PROCEEDINGS OF THE
AMERICAN MATHEMATICAL SOCIETY
Volume 140, Number 12, December 2012, Pages 4295–4308
S 0002-9939(2012)11340-1
Article electronically published on April 24, 2012

# NUMERICAL APPROXIMATION TO ODES USING THE ERROR FUNCTIONAL

L. BAYÓN, J. M. GRAU, M. M. RUIZ, AND P. M. SUÁREZ

(Communicated by Yingfei Yi)

ABSTRACT. In this paper we present a new method for solving systems of ordinary nonlinear differential equations with initial conditions. The method is based on the transformation of the problem to an optimal control problem. We then solve it with a technique based on the use of an integral form of the Euler equation combined with the shooting method and the cyclic coordinate descent method. Our method substantially improves a previous approach that uses iterative dynamic programming to solve the associated optimal control problem. We consider the error functional instead of the classical global error, the error functional obtained by our method being lower than that obtained by classical methods. The method presented in this paper allows us to solve a wide range of $n$th order ordinary nonlinear differential equations with initial conditions.

## 1. INTRODUCTION

For hundreds of years, ordinary differential equations (ODEs) have been used to model continuous systems in all scientific and engineering disciplines. Many mathematicians have studied the nature of these equations and many well-developed solution techniques exist. The most widely used mathematical formulation is that of an initial value problem (IVP) for a first-order system of ordinary nonlinear differential equations:

$$\begin{cases} x_i'(t) = f_i(t, x_1(t), ..., x_n(t)) \\ x_i(a) = y_i \end{cases} \tag{1.1}$$

with $i = 1, ..., n$. The numerical solution of ODEs is a well-studied problem in numerical analysis, and books on the subject abound ([5], [12], [13]). The most frequently employed numerical methods fall into the following categories: Taylor methods, Runge-Kutta methods, multistep methods, extrapolation methods and adaptive techniques. In this paper we present a new method for solving (1.1) based on transforming the IVP to an optimal control problem (OCP).

Other numerical approximations to ODEs using a variational approach are presented in [7] and in [1]. In [7] the authors use iterative dynamic programming (IDP) [10] to solve the OCP and obtain a piecewise-constant optimal control function. In

Received by the editors December 14, 2010 and, in revised form, June 3, 2011.
2010 *Mathematics Subject Classification.* Primary 65L05, 47J30.
*Key words and phrases.* Ordinary differential equation, optimal control, cyclic coordinate descent.
This work was supported by the Government of Principality of Asturias through PCTI: FICYT IB09-085 and by the Spanish Government (MICINN, project: MTM2010-15737).

[1] a suitable discretization of the error functional is pursued, and it is performed by using Hermite's interpolation and quadrature formulas.

We, however, propose a new methodology for solving the OCP, which has already proven successful within the framework of hydrothermal optimization ([3], [4]). Our method uses a variety of mathematical techniques, well known for the case of functions, though now adapted to the case of functionals, which are efficiently combined to afford a novel contribution. The technique is based on the use of an integral form of the Euler equation combined with an adapted version of the shooting method and the cyclic coordinate descent method. We shall compare the error obtained by our approach, classical methods and [7].

The paper is organized as follows. In Section 2, we state the simple case with a first-order ordinary nonlinear differential equation. In Section 3, we present the optimal control algorithm for this case. In Section 4, we state the general case of a first-order system of ordinary nonlinear differential equations. In Section 5, we describe the algorithm that provides the solution for the general case. The results of the application of the method to several numerical examples are presented in Section 6. Finally, Section 7 summarizes the main conclusions of our research.

## 2. Statement for a first-order equation

In this section we consider an IVP for a first-order ordinary nonlinear differential equation:

$$\begin{cases} \dot{x}(t) = f(t, x(t)), \\ x(a) = x_a. \end{cases} \tag{2.1}$$

Let us define the function $F : [a, b] \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$:

$$F(t, x(t), \dot{x}(t)) = [\dot{x}(t) - f(t, x(t))]^2 . \tag{2.2}$$

Next, we define the following minimization problem:

$$\begin{aligned} &\text{Minimize:} \quad E(x, \dot{x}) = \int_a^b F(t, x(t), \dot{x}(t))dt, \\ &\text{subject to:} \quad x(a) = x_a, \end{aligned} \tag{2.3}$$

where $E(x, \dot{x})$ is called the *error functional.* If the optimal solution of (2.3) is zero, since the function $F$ is continuous and nonnegative, then $F \equiv 0$. Thus, the first-order equation (2.2) will hold for all $t$, and the solution of (2.1) is obtained. We now formulate problem (2.3) as an OCP. We consider the state variable to be $x(t)$ and the control variable to be $\dot{x}(t)$. The OCP is thus

$$\begin{aligned} \min_{u(t)} \quad & E(x, u) = \int_a^b F(t, x(t), u(t))dt, \\ \text{s.t.} \quad & \dot{x}(t) = u(t), \\ & x(a) = x_a. \end{aligned} \tag{2.4}$$

We will rely on the strategy to seek a numerical approximation of (2.1) by using a discretization of (2.3). Citing [1]: "*One may be tempted to discard this approach as too naive to be fruitful...*." However, we have found that it leads to a quite accurate approximation and that it can be implemented very efficiently.

## 3. Optimal control algorithm I

In this section the standard Lagrange type problem (2.4) is formulated within the framework of optimal control ([14], [6]). The classical approach involves using Pontryagin's Minimum Principle (PMP), which results in a two-point boundary value problem (TPBVP).

Let $H$ be the Hamiltonian function associated with the problem

$$H(t,x,u,\lambda) = F(t,x,u) + \lambda \cdot u,$$

where $\lambda$ is the costate variable. In order for $u$ to be optimal, a nontrivial function $\lambda$ must necessarily exist, such that for almost every $t \in [a,b]$,

$$\dot{x} = H_\lambda(t,x,u,\cdot),$$

$$-\dot{\lambda} = H_x(t,\cdot,u,\lambda), \tag{3.1}$$

$$H(t,x,u,\lambda) = \min_{v(t)} H(t,x,v,\lambda), \tag{3.2}$$

$$x(a) = x_a;\ \lambda(b) = 0. \tag{3.3}$$

Let us term as the *coordination function* of $x$ the function in $[a,b]$, defined by

$$\mathbb{Y}_x(t) := -F_{\dot{x}}(t,x(t),\dot{x}(t)) + \int_a^t F_x(s,x(s),\dot{x}(s))ds.$$

We shall use Pontryagin's Minimum Principle (PMP) as the basis for proving this theorem.

**Theorem 1.** *If $x^* \in \widehat{C}^1$ is a solution of problem (2.4), then $\exists K \in \mathbb{R}$ such that*

$$\mathbb{Y}_{x^*}(t) = K. \tag{3.4}$$

*Proof.* Let $H$ be the Hamiltonian function associated with the problem

$$H(t,x,u,\lambda) = F(t,x,u) + \lambda \cdot u.$$

From (3.1), there exists a piecewise $C^1$ function $\lambda$ that satisfies

$$\dot{\lambda} = -H_x = -F_x. \tag{3.5}$$

From (3.5), it follows that

$$\lambda^*(t) = K - \int_a^t F_x(s,x^*(s),u(s))ds \tag{3.6}$$

with

$$K = \lambda^*(a).$$

From (3.2), it follows that $u(t)$ minimizes $H(t,x^*,\cdot,\lambda^*)$, for each $t$. Hence we have

$$F_u + \lambda^*(t) = 0. \tag{3.7}$$

From (3.6) and (3.7), we have

$$K = -F_{\dot{x}}(t,x^*(t),u(t)) + \int_a^t F_x(s,x^*(s),u(s))ds,$$

and the formula $\mathbb{Y}_{x^*}(t) = K$ of Theorem 1 is verified. □

We shall call this relation

$$-F_{\dot{x}}(t, x(t), \dot{x}(t)) + \int_a^t F_x(s, x(s), \dot{x}(s))ds = K, \quad \forall t \in [a, b], \tag{3.8}$$

the *coordination equation* for $x(t)$, and the constant $K$ will be termed the *coordination constant* of the extremal.

Thus the problem consists in finding the $K$ and the function $x_K(t)$ that satisfy (3.4) and (3.3). From the computational point of view, the construction can be performed using the same procedure as the simple shooting method [2], employing a discretized version of Equation (3.8). Thus, the method which we have developed to obtain the solution is based on the use of an integral form of the Euler equation combined with the simple shooting method. To undertake the approximate calculation of the solution, we use a numerical method similar to those used to solve differential equations in combination with an appropriate adaptation of the classical shooting method.

*Step* 1) *Approximate construction of* $x_K$.

The problem will consist in finding for each $K$ the function $x_K$ that satisfies $x_K(a) = x_a$ and condition (3.4) of Theorem 1. From the computational point of view, the approximate construction of $x_K$, which we shall call $\widetilde{x}_K$, can be performed using a discretized version of Equation (3.8). For example, the construction of each $\widetilde{x}_K$ can be performed by means of polygonals (the adapted Euler's method).

We denote

$$\mathbb{Y}_{\widetilde{x}_K}(t_n) = -F_{\dot{x}}(t_n, X_n, Y_n) + I_n,$$

and we consider the triple recurring sequence $(X_n, Y_n, I_n)$ with $h = \dfrac{b-a}{N}$, with

$$n = 0, \ldots, N-1$$

and $t_n = a + n \cdot h$, which represents the following approximations:

$$x_K(t_n) \approx \widetilde{x}_K(t_n) := X_n,$$

$$\dot{x}_K(t_n) \approx \widetilde{\dot{x}}_K(t_n) := Y_n,$$

$$x_K(t) \approx \widetilde{x}_K(t) := X_{n-1} + (t - t_{n-1}) \cdot Y_{n-1} \text{ in } [t_{n-1}, t_n],$$

$$\int_a^{t_n} F_x(s, x(s), \dot{x}(s))ds \approx I_n := \int_a^{t_n} F_x(s, \widetilde{x}_K(s), \widetilde{\dot{x}}_K(s))ds$$

and which obeys the following relation of recurrence:

$$X_0 = x_a; \; I_0 = 0.$$

$Y_n$ must be the solution of the coordination equation, i.e.,

$$-F_{\dot{x}}(t_n, X_n, Y_n) + I_n = K,$$

and, using an idea similar to Euler's method, we take

$$X_{n+1} = X_n + h \cdot Y_n$$

and finally

$$I_{n+1} = I_n + \int_{t_n}^{t_{n+1}} F_x(s, X_n + (s - t_n) \cdot Y_n, Y_n)ds,$$

where the integral could be calculated using any classical approximate method.

*Step 2) Construction of a sequence* $\{K^j\}_{j\in\mathbb{N}}$ *(the adapted shooting method).*

Varying the coordination constant, $K$, which is in fact the same as varying the value of $F_{\dot{x}}(a, x(a), \dot{x}(a))$, we construct a sequence $\{K^j\}_{j\in N}$ such that the $\lambda_{K^j}(b)$ converge at 0.

Stated more precisely, considering a specific tolerance $Tol$, we shall find the coordination constant $K^j$ that satisfies

$$|\lambda_{K^j}(b)| < Tol, \tag{3.9}$$

where

$$\lambda_{K^j}(b) = K^j - \int_a^b F_x(s, \tilde{x}_{K_j}(s), \tilde{\dot{x}}_{K_j}(s)ds.$$

The procedure is similar to the shooting method used to resolve a two-point boundary value problem (TPBVP). We implemented a Simple Shooting Method (SSM) and obtained good results. The secant method was used in the present paper, the algorithm showing rapid convergence to the optimal solution.

## 4. Statement for a first-order system

In this section we consider the general case of a first-order system of ordinary nonlinear differential equations:

$$\begin{cases} x_i'(t) = f_i(t, x_1(t), ..., x_n(t)), \\ x_i(a) = y_i, \end{cases} \quad i = 1, ..., n. \tag{4.1}$$

Let us define the function $F : [a, b] \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$:

$$F(t, x_1(t), ..., x_n(t), x_1'(t), ..., x_n'(t)) = \sum_{i=1}^{n} [x_i'(t) - f_i(t, x_1(t), ..., x_n(t))]^2. \tag{4.2}$$

Next, we define the following minimization problem:

(4.3)

$$\begin{aligned} &\text{Minimize:} \quad E(x_1, ..., x_n, x_1', ..., x_n') = \int_a^b F(t, x_1(t), ..., x_n(t), x_1'(t), ..., x_n'(t))dt, \\ &\text{subject to:} \quad x_1(a) = y_1; x_2(a) = y_2; ...; x_n(a) = y_n, \end{aligned}$$

where $E(x_1, ..., x_n, x_1', ..., x_n')$ is called the *error functional.* If the optimal solution of (4.3) is zero, since the function $F$ is continuous and nonnegative, then $F \equiv 0$. Thus, the first-order system (4.2) will hold for all $t$, and the solution of (4.1) is obtained. We now formulate problem (4.3) as an OCP. We denote $\mathbf{x}(t) = (x_1(t), ..., x_n(t))$, $\mathbf{y} = (y_1, ..., y_n)$, and $\mathbf{u}(t) = (u_1(t), ..., u_n(t))$. We consider the state variables to be $\mathbf{x}(t)$ and the control variables to be $\mathbf{x}'(t)$. The OCP is thus

$$\begin{aligned} \min_{\mathbf{u}(t)} \quad & E(\mathbf{x}, \mathbf{u}) = \int_a^b F(t, \mathbf{x}(t), \mathbf{u}(t))dt, \\ \text{s.t.} \quad & \mathbf{x}'(t) = \mathbf{u}(t), \\ & \mathbf{x}(a) = \mathbf{y}. \end{aligned} \tag{4.4}$$

## 5. Optimal control algorithm II

In this section the problem (4.4) is formulated within the framework of optimal control. The development is very similar to the one presented in Section 3.

Let $H$ be the Hamiltonian function associated with the problem

$$H(t, \mathbf{x}, \mathbf{u}, \lambda) = F(t, \mathbf{x}, \mathbf{u}) + \lambda \cdot \mathbf{u},$$

where $\lambda = (\lambda_1(t), ..., \lambda_n(t))$ is the costate vector. In order for $\mathbf{u}$ to be optimal, a nontrivial function $\lambda$ must necessarily exist such that for almost every $t \in [a, b]$, $i = 1, ..., n$

$$\begin{aligned} x_i' &= H_{\lambda_i}(t, \mathbf{x}, \mathbf{u}, \lambda_1, ..., \lambda_{i-1}, \cdot, \lambda_{i+1}, ..., \lambda_n), \\ -\lambda_i' &= H_{x_i}(t, x_1, ..., x_{i-1}, \cdot, x_{i+1}, ..., x_n, \mathbf{u}, \lambda), \\ H(t, \mathbf{x}, \mathbf{u}, \lambda) &= \min_{\mathbf{v}(t)} H(t, \mathbf{x}, \mathbf{v}, \lambda), \\ \mathbf{x}(a) = \mathbf{y};\ \lambda(b) &= \mathbf{0}. \end{aligned} \tag{5.1}$$

Let us term as the *i-th coordination function* of $\mathbf{x}$ the function in $[a, b]$, defined by

$$\mathbb{Y}^i_{\mathbf{x}}(t) := -F_{x_i'}(t, \mathbf{x}(t), \mathbf{x}'(t)) + \int_a^t F_{x_i}(s, \mathbf{x}(s), \mathbf{x}'(s))ds.$$

Proving this theorem is very easy (and similar to theorem 1) with the PMP:

**Theorem 2.** *If $\mathbf{x}^* \in \widehat{C}^1$ is a solution of our problem, then $\exists \mathbf{K} = (K_1, \ldots, K_n) \in \mathbb{R}^n$ such that*

$$\mathbb{Y}^i_{\mathbf{x}^*}(t) = K_i. \tag{5.2}$$

*We shall call this relation*

$$-F_{x_i'}(t, \mathbf{x}(t), \mathbf{x}'(t)) + \int_a^t F_{x_i}(s, \mathbf{x}(s), \mathbf{x}'(s))ds = K_i, \forall t \in [a, b], \tag{5.3}$$

*the i-th coordination equation for $\mathbf{x}(t)$, and the constant $K_i$ will be termed the i-th coordination constant of the extremal.*

5.1. **Algorithm for each $i$.** For each $i$ (assuming the rest of the variables are fixed), the problem consists in finding the $K_i$ and the function $x_i(t)$ that satisfy (5.2) and (5.1). The construction can be performed using the same procedure that we presented in Section 3.

*Step* 1) Approximate construction of $\widetilde{x}_i^{K_i}$ (*the adapted Euler method*).

The problem will consist in finding for each $K_i$ the function $\widetilde{x}_i^{K_i}$ that satisfies $\widetilde{x}_i^{K_i}(a) = y_i$ and condition (5.2) of Theorem 2.

*Step* 2) Construction of a sequence $\{K_i^j\}_{j \in \mathbb{N}}$ (*the adapted shooting method*).

Varying the coordination constant, we construct a sequence $\{K_i^j\}_{j \in \mathbb{N}}$ and search for the extremal that fulfills the second boundary condition $\lambda_i^{K_i^j}(b) = 0$.

5.2. **Algorithm for $i = 1, ..., n$.** To solve the variational problem (4.3) (with $i = 1, ..., n$), we propose an algorithm of its numerical resolution using a particular strategy related to the cyclic coordinate descent (CCD) method [9]. The classic CCD method minimizes a function of $n$ variables cyclically with respect to the coordinate variables. With our method, the problem could be solved like a sequence of problems whose error functional converges to zero.

The algorithm for the variational problem (4.3) (with $i = 1, ..., n$) carries out several iterations and at each $j$-$th$ iteration calculates $n$ stages, one for each $i$. At each stage, it calculates the optimal of $x_i(t)$, assuming the rest of the variables are fixed. For every $\mathbf{x} = (x_1, \ldots, x_n)$, we consider the functional $E^i_{\mathbf{x}}$ defined by

$$E^i_{\mathbf{x}}(v_i) = \int_a^b F^i_{\mathbf{x}}((t, x_1, ..., x_{i-1}, v_i, x_{i+1}, ..., x_n(t), x_1', ..., x_{i-1}', v_i', x_{i+1}', ..., x_n')dt.$$

We call the $i$-th *minimizing mapping* the mapping $\phi_i$ defined in the following way: for every $\mathbf{x}$

$$\phi_i(x_1, \ldots, x_i, \ldots, x_n) = (x_1, \ldots, x_i^*, \ldots, x_n),$$

where $x_i^*$ minimizes $E^i_{\mathbf{x}}$. We set $\Phi = (\phi_n \circ \phi_{n-1} \circ \cdots \circ \phi_2 \circ \phi_1)$ (i.e. the successive applications of $\{\phi_i\}_{i=1}^n$) and the recurrent sequence

$$\mathbf{x}^j = \Phi(\mathbf{x}^{j-1}),$$

$x^0$ being an admissible element (i.e. $x^0 \in \widehat{C}^1$ and $x^0(a) = y$). We thus have the descending sequence $\{x^j\}$ (i.e. $E(x^{j+1}) < E(x^j)$) whose limit $\lim_{j\to\infty} x^j$ is the solution to the problem.

Given a certain tolerance $\varepsilon$, it is simple to justify the convergence of the algorithm in a finite number of steps simply by considering the following solution set:

$$\{\mathbf{x} \mid E(\mathbf{x}) - E(\Phi(\mathbf{x})) < \varepsilon\}. \tag{5.4}$$

As can be seen, we propose a new method that employs diverse mathematical techniques which are well known for the case of functions. However, they are adapted here to the case of functionals and are efficiently combined to provide a novel contribution. A more detailed explanation of this algorithm can be consulted in Bayón et al. ([3], [4]), where the algorithm is used within the framework of hydrothermal optimization.

## 6. Numerical examples

The method presented in this paper allows us to solve a wide range of $n$th order ordinary nonlinear differential equations with initial conditions. A computer program was written (using the Mathematica package) to apply the results obtained in this paper. This section presents four examples. In subsections 6.1, 6.2 and 6.3 we present a comparison with the IDP technique, considering three of the ODEs that appears in [7]. Then, in subsection 6.4, we present one more sophisticated example: a stiff ODE.

6.1. **Example 1.** First, let us consider the following IVP:

$$\dot{x} - x - e^{t-1} = 0; \quad x(0) = 0. \tag{6.1}$$

A very simple example is likewise presented in [7]. The solution of this equation is $x(t) = te^{t-1}$. Applying the above development, problem (6.1) changes to the following form:

$$\begin{aligned} \min_{u(t)} \quad & E(x,u) = \int_0^b \left[u(t) - x(t) - e^{t-1}\right]^2 dt, \\ \text{s.t.} \quad & \dot{x}(t) = u, \\ & x(0) = 0, \end{aligned}$$

and we consider $b = 1$. In Table 1 we present the influence of the tolerance $Tol$ (see (14)) and the discretization $N$ on CPU time. We see that the CPU time increases linearly with increasing discretization, as was to be expected. However, the influence of the tolerance on CPU time is almost negligible, since the number of iterations needed in the previously developed algorithm barely increases.

Table 1. CPU time (sec) versus Tolerance $Tol$ and discretization $N$.

| $Tol \setminus N$ | 10 | 100 | 1000 | 10000 |
|---|---|---|---|---|
| $10^{-15}$ | 0.015 | 0.109 | 0.998 | 9.922 |
| $10^{-30}$ | 0.015 | 0.219 | 2.637 | 22.947 |
| $10^{-40}$ | 0.031 | 0.249 | 2.917 | 29.678 |

On the other hand, the effect of the tolerance chosen to impose the condition that the extremal satisfies the transversality condition $\lambda_i^{K_i^j}(b) = 0$ is also interesting. It should be noted that numerous trials have shown that there is no need to resort to very low tolerances, as the obtained solution and hence the error functional remain practically constant for tolerances below $10^{-15}$, only managing to slow down the algorithm unnecessarily. In Table 2 below we present the influence of discretization on the functional error, considering the tolerance to be fixed at $10^{-15}$.

Table 2. Error functional versus discretization $N$.

| $N$ | 10 | 100 | 1000 | 10000 |
|---|---|---|---|---|
| $E$ | $1.747\ 10^{-12}$ | $1.29119\ 10^{-20}$ | $1.2365\ 10^{-28}$ | $5.7167\ 10^{-33}$ |

In this table we see that very small discretizations (in the order of $N = 10$) are sufficient to obtain similar functional errors to those of traditional methods, as we shall see next in Table 3.

We now compare our solution, $x(t)$ (with $N = 100$ and $Tol = 10^{-15}$), with that obtained using IDP in [4] and with that obtained using the NDSolve instruction in the commercial software package, Mathematica [8]. Said software incorporates a variety of classical methods which we have employed using their default parameters. Instead of the classical global error, we consider the error functional $E$.

Table 3. Error functional $E$ of IVP and CPU time.

| | $E$ | CPU time (sec) |
|---|---|---|
| Our solution | $1.29119\ 10^{-20}$ | 0.109 |
| IDP | $6.8757\ 10^{-4}$ | - |
| Explicit Euler | $1.6610\ 10^{-8}$ | 0.031 |
| Explicit Midpoint | $1.2118\ 10^{-13}$ | 0.015 |
| Explicit Runge-Kutta | $2.8970\ 10^{-6}$ | 0.062 |
| Implicit Runge-Kutta | $1.6824\ 10^{-14}$ | 0.046 |
| Predictor-corrector Adams | $6.1157\ 10^{-11}$ | 0.016 |
| Backward differentiation formulae (BDF) | $1.6867\ 10^{-12}$ | 0.016 |

It can be seen from Table 3 that the error functional obtained by our method is lower than that obtained by classical methods and very lower than that obtained by the IDP method. This result follows from the fact that the aim of classical methods is to minimize the global error. Table 3 also shows the CPU time consumed by our algorithm and by classical methods (the value relative to IDP is not available in [4]). In conclusion, it may be stated that our method is not significantly more complex than classical methods when suitable discretizations and tolerances are employed.

Nonetheless, the global error obtained by means of our method (see Table 4) is reasonably satisfactory (in the order of $1.2\ 10^{-3}$), having obtained $x(1) = 0.9987$ versus $x(1) = 1.00037$ using explicit Runge-Kutta, not to mention $x(1) = 0.9918$ obtained using IDP.

Table 4. Global error of IVP.

| | $E$ |
|---|---|
| Our solution | 0.001248 |
| IDP | 0.008200 |
| Explicit Euler | 0.000163 |
| Explicit Midpoint | $4.57\ 10^{-7}$ |
| Explicit Runge-Kutta | 0.000366 |
| Implicit Runge-Kutta | $1.66\ 10^{-7}$ |
| Predictor-corrector Adams | $5.19\ 10^{-8}$ |
| Backward differentiation formulae (BDF) | $1.76\ 10^{-7}$ |

The secant method was used to calculate the approximate value of $K$ for which $\lambda^K(b) = 0$ (transversality condition) with $Tol = 10^{-30}$. The algorithm shows a rapid convergence to the optimal solution for a wide range of $K_{\min}$ and $K_{\max}$. For example, for $K_{\min} = -1$ and $K_{\max} = 5$, in 6 iterations (see Figure 1) our method gives the approximate optimal solution presented in Figure 2.

6.2. **Example 2.** Next, let us consider the following IVP:

$$y'' + t^2 y = 0; \quad y(0) = 0; y'(0) = 0.1; \tag{6.2}$$

other examples are likewise presented in [7]. Let

$$x_1(t) = y(t); \quad x_2(t) = y'(t).$$

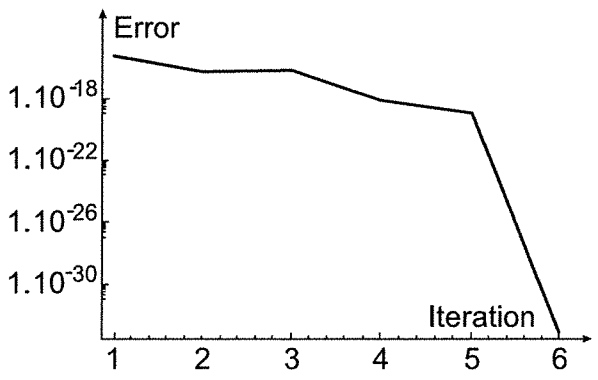


FIGURE 1. Convergence of the algorithm.

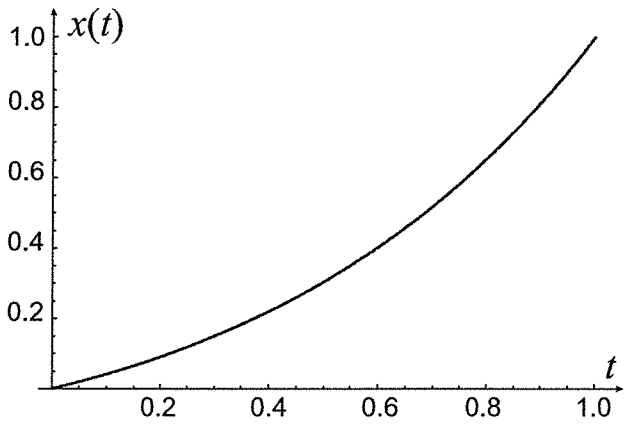


FIGURE 2. Optimal solution of IVP.

Then (6.2) transforms into a first-order system:

$$\begin{cases} x_1'(t) = x_2(t), \\ x_2'(t) = -t^2 x_1(t), \\ x_1(0) = 0; x_2(0) = 0.1. \end{cases}$$

Applying the above development, problem (6.2) changes to the following form:

$$\begin{aligned} \min_{u_1(t),u_2(t)} \quad & E(x_1, x_2, u_1, u_2) = \int_0^b \left[(u_1(t) - x_2(t))^2 + (u_2(t) + t^2 x_1(t))^2\right] dt, \\ \text{s.t.} \quad & x_1'(t) = u_1; \quad x_2'(t) = u_2, \\ & x_1(0) = 0; \quad x_2(0) = 0.1. \end{aligned}$$

We consider $b = 1$, a discretization of 100 subintervals, a tolerance for the transversality condition $Tol = 10^{-15}$ (see (14)), and a tolerance $\varepsilon = 10^{-5}$ (see (22)) for the cyclic coordinate descent. In 8 iterations of the cyclic coordinate descent, our method gives the approximate optimal solution presented in Figure 3. We now compare our solution, $y(t)$, with that obtained using IDP in [7] and with that obtained using the NDSolve instruction in the commercial software package, Mathematica [8].

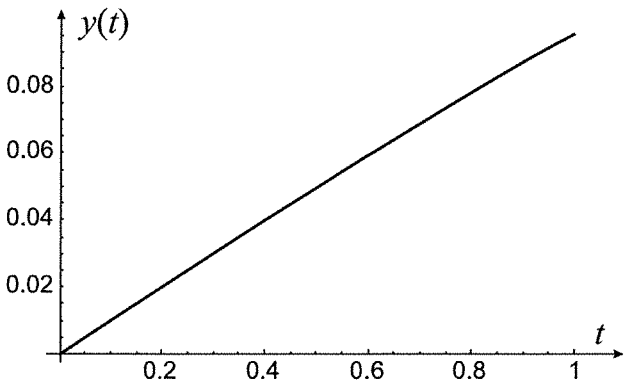


FIGURE 3. Optimal solution.

Yet again it can be observed (see Table 5) that the error functional obtained by our method is lower than that obtained by IDP and that obtained by classical methods. The global error obtained by means of our method is satisfactory (in the order of 2.3 $10^{-4}$), having obtained $y(1) = 0.09530$ versus $y(1) = 0.095069$ using Runge-Kutta, not to mention, $y(1) = 0.0881$ obtained using IDP.

Table 5. Error functional $E$ and CPU time.

| | $E$ | CPU time (sec) |
|---|---|---|
| Our solution | $4.8\ 10^{-13}$ | 1.46 |
| IDP | $1.2\ 10^{-3}$ | - |
| Euler | $4.9\ 10^{-3}$ | 0.016 |
| Midpoint | $6.4\ 10^{-5}$ | 0.015 |
| Runge-Kutta | $8.1\ 10^{-14}$ | 0.015 |
| Predictor-corrector Adams | $1.5\ 10^{-11}$ | 0.016 |

Unlike the previous example, in this example it is necessary to use the cyclic coordinate descent. As we have seen in Table 5 the algorithm slows down somewhat, but without abandoning the linear complexity with respect to the discretization.

6.3. **Example 3.** Next, we apply the same above development for the following nonlinear IVP:

$$y'' - (\sin t)^2 y'^2 + 2y = 0; \quad y(0) = 0; y'(0) = 0.1; \tag{6.3}$$

other examples are likewise presented in [7]. Once more we consider $b = 1$ and a discretization of 1000 subintervals. In this case we obtain the approximate optimal solution presented in Figure 4. If we compare the error functional of our solution $y(t)$ with that obtained using IDP and with that obtained using the NDSolve instruction of Mathematica, then yet again it can be observed (see Table 6) that the error functional obtained by our method is lower than that obtained by IDP and that obtained by the classical Euler and Midpoint methods.

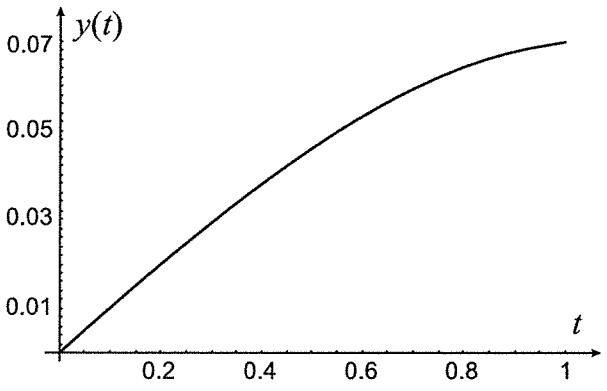


FIGURE 4. Optimal solution.

Table 6. Value of the error functional $E$.

| | $E$ |
|---|---|
| Our solution | $1.6\ 10^{-8}$ |
| IDP | $6.1\ 10^{-3}$ |
| Euler | $3.8\ 10^{-2}$ |
| Midpoint | $1.8\ 10^{-7}$ |
| Runge-Kutta | $5.2\ 10^{-14}$ |
| Predictor-corrector Adams | $4.1\ 10^{-9}$ |

6.4. **Stiff ODEs.** Stiffness is a subtle, difficult and important concept in the numerical solution of ordinary differential equations. In mathematics, a stiff equation [11] is a differential equation for which certain numerical methods for solving the equation are numerically unstable, unless the step size is taken to be extremely small. Normally the equation includes some terms that can lead to rapid variation in the solution. The numerical methods used to solve IVPs can be categorized as implicit or explicit according to whether or not they require the solution of a nonlinear system of equations at every integration step. Since the solution of a nonlinear system is computationally expensive, explicit methods are preferred whenever they can be used with reasonable stepsizes. The computational cost of the implicit methods, however, may be justifiable, since for certain IVPs (stiff IVPs) an implicit method is much more efficient than comparable explicit ones. Let us see how our method is capable of addressing problems of this nature.

The behaviour of numerical methods with respect to stiff problems can be analyzed by applying these methods to the Dalquist's test equation:

$$\dot{x} = kx;\ x(0) = 1;\ t \geq 0,$$

where $k \in C$ and $|k|$ is large. The solution of this equation is $x(t) = e^{kt}$ (see Figure 5). This solution approaches zero as $t \to \infty$ when $Re(k) < 0$. If the numerical method also exhibits this behaviour, then the method is said to be $A$-stable. We consider three values of $k$, namely $-1$, $-15$ and $-60$. We consider $b = 1$ and a discretization of $N = 1000$ subintervals. Table 7 shows the error functional obtained with our method and with the NDSolve instruction, using three methods: Explicit Runge-Kutta, Implicit Runge-Kutta and Backward Differentiation Formulae

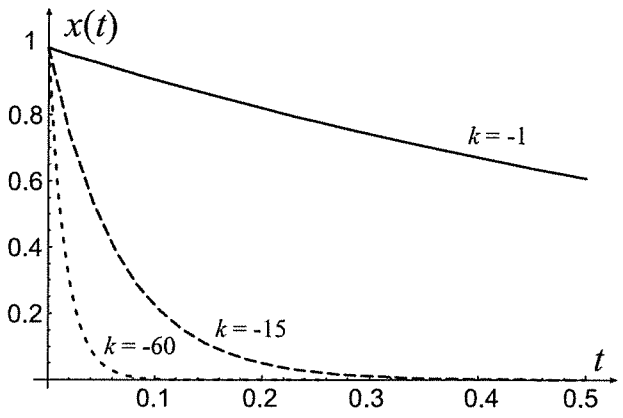


FIGURE 5. Optimal solution.

(BDF). Table 7 shows, once again, that our method is superior to the three classical methods, the poor performance of the Implicit Runge-Kutta method being notably surprising, above all, when considering the value of $k = -60$.

Table 7. Error functional of the stiff test equation.

| | $k = -1$ | $k = -15$ | $k = -60$ |
|---|---|---|---|
| Our solution | $8.012\ 10^{-36}$ | $6.5488\ 10^{-32}$ | $9.31085\ 10^{-13}$ |
| Explicit Runge-Kutta | $1.429\ 10^{-11}$ | $4.9975\ 10^{-6}$ | $2.55882\ 10^{-6}$ |
| Implicit Runge-Kutta | $1.429\ 10^{-11}$ | 0.00256469 | 18.5127 |
| BDF | $6.3618\ 10^{-13}$ | $8.05371\ 10^{-11}$ | $3.44955\ 10^{-10}$ |

It is well known that the MATLAB solver ode45 (used most of the time) [13] is slow when the problem is stiff. In such cases, there are more appropriate solvers such as, for example, ode15s. Although the goal of our method is to minimize the error functional, the global error committed is very satisfactory in numerous cases, such as the one we are dealing with here. In Table 8 we compare our solution with that obtained with these MATLAB solvers.

Table 8. Global error of the stiff test equation.

| | $k = -1$ | $k = -15$ | $k = -60$ |
|---|---|---|---|
| Our solution | $1.84016\ 10^{-4}$ | $3.285966\ 10^{-8}$ | $8.680544\ 10^{-8}$ |
| ode45 | $1.20903\ 10^{-9}$ | $1.609772\ 10^{-8}$ | $2.934628\ 10^{-7}$ |
| ode15s | $4.16291\ 10^{-4}$ | $6.409395\ 10^{-8}$ | $8.525057\ 10^{-9}$ |

As can be seen, our method is better (from the point of view of the global error) than ode15s when the problem is not stiff ($k = -1, -15$) and better than ode45 when the problem is stiff ($k = -60$). Hence, our method is robust, and it is not necessary to test the nature of the problem when choosing the most suitable solver (often we do not know if a particular ODE model is stiff).

The CPU time used was 0.7 sec on a personal computer (Pentium IV/$2GHz$). To sum up, our method may be used to stiff problems, and there is no need to take small steps to obtain satisfactory results.

## 7. Conclusions and future perspectives

In this paper we have presented a new optimal control technique for solving ordinary differential equations. Our method substantially improves a previous approach that uses iterative dynamic programming to solve the associated optimal control problem. We consider the error functional instead of the classical global error, the error functional obtained by our method being lower than that obtained by classical methods. The global error, which is the error normally considered, is not always really important, especially in problems of variational origin.

It is true that the proposed algorithm cannot improve the running times of other methods, e.g. that of Euler, as it is in fact an adaptation of said method (with a similar number of operations) and requires several runs that are moderately small in number and which depend on the tolerance. However, and precisely for this reason, once a tolerance has been fixed, the computational complexity is the same as that of Euler's method: $O(n)$, where $n$ represents the number of intervals considered in the discretization. In short, although our method is no faster than classical approaches, it runs in linear time and thus does not entail any risk in this respect.

Finally, it should be noted that our method may be applicable to initial value problems of a very general nature, as well as to boundary value problems. It would also be worth studying the application of this method to obtain the numerical solution of differential equations with algebraic equality (DAE) and inequality (DAI) constraints or even for nonsmooth differential equations.

## REFERENCES

1. S. Amat and P. Pedregal, A variational approach to implicit ODEs and differential inclusions, *ESAIM: COCV* 15(1): 139-148 (2009). MR2488572 (2010f:34006)
2. U.M. Ascher, R. Mattheij, and R.D. Russell, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, Classics in Applied Mathematics, 13, SIAM, PA, 1995. MR1351005 (96f:65075)
3. L. Bayón, J.M. Grau, M.M Ruiz and P.M. Suárez, A Bolza problem in hydrothermal optimization, *Appl. Math. Comput.* 184(1): 12-22 (2007). MR2295455
4. L. Bayón, J.M. Grau, M.M Ruiz and P.M. Suárez, A constrained and nonsmooth hydrothermal problem, *Appl. Math. Comput.* 209(1): 10-18 (2009). MR2493281 (2010h:80022)
5. R. Burden and J. Faires, *Numerical Analysis*, Thomson Brooks/Cole, Belmont, CA, 2005.
6. F.H. Clarke, Y.S. Ledyaev, R.J. Stern, and P.R. Wolenski, *Nonsmooth Analysis and Control Theory*, Springer, NY, 1998. MR1488695 (99a:49001)
7. S. Effati and H. Roohparvar, Iterative dynamic programming for solving linear and nonlinear differential equations, *Appl. Math. Comput.* 175: 247-257 (2006). MR2216338
8. S. Hassani, *Mathematical Methods Using Mathematica,* Springer, NY, 2003.
9. Z.Q. Luo and P. Tseng, On the convergence of the coordinate descent method for convex differentiable minimization, *J. Optim. Theory Appl.* 72(1): 7-35 (1992). MR1141764 (92k:90092)
10. R. Luus, *Iterative Dynamic Programming*, Chapman & Hall/CRC Press, Boca Raton, FL, 2000. MR1750212 (2001i:49002)
11. H. Ramos and J. Vigo-Aguiar, An almost $L$-stable BDF-type method for the numerical solution of stiff ODEs arising from the method of lines, *Numer. Meth. Part. D. E.* 23(5): 1110-1121 (2007). MR2340663 (2008j:65095)
12. L. F. Shampine, *Numerical Solution of Ordinary Differential Equations*, Chapman & Hall/CRC Press, NY, 1994. MR1261869 (94j:65007)
13. L. F. Shampine, I. Gladwell, and S. Thompson, *Solving ODEs with MATLAB*, Cambridge University Press, Cambridge, UK, 2003. MR1985643 (2004f:65221)
14. R. Vinter, *Optimal Control*, Birkhäuser, Boston, 2000. MR1756410 (2001c:49001)

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF OVIEDO, E.P.I. CAMPUS OF VIESQUES, GIJÓN, 33203, SPAIN
*E-mail address*: bayon@uniovi.es

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF OVIEDO, E.P.I. CAMPUS OF VIESQUES, GIJÓN, 33203, SPAIN

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF OVIEDO, E.P.I. CAMPUS OF VIESQUES, GIJÓN, 33203, SPAIN

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF OVIEDO, E.P.I. CAMPUS OF VIESQUES, GIJÓN, 33203, SPAIN