Engineering Applications of Artificial Intelligence 97 (2021) 104086

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Optimizing functionals using Differential Evolution

K.B. Cantún-Avila [a], D. González-Sánchez [b], S. Díaz-Infante [b], F. Peñuñuri [a,*]

[a] Facultad de Ingeniería, Universidad Autónoma de Yucatán, A.P. 150, Cordemex, Mérida, Yucatán, Mexico
[b] Departamento de Matemáticas, CONACYT-Universidad de Sonora, C.P 83000, Hermosillo, Sonora, Mexico

ARTICLE INFO

Keywords:
Functional optimization
Differential Evolution
Large-scale global optimization
Automatic differentiation
Cubic spline interpolation

ABSTRACT

Metaheuristic algorithms are typically used for optimizing a function $f : A \to \mathbb{R}$, where A is a subset of $\mathbb{R}^N$. Nevertheless, many real-life problems require A to be a set of functions which makes f a functional. In this paper, we present a methodology to address the optimization of functionals by using the evolutionary algorithm known as Differential Evolution. Unlike traditional techniques where continuity and differentiability assumptions are required to solve some associated differential equations—like calculus of variations, Pontryagin's principle or dynamic programming, the optimization is carried out directly on the functional without the need of any of the assumptions mentioned before. Lagrangians involving derivatives are considered, these derivatives are computed implementing Automatic Differentiation with dual numbers. To the best of our knowledge, this is the first time that a metaheuristic optimization approach has been applied to directly optimize a broad variety of functionals. The effectiveness of our methodology is validated by solving two problems. The first problem is related to the implementation of quarantine and isolation in SARS epidemics and the second validation problem deals with the well-known brachistochrone curve problem. The results of both validation problems are in outstanding agreement with those obtained with the application of traditional techniques, specifically with the Forward–Backward-Sweep method in the first problem, and with the calculus of variations for the latter problem. We also found that interpolation may be employed to solve the large scale global optimization problems arisen in the optimization of functionals.

1. Introduction

Metaheuristic methods (Abdel-Basset et al., 2018) have been employed to solve many optimization problems arising in science and engineering. Recent studies have used common variants of metaheuristic algorithms to study a wide range of subjects. The Genetic Algorithm is used to achieve facial recognition as described in Zhi and Liu (2019), while in Biswas et al. (2019) the Differential evolution algorithm is used to estimate the parameters describing a photovoltaic solar cell, and in Altan et al. (2019) the cuckoo search algorithm is used as the optimization algorithm in a model for digital currency forecasting. Other examples include the use of the Particle Swarm Optimization algorithm to model the nonlinear relationship between CO_2 emissions and economic growth in Wang and Li (2019). The examination timetabling problem was addressed using the simulated annealing algorithm in Leite et al. (2019) and, in Peng et al. (2019), the ant colony algorithm is used to identify ecological security patterns in the city of Beijing. In more recent studies, the partitioning problem in hardware/software co-design was address implementing the tabu search algorithm (Neng et al., 2020), the generation cost and fuel emission in a hydrothermal scheduling problem was minimized using the sine cosine algorithm (Dasgupta et al., 2020), and the packing density for cementitious pastes was predicted using the gravitational search algorithm (Banyhussan et al., 2020).

Metaheuristic methods are devised for optimizing a function $f_{ob} : A \to \mathbb{R}$, where the domain A is a continuum subset of $\mathbb{R}^N$ (if the domain is a discrete set, the floor or ceiling functions can be used to map from the continuum case to the discrete one). However, many real-life problems require the set A to be a space of functions, thus f_{ob} becomes a functional (see Eq. (1)). Despite the large amount of work on metaheuristic optimization, the direct application of these algorithms to the optimization of functionals has not been properly studied. Although the bibliography is scarce, we can mention (Yan and Zou, 2008) where a study about suboptimal control is conducted. However, the dimensions of the individuals to evolve are too small and the methodology is vaguely presented. Moreover, to the best of our knowledge, the optimization of a functional whose Lagrangian involves derivatives—as in Eq. (1)—has not been carried out for any metaheuristic algorithm in the literature. In this study, we present a methodology that allows the optimization of functionals using the Differential Evolution algorithm as optimization method and Automatic

* Corresponding author.
E-mail addresses: karla.cantun@correo.uady.mx (K.B. Cantún-Avila), dgonzalezsa@conacyt.mx (D. González-Sánchez), saul.diazinfante@unison.mx (S. Díaz-Infante), francisco.pa@uady.mx (F. Peñuñuri).

https://doi.org/10.1016/j.engappai.2020.104086
Received 30 May 2020; Received in revised form 2 November 2020; Accepted 4 November 2020
Available online 13 November 2020
0952-1976/

Differentiation for computing derivatives (Griewank, 1989; Neidinger, 2010; Yu and Blair, 2013). The proposed methodology would allow to study many optimal control problems subject to certain dynamics, as in Eq. (2). These problems are of current relevance as most of the epidemic control problems fall into this category.

The optimization of functionals has been widely studied with different theoretical approaches. Calculus of variations (van Brunt, 2004), Pontryagin's maximum principle (Pontryagin et al., 1962), and dynamic programming (Fleming and Soner, 2006) are the most commonly used techniques. In general terms, calculus of variations requires to solve a set of ordinary differential equations. The Pontryagin's principle requires, in addition to solving a system of differential equations, satisfying an optimization condition and some sufficient conditions (Pontryagin et al., 1962, p. 19). Finally, the dynamic programming approach consists in solving a partial differential equation, the Hamilton–Jacobi–Bellman equation, with a process that also involves an optimization condition (Fleming and Soner, 2006, p. 12). Contrasting to these traditional approaches, when metaheuristic algorithms are used to directly optimize functionals, the aforementioned conditions are no longer needed. Therefore, the kind of functionals that could be optimized is broader and the considerable amount of mathematical knowledge and specialization used to obtain the requirements for the traditional approaches, can be spared. As a result, an interesting field of application could open up for many researchers working on global metaheuristic optimization.

Among metaheuristic methods for global optimization, evolutionary algorithms (Metaxiotis and Liagkouras, 2012; Vikhar, 2016) have a good reputation. Such algorithms emulate Darwinian natural evolution, essentially by introducing operators for mutation, crossover and selection. One of the earliest studies using evolutionary methods is the one by Box (1957), which in 1957 presented a study using evolutionary operations as a method for increasing industrial productivity. In the following years, further studies applying the evolutionary ideas appeared (Friedberg, 1958; Friedberg et al., 1959; Bremermann, 1962). A more detailed and comprehensive summary of the history of evolutionary computation can be found in Jong et al. (1997). Nevertheless, it can be said that the first evolutionary algorithm, as they are currently known, was the so called Genetic Algorithm. The algorithm was developed by Holland (1975), who also introduced the widely recognized schema theorem as a mathematical justification to this algorithm. Since then, many variations of evolutionary algorithms have appeared, being Differential Evolution (DE), introduced by Storn and Price (1997), one of the simpler, more efficient and effective, and more widely used optimization algorithm (Bilal et al., 2020). There is a large amount of published studies and review papers on DE in several areas of engineering. For instance, a detailed survey of the method can be found in Neri (2010) and a review of major application areas of DE is presented in Plagianakos et al. (2008). On more recent articles, an extensive survey of the DE method (Bilal et al., 2020) and a review on large-scale global optimization (LSGO) problems (Maučec and Brest, 2019) are provided.

When optimizing a functional using DE, a large dimension of the individuals to evolve could be necessary. Therefore, we need to solve an LSGO problem. Some variants of DE have proved to be effective to solve LSGO problems (Maučec and Brest, 2019). Unfortunately, the original – or classical – version of DE may have slow convergence and stagnation when applied to an LSGO problem (Maučec and Brest, 2019). However, we demonstrate that an interpolation, as presented in 2.1, yields a successful optimization of the originated LSGO problem using the DE method.

The remaining of the paper is organized as follows. In Section 2, we present a direct methodology for addressing the optimization of functionals. After briefly reviewing the classical DE method, we introduce the way in which a functional can be optimized and how dual numbers can be used for computing derivatives. This section also explains how an interpolation can be used to solve the LSGO problem arising in this study. An important problem related to quarantine and isolation in SARS epidemics is presented in Section 3. Section 4 presents the classical problem of the brachistochrone. This case is of interest because the functional to optimize involves derivatives. It is extremely difficult for the classical DE method to optimize this problem. In fact, a direct application of this method without the aforementioned interpolation approach, fails to achieve convergence. Finally, the conclusions of this study are presented in Section 5.

2. Methodology for optimizing functionals using DE

2.1. The differential evolution algorithm

The DE algorithm is a simple and effective evolutionary optimization method which has been applied to many real-life problems giving excellent results. The original version of DE can be implemented by defining 5 operators (Peñuñuri et al., 2016): the initial population generation operator $\mathbf{X_0}$, the mutation operator $\mathbf{M}$, the crossover operator $\mathbf{C}$, the selection operator $\mathbf{S}$, and the operator $\mathbf{Best}$, which extracts the best individual after the evolutionary process. The implicated variables in the optimization problem are: the number of individuals of the population N_p, the search space $\mathcal{V}$, the number of generations which would determine the stopping criterion, the crossover probability C_r, the mutation scale factor F, and the objective function to optimize f_{ob}. The method can be coded following Algorithm 1, in which the initial population is stored in the matrix X. Subsequently, the population X is mutated and stored in matrix M, the crossover between X and M is stored in matrix C, and the selection operator produces the new population, which is reassigned to X. After the evolutionary process, the best individual is stored in vector $\mathbf{x}_{best}$. The explicit form for the DE operators can be found elsewhere in the literature, particularly in Price et al. (2005). In the next section, it is explained how this algorithm can be used to optimize functionals.

Algorithm 1 Differential Evolution

$X \leftarrow \mathbf{X}_0(Np, \mathcal{V})$
while (the stopping criterion has not been met) **do**
  $M \leftarrow \mathbf{M}(X, F, \mathcal{V})$
  $C \leftarrow \mathbf{C}(X, M, C_r)$
  $X \leftarrow \mathbf{S}(X, C, f_{ob})$
end while
$\mathbf{x}_{best} \leftarrow \mathbf{Best}(X, f_{ob})$

2.2. Optimization of a functional by DE

The proposed problem is to optimize an expression of the form

$$\mathcal{J}(q(t)) = \int_{t_0}^{t_f} \mathcal{L}(q(t), \dot{q}(t), f(x(t)), t)\, dt, \tag{1}$$

where $q(t) = (q_1(t), q_2(t), \ldots, q_n(t))$ for some integer n, $f(x)$ is some function $f : \mathbb{R}^N \to \mathbb{R}$, and t the independent variable. Following the nomenclature used in mechanics, we will call $\mathcal{L}$ the Lagrangian. In some applications, the Lagrangian $\mathcal{L}$ will not necessarily depend on all the variables. For instance, in a mechanical system $\mathcal{L} = \mathcal{L}(q, \dot{q}, t)$ and, for a conservative system, $\mathcal{L} = \mathcal{L}(q, \dot{q})$. This problem is well known in the area of calculus of variations (van Brunt, 2004). On the other hand, for a typical optimal control problem, the Lagrangian is of the form $\mathcal{L}(q, f, t)$, where q is the control variable, x is the state variable and t the independent variable. These variables satisfy the dynamics

$$\dot{x} = F(q, x, t). \tag{2}$$

In our formulation, the Lagrangian for optimal control problems takes the form $\mathcal{L}(q, f, t)$, where q is the control and f is any real-valued transformation of the state x. For instance, the problem we address in Section 3 considers the dynamics (14) and a linear transformation f.

Regarding the DE terminology, $\mathcal{J}(q(t))$ is the objective function. However, it depends on another functions, i.e., it is a functional. Therefore, we are looking for a function which makes the functional (1) optimal, rather than a point which makes a function optimal. However, it is not clear how the DE algorithm can be applied to this case, since the DE algorithm is designed for optimizing an objective function $f_{ob} : \mathbb{R}^N \to \mathbb{R}$.

From a numerical point of view, a single real-valued function $q_j(t)$ can be seen as a set of points $(t_i, q_j(t_i))$, $i = 1, \ldots, m$. Thus, the numbers $q_j(t_i)$ can be stored in a vector which would be the individual subject to the evolutionary process. Consequently, we can take the functions $q_j(t)$ as piecewise functions given by

$$q_j(t) = \begin{cases} u_j^i & \text{if } t \in I_i \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where the interval $I = [t_0, t_f]$ has been discretized in p disjoint intervals I_i and with $p+1 \geq m \geq 3$. From a programming point of view, it would be convenient, for the DE method, to define the q_j functions having an extra parameter $\mathbf{r}$, as seen in Algorithm 2. For instance, if the domain $[t_0, t_f]$ is divided in 2 disjoint intervals, $q_j(t_0 \leqslant t < t_1, [u_j^1, u_j^2, u_j^3]) = u_j^1$, $q_j(t_1 \leqslant t < t_2, [u_j^1, u_j^2, u_j^3]) = u_j^2$, $q_j(t = t_f, [u_j^1, u_j^2, u_j^3]) = u_j^3$. Therefore, the complete individual subject to the evolutionary process would be

$$\mathbf{x} = [u_1^1, u_1^2 \ldots, u_1^m; u_2^1, u_2^2 \ldots, u_2^m; \ldots; u_n^1, u_n^2 \ldots, u_n^m]. \tag{4}$$

Algorithm 2 q_j function

Parameters: t_0, t_f
Input: $t, \mathbf{r}$
Output: fr

function $q_j(t, \mathbf{r})$
  $eps \leftarrow$ a small quantity epsilon
  $m \leftarrow$ length of $\mathbf{r}$
  $dt \leftarrow (t_f - t_0)/m$
  $ith \leftarrow floor(t/dt) + 1$
  $if(abs(t - tfi) \leq eps)\ ith \leftarrow m$
  $fr \leftarrow \mathbf{r}(ith)$, the component ith of $\mathbf{r}$
end function

Consequently, the DE method can be applied as usual for the case in which the Lagrangian $\mathcal{L}$ does not depend on the derivatives $\dot{q}_j$. Nevertheless, an acceptable answer for the functions q_j may require a large dimension for $\mathbf{x}$—the product mn. As a result, the problem to be considered is an LSGO problem. As mentioned before, the DE method could fail in optimizing this kind of problems. In fact, the problem addressed in Section 4 exemplifies this issue when the DE method fails to achieve convergence for dimensions of order 15 or greater (for a population of 300 individuals and 40 000 generations). Nonetheless, the successful optimization is achieved by making an interpolation of n_i points between the m points of the q_j function. Notice that this interpolation does not change the dimensionality of the problem since the individual to evolve still has mn components. However, the objective function has the extra information coming from the n_i interpolated points between each $q_j(t_k)$ and $q_j(t_{k+1})$. Once the optimization has been conducted for small dimensions, we can construct as many points as desired by interpolation. Observe that an interpolation approach is plausible since we are looking for a function which, in principle, is a continuous curve (or, at most, having discontinuities of measure 0). Moreover, the effectiveness of this approach is demonstrated experimentally in Section 4.

In the general case $\mathcal{L}(q, \dot{q}, f, t)$, the derivative $\dot{q}$ introduces a considerable challenge and perhaps this is one of the reasons why the classical DE method fails to achieve convergence even for small dimensions. In contrast, the classical DE method is successful in optimizing the example presented in Section 3, where individuals of 722 dimensions were used.

2.3. Automatic differentiation through dual numbers

Automatic Differentiation (AD) (Griewank, 1989; Neidinger, 2010; Yu and Blair, 2013) is an algorithmic way of implementing the chain rule for computing derivatives. It can be implemented by using dual numbers (Fike and Alonso, 2011; Yu and Blair, 2013; Peón-Escalante et al., 2020). Below, we briefly review the essential ideas of this AD implementation.

A dual number $\hat{r}$ is a number of the form

$$\hat{r} = a + \epsilon\, b, \tag{5}$$

where a and b are real numbers and $\epsilon^2 = 0$. Such numbers were introduced by Clifford in the late nineteenth century (Clifford, 1873). Dual numbers are not a field (a pure dual number has no inverse). However, they are a commutative ring whose algebra allows the computation of derivatives.

For an analytical function $g(x)$ we can write

$$g(x + h) = g(x) + g'(x)h + 1/2\, g''(x)h^2 + O(h^3), \tag{6}$$

and thus evaluating in the dual number $\hat{x} = x + \epsilon$ we obtain

$$g(x + \epsilon) = g(x) + g'(x)\epsilon, \tag{7}$$

where the other terms are zero in virtue of $\epsilon^2 = 0$. Notice that (7) is a dual number with the real part $g(x)$ and dual part $g'(x)$. The generalization for computing second order derivatives is made by choosing a number of the form

$$\tilde{r} = a + b\,\epsilon_1 + c\,\epsilon_2, \tag{8}$$

with the multiplication table

	1	ϵ_1	ϵ_2
1	1	ϵ_1	ϵ_2
ϵ_1	ϵ_1	$2\epsilon_2$	0
ϵ_2	ϵ_2	0	0

. (9)

Again, evaluating Eq. (6) in $\tilde{r} = x + \epsilon_1$ and according to Eq. (9), we obtain

$$g(x + \epsilon) = g(x) + g'(x)\epsilon_1 + g''(x)\epsilon_2. \tag{10}$$

As can be observed, the real part in (10) is $g(x)$, the ϵ_1-part is $g'(x)$, and the ϵ_2-part is $g''(x)$. Taking the normal cubic spline interpolation method (Bartels et al., 1987) in the context of dual numbers and applying it to points given by the function $q_j(t)$, we are able to estimate $q_j(t)$, $\dot{q}_j(t)$ and $\ddot{q}_j(t)$. Although the interpolation method will give $\ddot{q}_j(t_0) = \ddot{q}_j(t_f) = 0$, this is not a limitation, since the Lagrangian is of the form $\mathcal{L}(q_j, \dot{q}_j, f, t)$ which does not depend on $\ddot{q}_j$.

As additional material to this article we provide a Fortran implementation of the dual numbers as well as the interpolation method in its dual version. See Appendix for more details. Our methodology can be applied, in general, for

$$\mathcal{J}(q_j) = \mathbf{O}_{t_0}^{t_f}(\mathcal{L}(q_j, \dot{q}_j, f, t)), \tag{11}$$

where $\mathbf{O}_{t_0}^{t_f}$ is any operator acting on the Lagrangian function. However, we have decided to exemplify with $\mathbf{O}_{t_0}^{t_f} = \int_{t_0}^{t_f}$.

3. Quarantine and isolation for the SARS epidemics

To illustrate our methodology, we consider the optimal control model reported in Yan and Zou (2008). The general scope is presented as follows.

When a new virus emerges, it is very likely that there is no effective vaccine or medical treatment to control an outbreak of the disease. Therefore, the most effective, and sometimes the only control strategies available to mitigate its spread, are isolation or quarantine. The 2003

Table 1
Description and value of parameters for the SARS model Eq. (14).

Parameters	Values	Description
β	0.2	Transmission coefficient.
ε_E, ε_Q, ε_J	0.3, 0, 0.1	Modification parameter for exposed, quarantine and isolation classes.
μ	0.000034	Natural death rate.
Λ	408.09	Constant recruitment rate.
p	0	Net inflow of asymptomatic individuals.
k_1	0.1	Transfer rate from class of asymptomatic to symptomatic.
k_2	0.125	Transfer rate from the quarantine class to isolation.
d_1, d_2	0.0079, 0.0068	Per-capita disease induced death rates for the symptomatic individuals and isolated individuals.
σ_1, σ_2	0.0337, 0.0386	Per-capita recovery rates for the symptomatic individuals and isolated individuals.
t_f	360 days	Final time.
B_1, B_2, B_3, B_4	1	Cost for E, Q, I, J classes, respectively.
C_1, C_2	300, 600	Costs for Isolation and Quarantine policies.
u_{1L}, u_{2L}	0.05	Lower bound for each control.
u_{1U}, u_{2U}	0.5	Upper bound for each control.

severe acute respiratory syndrome (SARS) breakout in China (Gumel et al., 2004) is a relevant example of this situation.

Yan and Zou reported in Yan and Zou (2008) an optimally controlled version of Gumel et al. (2004). They obtain a model where isolation u_1 and quarantine u_2 mitigation strategies are optimal controls. It this model, the population of size N is split into six classes, namely, susceptible S, asymptomatic E, quarantined Q, symptomatic I, isolated J, and recovered R individuals. Thus, according to the cost functional

$$\mathcal{J}\left(u_1,u_2\right)=\int_0^{t_f}\left[B_1E(t)+B_2Q(t)+B_3I(t)+B_4J(t)+\frac{C_1}{2}u_1^2(t)+\frac{C_2}{2}u_2^2(t)\right]dt, \tag{12}$$

and the space of admissible functions

$$\mathcal{U}:=\{(u_1,u_2):u_i:[0,t_f]\to[u_{iL},u_{iU}],\ \text{piecewise continuous}\}, \tag{13}$$

the corresponding optimal control problem (OCP) reads

$$\min_{u\in\mathcal{U}}\mathcal{J}(u_1,u_2)$$

subject to the dynamics

$$\begin{aligned}
\dot{S}&=\Lambda-\frac{S\left(\beta I+\varepsilon_E\beta E+\varepsilon_Q\beta Q+\varepsilon_J\beta J\right)}{N}-\mu S\\
\dot{E}&=p+\frac{S\left(\beta I+\varepsilon_E\beta E+\varepsilon_Q\beta Q+\varepsilon_J\beta J\right)}{N}-\left(u_1(t)+k_1+\mu\right)E\\
\dot{Q}&=u_1(t)E-\left(k_2+\mu\right)Q\\
\dot{I}&=k_1E-\left(u_2(t)+d_1+\sigma_1+\mu\right)I\\
\dot{J}&=u_2(t)I+k_2Q-\left(d_2+\sigma_2+\mu\right)J\\
\dot{R}&=\sigma_1I+\sigma_2J-\mu R.\\
S(0)&=S_0,\ E(0)=E_0,\ Q(0)=Q_0,\ I(0)=I_0,\\
J(0)&=J_0,\ R(0)=R_0\\
u_{1L}&\leqslant u_1\leqslant u_{1U},\quad u_{2L}\leqslant u_2\leqslant u_{2U}.
\end{aligned} \tag{14}$$

Table 1 presents the description of the parameters in Eq. (14) and their values. Additionally, we take the initial conditions: $S_0=12\times10^6$, $E_0=1565$, $Q_0=292$, $I_0=695$, $J_0=326$, $R_0=20$.

Table 2
Statistics of the optimizations. Each error value is obtained from the mean of 50 optimizations, the standard deviation of these optimizations is tabulated in column 3.

Number of generations (ng)	Error (%) Eq. (15)	Standard deviation of the error
1000	1.26868	0.02633
2000	0.31472	0.00981
3000	0.10105	0.00252
4000	0.03316	0.00100
5000	0.01146	0.00064
6000	0.00394	0.00025
8000	0.00067	0.00017
10 000	0.00016	0.00008
15 000	0.00009	0.00002
20 000	0.00002	0.00002

In this case, the Lagrangian is of the form $\mathcal{L}(q(t),f(x(t)))$, where $q_1(t)=u_1(t)$, $q_2(t)=u_2(t)$, and $f(x(t))=B_1E(t)+B_2Q(t)+B_3I(t)+B_4J(t)$. There is no explicit dependence on t nor on the derivatives of u_1,u_2. However, this problem demands to solve the dynamics given by the system of ordinary differential equations in Eq. (14). To compute the integral in Eq. (12), we first approximate Eq. (14) using the standard fourth order Runge–Kutta method. Yan and Zou report in Yan and Zou (2008) a suboptimal solution for this optimal control problem. The authors used dimension 3 for the individuals to evolve. In contrast, our experiments reported that an optimal control requires a dimension of order 722.

Due to the large dimensionality of the individuals, taking a small number of generations may lead to significant noise in the solution, as illustrated in Fig. 1. Hence, we take a population of 300 individuals, with 10000 generations for the DE algorithm which eventually eliminates the noise. In Fig. 2, the optimal control policies computed with the DE method are presented. The mutation scale factor was $F=1$ and the crossover probability was $C_r=0.3$. According to this problem, it is not necessarily true that the classical DE fails in optimizing LSGO problems. Fig. 3 illustrates the evolution of the objective function depending on the number of generations (a) and as the function of time in (b). The computations were made on an Intel(R) Core(TM) i5-4460 computer with a CPU @ 3.20 GHz of frequency, running the Intel Fortran compiler. In this figure, we can see that for a number of generations of 8000 ($\sim$ 370 s), the convergence is reached with a minimum value min = 233590.893. The percent error defined as

$$\text{error}=\left|\frac{\min-f_{ob}(ng)}{\min}\right|\times100 \tag{15}$$

for 8000 generations is 0.00067% as reported in Table 2. In this table, the mean percent error and the standard deviation for a set of 50 optimizations are also presented. The relative small values for the standard deviations show that the optimizations have a very good stability.

We compared the results of the proposed methodology with the results of the well established forward–backward-sweep method (FBS) (Lenhart and Workman, 2007). Fig. 4 displays the differences given by

$$\delta u_i(t)=|u_i^{DE}(t)-u_i^{FBS}(t)|,\ t\in[0,t_f],\quad i=1,2 \tag{16}$$

between the optimal policies calculated with the DE and the FBS methods at time t. Since the difference between the polices is of order 10^{-3}, both methods give similar results. Moreover, after substituting the controls in Eq. (14) to compute the number of infected individuals shown in Fig. 5(a), the difference between the DE and the FBS method drops to 10^{-11}.

Applying constant control policies $u_1=u_2=0.2$ and the optimal strategies, Fig. 5(a) contrasts the respective performances to mitigate the whole infected classes $E+Q+I+J$. Optimal signals in Fig. 2 reduce the incidence of SARS infected individuals in less time. Optimal controls also are cheaper in terms of cost functional (12) and mitigate

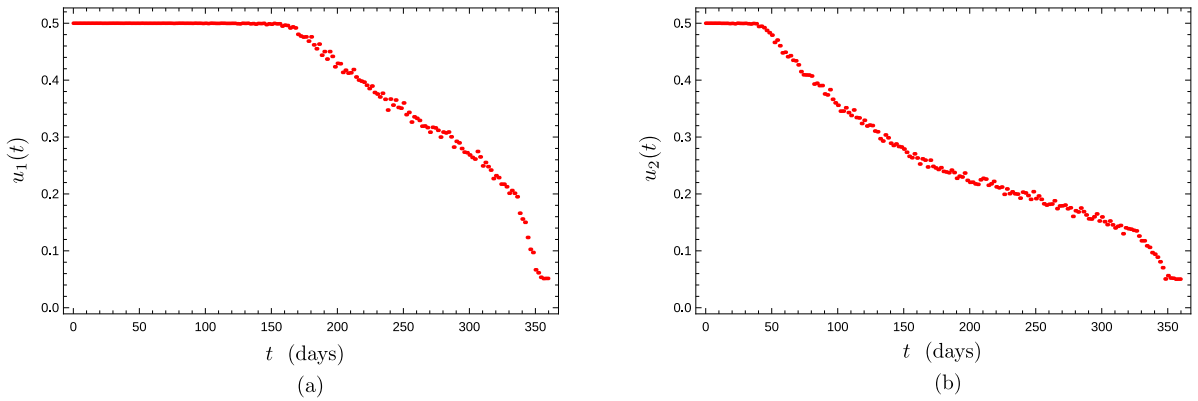

Fig. 1. Optimal controls u_1 and u_2, with $p = 720$ (the number of disjoint intervals) and 3000 generations.

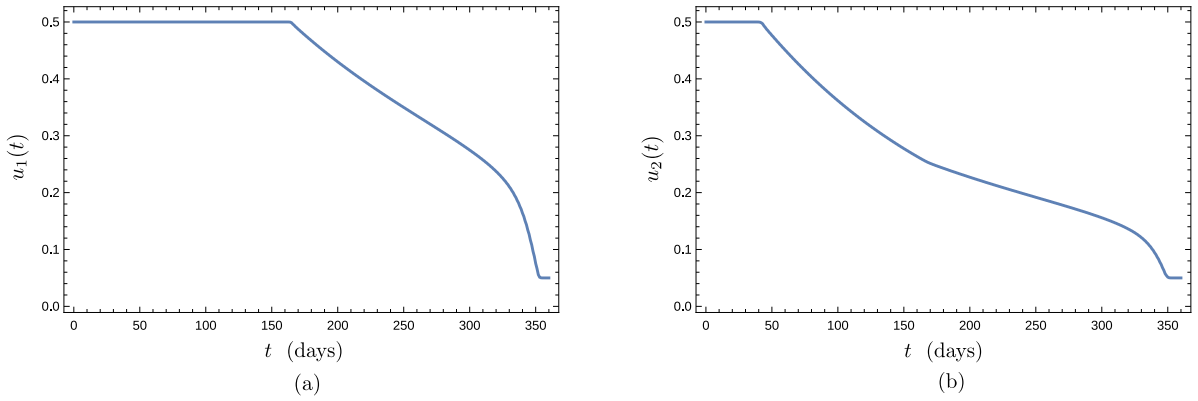

Fig. 2. Optimal controls u_1 and u_2, with $p = 720$ and 10 000 generations.

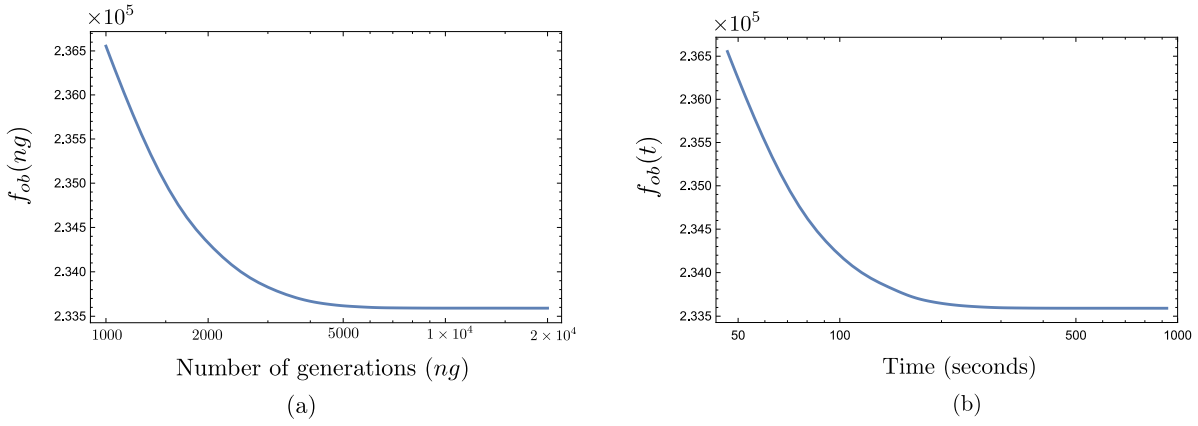

Fig. 3. Convergence of f_{ob} as a function of (a) the number of generations and (b) the time. The plots correspond to the mean value of f_{ob} for 50 optimizations.

better the spread of SARS than constant policies. For reference, in Fig. 5(b), the uncontrolled infected dynamics is illustrated; notice that the order of infected individuals rises to 10^6.

4. Brachistochrone curve problem

A classical problem of optimizing a functional, which eventually would originate the calculus of variations, is the brachistochrone curve problem. The problem was proposed by Johann Bernoulli to the readers of Acta Eruditorum in June, 1696 (Bernoulli, 1696). He wrote the following:

"Given two points A and B in a vertical plane, what is the curve traced out by a point acted on only by gravity, which starts at A and reaches B in the shortest time".

The solution is obtained by optimizing the time necessary for a particle to travel from point A to point B. This time is given by Eq. (17)

$$T = \int_A^B \frac{ds}{v}, \tag{17}$$

being ds the infinitesimal arc length and v the speed of the particle.

Without loss of generality, point A is taken as $(0, 0)$ and point B is some point in the fourth quadrant. From the energy conservation law we obtain

$$T = \int_0^{X_B} \frac{\sqrt{1 + y'^2}}{\sqrt{-2gy}}\, dx, \tag{18}$$

where g is the acceleration of gravity. Notice that the integrand in the previous equation is of the form $\mathcal{L} = \mathcal{L}(y(x), y'(x))$ which depends

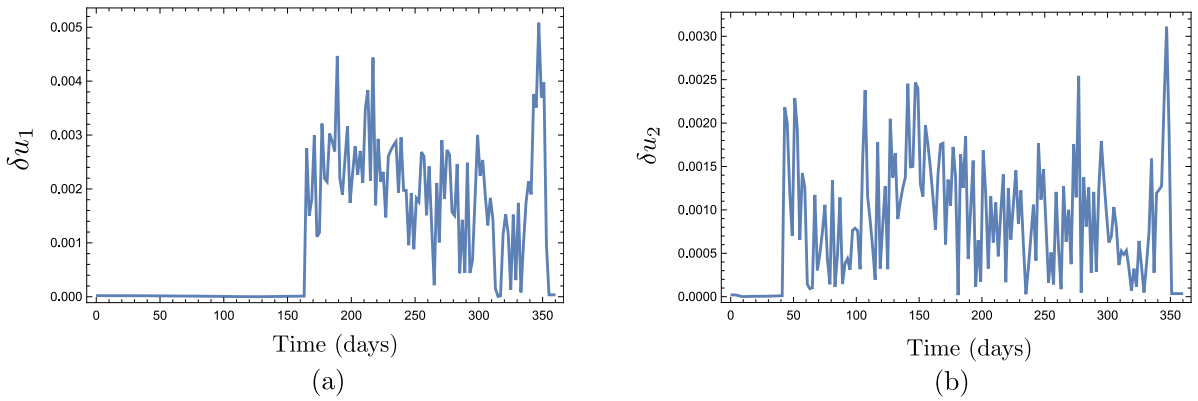

Fig. 4. Absolute differences between the optimal control signals computed with the DE and FBSM methods. Respectively, (a) and (b) correspond to the absolute differences of control signals $u_1(t)$ and $u_2(t)$.

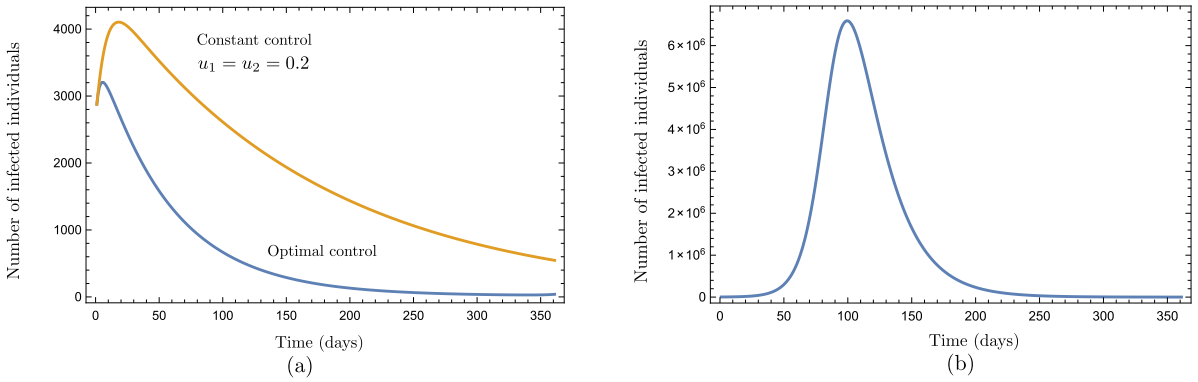

Fig. 5. Number of infected Individuals. In (a) the optimal controlled infected dynamics and the dynamics with constant controls $u_1 = 0.2$, $u_2 = 0.2$ are displayed, and (b) displays the number of infected individuals without control.

explicitly on a derivative of y. The cubic spline interpolation method in its dual form will be applied to the points $(x_1, y(x_1)), \dots (x_m, y(x_m))$ generated using Eq. (3) to compute the aforementioned derivatives. In the normal cubic spline interpolation method, the interpolated and the original points are given in the form $(x(t), y(t))$ for some dimensionless parameter $t \in [0, 1]$. In its dual form, in addition to these points, the method also yields $\dot{x}(t)$ and $\dot{y}(t)$, as mentioned above. Thus, it is convenient to rewrite Eq. (18) as follows

$$T = \int_0^{X_B} \sqrt{\frac{\dot{x}^2 + \dot{y}^2}{-2\,g\,y\,\dot{x}^2}}\, dx, \tag{19}$$

and, since $\dot{x}$ is constant for the cubic spline interpolation method, optimizing Eq. (19) is equivalent to optimize

$$\tau = \int_0^{X_B} \sqrt{\frac{\dot{x}^2 + \dot{y}^2}{-y}}\, dx. \tag{20}$$

It is important to mention that $\dot{x}$ in Eq. (20) is not a new generalized coordinate. Its existence is artificial due to the interpolation method. After applying Eq. (3) we will have the set of points $X = \{(x_1, y(x_1)), \dots, (x_m, y(x_m))\}$, and thus the individual to evolve would be

$$\mathbf{x} = [y^1, y^2, \dots y^m]. \tag{21}$$

As a concrete example, let us find the curve of fastest descent that connects the points $A = (0, 0)$ and $B = (\pi, -2)$. The exact solution, in the parametric form, is given by

$$\mathbf{r}(\theta) = [\theta - \sin\theta, \cos\theta - 1], \ \theta \in [0, \pi]. \tag{22}$$

Fig. 6 illustrates the exact solution and the solution obtained through DE. In (a), the domain $[0, \pi]$ was divided into 10 disjoint intervals. An interpolation of 10 points was used in each of the 10 disjoint intervals—between the points (x_k, y_k) and (x_{k+1}, y_{k+1}), $k = 1, \dots, 10$. In (b), 20 disjoint intervals with 150 points of interpolation were used. The information generated by the interpolation was available to the objective function. However, the dimension of the individuals was always 11 for (a) and 21 for (b), allowing a successful and efficient optimization. In addition, a population of 100 individuals with 1000 generations and 15 000 generations, for (a) and (b) respectively, were used. The mutation scale factor was $F = 1$ and the crossover probability $C_r = 0.9$. It is worthwhile to mention that the generated points by a normal cubic spline interpolation between the 21 points practically coincides with the exact solution.

Table 3
Statistics of the optimizations. Each error value is obtained from mean of 50 optimizations, the standard deviation of these optimizations is tabulated in column 3.

Number of generations (ng)	Error (%) Eq. (15)	Standard deviation of the error
1000	0.80957	0.57864
2000	0.12503	0.13501
3000	0.03362	0.05459
4000	0.01183	0.02234
5000	0.00412	0.00626
6000	0.00169	0.00218
7000	0.00086	0.00087
8000	0.00057	0.00038
9000	0.00047	0.00019
10 000	0.00042	0.00010

Fig. 7 illustrates the evolution of the objective function depending on the number of generations in (a) and as function of time in (b). In this figure, it can be observed that the convergence to the minimum (min = 0.69169) is achieved after approximately 8000 generations. This

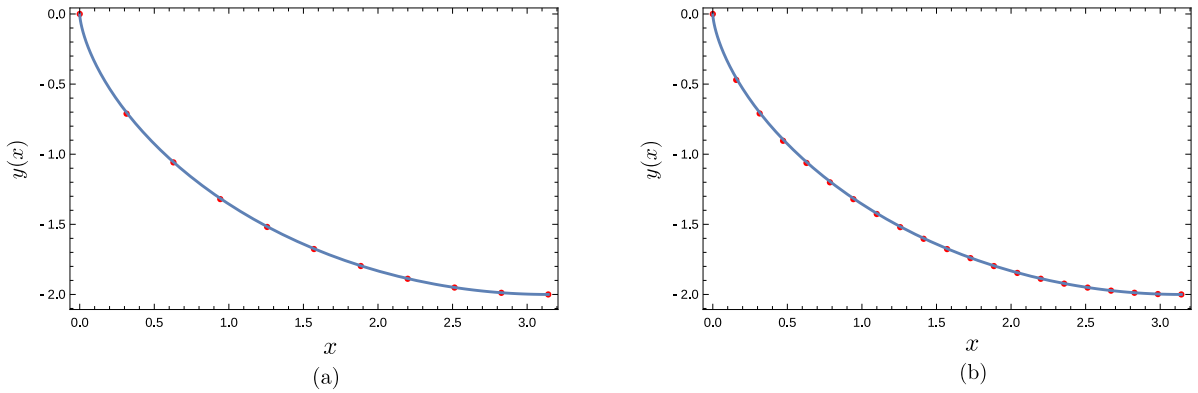

Fig. 6. Optimal solution to the Brachistochrone curve problem. The dots are the DE solution for (a) $p = 10$ and 1000 generations (b) $p = 20$ and 15 000 generations. The solid line is the exact solution.

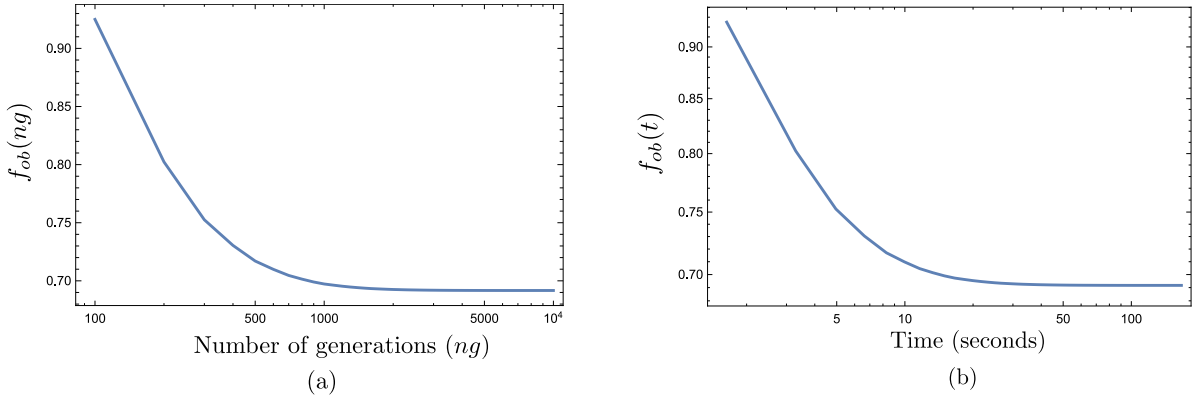

Fig. 7. Convergence of f_{ob} as a function of (a) the number of generations and (b) the time. The plots correspond to the mean value of f_{ob} for 50 optimizations.

corresponds to approximately 133 s for the aforementioned computer properties. The percent error and its standard deviation are presented in Table 3, for 50 random experimental realizations. The DE method was unable to optimize Eq. (20) without the aforementioned interpolation approach. The dependence of the Lagrangian on the derivatives appears to introduce a considerable complexity to the problem. We think our study can be used for testing another variants of the DE method and, in general, for testing any heuristic or metaheuristic method.

5. Conclusions

Differential evolution can be used for directly optimizing functionals. This kind of optimization includes problems which are usually studied in calculus of variations as well as the so-called optimal control problems. Our approach relies on the Differential Evolution as the optimization method, Automatic differentiation – via dual numbers – in the cases where the Lagrangian involves derivatives, and cubic spline interpolation. These interpolations introduce new information in the objective function without the need of increasing the dimensionality of the individuals allowing the DE method for a successful optimization. To the best of our knowledge, there are no studies using any metaheuristic algorithm to address the case when the Lagrangian depends on the velocities—as in Eq. (1). Here, by means of numerical experiments, we have demonstrated that our approach could be a good option to cope with optimization of this type of functionals. The example in Section 3 proves that, contrary to common believe, the classical DE method will not necessarily fail in optimizing LSGO problems. The presented methodology contributes to address optimal control problems evolving according to dynamics given by differential equations. Epidemiological models belong to the latter category and, with the world immersed in the COVID-19 pandemic, we hope our methodology can contribute to the study of this kind of mathematical models. Our method does not require additional knowledge on variational techniques, dynamic programming or optimal control theory methods and could be use to address the previous or any other problem involving the optimization of a functional.

CRediT authorship contribution statement

K.B. Cantún-Avila: Methodology, Software, Writing - original draft. **D. González-Sánchez:** Validation, Formal analysis. **S. Díaz-Infante:** Methodology, Software, Validation. **F. Peñuñuri:** Conceptualization, Methodology, Software, Validation, Investigation, Writing - original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix. Normal cubic spline interpolation method

A cubic spline for a set of m data points $\{(x_1, y_1), \ldots, (x_m, y_m)\}$, $m > 1$, is a spline constructed of piecewise third-order polynomials

$$Y_i(t) = a_i + b_i t + c_i t^2 + d_i t^3; \quad t \in [0, 1], \quad i = 1, \ldots, m-1, \tag{A.1}$$

which pass through such set of points.

The coefficients of Y_i are given by Bartels et al. (1987)

$$a_i = y_i$$
$$b_i = D_i$$

$$c_i = 3(y_{i+1} - y_i) - 2D_i - D_{i+1}$$
$$d_i = 2(y_i - y_{i+1}) + D_i + D_{i+1} \tag{A.2}$$

where the D_s numbers are determined by solving the symmetric tridiagonal system:

$$\mathbf{T}\,\mathbf{D} = \mathbf{R} \tag{A.3}$$

with

$$\mathbf{T} = \begin{pmatrix} 2 & 1 & & & & & \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & 4 & 1 & \\ & & & & 1 & 4 & 1 \\ & & & & & 1 & 2 \end{pmatrix} \tag{A.4}$$

$$\mathbf{D} = \begin{pmatrix} D_1 \\ D_2 \\ D_3 \\ \vdots \\ D_{m-2} \\ D_{m-1} \\ D_m \end{pmatrix} \tag{A.5}$$

$$\mathbf{R} = 3\begin{pmatrix} y_2 - y_1 \\ y_3 - y_1 \\ y_4 - y_2 \\ \vdots \\ y_{m-1} - y_{m-3} \\ y_m - y_{m-2} \\ y_m - y_{m-1} \end{pmatrix}. \tag{A.6}$$

The inverse of a $m \times m$ nonsingular tridiagonal matrix

$$\mathbf{M} = \begin{pmatrix} a_1 & b_1 & & & \\ c_1 & a_2 & b_2 & & \\ & c_2 & \ddots & \ddots & \\ & & \ddots & \ddots & b_{m-1} \\ & & & c_{m-1} & a_m \end{pmatrix},$$

can be written as (Usmani, 1994a,b):

$$\mathbf{M}^{-1}_{ij} = \begin{cases} (-1)^{i+j}\, b_i \ldots b_{j-1}\, \theta_{i-1}\, \phi_{j+1}/\theta_m & \text{if } i \leqslant j, \\ (-1)^{i+j}\, c_j \ldots c_{i-1}\, \theta_{j-1}\, \phi_{i+1}/\theta_m & \text{if } i > j, \end{cases} \tag{A.7}$$

where θ_i is obtained by solving the recurrence equation

$$\theta_i = a_i\, \theta_{i-1} - b_{i-1} c_{i-1} \theta_{i-2}, \text{ for } i = 2, \ldots, m, \tag{A.8}$$

with $\theta_0 = 1$ and $\theta_1 = a_1$. Then, ϕ_i is obtained by solving the recurrence equation

$$\phi_i = a_i\, \phi_{i+1} - b_i c_i \phi_{i+2}, \text{ for } i = m-1, \ldots, 1, \tag{A.9}$$

with $\phi_{m+1} = 1$ and $\phi_m = a_m$.

After solving Eqs. (A.7), (A.8), and (A.9) for the particular case of Eq. (A.4), we obtain

$$\theta_s = \begin{cases} \dfrac{\left(2-\sqrt{3}\right)^s + \left(2+\sqrt{3}\right)^s}{2} & \text{if } s \neq m, \\ \dfrac{\left(2+\sqrt{3}\right)^m \left(-3+2\sqrt{3}\right) - \left(2-\sqrt{3}\right)^m \left(3+2\sqrt{3}\right)}{2} & \text{if } s = m, \end{cases} \tag{A.10}$$

$$\phi_s = \begin{cases} \dfrac{\left(2-\sqrt{3}\right)^{1+m}\left(2+\sqrt{3}\right)^s + (2-\sqrt{3})^s\,(2+\sqrt{3})^{1+m}}{2} & \text{if } s \neq 1, \\ \dfrac{\left(2+\sqrt{3}\right)^m \left(-3+2\sqrt{3}\right) - \left(2-\sqrt{3}\right)^m \left(3+2\sqrt{3}\right)}{2} & \text{if } s = 1. \end{cases} \tag{A.11}$$

Defining inv as

$$\text{inv}(s,k) = (-1)^{s+k} \theta_{s-1}\, \phi_{k+1}/\theta_m, \tag{A.12}$$

and writing explicitly ϕ_{k+1}/θ_m, we obtain

$$\text{inv}(s,k) = \frac{(-1)^{s+k}}{2} \frac{1 + (\alpha^\dagger/\alpha)^{s-1}}{\beta^\dagger - (\alpha^\dagger/\alpha)^m\, \beta}$$
$$\times \left[\alpha\, \mathrm{e}^{(k+1)\ln\alpha^\dagger + (s-1)\ln\alpha} + \alpha^\dagger\, \mathrm{e}^{(s+k)\ln\alpha + m\,\ln(\alpha^\dagger/\alpha)}\right], \tag{A.13}$$

with

$$\alpha = 2 + \sqrt{3}$$
$$\alpha^\dagger = 2 - \sqrt{3}$$
$$\beta = 2\sqrt{3} + 3$$
$$\beta^\dagger = 2\sqrt{3} - 3. \tag{A.14}$$

From this, the inverse of the matrix $\mathbf{T}$ is

$$\mathbf{T}^{-1}_{sk} = \begin{cases} \text{inv}(s,k) & \text{if } s \leqslant k, \\ \text{inv}(k,s) & \text{if } s > k. \end{cases} \tag{A.15}$$

Thus, the coefficients D_s are given by

$$D_s = \sum_{k=1}^{m} \mathbf{T}^{-1}_{sk} \mathbf{R}_k, \quad s = 1, \ldots, m. \tag{A.16}$$

Now that all the coefficients have been determined, the parametric equation for the interpolated points will be

$$\mathbf{r}_i(t) = \left\{x_i + (x_{i+1} - x_i)\, t,\ Y_i(t)\right\}. \tag{A.17}$$

By implementing Eq. (A.17) in the context of dual numbers, we are capable of computing the derivatives of $\mathbf{r}$ as well as the derivatives of $f(\mathbf{r}(t))$ or $\mathbf{r}(f(t))$, for a differentiable function f. The Fortran code for this implementation can be downloaded from Peñuñuri et al. (2020).

References

Abdel-Basset, M., Abdel-Fatah, L., Sangaiah, A.K., 2018. Chapter 10 - metaheuristic algorithms: A comprehensive review. In: Sangaiah, A.K., Sheng, M., Zhang, Z. (Eds.), Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications, Intelligent Data-Centric Systems. Academic Press, pp. 185–231. http://dx.doi.org/10.1016/B978-0-12-813314-9.00010-4.

Altan, A., Karasu, S., Bekiros, S., 2019. Digital currency forecasting with chaotic meta-heuristic bio-inspired signal processing techniques. Chaos Solitons Fractals 126, 325–336. http://dx.doi.org/10.1016/j.chaos.2019.07.011.

Banyhussan, Q.S., Hanoon, A.N., Al-Dahawi, A., Yıldırım, G., Abdulhameed, A.A., 2020. Development of gravitational search algorithm model for predicting packing density of cementitious pastes. J. Build. Eng. 27, 100946. http://dx.doi.org/10.1016/j.jobe.2019.100946.

Bartels, R.H., Beatty, J.C., Barsky, B.A., 1987. An Introduction to Splines for Use in Computer Graphics and Geometric Modeling. Morgan Kaufmann, Los Altos, CA.

Bernoulli, J., 1696. Problema novum ad cujus solutionem mathematici invitantur. a new problem to whose solution mathematicians are invited. Acta Erud. 18, 269.

Bilal, Pant, M., Zaheer, H., Garcia-Hernandez, L., Abraham, A., 2020. Differential evolution: A review of more than two decades of research. Eng. Appl. Artif. Intell. 90, 103479. http://dx.doi.org/10.1016/j.engappai.2020.103479.

Biswas, P.P., Suganthan, P., Wu, G., Amaratunga, G.A., 2019. Parameter estimation of solar cells using datasheet information with the application of an adaptive differential evolution algorithm. Renew. Energy 132, 425–438. http://dx.doi.org/10.1016/j.renene.2018.07.152.

Box, G.E.P., 1957. Evolutionary operation: A method for increasing industrial productivity. J. R. Stat. Soc. C 6 (2), 81–101.

Bremermann, H., 1962. Optimization through evolution and recombination. In: Yovits, M., Goldstein, G., Jacobi, G. (Eds.), Self-Organizing Systems. Spartan, Washington, DC.

Clifford, W., 1873. Preliminary sketch of biquaternions. Proc. Lond. Math. Soc. 1 (1–4), 381–395.

Dasgupta, K., Roy, P.K., Mukherjee, V., 2020. Power flow based hydro-thermal-wind scheduling of hybrid power system using sine cosine algorithm. Electr. Power Syst. Res. 178, 106018. http://dx.doi.org/10.1016/j.epsr.2019.106018.

Fike, J., Alonso, J., 2011. The development of hyper-dual numbers for exact second-derivative calculations, AIAA Paper 2011-886. http://dx.doi.org/10.2514/6.2011-886.

Fleming, W.H., Soner, H.M., 2006. Controlled Markov Processes and Viscosity Solutions, second ed. In: Stochastic Modelling and Applied Probability, vol. 25, Springer, New York.

Friedberg, R.M., 1958. A learning machine: Part I. IBM J. Res. Dev. 2, 2–13.

Friedberg, R.M., Dunham, B., North, J.H., 1959. A learning machine: Part II. IBM J. Res. Dev. 3, 282–287.

Griewank, A., 1989. On automatic differentiation. In: Iri, M., Tanabe, K. (Eds.), Mathematical Programming: Recent Developments and Applications. Kluwer, Dordrecht, The Netherlands.

Gumel, A., Ruan, S., Day, T., Watmough, J., Brauer, F., Van Den Driessche, P., Gabrielson, D., Bowman, C., Alexander, M., Ardal, S., Wu, J., Sahai, B., 2004. Modelling strategies for controlling sars outbreaks. Philos. Trans. R. Soc. B 271 (1554), 2223–2232. http://dx.doi.org/10.1098/rspb.2004.2800.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press, Cambridge, MA.

Jong, K.D., Fogel, D.B., Schwefel, H.-P., 1997. A history of evolutionary computation. In: Thomas, B., David, F., Zbigniew, M. (Eds.), Handbook on Evolutionary Computation (Part A, Section 2.3). IOP Publishing Ltd and Oxford University Press.

Leite, N., Melício, F., Rosa, A.C., 2019. A fast simulated annealing algorithm for the examination timetabling problem. Expert Syst. Appl. 122, 137–151. http://dx.doi.org/10.1016/j.eswa.2018.12.048.

Lenhart, S., Workman, J., 2007. Optimal Control Applied to Biological Models. In: Chapman & Hall/CRC Mathematical and Computational Biology, CRC Press.

Maučec, M.S., Brest, J., 2019. A review of the recent use of differential evolution for large-scale global optimization: An analysis of selected algorithms on the CEC 2013 LSGO benchmark suite. Swarm Evol. Comput. 50, 100428. http://dx.doi.org/10.1016/j.swevo.2018.08.005.

Metaxiotis, K., Liagkouras, K., 2012. Multiobjective evolutionary algorithms for portfolio management: A comprehensive literature review. Expert Syst. Appl. 39 (14), 11685–11698. http://dx.doi.org/10.1016/j.eswa.2012.04.053.

Neidinger, R.D., 2010. Introduction to automatic differentiation and MATLAB object-oriented programming. SIAM Rev. 52 (3), 545–563. http://dx.doi.org/10.1137/080743627.

Neng, H., Fazhi, H., Yi, Z., Yilin, C., 2020. An efficient gpu-based parallel tabu search algorithm for hardware/software co-design. Front. Comput. Sci. 14, 145316. http://dx.doi.org/10.1007/s11704-019-8184-3.

Neri, F., 2010. Recent advances in differential evolution: a survey and experimental analysis. Artif. Intell. Rev. 33, 61–106. http://dx.doi.org/10.1007/s10462-009-9137-2.

Peñuñuri, F., Cab, C., Carvente, O., Zambrano-Arjona, M., Tapia, J., 2016. A study of the classical differential evolution control parameters. Swarm Evol. Comput. 26, 86–96. http://dx.doi.org/10.1016/j.swevo.2015.08.003.

Peñuñuri, F., Cantún, K.B., González, D., Díaz-Infante, S., 2020. Additional material to article: Optimizing functionals using differential evolution, Mendeley Data, v1. http://dx.doi.org/10.17632/r7mdjt7gh9.3.

Peng, J., Zhao, S., Dong, J., Liu, Y., Meersmans, J., Li, H., Wu, J., 2019. Applying ant colony algorithm to identify ecological security patterns in megacities. Environ. Model. Softw. 117, 214–222. http://dx.doi.org/10.1016/j.envsoft.2019.03.017.

Peón-Escalante, R., Jiménez, F.C., Soberanis, M.E., Peñu nuri, F., 2020. Path generation with dwells in the optimum dimensional synthesis of stephenson III six-bar mechanisms. Mech. Mach. Theory 144, 103650. http://dx.doi.org/10.1016/j.mechmachtheory.2019.103650.

Plagianakos, V., Tasoulis, D., Vrahatis, M., 2008. A review of major application areas of differential evolution. In: Chakraborty, u.k. (Ed.), Advances in Differential Evolution. In: Studies in Computational Intelligence, vol. 143. http://dx.doi.org/10.1007/978-3-540-68830-3_8.

Pontryagin, L.S., Boltyanskii, V.G., Gamkrelidze, R.V., Mishchenko, E.F., 1962. In: Neustadt, L.W. (Ed.), The Mathematical Theory of Optimal Processes. Interscience Publishers John Wiley & Sons, Inc., New York-London, Translated from the Russian by K. N. Trirogoff.

Price, K.V., Storn, R.M., Lampinen, J.A., 2005. Differential Evolution: A Practical Approach to Global Optimization. Springer-Verlag, Berlin.

Storn, R., Price, K., 1997. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. 11 (4), 341–352. http://dx.doi.org/10.1023/A:1008202821328.

Usmani, R.A., 1994a. Inversion of a tridiagonal Jacobi matrix. Linear Algebra Appl. 212–213, 413–414. http://dx.doi.org/10.1016/0024-3795(94)90414-6.

Usmani, R., 1994b. Inversion of Jacobi's tridiagonal matrix. Comput. Math. Appl. 27 (8), 59–66. http://dx.doi.org/10.1016/0898-1221(94)90066-3.

van Brunt, B., 2004. The Calculus of Variations. Springer-Verlag, New York.

Vikhar, P.A., 2016. Evolutionary algorithms: A critical review and its future prospects. In: 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC). pp. 261–265.

Wang, Z.-X., Li, Q., 2019. Modelling the nonlinear relationship between CO_2 emissions and economic growth using a PSO algorithm-based grey Verhulst model. J. Cleaner Prod. 207, 214–224. http://dx.doi.org/10.1016/j.jclepro.2018.10.010.

Yan, X., Zou, Y., 2008. Optimal and sub-optimal quarantine and isolation control in SARS epidemics. Math. Comput. Modelling 47 (1), 235–245. http://dx.doi.org/10.1016/j.mcm.2007.04.003.

Yu, W., Blair, M., 2013. DNAD, a simple tool for automatic differentiation of Fortran codes using dual numbers. Comput. Phys. Comm. 184, 1446–1452. http://dx.doi.org/10.1016/j.cpc.2012.12.025.

Zhi, H., Liu, S., 2019. Face recognition based on genetic algorithm. J. Vis. Commun. Image Represent. 58, 495–502. http://dx.doi.org/10.1016/j.jvcir.2018.12.012.