LEAST-SQUARES SOLUTIONS
OF NONLINEAR DIFFERENTIAL EQUATIONS

Daniele Mortari, Hunter Johnston, and Lidia Smith

This study shows how to obtain least-squares solutions to initial and boundary value problems of nonlinear differential equations. The proposed method begins using an approximate solution obtained by any existing integrator. Then, a least-squares fitting of this approximate solution is obtained using a constrained expression, introduced in Ref. [10]. This expression has embedded the differential equations constraints. The resulting expression is then used as an initial guess in a Newton iterative process that increases the solution accuracy to machine error level in no more than two iterations for most of the problems considered. For non-smooth solutions or for long integration times, a piecewise approach is proposed. The highly accurate value estimated at the final time is then used as the new initial guess for the next time range, and this process is repeated for subsequent time ranges. This approach has been validated for the simple oscillator and Duffing oscillator for 10,000 subsequent time ranges obtaining a 10^{-12}-level of final accuracy. A final numerical test is provided for a boundary value problem with a known solution, requiring 18 iterations to reach machine error accuracy.

Acronyms used throughout this paper

NDE → Nonlinear Differential Equation
LDE → Linear Differential Equation
IVP → Initial Value Problem
BVP → Boundary Value Problem
MVP → Multi-point Value Problem
LS → Least-Squares

INTRODUCTION

The n-th order nonlinear Differential Equation (DE) is the equation

\[ y^{(n)} = f \left( y^{(n-1)}, y^{(n-2)}, \ldots, \dot{y}, y, t \right), \]

where \( y^{(k)} = \frac{d^k y}{dt^k} \), \( f(\cdot) \) is a nonlinear function of its arguments, and \( n \geq 1 \). This type of equation appears in many problems and in almost all scientific disciplines.

Equation (1) can be solved by many existing approaches, most of which are based on the Runge-Kutta family of integrators [1]. Other methods include: Gauss-Jackson [2], time domain collocation

* Aerospace Engineering, Texas A&M University, College Station, TX. IEEE and AAS Fellow, AIAA Associate Fellow. E-mail: MORTARI@TAMU.EDU
† Aerospace Engineering, Texas A&M University, College Station, TX. E-mail: HUNTERJOHNSTON@TAMU.EDU
‡ Mathematics, Blinn College, College Station, TX. E-mail: LIDIA.SMITH@BLINN.EDU
techniques [3, 4], and Modified Chebyshev-Picard Iteration [5, 6, 7] (a path-length integral approximation which has been recently proven to be highly effective). All of these methods are based on low-order Taylor expansions, which limit the step size that can be used to propagate the solution.

In the Collocation Methods [8] (CM), the solution components are approximated by piecewise polynomials on a mesh. The coefficients of the polynomials form the unknowns to be computed. The approximation to the solution must satisfy the constraint conditions and the DE at collocation points in each mesh subinterval. In the CM the placement of the collocation points is not arbitrary. A modified Newton-type method, known as quasi-linearization, is then used to solve the nonlinear equations for the polynomial coefficients. The mesh is then refined by attempting to equidistribute the estimated error over the whole interval, and therefore an initial estimation of the solution across the mesh is required. A common weakness of all methods based on low-order Taylor expansion is that they are not effective in enforcing algebraic constraints. This is the actual strength of the proposed Least-Squares (LS) method, since the DE constraints are embedded in the searched solution expression (constrained expression). This means that all solutions generated by this method perfectly (meaning analytically) satisfy the DE constraints. The differences between the LS approach and the CM are several. The CM requires the solution satisfying the nonlinear DE in certain points (collocation points) and where the nonlinear DE constraints are defined. The LS uses constrained expressions to always satisfy the nonlinear DE constraints. It then looks for the solution as a linear combination of basis functions. In general, the LS does not satisfy the nonlinear DE at all points, except for the constraints points. Therefore, while the CM uses piecewise approximation polynomials, LS uses a single analytical expression for the whole range.

Spectral methods [9] are also used to numerically solve DE. In these methods the solution is expressed as a sum of certain “basis functions” (generally Fourier series) and the coefficients in the sum are computed to satisfy the DE as closely as possible. The boundary conditions must then be enforced, which is done by replacing one (or more) of the equations with the constraints conditions. The proposed method proceeds in the reverse sequence: it first takes care of the constraints by deriving a constrained expression [10] (equation that always satisfies the DE constraints) and then finds the least-squares solution by expressing the free function, \( g(t) \), of the constrained expression as a linear combination of “basis functions.”

The least-squares method proposed in this study has three steps (see Flowchart given in Fig. 1).

1. The first step consists of obtaining an initial solution (a set of \([t_k, y_k]\) points) using any existing integrator (e.g., Runge-Kutta);

2. Then it performs least-squares fitting of this initial solution using a constrained expression, introduced in Ref. [10], that has embedded the DE constraints. The least-squares minimizes the \( L_2 \) norm of the DE residuals. The constrained expression is provided in terms of an unknown function, \( g(t) \), which is expressed as a linear combination of \( m \) independent known basis functions, \( g(t) = \xi_0^T h(t) \), where \( \xi_0 \) is a vector of \( m \) unknown coefficients and \( h(t) \) is a vector containing the selected set of basis functions [11]. Vector \( \xi_0 \) constitutes the initial LS estimate;

3. In the third step the nonlinear DE is linearized around the approximated solution vector, \( \xi_0 \), and a Newton iterative LS technique updates the coefficients vector, \( \xi \), until convergence is obtained (see Fig. 1). The final solution has \( g(t) = \xi_f^T h(t) \), where \( \xi_f \) is the final vector of coefficients.
When the solution is non-smooth the number of basis functions would greatly increase in order to capture the highly oscillating behavior, which would create numerical issues because of the need for high degree polynomials. However, for non-smooth solutions, a strategy limiting the number of basis functions is proposed by splitting the whole range of integration in smaller subranges. Since for a short integration range the proposed method provides machine error accuracy, the estimation at the final time of a short range of integration can be used as the initial guess for the next subrange. This approach has been tested for 10,000 consecutive subranges for: 1) simple oscillator (known solution) and 2) Duffing oscillator (unknown solution). These numerical tests show that the solution accuracy provided is at a $10^{-12}$-level. Using this technique if the solution is not smooth, it is still possible to obtain highly accurate solutions by keeping low the number of basis functions.

Since the proposed method deeply relies on linear LS solver, a summary of most of the known alternative approaches to perform linear LS is provided in Appendix. Particular care is given to the technique selected to solve the linear LS problems. Specifically, our analysis shows, for our purpose, that the best LS approach consists of using the QR decomposition on a scaled (setting the norm of all columns to unity) coefficient matrix. This shows a substantial reduction of the condition number over the alternative approaches.

**Orthogonal basis functions defined in a specific range**

Orthogonal basis functions are used here as basis functions. For this reason, the most well known Orthogonal Polynomials (OP) are summarized in Appendix, where the following basis functions can be found: 1) Chebyshev OP of the first kind, 2) Legendre OP, 3) Laguerre OP, 4) two Hermite OP, and 5) Bessel OP, all defined in $x \in [-1, +1]$. For completeness, a Fourier Basis is also provided.

For generic basis functions defined in $[x_0, x_f]$ the new variable $x$ is set linearly to $t \in [t_0, t_f]$, as

$$x = x_0 + \frac{x_f - x_0}{t_f - t_0} (t - t_0) \quad \leftrightarrow \quad t = t_0 + \frac{t_f - t_0}{x_f - x_0} (x - x_0),$$

(2)

where the final time, $t_f$, is specifically defined in BVP while it can be considered as the integration
upper limit time in IVP. Setting the range ratio, \( c = \frac{x_f - x_0}{t_f - t_0} \), the derivatives in terms of the new variable are,
\[
\frac{d^k y}{dt^k} = \alpha^k \frac{d^k y}{dx^k} \quad \text{where} \quad k \in [0, n].
\]
Therefore, the nonlinear DE given in Eq. (1), written in the new \( x \) variable becomes,
\[
L\left( c^n y^{(n)}, c^{n-1} y^{(n-1)}, c^{n-2} y^{(n-2)}, \cdots, y, x \right) = 0,
\]
where the constraints becomes,
\[
\frac{d^k y}{dx^k}(t_j) = y_j^{(k)} = \alpha^k \frac{d^k y}{dx^k}(x_j).
\]

**LEAST-SQUARES USING CONSTRAINED EXPRESSIONS**

The proposed method starts by integrating the DE with an existing numerical integrator.* The numerical integrator provides an approximated solution made by \( n \) vectors, \([t, y, y', \cdots, y^{(n-2)}, y^{(n-1)}] \), whose values are provided at \( N \) times \( t_k \), ranging from \( t_0 \) to \( t_f \), and the derivatives of these vectors are specified in terms of time. Therefore, they must be converted in terms of \( x \in [x_0, x_f] \), as described in Eqs. (2) and (3). The constrained expression and its derivatives can be then written in the following form,
\[
\begin{align*}
\dot{y} &= \alpha^T(x) \xi + b(x) \\
\dot{y}' &= \alpha'^T(x) \xi + b'(x) \\
\vdots \\
\dot{y}^{(n-2)} &= \frac{d^{(n-2)} y}{dx^{(n-2)}} = \alpha^{(n-2)}(x) \xi + \frac{d^{(n-2)} b(x)}{dx^{(n-2)}} \\
\dot{y}^{(n-1)} &= \frac{d^{(n-1)} y}{dx^{(n-1)}} = \alpha^{(n-1)}(x) \xi + \frac{d^{(n-1)} b(x)}{dx^{(n-1)}}.
\end{align*}
\]

Using the vectors provided by the integrator, an initial estimation of coefficient vector, \( \xi_0 \), can be computed from the linear system,
\[
\begin{bmatrix}
\alpha^T(x) \\
\alpha'^T(x) \\
\vdots \\
\alpha^{(n-2)}(x) \\
\alpha^{(n-1)}(x)
\end{bmatrix}
\begin{bmatrix}
\xi_0 \\
y - b(x) \\
\dot{y} - b'(x) \\
\vdots \\
c^{n-2} y^{(n-2)} - b^{(n-2)}(x) \\
c^{n-1} y^{(n-1)} - b^{(n-1)}(x)
\end{bmatrix}
= b.
\]

Equation (6) is solved by least-squares providing the first estimate, \( \xi_0 \). This estimate is then used as an initial guess for an iterative Newton approach to find \( \xi_f \) by linearizing the DE around the estimated solution. The \( k \)-th iteration of this Newton approach is,
\[
L_k + \begin{bmatrix} \partial L \end{bmatrix}^T_k (\xi_{k+1} - \xi_k) \approx 0, \quad \text{where} \quad \frac{dL}{d\xi} = \sum_{i=0}^{n} \left( \frac{\partial L}{\partial y^{(i)}} \cdot \frac{\partial y^{(i)}}{\partial \xi} \right) \quad \text{and} \quad \frac{\partial y^{(i)}}{\partial \xi} = a^{(i)},
\]
and where \( L_k \) is the vector of the DE residuals specified using \( \xi_k \) at all values of vector \( x \). The convergence is obtained when the \( L_2 \) norm, \( L_2[L(x, \xi_k)] < \varepsilon \), where \( \varepsilon \) is a given tolerance.

*For the numerical examples considered in this article, the Runge-Kutta-Fehlberg method, implemented in MATLAB as “ode45,” is adopted because it is the most widely method used in commercial applications.
INITIAL VALUE PROBLEM

Even though the proposed method can be applied to nonlinear DE of any order, let’s provide a detailed explanation when applied to 2-nd order IVP,

\[ \dot{y} = f(\dot{y}, y, t) \quad \text{subject to:} \begin{align*} y(t_0) &= y_0 \quad \text{(8)} \end{align*} \]

Assuming the notation already introduced, the first and second derivatives are,

\[ \frac{dy}{dt} = \dot{y}, \quad \frac{d^2y}{dt^2} = \ddot{y}, \quad \frac{dy}{dx} = y', \quad \text{and} \quad \frac{d^2y}{dx^2} = y''. \quad \text{(9)} \]

This allows us to write the DE, Eq. (8), in terms of the new variable,

\[ L \left( x^2 \ddot{y}, cy', y, x \right) = 0 \quad \text{subject to:} \begin{align*} y(-1) &= y_0 \quad \text{(10)} \\
 y'(-1) &= \frac{\dot{y}_0}{c} = y'_0 \end{align*} \]

The constrained expression is

\[ y(x) = g(x) + (y_0 - g_0) + (x + 1) (y'_0 - g'_0). \quad \text{(11)} \]

By setting our new variable, \( g(x) \), as a linear combination of known basis functions,

\[ g(x) = \xi^T h(x), \quad g_0 = \xi^T h(-1) = \xi^T h_0, \quad \text{and} \quad g'_0 = \xi^T h'(-1) = \xi^T h'_0, \quad \text{(12)} \]

the constrained expression and its derivatives become

\[ \begin{cases} y(x) = \xi^T h(x) + (y_0 - \xi^T h_0) + (x + 1) (y'_0 - \xi^T h'_0) = a^T(x) \xi + b(x) \\
y'(x) = \xi^T h'(x) + y'_0 - \xi^T h'_0 = a'^T(x) \xi + b'(x) \\
y''(x) = \xi^T h''(x) = a''^T(x) \xi + b''(x) \end{cases} \quad \text{(13)} \]

where \( a \) and \( b \) and their derivatives, appearing in Eq. (6), are

\[ \begin{cases} a(x) = h(x) - h_0 - (x + 1) h'_0 \\
a'(x) = h'(x) - h'_0 \\
a''(x) = h''(x) \end{cases} \quad \quad \begin{cases} b(x) = y_0 + (x + 1) y'_0 \\
b'(x) = y'_0 \\
b''(x) = 0. \end{cases} \]

The expressions of \( y(x) \) and \( y'(x) \) provided in Eq. (13) are then substituted into the constrained expression, Eq. (11). The unknown coefficient vector, \( \xi \), is then derived by fitting this constrained expression with the solution provided by a numerical integrator, \([x_k, y_k]\), by LS. This provides the initial guess, \( \xi_0 \), shown in the flowchart of Fig. 1. Using this initial guess, the iterative Newton iteration process given in Eq. (7), can be started where

\[ \frac{dL}{d\xi} = \frac{\partial L}{\partial y'} \cdot \frac{\partial y'}{\partial \xi} + \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial \xi} + \frac{\partial L}{\partial y'} \cdot \frac{\partial y'}{\partial \xi}. \quad \text{(14)} \]

Specifically, \( L_k \) is specified for \( N \) values of \( x \in [-1, +1] \). Thus, obtaining a set of \( N \gg m \) equations that can be solved by LS using scaled QR method. The convergence is achieved by checking the \( L_2 \) norm of the residuals, \( L(x, \xi_k) \).
NUMERICAL EXAMPLES

In this test section, a nonlinear DE problem with a known solution has been selected to compare all three estimated solution accuracies (see flowchart in Fig. 1), \( [t_k, y_k], \xi_0, \) and \( \xi_f \). This section is also dedicated to numerically showing the high accuracy provided by the proposed LS approach along with the number of required iterations. This is done for two IVP applied to nonlinear DE with known solutions. After this, an algorithm to perform long nonlinear DE integration by LS is provided and used to integrate the Duffing oscillator for a long integration range.

Example #1

Consider the 1st order nonlinear DE problem

\[
\dot{y} = f(y, t) = (1 - 2t)y^2 \quad \text{subject to: } y(0) = y_0 \rightarrow y(t) = \frac{y_0}{t(t - 1)y_0 + 1}. \tag{15}
\]

Equation (10) becomes,

\[
\mathcal{L}(c y', y, x) = cy' - (1 - 2t)y^2 = 0 \quad \text{subject to: } y(-1) = y_0. \tag{16}
\]

The constrained expression and its first derivative are,

\[
y(x) = [h(x) - h_0]^T \xi + y_0 \quad \text{and} \quad y'(x) = \xi^T h'(x). \tag{17}
\]

Therefore, Eq. (14) can be written as,

\[
\frac{d\mathcal{L}}{d\xi} = c h'(x) - 2(1 - 2t)y [h(x) - h_0], \tag{18}
\]

and Eq. (7) becomes,

\[
\{c h' - 2(1 - 2t)y_k (h - h_0)\}^T (\xi_{k+1} - \xi_k) \approx (1 - 2t)y_k^2 - c y'_k. \tag{19}
\]

Expressing \( t \) as a function of \( x \), this equation can be written in a compact form as,

\[
a_k^T(x)(\xi_{k+1} - \xi_k) \approx b_k(x), \tag{20}
\]

which can be specified for \( N \) values of \( x \) between \( x_0 \) and \( x_f \). This leads to a set of \( N \gg m \) equations that is solved by LS using the scaled QR method given in Eq. (39) of Appendix.

Fig. 2 shows the results obtained by the proposed method for the IVP problem given in Eq. (15). Using \( m = 60 \) basis functions and \( N = 200 \) points, the solution was computed by the method detailed above and the LS solution displayed was reached after two iterations. In the top left plot of Fig. 2 the computation speed is of the method is compared to measured error and the bottom left plot shows the true (analytical) solution. Additionally, the right plot shows the LS solution error over the range of the solution after two iterations and is compared to results obtained using \texttt{ode45} and the Chebfun [14] MATLAB implementation. The the plots clearly show a noticeable speed advantage over both \texttt{ode45} and Chebfun when highly accurate solutions are need.
Example #2

Consider the nonlinear DE autonomous IVP problem,
\[ \dot{y} = y^2 \quad \text{subject to:} \quad y(t_0) = y_0 \rightarrow y(t) = \frac{y_0}{1 - y_0(t - t_0)}. \]  

(21)

This autonomous nonlinear DE, which is singular at \( t = t_0 + \frac{1}{y_0} \), can be written in terms of \( x \) as,
\[ L(y', y) = cy' - y^2 = 0 \quad \text{subject to:} \quad y(-1) = y_0, \]

(22)

where,
\[ y(x) = g(x) + (y_0 - g_0) = (h - h_0)^T \xi + y_0 \quad \text{and} \quad y'(x) = g'(x) = h' \xi. \]

(23)

Substituting, Eq. (14) becomes,
\[ \frac{dL}{d\xi} = c h' - 2[(h - h_0)^T \xi + y_0](h - h_0). \]

(24)

Fig. 3 shows the results obtained by the proposed method for the IVP problem given in Eq. (21). Using \( m = 30 \) basis functions and \( N = 100 \) points, the solution was computed by the method detailed above and the LS solution displayed was reached after two iterations. In the top left plot of Fig. 3 the computation speed is of the method is compared to measured error and the bottom left plot shows the true (analytical) solution. Additionally, the right plot shows the LS solution error over the range of the solution after two iterations and is compared to results obtained using ode45 and the Chebfun [14] MATLAB implementation. The the plots clearly show a noticeable speed advantage over both ode45 and Chebfun when highly accurate solutions are need.
LONG INTEGRATION TIME APPROACH FOR INITIAL VALUE PROBLEMS

As the integration range increases, more basis functions are needed to describe the solution and there becomes a larger matrix to invert for the LS approach. This also happens if the solution is highly oscillating, as in the case of the Duffing equation, as polynomials (even orthogonal) poorly describe oscillating behaviors. Therefore, the proposed iterative LS solution may fail for long integration time ranges or for solutions with many oscillations because the increase in the number of basis functions introduces numerical issues related to the size of the matrix to be inverted (and increases the computational time as well). Thus, increasing the number of basis functions is not a “solution” for long integration ranges or for highly oscillating behaviors. For long integration ranges the LS solution can be kept accurate by the integration strategy described in Fig. 4.

The whole integration range is split in $N_p$ parts, each with the time range, $t \in [t_{0k}, t_{fk}]$, $k = 1, 2, \cdots$. Starting with the original initial conditions at time $t_{01}$, the DE is integrated by a numerical method (indicated in the figure by RK). Then, an initial LS solution estimate ($\xi_0$) is obtained by fitting the RK solution using a constrained expression with $m$ basis functions. After this, as described in the previous section, the Newton iterative approach (indicated by I-LS) is applied using few iterations (typically one or two) to provide the final solution estimate ($\xi_f$). The first I-LS solution is provided up to time $t_{f1}$. The final values (at time $t_{f1}$) of this highly accurate solution constitute the initial conditions for the second time segment, which ranges from $t_{02} = t_{f1}$ to $t_{f2}$. This process is then repeated as sketched in Fig. 4.

To quantify the accuracy loss by this approach in subsequent time segments, this integration strategy has been applied to the simple oscillator,

$$\ddot{y} + \omega^2 y = 0 \quad \text{subject to: } y(t_0) = y_0, \text{ and } \dot{y}(t_0) = \dot{y}_0$$

(25)
Figure 4. Long integration time approach flowchart

whose general solutions is,

\[ y(t) = A \cos(\omega t + \varphi) \]

where:

\[ \varphi = \tan^{-1} \left( -\frac{\dot{y}_0}{\omega y_0} \right) - \omega t_0 \]

\[ A = \frac{y_0}{\cos(\omega t_0 + \varphi)} \]  \hspace{1cm} (26)

where \( \omega = 2\pi \) (period is 1 sec), \( y_0 \in [0,1] \), and \( \dot{y}_0 \in [0,1] \). The time length of each segment has been randomly selected with uniform distribution, \( t_{f_k} - t_{0_k} = \Delta t_k \in [0.5, 1.5] \) sec (around the period of 1 sec). Fig. 5 shows the results obtained by the proposed method when applied to the simple oscillator using \( N_p = 10,000 \) time segments, \( m = 25 \) basis functions, \( N = 100 \) points, and only one iteration per time segment.

Figure 5. Results for long integration \((t \in [0, 10,000])\) of simple oscillator
In the left plot of Fig. 5 the maximum absolute error experienced in each time segment (with respect to the true values) is plotted for the LS (ξ₀, with black marks) and the I-LS (ξ₁, with red marks) solutions. The right plot of Fig. 5 shows the histograms of the logarithm in base 10 of these two maximum errors. These two plots are provided to show the upper bound of the true error and to highlight that the accuracy loss during this process is limited. Actually, it seems that some “compensation” effect even occurs in some regions. This test shows that the error accuracy is still better than 10⁻¹⁰ after 10,000 integrations in consecutive ranges.

**DUFFING EQUATION**

The Duffing equation is an example of a dynamical system that exhibits chaotic behavior. The Duffing equation is a nonlinear 2-nd order nonlinear DE used to model certain damped and driven oscillators. The equation is,

\[
\ddot{y} + \delta \dot{y} + \alpha y + \beta y^3 - \gamma \cos(\omega t) = 0
\]  

where, \(\alpha\) controls the linear stiffness, \(\beta\) controls the amount of non-linearity in the restoring force, \(\delta\) controls the amount of damping, \(\gamma\) is the amplitude of the periodic driving force, and \(\omega\) is the angular frequency of the periodic driving force. In general, the Duffing equation does not admit an exact, analytic solution. Therefore, a numerical method is used to obtain an approximate solution.

For numerical tests, let us assume the typical values adopted in the literature: \(\alpha = -1, \beta = +1, \delta = 0.3, \omega = 1.2\) and \(\gamma = 0.4\) (usually, \(\gamma \in [0.2, 0.65]\)), with initial conditions \(y(t_0) = y_0 = 1\) and \(\dot{y}(t_0) = \dot{y}_0 = 0\). Using basis functions defined in \(x \in [-1, +1]\), the DE is written as,

\[
L(y'', y', y, t) = c^2 y'' + \delta c y' + \alpha y + \beta y^3 - \gamma \cos(\omega t) = 0,
\]  

where \(t = t_0 + (t_f - t_0)(x + 1)/2\) and where \(y\) is provided by the constrained expression

\[
y(x) = g(x) + (y_0 - g_0) + (x + 1) \left( \frac{\dot{y}_0}{c} - y_0 \right) = (h - h_0)^T \xi + y_0 + (x + 1) \left( \frac{\dot{y}_0}{c} - h_0^T \xi \right)
\]  

and its derivatives,

\[
y' = (h' - h_0')^T \xi + \frac{\dot{y}_0}{c} \quad \text{and} \quad y'' = h''^T \xi.
\]  

The derivative in Eq. (14) are,

\[
\frac{dL}{d\xi} = \frac{\partial L}{\partial y''} \frac{dy''}{d\xi} + \frac{\partial L}{\partial y'} \frac{dy'}{d\xi} + \frac{\partial L}{\partial y} \frac{dy}{d\xi} = c^2 h'' + \delta c (h' - h_0') + (\alpha + 3\beta y^2)[h - h_0 - (x + 1) h_0]
\]  

Fig. 6 shows the results obtained by the proposed LS approach when applied to the Duffing oscillator. This figure shows the \(L_2\) norm of \(L(x, \xi)\) when fitting the \texttt{ode45} \((\xi = \xi_0)\) solution and for one and two iterations of the I-LS proposed approach \((\xi_1\) and \(\xi_2\) solutions). In this example, by just two iterations, 11 orders of magnitude accuracy gain is observed using \(m = 45\) basis functions and \(N = 100\) points. Fig. 7 shows the results obtained when a long integration range is performed for the Duffing oscillator as described by the flowchart shown in Fig. 4. The long integration range considered is \(N_p = 10,000\) times the 4-sec time range considered in Fig. 6. The number of basis function and points per time segment adopted is the same as that adopted for a single 4-sec time segment \((m = 45\) and \(N = 100\) points). The left plot shows \(\max(|L_k|)\) obtained when fitting the \texttt{ode45} solution \((\xi_0)\) and when using one and two iterations of the I-LS proposed approach \((\xi_1\) and \(\xi_2\) solutions). Again, we outline that the accuracy loss during this process is limited and the \(L_2\) norm of \(L(x, \xi_f)\) obtained in the last range is still better than \(10^{-10}\).
BOUNDARY VALUE PROBLEM

A 2-nd order boundary value problem is the nonlinear DE,

$$\mathcal{L}(c^2y'', cy', y, x) = 0 \quad \text{subject to:} \quad \begin{cases} y(-1) = y_0 \\ y(+1) = y_f, \end{cases}$$

whose the constrained expression is,

$$y(x) = g(x) + \frac{1-x}{2} (y_0 - g_0) + \frac{x-1}{2} (y_f - g_f).$$

Setting $g(x) = \xi^T h(x)$, the constrained expression and its derivatives become,

$$\begin{aligned}
    y(x) &= \xi^T h(x) + \frac{1-x}{2} (y_0 - \xi^T h_0) + \frac{x-1}{2} (y_f - \xi^T h_f) \\
    y'(x) &= \xi^T h'(x) - \frac{1}{2} (y_0 - \xi^T h_0) + \frac{1}{2} (y_f - \xi^T h_f) \\
    y''(x) &= \xi^T h''(x)
\end{aligned}$$

and the procedure becomes identical to that presented in the previous section for IVP.

Numerical example

The proposed LS approach is tested for the following BVP,

$$\ddot{y} + y \dot{y} = e^{-2t} (\cos t - \sin t) - 2e^{-t} \cos t \quad \text{subject to:} \quad \begin{cases} y(0) = 0 \\ y(\pi) = 0, \end{cases}$$

whose general solution is,

$$y(t) = e^{-t} \sin t.$$
Consider this problem be solved by a *perfect shooting method*, giving the exact solution, $\dot{y}(0) = 1$. The BVP is then transformed into an IVP and the initial solution can be provided by `ode45`. The results obtained by the proposed LS approach are shown in Fig. 8.

In the top left plot the MATLAB `ode45` solution (function in black and derivative in red) is shown. The error with respect to the true solution is provided in the bottom left plot. The solution obtained by `ode45` is first (LS) fitted using the constrained expression given in Eq. (34). This constitutes the initial guess ($\xi_0$) of the proposed iterative approach. The right plot of Fig. 8 shows the maximum (black marker) and the mean (red marker) errors obtained at each iteration. This plot shows that, for this specific BVP example, using $m = 50$ basis functions and a $N = 100$ point discretization, the LS approach convergence is slower as compared to that obtained by the previously shown IVP examples. Machine error accuracy is, however, obtained at the cost of 18 iterations. The reason for the slow convergence is most likely related to the small update experienced (for this case) by the Newton iterative process. A higher order zero finder, such as the 3-rd order Halley iteration, may provide less iterations, but consequently at higher computational costs.

**CONCLUSIONS**

This study shows how to obtain least-squares solutions for initial and boundary values problems applied to nonlinear differential equations. The proposed method extends to the nonlinear differential equations the approach used in Ref. [11] for nonhomogeneous linear differential equations with nonconstant coefficients. This is done by searching the solution as *constrained expressions*, introduced in Ref. [10], which are expressions with embedded differential equation constraints. These expressions contain a function, $g(x)$, which can be freely selected, that is expressed as a linear combination of a basis functions set, such as Chebyshev or Lagrange orthogonal polynomials.
Figure 8. Boundary value problem applied to nonlinear differential equation

The coefficient vector of this linear combination, $\xi$, constitutes the unknown vector to optimize.

The proposed iterative least-square method has three steps. First, the differential equation is integrated using a numerical integrator. Second, an approximation of the unknown vector, $\xi_0$, is obtained by performing least-squares best fitting of the solution obtained by the integrator using a constrained expression for the differential equation constraints. Third, an iterative Newton solution is obtained by linearizing the differential equation around the estimated solution until the convergence reaches the machine error solution accuracy level.

The paper quantifies, by numerical examples, the accuracy obtained when applying the proposed iterative least-square method to initial and boundary values problems. These examples includes two initial values problems applied to second order differential equations and one applied to the second order Duffing oscillator. In particular, an algorithm to apply the proposed method to long integration range problems is proposed and tested for problems with a known solution (simple oscillator) and an unknown solution (Duffing oscillator). For the numerical examples considered, the machine error is obtained with no more that 2 iterations, while a slower convergence is experienced for the boundary values problem considered. Additionally, for the initial value problems, the solutions are compared for both speed and accuracy with ode45 and the Chebfun [14] MATLAB implementation which uses an adaptive spectral-collocation method.

This article just introduces the proposed approach, supported by a small set of numerical examples. Future studies will be dedicated to a detailed analysis on the number and type of basis functions as well as on the number and distribution type of the discretization points.
LINEAR LEAST-SQUARES METHODS

There are different numerical techniques to compute the linear least-squares (LS) solution of \( A \xi = b \). The most common methods are:

- The simplest approach consists with the (classic) solution,
  \[
  \xi = (A^T A)^{-1} A^T b.
  \] (37)

- using the QR decomposition,
  \[
  A = QR \rightarrow \xi = R^{-1} Q^T b,
  \]
  where \( Q \) is an orthogonal matrix and \( R \) an upper triangular matrix.

- using the SVD decompositions,
  \[
  A = U \Sigma V^T \rightarrow \xi = A^+ b = V \Sigma^+ U^T b
  \]
  where \( U \) and \( V \) are two orthogonal matrices, and where \( \Sigma^+ \) is the pseudo-inverse of \( \Sigma \), which is formed by replacing every non-zero diagonal entry by its reciprocal and transposing the resulting matrix.

- using the Cholesky decomposition,
  \[
  A^T A \xi = U^T U \xi = A^T b \rightarrow \xi = U^{-1} (U^{-T} A^T b)
  \]
  where \( U \) is a upper triangular and, consequently, \( U^{-1} \) and \( U^{-T} \) are easy to compute.

Particular attention must be given to reduce the condition number by scaling. The correct method of scaling is by scaling the columns of matrix \( A \),

\[
A (SS^{-1}) \xi = (A S) (S^{-1} \xi) = B \eta = b \rightarrow \xi = S \eta = S (B^T B)^{-1} B^T b
\] (38)

where \( S \) is the \( m \times m \) scaling diagonal matrix whose (diagonal) elements are the inverse of the norms of the corresponding \( A \) matrix columns, \( s_{kk} = |a_k|^{-1} \) or the maximum absolute value, \( s_{kk} = \max_i |a_k(i)| \). The purpose of scaling is to decrease the condition number of the matrix to invert.

In this article a scaled QR approach has been selected. This approach performs the QR decomposition of the scaled matrix,

\[
B = A S = Q R \rightarrow \xi = S R^{-1} Q^T b.
\] (39)

A weighted LS solution can be obtained by introducing an \( n \times n \) diagonal matrix of weights, \( W \). This allows to increase the accuracy on time intervals of particular interest, as around initial or final times,

\[
W A \xi = W b \rightarrow \xi = (A^T W^2 A)^{-1} A^T W b.
\] (40)

Scaling the rows of matrix \( A \) is equivalent to perform weighted LS, as it can be easily proven.
BASIS FUNCTIONS

Since the proposed method uses a set of basis functions, a summary of the candidate orthogonal polynomial basis functions is provided. Note that whatever basis functions type is adopted, constant and linear terms cannot be used because they have already been adopted to derive the constrained expressions for both IVP and BVP.

Chebyshev Orthogonal Polynomials

Chebyshev Orthogonal Polynomials of the first kind (COP), \( T_k(x) \), are defined in the \( x \in [-1, +1] \) range and they are generated using the recursive function,

\[
T_{k+1} = 2x T_k - T_{k-1} \quad \text{starting from:} \quad \begin{cases} T_0 = 1 \\ T_1 = x \end{cases}
\] (41)

All derivatives of COP can be computed in a recursive way, starting from

\[
\frac{dT_0}{dx} = 0, \quad \frac{dT_1}{dx} = 1 \quad \text{and} \quad \frac{d^d T_0}{dx^d} = \frac{d^d T_1}{dx^d} = 0 \quad (\forall d > 1),
\] (42)

while the subsequent derivatives of Eq. (41) give for \( k \geq 1 \),

\[
\frac{dT_{k+1}}{dx} = 2 \left( T_k + x \frac{dT_k}{dx} \right) - \frac{dT_{k-1}}{dx}
\]

\[
\frac{d^2 T_{k+1}}{dx^2} = 2 \left( 2 \frac{dT_k}{dx} + x \frac{d^2 T_k}{dx^2} \right) - \frac{d^2 T_{k-1}}{dx^2}
\]

\[
\vdots
\]

\[
\frac{d^d T_{k+1}}{dx^d} = 2 \left( d \frac{d^{d-1} T_k}{dx^{d-1}} + x \frac{d^d T_k}{dx^d} \right) - \frac{d^d T_{k-1}}{dx^d}; \quad (\forall d \geq 1).
\] (43)

In particular,

\[
T_k(-1) = (-1)^k, \quad \frac{dT_k}{dx} \bigg|_{x=-1} = (-1)^{k+1} k^2, \quad \frac{d^2 T_k}{dx^2} \bigg|_{x=-1} = (-1)^k \frac{k^2 (k^2 - 1)}{3}
\] (44)

and

\[
T_k(1) = 1, \quad \frac{dT_k}{dx} \bigg|_{x=1} = k^2, \quad \frac{d^2 T_k}{dx^2} \bigg|_{x=1} = \frac{k^2 (k^2 - 1)}{3}.
\] (45)

Legendre Orthogonal Polynomials

Legendre Orthogonal Polynomials (LeP), \( L_k(x) \), are defined in the \( x \in [-1, +1] \) range and they are generated using the recursive function,

\[
L_{k+1} = \frac{2k+1}{k+1} x L_k - \frac{k}{k+1} L_{k-1} \quad \text{starting:} \quad \begin{cases} L_0 = 1 \\ L_1 = x \end{cases}
\] (46)

All derivatives of LOP can be computed in a recursive way, starting from

\[
\frac{dL_0}{dx} = 0, \quad \frac{dL_1}{dx} = 1 \quad \text{and} \quad \frac{d^d L_0}{dx^d} = \frac{d^d L_1}{dx^d} = 0 \quad (\forall d > 1),
\] (47)
while the subsequent derivatives of Eq. (46) for \( k \geq 1 \), can be computed in cascade,

\[
\frac{dL_{k+1}}{dx} = \frac{2k + 1}{k + 1} \left( L_k + x \frac{dL_k}{dx} \right) - \frac{k}{k + 1} \frac{dL_{k-1}}{dx} \\
\frac{d^2L_{k+1}}{dx^2} = \frac{2k + 1}{k + 1} \left( 2 \frac{dL_k}{dx} + x \frac{d^2L_k}{dx^2} \right) - \frac{k}{k + 1} \frac{d^2L_{k-1}}{dx^2} \\
\vdots \\
\frac{d^dL_{k+1}}{dx^d} = \frac{2k + 1}{k + 1} \left( d \frac{d^{d-1}L_k}{dx^{d-1}} + x \frac{d^dL_k}{dx^d} \right) - \frac{k}{k + 1} \frac{d^dL_{k-1}}{dx^d}; \quad (\forall \ d \geq 1).
\]

(48)

**Laguerre Orthogonal Polynomials**

Laguerre Orthogonal Polynomials (LaP), \( L_k(x) \), are generated using the recursive function,

\[
L_{k+1}(x) = \frac{2k + 1 - x}{k + 1} L_k(x) - \frac{k}{k + 1} L_{k-1}(x) \quad \text{starting:} \quad \begin{cases} 
L_0 = 1 \\
L_1 = 1 - x.
\end{cases}
\]

(49)

All derivatives of LOP can be computed in a recursive way, starting from

\[
\frac{dL_0}{dx} = 0, \quad \frac{dL_1}{dx} = -1 \quad \text{and} \quad \frac{d^dL_0}{dx^d} = \frac{d^dL_1}{dx^d} = 0 \quad (\forall \ d > 1),
\]

then

\[
\frac{dL_{k+1}}{dx} = \frac{2k + 1 - x}{k + 1} \frac{dL_k}{dx} - \frac{1}{k + 1} L_k - \frac{k}{k + 1} \frac{dL_{k-1}}{dx} \\
\frac{d^2L_{k+1}}{dx^2} = \frac{2k + 1 - x}{k + 1} \frac{d^2L_k}{dx^2} - \frac{2}{k + 1} \frac{dL_k}{dx} - \frac{k}{k + 1} \frac{d^2L_{k-1}}{dx^2} \\
\vdots \\
\frac{d^dL_{k+1}}{dx^d} = \frac{2k + 1 - x}{k + 1} \frac{d^dL_k}{dx^d} - \frac{d}{k + 1} \frac{d^{d-1}L_k}{dx^{d-1}} - \frac{k}{k + 1} \frac{d^dL_{k-1}}{dx^d} 
\]

(50)

**Hermite Orthogonal Polynomials**

There are two Hermite Orthogonal Polynomials (HOP), the probabilists, indicated by \( E_k(x) \) and the physicists, indicated by \( H_k(x) \). They both are generated using the recursive function.

The probabilists are defined as

\[
E_{k+1}(x) = x E_k(x) - k E_{k-1}(x) \quad \text{starting:} \quad \begin{cases} 
E_0(x) = 1 \\
E_1(x) = x
\end{cases}
\]

(51)

All derivatives can be computed in a recursive way, starting from

\[
\frac{dE_0}{dx} = 0, \quad \frac{dE_1}{dx} = 1 \quad \text{and} \quad \frac{d^dE_0}{dx^d} = \frac{d^dE_1}{dx^d} = 0 \quad (\forall \ d > 1),
\]
then
\[ \frac{dE_{k+1}}{dx} = E_k + x \frac{dE_k}{dx} - k \frac{dE_{k-1}}{dx} \]
\[ \frac{d^2E_{k+1}}{dx^2} = 2 \frac{dE_k}{dx} + x \frac{d^2E_k}{dx^2} - k \frac{d^2E_{k-1}}{dx^2} \]
\[ \vdots \]
\[ \frac{d^dE_{k+1}}{dx^d} = d \frac{d^{d-1}E_k}{dx^{d-1}} + x \frac{d^dE_k}{dx^d} - k \frac{d^dE_{k-1}}{dx^d} \] (52)

The physicists are defined as
\[ H_{k+1}(x) = 2x H_k(x) - 2k H_{k-1}(x) \]
starting:
\[
\begin{align*}
H_0(x) &= 1 \\
H_1(x) &= 2x
\end{align*}
\] (53)

All derivatives can be computed in a recursive way, starting from
\[ \frac{dH_0}{dx} = 0, \quad \frac{dH_1}{dx} = 2 \quad \text{and} \quad \frac{d^dH_0}{dx^d} = \frac{d^dH_1}{dx^d} = 0 \quad (\forall \ d > 1), \]
then
\[ \frac{dH_{k+1}}{dx} = 2H_k + 2x \frac{dH_k}{dx} - 2k \frac{dH_{k-1}}{dx} \]
\[ \frac{d^2H_{k+1}}{dx^2} = 4 \frac{dH_k}{dx} + 2x \frac{d^2H_k}{dx^2} - 2k \frac{d^2H_{k-1}}{dx^2} \]
\[ \vdots \]
\[ \frac{d^dH_{k+1}}{dx^d} = 2d \frac{d^{d-1}H_k}{dx^{d-1}} + 2x \frac{d^dH_k}{dx^d} - 2k \frac{d^dH_{k-1}}{dx^d} \] (54)

Bessel Orthogonal Polynomials

Bessel Orthogonal Polynomials (BOP), \( B_k(x) \), are generated using the recursive function,
\[ B_{k+1}(x) = (2k + 1)x B_k(x) + B_{k-1}(x) \]
starting:
\[
\begin{align*}
B_0(x) &= 1 \\
B_1(x) &= 1 + x
\end{align*}
\] (55)

All derivatives of BOP can be computed in a recursive way, starting from
\[ \frac{dB_0}{dx} = 0, \quad \frac{dB_1}{dx} = 1 \quad \text{and} \quad \frac{d^dB_0}{dx^d} = \frac{d^dB_1}{dx^d} = 0 \quad (\forall \ d > 1), \]
then
\[ \frac{dB_{k+1}}{dx} = (2k + 1)B_k + (2k + 1)x \frac{dB_k}{dx} + \frac{dB_{k-1}}{dx} \]
\[ \frac{d^2B_{k+1}}{dx^2} = 2(2k + 1) \frac{dB_k}{dx} + (2k + 1)x \frac{d^2B_k}{dx^2} + \frac{d^2B_{k-1}}{dx^2} \]
\[ \vdots \]
\[ \frac{d^dB_{k+1}}{dx^d} = d(2k + 1) \frac{d^{d-1}B_k}{dx^{d-1}} + (2k + 1)x \frac{d^dB_k}{dx^d} + \frac{d^dB_{k-1}}{dx^d} \] (56)
In addition, and for completeness, a Fourier basis functions is provided.

**Fourier Basis**

Fourier Series (FS) provides an approximation of the $g(t)$ function

$$g(t) = a_0 + \sum_{k=1}^{m} \left[ a_k \cos(k\omega_0 t) + b_k \sin(k\omega_0 t) \right]$$  \hspace{1cm} (57)

The first two derivatives of FS are

$$\dot{g}(t) = \sum_{k=1}^{m} \left[ -a_k \sin(k\omega_0 t) + b_k \cos(k\omega_0 t) \right] (k\omega_0)$$  \hspace{1cm} (58)

and

$$\ddot{g}(t) = \sum_{k=1}^{m} \left[ -a_k \cos(k\omega_0 t) - b_k \sin(k\omega_0 t) \right] (k\omega_0)^2$$  \hspace{1cm} (59)

where the basic frequency is selected as, $\omega_0 = \frac{2\pi}{t_f - t_0}$, that is one period for the $[t_0, t_f]$ time interval.

**REFERENCES**


