Computational methods for nonlinear dynamical systems

Xuechuan WANG* and Satya N. ATLURI**
* National Key Laboratory of Aerospace Flight Dynamics,
Northwestern Polytechnical University, Xi’an, Shaanxi, People’s Republic of China, 710072
E-mail: xcwang91@gmail.com
**Center for Advanced Research in the Engineering Sciences,
Institute for Materials, Manufacturing, and Sustainment,
Texas Tech University, Lubbock, Texas 79415

Received: 19 January 2017; Revised: 2 February 2017; Accepted: 6 February 2017

Abstract
A variety of computational methods that have a wide range of applications to nonlinear dynamical problems are presented in this review. Most of the methods are used in isolated ways in literature. The relationships underlying these methods and the distinctions between some methods are often not clarified. This may lead to confusion in understanding these methods, as well as unnecessary efforts in conducting further researches. In this paper, the methods are arranged in a unified manner. Four groups of distinct methods, namely the weighted residual methods, the finite difference methods, the asymptotic methods and the variational iteration-collocation methods, are introduced. The weighted residual methods comprise the collocation, the finite volume, the finite element, the boundary element methods, etc. Both the corresponding global and local methods are introduced. Depending on whether the problem is expressed in primal or mixed formulation, the weighted residual methods can be divided into primal or mixed methods. The finite difference methods are divided into explicit and implicit classes. Some commonly used methods such as the Runge-Kutta and Newmark method are introduced. The asymptotic methods are among the principal methods of nonlinear analyses. Some representative methods such as the perturbation method, Adomian decomposition method and variational iteration method are presented. The recently proposed variational iteration-collocation method is a kind of semi-analytical iteration method. It is applicable to strongly nonlinear problems and capable of achieving high accuracy and efficiency. All the general formulations of the aforementioned methods are derived. Some numerical examples are also used to illustrate these methods when necessary.

Key words: Weighted residual method, Finite difference method, Asymptotic method, Harmonic balance method, Collocation method, Variational iteration method, Nonlinear problems, Time integrators

1. Introduction
Nonlinear dynamical systems are ubiquitous in science and engineering. Although the linearization is often utilized to simplify the original problem, it is not applicable in some certain cases. Some dynamical phenomena in nonlinear systems are very distinctive, and cannot occur in linear systems (Guckenheimer and Holmes, 1983, Hayashi, 1964, Mickens, 1981, Minorsky, 1947). Sometimes, the consideration of nonlinearity is so crucial to reveal the true dynamical behaviors that the nonlinear problem must be treated as it is. In this review, the principal aim is to introduce the various computational methods for solving nonlinear problems. Hence the attention is focused on the developments of these techniques and their performance, rather than investigating the unique and intriguing features of nonlinear dynamical systems. Among the works to be mentioned in the following, some are from the literature in the field of nonlinear mechanics, the others are contributed by the authors themselves. The contents are divided into four parts, including the weighted residual methods (Section 2, 3), the finite difference methods (Section 4), the asymptotic methods (Section 5), and the variational iteration-collocation methods (Section 6, 7).

With the aid of modern computers, the most straightforward approach to investigate the solution trajectory of a nonlinear dynamical system could be the numerical integration methods. There are literally hundreds of different methods available for this purpose.
numerical integration methods, of which the majority are finite difference methods such as the Runge-Kutta method (Filberg, 1969, Filippi and Gräf, 1986), the Hilber-Hughes-Taylor α method (Hilber, etc., 1977) and the Newmark β method (Newmark, 1959). These methods directly use the definition of a differential and the Taylor’s theorem to discretize the original differential system and provide piecewise approximations in very small time segments. Depending on whether the approximation in each segment is forward or backward, the methods can normally be classified into explicit and implicit methods (Dokainish and Subbaraj, 1989, Subbaraj and Dokainish, 1989). In general, the implicit methods are more stable than their explicit counterparts, but the former usually involves the solution of nonlinear algebraic equations

For nonlinear dynamical systems, long term responses are often desired. There is the well-known class of methods named Galerkin weak-form methods in the area of numerical analysis. As a special version of the Galerkin’s approach, the Harmonic Balance (HB) method (Liu and Dowell, 2005, 2007, Thomas, etc., 2003) is well-developed, to predict the periodic steady-state solutions for autonomous and non-autonomous nonlinear systems. It is a global method that approximates the steady-state responses with trial functions composed of harmonics or sub-harmonics. The HB method is a frequency domain method. Through the balance of the harmonics, the fundamental frequency (ω) and the amplitude of the periodic motion can be obtained. However, it is difficult to rewrite nonlinear terms in the form of harmonics series, even with the use of symbolic computation; thus the resulting algebraic equations are usually very complex. To simplify the symbolic calculations of HB method, the High Dimensional Harmonic Balance (HDHB) method approximates the nonlinear terms using the relationship between its harmonic form and the values of it at a set of collocation points. This approach is much more convenient, but it is also found to be accompanied by the aliasing phenomenon because of the simplification. The HDHB method is later proved to be exactly the same as the Time Domain Collocation (TDC) method (Dai, etc., 2011, 2012, 2014), which is a global collocation method or the spectral collocation method in the time domain.

Besides the periodic responses, the non-periodic responses, including chaotic motions, are more common in nonlinear dynamical systems. Due to the lack of periodicity, the accurate prediction of long term non-periodic responses for the whole domain is difficult. Normally, the approximation is made only in a finite time interval (t_{i+1} - t_i). Since the response is non-periodic, the trial function can be any appropriate function besides the harmonics. Take the Radial Basis Functions (RBF) for example. By assuming the trial function in terms of RBFs, a RBF Collocation method is proposed (Elgohary, etc., 2014, 2015) to solve the initial value and boundary value problems of nonlinear systems. Naturally, to achieve long-term prediction of non-periodic motion of nonlinear systems, the local methods are much more preferred than their global counterparts. By dividing the entire domain into finite but small segments and approximating the solution on each segment separately, the local methods can be more efficient and reliable (Alturi, 2005, Dong, etc. 2014).

To approximate the solution of a nonlinear dynamical system with arbitrary boundary conditions, the methods based on the weighted residual principle, which is more general than the idea of the global Galerkin’s method, can be used. In the works of Atluri (2005) and Dong, etc. (2014), these methods are illustrated clearly and systematically, including the Finite Element Method (FEM), the Finite Volume Method (FVM), the Boundary Element Method (BEM) and the Meshless Local Petrov Galerkin (MLPG) method, etc. These methods are developed using different test functions and trial functions. Among them, the collocation method is the simplest, using the Dirac Delta functions as the test functions. The selection of trial functions is flexible. Depending on the problem, one can use harmonics, polynomials, Radial Basis Functions (RBFs) and Moving Least Square (MLS) functions. But no matter which trial function is used, it will eventually lead to a system of nonlinear algebraic equations (NAEs). Therefore, the problem ultimately becomes how to solve the NAEs. For that, many NAE solvers are proposed. The most popular Newton-like methods mostly involve the inverse of the Jacobian matrix, and can be very sensitive to the initial guess of the solution. Many efforts have been made to solve the NAEs without inverting the Jacobian and without being sensitive to the initial guess. Among there are the scalar-homotopy iterative algorithms developed by Liu, etc. (2009), and Liu and Atluri (2012).

In the area of solving strongly nonlinear dynamical problems, the analytical asymptotic methods are also commonly used, such as the Modified Lindstedt-Poincare Method (Cheung, etc., 1991), the Homotopy Perturbation Method (He, 1999, 2006), the Variational Iteration Method (VIM) (He, 1999, 2000), the Picard Iteration Method (PIM) (Fukushima, 1997), the Adomian Decomposition Method (ADM) (Adomian, 1988), and so on. These methods start from the solution of a linearized problem and iteratively correct the initial guess so that it approaches the real solution of the nonlinear problem. It implies that the solution of the nonlinear problem can be obtained by constructing an iterative formula involving a functional of the original equations, instead of transforming the nonlinear differential equations into NAEs.
to be solved. In the paper of Wang and Atluri (2016a), it is shown that the PIM and ADM can be derived as specific forms of the VIM. However, the VIM is a global method with low numerical accuracy, and the correctional formula of VIM is hard to be implemented, as it needs a lot of complex symbolic computations. To improve the accuracy and to simplify the implementation of VIM, Wang and Atluri (2016a) proposed a Local Variational Iteration Method (LVIM), where some modifications are also made to the derivation of generalized Lagrange multipliers in comparison to the VIM of He (2000). Using the LVIM, the corrected analytical solution can be obtained with much less difficulty and can be easily applied in each local time segments $t_{i+1} - t_i$. Compared to the VIM, The LVIM is much more convenient in the prediction of long term responses of nonlinear systems, and the computational cost is very low. The symbolic calculation in LVIM are still a heavy burden, as the iteration goes on. In each iteration loop, the approximated solution includes more terms, thus the correctional process will be hindered. It is often impossible to repeat the iterative process for multiple times to achieve very accurate results. Although we have made some improvements in LVIM, higher-order approximation of Lagrange multipliers and more iteration steps mean much heavier burden of symbolic manipulations. Hence, the method is usually limited to low-order approximations. As an asymptotic method, the LVIM is still plagued by the difficulties of symbolic manipulations, which is a common problem in the methods which seek purely analytical solutions of nonlinear problems.

In the concepts of the weighted residual methods such as the Harmonic Balance method, the Time Domain Collocation method, and the various collocation methods, they approximate the solution in a weak sense, i.e. the original system is approximated by a set of weighted residual formulae. Thereby, the problem can be solved in a semi-analytical way which is much more convenient than a purely analytical approach. With this in mind, we apply the weighted residual principle on the correctional formula of LVIM. The selection of both test functions and trial functions can be flexible. For example, if the Dirac Delta functions are selected as the test functions, the LVIM will give rise to an iterative formula in the collocation form. The question is, how to turn the iterative formula of LVIM into iterative formula for values of the solution $x(t)$ at collocation points and make it easy to implement.

In order to obtain applicable iterative formulae in weighted residual form, three modifications of the LVIM are derived in this paper. The first modification, denoted here as Algorithm-1, is obtained by transforming the original integral form of LVIM into a differential form. Then it is found that the matrix of generalized Lagrange multipliers can be eliminated in Algorithm-1, using the constraints on the Lagrange multipliers. The resulting formula is very concise. As will be illustrated below, the Algorithm-1 is actually the generalization of the Newton’s iterative formula in a function space of first order differential equations. The Algorithm-1 is totally equivalent with the iterative formula of LVIM since it is obtained by reversible transformations. The other two modifications of LVIM, denoted here as Algorithm-2 and Algorithm-3, are obtained by approximating the generalized Lagrange multipliers with power series and exponential series respectively. Although they are not as concise as the Algorithm-1, the implementations of Algorithm-2 and Algorithm-3 are very easy. Based on these modifications of LVIM, the solution of the nonlinear problem is further approximated using trial functions. For convenience, the test functions are selected as Dirac Delta functions. They provide three kinds of algebraic iterative formulae for values of $x$ at the collocation points. Once the problem is solved in weighted residual forms, that is to say, the values of $x$ at collocation points are solved for, the solutions can be obtained by interpolation. Unlike the conventional collocation method, this approach is free from constructing and solving the NAEs. As will be shown in the following sections, the proposed iterative formulae also illustrate the relationship between the Newton’s method and the variational iteration method. The proposed iterative algorithms in this paper are very accurate and efficient. Although they are local methods, the convergence domain of them are much larger than the finite difference methods. Moreover, the accumulation of computational error is relatively very small because in each segment $t_{i+1} - t_i$, the approximation is made at all the collocated points, while the finite difference methods only approximate the end point. In all, the application of weighted residual principle on Algorithm-1, Algorithm-2, and Algorithm-3 respectively provides a class of powerful tools for solving a system of strongly nonlinear differential equations and for investigating chaos and other non-periodic responses.

2. A simplified beam model and the weighted residual methods

Before treating nonlinear dynamical systems, the static problem of a beam on elastic foundation (Atluri, 2005) is considered in this section. With this problem as an example, the various computational methods that are rooted in the idea of weighted residuals are introduced.
2.1 Problem description

The deflection of a beam on elastic foundation is simply described by a 4th order ordinary differential equation in dimensionless form

\[ y'''' + y - f(x) = 0, \quad x \in \Omega, \]  \hspace{1cm} (2.1.1)

where \( y \) is the normalized vertical displacement (deflection), \( f \) is the normalized distributed load on the beam, and \( \Omega = \{ x \mid 0 < x < 1 \} \) is the domain of interest. The boundary conditions are expressed as

\[ y = \overline{y} \text{ at } S, \quad y' = \overline{y}' \text{ at } S', \quad y'' = \overline{y}'' \text{ at } S'', \quad y''' = \overline{y}''' \text{ at } S''', \]

in which \( S, S', S'', \) and \( S''' \) are the boundaries where \( y \) (displacement), \( y' \) (rotation), \( y'' \) (moment), and \( y''' \) (shear) are prescribed, respectively.

Depending on the boundary conditions, the problem can be either well-posed or ill-posed. For the well-posed problem, the prescriptions of \( y'''' \) and \( y''' \) are mutually disjoint, i.e.

\[ S \cup S''' = S' \cup S'' = \partial \Omega, \quad S \cap S''' = S' \cap S'' = \emptyset, \]  \hspace{1cm} (2.1.2)

with \( \partial \Omega \) being the boundary points \( x = 0,1 \). The well-posed problem is physically consistent with the solid mechanics used to describe the beam, in which the shear force reaction \( y''' \) can be obtained once the deflection \( y \) is prescribed, and the moment reaction \( y'' \) can be obtained once the rotation \( y' \) is prescribed. Thus the well-posed problem is “natural”, meaning that a unique solution exists and is not very sensitive to small variations of boundary values, although the solution does change continuously with the boundary condition. A problem that is not well-posed, i.e. Eq. (2.1.2) does not hold, is termed ill-posed. The ill-posed problems are very common in various engineering applications such as structural health monitoring, system control, and medical imaging.

2.2 Primal methods

2.2.1 Global unsymmetric weak form I

To solve Eq. (2.1.1), a trial function \( u \) is used to approximate the true solution. It results to a residual error in the governing equation, which is

\[ R = u'''' + u - f \neq 0, \quad x \in \Omega, \]  \hspace{1cm} (2.2.1)

Although it is usually difficult to make the residual error \( R \) diminish in general nonlinear problems, it is relatively easier and often feasible to make the weighted residuals equal to zero. With a test function \( v \), the weighted residual form of the governing equation is written as

\[ \int_{\Omega} v R dx = \int_{\Omega} v(u'''' + u - f) dx = 0. \]  \hspace{1cm} (2.2.2)

Conventionally, the weighted residual form is also referred to as the weak form.

Since Eq. (2.2.2) involves the fourth-order derivative of the trial function \( u \), it is required that the third order derivative of \( u \) should be continuous, or briefly, \( u \) should be \( C^3 \) functions. Some commonly used \( C^3 \) functions include harmonics, polynomials, radial basis functions (RBFs), and moving least squares functions (MLSFs) (Dong, 2014). As to the test function \( v \), there is no requirement on its continuity. The selection of trial and test functions is very flexible, leading to various combinations. Among them, two cases are of certain importance in the application to nonlinear dynamical problems and will be further explored in section 3.

Case 1: test functions are selected as Dirac Delta functions, leading to the collocation method;

Case 2: test functions and trial functions are both selected as the same orthogonal basis functions, leading to the Galerkin method.

The collocation method is the simplest and the most straightforward among the various branches of weighted residual method. Based on the types of trial functions, the collocation methods are divided into different classes. Three kinds of commonly used trial functions are listed herein:

Harmonics:

\[ u = \sum_{n=1}^{N} a_n \sin(n\pi x) + \sum_{n=0}^{N} b_n \cos(n\pi x) = b_0 + \sum_{n=1}^{N} [a_n \sin(n\pi x) + b_n \cos(n\pi x)]. \]  \hspace{1cm} (2.2.3)

Polynomials (first kind of Chebyshev polynomials):

\[ u = \sum_{n=0}^{N} T_n(x), \quad \text{where} \quad T_0(x) = 1, \quad T_1(x) = x, \quad \ldots, \quad T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad -1 \leq x \leq 1. \]  \hspace{1cm} (2.2.4)
Radial Basis Function (Gaussian)
\[ u = \sum_{n=1}^{N} a_n \phi(x-x_n) = \sum_{n=1}^{N} a_n e^{-(x-x_n)^T} , \quad \text{where} \quad x_n \quad (n = 1, \ldots, N) \quad \text{are the supporting nodes.} \quad (2.2.5) \]

Note that there are numerous types of trial functions (Buhmann, 2003, Chen, etc., 2013). Each of them possesses unique property and behaves differently in application. In perspective of computational accuracy and efficiency, the orthogonal functions are often preferred (Abramowitz and Stegun, 1968), in that they can bring great convenience in numerical computation as well as symbolic operation using the orthogonal property.

No matter which kind of trial function is used, we can always write the trial function in the form as:
\[ u = \Phi(x) \alpha , \quad (2.2.6) \]
where the columns of \( \Phi(x) \) represent each of the basis functions, and \( \alpha \) is the vector of undetermined coefficients.

As is stated above, when the Dirac Delta functions are used as test functions, i.e., \( v = \delta(x-x_I) \) for a group of pre-selected points \( x_I \quad (I = 1, \ldots, N) \) along the beam, the weighted residual formulation (2.2.2) leads to the point collocation method (Bellomo, etc., 2008):
\[ u''''(x_I) + u(x_I) = \left[ \Phi''''(x_I) + \Phi(x_I) \right] \alpha = f(x_I) . \quad (2.2.7) \]

The boundary conditions can be incorporated in the formulation by collocating points on the boundaries:
\[ u(x_1) = \Phi(x_1) \alpha = \Phi, \quad \text{for} \quad x_1 \in S ; \\
 u'(x_1) = \Phi'(x_1) \alpha = \Phi', \quad \text{for} \quad x_1 \in S' ; \\
 u''(x_1) = \Phi''(x_1) \alpha = \Phi'' , \quad \text{for} \quad x_1 \in S'' ; \\
 u'''(x_1) = \Phi'''(x_1) \alpha = \Phi''' , \quad \text{for} \quad x_1 \in S''' . \quad (2.2.8) \]

By solving the algebraic equations consist of Eqs. (2.2.7) and Eqs. (2.2.8), the coefficients \( \alpha \) of basis functions can be obtained, thus an approximated solution is provided by Eq. (2.2.6).

Other than Dirac Delta function, we can also use Heaviside functions as test functions in Eq. (2.2.2), leading to the finite volume method (Dong, 2014).

2.2.2 Global symmetric weak form

In Eq. (2.2.2), the continuity requirements of the trial function \( u \) and the test function \( v \) are different, hence the weak form in Eq. (2.2.2) is unsymmetric. For some methods, such as the Galerkin-based finite element method, the symmetric weak form is preferred. By integrating Eq. (2.2.2) by parts twice, the symmetric weak form is derived as following:
\[ \left[ n_u v \right]_{\Omega} - \left[ n \dot{u} \dot{v} \right]_{\Omega} + \int_{\Omega} \left( u''''v + uv' \right) dx = 0 . \quad (2.2.9) \]

For Eq. (2.2.9), both the trial and test function are required to be \( C^1 \) continuous. Therefore, it allows us to use simple functions, instead of complex \( C^1 \) continuous functions, to approximate the solution. As aforementioned, the trial and test function can be chosen as harmonics, polynomials and RBFS, etc., leading to the Galerkin method. However, these functions (harmonics, polynomials, RBFS) often lead to a dense, ill-conditioned system of algebraic equations. Suppose that the trial function is
\[ u = \sum_{n=0}^{N} \left[ a_n \sin(n\pi x) + b_n \cos(n\pi x) \right] . \quad (2.2.10) \]

The basis functions in Eq. (2.2.10) are defined in the whole domain and are mostly nonzero for an arbitrary \( x \) in the domain \( \Omega \). This kind of basis functions is classified as global functions. In practice, it is more favorable to use local functions instead of global ones. For instance, we define the trial function as element based Hermite Interpolation as following
\[ u = \sum_{n=1}^{N} h_n(x) , \quad \text{with the Hermite Interpolation function} \quad h_n(x) \quad \text{defined in} \quad \Omega_n . \quad (2.2.11) \]

Then only one local basis function \( h_n(x) \) need to be considered for \( x \in \Omega_n \), where \( \Omega_n \quad (n = 1, \ldots, N) \) are non-overlapping subdomains in \( \Omega \). By adopting the local trial functions, the weak form (2.2.9) leads to the finite element method (Zienkiewicz and Morice, 1971, Dong and Atluri, 2011, Zielinski and Herrera, 1987), where the resulted system
of algebraic equations is sparse.

2.2.3 Global unsymmetric weak form II

Further integrating Eq. (2.2.9) by parts for twice yields another unsymmetric weak form:

\[ [n, u''']_{\partial \Omega} - [n, u''']_{\partial \Omega} + [n, u''']_{\Omega} + \int_{\Omega} (v''' + v) dx - \int_{\Omega} f dx = 0 \theta, \]  (2.2.12)

in which there is no continuity requirement for the trial function. If one can find the Green’s function for

\[ v''' + v = \delta(x - \eta), \]  (2.2.13)

i.e. the fundamental solution of an infinite beam on an elastic foundation, Eq. (2.2.12) will reduce to the boundary integral equation

\[ C(\eta)u(\eta) - \int_{\partial \Omega} f dx + (n, u''')_{\partial \Omega} - (n, u''')_{\partial \Omega} + (n, u''')_{\partial \Omega} = 0, \]  (2.2.14)

where

\[ C(\eta) = \begin{cases} 1, & \eta \notin \partial \Omega \\ 0.5, & \eta \in \partial \Omega \end{cases}. \]

In Eq. (2.2.14), the trial function only appears in the boundary terms, thus it leads to the boundary element method (Okada, etc., 1988, Atluri, 2004).

2.2.4 Local weak forms

In the previously developed methods, the global weak form is considered in the whole domain \( \Omega \). By dividing \( \Omega \) into subdomains \( \Omega_i \) (over-lapping or non-over-lapping), local weak forms are defined as

\[ \int_{\Omega_i} vR dx = \int_{\Omega_i} v(u''' + u - f) dx = 0. \]  (2.2.15)

The corresponding local boundary conditions are set by the continuous requirement of \( u \) on the local boundaries \( \partial \Omega_i \).

If \( \Omega_i \) are non-over-lapping subdomains, the local boundary conditions can be expressed as

\[ u(\partial \Omega_i^+) = u(\partial \Omega_i^-), \quad u'(\partial \Omega_i^+) = u'(\partial \Omega_i^-), \quad u'''(\partial \Omega_i^+) = u'''(\partial \Omega_i^-), \quad u''''(\partial \Omega_i^+) = u''''(\partial \Omega_i^-). \]  (2.2.16)

In the local weak forms, the trial and test functions can be much simpler than those in the global weak forms (2.2.1). As a subdomain is much smaller than the original domain, it is relatively very easy to obtain highly accurate local solutions, even when the trial functions are composed by a small group of basis functions. In addition to that, the local weak forms are also more efficient in computation, because the resulted system of algebraic equations is partitioned into blocks.

The local weak forms can also be rewritten into symmetric and unsymmetric forms, therefore it enables us to develop local collocation, local finite volume, local Galerkin, local finite element, and local boundary element method (Atluri, etc., 1999, 2000, 2002, Zhang, etc., 2014).

Herein, it should be emphasized that the local methods are different from the global methods using local trial functions. In the former, the weak forms are satisfied in each local subdomain, and the continuity of the solution is ensured by the local boundary conditions; while in the latter, the weak form is satisfied in the whole domain, and the local trial function needs to be selected carefully to satisfy the continuous requirement a-priori.

2.3 Mixed methods

In the primal problem description, the governing equation (2.2.1) is of 4th order. This brings much inconvenience to the computation, for that the trial function has to meet high order continuous requirement. It not only makes the trial function quite complex, but also easily leads to ill-conditioned matrix by including higher order derivatives of the trial function. To decrease the order of the governing equation (2.2.1), extra variables are introduced. By adding the moment \( m \), Eq. (2.2.1) is rewritten as
or equivalently in matrix form as

\[
\begin{align*}
A P'' + B P - g &= 0, \\
C Z' + D Z - h &= 0,
\end{align*}
\]

where

\[
A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \quad g = \begin{bmatrix} 0 \\ f \end{bmatrix}, \quad P = \begin{bmatrix} u \\ m \end{bmatrix}.
\]

Considering a vector test function \( V \), the weak form of Eq. (2.3.2) can be obtained

\[
\int_{\Omega} (A P'' + B P - g) V \, dx = 0.
\]

The required continuity of the trial function is reduced from \( C^3 \) to \( C^1 \).

By introducing displacement \( u \), rotation \( \theta \), moment \( m \) and shear \( q \) as independent variables, the original governing equation is expressed as

\[
\begin{align*}
\dot{u} &= \theta \\
\dot{\theta} &= m \\
\dot{m} &= q \\
\dot{q} &= u - f
\end{align*}
\]

The corresponding matrix form is

\[
C Z' + D Z - h = 0,
\]

where

\[
C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad g = \begin{bmatrix} 0 \\ 0 \\ 0 \\ f \end{bmatrix}, \quad Z = \begin{bmatrix} u \\ \theta \\ m \\ q \end{bmatrix}.
\]

The weighted residual weak form can be written as

\[
\int_{\Omega} (C Z' + D Z - h)^T W \, dx = 0.
\]

Now the continuous requirement of trial functions is reduced to \( C^0 \), that is to say, the solution of the problem considered can be approximated by linear interpolation.

Based on the mixed forms, one can develop various mixed weighted residual methods (Dong and Atluri, 2011, Lee and Pian, 1978), in reference to the primal methods introduced in subsection 2.2.

### 3. Application of weighted residual methods in nonlinear dynamical problems

Two classical problems in nonlinear dynamics are considered here: the Duffing oscillator and the Van Der Pol oscillator. The governing equations of them are both expressed as 2nd order differential equations.

**Duffing Oscillator:**

\[
x'' + \mu x' + \varepsilon x + \xi x^3 = f(t),
\]

**Van der Pol Oscillator:**

\[
x'' - \mu (1 - x^2) x' + x = f(t),
\]

where \( x \) is a variable dependent on time \( t \), while \( \mu, \varepsilon, \xi \) are system parameters and \( f(t) \) is external forces.

Due to the existence of nonlinear terms (cubic nonlinear term and nonlinear damping term respectively) in Eqs. (3.1 & 3.2), it is barely possible to obtain analytical solutions in closed forms. Therefore, the numerical methods and approximate approaches are often used to reveal the nonlinear dynamical behaviors. In this section, the weighted residual methods in the time domain are employed, from which one can accurately approximate the true solution and investigate some characteristics of the nonlinear dynamical systems in a semi-analytical way.
3.1 Transient motions

The transient motion of a nonlinear system is usually irregular and fast-changing. It occurs before the system settles down to a steady state, such as a fixed point or a limit cycle. To capture the transient motion of a nonlinear system for long term is difficult. Normally, it is achieved by dividing the time domain into many small intervals and approximating the solution in each interval successively.

3.1.2 Collocation method

Denoting the local time interval as $\Omega$, the local weak form of Eq. (3.1) is written as

$$
\int_{\Omega_j} v Rdt = \int_{\Omega_j} v(u'' + \mu u' + \varepsilon u + \xi u^3 - f)dt = 0,
$$

where $v$ is test function, $u$ is trial function, and $R$ stands for the residual error.

Let the test function $v$ be Dirac Delta functions: $v = \delta(t - t_m), t_m \in \Omega, m = 1, 2, ..., M$. Eq. (3.1.1) gives rise to the collocation method

$$
u''(t_m) + \mu u'(t_m) + \varepsilon u(t_m) + \xi u(t_m)^3 - f(t_m) = 0,
$$

The local initial conditions are satisfied by imposing

$$u(t_0) = x(t_0), \quad u'(t_0) = x'(t_0),$$

where $t_0$ is the initial time in the local interval.

Suppose the trial function $u$ is composed of a set of basis functions (harmonics, polynomials, RBFs, etc.)

$$u(t) = \sum_{n=1}^{N} \alpha_n \phi_n(t) = \Phi(t) \alpha.$$

Substituting it into Eq. (3.1.2), we have

$$
\left[ \Phi''(t_m) + \mu \Phi'(t_m) + \varepsilon \Phi(t_m) \right] \alpha + \xi \left[ \Phi(t_m) \alpha \right]^3 - f(t_m) = 0,
$$

where $m = 1, 2, ..., M$. By solving the system of nonlinear algebraic equations given in (3.1.5), the coefficients $\alpha$ can be determined. For the convenience of computation, Eq. (3.1.5) can be rewritten as

$$
\left[ \Phi''(t_m) + \mu \Phi'(t_m) + \varepsilon \Phi(t_m) \right] \alpha - \mu \left[ 1 - \left[ \Phi(t_m) \alpha \right] \right] \left[ \Phi'(t_m) \alpha \right] - f(t_m) = 0
$$

from which the values of collocation points $u(t_m)$ are determined first. The harmonic coefficients can be determined afterwards using the transformation

$$
\alpha = \left[ \Phi(t_1) \right]^{-1} \begin{bmatrix} u(t_1) \\ u(t_2) \\ \vdots \\ u(t_M) \end{bmatrix}.
$$

Similar to the preceding process introduced to solve Duffing equation, the approximated solution of Van der Pol oscillator can be obtained by solving

$$
\left[ \Phi''(t_m) + \Phi(t_m) \right] \alpha = \left[ 1 - \left[ \Phi(t_m) \alpha \right] \right] \left[ \Phi'(t_m) \alpha \right] - f(t_m) = 0
$$

in each local time interval $\Omega$. The number of collocation points $M$ can be equal to or more than the number of basis functions $N$.

3.1.3 Galerkin Method

Let the test and the trial function in Eq. (3.3) be the same set of orthogonal functions $\Phi(t)$ in the local time domain $\Omega$, then

$$v(t) = u(t) = \sum_{n=1}^{N} \alpha_n \phi_n(t) = \Phi(t) \alpha.$$

[DOI: 10.1299/mer.17-00040] © 2017 The Japan Society of Mechanical Engineers
The local weak form becomes

$$\int_{\Omega} \Phi(t)a \left[ \Phi''(t) + \mu \Phi'(t) + \varepsilon \Phi(t) \right] dt = 0 \, . \quad (3.1.10)$$

Using the orthogonal property of $\Phi(t)$, saying

$$\int_{\Omega} \Phi_i(t)\Phi_j(t) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \, , \quad (3.1.11)$$

Eq. (3.1.10) leads to a system of equations

$$\int_{\Omega} \Phi_i(t) \left[ \Phi''(t) + \mu \Phi'(t) + \varepsilon \Phi(t) \right] dt = 0 \, , \quad (3.1.12)$$

where $n = 1, 2, ..., N$. For the existence of cubic nonlinear term, the integration in Eq. (3.1.12) could be troublesome when $N$ is relatively large. In order to explicitly express Eq. (3.1.12) as algebraic equations of the coefficients $a$, Mathematica can be used to handle with the complex symbolic manipulation.

For demonstration, the Chebyshev polynomials of the first kind $T_n(\tau)$ are used as basis functions (Bai, and Junkins, 2011, Woollands, etc., 2015). Because the orthogonality of $T_n(\tau)$ is valid for $-1 \leq \tau \leq 1$, time $t$ is replaced by the rescaled time $\tau$, where

$$\tau = 2(t-t_0)/(t_f-t_0)-1 \, . \quad (3.1.13)$$

The final algebraic form of Galerkin method is much more complex than collocation method. Normally, the explicit expression of Eq. (3.1.12) can not be written in matrix and vector form. Thus, it is often laborious to solve Eq. (3.1.12), even with the help of Mathematica and MATLAB. However, the solution of Galerkin method is usually more accurate than that of collocation method when the same number of basis functions are used.

3.2 Periodic motions

In nonlinear dynamics, the periodic motion is related to the limit cycle oscillation (LCO). As one of the most commonly observed phenomenon in nonlinear dynamical systems, the existence of LCOs as well as their shape and period information is of great interest in both theoretical studies and engineering applications.

3.2.1 Collocation method

To approximate the periodic solution, the trial function is selected as periodic basis functions such as harmonics with unknown frequency $\omega$.

$$u(t) = b_0 + \sum_{n=1}^{N} \left( a_n \sin n\omega t + b_n \cos n\omega t \right) \, . \quad (3.2.1)$$

The weak form of Eq. (3.1) is written as

$$\int_{\Omega} v dt = \int_{\Omega} v (u'' + \mu u' + \varepsilon u + \xi u - f) dt = 0 \, , \quad (3.2.2)$$

with $v$ being Dirac Delta functions, and $\Omega$, being a time interval, of which the length is the period $T$ of the limit cycle oscillation. Substituting Eq. (3.2.1) into Eq. (3.2.2), we get a system of nonlinear algebraic equations (NAEs) in matrix form

$$\left[ \omega^2 E(\omega) A^2 + \mu \omega E(\omega) A + \varepsilon E(\omega) Q + \xi [E(\omega) Q]^3 \right] - F = 0 \, , \quad (3.2.3)$$

where
The coefficients $Q$ and frequency $\omega$ are unknowns to be determined.

To solve the NAEs in (3.2.3), Newton-Raphson (NR) iteration method is mostly used because of its simplicity and high convergence speed. However, the NR method is sensitive to the initial values used to initialize the iteration, and it needs to compute the Jacobian matrix and its inverse. In the case where proper initial values can not be obtained, or the Jacobian matrix is ill-conditioned, the NR method may fail to work. To remedy that, one may use the vector/scalar homotopy method as alternative (Liu, etc. 2009, 2011, 2012), which is insensitive to initial values and free of inverting the Jacobian matrix.

### 3.2.2 Galerkin method (harmonic balance method)

Let both the trial and test function be composed of the same harmonics:

$$v(t) = u(t) = b_0 + \sum_{n=1}^{N} (a_n \sin nt + b_n \cos nt).$$  \hfill (3.2.4)

The weak form (3.2.2) can be discretized as

$$\int_{\Omega} (u'' + \mu u' + \xi u - f) dt = 0, \quad n = 1, 2, ..., N.$$  \hfill (3.2.5)

To solve Eq. (3.2.5), the coefficient $a_1$ or any other coefficient may be prescribed as zero. In this way, there is $2N+1$ equations for $2N+1$ unknowns, thus the system is solvable.

The weak form of Galerkin method is described in Eq. (3.2.5). The corresponding NAEs can also be obtained by substituting the trial function in Eq. (3.2.4) into

$$u'' + \mu u' + \xi u - f = 0,$$  \hfill (3.2.6)

and eliminating the coefficients of each harmonics $\sin nt$ and $\cos nt$ separately. This approach is exactly the harmonic balance method (Liu and Dowell, 2004, 2005, 2007).

It should be noted that the collocation method and the harmonic balance method may produce approximated solutions that are physically meaningless. To relieve that, more collocation points or harmonics may be added to the approximations. Normally, the more harmonics are included in the trial function, the more accurate approximation can be obtained. However, as the number of harmonics increases, the resulting system of NAEs becomes more complex and thus more computational resource will be spent.

Compared to the purely numerical methods, the advantages of weighted residual methods on predicting the periodic motion includes:

(i) The periodic solution is approximated semi-analytically, in which the frequency and the shape of the LCOs are explicitly provided.

(ii) Both the stable and unstable LCOs can be predicted by weighted residual methods, while the numerical integration can only reveal the stable ones.

(iii) It is more convenient to use weighted residual methods to study the evolution of LCOs for varying system parameters, than using numerical methods.

### 4. Finite difference methods

Consider a second-order nonlinear dynamic system, which is recast into a system of first-order ordinary differential equations (ODEs):
This system can be further rewritten as matrix form

\[
x' = g(x, f, t),
\]

where \( x = [x_1, x_2]^T \), \( f \) is the external force applied to the system. For a specified set of initial conditions, Eq. (4.2) can be solved with various implicit and explicit numerical integration methods.

4.1 Explicit methods:

In explicit methods, the future unknown state is directly expressed in terms of the currently-unknown state with an explicit formula. Euler method is the simplest explicit method

\[
x(t + \Delta t) = x(t) + \Delta t g(x(t), f(t), t),
\]

which is a first-order Taylor series expansion in the time domain.

Another explicit method is the second order central difference method:

\[
\begin{align*}
x_2(t + \Delta t) &= x_2(t - \frac{\Delta t}{2}) + \Delta t g_2(x(t), f(t)) \\
x_1(t + \Delta t) &= x_1(t) + \Delta t x_2(t + \frac{\Delta t}{2})
\end{align*}
\]

This method is presented by (Belytschko, etc., 1976), and has been widely used for transient finite element analyses of large scale nonlinear structures, such as crash simulation of automobiles.

The family of Runge-Kutta (RK) method is considered as the most widely used explicit methods for numerical integration. The first order RK method is simply the forward Euler method in Eq. (4.1.1). Among the various RK methods, the classical 4th-order RK method is the most commonly used. It evaluates the future unknown state by adding four increments to the currently-known system state:

\[
x(t + \Delta t) = x(t) + \frac{\Delta t}{6} (k_1 + 2k_2 + 2k_3 + k_4),
\]

where

\[
\begin{align*}
k_1 &= F(t, x(t)) \\
k_2 &= F(t + \frac{\Delta t}{2}, x(t) + \frac{\Delta t}{2} k_1) \\
k_3 &= F(t + \frac{\Delta t}{2}, x(t) + \frac{\Delta t}{2} k_2) \\
k_4 &= F(t + \Delta t, x(t) + \Delta t k_3)
\end{align*}
\]

Against to the fixed step size in RK4, the adaptive step-size 4th-order RK methods are developed by (Fehlberg, 1969) and are now known as the Runge-Kutta-Fehlberg (RKF) methods. Several higher order adaptive RKF methods (Filippi and Graef, 1986), are widely used for very high accuracy applications such as orbit propagation problems (Fox, 1984).

4.2 Implicit methods:

Implicit methods approximate the unknown future state in a backward manner, which normally results in a set of linear or nonlinear algebraic equations. The backward Euler method is an illustration of this concept:

\[
x(t + \Delta t) = x(t) + \Delta t g(x(t) + \Delta t f(t + \Delta t), f(t + \Delta t), t + \Delta t).
\]

In (Newmark, 1959), Newmark introduces the Newmark-\( \beta \) method based on the extended mean value theorem, which is among the most widely-used implicit methods for numerically evaluating the dynamical response of engineering structures.

A generalization of the Newmark-\( \beta \) method is introduced by (Hilber, etc., 1977):

\[
\begin{align*}
x_1(t + \Delta t) &= x_1(t) + \Delta t x_2(t) + \frac{1}{2} \Delta t^2 [1 - 2\beta] a(t + \Delta t) + 2\beta a(t + \Delta t)] \\
x_2(t + \Delta t) &= x_2(t) + \Delta t [(1 - \gamma) a(t) + \gamma a(t + \Delta t)] \\
a(t + \Delta t) &= (1 + a) x_1(t + \Delta t) - a x'_1(t)
\end{align*}
\]

with \( \gamma = \frac{1 - 2\alpha}{2}, \quad \beta = \left(\frac{1 - 2\alpha}{2}\right)^2, \quad \alpha \in [-\frac{1}{3}, 0] \). This method is known as Hilbert-Hughes-Taylor or HHT-\( \alpha \) method.
For $\alpha = 0$, the HHT-\(\alpha\) method collapses to the well-known Newmark method.

For the above numerical methods, the step size plays an important role in the computational accuracy. Normally the step size must be relatively small to obtain an accurate solution for nonlinear problem. If an implicit method is used, it will be necessary to solve a set of linear/nonlinear algebraic equations, which brings in extra computational burden.

5. Asymptotic methods

One way to obtain analytical approximated solutions of nonlinear problems in time domain is using asymptotic method (Bogoliubov and Mitropolsky, 1961, He, 2006). It is often used to study the local behavior of a system in the neighborhood of a nominal solution. The most well-known asymptotic method for solving nonlinear differential equations should be the perturbation method (Hayashi, 1964, Nayfeh, 1981).

In earlier times, the use of the perturbation method is limited to the astronomical calculations. The scholars such as Laplace and Lagrange used it to investigate the perturbed motion of planets around the sun. Then Urbain Le Verrier successfully predicted the existence of the planet Neptune with this method. After this remarkable event, the application of perturbation method was gradually broadened to a more general field of nonlinear mechanics.

5.1 Perturbation method

This method assumes that the nonlinear terms in the equations are associated with small parameters. In that case, the solution can be obtained by adding small corrections to the nominal solution that is readily solved for. To explain the details of perturbation method, a second order differential equation is taken for instance.

\[
x'' + \varepsilon x + \mu x^2 = 0.
\]  (5.1.1)

Since $\mu$ is a small parameter, the solution of linear equation $x'' + \varepsilon x = 0$ is considered close enough to the true solution of Eq. (5.1.1), and is denoted as $x_0$. Obviously, a nonzero residual occurs by substituting $x_0$ into Eq. (5.1.1).

\[
R_0 = x_0'' + \varepsilon x_0 + \mu x_0^3 = O(\mu).
\]  (5.1.2)

To eliminate the residual $R_0$, a small correction $x_1$ is added to $x_0$, leading to

\[
R_1 = (x_0'' + x_1'') + \varepsilon (x_0 + x_1) + \mu (x_0 + x_1)^3 = x_1'' + \varepsilon x_1 + \mu x_1^3 + \mu (3x_0^2 x_1 + 3x_0 x_1^2 + x_1^3).
\]  (5.1.3)

Let $x_1'' + \varepsilon x_1 + \mu x_1^3 = 0$, we have

\[
R_0 = \mu (3x_0^2 x_1 + 3x_0 x_1^2 + x_1^3) = O(\mu^2).
\]

Then by adding another correction $x_2$ to the previously obtained solution, the residual of Eq. (5.1.1) is obtained as

\[
R_2 = x_0'' + x_1'' + x_2'' + \varepsilon (x_0 + x_1 + x_2) + \mu (x_0 + x_1 + x_2)^3
\]

\[
= x_2'' + \varepsilon x_2 + 3\mu x_2 x_1 + \mu (3x_0 x_1^2 + x_1^3 + 3x_0^2 x_2 + 6x_0 x_1 x_2).
\]  (5.1.4)

Let $x_2'' + \varepsilon x_2 + 3\mu x_2 x_1 = 0$, $R_2$ becomes

\[
R_2 = \mu (3x_0 x_1^2 + x_1^3 + 3x_0^2 x_2 + 6x_0 x_1 x_2 + 3x_0^2 x_2 + 3x_0 x_2^2 + 3x_0^2 x_2 + x_2^3) = O(\mu^4).
\]  (5.1.5)

Repeating this process leads to the solution of Eq. (5.1.1) in series form

\[
x(t) = x_0(t) + x_1(t) + x_2(t) + ...,
\]  (5.1.6)

where $x_0$, $x_1$, $x_2$, ... are obtained by solving

\[
\begin{align*}
x_0'' + \varepsilon x_0 &= 0 \\
x_1'' + \varepsilon x_1 + \mu x_1^3 &= 0 \\
x_2'' + \varepsilon x_2 + 3\mu x_2 x_1 &= 0 \\
&\vdots
\end{align*}
\]  (5.1.7)

The preceding description of the perturbation method looks concise and clear. However, by proceeding in this way, a difficulty may often be encountered in the form of “secular terms”, where the convergence of the solution (5.1.6) is destroyed by infinitely growing terms as $t \to \infty$. In the following, Eq. (5.1.1) is used as an example to illustrate the appearance of secular terms.

By solving Eq. (5.1.7), the nominal solution $x_0$ is obtained first

\[
x_0 = C_1 \cos(\sqrt{\varepsilon} t) + C_2 \sin(\sqrt{\varepsilon} t).
\]  (5.1.8)

Hence the second equation becomes
\[ x_i'' + \epsilon x_i + \frac{\mu}{4} \left[ 3(C_i + C_i C_i^2) \cos(\sqrt{\epsilon}t) + (C_i^3 - 3C_i^2 C_i) \cos(3\sqrt{\epsilon}t) + 3(C_i^2 C_i + C_i^2) \sin(\sqrt{\epsilon}t) + (3C_i^2 C_i^2 - C_i^3) \sin(3\sqrt{\epsilon}t) \right]. \]

Since it is a linear differential equation, its solution \( x_i \) can be obtained by the superposition of multiple simpler solutions. Specifically, we focus on the solution \( x_{i,1} \) of
\[ x_{i,1}'' + \epsilon x_{i,1} + \cos(\sqrt{\epsilon}t) = 0, \]
which is solved as
\[ x_{i,1} = C_i \cos(\sqrt{\epsilon}t) + C_2 \sin(\sqrt{\epsilon}t) - \frac{\cos(\sqrt{\epsilon}t) \cos(2\sqrt{\epsilon}t) - 2\tau \sqrt{\epsilon} \sin(\sqrt{\epsilon}t) - \sin(\sqrt{\epsilon}t) \sin(2\sqrt{\epsilon}t)}{4\epsilon}. \]

It can be seen the secular term appears as \(-2\epsilon \sqrt{\epsilon} \sin(\sqrt{\epsilon}t)\).

The reason why secular terms appear is as follows. By solving the first equation, the frequency of \( x_0 \) is prescribed as \( \sqrt{\epsilon} \), while the true periodic solution of the nonlinear system, if there is any, may not be of the same frequency. A particularly convincing example is given by the expanding a simple periodic function \( \sin(\tau \omega_0 + \delta t) \)
\[ \sin(\tau \omega_0 + \delta t) = \sin(\tau \omega_0 + \delta t) \cos(\frac{\delta^2 t^2}{2!} - \sin(\tau \omega_0) - \cdots), \]
where \( \delta \) is a small variation of the frequency.

To eliminate the secular terms, a rescaled time \( \tau = \omega t \) is used in Eq. (5.1.1), with \( \omega \) being the frequency of the true solution, which is unknown. Hence Eq. (5.1.1) becomes
\[ \omega^2 x'' + \epsilon x + \mu x^3 = 0, \]
and Eq. (5.1.7) is rewritten as
\[
\begin{align*}
\omega_0^2 x_0'' + \epsilon x_0 &= 0 \\
\omega_0^2 x_1'' + \omega_0^2 x_0'' + \epsilon x_1 + \mu x_0^3 &= 0 \\
\omega_0^2 x_2'' + \omega_0^2 x_1'' + \omega_0^2 x_0'' + \epsilon x_2 + 3\mu x_0^2 x_1 &= 0 \\
&\vdots
\end{align*}
\]
(5.2.22)

Since \( x_i(\tau + 2\pi) = x_i(\tau) \), we have the following conditions to help solving Eq. (5.1.22), namely
\[ x_0(0) = A, \quad x_{i,1}(0) = 0, \quad \dot{x}_i(0) = 0, \quad i = 0, 1, 2, \cdots. \]
The frequency is then determined by \( \omega^2 = \omega_0^2 + \omega_0^2 + \omega_0^2 + \cdots \). This is the modified perturbation method.

At last, it should be noted that the convergence of perturbation method is not guaranteed. Although a process of solving the given example is demonstrated above, it is more like as a guideline rather than a presentation of rules. In practice, the application of perturbation method could be very flexible and experience is valuable.

### 5.2 Adomian decomposition method

This method is essentially another version of perturbation method, but it introduces the notion of Adomian polynomials and presents the iterative formulas for solving nonlinear systems in a particular form (Adomian, 1988, Wazeaz, 2000, Al-Sawalha, etc., 2009, Gosh, etc., 2007). A brief description of the Adomian decomposition method (ADM) is made as the following.

Considering a one dimensional nonlinear dynamical system \( Lx + Rx + Nx = f(t) \), where \( L \) is the highest-ordered derivative, \( R \) and \( N \) are the linear and nonlinear operators on \( x \), it can be rewritten as
\[ Lx = f(t) - Rx - Nx. \]
(5.2.1)
The solution is supposed to be found as a series of functions
\[ x = \sum_{n=0} \mathcal{A}_n. \]
(5.2.2)
Substituting Eq. (5.2.2) into Eq. (5.2.1), we have
\[ Lx = f(t) - Rx - N(x_0 + x_1 + x_2 + \cdots). \]
(5.2.3)
The nonlinear term can be equivalently expanded into Taylor series
\[ N(x) = N(x_0) + N'(x_0)(x_1 + x_2 + \cdots) + \frac{N''(x_0)}{2!}(x_1 + x_2 + \cdots)^2 + \cdots. \]
(5.2.4)
After rearrangement, it is expressed as
From Eq. (5.2.5), the Adomian polynomials are defined as
\[ A_0 = N(x_0), \quad A_1 = N'(x_0) x_1, \quad A_2 = N''(x_0) x_1^2 + \frac{N'''(x_0)}{2!} x_1^3, \quad \text{etc}. \]

Therefore, Eq. (5.2.3) can be decomposed as
\[
\begin{align*}
Lx_0 &= f(t) - Rx_0 \\
Lx_{n+1} &= -Rx_n - A_n,
\end{align*}
\]  

where \( n = 0, 1, 2, \ldots \). By solving Eq. (5.2.6), analytical solutions can be obtained for the nonlinear system. However, the convergence of the Adomian series is not guaranteed. Although it is often declared that the ADM is not restricted to small assumptions or weak nonlinearity, mostly it works only when the nonlinear term is relatively not very significant.

It is well known that a higher order differential equation can always be transformed into a system of first order differential equations. For example, the equation
\[
\frac{d^n x}{d\tau^n} = f(d^{n-1} x, d^{n-2} x, \ldots, x, \tau)
\]  

can be rewritten as a system of equations like
\[
\frac{dx_0}{d\tau} = x_1, \quad \frac{dx_1}{d\tau} = x_2, \ldots, \quad \frac{dx_{n-1}}{d\tau} = f\left(\frac{dx_{n-1}}{d\tau}, \ldots, x_1, x_0, \tau\right)
\]  

by introducing the variables \( x_0, \ldots, x_{n-1} \). Therefore, the nonlinear ordinary differential equations can be expressed in a general form as
\[
\frac{dx}{d\tau} = F(x, \tau), \quad \text{where} \quad x = (x_1, x_2, \ldots)^T, \quad F = (f_1, f_2, \ldots)^T, \quad \tau \in [t_0, t_1],
\]

where, here onwards, a bold symbol indicates a vector or a matrix, and for brevity, the differential operator \( d / d\tau \) is denoted as \( L \) in the following sections. In the preceding equation, \( F(x, \tau) \) is a nonlinear function of the state vector \( x \) and the independent variable \( \tau \).

This equation can be recast as
\[
x = x(t_0) + L^{-1} F(x, \tau),
\]

with \( x(t_0) \) being the initial condition.

The innovative part of ADM is that it approximates both the solution and the nonlinear part as sequences of functions and introduces the Adomian polynomials so that the original problem will be solved progressively.

\[
x = \sum_{n=0}^{\infty} x_n, \quad F(x, \tau) = \sum_{n=0}^{\infty} A_n(x_0, x_1, \ldots, x_n).
\]

This enables us to rewrite the Eq. (5.2.10) as
\[
\sum_{n=0}^{\infty} x_n = x(t_0) + L^{-1} \sum_{n=0}^{\infty} A_n(x_0, x_1, \ldots, x_n).
\]

The solution can thus be derived recursively in the following way
\[
\overline{x}_i = x(t_0) + L^{-1} A_i,
\]
\[
\overline{x}_{i+1} = L^{-1} A_{i+1}.
\]

The Adomian polynomials \( A_n \) are generated by simply rearranging the Taylor series expansion of \( F(x, \tau) \) about \( \overline{x}_0 = x(t_0) \). They are expressed as
\[
A_0 = F(\overline{x}_0), \quad A_1 = F'(\overline{x}_0)\overline{x}_1, \quad A_2 = F'(\overline{x}_0)\overline{x}_2 + F''(\overline{x}_0)\frac{\overline{x}_1^2}{2!}, \ldots
\]
where

\[
F'(x_0) = \frac{\partial F}{\partial x_0} = \begin{bmatrix}
\frac{\partial F_1}{\partial x_{1,0}} & \frac{\partial F_1}{\partial x_{2,0}} & \cdots \\
\frac{\partial F_2}{\partial x_{1,0}} & \frac{\partial F_2}{\partial x_{2,0}} & \cdots \\
\vdots & \vdots & \ddots
\end{bmatrix}, \quad \text{and} \quad F''(x_0) = \frac{\partial^2 F}{\partial x_0^2} = \begin{bmatrix}
\frac{\partial^2 F_1}{\partial x_{1,0}^2} & \frac{\partial^2 F_1}{\partial x_{1,0} \partial x_{2,0}} & \cdots \\
\frac{\partial^2 F_2}{\partial x_{1,0}^2} & \frac{\partial^2 F_2}{\partial x_{1,0} \partial x_{2,0}} & \cdots \\
\vdots & \vdots & \ddots
\end{bmatrix}.
\]

5.3 Picard iteration method

Picard iteration method takes a very simple form. However, its application is very limited, since it needs to integrate the nonlinear terms in each iteration step, which could be very difficult to implement. Compared to Adomian’s method, the Picard iteration lacks the ease of computation and the ability to solve a wide class of equations. But by combining with other computational techniques, it is still possible to make advances. For example, the MCPI method (Woolland, Younes and Junkins, 2015) combines the Chebyshev polynomials with the Picard’s method and is applied to the two body gravitational integration problem. It is shown that the integration process of Picard’s method becomes very simple and the method achieves high accuracy and efficiency.

Consider an initial value problem (IVP) governed by a system of first-order differential equations

\[
L x = F(x, \tau), \quad \tau \in [t_0, t], \quad x(t_0) = [x_1(t_0), x_2(t_0), \ldots]^T.
\]

It is equal to the associated integral equations

\[
x(t) = x(t_0) + \int_{t_0}^{t} F[\tau, x(\tau)] d\tau, \quad \tau \in [t_0, t].
\]

The PIM solves this problem in a recursive way by constructing a series of approximating functions. The process works as

1. Give an initial guess of the solution \( x_0(\tau) \) that satisfies the initial condition \( x_0(t_0) = x(t_0) \).

2. Substitute it into the recursive formula that holds

\[
x_{n+1}(t) = x(t_0) + \int_{t_0}^{t} F[\tau, x_n(\tau)] d\tau,
\]

for \( n \geq 0 \).

5.4 Variational iteration method

The VIM can solve a large class of nonlinear problems, including both the systems expressed as either ordinary or partial differential equations. The foundation of VIM is the use of generalized Lagrange multipliers, as proposed by Inokuti, etc. (1978). This method is similar to the Newton’s approach for solving nonlinear algebraic equations, except that it is extended to the function space. Originally, it was used to correct the solution of a linearized problem at a certain point so as to obtain a more precise approximation. By using this approach at an arbitrary point, we can obtain the corrected approximation for the whole domain, and thus it leads to the VIM for the corrected analytical solution for the nonlinear dynamical system.

5.4.1 Formulation of VIM

Consider the following general nonlinear system

\[
L x = F(x, \tau), \quad \tau \in [t_0, t],
\]

where \( L \) is the first order differential operator and \( F \) is a nonlinear operator. The solution of this system can be approximated with an initial approximation \( x_n(t) \) and the correctional formula as

\[
x_{n+1}(t) = x_n(t) + \int_{t_0}^{t} \lambda(\tau) [L x_n(\tau) - F[x_n(\tau), \tau)] d\tau,
\]

[DOI: 10.1299/mer.17-00040] © 2017 The Japan Society of Mechanical Engineers
where $\lambda(\tau)$ is a matrix of Lagrange multipliers which are yet to be determined. Suppose $\Pi[x(\tau), \lambda(\tau)]$ is a vector function of $x(\tau)$ and $\lambda(\tau)$, where $t_0 \leq \tau \leq t$.

$$
\Pi[x(t), \lambda(t)] = x(\tau)|_{t=t_0} + \int_{t_0}^{t} \lambda(\tau)\{Lx(\tau) - F[x(\tau), \tau]\}d\tau.
$$

(5.4.3)

Let $\hat{x}(\tau)$ be the exact solution of $Lx(\tau) = F[x(\tau), \tau]$. Naturally, it satisfies the expression

$$
\Pi[\hat{x}(\tau), \lambda(\tau)] = \hat{x}(\tau)|_{t=t_0} + \int_{t_0}^{t} \lambda(\tau)\{L\hat{x}(\tau) - F[\hat{x}(\tau), \tau]\}d\tau = \hat{x}(\tau)|_{t=t_0}.
$$

(5.4.4)

Now we want to make the function $\Pi[x(t), \lambda(t)]$ stationary about $x$ at $x(t) = \hat{x}(t)$. Firstly, the variation of $\Pi[x(t), \lambda(t)]$ is derived as

$$
\delta\Pi[x(t), \lambda(t)] = \delta x(t)|_{t=t_0} + \int_{t_0}^{t} \delta \lambda(\tau)\{Lx(\tau) - F[x(\tau), \tau]\}d\tau + \int_{t_0}^{t} \lambda(\tau)\{L\delta x(\tau) - F[\hat{x}(\tau), \tau]\}d\tau
$$

$$
= \int_{t_0}^{t} \delta \lambda(\tau)\{Lx(\tau) - F[x(\tau), \tau]\}d\tau + \lambda(\tau)\delta x(t)|_{t=t_0} + \int_{t_0}^{t} \lambda(\tau)\delta x(\tau)|_{t=t_0}d\tau
$$

$$
- \int_{t_0}^{t} [L\lambda(\tau) + \lambda(\tau)\frac{\partial F(x, \tau)}{\partial x}]\delta x(\tau)d\tau - \int_{t_0}^{t} \lambda(\tau)\frac{\partial F(x, \tau)}{\partial \tau}\delta x(\tau)d\tau.
$$

(5.4.5)

If $F$ is not an explicit function of $\tau$, the term $[\frac{\partial F(x, \tau)}{\partial \tau}]\delta x(\tau)$ can be omitted in the preceding formula. Then we collect the terms including $\delta x(t)|_{t=t_0}$ and $\delta x(\tau)$,

$$
\delta x(t)|_{t=t_0} + \lambda(\tau)\delta x(\tau)|_{t=t_0} + \int_{t_0}^{t} [L\lambda(\tau) + \lambda(\tau)\frac{\partial F(x, \tau)}{\partial x}]\delta x(\tau)d\tau.
$$

(5.4.6)

Note that the boundary value of $x(\tau)$ at $\tau = t_0$ is prescribed, that is to say $\delta x(t)|_{t=t_0} = 0$. Thus the stationary condition for $\Pi[x(\tau), \lambda(\tau)]$ is obtained as

$$
\begin{cases}
\delta x(t)|_{t=t_0} : diag[1, 1, \ldots] + \lambda(\tau)|_{t=t_0} = 0 \\
\delta x(\tau) : L\lambda(\tau) + \lambda(\tau)\frac{\partial F(x, \tau)}{\partial x} = 0 \\
\delta \lambda(\tau) : Lx(\tau) = F[x(\tau), \tau]
\end{cases}.
$$

(5.4.7)

Noting that the exact solution $\hat{x}$ is unknown, therefore the truly optimal $\lambda(\tau)$ is not available herein. As an alternative, $\lambda(\tau)$ is approximated by replacing $\hat{x}$ with $x_n$. If $x_n$ is a neighbored function of $\hat{x}$, i.e. $\hat{x} - x_n = \delta \hat{x}$, the error caused will not exceed $O^2(\delta \hat{x})$.

In the VIM of He (1999), sometimes the nonlinear term is considered as restricted from variation, in order to simplify the calculation of $\lambda$. As an asymptotic method for solving nonlinear problems, VIM provides plenty of freedom to the user. It does not restrict the selection of initial guess functions and even allows the existence of unknown parameters in it. Further, depending on the selection of a restricted variation, the matrix of generalized Lagrange multipliers $\lambda$ can be determined in various ways. The less restricted the variational terms are, the more accurate will the Lagrange multipliers be.

5.4.2 VIM and its relationship to PIM and ADM

Instead of PIM, if VIM of He (1999) is utilized to solve the IVP governed by Eq. (1), we will get the corresponding correctional formula with $\lambda$.

[DOI: 10.1299/mer.17-00040] © 2017 The Japan Society of Mechanical Engineers
\[ x_{n+1}(t) = x_n(t) + \int_0^t \lambda [Lx_n - \tilde{F}(x_n, \tau)] d\tau = 0, \]  
(5.4.8)

where \( \tilde{F}(x_n, \tau) \) is considered as being restricted from variation. The matrix of Lagrange multipliers \( \lambda \) are determined from the following restricted stationary conditions:

\[
\begin{cases}
\lambda(\tau)_{|_{\tau=0}} + \text{diag}[1,1,\ldots] = 0 \\
L\lambda(\tau) = 0
\end{cases}
(5.4.9)
\]

The Lagrange multipliers, therefore, can be identified as \( \lambda(\tau) = \text{diag}[-1,-1,\ldots] \). As a result, we obtain the following iteration formula

\[ x_{n+1}(t) = x_n(t) - \int_0^t \{Lx_n(\tau) - F[x_n(\tau), \tau]\} d\tau = x_n(t_0) + \int_0^t F[x_n(\tau), \tau] d\tau. \]  
(5.4.10)

Noting that \( x_n(t_0) \) is equal to the initial value \( x(t_0) \), it is exactly the Picard iteration.

In the paper of He (1999), a first-order differential equation is considered and the ADM is proved to be a specific version of VIM. This work is reviewed in the following.

Consider the equation \( Lx + Rx + Nx = g(\tau) \), in which \( L \) is a first-order differential operator, while \( R \) is a linear operator. For this equation, the correctional formula of VIM is

\[ x_{n+1} = x_n + L^{-1}\{Lx_n + Rx_n + Nx_n - g(\tau)\}. \]  
(5.4.11)

Considering \( Rx_n + Nx_n \) as restricted from variation, \( \lambda \) can be easily obtained as \(-1\) in this case. Suppose \( x_0 = \bar{x}_0 = x(t_0) - L^{-1}g(\tau) \), substituting it into the formula gives

\[ x_1 = \bar{x}_0 - L^{-1}\{L\bar{x}_0 + R\bar{x}_0 + N\bar{x}_0\} = \bar{x}_0 + \bar{x}_1. \]  
(5.4.12)

Note that \( L^{-1}L\bar{x}_0 = 0 \). Therefore we have

\[ \bar{x}_1 = -L^{-1}\{L\bar{x}_0 + R\bar{x}_0 + N\bar{x}_0\} = -L^{-1}R\bar{x}_0 - L^{-1}A_0. \]  
(5.4.13)

Similarly the iteration process gives

\[ x_2 = (\bar{x}_0 + \bar{x}_1) - L^{-1}\{L(\bar{x}_0 + \bar{x}_1) + R(\bar{x}_0 + \bar{x}_1) + N(\bar{x}_0 + \bar{x}_1)\} = \bar{x}_0 + \bar{x}_1 + \bar{x}_2. \]  
(5.4.14)

If \( \bar{x}_1 \) is regarded as relatively small, expanding the nonlinear functional about \( \bar{x}_0 \) and ignoring the higher-level small functions leads to \( N(\bar{x}_0 + \bar{x}_1) = N(\bar{x}_0) + \bar{x}_1N'(\bar{x}_0) \). Then we have

\[
\begin{align*}
\bar{x}_2 &= -L^{-1}\{L(\bar{x}_0 + \bar{x}_1) + R(\bar{x}_0 + \bar{x}_1) + N(\bar{x}_0 + \bar{x}_1)\} \\
&= -L^{-1}\{L\bar{x}_1 + R(\bar{x}_0 + \bar{x}_1) + N(\bar{x}_0) + \bar{x}_1N'(\bar{x}_0)\} \\
&= -L^{-1}\{L(-L^{-1}R\bar{x}_0 - L^{-1}A_0) + R(\bar{x}_0 + \bar{x}_1) + A_0 + \bar{x}_1N'(\bar{x}_0)\} \\
&= -L^{-1}R\bar{x}_0 - L^{-1}A_0.
\end{align*}
\]  
(5.4.15)

Further, in the \( n \)th step of iteration, regard \( \bar{x}_{n-1} \) as the relatively small function and omit the terms \( O(\bar{x}_{m1}, \bar{x}_{m2}, \ldots, \bar{x}_{mk}) \), where \( m1 + m2 + \ldots + mk \geq n \). It leads to \( \bar{x}_n = -L^{-1}R\bar{x}_{n-1} - L^{-1}A_{n-1} \), which is the same as ADM.

It is worth to note that the above comparison can also be made in the case of multi-dimensional system. Herein, a general form of the nonlinear ordinary differential equations is taken for illustration.

\[ Lx = F(x, \tau), \]  
(5.4.16)

The ADM gives an iteration formula as

\[ x_{n+1} = L^{-1}A_n, \quad x_n = \sum_{i=0}^n \bar{x}_i, \]  
(5.4.17)
while the correctional formula of VIM is

\[ x_{n+1}(t) = x_n(t) + \int_{t_n}^{t} \lambda(t)[Lx_n - F(x_n, t)]d\tau. \]  

(5.4.18)

Set \( \lambda \) as \( \text{diag}[-1, -1, \ldots] \) and follow the manipulations we made to the one dimensional case. Suppose \( x_0 = \overline{x}_0 \), from the correctional formula of VIM we have

\[ x_1 = \overline{x}_0 + L^{-1}F(\overline{x}_0, \tau) = \overline{x}_0 + L^{-1}A_0 = \overline{x}_0 + \overline{x}_1. \]  

(5.4.19)

Accordingly,

\[ x_2 = \overline{x}_0 + L^{-1}F[\overline{x}_0 + \overline{x}_1, \tau]. \]  

(5.4.20)

Regard \( \overline{x}_1 \) as a relatively small amplitude vector of functions. The nonlinear term can be rewritten as

\[ F[(\overline{x}_0 + \overline{x}_1, \tau) = F(\overline{x}_0) + F'(\overline{x}_0)\overline{x}_1 + O(\overline{x}_1^2). \]  

(5.4.21)

Omitting the small terms \( O(\overline{x}_1^2) \), it leads to

\[ x_2 = \overline{x}_0 + L^{-1}[F(\overline{x}_0) + F'(\overline{x}_0)\overline{x}_1] = \overline{x}_0 + L^{-1}(A_0 + A_1) = \overline{x}_0 + \overline{x}_1 + \overline{x}_2. \]  

(5.4.22)

As the iteration goes on, we have

\[ x_n = \overline{x}_0 + L^{-1}[F(\overline{x}_0) + F'(\overline{x}_0)(\overline{x}_1 + \overline{x}_2 + \ldots + \overline{x}_{n-1}) + \frac{1}{2!}F''(\overline{x}_0)(\overline{x}_1 + \overline{x}_2 + \ldots + \overline{x}_{n-1})^2 + \ldots] \]  

where

\[ F''(\overline{x}_0) = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \]

(5.4.23)

which becomes the ADM correctional formula by omitting the small terms \( O(\overline{x}_m \overline{x}_n \ldots \overline{x}_{mn}) \).

From the statements above, it is clear that the ADM and PIM can both be regarded as variants of VIM. Although they are treated as different methods in literature, and were developed into various forms such as modified ADM and MCPI method, a common principle, namely the general use of Lagrange multipliers guides all of them.

### 5.4.3 Limitations of Global VIM

It has been shown that the VIM is effective in approximating the periodic motions of nonlinear systems, especially the limit cycle oscillation. To verify that, an unforced Duffing equation is solved with VIM herein. The governing equation is

\[ \begin{cases} \dot{x}_1 - x_2 = 0 \\ \dot{x}_2 - cx_2 + k_1x_1 + k_2x_1^3 = 0 \end{cases}, \]  

(5.4.24)

where \( c = 0, k_1 = 1, k_2 = 1 \).

In the approach proposed by He (1999), the Lagrange multiplier \( \lambda \) is derived with the nonlinear term being restricted from variation, and it is shown to be tenable in the unforced case with the initial conditions \( x_1(t_0) = A, x_2(t_0) = 0 \) and initial guess \( x_1(t) = A \cos \omega t \). The correctional formula is

\[ \begin{cases} x_1 \\ x_2 \end{cases}_{n+1} = \begin{cases} x_1 \\ x_2 \end{cases}_n + \int_{t_n}^{t} \begin{bmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{bmatrix} \begin{bmatrix} \dot{x}_1 - x_2 \\ \dot{x}_2 - cx_2 + k_1x_1 + k_2x_1^3 \end{bmatrix} d\tau, \]  

(5.4.25)

With the nonlinear term \( \dot{x}_1^3 \) restricted from variation, the matrix of Lagrange multipliers satisfies the condition:

\[ \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}; \]  

(5.4.26)
\[
\mathbf{\lambda}(\tau) = \begin{bmatrix}
-\cos(\tau - t) & \sin(\tau - t) \\
-\sin(\tau - t) & -\cos(\tau - t)
\end{bmatrix}.
\] (5.4.27)

The solution corrected only once with VIM can already give a result comparable with that of the RK4 method. The example of an unforced Duffing equation shows that the VIM could be very efficient in approximating the periodic solution of nonlinear systems by eliminating the secular term appearing in the correctional formula. The numerical results are plotted in Fig. 5.4.1 with the initial condition \(x(0) = 1, \dot{x}(0) = 0\) and initial guess \(x_1(t) = \cos \omega t\). To eliminate the secular term, the frequency is determined to be \(\omega = 1.32774\).

![Graph of x(t) vs. Time t]

However, in a more general case with \(\mathbf{x}(t_0) = \begin{bmatrix} A \\ B \end{bmatrix}^T\), the performance of VIM is far inferior to the preceding one. The initial guess is selected as \(x(t) = A \cos \omega t + (B/\omega) \sin \omega t\). Let \(A = 1\) and \(B = 1\), the variational iteration formula gives the corrected solution, in which the terms of \(\sin t\) and \(\cos t\) are

\[
\begin{align*}
&-4 - 17 \omega^2 + 9 \omega^4 \cos t, \\
&1 - 10 \omega^2 + 9 \omega^4 - \sin t.
\end{align*}
\] (5.4.28)

It is easily identified that the coefficients of \(\sin t\) and \(\cos t\) can not both be equal to zero. Here only the term of \(\cos t\) is eliminated, leading to the approximation \(\omega \approx 1.4493\). Substituting it into the corrected solution, we have

\[
\begin{align*}
x_1(t) &= 1.00598 \cos(1.4493 t) - 0.00597962 \cos(4.34791 t) - 0.111705 \sin(t) + \\
&0.694112 \sin(1.4493 t) + 0.0243163 \sin(4.34791 t)
\end{align*}
\] (5.4.29)

This result is compared with that of RK4 in Fig. 5.4.2.

It is believed that if both terms of \(\sin t\) and \(\cos t\) vanish, the corrected solution will be more accurate. But with just one parameter \(\omega\) in the initial guess, this will be an impossible mission. Thus we tried to bring in two parameters \(\omega_1, \omega_2\) in the initial guess, which takes the form \(x_1(t) = A \cos \omega_1 t + (B/\omega_2) \sin \omega_2 t\). If the resulted corrected solution can be obtained analytically, the accuracy will be much improved. But along with it, the calculation of variational iteration formula will become much more complex. We found that even with the help of Mathematica, this job is still too burdensome to be accomplished.

Moreover, it is also indicated that a carefully selected initial guess is important to ensure the convergence of this method. If the approximate function takes the form as \(x_1(t) = A + Bt\) rather than \(x_1(t) = A \cos \omega t\), the result given by VIM will be more and more divergent from the true solution with the iteration. It is illustrated in the case \(A = 1\) and \(B = 1\). With VIM, the initial guess \(x_{1,0}(t) = 1 + t\) is corrected for 2 times and the results are given as follows.
\[ x_{1,1}(t) = 5 + 3t - 3t^2 - t^3 - 4\cos t - 2\sin t \quad (5.4.30) \]

\[ x_{2,1}(t) = 393349 + 252477t + t^9 + \left( -\frac{7083587}{18} + \frac{999t}{4} - \frac{3t^7}{7} \right) \cos(t) + \left( \frac{1666}{9} + 6t - 50t^2 - 6t^3 \right) \cos(2t) \]

\[ -\frac{1}{2} \cos(3t) + \left( -\frac{505575}{2} + 378t + \frac{6t^7}{7} \right) \sin t + (32 + \frac{424}{3} t - 8t^3)\sin(2t) - \frac{11}{4} \sin(3t) \quad . \quad (5.4.31) \]

The solution \( x_{1,1}(t) \) is compared with that of RK4 in Fig. 5.4.3.

Fig. 5.4.2 Solid line: RK4. Dash line: VIM

Fig. 5.4.3 Solid line: RK4. Dotted line: VIM

The above numerical examples and analysis show that VIM is fastidious to the initial condition of the problem, and too sensitive to the initial guess function, which is analogous to the Newton’s iteration method for solving nonlinear
algebraic equations (NAEs). This method also fails to predict the long-term responses of some complex nonlinear phenomenon such as chaos or quasi periodic motion, because these patterns of motion are non-periodic and can hardly be approximated by any analytical functions.

5.5 Local variational iteration method

As is shown in Atluri (2005) and Dong, Alotaibi, Mohiuddine and Atluri (2014), local approximations are far more effective in solving initial value and boundary value problems. We use these ideas in developing a local variational iteration method.

To remedy the drawbacks shown above, we propose to further improve the VIM. Here is the primary idea. Firstly, the nonlinear term should be kept in the derivation of Lagrange multipliers so that a better correction can be obtained for arbitrary approximation. And then the entire time domain is divided into small intervals and the LVIM is applied repeatedly in each time interval \( t_{i-1} \leq \tau \leq t_i \). In each interval \( t_{i-1} \leq \tau \leq t_i \), we approximate locally using an arbitrary function \( x(\tau) = \mathbf{A} + \mathbf{B} \tau \) for instance; then the corrected solution will be a function of \( \mathbf{A} \) and \( \mathbf{B} \), i.e. the initial condition of each interval, thus the solution in the entire time domain can be obtained by the repetition of \( \mathbf{A} \) and \( \mathbf{B} \) step by step.

With this in mind, the IVP of the unforced Duffing equation is solved with the initial condition as \( x_1(t_0) = 1 \), \( x_2(t_0) = 1 \). The correctional formula is

\[
\begin{bmatrix}
  x_1 \\
  x_2 \\
end{bmatrix}_{n+1} =
\begin{bmatrix}
  x_1 \\
  x_2 \\
end{bmatrix}_n + \int_{t_{i-1}}^{t_i} \begin{bmatrix}
  \lambda_{a1} \\
  \lambda_{a2} \\
end{bmatrix}
\begin{bmatrix}
  \dot{x}_1 - x_2 \\
  \dot{x}_2 - c x_2 + k_1 x_1 + k_2 x_1^3 \\
end{bmatrix}
\ d\tau,
\]

where \( t_{i-1} \leq \tau \leq t_i \). The stationary condition is

\[
\delta x(t) \bigg|_{t_{i-1}}^{t_i} : \lambda(t) \bigg|_{t_{i-1}}^{t_i} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix};
\]

\[
\delta x(t) : -\dot{\lambda}(t) + \lambda(t)
\begin{bmatrix}
  0 \\
  k_1 + 3 k_2 x_1^2 \\
end{bmatrix}
_0
\begin{bmatrix}
  -1 \\
  -c \\
end{bmatrix}
= 0, \; t_{i-1} \leq \tau \leq t_i \; \text{and} \; t_{i-1} \leq t \leq t_i.
\]

Instead of neglecting the nonlinear term, we kept it in the derivation of \( \lambda \). This makes the differential equations more difficult to be solved analytically. But an exact analytical solution is not necessary here. The differential transform (DT) method is introduced to obtain the approximated \( \lambda \) in the form of power series. With the nonlinear term included, the approximated \( \lambda \) is actually more precise than that in the VIM of He (1999).

Supposing that the initial guess of \( x_1 \) takes the form as \( x_{1,0}(\tau) = \mathbf{B} \tau + \mathbf{A} \), an approximated \( \lambda \) can be obtained using the DT method. Herein only the first four DT terms of \( \lambda(\tau) \) is kept.

\[
\lambda_{a1} = -1 + \frac{1}{2}(t + \tau)^2 \left( k_1 + 3 (A + B \tau)^2 k_2 \right) + \frac{1}{3}(t + \tau)^3 \left( 6 B (A + B \tau) k_2 - \frac{1}{2} c \left( k_1 + 3 (A + B \tau)^2 k_2 \right) \right),
\]

\[
\lambda_{a2} = -t + \tau - \frac{1}{2} \left( -t + \tau \right)^2 - \frac{1}{2} \left( -t + \tau \right)^3 \left( \frac{c^2}{2} + \frac{1}{2} \left( -k_1 - 3 (A + B \tau)^2 k_2 \right) \right),
\]

\[
\lambda_{a3} = \left( -t + \tau \right)^2 \left( k_1 - 3 (A + B \tau)^2 k_2 \right) + \frac{1}{2} \left( -t + \tau \right)^3 \left( 6 B (A + B \tau) k_2 + c \left( k_1 + 3 (A + B \tau)^2 k_2 \right) \right)
\]

\[
\lambda_{a4} = \left( -t + \tau \right)^3 \left( -3 B^2 k_2 + 6 Bc (A + B \tau) k_2 + \frac{1}{2} \left( k_1 + 3 (A + B \tau)^2 k_2 \right) \left( -c^2 + k_1 + 3 (A + B \tau)^2 k_2 \right) \right),
\]

(5.5.4)
\[ \lambda_{22} = -1 + c(-t + \tau) + \frac{1}{2}(-t + \tau)^2\left(-c^2 + k_1 + 3\left(A + B\tau\right)^2 k_2\right) + \frac{1}{3}(-t + \tau)^3 \]
\[ \left(-\frac{1}{2}c(-c^2 + k_1 + 3\left(A + B\tau\right)^2 k_2) + \frac{1}{2}\left(6B\left(A + B\tau\right)k_2 - c\left(k_1 + 3\left(A + B\tau\right)^2 k_2\right)\right)\right), \]

where \( t_{i-1} \leq t \leq t_i \) and \( t_{i-1} \leq \tau \leq t_i \). Using the correctional formula for one time, we have
\[
x_{i,1} = A + Bt + \frac{1}{5040}t^2 \left\{ 42 \left\{ 5Bc\left(12 + ct(4 + ct)\right) + k_1 \left\{ -5A\left(12 + ct(4 + ct)\right) - Bt\left(20 + ct(10 + ct)\right) + r^2(5A + Bt)k_1\right\} \right\} - 6 \left\{ 35A^3\left(12 + ct\right) + 21A^2Bt\left(20 + ct\left(10 + ct\right)\right) + 7AB^2r^2\left(30 + ct\left(12 + ct\right)\right) \right\} k_2 \right. \\
\left. + B^3r^4\left(42 + ct\left(14 + ct\right)\right) - 4r^2\left(35A^3 + 21A^2Bt + 7AB^2r^2B^3r^3\right)k_i \right\} k_2 + 5r^2\left(126A^5 + 126A^4Bt + 84A^3B^2r^2 + 36A^2B^3r^3 + 9AB^4r^4 + B^5r^5\right)k_2^2 \right\}, \quad (5.5.5)
\]

where \( t_{i-1} \leq t \leq t_i \).

In Fig. 5.5.1 (a), the preceding solution is compared with the numerical results obtained by MATLAB built-in ODE45, for which the relative and absolute accuracy are both set as \(-10^{15}\). Fig. 5.5.1 (b) shows the computational error of the LVIM with respect to ODE45. The time step size is set as \( \Delta t = 0.01 \) and the simulation is carried in \( t = [0, 100] \). From Fig. 5.5.1 (b), it can be seen the error of the LVIM is less than \(-10^{-3}\) even for \( t = 100 \).

To further inspect the performance of the LVIM in predicting complicated responses, the forced Duffing equation is investigated.
\[
\ddot{x} + cx + k_1x + k_2x^3 = f \cos(\omega t), \quad c = 0.15, \quad k_1 = -1, \quad k_2 = 1, \quad f = 0.41, \quad \omega = 0.4. \quad (5.5.6)
\]
It is shown to be a chaotic system by RK4 and other reliable methods. Unsurprisingly, the VIM of He (1999) fails to solve it because the solution is too complicated to be computable. An approximation is made to the forced term of the system. Since it is involved in the integration, a simple form of it will accelerate the calculation. For that, the forced term is expanded into power series as well. The stationary condition for \( \dot{\lambda} \) is
\[
\delta \ddot{x}(t) : 1 - \dot{\lambda}(\tau) \bigg|_{\tau=t} = 0, \quad \delta \dot{x}(t) : \dot{\lambda}(\tau) \bigg|_{\tau=t} = 0, \quad \delta \ddot{x}(\tau) : \ddot{\lambda}(\tau) - c\dot{\lambda}(\tau) + k_1\dot{\lambda}(\tau) + 3k_2\dot{x}_n^2\dot{\lambda}(\tau) = 0, \quad (5.5.7)
\]
where \( t_{i-1} \leq \tau \leq t_i \) and \( t_{i-1} \leq t \leq t_i \).
Suppose the initial guess in each interval $t_{i-1} \leq \tau \leq t_i$ is $x_i(\tau) = A + B\tau$. With DT method, we can get the fourth-order approximated $\dot{\lambda}(\tau,t)$ in terms of $A$ and $B$, which is

$$
\dot{\lambda}(\tau,t) = -t + \frac{1}{2}c(t-\tau)^2 + \tau + \frac{1}{6}(t-\tau)^3 \left(-c^2 + k_1 + 3(A+B)^2 k_2\right) - \frac{1}{24}(t-\tau)^4 \left(-c^3 + 2ck_1 + 6(A+Bt)(2B + Ac + Bct)k_2\right)
$$

(5.5.8)

Then substituting it into the correction formula, with the forced term being replaced by power series, will lead to the corrected function $x(t)$

$$
x(t) = A + Bt + \frac{1}{2}(-Bc + f\cos(C\omega) - Ak_1 - A^2 k_2) t^2 + \frac{1}{6} \left(\frac{Bc^2 - cf\cos(C\omega) - f\omega\sin(C\omega)}{(-B + Ac)k_1 + A^2 (-3B + Ac) k_2} + \frac{Bc^2 - cf\cos(C\omega) + f\omega\sin(C\omega)}{6B^2 - 6Ac + A^2c^2 + 3A f\cos(C\omega)} k_2 + 3A^2 k_2^2 + \frac{Bc^2 + c^2 f\cos(C\omega) - c f\omega^2 \cos(C\omega) - c f\omega \sin(C\omega) + f\omega^3 \sin(C\omega)}{120} \right) t^4 + O(t^6),
$$

(5.5.9)

where $t_{i-1} \leq t \leq t_i$. It has a compacted form with $A$, $B$, $t_{i-1}$ (denoted as $C$ in the corrected function) as the coefficients. When applied to multiple time intervals, it only needs to recompute $A$, $B$, and $t_0$, which correspond to

![Figure 5.5.1](image)

**Figure 5.5.1** Comparison of the LVIM and the ODE45. The time step size of LVIM is selected as (a) $\Delta t = 1$ and (b) $\Delta t = 0.01$. 
the final position, velocity and time instant of the last interval. The LVIM is very fast, precise and efficient on solving nonlinear problems. Fig. 5.5.2 shows the chaotic motions of a forced Duffing oscillator predicted by both LVIM and RK4 methods. The time step size of LVIM is set as 0.2, while that of RK4 needs to be set as 0.02 in order to achieve the same accuracy. The numerical simulation carried out with MATLAB shows the LVIM is almost 10 times faster than RK4 in this case. The computation time of the LVIM and the RK4 are 0.013s and 0.135s, respectively.

Similar to the modified ADM, the variational iteration method is further improved herein by bringing in some local approximation techniques, so as to be practical in predicting long term motion and complex dynamical responses. The LVIM makes the derivation of Lagrange multipliers \( \lambda \) and the calculation of the variational iteration formula much simpler, yet still provide reliable solution in each sub time domain.

6. Local variational iteration-collocation methods

A new class of time-integrators is presented for strongly nonlinear dynamical systems. These algorithms are far superior to the currently common time integrators in computational efficiency and accuracy. These three algorithms are based on a Local Variational Iteration Method (LVIM) applied over a finite interval of time. By using Chebyshev polynomials as trial functions and Dirac-Delta functions as the test functions over the finite time interval, the three algorithms are developed into three different discrete time-integrators through the collocation method.

6.1 Local variational iteration method and its modifications

To remedy the drawbacks of VIM, the LVIM was proposed as an alternative in Wang and Atluri (2016a). With LVIM, the solution of the nonlinear system is approximated locally in a sub-interval of time, \( t_i \) to \( t_{i+1} \), by the correctional formula

\[
x_{n+1}(t) = x_n(t) + \int_{t_i}^{t_{i+1}} \lambda(\tau) [L x_n(\tau) - F[x_n(t), \tau]] d\tau, \quad t \in [t_i, t_{i+1}].
\]

where \( \lambda(\tau) \) is the matrix of the Lagrange multipliers. Based on the variational principle, the optimal \( \lambda(\tau) \) needs to satisfy the following constraints:
\[
\left\{ \begin{align*}
\delta x_n(\tau)_{\tau=t} :& \quad I + \lambda(\tau)_{\tau=t} = 0 \\
\delta x_n(\tau) :& \quad L \lambda(\tau) + \lambda(\tau) \frac{\partial F(x_n, \tau)}{\partial x_n} = 0, \quad \tau \in [t_i, t_{i+1}].
\end{align*} \right. \tag{6.1.2}
\]

In LVIM, the term \( \partial F/\partial x \) is always retained in the evolution equation for \( \lambda \) as in Eq. (6.1.2), so that a better correction can be obtained for arbitrary initial guess. Furthermore, the entire time domain is divided into small intervals and the LVIM is applied repeatedly in each finite time interval \( \Delta t = t_{i+1} - t_i \).

In Wang and Atluri (2016a), the derivation of the generalized Lagrange multipliers is accomplished using an approximation technique, i.e. the differential transform (DT) method (Chu and Lo, 2011, Jang, etc., 2000). In each interval \( t_i \leq \tau \leq t_{i+1} \), we approximate \( x(\tau) \) locally, using a very simple function \( x(\tau) = A + B \tau \) for instance; then the corrected solution will be a function of \( A \) and \( B \), i.e. the initial condition of each interval, thus the solution in the entire time domain can be obtained by the repetitive solution in terms of \( A \) and \( B \) step by step. In this way, the LVIM enables the initial guess function to take a very simple form and provides a convenient approach for the implementation of the correctional formula. Note that the corrected approximation in each interval takes the same expression. Therefore, it only needs to go through the symbolic calculations for one time.

In spite of all the merits of LVIM, we found that the resulting analytical solution remains to be a very lengthy expression. If a highly accurate solution is desired, the correction would have to be conducted for multiple times. In that case, the symbolic calculation would still be troublesome and the solution would be too complicated to be interpreted. Considering that, some further modifications are made to the LVIM in the following sections.

6.1.1 Elimination of generalized Lagrange multipliers and Algorithm-1

By differentiating the correctional formula of LVIM in Eq. (6.1.1) and considering the constraints on \( \lambda(\tau) \) [Eq. (6.1.2)], we have

\[
\frac{dx_{n+1}}{dt} = \frac{dx_n}{dt} + \lambda(\tau)_{\tau=t} \left[ \frac{dx_n}{dt} - F(x_n, t) \right] + \int_0^\tau \frac{\partial \lambda}{\partial t} [L x_n - F(x_n, \tau)] d\tau \\
= F(x_n, t) + \int_0^\tau \frac{\partial \lambda}{\partial t} [L x_n - F(x_n, \tau)] d\tau, \quad t \in [t_i, t_{i+1}]. \tag{6.1.3}
\]

To further simplify the preceding formula, the following property of \( \lambda(\tau) \) is used.

**Assertion:**

The matrix of generalized Lagrange multipliers \( \lambda(\tau) \) in Eq. (6.1.2) happens to be the solution \( \lambda(t) \) of the following ordinary differential equations:

\[
\left\{ \begin{align*}
1 + \lambda(t)_{\tau=t} &= 0 \\
\frac{\partial \lambda(t)}{\partial t} - J(t) \lambda(t) &= 0 \quad \tau \in [t_i, t_{i+1}].
\end{align*} \right. \tag{6.1.4}
\]

**Proof:**

Let \( J(x_n, \tau) = \partial F(x_n, \tau)/\partial x_n \), the generalized Lagrange multipliers can be identified in the Magnus approach, through which the solution of a linear ordinary differential equations with variable coefficients:

\[
Y'(t) = A(t)Y(t), \quad Y(t_0) = Y_0.
\]

can be expressed as

\[
Y(t) = \{\exp[\Omega(t, t_0)]\} Y_0 = \{\exp[\sum_{k=1}^\infty \Omega_k(t)]\} Y_0. \tag{6.1.5}
\]

The first three terms of Magnus expansion \( \Omega_k(t) \) read

\[
\Omega_1(t) = \int_0^t A(t) dt, \quad \Omega_2(t) = \frac{1}{2} \int_0^t \int_0^t \left[ A(t_1), A(t_2) \right] dt_1 dt_2.
\]

\[
\Omega_3(t) = \int_0^t A(t_1) dt_1, \quad \Omega_4(t) = \frac{1}{2} \int_0^t \int_0^t \left[ A(t_1), A(t_2) \right] dt_1 dt_2.
\]

\[
\Omega_5(t) = \int_0^t A(t_1) dt_1, \quad \Omega_6(t) = \frac{1}{2} \int_0^t \int_0^t \left[ A(t_1), A(t_2) \right] dt_1 dt_2.
\]

\[
\Omega_n(t) = \int_0^t A(t_1) dt_1, \quad \Omega_n(t) = \frac{1}{2} \int_0^t \int_0^t \left[ A(t_1), A(t_2) \right] dt_1 dt_2.
\]

\[
\Omega_n(t) = \int_0^t A(t_1) dt_1, \quad \Omega_n(t) = \frac{1}{2} \int_0^t \int_0^t \left[ A(t_1), A(t_2) \right] dt_1 dt_2.
\]

\[
\Omega_n(t) = \int_0^t A(t_1) dt_1, \quad \Omega_n(t) = \frac{1}{2} \int_0^t \int_0^t \left[ A(t_1), A(t_2) \right] dt_1 dt_2.
\]

\[
\Omega_n(t) = \int_0^t A(t_1) dt_1, \quad \Omega_n(t) = \frac{1}{2} \int_0^t \int_0^t \left[ A(t_1), A(t_2) \right] dt_1 dt_2.
\]

\[
\Omega_n(t) = \int_0^t A(t_1) dt_1, \quad \Omega_n(t) = \frac{1}{2} \int_0^t \int_0^t \left[ A(t_1), A(t_2) \right] dt_1 dt_2.
\]

\[
\Omega_n(t) = \int_0^t A(t_1) dt_1, \quad \Omega_n(t) = \frac{1}{2} \int_0^t \int_0^t \left[ A(t_1), A(t_2) \right] dt_1 dt_2.
\]
\[ \Omega_\lambda(t) = \frac{1}{6} \int_0^t dt_1 \int_0^{t_1} dt_2 \int_0^{t_1} dt_3 (A(t_1), [A(t_2), A(t_3)]) + [A(t_2), [A(t_3), A(t_1)]], \] (6.1.7)

where \([A,B] \equiv AB - BA\) is the commutator of \(A\) and \(B\).

By transposing the constraints of \(\tau_\lambda\) in Eq. (6.1.2) as
\[
\begin{align*}
\lambda_\tau = & \exp \int_0^t (-J^\tau) d\tau_1 + \frac{1}{2} \int_0^t d\tau_1 \int_0^{\tau_1} d\tau_2 [-J^\tau(\tau_1), -J^\tau(\tau_2)] + \ldots \lambda_\tau, \quad \tau \in [t_1, t] \nonumber \\
& = -\exp \{-\int_0^t (J^\tau) d\tau_1 - \frac{1}{2} \int_0^t d\tau_1 \int_0^{\tau_1} d\tau_2 [-J(\tau_1), -J(\tau_2)] + \ldots \} \nonumber \\
& = -\exp \{\int_0^t (J) d\tau_1 + \frac{1}{2} \int_0^t d\tau_1 \int_0^{\tau_1} d\tau_2 [J(\tau_1), J(\tau_2)] + \ldots \}
\end{align*}
\] (6.1.9)

Inverting the expression leads to
\[
\lambda_\lambda = \lambda_\lambda \left|_{t_1} \right. \exp \left\{ \int_0^t (-J)d\tau_1 + \frac{1}{2} \int_0^t d\tau_1 \int_0^{\tau_1} d\tau_2 [-J(\tau_1), -J(\tau_2)] + \ldots \right\} 
\]
\[
= -\exp \{-\int_0^t (J) d\tau_1 - \frac{1}{2} \int_0^t d\tau_1 \int_0^{\tau_1} d\tau_2 [-J(\tau_1), -J(\tau_2)] + \ldots \} = -\exp \{\int_0^t (J) d\tau_1 + \frac{1}{2} \int_0^t d\tau_1 \int_0^{\tau_1} d\tau_2 [J(\tau_1), J(\tau_2)] + \ldots \}.
\] (6.1.10)

We notice that
\[
\begin{align*}
\int_0^t d\tau_1 \int_0^{\tau_1} d\tau_2 [J(\tau_1), J(\tau_2)] - \int_0^t d\tau_1 \int_0^{\tau_1} d\tau_2 [J(\tau_1), J(\tau_2)] &= 0 \\
& = 0
\end{align*}
\] (6.1.11)

and the other terms in the Magnus series also possess a similar characteristic. The expression of \(\lambda(\tau)\) can thus be rewritten as
\[
\lambda(\tau) = -\exp \{\int_0^t (J) d\tau_1 + \frac{1}{2} \int_0^t d\tau_1 \int_0^{\tau_1} d\tau_2 [J(\tau_1), J(\tau_2)] + \ldots \}, \quad \tau \in [t_1, t] \nonumber \\
\] (6.1.12)

It is exactly \(\lambda(t)\) in the form of Magnus series.

Using the equivalence between \(\lambda(\tau)\) and \(\lambda(t)\), we obtain from Eqs. (6.1.3) and (6.1.4)
\[
\begin{align*}
\frac{dX_{n+1}}{dt} &= F(x_n, t) + \int_0^t \frac{\partial \lambda(\tau)}{\partial t} [Lx_n - F(x_n, \tau)] d\tau \\
& = F(x_n, t) + \int_0^t \frac{\partial \lambda(\tau)}{\partial t} [Lx_n - F(x_n, \tau)] d\tau \\
& = F(x_n, t) + J(x_n, t) \int_0^t \lambda(\tau)[Lx_n - F(x_n, \tau)] d\tau \\
& = F(x_n, t) + J(x_n, t) \int_0^t \lambda(\tau)[Lx_n - F(x_n, \tau)] d\tau \\
& = F(x_n, t) + J(x_n, t) (x_{n+1} - x_n)
\end{align*}
\] (6.1.13)

This recursive formula for \(\dot{x}\) can be rewritten as

Algorithm-1: \[
\begin{align*}
\frac{dX_{n+1}}{dt} - J(x_n, t)X_{n+1} &= F(x_n, t) - J(x_n, t)x_n, \quad t \in [t_{n-1}, t_i].
\end{align*}
\]
6.1.2 Polynomial approximations of generalized Lagrange multipliers and Algorithm-2

It is difficult to explicitly derive the generalized Lagrange multipliers, since the constraints include \( n \). However, with the help of the differential transform method, it is convenient to approximate the generalized Lagrange multipliers in a series of polynomials

\[
\lambda(\tau) \approx T_k[\lambda] + T_1[\lambda](\tau - t) + \cdots + T_n[\lambda](\tau - t)^K,
\]

where \( T_k[\lambda] \) is the \( k \) th order differential transformation of \( \lambda(\tau) \), i.e.

\[
T_k[\lambda] = \frac{1}{k!} \frac{d^k \lambda(\tau)}{d\tau^k}.
\]

From the constraints on \( \lambda(\tau) \), \( T_k[\lambda] \) can be determined in an iterative way as following:

\[
T_k[\lambda] = \text{diag}[-1,-1,\ldots], \quad T_{k+1}[\lambda] = -\frac{T_k[\lambda J]}{k+1}, \quad 0 \leq k \leq K + 1.
\]

In the VIM of He (1999), the nonlinear term in the dynamical system is normally restricted from variation to simplify the derivation of \( \lambda(\tau) \). If we also restrict the nonlinear term \( F(x, \tau) \) from variation herein, the matrix of generalized Lagrange multipliers can be easily determined as \( \lambda(\tau) = -I \). Substituting it into the correctional formula of LVIM leads to the Picard iteration method

\[
x_{n+1}(t) = x(t) + \int_{t_n}^{t} F[x_n(\tau), \tau] d\tau.
\]

If we differentiate this formula, we have

\[
\frac{dx_{n+1}}{dt} = F(x_n, t), \quad t \in [t_n, t_{n+1}].
\]

However, if the nonlinear term is included in the variation as in the present paper, the resulting \( \lambda(\tau) \) should be a function of \( \tau \) and \( t \). Making differentiations to the correctional formula of LVIM for \( K + 1 \) times will lead to

\[
x^{(K+1)}_{n+1} = x^{(K+1)}_n + \left( \lambda \frac{\partial}{\partial t} G \right)^{(K)} + \left( \lambda \frac{\partial}{\partial t} G \right)^{(K-1)} + \cdots + \left( \lambda \frac{\partial}{\partial t} G \right)^{(1)} + \int_{t_n}^{t} \left( \lambda \frac{\partial}{\partial t} G \right)^{(0)} d\tau,
\]

where \( G = Lx_n(\tau) - F[x_n(\tau), \tau] \).

If we simply enforce \( x(\tau) \) to be a constant \( x(t_n) \), then \( \lambda(\tau) \) can be approximated by truncated polynomial series in which the highest order of \( t \) is \( K \). The last term of the preceding expression can be omitted, thus we will obtain a correctional formula without the integral operator.

\[
x^{(K+1)}_{n+1} = x^{(K+1)}_n + \left( \lambda \frac{\partial}{\partial t} G \right)^{(K)} + \left( \lambda \frac{\partial}{\partial t} G \right)^{(K-1)} + \cdots + \left( \lambda \frac{\partial}{\partial t} G \right)^{(1)}.
\]

However, this formula is not complete. It should be noted that each time that we differentiate the correctional formula, the information of constant terms in it is lost. For that reason, before each differentiation is made, the initial value \( x^{(K)}_{n+1}(t_n), \quad k = 0, 1, 2, \ldots, K \) is kept as additional constrains to the correctional formula. In the expression above, \( T_k[\lambda] \) \( (k = 0, 1, 2, \ldots, K) \) are constants. So \( x^{(K+1)}_{n+1} \) is a simple linear combination of \( x^{(K+1)}_n \) and \( G \). Clearly, if the iteration become static, i.e. \( x_{n+1} = x_n \), the correctional formula will guarantee

\[
G = Lx_n(\tau) - F[x_n(\tau), \tau] = 0.
\]

By approximating the generalized Lagrange multipliers in a series of polynomials, the LVIM can also be altered in the following way, for \( x_{n+1} \):
\[ x_{n+1}(t) = x_n(t) + \int_t^t \lambda(\tau) \{ Lx_n(\tau) - F[x_n(\tau), \tau] \} d\tau \]
\[ = x_n(t) + \int_t^t \{ T_0[\lambda] + \tau \} Gd\tau + \int_t^t \{ T_1[\lambda] \} Gd\tau + \int_t^t \{ T_2[\lambda] \} Gd\tau + \int_t^t \{ T_3[\lambda] \} Gd\tau \]  

(6.1.22)

Considering that \( T_i[\lambda], \ k = 0,1,...K \) are functions of \( x_n(t) \) and \( t \), the preceding expression can be rewritten as

**Algorithm-2:** \[ x_{n+1}(t) = x_n(t) + A_0(t) \int_t^t Gd\tau + A_1(t) \int_t^t \tau Gd\tau + \cdots + A_K(t) \int_t^t \tau^K Gd\tau . \]

where \( t \in [t_i, t_{i+1}] \). The coefficient matrices \( A_k(t), \ k = 0,1,...K \) are combinations of \( T_i[\lambda] \) and \( t \). It is worth to note that the Algorithm-2 can be easily generalized into higher order systems, although it is derived in the case of first order differential equations.

### 6.1.3 Exponential approximations of generalized Lagrange multipliers and Algorithm-2

As is stated in subsection 2.1, the matrix of generalized Lagrange multipliers can be expressed in the form of exponential functions. Since the exponential functions can be equivalently defined by power series, we have

\[ \lambda(\tau) = \exp \{ \int_0^\tau \tilde{G}(x_n, \zeta)d\zeta \} - \{ \text{diag}[1,1,...] + \int_0^\tau \tilde{G}(x_n, \zeta)d\zeta + \frac{1}{2!} \int_0^\tau \tilde{G}(x_n, \zeta)d\zeta^2 + \cdots \} \]  

(6.1.23)

We then substitute it into \( \partial \lambda(\tau)/\partial t = J(x_n, t)\tilde{\lambda}(\tau) \). If the higher order terms of \( \lambda(\tau) \) are ignored, \( \partial \lambda(\tau)/\partial t \) can be approximated by \( \partial \lambda(\tau)/\partial t = -J(x_n, t) \). Accordingly, it gives rise to the following iterative formula.

\[ \frac{dx_{n+1}}{dt} = \frac{dx_n}{dt} + \lambda(t) \{ Lx_n(t) - F[x_n(t), t] \} + \int_t^t \frac{\partial \lambda}{\partial t} \{ Lx_n(t) - F[x_n(t), t] \} d\tau \]
\[ = F(x_n, t) - J(x_n, t) \int_t^t \{ Lx_n(t) - F[x_n(t), t] \} d\tau \]
\[ = F(x_n, t) - J(x_n, t) [x_n - \int_t^t F(x_n, \tau) d\tau] \]  

(6.1.24)

or simply

**Algorithm-3:** \[ \frac{dx_{n+1}}{dt} = F(x_n, t) - J(x_n, t) [x_n - \int_t^t F(x_n, \tau) d\tau] . \]

where \( t \in [t_i, t_{i+1}] \).

In summary, the three proposed modifications of the Local Variational Iteration Method are listed as below.

### Table 6.1.1 Three modifications of LVIM with \( t \in [t_i, t_{i+1}] \):

<table>
<thead>
<tr>
<th>Algorithm-1</th>
<th>( \frac{dx_{n+1}}{dt} - J(x_n, t)x_n = F(x_n, t) - J(x_n, t)x_n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm-2</td>
<td>( x_{n+1}(t) = x_n(t) + A_0(t) \int_t^t Gd\tau + A_1(t) \int_t^t \tau Gd\tau + \cdots + A_K(t) \int_t^t \tau^K Gd\tau )</td>
</tr>
<tr>
<td>Algorithm-3</td>
<td>( \frac{dx_{n+1}}{dt} = F(x_n, t) - J(x_n, t) [x_n - \int_t^t F(x_n, \tau) d\tau] )</td>
</tr>
</tbody>
</table>

It is not hard to see that direct applications of the Algorithm-1, 2, and 3 with an initial analytical guess still needs very lengthy symbolic computations. Instead of using analytical initial guess functions, we satisfy the iterative equations of Algorithm-1, 2, and 3, in a weak-form by using trial functions and test functions in the time interval \( t_i \leq t \leq t_{i+1} \). Depending on the selection of test functions and trial functions, there are various weak forms, of which the collocation form is the simplest. To explain the idea of weak forms of the three modifications clearly, the collocation method is
introduced first.

6.2 Implementation of the Algorithm-1, 2 and 3, using the collocation method

Considering a vector of trial functions $\mathbf{u}$, the residual error of the first order differential equations considered in this paper is

$$ R = \mathbf{L} \mathbf{u} - \mathbf{F}(\mathbf{u}, t) \neq \mathbf{0}, \quad t \in [t_i, t_{i+1}] . \tag{6.2.1} $$

With a diagonal matrix of test functions $\mathbf{v} = \text{diag}[v, v, \ldots]$, the local weighted residual weak-form of the formulation is written as

$$ \int_{t_i}^{t_{i+1}} \mathbf{v} R \, dt = \int_{t_i}^{t_{i+1}} \mathbf{v} [\mathbf{L} \mathbf{u} - \mathbf{F}(\mathbf{u}, t)] \, dt = \mathbf{0} . \tag{6.2.2} $$

Let the trial function $u_e$ be the linear combinations of basis functions $\phi_{e,ab}(t)$

$$ u_e = \sum_{ab=1}^{N} \alpha_{e,ab} \phi_{e,ab}(t) = \mathbf{\Phi}_e(t) \mathbf{A}_e . \tag{6.2.3} $$

where $u_e$ represents an element of $\mathbf{u}$, the columns of $\mathbf{\Phi}_e$ represent each of the independent basis functions, and the vector $\mathbf{A}_e$ contains undetermined coefficients.

We use the Dirac Delta functions as the test function, i.e. $v = \delta(t - t_m)$ for a group of pre-selected points $t_m$ ($m = 1, 2, \ldots, M$) in the local domain $t \in [t_i, t_{i+1}]$. It leads to the local collocation method

$$ \mathbf{L} \mathbf{u}(t_m) - \mathbf{F}(\mathbf{u}(t_m), t_m) = \mathbf{0}, \quad t_m \in [t_i, t_{i+1}] . \tag{6.2.4} $$

The value of $u_e$ and the differentiation of $u_e$ at time point $t_m$ can be expressed as

$$ u_e(t_m) = \sum_{ab=1}^{N} \alpha_{e,ab} \phi_{e,ab}(t_m) = \mathbf{\Phi}_e(t_m) \mathbf{A}_e \quad \text{and} \\
L u_e(t_m) = \sum_{ab=1}^{N} \alpha_{e,ab} L \phi_{e,ab}(t_m) = L \mathbf{\Phi}_e(t_m) \mathbf{A}_e . \tag{6.2.6} $$

In matrix form, they can be written as $\mathbf{U}_e = \mathbf{B}_e \mathbf{A}_e$ and $\mathbf{L} \mathbf{U}_e = \mathbf{L} \mathbf{B}_e \mathbf{A}_e$ separately, where

$$ \mathbf{U}_e = [u_e(t_1), u_e(t_2), \ldots, u_e(t_M)]^T \quad \text{and} \quad \mathbf{B}_e = [\mathbf{\Phi}_e(t_1)^T, \mathbf{\Phi}_e(t_2)^T, \ldots, \mathbf{\Phi}_e(t_M)^T]^T . $$

By simple transformations, we have $\mathbf{L} \mathbf{U}_e = \mathbf{L} \mathbf{B}_e \mathbf{A}_e = (\mathbf{L} \mathbf{B}_e) \mathbf{B}_e^{-1} \mathbf{U}_e$, thus $\mathbf{L} \mathbf{U}_e$ is expressed in the form of $\mathbf{U}_e$. Further we can derive the collocation of higher order differentiations as

$$ L^k \mathbf{U}_e = L^k \mathbf{B}_e \mathbf{A}_e = (L^k \mathbf{B}_e) \mathbf{B}_e^{-1} \mathbf{U}_e . \tag{6.2.7} $$

Now we can rewrite the formulation of local collocation method as a system of NAEs of $\mathbf{U}$

$$ \mathbf{E} \mathbf{U} - \mathbf{F}(\mathbf{U}, t) = \mathbf{0} , \tag{6.2.8} $$

where $\mathbf{t} = [t_1, t_2, \ldots, t_M]$ , $\mathbf{E} = \text{diag}([\mathbf{L} \mathbf{B}_1 \mathbf{B}_1^{-1}, \ldots, \mathbf{L} \mathbf{B}_M \mathbf{B}_M^{-1}, \ldots])$ and $\mathbf{U} = [\mathbf{U}_1^T, \ldots, \mathbf{U}_M^T]^T$.

Following the concept of collocation method, we approximate the guess function $\mathbf{x}_n(t)$ and the corrected function $\mathbf{x}_{n+1}(t)$ in the three modifications of LVIM respectively by $\mathbf{u}_n(t)$ and $\mathbf{u}_{n+1}(t)$, which are composed of the same set of basis functions $\mathbf{\Phi}$. By collocating at a set of points in the local time domain of the system, the iterative formulas of Algorithm-1, 2, and 3 are then reduced to algebraic iterative formulas for the values of the solution at collocation points.

6.2.1 Collocation of the Algorithm-1

Suppose that the solution of the nonlinear problem $\mathbf{L} \mathbf{x} = \mathbf{F}(\mathbf{x}, t)$, in a sub-interval $t \in [t_i, t_{i+1}]$, is approximated by $\mathbf{u}$. From the Algorithm-1, we have

$$ \mathbf{L} \mathbf{u}_{n+1}(t_m) - \mathbf{J}[\mathbf{u}_n(t_m), t_m] \mathbf{u}_{n+1}(t_m) = \mathbf{F}(\mathbf{u}_n(t_m), t_m) - \mathbf{J}[\mathbf{u}_n(t_m), t_m] \mathbf{u}_n(t_m) . \tag{6.2.9} $$
where \( m_t (m = 1, 2, \ldots, M) \) are the collocation points in the time interval \( t_i \) to \( t_{i+1} \). Following the transformations made in section 3, this formulation can be rewritten in matrix form as

\[
\mathbf{E} \mathbf{U}^{n+1} - \mathbf{J}(\mathbf{U}_n, t) \mathbf{U}^{n+1} = \mathbf{F}(\mathbf{U}_n, t) - \mathbf{J}(\mathbf{U}_n, t) \mathbf{U}_n.
\]  

(6.2.10)

By rearranging the matrices, it can be further expressed as

\[
\mathbf{U}^{n+1} = \mathbf{U}_n - \left[ \mathbf{E} - \mathbf{J}(\mathbf{U}_n, t) \right]^{-1} \left[ \mathbf{E} \mathbf{U}_n - \mathbf{F}(\mathbf{U}_n, t) \right].
\]  

(6.2.11)

Interestingly, this recursive formula happens to be the Newton-Raphson iterative algorithm for solving the NAEs generated by conventional collocation method. It gives a straightforward illustration of the relationship between Newton’s method and the variational iteration method, which is implied in the work of Inokuti, Sekine and Mura (1978).

### 6.2.2 Collocation of the Algorithm-2

Firstly, the Picard iteration method is derived here as a special case of LVIM to give a brief explanation. Considering the first order differential equations, the sequence of approximations given by Picard iteration is

\[
\frac{dx_{n+1}}{dt} = F(x_n, t), \quad x_{n+1}(t_i) = x(t_i) \quad \text{or} \quad \int_{t_i}^{t_{i+1}} F(x_n(\tau), \tau) d\tau, \quad t \in [t_i, t_{i+1}].
\]  

(6.2.12)

The adoption of collocation method will then generate an iteration formula for values of the solution at collocation points \( m_t (m = 1, 2, \ldots, M) \), which is

\[
\mathbf{E} \mathbf{U}^{n+1} = \mathbf{F}(\mathbf{U}_n, t), \quad \mathbf{u}^{n+1}(t_i) = x(t_i) \quad \text{or} \quad \mathbf{U}^{n+1} = \mathbf{U}_n - \mathbf{E} \left[ \mathbf{U}_n - \mathbf{F}(\mathbf{U}_n, t) \right].
\]  

(6.2.13)

The coefficient matrix \( \mathbf{E} \) is composed by \( \mathbf{E} = \text{diag}([L^{-1}B_1^{-1}, \ldots, L^{-1}B_k^{-1}, \ldots]). \) It should be noted that these two iterative formulas are not the same in practice. In the first formula, the boundary conditions are enforced by adding constrains \( \mathbf{u}_{n+1}(t_i) = x(t_i) \), which will introduce extra computational error. Conversely, the second formula is not plagued by this problem because the boundary conditions are naturally satisfied.

Obviously, if the basis functions are selected as the Chebyshev polynomials of the first kind, the second iterative formula stated above will give rise to the MCPI method (Bai and Junkins, 2011). The iteration formula works if only the collocations \( \mathbf{U}_n \) is updated in each step, so it is very simple and direct.

**Remark 1:**

The MCPI method is exactly the same as the zeroth order approximation of Algorithm-2, using the collocation method with Chebyshev polynomials as basis functions.

In a general case, if \( x(\tau) \) is approximated by \( x(t_i) \), we have

\[
x^{(K)}_{n+1} = x^{(K)}_{n} + \lambda G^{(K)}_{n} - T_t^{(K)} \lambda \frac{G^{(K)}_{n+1}(t_i)}{r}, \quad \mathbf{G}^{(K)}_{n+1, r} = \mathbf{G}^{(K)}_{n, r} + \lambda \mathbf{G}^{(K)}_{n+1, r}.
\]  

(6.2.15)

By collocating points in the local domain, it leads to

\[
\mathbf{u}^{(K)}_{n+1}(t_m) = \mathbf{u}^{(K)}_{n}(t_m) + T_0 \mathbf{G}^{(K)}_{n} - T_t^{(K)} \mathbf{G}^{(K)}_{n}(t_m) + \lambda \mathbf{G}^{(K)}_{n+1}(t_m),
\]  

(6.2.16)

where \( \mathbf{G}^{(K)}_{n, r} \) and \( T_t^{(K)} \) are denoted as \( \mathbf{G} \) and \( T_t \) for simplicity. It can be rewritten in matrix form as

\[
\mathbf{E}_{K+1} \mathbf{U}^{n+1} = \mathbf{E}_{K+1} \mathbf{U}_n + T_0 \mathbf{G}^{(K)}(t) - T_t \mathbf{G}^{(K)}(t) + \lambda \mathbf{G}^{(K)}(t_m),
\]  

(6.2.17)

of which \( \mathbf{E}_{K+1} = \text{diag}([LB_1^{-1}, \ldots, (LB_k^{-1}), \ldots]) \).

This iterative formula is very simple since there is no need to calculate the Jacobian matrix of nonlinear terms and its inverse, and all the matrices of coefficients are constant. However, the convergence domain of it could be very limited.
because \( x(\tau) \) is approximated too roughly.

Normally, it is more reasonable to use the integral form of the Algorithm-2:

\[
x_{n+1}(t) = x_n(t) + A_0(t)\int_0^t Gd\tau + A_1(t)\int_0^t \tau Gd\tau + \ldots + A_k(t)\int_0^t \tau^k Gd\tau.
\]  

(6.2.18)

The adoption of collocation method leads to

\[
U_{n+1} = U_n + A_0(t)\tilde{E}G(t) + A_1(t)\tilde{E}[\mathbf{t} \cdot \mathbf{G}(t)] + \ldots + A_k(t)\tilde{E}[\mathbf{t}^k \cdot \mathbf{G}(t)].
\]  

(6.2.18)

The integral form is superior to the differential form in the several aspects:

1. The integral form is mathematically correct. It considers the change of \( \lambda(\tau) \) along with \( x_n(\tau) \) in each iteration step, while the differential form fails to achieve that.

2. The integral form satisfies the initial conditions inherently. In contrast, the differential form needs additional constrains to guarantee that, which will bring in extra computational errors.

3. In the integral form, there is no need to calculate the inverse of \( E_x^T \), which could get ill-conditioned easily for high order interpolations (Dong, etc., 2014).

Overall, the integral form is more robust than the differential form, although the computational burden is slightly heavier. Unlike the collocation form of Algorithm-1, the collocation form of Algorithm-2 does not need to calculate the Jacobian matrix of nonlinear terms or its inverse in the computation. For high dimensional systems with complex structures, the simple operations of this formula can speed up the computation.

With \( T_0[\lambda] \), \( T_1[\lambda] \) and \( T_2[\lambda] \), the first three low order correctional formulas in differential form can be obtained by using the collocation formula of Algorithm-2.

Table 6.2.1 Differential forms of collocated Algorithm-2

<table>
<thead>
<tr>
<th>Approximation of ( \lambda )</th>
<th>Correctional formula in differential form</th>
</tr>
</thead>
<tbody>
<tr>
<td>0th order: ( \lambda = T_0 )</td>
<td>( E_1U_{n+1} = E_1U_n + T_0G(t) )</td>
</tr>
<tr>
<td>1st order: ( \lambda = T_0 + T_1(\tau - t) )</td>
<td>( E_2U_{n+1} = E_2U_n + T_0G^{(1)}(t) - T_1G(t) )</td>
</tr>
<tr>
<td>2nd order: ( \lambda = T_0 + T_1(\tau - t) + T_2(\tau - t)^2 )</td>
<td>( E_3U_{n+1} = E_3U_n + T_0G^{(2)}(t) - T_1G^{(1)}(t) + 2T_2G(t) )</td>
</tr>
</tbody>
</table>

Similarly, the integral forms are obtained as

Table 6.2.2 Integral forms of collocated Algorithm-2

<table>
<thead>
<tr>
<th>Approximation of ( \lambda )</th>
<th>Correctional formula in integral form</th>
</tr>
</thead>
<tbody>
<tr>
<td>0th order: ( \lambda = T_0 )</td>
<td>( U_{n+1} = U_n + T_0\tilde{E}G )</td>
</tr>
<tr>
<td>1st order: ( \lambda = T_0 + T_1(\tau - t) )</td>
<td>( U_{n+1} = U_n + (T_0 - [T_1 \cdot \mathbf{t}])\tilde{E}G + T_1\tilde{E}([\mathbf{t} \cdot \mathbf{G}]) )</td>
</tr>
<tr>
<td>2nd order: ( \lambda = T_0 + T_1(\tau - t) + T_2(\tau - t)^2 )</td>
<td>( U_{n+1} = U_n + (T_0 - [T_1 \cdot \mathbf{t}] + [T_2 \cdot \mathbf{t}^2])\tilde{E}G + (T_1 - 2[T_2 \cdot \mathbf{t}])\tilde{E}([\mathbf{t} \cdot \mathbf{G}]) + T_2\tilde{E}([\mathbf{t}^2 \cdot \mathbf{G}]) )</td>
</tr>
</tbody>
</table>
6.2.3 Collocation of the Algorithm-3

Since there exists the integration of nonlinear terms, we need to approximate the nonlinear terms with a set of basis functions first, so that it can be written as

$$F_e(x_n, t) = \sum_{n=1}^{N} \eta_n \phi_n(t) = \Phi_e(t)Y_e,$$  \hspace{0.5cm} (6.2.19)

where $Y_e$ can be determined from the relationship $F_e[u(t_m), t_m] = \Phi_e(t_m)Y_e, \ m = 1, 2, \ldots, M$. Therefore, the integral terms at collocation point $t_m$ can be expressed as

$$\int_{t_i}^{t_f} F_e(u_n, \tau)d\tau = L^{-1}\Phi_e(t_m)Y_e = [L^{-1}\Phi_e(t_m)]B^{-1}F_e(U_{an}, t),$$  \hspace{0.5cm} (6.2.20)

where $B = \{\Phi_e(t_1)^T, \Phi_e(t_2)^T, \ldots, \Phi_e(t_M)^T\}^T$ and $L^{-1}\Phi_e$ is the integration of the basis functions. Substituting it into the collocation form of the Algorithm-3 leads to

$$EU_{n+1} = F(U_n, t) - J(U_n, t)^T[E(U_n - F(U_n, t)],$$  \hspace{0.5cm} (6.2.21)

Compared to the collocation form of Algorithm-2, this iteration formula [Eq. (6.2.21)] includes the Jacobian matrix of nonlinear terms, but there is still no need to calculate the inverse of Jacobian matrix.

6.3 Application to forced Duffing equation

Similar to the conventional collocation methods, the basis functions will determine the specific forms of the iterative schemes proposed in section 5, while the selection of basis functions could be fairly flexible. The common types of basis functions used in collocation method include harmonics, polynomials, Radial Basis Functions (RBFs), Moving Least Square (MLS) functions and etc. For the sake of brevity and clarity, only one kind of basis functions is used in this paper to illustrate the methods of section 6, which is the first kind of Chebyshev polynomials. All the discussions made in the following are based on this type of basis functions. For clarity, we use the name “Chebyshev Local Iterative Collocation (CLIC) method” to denote the proposed methods with Chebyshev polynomials as basis functions. Thus, as counterparts of Eqs. (6.2.11), (6.2.18), and (6.2.21), we have CLIC-1, CLIC-2 and CLIC-3 respectively.

With the first kind of Chebyshev polynomials as the basis functions, the trial functions are expressed as

$$u_i(\xi) = \sum_{n=0}^{N} \alpha_n T_n(\xi), \ \xi = \frac{2t - (t_i + t_{i+1})}{t_{i+1} - t_i}.$$  \hspace{0.5cm} (6.2.22)

where $T_n(t)$ denotes the $n$th Chebyshev polynomial. A rescaled time $\xi$ is introduced herein so that the Chebyshev polynomials are defined in a valid range $-1 \leq \xi \leq 1$.

The Chebyshev polynomials of the first kind are defined by the recurrence relation

$$T_0(\xi) = 1, \ T_1(\xi) = \xi, \ T_{n+1}(\xi) = 2\xi T_n(\xi) - T_{n-1}(\xi),$$  \hspace{0.5cm} (6.2.23)

or by trigonometric functions

$$T_n(\xi) = \cos[n \arccos(\xi)], \ -1 \leq \xi \leq 1.$$  \hspace{0.5cm} (6.2.24)

The differentiations of $T_n(\xi)$ can be obtained by the properties of Chebyshev polynomials or simply by

$$\begin{align*}
LT_n(\xi) &= \frac{n \sin[n \arccos(\xi)]}{\sqrt{1-\xi^2}} \\
LT_n(\xi) \big|_{\xi=1} &= (\pm 1)^{n+1} n^2.
\end{align*}$$  \hspace{0.5cm} (6.2.25)

For higher order differentiations of Chebyshev polynomials, one may seek the help of Mathematica. With the properties of Chebyshev polynomials, the integration can be calculated conveniently by

$$\int T_n(\xi)d\xi = \frac{1}{2} \left( \frac{T_{n+1}}{n+1} - \frac{T_{n-1}}{n-1} \right).$$  \hspace{0.5cm} (6.2.26)

For better approximation, the collocation points are selected as the Chebyshev-Gauss-Lobatto (CGL) nodes, which are calculated from
Consider a Duffing-Holmes’s oscillator, of which the governing equation is
\[
\ddot{x} + c\dot{x} + k_1 x + k_2 x^3 = f \cos(\omega t), \quad x(0) = a, \quad \dot{x}(0) = b.
\] (6.2.27)

It can be rewritten as a system of first order differential equations
\[
\begin{align*}
\dot{x}_1 &= x_2, \\
\dot{x}_2 &= -c x_2 - k_1 x_1 - k_2 x_1^3 + f \cos(\omega t),
\end{align*}
\] (6.2.28)
in which the variables and nonlinear terms are
\[
x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad F(x, t) = \begin{bmatrix} x_2 \\ -c x_2 - k_1 x_1 - k_2 x_1^3 + f \cos(\omega t) \end{bmatrix}.
\] (6.2.29)

Accordingly, the Jacobian matrix is
\[
\begin{bmatrix}
0 & 1 \\
-k_1 - 3k_2 x_1^2 & -c
\end{bmatrix}.
\] (6.2.30)

The Duffing-Holmes’s equation is solved with the proposed CLIC methods. Both the 0th order and the 1st order correctional formulas of the CLIC-2 are used. In each local domain, the iteration of the proposed methods stops when
\[
U_n - U_{n+1} \leq 10^{-12}.
\]

The results of ODE45 are used as benchmarks to measure the computational errors. The relative accuracy and the absolute accuracy of ODE45 are set to be \(10^{-15}\).

The chaotic phase portrait and time response curves obtained by the CLIC methods and the ODE45 are plotted in Fig. 6.3.1, in which the Chebyshev polynomials of the first kind are used as the basis functions. The repetitive time interval for the proposed methods is set as \(\Delta t = 2\). The number of basis functions and collocation points are \(N = M = 32\). Obviously, the results are consistent even for \(t > 200\). For comparison, the RK4 is also used and the step length of RK4 is selected as 0.001. The computational errors of the methods, including RK4, are recorded and plotted in Fig. 6.3.2.
Figure 6.3.1 (a) Phase portrait obtained by ODE45. (b) Response curves obtained by the CLIC methods (dots) and ODE45 (solid line).
Figure 6.3.2 Discrepancies between CLIC methods, RK4 method and ODE45 on $x(t)$. (a) CLIC-1, (b) CLIC-2 (zeroth order), (c) CLIC-2 (first order), (d) CLIC-3.

As is shown in Fig. 6.3.2, the accuracy of the proposed methods is very high. Although the time intervals of the proposed methods are 2000 times larger than the time-step size of RK4, the computational errors of them are $10^5$ times smaller than that of RK4 on the whole. The accuracy can be improved further by including more Chebyshev polynomials in the interpolation or reducing the size of the time intervals of the present algorithms.

Figure 6.3.3 Iterative steps of CLIC methods.
Table 6.3.1 Performance of the methods on solving the forced Duffing equation.

<table>
<thead>
<tr>
<th>Methods</th>
<th>CLIC-1</th>
<th>CLIC-2 (0th order)</th>
<th>CLIC-2 (1st order)</th>
<th>CLIC-3</th>
<th>RK4</th>
<th>ODE45</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterative steps</td>
<td>546</td>
<td>2159</td>
<td>1195</td>
<td>1174</td>
<td>200000</td>
<td>188625</td>
</tr>
<tr>
<td>Total computational time</td>
<td>0.389</td>
<td>0.714</td>
<td>1.316</td>
<td>1.161</td>
<td>0.814</td>
<td>3.706</td>
</tr>
<tr>
<td>Maximal error</td>
<td>7.602e-07</td>
<td>5.178e-06</td>
<td>7.006e-07</td>
<td>3.414e-07</td>
<td>0.038</td>
<td>\</td>
</tr>
</tbody>
</table>

For a comprehensive evaluation of the proposed methods, the indices of computational efficiency, including the iterative steps, the computational time and the maximal errors are compared in Fig. 6.3.3 and Table 6.3.1. It can be seen that the CLIC-1, is the fastest method in this case. The CLIC-2 of the first order and the CLIC-3 behave very similarly in both the convergence speed and the computational speed. From Fig. 6.3.3, it is found that the iterative steps of CLIC-2 of the 0th order, which can also be referred to as MCPI method, is almost twice of the CLIC-3 or the CLIC-2 of the first order, and is almost four times of the CLIC-1.

Compared with RK4, the CLIC-1 and CLIC-2 (0th order) are superior in all aspects, especially CLIC-1. The other two proposed methods CLIC-2 (1st order) and CLIC-3 are also very efficient considering their high accuracy, although they cost a bit more time than the RK4. It is also noticed that the accuracy of the CLIC-3 is the highest while the CLIC-2 (0th order) is the lowest among all the proposed methods.

7. Application of variational iteration-collocation method in orbital mechanics

The variational iteration-collocation method is applicable to investigate long term dynamical behaviors of a broad class of strongly nonlinear systems, regardless of whether the system is high dimensional or low dimensional, autonomous or non-autonomous. To demonstrate the high accuracy and efficiency of the variational iteration-collocation method in practical problems, the orbit propagation and Lambert’s problem in perturbed environment are tackled.

Mathematically, the orbit propagation and Lambert’s problem correspond to the initial value problem (IVP) and two-point boundary value problem (TPBVP) respectively. In the case of solving the initial value problem, the initial conditions are automatically satisfied by the correctional formulas. Normally the number of collocation points $N$ and the number of basis functions $N$ are set to be equal, so that there are enough constrains to determine the coefficients of the basis functions. For a long time interval, the solution can be obtained in piecewise, by repetitively using the proposed method on each local time interval $t_i \leq t \leq t_{i+1}$. Since the basis functions we adopted in this paper is the Chebyshev polynomials of the first kind, the corresponding algorithm for solving initial value problems are named as Chebyshev Local Iterative Collocation (CLIC) method.

For solving the two-point boundary value problem, the boundary conditions can be enforced by imposing extra constrains on the correctional formulas. Take the first order correctional formula for instance, one may simply collocate at the boundary point and add the boundary condition as a part of the correctional formula, thus lead to

$$\begin{bmatrix} \mathbf{I} \\ \mathbf{E}_b \end{bmatrix} \mathbf{U}_{n+1} = \begin{bmatrix} \mathbf{U}_n + (\mathbf{T}_b - [\mathbf{T}_1 \cdot \mathbf{t}] \dot{\mathbf{E}}G + \mathbf{T}_1 \dot{\mathbf{E}}(\mathbf{t} \cdot \mathbf{G}) \\ \mathbf{U}_b \end{bmatrix},$$

(7.1)

$$\mathbf{E}_b \mathbf{U}_{n+1} = \mathbf{U}_b$$

describes the boundary condition of the problem. It should be noted that there could be other approaches to incorporate the boundary conditions, such as artificially adapting the first several coefficients of the basis functions in each iteration step, as is adopted by the MCPI method (Bai and Junkins, 2011, Woollands, etc., 2015, Fukushima, 1997).

7.1 Perturbed orbit propagation

In Earth-Centric Initial (ECI) frame, the governing equations of a spacecraft considering $J_2$ perturbation in the
gravity potential field can be expressed as

\[
\begin{aligned}
\dot{x} &= v_x, \quad \dot{y} = v_y, \quad \dot{z} = v_z, \\
\dot{v}_x &= -\frac{x\mu[2(x^2 + y^2 + z^2)^2 + 3(x^2 + y^2 - 4z^2)JR_e^2]}{2(x^2 + y^2 + z^2)^{3/2}}, \\
\dot{v}_y &= -\frac{y\mu[2(x^2 + y^2 + z^2)^2 + 3(x^2 + y^2 - 4z^2)JR_e^2]}{2(x^2 + y^2 + z^2)^{3/2}}, \\
\dot{v}_z &= -\frac{z\mu[2(x^2 + y^2 + z^2)^2 + 3(x^2 + y^2 - 2z^2)JR_e^2]}{2(x^2 + y^2 + z^2)^{3/2}},
\end{aligned}
\]  

(7.1.1)

where \( R_e \) is the radius of Earth, \( \mu \) is the gravitational constant.

\[
R_e = 67378137 \text{ m}, \quad \mu = 3.986 \times 10^{14} \text{ m}^3 \cdot \text{s}^{-2}, \quad J_2 = 1.082622675.
\]

The corresponding Hamiltonian per unit mass is

\[
H = \frac{1}{2}(v_x^2 + v_y^2 + v_z^2) - \frac{\mu}{\sqrt{x^2 + y^2 + z^2}} \left[ 1 - \frac{R_e^2}{2(x^2 + y^2 + z^2)} \left( \frac{3z^2}{x^2 + y^2 + z^2} - 1 \right) J_2 \right].
\]  

(7.1.2)

With the proposed CLIC method, two orbit propagation problems of the preceding system is solved. In case 1, the eccentricity of the orbit is \( e \approx 0.1 \), the initial conditions are

\[
(x \quad y \quad z) = (-0.3886 \quad 7.7388 \quad 0.6736) \times 10^6 \text{ m}, \quad (v_x \quad v_y \quad v_z) = (-3.5794 \quad 0 \quad 6.1997) \times 10^3 \text{ m}.
\]
The number of Chebyshev polynomials and collocation points are set as $N = M = 13$. The size of each local time interval is selected as $\Delta t = 500s$. The resulted orbit trajectory is presented in Fig. 7.1.1 (a). To assess the accuracy of the computational result, the relative error $\varepsilon$ of the Hamiltonian is recorded and plotted in Fig. 7.1.1 (b). $\varepsilon$ is calculated from

$$
\varepsilon = \left| \frac{H - H_0}{H_0} \right|,
$$

where $H_0$ is the Hamiltonian evaluated from the initial conditions, and $H$ is the Hamiltonian at each collocation point. The simulation is carried out for 100,000s, which is almost 15 revolutions. As is shown in Fig. 1, the solution obtained by the proposed CLIC method is of high accuracy. The relative computational error of the Hamiltonian is kept less than $10^{-13}$ during the simulation. This is comparable to the most accurate numerical result that can be obtained by the ODE45 function of MATLAB and the 12th/10th order Runge-Kutta-Nystrom integrator. Moreover, the CLIC method is also very efficient and computationally cost-saving. The total computational time is about 0.9s in the environment of MATLAB.

The initial conditions of case 2 ($e \approx 0.9$) are

$$
(x, y, z) = (-1.4, 2.1, 2.4249) \times 10^7 m, \quad (v_x, v_y, v_z) = (-1.6873, 0, 2.9226) \times 10^3 m
$$

The number of Chebyshev polynomials and collocation points are set as $N = M = 101$. The size of each local time interval is selected as $\Delta t = 5000s$. The total computational time is about 4s. The numerical result in this example agrees with the previous one. The computational error of Hamiltonian is kept under $10^{-13}$ during the simulation time of 100,000s.

For comparison, the MCPI method is also applied to the preceding examples. In each case, the same parameters $N$, $M$ and $\Delta t$ in CLIC method are used. The computational errors of MCPI method are plotted in Fig. 7.1.1 (b) and Fig. 7.1.2 (b). The performance indices of MCPI and CLIC, including the total computational time, the mean iteration steps in each local time interval, and the maximum value of $\varepsilon$, are listed in Table 7.1.1. As is shown, both MCPI method and CLIC method are computationally efficient. However, the numbers of mean iteration steps indicate that the CLIC method converges faster than MCPI method, and at the same time, the computational error $\varepsilon$ of CLIC is much lower than MCPI.

![Figure 7.1.1 (a) Orbit trajectory for low eccentricity case and (b) computational error of CLIC method and MCPI method](image-url)
Figure 7.1.2 (a) Orbit trajectory for high eccentricity case and (b) computational error of CLIC method and MCPI method

Table 7.1.1 The performance of CLIC method vs. MCPI method.

<table>
<thead>
<tr>
<th>Cases</th>
<th>Methods</th>
<th>Computational Time</th>
<th>Mean Iteration Steps</th>
<th>Maximum of ε</th>
</tr>
</thead>
<tbody>
<tr>
<td>e ≈ 0.1</td>
<td>CLIC</td>
<td>0.9 s</td>
<td>7</td>
<td>8.194e-14</td>
</tr>
<tr>
<td></td>
<td>MCPI</td>
<td>1.1 s</td>
<td>13</td>
<td>3.201e-10</td>
</tr>
<tr>
<td></td>
<td>CLIC</td>
<td>4 s</td>
<td>6</td>
<td>1.065e-13</td>
</tr>
</tbody>
</table>
In the above simulations, the step size of CLIC method is fixed and is set to be relatively small, so that the method is ensured to converge in each local time interval. It has a drawback that a large number of steps are over calculated, thus leading to the waste of computational resource. For example, in the high eccentricity case, the solution of the initial value problem during $0 \leq t \leq 50,000$ s can actually be obtained in one step instead of ten steps, and still enjoy a very low computational error ($\varepsilon$ being less than $10^{-14}$) using CLIC method. Being aware of that, one may use adaptive steps in practice to further improve the computational efficiency and accuracy of this method.

7.2 Perturbed Lambert’s problem

Two general transfer orbits are obtained using the proposed iterative collocation method. In the low earth orbit transfer, the initial position is $\left(-0.3889, 7.7388, 0.6736\right) \times 10^6$ m, the final position is $\left(-3.6515, -4.2152, 6.3103\right) \times 10^6$ m, and the transfer time is set as $t = 2500$ s. In the high eccentricity orbit transfer, the initial and final positions are selected as $\left(-1.4, 2.1, 2.4249\right) \times 10^7$ m and $\left(-3.1497, -0.0462, 5.4554\right) \times 10^7$ m respectively, while the transfer time is chosen as $T = 25000$ s.

In solving these two cases of orbit transfers, the number of collocation points and Chebyshev polynomials are both selected as $N = M = 64$. The initial guess for the solution is selected as a uniform straight line connecting the initial and final position.

Table 7.2.1 Two general transfer orbits obtained using the iterative collocation method

<table>
<thead>
<tr>
<th>Cases</th>
<th>Boundary velocities, m/s</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Initial</td>
<td>Final</td>
</tr>
<tr>
<td>Low Eccentricity</td>
<td>-3579.396500</td>
<td>1798.099253</td>
</tr>
<tr>
<td></td>
<td>0.008964</td>
<td>-5510.306049</td>
</tr>
<tr>
<td></td>
<td>6199.705320</td>
<td>-3124.253368</td>
</tr>
<tr>
<td>High Eccentricity</td>
<td>-1687.308996</td>
<td>20.559111</td>
</tr>
<tr>
<td></td>
<td>-0.025078</td>
<td>-1124.689391</td>
</tr>
<tr>
<td></td>
<td>2922.606386</td>
<td>-35.693697</td>
</tr>
</tbody>
</table>

![Graph](image-url)
The results obtained by the proposed iterative collocation method are plotted in Fig. 7.2.1, and are compared with the numerical solution obtained using the MATLAB built-in ODE45 function, of which the relative tolerance and the absolute tolerance are both set to $10^{-15}$. The discrepancies on final positions and velocities are measured and listed in Table 7.2.1 to assess the accuracy of the iterative collocation method. As is shown, with 64 CGL nodes as collocation points, the Lambert’s problems considering $J_2$ perturbation is solved in very high accuracy.

### 8. Conclusion

We presented a variety of computational methods for solving nonlinear dynamical systems. Based on the principle ideas these methods rooted in, they are classified into four groups. The methods in each group are tightly related since they evolve from the same origin. Ignoring the details in implementation, it is shown that the collocation, finite volume, Galerkin, finite element, boundary element, meshless method are all based on the theory of weighted residuals. Due to the different choices of primal or mixed formulations, global or local, symmetric or unsymmetric weak forms, global or local interpolations of trial functions and test functions, various methods can be developed for both well-posed and ill-posed problems. The finite difference methods are also widely used on investigating the dynamical behaviors of nonlinear systems. In solving initial value problems, the step size of finite difference method is normally selected very small to ensure the accuracy. For long term responses, the integration using finite difference method may need numerous steps since the approximation in each step can only be made in a very small interval. The computational error may accumulate rapidly in solving nonlinear systems if some common explicit methods such as classical RK4 method is used. The implicit method such as Newmark method is more stable than the explicit method. However, it is always accompanied with solving linear or nonlinear algebraic systems, which increases the computational burden. The history of asymptotic methods is inseparable from the nonlinear dynamics. Among them, the most representative is perturbation method. It is vastly used in nonlinear analysis due to its analytical properties. The theory of perturbation can be used flexibly in various problems. However, it is subjected to small parameter assumptions or weak nonlinearities, which is the same to some other methods like averaging method and Adomian decomposition method. Recently, some asymptotic methods that are applicable to strongly nonlinear problems are developed in literature, including the homotopy perturbation method and the variational iteration method.

The variational iteration-collocation methods are iterative semi-analytical algorithms for predicting long-term responses of strongly nonlinear dynamical systems, based on a local application of the variational iteration method,
wherein variations of the nonlinear term are retained. We labeled this improved sequential local variational iteration method as LVIM, and then further modified it into Algorithm-1, Algorithm -2, and Algorithm -3, respectively, wherein the variational iterative formulas: (1) do not contain the Lagrange multipliers; (2) the Lagrange multipliers are approximated by polynomial series; and (3) the Lagrange multipliers are approximated by exponential functions. The weak-form solutions of the iterative formulas of Algorithm-1, 2, and 3, are derived respectively, by using Dirac Delta functions as test functions in each local time interval. It is shown that the variational iteration-collocation methods are far superior in computational accuracy as well as time, as compared to classical RK4. The implementation of variational iteration-collocation method is also very simple and free from complex symbolic or numerical computations. In sum, the class of variational iteration-collocation methods are of high accuracy and efficiency. They are readily applicable to investigate long term dynamical behaviors of a broad class of strongly nonlinear systems, regardless of whether the system is high dimensional or low dimensional, autonomous or non-autonomous. The underlying applications of the variational iteration-collocation methods cover many different areas such as the vibration of aeroelastic structures, the perturbed orbital propagations, and even the stochastic problems.

References

Abramowitz, M. and Stegun, I., Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables (1968), US. Department of Commerce.


Fukushima, T., Picard iteration method, Chebyshev polynomial approximation, and global numerical integration of
Liu, C. S. and Atluri, S. N., An iterative algorithm for solving a system of nonlinear algebraic equations, \( \mathbf{F}(\mathbf{x}) = 0 \), using the system of ODEs with an optimum \( \alpha \) in \( \dot{x} = \alpha \mathbf{F} + (1-\alpha)\mathbf{B}^T \mathbf{F} \); \( B_j = \frac{\partial F_j}{\partial x_j} \), Computer Modeling in Engineering and Sciences, Vol. 73, No. 4 (2011), pp. 395-431. 
Liu, C. S. and Atluri, S. N., A globally optimal iterative algorithm using the best decent vector \( \dot{x} = \alpha \mathbf{F} + \mathbf{B}^T \mathbf{F} \), with the critical value \( \alpha_c \), for solving a system of nonlinear algebraic equations \( \mathbf{F}(\mathbf{x}) = 0 \), CMES: Computer Modeling in Engineering & Sciences, Vol. 84, No. 6 (2012), pp. 575-601. 
Li, D., Guo, S. and Xiang, J., Aeroelastic Dynamic Response and Control of an Airfoil Section With Control Surface
Strogatz S. H., Nonlinear Dynamics and Chaos: with Applications to Physics, Biology, Chemistry, and Engineering (1994), Addison-Wesley.
Wang, X. C. and Atluri, S. N., A Novel Class of Highly Efficient & Accurate Time-Integrators in Nonlinear Computational Mechanics, Computational Mechanics. (To be published)
Zhang, T., He, Y., Dong, L., Li, S., Alotaibi, A., and Atluri, S. N., Meshless local petrov-galerkin mixed collocation