Soft Computing manuscript No.
(will be inserted by the editor)

Using Covariance Matrix Adaptation Evolution Strategies for Solving Different Types of Differential Equations

Jose M. Chaquet · Enrique J. Carmona

the date of receipt and acceptance should be inserted later

Abstract A novel mesh-free heuristic method for solving differential equations is proposed. The new approach can cope with linear, nonlinear, and partial differential equations (DE), and systems of DEs. Candidate solutions are expressed using a linear combination of kernel functions. Thus, the original problem is transformed into an optimization problem that consists in finding the parameters that define each kernel. The new optimization problem is solved applying a Covariance Matrix Adaptation Evolution Strategy (CMA-ES). To increase the accuracy of the results, a Downhill Simplex local search is applied to the best solution found by the mentioned evolutionary algorithm. Our method is applied to 32 differential equations extracted from the literature. All problems are successfully solved, achieving competitive accuracy levels when compared to other heuristic methods. A simple comparison with numerical methods is performed using two partial differential equations to show the pros and cons of the proposed algorithm. To verify the potential of this approach with a more practical problem, an electric circuit is analyzed in depth. The method can obtain the dynamic behavior of the circuit in a parametric way, taking into account different component values.

Differential equations Covariance Matrix Adaptation Evolution Strategies Gaussian kernel Downhill Simplex Algorithm

Jose M. Chaquet · Enrique J. Carmona
Dpto. de Inteligencia Artificial, Escuela Técnica Superior de Ingeniería Informática, Universidad Nacional de Educación a Distancia, Madrid, Spain
Tel.:+34 91 398 7301
E-mail: jose.chaquet@gmail.com
E-mail: ecarmona@dia.uned.es

1 Introduction

Calculus is the mathematical study of the *rate of change*. It has two major branches, *differential* calculus (concerning rates of change and slopes of curves), and *integral* calculus (concerning accumulation of quantities and the areas under and between curves). These two branches are related to each other by the fundamental theorem of calculus (Spivak, 1980). The modern development of calculus is usually credited to Isaac Newton (1643-1727) and Gottfried Leibniz (1646-1716), who provided independent and unified approaches to differentiation and derivatives.

Once the concept of derivative was clearly established, more complex mathematical objects were developed such as differential equations (DEs). A differential equation is just an algebraic relation between functions and their derivatives. These mathematical entities allow scientists to understand a wide range of complex phenomena. Many fundamental laws of physics and chemistry can be formulated as differential equations. DEs are also useful to model different problems in a lot of scientific fields, such as biology, economics or engineering. Moreover, when the same differential equation describes different phenomena, it has been used as unifying principle. As an example, the propagation of light and sound in the atmosphere, or the waves on the surface of a pond may be described by the same second-order partial differential equation, the *wave* equation. In the same way, heat conduction in a solid is governed by another second-order partial differential equation, the *heat* equation. The *Black-Scholes* equation in finance is, for instance, related to the *heat* equation.

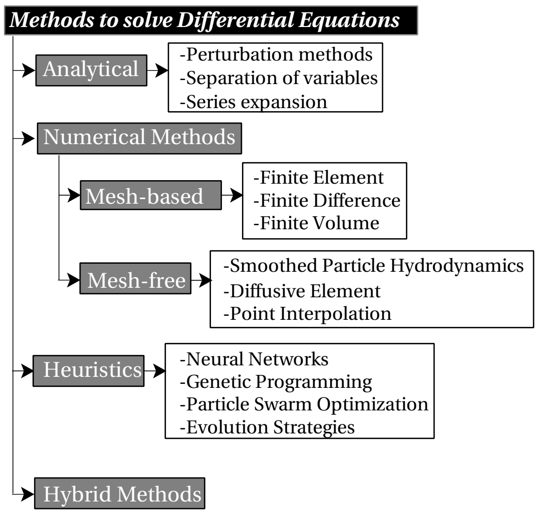

Figure 1: A possible taxonomy of methods to solve differential equations.

Some simple differential equations admit solutions given by explicit formulas. But in the general case, only approximate solutions can be found. Several paradigms exist in the literature to solve the equations. Fig. 1 shows a possible taxonomy of existing methods to solve differential equations. A first group can be formed with analytical methods, which try to find exact solutions. However, as it has been commented, very few problems admit this approach. The second group of methods transforms the original differential equations into a system of algebraic equations and solves them by numerical methods. Inside this category, two subfamilies can be distinguished according to whether or not a connectivity mesh is needed. Mesh-based schemes can be classified into *finite element* method (FEM) (Suli and Mayers, 2003), *finite difference* method (FDM) (Ozisik, 1994), or *finite volume* method (FVM) (Leveque, 2002). On the other hand, mesh-free algorithms, such as *smoothed particle hydrodynamics* (SPH), *diffusive element* method (DEM) and *point interpolation* method (PIM), do not require mesh connectivity (Liu, 2010). However, the final algebraic equations obtained are solved using numerical methods.

A radically different type of approach, denoted by *Heuristics* at Fig. 1, consists in transforming the problem into one of optimization, where a candidate solution is tested according to a fitness or cost function which measures how the differential equation is fulfilled. Generally speaking, these methods are also mesh-free, but are more flexible because can cope with different types of equations. Several paradigms have been reported in this field, such as cellular automata (Puffer et al., 1995), artificial neural networks (Lagaris et al., 1998; He et al., 2000b; Parisi et al., 2003; Sun et al., 2003; Choi and Lee, 2009; Shirvany et al., 2009; Tsoulos et al., 2009; Chen et al., 2011; Kumar and Yadav, 2011; Yazdi et al., 2011; Mosleh, 2013; Rudd and Ferrari, 2015), genetic algorithms (GAs) (MacNeil, 2012), genetic programming (GP) (Howard and Roberts, 2001; Kirstukas et al., 2005; Bryden et al., 2006; Sobester et al., 2008; Balasubramaniam and Kumar, 2009; Seaton et al., 2010; Howard et al., 2011), particle swarm optimization (PSO) (Khan et al., 2009; Babaei, 2013), evolution strategies (ES) (Chaquet and Carmona, 2012), differential evolution (Panagant and Bureerat, 2014) or support vector machines (Mehrkanoon and Suykens, 2015).

Finally, a fourth approach can be adopted, where numerical and heuristic methods are combined (He et al., 2000a; El-Emam and Al-Rabeh., 2011).

Although it is out of the present work scope, where we focus on classic differential equations, there is an increasing interest in other types of differential equations such as DE with uncertainty (Allahviranloo et al., 2012; Yao, 2015) or Grey DE (Guo and Guo, 2009).

The most widely used schemes to solve differential equations are mesh-based numerical methods. In this type of schemes, the computational domain must be discretized by means of a *mesh* (also called a *grid*). A mesh is a set of nodes and a type of connectivity. The latter defines how nodes are joined to build computational volumes. In some occasions, the generation of a grid could be even more difficult than obtaining the solution of the equation. Therefore, grid generation is only a means to an end and still has a component of both art and science (Thompson et al., 1999). For example, the general problem of tetrahedral meshing is challenging and remains unsolved (Bronson et al., 2014). Although in recent years mesh generation techniques have improved a lot, it would be premature to state that further improvements are not needed, especially in terms of efficiency and robustness.

An increasing interest in heuristic methods is observed in the literature. Although these methods are less efficient than numerical methods from a computational cost point of view, they have several advantages: a connectivity grid is not needed and they can handle a wide variety of differential equations with a straightforward setup. Moreover, numerical schemes need be adapted for each equation type.

We present a novel heuristic method for solving differential equations. The new approach can cope with linear and nonlinear, ordinary, partial or systems of differential equations. Candidate solutions are expressed as a weighted sum of *Gaussian* functions. The original problem is transformed into an optimization problem that consists in finding the parameters that define each Gaussian. The new optimization problem is solved applying a *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES). Besides, to increase the accuracy of the results, a local search is applied to the best solution found by the CMA-ES using the *Downhill Simplex* (DS) algorithm. To test the proposed method, 32 problems extracted from the literature are successfully solved, providing evidence about the robustness of the algorithm.

A simple comparison with numerical methods is performed using two partial differential equations to show the pros and cons of the proposed algorithm in terms of accuracy, flexibility, storing requirements, robustness and computational elapsed time.

To verify the potential of the approach in a more practical problem, a study of an RLC electric circuit is presented. The method can obtain the dynamic behavior of the circuit in a parametric form taking into account different component values. This allows us to study different analysis and design problems and solve them in a very efficient manner. A linear circuit is solved and the evolved solution is compared with the analytical solution. A non-linear case of the same problem is also commented and solved.

The rest of the paper is organized as follows: In Section 2, CMA-ES and DS methods are briefly introduced. The mathematical problem description is detailed in Section 3. Then, in Section 4, a description of the proposed approach is given. In Section 5, a set of test cases extracted from the literature is described and experimental results are reported. Section 6 gives some qualitative and quantitative comparisons with both other evolutionary approaches and numerical methods. A practical problem in which our method can take advantage is solved in Section 7, where one design exercise of an electric circuit is presented. Finally, the conclusions are outlined in Section 8.

2 Background

CMA-ES (Hansen, 2006) is a stochastic, derivative-free method for numerical optimization of non-linear or non-convex continuous optimization problems. It belongs to the family of evolutionary algorithms. New candidate solutions are sampled according to a multivariate normal distribution. Pairwise dependencies between the variables in this distribution are represented by a covariance matrix.

CMA-ES is considered as the state-of-the-art in evolutionary computation and has been adopted as one of the standard tools for continuous optimization. A lot of applications can be found in the literature (Colutto et al., 2010; Peterson, 2011).

Algorithm 1 : CMA-ES.

```
Input:
    $\sigma_0$: Initial mutation step
    $\xi_0$: Initial mean value of the distribution
    $\mu, \lambda$: Population and offspring sizes
    $F(\cdot)$: Fitness function
Output:
    $\mathbf{S}_1$: Optimum solution.

0: Initialize $C = I, \sigma = \sigma_0$ and $\xi = \xi_0$
1: Until termination criteria do
    for $i = 1 \ldots \lambda$
        $\mathbf{S}_i$=Sample_multivariate_normal_distribution($\xi, \sigma^2 C$)
        $F_i = F(\mathbf{S}_i)$//Evaluate fitness
    Sort solutions: $F(\mathbf{S}_1) \leq F(\mathbf{S}_2) \leq \cdots \leq F(\mathbf{S}_\lambda)$
    Selection and recombination ($\mathbf{S}_1, \cdots, \mathbf{S}_\lambda; \mu$)
    $\xi$ =Update mean
    $\sigma$ =Update step size
    $C$ =Update covariance matrix
2: Return $\mathbf{S}_1$
```

CMA-ES uses a second order approach to estimate a covariance matrix C within an iterative procedure. The covariance matrix is positive definite and plays a similar role to the Hessian matrix in gradient-based algorithms. This makes the method feasible on non-separable and/or badly conditioned problems (Hansen and Kern, 2004). Another interesting CMA-ES property is its feasibility on non-smooth and even non-continuous problems. Finally, it does not require a tedious parameter tuning.

Algorithm 1 describes the main steps of the CMA-ES algorithm. The covariance matrix C is initialized with the identity matrix I. Other parameters, such as the step size σ and the population mean ξ are also initialized at the beginning. Then, in each generation, λ new individuals are sampled from a multivariate normal distribution of mean ξ, and covariance $\sigma^2 C$. The individuals are sorted according to the fitness values. Two main principles are exploited by the CMA-ES algorithm for the adaptation of the distribution parameters. The first one, the *maximum-likelihood* principle, is based on the idea of increasing the probability of generating successful candidate solutions. The mean of the distribution is updated maximizing the likelihood of previous successful candidate solutions. The second principle, called *search* or *evolution paths*, records two paths related to the evolution over time of the mean of the distribution. These paths contain significant information about the correlation between consecutive steps. More details about how the mean, step size and covariance matrix are updated can be consulted in (Hansen, 2011).

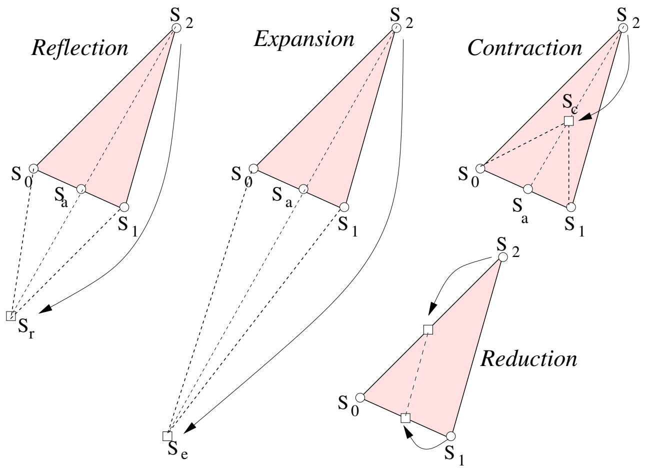

Figure 2: Geometric operations performed by Downhill Simplex method in a simplex. Example for a two-dimensional search space.

As it was commented in the introduction, a local search is applied in a second phase to improve the quality of the evolved solution. For that, the *Downhill Simplex* (DS) algorithm is used, also known as *Nelder-Mead* or *Amoeba* (Nelder and Mead, 1965). DS is a technique for minimizing an objective function in a multidimensional space. The method uses the concept of *simplex*, which is a special polytope of $N+1$ vertices in a N-dimensional space. The algorithm generates a new vertex by extrapolating the behavior of the objective function, measured at each test point arranged as a simplex, and applying geometric rules such as *reflections*, *expansions*, *contractions* and *reductions*. Fig. 2 shows these geometric operations in a two-dimensional problem ($N=2$). The parameters α, β, δ and ζ are, respectively, the *reflection*, *expansion*, *contraction* and *shrink* coefficients. The algorithm then chooses to successively replace the worst of these vertices with the new test point, and so the search progresses. Algorithm 2 sketches the main steps of the Downhill Simplex method.

Algorithm 2 : Downhill Simplex.

```
Input:
  {$\mathbf{S}_0, \cdots, \mathbf{S}_N$}: Initial simplex
  $\alpha, \beta, \delta, \zeta$: Geometric coefficients
  $F(\cdot)$: Fitness function
Output:
  $\mathbf{S}_0$: Optimum solution.

0: Given a simplex $\{\mathbf{S}_0, \cdots, \mathbf{S}_N\}$ while not happy
1: Sort: $F(\mathbf{S}_0) \le F(\mathbf{S}_1) \le \cdots \le F(\mathbf{S}_N)$
2: $\mathbf{S}_a = \frac{1}{N-1}\sum_{i=0}^{N-1}\mathbf{S}_i$
3: $\mathbf{S}_r = \mathbf{S}_a + \alpha(\mathbf{S}_a - \mathbf{S}_N)$ //Reflection
   if $F(\mathbf{S}_0) \le F(\mathbf{S}_r) < F(\mathbf{S}_{N-1})$ then
       $\mathbf{S}_N = \mathbf{S}_r$
       go to 0
4: if $F(\mathbf{S}_r) < F(\mathbf{S}_0)$ then //Expansion
       $\mathbf{S}_e = \mathbf{S}_a + \beta(\mathbf{S}_a - \mathbf{S}_N)$
   if $F(\mathbf{S}_e) < F(\mathbf{S}_r)$ then
       $\mathbf{S}_N = \mathbf{S}_e$
       go to 0
   else
       $\mathbf{S}_N = \mathbf{S}_r$
       go to 0
5: $\mathbf{S}_c = \mathbf{S}_a + \delta(\mathbf{S}_a - \mathbf{S}_N)$ //Contraction
   if $F(\mathbf{S}_c) < F(\mathbf{S}_N)$ then
       $\mathbf{S}_N = \mathbf{S}_c$
       go to 0
6: $\mathbf{S}_i = \mathbf{S}_0 + \zeta(\mathbf{S}_i - \mathbf{S}_0)$    $\forall i \in \{1, \ldots, N\}$//Reduction
   go to 0
```

3 Statement of the problem

Using the same notation as (Sobester et al., 2008) and (Chaquet and Carmona, 2012), we present the problem to solve differential equations expressed by the following general expression:

$$\mathbf{L}\mathbf{y}(\mathbf{x}) = \mathbf{f}(\mathbf{x}) \;\; in \; \Omega \subset \mathbb{R}^d \tag{1}$$

subject to the boundary conditions:

$$\mathbf{B}\mathbf{y}(\mathbf{x}) = \mathbf{g}(\mathbf{x}) \;\; on \; \partial\Omega, \tag{2}$$

where $\mathbf{L}$ and $\mathbf{B}$ are differential operators, $\mathbf{x} \in \mathbb{R}^d$, and $\mathbf{y}(\mathbf{x})$ denotes the unknown solution vector. Functions $\mathbf{f}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ denote source terms, so only depend on $\mathbf{x}$, but not on $\mathbf{y}$ or its derivatives. From a general point of view, $\mathbf{y}(\mathbf{x})$, $\mathbf{f}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ belong to the set of vector-valued functions $\mathbb{R}^d \rightarrow \mathbb{R}^m$. The domain $\Omega \subset \mathbb{R}^d$ is bounded and $\partial\Omega$ denotes its boundary. Strictly speaking, this notation corresponds to elliptic equations, where the boundary conditions must be imposed in the whole domain boundary. In other problems, such as initial value differential equations, boundary conditions are given in a subset of $\partial\Omega$. Note that if $d=1$ and $m=1$, we have an ordinary differential equation (ODE) problem, which can be linear (LODE) or non linear (NLODE). If $d=1$ and $m>1$, a system of differential equations (SODE) problem is managed. Finally, if $d>1$ and $m=1$, a partial differential equation (PDE) problem is established. The solution satisfying (1) and (2) can be computed solving the following constrained optimization problem:

$$\begin{array}{ll} Minimize: & \int_{\Omega} \|\mathbf{L}\mathbf{y}(\mathbf{x}) - \mathbf{f}(\mathbf{x})\|^2 d\mathbf{x} \\ Subject\ to: & \int_{\partial\Omega} \|\mathbf{B}\mathbf{y}(\mathbf{x}) - \mathbf{g}(\mathbf{x})\|^2 d\mathbf{x} = 0 \end{array} \tag{3}$$

where $\|\cdot\|$ denotes the Euclidean norm in $\mathbb{R}^d$ space.

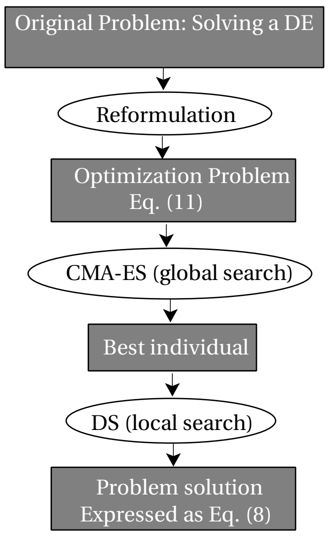
Original Problem: Solving a DE
Reformulation
Optimization Problem Eq. (11)
CMA-ES (global search)
Best individual
DS (local search)
Problem solution Expressed as Eq. (8)

Figure 3: Block diagram of the proposed method. The global search of the Differential Equation (DE) solution is made using the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), and the local search by means of the Downhill Simplex (DS) algorithm.

The problem is then discretized using a set of n_C collocation points situated within the domain:

$$C = \{(\mathbf{x}_i) \mid_{i=1,\cdots,n_C} \subset \Omega\}, \tag{4}$$

and n_B points located on the boundary:

$$B = \{(\mathbf{x}_j) \mid_{j=1,\cdots,n_B} \subset \partial\Omega\}. \tag{5}$$

Finally the original problem is transformed into an optimization problem without constraints where the cost function to minimize is:

$$F(\mathbf{y}) = \sum_{i=1}^{n_C} \|\mathbf{Ly}(\mathbf{x}_i) - \mathbf{f}(\mathbf{x}_i)\|^2 + \sum_{j=1}^{n_B} \|\mathbf{By}(\mathbf{x}_j) - \mathbf{g}(\mathbf{x}_j)\|^2. \tag{6}$$

4 Method Description

The proposed method will be described in this section. A block diagram of the algorithm is shown in Fig. 3. As we can see, the original differential equation is reformulated into a new optimization problem. The solution is obtained making a global search by means of an evolutionary algorithm (CMA-ES). A local search is performed on the best individual obtained by CMA-ES employing a Downhill Simplex (DS) algorithm. The particular coding of candidate solutions and the fitness function used by both search algorithms are explained in subsections 4.1 and 4.2, respectively. Finally, the global search (CMA-ES) and the local search (DS algorithm) are explained in subsections 4.3 and 4.4.

4.1 Candidate solution encoding

Encoding is a very important issue when an evolutionary algorithm is designed. Ideally, the exact solution should be expressed with the encoding selected, and at the same time, this encoding should be as straightforward as possible. In this context, the use of a functional basis considerably reduces the size of the search space. That is, instead of searching generic symbolic expressions, each candidate solution component $y(\mathbf{x})$ is expressed as a linear combination of basis functions. In a previous work by the same authors (Chaquet and Carmona, 2012), a harmonic approach was adopted using a Fourier series expansion. With this strategy, each harmonic affects to the whole solution, needing a complex scheme for handling the *unknowns* (variables whose value is unknown). Additionally, *Radial basis functions* have been adopted here in order to decouple in some way the search of each unknown. A radial function, also called *kernel*, is a real-valued function whose value depends only on the distance from the origin, so that $\phi(\mathbf{x}) = \phi(\|\mathbf{x}\|)$, or, alternatively, on the distance from some other point $\mathbf{c}$, called *center*, so that $\phi(\mathbf{x}, \mathbf{c}) = \phi(\|\mathbf{x} - \mathbf{c}\|)$. The norm $\|\cdot\|$ is usually the Euclidean distance, although other distance functions are also possible. Among the family of radial basis functions, *Gaussian* kernels are chosen because of their good behavior to approximate any continuous function (Hangelbroek and Ron, 2010). A Gaussian kernel is defined as

$$\Phi(\mathbf{x}, \mathbf{c}) = \exp\left(-\gamma \|\mathbf{x} - \mathbf{c}\|^2\right), \tag{7}$$

where $\mathbf{c} \in \mathbb{R}^d$ is a vector defining the center of the Gaussian, and $\gamma > 0$ is a scalar which controls the shape of the kernel. A candidate solution $y(\mathbf{x})$ is then expressed combining n Gaussian kernels in the following way:

$$y(\mathbf{x}) = \sum_{i=1}^{n} w_i \Phi(\mathbf{x}, \mathbf{c}_i) = \sum_{i=1}^{n} w_i \exp\left[-\sum_{j=1}^{d} \gamma_i (x_j - c_{ij})^2\right]. \tag{8}$$

For each Gaussian Φ_i, we have $d+2$ degrees of freedom: d components of the center $\mathbf{c}_i$, the shape parameter γ_i and the weight w_i. Considering all n kernels, the problem can be seen as an optimization problem with

$n \cdot (d+2)$ degrees of freedom. Additionally, if the function to be estimated is a vector function with $m > 1$ dimensions in the output domain (SODE), we apply the previous ideas to each dimension. Therefore, the following degrees of freedom or *number of unknowns* (variables) are obtained:

$$N = m \cdot n \cdot (d+2) \, . \tag{9}$$

For instance, in the case of an ordinary DE ($m = d = 1$), each genotype would be formed by:

$$[w_1, \gamma_1, c_1, \cdots, w_n, \gamma_n, c_n] \, . \tag{10}$$

Finally, as it was stated in Section 3, the original problem has been transformed into an optimization problem without constraints. In particular, the shape parameters γ_i are not limited to positive values. Although standard Gaussian kernels have positive values of γ_i, no constrains are applied to this type of parameter because better convergence has been observed when $\gamma_i \in \mathbb{R}$. Note that a Gaussian kernel is not a probability distribution function. Therefore, because the concept of probability is not involved in the kernel computation, it is possible to use gamma values less than or equal to zero. In this case, we could speak of *degenerate Gaussian kernels*. For example, for $\gamma = 0$, a constant function can be approximated with only one kernel. On the other hand, when $\gamma < 0$, the resulting degenerate Gaussian kernel is not bounded above in $(-\infty, +\infty)$, that is, $\lim_{x \to \pm\infty} \Phi(x, c) = \infty$. However, this is not a problem because we are only interested in solving differential equations in bounded domains. In section 5.2, an example with $\gamma < 0$ will be provided in order to show the benefits of not limiting the value of this parameter.

4.2 Fitness Function

A fitness function assigns to each individual in the population a measure of how close it is to achieve its aims, i. e., how close the individual represents a correct solution of the differential equation. In the present work, a fitness function is built modifying the cost function given by Eq. (6) as follows:

$$F(\mathbf{y}) = \left[\sum_{i=1}^{n_C} \xi(\mathbf{x}_i) \left\| \mathbf{L}\mathbf{y}(\mathbf{x}_i) - \mathbf{f}(\mathbf{x}_i) \right\|^2 + \right. \\ \left. \varphi \sum_{j=1}^{n_B} \left\| \mathbf{B}\mathbf{y}(\mathbf{x}_j) - \mathbf{g}(\mathbf{x}_j) \right\|^2 \right] / \left[m \cdot (n_C + n_B) \right] , \tag{11}$$

where a weighting factor, $\xi(\mathbf{x}) \in \mathbb{R}^+$, only dependent on the collocation points, $\mathbf{x}_i$, and a penalty parameter, φ, are introduced. Note that the cost function is obtained dividing the residuals by the total number of collocation points $m \cdot (n_C + n_B)$ in a similar way as in (Parisi et al., 2003; Chaquet and Carmona, 2012). Other authors (Lagaris et al., 1998; Sobester et al., 2008; Balasubramaniam and Kumar, 2009) do not make this normalization, which makes the fitness function values more dependent on the number of collocation points. The penalty parameter φ controls the relative weights assigned to boundary condition points compared with the inner collocation points. Note that, typically, $n_C \gg n_B$.

The weighting factor $\xi(\mathbf{x})$ can be used to modify the algorithm convergence behavior increasing the relative errors in some domain locations. In the present work, $\xi(\mathbf{x})$ is employed to increase the weights of the inner collocation points closer to boundary points in the following way:

$$\xi(\mathbf{x}_i) = \frac{1 + \kappa \left(1 - \frac{\min_{\forall \mathbf{x}_j \in B} \|\mathbf{x}_i - \mathbf{x}_j\|}{\max_{\forall \mathbf{x}_k \in C} \left(\min_{\forall \mathbf{x}_j \in B} \|\mathbf{x}_k - \mathbf{x}_j\| \right)} \right)}{1 + \kappa} , \tag{12}$$

where $\kappa \geq 0$ is a user parameter called *inner weighting factor*. The expression (12) assigns a maximum value of 1 to all the collocation points closest to the boundary $\partial\Omega$, and a value of $1/(1+\kappa)$ to those collocation points located at a maximum distance from $\partial\Omega$, i. e., from the most interior points. When $\kappa = 0$, the standard cost function is recovered because $\xi(\mathbf{x}) = 1$. Note that the weighting factor only depends on the collocation points. Threfore, it can be computed in a pre-processing step, not adding any extra computational cost to the algorithm. The method maintains its mesh-free quality because connectivity is not needed to obtain $\xi(\mathbf{x})$.

According to Eq. (11), not only the candidate solution is needed, but also its derivatives. For that reason, we have chosen the Gaussian kernel: it is infinitely differentiable and its derivatives can be easily precalculated. Nevertheless, there is not any general expression to obtain all the derivatives for any arbitrary order as it is the case of trigonometric functions (Chaquet and Carmona, 2012). The first derivative of the Gaussian kernel respect to component k is:

$$\frac{\partial y(\mathbf{x})}{\partial x_k} = -2 \sum_{i=1}^{n} w_i \gamma_i (x_k - c_{ik}) \, \Phi(\mathbf{x}, \mathbf{c}_i) \, . \tag{13}$$

The second derivative is a bit more complex. If $k \neq l$ we have:

$$\frac{\partial^2 y(\mathbf{x})}{\partial x_k \partial x_l} = 4 \sum_{i=1}^{n} w_i \gamma_i^2 (x_k - c_{ik}) (x_l - c_{il}) \, \Phi(\mathbf{x}, \mathbf{c}_i) \, , \tag{14}$$

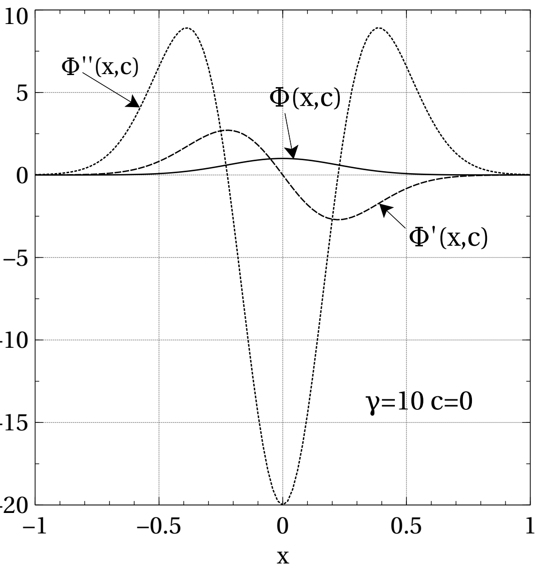

Figure 4: A Gaussian kernel $\Phi(x,c)$ and its first and second derivatives.

otherwise:

$$\frac{\partial^2 y(\mathbf{x})}{\partial x_k^2} = \sum_{i=1}^{n} w_i \gamma_i \left[4\gamma_i (x_k - c_{ik})^2 - 2\right] \Phi(\mathbf{x}, \mathbf{c}_i). \quad (15)$$

Higher order derivatives can be also obtained by hand. In Fig. 4, a Gaussian kernel $\Phi(x,c)$ in one dimension with center in the origin $c=0$ and $\gamma=10$ is plotted. The first and second derivatives are also plotted. We can observe that although the function's influence zone is located near its center, this is not true for the first and second derivatives. In fact, the first derivative has a zero value at the center, a global maximum at $x = c - 1/\sqrt{2\gamma}$ and a global minimum at $x = c + 1/\sqrt{2\gamma}$. The second derivative has two local maxima and one global minimum. $\Phi(x,c)$ and $\Phi''(x,c)$ are even functions, whereas $\Phi'(x,c)$ is an odd function. Therefore, the problem of adjusting simultaneously the values and derivatives of a function using kernels is not straightforward.

4.3 Global search of the solution: CMA-ES

Among the different paradigms of evolutionary computation, CMA-ES has been selected. As can be inferred from (Chaquet and Carmona, 2012) work, those problems that result from transforming a DE into an optimization problem, can be non-separable. In this kind of problems, CMA-ES has proved to be competitive.

CMA-ES is used to search the optimal weights, centers and gammas that minimize the fitness function expressed in Eq. (11). First of all, the unknowns are randomly initialized. As we see in Fig. 4, the influence zones of the kernel derivatives are located far away from their centers. Therefore, the initial values of centers, c_{ik}, are set randomly in an *extended range* in the form:

$$c_{ik} \in [x_{k,min} - \beta R_k, x_{k,max} + \beta R_k] \quad (16)$$

being $R_k = x_{k,max} - x_{k,min}$ the original range width for dimension k-th, and β an initialization parameter which controls the extended range width. The domain range $[x_{k,min}, x_{k,max}]$ is sampled with all the collocation points within sets C and B (see Eq. (4) and (5), respectively). As it was commented in Section 4.1, no constraints are considered in the variables (unknowns). Therefore it must be said that Eq. (16) is only applied on the first generation, but not in the rest of generations.

All the default parameters for CMA-ES proposed by Hansen in his CMA-ES public implementation (Hansen, 2011) are adopted, except the offspring number, λ, and the population size, μ. The default value for the offspring number is:

$$\lambda_{default} = 4 + \lfloor 3 \ln N \rfloor, \quad (17)$$

and the population size is obtained dividing by 2:

$$\mu_{default} = \lfloor \lambda_{default}/2 \rfloor. \quad (18)$$

In the above expressions, N is the number of unknowns. In our case, the offspring number is slightly increased multiplying the default value by a constant and, consequently, given that μ depends on λ, the population size is also increased. The final values for this couple of parameters were

$$\left.\begin{array}{c} \lambda = 3 \cdot \lambda_{default} \\ \mu = \lfloor \lambda/2 \rfloor = \lfloor 3 \cdot \lambda_{default}/2 \rfloor \approx 3 \cdot \mu_{default} \end{array}\right\}. \quad (19)$$

The value of N, necessary to compute $\lambda_{default}$, is given by the Eq. (9). These changes have experimentally shown to produce better results and decrease the dispersion between different runs. This strategy is in line with the results obtained in (Hansen and Kern, 2004), where the benefit of increasing the value of λ (with respect to its default value) in multimodal problems with high dimensionality is experimentally shown.

Finally, CMA-ES is run (see Algorithm 1) and several generations are computed until some of the default stop criteria (Hansen, 2011) are met. Then, the best individual found in the evolutionary process is returned by the algorithm.

Table 1: Test cases: differential equations, ranges and boundary conditions. Last column shows the papers where each problem is solved. The following notation is used: [a] Tsoulos and Lagaris (2006), [b] Tsoulos et al. (2009), [c] Chaquet and Carmona (2012), [d] Lagaris et al. (1998), [e] Yazdi and Pourreza (2010), [f] Chen et al. (2011), [g] Babaei (2013), [h] Sobester et al. (2008) and [i] Panagant and Bureerat (2014).

Case	**Equation**	**Range**	**Boundary Conditions**	**Used in**
LODE1	$y' = (2x - y)/x$	$x \in [1, 2]$	$y(1) = 3$	[a,b,c]
LODE2	$y' = (1 - y\cos(x))/\sin(x)$	$x \in [1, 2]$	$y(1) = 3/\sin 1$	[a,b,c]
LODE3	$y'' = 6y' - 9y$	$x \in [0, 1]$	$y(0) = 0; y'(0) = 2$	[a,b,c]
LODE4	$y'' + \frac{1}{5}y' + y = -\frac{1}{5}e^{-x/5}\cos(x)$	$x \in [0, 1]$	$\left.\begin{array}{c} y(0) = 0 \\ y'(0) = \sin(0.1)/e^{0.2} \end{array}\right\}$	[a,b,c]
LODE5	$xy'' + y' = \cos(x)$	$x \in [0, 1]$	$y(0) = 0; y'(0) = 1$	[a,b,c,d]
LODE6	$y'' + 2xy = 0$	$x \in [0, 1]$	$y(0) = 0; y'(0) = 1$	[a,b,c]
LODE7	$y''\left(x^2 + 1\right) - 2xy - x^2 - 1 = 0$	$x \in [0, 1]$	$y(0) = 0; y'(0) = 1$	[a,b,c]
LODE8	$y' + 2y = 1$	$x \in [0, 10]$	$y(0) = 1$	[c,e]
LODE9	$y' + 2y = \sin(x)$	$x \in [0, 10]$	$y(0) = 1$	[c,e]
LODE10	$y'' = -16\pi^2\sin(4\pi x)$	$x \in [0, 1]$	$y(0) = 2; y(1) = 2$	[c,f]
LODE11	$u'' + 2u' + 5u = 0$	$x \in [0, \pi]$	$u(0) = 0; u'(0) = 1$	[g]
NLODE1	$y' = 1/(2y)$	$x \in [1, 4]$	$y(1) = 1$	[a,b,c]
NLODE2	$(y')^2 + \log y = \cos^2 x + 2\cos x + 1 + \log(x + \sin x)$	$x \in [1, 2]$	$y(1) = 1 + \sin 1$	[a,b,c]
NLODE3	$y''y' = -4/x^3$	$x \in [1, 2]$	$y(1) = 0; y'(1) = 2$	[a,b,c]
NLODE4	$x^2y'' + \left(xy'\right)^2 + 1/\log x = 0$	$x \in [e, 2e]$	$y(e) = 0; y'(e) = 1/e$	[a,b,c]
NLODE5	$y'' - yy'/\left(x\sin x^2\right) = -4x^2\sin x^2$	$x \in [1, 2]$	$y(1) = \sin 1; y(2) = \sin 4$	[c]
NLODE6	$10^{-4}y'' + y - y^3 = 0$	$x \in [-1, 1]$	$y(-1) = -1; y(1) = 1$	[c,f]
SODE1	$\left.\begin{array}{c} y_1' = \cos x + y_1^2 + y_2 - \left(x^2 + \sin^2 x\right) \\ y_2' = 2x - x^2\sin x + y_1y_2 \end{array}\right\}$	$x \in [0, 1]$	$\left.\begin{array}{c} y_1(0) = 0 \\ y_2(0) = 0 \end{array}\right\}$	[a,b,c,d]
SODE2	$\left.\begin{array}{c} y_1' = (\cos x - \sin x)/y_2 \\ y_2' = y_1y_2 + e^x - \sin x \end{array}\right\}$	$x \in [0, 1]$	$\left.\begin{array}{c} y_1(0) = 0 \\ y_2(0) = 1 \end{array}\right\}$	[a,b,c]
SODE3	$\left.\begin{array}{c} y_1' = \cos x \\ y_2' = -y_1 \\ y_3' = y_2 \\ y_4' = -y_3 \\ y_5' = y_4 \end{array}\right\}$	$x \in [0, 1]$	$\left.\begin{array}{c} y_1(0) = 0 \\ y_2(0) = 1 \\ y_3(0) = 0 \\ y_4(0) = 1 \\ y_5(0) = 0 \end{array}\right\}$	[a,b,c]
SODE4	$\left.\begin{array}{c} y_1' = -\sin(e^x)/y_2 \\ y_2' = -y_2 \end{array}\right\}$	$x \in [0, 1]$	$\left.\begin{array}{c} y_1(0) = \cos 1 \\ y_2(0) = 1 \end{array}\right\}$	[a,b,c]
SODE5	$\left.\begin{array}{c} I' = -I - V \\ V' = 2I - V \end{array}\right\}$	$t \in [0, 1.5]$	$\left.\begin{array}{c} I(0) = 2 \\ V(0) = 2 \end{array}\right\}$	[g]
SODE6	$\left.\begin{array}{c} \left[1 + \left(y'\right)^2\right] y = k^2 \\ k' = 0 \end{array}\right\}$	$x \in [0, 1]$	$\left.\begin{array}{c} y(0) = 0 \\ y(1) = 2 \end{array}\right\}$	[g]
PDE1	$\Psi_{xx} + \Psi_{yy} = e^{-x}\left(x - 2 + y^3 + 6y\right)$	$x, y \in [0, 1]$	$\left.\begin{array}{c} \Psi(0, y) = y^3 \\ \Psi(1, y) = \left(1 + y^3\right)e^{-1} \\ \Psi(x, 0) = xe^{-x} \\ \Psi(x, 1) = (x + 1)e^{-x} \end{array}\right\}$	[a,b,c,d,h,i]
PDE2	$\Psi_{xx} + \Psi_{yy} = -2\Psi$	$x, y \in [0, 1]$	$\left.\begin{array}{c} \Psi(0, y) = 0 \\ \Psi(1, y) = \sin(1)\cos(y) \\ \Psi(x, 0) = \sin(x) \\ \Psi(x, 1) = \sin(x)\cos(1) \end{array}\right\}$	[a,b,c,i]
PDE3	$\Psi_{xx} + \Psi_{yy} = 4$	$x, y \in [0, 1]$	$\left.\begin{array}{c} \Psi(0, y) = y^2 + y + 1 \\ \Psi(1, y) = y^2 + y + 3 \\ \Psi(x, 0) = x^2 + x + 1 \\ \Psi(x, 1) = x^2 + x + 3 \end{array}\right\}$	[a,b,c,i]
PDE4	$\Psi_{xx} + \Psi_{yy} = -\Psi\left(x^2 + y^2\right)$	$x, y \in [0, 1]$	$\left.\begin{array}{c} \Psi(0, y) = 0 \\ \Psi(1, y) = \sin(y) \\ \Psi(x, 0) = 0 \\ \Psi(x, 1) = \sin(x) \end{array}\right\}$	[a,c,i]
PDE5	$\Psi_{xx} + \Psi_{yy} = 4x\cos x + \left(5 - x^2 - y^2\right)\sin x$	$x^2 + y^2 \leq 1$	$\Psi(x, y) = 0 \; in \; \partial\Omega$	[c,h,i]
PDE6	$\Psi_{xx} + \Psi_{yy} = 2e^{(x-y)}$	$R^2(\theta) \leq \cos(2\theta) + \sqrt{1.1\sin^2(2\theta)}$	$\left.\begin{array}{c} \Psi = e^x\left(e^{-y} + \cos y\right) \\ in \; \partial\Omega \end{array}\right\}$	[c,h,i]
PDE7	$u_x + u_t = 0$	$x, t \in [0, 2]$	$\left.\begin{array}{c} u(0, t) = e^{-2(t-1)^2} \\ u(x, 0) = e^{-2(x+1)^2} \end{array}\right\}$	-
PDE8	$u_{xx} - u_{tt} = 0$	$x, t \in [0, 1]$	$\left.\begin{array}{c} u(0, t) = 0 \\ u(1, t) = 0 \\ u(x, 0) = 0 \\ u_t(x, 0) = \pi\sin(\pi x) \end{array}\right\}$	-
PDE9	$u_t - uu_x = u_{xx}$	$x, t \in [0, 1]$	$\left.\begin{array}{c} u(0, t) = 1 + 2/(t + 1) \\ u(x, 0) = 1 + 2/(x + 1) \\ u_x(0, t) = -2/(t + 1)^2 \end{array}\right\}$	-

4.4 Local search of the solution: Downhill Simplex method

Once the CMA-ES algorithm finishes, a Downhill Simplex (DS) method is applied on the best individual obtained in the last generation of the evolutionary algorithm. The first simplex is computed applying sequentially a random increment to each unknown, maintaining the rest of unknowns unmodified. Expressed mathematically, let $\mathbf{S}_0$ be the unknown vector obtained from the CMA-ES, and formed by all the parameters: w_i, γ_i, and c_{ij}. As it was commented in subsection 4.1, the size of that vector is $N = n \cdot m \cdot (2 + d)$. A simplex is formed by the $N+1$ nodes $\{\mathbf{S}_0, \mathbf{S}_1, \cdots, \mathbf{S}_N\}$. The nodes of the initial simplex $\mathbf{S}_k = \{s_{k1}, s_{k2}, \cdots, s_{kN}\}$, $\forall k = 1, ..., N$, are computed applying the following rule:

$$s_{kl} = \begin{cases} s_{0l}, & if\ k \neq l \\ s_{0l} + \rho_k, & if\ k = l \end{cases} \quad k = 1, \cdots, N, \tag{20}$$

where ρ_k is a random number extracted from a uniform distribution over the range $[-\triangle, \triangle]$, being $\triangle \in \mathbb{R}^+$ a user parameter.

Then, several iterations on the simplex are performed applying the DS rules (see Fig. 2). Fitness values of the simplex nodes drive the search process, as it is sketched in Algorithm 2. Standard values for the geometric coefficients are $\alpha = 1$, $\beta = 2$, $\delta = -1/2$ and $\zeta = 1/2$. They have been kept constant in all the runs of the preset work. Several stop criteria are checked: a maximum number of fitness evaluations, a minimum distance from the best and worst simplex nodes (*parameter convergence criterion*), and a minimum fitness value difference from the best and worst nodes (*target convergence criterion*).

DS method, as other heuristics algorithms, can converge to local optimum depending on initialization issues. To reduce this effect, the algorithm is restarted several times recomputing the first simplex.

5 Experiments and Results

This section presents the experimental results obtained applying the proposed method to solve a set of differential equations extracted from the literature. Subsection 5.1 introduces all the test cases (equations, boundary conditions and exact solutions). The references where these problems have been used are provided in order to facilitate a further benchmarking analysis. Subsection 5.2 gives the results obtained by the method, considering fitness function, errors and number of fitness evaluations. Finally, subsection 5.3 performs a sensitivity analysis of some parameters of the algorithm, such as the number of kernels.

5.1 Test cases

The proposed method has been applied to a set of 32 differential equations extracted from the literature (Lagaris et al., 1998; Tsoulos and Lagaris, 2006; Sobester et al., 2008; Tsoulos et al., 2009; Yazdi and Pourreza, 2010; Chen et al., 2011; Chaquet and Carmona, 2012; Babaei, 2013; Panagant and Bureerat, 2014). In order to verify the capabilities of the proposed approach, a wide range of problems is studied: LODEs, NLODEs, SODEs and PDEs. Table 1 shows all the problems used in the present work and the references of papers where they were also solved.

Several comments on some analyzed equations should be made. For example, the original problem LODE11 was presented in (Babaei, 2013) as an integro-differential equation $u' + 2u + 5\int_0^x u(t)\,dt = 1$ in the range $[0, \pi]$ with the boundary condition $u(0) = 0$. Taking derivatives and using the fundamental theorem of calculus (Spivak, 1980), the original problem is transformed into an ordinary differential equation as it is presented in Table 1.

In relation to SODE problems, SODE5 and SODE6 were also presented in (Babaei, 2013). The former describes a simple engineering application dealing with electric current and potential difference in an electric circuit. The later describes the *brachistochrome* problem, also known as the *curve of fastest decent*, which consists in finding the curve along with a particle slides from a given point to a lower one without friction in the shortest time. The problem is formulated as a system of differential equations because a constant k must be determined.

In relation to PDE problems, note that PDE5 and PDE6 are defined on non-rectangular domains. Besides, new partial differential equations, different from *Poisson* problems presented in (Sobester et al., 2008; Chaquet and Carmona, 2012), are defined. For example, PDE7 describes a *traveling wave* and PDE8 corresponds to the one dimensional *wave* equation with constant speed of wave propagation $c = 1$. Finally, PDE9 is the inviscid *Burgers' equation* which is a prototype for equations which can develop discontinuities (shock waves).

The exact solutions are provided in Table 2. The only exception is NLODE6, also known as the *Allen-Cahn* equation (Chen et al., 2011), which does not have a close analytical solution. The SODE6 solution is the parametric equation of a *cycloid.* The constant $k^2 \simeq 4.81125$ is obtained solving a transcendental equation.

Table 2: Exact solutions for the test cases.

Case	**Exact Solution**
LODE1	$y = x + 2/x$
LODE2	$y = (x+2)/\sin(x)$
LODE3	$y = 2xe^{3x}$
LODE4	$y = e^{-x/5}\sin(x)$
LODE5	$y = \int_0^x \frac{\sin(t)}{t} dt$
LODE6	$y = \int_0^x e^{-t^2} dt$
LODE7	$y = (x^2+1)\arctan(x)$
LODE8	$y = (e^{-2x}+1)/2$
LODE9	$y = [6e^{-2x} + 2\sin(x) - \cos(x)]/5$
LODE10	$y = 2 + \sin(4\pi x)$
LODE11	$u = 0.5e^{-x}\sin(2x)$
NLODE1	$y = \sqrt{x}$
NLODE2	$y = x + \sin(x)$
NLODE3	$y = \log(x^2)$
NLODE4	$y = \log(\log(x))$
NLODE5	$y = \sin(x^2)$
NLODE6	No analytical solution
SODE1	$\left.\begin{array}{c} y_1 = \sin x \\ y_2 = x^2 \end{array}\right\}$
SODE2	$\left.\begin{array}{c} y_1 = \sin(x)/e^x \\ y_2 = e^x \end{array}\right\}$
SODE3	$\left.\begin{array}{c} y_1 = y_3 = y_5 = \sin x \\ y_2 = y_4 = \cos x \end{array}\right\}$
SODE4	$\left.\begin{array}{c} y_1 = \cos(e^x) \\ y_2 = e^{-x} \end{array}\right\}$
SODE5	$\left.\begin{array}{c} I = e^{-t}\left[2\cos\left(\sqrt{2}t\right) - \sqrt{2}\sin\left(\sqrt{2}t\right)\right] \\ V = 2e^{-t}\left[\cos\left(\sqrt{2}t\right) + \sqrt{2}\sin\left(\sqrt{2}t\right)\right] \end{array}\right\}$
SODE6	$\left.\begin{array}{c} x = k^2(\theta - \sin\theta)/2 \\ y = k^2(1 - \cos\theta)/2 \\ \frac{2}{k^2} = \arccos\left(1 - \frac{4}{k^2}\right) - \sin\arccos\left(1 - \frac{4}{k^2}\right) \end{array}\right\}$
PDE1	$\Psi = (x + y^3)e^{-x}$
PDE2	$\Psi = \sin(x)\cos(y)$
PDE3	$\Psi = x^2 + y^2 + x + y + 1$
PDE4	$\Psi = \sin(xy)$
PDE5	$\Psi = (x^2 + y^2 - 1)\sin x$
PDE6	$\Psi = e^{(x-y)} + e^x\cos y$
PDE7	$u = e^{-2(t-x-1)^2}$
PDE8	$u = \sin(\pi t)\cdot\sin(\pi x)$
PDE9	$u = 1 + 2/(x + t + 1)$

5.2 Results

The proposed method has been implemented in C++ and we have integrated the public implementation in ANSI C of CMA-ES algorithm provided by Hansen (Hansen, 2011). The source code has been compiled with GNU gcc compiler using Ubuntu Linux operating system. The runs have been performed in a PC with Intel Core 2 Quad CPU Q9400 at 2.66GHz and 3.8 GB of RAM memory. The memory requirements of the test problems are not demanding.

Due to the stochastic nature of the method, it was run 50 times on each differential equation described previously, using different seeds for the random number generator; averages and standard deviations were also computed. A total of 100 equidistant collocation points have been used in all cases, except in those partial differential equations defined in a non-rectangular domain (PDE5 and PDE6). In these two cases, the same point distribution used in (Sobester et al., 2008) and (Chaquet and Carmona, 2012) was imposed (77 and 80 points, respectively) in order to make a fair comparison.

In the same way as (Chaquet and Carmona, 2012), the *Root of the Mean Squared Error* (RMSE) between the final evolved solution $\mathbf{y}$ and the exact solution $\mathbf{y}_{exact}$ is computed to measure the quality of the solutions obtained by the method:

$$RMSE^2 = \left[\sum_{i=1,\mathbf{x}_i\in C}^{n_C} \left\|\mathbf{y}(\mathbf{x}_i) - \mathbf{y}_{exact}(\mathbf{x}_i)\right\|^2 + \sum_{j=1,\mathbf{x}_j\in B}^{n_B} \left\|\mathbf{y}(\mathbf{x}_j) - \mathbf{y}_{exact}(\mathbf{x}_j)\right\|^2\right] / \left[m\cdot(n_C + n_B)\right]. \tag{21}$$

The fitness function value depends on how the differential equations are provided to Eq. (11). For instance, LODE1 equation can be provided as $y' - (2x - y)/x$, $y'x - (2x - y)$ or even $[y' - (2x - y)/x]/k$. Although the same problem is solved, the fitness functions are different. In particular, the first and third functions have a similar fitness landscape, but with a different scale factor given by the constant k. The second function has a different fitness landscape, so could have a different behavior during the evolving process. Nevertheless, RMSE provides a good way to analyze the quality of the solution found because is independent of the mentioned problems, and allows us to compare more fairly the results of the different methods published in the literature. Obviously, RMSE can be only used when the exact solution is known. It is important to highlight that the RMSE values are not used to solve the differential equations, only the fitness function defined by Eq. (11) is used by the evolutionary and local search algorithms.

Because NLODE6 does not have analytical solution, the *exact* solution for this problem has been obtained using a numerical approximation computed with a finite difference approach solving the discretized equation with a four order Runge-Kutta relaxation method (Press et al., 2002). The domain range is split in 1000 nodes.

Tables 3 and 4 list the algorithm parameter configuration used in all test cases regarding the CMA-ES and DS methods. As it was commented in Section 4.3, all default CMA-ES parameters are adopted, except the size of the population and the offspring number. Therefore, according to Eq. (9) and Eq. (19), the population

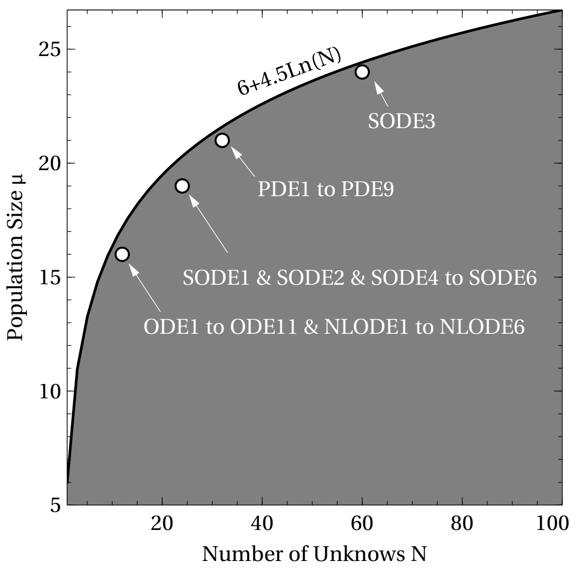

Figure 5: Population size μ of the CMA-ES algorithm according to the number of unknowns N. Dots mark the population size used to solve the different types of test problems.

size μ is related with the number of unknowns N:

$$\mu = \left\lfloor 6 + \frac{3}{2} \lfloor 3 \cdot \ln N \rfloor \right\rfloor . \tag{22}$$

Fig. 5 plots the relation between μ and N for all the test problems. The number of unknowns depends on the number of kernels n, the number of dependent variables m, and the problem dimensionality d, according to Eq. (9). The experimental criterion used to select the number of kernels was the following: four kernels for each dependent variable in LODEs, NLODEs and SODEs, and eight in PDEs. Thus, for LODEs and NLODEs, the number of kernels was always four. On the other hand, if the number of equations (dependent variables) in a SODE is m, the number of kernels used was $4m$. Regarding PDEs, all of them have two independent variables, which make the problem harder to be solved compared with ODEs. Therefore, the number of kernels for PDEs has been doubled respect to ODEs. As we can see in Fig. 5, SODE3 has more unknowns than the other SODEs because five dependent variables must be solved ($m = 5$) instead of only two ($m = 2$) for the rest of SODEs. The centers are initialized randomly in an extended range whose size is controlled by a user parameter, β, according to Eq. (16).

Some parameters affect both CMA-ES and DS algorithms: the boundary condition penalty φ is set to

Table 3: CMA-ES parameter configuration.

Parameter	Values
Initial weights	Randomly in $w_i \in [-0.01, 0.01]$
Initial γ_i	Randomly in $\gamma_i \in (0, 1]$
Initial centers c_{ik}	Randomly, Eq. (16) and $\beta = 2$
Initial mutation step σ	0.01
Offspring number	$\lambda = 3 \cdot \lambda_{default}$ (see Eq. (17))
Population size	$\mu = \lfloor \lambda/2 \rfloor$
Stop criteria	Default, see (Hansen, 2011)

Table 4: DS parameter values.

Parameter	Values
Number of restarts	10
Increment for first simplex	$\triangle = 10^{-2}$. Eq. (20).
Stop criterion	Fit. evaluations $= 2 \cdot 10^4$
Parameter conv. criterion	10^{-20}
Target conv. criterion	10^{-20}

300, the inner weighting factor κ used is 30, and the maximum number of fitness evaluations allowed is 10^6.

The results are given in Table 5. Data are grouped in two categories: those obtained at the end of the CMA-ES phase, and the final ones at the end of the DS phase. Average values and the standard deviations are reported for each problem. By construction of the local search algorithm (see Section 4.4), its results (column marked as “Fitness CMA-ES+DS”) always equal or outperform the fitness values of the CMA-ES algorithm (column marked as “Fitness CMA-ES”). In some cases, the DS phase improves significantly the fitness value by several orders of magnitude as in LODE3, LODE5, LODE6, NLODE3, NLODE4 or SODE4 problems.

In order to obtain the percentage of cases in which the DS phase improves the RMSE value obtained from the evolutionary algorithm’s best solution, a statistical significance test was performed. A *Kolmogorov-Smirnov* normality test revealed that the null hypothesis was rejected in 94% of the analyzed problems ($\alpha = 0.05$). Therefore, a non-parametric *Wilcoxon sum-rank* test (one-tailed) was applied. The results of this test shown that the median of RMSE values after applying the local search was lower than the median of RMSE values associated to CMA-ES phase in 34% of the analyzed problems ($\alpha = 0.05$). Therefore, we can say that the DS phase is useful and allows us to improve (or maintain) the results obtained by the evolutionary algorithm, without modifying the control parameters, and with a limited cost, as we can see comparing the number of fitness evaluation before and after the DS phase (two last columns of the table 5).

Table 5: Experimental results. Each case was run 50 times using different seeds for the random number generator. In columns "Fitness" and "RMSE", the best values obtained are highlighted in bold letters.

Case	n (Eq. (9))	Fitness (CMA-ES)	Fitness (CMA-ES+DS)	RMSE (CMA-ES)	RMSE (CMA-ES+DS)	Fitness Eval.$\times 10^5$ (CMA-ES)	Fitness Eval.$\times 10^5$ (CMA-ES+DS)
LODE1	4	$(8.73 \pm 34.5)\,10^{-8}$	$\mathbf{(1.18 \pm 2.87)\,10^{-9}}$	$(1.41 \pm 3.77)\,10^{-5}$	$\mathbf{(2.35 \pm 2.52)\,10^{-6}}$	1.08 ± 0.481	1.92 ± 0.473
LODE2	4	$(7.32 \pm 47.7)\,10^{-7}$	$\mathbf{(6.82 \pm 47.7)\,10^{-7}}$	$(2.70 \pm 14.4)\,10^{-5}$	$\mathbf{(2.14 \pm 14.4)\,10^{-5}}$	0.958 ± 0.749	1.77 ± 0.802
LODE3	4	$(1.42 \pm 8.87)\,10^{-1}$	$\mathbf{(3.47 \pm 17.5)\,10^{-4}}$	$(2.51 \pm 16.1)\,10^{-1}$	$\mathbf{(2.13 \pm 7.53)\,10^{-3}}$	0.691 ± 0.215	1.71 ± 0.312
LODE4	4	$(4.31 \pm 20.2)\,10^{-5}$	$\mathbf{(1.81 \pm 11.6)\,10^{-5}}$	$(2.08 \pm 9.50)\,10^{-4}$	$\mathbf{(1.03 \pm 6.20)\,10^{-4}}$	0.928 ± 0.656	1.64 ± 0.678
LODE5	4	$(4.45 \pm 0.222)\,10^{-7}$	$\mathbf{(7.32 \pm 0.373)\,10^{-10}}$	$(5.64 \pm 26.2)\,10^{-5}$	$\mathbf{(3.01 \pm 8.13)\,10^{-6}}$	1.04 ± 0.570	1.71 ± 0.723
LODE6	4	$(2.71 \pm 18.0)\,10^{-8}$	$\mathbf{(9.87 \pm 42.2)\,10^{-10}}$	$(1.07 \pm 5.75)\,10^{-6}$	$\mathbf{(1.72 \pm 2.85)\,10^{-7}}$	1.14 ± 0.786	1.78 ± 0.856
LODE7	4	$(4.42 \pm 17.9)\,10^{-6}$	$\mathbf{(2.39 \pm 9.47)\,10^{-6}}$	$(1.40 \pm 5.30)\,10^{-5}$	$\mathbf{(6.71 \pm 2.33)\,10^{-6}}$	2.77 ± 1.73	3.41 ± 1.65
LODE8	4	$(1.20 \pm 2.34)\,10^{-7}$	$\mathbf{(2.96 \pm 8.54)\,10^{-8}}$	$(9.07 \pm 8.41)\,10^{-5}$	$\mathbf{(4.87 \pm 5.41)\,10^{-5}}$	0.439 ± 0.161	1.43 ± 0.235
LODE9	4	$(1.01 \pm 0.403)\,10^{-1}$	$\mathbf{(9.70 \pm 4.26)\,10^{-2}}$	$(2.18 \pm 0.770)\,10^{-1}$	$\mathbf{(2.16 \pm 0.773)\,10^{-1}}$	0.825 ± 0.281	1.76 ± 0.405
LODE10	4	$(1.22 \pm 1.14)\,10^{3}$	$\mathbf{(9.28 \pm 10.2)\,10^{2}}$	$\mathbf{1.88 \pm 1.33}$	1.90 ± 1.41	0.522 ± 0.234	1.41 ± 0.502
LODE11	4	$(4.32 \pm 20.4)\,10^{-4}$	$\mathbf{(2.24 \pm 14.3)\,10^{-4}}$	$(1.40 \pm 5.41)\,10^{-3}$	$\mathbf{(9.16 \pm 39.4)\,10^{-4}}$	2.32 ± 1.44	2.78 ± 1.44
NLODE1	4	$(1.05 \pm 2.26)\,10^{-7}$	$\mathbf{(6.61 \pm 18.2)\,10^{-9}}$	$(6.40 \pm 8.70)\,10^{-5}$	$\mathbf{(1.61 \pm 2.12)\,10^{-5}}$	0.914 ± 0.446	1.88 ± 0.453
NLODE2	4	$(2.79 \pm 19.5)\,10^{-4}$	$\mathbf{(2.79 \pm 19.45)\,10^{-4}}$	$(4.84 \pm 33.8)\,10^{-4}$	$\mathbf{(4.83 \pm 33.8)\,10^{-4}}$	1.10 ± 0.589	1.77 ± 0.686
NLODE3	4	$(1.34 \pm 7.1)\,10^{-2}$	$\mathbf{(1.19 \pm 5.02)\,10^{-7}}$	$(4.83 \pm 24.2)\,10^{-3}$	$\mathbf{(2.05 \pm 5.04)\,10^{-6}}$	1.31 ± 0.542	2.26 ± 0.475
NLODE4	4	$(6.85 \pm 16.2)\,10^{-3}$	$\mathbf{(1.05 \pm 4.71)\,10^{-6}}$	$(4.78 \pm 10.4)\,10^{-2}$	$\mathbf{(8.10 \pm 39.9)\,10^{-3}}$	0.717 ± 0.392	1.78 ± 0.360
NLODE5	4	$(3.76 \pm 21.8)\,10^{-4}$	$\mathbf{(5.213 \pm 17.9)\,10^{-5}}$	$(2.24 \pm 7.48)\,10^{-4}$	$\mathbf{(8.51 \pm 12.1)\,10^{-5}}$	2.76 ± 2.13	3.42 ± 1.96
NLODE6	4	$(3.79 \pm 2.07)\,10^{-3}$	$\mathbf{(3.35 \pm 1.66)\,10^{-3}}$	$(5.30 \pm 3.53)\,10^{-1}$	$\mathbf{(5.32 \pm 3.57)\,10^{-1}}$	1.21 ± 0.649	2.10 ± 0.904
SODE1	4	$(1.41 \pm 7.66)\,10^{-9}$	$\mathbf{(1.16 \pm 6.74)\,10^{-9}}$	$(2.23 \pm 4.42)\,10^{-6}$	$\mathbf{(1.86 \pm 3.95)\,10^{-6}}$	3.66 ± 1.39	5.04 ± 1.41
SODE2	4	$(3.18 \pm 4.90)\,10^{-10}$	$\mathbf{(2.47 \pm 4.16)\,10^{-10}}$	$(1.61 \pm 1.23)\,10^{-6}$	$\mathbf{(1.40 \pm 1.12)\,10^{-6}}$	3.75 ± 1.41	4.98 ± 1.37
SODE3	4	$(9.21 \pm 30.0)\,10^{-9}$	$\mathbf{(9.13 \pm 30.0)\,10^{-9}}$	$(8.28 \pm 9.54)\,10^{-6}$	$\mathbf{(8.19 \pm 9.46)\,10^{-6}}$	9.54 ± 0.833	9.95 ± 0.175
SODE4	4	$(1.45 \pm 4.35)\,10^{-1}$	$\mathbf{(1.17 \pm 3.97)\,10^{-1}}$	$\mathbf{6.67 \pm 26.4}$	8.37 ± 31.7	4.49 ± 2.36	6.00 ± 2.12
SODE5	4	$(1.21 \pm 3.28)\,10^{-2}$	$\mathbf{(1.21 \pm 3.27)\,10^{-2}}$	$(2.88 \pm 7.79)\,10^{-2}$	$\mathbf{(2.88 \pm 7.78)\,10^{-2}}$	4.58 ± 2.67	5.89 ± 2.36
SODE6	4	$(2.07 \pm 1.39)\,10^{-2}$	$\mathbf{(1.34 \pm 1.14)\,10^{-2}}$	$(2.83 \pm 1.46)\,10^{-2}$	$\mathbf{(1.92 \pm 1.27)\,10^{-2}}$	1.47 ± 0.465	3.21 ± 0.734
PDE1	8	$(2.82 \pm 5.50)\,10^{-5}$	$\mathbf{(2.02 \pm 2.82)\,10^{-5}}$	$(1.98 \pm 1.60)\,10^{-4}$	$\mathbf{(1.75 \pm 1.14)\,10^{-4}}$	5.14 ± 1.63	7.10 ± 1.52
PDE2	8	$(1.02 \pm 4.36)\,10^{-5}$	$\mathbf{(8.77 \pm 36.1)\,10^{-6}}$	$(9.88 \pm 12.2)\,10^{-5}$	$\mathbf{(9.48 \pm 11.3)\,10^{-5}}$	6.85 ± 2.59	8.13 ± 1.80
PDE3	8	$(1.98 \pm 2.95)\,10^{-7}$	$\mathbf{(1.72 \pm 2.48)\,10^{-7}}$	$(1.19 \pm 0.963)\,10^{-5}$	$\mathbf{(1.09 \pm 0.846)\,10^{-5}}$	6.41 ± 1.78	8.03 ± 1.40
PDE4	8	$(2.44 \pm 4.90)\,10^{-4}$	$\mathbf{(2.03 \pm 3.53)\,10^{-4}}$	$(7.31 \pm 5.80)\,10^{-4}$	$\mathbf{(7.02 \pm 5.28)\,10^{-4}}$	5.16 ± 1.96	7.02 ± 1.72
PDE5	8	$(7.56 \pm 19.8)\,10^{-4}$	$\mathbf{(6.26 \pm 15.4)\,10^{-4}}$	$(8.07 \pm 7.97)\,10^{-4}$	$\mathbf{(7.53 \pm 6.39)\,10^{-4}}$	7.79 ± 2.29	8.94 ± 1.63
PDE6	8	$(3.12 \pm 8.20)\,10^{-4}$	$\mathbf{(1.96 \pm 5.01)\,10^{-4}}$	$(5.76 \pm 7.37)\,10^{-4}$	$\mathbf{(4.75 \pm 5.26)\,10^{-4}}$	4.71 ± 1.41	6.70 ± 1.40
PDE7	8	$(2.25 \pm 4.15)\,10^{-3}$	$\mathbf{(2.07 \pm 4.11)\,10^{-3}}$	$(1.60 \pm 1.39)\,10^{-2}$	$\mathbf{(1.55 \pm 1.37)\,10^{-2}}$	3.97 ± 1.96	5.80 ± 1.86
PDE8	8	$(6.44 \pm 12.7)\,10^{-1}$	$\mathbf{(4.36 \pm 7.63)\,10^{-1}}$	$(5.77 \pm 6.11)\,10^{-2}$	$\mathbf{(5.15 \pm 5.29)\,10^{-2}}$	3.78 ± 2.10	5.62 ± 1.88
PDE9	8	$(1.22 \pm 1.53)\,10^{-5}$	$\mathbf{(7.69 \pm 8.18)\,10^{-6}}$	$(3.77 \pm 2.64)\,10^{-4}$	$\mathbf{(3.47 \pm 2.45)\,10^{-4}}$	4.01 ± 1.07	6.01 ± 1.07

Figures 6 to 10 exemplify how the proposed method behaves when it is applied to different problems. For example, Fig. 6 provides a typical run for PDE8 problem showing the evolution of fitness function and the RMSE versus the number of fitness evaluations. It is also shown the step size and the *condition number* of the CMA-ES algorithm, which is defined as the ratio between the largest and smallest eigenvalue of the covariance matrix C. We can appreciate the auto-adaptation capabilities of the CMA-ES from the evolution of the covariance matrix eigenvalues and mutation step σ, according to the local landscape of the fitness function. Figure 7 shows the evolution of the Gaussian kernels for a typical run of NLODE4. Note how a big variation is observed in most of the parameters at the first generations, where the CMA-ES explores the solution space. Afterwards, the rate of change decreases because CMA-ES and DS perform an exploitation of the best solutions. Figure 8 shows the approximate solution as an addition of four Gaussian kernels that form the solution of NLODE4. Note that in this particular situation, the four centers c_i are outside the independent variable range and one kernel has a negative value for γ. Figure 9 shows a comparison between the exact and the approximated solutions for NLODE4. Finally, Fig. 10 shows a comparison between the approximated solution found by the proposed method and the exact solution for the *wave* equation (PDE8). The solution represents the vibration of a string in his fundamental harmonic scale. Only five time instants are plotted for t from 0 to 0.5, although the equation is solved in the range $t \in [0, 1]$. As we can observe in the close-up view at the bottom figure, the solution evolved by the method is in good agreement with the exact one.

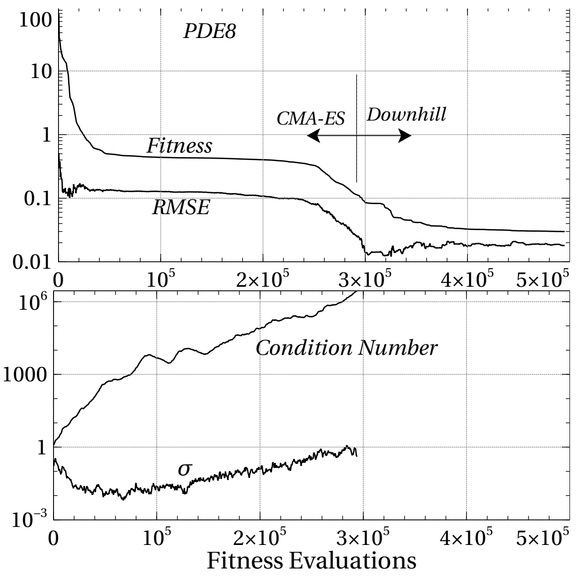

Figure 6: A typical run of PDE8 showing the evolution of the fitness value and RMSE (top view), and the step size and condition number of covariance matrix (bottom view) of the CMA-ES phase.

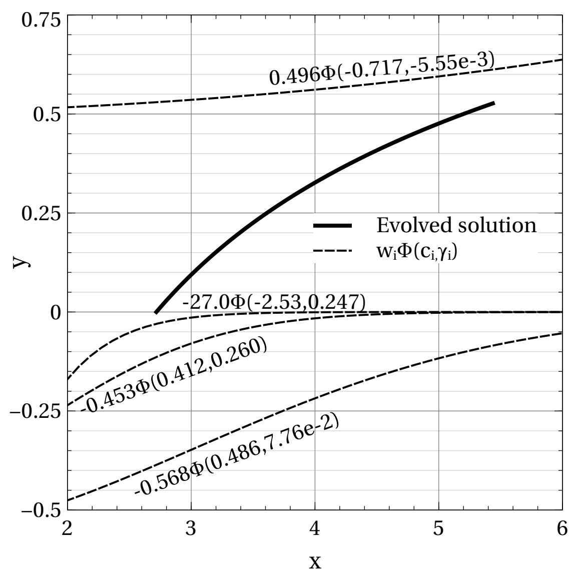

Figure 8: Final evolved solution obtained in a typical run of NLODE4. The four Gaussian kernels forming the evolved solution are also shown. Note that kernel $\Phi\,(-0.717, -5.55e-3)$ has a negative gamma value.

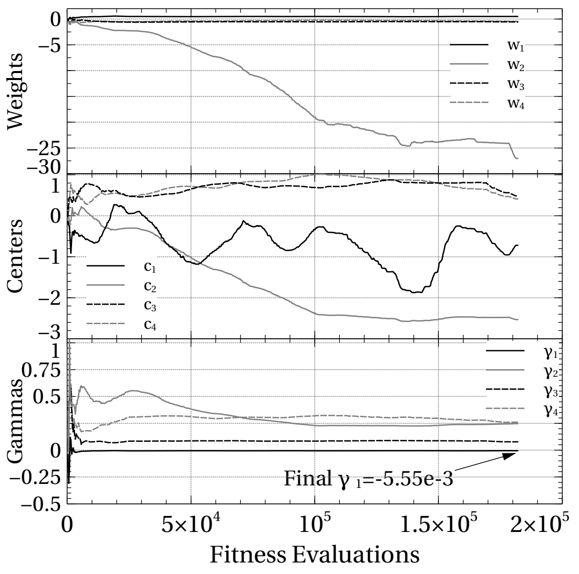

Figure 7: A typical run of NLODE4 showing the evolution of the Gaussian kernels: w_i, c_i and γ_i.

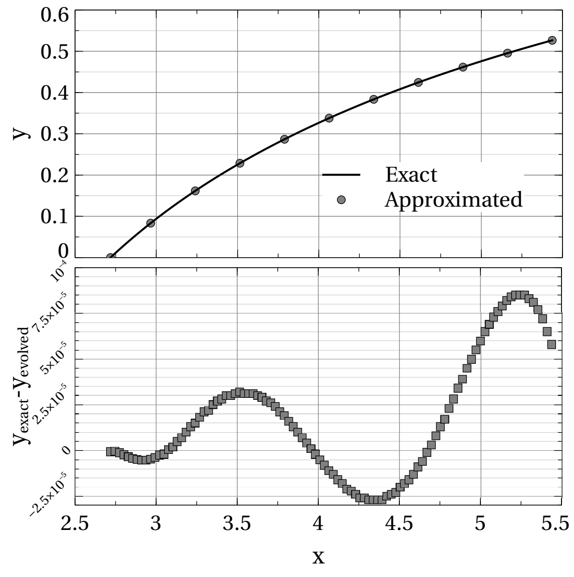

Figure 9: Comparison between the final solution obtained in a typical run of NLODE4 with the exact one (top view) and the error between the evolved solution and the exact one (bottom view). For the sake of clarity, in the top figure only 11 points are plot for the evolved solution, although 100 collocation points were taken into account. Note the order of magnitude of the error in the lower figure.

5.3 Sensitivity to the number of kernels and the inner weighting factor (κ)

The majority of the solutions obtained present fitness values with orders of magnitude below 10^{-3}. As we can see in Table 5, values below 10^{-9} are reached in some cases. The effect of increasing the number of kernels in the solution quality is studied in this section. To do this, we arbitrarily select all the problems whose orders

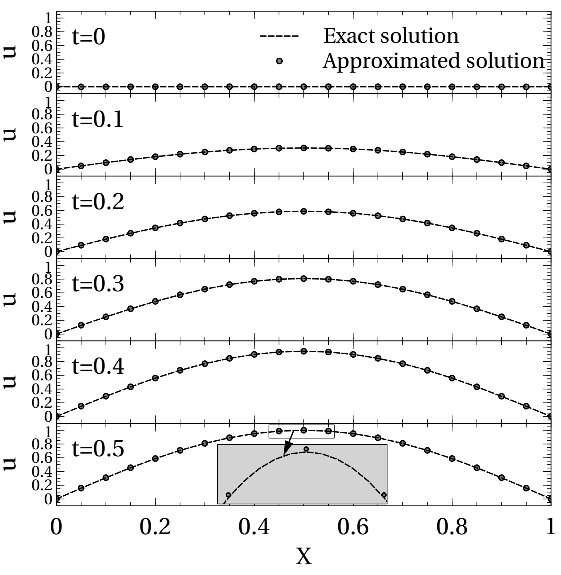

Figure 10: Comparison between a typical solution obtained by the proposed method and the exact one for PDE8 (*wave* equation).

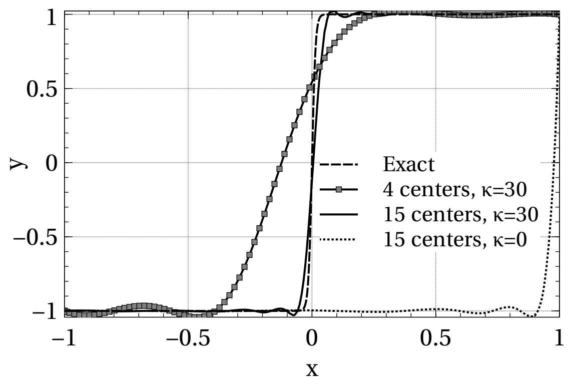

Figure 11: Comparison between evolved solutions and the exact one for NLODE6 equation.

of magnitude associated to the solution fitness value are higher than 10^{-3}. We will vary the number of kernels, but the rest of the control parameters described in Tables 3 and 4 are maintained.

As we see in Table 6, better fitness values are obtained when the number of kernels is increased for all the cases, except for SODE4 problem, where the fitness values are very similar. However, for this particular problem, an increment of the number of kernels from four (Table 5) to six (Table 6) is enough to decrease the fitness values in eight orders of magnitude. On the other hand, the value of RMSE also improves or is approximately maintained when the number of kernels grows. Here, it is important to remain the reader that the RMSE value is used neither in the evolutionary algorithm nor in the local search method. Therefore, from the evidence here presented, we can say that, when the evolved solution is not as accurate as desired, we should increase the number of kernels to improve such accuracy. Finally, in relation to the average number of fitness evaluation, obviously, it increases as the size of the problem does with the number of kernels.

As it was commented in Section 5.2, the inner weighting factor κ was set to 30 for all the cases. It was not observed a high sensitivity in the experiments regarding to κ, except for the NLODE6 case. As we see in Fig. 11, this equation is hard to solve because it has a strong gradient in the origin. For example, when $\kappa = 0$, the algorithm is not capable to locate the appropriate transition from -1 to 1 at the origin. When an appropriate value of κ is set ($\kappa = 30$), we can observe how the quality of the solution only depends on the number of centers, increasing the former when the latter does.

6 Discussion

In this section, some qualitative and quantitative comparisons against other evolutionary approaches reported in the literature are presented (Section 6.1). Then, two PDEs are solved using two different numerical methods and the results are compared with the proposed algorithm (Section 6.2).

6.1 Comparison with other evolutionary approaches

It is difficult to make a quantitative comparative study with other reported approaches because the fitness values are highly dependent on the solver parameters and on how differential equation is provided to Eq. (11) (see Section 5.2). For a correct comparison of the solution quality, RMSE values should be used. However, the RMSE is not generally reported in previous contributions. As it was described in Section 5.3, the direct way to increase the solution accuracy in our method depends strongly on the number of kernels used. In order to measure the performance of our approximation in relation to other methods, we maintained here the set-up used in Section 5, except the number of kernels. That is, when the RMSE value obtained in Table 5 was lower than the best result obtained by one of the techniques used in the comparison, we increased the number of kernels to investigate the potential of our method. In this regard, we also reused the results obtained in Table 6.

Table 6: Effect of increasing the number of centers (or kernels). The best values obtained are highlighted in bold letters in "Fitness" and "RMSE" columns.

Case	**Centers n**	**Unknowns N** Eq. (9)	**Population μ** Eq. (22)	**Fitness (CMA-ES+DS)**	**RMSE (CMA-ES+DS)**	**Fit. Evaluations (CMA-ES+DS)**
LODE9	6	18	18	$(4.14 \pm 5.46)\,10^{-2}$	$(1.04 \pm 1.08)\,10^{-1}$	$(3.36 \pm 0.843)\,10^{5}$
	10	30	21	$(2.48 \pm 16.3)\,10^{-3}$	$(1.08 \pm 3.48)\,10^{-2}$	$(5.90 \pm 1.05)\,10^{5}$
	14	42	22	$\mathbf{(7.44 \pm 43.7)\,10^{-6}}$	$\mathbf{(4.49 \pm 20.4)\,10^{-4}}$	$(9.44 \pm 0.80)\,10^{5}$
LODE10	6	18	18	$(1.18 \pm 3.66)\,10^{2}$	$(2.26 \pm 6.14)\,10^{-1}$	$(3.03 \pm 1.50)\,10^{5}$
	10	30	21	$(4.72 \pm 23.1)\,10^{-3}$	$(5.18 \pm 9.21)\,10^{-4}$	$(6.15 \pm 1.40)\,10^{5}$
	14	42	22	$\mathbf{(1.32 \pm 5.04)\,10^{-3}}$	$\mathbf{(4.54 \pm 6.74)\,10^{-4}}$	$(9.27 \pm 1.06)\,10^{5}$
NLODE6	6	18	18	$(1.87 \pm 1.43)\,10^{-3}$	$(4.12 \pm 3.54)\,10^{-1}$	$(3.88 \pm 1.36)\,10^{5}$
	10	30	21	$(1.30 \pm 1.36)\,10^{-3}$	$(3.05 \pm 2.99)\,10^{-1}$	$(6.19 \pm 1.14)\,10^{5}$
	14	42	22	$\mathbf{(6.26 \pm 8.50)\,10^{-4}}$	$\mathbf{(2.02 \pm 2.74)\,10^{-1}}$	$(8.89 \pm 1.15)\,10^{5}$
SODE4	6	36	21	$\mathbf{(2.18 \pm 5.89)\,10^{-9}}$	$\mathbf{(2.76 \pm 3.23)\,10^{-6}}$	$(8.15 \pm 2.01)\,10^{5}$
	10	60	24	$(3.58 \pm 5.56)\,10^{-9}$	$(4.47 \pm 3.13)\,10^{-6}$	$(9.55 \pm 1.15)\,10^{5}$
	14	84	25	$(6.57 \pm 8.52)\,10^{-9}$	$(6.38 \pm 4.45)\,10^{-6}$	$(9.49 \pm 1.09)\,10^{5}$
SODE5	6	36	21	$(2.63 \pm 12.9)\,10^{-9}$	$(3.53 \pm 3.98)\,10^{-6}$	$(7.86 \pm 1.80)\,10^{5}$
	10	60	24	$(6.26 \pm 11.7)\,10^{-10}$	$\mathbf{(2.28 \pm 1.95)\,10^{-6}}$	$(8.83 \pm 1.40)\,10^{5}$
	14	84	25	$\mathbf{(6.22 \pm 5.37)\,10^{-10}}$	$(2.38 \pm 1.24)\,10^{-6}$	$(8.76 \pm 1.51)\,10^{5}$
SODE6	6	36	21	$(1.43 \pm 0.208)\,10^{-2}$	$(2.11 \pm 0.288)\,10^{-2}$	$(4.29 \pm 0.77)\,10^{5}$
	10	60	24	$(1.02 \pm 0.303)\,10^{-2}$	$(1.64 \pm 0.374)\,10^{-2}$	$(7.30 \pm 1.91)\,10^{5}$
	14	84	25	$\mathbf{(8.70 \pm 4.88)\,10^{-3}}$	$\mathbf{(1.38 \pm 0.556)\,10^{-2}}$	$(8.92 \pm 1.34)\,10^{5}$
PDE7	6	24	19	$(7.95 \pm 11.7)\,10^{-2}$	$(2.98 \pm 2.52)\,10^{-2}$	$(3.87 \pm 2.15)\,10^{5}$
	10	40	22	$(7.80 \pm 10.2)\,10^{-4}$	$(9.88 \pm 10.0)\,10^{-3}$	$(7.73 \pm 1.95)\,10^{5}$
	14	56	24	$\mathbf{(2.78 \pm 4.02)\,10^{-4}}$	$\mathbf{(6.19 \pm 6.41)\,10^{-3}}$	$(9.56 \pm 1.04)\,10^{5}$
PDE8	6	24	19	1.20 ± 1.90	$(1.00 \pm 0.729)\,10^{-1}$	$(3.38 \pm 1.64)\,10^{5}$
	10	40	22	$(1.97 \pm 6.39)\,10^{-2}$	$(7.00 \pm 14.3)\,10^{-3}$	$(8.75 \pm 1.60)\,10^{5}$
	14	56	24	$\mathbf{(8.23 \pm 14.8)\,10^{-4}}$	$\mathbf{(1.50 \pm 1.90)\,10^{-3}}$	$(9.95 \pm 0.215)\,10^{5}$

We begin this discussion focusing on those works where RMSE values (or at least, some other measure of the solution accuracy) are managed (Sobester et al., 2008; Chaquet and Carmona, 2012; Babaei, 2013; Panagant and Bureerat, 2014). The results of the comparison are shown in Table 7. For example, in a previous work by the same authors (Chaquet and Carmona, 2012), the candidate solutions were expressed with partial sums of Fourier series. Additionally, an *ad hoc* method, which is based on the successive application of several stages of standard Evolution Strategies (ES) to tune the harmonics, was used. The present approach outperforms the results of (Chaquet and Carmona, 2012) in terms of lower values of RMSE (see Table 7). However, in order to confirm this, a *Wilcoxon's sum-rank* test (one-tailed), was applied. In this study, the null hypothesis (the medians are equal) was rejected in 100% of the analyzed problems ($\alpha = 0.05$). Therefore, we have statistical evidence to say that the median of RMSE values obtained with our method is always lower than those obtained in (Chaquet and Carmona, 2012).

As an example, Fig. 12 shows a typical run of the proposed algorithm compared with the method described in (Chaquet and Carmona, 2012) for NLODE1 problem. Regarding the set-up for this comparison, the same parameters already commented in Tables 3 and 4 have been employed for the algorithm, i. e., four kernels are adjusted (12 unknowns) with a population size $(\mu, \lambda) = (16, 33)$. In (Chaquet and Carmona, 2012), the method uses 10 harmonics and a population of $(\mu, \lambda) = (10, 400)$. Note how the CMA-ES algorithm needs a lower λ value, which turns in a lower number of fitness evaluations. In any case, due to the stochastic nature of the algorithms, the average values are compared after running both algorithms 50 times. Our current method obtains a RMSE of $(1.61{\pm}2.12){\cdot}10^{-5}$, needing $(1.88{\pm}0.453){\cdot}10^{5}$ fitness evaluations. In (Chaquet and Carmona, 2012), the RMSE obtained was $(7.42 \pm 0.968) \cdot 10^{-5}$, using $(4.87 \pm 1.67) \cdot 10^{5}$ fitness evaluations. In the present approach, all the problems addressed in (Chaquet and Carmona, 2012) have been successfully solved, meanwhile, in the harmonic evolutionary solver, LODE3 was not correctly handled. As it is described in (Chaquet and Carmona, 2012), the number of unknowns exponentially increases with the space dimension. This drawback does not exist in our approximation. For example, for a PDE with two dependent variables, the election of 10 centers in our methods implies to tune 40 unknowns. On the other hand, the use of 10 harmonics in (Chaquet and Carmona, 2012) implies the tuning of 100 unknowns, that is, more than twice as many. Thanks to the good performance of CMA-ES, all the unknowns can be adjusted simultaneously. However, in (Chaquet and Carmona, 2012), different ES steps adjust differ-

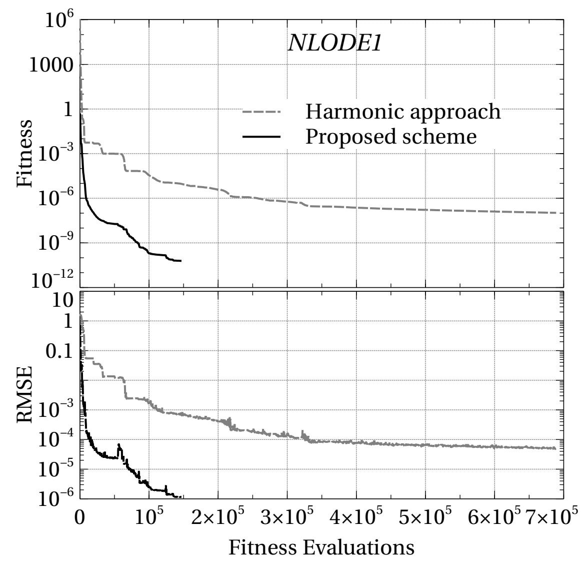

Figure 12: Comparison between the proposed method and a Fourier-based approach described in (Chaquet and Carmona, 2012) for a typical run of NLODE1 problem.

ent harmonic coefficients. Therefore, our approximation greatly simplifies the procedure used to tune the variables (unknowns).

Another approach using Fourier series can be seen in (Babaei, 2013) where harmonics are tuned by PSO techniques. There, no PDE was addressed. In that work, some tuned Fourier coefficients are given, so it is possible to calculate and compare the RMSE values obtained. Note that the raw data and their standard deviations are not available in (Babaei, 2013). Therefore, assuming normality hypothesis in the data distribution, a *Student's t*-test (one-tailed) was made in order to compare the means of the RMSE values obtained for LODE11, SODE5 and SODE6. The test results revealed that there is sufficient evidence, at the $\alpha = 0.05$ level, to conclude that, in two of the three mentioned problems, our results are more accurate than those obtained in (Babaei, 2013).

In (Panagant and Bureerat, 2014), the RMSE values are also reported. There, only PDEs are solved. Solutions are approximated by polynomial functions whose coefficients are tuned using a method based on Differential Evolution. However, the raw data, the number of executions for each problem, the dispersion of the results, and the number of unknowns used are not provided. Thus, assuming normality hypothesis in the data distribution, a *Student's t-test* (one-tailed) was made in order to compare the means of RMSE values for PDE1 to PDE6 (see Table 7). The test results revealed that the null hypothesis (the means are equal) was rejected for 100% of the mentioned problems. Therefore, there is sufficient evidence ($\alpha = 0.05$) to conclude that our results are more accurate than those obtained in (Panagant and Bureerat, 2014).

In (Sobester et al., 2008), a Genetic Programming approach for solving differential equations is presented. There, the solutions are split into two terms in order to fulfill the boundary conditions by construction. This particular approach can be adopted only in some specific geometries. In that paper, the MSE (mean square error) is used as a measure of accuracy. However, it is straightforward to obtain the RMSE values ($RMSE^2 = MSE$). As we can see in Table 7, the accuracy of our approach is better than the results presented in (Sobester et al., 2008). This conclusion is confirmed assuming normality hypothesis in the data distribution (the raw data are not available) and applying a *Student's t*-test (one-tailed). Thus, we can affirm that there is enough evidence ($\alpha = 0.05$) to reject the null hypothesis and, therefore, to state that the mean of RMSE values in our case is lower than those obtained in (Sobester et al., 2008) for PDE1, PDE5 and PDE6.

The dispersion in the results is an important issue when comparing stochastic algorithms. A good algorithm should give similar results in all the executions. However, very few works in the literature give dispersion rates. For example, in (Tsoulos and Lagaris, 2006), a differential equation solver using Genetic Programming is presented. Although RMSE values are not provided, the authors give minimum, maximum and average of the number of generations needed to find a correct solution as measures of dispersion. Knowing the population size used, it is possible to perform a comparison with our approach in terms of the number of fitness evaluations: the average order of magnitude calculated was 10^5, i.e., the same as that obtained by our method. Additionally, all the problems solved there were also successfully solved here.

In (Shirvany et al., 2009), a PDE using complex numbers on a triangle shape domain is reported. The RMSE is computed approximating the exact solution by a numerical method solution, obtaining a value around 10^{-4}. This number is of the same order as the RMSE average obtained for all the PDEs when our method is used (a variation from 10^{-2} to 10^{-5} is observed). However, the number of unknowns that must be tuned in (Shirvany et al., 2009) is much higher than that used in our method. A total of 132 neurons are needed there, against 32 unknowns used here. In a similar way, in (Yazdi and Pourreza, 2010), more than 70 neurons are

Table 7: Comparison of the obtained RMSE, considering only those works where that type of error is reported (Sobester et al., 2008; Chaquet and Carmona, 2012; Babaei, 2013; Panagant and Bureerat, 2014). Standard deviations are not always provided in the referenced works. The best results are marked in bold letter. The final number of centers n, used by our approach for this comparison, is also provided in the last column.

Problem	**RMSE in Sobester08**	**RMSE in Chaquet12**	**RMSE in Babai13**	**RMSE in Panagant14**	**RMSE in this work| centers n used**
LODE1	-	$(3.70 \pm 0.166) \cdot 10^{-5}$	-	-	$\mathbf{(2.35 \pm 2.52) \cdot 10^{-6}} \vert 4$
LODE2	-	$(6.59 \pm 0.255) \cdot 10^{-5}$	-	-	$\mathbf{(2.14 \pm 14.4) \cdot 10^{-5}} \vert 4$
LODE3	-	13.16 ± 0.013	-	-	$\mathbf{(2.13 \pm 7.53) \cdot 10^{-3}} \vert 4$
LODE4	-	$(1.65 \pm 0.87) \cdot 10^{-6}$	-	-	$\mathbf{(4.17 \pm 9.16) \cdot 10^{-7}} \vert 6$
LODE5	-	$(9.90 \pm 0.38) \cdot 10^{-5}$	-	-	$\mathbf{(3.01 \pm 8.13) \cdot 10^{-6}} \vert 4$
LODE6	-	$(5.44 \pm 3.64) \cdot 10^{-6}$	-	-	$\mathbf{(1.72 \pm 2.85) \cdot 10^{-7}} \vert 4$
LODE7	-	$(1.25 \pm 0.26) \cdot 10^{-5}$	-	-	$\mathbf{(6.71 \pm 2.33) \cdot 10^{-6}} \vert 4$
LODE8	-	$(1.22 \pm 0.001) \cdot 10^{-2}$	-	-	$\mathbf{(4.87 \pm 5.41) \cdot 10^{-5}} \vert 4$
LODE9	-	$(1.51 \pm 0.001) \cdot 10^{-2}$	-	-	$\mathbf{(4.49 \pm 20.4) \cdot 10^{-4}} \vert 14$
LODE10	-	$(3.15 \pm 0.0519) \cdot 10^{-2}$	-	-	$\mathbf{(4.54 \pm 6.74) \cdot 10^{-4}} \vert 14$
LODE11	-	-	$4.65 \cdot 10^{-3}$	-	$\mathbf{(9.16 \pm 39.4) \cdot 10^{-4}} \vert 4$
NLODE1	-	$(7.42 \pm 0.968) \cdot 10^{-5}$	-	-	$\mathbf{(1.61 \pm 2.12) \cdot 10^{-5}} \vert 4$
NLODE2	-	$(5.90 \pm 0.549) \cdot 10^{-6}$	-	-	$\mathbf{(3.79 \pm 3.31) \cdot 10^{-7}} \vert 6$
NLODE3	-	$(3.64 \pm 0.49) \cdot 10^{-5}$	-	-	$\mathbf{(2.05 \pm 5.04) \cdot 10^{-6}} \vert 4$
NLODE4	-	$(8.32 \pm 0.79) \cdot 10^{-5}$	-	-	$\mathbf{(8.21 \pm 33.0) \cdot 10^{-7}} \vert 6$
NLODE5	-	$(3.19 \pm 0.59) \cdot 10^{-6}$	-	-	$\mathbf{(1.59 \pm 1.74) \cdot 10^{-6}} \vert 8$
NLODE6		$(3.03 \pm 0.0009) \cdot 10^{-1}$			$\mathbf{(2.02 \pm 2.74) \cdot 10^{-1}} \vert 14$
SODE1	-	$(7.465 \pm 1.66) \cdot 10^{-5}$	-	-	$\mathbf{(1.86 \pm 3.95) \cdot 10^{-6}} \vert 4$
SODE2	-	$(3.90 \pm 0.37) \cdot 10^{-5}$	-	-	$\mathbf{(1.40 \pm 1.12) \cdot 10^{-6}} \vert 4$
SODE3	-	$(8.51 \pm 4.02) \cdot 10^{-5}$	-	-	$\mathbf{(8.19 \pm 9.46) \cdot 10^{-6}} \vert 4$
SODE4	-	$(4.72 \pm 0.261) \cdot 10^{-5}$	-	-	$\mathbf{(2.76 \pm 3.23) \cdot 10^{-6}} \vert 6$
SODE5	-	$(1.20 \pm 0.007) \cdot 10^{-4}$	$3.88 \cdot 10^{-3}$	-	$\mathbf{(2.28 \pm 1.95) \cdot 10^{-6}} \vert 10$
SODE6	-	-	$1.78 \cdot 10^{-2}$	-	$\mathbf{(1.38 \pm 0.56) \cdot 10^{-2}} \vert 14$
PDE1	$(6.9 \pm 8.3) \cdot 10^{-4}$	$(6.37 \pm 0.73) \cdot 10^{-3}$	-	$7.25 \cdot 10^{-4}$	$\mathbf{(6.20 \pm 3.36) \cdot 10^{-5}} \vert 14$
PDE2	-	$(1.16 \pm 0.21) \cdot 10^{-3}$	-	$2.45 \cdot 10^{-4}$	$\mathbf{(9.48 \pm 11) \cdot 10^{-5}} \vert 8$
PDE3	-	$(5.90 \pm 0.79) \cdot 10^{-3}$	-	$9.48 \cdot 10^{-6}$	$\mathbf{(5.02 \pm 2.19) \cdot 10^{-6}} \vert 10$
PDE4	-	$(1.23 \pm 0.03) \cdot 10^{-3}$	-	$6.60 \cdot 10^{-3}$	$\mathbf{(7.02 \pm 5.2) \cdot 10^{-4}} \vert 8$
PDE5	$(1.4 \pm 2.7) \cdot 10^{-2}$	$(9.06 \pm 1.29) \cdot 10^{-4}$	-	$3.72 \cdot 10^{-2}$	$\mathbf{(1.17 \pm 0.73) \cdot 10^{-4}} \vert 14$
PDE6	$(2.0 \pm 2.1) \cdot 10^{-2}$	$(1.79 \pm 0.03) \cdot 10^{-2}$	-	$3.82 \cdot 10^{-1}$	$\mathbf{(1.80 \pm 1.15) \cdot 10^{-4}} \vert 10$

needed for obtaining a solution in LODE9. In our case, good accuracy (RMSE order of magnitude is equal to 10^{-4}) has been obtained with 42 unknowns (see Table 6).

From a qualitative point of view, some advantages of the current approach can be enumerated. Firstly, it is straightforward to compute the derivatives because all the solutions are only expressed as a sum of Gaussian functions. Derivatives of this type of kernels can be precalculated. Secondly, low dispersion in the results has been observed, so this finding provides evidence about the robustness of our method. Works based on GP reported a higher dispersion. And finally, our approach can be applied to different types of differential equations. Some authors (Lagaris et al., 1998; Sobester et al., 2008; Yazdi and Pourreza, 2010) use some particular methods for dealing with the boundary conditions, facilitating the optimization process by means of eliminating constraints. Nevertheless, these methods are problem dependent and cannot be applied to other types of DEs. Our method does not assume any particular structure in the boundary conditions, making the method suitable for different types of problems (LODE, NLODE, SODE, and PDE).

6.2 Comparison with classical methods

In this subsection, a comparison between the proposed algorithm and numerical methods is presented. Our intention here is not giving an exhaustive comparison with this kind of methods. Numerical methods are much more mature than evolutionary schemes, can cope with a great variety of difficult problems and are faster than evolutionary approaches. However, numerical methods are usually specific for each type of equation. Alternatively, an evolutionary algorithm could cope with different types of differential equations if they can be transformed into an optimization problem. An example of this is analyzed here. For that, two partial differential equations are chosen: PDE1 (*Poisson* equation) and PDE8 (*wave* equation). Although both are second order equations, the former is *elliptic* and the latter *hyperbolic*. This means that a different numerical method should be employed to solve each one of them. On the other hand, as we will show in this section, the proposed evolutionary method can cope with the two problems without any change in the algorithm or in its parameter configuration.

A brief description of two simple finite difference numerical methods used in the comparison is provided. The first method, which is used to solve PDE1, is an explicit four order Runge-Kutta (RK4) (Press et al., 2002). Let $\Psi_{xx} + \Psi_{yy} = f(x, y)$ be the DE. After a discretization process on the grid nodes and using central differences, the residual at point (i, j) of a candidate solution field Ψ is computed as:

$$\begin{aligned} R_{i,j}(\Psi) = \frac{\Psi_{i-1,j} - 2\Psi_{ij} + \Psi_{i+1,j}}{\triangle x^2} + \\ + \frac{\Psi_{i,j-1} - 2\Psi_{ij} + \Psi_{i,j+1}}{\triangle y^2} - f(x_i, y_j), \end{aligned} \tag{23}$$

where constant grid sizes $\triangle x$ and $\triangle y$ are assumed. Note that the collocation points must be arranged in such a way that a structured grid (or mesh) is formed assigning two indices (i, j) to each node. In this way, for a given point (or node of the mesh), its neighbors can be identified to compute the derivatives. This special arrangement is not needed in our mesh-free approach.

At the boundary condition points, the residual is set to 0. Defining a pseudo time parameter, τ, which controls the convergence rate and the stability of the method, the RK4 consists in the following sequence:

$$\left.\begin{array}{c} \Psi^* = \Psi \\ \Psi^* = \Psi + \tau R(\Psi^*)/4 \\ \Psi^* = \Psi + \tau R(\Psi^*)/3 \\ \Psi^* = \Psi + \tau R(\Psi^*)/2 \\ \Psi = \Psi^* \end{array}\right\} \tag{24}$$

The above sequence is repeated until some convergence criterion is fulfilled, typically when a maximum number of iterations is reached or when the norm of the residual array, R, is below a predefined tolerance. We have used 2000 iterations and $\tau = 10^{-3}$ for PDE1. We can see a schematic of the numerical method in Fig. 13a.

On the other hand, some modifications must be done to the numerical scheme to solve the *wave* equation (PDE8). In particular, the coefficient for the numerical approximation of the second derivative u_{tt} in Eq. (23) changes the sign. Besides, the boundary conditions must be applied in a different way because they are not only set in u, but also in its first derivative u_t. However, RK4 does not work for *wave* equation because the algorithm is not stable, even for very low values of τ.

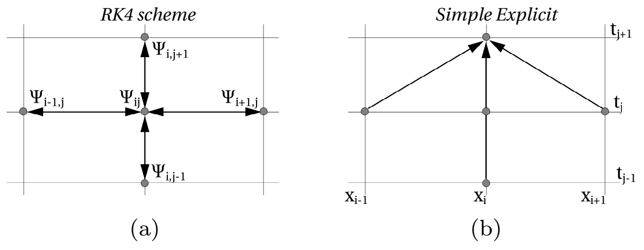

Figure 13: Numerical schemes for comparison. Arrows indicate that one node is affecting another. Note that, with the *Runge-Kutta* scheme, point (i, j) affects all its neighbors. On the contrary, with the *Explicit* algorithm, to compute the solution at time instant t_{j+1}, we need the values at previous time instants t_j and t_{j-1}, but the solution at future time instants $(t_{j+1}, t_{j+2}, \ldots)$ is not needed.

To solve the PDE8, a different approach must be followed. A simple explicit scheme can be used. Being the equation $u_{tt} = u_{xx}$, and also using central differences, the field can be computed in a straightforward way in just a single iteration:

$$u_{i,j+1} = -u_{i,j-1} + 2\left(1 - \alpha^2\right) u_{ij} + \alpha^2 \left(u_{i+1,j} + u_{i-1,j}\right), \tag{25}$$

where $\alpha = \triangle t / \triangle x$. Note that the first and second time instant ($u_{i,0}$ and $u_{i,1}$) must be obtained from the boundary conditions. The numerical scheme is outlined in Fig. 13b. Alternatively, some changes must be applied to Eq. (25) and the boundary condition to solve the *Poisson* equation (PDE1) by means of the simple explicit method.

Table 8 shows the results when PDE1 and PDE8 are solved with the evolutionary algorithm and with both numerical methods. As we can observe, each numerical method works well in one of the two problems. On the other hand, the evolutionary algorithm works properly in both cases. When the numerical schemes converge, RMSE values are similar (same order of magnitude) to those obtained by our method. However, the time required by the numerical methods is several orders of magnitude lower.

Other advantage of evolutionary algorithm is its lower memory requirements: using eight centers, a total of 32 unknowns must be stored. Numerical methods must keep as many values as collocation points (100 in these examples). Therefore, our solution requires 3 times less memory; and even more importantly, the solution is symbolically stored, so new solution values, different from the collocation points, can be obtained without performing any interpolation.

Table 8: Comparison between numerical methods and our approach for PDE1 and PDE8.

Case	**Method**	**RMSE**	**Iterations/ Fitness Eval.**	**Absolute Elapsed time**	**Relative Elapsed time**
PDE1	Evolutionary	$8.3 \cdot 10^{-6}$	$1.4 \cdot 10^{5}$	$68s$	$8.1 \cdot 10^{-2}$
	RK4	$6.5 \cdot 10^{-5}$	$8.0 \cdot 10^{3}$	$0.2s$	$2.4 \cdot 10^{-4}$
	Simple Explicit	1.1	1	$0.003s$	$3.6 \cdot 10^{-6}$
PDE8	Evolutionary	$7.8 \cdot 10^{-3}$	$4.2 \cdot 10^{5}$	14 *minutes*	1
	RK4	∞	∞	∞	∞
	Simple Explicit	$4.7 \cdot 10^{-3}$	1	$0.004s$	$4.7 \cdot 10^{-6}$

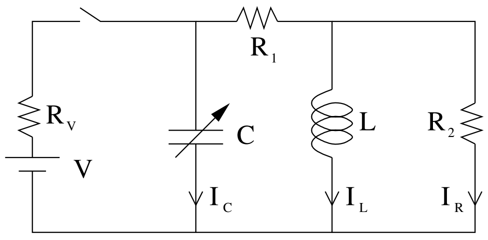

Figure 14: RLC circuit.

This comparison should be carefully taken into consideration. For example, a more complex numerical method, such as an implicit method, could solve both equations. Here we only want to give some experimental feedback about how an evolutionary approach could be more flexible and with a more straightforward setup than a numerical method.

7 Other Applications: electric circuit analysis

A more challenging and practical problem will be handled in this section. In particular, the dynamic behavior of a RLC circuit is modeled by a parametric differential equation whose solution will allow us to solve different types of analysis and design problems. First of all, a linear circuit is presented. The obtained results for each problem are validated with the exact solutions. Afterwards, a non-linear variation of the same problem is also solved with the proposed method.

7.1 Linear RLC electric circuit

The linear RLC circuit described in Fig. 14 consists of a total of six linear components: a constant voltage generator V, three resistors (R_V, R_1 and R_2), a variable capacitor C and an inductor L. We are interested in the transient behavior of the circuit when the switch is closed. Calling I_C, I_L and I_R the electric current trough the capacitor, the inductor and the resistor R_2, respectively, and applying Kirchhoff's current law, the following differential equation system is obtained:

$$\left.\begin{array}{c} V = (I_L + I_R + I_C)\,R_V + (I_L + I_R)\,R_1 + I_R R_2 \\ L\dfrac{dI_L}{dt} = R_2 I_R \\ I_C = CR_1\left(\dfrac{dI_L}{dt} + \dfrac{dI_R}{dt}\right) + CR_2\dfrac{dI_R}{dt} \end{array}\right\}, \tag{26}$$

The switch is connected at time $t = 0$. The boundary conditions are $I_L(0) = I_R(0) = I_C(0) = 0$. Note that it is a system of three equations for three unknowns (the electric currents I_L, I_R and I_C). There are a total of six parameters which determine the system: V, R_V, R_1, R_2, C and L. After some algebra, it is possible to simplify the problem to the following second order differential equation:

$$\alpha\frac{d^2I}{d\tau^2} + \frac{dI}{d\tau} + I = 1, \tag{27}$$

with boundary conditions $I(0) = (dI/d\tau)_{\tau=0} = 0$. The new variable I represents the ratio between the current in the inductor I_L at a given dimensionless time τ and the current at the steady state $I_L(\infty)$, that is:

$$I = \frac{I_L(\tau)}{I_L(\infty)} = \frac{R_V + R_1}{V} I_L(\tau)\,. \tag{28}$$

A change of variables was also performed: $\tau = t/T$, where the characteristic time T of the circuit is defined as

$$T = \frac{CR_1R_2R_V + L\,(R_1 + R_2 + R_V)}{R_2\,(R_1 + R_V)}. \tag{29}$$

The coefficient α at the Eq. (27) is the unique parameter which determines the dynamic behavior of the system. Therefore, we have transformed the original problem defined with six parameters into a new equation with only one coefficient. The relation between the original parameters and α is the following:

$$\alpha = \frac{CL}{T^2}\frac{R_V}{R_2}\frac{(R_1 + R_2)}{(R_1 + R_V)}, \tag{30}$$

The exact solution of Eq. (27) is known, so a validation of the solution obtained by our evolutionary approach can be performed. More specifically, the exact solution depends on the sign of $1 - 4\alpha$, which determines the roots of the characteristic polynomial of the differential equation. Thus, if $1 - 4\alpha > 0$, there are two different real roots $r_1 = \left(-1 + \sqrt{1-4\alpha}\right)/2\alpha$ and $r_2 = \left(-1 - \sqrt{1-4\alpha}\right)/2\alpha$, and the exact solution is $I = 1 + K_1 e^{r_1 t} + K_2 e^{r_2 t}$ with $K_1 = -1 - K_2$ and $K_2 = r_1/\left(r_2 - r_1\right)$. When $1 - 4\alpha < 0$, the characteristic polynomial has two complex roots $\mu \pm \lambda i$, being $\mu = -1/2\alpha$ and $\lambda = \sqrt{4\alpha - 1}/2\alpha$, and the exact solution is expressed as $I = 1 + e^{\mu t}\left(-\cos(\lambda t) + \frac{\mu}{\lambda}\sin(\lambda t)\right)$. Finally, if $\alpha = 1/4$ the polynomial has one repeated real root $r = -2$, and the solution is $I = 1 + e^{rt}\left(-1 + rt\right)$.

Using Eq. (29) and Eq. (30), it is possible to obtain an expression for α dependent only on the ratio L/C and the resistor values, i. e., $\alpha \equiv \alpha(\frac{L}{C}, R_1, R_2, R_V)$. Assuming that all resistances were known, it would be possible to obtain the variation range for the differential equation coefficient:

$$\alpha \in \left(0, \frac{\left(R_1 + R_2\right)\left(R_1 + R_V\right)}{4R_1\left(R_1 + R_2 + R_V\right)}\right], \tag{31}$$

where the minimum value of $\alpha \to 0$ is obtained when $L/C \to 0$ or $L/C \to \infty$. The maximum shown in Eq. (31) is obtained for $L/C = R_1 R_2 R_V/(R_1 + R_2 + R_V)$.

Once the problem and the exact solution have been described, an approximated solution will be obtained by our method. In this context, we are going to show the potential of our approach in tasks such as analysis, design and optimization in electric circuits. For that aim, the system will not be solved for a specific configuration, but it will be characterized in a *range* of the design space. In particular, the system will be simultaneously solved for a value range of α. Due to the problem is two dimensional (two independent variables, α and τ, are handled), the solution to the new problem will be obtained from the following particularization of Eq. (8):

$$I\left(\tau, \alpha\right) = \sum_{i=1}^{n} w_i \exp\left\{-\gamma_i\left[\left(\tau - c_{i\tau}\right)^2 + \left(\alpha - c_{i\alpha}\right)^2\right]\right\} \tag{32}$$

Considering that, in the circuit of Fig. 14, the following component values are known: $V = 12$ Volts, $R_V = 50\Omega$, $R_1 = R_2 = 10^5\Omega$ and $L = 10^{-3}H$, and the variable capacitor is in the range $C \in \left[1.60 \cdot 10^{-9}, 3.92 \cdot 10^{-8}\right] F$, then the following problems can be defined:

- Problem 1: What is the current through the inductor L at time $t = 1\mu s$ for all the possible values of the capacitor?

Table 9: Range of α and τ for Problems 1 and 2.

C	α	τ **at** $t = 1\mu s$
$1.60 \cdot 10^{-9}$	0.16	10
$3.92 \cdot 10^{-8}$	0.01	0.5

- Problem 2: What is the maximum value of C which ensures that the current through the inductor L at time $t = 1\mu s$ is at least the 95% of the current at the steady state?
- Problem 3: What is the minimum value of C which ensures that the maximum voltage in the inductor is lower than 0.5 Volts?

In order to answer these questions, the Eq. (27) is solved with our evolutionary algorithm. First, a range for the collocation points must be selected. Knowing the values of the linear components of the circuit and using Eq. (29) and (30), we can obtain the ranges for α and τ. In particular, substituting the known values of R_1, R_2 and R_V in Eq. (31), it is possible to obtain that the maximum value of α is $\alpha_{max} \simeq 0.25$. In the same way, using the extreme values of the capacitor and the known value of the inductor, the maximum values for α and τ can be obtained for the two first problems (see Table 9). Nevertheless, knowing that $\alpha \in [0.01, 0.16]$, and in order to show the potential of the method to deal with different types of solutions, the range of α is extended until its maximum, that is, we will work with $\alpha \in [0.01, 0.25]$. It is not possible to know which range for τ is needed to solve the Problem 3, so a maximum value of $\tau = 10$, required for the Problems 1 and 2, is assumed. Fig. 15 shows the design region considered. As it is shown in this figure, 13 different values of α in the ranges $\alpha \in [0.01, 0.25]$ are defined. The non-dimensional temporal variable τ is set in $[0, 10]$ using 101 points (constant intervals of 0.1). Therefore, a total of 1313 collocation points are employed. The same parameters shown in Tables 3 and 4 are used by the algorithm, except the maximum number of fitness evaluation, which has been increased to 10^7. Table 10 shows the fitness and RMSE values for different number of kernels. Note how the accuracy improves when the number of kernels is increased. As in the benchmark problems shown in the previous sections, low dispersion is also observed here.

We have only to deal with two types of solutions depending on the roots of the characteristic polynomial. Note that with the component values considered, it is not possible to obtain complex roots in the characteristic polynomial. Figure 16 compares the evolved solu-

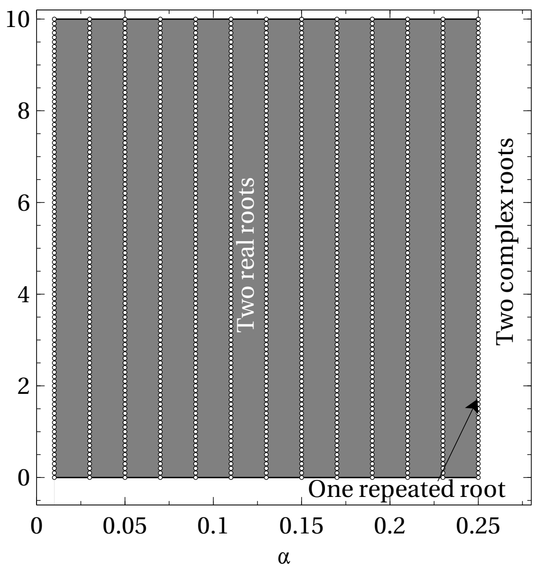

Figure 15: Design range showing α and τ parameters. The collocation points provided to the evolutionary algorithm are marked with circles.

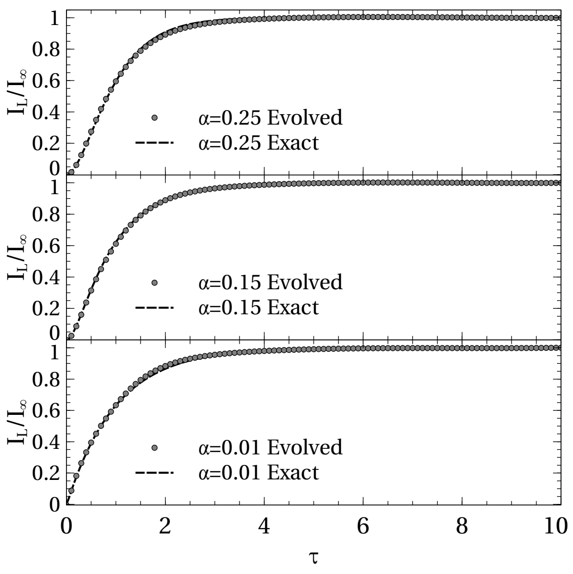

Figure 16: Current ratio I_L/I_∞ versus time τ for 3 values of α used as collocation points. Comparison between the exact solution and the evolved approximation using 15 centers.

Table 10: Results of the RLC circuit problem for different number of kernels. Average values and standard deviations are provided.

n	**Fitness**	**RMSE**	**Fit. Eval.**
5	$(1.71 \pm 0.70)\,10^{-3}$	$(8.76 \pm 1.90)\,10^{-3}$	$(3.03 \pm 0.52)\,10^{5}$
10	$(4.94 \pm 1.75)\,10^{-4}$	$(5.74 \pm 1.25)\,10^{-3}$	$(6.89 \pm 1.27)\,10^{5}$
15	$(3.84 \pm 1.69)\,10^{-4}$	$(4.50 \pm 1.36)\,10^{-3}$	$(1.14 \pm 0.21)\,10^{6}$

tion I along the time with the exact solution for three different values of α. One typical run of the evolutionary algorithm with 15 kernels is used. Note that a very good agreement is observed in all the curves. It should also be noted that all solutions are monotone (always lower than one) because all the roots of the characteristic polynomial are real, whereas the solution with complex roots would oscillate around one.

Once the differential equation Eq. (27) has been solved, a symbolic solution parametrized with the coefficient α is obtained, so we can solve the three mentioned problems above in a very efficient way. In fact, the solution to the Problem 1 is obtained from our approximation to the differential equation, see Eq. (32), where α and τ are replaced conveniently using Eq. (29) and (30), and expressing $I = f(C)$. Figure 17 shows the solution current ratio, I, at time $t = 1\mu s$ versus the value of C (top figure). Two curves are plot, one obtained with the exact solution and, the other, with the approximated solution given by the evolutionary algorithm. In order to see if we are inside the region defined by the collocation points, α vs. C (middle figure), and τ vs. C (bottom figure) are also plotted. Additionally, we can also obtain the solution to the Problem 2 from Fig. 17: the x-coordinate of the point obtained as the intersection between the current ratio curve and the horizontal line $I_L/I_\infty = 0.95$ will be the desired answer. The exact solution is $C = 6.54 \cdot 10^{-9}F$, meanwhile the evolved solution gives $C = 6.74 \cdot 10^{-9}F$, obtaining a relative error of 3%.

The Problem 3 is more challenging because it implies obtaining the voltage in the inductor. According to the characteristic equation of the inductor, the voltage V_L can be obtained as the derivative of the current I_L. This shows the benefits of the proposed approach: it is straightforward to obtain the derivative of the solution because it is given in a symbolic manner:

$$V_L = \frac{LI_\infty}{T}\frac{dI}{d\tau}. \tag{33}$$

As we see in Eq. (33), the voltage in the inductor is proportional to the derivative of the current ratio I. Figure 18 shows the voltage for several values of C both for the exact solution and the evolved one. As we see in the mentioned figure, we can plot the curve which reaches a maximum of $0.5V$. Thus, the exact solution to the Problem 3 is $C = 3.657 \cdot 10^{-9}F$, meanwhile the evolved solution gives a capacitor of $3.61 \cdot 10^{-9}F$, obtaining a relative error of 1.3%. This maximum voltage is produced in a physical time $t = 4.8 \cdot 10^{-8}$ s, which

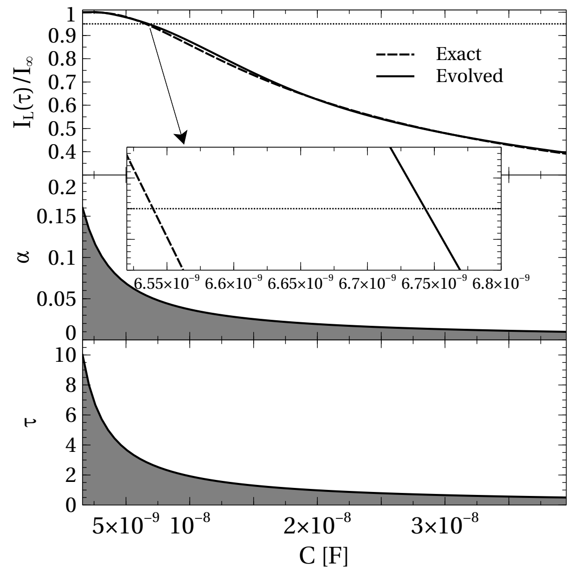

Figure 17: The current ratio I_L/I_∞ through the inductor at $t = 1\mu s$ (top figure), the coefficient of the differential equation α (middle figure) and the non-dimensional time τ (bottom figure) are plotted versus the value of the capacitor (see circuit of Fig. 14). This figure allows us to solve the problems 1 and 2 defined in section 7.1.

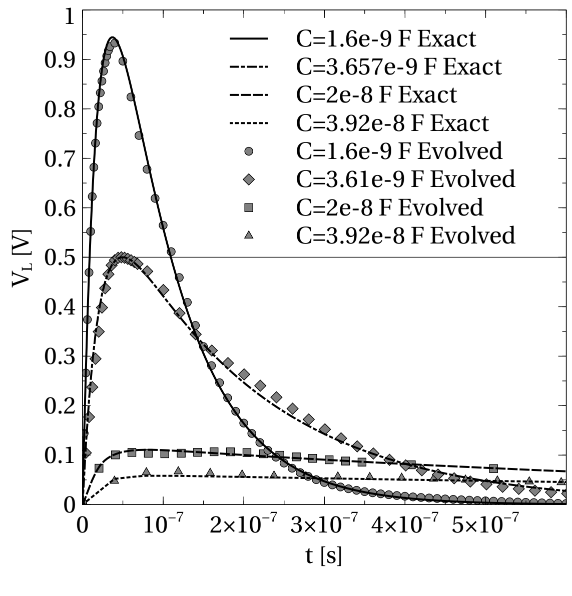

Figure 18: Voltage in the inductor versus the physical time t for different values of the capacitor C (see circuit of Fig. 14).

corresponds to a non-dimensional time $\tau = 0.24 < 10$. Therefore, it is not needed to increase the design range for collocation points shown in Fig. 15.

As a conclusion, we can say that although the proposed approach requires more computational effort than a numerical method, it could be more efficient in problems like those presented here. The proposed method obtains a symbolic solution for all values of α parameter inside a range, so derivatives of any dependent variable can be computed easily. On the contrary, a traditional method only can obtain the solution for a unique value of the α coefficient in each run.

7.2 Non-linear RLC electric circuit

In this section, a more challenging problem is sketched. Based on the circuit of Fig. 14, the constant voltage generator V is substituted by a nonlinear generator which depends on time and the current through the inductor:

$$V(t, I_L) = V_0\left(1 + 0.1\arctan\left(kI_L^2\right) + 0.1\sin(\omega t)\right), \tag{34}$$

where V_0, k and ω are constant parameters. The aim of this study case is to show the capabilities of the method to solve non-linear problems. The original Eq. (34) turns

$$\begin{array}{c}\alpha\dfrac{d^2I}{d\tau^2} + \dfrac{dI}{d\tau} + I = \\ 1 + 0.1\arctan\left(kI_\infty^2 I^2\right) + 0.1\sin(\omega T\tau)\end{array}, \tag{35}$$

where $I = I_L/I_\infty$, being $I_\infty = V_0/(R_1 + R_V)$. Note that now I_∞ is just a definition and does not correspond to the steady state current. The same setup as that one used in the linear counterpart is employed. Using 15 kernels, Fig. 19 shows the solution obtained for several values of α. Parameters k and ω are set to $1/I_\infty^2$ and $1/T$ respectively. The ranges of α and τ are the same as those ones used in the linear circuit because the same component values are assumed (except for the voltage generator).

8 Conclusions

A novel mesh free approach for solving differential equations based on CMA-ES has been presented. Unlike numerical methods, the proposed algorithm does not assume any particular structure of the differential equation. The approach has been applied to different kinds of problems: linear and nonlinear ODEs, SODEs and PDEs. Candidate solutions are built using Gaussian kernels which allows us to compute in advance all the kernel derivatives. No restrictions have been imposed on Gaussian parameters, allowing even zero or negative values for γ because we observed that the capabilities to approximate functions are enhanced. To increase the

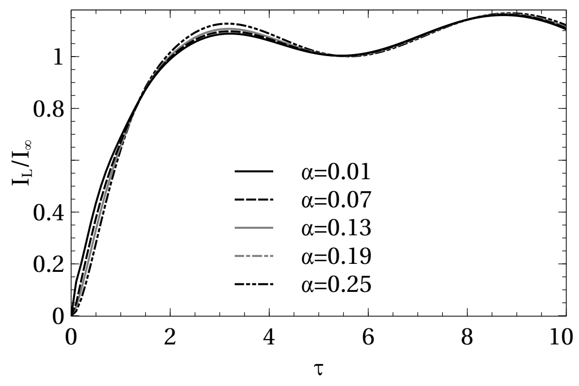

Figure 19: Current ratio curve I_L/I_∞ versus the non-dimensional time τ evolved by the evolutionary algorithm for a non-linear RLC circuit (see Fig. 14), where $V \equiv V(t, I_L)$ (see Eq. (34)). The current ratio is shown for several values of α parameter.

accuracy of the best solution found by the CMA-ES algorithm, a local search based on DS method is applied. The DS phase improves or at least maintains the quality of the evolved solution and also decreases the dispersion when several runs are used.

The proposed method has been tested in 32 different problems extracted from the literature. All the test cases have been successfully solved using the same parameter configuration. Assuming that the exact solution for each problem is known, the RMSE is used for comparing the quality of the solutions obtained: it has been observed that the quality is improved when the number of kernels increases.

Some qualitative and quantitative comparisons between our method and other evolutionary approaches reported in the literature have been presented. Several advantages of our algorithm have been commented. For example, it is easily configurable and achieves a competitive accuracy, no restrictions are used in the process of finding the solution, and a low dispersion is also observed in the results obtained from 50 runs made for each problem analyzed.

A comparative analysis was also performed solving two problems with two different numerical methods. Numerical methods are very efficient, are well developed and can cope with the majority of real problems. However, from the moment in which the problem of solving a differential equation is transformed into an optimization problem, the proposed method, in particular, and evolutionary algorithms, in general, have interesting properties (mesh-free, mathematical function as solution, etc.) that can be useful in this kind of problems. On the other hand, the main drawbacks of evolutionary methods are their non-deterministic behavior and their low convergence speed.

To show the capabilities of the proposed approach in a more practical problem, a RLC circuit analysis is presented. The circuit is modeled by a parametric differential equation and the solution is used to solve different analysis and design problems. Good accuracy in the results is observed compared with the exact solution. A non-linear version of the same circuit is also commented.

As it has been reported in this work, the number of kernels has a strong influence on the solution accuracy. Besides, only one type of kernel has been employed (Gaussian kernel) in our approach. Therefore, it could be interesting to test other types. For example, the automatic selection of the best combination of different types of kernels could improve the search of the solution. In the same sense, the optimum number of kernels could also be selected by the algorithm in an automatic way, considering a trade-off between accuracy and computational cost.

Acknowledgements Authors wish to thanks Nikolaus Hansen for sharing with the research community CMA-ES implementations in several programming languages. In particular, the ANSI C implementation[1] has been used in the present work.

Compliance with ethical standards

Conflict of interest The authors declare that have no conflict of interest.

References

T. Allahviranloo, S. Salahshour, and S. Abbasbandy. Explicit solutions of fractional differential equations with uncertainty. *Soft Computing*, 16:297–302, 2012.

M. Babaei. A general approach to approximate solutions of nonlinear differential equations using particle swarm optimization. *Applied Soft Computing*, 13(7): 3354 – 3365, 2013.

P. Balasubramaniam and A. V. A. Kumar. Solution of matrix Riccati differential equation for nonlinear singular system using genetic programming. *Genetic Programming and Evolvable Machines*, 10:71–89, 2009.

J. R. Bronson, S. P. Sastry, J. A. Levine, and R. T. Whitaker. Adaptive and unstructured mesh cleaving. In *23rd International Meshing Roundtable (IMR23)*, volume 82, pages 266–278, 2014.

K. M. Bryden, D. A. Ashlock, S. Corns, and S. J. Willson. Graph-based evolutionary algorithms. *IEEE*

[1] https://www.lri.fr/~hansen/cmaes_inmatlab.html

Transactions on Evolutionary Computation, 10(5): 550–567, 2006.

J. M. Chaquet and E. J. Carmona. Solving differential equations with Fourier series and evolution strategies. *Applied Soft Computing*, 12:3051–3062, 2012.

H. Chen, L. Kong, and W.-J. Leng. Numerical solution of PDEs via integrated radial basis function networks with adaptive training algorithm. *Applied Soft Computing*, 11:855–860, 2011.

B. Choi and J.H. Lee. Comparison of generalization ability on solving differential equations using backpropagation and reformulated radial basis function networks. *Neurocomputing*, 73:115–118, 2009.

S. Colutto, F. Fruhauf, M. Fuchs, and O. Scherzer. The CMA-ES on Riemannian manifolds to reconstruct shapes in 3-D voxel images. *IEEE Transactions on Evolutionary Computation*, 14(2):227–245, 2010.

N. N. El-Emam and R. H. Al-Rabeh. An intelligent computing technique for fluid flow problems using hybrid adaptive neural network and genetic algorithm. *Applied Soft Computing*, 11:3283–3296, 2011.

R. Guo and D. Guo. Random fuzzy variable foundation for grey differential equation modeling. *Soft Computing*, 13:185–201, 2009.

T. Hangelbroek and A. Ron. Nonlinear approximation using Gaussian kernels. *Journal of Functional Analysis*, 259(1):203 – 219, 2010.

N. Hansen. The CMA evolution strategy: A comparing review. In J.A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, editors, *Towards a New Evolutionary Computation*, volume 192 of *Studies in Fuzziness and Soft Computing*, pages 75–102. Springer Berlin Heidelberg, 2006.

N. Hansen. The CMA evolution strategy: A tutorial. `https://www.lri.fr/~hansen/cmatutorial.pdf`, June 2011.

N. Hansen and S. Kern. Evaluating the CMA evolution strategy on multimodal test functions. In *Eighth International Conference on Parallel Problem Solving from Nature PPSN VIII*, pages 282–291. Springer, 2004.

J. He, J. Xu, and X. Yao. Solving equations by hybrid evolutionary computation techniques. *IEEE Transactions on Evolutionary Computation*, 4(3):295–304, 2000a.

S. He, K. Reif, and R. Unbehauen. Multilayer neural networks for solving a class of partial differential equations. *Neural Networks*, 13:385–396, 2000b.

D. Howard, A. Brezulianu, and J. Kolibal. Genetic programming of the stochastic interpolation framework: convection-diffusion equation. *Soft Computing*, 15: 71–78, 2011.

Daniel Howard and Simon C. Roberts. Genetic programming solution of the Convection-diffusion equation. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, GECCO'01, pages 34–41. Morgan Kaufmann Publishers Inc., 2001.

J.A. Khan, R.M.A. Zahoor, and I.M. Qureshi. Swarm intelligence for the solution of problems in differential equations. In *Environmental and Computer Science, 2009. ICECS '09. Second International Conference on*, pages 141–147, 2009.

S. J. Kirstukas, K. M. Bryden, and D. A. Ashlock. A hybrid genetic programming approach for the analytical solution of differential equations. *International Journal of General Systems*, 34:279–299, 2005.

M. Kumar and N. Yadav. Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations. A survey. *Computers and Mathematics with Applications*, 62:3796–3811, 2011.

I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 5:987–1000, 1998.

R. J. Leveque. *Finite Volume Methods for Hyperbolic Problems.* Cambridge University Press, 2002.

G. R. Liu. *Meshfree Methods. Moving Beyond the Finite Element Method.* CRC Press, Inc., 2010.

P.E. MacNeil. A technique for generating approximate solutions and its application to coulomb interactions. In *Southeastcon, 2012 Proceedings of IEEE*, pages 1–5, 2012.

S. Mehrkanoon and J. A. K. Suykens. Learning solutions to partial differential equations using LS-SVM. *Neurocomputing*, 159:105–116, 2015.

M. Mosleh. Fuzzy neural network for solving a system of fuzzy differential equations. *Applied Soft Computing*, 13(8):3597–3607, 2013.

J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

M. N. Ozisik. *Finite Difference Methods in Heat Transfer.* CRC Press, Inc., 1994.

N. Panagant and S. Bureerat. Solving partial differential equations using a new differential evolution algorithm. *Mathematical Problems in Engineering*, 2014 (747490):1–10, 2014.

D. R. Parisi, M. C. Mariani, and M. A. Laborde. Solving differential equations with unsupervised neural networks. *Chemical Engineering and Processing*, 42: 715–721, 2003.

L. E. Peterson. Covariance matrix self-adaptation evolution strategies and other metaheuristic techniques for neural adaptive learning. *Soft Computing*, 15:598, 2011.

W. H. Press, W. T. Vetterling, S. A. Teukolsky, and B. P. Flannery. *Numerical Recipes in C++: the art of scientific computing.* Cambridge University Press, New York, NY, USA, 2nd edition, 2002.

F. Puffer, R. Tetzlaff, and D. Wolf. A learning algorithm for cellular neural networks (CNN) solving nonlinear partial differential equations. In *International Symposium on Signals, Systems, and Electronics (ISSSE '95)*, pages 501–504, 1995.

K. Rudd and S. Ferrari. A constrained integration (cint) approach to solving partial differential equations using artificial neural networks. *Neurocomputing*, 155: 277–285, 2015.

T. Seaton, G. Brown, and J. F. Miller. Analytic solutions to differential equations under graph-based genetic programming. In Anna Isabel Esparcia-Alcázar, Anikó Ekárt, Sara Silva, Stephen Dignum, and A. Şima Uyar, editors, *Genetic Programming: 13th European Conference, EuroGP 2010*, pages 232–243. Springer Berlin Heidelberg, 2010.

Y. Shirvany, M. Hayati, and R. Moradian. Multilayer perceptron neural networks with novel unsupervised training method for numerical solution of the partial differential equations. *Applied Soft Computing*, 9:20–29, 2009.

A. Sobester, P. B. Nair, and A. J. Keane. Genetic programming approaches for solving elliptic partial differential equations. *IEEE Transactions on Evolutionary Computation*, 12:469–478, 2008.

M. Spivak. *Calculus.* Publish or Perish, 4 edition, 1980.

E. Suli and D. F. Mayers. *An Introduction to Numerical Analysis.* Cambridge University Press, 2003.

Mingui Sun, Xiaopu Yan, and R.J. Sclabassi. Solving partial differential equations in real-time using artificial neural network signal processing as an alternative to finite-element analysis. In *International Conference on Neural Networks and Signal Processing, 2003*, volume 1, pages 381–384, 2003.

J. F. Thompson, B. K. Soni, and N. P. Weatherill. *Handbook of Grid Generation.* CRC Press, 1999.

I. G. Tsoulos and I. E. Lagaris. Solving differential equations with genetic programming. *Genetic Programming and Evolvable Machines*, 7:33–54, 2006.

I. G. Tsoulos, D. Gavrilis, and E. Glavas. Solving differential equations with constructed neural networks. *Neurocomputing*, 72:2385–2391, 2009.

K. Yao. Uncertain differential equation with jumps. *Soft Computing*, 19:2063–2069, 2015.

H. S. Yazdi and R. Pourreza. Unsupervised adaptive neural-fuzzy inference system for solving differential equations. *Applied Soft Computing*, 10:267–275, 2010.

H. S. Yazdi, M. Pakdaman, and H. Modaghegh. Unsupervised kernel least mean square algorithm for solving ordinary differential equations. *Neurocomputing*, 74:2062–2071, 2011.