Solving Differential Equations with Fourier Series and Evolution Strategies

Jose M. Chaquet*, Enrique J. Carmona

Dpto. de Inteligencia Artificial, Escuela Técnica Superior de Ingeniería Informática, Universidad Nacional de Educación a Distancia, Madrid, Spain

Abstract

A novel mesh-free approach for solving differential equations based on Evolution Strategies (ESs) is presented. Any structure is assumed in the equations making the process general and suitable for linear and nonlinear ordinary and partial differential equations (ODEs, PDEs), as well as systems of ordinary differential equations (SODEs). Candidate solutions are expressed as partial sums of Fourier series. Taking advantage of the decreasing absolute value of the harmonic coefficients with the harmonic order, several ES steps are performed. Harmonic coefficients are taken into account one by one starting with the lower order ones. Experimental results are reported on several problems extracted from the literature to illustrate the potential of the proposed approach. Two cases (an initial value problem and a boundary condition problem) have been solved using numerical methods and a quantitative comparative is performed. In terms of accuracy and storing requirements the proposed approach outperforms the numerical algorithm.

Keywords: Differential equations, Fourier series, evolution strategies, mesh-free methods, harmonic analysis

1. Introduction

Differential equations are mathematical equations for one or several unknown functions that relate the values of the functions themselves and their derivatives of various orders. Differential equations play a prominent role in engineering, physics, economics, and other disciplines. Some important examples are the Newton's Second Law in dynamics, the Maxwell's equations in electromagnetism, the heat equation in thermodynamics, Einstein's field equation in general relativity, Schrödinger equation in quantum mechanics or the Navier-Stokes equations in fluid dynamics [6].

*Corresponding author. Tel.: +34 91 398 7301

Email addresses: jose.chaquet@gmail.com (Jose M. Chaquet), ecarmona@dia.uned.es (Enrique J. Carmona)

Some simple differential equations admit solutions given by explicit formulas. But in the general case, only approximate solutions can be found. Among the engineering community, the most popular methods for solving differential equations use numerical analysis techniques such as finite element method (FEM) [21], finite difference method [14], or finite volume method [11]. These approaches relied on a grid or a mesh for discretizing the equations, bringing them into a finite-dimensional subspace. The original problem is reduced to the solution of algebraic equations.

On the other hand, mesh-free methods work with a set of arbitrary distributed points without using any mesh that provides the connectivity of these nodes. Some examples of mesh-free methods are Smoothed particle hydrodynamics (SPH), Diffusive element method (DEM) and Point Interpolation Method (PIM) among others [12].

Other mesh-free methods have its inspiration in the artificial intelligence field. For instance Legaris et al. [10] use a feed forward neural network to codify the solution of this type of problems. The trial solutions are computed as a sum of two parts. The first part satisfies the initial/boundary conditions and contains no adjustable parameters. The second part is constructed so as not to affect the initial/boundary conditions. This approach is problem dependent and in some cases could be difficult to split the candidate solutions in the two terms. Neural network weights and bias are optimized using a quasi-Newton Broyden-Fletcher-Goldfarb-Shanno method. This approach has been successfully applied to a system of partial differential equations which models a non-steady fixed bed non-catalytic solid-gas reactor [15]. In this last work, the boundary condition error is added to the cost or fitness function as a penalty term.

Nowadays an increasing interest in solving differential equations using artificial neural networks is observed. In [19] a Multilayer Perceptron and Radial Basis Function neural network is successfully applied to the nonlinear Schröndinger equation in hydrogen atom. Yazdi et al. [24] combine a neural network and a fuzzy system to solve some simple first and second order ordinary linear differential equations. Fast convergence is achieved training the adaptive network-based fuzzy inference system in unsupervised way. In [3] other mesh-free numerical method for solving PDEs based on integrated radial basis function networks with adaptive residual subsampling training scheme is presented. Numerical experiments solving several PDEs show that this algorithm with the adaptive procedure requires fewer neurons to attain the desired accuracy than conventional radial basis function networks. A different approach dealing with neural networks consist of solving a family of differential equations using traditional methods and training a neural network for building surrogate models. Following this line, work [5] presents a new hybrid adaptive neural network with modified adaptive smoothing errors based on genetic algorithm to construct a learning system for complex problem solving in fluid dynamics. The system can predict an incompressible viscous fluid flow represents by stream function through symmetrical backward-facing steps channels.

Recently new methods for solving differential equations using Genetic Programming (GP) have been reported. These approaches can be considered mesh-

free methods because the derivatives are computed symbolically, so any node connectivity is needed. Sobester et al. [20] propose a technique for the mesh-free solution of elliptic partial differential equations where least-squares collocation principle has been employed to define an appropriate objective function, which is optimized using GP. In that work no particular function basis is used, but symbolic regression is performed. This makes the search space very large. Another GP approach can be seen in [8] where polynomials are used for solving the convective-diffusion equation. In the same line of research, Tsoulos and Legaris [23] use a similar technique, with the novelty of evolving the candidate solutions using Grammatical Evolution (GE) [13]. This technique has been employed successfully for solving the matrix Ricatti differential equation for nonlinear singular system [1]. GE has been used as well for enhanced the constructed neural method in [22]. The main advantage of this last approach is that the user does not choose a priori the number of neuron cells. In that contribution local search is employed over some individuals.

Seaton et al. [18] investigate the influence of the problem complexity and perform a search analysis when differential equations are solved within an evolutionary framework. They show that reducing the search space can improve significantly the algorithm performances. A possible approach for reducing the search space dimension is using some kind of function basis for building candidate solutions. This idea is used by Kirstukas et al. [9], where a hybrid GP approach from an engineering perspective is employed. In that approach, for the particular case of linear differential equations, a modified Gram-Schmidt algorithm is used to reduce the set of general solutions located by GP to a function basis set.

In the present work a novel mesh-free method for solving differential equations is reported. Candidate solutions are expressed as partial sums of Fourier series. In order to simplify the problem, an even periodic expansion of the solutions is done in such a way that all the sine coefficients are vanished. This representation can be regarded equivalent to a Discrete Cosine Transform (DCT) [17] which has been successfully used in several science and engineering applications, as for lossy compression of audio (MP3) and image (JPEG). With the chosen solution representation, the problem of solving differential equations is transformed into an optimization one, where the differential equation residuals and the boundary condition errors are minimized. The optimal Fourier coefficients are sought using Evolution Strategies (ESs). In order to systematize the process, the harmonic searching is done in a progressive way starting with the lowest order harmonic and using a different ES cycle to find the optimum value for each one.

The rest of the paper is organized as follows: In Section 2 a description of the proposed approach is given. In Section 3 a set of test cases extracted from the literature is described and experimental results are reported. Section 4 gives some qualitative and quantitative comparisons with numerical methods and other evolutionary approaches. Finally, the conclusions and some future work guides are outlined in Section 5.

2. Method description

In this section the proposed method is described. First the mathematical statement of the problem is given in subsection 2.1. The particular coding of candidate solutions using Fourier series is explained in subsection 2.2. Each optimal harmonic coefficient is sought using several ES steps. Subsection 2.3 describes these particular steps, and subsection 2.4 explains how the steps are combined for solving the global optimization problem.

2.1. Statement of the problem

Using the same notation than Sobester et al. [20] but extending the original problem to systems of differential equations, we consider the general equation

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

subject to the boundary conditions

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

where $\mathbf{L}$ and $\mathbf{B}$ are differential operators in the space $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{y}\left(\mathbf{x}\right)$ denotes the unknown solution vector. Functions $\mathbf{f}\left(\mathbf{x}\right)$ and $\mathbf{g}\left(\mathbf{x}\right)$ denote source terms, so only depend on $\mathbf{x}$, but not on $\mathbf{y}$ or its derivatives. From a general point of view, $\mathbf{y}\left(\mathbf{x}\right)$, $\mathbf{f}\left(\mathbf{x}\right)$ and $\mathbf{g}\left(\mathbf{x}\right)$ belong to the set of vector-valued functions $\mathbb{R}^d \rightarrow \mathbb{R}^m$. $\Omega \subset \mathbb{R}^d$ is a bounded domain and $\partial\Omega$ denotes its boundary[1]. Note that if $d = 1$ and $m = 1$, we have an ODE problem. If $d = 1$ and $m > 1$, a SODE problem is managed and, finally, if $d > 1$ and $m = 1$, a PDE problem is established. The solution vector satisfying (1) and (2) can be computed solving the following *Constrained Optimization Problem* (COP):

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

where $\|\cdot\|$ denotes the Euclidean norm in $\mathbb{R}^d$ space. This problem is discretized using a set of collocation points $C = \{(\mathbf{x}_i) \mid_{i=1,\cdots,n_C} \subset \Omega\}$ situated within the domain and as well on the boundary $B = \{(\mathbf{x}_j) \mid_{j=1,\cdots,n_B} \subset \partial\Omega\}$. Finally the original COP is transformed into a *Free Constrained Optimization Problem* defining a cost function as follows

$$F(\mathbf{y}) = \frac{1}{d \cdot (n_C + n_B)} \left[\sum_{i=1}^{n_C} \left\|\mathbf{L}\mathbf{y}\left(\mathbf{x}_i\right) - \mathbf{f}\left(\mathbf{x}_i\right)\right\|^2 + \varphi \sum_{j=1}^{n_B} \left\|\mathbf{B}\mathbf{y}\left(\mathbf{x}_j\right) - \mathbf{g}\left(\mathbf{x}_j\right)\right\|^2 \right], \tag{4}$$

[1] This notation corresponds to elliptic equations appearing in the solution of boundary value problems. Other kind of differential equations such as initial value problems can be treated in a similar way.

where φ is a penalty parameter. Note that the cost function is obtained dividing the residuals by the total number of collocation points $d \cdot (n_C + n_B)$ in a similar way than Parisi et al. [15]. Other authors [1, 10, 20] do not make this normalization, which makes their values more dependent on the number of collocation points.

2.2. Candidate solutions

In the proposed approach, each component $y(\mathbf{x})$ of the trial solution is expressed as a partial sum of a Fourier series. The periodic expansion of $y(\mathbf{x})$ from the original definition range to all $\mathbb{R}^d$ is always performed using even functions. Therefore all the sine Fourier coefficients are vanished. In order to define this expansion, first some notation must be introduced. For each coordinate x_k with $k = 1, \cdots, d$, variables $x_{k,min}$ and $x_{k,max}$ are defined as the minimum and maximum values among the inner collocation points C and the boundary condition points B. Using these values and an user defined parameter $\xi \geq 0$ called *range extension*, a new coordinate origin c_k and a semi-period L_k are defined as

$$c_k = x_{k,min} - \xi \left(x_{k,max} - x_{k,min} \right) \tag{5}$$

$$L_k = \left(x_{k,max} - x_{k,min} \right) (1 + 2\xi) \tag{6}$$

Then each component $y(\mathbf{x})$ of the solution vector is expressed as a partial sum of Fourier Cosine series:

$$y(x_1, \ldots, x_d) = \frac{a_0}{2} + \sum_{n_1,\ldots,n_d=1}^{N} a_{n_1,\ldots,n_d} \prod_{k=1}^{d} \cos\left(\frac{\pi n_k}{L_k} \left(x_k - c_k \right) \right) \tag{7}$$

where a_0 and $a_{n_1,\ldots,n_d}$ are the unknown coefficients or *harmonics* and N is an user defined parameter which determines the number of harmonics used. The total number of harmonics for each component will be $1 + N^d$. By definition, this expanded function is periodic in each dimension with period $2L_k$ and, in addition, is defined everywhere, continuous and infinitely differentiable. However, according to Eq. (4), this function will be only evaluated in the original definition range. Therefore the expansion can be done anyhow with the following constraints: the expanded function must be periodic, even and solution to the original problem in Ω and $\partial\Omega$.

Range extension parameter ξ is needed in order to suppress the intrinsic limitations of even functions at the boundaries regarding the first partial derivative:

$$\left. \frac{\partial y(x_1, \ldots, x_d)}{\partial x_k} \right|_{x_k = c_k} = 0. \tag{8}$$

Note that if $\xi = 0$ the null first derivative will be obtained at points $x_k = x_{k,min}$, which could be interesting in some particular cases. Nevertheless, a general problem will have not-null first derivatives at boundaries. Because non discontinuities are introduced, neither in the expanded function itself nor in their

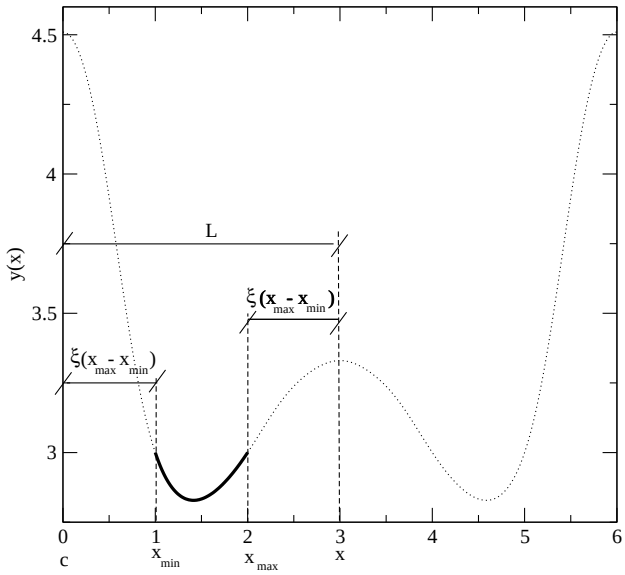

Figure 1: Even periodic expansion example of a function originally defined in the range $[1, 2]$.

derivatives, convergence problems regarding Gibbs phenomenon (large oscillations of the nth partial sum of the Fourier series near the discontinuity jump) are avoided. According to expression (7), Fig. 1 shows an even periodic expansion example of a function originally defined in the range $[1, 2]$, in one dimension ($d = 1$) with a *range expansion* $\xi = 1$. Note the null first derivatives at points $x = 0$, $x = 3$ and $x = 6$.

Computing the cost function (4) implies obtaining derivatives of expression (7). A generic derivative operator D can be expressed as

$$D = \frac{\partial^{\lambda_1}}{\partial x_1^{\lambda_1}} \frac{\partial^{\lambda_2}}{\partial x_2^{\lambda_2}} \cdots \frac{\partial^{\lambda_d}}{\partial x_d^{\lambda_d}}. \tag{9}$$

Using Eq. (7), the differential operator D applied to the dependant variable y yields

$$D\left(y\right) = \sum_{n_1,\ldots,n_d=1}^{N} a_{n_1,\ldots,n_d} \prod_{k=1}^{d} \left(\frac{\pi n_k}{L_k}\right)^{\lambda_k} \cos^{(\lambda_k)}\left(\frac{\pi n_k}{L_k}\left(x_k - c_k\right)\right) \tag{10}$$

where the nth derivative of cosine function can be computed as

$$\cos^{(n)} x = \begin{cases} \cos x & if \quad n\%4 = 0 \\ -\sin x & if \quad n\%4 = 1 \\ -\cos x & if \quad n\%4 = 2 \\ \sin x & if \quad n\%4 = 3 \end{cases}. \tag{11}$$

2.3. Evolution Strategy

The optimization problem of searching the best set of harmonic coefficients is solved using a ES which is an optimization technique based on ideas of adaptation and evolution [2]. Among all the Evolutionary Computing paradigms, ES has been chosen because they are typically used for continuous parameter optimization problems and its very useful feature: self-adaptation of strategy parameters [4]. There is a strong emphasis on mutation for creating offspring. In this approach uncorrelated mutation with several step sizes is used. Regarding the genotype coding, each component of the solution vector is represented by

$$\left[\underbrace{a_0, a_{1,\cdots,1}, \cdots, a_{N,\cdots,N}}_{\mathbf{a}}, \underbrace{\sigma_0, \sigma_{1,\cdots,1}, \cdots, \sigma_{N,\cdots,N}}_{\boldsymbol{\sigma}}\right]. \tag{12}$$

The first part of the genotype, $\mathbf{a}$, codifies all the harmonic coefficients needed for building the individual's phenotype using Eq. (7). For each individual, a fitness value can be computed using Eq. (4). The second part of the vector, $\boldsymbol{\sigma}$, codifies the mutation strengths for each harmonic.

Within one ES generational cycle, λ offspring individuals are generated from a set of μ parent individuals using recombination and mutation operators. Then selection operator chooses those individuals which will form the population in the next generation. The process will be repeated in a close loop until some stop condition is fulfilled.

Global discrete recombination is used for the harmonics, and global intermediate recombination is performed for the mutation strengths. That is,

$$\begin{aligned} a_{\mathbf{n}}^{child} &= a_{\mathbf{n}}^{parent1} \quad or \quad a_{\mathbf{n}}^{parent2} \\ \sigma_{\mathbf{n}}^{child} &= \left(\sigma_{\mathbf{n}}^{parent1} + \sigma_{\mathbf{n}}^{parent2}\right)/2 \end{aligned} \tag{13}$$

where $\mathbf{n} \equiv n_1, \cdots, n_d$ is an index vector according to expression (7) and the parents are chosen randomly among all the population. This scheme of recombination is the most used in ES implementations because preserves diversity within the phenotype space, allowing the trial of very different combinations of values, whilst the averaging effect of intermediate recombination assures a more cautious adaptation of mutation strengths [4].

Each λ offspring individual is mutated using independent random samples from a standard normal distribution $N(0,1)$

$$\sigma'_{\mathbf{n}} = \sigma_{\mathbf{n}} \exp\left(\tau' N(0,1) + \tau N_i(0,1)\right) \tag{14}$$

$$a'_{\mathbf{n}} = a_{\mathbf{n}} + \sigma'_{\mathbf{n}} N_i(0,1) \tag{15}$$

where τ and τ' are the learning rates defined as $\tau' = f_\tau/\sqrt{2\left(1+N^d\right)}$ and $\tau = f_\tau/\sqrt{2\sqrt{1+N^d}}$ being f_τ a learning rate factor defined by the user. In order to avoid too small mutation steps, a threshold ϵ is applied in the following way:

$$if\ \sigma'_{\mathbf{n}} < \epsilon \Longrightarrow \sigma'_{\mathbf{n}} = \epsilon \tag{16}$$

The threshold used in the present work is $\epsilon = 10^{-20}$. In order to guarantee extinction of misfit individuals, the classical selection process (μ, λ) is used. However, it is modified adding elitism of one individual. That is, the next generation is formed by the best μ individuals among the λ mutated offspring and the best individual among all the parents. In this way, the fitness of the best individual in the population is a monotonic function with the generation number. Two stop conditions are checked: maximum number of generations or unchanged fitness of the best individual during a predefined number of generations. The tolerance used in the present work to distinguish unchanged fitness values is 10^{-9}.

2.4. Global algorithm

In this section a global strategy for solving the original problem is presented. The basic idea consists of introducing harmonics one by one in the evolutionary process starting with the lower order ones. This strategy is based on the assumption that the absolute values of Fourier coefficients decrease when the harmonic number is increased. There are several works that gives bounds to the Fourier coefficients assuming some properties to the original function [7]. For instance, if we assume that y is an absolutely continuous function of one real variable, then the nth Fourier coefficient a_n fulfils

$$|a_n| \leq \frac{K}{n}, \tag{17}$$

where the constant K only depends on y but not on n. Better bounds can be given if more features are assumed on y. This property has been observed experimentally in all the test problems. In Fig. 2 the absolute value of the first ten Fourier coefficients $|a_n|$ are shown for ODE1 case (the test cases will be introduced in section 3). We can observe the decay of the absolute values when the harmonic number is increased. Note the logarithm scale on the vertical axis.

The global algorithm consists of several basic steps, called *ES steps*. Each *ES step* is instantiated as it was explained in subsection 2.3. In addition, a global flag of three possible values (*active*, *inactive* and *frozen*), named $flag_{\mathbf{n}}$, is associated to each harmonic coefficient $a_{\mathbf{n}}$. Each $flag_{\mathbf{n}}$ is initialized before running an *ES step*, does not evolve and has the same value for all individuals in the population. If a harmonic is *active*, it participates for computing the fitness value of the individual, using Eq. (4), and is evolved by the *ES step*. If a harmonic is *inactive*, it is not used (zero value) for computing the fitness value and is not evolved. Finally, if the harmonic is *frozen*, it is used for computing the fitness value but is not evolved by the *ES step*. Representing each *ES step* by $ES\,[m_{low}, m_{high}]$, each harmonic in an individual is classified as follows: harmonic $a_{n_1,\ldots,n_d}$ will be *active* if there is at least one index $n_i \in [m_{low}, m_{high}]$ and the remaining indexes are $n_j \leq m_{high}$. A harmonic will be considered *frozen* when for all indexes $n_j < m_{low}$. Finally, a harmonic will be considered *inactive*

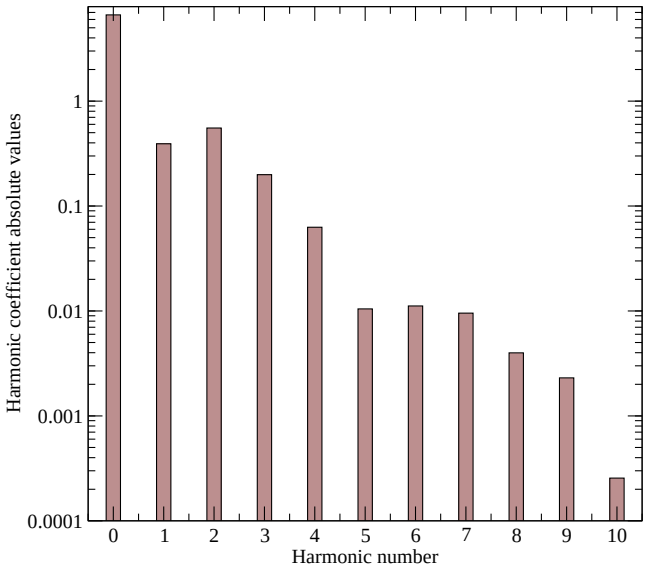

Figure 2: First 10 Fourier coefficients computed for ODE1 case.

if there is at least one index $n_j > m_{high}$. Using the aforementioned notation, the global strategy can be implemented by a sequence of *ES steps* represented by the algorithm shown in Fig. 3.

Note that there is always an *ES step* of type $ES\,[0, m]$ after of an *ES step* of type $ES\,[m, m]$. As can be seen easily, a new harmonic is tuned (*activated*) in an *ES step* of type $ES\,[m, m]$. Then, all the harmonics lower or equal than the new one are tuned in a step of type $ES\,[0, m]$. With this policy the search space dimension is reduced making the searching process more systematic and the optimization problem easier. The steps $ES\,[0, 0]$ and $ES\,[1, 1]$ are not considered because the algorithm starts tuning the two first harmonic simultaneously ($ES\,[0, 1]$). The final step is called *Fine Tuning* phase and its aim is to adjust finely all the harmonics simultaneously, so all of them are activated. The stop criterion for each *ES step* is fulfilled when the best fitness during a predefined number of consecutive generations is not modified within a given tolerance or when a predefined maximum number of generations G are fulfilled. These three parameters (number of generations, tolerance and G) are input parameter for the algorithm.

Before running each *ES step*, the population must be initialized. The initialization policy is different for *ES steps* of types $ES\,[m, m]$, $ES\,[0, m]$ or the *Fine Tuning* phase. The population initialization in each *ES step* of type $ES\,[m, m]$ can be summarized as follows:

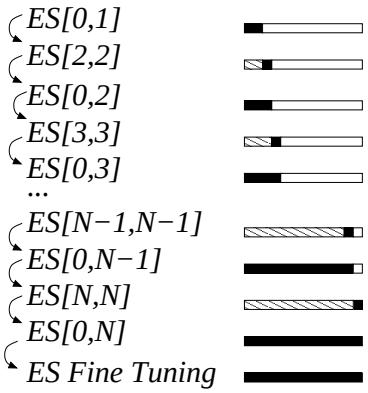

Figure 3: Global strategy as a sequence of *ES steps*. Each bar represents, going from left to right, the *frozen* harmonics (pattern filled), the *active* (black color) and the *inactive* ones (white color).

$$a_{\mathbf{n}} = \begin{cases} U(\alpha,\beta) & if \quad flag_{\mathbf{n}} = active \\ \hat{a}_{\mathbf{n}} & if \quad flag_{\mathbf{n}} = frozen \\ 0 & if \quad flag_{\mathbf{n}} = inactive \end{cases}, \tag{18}$$

where $U(\alpha,\beta)$ is a random sample from a continuous uniform distribution in the range $[\alpha,\beta]$, being α and β user defined parameters. The symbol $\hat{a}_{\mathbf{n}}$ denotes the harmonic with index $\mathbf{n}$ of the best individual at the final population of the previous *ES step* run. This policy avoids the algorithm being trapped in local optima when a new harmonic is used. On the other hand, it is not performed any particular initialization in steps of type $ES[0,m]$ or *Fine Tuning* steps. Therefore the initial population is copied from the final population of the previous *ES step*.

Mutation strengths are initialized in a similar way. For steps of type $ES[m,m]$ it is used the next expression

$$\sigma_{\mathbf{n}} = \begin{cases} U(\gamma,\delta) & if \quad flag_{\mathbf{n}} = active \\ 0 & if \quad flag_{\mathbf{n}} = inactive \quad or \quad frozen \end{cases}, \tag{19}$$

where the range of the random variable $U(\gamma,\delta)$ is as well a user defined parameter. Initialization for $ES[0,m]$ is done in a range ten times lower than in expression (19) in order to re-adjust lower harmonics when the new harmonic has been computed:

$$\sigma_{\mathbf{n}} = \begin{cases} U(\gamma,\delta)/10 & if \quad flag_{\mathbf{n}} = active \\ 0 & if \quad flag_{\mathbf{n}} = inactive \quad or \quad frozen \end{cases}. \tag{20}$$

Finally, all the mutation strengths in *Fine Tuning* phase are initialized to a small value (10^{-7} in the experiments).

In PDEs, an increment of the harmonic order implies a non linear increment in the number of harmonics depending on the problem dimension. For instance,

for a two dimensional problem, going from $ES\,[0,3]$ to $ES\,[0,4]$ implies an increment in the number of harmonics of 7.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \Longrightarrow \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}. \tag{21}$$

Therefore, in PDE problems, the maximum number of new coefficients to tune in each $ES\,[m,m]$ phase could be specified as another algorithm input. In this way, this type of phases would be split into several sub-phases.

3. Experimental results

In this section experimental results are presented. First a detailed description of each test problem is given. Then a quality measure of the final solutions, that is different from the fitness function, is explained. Due to the stochastic nature of the algorithm, results are presented as an average of 10 independent runs for each problem. Standard deviation values are given as well. Of a total of 26 test cases studied, we have only identified some convergence problems in one of them. In that problematic case, further runs are reported in order to clarify the behaviour of the algorithm.

3.1. Test cases

The proposed method was tested on several problems: linear and nonlinear ODEs, SODEs and PDEs with two variables and Dirichlet boundary conditions. These test cases have been extracted from several previous works [10, 23, 20, 22, 24, 3]. In table 1 all the problems are given: differential equations, independent variable ranges and boundary conditions. The exact solutions are provided as well in table 2 with the exception of Allen-Cahn equation [3] in NLODE6, which does not have a close analytical solution. Note that PDE5 and PDE6 are defined on non-rectangular domains (Fig. 4).

3.2. Quality solution measure: Root of the Mean Squared Error

Fitness function value is not a good measure for comparing the final solutions obtained in different problems because it is very dependent on some algorithm parameters as for example the boundary condition penalty φ. Moreover, the fitness value can be modified depending on how the differential equation is provided to Eq. (4). For instance, ODE1 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 function values are modified: First and last equations have the same fitness landscape, but with a scale value given by the

Case	Equation	Range	Boundary Conditions
ODE1	$y' = (2x - y)/x$	$x \in [1, 2]$	$y(1) = 3$
ODE2	$y' = (1 - y\cos(x))/\sin(x)$	$x \in [1, 2]$	$y(1) = 3/\sin 1$
ODE3	$y'' = 6y' - 9y$	$x \in [0, 1]$	$y(0) = 0; y'(0) = 2$
ODE4	$y'' + \frac{1}{5}y' + y = -\frac{1}{5}e^{-x/5}\cos(x)$	$x \in [0, 1]$	$y(0) = 0; y'(0) = \sin(0.1)/e^{0.2}$
ODE5	$y'' + \frac{1}{5}y' = \frac{1}{5}\cos(x)$	$x \in [0, 1]$	$y(0) = 0; y'(0) = 1$
ODE6	$y'' + 2xy = 0$	$x \in [0, 1]$	$y(0) = 0; y'(0) = 1$
ODE7	$y''\left(x^2 + 1\right) - 2xy - x^2 - 1 = 0$	$x \in [0, 1]$	$y(0) = 0; y'(0) = 1$
ODE8	$y' + 2y = 1$	$x \in [0, 10]$	$y(0) = 1$
ODE9	$y' + 2y = \sin(x)$	$x \in [0, 10]$	$y(0) = 1$
ODE10	$y'' = -16\pi^2\sin(4\pi x)$	$x \in [0, 1]$	$y(0) = 2; y(1) = 2$
NLODE1	$y' = 1/(2y)$	$x \in [1, 4]$	$y(1) = 1$
NLODE2	$(y')^2 + \log y = \cos^2 x + 2\cos x + 1 + \log(x + \sin x)$	$x \in [1, 2]$	$y(1) = 1 + \sin 1$
NLODE3	$y''y' = -4/x^3$	$x \in [1, 2]$	$y(1) = 0; y'(1) = 2$
NLODE4	$x^2y'' + (xy')^2 + 1/\log x = 0$	$x \in [e, 2e]$	$y(e) = 0; y'(e) = 1/e$
NLODE5	$y'' - yy'/\left(x\sin x^2\right) = -4x^2\sin x^2$	$x \in [1, 2]$	$y(1) = \sin 1; y(2) = \sin 4$
NLODE6	$10^{-4}y'' + y + y^3 = 0$	$x \in [-1, 1]$	$y(-11) = -1; y(1) = 1$
SODE1	$y_1' = \cos x + y_1^2 + y_2 - \left(x^2 + \sin^2 x\right)$ $y_2' = 2x - x^2\sin x + y_1y_2$	$x \in [0, 1]$	$y_1(0) = 0$ $y_2(0) = 0$
SODE2	$y_1' = (\cos x - \sin x)/y_2$ $y_2' = y_1y_2 + e^x - \sin x$	$x \in [0, 1]$	$y_1(0) = 0$ $y_2(0) = 1$
SODE3	$y_1' = \cos x$ $y_2' = -y_1$ $y_3' = y_2$ $y_4' = -y_3$ $y_5' = y_4$	$x \in [0, 1]$	$y_1(0) = 0$ $y_2(0) = 1$ $y_3(0) = 0$ $y_4(0) = 1$ $y_5(0) = 0$
SODE4	$y_1' = -\sin(e^x)/y_2$ $y_2' = -y_2$	$x \in [0, 1]$	$y_1(0) = \cos 1$ $y_2(0) = 1$
PDE1	$\nabla^2\Psi(x, y) = e^{-x}\left(x - 2 + y^3 + 6y\right)$	$x, y \in [0, 1]$	$\Psi(0, y) = y^3; \Psi(1, y) = \left(1 + y^3\right)e^{-1}$ $\Psi(x, 0) = xe^{-x}; \Psi(x, 1) = (x + 1)e^{-x}$
PDE2	$\nabla^2\Psi(x, y) = -2\Psi$	$x, y \in [0, 1]$	$\Psi(0, y) = 0; \Psi(1, y) = \sin(1)\cos(y)$ $\Psi(x, 0) = \sin(x); \Psi(x, 1) = \sin(x)\cos(1)$
PDE3	$\nabla^2\Psi(x, y) = 4$	$x, y \in [0, 1]$	$\Psi(0, y) = y^2 + y + 1; \Psi(1, y) = y^2 + y + 3$ $\Psi(x, 0) = x^2 + x + 1; \Psi(x, 1) = x^2 + x + 3$
PDE4	$\nabla^2\Psi(x, y) = -\Psi\left(x^2 + y^2\right)$	$x, y \in [0, 1]$	$\Psi(0, y) = 0; \Psi(1, y) = \sin(y)$ $\Psi(x, 0) = 0; \Psi(x, 1) = \sin(x)$
PDE5	$\nabla^2\Psi(x, y) = 4x\cos x + \left(5 - x^2 - y^2\right)\sin x$	$x^2 + y^2 \leq 1$	$\Psi(x, y) = 0$ *in* $\partial\Omega$
PDE6	$\nabla^2\Psi(x, y) = 2e^{(x-y)}$	$R^2(\theta) \leq \cos(2\theta) + \sqrt{1.1\sin^2(2\theta)}$	$\Psi(x, y) = e^{(x-y)} + e^x\cos y$ *in* $\partial\Omega$

Table 1: Test cases: differential equations, ranges and boundary conditions.

Case	Exact Solution
ODE1	$y = x + 2/x$
ODE2	$y = (x+2)/\sin(x)$
ODE3	$y = 2xe^{3x}$
ODE4	$y = e^{-x/5}\sin(x)$
ODE5	$y = \int_0^x \frac{\sin(t)}{t} dt$
ODE6	$y = \int_0^x e^{-t^2} dt$
ODE7	$y = \left(x^2+1\right)\arctan(x)$
ODE8	$y = \left(e^{-2x}+1\right)/2$
ODE9	$y = \left[6e^{-2x} + 2\sin(x) - \cos(x)\right]/5$
ODE10	$y = 2 + \sin(4\pi x)$
NLODE1	$y = \sqrt{x}$
NLODE2	$y = x + \sin(x)$
NLODE3	$y = \log\left(x^2\right)$
NLODE4	$y = \log(\log(x))$
NLODE5	$y = \sin\left(x^2\right)$
NLODE6	No analytical solution
SODE1	$y_1 = \sin x$, $y_2 = x^2$
SODE2	$y_1 = \sin(x)/e^x$, $y_2 = e^x$
SODE3	$y_1 = y_3 = y_5 = \sin x$, $y_2 = y_4 = \cos x$
SODE4	$y_1 = \cos(e^x)$, $y_2 = e^{-x}$
PDE1	$\Psi(x,y) = \left(x+y^3\right)e^{-x}$
PDE2	$\Psi(x,y) = \sin(x)\cos(y)$
PDE3	$\Psi(x,y) = x^2 + y^2 + x + y + 1$
PDE4	$\Psi(x,y) = \sin(xy)$
PDE5	$\Psi(x,y) = \left(x^2+y^2-1\right)\sin x$
PDE6	$\Psi(x,y) = e^{(x-y)} + e^x\cos y$

Table 2: Exact solutions for the test cases.

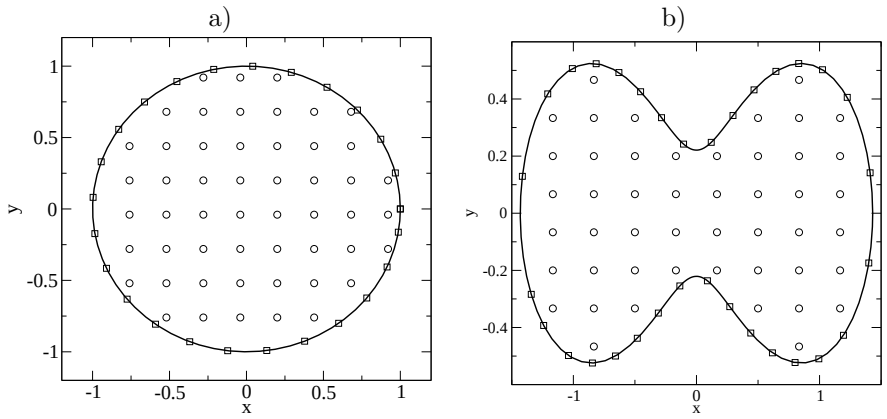

Figure 4: Non-rectangular domains. Unit circle for PDE5 (a), and Cassini's oval for PDE6 (b).

constant k. Second equation has a different fitness landscape so could have a different behaviour during the evolving process.

Therefore we believe that a new metric for measuring the final solution quality obtained is necessary. Concretely, in this work we propose using the *Root of the Mean Squared Error* (RMSE) between the computed final solution $\mathbf{y}$ and the exact solution $\mathbf{y}_{exact}$:

$$RMSE = \sqrt{\frac{\sum_{i=1,\mathbf{x}_i \in C}^{n_C} \left\| \mathbf{y}\left(\mathbf{x}_i\right) - \mathbf{y}_{exact}\left(\mathbf{x}_i\right) \right\|^2 + \sum_{j=1,\mathbf{x}_j \in B}^{n_B} \left\| \mathbf{y}\left(\mathbf{x}_j\right) - \mathbf{y}_{exact}\left(\mathbf{x}_j\right) \right\|^2}{d \cdot \left(n_C + n_B\right)}}. \tag{22}$$

Other authors use the same measure for determine the achieved accuracy [19]. This error does not deal with the differential equation residuals, but measures distances between the computed solution and the exact one. Obviously RMSE can be only used when the exact solution is known and for benchmarking purposes. Because NLODE6 does not have analytical solution, the RMSE for this problem has been computed using a numeric approximation of the solution computed with a Runge-Kutta relaxation method [16] discretizing the domain range in 1000 nodes.

3.3. Results

The method was run 10 times on every differential equation described previously using different seeds for the random number generator and averages were taken. A total of 100 equidistant collocation points have been used in all cases, except for PDE5 and PDE6. In those non-rectangular domain cases, 51 internal points and 25 boundary points have been used for PDE5 (Fig. 4a), and 48 internal and 32 boundary points in PDE6 (Fig. 4b) as in [20]. A maximum harmonic order $N = 10$ has been used, which gives a total of 11 harmonics for

ODEs and NLODEs problems, 22 harmonics for SODEs (systems of two equations each) cases except SODE3 (system of five equations) with 55 harmonics, and 101 harmonics for PDEs.

Table 3 lists the algorithm parameter values which have been used for all test cases and the good results obtained provide evidence about the robustness of the method. However, more improvements could be obtained if the parameters were tuned specifically for each case. In this way, we can differentiate those parameters which are directly sensitive to the solution quality (maximum harmonic order N and range extension ξ) and those which affect to the convergence process (rest of algorithm parameters in Table 3). Regarding the former, the required maximum harmonic order N depends on the variations of the solution function and its derivatives. Generally speaking, more harmonics implies better accuracy in terms of a lower value of RMSE. As it will further be shown, good results have been obtained for $N = 10$ because the solution functions do not have high frequency harmonics. Nevertheless, in some punctual cases higher harmonics are needed. This is the case of ODE3, ODE8, ODE9, ODE10 and NLODE6 problems as it will further be described. The range extension ξ should be higher than 0 when no null first derivatives are desired in the domain boundaries. Good results have been obtained for $\xi = 1$. Low sensitivity has been observed when this value is increased or reduced, except for values close to 0. Concerning the second type the parameters, the initialization values have a high sensitivity in the results. According to Table 3, the best results have been obtained using low values for the initial coefficients, and a similar maximum value for the initial mutation strengths. Regarding the population, as expected, better convergence is achieved increasing its size. A trade-off among performance and computational cost is achieved using a population size of $\mu = 10$. A higher selective pressure than the standard one ($\lambda/\mu \simeq 7$ according to [4]) has been observed more effective. The learning rate factor could be increased in some cases up to 2 or 3 for speed-up the convergence process. According to the definition of the fitness function, Eq. (4), the boundary condition penalty φ should be of the same order than the number of collocation points and the boundary point ratio n_C/n_B. Finally, in simple cases as ODEs, the stop criteria could be relaxed in order to find the solution in less number of generations.

Table 4 shows the result of the proposed method over all the test cases. Average values of the 10 runs are listed for fitness and it can be seen the RMSE values of the best individual in the last generation and the number of generations used. This table also gives the standard deviation of these values, showing low dispersion in the results. As we can appreciate, ODE3 problem results are very different from the rest of test cases. In the next subsection further details will be given. Excluding ODE3 problem, good results have been obtained with RMSE values between 10^{-6} and 10^{-1} for one-dimensional problems, and between 10^{-4} and 10^{-2} for PDEs. As an example, Fig. 5 and 6 show plots related to the evolution of characteristic parameters of the algorithm and the quality of the solution obtained for a run of the NLODE4 case. In particular, Fig. 5a and Fig. 5b show the fitness value and harmonic coefficients of the best individual over the generations. Fig. 6a shows the evolution of the mutation strengths of the

Parameter	Values
Maximum harmonic order N	10
Initial coefficients	Between $\alpha = -10^{-3}$ and $\beta = 10^{-3}$
Initial mutation strength	Between $\gamma = 3 \cdot 10^{-4}$ and $\delta = 3 \cdot 10^{-3}$
Parent selection	$(\mu, \lambda) = (10, 400)$
Learning Rate factor f_τ	1
Boundary Condition Penalty φ	300
Range extension ξ	1
Stop criteria	15 generations with unchanged fitness or 80 generations
Maximum Generations G	3000
Max. new harmonics in each step	8

Table 3: Numerical values for the parameters of the method.

Case	Fitness	RMSE	Generations	$\sigma_{Fitness}$	σ_{RMSE}	$\sigma_{Generations}$
ODE1	$5.94 \cdot 10^{-7}$	$3.70 \cdot 10^{-5}$	2581	$6.30 \cdot 10^{-8}$	$1.66 \cdot 10^{-6}$	476
ODE2	$2.51 \cdot 10^{-6}$	$6.59 \cdot 10^{-5}$	3000	$1.28 \cdot 10^{-7}$	$2.55 \cdot 10^{-6}$	0
ODE3	7.48	13.16	3000	$1.32 \cdot 10^{-2}$	$1.35 \cdot 10^{-2}$	0
ODE4	$4.96 \cdot 10^{-6}$	$1.65 \cdot 10^{-6}$	1113	$3.33 \cdot 10^{-9}$	$8.72 \cdot 10^{-7}$	38
ODE5	$4.31 \cdot 10^{-8}$	$9.90 \cdot 10^{-5}$	819	$3.65 \cdot 10^{-9}$	$3.83 \cdot 10^{-6}$	21
ODE6	$2.91 \cdot 10^{-8}$	$5.44 \cdot 10^{-6}$	801	$1.47 \cdot 10^{-8}$	$3.64 \cdot 10^{-6}$	74
ODE7	$9.34 \cdot 10^{-6}$	$1.25 \cdot 10^{-5}$	3000	$4.65 \cdot 10^{-7}$	$2.64 \cdot 10^{-6}$	0
ODE8	$1.51 \cdot 10^{-3}$	$1.22 \cdot 10^{-2}$	3000	$2.00 \cdot 10^{-5}$	$1.00 \cdot 10^{-4}$	0
ODE9	$9.98 \cdot 10^{-3}$	$3.15 \cdot 10^{-2}$	3000	$2.91 \cdot 10^{-4}$	$5.29 \cdot 10^{-4}$	0
ODE10	$1.46 \cdot 10^{2}$	$3.90 \cdot 10^{-2}$	3000	$2.06 \cdot 10^{1}$	$1.65 \cdot 10^{-3}$	0
NLODE1	$2.26 \cdot 10^{-7}$	$7.42 \cdot 10^{-5}$	1218	$9.63 \cdot 10^{-8}$	$1.81 \cdot 10^{-5}$	419
NLODE2	$7.87 \cdot 10^{-8}$	$5.90 \cdot 10^{-6}$	1349	$7.07 \cdot 10^{-9}$	$5.49 \cdot 10^{-7}$	106
NLODE3	$1.12 \cdot 10^{-5}$	$3.64 \cdot 10^{-5}$	3000	$6.87 \cdot 10^{-7}$	$4.92 \cdot 10^{-6}$	0
NLODE4	$3.67 \cdot 10^{-7}$	$8.32 \cdot 10^{-5}$	2187	$8.14 \cdot 10^{-9}$	$7.91 \cdot 10^{-6}$	49
NLODE5	$3.58 \cdot 10^{-7}$	$3.19 \cdot 10^{-6}$	2755	$1.03 \cdot 10^{-7}$	$5.96 \cdot 10^{-7}$	78
NLODE6	$3.12 \cdot 10^{-2}$	$3.03 \cdot 10^{-1}$	3000	$1.97 \cdot 10^{-4}$	$9.49 \cdot 10^{-4}$	0
SODE1	$2.13 \cdot 10^{-7}$	$7.67 \cdot 10^{-5}$	1117	$2.18 \cdot 10^{-8}$	$1.66 \cdot 10^{-5}$	147
SODE2	$2.43 \cdot 10^{-8}$	$3.90 \cdot 10^{-5}$	2701	$2.43 \cdot 10^{-8}$	$3.77 \cdot 10^{-6}$	166
SODE3	$1.73 \cdot 10^{-7}$	$8.51 \cdot 10^{-5}$	1149	$5.88 \cdot 10^{-8}$	$4.02 \cdot 10^{-5}$	48
SODE4	$1.17 \cdot 10^{-6}$	$4.72 \cdot 10^{-5}$	3000	$9.90 \cdot 10^{-8}$	$2.61 \cdot 10^{-6}$	0
SODE5	$3.58 \cdot 10^{-7}$	$3.19 \cdot 10^{-6}$	2755	$1.03 \cdot 10^{-7}$	$3.18 \cdot 10^{-6}$	78
PDE1	$1.56 \cdot 10^{-2}$	$6.37 \cdot 10^{-3}$	2700	$3.01 \cdot 10^{-3}$	$7.33 \cdot 10^{-4}$	57
PDE2	$7.18 \cdot 10^{-4}$	$1.16 \cdot 10^{-3}$	1956	$1.77 \cdot 10^{-4}$	$2.14 \cdot 10^{-4}$	139
PDE3	$1.70 \cdot 10^{-2}$	$5.90 \cdot 10^{-3}$	2564	$3.63 \cdot 10^{-3}$	$7.99 \cdot 10^{-4}$	156
PDE4	$6.90 \cdot 10^{-4}$	$1.23 \cdot 10^{-3}$	2066	$6.76 \cdot 10^{-5}$	$3.62 \cdot 10^{-5}$	102
PDE5	$9.24 \cdot 10^{-4}$	$9.06 \cdot 10^{-4}$	2599	$2.10 \cdot 10^{-4}$	$1.29 \cdot 10^{-4}$	66
PDE6	$2.22 \cdot 10^{-1}$	$1.79 \cdot 10^{-2}$	2979	$6.24 \cdot 10^{-2}$	$3.84 \cdot 10^{-3}$	42

Table 4: Experimental results.

Case	Max. Harmonic Order	Fitness	RMSE
ODE8	10	$1.51 \cdot 10^{-3}$	$1.22 \cdot 10^{-2}$
	20	$1.35 \cdot 10^{-4}$	$2.84 \cdot 10^{-3}$
	30	$1.84 \cdot 10^{-5}$	$9.24 \cdot 10^{-4}$
ODE9	10	$1.00 \cdot 10^{-2}$	$3.16 \cdot 10^{-2}$
	20	$8.32 \cdot 10^{-4}$	$7.08 \cdot 10^{-3}$
	30	$1.18 \cdot 10^{-4}$	$2.32 \cdot 10^{-3}$
ODE10	10	$1.46 \cdot 10^{2}$	$3.39 \cdot 10^{-2}$
	20	$3.27 \cdot 10^{-1}$	$1.34 \cdot 10^{-3}$
	30	$5.65 \cdot 10^{-3}$	$5.33 \cdot 10^{-4}$
NLODE6	10	$3.13 \cdot 10^{-2}$	$3.03 \cdot 10^{-1}$
	20	$1.80 \cdot 10^{-2}$	$2.17 \cdot 10^{-1}$
	30	$1.25 \cdot 10^{-2}$	$1.73 \cdot 10^{-1}$

Table 5: Harmonic number analysis for ODE8, ODE9, ODE10 and NLODE6 problems.

best individual with the generation number. Fig. 6b compares the exact solution with the computed one. We can appreciate the good matching obtained.

Observing the RMSE for one dimensional problems in table 4, we can see that all the cases have achieved a good accuracy with RMSEs values between 10^{-5} and 10^{-6} except for cases ODE3, ODE8, ODE9, ODE10 and NLODE6. These last cases have a more complicated shape and more than 10 harmonics are needed for approximate the solution. In table 5 the fitness and RMSEs values running the algorithm with the same solver parameters (table 3) but using 10, 20 and 30 harmonics are given. As expected, the accuracy is increased when more harmonics are used. Fig. 7 compares the solutions obtained for NLODE6 case with the numerical approximation using a numerical method. We can appreciate that due to the strong variation of the function near the origin 10 harmonics are not enough for a proper representation of the solution.

3.4. ODE3 discussion

Not as good results have been obtained for ODE3 problem, with a RMSE of several orders of magnitude bigger than the rest ODEs. Studding the differences between this problem and the others, we can see that the variation range of the derivatives appearing in the differential equation is around one order of magnitude higher than in the other cases:

$$\left.\begin{array}{l} \dfrac{y'(1)}{y'(0)} = 4e^3 \simeq 80 \\ \dfrac{y''(1)}{y''(0)} = \dfrac{30}{12}e^3 \simeq 50 \end{array}\right\}. \tag{23}$$

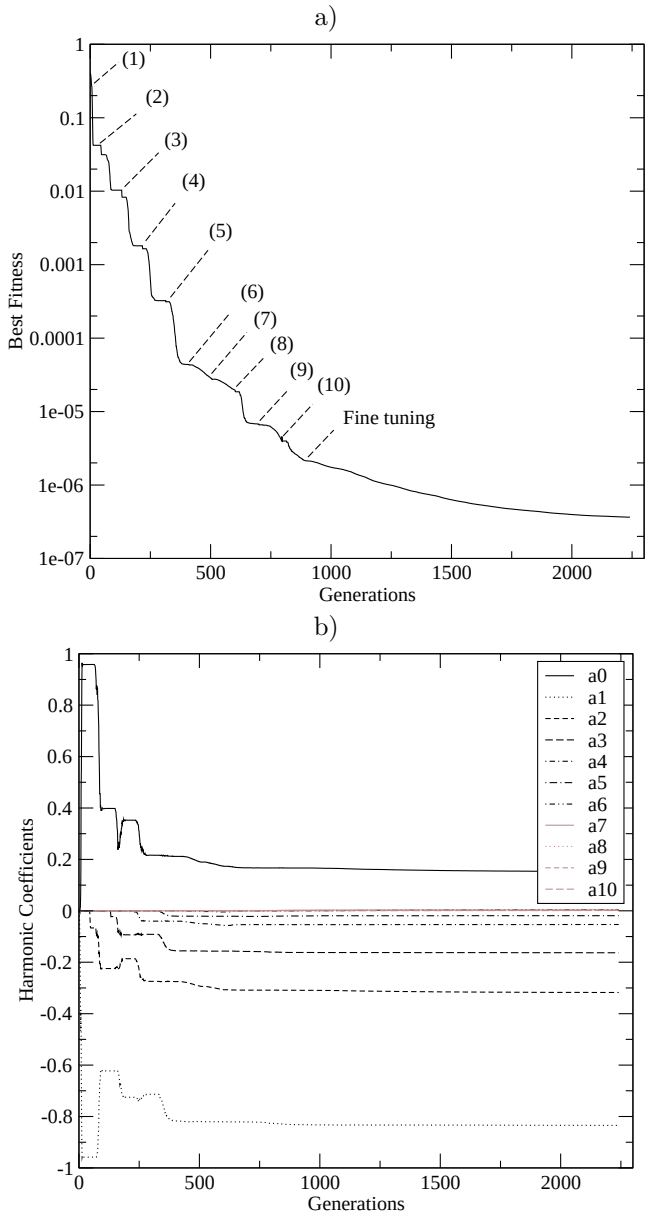

Figure 5: Fitness value (a) and harmonic coefficients (b) of the best individual over the generations for one run of NLODE4 case. In (a) number in parentheses indicate the number of *active* plus *frozen* harmonics. *Fine tuning* step is shown as well.

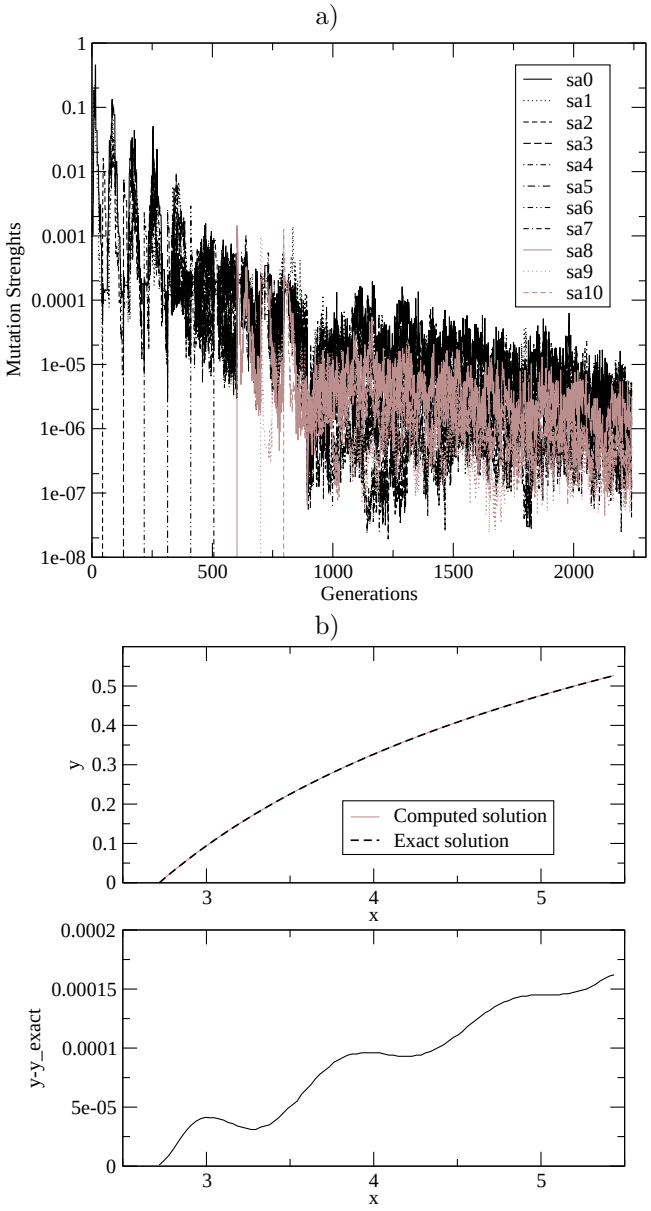

Figure 6: Mutation strengths of the best individual over the generations (a), comparison (b, top) and difference (b, bottom) between the exact solution and the computed one for one run of NLODE4 case. Mutation strength in (a) of *inactive* harmonics are considered out of the plot with a value close to 0.

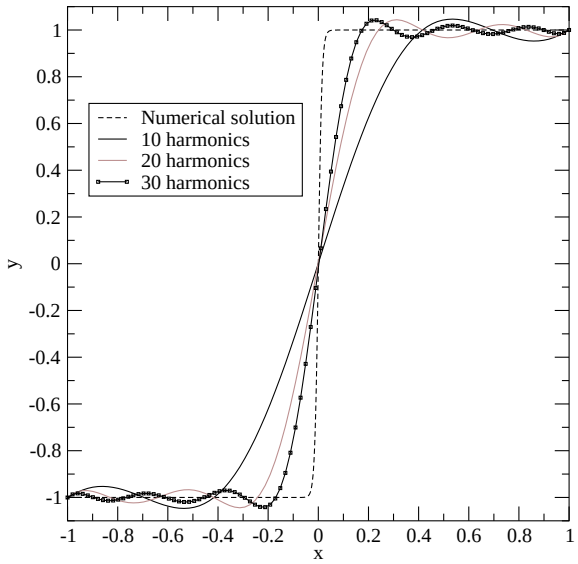

Figure 7: Comparison of the computed solution using 10, 20 and 30 harmonics with the numerical approximation for NLODE6 case.

In order to check if these ratios are the cause of the bad results, the same problem ODE3 has been solved using the same solver parameters (table 3) but reducing the independent variable range from the original $x \in [0, 1]$ to $x \in [0, 0.2]$. In this way, the above ratios turn

$$\left.\begin{array}{l} \dfrac{y'(0.2)}{y'(0)} = \dfrac{3.2}{2} e^{0.6} \simeq 2.9 \\ \dfrac{y''(0.2)}{y''(0)} = \dfrac{15.6}{12} e^{0.6} \simeq 2.4 \end{array}\right\}. \tag{24}$$

Algorithm performance has been recovered for this new problem, obtaining an average fitness of $5.48 \cdot 10^{-4}$ and an average RMSE of $1.02 \cdot 10^{-4}$.

Focusing in the original ODE3 problem, a study on the influence of the number of harmonic coefficients in the result quality has been performed. Several runs with an unlimited number of generations have been done. Results are shown in table 6. As we can see the results improve when the number of harmonic coefficients is increased. More than 20 harmonics are needed for having a RMSE close to the rest of cases. Other important problem detected is the slow convergence rate, needing a higher number of generations.

4. Comparative study

In this section some comparatives with other approaches are described. Two non-linear ODEs are solved using a numerical method and the results are com-

Max. Harmonic Order	Fitness	RMSE	Generations
5	7.60	13.03	20053
10	1.49	4.45	194630
15	$2.40 \cdot 10^{-1}$	2.00	230060
20	$3.93 \cdot 10^{-2}$	$5.83 \cdot 10^{-1}$	380060

Table 6: Harmonic number analysis for original ODE3 problem.

pared with the proposed algorithm. Then, some qualitative comparisons against other Evolutionary approaches reported in the literature are presented.

4.1. Comparative with numerical methods

Maybe the greater advantage of the proposed approach is that it is a generic framework, that is, it does not depend on the type of differential equation. On the other hand, numerical methods are specific for each equation type. For instance, classical Runge-Kutta methods are used for initial value problems due to their higher accuracy features compared with more efficient algorithms as Euler's method. Classical Runge-Kutta methods are explicit, and are unsuitable for stiff systems because of their small region of stability. On the contrary, implicit Runge-Kutta methods have a large region of absolute stability [21]. Boundary value problems requires different algorithms, such as shooting method for one-dimensional problems or the finite element method for more general domains. Even for the same equation, in some problems depending on the boundary conditions the numerical scheme must be changed due to stability reasons [25]. Furthermore, the implementation of a new numerical method could turn difficult because it is necessary to take into account several issues as the discretization order, the algorithm stability, the convergence speed, how to fulfill the boundary conditions, etc. In the method described in this work, the original problem is transformed into an optimization one according to Eq. (4), so the problem of choosing the most appropriate numeric method disappears.

For making a quantitative comparison with a numerical method, two non-linear ordinary differential equations from the test case suite (table 1) have been solved with traditional numerical methods. The first problem, NLODE2 is a first order non-liner differential equation. The domain interval $x \in [1, 2]$ is discretized into N equidistant nodes. This system can be solved in a very efficient way using a fourth order Runge-Kutta (RK4) algorithm [16]. Once the solution is obtained, the RMSE is computed according to expression (22). The RMSE can be computed on different grids using a liner interpolation obtaining the values of the dependant variable y_i on each grid node x_i. In table 7 the results obtained for two different grid sizes (10^2 and 10^3 nodes) are compared with the evolutionary approach. The RMSE are computed on three different grid sizes (10^2, 10^3 and 10^4 nodes).

Method	**Grid size**	$RMSE_{10^2}$	$RMSE_{10^3}$	$RMSE_{10^4}$
RK4	10^2	$2.23 \cdot 10^{-12}$	$8.91 \cdot 10^{-6}$	$8.91 \cdot 10^{-6}$
RK4	10^3	$8.71 \cdot 10^{-8}$	$3.80 \cdot 10^{-15}$	$8.75 \cdot 10^{-8}$
Evolutionary	10^2	$5.98 \cdot 10^{-6}$	$5.99 \cdot 10^{-6}$	$5.99 \cdot 10^{-6}$

Table 7: Comparison of the numerical method solution with the Evolutionary approach for NODE2 case. Two grid sizes of 10^2 and 10^3 have been used. The RMSE are computed on three grid sizes using a linear interpolation for the numerical solution and the equation (7) for the evolutionary one.

We can see that the RMSE of evolutionary solution is not dependant on the grid size. The RK4 solutions have a high accuracy in the grid used in the computation, but this accuracy decreases when other grid is used due to the linear interpolation. The RK4 algorithm with the same grid size as the evolutionary solution has a less accuracy in the finer grids (10^3 and 10^4 nodes). As expected, if the number of nodes is increased, the accuracy achieved is higher and better than the evolutionary approach. It is worth to point out that the solution representation of the evolutionary approach only needs to store 11 numbers (the harmonic coefficients), whereas the RK4 solution requires to store the values of x and y in all the grid nodes, i. e. the RK4 solution in the finer grid is around 200 times bigger than the evolutionary solution.

For the next comparative NLODE5 has been selected. It is as well a non linear differential equation, but the boundary conditions are given in different points. This type of cases is called two point boundary problems. The crucial distinction between initial value problems and two point boundary value problems is that in the former case we are able to start an acceptable solution at its beginning, while in the present case, the boundary conditions at the starting point do not determine a unique solution to start with. For this reason, two point boundary value problems require considerably more effort to solve than do initial value problems. The *shooting method* [16] has been used to solve this problem. In this method the original problem is transformed into a root finding problem. We choose values for all of the dependent variables at one boundary. We then integrate the ODE by initial value methods, arriving at the other boundary. We find discrepancies from the desired boundary values there, so the initial boundary condition is readjusted. The iteration process is stopped when no improve is detected in the error of the boundary condition at the left side of the domain interval. Each iteration is solved using a RK4 algorithm transforming the initial two order equation into the following equivalent first order system:

$$\left.\begin{array}{c} y' = z \\ z' = \frac{yz}{x \sin x^2} - 4x^2 \sin x^2 \end{array}\right\}, \tag{25}$$

Table 8 shows a comparative of the solutions obtained using two grid sizes. As before, the RMSE values are computed in three different grids using a linear interpolation in the numerical solutions, and equation (7) for the evolutionary one.

We see that as in the previous case a finer grid is needed for achieving

Method	**Grid size**	$RMSE_{10^2}$	$RMSE_{10^3}$	$RMSE_{10^4}$
Shooting	10^2	$3.92 \cdot 10^{-7}$	$5.67 \cdot 10^{-5}$	$5.68 \cdot 10^{-5}$
Shooting	10^3	$5.54 \cdot 10^{-7}$	$3.89 \cdot 10^{-10}$	$5.60 \cdot 10^{-7}$
Evolutionary	10^2	$3.12 \cdot 10^{-6}$	$3.13 \cdot 10^{-6}$	$3.14 \cdot 10^{-6}$

Table 8: Comparison numerical method solution with the Evolutionary approach for NODE5 case. Two grid sizes of 10^2 and 10^3 have been used in the numerical method. The RMSE is obtained using a linear interpolation for the numerical solution and the equation (7) for the evolutionary one in three different grids of 10^2, 10^3 and 10^4 nodes.

a better RMSE value than in the evolutionary approach. When the RMSE is computed in a different grid than the one used in the *Shooting* algorithm, the errors increase. On the other hand, the RMSE values of the Evolutionary solutions are not affected by the grid size. Note that the numerical integration is performed using a fourth order Runge-Kutta algorithm, which is a high order method, i. e. the errors depends on the grid size rise to the power of four. The memory requirements of the *Shooting* solution using 10^3 nodes is around 200 times bigger than the evolutionary solution (arrays of x and y values must be stored, meanwhile only the harmonic coefficients must be kept in evolutionary solution).

With these two examples, we can say that in some cases the evolutionary approach can achieve a more accurate solution using less number of nodes. The solutions obtained are coded in a more compact way requiring significantly less amount of memory. Nevertheless, a major drawback is the CPU time consuming. In this comparison, the Evolutionary approach consumes around 5000 more time than the numerical approach. This number could be decreased in other problems where efficient numerical methods as RK4 and *Shooting* can not be applied, as for instance in PDEs.

4.2. Comparative with other Evolutionary Computing approaches

It is difficult to make a quantitative comparative study with other reported approaches. Although the same problems described in [22] have been used with the same collocation points, the comparative is not straightforward because, as it was already commented in subsection 3.2, the fitness values are high dependent on the solver parameters and on how differential equations are provided to expression (4). A correct comparison of the solution quality should be done with the RMSE values, but these quantities are not reported in previous contributions. However, Tsoulos et al. [22] obtained an average fitness values between 10^{-5} and 10^{-9}, meanwhile in the present work the fitness values are in the range of 10^{-2} and 10^{-8}. Therefore it seems that neural networks can approximate the solution functions with better accuracy. Moreover, no problems have been reported for ODE3 case in [22, 23]. It is important to notice that in the present contribution all the test cases have been run with the same solver parameters in order to present a systematic method. However, better results could be obtained if these solver parameters were adjusted by trial and error for each problem.

In contribution [19] 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. The RMSE obtained is around 10^{-4}, which is better than those obtained in the present paper. Nevertheless, the number of unknowns that must be tuned in [19] is much higher than in the present work. A total of 132 neurons are needed, so considering the weights and bias, this implies a number of unknowns around 260 against 100 of the present contribution. In a similar way, good accuracy has been obtained for ODE9 using 30 harmonics, whereas in [3] more than 70 neurons are needed for obtaining a similar solution.

It can be remarked that in some other previous contributions [10, 22, 23] PDE's results are of the same quality than ODEs. In the present work, the PDE's RMSE is around two orders of magnitude bigger compared with ODEs and SODEs. This could be explained noting that the number of unknown coefficients needed by a neural network for solving a PDE problem scale linearly with the dependent variable dimension. On the other hand, using an harmonic approach, the number of harmonics increases quadratically for 2D problems, as it was shown in expression (21).

Because local search is used in other approaches [10, 15, 22], but not in the present contribution, it is difficult as well to make a quantitative comparison of the required computational power. It should be compared not the generation number, but the number of fitness evaluations. But these quantities have not been reported in the approaches mentioned. Furthermore, in our method, the fitness evaluation number could be reduced taking into account that is cheaper to evaluate the fitness function when the number of *active* harmonics is lower in the first *ES steps* than in the last *ES steps*.

From a qualitative point of view, some advantages of the current approach can be enumerated. Firstly, in this contribution is straightforward to compute the derivatives because all the solutions are only expressed as sum of cosine functions. In works based on GP [1, 20, 23], an automatic differentiation engine must be used. In neural networks approaches [10, 19, 22, 15, 3], the activation function must be differentiated, which could be hard depending on the chosen function. Furthermore, if more than one hidden layer is used, the symbolic derivatives implementation could turn very complex.

Secondly, in this work, several steps of a classical ES are used, guiding the search process in a more efficient way. Other approaches [10, 15, 22] use local search, which makes the implementation and analyis more difficult. Nevertheless, the proposed approach can deal naturally with local search phases, but we have wanted to test the skills of our method in the simplest way possible. In any case, the addition of this kind of search could accelerate the convergence velocity.

Thirdly, 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, the proposed approach can be applied to any kind of differential equations. Some authors [10, 20, 24] use some particular methods for dealing with the boundary conditions, facilitating the optimization process eliminating

the constraints. Nevertheless, these methods are problem dependant and can not be applied to all problems. Our proposed method do not assumed any particular structures in the boundary conditions. That is, it is straightforward to assign a fitness value to each individual in the population transforming the original problem into an optimization one according to Eq. (4), even in those problem with complex geometries and boundary conditions, such as PDE6 (see Table 1) where the definition domain is a region different from a classical 2-dimensional interval.

5. Conclusions

A novel mesh-free approach for solving differential equations based on ESs has been presented. Unlike numerical methods, the proposed algorithm is general and does not assume any structure of the differential equations. Therefore the approach is suitable both for linear and nonlinear ODEs, SODEs and PDEs. Candidate solutions are built using Fourier series. The periodic expansion of the solutions is done in such a way that all the sine coefficients are vanished and the constraint of null first derivative at the borders of the original range is avoided. This method allows computing symbolically all the needed derivatives, so no mesh connectivity is needed. Only a set of collocation points must be provided. Taking advantage on the decreasing absolute value of the harmonic coefficients with the harmonic order, several ES steps are performed. The harmonic coefficients are taking into account one by one starting with the lower order ones. In this way the search space dimension is reduced making the search process more systematic and easier.

The proposed method has been tested in a set of 26 different problems extracted from the literature. All the test cases have been successfully solved using the same set of algorithm parameters. Assuming that the exact solution is known for the test cases, the RMSE is used for comparing the quality of the results. The achieved RMSE is around 10^{-5} for the majority of ODEs and SODEs and around 10^{-3} for PDEs with rectangular boundaries. Lower convergence has been achieved in PDE6 case, with a RMSE of 10^{-2}. It has been shown that depending on the solution's shape, some cases need more than 10 harmonics for a correct representation.

A quantitative analysis was performed solving two cases with a numerical method. This analysis has shown that the evolutionary algorithm could outperform the numerical approach in terms of accuracy, compactness and storage requirements of the solution because the evolutionary approach produces a mathematical function, whereas numerical methods give arrays of values on the grid nodes. In addition, in the proposed algorithm like other mesh-free approaches, the connectivity in the mesh is not necessary, making the preprocessing steps straightforward compared with a numerical approach.

The results obtained are encouraging but several improvements can be done in cases where a high number of harmonics is needed for building candidate solutions. In this type of problems, efficiency of the algorithm could be improved using a parallel implementation, which is relatively straightforward because of the

intrinsic parallel nature of the evolutionary algorithms. Several sub-populations could be computed in different processor units. Another way of improving the algorithm performance is using some kind of local search in the evolutionary algorithm. This can be done in an easy way without modifying the ESs phases.

The proposed approach can naturally deal with differential equation problems defined on non-rectangular boundaries as it has been shown in PDE5 and PDE6 cases. Such problems are very interesting and arise in many real engineering applications. Nevertheless the behaviour of the Fourier expansions and the ES dynamics must be investigated in more complex geometries. Another important direction of research could be the application of the proposed method to another kind of differential equations like PDEs with initial value problems.

References

[1] 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.

[2] H. G. Beyer and H. P. Schwefel. Evolution strategies - a comprehensive introduction. *Natural Computing*, 1:3–52, 2002.

[3] 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.

[4] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag Berlin Heidelberg, 2 edition, 2007.

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

[6] S. J. Farlow. *Partial Differential Equations for Scientists and Engineers*. Dover Publications, Inc., 1993.

[7] B. L. Ghodadra. Order of magnitude of multiple fourier coefficients of functions of bounded p-variation. *Acta Mathematica Hungarica*, 22(3):187–198, 2010.

[8] D. Howard and S. C. Roberts. Genetic programming solution of the convection-diffusion equation. *Proceedings of Genetic Evolutionary Computation Conference (GECCO-2001)*, pages 34–41, 2001.

[9] 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.

[10] 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.

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

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

[13] M. O'Neill and C. Ryan. Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5:349–358, 2001.

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

[15] 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.

[16] 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.

[17] K. R. Rao and P. Yip. *Discrete Cosine Transform. Algorithms, Advantages and Aplications*. Academic Press, Inc, 1990.

[18] T. Seaton, G. Brown, and J. F. Miller. Analytic solutions to differential equations under graph-based genetic programming. *EuroGP LNCS*, 6021:232–243, 2010.

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

[20] 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.

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

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

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

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

[25] Xiaolu Zhao. Stream function solution of transonic flow along s2 stream-surface of axial turbomachines. *Journal of Engineering for Gas Turbines and Power*, 108(1):138–143, 1986.