\phi(\mathbf{x})]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} < 0 \\ [\mathbf{u}^T & -\gamma\mathbf{u}^T - (1+\text{co}\{-\kappa\,;\,+\kappa\})\nabla^T\phi(\mathbf{x})]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = 0 \end{cases} \qquad (38)$$

where $\text{co}\{-\kappa\,,\,+\kappa\} = [-\kappa\,,\,+\kappa]$.

The point $\mathbf{w}^* := [\mathbf{x}_*^T\ \mathbf{0}^T]^T$, being $\mathbf{x}_*$ any zero gradient point, is an equilibrium point of (37).

The Lie derivative of the candidate Liapunov function (4) along the trajectories determined by $\dot{\mathbf{w}}(t)$ is:

$$\begin{aligned}\dot{V} \quad &\in \bar{\mathcal{L}}_{\mathcal{F}} V(\mathbf{w}) = \{\nabla^T V(\mathbf{w})\dot{\mathbf{v}} \,|\, \mathbf{v} \in \mathcal{F}(\mathbf{w})\} = \\ &= \begin{cases} -\kappa|\nabla^T\phi(\mathbf{x})\mathbf{u}| - \gamma\mathbf{u}^T\mathbf{u} < 0 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} \neq 0 \\ -\gamma\mathbf{u}^T\mathbf{u} - [-\kappa\,,\,+\kappa]\nabla^T\phi(\mathbf{x})\mathbf{u} = -\gamma\mathbf{u}^T\mathbf{u} \leq 0 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = 0 \end{cases}\end{aligned}$$

An analysis completely similar to that made in the former subsection allows to conclude that the trajectories determined by (37) converge to a point $\mathbf{w}^*$.

This algorithm will be denoted as HBF2.

7c) Third choice

$\gamma > 0$ as a constant value.

$$\mu(\mathbf{x},\mathbf{u}) = \begin{cases} 1 + \kappa\frac{1}{\nabla^T\phi(\mathbf{x})\mathbf{u}} & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| > \epsilon \\ 1 & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| \leq \epsilon \end{cases} \tag{39}$$

where $\kappa > 0$ and $\epsilon > 0$ is a constant value small enough. Note that, for $|\nabla^T\phi(\mathbf{x})\mathbf{u}| \leq \epsilon$, this algorithms becomes equal to (6). Here again, this is a nonsmooth system which can be represented by the following set valued map:

$$\dot{\mathbf{w}} = \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{u} \\ -\gamma\mathbf{u} - \mu(\mathbf{x},\mathbf{u})\nabla\phi(\mathbf{x}) \end{bmatrix} \in \mathcal{F}(\mathbf{w}(t)) \tag{40}$$

where $\mathbf{w}(t) := [\mathbf{x}(t)^T\ \mathbf{u}(t)^T]^T$ and the set valued map $\mathcal{F}(\mathbf{w})$ can be represented as:

$$\mathcal{F}(\mathbf{w}) = \begin{cases} [\mathbf{u}^T \ -\gamma\mathbf{u}^T - \left(1 + \kappa\frac{1}{\nabla^T\phi(\mathbf{x})\mathbf{u}}\right)\nabla^T\phi(\mathbf{x})]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} > \epsilon \\ [\mathbf{u}^T \ -\gamma\mathbf{u}^T - \left(1 + \kappa\frac{1}{\nabla^T\phi(\mathbf{x})\mathbf{u}}\right)\nabla^T\phi(\mathbf{x})]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} < -\epsilon \\ [\mathbf{u}^T \ -\gamma\mathbf{u}^T - \nabla^T\phi(\mathbf{x})]^T & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| < \epsilon \\ [\mathbf{u}^T \ -\gamma\mathbf{u}^T - \text{co}\left\{1\,,\,1 + \frac{\kappa}{\epsilon}\right\}\nabla^T\phi(\mathbf{x})]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = \epsilon \\ [\mathbf{u}^T \ -\gamma\mathbf{u}^T - \text{co}\left\{1\,,\,1 - \frac{\kappa}{\epsilon}\right\}\nabla^T\phi(\mathbf{x})]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = -\epsilon \end{cases}$$

Note that $\mathbf{w}^* := [\mathbf{x}_*^T\ \mathbf{0}^T]^T$, being $\mathbf{x}_*$ any zero gradient point, is an equilibrium point of (40).

The Lie derivative of the candidate Liapunov function (4) along the trajectories determined by $\dot{\mathbf{w}}(t)$ is:

$$\begin{aligned}\dot{V} \quad &\in \bar{\mathcal{L}}_{\mathcal{F}} V = \{\nabla^T V(\mathbf{w})\mathbf{v} \mid \mathbf{v} \in \mathcal{F}(\mathbf{w})\} = \\ &= \begin{cases} -\kappa - \gamma\mathbf{u}^T\mathbf{u} < 0 & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| > \epsilon \\ -\gamma\mathbf{u}^T\mathbf{u} \leq 0 & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| < \epsilon \\ -\gamma\mathbf{u}^T\mathbf{u} + [-\kappa\,,\,0] < 0 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = \epsilon \\ -\gamma\mathbf{u}^T\mathbf{u} - [0\,,\,\kappa] < 0 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = -\epsilon \end{cases}\end{aligned}$$

An analysis completely similar to that made in the former subsection allows to conclude that the trajectories determined by (40) converge to a point $\mathbf{w}^*$.

This algorithm will be denoted as HBF3.

7d) Fourth choice

$\gamma > 0$ as a constant value.

$\mu(\mathbf{x},\mathbf{u}) = 1 + \kappa\nabla^T\phi(\mathbf{x})\mathbf{u}$
where $\kappa > 0$. Note that this is a smooth system, which can be represented by the following first order ODE:

$$\dot{\mathbf{w}}(t) = \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{u} \\ -\gamma\mathbf{u} - (1 + \kappa\nabla^T\phi(\mathbf{x})\mathbf{u})\nabla\phi(\mathbf{x}) \end{bmatrix} \tag{41}$$

where $\mathbf{w}(t) := [\mathbf{x}(t)^T\ \mathbf{u}(t)^T]^T$ and $\mathbf{w}^* := [\mathbf{x}_*^T\ \mathbf{0}^T]^T$, being $\mathbf{x}_*$ any zero gradient point, is an equilibrium point of (41).

The derivative of the candidate Liapunov function (4) along the trajectories determined by (41) is:

$$\dot{V}(\mathbf{w}) = \nabla^T V(\mathbf{w})\dot{\mathbf{w}} = -\kappa(\nabla^T\phi(\mathbf{x})\mathbf{u})^2 - \gamma\mathbf{u}^T\mathbf{u} \leq 0$$

By the Barbalat's lemma, since $\dot{V}$ is uniformly continuous, $\dot{V}$ tends to zero when t tends to infinity, so $\mathbf{u} \to 0$. By the controller equation $\nabla\phi(\mathbf{x})$ also tends to zero, and the system can stop at any zero gradient point.

This algorithm will be denoted as HBF4.

7e) Fifth choice

$$\begin{aligned} \gamma(\mathbf{x},\mathbf{u}) &= \begin{cases} \frac{b}{\mathbf{u}^T\mathbf{u}} & \text{if } \mathbf{u}^T\mathbf{u} > \epsilon \\ 1 & \text{if } \mathbf{u}^T\mathbf{u} \leq \epsilon \end{cases} \\ \mu(\mathbf{x},\mathbf{u}) &= \begin{cases} 1 + \frac{a}{\nabla^T\phi(\mathbf{x})\mathbf{u}} & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| > \epsilon \\ 1 & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| \leq \epsilon \end{cases} \end{aligned} \tag{42}$$

where $a > 0$, $b > 0$ and the parameter ϵ is a positive constant value small enough. This is also a nonsmooth system which can be represented by the following piecewise continuous set valued map:

$$\dot{\mathbf{w}}(t) = \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{u} \\ -\gamma(\mathbf{x},\mathbf{u})\mathbf{u} - \mu(\mathbf{x},\mathbf{u})\nabla\phi(\mathbf{x}) \end{bmatrix} \in \mathcal{F}(\mathbf{w}(t)) =$$

$$\begin{cases} [\mathbf{u}^T\ -\frac{b}{\mathbf{u}^T\mathbf{u}}\mathbf{u}^T - \left(1 + \frac{a}{\nabla^T\phi\mathbf{u}}\right)\nabla^T\phi(\mathbf{x})]^T & \text{if } \mathbf{u}^T\mathbf{u} > \epsilon \text{ and } |\nabla^T\phi(\mathbf{x})\mathbf{u}| > \epsilon \\ [\mathbf{u}^T\ -\mathbf{u}^T - \left(1 + \frac{a}{\nabla^T\phi\mathbf{u}}\right)\nabla^T\phi(\mathbf{x})]^T & \text{if } \mathbf{u}^T\mathbf{u} < \epsilon \text{ and } |\nabla^T\phi(\mathbf{x})\mathbf{u}| > \epsilon \\ [\mathbf{u}^T\ -\frac{b}{\mathbf{u}^T\mathbf{u}}\mathbf{u}^T - \nabla^T\phi(\mathbf{x})]^T & \text{if } \mathbf{u}^T\mathbf{u} > \epsilon \text{ and } |\nabla^T\phi(\mathbf{x})\mathbf{u}| < \epsilon \\ [\mathbf{u}^T\ -\mathbf{u}^T - \nabla^T\phi(\mathbf{x})]^T & \text{if } \mathbf{u}^T\mathbf{u} < \epsilon \text{ and } |\nabla^T\phi(\mathbf{x})\mathbf{u}| < \epsilon \\ [\mathbf{u}^T\ -\left[1\,,\,\frac{b}{\epsilon}\right]\mathbf{u}^T - \left(1 + \frac{a}{\nabla^T\phi\mathbf{u}}\right)\nabla^T\phi(\mathbf{x})]^T & \text{if } \mathbf{u}^T\mathbf{u} = \epsilon \text{ and } |\nabla^T\phi(\mathbf{x})\mathbf{u}| > \epsilon \\ [\mathbf{u}^T\ -\left[1\,,\,\frac{b}{\epsilon}\right]\mathbf{u}^T - \nabla^T\phi(\mathbf{x})]^T & \text{if } \mathbf{u}^T\mathbf{u} = \epsilon \text{ and } |\nabla^T\phi(\mathbf{x})\mathbf{u}| < \epsilon \\ [\mathbf{u}^T\ -\frac{b}{\mathbf{u}^T\mathbf{u}}\mathbf{u}^T - \left[1\,,\,1 + \frac{a}{\epsilon}\right]\nabla^T\phi(\mathbf{x})]^T & \text{if } \mathbf{u}^T\mathbf{u} > \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} = \epsilon \\ [\mathbf{u}^T\ -\frac{b}{\mathbf{u}^T\mathbf{u}}\mathbf{u}^T - \left[1\,,\,1 - \frac{a}{\epsilon}\right]\nabla^T\phi(\mathbf{x})]^T & \text{if } \mathbf{u}^T\mathbf{u} > \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} = -\epsilon \\ [\mathbf{u}^T\ -\mathbf{u}^T - \left[1\,,\,1 + \frac{a}{\epsilon}\right]\nabla^T\phi(\mathbf{x})]^T & \text{if } \mathbf{u}^T\mathbf{u} < \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} = \epsilon \\ [\mathbf{u}^T\ -\mathbf{u}^T - \left[1\,,\,1 - \frac{a}{\epsilon}\right]\nabla^T\phi(\mathbf{x})]^T & \text{if } \mathbf{u}^T\mathbf{u} < \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} = -\epsilon \\ [\mathbf{u}^T\ -\left[1\,,\,\frac{b}{\epsilon}\right]\mathbf{u}^T - \left[1\,,\,1 + \frac{a}{\epsilon}\right]\nabla^T\phi(\mathbf{x})]^T & \text{if } \mathbf{u}^T\mathbf{u} = \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} = \epsilon \\ [\mathbf{u}^T\ -\left[1\,,\,\frac{b}{\epsilon}\right]\mathbf{u}^T - \left[1\,,\,1 - \frac{a}{\epsilon}\right]\nabla^T\phi(\mathbf{x})]^T & \text{if } \mathbf{u}^T\mathbf{u} = \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} = -\epsilon \end{cases} \tag{43}$$

Note that $\mathbf{w}^* := [\mathbf{x}_*^T\ \mathbf{0}^T]^T$, being $\mathbf{x}_*$ any zero gradient point, is an equilibrium point of (43).

The Lie derivative of the candidate Liapunov function (4) along the trajectories determined by $\dot{\mathbf{w}}(t)$ is:

$$\dot{V} \in \bar{\mathcal{L}}_{\mathcal{F}} V = \{\nabla^T V(\mathbf{w})\mathbf{v} \mid \mathbf{v} \in \mathcal{F}(\mathbf{w})\} =$$
$$= \begin{cases} -b - a < 0 & \text{if } \mathbf{u}^T\mathbf{u} > \epsilon \text{ and } |\nabla^T\phi(\mathbf{x})\mathbf{u}| > \epsilon \\ -\mathbf{u}^T\mathbf{u} - a < 0 & \text{if } \mathbf{u}^T\mathbf{u} < \epsilon \text{ and } |\nabla^T\phi(\mathbf{x})\mathbf{u}| > \epsilon \\ -b < 0 & \text{if } \mathbf{u}^T\mathbf{u} > \epsilon \text{ and } |\nabla^T\phi(\mathbf{x})\mathbf{u}| < \epsilon \\ -\mathbf{u}^T\mathbf{u} \leq 0 & \text{if } \mathbf{u}^T\mathbf{u} < \epsilon \text{ and } |\nabla^T\phi(\mathbf{x})\mathbf{u}| < \epsilon \\ -[\epsilon\,,\, b] - a < 0 & \text{if } \mathbf{u}^T\mathbf{u} = \epsilon \text{ and } |\nabla^T\phi(\mathbf{x})\mathbf{u}| > \epsilon \\ -[\epsilon\,,\, b] < 0 & \text{if } \mathbf{u}^T\mathbf{u} = \epsilon \text{ and } |\nabla^T\phi(\mathbf{x})\mathbf{u}| < \epsilon \\ -b - [0\,,\, a] < 0 & \text{if } \mathbf{u}^T\mathbf{u} > \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} = \epsilon \\ -b - [0\,,\, a] < 0 & \text{if } \mathbf{u}^T\mathbf{u} > \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} = -\epsilon \\ -\mathbf{u}^T\mathbf{u} - [0\,,\, a] < 0 & \text{if } \mathbf{u}^T\mathbf{u} < \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} = \epsilon \\ -\mathbf{u}^T\mathbf{u} - [0\,,\, a] < 0 & \text{if } \mathbf{u}^T\mathbf{u} < \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} = -\epsilon \\ -[\epsilon\,,\, b] - [0\,,\, a] < 0 & \text{if } \mathbf{u}^T\mathbf{u} = \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} = \epsilon \\ -[\epsilon\,,\, b] - [0\,,\, a] < 0 & \text{if } \mathbf{u}^T\mathbf{u} = \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} = -\epsilon \end{cases}$$

An analysis completely similar to that made in the former subsection allows to conclude that the trajectories determined by (43) converge to the point $\mathbf{w}^*$.

This algorithm will be denoted as HBF5.

7f) Sixth choice

$$\mu = \begin{cases} \mu_1 > 1 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} > 0 \\ \mu_2 \mid 0 < \mu_2 < 1 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} \leq 0 \end{cases}$$
$$\gamma(\mathbf{x}, \mathbf{u}) = \begin{cases} (1-\mu)\frac{\nabla^T\phi(\mathbf{x})\mathbf{u}}{\mathbf{u}^T\mathbf{u}} + a & \text{if } \mathbf{u}^T\mathbf{u} > \epsilon \\ a & \text{if } \mathbf{u}^T\mathbf{u} \leq \epsilon \end{cases} \tag{44}$$

where $a > 0$ and $\epsilon > 0$ is a constant value small enough.

The system (33) becomes nonsmooth with these parameters, and it can be represented by the following set valued map:

$$\dot{\mathbf{w}}(t) = \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{u} \\ -\mu\nabla\phi(\mathbf{x}) - \gamma(\mathbf{x}, \mathbf{u})\mathbf{u} \end{bmatrix} \in \mathcal{F}(\mathbf{w}) =$$
$$\begin{cases} [\mathbf{u}^T & -\mu_1\nabla^T\phi(\mathbf{x}) - \left[(1-\mu_1)\frac{\nabla^T\phi(\mathbf{x})\mathbf{u}}{\mathbf{u}^T\mathbf{u}} + a\right]\mathbf{u}^T]^T & \text{if } \mathbf{u}^T\mathbf{u} > \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} > 0 \\ [\mathbf{u}^T & -\mu_1\nabla^T\phi(\mathbf{x}) - a\mathbf{u}^T]^T & \text{if } \mathbf{u}^T\mathbf{u} < \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} > 0 \\ [\mathbf{u}^T & -\mu_2\nabla^T\phi(\mathbf{x}) - \left[(1-\mu_2)\frac{\nabla^T\phi(\mathbf{x})\mathbf{u}}{\mathbf{u}^T\mathbf{u}} + a\right]\mathbf{u}^T]^T & \text{if } \mathbf{u}^T\mathbf{u} > \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} < 0 \\ [\mathbf{u}^T & -\mu_2\nabla^T\phi(\mathbf{x}) - a\mathbf{u}^T & \text{if } \mathbf{u}^T\mathbf{u} < \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} < 0 \\ [\mathbf{u}^T & -\mu_1\nabla^T\phi(\mathbf{x}) - \left[a\,,\, (1-\mu_1)\frac{\nabla^T\phi(\mathbf{x})\mathbf{u}}{\epsilon} + a\right]\mathbf{u}^T]^T & \text{if } \mathbf{u}^T\mathbf{u} = \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} > 0 \\ [\mathbf{u}^T & -\mu_2\nabla^T\phi(\mathbf{x}) - \left[a\,,\, (1-\mu_2)\frac{\nabla^T\phi(\mathbf{x})\mathbf{u}}{\epsilon} + a\right]\mathbf{u}^T]^T & \text{if } \mathbf{u}^T\mathbf{u} = \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} < 0 \\ [\mathbf{u}^T & -[\mu_1\,,\, \mu_2]\nabla^T\phi(\mathbf{x}) - a\mathbf{u}^T]^T & \text{if } \mathbf{u}^T\mathbf{u} > \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} = 0 \\ [\mathbf{u}^T & -[\mu_1\,,\, \mu_2]\nabla^T\phi(\mathbf{x}) - a\mathbf{u}^T]^T & \text{if } \mathbf{u}^T\mathbf{u} < \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} = 0 \\ [\mathbf{u}^T & -[\mu_1\,,\, \mu_2]\nabla^T\phi(\mathbf{x}) - a\mathbf{u}^T]^T & \text{if } \mathbf{u}^T\mathbf{u} = \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} = 0 \end{cases} \tag{45}$$

where $\mathbf{w}(t) = [\mathbf{x}(t)^T\ \mathbf{u}(t)^T]^T$. Defining $\mathbf{w}^* = [\mathbf{x}_*^T\ \mathbf{0}^T]^T$, being $\mathbf{x}_*$ any zero gradient point, we note that $\mathbf{0} \in \mathcal{F}(\mathbf{w}^*)$, hence $\mathbf{w}^*$ is an equilibrium point of (45).

The Lie derivative of the candidate Liapunov function (4) along the trajectories defined by (45) is:

$$\dot{V}(\mathbf{w}) \in \bar{\mathcal{L}}_{\mathcal{F}} V(\mathbf{w}) = \{\nabla^T V(\mathbf{w})\mathbf{v} \mid \mathbf{v} \in \mathcal{F}(\mathbf{w})\} =$$

$$= \begin{cases} -a\mathbf{u}^T\mathbf{u} < 0 & \text{if } \mathbf{u}^T\mathbf{u} > \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} > 0 \\ (1-\mu_1)\nabla^T\phi(\mathbf{x})\mathbf{u} - a\mathbf{u}^T\mathbf{u} < 0 & \text{if } \mathbf{u}^T\mathbf{u} < \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} > 0 \\ -a\mathbf{u}^T\mathbf{u} < 0 & \text{if } \mathbf{u}^T\mathbf{u} > \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} < 0 \\ (1-\mu_2)\nabla^T\phi(\mathbf{x})\mathbf{u} - a\mathbf{u}^T\mathbf{u} < 0 & \text{if } \mathbf{u}^T\mathbf{u} < \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} < 0 \\ -a\epsilon + [0\,,\,1](1-\mu_1)\nabla^T\phi(\mathbf{x})\mathbf{u} < 0 & \text{if } \mathbf{u}^T\mathbf{u} = \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} > 0 \\ -a\epsilon + [0\,,\,1](1-\mu_2)\nabla^T\phi(\mathbf{x})\mathbf{u} < 0 & \text{if } \mathbf{u}^T\mathbf{u} = \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} < 0 \\ -a\mathbf{u}^T\mathbf{u} < 0 & \text{if } \mathbf{u}^T\mathbf{u} > \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} = 0 \\ -a\mathbf{u}^T\mathbf{u} \leq 0 & \text{if } \mathbf{u}^T\mathbf{u} < \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} = 0 \\ -a\epsilon < 0 & \text{if } \mathbf{u}^T\mathbf{u} = \epsilon \text{ and } \nabla^T\phi(\mathbf{x})\mathbf{u} = 0 \end{cases}$$

Hence $\bar{\mathcal{L}}_{\mathcal{F}}(\mathbf{w}) < 0$ for all $\mathbf{w}$ such that $\mathbf{u} \neq \mathbf{0}$, and $\bar{\mathcal{L}}_{\mathcal{F}}(\mathbf{w}) = 0$ for all $\mathbf{w}$ such that $\mathbf{u} = \mathbf{0}$. Making an analysis completely similar to that made in the former sections, $\mathbf{w}(t)$ tends to the invariant set $\{\mathbf{w}^*\}$, that is, to a zero gradient point.

This algorithm will be denoted HBF6.

Surely, there exist other choices of the parameters $\mu(\mathbf{x}, \mathbf{u})$ and $\gamma(\mathbf{x}, \mathbf{u})$ that make $\bar{\mathcal{L}}_{\mathcal{F}} V(\mathbf{w})$ negative for all $\mathbf{u} \neq \mathbf{0}$. The ones presented here are only the simplest examples.

8) CG

The conjugate gradient algorithm (12), as mentioned above, can also be viewed as a closed loop control system. Equation (12) is rewritten here for convenience.

$$\begin{aligned} \dot{\mathbf{x}} &= \alpha(\mathbf{x}, \mathbf{u})\mathbf{u} \\ \dot{\mathbf{u}} &= -\nabla\phi(\mathbf{x}) - \beta(\mathbf{x}, \mathbf{u})\mathbf{u} \end{aligned} \tag{46}$$

Choosing (4) as candidate Liapunov function, and using (46) as the choices of plant and controller, respectively, the candidate Liapunov function derivative yields:

$$\dot{V} = (\alpha(\mathbf{x}, \mathbf{u}) - 1)\mathbf{u}^T\nabla\phi(\mathbf{x}) - \beta(\mathbf{x}, \mathbf{u})\mathbf{u}^T\mathbf{u}$$

Observe that the choice $\alpha = 1$ results $\dot{V} < 0$ for all $\beta > 0$ and $\mathbf{u} \neq \mathbf{0}$, but in this case (46) becomes equivalent to the HBF ODE (6) with the parameter $\beta = \gamma$.

Here again, different choices of the parameters $\alpha(\mathbf{x}, \mathbf{u})$ and $\beta(\mathbf{x}, \mathbf{u})$ lead to different algorithms.

Remark 3.1 *Observe that, if $\alpha(\mathbf{x}, \mathbf{u})$ is assumed differentiable in $\mathbf{x}, \mathbf{u}$, then (46) can be rewritten (in terms of $\ddot{\mathbf{x}}, \dot{\mathbf{x}}, \mathbf{x}$) as follows:*

$$\ddot{\mathbf{x}} + \beta(\mathbf{x}, \mathbf{u})\dot{\mathbf{x}} + \alpha(\mathbf{x}, \mathbf{u})\nabla\phi(\mathbf{x}) - \left[\frac{\partial\alpha}{\partial\mathbf{x}}\dot{\mathbf{x}} + \frac{\partial\alpha}{\partial\mathbf{u}}\dot{\mathbf{u}}\right]\mathbf{u} = \mathbf{0}$$

If it is assumed that α depends only on $\mathbf{x}$ (and not on $\mathbf{u}$), then (46) can be reduced to second order form, yielding (assuming $\alpha(\mathbf{x}) \neq 0$, for all $\mathbf{x}$):

$$\ddot{\mathbf{x}} + \left[\beta(\mathbf{x}) - \frac{\nabla\alpha(\mathbf{x})^T\dot{\mathbf{x}}}{\alpha(\mathbf{x})}\right]\dot{\mathbf{x}} + \alpha(\mathbf{x})\nabla\phi(\mathbf{x}) = 0 \tag{47}$$

so that the state-dependent term $\alpha(\mathbf{x})$ is seen to give rise to a nonlinear damping term (the term within square brackets in (47)). Of course, a switching law of the parameter α for the CG ODE is discontinuous, so reduction to the second order form above is not possible, but it does serve as an indication that the CG dynamics are quite different and more complex than the HBF dynamics.

In the sequel, some choices of the parameters $\alpha(\mathbf{x},\mathbf{u})$ and $\beta(\mathbf{x},\mathbf{u})$ will be presented.

8a) First choice
$\beta > 0$ as a constant value.

$$\alpha(\mathbf{x},\mathbf{u}) = \begin{cases} 1+\beta\frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi(\mathbf{x})\mathbf{u}} - a & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} > \epsilon \\ 1+\beta\frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi(\mathbf{x})\mathbf{u}} + b & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} < -\epsilon \\ 1 & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| \le \epsilon \end{cases} \tag{48}$$

where $a > 0$, $b > 0$ and $\epsilon > 0$ is a constant value small enough.

Also here, as in the case 7a), the parameters a and b can be variables. For example, the choices

$a = \beta\frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi(\mathbf{x})\mathbf{u}} + c,\ \ c > 0$ leads to $\alpha = 1 - c$ if $\nabla^T\phi(\mathbf{x})\mathbf{u} > \epsilon$

$b = -\beta\frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi(\mathbf{x})\mathbf{u}} + d,\ \ d > 0$ leads to $\alpha = 1 + d$ if $\nabla^T\phi(\mathbf{x})\mathbf{u} < -\epsilon$

With these choices of the parameters β and $\alpha(\mathbf{x},\mathbf{u})$, the algorithm (46) becomes nonsmooth, and can be represented by the following set valued map:

$$\begin{aligned} \dot{\mathbf{w}}(t) &= \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \alpha(\mathbf{x},\mathbf{u})\mathbf{u} \\ -\nabla\phi(\mathbf{x}) - \beta\mathbf{u} \end{bmatrix} \in \mathcal{F}(\mathbf{w}) = \\ &= \begin{cases} \left[\left(1+\beta\frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi(\mathbf{x})\mathbf{u}} - a\right)\mathbf{u}^T \ \ -\nabla^T\phi(\mathbf{x}) - \beta\mathbf{u}^T\right]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} > \epsilon \\ \left[\left(1+\beta\frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi(\mathbf{x})\mathbf{u}} + b\right)\mathbf{u}^T \ \ -\nabla^T\phi(\mathbf{x}) - \beta\mathbf{u}^T\right]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} < -\epsilon \\ [\mathbf{u}^T \ \ -\nabla^T\phi(\mathbf{x}) - \beta\mathbf{u}^T]^T & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| < \epsilon \\ \left[[1\,,\,1+\beta\frac{\mathbf{u}^T\mathbf{u}}{\epsilon} - a]\mathbf{u}^T \ \ -\nabla^T\phi(\mathbf{x}) - \beta\mathbf{u}^T\right]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = \epsilon \\ \left[[1\,,\,1-\beta\frac{\mathbf{u}^T\mathbf{u}}{\epsilon} + b]\mathbf{u}^T \ \ -\nabla^T\phi(\mathbf{x}) - \beta\mathbf{u}^T\right]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = -\epsilon \end{cases} \end{aligned} \tag{49}$$

where $\mathbf{w}(t) = [\mathbf{x}(t)^T \ \ \mathbf{u}(t)^T]^T$. Defining $\mathbf{w}^* = [\mathbf{x}_*^T \ \ \mathbf{0}^T]^T$, being $\mathbf{x}_*$ any zero gradient point, $\mathbf{0} \in \mathcal{F}(\mathbf{w}^*)$, hence $\mathbf{w}^*$ is an equilibrium point of (49).

Choosing (4) as candidate Liapunov function, the Lie derivative of $V(\mathbf{w})$ along the trajectories defined by (49) is:

$$\begin{aligned} \dot{V}(\mathbf{w}) &\in \bar{\mathcal{L}}_{\mathcal{F}}V(\mathbf{w}) = \left\{\nabla^T V(\mathbf{w})\mathbf{v} \mid \mathbf{v} \in \mathcal{F}(\mathbf{w})\right\} = \\ &= \begin{cases} -a\nabla^T\phi(\mathbf{x})\mathbf{u} < 0 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} > \epsilon \\ b\nabla^T\phi(\mathbf{x})\mathbf{u} < 0 & \text{if } \nabla\phi(\mathbf{x})^T\mathbf{u} < -\epsilon \\ -\beta\mathbf{u}^T\mathbf{u} \le 0 & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| < \epsilon \\ [-a\epsilon\,,\,-\beta\mathbf{u}^T\mathbf{u}] < 0 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = \epsilon \\ [-b\epsilon\,,\,-\beta\mathbf{u}^T\mathbf{u}] < 0 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = -\epsilon \end{cases} \end{aligned}$$

Here again, an analysis similar to that made in the former subsections, allow us to conclude that $\mathbf{u}$ and $\dot{\mathbf{u}}$ tend to zero, and, by the controller equation, $\nabla\phi(\mathbf{x})$ also tends to zero. Therefore, the trajectories generated by this algorithm can

stop at any zero gradient point, depending on the choice of the parameters β, a and b the possibility to pass through basins of attraction of local minima to converge to the global minimum of the function.

This algorithm will be denoted as CG1.

8b) Second choice

$\beta > 0$ as a constant value.

$\alpha(\mathbf{x}, \mathbf{u}) = 1 - \kappa \operatorname{sgn}(\nabla^T \phi(\mathbf{x})\mathbf{u})$

where $\kappa > 0$. Here again, the system (46) becomes nonsmooth, and can be represented by the following set valued map:

$$\begin{aligned} \dot{\mathbf{w}}(t) &= \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \alpha(\mathbf{x}, \mathbf{u})\mathbf{u} \\ -\nabla\phi(\mathbf{x}) - \beta\mathbf{u} \end{bmatrix} \in \mathcal{F}(\mathbf{w}) = \\ &= \begin{cases} [(1-\kappa)\mathbf{u}^T \quad -\nabla^T\phi(\mathbf{x}) - \beta\mathbf{u}^T]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} > 0 \\ [(1+\kappa)\mathbf{u}^T \quad -\nabla^T\phi(\mathbf{x}) - \beta\mathbf{u}^T]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} < 0 \\ [(1-[-\kappa\,,+\kappa])\mathbf{u}^T \quad -\nabla^T\phi(\mathbf{x}) - \beta\mathbf{u}^T]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = 0 \end{cases} \end{aligned} \tag{50}$$

where $\mathbf{w}(t) = [\mathbf{x}(t)^T \quad \mathbf{u}(t)^T]^T$. Here again, note that the only one equilibrium point is $\mathbf{w}^* = [\mathbf{x}_*^T \; \mathbf{0}^T]^T$, being $\mathbf{x}_*$ any zero gradient point, for all $\kappa > 0$. Note that this equilibrium point includes the case $\kappa = 1$, because if $\dot{\mathbf{u}} = \mathbf{0}$, then $\nabla\phi(\mathbf{x}) = -\beta\mathbf{u}$, so $\nabla^T\phi(\mathbf{x})\mathbf{u} \leq 0$ and the first case cannot reach an equilibrium point; in the second case $(1+\kappa)\mathbf{u} = \mathbf{0}$, which implies $\mathbf{u} = \mathbf{0}$, so $\nabla^T\phi(\mathbf{x})\mathbf{u} = 0$ and this second case cannot reach the equilibrium point; in the third case if $\nabla^T\phi(\mathbf{x})\mathbf{u} = 0$ and $\nabla^T\phi(\mathbf{x}) = -\beta\mathbf{u}$, then $\mathbf{u} = \mathbf{0}$ and $\nabla\phi(\mathbf{x}) = \mathbf{0}$ is the only one equilibrium point.

Choosing (4) as candidate Liapunov function, its Lie derivative along the trajectories defined by (50) is:

$$\begin{aligned} \dot{V} &\in \bar{\mathcal{L}}_{\mathcal{F}} V(\mathbf{w}) = \{\nabla^T V(\mathbf{w})\mathbf{v} \mid \mathbf{v} \in \mathcal{F}(\mathbf{w})\} = \\ &= \begin{cases} -\kappa\nabla^T\phi(\mathbf{x})\mathbf{u} - \beta\mathbf{u}^T\mathbf{u} < 0 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} > 0 \\ \kappa\nabla^T\phi(\mathbf{x})\mathbf{u} - \beta\mathbf{u}^T\mathbf{u} < 0 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} < 0 \\ -\beta\mathbf{u}^T\mathbf{u} \leq 0 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = 0 \end{cases} \end{aligned}$$

An analysis completely similar to that made in the former subsections allow us to conclude that the point $\mathbf{w}^*$ is asymptotically stable.

This algorithm will be denoted as CG2.

8c) Third choice

$\beta > 0$ as a constant value.

$$\alpha(\mathbf{x}, \mathbf{u}) = \begin{cases} 1 - \frac{\kappa}{\nabla^T\phi(\mathbf{x})\mathbf{u}} & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| > \epsilon \\ 1 & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| \leq \epsilon \end{cases} \tag{51}$$

where $\kappa > 0$ and $\epsilon > 0$ is a constant value small enough.

Here again, for $|\nabla^T\phi(\mathbf{x})\mathbf{u}| < \epsilon$, the algorithm (46) becomes equal to the HBF ODE (6), with the parameter $\beta = \gamma$.

This is a nonsmooth system which can be represented by the following set valued map:

$$\dot{\mathbf{w}}(t) = \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \alpha(\mathbf{x},\mathbf{u})\mathbf{u} \\ -\nabla\phi(\mathbf{x}) - \beta\mathbf{u} \end{bmatrix} \in \mathcal{F}(\mathbf{w}) =$$
$$= \begin{cases} [\left(1 - \frac{\kappa}{\nabla^T\phi(\mathbf{x})\mathbf{u}}\right)\mathbf{u}^T \ -\nabla^T\phi(\mathbf{x}) - \beta\mathbf{u}^T]^T & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| > \epsilon \\ [\mathbf{u}^T \ -\nabla^T\phi(\mathbf{x}) - \beta\mathbf{u}^T]^T & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| < \epsilon \\ [[1-\kappa/\epsilon\,,\,1]\mathbf{u}^T \ -\nabla^T\phi(\mathbf{x}) - \beta\mathbf{u}^T]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = \epsilon \\ [[1\,,\,1+\kappa/\epsilon]\mathbf{u}^T \ -\nabla^T\phi(\mathbf{x}) - \beta\mathbf{u}^T]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = -\epsilon \end{cases} \tag{52}$$

where $\mathbf{w}(t) := [\mathbf{x}(t)^T \ \ \mathbf{u}(t)^T]^T$. Here again, the only one equilibrium point is $\mathbf{w}^* = [\mathbf{x}_*^T \ \mathbf{0}^T]^T$, where $\mathbf{x}_*$ is any zero gradient point, because $\mathbf{0} \in \mathcal{F}(\mathbf{w}^*)$ and $\mathbf{0} \notin \mathcal{F}(\mathbf{w})$ for all $\mathbf{w} \neq \mathbf{w}^*$.

Choosing (4) as candidate Liapunov function, its Lie derivative along the trajectories defined by (52) is:

$$\dot{V} \in \bar{\mathcal{L}}_{\mathcal{F}}V(\mathbf{w}) = \{\nabla^T V(\mathbf{w})\mathbf{v} \mid \mathbf{v} \in \mathcal{F}(\mathbf{w})\} =$$
$$= \begin{cases} -\kappa - \beta\mathbf{u}^T\mathbf{u} < 0 & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| > \epsilon \\ -\beta\mathbf{u}^T\mathbf{u} \leq 0 & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| < \epsilon \\ [-\kappa\,,\,0] - \beta\mathbf{u}^T\mathbf{u} < 0 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = \epsilon \\ [-\kappa\,,\,0] - \beta\mathbf{u}^T\mathbf{u} < 0 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = -\epsilon \end{cases}$$

Here again, the system (52) tends to the largest invariant set asymptotically, and this set is $\{\mathbf{w}^*\}$.

This algorithm will be denoted as CG3.

8d) Fourth choice

$\beta > 0$ as a constant value.

$\alpha(\mathbf{x},\mathbf{u}) = 1 - \kappa\nabla^T\phi(\mathbf{x})\mathbf{u}$

where $\kappa > 0$. Note that this is a smooth system, which can be represented by the following first order ODE:

$$\dot{\mathbf{w}}(t) = \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} (1 - \kappa\nabla^T\phi(\mathbf{x})\mathbf{u})\mathbf{u} \\ -\nabla\phi(\mathbf{x}) - \beta\mathbf{u} \end{bmatrix} \tag{53}$$

where $\mathbf{w}(t) = [\mathbf{x}(t)^T \ \mathbf{u}(t)^T]^T$.

Choosing (4) as candidate Liapunov function, its derivative along the trajectories determined by (53) is:

$$\dot{V}(\mathbf{w}) = \nabla^T V\dot{\mathbf{w}} = -\kappa(\nabla^T\phi(\mathbf{x})\mathbf{u})^2 - \beta\mathbf{u}^T\mathbf{u} \leq 0$$

By Barbalat's lemma, as $\dot{V}$ is uniformly continuous, $\dot{V}$ tends to zero when t tends to infinity. Hence, $\nabla^T\phi(\mathbf{x})\mathbf{u}$ and $\mathbf{u}$ tend to zero too. By the controller equation, $\nabla\phi(\mathbf{x})$ tends to zero too. Therefore, the trajectories generated by this system tend asymptotically to the only one equilibrium point, which is $\mathbf{w}^* = [\mathbf{x}_*^T \ \mathbf{0}^T]^T$, where $\mathbf{x}_*$ is any zero gradient point.

This algorithm will be denoted as CG4.

8e) Fifth choice

$$\begin{array}{l} \beta(\mathbf{x},\mathbf{u}) = \left\{ \begin{array}{ll} \frac{b}{\mathbf{u}^T\mathbf{u}} & \text{if } \mathbf{u}^T\mathbf{u} > \epsilon \\ 1 & \text{if } \mathbf{u}^T\mathbf{u} \leq \epsilon \end{array} \right. \\ \alpha(\mathbf{x},\mathbf{u}) = \left\{ \begin{array}{ll} 1 - \frac{a}{\nabla^T\phi(\mathbf{x})\mathbf{u}} & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| > \epsilon \\ 1 & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| \leq \epsilon \end{array} \right. \end{array} \tag{54}$$

where $a > 0$, $b > 0$ and $\epsilon > 0$ is a constant value small enough.

This is a nonsmooth system, which can be represented by the following nonautonomous piecewise continuous Filippov set valued map:

$$\begin{array}{rl} \dot{\mathbf{w}} & = \left[\begin{array}{c} \dot{\mathbf{x}} \\ \dot{\mathbf{u}} \end{array} \right] = \left[\begin{array}{c} \alpha(\mathbf{x},\mathbf{u})\mathbf{u} \\ -\nabla\phi(\mathbf{x}) - \beta(\mathbf{u})\mathbf{u} \end{array} \right] \in \mathcal{F}(\mathbf{w}) = \\ & = \left\{ \begin{array}{ll} [\left(1 - \frac{a}{\nabla^T\phi(\mathbf{x})\mathbf{u}}\right)\mathbf{u}^T \quad - \nabla^T\phi(\mathbf{x}) - \frac{b}{\mathbf{u}^T\mathbf{u}}\mathbf{u}^T]^T & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| > \epsilon \text{ and } \mathbf{u}^T\mathbf{u} > \epsilon \\ [\left(1 - \frac{a}{\nabla^T\phi(\mathbf{x})\mathbf{u}}\right)\mathbf{u}^T \quad - \nabla^T\phi(\mathbf{x}) - \mathbf{u}^T]^T & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| > \epsilon \text{ and } \mathbf{u}^T\mathbf{u} < \epsilon \\ [\mathbf{u}^T \quad - \nabla^T\phi(\mathbf{x}) - \frac{b}{\mathbf{u}^T\mathbf{u}}\mathbf{u}^T]^T & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| < \epsilon \text{ and } \mathbf{u}^T\mathbf{u} > \epsilon \\ [\mathbf{u}^T \quad - \nabla^T\phi(\mathbf{x}) - \mathbf{u}^T]^T & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| < \epsilon \text{ and } \mathbf{u}^T\mathbf{u} < \epsilon \\ [\left(1 - \frac{a}{\nabla^T\phi(\mathbf{x})\mathbf{u}}\right)\mathbf{u}^T \quad - \nabla^T\phi(\mathbf{x}) - \left[1\,,\,\frac{b}{\epsilon}\right]\mathbf{u}^T]^T & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| > \epsilon \text{ and } \mathbf{u}^T\mathbf{u} = \epsilon \\ [\mathbf{u}^T \quad - \nabla^T\phi(\mathbf{x}) - \left[1\,,\,\frac{b}{\epsilon}\right]\mathbf{u}^T]^T & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| < \epsilon \text{ and } \mathbf{u}^T\mathbf{u} = \epsilon \\ [\left[\left(1 - \frac{a}{\epsilon}\right)\,,\,1\right]\mathbf{u}^T \quad - \nabla^T\phi(\mathbf{x}) - \frac{b}{\mathbf{u}^T\mathbf{u}}\mathbf{u}^T]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = \epsilon \text{ and } \mathbf{u}^T\mathbf{u} > \epsilon \\ [\left[\left(1 - \frac{a}{\epsilon}\right)\,,\,1\right]\mathbf{u}^T \quad - \nabla^T\phi(\mathbf{x}) - \mathbf{u}^T]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = \epsilon \text{ and } \mathbf{u}^T\mathbf{u} < \epsilon \\ [\left[1\,,\,\left(1 + \frac{a}{\epsilon}\right)\right]\mathbf{u}^T \quad - \nabla^T\phi(\mathbf{x}) - \frac{b}{\mathbf{u}^T\mathbf{u}}\mathbf{u}^T]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = -\epsilon \text{ and } \mathbf{u}^T\mathbf{u} > \epsilon \\ [\left[1\,,\,\left(1 + \frac{a}{\epsilon}\right)\right]\mathbf{u}^T \quad - \nabla^T\phi(\mathbf{x}) - \mathbf{u}^T]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = -\epsilon \text{ and } \mathbf{u}^T\mathbf{u} < \epsilon \\ [\left[\left(1 - \frac{a}{\epsilon}\,,\,1\right)\right]\mathbf{u}^T \quad - \nabla^T\phi(\mathbf{x}) - \left[1\,,\,\frac{b}{\epsilon}\right]\mathbf{u}^T]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = \epsilon \text{ and } \mathbf{u}^T\mathbf{u} = \epsilon \\ [\left[1\,,\,\left(1 + \frac{a}{\epsilon}\right)\right]\mathbf{u}^T \quad - \nabla^T\phi(\mathbf{x}) - \left[1\,,\,\frac{b}{\epsilon}\right]\mathbf{u}^T]^T & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = -\epsilon \text{ and } \mathbf{u}^T\mathbf{u} = \epsilon \end{array} \right. \end{array} \tag{55}$$

where $\mathbf{w}(t) = [\mathbf{x}(t)^T \ \mathbf{u}(t)^T]^T$. Note that the point $\mathbf{w}^* = [\mathbf{x}_*^T \ \mathbf{0}^T]^T$ is the only one equilibrium point of (55) because $\mathbf{0} \in \mathcal{F}(\mathbf{w}^*)$ and $\mathbf{0} \notin \mathcal{F}(\mathbf{w})$ for all $\mathbf{w} \neq \mathbf{w}^*$.

Choosing (4) as candidate Liapunov function, the Lie derivative of (4) along the trajectories determined by (55) can be defined as:

$$\begin{array}{rl} \dot{V} & \in \bar{\mathcal{L}}_{\mathcal{F}} V(\mathbf{w}) = \{\nabla^T V\mathbf{v} \mid \mathbf{v} \in \mathcal{F}(\mathbf{w})\} = \\ & = \left\{ \begin{array}{ll} -a - b < 0 & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| > \epsilon \text{ and } \mathbf{u}^T\mathbf{u} > \epsilon \\ -a - \mathbf{u}^T\mathbf{u} < 0 & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| < \epsilon \text{ and } \mathbf{u}^T\mathbf{u} < \epsilon \\ -b < 0 & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| < \epsilon \text{ and } \mathbf{u}^T\mathbf{u} > \epsilon \\ -\mathbf{u}^T\mathbf{u} \leq 0 & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| < \epsilon \text{ and } \mathbf{u}^T\mathbf{u} < \epsilon \\ -a - [\epsilon\,,\,b] < 0 & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| > \epsilon \text{ and } \mathbf{u}^T\mathbf{u} = \epsilon \\ -[\epsilon\,,\,b] < 0 & \text{if } |\nabla^T\phi(\mathbf{x})\mathbf{u}| < \epsilon \text{ and } \mathbf{u}^T\mathbf{u} = \epsilon \\ [-a\,,\,0] - b < 0 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = \epsilon \text{ and } \mathbf{u}^T\mathbf{u} > \epsilon \\ [-a\,,\,0] - \mathbf{u}^T\mathbf{u} < 0 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = \epsilon \text{ and } \mathbf{u}^T\mathbf{u} < \epsilon \\ [-a\,,\,0] - b < 0 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = -\epsilon \text{ and } \mathbf{u}^T\mathbf{u} > \epsilon \\ [-a\,,\,0] - \mathbf{u}^T\mathbf{u} < 0 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = -\epsilon \text{ and } \mathbf{u}^T\mathbf{u} < \epsilon \\ [-a\,,\,0] - [\epsilon\,,\,b] < 0 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = \epsilon \text{ and } \mathbf{u}^T\mathbf{u} = \epsilon \\ [-a\,,\,0] - [\epsilon\,,\,b] < 0 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} = -\epsilon \text{ and } \mathbf{u}^T\mathbf{u} = \epsilon \end{array} \right. \end{array}$$

Here again, an analysis completely similar to that made in the former subsection allow us to conclude that the trajectories generated by (55) converge asymptotically to an equilibrium point $\mathbf{w}^*$.

This algorithm will be denoted as CG5.

With the presented choices of plants and controllers, as well as with the mentioned choices of the parameters, we can enunciate the following theorem:

Theorem 3.2 *The neural nets reported in Table 1, with their parameters chosen as showed in Table 2, make the time derivative (5) of the control Liapunov function (4) negative definite, thus guaranteing that the trajectories generated by these systems converge asymptotically to a state such that* $[\nabla\phi(\mathbf{x})\ \mathbf{u}] = [\mathbf{0}\ \mathbf{0}]$.

Note that the equilibrium point $[\nabla\phi(\mathbf{x})\ \mathbf{u}] = [\mathbf{0}\ \mathbf{0}]$ does not necessarily correspond to the global minimum of the objective function, but only to a local one. However, Lemma 2.1 suggests that there always exist parameter values for each algorithm, such that the trajectories converge to the global minimum $\mathbf{x}^*$.

It is clear that there exist other choices of $\alpha(\mathbf{x}, \mathbf{u})$ and $\beta(\mathbf{x}, \mathbf{u})$ that make the Lie derivative of (4) negative definite. The choices presented here are only some of the simplest ones.

The general conclusion so far is that each new choice of plant and controller, as well as each choice of control Liapunov function leads to a different algorithm.

4 Computational experiments

In this section, the algorithms presented in the previous section will be tested with a suite of benchmark scalar functions, well known in the optimization literature.

These function are a scalar function of one variable, the inverted camelback function, the Rastrigin function, the Ackley function, and the Griewank function. All these functions are nonconvex, multimodal and have a unique finite global minimum.

The nets were implemented with MATLAB 2007, and the corresponding ODEs discretized by the forward Euler method with a constant step size of $0.01\,s$. The simulations were carried out during $10\,s$.

4.1 Simulations with a scalar nonconvex function of one variable

The function chosen in this subsection is the shifted sinc function:

$$\phi(x) = -\frac{\sin(x)}{x} + 1 \ : \ \mathbb{R} \to \mathbb{R} \tag{56}$$

This function has multiple local minima and only one global minimum at $x^* = 0$. The value of the function at the global minimum is $\phi(x^*) = 0$.

Figure 4 shows the value of the function vs. the variable x.

The initial point was chosen $x_0 = -20$, which is a local minimum, and the initial value of the control variable as $u_0 = 20$. In all the tests, in the algorithms which have a parameter ϵ, it was chosen as $\epsilon = 0.1$. The other parameters were adjusted by hand tuning in such a way to achieve the fastest convergence to the global minimum. These parameters are reported in Table 3.

Table 1: Neural networks represented as second order continuous time algorithms to minimize nonconvex scalar functions deduced with CLF methodology

name	plant	controller	ODE
MI1	$\dot{\mathbf{x}} = \mathbf{u}$	$\dot{\mathbf{u}} = -\kappa \operatorname{sgn}(\mathbf{u}) - \nabla\phi$	$\ddot{\mathbf{x}} + \kappa \operatorname{sgn}(\dot{\mathbf{x}}) + \nabla\phi = \mathbf{0}$
MI2	$\dot{\mathbf{x}} = \mathbf{u}$	$\dot{\mathbf{u}} = -\kappa(\nabla^2\phi)^2\mathbf{u} - \nabla\phi$	$\ddot{\mathbf{x}} + \kappa(\nabla^2\phi)^2\dot{\mathbf{x}} + \nabla\phi = \mathbf{0}$
MI3	$\dot{\mathbf{x}} = \nabla^{-2}\phi\mathbf{u}$	$\dot{\mathbf{u}} = -\kappa\mathbf{u} - \nabla^{-2}\phi\nabla\phi$	$\ddot{\nabla}\phi + \kappa\dot{\nabla}\phi + \nabla^{-2}\phi\nabla\phi = \mathbf{0}$
MI4	$\dot{\mathbf{x}} = \nabla^2\phi\mathbf{u}$	$\dot{\mathbf{u}} = -\kappa\mathbf{u} - \nabla^2\phi\nabla\phi$	$\ddot{\mathbf{x}} \quad +(\kappa I - \dot{\nabla}^2\phi\nabla^{-2}\phi)\dot{\mathbf{x}}$ $+(\nabla^2\phi)^2\nabla\phi = \mathbf{0}$
MI5	$\dot{\mathbf{x}} = \nabla^{-2}\phi\mathbf{u}$	$\dot{\mathbf{u}} = \quad -(I - \gamma I + a\nabla^{-2}\phi)\mathbf{u}$ $-(\gamma I + b\nabla^{-2}\phi)\nabla\phi$	$\ddot{\nabla}\phi \quad +(I - \gamma I + a\nabla^{-2}\phi)\dot{\nabla}\phi$ $+(\gamma I + b\nabla^{-2}\phi)\nabla\phi = \mathbf{0}$
DIN	$\dot{\mathbf{x}} = \mathbf{u}$	$\dot{\mathbf{u}} = -\nabla\phi - (aI + b\nabla^2\phi)\mathbf{u}$	$\ddot{\mathbf{x}} + (aI + b\nabla^2\phi)\dot{\mathbf{x}} + \nabla\phi = \mathbf{0}$
HBF	$\dot{\mathbf{x}} = \mathbf{u}$	$\dot{\mathbf{u}} = -\nabla\phi - \gamma\mathbf{u}$	$\ddot{\mathbf{x}} + \gamma\dot{\mathbf{x}} + \nabla\phi = \mathbf{0}$
gHBF	$\dot{\mathbf{x}} = \mathbf{u}$	$\dot{\mathbf{u}} = -\gamma(\mathbf{x},\mathbf{u})\mathbf{u} - \mu(\mathbf{x},\mathbf{u})\nabla\phi$	$\ddot{\mathbf{x}} + \gamma(\mathbf{x},\mathbf{u})\dot{\mathbf{x}} + \mu(\mathbf{x},\mathbf{u})\nabla\phi = \mathbf{0}$
CG	$\dot{\mathbf{x}} = \alpha(\mathbf{x},\mathbf{u})\mathbf{u}$	$\dot{\mathbf{u}} = -\nabla\phi - \beta(\mathbf{x},\mathbf{u})\mathbf{u}$	$\ddot{\mathbf{x}} \quad +\alpha\nabla\phi + \beta\dot{\mathbf{x}}$ $-\left[\frac{\partial\alpha}{\partial\mathbf{x}}^T\dot{\mathbf{x}} + \frac{\partial\alpha}{\partial\mathbf{u}}^T\dot{\mathbf{u}}\right]\mathbf{u} = \mathbf{0}$

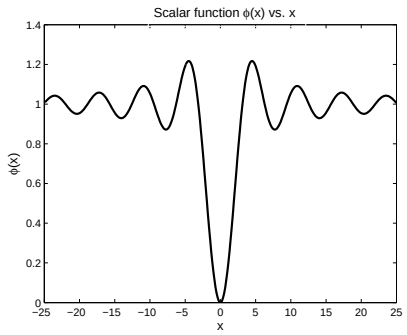

Figure 4: Scalar function of one variable vs. its independent variable.

Table 2: Parameters and switching laws used by the different nets

name	parameters
MI1	$\kappa > 0$
MI2	$\kappa > 0$
MI3	$\kappa > 0$
MI4	$\kappa > 0$
MI5	$\gamma \in \mathbb{R},\ \ a + b > 0$
DIN	$a > 0,\ \ b > 0$
HBF	$\gamma > 0$
HBF1	$\gamma > 0, \quad \mu(\mathbf{x},\mathbf{u}) = \begin{cases} 1 - \gamma\frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi(\mathbf{x})\mathbf{u}} + a & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} > \epsilon \\ 1 - \gamma\frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi(\mathbf{x})\mathbf{u}} - b & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} < -\epsilon \\ 1 & \text{if } \lvert\nabla^T\phi(\mathbf{x})\mathbf{u}\rvert \leq \epsilon \end{cases},\ a > 0,\ b > 0$
HBF2	$\gamma > 0, \quad \mu(\mathbf{x},\mathbf{u}) = 1 + \kappa\,\mathrm{sgn}(\nabla^T\phi(\mathbf{x})\mathbf{u}),\ \kappa > 0$
HBF3	$\gamma > 0, \quad \mu(\mathbf{x},\mathbf{u}) = \begin{cases} 1 + \kappa\frac{1}{\nabla^T\phi(\mathbf{x})\mathbf{u}} & \text{if } \lvert\nabla^T\phi(\mathbf{x})\mathbf{u}\rvert > \epsilon \\ 1 & \text{if } \lvert\nabla^T\phi(\mathbf{x})\mathbf{u}\rvert \leq \epsilon \end{cases},\ \kappa > 0$
HBF4	$\gamma > 0, \quad \mu(\mathbf{x},\mathbf{u}) = 1 + \kappa\nabla^T\phi(\mathbf{x})\mathbf{u},\ \kappa > 0$
HBF5	$\gamma(\mathbf{x},\mathbf{u}) = \begin{cases} \frac{b}{\mathbf{u}^T\mathbf{u}} & \text{if } \mathbf{u}^T\mathbf{u} > \epsilon \\ 1 & \text{if } \mathbf{u}^T\mathbf{u} \leq \epsilon \end{cases}$ $\mu(\mathbf{x},\mathbf{u}) = \begin{cases} 1 + \frac{a}{\nabla^T\phi(\mathbf{x})\mathbf{u}} & \text{if } \lvert\nabla^T\phi(\mathbf{x})\mathbf{u}\rvert > \epsilon \\ 1 & \text{if } \lvert\nabla^T\phi(\mathbf{x})\mathbf{u}\rvert \leq \epsilon \end{cases},\ a > 0,\ b > 0$
HBF6	$\mu = \begin{cases} \mu_1 > 1 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} > 0 \\ \mu_2 \mid 0 < \mu_2 < 1 & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} \leq 0 \end{cases}$ $\gamma(\mathbf{x},\mathbf{u}) = \begin{cases} (1-\mu)\frac{\nabla^T\phi(\mathbf{x})\mathbf{u}}{\mathbf{u}^T\mathbf{u}} + a & \text{if } \mathbf{u}^T\mathbf{u} > \epsilon \\ a & \text{if } \mathbf{u}^T\mathbf{u} \leq \epsilon \end{cases},\ a > 0$
CG1	$\beta > 0,\ \ \alpha(\mathbf{x},\mathbf{u}) = \begin{cases} 1 + \beta\frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi(\mathbf{x})\mathbf{u}} - a & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} > \epsilon \\ 1 + \beta\frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi(\mathbf{x})\mathbf{u}} + b & \text{if } \nabla^T\phi(\mathbf{x})\mathbf{u} < -\epsilon \\ 1 & \text{if } \lvert\nabla^T\phi(\mathbf{x})\mathbf{u}\rvert \leq \epsilon \end{cases},\ a > 0,\ b > 0$
CG2	$\beta > 0,\ \ \alpha(\mathbf{x},\mathbf{u}) = 1 - \kappa\,\mathrm{sgn}(\nabla^T\phi\mathbf{u}),\ \kappa > 0$
CG3	$\beta > 0,\ \ \alpha(\mathbf{x},\mathbf{u}) = \begin{cases} 1 - \frac{\kappa}{\nabla^T\phi(\mathbf{x})\mathbf{u}} & \text{if } \lvert\nabla^T\phi(\mathbf{x})\mathbf{u}\rvert > \epsilon \\ 1 & \text{if } \lvert\nabla^T\phi(\mathbf{x})\mathbf{u}\rvert \leq \epsilon \end{cases},\ \kappa > 0$
CG4	$\beta > 0,\ \ \alpha(\mathbf{x},\mathbf{u}) = 1 - \kappa\nabla^T\phi\mathbf{u},\ \kappa > 0$
CG5	$\beta(\mathbf{x},\mathbf{u}) = \begin{cases} \frac{b}{\mathbf{u}^T\mathbf{u}} & \text{if } \mathbf{u}^T\mathbf{u} > \epsilon \\ 1 & \text{if } \mathbf{u}^T\mathbf{u} \leq \epsilon \end{cases}$ $\alpha(\mathbf{x},\mathbf{u}) = \begin{cases} 1 - \frac{a}{\nabla^T\phi(\mathbf{x})\mathbf{u}} & \text{if } \lvert\nabla^T\phi(\mathbf{x})\mathbf{u}\rvert > \epsilon \\ 1 & \text{if } \lvert\nabla^T\phi(\mathbf{x})\mathbf{u}\rvert \leq \epsilon \end{cases},\ a > 0,\ b > 0$

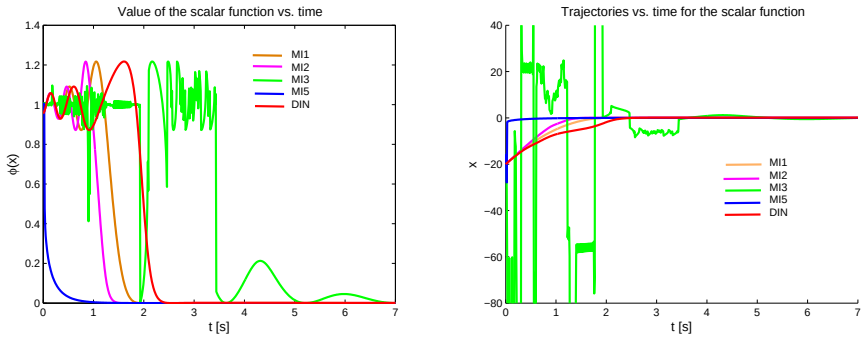

Figure 5: Value of the function and trajectories vs. time with the scalar function of one variable for the trajectories generated by the algorithms MI1, MI2, MI3, MI5 and DIN. Initial point $x_0 = -20$, initial value of the control variable $u_0 = 20$.

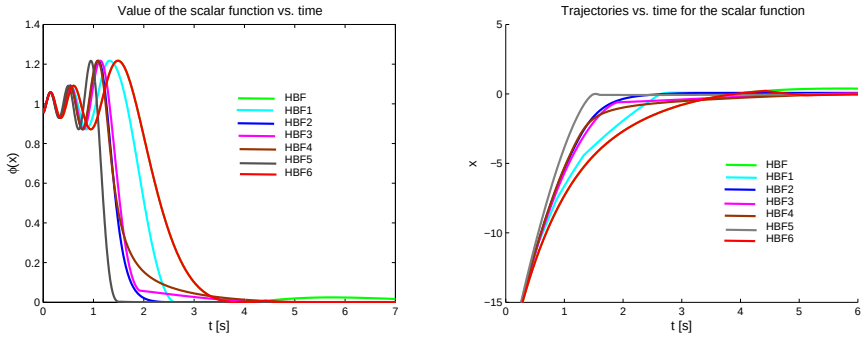

Figure 6: Value of the function and trajectories vs. time with the scalar function of one variable for the trajectories generated by the gHBF algorithms. Initial point $x_0 = -20$, initial value of the control variable $u_0 = 20$.

Figure 5 shows the value of the function and the trajectories vs. time for the trajectories generated by the algorithms MI1, MI2, MI3, MI5 and DIN. It was not possible to find values of the parameters to achieve convergence to the global minimum for the algorithm MI4.

Figure 6 shows the value of the function and the trajectories vs. time for the trajectories generated by the gHBF algorithms. Note that all the variations of the HBF algorithm achieve faster convergence than that of the HBF algorithm.

Figure 7 shows the value of the function and the trajectories vs. time for the trajectories generated by the CG algorithms. Note that, excepting the CG4 algorithm, all the others present a very much faster convergence than that presented by the gHBF algorithms.

Finally, we emphasize that, excepting the MI4 algorithm, all the other ones generated trajectories that were able to pass three local minima (including the initial point), to converge to the global minimum.

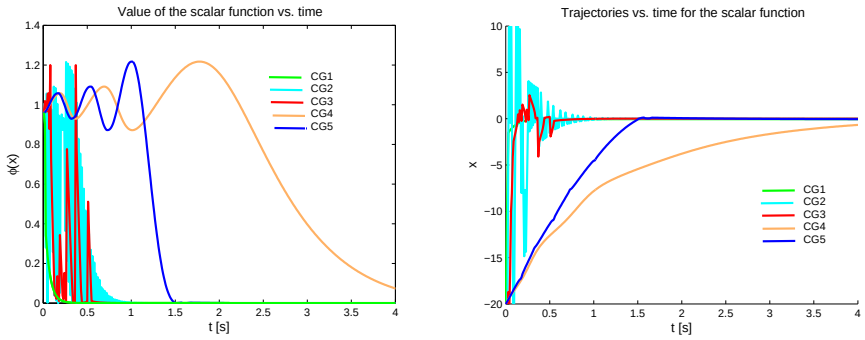

Figure 7: Value of the function and trajectories vs. time with the scalar function of one variable for the trajectories generated by the CG algorithms. Initial point $x_0 = -20$, initial value of the control variable $u_0 = 20$.

Table 3: Parameters used by the algorithms in the experiments reported in section 4.1 with the scalar function of one variable, where - means that the trajectory does not converge to a minimum in 10 s.

	Scalar function
MI1	$\kappa = 10.1$
MI2	$\kappa = 69$
MI3	$\kappa = 0.9$
MI4	-
MI5	$\gamma = 0.2, \quad a = 1, \quad b = 1$
DIN	$a = 1, \quad b = 13$
HBF	$\gamma = 1$
HBF1	$\gamma = 15, \quad \epsilon = 0.1$ $\mu = \begin{cases} 1 - \gamma \frac{\mathbf{u}^T \mathbf{u}}{\nabla^T \phi \mathbf{u}} + 18 & \nabla^T \phi \mathbf{u} > \epsilon \\ 1 - \gamma \frac{\mathbf{u}^T \mathbf{u}}{\nabla^T \phi \mathbf{u}} - 4 & \nabla^T \phi \mathbf{u} < -\epsilon \\ 1 & \lvert \nabla^T \phi \mathbf{u} \rvert \leq \epsilon \end{cases}$
HBF2	$\gamma = 0.6, \quad \kappa = 22.5$
HBF3	$\gamma = 0.61, \quad \kappa = 27.7, \quad \epsilon = 0.1$
HBF4	$\gamma = 0.6, \quad \kappa = 17.5$
HBF5	$a = 8, \quad b = 127.3, \quad \epsilon = 0.1$
HBF6	$\epsilon = 0.1$ $\mu_1 = 1297, \quad \mu_2 = 0.1$ $a = 1$
CG1	$\beta = 10, \quad \epsilon = 0.1$ $\alpha = \begin{cases} 34.1 & \nabla^T \phi \mathbf{u} > \epsilon \\ 1 & \nabla^T \phi \mathbf{u} < -\epsilon \\ 1 & \lvert \nabla^T \phi \mathbf{u} \rvert \leq \epsilon \end{cases}$
CG2	$\beta = 5, \quad \kappa = 104$
CG3	$\beta = 5, \quad \kappa = 14.07, \quad \epsilon = 0.1$
CG4	$\beta = 1, \quad \kappa = 0.28$
CG5	$\epsilon = 0.1, \quad a = 0.1, \quad b = 130$

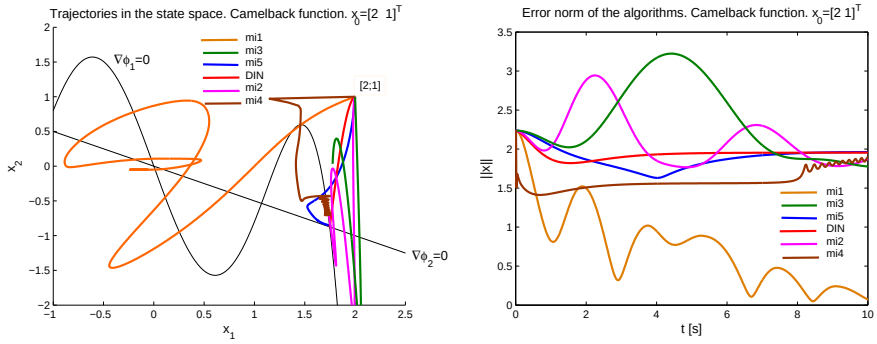

Figure 8: Trajectories in the state space and error norms vs. time with the inverted camelback function for the trajectories generated by the algorithms MI1, MI2, MI3, MI4, MI5 and DIN. Initial point $\mathbf{x}_0 = [2\ 1]^T$, initial value of the control variable $\mathbf{u}_0 = [0\ 0]^T$.

4.2 Simulations with the inverted camelback function

The inverted camelback function, popular in the literature on global optimization, was chosen as objective function in this subsection:

$$\phi(x_1, x_2) = ax_1^2 + bx_1^4 + cx_1^6 - x_1x_2 + dx_2^2 + ex_2^4 \ : \ \mathbb{R}^2 \rightarrow \mathbb{R} \qquad (57)$$

with the following constants $a = -2$, $b = 1.05$, $c = -\frac{1}{6}$, $d = -1$ and $e = 0$. With these constants, the inverted camelback function has a global minimum at $\mathbf{x}^* = [0\ \ 0]^T$, two local minima at $\mathbf{x} = [-1.7475\ \ 0.8737]^T$ and $\mathbf{x} = [1.7475\ \ -0.8737]^T$, and two saddle points at $\mathbf{x} = [-1.0705\ \ 0.5352]^T$ and $\mathbf{x} = [1.0705\ \ -0.5352]^T$. The value of the inverted camelback function at the global minimum is $\phi(\mathbf{x}^*) = 0$.

The algorithms were tested from three initial points, $\mathbf{x}_0 = [2\ 1]^T$, $\mathbf{x}_0 = [-1.5\ \ -1.5]^T$ and $\mathbf{x}_0 = [1.7475\ \ -0.8737]^T$, which corresponds to a local minimum. From the initial point $\mathbf{x}_0 = [2\ 1]^T$ the initial value of the control variable was chosen as $\mathbf{u}_0 = [0\ 0]^T$, from the initial point $\mathbf{x}_0 = [-1.5\ \ -1.5]^T$, the initial value of the control variable was chosen as $\mathbf{u}_0 = [1\ 1]^T$, and from the initial point $\mathbf{x}_0 = [1.7475\ \ -0.8737]^T$, the initial value of the control variable was chosen as $\mathbf{u}_0 = [-3\ 3]^T$.

Figure 8 shows the trajectories of the algorithms MI1, MI2, MI3, MI4, MI5 and DIN in the state space and the norms of the state variable $\|\mathbf{x}\|$ versus time from the initial point $\mathbf{x}_0 = [2\ 1]^T$ with initial value of the control variable $\mathbf{u}_0 = [0\ 0]^T$. The parameters used by the algorithms were chosen by hand tuning in such a way to try convergence to the global minimum. These parameters are reported in Table 4. Note that only the trajectory generated by MI1 algorithm achieves this goal, even though the convergence is slow (greater than $10\,s$).

Figure 9 shows the trajectories generated by the gHBF algorithms in the state space and the error norms of the trajectories versus time from the initial point $\mathbf{x}_0 = [2\ 1]^T$ with initial value of the control variable $\mathbf{u}_0 = [0\ 0]^T$. The parameter γ in the HBF algorithm was chosen as $\gamma = 1.7$, because this is the maximum value of this parameter for the trajectory to converge to the global minimum. In the other algorithms that have a constant friction parameter γ,

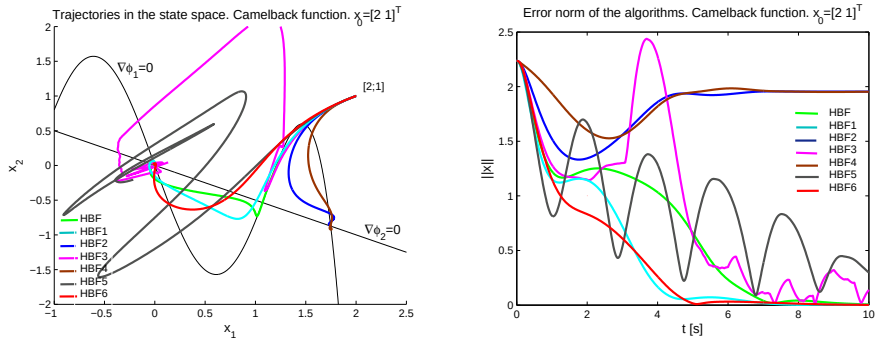

Figure 9: Trajectories in the state space and error norms vs. time with the inverted camelback function for the trajectories generated by the gHBF algorithms. Initial point $\mathbf{x}_0 = [2\ 1]^T$, initial value of the control variable $\mathbf{u}_0 = [0\ 0]^T$.

that is, HBF1, HBF2, HBF3 and HBF4, this parameter was also chosen as $\gamma = 1.7$ for comparative purposes. In the algorithms that have a parameter ϵ, it was chosen as $\epsilon = 0.1$. The other parameters were chosen by hand tuning in such a way to achieve the fastest convergence to the global minimum, in the cases where this convergence was possible. All the parameters used by the algorithms are reported in Table 4. Note that the trajectories generated by the algorithms HBF2 and HBF4 do not converge to the global minimum, but to the closer local one. It is possible that with another value of the parameter γ, different to $\gamma = 1.7$, the goal may be achieved, but this possibility was not tested. Note that the trajectories generated by the algorithms HBF1 and HBF6 converge to the global minimum faster than the trajectory generated by the HBF algorithm.

Figure 10 shows the trajectories generated by the CG algorithms in the state space and the error norms of the trajectories versus time from the initial point $\mathbf{x}_0 = [2\ 1]^T$ with initial value of the control variable $\mathbf{u}_0 = [0\ 0]^T$. In the algorithms that have a constant parameter β, i.e. CG1, CG2, CG3 and CG4, it was chosen as $\beta = 1.7$, the same value of γ used in the gHBF algorithms, for comparative purposes with respect to the curves showed in figure 9. In the algorithms that have a parameter ϵ (CG1, CG3 and CG5), it was chosen as $\epsilon = 0.1$. The other parameters were chosen by hand tuning in such a way to achieve the fastest convergence to the global minimum. These parameters are reported in Table 4. Note that all the trajectories converge to the global minimum, and faster than the trajectories generated by the gHBF, MI and DIN algorithms from the same initial point.

Figure 11 shows the trajectories generated by the algorithms MI1, MI2, MI3, MI4, MI5 and DIN in the state space and the error norms versus time from the initial point $\mathbf{x}_0 = [-1.5\ \ -1.5]^T$ with initial value of the control variable $\mathbf{u}_0 = [1\ 1]^T$. The values of the parameters used by the algorithms were chosen by hand tuning in such a way to try convergence to the global minimum. These parameters are reported in Table 4. Note that MI3 and MI5 do not achieve convergence to the global minimum, but to the closer local one.

Figure 12 shows the trajectories generated by the gHBF algorithms in the state space and the error norms of the trajectories versus time from the initial

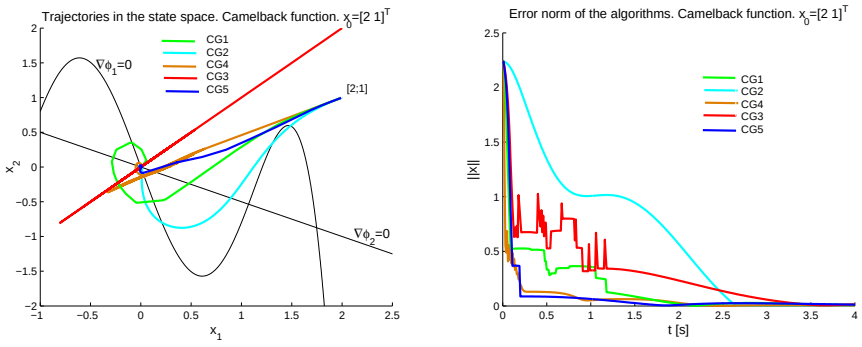

Figure 10: Trajectories in the state space and error norms vs. time with the inverted camelback function for the trajectories generated by the CG algorithms. Initial point $\mathbf{x}_0 = [2\ 1]^T$, initial value of the control variable $\mathbf{u}_0 = [0\ 0]^T$.

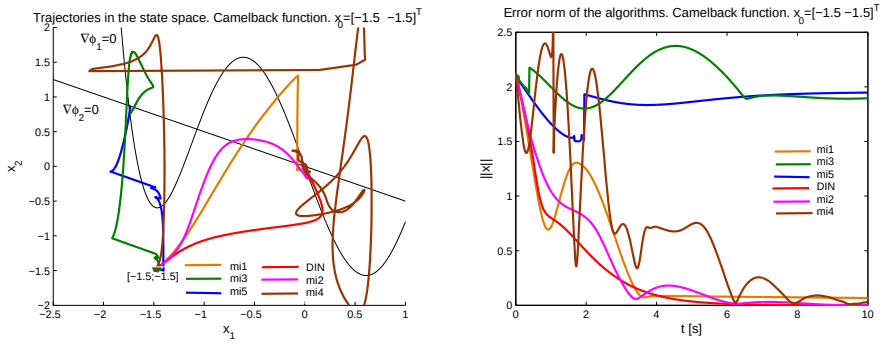

Figure 11: Trajectories in the state space and error norms vs. time with the inverted camelback function for the trajectories generated by the algorithms MI1, MI2, MI3, MI4, MI5 and DIN. Initial point $\mathbf{x}_0 = [-1.5\ \ -1.5]^T$, initial value of the control variable $\mathbf{u}_0 = [1\ 1]^T$.

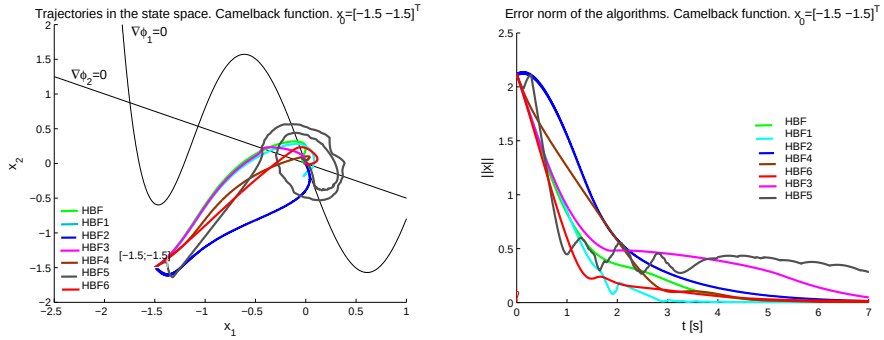

Figure 12: Trajectories in the state space and error norms vs. time with the inverted camelback function for the trajectories generated by the gHBF algorithms. Initial point $\mathbf{x}_0 = [-1.5 \ \ -1.5]^T$, initial value of the control variable $\mathbf{u}_0 = [1 \ 1]^T$.

point $\mathbf{x}_0 = [-1.5 \ \ -1.5]^T$ and with an initial value of the control variable $\mathbf{u}_0 = [1 \ 1]^T$. In all the algorithms that have a constant γ parameter, that is, HBF, HBF1, HBF2, HBF3 and HBF4, it was chosen as $\gamma = 2.2$ because this is the value (tested by hand tuning) which achieve the fastest convergence to the global minimum of the trajectory generated by the HBF algorithm. The parameter ϵ was chosen as $\epsilon = 0.1$ in all the algorithms that use this parameter (HBF1, HBF3, HBF5 and HBF6). The other parameters were chosen, also by hand tuning, in such a way to achieve the fastest convergence to the global minimum. These parameters are reported in Table 4. The trajectories generated by all the gHBF algorithms converge to the global minimum from this initial point, but only the one generated by HBF1 is very much faster than the one generated by the HBF algorithm.

Figure 13 shows the trajectories generated by the CG algorithms in the state space and the error norms of the trajectories versus time from the initial point $\mathbf{x}_0 = [-1.5 \ -1.5]^T$ with initial value of the control variable $\mathbf{u}_0 = [1 \ 1]^T$. In the algorithms that have a constant parameter β, that is CG1, CG2, CG3 and CG4, it was chosen as $\beta = 2.2$, the same value of γ used in the gHBF algorithms from this initial point, for comparative purposes. In all the cases, in the algorithms that have a parameter ϵ, it was chosen as $\epsilon = 0.1$. The other parameters were chosen by hand tuning in such a way to achieve the fastest convergence to the global minimum. These parameters are reported in Table 4. All the trajectories converge to the global minimum from this initial point, and the ones generated by CG1, CG2 and CG4 converge very much faster than their gHBF, DIN and MI competitors.

Figure 14 shows the trajectories generated by the algorithms MI1, MI2, MI3, MI4, MI5 and DIN in the state space and the error norms of the trajectories versus time from the initial point $\mathbf{x}_0 = [1.7475 \ \ -0.8737]^T$ and with an initial value of the control variable $\mathbf{u}_0 = [-3 \ 3]^T$. The parameters were chosen by hand tuning in such a way to achieve convergence to the global minimum, in the cases where it was possible. These parameters are reported in Table 4. Note that only the trajectories generated by MI1, MI2 and DIN converge to the global minimum.

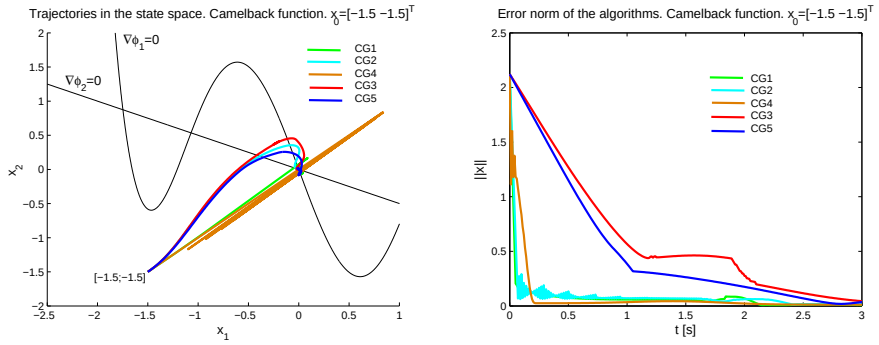

Figure 13: Trajectories in the state space and error norms vs. time with the inverted camelback function for the trajectories generated by the CG algorithms. Initial point $\mathbf{x}_0 = [-1.5 \ -1.5]^T$, initial value of the control variable $\mathbf{u}_0 = [1 \ 1]^T$.

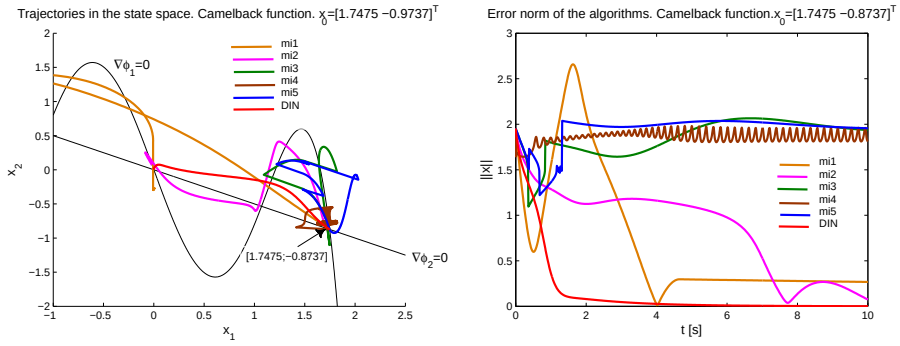

Figure 14: Trajectories in the state space and error norms vs. time with the inverted camelback function for the trajectories generated by the algorithms MI1, MI2, MI3, MI4, MI5 and DIN. Initial point $\mathbf{x}_0 = [1.7475 \ -0.8737]^T$, initial value of the control variable $\mathbf{u}_0 = [-3 \ 3]^T$.

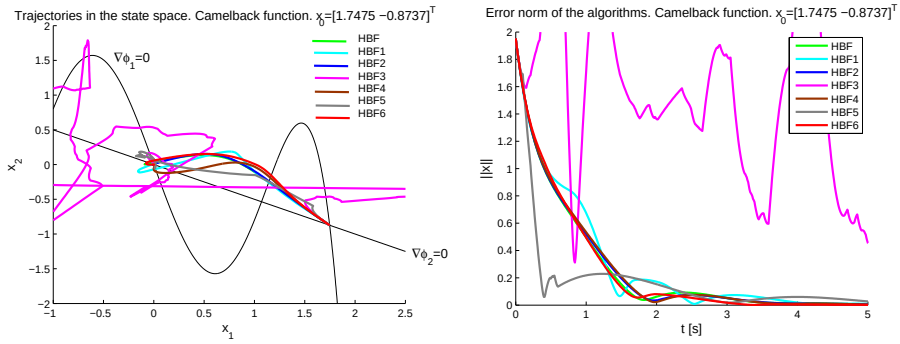

Figure 15: Trajectories in the state space and error norms vs. time with the inverted camelback function for the trajectories generated by the gHBF algorithms. Initial point $\mathbf{x}_0 = [1.7475\ -0.8737]^T$, initial value of the control variable $\mathbf{u}_0 = [-3\ 3]^T$.

Figure 15 shows the trajectories generated by the gHBF algorithms in the state space and the error norms of the trajectories versus time from the initial point $\mathbf{x}_0 = [1.7475\ -0.8737]^T$ and with an initial value of the control variable $\mathbf{u}_0 = [-3\ 3]^T$. In all the algorithms that have a constant parameter γ, that is, HBF, HBF1, HBF2, HBF3 and HBF4, it was chosen as $\gamma = 2.3$ because this is the value (tested by hand tuning) to achieve the fastest convergence to the global minimum of the HBF algorithm. The parameter ϵ was chosen as $\epsilon = 0.1$ in all the algorithms that use this parameter (HBF1, HBF3, HBF5 and HBF6). The other parameters were chosen, also by hand tuning, in such a way to achieve the fastest convergence to the global minimum. These parameters are reported in Table 4. The trajectories generated by all the gHBF algorithms converge to the global minimum from this initial point, but only the one generated by HBF6 is a little faster than the one generated by the HBF algorithm. In HBF2 and HBF4, the parameter κ was chosen as $\kappa = 0.1$, i.e. these algorithms becomes similar to the HBF algorithm.

Figure 16 shows the trajectories generated by the CG algorithms in the state space and the error norms of the trajectories versus time from the initial point $\mathbf{x}_0 = [1.7475\ -0.8737]^T$ and with an initial value of the control variable $\mathbf{u}_0 = [-3\ 3]^T$. In all the algorithms that have a constant parameter β, that is, CG1, CG2, CG3 and CG4, it was chosen as $\beta = 2.3$ because this is the value of the friction parameter γ used in the gHBF algorithms from this initial point. The parameter ϵ was chosen as $\epsilon = 0.1$ in all the algorithms that use this parameter. The other parameters were chosen, also by hand tuning, in such a way to achieve the fastest convergence to the global minimum. These parameters are reported in Table 4. The trajectories generated by all the CG algorithms converge to the global minimum from this initial point, and those generated by CG1, CG2 and CG5, converge very much faster than the one generated by the HBF algorithm.

Table 4 reports the parameters used with the inverted camelback function.

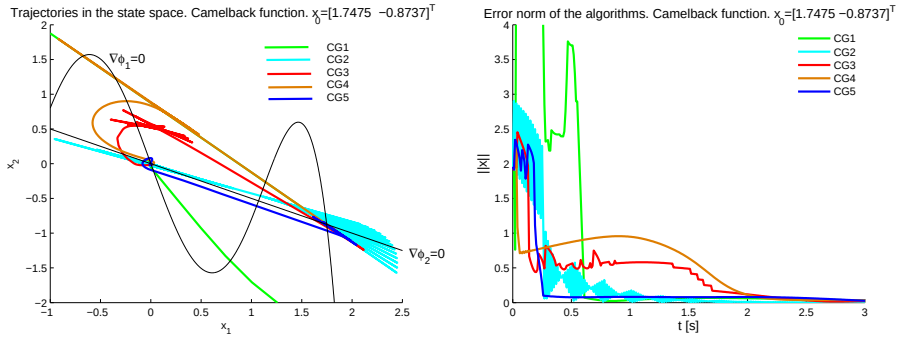

Figure 16: Trajectories in the state space and error norms vs. time with the inverted camelback function for the trajectories generated by the CG algorithms. Initial point $\mathbf{x}_0 = [1.7475 \;\; -0.8737]^T$, initial value of the control variable $\mathbf{u}_0 = [-3 \; 3]^T$.

Table 4: Parameters used by the algorithms in the experiments reported in section 4.2 with the inverted camelback function.

	Camelback		
	$\mathbf{x}_0 = [2\ 1]^T \quad \mathbf{u}_0 = [0\ 0]^T$	$\mathbf{x}_0 = [-1.5 \;\; -1.5]^T \quad \mathbf{u}_0 = [1\ 1]^T$	$\mathbf{x}_0 = [1.7475 \;\; -0.8737]^T \quad \mathbf{u}_0 = [-3\ 3]^T$
MI1	$\kappa = 0.4$	$\kappa = 1.2$	$\kappa = 0.97$
MI2	$\kappa = 0.1$	$\kappa = 0.5$	$\kappa = 0.4$
MI3	$\kappa = 0.1$	$\kappa = 0.4$	$\kappa = 0.7$
MI4	$\kappa = 12$	$\kappa = 1.3$	$\kappa = 14$
MI5	$\gamma = 0.5, \quad a = 3, \quad b = 0.7$	$\gamma = 0.2, \quad a = 1, \quad b = 0.7$	$\gamma = 0.2, \quad a = 0.5, \quad b = 0.8$
DIN	$a = 1, \quad b = 1$	$a = 1, \quad b = 1$	$a = 2.2, \quad b = 1.1$
HBF	$\gamma = 1.7$	$\gamma = 2.2$	$\gamma = 2.3$
HBF1	$\gamma = 1.7, \quad \epsilon = 0.1$ $\mu = \begin{cases} 1 - \gamma \frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi\mathbf{u}} + 20 & \nabla^T\phi\mathbf{u} > \epsilon \\ 1.2 & \nabla^T\phi\mathbf{u} < -\epsilon \\ 1 & \lvert\nabla^T\phi\mathbf{u}\rvert \leq \epsilon \end{cases}$	$\gamma = 2.2, \quad \epsilon = 0.1$ $\mu = \begin{cases} 1 - \gamma \frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi\mathbf{u}} + 100 & \nabla^T\phi\mathbf{u} > \epsilon \\ 1 - \gamma \frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi\mathbf{u}} - 1 & \nabla^T\phi\mathbf{u} < -\epsilon \\ 1 & \lvert\nabla^T\phi\mathbf{u}\rvert \leq \epsilon \end{cases}$	$\gamma = 2.3 \quad \epsilon = 0.1$ $\mu = \begin{cases} 1 - \gamma \frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi\mathbf{u}} + 10 & \nabla^T\phi\mathbf{u} > \epsilon \\ 2.8 & \nabla^T\phi\mathbf{u} < -\epsilon \\ 1 & \lvert\nabla^T\phi\mathbf{u}\rvert \leq \epsilon \end{cases}$
HBF2	$\gamma = 1.7, \quad \kappa = 0.5$	$\gamma = 2.2, \quad \kappa = 85$	$\gamma = 2.3, \quad \kappa = 0.1$
HBF3	$\gamma = 1.7, \quad \kappa = 0.5, \quad \epsilon = 0.1$	$\gamma = 2.2, \quad \kappa = 0.6, \quad \epsilon = 0.1$	$\gamma = 2.3, \quad \kappa = 37.4, \quad \epsilon = 0.1$
HBF4	$\gamma = 1.7, \quad \kappa = 0.5$	$\gamma = 2.2, \quad \kappa = 0.2$	$\gamma = 2.3, \quad \kappa = 0.1$
HBF5	$a = 0.1, \quad b = 0.5, \quad \epsilon = 0.1$	$a = 10, \quad b = 0.1, \quad \epsilon = 0.1$	$a = 90, \quad b = 0.1, \quad \epsilon = 0.1$
HBF6	$\epsilon = 0.1$ $\mu_1 = 1.5, \quad \mu_2 = 0.6$ $a = 1.7$	$\epsilon = 0.1$ $\mu_1 = 10, \quad \mu_2 = 0.3$ $a = 2$	$\epsilon = 0.1$ $\mu_1 = 5.8, \quad \mu_2 = 0.9$ $a = 2.2$
CG1	$\beta = 1.7, \quad \epsilon = 0.1$ $\alpha = \begin{cases} 1 + \beta \frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi\mathbf{u}} - 1 & \nabla^T\phi\mathbf{u} > \epsilon \\ 1 + \beta \frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi\mathbf{u}} + 100 & \nabla^T\phi\mathbf{u} < -\epsilon \\ 1 & \lvert\nabla^T\phi\mathbf{u}\rvert \leq \epsilon \end{cases}$	$\beta = 2.2, \quad \epsilon = 0.1$ $\alpha = \begin{cases} 1 + \beta \frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi\mathbf{u}} - 10 & \nabla^T\phi\mathbf{u} > \epsilon \\ 27 & \nabla^T\phi\mathbf{u} < -\epsilon \\ 1 & \lvert\nabla^T\phi\mathbf{u}\rvert \leq \epsilon \end{cases}$	$\beta = 2.3, \quad \epsilon = 0.1$ $\alpha = \begin{cases} 1 + \beta \frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi\mathbf{u}} - 8 & \nabla^T\phi\mathbf{u} > \epsilon \\ 135 & \nabla^T\phi\mathbf{u} < -\epsilon \\ 1 & \lvert\nabla^T\phi\mathbf{u}\rvert \leq \epsilon \end{cases}$
CG2	$\beta = 1.7, \quad \kappa = 1$	$\beta = 2.2, \quad \kappa = 22$	$\beta = 2.3, \quad \kappa = 24.2$
CG3	$\beta = 1.7, \quad \epsilon = 0.1, \quad \kappa = 40$	$\beta = 2.2, \quad \epsilon = 0.1, \quad \kappa = 1$	$\beta = 2.3, \quad \epsilon = 0.1, \quad \kappa = 4.2$
CG4	$\beta = 1.7, \quad \kappa = 1580$	$\beta = 2.2, \quad \kappa = 42$	$\beta = 2.3, \quad \kappa = 9.2$
CG5	$\epsilon = 0.1, \quad a = 50, \quad b = 70$	$\epsilon = 0.1, \quad a = 2, \quad b = 5$	$a = 12, \quad b = 80, \quad \epsilon = 0.1$

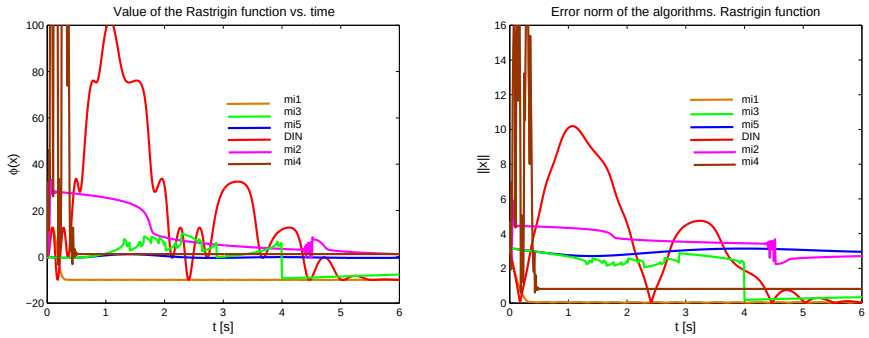

Figure 17: Value of the Rastrigin function and error norms vs. time for the trajectories generated by the algorithms MI1, MI2, MI3, MI4, MI5 and DIN. Initial point $x_{0_i} = -1$, initial value of the control variable $u_{0_i} = 6$ for all $i \in \{1, \cdots, 10\}$.

4.3 Simulations with the Rastrigin function

The Rastrigin function, popular in the global optimization literature, was chosen as objective function in this section:

$$\phi(\mathbf{x}) = \sum_{i=1}^{N} x_i^2 - \cos(2\pi x_i) \ : \ \mathbb{R}^N \to \mathbb{R} \tag{58}$$

This function has several local minima and a unique global minimum at $\mathbf{x}^* = \mathbf{0}$. The value of the function at the minimum is $\phi(\mathbf{x}^*) = -N$, where N is the number of variables. This number was chosen as $N = 10$ for the numerical experiments reported here. In all the experiments, the initial point was chosen as $x_{0_i} = -1$ and the initial value of the control variable as $u_{0_i} = 6$ for all $i \in \{1, \cdots, N\}$.

Figure 17 shows the value of the Rastrigin function versus time and the error norms of the trajectories versus time for the trajectories generated by the algorithms MI1, MI2, MI3, MI4, MI5 and DIN. The parameters used by the algorithms were chosen by hand tuning in such a way to achieve the fastest convergence to the global minimum, in the cases where convergence was possible. These parameters are reported in Table 5. Note that only the trajectories generated by MI1, MI3 and DIN converge to the global minimum, the other trajectories converge to a local minimum, and only the trajectory generated by MI1 presents a fast convergence.

Figure 18 shows the value of the function and the error norms versus time of the trajectories generated by the gHBF algorithms. In the algorithms that have a constant friction parameter γ (HBF, HBF1, HBF2, HBF3 and HBF4) it was chosen as $\gamma = 5$. The other parameters were chosen by hand tuning in such a way to achieve the fastest convergence to the global minimum. These parameters are reported in Table 5. In algorithms that have a parameter ϵ (HBF1, HBF3, HBF5 and HBF6), it was chosen as $\epsilon = 0.1$. It was not possible to find parameters of the HBF5 algorithm to achieve trajectory convergence to a minimum, and the trajectory generated by HBF4 converges to a local minimum. The trajectories generated by HBF1, HBF2, HBF3 and HBF6 converge faster

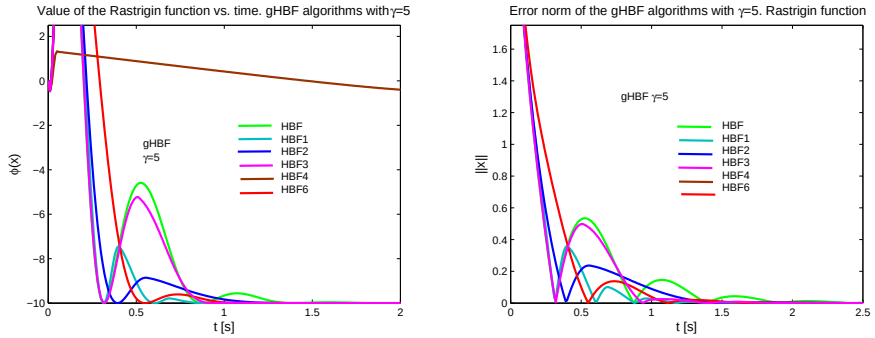

Figure 18: Value of the Rastrigin function and error norms vs. time for the trajectories generated by the gHBF algorithms with parameter $\gamma = 5$. Initial point $x_{0_i} = -1$, initial value of the control variable $u_{0_i} = 6$ for all $i \in \{1, \cdots, 10\}$.

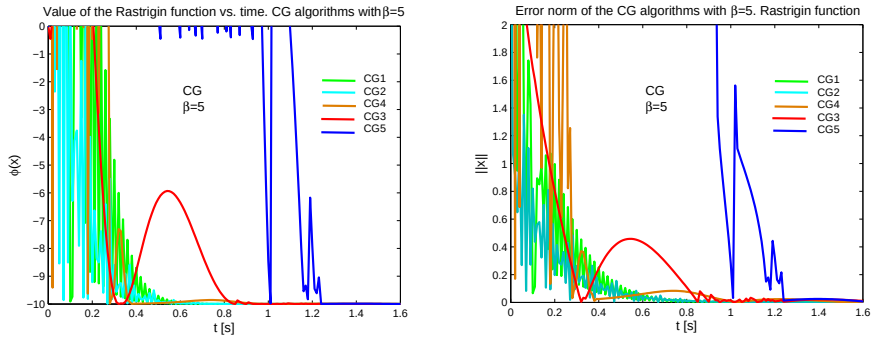

Figure 19: Value of the Rastrigin function and error norms vs. time for the trajectories generated by the CG algorithms with parameter $\beta = 5$. Initial point $x_{0_i} = -1$, initial value of the control variable $u_{0_i} = 6$ for all $i \in \{1, \cdots, 10\}$.

than the one generated by the HBF algorithm.

Figure 19 shows the value of the function and the error norms versus time of the trajectories generated by the CG algorithms. In the algorithms that have a constant parameter β (CG1, CG2, CG3 and CG4), it was chosen as $\beta = 5$ to compare with the trajectories generated by the algorithms gHBF with the same value of the parameter γ, which are showed in figure 18. In the algorithms that have a parameter ϵ (CG1, CG3 and CG5), it was chosen as $\epsilon = 0.1$. Also here, the other parameters were chosen by hand tuning to achieve the fastest convergence to the global minimum. Note that all the trajectories converge to the global minimum and faster than the ones generated by the gHBF algorithms, specially those generated by CG1 and CG2 .

Figure 20 shows the value of the function and the error norms versus time of the trajectories generated by the gHBF algorithms, from the same initial point $x_{0_i} = -1$ and with the same initial value of the control variable $u_{0_i} = 6$ for all $i \in \{1, \cdots, 10\}$, but in this case, in the algorithms that have a constant friction parameter γ (HBF, HBF1, HBF2, HBF3 and HBF4), it was chosen as $\gamma = 10$. In the algorithms that have a parameter ϵ (HBF1, HBF3, HBF5 and HBF6),

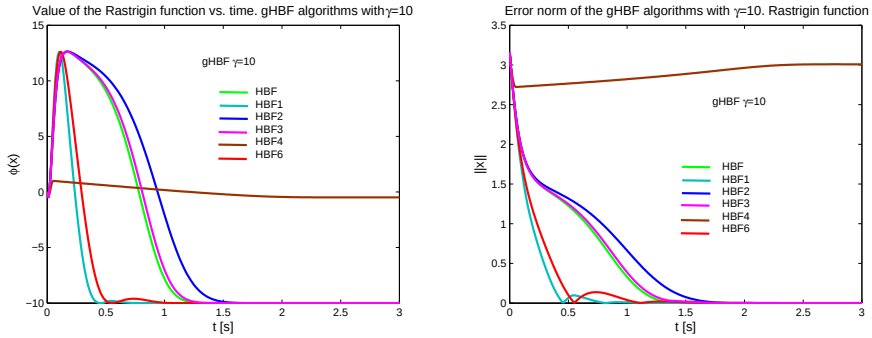

Figure 20: Value of the Rastrigin function and error norms vs. time for the trajectories generated by the gHBF algorithms with parameter $\gamma = 10$. Initial point $x_{0_i} = -1$, initial value of the control variable $u_{0_i} = 6$ for all $i \in \{1, \cdots, 10\}$.

it was chosen as $\epsilon = 0.1$. Also here, the other parameters were chosen by hand tuning in such a way to achieve the fastest convergence to the global minimum. These parameters are reported in Table 5. The trajectory generated by HBF6 is the same showed in figure 18, because this algorithm also has not a constant parameter γ, and the curves were repeated here only for comparative purposes. The trajectories generated by the algorithms HBF1 and HBF6 converge to the global minimum faster than the one generated by the HBF with this value of the parameter γ.

Figure 21 shows the value of the function and the error norms versus time of the trajectories generated by the CG algorithms. In the algorithms that have a constant parameter β (CG1, CG2, CG3 and CG4), it was chosen as $\beta = 10$, for comparative purposes with respect to the trajectories generated by the gHBF algorithms with the same value of the parameter γ and reported in figure 20. In the algorithms that have a parameter ϵ (CG1, CG3 and CG5), it was chosen as $\epsilon = 0.1$. Here again, the other parameters were chosen by hand tuning in such a way to achieve the fastest convergence to the global minimum. These parameters are reported in Table 5. Also here, CG5 algorithm does not have a constant parameter β, so the curves corresponding to its trajectory are the same that those showed in figure 19 and they are repeated here for comparative purposes. Note that the convergence of the trajectories is faster than the convergence of the ones generated by the gHBF algorithms with the same value of the parameter γ, which are showed in figure 20, specially the ones generated by CG1 and CG2.

Table 5 reports the parameters used by the algorithms with the Rastrigin function.

4.4 Simulations with the Ackley function

The Ackley function can be expressed as:

$$\phi(\mathbf{x}) = -20\exp^{-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}} - \exp^{\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i)} \; : \; \mathbb{R}^N \to \mathbb{R} \qquad (59)$$

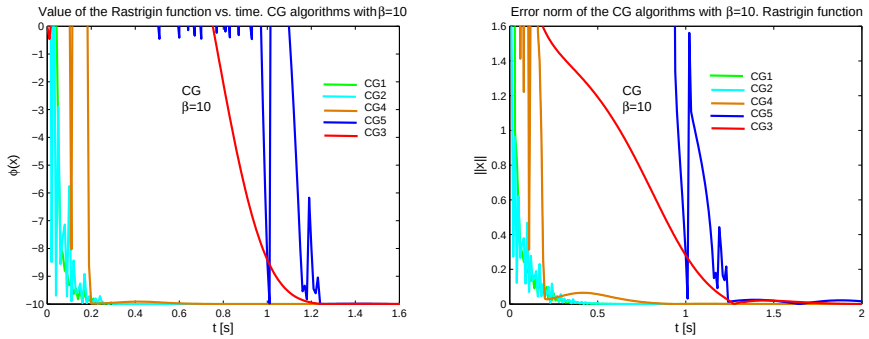

Figure 21: Value of the Rastrigin function and error norms vs. time for the trajectories generated by the CG algorithms with parameter $\beta = 10$. Initial point $x_{0_i} = -1$, initial value of the control variable $u_{0_i} = 6$ for all $i \in \{1, \cdots, 10\}$.

Table 5: Parameters used by the algorithms in the experiments reported in section 4.3 with the Rastrigin function, where - means that the trajectory does not converge to a minimum in $10\,s$.

	Rastrigin	
MI1	$\kappa = 20$	
MI2	$\kappa = 0.2$	
MI3	$\kappa = 0.2$	
MI4	$\kappa = 20$	
MI5	$\gamma = 1, \;\; a = 20, \;\; b = 20$	
DIN	$a = 0.1, \;\; b = 0.1$	
HBF	$\gamma = 5$	$\gamma = 10$
HBF1	$\gamma = 5, \;\; \epsilon = 0.1$ $\mu = \begin{cases} 1 - \gamma \frac{\mathbf{u}^T \mathbf{u}}{\nabla^T \phi \mathbf{u}} + 9 & \nabla^T \phi \mathbf{u} > \epsilon \\ 1.7 & \nabla^T \phi \mathbf{u} < -\epsilon \\ 1 & \lvert \nabla^T \phi \mathbf{u} \rvert \leq \epsilon \end{cases}$	$\gamma = 10, \;\; \epsilon = 0.1$ $\mu = \begin{cases} 1 - \gamma \frac{\mathbf{u}^T \mathbf{u}}{\nabla^T \phi \mathbf{u}} + 7 & \nabla^T \phi \mathbf{u} > \epsilon \\ 1.7 & \nabla^T \phi \mathbf{u} < -\epsilon \\ 1 & \lvert \nabla^T \phi \mathbf{u} \rvert \leq \epsilon \end{cases}$
HBF2	$\gamma = 5, \;\; \kappa = 0.7$	$\gamma = 10, \;\; \kappa = 0.2$
HBF3	$\gamma = 5, \;\; \kappa = 4, \;\; \epsilon = 0.1$	$\gamma = 10, \;\; \kappa = 0.8, \;\; \epsilon = 0.1$
HBF4	$\gamma = 5, \;\; \kappa = 1$	$\gamma = 10, \;\; \kappa = 1$
HBF5	-	
HBF6	$\epsilon = 0.1$ $\mu_1 = 10, \;\; \mu_2 = 0.3$ $a = 7$	
CG1	$\beta = 5, \;\; \epsilon = 0.1$ $\alpha = \begin{cases} 1 + \beta \frac{\mathbf{u}^T \mathbf{u}}{\nabla^T \phi \mathbf{u}} - 20 & \nabla^T \phi \mathbf{u} > \epsilon \\ 1 + \beta \frac{\mathbf{u}^T \mathbf{u}}{\nabla^T \phi \mathbf{u}} + 20 & \nabla^T \phi \mathbf{u} < -\epsilon \\ 1 & \lvert \nabla^T \phi \mathbf{u} \rvert \leq \epsilon \end{cases}$	$\beta = 10, \;\; \epsilon = 0.1$ $\alpha = \begin{cases} 1 + \beta \frac{\mathbf{u}^T \mathbf{u}}{\nabla^T \phi \mathbf{u}} - 9 & \nabla^T \phi \mathbf{u} > \epsilon \\ 1 + \beta \frac{\mathbf{u}^T \mathbf{u}}{\nabla^T \phi \mathbf{u}} + 10 & \nabla^T \phi \mathbf{u} < -\epsilon \\ 1 & \lvert \nabla^T \phi \mathbf{u} \rvert \leq \epsilon \end{cases}$
CG2	$\beta = 5, \;\; \kappa = 10$	$\beta = 10, \;\; \kappa = 7$
CG3	$\beta = 5, \;\; \epsilon = 0.1, \;\; \kappa = 1.5$	$\beta = 10, \;\; \epsilon = 0.1, \;\; \kappa = 0.2$
CG4	$\beta = 5, \;\; \kappa = 0.4$	$\beta = 10, \;\; \kappa = 0.5$
CG5	$\epsilon = 0.1, \;\; a = 200, \;\; b = 100$	

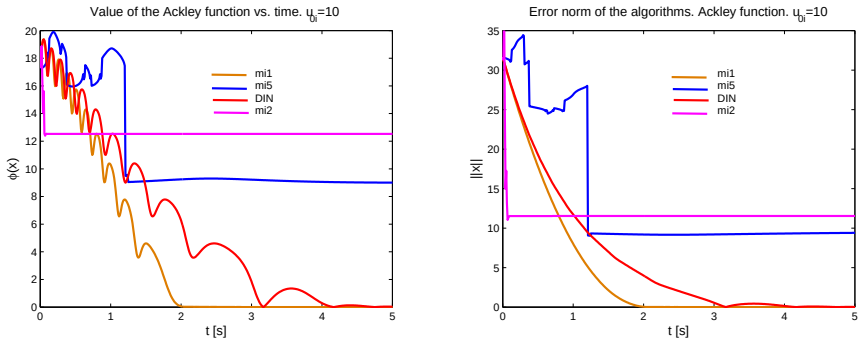

Figure 22: Value of the Ackley function and error norms vs. time for the trajectories generated by the algorithms MI1, MI2, MI3, MI4, MI5 and DIN. Initial point $x_{0_i} = -10$, initial value of the control variable $u_{0_i} = 10$ for all $i \in \{1, \cdots, 10\}$.

where N is the number of the variables which was chosen as $N = 10$ for the numerical experiments reported here.

The Ackley function has several local minima and a unique global minimum at $\mathbf{x}^* = \mathbf{0}$. The value of the function at the global minimum is $\phi(\mathbf{x}^*) = 0$. In all the experiments, the initial point was chosen as $x_{0_i} = -10$ and the initial value of the control variable as $u_{0_i} = 10$, and $u_{0_i} = 20$ for all $i \in \{1, \cdots, N\}$.

Figure 22 shows the value of the Ackley function versus time and the error norms of the trajectories versus time for the trajectories generated by the algorithms MI1, MI2, MI3, MI4, MI5 and DIN for the initial value of the control variable $u_{0_i} = 10$. The parameters were chosen by hand tuning in such a way to achieve convergence to the global minimum, in the cases where it was possible, and to achieve the fastest convergence. These parameters are reported in Table 6. Note that only the trajectories generated by MI1 and DIN converge to the global minimum, whereas trajectories generated by MI5 and MI2 converge to local minima. It was not possible to find parameters by hand tuning to achieve convergence of the trajectories generated by MI3 and MI4.

Figure 23 shows the value of the Ackley function and the error norms versus time for the trajectories generated by the gHBF algorithms for the initial value of the control, variable $u_{0_i} = 10$. In the algorithms that have a constant parameter γ (HBF, HBF1, HBF2, HBF3 and HBF4) it was chosen as $\gamma = 1$. In the algorithms that have a parameter ϵ (HBF1, HBF3, HBF5 and HBF6), it was chosen as $\epsilon = 0.1$. The other parameters were chosen by hand tuning in such a way to achieve the fastest convergence to the global minimum. These parameters are reported in Table 6. Note that the trajectories generated by HBF4 and HBF5 converge to local minima. Another once, it is possible that with a different value of the parameter γ, the trajectory generated by HBF4 converges to the global minimum, but this possibility was not tested. The trajectories generated by HBF1, HBF2, HBF3 and HBF6 converge to the global minimum faster than the one generated by HBF.

Figure 24 shows the value of the function and the error norms versus time for the trajectories generated by the CG algorithms for the initial value of the control variable $u_{0_i} = 10$. In the algorithms that have a constant parameter β

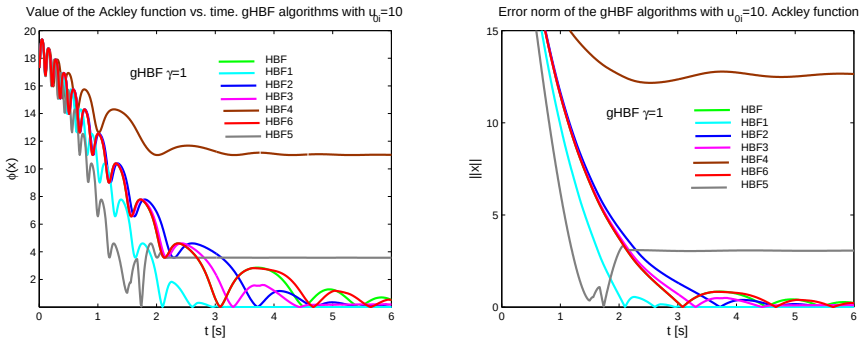

Figure 23: Value of the Ackley function and error norms vs. time for the trajectories generated by the gHBF algorithms with $\gamma = 1$. Initial point $x_{0_i} = -10$, initial value of the control variable $u_{0_i} = 10$ for all $i \in \{1, \cdots, 10\}$.

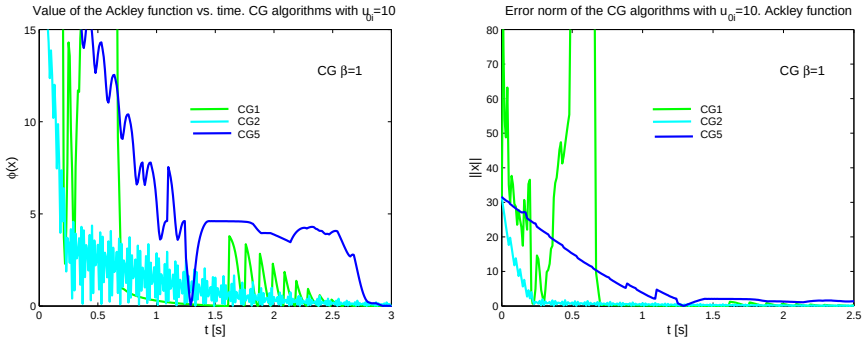

Figure 24: Value of the Ackley function and error norms vs. time for the trajectories generated by the CG algorithms with $\beta = 1$. Initial point $x_{0_i} = -10$, initial value of the control variable $u_{0_i} = 10$ for all $i \in \{1, \cdots, 10\}$.

(CG1, CG2, CG3 and CG4), it was chosen as $\beta = 1$ for comparative purposes with respect to the trajectories generated by the gHBF algorithms with the same value of the parameter γ, which are showed in figure 23. In the algorithms that have a parameter ϵ (CG1, CG3 and CG5), it was chosen as $\epsilon = 0.1$. The other parameters were chosen to try the fastest convergence to the global minimum and they are reported in Table 6. Note that, with this value of β, only trajectories generated by CG1, CG2 and CG5 (which does not have a constant β) achieve convergence to the global minimum. Of course, it is possible that with another value of the parameter β, trajectories generated by CG3 and CG4 may converge, but this possibility was not tested. However, such as happens with the experiments reported in the former subsections, the convergence of the trajectories generated by the CG algorithms is faster than the convergence of their gHBF, MI and DIN competitors.

Figure 25 shows the value of the Ackley function versus time and the error norms of the trajectories versus time for the trajectories generated by the algorithms MI1, MI2, MI3, MI4, MI5 and DIN for the initial value of the control variable $u_{0_i} = 20$. The parameters were chosen by hand tuning in such a way

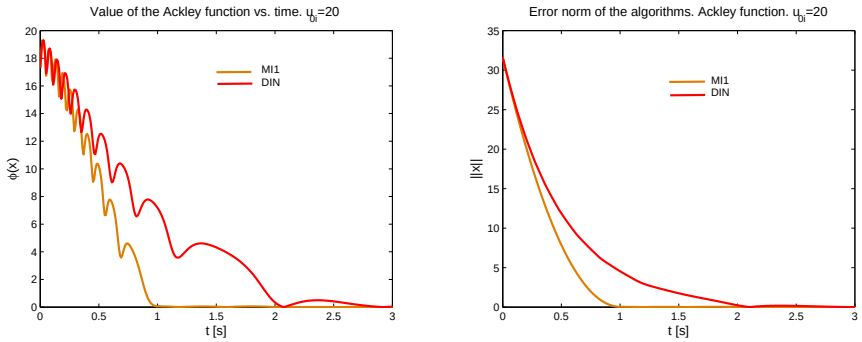

Figure 25: Value of the Ackley function and error norms vs. time for the trajectories generated by the algorithms MI1, MI2, MI3, MI4, MI5 and DIN. Initial point $x_{0_i} = -10$, initial value of the control variable $u_{0_i} = 20$ for all $i \in \{1, \cdots, 10\}$.

to achieve convergence to the global minimum, in the cases where it was possible, and to achieve the fastest convergence. These parameters are reported in Table 6. Note that only the trajectories generated by MI1 and DIN converge to the global minimum. It was not possible to find parameters by hand tuning to achieve convergence of the trajectories generated by the other algorithms.

Figure 26 shows the value of the Ackley function and the error norms versus time for the trajectories generated by the gHBF algorithms for the initial value of the control, variable $u_{0_i} = 20$. In the algorithms that have a constant parameter γ (HBF, HBF1, HBF2, HBF3 and HBF4) it was chosen as $\gamma = 2$. In the algorithms that have a parameter ϵ (HBF1, HBF3, HBF5 and HBF6), it was chosen as $\epsilon = 0.1$. The other parameters were chosen by hand tuning in such a way to achieve the fastest convergence to the global minimum. These parameters are reported in Table 6. It was not possible to find parameters by hand tuning to achieve convergence to the global minimum for the algorithms HBF4 and HBF5. Another once, it is possible that with a different value of the parameter γ, the trajectory generated by HBF4 converges to the global minimum, but this possibility was not tested. All the trajectories that converge to the global minimum do it faster than the trajectory generated by the HBF algorithm.

Figure 27 shows the value of the function and the error norms versus time for the trajectories generated by the CG algorithms for the initial value of the control variable $u_{0_i} = 20$. In the algorithms that have a constant parameter β (CG1, CG2, CG3 and CG4), it was chosen as $\beta = 2$ for comparative purposes with respect to the trajectories generated by the gHBF algorithms with the same value of the parameter γ, which are showed in figure 26. In the algorithms that have a parameter ϵ (CG1, CG3 and CG5), it was chosen as $\epsilon = 0.1$. The other parameters were chosen to try the fastest convergence to the global minimum and they are reported in Table 6. Note that all the trajectories converge to the global minimum and very much faster than their gHBF, MI and DIN competitors

Table 6 reports the parameters used by the algorithms with the Ackley function.

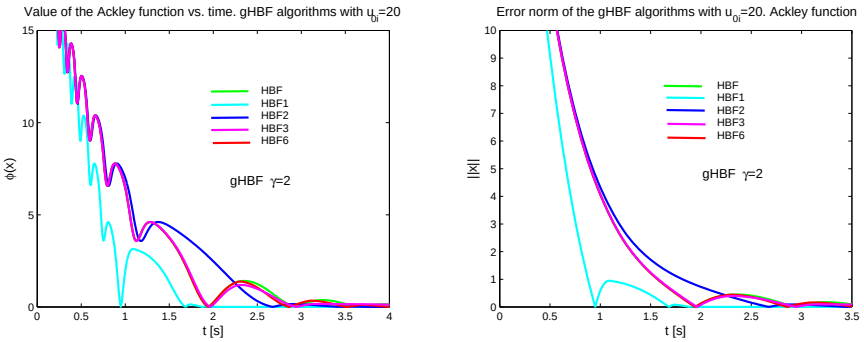

Figure 26: Value of the Ackley function and error norms vs. time for the trajectories generated by the gHBF algorithms with $\gamma = 2$. Initial point $x_{0_i} = -10$, initial value of the control variable $u_{0_i} = 20$ for all $i \in \{1, \cdots, 10\}$.

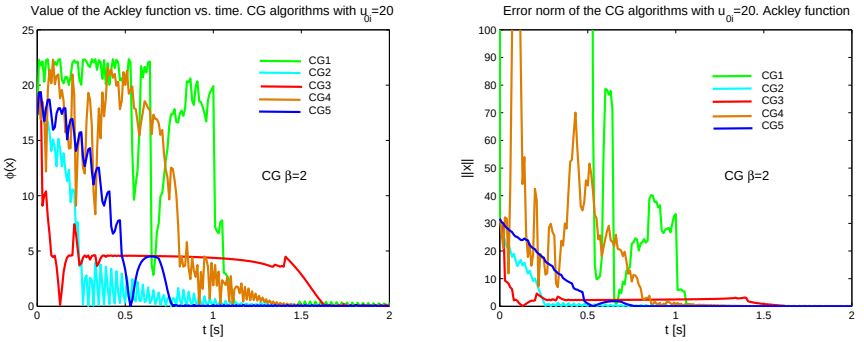

Figure 27: Value of the Ackley function and error norms vs. time for the trajectories generated by the CG algorithms with $\beta = 2$. Initial point $x_{0_i} = -10$, initial value of the control variable $u_{0_i} = 20$ for all $i \in \{1, \cdots, 10\}$.

Table 6: Parameters used by the algorithms in the experiments reported in section 4.4 with the Ackley function, where - means that the trajectory does not converge to a minimum in $10\,s$.

	Ackley	
	$u_{0_i} = 10 \;\; \forall i \in \{1, \cdots, 10\}$	$u_{0_i} = 20 \;\; \forall i \in \{1, \cdots, 10\}$
MI1	$\kappa = 5.2$	$\kappa = 20.4$
MI2	$\kappa = 18$	-
MI3	-	-
MI4	-	-
MI5	$\gamma = 0.4, \;\; a = 0.5, \;\; b = 1$	-
DIN	$a = 1, \;\; b = 0.3$	$a = 2, \;\; b = 0.4$
HBF	$\gamma = 1$	$\gamma = 2$
HBF1	$\gamma = 1, \;\; \epsilon = 0.1$ $\mu = \begin{cases} 1 - \gamma \frac{\mathbf{u}^T \mathbf{u}}{\nabla^T \phi \mathbf{u}} + 11 & \nabla^T \phi \mathbf{u} > \epsilon \\ 2 & \nabla^T \phi \mathbf{u} < -\epsilon \\ 1 & \lvert \nabla^T \phi \mathbf{u} \rvert \leq \epsilon \end{cases}$	$\gamma = 2, \;\; \epsilon = 0.1$ $\mu = \begin{cases} 1 - \gamma \frac{\mathbf{u}^T \mathbf{u}}{\nabla^T \phi \mathbf{u}} + 30 & \nabla^T \phi \mathbf{u} > \epsilon \\ 2 & \nabla^T \phi \mathbf{u} < -\epsilon \\ 1 & \lvert \nabla^T \phi \mathbf{u} \rvert \leq \epsilon \end{cases}$
HBF2	$\gamma = 1, \;\; \kappa = 0.3$	$\gamma = 2, \;\; \kappa = 0.6$
HBF3	$\gamma = 1, \;\; \kappa = 1.5, \;\; \epsilon = 0.1$	$\gamma = 2, \;\; \kappa = 0.5, \;\; \epsilon = 0.1$
HBF4	$\gamma = 1, \;\; \kappa = 0.2$	-
HBF5	$a = 120, \;\; b = 230, \;\; \epsilon = 0.1$	$a = 607, \;\; b = 986, \;\; \epsilon = 0.1$
HBF6	$\epsilon = 0.1$ $\mu_1 = 10, \;\; \mu_2 = 0.3$ $a = 1$	$\epsilon = 0.1$ $\mu_1 = 3.5, \;\; \mu_2 = 0.9$ $a = 2$
CG1	$\beta = 1, \;\; \epsilon = 0.1$ $\alpha = \begin{cases} 1 + \beta \frac{\mathbf{u}^T \mathbf{u}}{\nabla^T \phi \mathbf{u}} - 5 & \nabla^T \phi \mathbf{u} > \epsilon \\ 1 + \beta \frac{\mathbf{u}^T \mathbf{u}}{\nabla^T \phi \mathbf{u}} + 28 & \nabla^T \phi \mathbf{u} < -\epsilon \\ 1 & \lvert \nabla^T \phi \mathbf{u} \rvert \leq \epsilon \end{cases}$	$\beta = 2, \;\; \epsilon = 0.1$ $\alpha = \begin{cases} 1 + \beta \frac{\mathbf{u}^T \mathbf{u}}{\nabla^T \phi \mathbf{u}} - 12 & \nabla^T \phi \mathbf{u} > \epsilon \\ 1 + \beta \frac{\mathbf{u}^T \mathbf{u}}{\nabla^T \phi \mathbf{u}} + 5 & \nabla^T \phi \mathbf{u} < -\epsilon \\ 1 & \lvert \nabla^T \phi \mathbf{u} \rvert \leq \epsilon \end{cases}$
CG2	$\beta = 1, \;\; \kappa = 8$	$\beta = 2, \;\; \kappa = 3.2$
CG3	-	$\beta = 2, \;\; \kappa = 11.6$
CG4	-	$\beta = 2, \;\; \kappa = 0.6$
CG5	$\epsilon = 0.1, \;\; a = 2.5, \;\; b = 285$	$\epsilon = 0.1, \;\; a = 9.5, \;\; b = 3000$

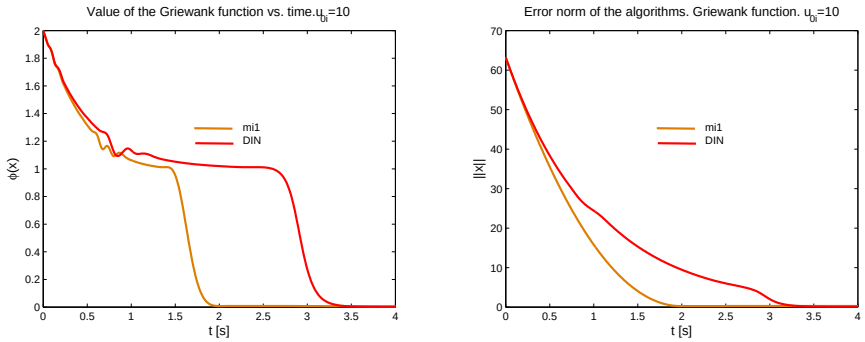

Figure 28: Value of the Griewank function and error norms vs. time for the trajectories generated by the algorithms MI1, MI2, MI3, MI4, MI5 and DIN. Initial point $x_{0_i} = -20$, initial value of the control variable $u_{0_i} = 20$ for all $i \in \{1, \cdots, 10\}$.

4.5 Simulations with the Griewank function

The Griewank function can be expressed as:

$$\phi(\mathbf{x}) = \sum_{i=1}^{N} \frac{x_i^2}{4000} - \prod_{i=1}^{N} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \ : \ \mathbb{R}^N \to \mathbb{R} \tag{60}$$

where N is the number of variables which was chosen as $N = 10$ for the numerical experiments reported here.

The Griewank function has several local minima and a unique global minimum at $\mathbf{x}^* = \mathbf{0}$. The value of the function at the global minimum is $\phi(\mathbf{x}^*) = 0$. In all the experiments reported here, the initial point was chosen as $x_{0_i} = -20$ and the initial value of the control variable as $u_{0_i} = 20$ and $u_{0_i} = 30$, for all $i \in \{1, \cdots, N\}$.

Figure 28 shows the value of the Griewank function versus time and the error norms of the trajectories versus time for the trajectories generated by the algorithms MI1, MI2, MI3, MI4, MI5 and DIN for the initial value of the control variable $u_{0_i} = 20$. The parameters of the algorithms were chosen by hand tuning in such a way to achieve convergence to the global minimum, in the cases where it was possible, and to achieve the fastest convergence. These parameters are reported in Table 7. Note that, such as happened with the Ackley function (figure 22), only MI1 and DIN converge to the global minimum, and in this case it was not possible to find parameters by hand tuning to achieve convergence of the trajectories generated by MI2, MI3, MI4 and MI5.

Figure 29 shows the value of the function and the error norms versus time for the trajectories generated by the gHBF algorithms for the initial value of the control variable $u_{0_i} = 20$. In the algorithms that have a constant parameter γ (HBF, HBF1, HBF2, HBF3 and HBF4) it was chosen as $\gamma = 1$. In all the algorithms that have a parameter ϵ (HBF1, HBF3, HBF5 and HBF6), it was chosen as $\epsilon = 0.1$. The other parameters were chosen by hand tuning in such a way to achieve the fastest convergence to the global minimum. These parameters are reported in Table 7. All the trajectories converge to the global minimum, but approximately at the same time that the one generated by the

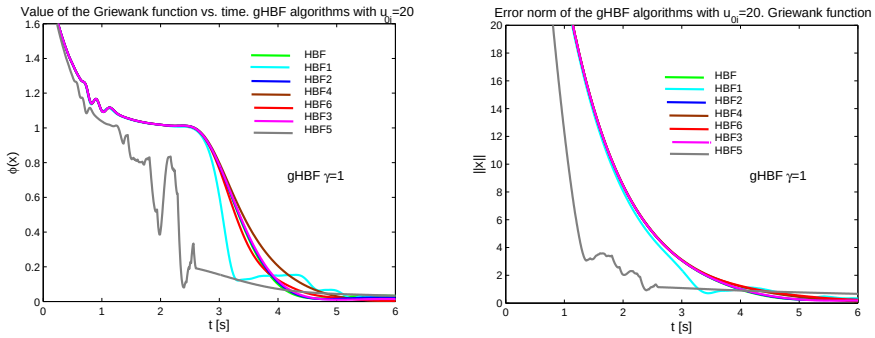

Figure 29: Value of the Griewank function and error norms vs. time for the trajectories generated by the gHBF algorithms with $\gamma = 1$. Initial point $x_{0_i} = -20$, initial value of the control variable $u_{0_i} = 20$ for all $i \in \{1, \cdots, 10\}$.

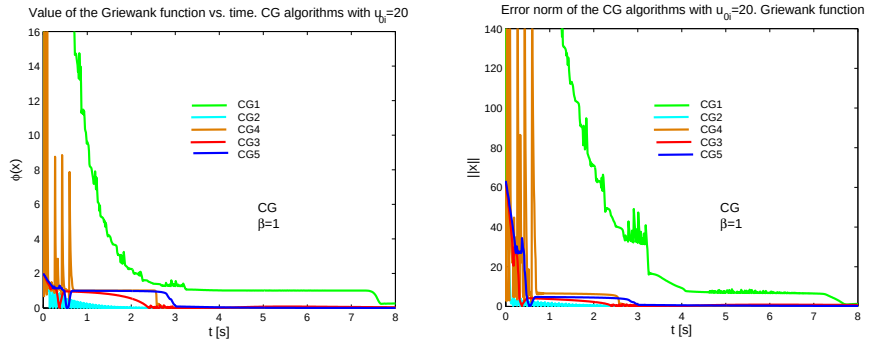

Figure 30: Value of the Griewank function and error norms vs. time for the trajectories generated by the CG algorithms with $\beta = 1$. Initial point $x_{0_i} = -20$, initial value of the control variable $u_{0_i} = 20$ for all $i \in \{1, \cdots, 10\}$.

HBF algorithm.

Figure 30 shows the value of the function and the error norms versus time for the trajectories generated by the CG algorithms for the initial value of the control variable $u_{0_i} = 20$. In the algorithms that have a constant parameter β (CG1, CG2, CG3 and CG4), it was chosen as $\beta = 1$ to compare with the trajectories generated by the gHBF algorithms, with the same value of the parameter γ, and showed in figure 29. In all the algorithms that have a parameter ϵ (CG1, CG3 and CG5), it was chosen as $\epsilon = 0.1$. The other parameters were chosen to try the fastest convergence and they are reported in Table 7. Note that, excepting the trajectory generated by the CG1 algorithm, the others converge to the global minimum very much faster than their gHBF and DIN competitors.

Figure 31 shows the value of the Griewank function versus time and the error norms of the trajectories versus time for the trajectories generated by the algorithms MI1, MI2, MI3, MI4, MI5 and DIN for the initial value of the control variable $u_{0_i} = 30$. The parameters of the algorithms were chosen by hand tuning in such a way to achieve convergence to the global minimum, in the cases where it was possible, and to achieve the fastest convergence. These parameters are

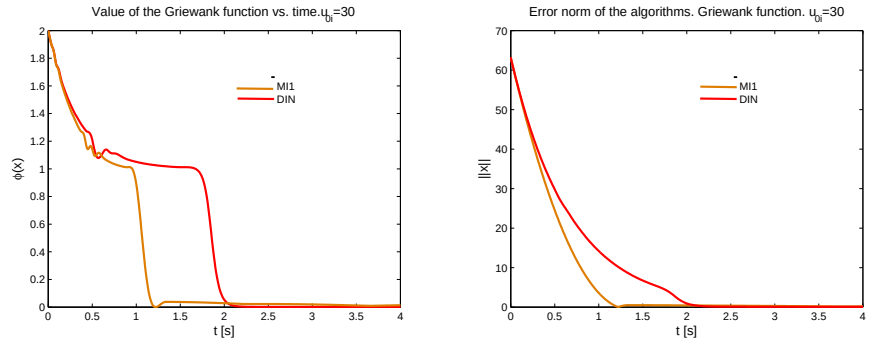

Figure 31: Value of the Griewank function and error norms vs. time for the trajectories generated by the algorithms MI1, MI2, MI3, MI4, MI5 and DIN. Initial point $x_{0_i} = -20$, initial value of the control variable $u_{0_i} = 30$ for all $i \in \{1, \cdots, 10\}$.

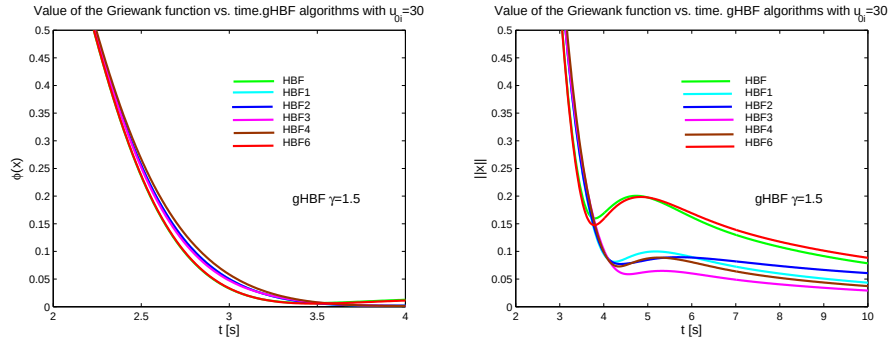

Figure 32: Value of the Griewank function and error norms vs. time for the trajectories generated by the gHBF algorithms with $\gamma = 1.5$. Initial point $x_{0_i} = -20$, initial value of the control variable $u_{0_i} = 30$ for all $i \in \{1, \cdots, 10\}$.

reported in Table 7. Note that, such as happened with the Ackley function (figure 22) and in the former case (figure 28), only MI1 and DIN converge to the global minimum, and in this case it was not possible to find parameters by hand tuning to achieve convergence of the trajectories generated by MI2, MI3, MI4 and MI5.

Figure 32 shows the value of the function and the error norms versus time for the trajectories generated by the gHBF algorithms for the initial value of the control variable $u_{0_i} = 30$. In the algorithms that have a constant parameter γ (HBF, HBF1, HBF2, HBF3 and HBF4) it was chosen as $\gamma = 1.5$. In all the algorithms that have a parameter ϵ (HBF1, HBF3, HBF5 and HBF6), it was chosen as $\epsilon = 0.1$. The other parameters were chosen by hand tuning in such a way to achieve the fastest convergence to the global minimum. These parameters are reported in Table 7. All the trajectories converge to the global minimum, but approximately at the same time that the one generated by the HBF algorithm.

Figure 33 shows the value of the function and the error norms versus time

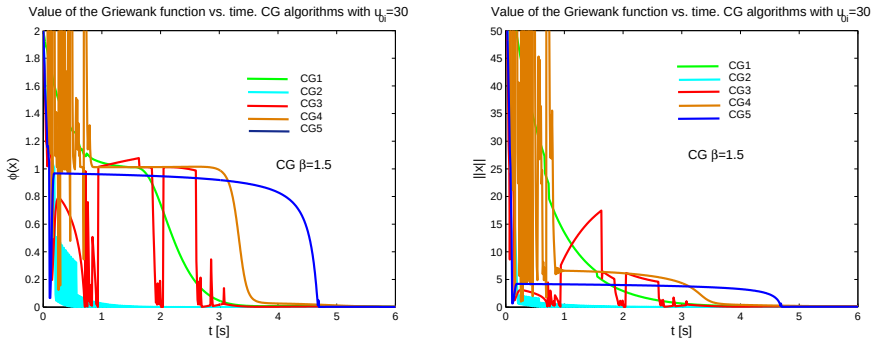

Figure 33: Value of the Griewank function and error norms vs. time for the trajectories generated by the CG algorithms with $\beta = 1.5$. Initial point $x_{0_i} = -20$, initial value of the control variable $u_{0_i} = 30$ for all $i \in \{1, \cdots, 10\}$.

for the trajectories generated by the CG algorithms for the initial value of the control variable $u_{0_i} = 30$. In the algorithms that have a constant parameter β (CG1, CG2, CG3 and CG4), it was chosen as $\beta = 1.5$ to compare with the trajectories generated by the gHBF algorithms, with the same value of the parameter γ, and showed in figure 32. In all the algorithms that have a parameter ϵ (CG1, CG3 and CG5), it was chosen as $\epsilon = 0.1$. The other parameters were chosen to try the fastest convergence and they are reported in Table 7. Note that the trajectory generated by the CG2 algorithm converges to the global minimum very much faster than their gHBF and DIN competitors.

Table 7 reports the parameters used by the algorithms with the Griewank function.

4.6 Comparative analysis of the performances of the algorithms

To make a comparative analysis of the performance of the algorithms, two times of convergence are presented in Table 8. These two times refer to when the trajectories generated by the algorithms get into and remain inside $\mathcal{B}(\mathbf{x}^*, 0.8)$ and $\mathcal{B}(\mathbf{x}^*, 0.1)$; that is, since $\mathbf{x}^* = \mathbf{0}$ in all the functions, the times t_{c_1} and t_{c_2} such that $\|\mathbf{x}(t)\| < 0.8$ for all $t > t_{c_1}$ and $\|\mathbf{x}(t)\| < 0.1$ for all $t > t_{c_2}$, respectively.

The behavior of the trajectories generated by the algorithms presented here, tested with this small suite of benchmark functions, and showed in figures 4 to 26, as well as the times of convergence reported in Table 8, lead us to some conclusions.

The MI algorithms have the simplest structure, in general with only one parameter to adjust. However, only MI1 presents convergence in all the cases tested, and, excepting with the inverted camelback function, convergence is fast and with a good transient.

The trajectories generated by the DIN algorithm converge in almost all the cases tested, and, excepting for the Rastrigin function, a little faster than those

Table 7: Parameters used by the algorithms in the experiments reported in section 4.5 with the Griewank function, where - means that the trajectory does not converge to a minimum in $10\,s$.

	Griewank	
	$u_{0_i} = 20\ \forall i \in \{1, \cdots, 10\}$	$u_{0_i} = 30\ \forall i \in \{1, \cdots, 10\}$
MI1	$\kappa = 10.1$	$\kappa = 22.5$
MI2	-	-
MI3	-	-
MI4	-	-
MI5	-	-
DIN	$a = 1,\ \ b = 30$	$a = 1.49,\ \ b = 57$
HBF	$\gamma = 1$	$\gamma = 1.5$
HBF1	$\gamma = 1,\ \ \epsilon = 0.1$ $\mu = \begin{cases} 100 & \nabla^T\phi\mathbf{u} > \epsilon \\ 30 & \nabla^T\phi\mathbf{u} < -\epsilon \\ 1 & \lvert\nabla^T\phi\mathbf{u}\rvert \leq \epsilon \end{cases}$	$\gamma = 1.5,\ \ \epsilon = 0.1$ $\mu = \begin{cases} 2.2 & \nabla^T\phi\mathbf{u} > \epsilon \\ 0.1 & \nabla^T\phi\mathbf{u} < -\epsilon \\ 1 & \lvert\nabla^T\phi\mathbf{u}\rvert \leq \epsilon \end{cases}$
HBF2	$\gamma = 1,\ \ \kappa = 0.2$	$\gamma = 1.5,\ \ \kappa = 0.6$
HBF3	$\gamma = 1,\ \ \kappa = 0.2,\ \ \epsilon = 0.1$	$\gamma = 1.5,\ \ \kappa = 0.4,\ \ \epsilon = 0.1$
HBF4	$\gamma = 1,\ \ \kappa = 0.2$	$\gamma = 1.5,\ \ \kappa = 0.9$
HBF5	$a = 380,\ \ b = 980,\ \ \epsilon = 0.1$	-
HBF6	$\epsilon = 0.1$ $\mu_1 = 10,\ \ \mu_2 = 0.3$ $a = 1$	$\epsilon = 0.1$ $\mu_1 = 1,\ \ \mu_2 = 0.9$ $a = 1.5$
CG1	$\beta = 1,\ \ \epsilon = 0.1$ $\alpha = \begin{cases} 1 + \beta\frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi\mathbf{u}} - 150 & \nabla^T\phi\mathbf{u} > \epsilon \\ 1 + \beta\frac{\mathbf{u}^T\mathbf{u}}{\nabla^T\phi\mathbf{u}} + 260 & \nabla^T\phi\mathbf{u} < -\epsilon \\ 1 & \lvert\nabla^T\phi\mathbf{u}\rvert \leq \epsilon \end{cases}$	$\beta = 1.5,\ \ \epsilon = 0.1$ $\alpha = \begin{cases} 9 & \nabla^T\phi\mathbf{u} > \epsilon \\ 0.9 & \nabla^T\phi\mathbf{u} < -\epsilon \\ 1 & \lvert\nabla^T\phi\mathbf{u}\rvert \leq \epsilon \end{cases}$
CG2	$\beta = 1,\ \ \kappa = 7$	$\beta = 1.5,\ \ \kappa = 3$
CG3	$\beta = 1,\ \ \kappa = 3.5,\ \ \epsilon = 0.1$	$\beta = 1.5,\ \ \kappa = 15.5,\ \ \epsilon = 0.1$
CG4	$\beta = 1,\ \ \kappa = 40$	$\beta = 1.5,\ \ \kappa = 49.2$
CG5	$\epsilon = 0.1,\ \ a = 2,\ \ b = 720$	$\epsilon = 0.1,\ \ a = 8,\ \ b = 950$

Table 8: Times of convergence of the trajectories for the experiments reported here, expressed in s. Convergence occurs when $\|\mathbf{x}(t_c) - \mathbf{x}^*\| < \delta$, and remains for all $t > t_c$, where the first time corresponds to $\delta = 0.8$ and the second time corresponds to $\delta = 0.1$. $\mathbf{x}_{0_1} = [2\ \ 1]^T$, $\mathbf{x}_{0_2} = [-1.5\ \ -1.5]^T$, $\mathbf{x}_{0_3} = [1.7475\ \ -0.8737]^T$ and - means that the trajectory does not achieve the goal in $10\,s$. The fastest times in each case are boldfaced.

	Scalar	**Camelback**			**Rastrigin**		**Ackley**		**Griewank**	
		$\mathbf{x}_0 = \mathbf{x}_{0_1}$	$\mathbf{x}_0 = \mathbf{x}_{0_2}$	$\mathbf{x}_0 = \mathbf{x}_{0_3}$	$\gamma = \beta = 5$	$\gamma = \beta = 10$	$u_{0_i} = 10$	$u_{0_i} = 20$	$u_{0_i} = 20$	$u_{0_i} = 30$
MI1	1.61 1.87	5.76 9.98	2.64 3.5	3.26 -	**0.18 0.31**		1.72 **1.95**	0.85 0.96	1.81 -	1.17 -
MI2	1.26 1.47	- -	2.01 5.31	6.6 9.88	- -		- -	- -	- -	- -
MI3	4.68 9.83	- -	- -	- -	4 -		- -	- -	- -	- -
MI4	- -	- -	2.69 7.59	- -	- -		- -	- -	- -	- -
MI5	0.24 1.3	- -	- -	- -	- -		- -	- -	- -	- -
DIN	2.17 2.45	- -	1.01 3.91	0.76 1.55	4.36 5.84		2.93 4.05	1.82 2.66	3.22 4.29	2.02 2.46
HBF	3.0 9.83	4.71 6.64	1.04 3.63	0.61 1.61	0.2 1.22	0.77 1.18	3.9 7.49	1.64 3.5	4.14 -	2.82 8.46
HBF1	2.36 2.6	2.88 4.25	1.05 2.46	0.86 2.27	0.21 0.7	0.23 0.41	1.98 2.85	1.28 1.65	9.5 -	2.86 3.97
HBF2	1.74 2.39	- -	1.67 5.52	0.62 1.7	0.22 0.97	0.92 1.46	3.31 5.25	1.95 2.56	4.17 -	2.86 3.99
HBF3	1.81 4.63	4.86 -	1.13 6.37	4.56 8.28	0.2 0.84	0.79 1.22	2.97 8.75	1.65 2.84	4.20 -	2.85 4.02
HBF4	2.29 5.05	- -	1.67 3.70	0.64 1.72	- -	- -	- -	- -	4.16 -	2.89 4
HBF5	1.33 1.47	7.72 -	0.77 7.89	0.25 2.34	- -		- -	- -	- -	- -
HBF6	3.0 4.65	2.22 4.53	0.86 3.25	0.65 1.56	0.29 0.83		3.9 6.84	1.64 3.35	4.13 -	2.82 9.1
CG1	**0.1 0.37**	0.08 2.05	**0.04** 1.7	0.6 0.61	0.23 0.53	**0.05 0.2**	1.89 2.53	1.08 2.22	- -	2.82 5.99
CG2	0.63 1.05	1.74 2.52	0.05 0.64	0.31 1.12	0.18 0.49	0.04 0.25	**1.1** 2.97	**0.49** 1.46	**1.55 3.72**	**0.93 2.08**
CG3	0.55 0.71	0.68 2.71	0.82 2.59	0.3 2.1	0.21 0.81	0.74 1.15	- -	1.51 1.75	7.4 -	3.09 7.39
CG4	3.83 5.39	**0.03** 0.76	0.11 **0.19**	1.33 2.04	0.28 0.36	0.19 0.2	- -	1.11 1.49	2.72 -	3.56 6.77
CG5	1.39 1.73	0.10 **0.20**	0.70 2.36	**0.21 0.27**	1.1 1.24		2.69 2.82	0.73 **0.77**	3.02 -	4.67 4.7

generated by the traditional HBF algorithm. With the inverted camelback function from $\mathbf{x}_0 = [2\ 1]^T$, the trajectory converges to the closer local minimum excepting that a very small value of the parameter b were used, and in this case the algorithm becomes similar to the HBF, zeroing the spatial damping term.

Among the gHBF algorithms, the HBF1, HBF2, HBF3 and HBF6 presented good performances, converging in almost all the cases tested. However, in HBF2 and HBF3, in some cases it was necessary to choose very small values of the parameter κ, so the parameter μ approximates to one and the algorithms becomes similar to the HBF. The HBF1 and HBF6 present the lowest times of convergence, lower than the times presented by the HBF algorithm.

All the CG algorithms presented good performances and fast times of convergence. In the majority of the cases, they are very much faster than the trajectories generated by the gHBF algorithms with the same value of the friction parameter (γ in the gHBF, and β in the CG). The best performances were presented by CG2 and CG5.

Finally, we note that with the Ackley function, from the initial point and with the initial values of the control variable chosen, and with the values of the parameters used, several algorithms did not achieve convergence, with both of the used criterions. Once again, we emphasize that it is possible that from another initial point, with another initial value of the control variable, or a different value of the friction parameter (γ in gHBF and β in CG) the goal may be achieved. This possibility was not tested. With the Griewank function, with the initial value of the control variable $u_{0_i} = 20$, we note that several trajectories get into $\mathcal{B}(\mathbf{x}^*, 0.8)$, but only two of them (DIN and CG2) get into $\mathcal{B}(\mathbf{x}^*, 0.1)$ until $10\,s$, the final time used in the experiments. With the initial value of the control variable $u_{0_i} = 30$, it only happens with the MI1 algorithm.

4.7 Analysis of the discontinuities of the trajectories

The algorithms deduced here are characterized by continuous time ODEs. These algorithms generate continuous trajectories from an initial point $\mathbf{x}_0$ to, if convergence is achieved, the global minimum $\mathbf{x}^*$. It is true even in the cases where the update law of the state variable is discontinuous. For example, the first order ODE with a discontinuous update law:

$$\dot{\mathbf{x}} = -\text{sgn}(\mathbf{x})$$

generates a continuous trajectory along the time from an initial point $\mathbf{x}_0$ given by:

$$\mathbf{x}(t) = \mathbf{x}_0 - \text{sgn}(\mathbf{x}_0)t$$

which converges to the equilibrium point $\mathbf{x} = \mathbf{0}$. One notable feature presented by ODEs with discontinuous right-hand side is that they can present convergence in *finite time*; in this case, the convergence time is $T = \max_i |x_{0_i}|$.

The notable features, true for many vector ODEs with discontinuous right-hand sides, are the finite time convergence and the fact that there is a continuous solution from the initial condition to the equilibrium point (see [32, 31] for further details).

However, observing the plots presented in the experiments reported here, we note that there are several discontinuous trajectories. This phenomenon is due to the discretization of the algorithms made with the forward Euler method

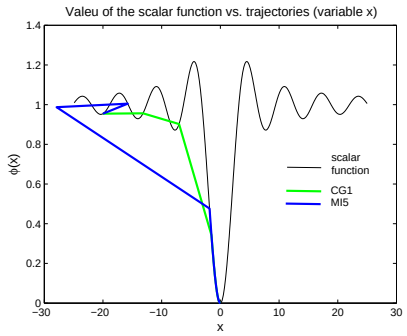

Figure 34: Value of the scalar function vs. trajectory x for the trajectories generated by the CG1 and MI5 algorithms. Initial point $x_0 = -20$, initial value of the control variable $u_0 = 20$.

with a constant step size $h = 0.01\,s$. This means that the state variables are calculated as:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h\dot{\mathbf{x}}$$

where k is a natural number, such that $t = k\,h$.

We also note that, in the cases where the update law of the state variable $\dot{\mathbf{x}} = \mathbf{u}$, there are no discontinuous trajectories, independently of the update law of the control variable $\dot{\mathbf{u}}$, whereas in the cases where the update law $\dot{\mathbf{x}} \neq \mathbf{u}$, for example, in the CGs cases where $\dot{\mathbf{x}} = \alpha(\mathbf{x}, \mathbf{u})\mathbf{u}$, discontinuous trajectories can be produced (if $\alpha(\mathbf{x}, \mathbf{u})$ is large enough). This means that, from the discretized state $\mathbf{x}_k$, the next state $\mathbf{x}_{k+1}$ is calculated as $\mathbf{x}_{k+1} = \mathbf{x}_k + h\alpha(\mathbf{x}_k, \mathbf{u}_k)\mathbf{u}_k$ and, if the product $h\alpha(\mathbf{x}_k, \mathbf{u}_k)\mathbf{u}_k$ is large enough, then this large jump occurs in the discrete state.

The discontinuities in the trajectories are not necessarily an undesirable feature. We note that this is exactly the reason because the convergence of some trajectories generated by the CG method is very much faster than those generated by its competitors.

In order to get a better feel for the interplay between the choice of the step size h and the switching law $\alpha(\cdot, \cdot)$, the scalar function (56) will be used. Figure 34 shows the value of the objective function vs. the trajectories generated by the two fastest algorithms, CG1 and MI5 with a step size $h = 0.01\,s$ and the same values of the parameters and initial conditions than those used in the former simulations. These trajectories converge to a ball $\mathcal{B}(\mathbf{x}^*, 0.8)$ in $0.1\,s$ and $0.24\,s$, respectively (see Table 8).

Note that the discontinuities in the trajectories are exactly what seem to speed up the convergence, reaching a point close to $\mathbf{x}^*$ very fast (actually, $0.03\,s$, before initiating a continuous trajectory to the minimum point).

Figure 35 shows the trajectories generated by the CG1 algorithm from the initial point $x_0 = -20$, with initial value of the control variable $u_0 = 10$, and the parameters $\beta = 10$, $\epsilon = 0.1$ and $\alpha = 34.1$ if $\nabla^T\phi\mathbf{u} > \epsilon$ and $\alpha = 1$ if $\nabla^T\phi\mathbf{u} \leq \epsilon$, but discretized with step sizes $h = 0.1$, 0.05, and 0.01, respectively. The figure shows clearly that for a fixed value of the parameter $\alpha(\mathbf{x}_k, \mathbf{u}_k)$, the greater the value of the step size h, the greater the initial jump in the trajectory. For example, from the initial conditions, with $\alpha(\mathbf{x}_0, \mathbf{u}_0)$, the step size $h = 0.1$

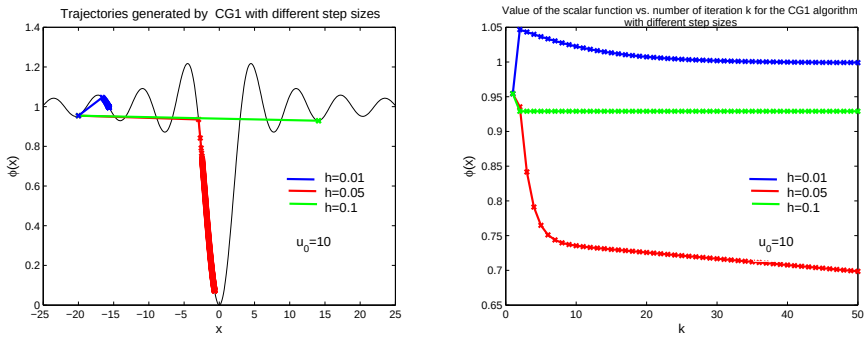

Figure 35: Trajectories generated by CG1 with the scalar function and value of the function vs. number of iteration k using step sizes $h = 0.1$, 0.05 and 0.01, with $x_0 = -20$ and $u_0 = 10$.

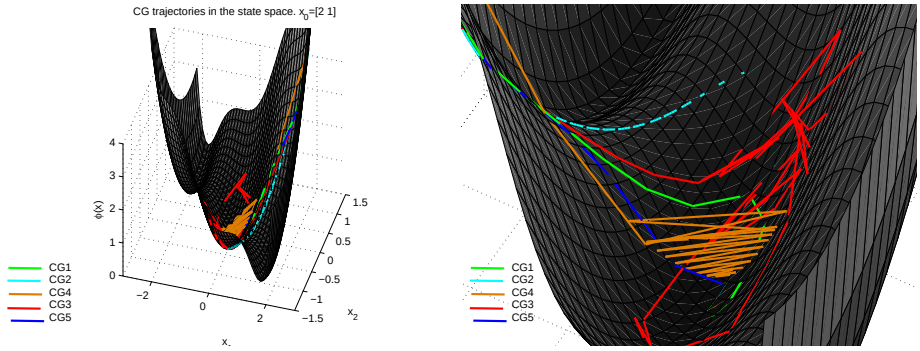

Figure 36: Trajectories generated by the CG algorithms for the camelback function in 3D, where the gray surface represents the value of the function. Initial point $\mathbf{x}_0 = [2\ 1]^T$, initial value of the control variable $\mathbf{u}_0 = [0\ 0]^T$.

produces the greater jump. Thus if $\alpha, \beta,$ and u_0 are adequately chosen, so that the large jump takes the state close to the global minimum, which also means that convergence is the fastest.

Figure 36 shows the trajectories generated by the CG algorithms with the inverted camelback function as objective function from the initial point $\mathbf{x}_0 = [2\ 1]^T$ in a three-dimensional plot. The figure to the right is a zoom of the one on the left, in order to visualize the discontinuities in some trajectories (specially, in those generated by CG1, CG3 and CG4). The figure shows the gray surface which represents the value of the objective function and the colored trajectories.

Note that the trajectory which seems to present more discontinuities (CG4) is that with fastest time of convergence ($0.03\,s$ to get into a ball $\mathcal{B}(\mathbf{x}^*, 0.8)$).

Figure 37 shows the trajectories generated by the CG algorithms with the inverted camelback function as objective function from the initial point $\mathbf{x}_0 = [-1.5\ -1.5]^T$ in a three-dimensional plot. The figure to the left has the opposite view point of the right one.

Note that, also here, the trajectory which seems to present more discontinuities (CG4) is one of the fastest ($0.11\,s$ to get into a ball $\mathcal{B}(\mathbf{x}^*, 0.8)$).

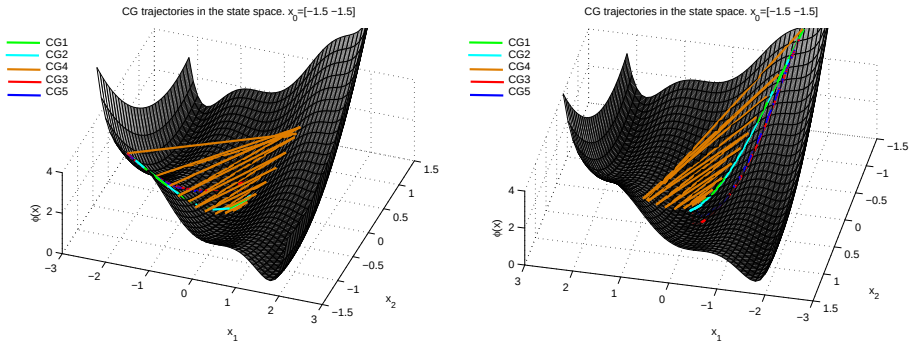

Figure 37: Trajectories generated by the CG algorithms for the camelback function in 3D, where the gray surface represents the value of the function. Initial point $\mathbf{x}_0 = [-1.5 \ -1.5]^T$, initial value of the control variable $\mathbf{u}_0 = [1 \ 1]^T$.

The discontinuities presented by some of the trajectories generated by the CG algorithms seems to be the main reason for their fast convergence to the minimum point.

4.8 Application with an unknown objective function

When the goal is to find the minimum of an unknown scalar objective function $\phi(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$, a strategy can be to choose several initial points $\mathbf{x}_0$ with several initial values of the control variable $\mathbf{u}_0$. From every initial point and with every initial value of the control variable, one algorithm is used with different values of the parameters, computing at each execution the minimal point founded. Of course, it is not guaranteed that the minimum founded is the global minimum of the function, but as greater the number of initial points and the number of the initial values of the control variable, the greater the possibility of finding the global minimum of the objective function.

As illustrative examples, the CG2 algorithm is chosen, because it has only two parameters to adjust and it presented an excellent performance with all the functions used in the former subsections, and for comparison the traditional HBF algorithm. The objective function chosen is the Rastrigin function (58) with 10 variables; this function presents a global minimum at $\mathbf{x}^* = \mathbf{0}$ and the value of the function at the global minimum is $\phi(\mathbf{x}^*) = -N = -10$.

Four initial points and four initial values of the control variable were used in this example. They are:

$\mathbf{x}_{0_1} = \mathbf{u}_{0_1} = [10 \ 10 \ 10 \ 10 \ 10 \ -10 \ -10 \ -10 \ -10 \ -10]^T$
$\mathbf{x}_{0_2} = \mathbf{u}_{0_2} = [-10 \ -10 \ -10 \ -10 \ -10 \ 10 \ 10 \ 10 \ 10 \ 10]^T$
$\mathbf{x}_{0_3} = \mathbf{u}_{0_3} = [10 \ 10 \ 10 \ 10 \ 10 \ 10 \ 10 \ 10 \ 10 \ 10]^T$
$\mathbf{x}_{0_4} = \mathbf{u}_{0_4} = [-10 \ -10 \ -10 \ -10 \ -10 \ -10 \ -10 \ -10 \ -10 \ -10]^T$

The parameter γ in the HBF algorithm was adjusted by hand tuning in such a way to achieve the fastest convergence to the global minimum (which is *a priori* known in this example). In the CG2 algorithm, 225 different values of the parameter κ and 225 different values of the parameter β, both of them ranging from 0.1 to 1000, were tested, and the fastest convergence to the minimum value reached by the trajectories was recorded.

Table 9: Parameters and time of convergence, expressed in s, of the algorithms HBF and CG2 from each initial point and with each initial value of the control variable. Convergence occurs when $\|\mathbf{x}(t_c) - \mathbf{x}^*\| < \delta$, and remains for all $t > t_c$, where t_1 corresponds to $\delta = 0.8$ and t_2 corresponds to $\delta = 0.1$.

	HBF			**CG2**		
	param.	t_1	t_2	param.	t_1	t_2
$\mathbf{x}_{0_1}$ $\mathbf{u}_{0_1}$	$\gamma = 2.3$	3.07	5.13	$\beta = 82.5$ $\kappa = 115$	0.03	0.03
$\mathbf{x}_{0_1}$ $\mathbf{u}_{0_2}$	$\gamma = 2.3$	1.92	3.99	$\beta = 97.5$ $\kappa = 95$	0.02	0.03
$\mathbf{x}_{0_1}$ $\mathbf{u}_{0_3}$	$\gamma = 2.2$	2.75	4.85	$\beta = 29$ $\kappa = 205$	0.24	0.32
$\mathbf{x}_{0_1}$ $\mathbf{u}_{0_4}$	$\gamma = 2.2$	2.75	4.85	$\beta = 29$ $\kappa = 205$	0.24	0.32
$\mathbf{x}_{0_2}$ $\mathbf{u}_{0_1}$	$\gamma = 2.3$	1.92	3.99	$\beta = 97.5$ $\kappa = 95$	0.02	0.03
$\mathbf{x}_{0_2}$ $\mathbf{u}_{0_2}$	$\gamma = 2.3$	3.07	5.13	$\beta = 82.5$ $\kappa = 115$	0.03	0.03
$\mathbf{x}_{0_2}$ $\mathbf{u}_{0_3}$	$\gamma = 2.2$	2.75	4.85	$\beta = 29$ $\kappa = 205$	0.24	0.32
$\mathbf{x}_{0_2}$ $\mathbf{u}_{0_4}$	$\gamma = 2.2$	2.75	4.85	$\beta = 29$ $\kappa = 205$	0.24	0.32
$\mathbf{x}_{0_3}$ $\mathbf{u}_{0_1}$	$\gamma = 2.2$	2.75	4.85	$\beta = 29$ $\kappa = 205$	0.24	0.32
$\mathbf{x}_{0_3}$ $\mathbf{u}_{0_2}$	$\gamma = 2.2$	2.75	4.85	$\beta = 29$ $\kappa = 205$	0.24	0.32
$\mathbf{x}_{0_3}$ $\mathbf{u}_{0_3}$	$\gamma = 2.3$	3.07	5.13	$\beta = 82.5$ $\kappa = 115$	0.03	0.03
$\mathbf{x}_{0_3}$ $\mathbf{u}_{0_4}$	$\gamma = 2.3$	1.92	3.99	$\beta = 97.5$ $\kappa = 95$	0.02	0.03
$\mathbf{x}_{0_4}$ $\mathbf{u}_{0_1}$	$\gamma = 2.2$	2.75	4.85	$\beta = 29$ $\kappa = 205$	0.24	0.32
$\mathbf{x}_{0_4}$ $\mathbf{u}_{0_2}$	$\gamma = 2.2$	2.75	4.85	$\beta = 29$ $\kappa = 205$	0.24	0.32
$\mathbf{x}_{0_4}$ $\mathbf{u}_{0_3}$	$\gamma = 2.3$	1.92	3.99	$\beta = 97.5$ $\kappa = 95$	0.02	0.03
$\mathbf{x}_{0_4}$ $\mathbf{u}_{0_4}$	$\gamma = 2.3$	3.07	5.13	$\beta = 82.5$ $\kappa = 115$	0.03	0.03

From every initial point and with every initial value of the control variable, both of the algorithms achieve the global minimum. The results are reported in Table 9. Note that, once again, the times of convergence of the trajectories generated by the CG2 algorithms are very much lower than the times of convergence of the trajectories generated by the traditional HBF algorithm.

5 Conclusions

The second order continuous time algorithms proposed in this paper, adequate for implementation as neural networks, were able to find minima of nonconvex scalar functions in a satisfactory way. The algorithms were tested with a small suite of benchmark functions. They were the shifted *sinc* function, inverted camelback, Rastrigin, Ackley and Griewank, which are all multimodal and have a unique finite global minimum.

The interpretation of the algorithms as closed loop control system has the advantage of to offer powerful tools for the design of new algorithms as well as for the analysis of the behavior of the existing ones. One tool, systematically used in this paper, is the control Liapunov function method. The algorithms deduced here are only some of the simplest examples which could be deduced using this method.

The authors feel that second-order neural networks, systematically designed using the powerful CLF technique, are promising, although more extensive numerical testing needs to be carried out, including the possibility of random initialization of parameters, instead of hand tuning, and combination with other

randomization methods to generate new starting points, such as the differential evolution (DE) method. Some preliminary progress in this area, using a combination of the Snyman-Fatti heavy ball method and a DE method has been made ([33]) and these ideas could be applied to propose a hybrid of DE and CG nets.

Comparing the performances of the different algorithms, we note that the goal of bypassing local minima is achieved only by some of the new algorithms and only after a careful adjustment of the parameters. However, this possibility of adjusting parameters does represent an advantage compared with algorithms based on first order ODEs, which will always stop at local minima, no matter what parameters are chosen.

This is a well known drawback of deterministic schemes for global minimization, but two observations are important here. First, it should be noted that this possibility of adjusting parameters is an advantage compared with neural nets based exclusively on gradient information (first-order ODEs in general), the trajectories of which will usually converge to local minima. Second, the numerical experiments carried out, although on a small set of benchmark functions, indicate that a few random initializations are usually sufficient to find suitable parameter values, so that a hybrid net (random initializations, followed by deterministic trajectory following) seems to have good potential for fast convergence to a global minimum, with fewer function evaluations and less computational burden compared to heuristic probabilistic algorithms, such as genetic algorithms.

Of the seventeen different nets tested (five MI, DIN, six gHBF and five CG), only the CG net presented very good behavior on all of the benchmark examples on which all the algorithms were tested, showing fast convergence and good transients and accuracy.

The CG method, with the **x**-dynamics being directly affected by the discontinuous switching law in α does better than the gHBF methods, in which the CLF designed discontinuous switching law directly affects the **u**-dynamics, while the **x**-dynamics are affected only after integration (hence in a continuous fashion).

In the light of the results obtained in this paper, the proposed CG net seem to be consistently better than the heavy ball method and its variants, which have received much attention in the literature. Further research on initialization of the CG net so as to promote convergence to the global minimum with a high probability needs to be carried out, and a theoretical analysis of the rate of convergence still needs to be carried out.

We end with a quote from Bertsekas [34, p.75] and a reflection: *Generally, there is a tendency to think that difficult problems should be addressed with sophisticated methods, such as Newton-like methods. This is often true, particularly for problems with nonsingular local minima that are poorly conditioned. However, it is important to realize that* often the reverse is true*, namely that for problems with "difficult" cost functions and singular local minima, it is best to use simple methods such as (perhaps diagonally scaled) steepest descent with simple stepsize rules such as a constant or diminishing stepsize. The reason is that methods that use sophisticated descent directions and stepsize rules often rely on assumptions that are likely to be violated in difficult problems. We also note that for difficult problems, it may be helpful to supplement the steepest descent method with features that allow it to deal better with multiple local minima*

and peculiarities of the cost function. An often useful modification is the heavy ball method ....

References

[1] A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*. Chichester: John Wiley, 1993.

[2] M. T. Chu, "Linear algebra algorithms as dynamical systems," *Acta Numerica*, pp. 1–86, 2008.

[3] A. Bhaya and E. Kaszkurewicz, *Control perspectives on numerical algorithms and matrix problems*, ser. Advances in Control. Philadelphia: SIAM, 2006.

[4] ——, "Steepest descent with momentum for quadratic functions is a version of the conjugate gradient method," *Neural Networks*, vol. 17, no. 1, pp. 65–71, 2004.

[5] B. T. Polyak, "Some methods of speeding up the convergence of iterative methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964, Zh. Vychisl. Mat. Mat. Fiz., pp. 791-803 (Russian edition).

[6] ——, *Introduction to optimization*. New York: Optimization software, 1987.

[7] H. Attouch, X. Goudou, and P. Redont, "The heavy ball with friction method. The continuous dynamical system," *Communications in contemporary math*, vol. 2, pp. 1–34, 2000.

[8] A. Cabot, "Inertial gradient-like dynamical system controlled by a stabilizing term," *Journal of Optimization Theory and Applications*, vol. 120, no. 2, pp. 275–303, 2004.

[9] A. Bhaya and E. Kaszkurewicz, "Iterative methods as dynamical systems with feedback control," in *Proc. 42nd IEEE Conference on Decision and Control*, Maui, Hawaii, USA, December 2003, pp. 2374–2380.

[10] ——, "A control-theoretic approach to the design of zero finding numerical methods," *IEEE Transactions on Automatic Control*, vol. 52, no. 6, pp. 1014–1026, June 2007.

[11] F. Pazos, A. Bhaya, and E. Kaszkurewicz, "Unified control Liapunov function based design of neural networks that aim at global minimization of nonconvex functions," in *Proc. of the International Joint Conference on Neural Networks*, Atlanta, U.S.A., 2009.

[12] A. Bhaya, F. Pazos, and E. Kaszkurewicz, "Comparative study of the CG and HBF ODEs used in the global minimization of nonconvex functions," in *Proceedings of the 19th. International Conference on Artificial Neural Networks*, Limassol, Cyprus, September 2009.

[13] F. Alvarez, H. Attouch, J. Bolte, and P. Redont, "A second-order gradient-like dissipative dynamical system with Hessian-driven damping. Application to optimization and mechanics," *J. Math. Pures Appl.*, vol. 81, pp. 747–779, 2002.

[14] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*. Englewood Cliffs, New Jersey: Prentice Hall, 1991.

[15] J.-P. Chehab and M. Raydan, "Implicit and adaptive inverse preconditioned gradient methods for nonlinear problems," *Applied Numerical Mathematics*, vol. 55, no. 1, pp. 32–47, 2005.

[16] J. Snyman and L. Fatti, "A multi-start global minimization algorithm with dynamic search trajectories," *Journal of Optimization Theory and Applications*, vol. 54, pp. 121–141, 1987.

[17] J. A. Snyman and S. Kok, "A reassessment of the Snyman-Fatti dynamic search trajectory method for unconstrained global optimization," *Journal of Global Optimization*, vol. 43, pp. 67–82, 2009, dOI: 10.1007/s10898-008-9293-y.

[18] K. Shimizu, H. Sugata, and T. Hagino, "Global optimization via multi-trajectory inertial system and chaos," in *1st International Conference on Control of Oscillations and Chaos*, vol. 2, Aug 1997, pp. 262–266, dOI:10.1109/COC.1997.631339.

[19] A. V. Levy and A. Montalvo, "The tunneling algorithm for the global minimization of functions," *SIAM Journal on Scientific and Statistical Computing*, vol. 6, pp. 15–29, 1985.

[20] P. V. Barhen, J. and D. Reister, "Trust: A deterministic algorithm for global optimization," *Science*, vol. 276, pp. 1094–1097, May 1997.

[21] J. Lee, "A novel three-phase trajectory informed search methodology for global optimization," *Journal of Global Optimization*, vol. 38, pp. 61–77, 2007, dOI: 10.1007/s10898-006-9083-3.

[22] Y. Shang, "Global search methods for solving nonlinear optimization problems," Ph.D. dissertation, University of Illinois, Urbana Champaign, U.S.A., 1997.

[23] N. Ampazis and S. Perantonis, "Two highly-efficient second order algorithms for training feedforward networks," *IEEE Transactions on Neural Networks*, vol. 13, no. 5, 2002.

[24] J. Egea, E. Vazquez, J. Banga, and R. Mart, "Improved scatter search for the global optimization of computationally expensive dynamic models," *Journal of Global Optimization*, 2007, dOI: 10.1007/s10898-007-9172-y.

[25] S. Ray and P. S. V. Nataraj, "An efficient algorithm for range computation of polynomials using the Bernstein form," *Journal of Global Optimization*, 2008, dOI: 10.1007/s10898-008-9382-y.

[26] J. Olensek, A. Burmen, J. Puhan, and T. Tuma, "Desa: a new hybrid global optmization method and its application to analog integrated circuit sizing," *Journal of Global Optimization*, 2008, dOI: 10.1007/s10898-008-9307-9.

[27] M. J. Hirsch, C. N. Meneses, P. M. Pardalos, and M. G. C. Resende, "Global optimization by continuous grasp," *Optimization Letters*, vol. 1, pp. 201–212, 2007, dOI: 10.1007/s11590-006-0021-6.

[28] M. Schnurr, "A second-order pruning step for verified global optimization," *Journal of Global Optimization*, 2008, dOI: 10.1007/s10898-008-9331-9.

[29] C. A. Floudas and C. E. Gounaris, "A review of recent advances in global optimization," *Journal of Global Optimization*, 2008, dOI: 10.1007/s10898-008-9332-8.

[30] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Networks*, vol. 12, pp. 145–151, 1999.

[31] J. Cortés, "Discontinuous dynamical systems. A tutorial on notions of solutions, nonsmooth analysis, and stability." *IEEE Control systems magazine*, pp. 36–73, June 2008.

[32] A. F. Filippov, *Differential Equations with Discontinuous Righthand Sides*. Dordrecht: Kluwer Academic, 1988.

[33] E. C. Laskari, K. E. Parsopoulos, and M. N. Vrahatis, "Evolutionary operators in global optimization with dynamic search trajectories," *Numerical Algorithms*, vol. 34, pp. 393–403, 2003.

[34] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed., ser. Optimization and Computation series. Belmont, MA: Athena Scientific, 1999.