Sofie E. Leon

Glaucio H. Paulino[1]
e-mail: paulino@illinois.edu

Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign, 205 North Matthews Avenue, Urbana, IL 61801

Anderson Pereira

Ivan F. M. Menezes

Group of Technology in Computer Graphics, Tecgraf, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, Rio de Janeiro, 22451, Brazil

Eduardo N. Lages
Center for Technology, Federal University of Alagoas, Maceio, Alagoas, 57072, Brazil

A Unified Library of Nonlinear Solution Schemes

Nonlinear problems are prevalent in structural and continuum mechanics, and there is high demand for computational tools to solve these problems. Despite efforts to develop efficient and effective algorithms, one single algorithm may not be capable of solving any and all nonlinear problems. A brief review of recent nonlinear solution techniques is first presented. Emphasis, however, is placed on the review of load, displacement, arc length, work, generalized displacement, and orthogonal residual control algorithms, which are unified into a single framework. Each of these solution schemes differs in the use of a constraint equation for the incremental-iterative procedure. The governing finite element equations and constraint equation for each solution scheme are combined into a single matrix equation, which characterizes the unified approach. This conceptual model leads naturally to an effective object-oriented implementation. Within the unified framework, the strengths and weaknesses of the various solution schemes are examined through numerical examples. [DOI: 10.1115/1.4006992]

1 Introduction

Nonlinear problems are prevalent in structural and continuum mechanics; however, one single nonlinear solution method may not be capable of solving any general nonlinear problem. Depending on the problem and the severity of the nonlinearities, modifications to solution algorithms are necessary to recover the entire equilibrium path. In an early work, Bergan et al. [1] stated:

"a computer program for nonlinear analysis should possess several alternative algorithms for the solution of the nonlinear system. These procedures should also allow for the possibility of an extensive control over the solution process by parameters that are input to the program. Such a scheme would lead to increased flexibility, and the experienced user has the possibility of obtaining improved reliability and efficiency for the solution of a particular problem."

This philosophy is the main motivating factor for the present work, which provides a modern theoretical and computational framework for the insightful statement by Bergan et al. [1].

Many authors have developed families of nonlinear solution schemes, which can be adjusted by the user depending on the problem. Mondkar and Powell [2] developed a library of algorithms based on the Newton-Raphson method. Seven solution schemes were formulated from 11 control parameters (stiffness update type and frequency, convergence tolerance, etc.) and tested on several nonlinear structural systems. Clarke and Hancock [3] used the concept of load increment from the standard or modified Newton-Raphson method to unify several nonlinear solution schemes through a single load factor. The specific incremental-iterative procedure depends on the chosen constraint equation, which is used to calculate the unifying load factor. The constraint equations are based on iterations at constant load, displacement, work, arc length, or minimum residual. Yang and Sheih [4] and Yang and Kuo [5] presented a similar library of nonlinear solvers unified through a single load parameter, and included the generalized displacement control method. More recently Rezaiee-Pajand et al. [6] unified five nonlinear solution schemes through a single general constraint equation. The schemes were identified by five different constraints, including minimizing error by means of its length, area, or perimeter, and then the strengths and weaknesses of each algorithm were evaluated.

The library of nonlinear solution schemes explored in this review is similar to its predecessors in that several solution schemes, defined by a constraint equation, are unified into a single space by means of a load parameter. The methods include load control, displacement control, work control, arc length control, generalized displacement control, and the orthogonal residual procedure, which until now has not been incorporated into a collection of unified schemes. The unified schemes are formulated and implemented such that (i) additional nonlinear solution schemes are readily incorporated and (ii) integration into a finite element analysis code is straightforward. The approach taken is aimed at widespread dissemination of the work and follows an educational philosophy. Moreover, a website with all of the components developed, including a tutorial, is provided to the interested reader.[2]

1.1 On Nonlinear Systems.

Nonlinear behavior can arise from either material or geometric nonlinearity. In the former, the constitutive relation describing the material is itself nonlinear and the structural response associated with physical phenomena such as plasticity or strain-softening must be captured. In the latter, nonlinearity is due to changes in geometry, arising from large strains and/or rotations, which enter the formulation from a nonlinear strain-displacement relationship, and may occur even if the constitutive relation is linear [7]. Furthermore, in geometric nonlinear problems, the applied loads will either have an effect on the deformed configuration, or the configuration will have an effect on the load (e.g., follower loads [8]).

Nonlinear problems arising from either geometric or material nonlinearity feature critical points along the solution path. Critical

[1]Corresponding author.

Manuscript received November 26, 2011; final manuscript received April 2, 2012; published online August 17, 2012. Assoc. Editor: J. N. Reddy.

[2]See www.ghpaulino.com/NLS_tutorial.html.

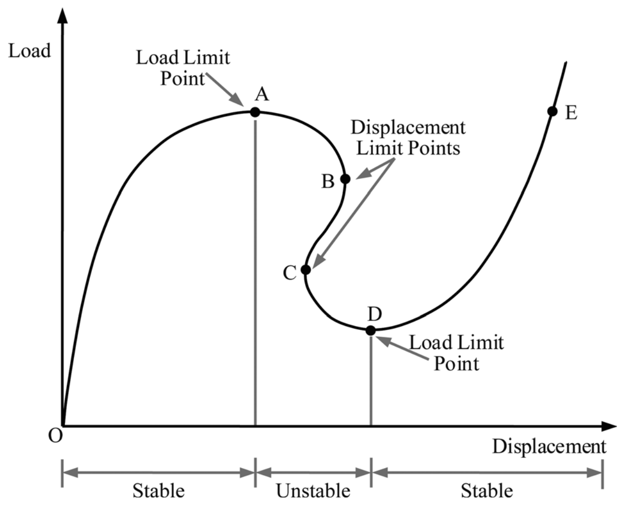


Fig. 1 Critical points in nonlinear equilibrium paths

points or stability points, shown in Fig. 1, are points on the solution path where the structure loses stability (e.g., buckling) or where bifurcation occurs (i.e., solution switches to two or more branches). Load limit points occur when a local maximum or minimum load is reached on the load versus displacement curve, as shown at points *A* and *D* in Fig. 1. A horizontal tangent is present at load limit points. Displacement limit points, shown at points *B* and *C* in Fig. 1, occur at vertical tangents on the solution curve. Displacement limit points are also commonly referred to as snap-back points or turning points in the literature. Methods capable of passing displacement limit points are said to capture snap-back behavior.

In this context, stability is directly related to load limit points, as shown in Fig. 1, where the region between the load limit points is unstable. The unstable region following a load limit point corresponds to a physical instability in the structural system, such as buckling. Using the theory of stability in a conservative system, a critical point occurs when the stiffness matrix is singular [9]. Some methods for tracing nonlinear load versus displacement curves are not capable of capturing behavior beyond a load limit point, and instead these methods yield snap-through behavior. The unstable region of the equilibrium path, shown as the dashed line in Fig. 2, is not traced; the curve snaps through and only the portions with increasing loads are captured.

Tracing an equilibrium path beyond the simple linear region and into a nonlinear region is an important task in structural analysis. While in some cases it may seem unnecessary to trace a path beyond the first load limit point, the full equilibrium path, including critical points and regions of instability, gives more information about the structural behavior than a simpler analysis [10]. Once a structure passes a load limit point, the nature of the unloading may be of importance to the analyst, rather than just the loading behavior. Additionally, information about the structural response past a displacement limit point may be of importance. For instance, if snap-back behavior was not captured in Fig. 1, the structure would appear to have a sharp drop in the load at point *B* and the nature of the unloading would be lost. For further discussion on nonlinear systems the reader is referred to Refs. [11–16].

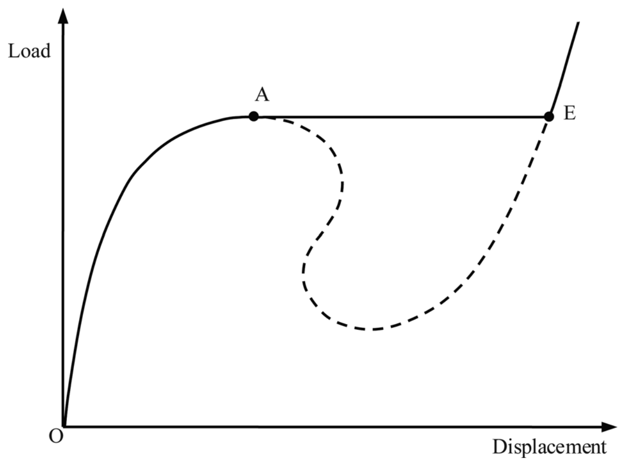


Fig. 2 Snap through behavior

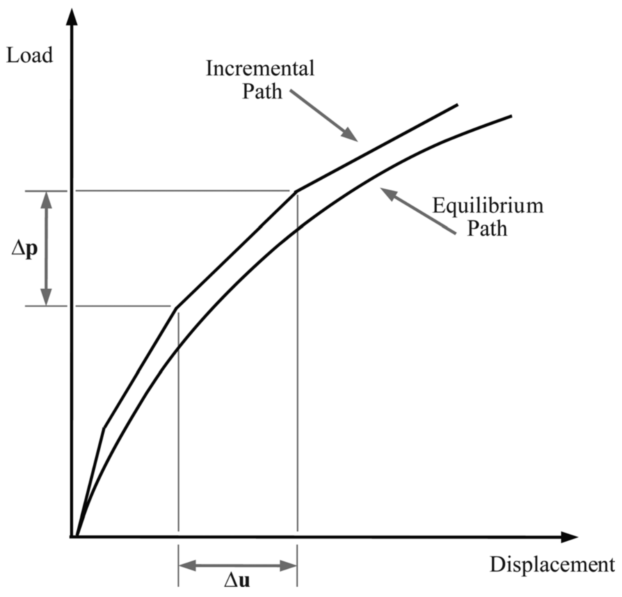


Fig. 3 Purely incremental procedure

1.2 Purely Incremental and Incremental-Iterative Methods. A nonlinear equilibrium path can be traced by means of either a purely incremental procedure or an incremental-iterative procedure. In an incremental procedure, shown in Fig. 3, the load is applied at relatively small load steps and the structure is assumed to respond linearly within each step. This method is simple to implement; however, as the solution progresses, it diverges considerably from the actual equilibrium path because equilibrium is not guaranteed at every step [2,5]. Conversely, in an incremental-iterative procedure, a series of iterations or corrections are performed at each incremental step until a specified convergence criterion is satisfied (see Fig. 4). Hence, the current increment in Fig. 4 advances the equilibrium curve from point *A* to point *B*, and the iterations occur within the increment, as indicated by the dotted line. If convergence is achieved before a maximum number of iterations are reached then equilibrium is satisfied for that step. Typically, convergence criteria is based on the ratio of the norm of the unbalanced forces to the norm of the applied loads at that incremental step. The incremental-iterative approach is accurate, but it comes with a higher computational cost. Incremental-iterative procedures are used for highly nonlinear behavior, and they will be the focus of this review.

The notation used in Fig. 4 denoting increments and iterations will be used in the remainder of this work: an increment is denoted with the superscript *i*, and iterations within the *i*th incremental step are denoted with the subscript *j*.

Due to the nature of the nonlinear problems, the internal forces, $\mathbf{q}_j^i \equiv \mathbf{q}(\mathbf{u}_j^i)$, are a function of the displacements, $\mathbf{u}_j^i$, and are not necessarily in equilibrium with the externally applied forces, $\mathbf{p}_j^i$. Thus, an unbalanced force or a residual $\mathbf{r}_j^i = \mathbf{p}_j^i - \mathbf{q}(\mathbf{u}_j^i)$ is generated.

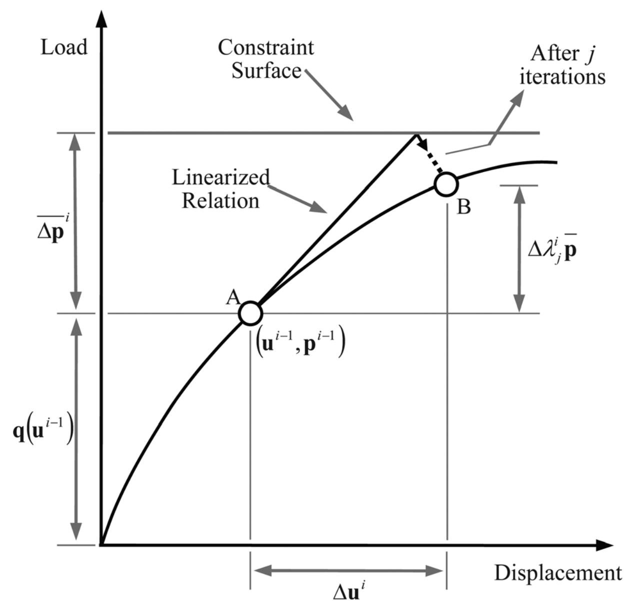
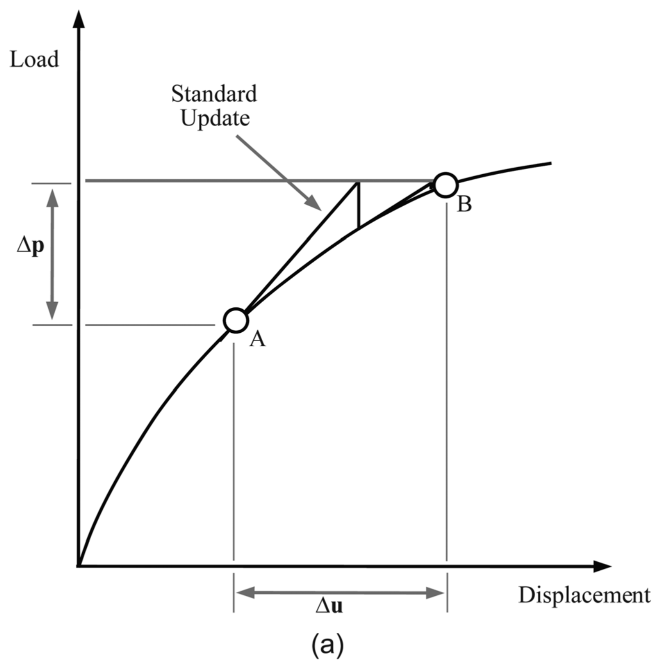

Load
Constraint Surface
After j iterations
Linearized Relation
$\overline{\Delta \mathbf{p}}^i$
$\Delta\lambda_j^i\overline{\mathbf{p}}$
A
B
$(\mathbf{u}^{i-1},\mathbf{p}^{i-1})$
$\mathbf{q}(\mathbf{u}^{i-1})$
$\Delta\mathbf{u}^i$
Displacement

Fig. 4 Incremental-iterative procedure



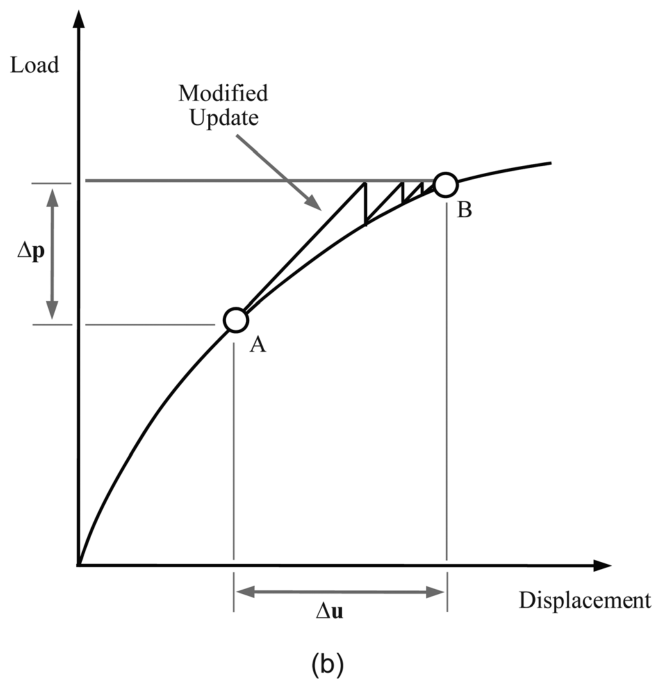


Fig. 5 (*a*) Standard and (*b*) modified updates to the tangent matrix

The displacement and external loads are computed by adding the contributions from the previously converged equilibrium configuration, (i.e., increment $i-1$) to the incremental updates at the current, jth, iteration of the current, ith, increment. Thus:

$$\mathbf{u}_j^i = \mathbf{u}^{i-1} + \Delta\mathbf{u}_j^i \tag{1}$$

$$\mathbf{p}_j^i = \mathbf{p}^{i-1} + \Delta\mathbf{p}_j^i \tag{2}$$

Similarly, the incremental updates at step i are computed by adding the contributions from the previous, $j-1$, iteration and iterative updates from the current, jth, iteration, i.e.,

$$\Delta\mathbf{u}_j^i = \Delta\mathbf{u}_{j-1}^i + \delta\mathbf{u}_j^i \tag{3}$$

$$\Delta\mathbf{p}_j^i = \Delta\mathbf{p}_{j-1}^i + \delta\mathbf{p}_j^i \tag{4}$$

where $\Delta\mathbf{u}_j^i$ and $\Delta\mathbf{p}_j^i$ are the incremental displacement and force vectors, respectively, and $\delta\mathbf{u}_j^i$ and $\delta\mathbf{p}_j^i$ are the iterative displacement and force vectors, respectively, in iteration j of step i. The residual vector is given by

$$\mathbf{r}_j^i = \mathbf{p}^{i-1} + \Delta\mathbf{p}_j^i - \mathbf{q}(\mathbf{u}^{i-1} + \Delta\mathbf{u}_j^i) \tag{5}$$

The governing system of nonlinear equations to be solved at the jth iteration of the ith incremental step is given by

$$\mathbf{K}_{j-1}^i\delta\mathbf{u}_j^i = \mathbf{p}_j^i - \mathbf{q}_{j-1}^i \tag{6}$$

where the tangent matrix, $\mathbf{K}_{j-1}^i$, may either be computed in accordance with a *standard* update method or with a *modified* update method. In the former method, the tangent matrix is calculated at the beginning of each iteration (Fig. 5(*a*)), while in the latter, the tangent matrix is only computed at the beginning of each incremental step and held constant for each iteration, i.e., $\mathbf{K}_{j-1}^i = \mathbf{K}_0^i$ for $j \geq 2$ (Fig. 5(*b*)). The modified method has a lower computational cost at each iteration than the standard version, but convergence is usually slower.

1.3 Paper Organization. First, a general review of nonlinear solution procedures is provided in Sec. 2. The unified nonlinear solution schemes are reviewed and cast into the $(N+1)$ dimensional space in Sec. 3. The object-oriented implementation of the unified schemes in the $(N+1)$ dimensional space is presented in Sec. 4. The unified schemes are evaluated through a set of numerical "experiments," which are shown and discussed in Sec. 5. Distribution of the code used to solve the example problems is described in Sec. 5.3. We conclude with a summary and potential research extensions in Sec. 6. Finally, the Appendix contains details of the analytical solution of one of examples explored in Sec. 5.1.

2 Related Work

To provide a comprehensive overview of the current state of the art in nonlinear solution procedures, methods that are not incorporated into the unified schemes are briefly reviewed here.

The literature on nonlinear solution schemes and their applications is extensive; therefore, the references in this section compose a rather incomplete list. For a more extensive discussion, the reader is directed to Refs. [17–21]. Different families of methods are addressed here including: (i) methods based on Adomian decomposition, (ii) methods based on Homotopy perturbation, (iii) variations of Newton's method (iv) methods to compute the locations of limit points, and (v) other libraries and toolkits.

(i) The Adomian decomposition method was originally developed to solve nonlinear stochastic differential equations [22]. Adomian and Rach [23] generalized the method to solve nonlinear systems of equations. The Adomian method consists of expressing the solution of a nonlinear equation as an infinite series. The nonlinear function used to represent the solution can be decomposed by the so-called Adomian polynomials. In a review by Adomian [17], it was stated that the method provides solutions that are more physically realistic than other methods because it solves nonlinear problems rather than linearizing them. Abbaoui and Cherruault [24] address difficulties associated with the Adomian decomposition method by computing Adomian polynomials and proving convergence of the decomposition series. Recent modifications of the Adomian decomposition technique have led to new methods with accelerated convergence [25–29]. Alternative decomposition methods have also been used to develop iterative schemes [30].

(ii) Another class of methods is based on coupling of homotopy and perturbation techniques. The homotopy is constructed using an embedding parameter $p \in [0, 1]$, then the perturbation technique is applied to arrive at an iterative scheme. For a more detailed explanation of the method, the reader is referred to the works by He [31–34]. Several authors have expanded upon this method to create new iteration schemes [35], or pass load limit points in nonlinear geometric analysis [36], for example. The method has also been used for various applications including nonlinear wave equations [37–39], nonlinear heat transfer [40–42], and calculating Adomian polynomials [43–45], to name a few.

(iii) Many methods are derived based on variations of Newton's method. For example, some methods have third-order convergence, unlike the traditional Newton method which is quadratically convergent, and they do not require calculation of second derivatives. However, these methods require evaluation of the function and first derivatives at multiple points [21,46]. Different quadrature formulas have been used to approximate the indefinite integral that arises with Newton's method to develop new incremental iterative procedures [47–49]. Another third order method was developed in Ref. [50] where Newton's method is used for the inverse function. While the previous methods do not require computation of second derivatives, Chebyshev-type methods approximate the second derivatives using evaluations of the function and first derivatives [51,52]. Families of higher order methods have been used to further improve convergence of Newton-Raphson type methods including a fourth- to eighth-order methods [53–55].

(iv) Some methods are successful at tracing beyond load and displacement limit points; however, many do not directly compute the location of such points. In the following studies, a rigorous approach is used to determine the location of stability points. Wriggers and Simo [56] appended the linearized eigenproblem as a constraint to characterize the presence of limit points to the nonlinear system of equations. The solution of the extended system therefore, includes the type of limit point (i.e., bifurcation or displacement limit point) in addition to the load factor and displacement field. Other methods for detecting limit points examine the determinant [57] or eigenvalues [58] of the tangent stiffness matrix, or use a homotopy method [59].

(v) In addition to individual solution schemes, libraries and tool kits for solving systems modeled by partial differential equations have also been developed. The Portable, Extensible Toolkit for Scientific Computation (PETSc) [60–62], for instance, is a suite of routines designed to solve large-scale applications through utilization of parallel linear and nonlinear solvers, including a parallel Newton-based solver. The libMesh library [63], which is a platform to perform numerical simulation of partial differential equations using arbitrary unstructured discretizations, utilizes the parallel solvers present in PETSc. Similarly, the Library of Continuation Algorithms, LOCA [64], was developed at Sandia National Laboratories in Albuquerque, New Mexico, to perform stability analysis on large scale problems by tracking multiple solution branches and bifurcation points. The software, LOCA, supports a variety of algorithms.

3 Unified Nonlinear Solution Schemes

Load control, displacement control, arc length control, work control, generalized displacement control, and orthogonal residual procedure schemes are presented in the context of the unified approach. The solution schemes are inherently different in their formulations and therefore feature unique constraint equations for the incremental-iterative procedure. The governing finite element equations and constraint equation are combined into a single matrix equation, which will be used to characterize the unified approach.

3.1 $N+1$ Dimensional Space Formulation. The formulation of the $(N+1)$ dimensional space begins with the general incremental-iterative procedure discussed in Sec. 1.2. The iterative load parameter, $\delta\lambda_j^i$, is introduced into Eqs. (2) and (4) by replacing $\delta\mathbf{p}_j^i$ with $\delta\lambda_j^i\overline{\mathbf{p}}$, where $\overline{\mathbf{p}}$ is a reference load vector [4], i.e.,

$$\mathbf{p}_j^i = \mathbf{p}^{i-1} + \Delta\mathbf{p}_{j-1}^i + \delta\lambda_j^i\overline{\mathbf{p}} \tag{7}$$

Equations (5), (6), and (7), are combined to give the following system of equations:

$$\mathbf{K}_{j-1}^i\delta\mathbf{u}_j^i = \mathbf{r}_{j-1}^i + \delta\lambda_j^i\overline{\mathbf{p}} \tag{8}$$

Equation (8) is a system of $(N+1)$ unknowns, N displacement components $(\delta\mathbf{u}_j^i)$ and one load parameter $(\delta\lambda_j^i)$, but only N equations. Therefore, an additional constraint equation must be added to the system, given by [5]

$$\mathbf{a}_j^i \cdot \delta\mathbf{u}_j^i + b_j^i\delta\lambda_j^i = c_j^i \tag{9}$$

Equations (8) and (9) yield a system of $(N+1)$ equations and $(N+1)$ unknowns

$$\begin{bmatrix} \mathbf{K}_{j-1}^i & -\overline{\mathbf{p}} \\ (\mathbf{a}_j^i)^T & b_j^i \end{bmatrix} \begin{Bmatrix} \delta\mathbf{u}_j^i \\ \delta\lambda_j^i \end{Bmatrix} = \begin{Bmatrix} \mathbf{r}_{j-1}^i \\ c_j^i \end{Bmatrix} \tag{10}$$

The augmented system matrix is no longer symmetric and has a significantly increased bandwidth due to the added load parameter. The solution of this system with a traditional method would be computationally undesirable with respect to both storage and efficiency. However, Batoz and Dhatt [65] presented a technique to overcome this problem, which consists of decomposing the iterative displacement vector into two parts

$$\delta\mathbf{u}_j^i = \delta\lambda_j^i \delta\mathbf{u}_{\mathrm{p}_j}^i + \delta\mathbf{u}_{\mathrm{r}_j}^i \tag{11}$$

then Eq. (8) becomes

$$\begin{aligned} \mathbf{K}_{j-1}^i \delta\mathbf{u}_{\mathrm{p}_j}^i &= \bar{\mathbf{p}} \\ \mathbf{K}_{j-1}^i \delta\mathbf{u}_{\mathrm{r}_j}^i &= \mathbf{r}_{j-1}^i \end{aligned} \tag{12}$$

It is clear that Eqs. (12) and (8) are mathematically equivalent by means of Eq. (11). Moreover, $\delta\mathbf{u}_{\mathrm{p}_j}^i$ and $\delta\mathbf{u}_{\mathrm{r}_j}^i$, are computed using the original system matrix, $\mathbf{K}_{j-1}^i$, and the banded and symmetric properties of the original system remain intact [66]. Finally, the load parameter is needed to compute the total displacement for the *j*th iteration of the *i*th incremental step. Solving Eq. (9) for the load parameter and combining with Eq. (11) yields

$$\delta\lambda_j^i = \frac{c_j^i - \mathbf{a}_j^i \cdot \delta\mathbf{u}_{\mathrm{r}_j}^i}{\mathbf{a}_j^i \cdot \delta\mathbf{u}_{\mathrm{p}_j}^i + b_j^i} \tag{13}$$

The constraint equation will be directly associated with a particular nonlinear solution scheme. Or, in other words, the formulation of each nonlinear solution scheme into the $(N+1)$ space will give rise to the constraint parameters, $\mathbf{a}_j^i$, b_j^i, and c_j^i.

3.2 Nonlinear Solution Schemes. The unified methods include load control, displacement control, work control, arc length control, generalized displacement control, and the orthogonal residual procedure. Each of the methods is reviewed and cast into the $(N+1)$ dimensional space in the following subsections.

3.2.1 Load Control Method (LCM). Traditionally, load control methods (LCM) or Newton-Raphson type methods have been the most popular for solving nonlinear system of equations [2,67]. Furthermore, many advances in nonlinear solvers consist of variations of the basic Newton-Raphson method [68,69].

External loads are computed at the first iteration of each incremental step and held constant throughout the remaining iterations in the step, as illustrated in Fig. 6. Hence, the load factor is

$$\delta\lambda_j^i = \begin{cases} \text{prescribed value} & \text{for} \quad j=1 \\ 0 & \text{for} \quad j \geq 2 \end{cases} \tag{14}$$

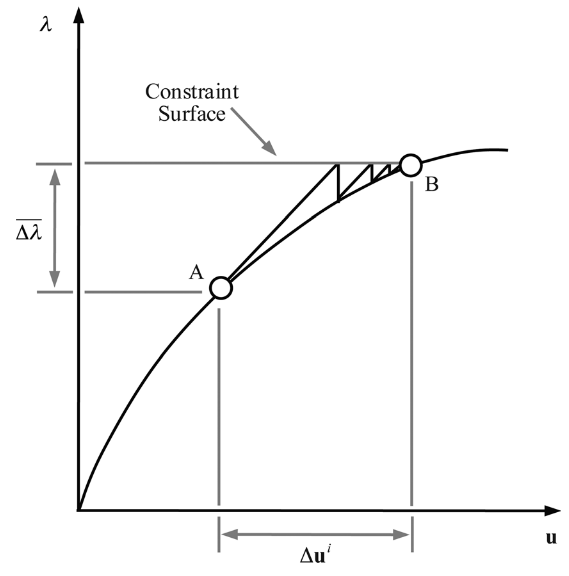


Fig. 6 Load control method

The constraint parameters are obtained by equating Eq. (14) and Eq. (13), such that

$$\mathbf{a}_j^i = 0 \tag{15}$$

$$b_j^i = 1 \tag{16}$$

$$c_j^i = \begin{cases} \overline{\Delta\lambda} & \text{for} \quad j=1 \\ 0 & \text{for} \quad j \geq 2 \end{cases} \tag{17}$$

where $\overline{\Delta\lambda}$ is a prescribed initial load factor. Because this method imposes the load factor, the system has only *N* unknowns and the decomposition of Eq. (8) is not needed. Instead, the system can be solved from Eq. (8) directly, i.e.,

$$\begin{aligned} \mathbf{K}_{j-1}^i \delta\mathbf{u}_1^i &= \overline{\Delta\lambda}\bar{\mathbf{p}} && \text{for} \quad j=1 \\ \mathbf{K}_{j-1}^i \delta\mathbf{u}_j^i &= \mathbf{r}_{j-1}^i && \text{for} \quad j \geq 2 \end{aligned} \tag{18}$$

This method is widely used and extremely robust, as indicated by Caballero et al. [70], who use it in modeling fracture of quasi-brittle material. This method has inherent weaknesses. Since the externally applied loads are kept constant, this method has difficulties near load limit points. Yang and Sheih [4] further showed that the constraint parameters imposed by this method will yield unbounded displacements near load limit points when the tangent matrix is nearly singular.

3.2.2 Displacement Control Method (DCM). Analogous to the LCM with a fixed load parameter, the displacement control method (DCM) uses a fixed displacement component as the control parameter to trace the equilibrium path. In multidegree of freedom systems, one displacement component is selected as the control component, denoted by $\delta u_{j_{\mathrm{ctrl}}}^i$, such that

$$\delta u_{j_{\mathrm{ctrl}}}^i = \begin{cases} \text{prescribed value} & \text{for} \quad j=1 \\ 0 & \text{for} \quad j \geq 2 \end{cases} \tag{19}$$

Equation (11) is solved for the control parameter with respect to the control component, giving

$$\delta\lambda_j^i = \frac{\delta u_{j_{\mathrm{ctrl}}}^i - \delta u_{\mathrm{r}_{j_{\mathrm{ctrl}}}}^i}{\delta u_{\mathrm{p}_{j_{\mathrm{ctrl}}}}^i} \tag{20}$$

On the first iteration, the residual will be zero, so $\delta\mathbf{u}_{\mathrm{r}_j}^i$ must also be zero, as evident from Eq. (12). The expression for $\delta\lambda_j^i$ reduces to the following:

$$\delta\lambda_j^i = \begin{cases} \dfrac{\delta u_{j_{\mathrm{ctrl}}}^i}{\delta u_{\mathrm{p}_{j_{\mathrm{ctrl}}}}^i} & \text{for} \quad j=1 \\ -\dfrac{\delta u_{\mathrm{r}_{j_{\mathrm{ctrl}}}}^i}{\delta u_{\mathrm{p}_{j_{\mathrm{ctrl}}}}^i} & \text{for} \quad j \geq 2 \end{cases} \tag{21}$$

The constraint parameters are obtained by comparing Eq. (21) and Eq. (13)

$$\mathbf{a}_j^i = \underbrace{[0, 0, \ldots 1, \ldots, 0]}_{\text{control displacement}} \tag{22}$$

$$b_j^i = 0 \tag{23}$$

$$c_j^i = \begin{cases} \overline{\Delta u} & \text{for} \quad j=1 \\ 0 & \text{for} \quad j \geq 2 \end{cases} \tag{24}$$

Finally, the load factor is given by

$$\delta\lambda_j^i = \begin{cases} \dfrac{\overline{\Delta u}}{\delta u^i_{\mathrm{p}_{j_{\mathrm{ctrl}}}}} & \text{for} \quad j = 1 \\ -\dfrac{\delta u^i_{\mathrm{r}_{j_{\mathrm{ctrl}}}}}{\delta u^i_{\mathrm{p}_{j_{\mathrm{ctrl}}}}} & \text{for} \quad j \geq 2 \end{cases} \tag{25}$$

where $\overline{\Delta u}$ is the prescribed initial displacement. A schematic of the DCM for a one dimensional problem is shown in Fig. 7. The prescribed displacement dictates the constraint surface on the first iteration in accordance with Eq. (25), then the direction of the load parameter changes for subsequent iterations. Notice that the ratio of the iterative load to the iterative displacement (i.e., $\overline{\Delta u}$ for $j = 1$ and $-\delta\mathbf{u}^i_{\mathrm{r}_{j_{\mathrm{ctrl}}}}$ for $j \geq 2$) remains fixed throughout the iterative process (see Eq. (25)).

The displacement control method can capture load limit points; however, it fails near displacement limit points. Yang and Sheih [4] showed that the control displacement, $\delta\mathbf{u}^i_{\mathrm{p}_{j_{\mathrm{ctrl}}}}$, approaches zero at displacement limit points, and the load parameter approaches infinity; hence numerical instability occurs at displacement limit points.

Variable Displacement Control Method (Variable DCM): In the standard DCM, the control parameter is selected intuitively or empirically, and remains fixed during the whole path-tracing process. To circumvent this drawback, Fujii et al. [71] devised a technique to systematically select the best control parameter. A decreasing component of the displacement vector may suggest that a displacement limit point is approaching in that component. However, the largest component of the displacement vector is the least likely component to experience a displacement limit point. The best candidate for the control component is therefore the one with the maximum absolute value of displacement. The sign of this component should also be retained to ensure the correct direction of equilibrium tracing. With this minor modification, the variable DCM can potentially capture displacement limit points. The constraint parameters and load factor for the variable DCM are the same as those of the standard DCM; however, the control displacement is not necessarily fixed throughout the incremental-iterative process. For example, the nonzero component of $\mathbf{a}_j^i$ in Eq. (22) may be different than the nonzero component of $\mathbf{a}_j^{i+1}$. Thus,

$$\mathbf{a}_j^i = \underbrace{[0, 0, \ldots 1, \ldots, 0]}_{\text{variable control displacement}}$$

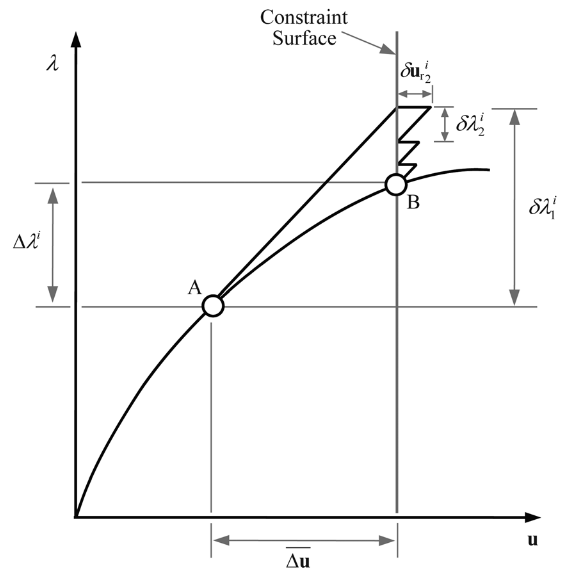


Fig. 7 Displacement control method

Simons and Bergan [72] also developed an extension of the traditional displacement control method that potentially captures snap-back. Unloading is steered via a reference displacement which is based on a linear combination of the displacement vectors of possible failure modes.

3.2.3 Arc Length Control Method (ALCM). The most typical example of a nonlinear solution scheme which considers simultaneous iteration on both the load and displacement variables is the arc length method (ALCM). The premise of the method is to constrain the solution path to an arc length, Δs_j^i. The arc length is calculated via a norm of the increment $(\Delta\mathbf{u}_j^i, \Delta\lambda_j^i)$. Several versions of the arc length method, including cylindrical, spherical, and elliptical can be represented by the general constraint equation

$$\Delta\mathbf{u}_j^i \cdot \Delta\mathbf{u}_j^i + \eta(\Delta\lambda_j^i)^2 = (\Delta s_j^i)^2 \tag{26}$$

where η is a non-negative real parameter, which is unique for each version of the arc length method. Figure 8 illustrates the incremental-iterative procedure of the arc length method for a one-dimensional problem. An initial arc length is determined in accordance with Eq. (26), then subsequent iterations lie on the constraint surface created by the arc. Iterations eventually converge at the intersection of the arc and the equilibrium path.

Spherical Arc Length Control Method: The spherical arc length method [10,73] sets the scaling parameter to one. The constraint equation becomes

$$\Delta\mathbf{u}_j^i \cdot \Delta\mathbf{u}_j^i + (\Delta\lambda_j^i)^2 = (\Delta s_j^i)^2 \tag{27}$$

The constraint equation now represents a sphere in three dimensional space, as shown in Fig. 9(*a*). Through several benchmark problems, Bellini and Chulya [74] demonstrated that in areas of very high slope, this method may have difficulties. The initial arc length may be adjusted such that it becomes very small; however, this increases the number of incremental steps to trace the entire equilibrium path, thereby increasing computational time.

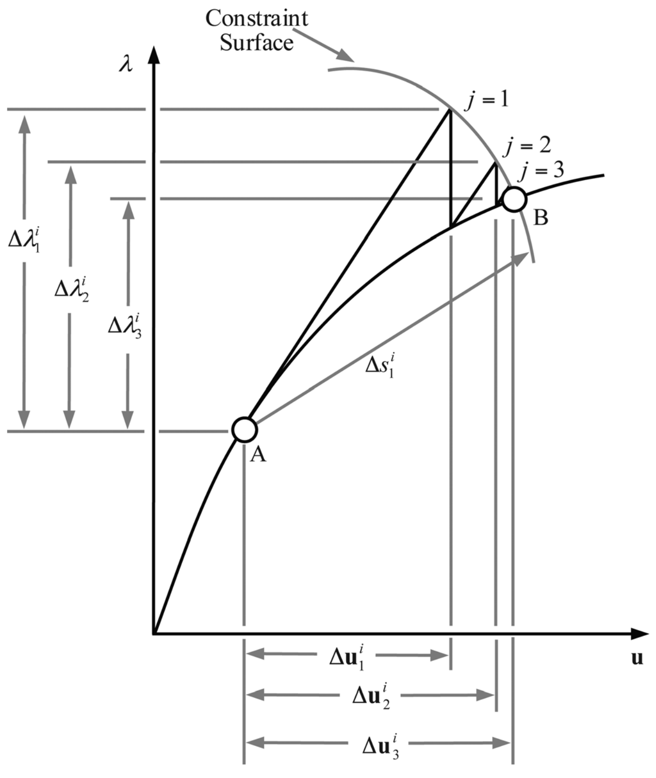


Fig. 8 Arc length control method

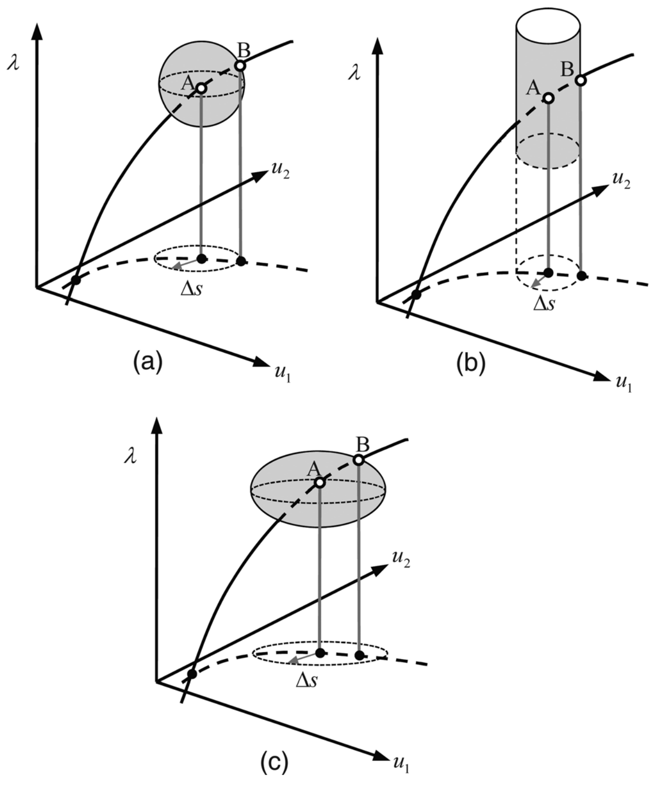


Fig. 9 (*a*) Spherical, (*b*) cylindrical and (*c*) elliptical version of the arc length control method

Cylindrical Arc Length Control Method: As indicated by Crisfield [10], the cylindrical arc length method consists of setting η to zero. Bellini and Chulya [74] illustrated the effectiveness of this version in capturing sharp turns at load limit points. The constraint equation represents a cylinder in three dimensional space,

$$\Delta\mathbf{u}_j^i \cdot \Delta\mathbf{u}_j^i = (\Delta s_j^i)^2 \tag{28}$$

as illustrated by Fig. 9(*b*).

Elliptical Arc Length Control Method: The most general form is the elliptical arc length method, where $\eta > 0$ and $\eta \neq 1$. Figure 9(*c*) shows that the constraint equation represents an ellipsoid in three dimensional space. Park [75] proposed a method where the scaling parameter, η, is the current stiffness parameter, introduced by Bergan et al. [1]. Bellini and Chulya [74] documented the success of this particular version of the elliptical arc length method.

Linearized Arc Length Control Method: The linearized arc length method was studied by Wempner [76] and Riks [67], and investigated further by Riks [9] and Crisfield [10]. The main feature of this version is that the iterations are processed on a hyperplane, as illustrated in Fig. 10(*a*). These methods are also referred to as orthogonal arc length methods because, in general, a norm constraint is imposed at the first iterative step, then an orthogonality condition is met in subsequent iterations of that increment. The orthogonality condition determines the type of linearized arc length method. For example, in the Fixed Normal Plane version, the iterative vectors $(\delta\mathbf{u}_j^i, \delta\lambda_j^i)$ are orthogonal to the initial incremental vector $(\Delta\mathbf{u}_1^i, \Delta\lambda_1^i)$. In the Updated Normal Plane version, the iterative vectors $(\delta\mathbf{u}_j^i, \delta\lambda_j^i)$ are orthogonal to the previous incremental vector $(\Delta\mathbf{u}_{j-1}^i, \Delta\lambda_{j-1}^i)$. The linearized versions are shown in Fig. 10(*b*).

The various versions of the ALCM are closely related, and Forde and Stiemer [77] unified the formulations of the normal and updated plane, and spherical versions through orthogonality relationships.

When the general nonlinear constraint equation of the arc length method is adopted, a quadratic equation in terms of $\Delta\lambda_j^i$ is

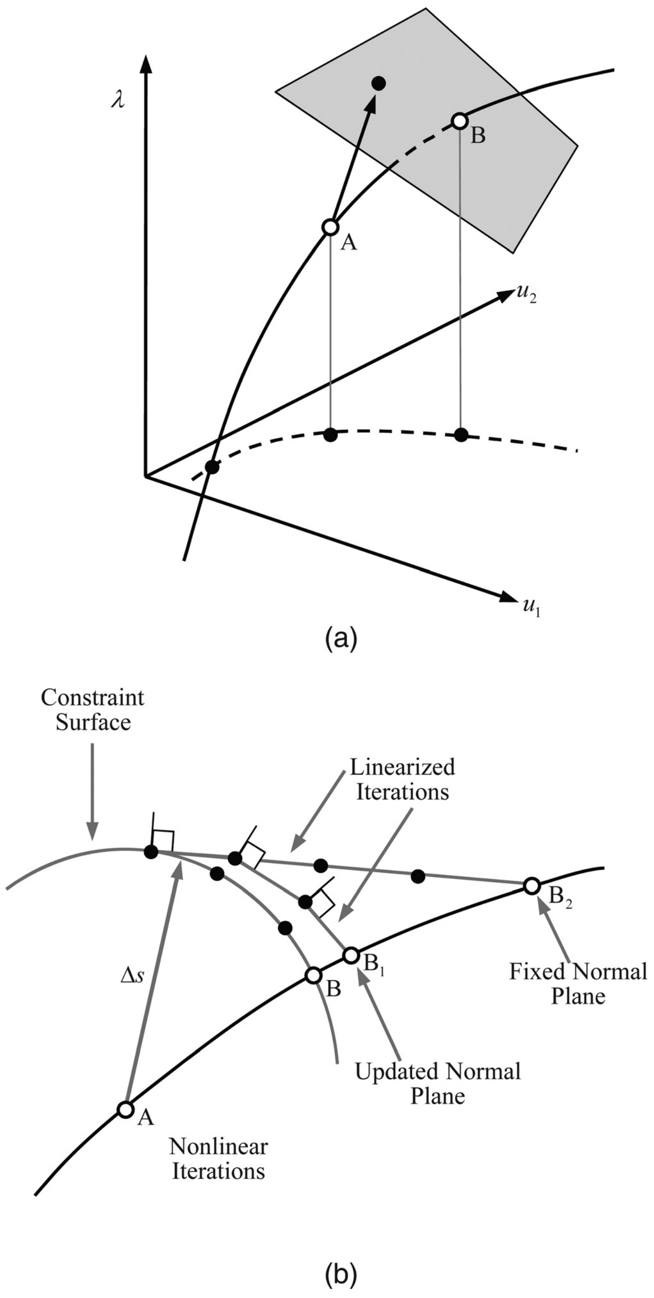


Fig. 10 (*a*) Schematic of the linearized arc length control method, (*b*) updated and fixed normal plane versions

obtained (i.e., Eq. (26)). It is necessary to adopt a set of rules to treat either real or complex roots obtained from the quadratic equation. For real roots, in general, the selected root is the one that corresponds to the smallest change in the direction of the iterative displacement vector compared to the previous displacement vector [74,78]. Lam and Morley [79] presented a methodology to avoid complex roots that can arise in the above mentioned quadratic equation by introducing a line search technique to find the real roots.

Ritto-Corrêa and Camotim [80] developed techniques to avoid complex roots of the quadratic equation using corrections to the incremental arc length size.

The constraint in Eq. (26) can be written with respect to the iterations rather than the increments

$$\delta\mathbf{u}_1^i \cdot \delta\mathbf{u}_j^i + \eta\delta\lambda_1^i\delta\lambda_j^i = (\Delta s_j^i)^2 \tag{29}$$

Then the following constraint parameters are obtained:

$$\mathbf{a}_j^i = \delta\mathbf{u}_1^i = \delta\lambda_1^i \delta\mathbf{u}_{\mathrm{p}1}^{\ i} \tag{30}$$

$$b_j^i = \eta\delta\lambda_1^i \tag{31}$$

$$c_j^i = \begin{cases} (\overline{\Delta S})^2 & \text{for} \quad j=1 \\ 0 & \text{for} \quad j \geq 2 \end{cases} \tag{32}$$

where $\overline{\Delta S}$ is the prescribed arc length to be assigned at the first iteration. The load factor is then given by

$$\delta\lambda_j^i = \begin{cases} \pm\dfrac{\overline{\Delta S}}{\sqrt{\delta\mathbf{u}_{\mathrm{p}1}^{\ i} \cdot \delta\mathbf{u}_{\mathrm{p}1}^{\ i} + \eta}} & \text{for} \quad j=1 \\ -\dfrac{\delta\mathbf{u}_1^i \cdot \delta\mathbf{u}_{\mathrm{r}j}^{\ i}}{\delta\mathbf{u}_1^i \cdot \delta\mathbf{u}_{\mathrm{p}j}^{\ i} + \eta\delta\lambda_1^i} & \text{for} \quad j \geq 2 \end{cases} \tag{33}$$

The sign of the load factor on the first iteration is not specified explicitly; however, it should depend on whether the systems is being loaded or unloaded. The load factor should be positive in the case of loading and negative for unloading.

The ability of the ALCM to change the sign of the load factor enables it to capture complex nonlinearities at load and displacement limit points; however, some weaknesses of the method have been identified in the literature. For instance, Carrera [81] documented the failure of several versions of the arc length method related to factors including the constraint equation, linearization, and computer precision and presented improved strategies. Ritto-Corrêa and Camotim [80] and Feng et al. [82] investigated scenarios where the ALCM converges incorrectly and presented a technique to determine the forward direction of the equilibrium path. Several authors have identified a potential shortcoming of this method associated with the units of terms in the load parameter. For example, $\delta\lambda_1^i$ is a scalar, while the displacements $\delta\mathbf{u}$ contain both translations and rotations, which are of different orders of magnitude [83]. In areas near displacement limit points with very high gradient, it is possible that $\delta\lambda_1^i$ is so large that the sign of $\delta\lambda_j^i$ depends only on the angle between $\delta\mathbf{u}_1^i$ and $\delta\mathbf{u}_{\mathrm{r}j}^{\ i}$. It follows that the sign of $\delta\lambda_j^i$ may change incorrectly, causing numerical divergence near displacement limit points [4]. It should be noted that the load factor should only change sign at areas of load limit points, not at displacement limit points. Al-Rasby [84] developed a method using diagonal scaling matrices to remove inconsistencies associated with mixed units (i.e., displacements, rotations, forces and moments). The ALCM has been used in a variety of applications including those associated with fracture [85–88], delamination [89,90], etc., and to improve nonlinear capabilities of numerical analysis techniques such as the Boundary Element Method (BEM) [91–95].

3.2.4 Work Control Method (WCM). Versions of the work control method (WCM) were studied by Simons and Powell [68], Bathe and Dvorkin [96], and Yang and McGuire [83]. One of the main motivations for this method was to overcome the issue of inconsistent physical units, as discussed for the ALCM. This method uses a constant work increment, δW_j^i, through the iterations of an incremental step

$$\delta W_j^i = \begin{cases} \text{prescribed value} & \text{for} \quad j=1 \\ 0 & \text{for} \quad j \geq 2 \end{cases} \tag{34}$$

The constraint equation is given by

$$\delta W_j^i = \delta\lambda_j^i \overline{\mathbf{p}} \cdot \delta\mathbf{u}_j^i \tag{35}$$

The constraint parameters can be determined directly from Eq. (35), i.e.,

$$\mathbf{a}_j^i = \delta\lambda_j^i \overline{\mathbf{p}} \tag{36}$$

$$b_j^i = 0 \tag{37}$$

$$c_j^i = \begin{cases} \overline{\Delta W} & \text{for} \quad j=1 \\ 0 & \text{for} \quad j \geq 2 \end{cases} \tag{38}$$

where $\overline{\Delta W}$ is the prescribed work increment. Equation (11) is substituted into $\delta\mathbf{u}_j^i$ in Eq. (35), and $\delta\mathbf{u}_{\mathrm{r}_j}^i$ is taken as zero on the first iteration, which yields

$$\delta\lambda_j^i = \begin{cases} \pm\sqrt{\left|\dfrac{\overline{\Delta W}}{\overline{\mathbf{p}} \cdot \delta\mathbf{u}_{\mathrm{p}1}^{\ i}}\right|} & \text{for} \quad j=1 \\ -\dfrac{\overline{\mathbf{p}} \cdot \delta\mathbf{u}_{\mathrm{r}1}^{\ i}}{\overline{\mathbf{p}} \cdot \delta\mathbf{u}_{\mathrm{p}1}^{\ i}} & \text{for} \quad j \geq 2 \end{cases} \tag{39}$$

Unlike the arc length method, the sign of the load parameter is easily determined. The term inside the square root, called the current stiffness parameter [83], indicates whether the system is being loaded or unloaded. The sign of this term should be applied to the load parameter: if the term is positive, the system is being loaded and the load parameter should increase; if it is negative, the load is decreasing and the load parameter should be negative.

Some weaknesses of the work control method have been examined by Yang and Sheih [4]. A potentially problematic situation occurs when there are a small number of degrees of freedom and the displacement associated with the major forcing direction tends to snap back (i.e., at a displacement limit point). The quantity $\overline{\mathbf{p}} \cdot \delta\mathbf{u}_{\mathrm{p}1}^{\ i}$ will tend to zero, forcing the load parameter to infinity. Thus, this method only has limited success near displacement limit points. Additionally, the presence of the reference load vector in the expression for $\delta\lambda_j^i$ may have adverse effects because it is somewhat arbitrary and does not necessarily represent the structural system.

Several authors have presented modifications for work control type methods. Lin et al. [97] developed a method to address the inconsistent units in the arc length method and weakness at snap-back of the work-control method using the work weighted state vector to control the incremental length throughout the solution tracing process. Chen and Blandford [98] presented a quadratically convergent algorithm that uses the incremental work to determine the size of the incremental step. A stabilized form of the work control method was presented by Kouhia [99] who reformulated the load parameter to be well defined even in areas of snap-back.

3.2.5 Generalized Displacement Control Method (GDCM). The generalized displacement control method (GDCM) was investigated by Yang and Sheih [4] as a result of the limitations discussed for other methods. Because the numerical stability of a nonlinear solution scheme depends on the selection of the constraint parameters, the following values were adopted:

$$\mathbf{a}_j^i = \delta\lambda_1^i \delta\mathbf{u}_{\mathrm{p}1}^{\ i-1} \tag{40}$$

$$b_j^i = 0 \tag{41}$$

$$c_j^i = \begin{cases} \text{generalized displacement} & \text{for} \quad j=1 \\ 0 & \text{for} \quad j \geq 2 \end{cases} \tag{42}$$

The constraint parameters are inserted into Eq. (13) such that

$$\delta\lambda_j^i = \frac{c_j^i - \delta\lambda_1^i \left(\delta\mathbf{u}_{\mathrm{p}1}^{\ i-1} \cdot \delta\mathbf{u}_{\mathrm{r}j}^{\ i}\right)}{\delta\lambda_1^i \left(\delta\mathbf{u}_{\mathrm{p}1}^{\ i-1} \cdot \delta\mathbf{u}_{\mathrm{p}_j}^i\right)} \tag{43}$$

which, simplifies to the following when $j=1$ because $\delta\mathbf{u}_{\mathrm{r}_1^i} = 0$ and $c_1^i = c$:

$$\delta\lambda_j^i = \begin{cases} \dfrac{c}{\delta\lambda_1^i \delta\mathbf{u}_{\mathrm{p}1}^{i-1} \cdot \delta\mathbf{u}_{\mathrm{p}1}^{i}} & \text{for} \quad j=1 \\ -\dfrac{\delta\mathbf{u}_{\mathrm{p}1}^{i-1} \cdot \delta\mathbf{u}_{\mathrm{r}j}^{i}}{\delta\mathbf{u}_{\mathrm{p}1}^{i-1} \cdot \delta\mathbf{u}_{\mathrm{p}_j}^{i}} & \text{for} \quad j \geq 2 \end{cases} \tag{44}$$

Moreover, c is solved for by letting $\delta\mathbf{u}_{\mathrm{p}1}^{1} = \delta\mathbf{u}_{\mathrm{p}1}^{0}$, which results in

$$c = (\delta\lambda_1^1)^2 \left(\delta\mathbf{u}_{\mathrm{p}1}^{1} \cdot \delta\mathbf{u}_{\mathrm{p}1}^{1}\right) \tag{45}$$

where $\delta\lambda_1^1 = \overline{\delta\lambda}$ is the prescribed initial control factor. The expression for $\delta\lambda_1^i$ becomes

$$\delta\lambda_1^i = \pm\overline{\delta\lambda}\left(\left|\frac{\delta\mathbf{u}_{\mathrm{p}1}^{1} \cdot \delta\mathbf{u}_{\mathrm{p}1}^{1}}{\delta\mathbf{u}_{\mathrm{p}1}^{i-1} \cdot \delta\mathbf{u}_{\mathrm{p}1}^{i}}\right|\right)^{\frac{1}{2}} \tag{46}$$

The GDCM adjusts the sign of the load parameter based on the stiffness of the system. The generalized stiffness parameter (GSP), defined below, will be positive for stiffening systems and negative for softening systems. The behavior of the GSP, including sign changes of the load parameter, is illustrated in Fig. 11. By definition

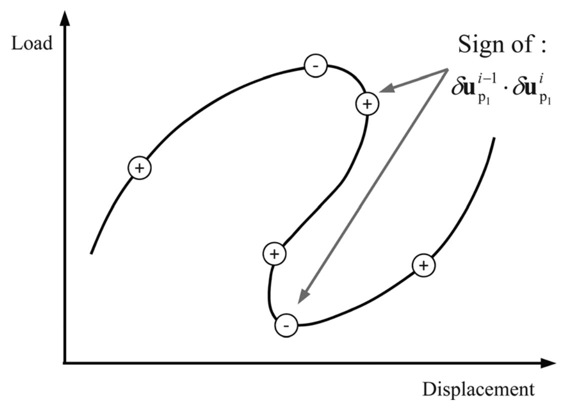
$$\mathrm{GSP} = \frac{\delta\mathbf{u}_{\mathrm{p}1}^{1} \cdot \delta\mathbf{u}_{\mathrm{p}1}^{1}}{\delta\mathbf{u}_{\mathrm{p}1}^{i-1} \cdot \delta\mathbf{u}_{\mathrm{p}1}^{i}} \tag{47}$$

Thus the load parameter at the first iteration of each incremental step is simply

$$\delta\lambda_1^i = \pm\overline{\delta\lambda}|\mathrm{GSP}|^{\frac{1}{2}} \tag{48}$$

Use of this physical quantity to represent the stiffness of the system makes this method computationally effective. The stiffness of the structure is measured with respect to the first incremental step, so stiffening and softening behavior are readily identified. Furthermore, the GSP changes sign only immediately after load limit points, meaning the direction of the load will only change at load limit points, not at displacement limit points. It can also be shown that the GSP remains bounded while tracing the equilibrium path.

The GDCM is successful at capturing complex nonlinear behavior at both load and displacement limit points, and has been used to study large deflection of trusses [100,101], and ultimate load carrying capacity of structures considering both member and structure instability [102]. The method has also been used to capture geometric and material nonlinearity associated with concrete-filled steel composite columns [103]. Additionally, the GDCM has been used by researchers modeling carbon nanotubes using



Load
Sign of :
$\delta\mathbf{u}_{\mathrm{p}_1}^{i-1} \cdot \delta\mathbf{u}_{\mathrm{p}_1}^{i}$
-
+
+
+
+
-
Displacement

Fig. 11 Generalized stiffness parameter used in the generalized displacement control method [4]

the truss rod model because of its accuracy and numerical stability at load and displacement limit points [104,105].

Connections have also been made between the GDCM and various versions of the ALCM. Cardoso and Fonseca [49] identified this method as an orthogonal arc length method. The constraint equation posed by the GDCM can be written as an orthogonal arc length constraint where adjustments in the radius of the arc are dependent on the value of the GSP. Additionally, a minor modification to the expression for $\mathbf{a}_j^i$ in Eq. (40) results in a version of the ALCM. Assuming $\mathbf{a}_j^i = \delta\lambda_1^i \delta\mathbf{u}_{\mathrm{p}1}^{i}$, and b_j^i and c_j^i are the same as in Eq. (40), then the load factor becomes

$$\delta\lambda_j^i = \begin{cases} \dfrac{c}{\delta\lambda_1^i \delta\mathbf{u}_{\mathrm{p}1}^{i} \cdot \delta\mathbf{u}_{\mathrm{p}1}^{i}} & \text{for} \quad j=1 \\ -\dfrac{\delta\mathbf{u}_{\mathrm{p}1}^{i} \cdot \delta\mathbf{u}_{\mathrm{r}j}^{i}}{\delta\mathbf{u}_{\mathrm{p}1}^{i} \cdot \delta\mathbf{u}_{\mathrm{p}_j}^{i}} & \text{for} \quad j \geq 2 \end{cases} \tag{49}$$

On the first iteration $(j=1)$ of the first incremental step $(i=1)$

$$c = \left(\overline{\delta\lambda}\right)^2 \delta\mathbf{u}_{\mathrm{p}1}^{1} \delta\mathbf{u}_{\mathrm{p}1}^{1} \tag{50}$$

Now the expression for $\delta\lambda_1^i$ becomes

$$\delta\lambda_1^i = \pm\overline{\delta\lambda}\left(\left|\frac{\delta\mathbf{u}_{\mathrm{p}1}^{1} \cdot \delta\mathbf{u}_{\mathrm{p}1}^{1}}{\delta\mathbf{u}_{\mathrm{p}1}^{i} \cdot \delta\mathbf{u}_{\mathrm{p}1}^{i}}\right|\right)^{\frac{1}{2}} \tag{51}$$

which means that the new GSP is

$$\mathrm{GSP} = \frac{\delta\mathbf{u}_{\mathrm{p}1}^{1} \cdot \delta\mathbf{u}_{\mathrm{p}1}^{1}}{\delta\mathbf{u}_{\mathrm{p}1}^{i} \cdot \delta\mathbf{u}_{\mathrm{p}1}^{i}} \tag{52}$$

Equation (51) can be rearranged as

$$\left(\delta\lambda_1^i\right)^2 \delta\mathbf{u}_{\mathrm{p}1}^{i} \cdot \delta\mathbf{u}_{\mathrm{p}1}^{i} = \left(\overline{\delta\lambda}\right)^2 \delta\mathbf{u}_{\mathrm{p}1}^{1} \cdot \delta\mathbf{u}_{\mathrm{p}1}^{1} \tag{53}$$

$$\delta\lambda_1^i \delta\mathbf{u}_{\mathrm{p}1}^{i} \cdot \delta\lambda_1^i \delta\mathbf{u}_{\mathrm{p}1}^{i} = \overline{\delta\lambda}\delta\mathbf{u}_{\mathrm{p}1}^{1} \cdot \overline{\delta\lambda}\delta\mathbf{u}_{\mathrm{p}1}^{1} \tag{54}$$

$$\delta\mathbf{u}_1^i \cdot \delta\mathbf{u}_1^i = \overline{\delta\mathbf{u}}_1^1 \cdot \overline{\delta\mathbf{u}}_1^1 \tag{55}$$

and since the incremental and iterative displacement at the first iteration of every incremental step are the same $\left(\delta\mathbf{u}_1^i = \Delta\mathbf{u}_1^i\right)$, then the constraint can be rewritten as

$$\Delta\mathbf{u}_1^i \cdot \Delta\mathbf{u}_1^i = \overline{\Delta\mathbf{u}}_1^1 \cdot \overline{\Delta\mathbf{u}}_1^1 = \left(\overline{\Delta s}_1^1\right)^2 \tag{56}$$

Notice that Eq. (56) is identical to that of the cylindrical ALCM presented in Eq. (28) for the first iteration $(j=1)$. Then, on subsequent iterations $(j \geq 2)$ we impose $\delta\mathbf{u}_1^i \cdot \delta\mathbf{u}_j^i = 0$. Thus, the minor modification to the parameters of the GDCM results in the linearized cylindrical ALCM.

3.2.6 Orthogonal Residual Procedure (ORP). The orthogonal residual procedure (ORP), investigated by Krenk [106], adjusts the load increment at each iterative step such that the current displacement increment is orthogonal to the current residual. The direction of the current displacement increment, $\Delta\mathbf{u}_j^i$, is taken as the best estimate of the direction of the actual displacement increment. The magnitude, however, will increase or decrease based on the projection of the residual force on the current displacement increment. The magnitude of the current displacement increment should not change; therefore, the orthogonality between the residual and the current displacement increment should be enforced, i.e.,

$$\mathbf{u}_j^i \cdot \Delta\mathbf{r}_j^i = 0 \tag{57}$$

as illustrated by Fig. 12(*a*).

To justify why the residual should not change the magnitude of the current displacement, consider the case where the residual and current displacement increment are not orthogonal, as shown in Fig. 12(*b*). First, note the order in which computations occur in the *j*th iteration. The incremental displacement is calculated using the tangent matrix and residual vector from the previous iteration

$$\delta\mathbf{u}_j^i = (\mathbf{K}_{j-1}^i)^{-1}\mathbf{r}_{j-1}^i \tag{58}$$

then the increment displacement is updated

$$\Delta\mathbf{u}_j^i = \Delta\mathbf{u}_{j-1}^i + \delta\mathbf{u}_j^i \tag{59}$$

and finally the residual is calculated. The direction of the current displacement increment should not change because it was calculated in the previous iteration and is taken as the best estimate of the actual displacement direction. If the residual is not orthogonal to the current displacement increment, then the iterative displacement increment, $\delta\mathbf{u}_j^i$, should have accounted for the induced displacement by the residual when it was calculated in the previous iteration. Changing the iterative displacement will also change the current displacement increment through Eq. (45). If the magnitude of the current displacement increment changes due to the residual, then the calculation of the current displacement increment in the previous iterative step was not optimal.

In the original ORP, a load factor, ξ_j^i, is used to increment the external load

$$\mathbf{p}_j^i = \mathbf{p}^{i-1} + \xi_j^i\overline{\Delta\mathbf{p}} \tag{60}$$

and the residual is given as follows:

$$\mathbf{r}_j^i = \mathbf{p}^{i-1} + \xi_j^i\overline{\Delta\mathbf{p}} - \mathbf{q}(\mathbf{u}^{i-1} + \Delta\mathbf{u}_j^i) \tag{61}$$

The optimal load increment factor is obtained by inserting Eq. (61) into the constraint in Eq. (57)

$$\xi_j^i = -\frac{\left[\mathbf{p}^{i-1} - \mathbf{q}(\mathbf{u}^{i-1} + \Delta\mathbf{u}_j^i)\right]\cdot\Delta\mathbf{u}_j^i}{\overline{\Delta\mathbf{p}}\cdot\Delta\mathbf{u}_j^i} \tag{62}$$

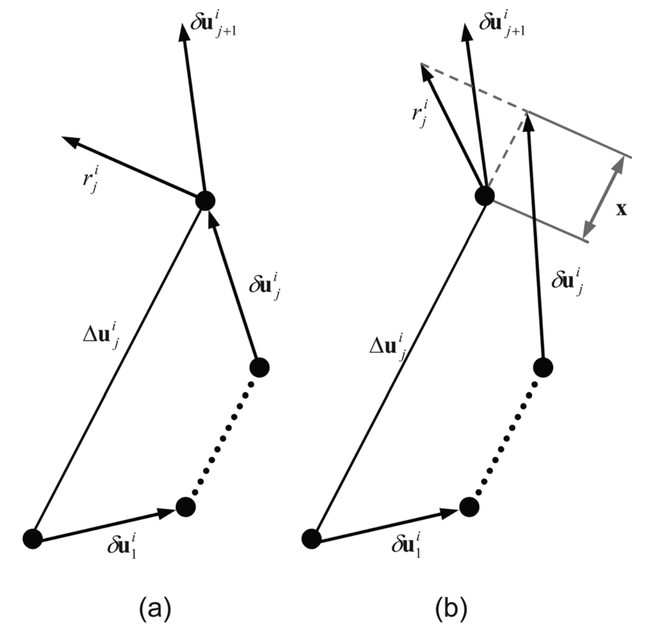


Fig. 12 (*a*) Orthogonality constraint for the orthogonal residual procedure, (*b*) error induced without the orthogonality constraint

The load increment factors, ξ_j^i, used in ORP are calculated at each iteration and are not dependent on any previous load increment factor. However, in the $(N+1)$ dimensional space each load increment factor is dependent on the previous, i.e.,

$$\Delta\lambda_j^i = \Delta\lambda_{j-1}^i + \delta\lambda_j^i \tag{63}$$

Figure 13 shows the dependence of $\Delta\lambda_j^i$ on previous iterations, while ξ_j^i is independent at each iteration.

To unify the ORP into the $(N+1)$ dimension space, the load factor, $\delta\lambda_j^i$, must be included in the constraint equation. The equivalence between the load factors is evident from Fig. 13

$$\mathbf{p}^{i-1} + \xi_j^i\overline{\Delta\mathbf{p}} = \lambda^{i-1}\overline{\mathbf{p}} + \Delta\lambda_{j-1}^i\overline{\mathbf{p}} + \delta\lambda_j^i\overline{\mathbf{p}} \tag{64}$$

Equation (64) is inserted into Eq. (61) to obtain the residual,

$$\mathbf{r}_j^i = (\lambda^{i-1} + \Delta\lambda_{j-1}^i + \delta\lambda_j^i)\overline{\mathbf{p}} - \mathbf{q}(\mathbf{u}^{i-1} + \Delta\mathbf{u}_j^i) \tag{65}$$

The load factor is found by solving Eq. (65)

$$\delta\lambda_j^i = \frac{\mathbf{r}_j^i + \left[\mathbf{q}(\mathbf{u}^{i-1} + \Delta\mathbf{u}_j^i)\right]}{\overline{\mathbf{p}}} - \left(\lambda^{i-1} + \Delta\lambda_j^{i-1}\right) \tag{66}$$

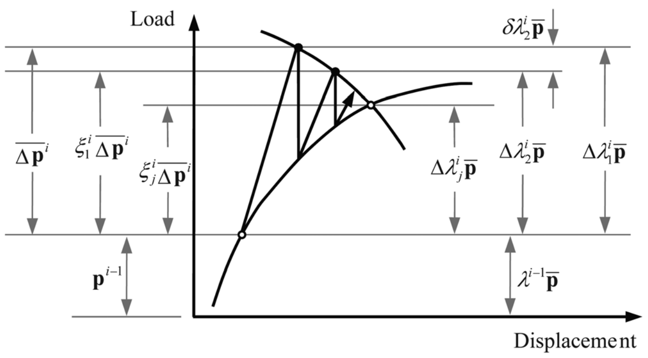
Then the orthogonality constraint is used to remove the residual term, and the resulting load factor is

$$\delta\lambda_j^i = \frac{\left[\mathbf{q}\left(\mathbf{u}^{i-1} + \Delta\mathbf{u}_j^i\right)\right]\cdot\Delta\mathbf{u}_j^i}{\left(\overline{\mathbf{p}}\cdot\Delta\mathbf{u}_j^i\right)} - \left(\lambda^{i-1} + \Delta\lambda_j^{i-1}\right) \tag{67}$$

Equation (67) is implemented in the $(N+1)$ dimensional space. At the first iteration of an incremental step (i.e., $j=1$), and initial load factor, $\overline{\delta\lambda}$, prescribed by the user, is assigned to $\delta\lambda_j^i$.

To improve the robustness of the ORP algorithm, certain conditions are checked/imposed throughout the incremental-iterative procedure. First, the direction of the displacement and load increments are reversed at load limit points. If the direction of the displacement increment is reversed relative to the previously converged displacement increment, then a load limit point has been passed and the sign of the load and displacement increments are changed. Secondly, maximum displacement and load factor increments are imposed to ensure they do not become unbounded near limit points. The maximum increments are computed as follows:

$$\text{If}\left\|\Delta\mathbf{u}_1^i\right\| > U_{\max} \Rightarrow \Delta\mathbf{u}_1^i = \frac{U_{\max}}{\left\|\Delta\mathbf{u}_1^i\right\|}\Delta\mathbf{u}_1^i \tag{68}$$



Load
$\delta\lambda^i_2\overline{\mathbf{p}}$
$\overline{\Delta\mathbf{p}}^i$
$\xi^i_1\overline{\Delta\mathbf{p}}^i$
$\xi^i_j\overline{\Delta\mathbf{p}}^i$
$\Delta\lambda^i_j\overline{\mathbf{p}}$
$\Delta\lambda^i_2\overline{\mathbf{p}}$
$\Delta\lambda^i_1\overline{\mathbf{p}}$
$\mathbf{p}^{i-1}$
$\lambda^{i-1}\overline{\mathbf{p}}$
Displacement

Fig. 13 Comparison of original ORP load factor and unified approach load factor

$$\text{If}\left\|\delta\mathbf{u}_j^i\right\| > U_{\max} \Rightarrow \delta\mathbf{u}_j^i = \frac{U_{\max}}{\left\|\delta\mathbf{u}_j^i\right\|}\delta\mathbf{u}_j^i \qquad (69)$$

where $U_{\max}$ is computed using a prescribed initial scale factor, β, such that

$$U_{\max} = \beta\left\|\Delta\mathbf{u}_1^i\right\| \qquad (70)$$

Finally, convergence problems may be addressed through the following modifications; however, they are not implemented here. If convergence is not met within the maximum iterations prescribed by the user, then the iterative procedure can be restarted with half the initial displacement, load increment, and maximum displacement, such that

$$\overline{\Delta\mathbf{p}} \leftarrow \frac{1}{2}\overline{\Delta\mathbf{p}} \qquad (71)$$

$$\Delta\mathbf{u}_1^i \leftarrow \frac{1}{2}\Delta\mathbf{u}_1^i \qquad (72)$$

$$U_{\max} \leftarrow \frac{1}{2}U_{\max} \qquad (73)$$

Information from the previous load increment may also be used to overcome convergence problems in the current increment. The load increment can be scaled by a ratio of the desired number of iterations, denoted Ite_D, and the actual number of iterations used to reach equilibrium in the last incremental step, denoted Ite_A [74,78]

$$\overline{\Delta\mathbf{p}}^{\,i} = \left(\frac{\text{Ite}_\text{D}}{\text{Ite}_\text{A}}\right)^{\alpha}\overline{\Delta\mathbf{p}}^{\,i-1} \qquad (74)$$

The exponent α is typically chosen to be in the range [0.5, 2.0]. Of course, this procedure is dependent on the behavior of the previous step; thus it may be ineffective for problems with very sudden changes. The original ORP has the potential to capture complex nonlinearities at load and displacement limit points, and it has recently been used in conjunction with the extended finite element method (XFEM) to solve nonlinear equations associated with cohesive crack growth [107,108].

Despite the reported success of the ORP, further investigation by Krenk and Hededal [109] and later by Kouhia [99] indicated that the original formulation has weaknesses around displacement limit points. Krenk and Hededal [109] modified the procedure by imposing a second orthogonality constraint that adjusts the displacement iteration in addition to the first constraint that adjusts the load increment. The original orthogonality constraint, $\mathbf{r}_j^i \cdot \Delta\mathbf{u}_j^i = 0$, and a modified Newton-Raphson approach with quasi-Newton modifications of the stiffness matrix are combined to obtain the second orthogonality condition between the current displacement iteration and internal force increment $\Delta\mathbf{q}_j^i \cdot \delta\mathbf{u}_j^i = 0$. This version of the method was successfully used by researchers investigating shell elements in a geometrically nonlinear analysis [110].

Kouhia [99] presented a stabilized version of the original ORP by relaxing the orthogonality constraint near displacement limit points. Similar to the approach to unify the ORP into the $(N+1)$ space, the load increment factors, ξ_j^i, were modified such that they would be additive within an incremental step. Near displacement limit points the load factor is

$$\delta\lambda_j^i = \frac{\Delta\mathbf{u}_j^i \cdot \mathbf{r}_{j-1}^i}{\text{sign}\left(\mathbf{u}_{j-1}^i \cdot \bar{\mathbf{p}}\right)\left\|\mathbf{u}_{j-1}^i\right\|\|\bar{\mathbf{p}}\|\left|\cos(\Delta\mathbf{u}_j^i, \bar{\mathbf{p}})\right| + \tau} \qquad (75)$$

where the following definition is used for the inner product:

$$\cos(\mathbf{a},\mathbf{b}) = \frac{\mathbf{a}\cdot\mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|} \qquad (76)$$

and τ is a dimensionless non-negative stabilization parameter that can be related to the current stiffness parameter proposed by Bergan et al. [1]. Hence, τ should be large at displacement limit points and small at load limit points. Kouhia [99] demonstrated the improvement of the stabilized version over the original ORP through numerical examples.

3.3 Incremental-Iterative Procedure in $N+1$ Space. The incremental-iterative procedure is illustrated in Fig. 14. The flow chart represents the operations performed on the jth iteration of the ith incremental step. The diamond-shaped boxes are conditional statements, and the computation of the load factor, indicated by the shaded elliptical box, is unique for each algorithm, as discussed in the previous sections.

4 Computational Implementation of the Unified Schemes

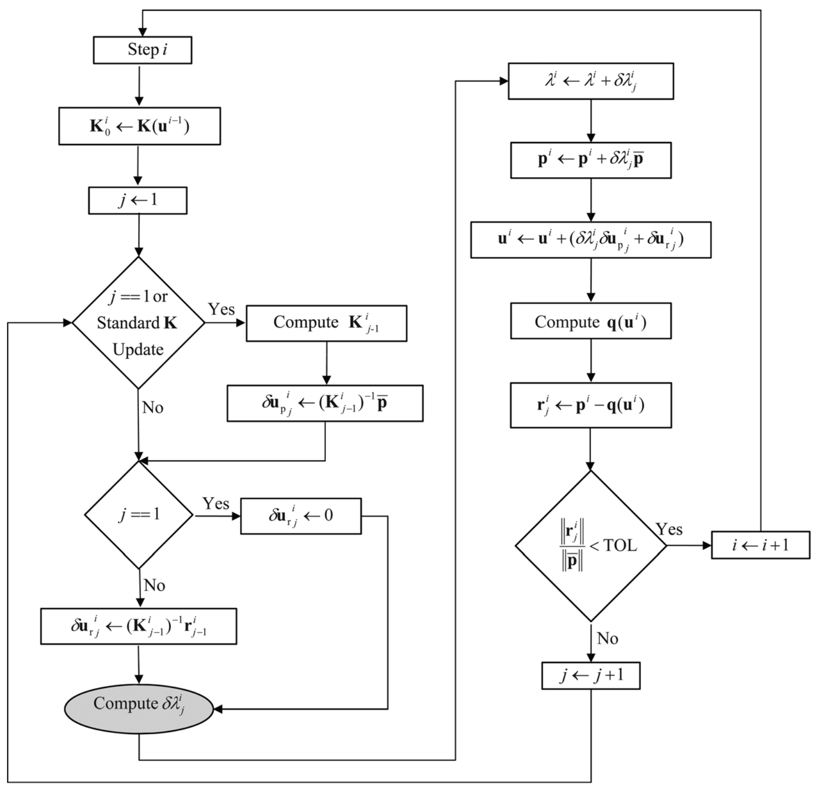
Object oriented design principles and programming can be utilized in the implementation of a unified solution scheme to provide a library that is flexible, extendible, and easy to understand. Most existing nonlinear solution algorithms are associated with large finite element software packages [62,111]. Modifying or extending these packages for solving different nonlinear systems of equations is not straightforward due the complexity of their object oriented structure. Furthermore, the cost of the steep learning curve for new developers who wish to utilize these packages tends to become discouragingly high with time. We aimed to streamline that complexity presenting a simple and robust object-oriented C++ library, called NLS++ (Nonlinear Solver). The library implements the nonlinear solution schemes discussed in the previous section. Through the unified approach, the solvers share a common interface and only vary in the computation of the load parameter, which depends on the constraint equation of each solver. This library is used to thoroughly test the solution schemes and characterize their performance in capturing various nonlinearities.

4.1 Class Hierarchy. The NLS++ code is organized into three distinct components: Model, Control, and LinearSolver. A separate application class creates instances of these components to execute the nonlinear analysis. Figure 15 shows the current class organization adopted by the NLS++ library.

4.1.1 Model Class. The purpose of Model class is to represent the problem to be solved. The primary function of the model class is to compute information associated with the system of equations, including the tangent matrix, and internal and external load vectors.

The base class/derived class paradigm is used to interface with the authors' applications. One application is included in the code distribution (see Sec. 5.3), and it includes model classes for bar and beam elements and for a few nonlinear functions. This application was used to generate the results presented in Sec. 5.

4.1.2 LinearSystem Class. The LinearSystem class is responsible for solving a linear system of equations, i.e., $\mathbf{Ax} = \mathbf{b}$. Hence, the LinearSystem needs to be extensible in order to accommodate different storage schemes (e.g., band, profile, sparse, etc.) and different numerical strategies (e.g., direct or iterative) for solving the system of equations. Several algorithms and corresponding software codes implementing linear solvers have appeared in recent years; for example, PARDISO [112,113] and UMFPACK [114]. The user can easily extend LinearSystem by selecting the most appropriate solver for his/her problem. Moreover, performance



Step i
$\mathbf{K}_0^i \leftarrow \mathbf{K}(\mathbf{u}^{i-1})$
$j \leftarrow 1$
$j == 1$ or Standard **K** Update
Yes
Compute $\mathbf{K}^i_{j\text{-}1}$
$\delta\mathbf{u}^i_{\mathrm{p}_j} \leftarrow (\mathbf{K}^i_{j-1})^{-1}\bar{\mathbf{p}}$
No
$j == 1$
Yes
$\delta\mathbf{u}^i_{\mathrm{r}_j} \leftarrow 0$
No
$\delta\mathbf{u}^i_{\mathrm{r}_j} \leftarrow (\mathbf{K}^i_{j-1})^{-1}\mathbf{r}^i_{j-1}$
Compute $\delta\lambda^i_j$
$\lambda^i \leftarrow \lambda^i + \delta\lambda^i_j$
$\mathbf{p}^i \leftarrow \mathbf{p}^i + \delta\lambda^i_j\bar{\mathbf{p}}$
$\mathbf{u}^i \leftarrow \mathbf{u}^i + (\delta\lambda^i_j\delta\mathbf{u}^i_{\mathrm{p}_j} + \delta\mathbf{u}^i_{\mathrm{r}_j})$
Compute $\mathbf{q}(\mathbf{u}^i)$
$\mathbf{r}^i_j \leftarrow \mathbf{p}^i - \mathbf{q}(\mathbf{u}^i)$
$\frac{\|\mathbf{r}^i_j\|}{\|\bar{\mathbf{p}}\|} < \mathrm{TOL}$
Yes
$i \leftarrow i+1$
No
$j \leftarrow j+1$

Fig. 14 Incremental-iterative procedure of the unified scheme

can be accelerated by using parallel processing, employing MPI-based implementations; for example PETSc [60–62]. Finally, the total runtime can be reduced for a sequence of linear systems that change slowly by using iterative solvers based on recycling subspaces of search spaces corresponding to previous solutions [115,116].

4.1.3 Control Class. The Control class is the engine of the NLS++ code, in the sense that it is responsible for tracing the equilibrium path by means of the incremental-iterative procedure. Classes can be derived from the Control class to obtain the various nonlinear solution schemes discussed in Sec. 3.2. The derived

Algorithm 1. Unified approach implemented in the Control class

```
1: for each iteration $j$ of step $i$ do
2:     Get global tangennt stiffiness matrix ($\mathbf{K}^i_j$) from Model class
3:     Compute $\delta\mathbf{u}^i_{\mathrm{p}_j}$ according to
           $\delta\mathbf{u}^i_{\mathrm{p}_j} \leftarrow (\mathbf{K}^i_{j-1})^{-1}\bar{\mathbf{p}}$
4:     Compute $\delta\mathbf{u}^i_{\mathrm{r}_j}$ according to
           $\delta\mathbf{u}^i_{\mathrm{r}_j} \leftarrow (\mathbf{K}^i_{j-1})^{-1}\mathbf{r}^i_{j-1}$
5:     Compute $\delta\lambda^i_j$ according to the selected solution scheme
6:     Update total load factor according to
           $\lambda^i \leftarrow \lambda^i + \delta\lambda^i_j$
7:     Update external load vector according to
           $\mathbf{p}^i \leftarrow \mathbf{p}^i + \delta\lambda^i_j\bar{\mathbf{p}}$
8:     Update total displacement vector according to
           $\mathbf{u}^i \leftarrow \mathbf{u}^i + \delta\lambda^i_j\delta\mathbf{u}^i_{\mathrm{p}_j} + \delta\mathbf{u}^i_{\mathrm{r}_j}$
9:     Get global internal force vector ($\mathbf{q}(\mathbf{u}^i)$) from Model class
10:    Compute unbalance load vector according to
           $\mathbf{r}^i_j \leftarrow \mathbf{p}^i - \mathbf{q}(\mathbf{u}^i)$
11:    if Convergence achieved then
12:        Next step $i \leftarrow i+1$
13:    else
14:        Next iteration $j \leftarrow j+1$
15:    end if
16: end for
```

Algorithm 2. Load factor implemented in the Load Control class

```
1: if ($j ==$ First Iteration) then
2:     $\delta\lambda^i_j \leftarrow$ Prescribed load increment
3: else
4:     $\delta\lambda^i_j \leftarrow 0$
5: end if
```

class has one particular function that is not implemented in the parent class, called Lambda, which computes the load factor for each nonlinear scheme. Algorithm 1 shows the main steps of the unified approach adopted here. For the sake of completeness, and also to illustrate how the incremental load factor ($\delta\lambda^i_j$) is obtained (see Step 5 of Algorithm 1), the main steps of the Load Control scheme are shown in Algorithm 2.

Algorithm 3. Application class implemented in the client of NLS++

```
1: Read model and algorithm input files
2: Create instance of Model class using data from model input file
3: Initialize instance of Model class
4: Create instance of LinearSystem class using data from algorithm input
   file
5: Create instance of Control class using data from algorithm input file
6: Call solver function of Control class
```

4.2 NLS++ Usage. The class hierarchy of NLS++ was designed in such way that the code would be executed by a simple application class, requiring minimal work from the end user. That application class of the NLS++ client requires two input files: a model file and an algorithm file. A finite element model file; for example, contains the element type, finite element mesh (i.e., nodes, elements, connectivity), boundary conditions, applied loads and displacements, and material properties. The algorithm file contains the type of nonlinear solution scheme, initial control

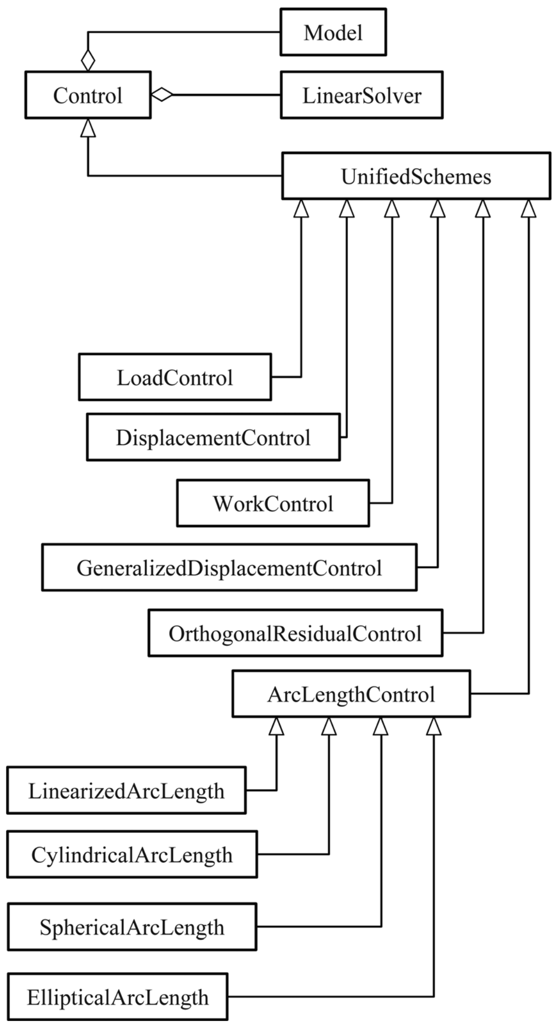


Fig. 15 NLS++ class organization

Table 1 Nonlinear solution scheme inputs

Algorithm	Initial control factor
LCM	Load increment, $\overline{\Delta\lambda}$
DCM	Displacement increment, $\overline{\Delta u}$
ALCM	Arc length increment, $\overline{\Delta S}$
WCM	Work increment, $\overline{\Delta W}$
GDCM	Load parameter, $\overline{\delta\lambda}$
ORP	Iterative load increment, $\overline{\delta\lambda}$

Table 2 Additional inputs for selected nonlinear solution schemes

Algorithm	Additional parameter 1	Additional parameter 2
DCM	Control degree of freedom	Constant or variable displacement
ALCM	Constant or variable arc length	n/a
ORP	Initial incremental scale factor, β	n/a

factor, additional algorithm inputs if necessary, maximum number of steps and iterations, convergence tolerance, and type of linear solver. Table 1 shows initial control factor required for each nonlinear solution scheme, and additional inputs required for a few of the nonlinear solution schemes are given in Table 2.

After input files are read in the application class, new instances of the above mentioned three classes (i.e., Model, LinearSystem, and Control) are created, the incremental-iterative procedure is called, and the analysis is executed. Algorithm 3 lists the steps of the application class.

5 Numerical Results

The performance of the unified solution schemes in capturing equilibrium paths of highly nonlinear problems is investigated. The example problems, input parameters, and resulting nonlinear equilibrium curves obtained with each scheme are given in the next subsection. A range of input parameters for each scheme (i.e., control factor, number of steps, scale factor) may result in converging solutions for a particular problem. However, in general, the input parameters reported in Sec. 5.1 are those that adequately captured the entire equilibrium curves as they are shown with the fewest number of steps. A qualitative evaluation of the solvers is given in Sec. 5.2, which includes a discussion of the sensitivity of the methods to the input parameters.

5.1 Applications and Examples. The ability of the solution schemes to capture nonlinear behavior is tested with several examples that feature a host of nonlinearities, including load and displacement limit points and large fluctuations in stiffness. The examples include two functions: one unidimensional and one two-dimensional, and three structural systems: the von Mises truss, twelve bar truss, and Lee frame.

Notice that the structural systems examined in this work are comprised of discrete elements (e.g., bars and beams); for an example, using continuum elements, the reader is referred to Lages et al. [66] where an earlier version of the NLS++ library was used to solve a compression test problem involving strain localization into a shear band. For additional examples of complex nonlinear systems, see the texts by Reddy [7] and Belytschko et al. [12], where in addition to structural systems, contact, fluid-flow, and coupled problems are considered. Structural and complex systems (involving smart structures) are analyzed and compared to experimental results in the text by Pai [117]. Doyle [118] places particular emphasis on stability of thin-walled structures, such as aircraft, ships, and containment vessels. Furthermore, numerical nonlinear analysis of composite materials, including laminated plates and shells, is examined in the text by Palazotto and Dennis [119].

Unless otherwise stated, all computation results were obtained with a maximum of 40 iterations per step, a convergence tolerance (see Fig. 14) of 10^{-4}, and a standard, rather than modified, update to the stiffness matrix. Additionally, the linearized version of the ALCM was employed in all examples.

5.1.1 Unidimensional Function. The first example to test the nonlinear solution schemes is a single-degree-of-freedom problem. The function was used by Chen and Blandford [98] to test their work increment control method. While the function is very simple to incorporate into NLS++, it features some complex nonlinearities including two load limit points and an infinite slope; thus making it a good candidate to evaluate the nonlinear solution schemes. The unidimensional function is given by

$$f(u) = -3\text{sign}(u)|u|^{\frac{1}{3}} + 4u + 1 \tag{77}$$

where the sign function is defined as

$$\text{sign}(x) = \begin{cases} -1 & \text{for} \quad x < 0 \\ 0 & \text{for} \quad x = 0 \\ 1 & \text{for} \quad x > 0 \end{cases} \tag{78}$$

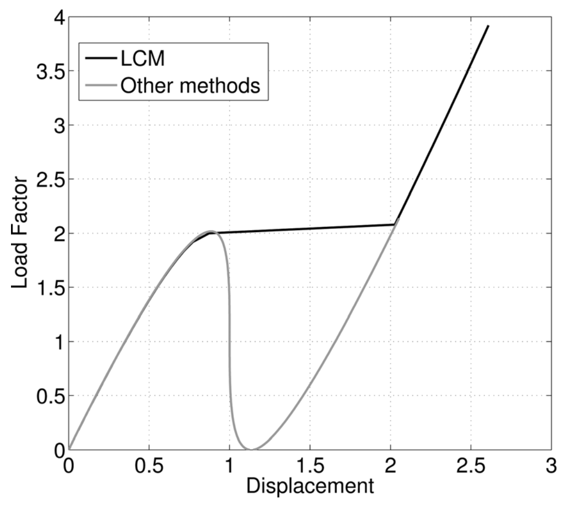


Fig. 16 Unidimensional function results

Table 3 Summary of the unidimensional function example

Method	Max. steps	Control factor	Scale factor	Behavior
LCM	50	0.08	n/a	Snaps through
DCM	100	0.02	n/a	Fully converged
ALCM*	325	0.02	n/a	Fully converged
WCM	95	0.001	n/a	Fully converged
GDCM	60	0.1	n/a	Fully converged
ORP	55	1.0	0.1	Fully converged
Convergence Tolerance = 0.1				

The internal load is simply given by Eq. (77). The derivative of Eq. (77) with respect to the degree of freedom, u, gives the tangent stiffness, i.e.,

$$\frac{df}{du} = -\frac{1}{|u|^{\frac{2}{3}}} + 4 \tag{79}$$

Each solution scheme was applied to the unidimensional function; the results are shown in Fig. 16. As expected, the LCM does not capture the full behavior at the load limit points. When the load factor reaches the first local maximum of 2, the method continues to increase load, therefore snapping through the softening behavior and only capturing stiffening behavior.

The remainder of the nonlinear solution schemes fully capture the behavior, which is expected because all of the methods are capable of capturing load limit points. It should be noted that methods that have difficulty near displacement limit points (i.e., displacement control method and work control method) were able to capture the full behavior of this example, because even though there is a vertical tangent at $u = 0$, there is no snap-back.

The parameters adopted in each algorithm and resulting behavior are given in Table 3. It should be noted that a relatively small control factor and large number of steps were required for the ALCM. There is very small change in the displacement corresponding to a very large change in the load near the vertical tangent. Other methods that increment the load and displacement independently (i.e., WCM, GDCM, ORP) can recover the curve between the maximum and minimum load with very few steps. However, the step size in the constant update arc length control method is defined by an arc of the same radius at every step, so the method cannot make large jumps in either load or displacement in one step. The tolerance was also adjusted to 10^{-1} to allow for the largest possible step size while still recovering the correct solution path.

5.1.2 Two-Dimensional Function. The next example is a function of two variables whose nonlinearity includes load and displacement limit points in both degrees of freedom. The problem is given by a vector of external and internal forces [120], shown below

$$\mathbf{p} = \begin{pmatrix} 40 \\ 15 \end{pmatrix} \tag{80}$$

$$\mathbf{q}(\mathbf{u}) = \begin{pmatrix} 10u_1 + 0.4u_2^3 - 5u_2^2 \\ 0.4u_1^3 - 3u_1^2 + 10u_2 \end{pmatrix} \tag{81}$$

The derivative of the internal force vector in Eq. (81) with respect to the degrees of freedom, u_1 and u_2, gives the components of the tangent stiffness matrix

$$K_{ij} = \frac{\partial q_i}{\partial u_j} \tag{82}$$

$$\mathbf{K}(\mathbf{u}) = \begin{bmatrix} 10 & 1.2u_2^2 - 10u_2 \\ 1.2u_1^2 - 6u_1 & 10 \end{bmatrix} \tag{83}$$

The equilibrium curves traced by each of the nonlinear solution schemes are shown in Fig. 17. The labels indicate the point where the method failed to accurately capture the equilibrium path. As expected, the LCM fails at the first load limit point, the DCM with either degree of freedom as the control fails at the first displacement limit point in the corresponding degree of freedom, and the variable DCM, ALCM, and GDCM capture the full equilibrium path. The WCM and ORP have difficulties in the first degree of freedom when the behavior of the system changes rapidly via a

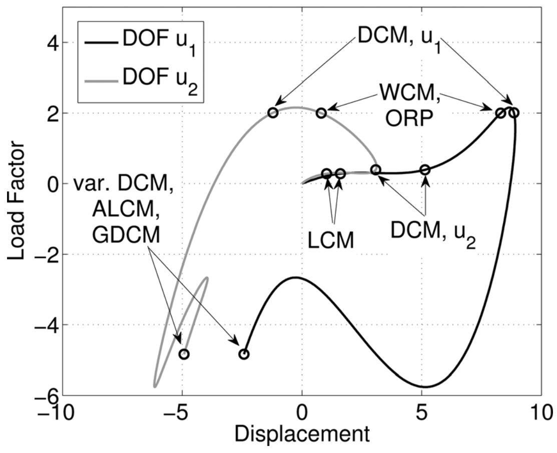


Fig. 17 Solution to the two-dimensional function example; labels indicate where the solution schemes failed. LCM fails at the first load limit point, the DCM fails at the displacement limit point in the degree of freedom corresponding to the control, the WCM and ORP fail near the first displacement limit point, and the variable DCM, ALCM, and GDCM capture the full equilibrium path.

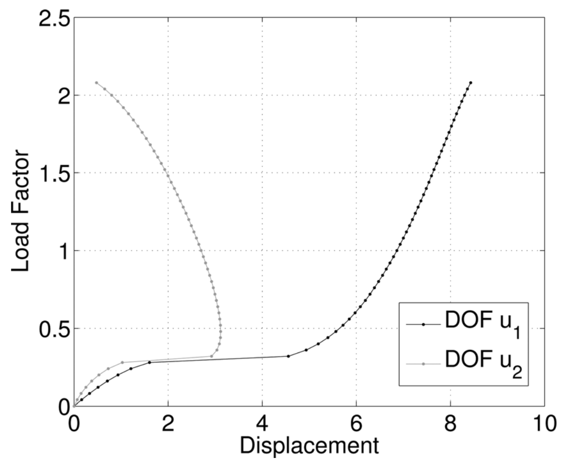


Fig. 18 Solution to the two-dimensional function example using the LCM. The method snaps through the first load limit point and fails at the second load limit points.

Table 4 Summary of the variable DCM for the two-dimensional function

Step	Control displacement	Snap-back
1–73	u_1	Step 58 in u_2 (Label 1 in Figure 18)
74–216	u_2	Step 158 in u_1 (Label 2 in Figure 4)
217–445	u_1	Step 266 in u_2 (Label 3 in Figure 4)
		Step 398 in u_2 (Label 4 in Figure 4)

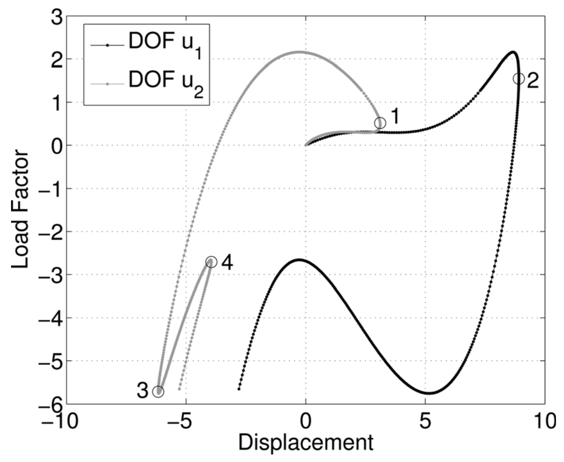


Fig. 19 Solution to the two-dimensional function example using the variable DCM. The labels indicate the locations where the control degree of freedom changes automatically.

load limit point followed immediately by a displacement limit point; accordingly, both methods fail in this area.

In this example, more explanation will be given to the behavior of the LCM and various versions of the DCM. First, the failure point of the LCM is shown at the first load limit point in Fig. 17 because the equilibrium path is not *accurately* captured beyond this point. The method actually snapped through at the first limit points on each curve and diverged at the next load limit point encountered, as shown in Fig. 18. The snap through behavior is expected because the load continues to increase after the initial load limit points; hence the solution scheme was able to converge to the next points. Next, a displacement limit point occurs in the second degree of freedom, however since the load is still increasing, the load control method captured this behavior. Divergence occurred at the next load limit point in the second degree of freedom. Snap through behavior could not have occurred because the load only decreases after this point.

Table 5 Summary of the two-dimensional function example

Method	Max. steps	Control factor	Scale factor	Behavior
LCM	100	0.04	n/a	Failed at load limit point
DCM^{s,1}	200	0.05	n/a	Failed at displacement limit point
DCM^{s,2}	200	0.02	n/a	Failed at displacement limit point
DCM^{v,1}	445	0.1	n/a	Fully converged
DCM^{v,2}	190	0.1	n/a	Fully converged
ALCM	700	0.05	n/a	Fully converged
WCM	300	0.01	n/a	Failed at displacement limit point
GDCM	650	0.01	n/a	Fully converged
ORP	100	0.1	1.0	Failed at displacement limit point

Note: ^{s,v} Standard, Variable; ^{1,2} Fixed control coordinate: $\mathbf{u}_1, \mathbf{u}_2$.

Each degree of freedom was selected as the control when the DCM was employed, and the labels in Fig. 17 indicate the resulting failure points. When the first degree of freedom, u_1, is the control, the equilibrium path is traced until the displacement limit point in the first degree of freedom is reached. Notice that the displacement limit point in the second degree of freedom is captured; this is expected because the displacement is only incremented for the first degree of freedom. When the second degree of freedom, u_2, is the control however, the solution scheme diverges much earlier at first the displacement limit point in the second degree of freedom.

The variable DCM captured the entire equilibrium path, shown in Fig. 17, because the control degree of freedom changes automatically when a displacement limit point is approaching in that degree of freedom. The control displacements and snap-back locations are listed in Table 4 and correspond to the labels in Fig. 19. From steps 1–73 the control displacement is u_1, and a displacement limit points is encountered at step 58 in degree of freedom u_2 (Label 1 in Fig. 19). At step 74, the control displacement changes to u_2, and snap-back is captured in degree of freedom u_1 at step 158 (Label 2 in Fig. 19). The final change is control degree of freedom occurs in step 217, after which point two snap-back points are passed in degree of freedom u_2 (Labels 3 and 4 in Fig. 19). These automatic changes in the control degree of freedom allow the method to recover the entire solution path. Unless otherwise noted, the variable DCM will be used in the remainder of the numerical examples.

Table 5 shows the parameters used and behavior captured for this two degree of freedom example. Notice in the table that the number of steps used for the arc length control method and the generalized displacement control method are similar, but the control factor for the arc length control method is five times greater. However, recall from Table 1 that the control factors represent different quantities for each scheme and cannot be directly compared. In the generalized displacement control method the control factor is only used once at the first iteration of the first step, and the load factor is adjusted by the algorithm for all subsequent steps and iterations. The control factor in the arc length control method, however, is used once at every step.

5.1.3 von Mises Truss. The von Mises truss is a two-degree of freedom system consisting of two prismatic bar (truss) elements loaded indirectly through a spring of stiffness C, as shown in Fig. 20. It has been studied by several authors, including Bergan [121], Bazant and Cedolin [122], and Yang and Sheih [4], among others. Large displacements and rotations (geometric nonlinearity) are accounted for through the Total Lagrangian (TL) formulation, in which static and kinematic variables refer to the original undeformed configuration, with rotated engineering strain measure

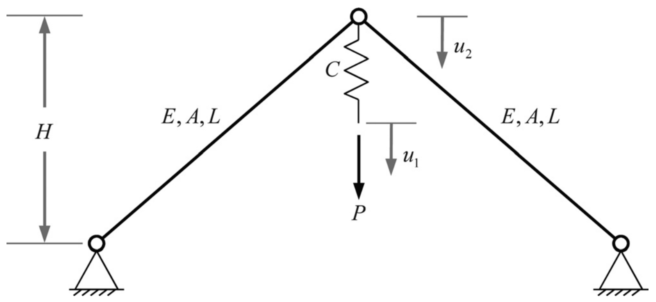


Fig. 20 von Mises truss schematic

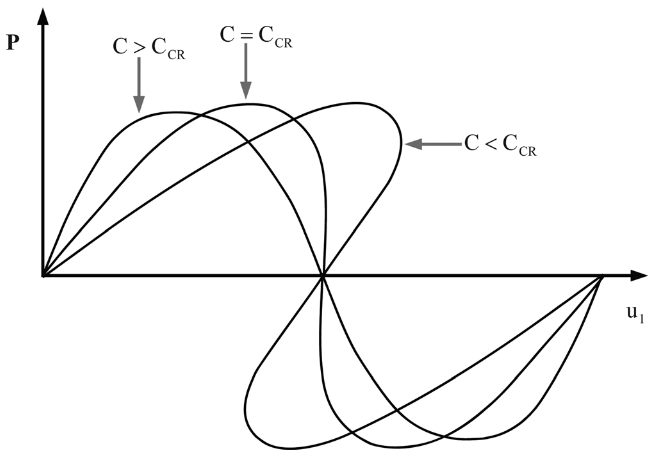


Fig. 21 Equilibrium paths for the von Mises truss with varying spring stiffness, *C*

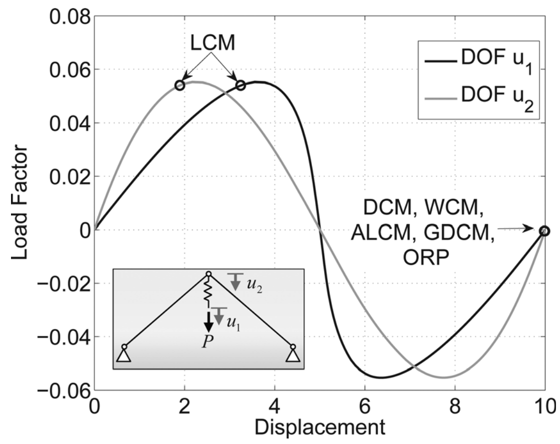


Fig. 22 Solution to the von Mises truss without snap-back; labels indicate where the solution schemes failed. LCM failed at the first load limit point, WCM failed at the first displacement limit point, and the variable DCM, ALCM, GDCM, and ORP captured the full equilibrium path.

[14] given by $\varepsilon = (L_{\text{deformed}} - L)/L$. Two constitutive relations are considered in the problem. The first is the linear Hooke's law; thus material nonlinearity is not present; in the second, material nonlinearity is considered by adopting an elastic/perfect plasticity model (see Appendix). Although very simple, this example can be used to demonstrate the ability of the nonlinear solution algorithms to capture both load and displacement limit points, as well as sudden changes of direction in the equilibrium paths. In both cases the following values are used (consistent units are assumed):

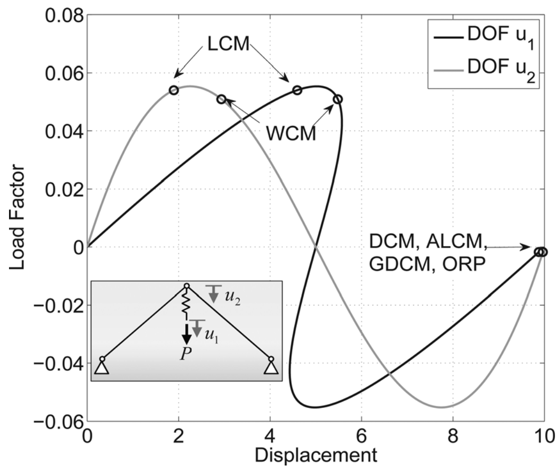


Fig. 23 Solution to the von Mises truss with snap-back; labels indicate where the solution schemes failed. LCM failed at the first load limit point, WCM failed at the first displacement limit point, and the variable DCM, ALCM, GDCM, and ORP captured the full equilibrium path.

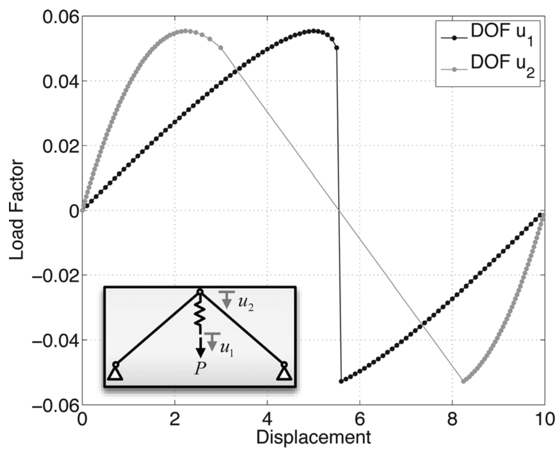


Fig. 24 Solution to the von Mises truss example with snap-back using the DCM with u_1 as the control displacement. The method snaps through at the displacement limit point and does not capture the full behavior.

Table 6 Summary of the von Mises truss example

Method	C	Max. steps	Control factor	Scale factor	Behavior
LCM	0.02	100	0.001	n/a	Fails at load limit point
DCM^{s,1}	0.02	100	0.1	n/a	Snaps through displacement limit point
DCM^{v,1}	0.02	100	0.3	n/a	Fully converged
ALCM	0.02	100	0.17	n/a	Fully converged
WCM	0.02	100	0.002	n/a	Fails at displacement limit point
WCM	0.04	50	0.001	n/a	Fully converged
GDCM	0.02	100	0.0025	n/a	Fully converged
ORP	0.02	190	0.0025	0.5	Fully converged
ORP	0.04	65	0.005	1.0	Fully converged

Note: ^{s,v} Standard, Variable; ^{1,2} Fixed control coordinate: u_1, u_2.

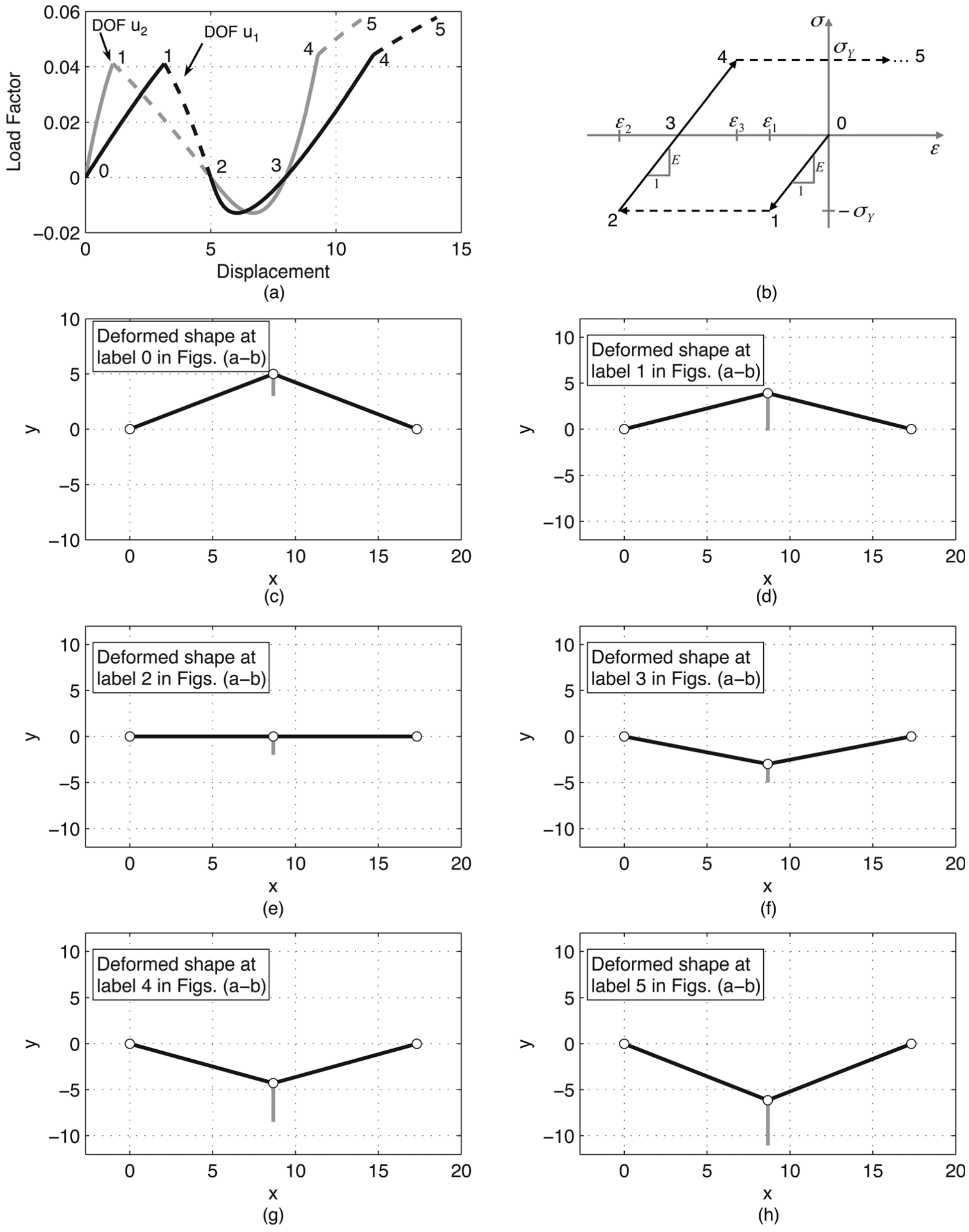


Fig. 25 Deformed shape of the von Mises truss with material and geometric nonlinearity

$H = 5.0$, $L = 10.0$, external load $= \{1.0, 0.0\}^T$. The reader is directed to Ref. [123] for more information on the derivation of the finite element equations for this and the remaining examples.

von Mises Truss: Elastic Case: The elastic material properties of the von Mises truss are $EA = 1.0$, where consistent units are assumed. The behavior of this structure depends strongly on the stiffness of the spring, C, (see Fig. 21), as snap-back behavior will only occur if it is below the critical value, $C_{cr} = 0.030940$ in this case (see Appendix for derivation). The critical spring stiffness can be obtained through a straight forward calculation using the stationary potential energy of the system. C was chosen to be 0.02 to obtain snap-back behavior, and 0.04 for no snap-back in the following numerical examples.

Figures 22 and 23 show the nonlinear behavior captured by each of the solution schemes for the system without and with snap-back, respectively. In both cases, the LCM fails at the load limit point, while the variable DCM, ALCM, GDCM, and ORP capture the full path. The WCM traces the entire equilibrium path, but fails when at the displacement limit point when snap-back is present. This behavior is expected from the WCM because the snap-back occurs in the major forcing direction. The external load vector contains only one load, and since snap-back occurs in the loading direction, the displacement increment in that direction is zero. The denominator of the load parameter in Eq. (29) becomes unbounded and the method diverges.

Analogous to the snap-through behavior with respect to increasing load seen with the LCM in unidimensional and two-dimensional examples, the standard DCM exhibits snap-through behavior with respect to increasing displacement. When snap-back is present, the standard DCM can only capture the full equilibrium path when degree of freedom u_2 is the control because there is no snap-back in this degree of freedom. When u_1 is chosen as the control displacement, the method snaps through at the displacement limit point and does capture the snap-back behavior, as illustrated in Fig. 24.

Although the ORP captures the full equilibrium paths for both the snap-back and no snap-back scenarios, the parameters are quite different between the two cases. The method proves to be

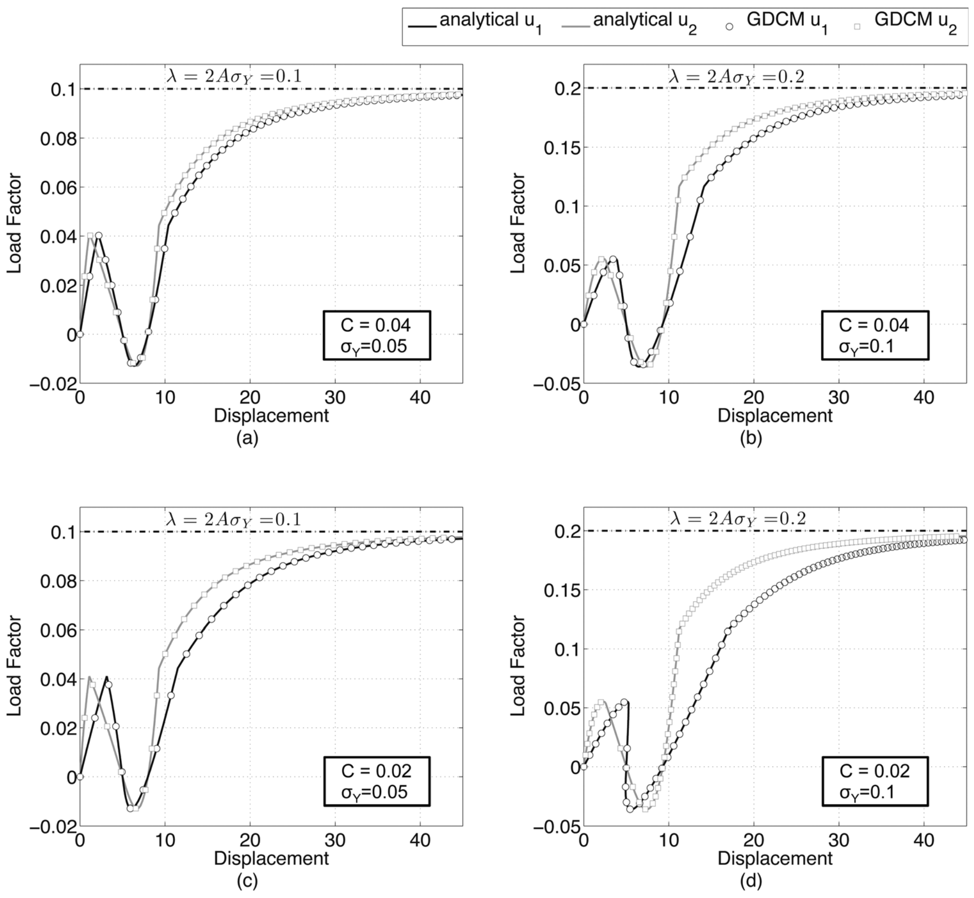


Fig. 26 Analytical solution and solution obtained with the GDCM for the elasto-plastic case of the von Mises truss: (*a*) C = 0.04 and $\sigma_Y = 0.05$; (*b*) C = 0.04 and $\sigma_Y = 0.1$; (*c*) C = 0.02 and $\sigma_Y = 0.05$; (*d*) C = 0.02 and $\sigma_Y = 0.1$

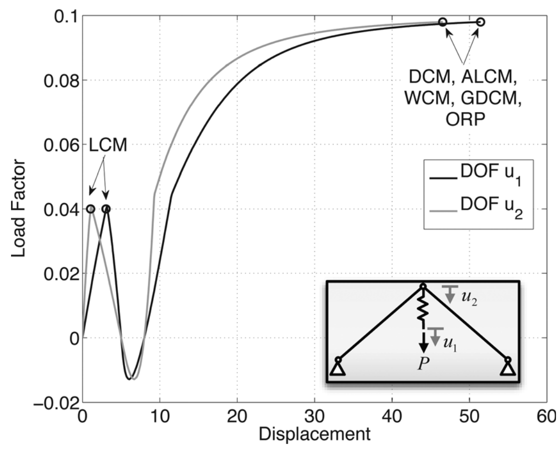


Fig. 27 Solution to the elasto-plastic case of the von Mises truss; labels indicate with the solution schemes failed. $\sigma_Y = 0.05$, C = 0.02.

Table 7 Summary of the von Mises truss: Elasto-plastic case with $\sigma_Y = 0.05$ and $C = 0.02$

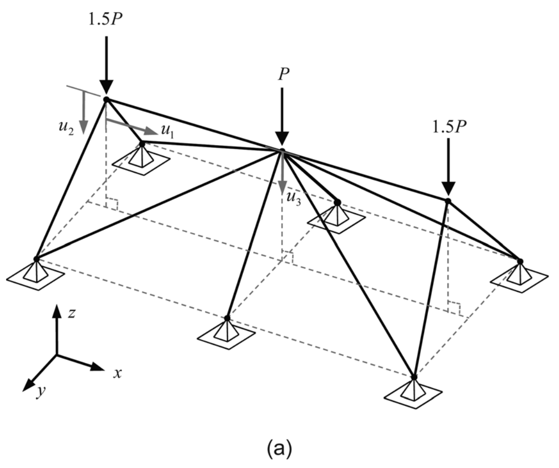
Method	Max. steps	Control factor	Scale factor	Behavior
LCM	100	0.001	n/a	Fails at load limit point
DCM^{v,1}	150	0.3	n/a	Fully converged
ALCM	350	0.2	n/a	Fully converged
WCM	80	0.001	n/a	Fully converged
GDCM	180	0.005	n/a	Fully converged
ORP	775	0.0025	0.5	Fully converged

Note: ^{v} Variable; ^{1} Fixed control coordinate: $\mathbf{u}_1$.

sensitive to input parameters and to the system configuration because a smaller control factor, smaller scale factor, and large number of steps are required in the system with snap-back behavior than the system without. Table 6 provides a summary of all parameters and the resulting behavior of each algorithm.

von Mises Truss: Elasto-Plastic Case: Material nonlinearity is considered in the von Mises truss by adopting an elasto-plastic constitutive model [124], with elastic modulus, E, and initial flow stress, σ_Y. We consider the case of perfect plasticity, in which the flow stress is varied to explore different structural responses.

The solution has four stages, which are derived analytically in the Appendix and are shown in Fig. 25 for the case of C = 0.02 and $\sigma_Y = 0.05$. The first stage is equal to the elastic solution; it starts at the undeformed configuration (Fig. 25(*c*)) and stops when the flow stress is reached in compression (Fig. 25(*d*)). The load versus displacement and stress versus strain trajectories for stage one are shown as the solid lines from labels "0" to "1" in Figs. 25(*a*) and 25(*b*), respectively. The second stage is a plateau (plastic) and it continues from the last point in stage one until $u_2 = H$ (Figs. 20 and 25(*e*)). The load versus displacement and stress



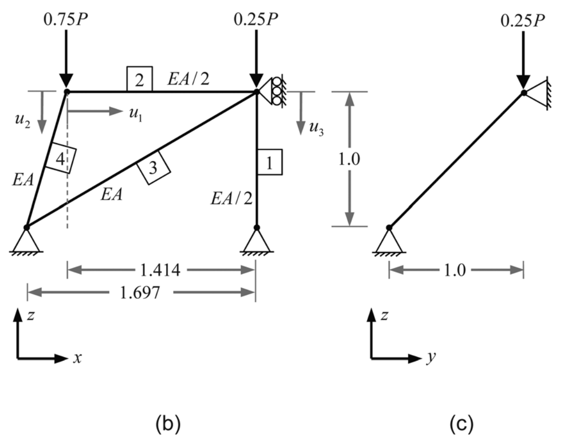
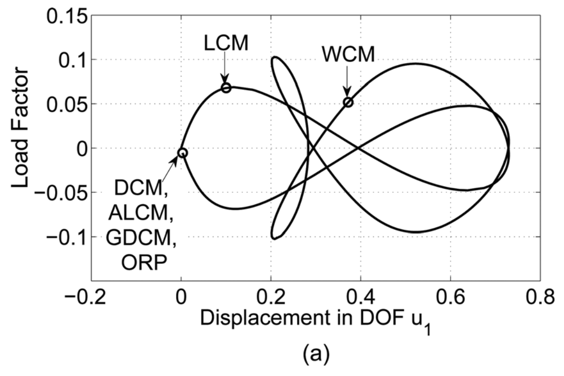
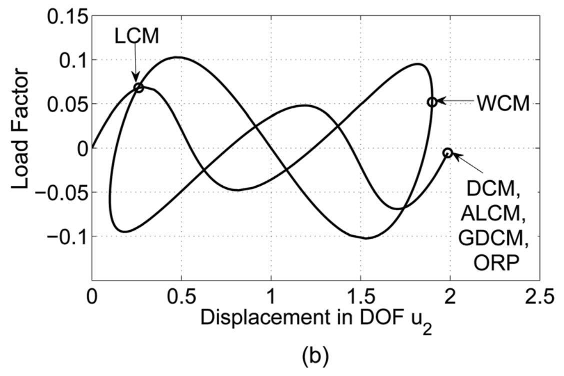


Fig. 28 Twelve-bar truss schematic (*a*) 3D view, (*b*) x-z view, (*c*) y-z view





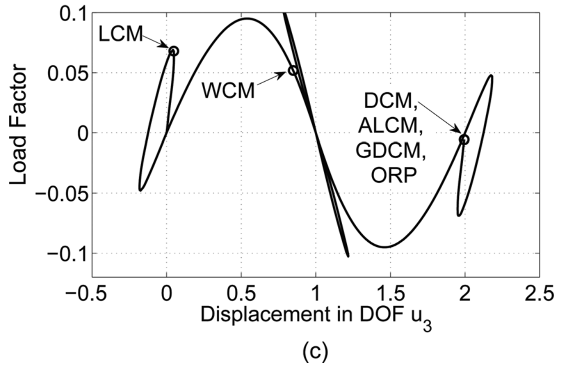
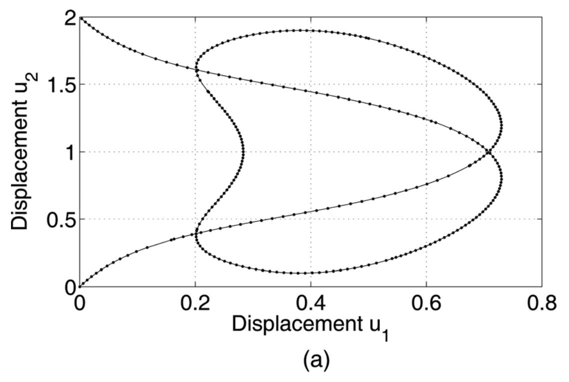


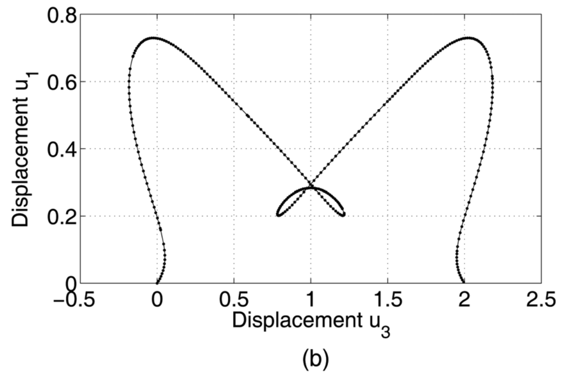
Fig. 29 Twelve bar truss results: Markers indicate where the solution schemes failed. LCM failed at the first load limit point, WCM failed at the first displacement limit point, and the variable DCM, ALCM, GDCM, and ORP captured the full equilibrium path. (*a*) DOF u_1, (*b*) DOF u_2, (*c*) DOF u_3.

versus strain trajectories for stage two are shown as the dashed lines from labels "1" to "2" in Figs. 25(*a*) and 25(*b*), respectively. The third stage is elastic and it continues from the last point in stage two until the flow stress is reached again, this time in tension (Fig. 25(*g*)). The load versus displacement and stress versus strain trajectories for stage three are shown as the solid lines from labels "2" to "4" in Figs. 25(*a*) and 25(*b*), respectively. Notice that the deformed configuration shown in Fig. 25(*f*) occurs during stage three. The fourth stage is again a plateau and it continues from the last point in stage three. The deformed configuration shown in Fig. 25(*h*) corresponds to the load and displacements at label "5" in Fig. 25(*a*). In this final stage, the equilibrium trajectories (u_1 versus P and u_2 versus P) are limited by the flow stress; hence there is a horizontal asymptote at $P = 2A\sigma_Y$ (see Fig. 26). In order to ensure that the first plasticity stage occurs, the initial flow stress must be less than the stress that that occurs at the beginning of stage three (i.e., when $u_2 = H$), otherwise the behavior will be entirely elastic until the flow stress is reached in tension. Additionally, depending on the value of the flow stress, the first plastic stage will either occur before or after the first load limit point of the elastic stage.

For the numerical study, the material properties are varied to illustrate different behavior of the von Mises truss with the elasto-plastic constitutive relation. The spring stiffness is either 0.04 or 0.02 to achieve no snap-back and snap-back in the elastic solution, respectively. The flow stress is either 0.05 or 0.1 so that the first plastic stage is reached before or after the load limit point of the elastic solution, respectively. The analytical solution and the solution obtained with the GDCM for C = 0.04 and $\sigma_Y = 0.05$ are shown in Fig. 26(*a*), for C = 0.04 and $\sigma_Y = 0.1$ in Fig. 26(*b*), for C = 0.02 and $\sigma_Y = 0.05$ in Fig. 26(*c*), and for C = 0.02 and $\sigma_Y = 0.1$ in Fig. 26(*d*). It should be noted that every third converged point obtained with the GDCM is plotted in Fig. 26. This figure demonstrates good agreement between the analytical solution and the solutions obtained with the unified schemes, where the GDCM is used as a representative scheme. The other methods were also tested for all cases, and good agreement with the analytical solution was achieved. Results for all methods are shown for the case where C = 0.02 and $\sigma_Y = 0.05$ in Fig. 27. All methods except the LCM captured the entire equilibrium path; as expected, the LCM failed at the first load limit point. A summary of the input parameters used to obtain the results in Fig. 27 is shown in Table 7.

5.1.4 Twelve-Bar Truss. This example consists of a 12-bar truss structure as illustrated in Fig. 28(*a*). It has been studied by





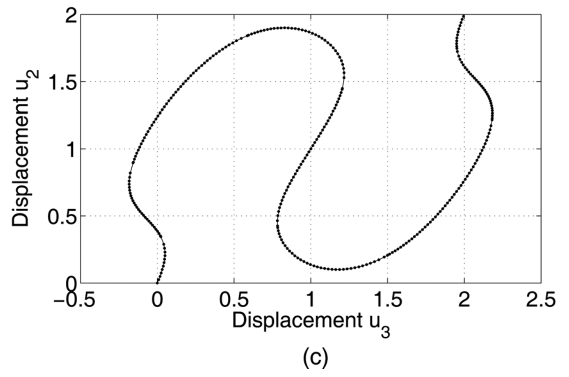


Fig. 30 Displacement-displacement curves for the twelve bar truss (*a*) u_2 versus u_1, (*b*) u_1 versus u_3, (*c*) u_2 versus u_3

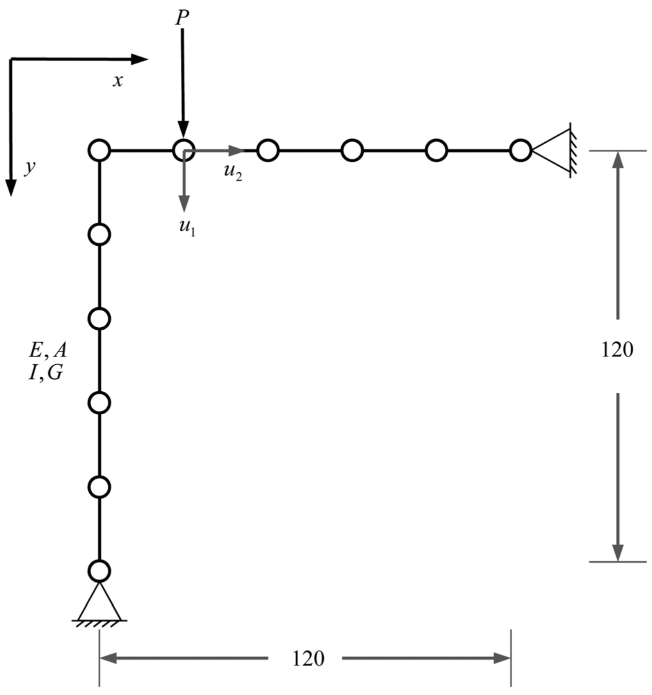


Fig. 31 Lee frame schematic

Table 8 Summary of the twelve bar truss example

Method	Max. steps	Control factor	Scale factor	Behavior
LCM	100	0.001	n/a	Failed at load limit point
DCM^{v,1}	465	0.01	n/a	Fully converged
ALCM	165	0.05	n/a	Fully converged
WCM	100	0.0002	n/a	Failed at displacement limit point
GDCM	115	0.025	n/a	Fully converged
ORP	650	0.0025	2	Fully converged

Note: ^{v} Variable, ^{1} Fixed control coordinate: $\mathbf{u}_1$.

Yang and Leu [125], Yang et al. [126], and Krenk and Hededal [109], who described the deformation behavior of the structure in detail, among others. This example features highly nonlinear behavior: the load changes direction eight times and the structure experiences several very large changes in stiffness through the load history.

The material properties of the twelve bar truss are assigned as $EA = 1.0$, and the external load vector is given by the reference load vector, $\{0.0, 0.75, 0.25\}^T$. Like the von Mises truss example, the nonlinear prismatic bar elements, the TL formulation, and the rotated engineering strain are used to include effects of geometric nonlinearity. The constitutive relationship is linear; thus material nonlinearity is not considered here. For the present work, double symmetry has been considered and; therefore, the deformations of the structure can be described by three displacement components (u_1, u_2, and u_3) as shown in Figs. 28(*b*) and 28(*c*).

The computational results are shown in Fig. 29. The LCM failed at the first load limit point, the WCM failed near snap-back points in the loading directions, and the ALCM, GDCM and ORP captured the full behavior. From the displacement-displacement curves shown in Fig. 30, it is clear that no linear combination of the displacement components increases monotonically; therefore, a traditional displacement control method could not capture the full behavior [109]. The variable DCM, however, can and does capture the full equilibrium path, as indicated in Fig. 29. A summary of the parameters and resulting behavior for each method is shown in Table 8.

5.1.5 Lee Frame. The Lee frame is a well-known example for evaluating nonlinear solvers, for which an analytical solution exists [127]. Schweizerhof and Wriggers [128] compared updated and spherical plane path following schemes using the Lee frame discretized with beam elements. Parente and Vaz [129] used this example discretized with quadratic isoparametric 8-node elements for shape design sensitivity analysis for nonlinear structures.

Deformation of the structure is characterized by large rigid body displacements and rotations resulting in instability. The behavior is highly nonlinear with two load limit points and snap-back behavior (i.e., displacement limit point). The Lee frame, shown in Fig. 31, was discretized with 10 beam elements, each with the following properties (consistent units are assumed): $EA = 4320$, $GJ = 2160$, $EI = 1440$. The deformed shape at different points during the incremental-iterative is illustrated in Fig. 32. The TL formulation is used to describe the nonlinear

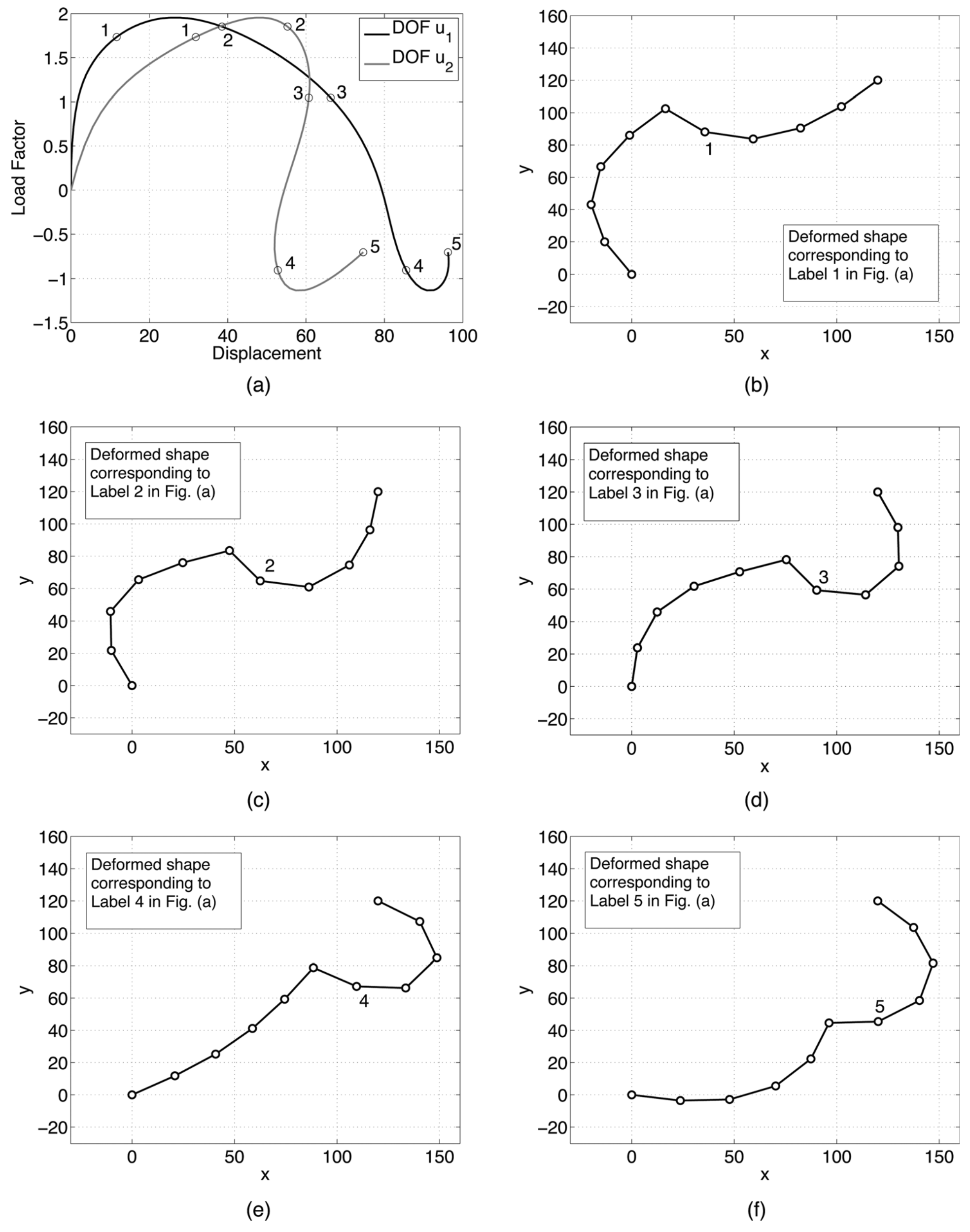


Fig. 32 Deformed shape of the Lee frame corresponding to (*b*) label 1, (*c*) label 2, (*d*) label 3, (*e*) label 4, (*f*) label 5 in (*a*) load factor versus displacement curve for DOFs u_1 and u_2 (see Fig. 31)

beam elements, which are capable of resisting longitudinal and transverse loads applied between supports as well as large rotations. Only geometric nonlinearity is considered, as a linear constitutive relationship is assumed.

The equilibrium paths for the Lee frame example are shown in Fig. 33, and the results are also summarized in Table 9. The nonlinear solution schemes generally behave as expected and as they did in the previous examples. The LCM only captured behavior up to the first load limit point. The WCM failed at the snap-back point in degree of freedom u_2, which is also the loading direction. The variable DCM, ALCM, GDCM, and ORP captured the full behavior. A summary of the input parameters used in this example is given in Table 9.

5.2 Evaluation of Solution Schemes. The robustness and sensitivity to input parameters of the methods is investigated in the section. The parameters for all of the schemes were varied then applied to the problems examined in the previous subsection. The resulting behavior is summarized and the schemes are qualitatively evaluated here.

First, it is well known that the LCM cannot capture the equilibrium path beyond load limit points. The method is inherently sensitive to the control parameter, which is the load increment. When the load parameter exceeds the first local external load maximum the method will either diverge or snap through the unstable behavior.

The variable DCM successfully captured the equilibrium solution with a control factor of 0.001, 0.01, or 0.1 for all examples in the previous section. However, this method will not be successful for problems with a single degree of freedom experiencing snap-back, and will have difficulty for problems where snap-back occurs in multiple degrees of freedom simultaneously.

The ALCM generally captures the entire equilibrium curve for the problems studied here when the control factor was selected between 0.001 and 1.0, and the value of the control factor may

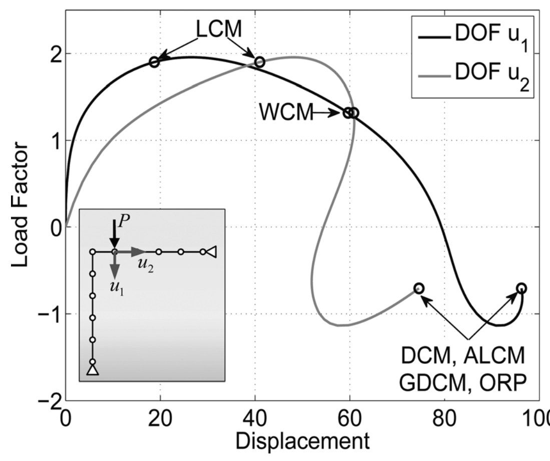


Fig. 33 Lee frame results: Markers indicate where the solution schemes failed. LCM failed at the first load limit point, WCM failed at the first displacement limit point, and the variable DCM, ALCM, GDCM, and ORP captured the full equilibrium path.

Table 9 Summary of Lee frame example

Method	Max. steps	Control factor	Scale factor	Behavior
LCM	100	0.05	n/a	Failed at load limit point
DCM^{v,1}	100	0.1	n/a	Fully converged
ALCM	80	5	n/a	Fully converged
WCM	300	0.004	n/a	Failed at first displacement limit point
GDCM	80	0.3	n/a	Fully converged
ORP	51,000	0.005	0.1	Fully converged

Note: ^{v} Variable, ^{1} Fixed control coordinate: $\mathbf{u}_1$.

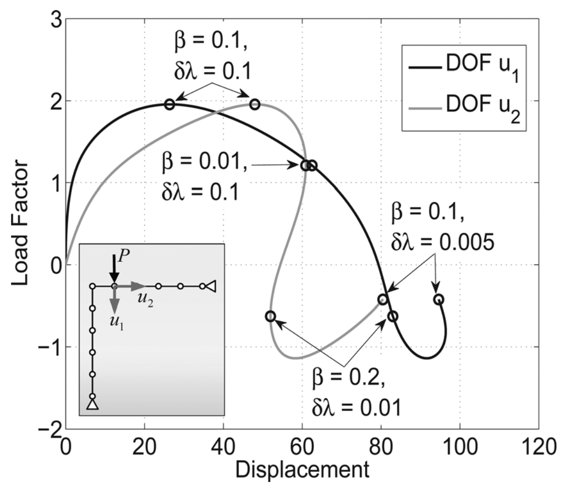


Fig. 34 Lee frame results using the ORP with different values for the control factor, *δλ*, and scale factor, *β*, which are indicated by the marker labels

even increase for problems with relatively high maximum displacements. For example, the maximum displacement in the 12-bar truss problem is about 2.2 and the ALCM captured the entire solution path with a control factor between 0.001 and 0.1. However, in the Lee frame problem the maximum displacement is around 95, and the control factor for the ALCM could be as high as 5.

Table 10 Behavior of the ORP applied to the Lee frame example

Control factor	Scale factor	Behavior
0.1	0.1	Failed at first load limit point (750 steps)
0.1	0.01	Failed at first displacement limit point (13,676 steps)
0.01	0.2	Failed at second displacement limit point (9,235 steps)
0.005	0.1	Fully converged (51,000 steps)

The WCM is able to capture the entire solution path for problems that do not feature snap-back behavior; however, a fairly small control factor is required. The method consistently diverges at displacement limit points, despite modifications to the input parameter.

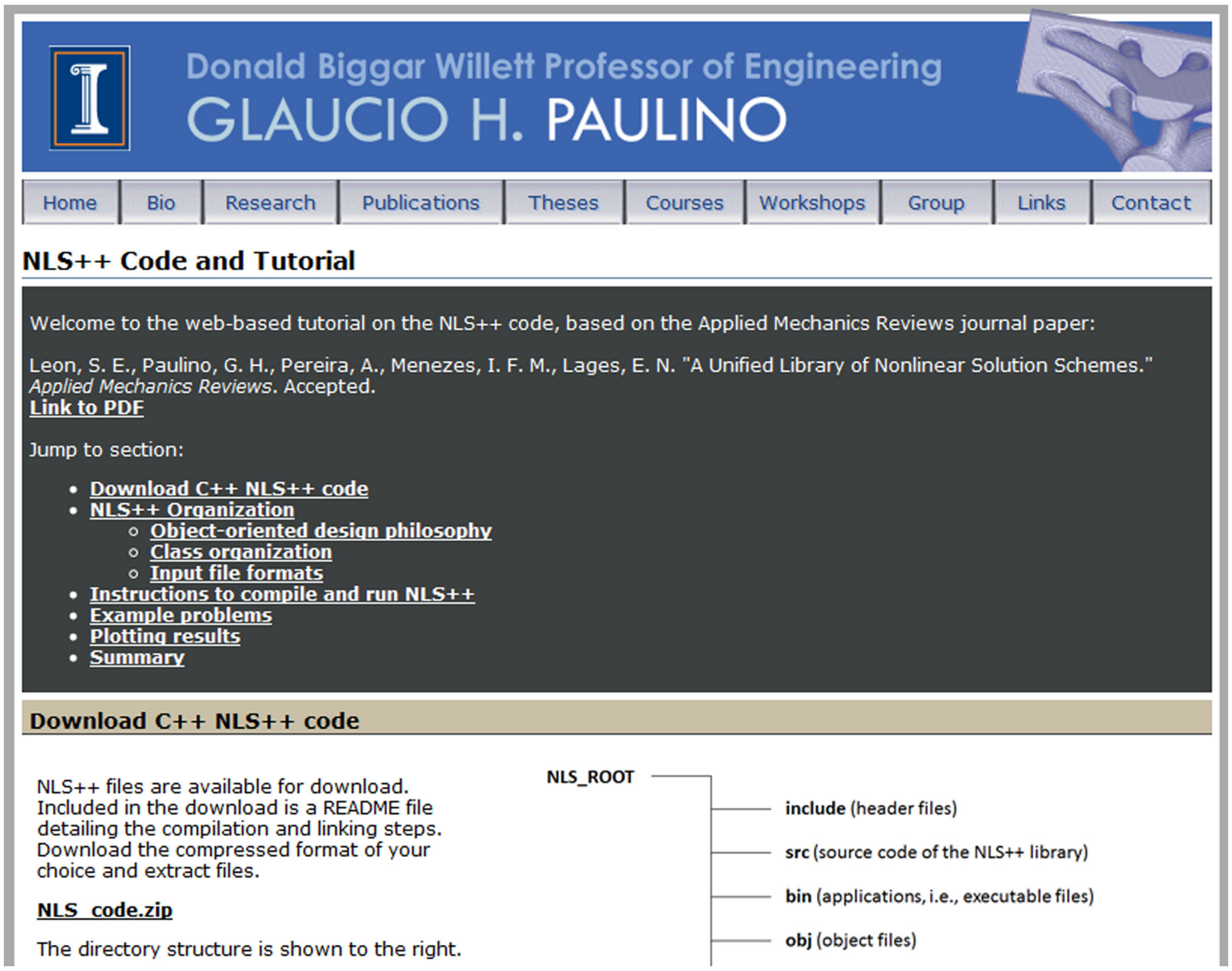
The GDCM generally captures the entire equilibrium curve for the problems studied here when the control factor was selected between 0.001 and 0.01. It should be noted that the GDCM takes a fewer number of steps for a given control factor compared to other schemes. For example, the GDCM required 234 steps to capture the equilibrium curve of the von Mises truss example with a control factor of 0.001. The variable DCM and the ALCM required 30,000 and 35,000 steps, respectively, for the same value of the control factor. When the control factor for the von Mises truss problem was chosen as 0.1 the variable DCM and the ALCM required 285 and 350 steps, respectively, but the GDCM could not accurately capture the solution path because too few steps were taken. Therefore, a smaller control is generally preferable for this method, but this does not imply that a large number of steps is required.

The ORP is the most sensitive of the unified schemes to the input parameters for the problems investigated in this work. Scale factors values ranging between 0.01 and 1.0 and control factors ranging between 0.001 and 1.0 were tested for all problems in the previous section. Generally, a smaller control factor and scale factor yields converged results; however, the number of steps required increases proportionally as the factors decrease. Additionally this trend was not uniform across all examples run in the previous section. To illustrate the sensitivity of the ORP to the input parameters, the Lee frame example is revisited. Figure 34 shows the different results obtained for the ORP depending on the selection of input parameters, given in Table 10. The final converged solution was obtained with a relatively small control factor and larger scale factor, which is not a trend that is observed for different problems. Additionally, 51,000 steps were required to capture the equilibrium path, which is 600 times more steps than was required with the ALCM, for example.

5.3 Distribution of NLS++ Source Code. The entire NLS++ library and a tutorial are available for download[3]. A screen shot of the website is shown in Fig. 35.

For Microsoft Visual Studio users, the code and Visual Studio project (*.prj) and solution (*.sln) files are available for download. In addition, the stand-alone source and library files are also available. The tutorial includes a detailed explanation of the code organization, which is meant to be supplementary to the information provide in Sec. 4. Next, step-by-step instructions for compiling and running the code using Visual Studio are provided. The input files used to generate the results in the previous section are distributed, along with a guide for the user to create his/her own set of input files. Finally, simple MATLAB codes, which plot the load versus displacement curves, are available.

[3]See www.ghpaulino.com/NLS_tutorial.html for download information.


Donald Biggar Willett Professor of Engineering
GLAUCIO H. PAULINO

NLS++ Code and Tutorial

Welcome to the web-based tutorial on the NLS++ code, based on the Applied Mechanics Reviews journal paper:

Leon, S. E., Paulino, G. H., Pereira, A., Menezes, I. F. M., Lages, E. N. "A Unified Library of Nonlinear Solution Schemes." *Applied Mechanics Reviews*. Accepted.
Link to PDF

Jump to section:

- Download C++ NLS++ code
- NLS++ Organization
  - Object-oriented design philosophy
  - Class organization
  - Input file formats
- Instructions to compile and run NLS++
- Example problems
- Plotting results
- Summary

Download C++ NLS++ code

NLS++ files are available for download. Included in the download is a README file detailing the compilation and linking steps. Download the compressed format of your choice and extract files.

NLS_code.zip

The directory structure is shown to the right.

NLS_ROOT
include (header files)
src (source code of the NLS++ library)
bin (applications, i.e., executable files)
obj (object files)

Fig. 35 Screen shot (shot of www.ghpaulino.com/NLS_tutorial.html)

6 Conclusions

In this paper, we review the formulation and application of nonlinear solution schemes with an emphasis on those schemes unified into the object-oriented library called NLS++. A variety of schemes are cast into the $(N+1)$ dimensional space then tested and evaluated with functions and structural systems exhibiting strong nonlinearities. The entire NLS++ library and a tutorial are available for download.[4] A brief review of other nonlinear solution techniques is also presented, however the references in this area are not exhaustive.

The full potential of NLS++ to capture highly nonlinear behavior can be achieved by integrating it into a general finite element analysis package. The software, TopFEM, is an object oriented finite element analysis code that utilizes TopS [130], a topological data structure for representing finite element meshes. At present it can represent large scale problems and can be used for complex analysis including topology optimization and dynamic fracture simulation. Integration of NLS++ into TopFEM would therefore greatly improve the capabilities of both object-oriented analysis codes.

Another area of further investigation lies in implementing an efficient linear solver, which is utilized in the incremental-iterative procedure. Iterative solvers, which use successive approximations beginning from an initial estimate, may be the most efficient and robust in the context of the NLS++. The conjugate gradient method for example, is very efficient for large problems; however, it is only applicable to symmetric, positive-definite systems [131]. Other optimization based solvers applicable to nonsymmetric matrices, such as the biconjugate gradient method or the generalized minimum residual method [132], may be explored for NLS++ in order to represent a larger class of problems [116,133].

[4]See www.ghpaulino.com/NLS_tutorial.html.

Acknowledgment

Sofie E. Leon gratefully acknowledges the support from the National Science Foundation Graduate Research Fellowship Program (GRFP). Anderson Pereira and Ivan F. M. Menezes acknowledge the financial support by Tecgraf (Group of Technology in Computer Graphics), PUC-Rio, Rio de Janeiro, Brazil.

Appendix

In this Appendix, we present key equations in the derivation of the analytical solution for the von Mises truss explored in Sec. 5.1.3.

A.1 Equilibrium Equations of the von Mises Truss. We use the principle of virtual work to obtain the analytical solution of the von Mises truss

$$\delta W_{\text{int}} = \delta W_{\text{ext}} \tag{A1}$$

$$2AL\sigma\delta\varepsilon + C(u_1 - u_2)(\delta u_1 - \delta u_2) = P\delta u_1 \tag{A2}$$

The strain measure chosen for these examples is the so-called rotated engineering strain [14]

$$\varepsilon = \frac{L_{\text{deformed}} - L}{L} \tag{A3}$$

where

$$L_{\text{deformed}} = \sqrt{L^2 - 2Hu_2 + u_2^2} \tag{A4}$$

and

$$\delta\varepsilon = \frac{u_2 - H}{L\sqrt{L^2 - 2Hu_2 + u_2^2}}\delta u_2 \tag{A5}$$

Then the equilibrium equations are

$$2A\sigma\frac{u_2 - H}{\sqrt{L^2 - 2Hu_2 + u_2^2}} - C(u_1 - u_2) = 0 \tag{A6}$$

and

$$C(u_1 - u_2) = P \tag{A7}$$

To trace the equilibrium paths, P versus u_1 and P versus u_2, these curves can be parametrized by u_2. The equilibrium equations are combined, which yields

$$P = 2A\sigma\frac{u_2 - H}{\sqrt{L^2 - 2Hu_2 + u_2^2}} \tag{A8}$$

and

$$u_1 = \frac{P}{C} + u_2 \tag{A9}$$

A.2 Elastic Case of the von Mises Truss. For the elastic case of the von Mises truss, the equilibrium equation (Eq. (91)) is obtained using the relation

$$\sigma = E\varepsilon \tag{A10}$$

A.2.1 Critical Spring Stiffness of the von Mises Truss. The stiffness of the spring in the von Mises truss dictates the overall behavior of the system. Stiffness below a critical value results in snap back behavior in the u_1 component, while values above do not. The critical case is realized when the tangent line to the P versus u_1 is vertical (i.e., $\partial u_1/\partial P = 0$), and when $u_1 = u_2 = H$, and $P = 0$. Then the expression for the critical value of the spring stiffness is

$$C_{cr} = 2\frac{EA}{L}\left(\frac{L}{\sqrt{L^2 - H^2}} - 1\right) \tag{A11}$$

and values from Sec. 5.1.3 give the critical spring stiffness for the configuration under investigation, $C_{cr} = 0.03094$.

A.3 Elastic-Plastic Case of the von Mises Truss. The analytical solution of the von Mises truss is based on the equilibrium equations derived in Sec. A.1. The stages of the solution and limiting/critical values of the flow stress material property are derived and discussed in the following subsections.

A.3.1 Stages of the Elasto-Plastic Solution. The stress-strain behavior of the elasto-plastic von Mises truss is comprised of two straight (linear) stages and two plateau stages (see Fig. 25(*b*)). A subscript in parentheses references the stage.

Stage 1: Elastic Compressive Stage: The first stage is equal to the elastic solution; therefore the $\sigma = E\varepsilon$. Stage 1 starts at the undeformed configuration (i.e., $u_1 = u_2 = 0$) and stops when the flow stress is reached in compression (i.e., when $\sigma = -\sigma_Y$). The limit on u_2 for stage 1 must be less than H, and is

$$u_{2\max}^{(1)} = H\left[1 - \sqrt{1 - \left(\frac{L}{H}\right)^2\left(\frac{\sigma_Y}{E}\right)\left(2 - \frac{\sigma_Y}{E}\right)}\right] \tag{A12}$$

Stage 2: Plateau Compressive Stage: The second stage is a plateau and the bars are in compression; therefore $\sigma = -\sigma_Y$. Stage 2 continues from the last point in stage 1 until the structure is flat, i.e.,

$$u_{2\max}^{(2)} = H \tag{A13}$$

Stage 3: Elastic Unloading/Reloading stage: The third stage is again elastic, and the stress is $\sigma = E(\varepsilon - \varepsilon_{\min}) - \sigma_Y$, where $\varepsilon_{\min}$ is strain at the end of stage 2, i.e., when $u_2 = H$

$$\varepsilon_{\min} = \sqrt{1 - \left(\frac{H}{L}\right)^2} - 1 \tag{A14}$$

Stage 3 continues from the last point in stage 2 until the flow stress is reached again, this time in tension. The limit on u_2 for stage 3 is found by equating $\sigma = E(\varepsilon - \varepsilon_{\min}) - \sigma_Y$ with $\sigma = \sigma_Y$, which results in

$$u_{2\max}^{(3)} = H + 2\sqrt{\frac{L^2\sigma_Y\left[\frac{\sigma_Y}{E} + \sqrt{1 - \left(\frac{H}{L}\right)^2}\right]}{E}} \tag{A15}$$

Stage 4: Plateau Tensile Stage: The fourth stage is a plateau, but now the bars are in tension, so $\sigma = \sigma_Y$. In this final stage there is a horizontal asymptote at $P_{\max} = 2A\sigma_Y$.

A.3.2 Limits on the Flow Stress. The flow stresses of the elasto-plastic von Mises truss in Sec. 5.1.3 are selected such that the material will reach the flow stress while the bars are in compression (i.e., before the structure reaches the plane configuration). Hence, the flow stress must be less than the maximum stress of the elastic stage, which occurs when $u_2 = H$.

$$\sigma_{Y,\max} = E\left[1 - \sqrt{1 - \left(\frac{H}{L}\right)^2}\right] \tag{A16}$$

For the values given in Sec. 5.1.3, $\sigma_{Y,\max} = 0.1340$. Additionally, depending on the value of the flow stress, the first plateau stage will start before or after the load limit point of the elastic solution, respectively. Therefore, the critical flow stress is equal to the stress at the load limit point

$$\sigma_{cr} = E\left\{1 - \frac{[(L^2 - H^2)L]^{\frac{1}{3}}}{L}\right\} \tag{A17}$$

For the values given in Sec. 5.1.3, $\sigma_{cr} = 0.0914$.

References

[1] Bergan, P. G., Horrigmoe, G., Brakeland, B., and Soreide, T. H., 1978, "Solution Techniques for Non-Linear Finite Element Problems," Int. J. Numer. Methods Eng., **12**(11), pp. 1677–1696.
[2] Mondkar, D. P., and Powell, G. H., 1978, "Evaluation of Solution Schemes for Nonlinear Structures," Comput. Struct., **9**(3), pp. 223–236.
[3] Clarke, M. J., and Hancock, G. J., 1990, "A Study of Incremental-Iterative Strategies for Non-Linear Analyses," Int. J. Numer. Methods Eng., **29**(7), pp. 1365–1391.

[4] Yang, Y.-B., and Shieh, M.-S., 1990, "Solution Method for Nonlinear Problems With Multiple Critical Points," AIAA J., **28**(12), pp. 2110–2116.
[5] Yang, Y.-B., and Kuo, S.-R., 1994, *Theory and Analysis of Nonlinear Framed Structures*, Prentice-Hall PTR, Englewood Cliffs, NJ.
[6] Rezaiee-Pajand, M., Tatar, M., and Moghaddasie, B., 2009, "Some Geometrical Bases for Incremental-Iterative Methods," Int. J. Eng. Trans. B: Appl., **22**(3), pp. 245–256.
[7] Reddy, J. N., 2004, *An Introduction to Nonlinear Finite Element Analysis*, Oxford University Press, New York.
[8] Zakharov, Y. V., Okhotkin, K. G., and Skorobogatov, A. D., 2004, "Bending of Bars Under a Follower Load," J. Appl. Mech. Tech.. Phys., **45**(5), pp. 756–763.
[9] Riks, E., 1979, "An Incremental Approach to the Solution of Snapping and Buckling Problems," Int. J. Solids Struct., **15**(7), pp. 529–551.
[10] Crisfield, M. A., 1981, "A Fast Incremental/Iterative Solution Procedure That Handles Snap-Through," Comput. Struct., **13**(1–3), pp. 55–62.
[11] Bathe, K. J., 1996, *Finite Element Procedures*, Prentice-Hall, Upper Saddle River, NJ.
[12] Belytschko, T., Liu, W., and Moran, B., 2000, *Nonlinear Finite Elements for Continua and Structures*, John Wiley & Sons, Inc., West Sussex, England.
[13] Bonet, J., 1997, *Nonlinear Continuum Mechanics for Finite Element Analysis*, Cambridge University Press, New York.
[14] Crisfield, M. A., 1991, *Non-Linear Finite Element Analysis of Solids and Structures. Volume 1: Essentials*, John Wiley & Sons, Inc., West Sussex, England.
[15] Crisfield, M. A., 1997, *Non-Linear Finite Element Analysis of Solids and Structures. Volume 2: Advanced Topics*, Wiley, John & Sons, Inc., West Sussex, England.
[16] Mang, H. A., Hofinger, G., and Jia, X., 2011, "On the Interdependency of Primary and Initial Secondary Equilibrium Paths in Sensitivity Analysis of Elastic Structures," Comput. Methods Appl. Mech. Eng., **200**(13–16), pp. 1558–1567.
[17] Adomian, G., 1988, "A Review of the Decomposition in Applied Mathematics," J. Math. Anal. Appl., **135**(2), pp. 501–544.
[18] He, J.-H., 2000, "A Review on Some New Recently Developed Nonlinear Analytical Techniques," Int. J. Nonlinear Sci. Numer. Simul., **1**(1), pp. 51–70.
[19] He, J.-H., 2006, "Some Asymptotic Methods for Strongly Nonlinear Equations," Int. J. Mod. Phys. B, **20**(10), pp. 1141–1199.
[20] He, J.-H., 2006, "Addendum New Interpretation of Homotopy Perturbation Method," Int. J. Mod. Phys. B, **20**(18), pp. 2561–2568.
[21] Babajee, D. K. R., and Dauhoo, M. Z., 2006, "An Analysis of the Properties of the Variants of Newton's Method With Third Order Convergence," Appl. Math. Comput., **183**(1), pp. 659–684.
[22] Adomian, G., 1983, *Stochastic Systems*, Academic Press, New York.
[23] Adomian, G., and Rach, R., 1985, "On the Solution of Algebraic Equations by the Decomposition Method," J. Math. Anal. Appl., **105**(1), pp. 141–166.
[24] Abbaoui, K., and Cherruault, Y., 1994, "Convergence of Adomian's Method Applied to Nonlinear Equations," Math. Comput. Model., **20**(9), pp. 69–73.
[25] Babolian, E., and Biazar, J., 2002, "Solution of Nonlinear Equations by Modified Adomian Decomposition Method," Appl. Math. Comput. **132**(1), pp. 167–172.
[26] Abbasbandy, S., 2003, "Improving Newton-Raphson Method for Nonlinear Equations by Modified Adomian Decomposition Method," Appl. Math. Comput., **145**(2–3), pp. 887–893.
[27] Chun, C., 2006, "A New Iterative Method for Solving Nonlinear Equations," Appl. Math. Comput., **178**(2), pp. 415–422.
[28] Darvishi, M. T., and Barati, A., 2007, "Super Cubic Iterative Methods to Solve Systems of Nonlinear Equations," Appl. Math. Comput., **188**(2), pp. 1678–1685.
[29] Darvishi, M. T., and Barati, A., 2007, "A Third-Order Newton-Type Method to Solve Systems of Nonlinear Equations," Appl. Math. Comput., **187**(2), pp. 630–635.
[30] Noor, M. A., and Noor, K. I., 2006, "Three-Step Iterative Methods for Nonlinear Equations," Appl. Math. Comput., **183**(1), pp. 322–327.
[31] He, J.-H., 1999, "Homotopy Perturbation Technique," Comput. Methods Appl. Mech. Eng., **178**(3–4), pp. 257–262.
[32] He, J.-H., 2000, "A Coupling Method of a Homotopy Technique and a Perturbation Technique for Non-Linear Problems," Int. J. Nonlinear Mech., **35**(1), pp. 37–43.
[33] He, J.-H., 2003, "Homotopy Perturbation Method: A New Nonlinear Analytical Technique," Appl. Math. Comput., **135**(1), pp. 73–79.
[34] He, J.-H., 2008, "An Elementary Introduction to Recently Developed Asymptotic Methods and Nanomechanics in Textile Engineering," Int. J. Mod. Phys. B, **22**(21), pp. 3487–3578.
[35] Golbabai, A., and Javidi, M., 2007, "A New Family of Iterative Methods for Solving System of Nonlinear Algebric Equations," Appl. Math. Comput., **190**(2), pp. 1717–1722.
[36] Jorabchi, K., and Suresh, K., 2011, "A Robust Continuation Method to Pass Limit-Point Instability" Finite Elem. Anal. Design, **47**(11), pp. 1253–1261.
[37] Liao, S.-J., and Cheung, K. F., 2003, "Homotopy Analysis of Nonlinear Progressive Waves in Deep Water," J. Eng. Math., **45**(2), pp. 105–116.
[38] He, J.-H., 2005, "Application of Homotopy Perturbation Method to Nonlinear Wave Equations," Chaos, Solitons Fractals, **26**(3), pp. 695–700.
[39] Sadighi, A., and Ganji, D. D., 2007, "Solution of the Generalized Nonlinear Boussinesq Equation Using Homotopy Perturbation and Variational Iteration Methods," Int. J. Nonlinear Sci. Numer. Simul., **8**(3), pp. 435–443.
[40] Abbasbandy, S., 2006, "The Application of Homotopy Analysis Method to Nonlinear Equations Arising in Heat Transfer," Phys. Lett. A, **360**(1), pp. 109–113.
[41] Ganji, D. D., and Sadighi, A., 2007, "Application of Homotopy-Perturbation and Variational Iteration Methods to Nonlinear Heat Transfer and Porous Media Equations," J. Comput. Appl. Math., **207**(1), pp. 24–34.
[42] Domairry, G., and Nadim, N., 2008, "Assessment of Homotopy Analysis Method and Homotopy Perturbation Method in Non-Linear Heat Transfer Equation," Int. Commun. Heat Mass Transfer, **35**(1), pp. 93–102.
[43] Ghorbani, A., and Saberi-Nadjafi, J., 2007, "He's Homotopy Perturbation Method for Calculating Adomian Polynomials," Int. J. Nonlinear Sci. Numer. Simul., **8**(2), pp. 229–232.
[44] Ozis, T., and Yildirim, A., 2008, "Comparison Between Adomian's Method and He's Homotopy Perturbation Method," Comput. Math. Appl., **56**(5), pp. 1216–1224.
[45] Ghorbani, A., 2009, "Beyond Adomian Polynomials: He Polynomials," Chaos, Solitons Fractals, **39**(3), pp. 1486–1492.
[46] Kou, J., Li, Y., and Wang, X., 2006, "A Modification of Newton Method With Third-Order Convergence," Appl. Math. Comput., **181**(2), pp. 1106–1111.
[47] Weerakoon, S., and Fernando, T. G. I., 2000, "A Variant of Newton's Method With Accelerated Third-Order Convergence," Appl. Math. Lett., **13**(8), pp. 87–93.
[48] Frontini, M., and Sormani, E., 2003, "Some Variant of Newton's Method With Third-Order Convergence," Appl. Math. Comput., **140**(2–3), pp. 419–426.
[49] Cardoso, E. L., and Fonseca, J. S. O., 2007, "The GDC Method as an Orthogonal Arc-Length Method," Commun. Numer. Methods Eng., **23**(4), pp. 263–271.
[50] Homeier, H. H. H., 2005, "On Newton-Type Methods With Cubic Convergence," J. Comput. Appl. Math., **176**(2), pp. 425–432.
[51] Jisheng, K., Yitian, L., and Xiuhua, W., 2006, "A Uniparametric Chebyshev-Type Method Free From Second Derivatives," Appl. Math. Comput., **179**(1), pp. 296–300.
[52] Babajee, D. K. R., Dauhoo, M. Z., Darvishi, M. T., Karami, A., and Barati, A., 2010, "Analysis of Two Chebyshev-Like Third Order Methods Free From Second Derivatives for Solving Systems of Nonlinear Equations," J. Comput. Appl. Math., **233**(8), pp. 2002–2012.
[53] King, R., 1973, "A Family of Fourth Order Methods for Nonlinear Equations," *SIAM* J. Numer. Anal., **10**(5), pp. 876–879.
[54] Sharma, J. R., and Guha, R. K., 2007, "A Family of Modified Ostrowski Methods With Accelerated Sixth Order Convergence," Appl. Math. Comput., **190**(1), pp. 111–115.
[55] Bi, W., Wu, Q., and Ren, H., 2009, "A New Family of Eighth-Order Iterative Methods for Solving Nonlinear Equations," Appl. Math. Comput., **214**(1), pp. 236–245.
[56] Wriggers, P., and Simo, J. C., 1990, "A General Procedure for the Direct Computation of Turning and Bifurcation Points," Int. J. Numer. Methods Eng., **30**(1), pp. 155–176.
[57] Planinc, I., and Saje, M., 1999, "A Quadratically Convergent Algorithm for the Computation of Stability Points: The Application of the Determinant of the Tangent Stiffness Matrix," Comput. Methods Appl. Mech. Eng., **169**(1–2), pp. 89–105.
[58] Fujii, F., and Okazawa, S., 1997, "Pinpointing Bifurcation Points and Branch-Switching," J. Eng. Mech., **123**(3), pp. 179–189.
[59] Fujii, F., and Ramm, E., 1997, "Computational Bifurcation Theory: Path-Tracing, Pinpointing and Path-Switching," Eng. Struct., **19**(5), pp. 385–392.
[60] Balay, S., Gropp, W. D., McInnes, L. C., and Smith, B. F., 1997, "Efficient Management of Parallelism in Object Oriented Numerical Software Libraries," *Modern Software Tools in Scientific Computing*, E. Arge, A. M. Bruaset, and H. P. Langtangen, eds., Birkhäuser Press, New York, pp. 163–202.
[61] Balay, S., Brown, J., Buschelman, K., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H., 2011, "PETSc Users Manual," Revision 3.2, Argonne National Laboratory, Tech. Report No. ANL-95/11.
[62] Balay, S., Brown, J., Buschelman, K., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H., 2011, "PETSc Web Page," http://www.mcs.anl.gov/petsc/
[63] Kirk, B. S., Peterson, J. W., Stogner, R. H., and Carey, G. F., 2006, "libMesh: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations," Eng. Comput., **22**(3–4), pp. 237–254.
[64] Salinger, A. G., Bou-Rabee, N. M., Pawlowski, R. P., Wilkes, E. D., Burroughs, E. A., Lehoucq, R. B., and Romero, L. A., 2002, "LOCA 1.0 Library of Continuation Algorithms: Theory and Implementation Manual," Sandia National Laboratories, Albuquerque, NM, Technical Report No. SAND2002-0396.
[65] Batoz, J.-L., and Dhatt, G., 1979, "Incremental Displacement Algorithms for Nonlinear Problems," Int. J. Numer. Methods Eng., **14**(8), pp. 1262–1267.
[66] Lages, E. N., Paulino, G. H., Menezes, I. F. M., and Silva, R. R., 1999, "Nonlinear Finite Element Analysis using an Object-Oriented Philosophy-Application to Beam Elements and to the Cosserat Continuum," Eng. Comput., **15**(1), pp. 73–89.
[67] Riks, E., 1972, "The Application of Newton's Method to the Problem of Elastic Stability," ASME J. Appl. Mech., **39**(4), pp. 1060–1066.
[68] Powell, G., and Simons, J., 1981, "Improved Iteration Strategy for Nonlinear Structures," Int. J. Numer. Methods Eng., **17**(10), pp. 1455–1467.
[69] Padovan, J., and Tovichakchaikul, S., 1982, "Self-Adaptive Predictor-Corrector Algorithms for Static Nonlinear Structural Analysis," Comput. Struct., **15**(4), pp. 365–377.

[70] Caballero, A., Willam, K. J., and Carol, I., 2008, "Consistent Tangent Formulation for 3D Interface Modeling of Cracking/Fracture in Quasi-Brittle Materials," Comput. Methods Appl. Mech. Eng., **197**(33–40), pp. 2804–2822.
[71] Fujii, F., Choong, K. K., and Gong, S.-X., 1992, "Variable Displacement Control to Overcome Turning Points of Nonlinear Elastic Frames," Comput. Struct., **44**(1–2), pp. 133–136.
[72] Simons, J., and Bergan, P. G., 1984, "Hyperplane Displacement Control Methods in Nonlinear Analysis," *Innovative Methods for Nonlinear Problems*, W. K. Liu, T. Belytschko, and K. C. Park, eds., Pineridge Press, Swansea, UK, pp. 345–364.
[73] Ramm, E., 1981, "Strategies for Tracing Nonlinear Responses Near Limit Points," *Nonlinear Finite Element Analysis in Structural Mechanics*, Springer-Verlag, New York, p. 68.
[74] Bellini, P. X., and Chuyla, A., 1987, "An Improved Automatic Incremental Algorithm for the Efficient Solution of Nonlinear Finite Element Equations," Comput. Struct., **26**(1–2), pp. 99–110.
[75] Park, K. C., 1982, "A Family of Solution Algorithms for Nonlinear Structural Analysis Based on Relaxation Equations," Int. J. Numer. Methods Eng., **18**(9), pp. 1337–1347.
[76] Wempner, G. A., 1971, "Discrete Approximations Related to Nonlinear Theories of Solids," Int. J. Solids Struct., **7**(11), pp. 1581–1599.
[77] Forde, B. W. R., and Stiemer, S. F., 1987, "Improved Arc Length Orthogonality Methods for Nonlinear Finite Element Analysis," Comput. Struct., **27**(5), pp. 625–630.
[78] Crisfield, M. A., 1983, "An Arc-Length Method Including Line Searches and Accelerations," Int. J. Numer. Methods Eng., **19**(9), pp. 1269–1289.
[79] Lam, W. F., and Morley, C. T., 1992, "Arc-Length Method for Passing Limit Points in Structural Calculation," J. Struct. Eng., **118**(1), pp. 169–185.
[80] Ritto-Correa, M., and Camotim, D., 2008, "On the Arc-Length and Other Quadratic Control Methods: Established, Less Known and New Implementation Procedures," Comput. Struct., **86**(11–12), pp. 1353–1368.
[81] Carrera, E., 1994, "A Study on Arc-Length-Type Methods and Their Operation Failures Illustrated by a Simple Model," Comput. Struct., **50**(2), pp. 217–229.
[82] Feng, Y. T., Peric, D., and Owen, D. R. J., 1996, "A New Criterion for Determination of Initial Loading Parameter in Arc-Length Methods," Comput. Struct., **58**(3), pp. 479–485.
[83] Yang, Y.-B., and McGuire, W., 1985, "A Work Control Method for Geometrically Nonlinear Analysis," Proceedings of the 1985 International Conference on Numerical Methods in Engineering: Theory and Applications, J. Middleton and G. N. Pande, eds., pp. 913–921.
[84] Al-Rasby, S. N., 1991, "Solution Techniques in Nonlinear Structural Analysis," Comput. Struct., **40**(4), pp. 985–993.
[85] Carol, I., López, C. M., and Roa, O., 2001, "Micromechanical Analysis of Quasi-Brittle Materials Using Fracture-Based Interface Elements," Int. J. Numer. Methods Eng., **52**(1–2), pp. 193–215.
[86] Gao, Y. F., and Bower, A. F., 2004, "A Simple Technique for Avoiding Convergence Problems in Finite Element Simulations of Crack Nucleation and Growth on Cohesive Interfaces," Model. Simul. Mater. Sci. Eng., **12**(3), pp. 453–463.
[87] Ngo, D., Park, K., Paulino, G. H., and Huang, Y., 2010, "On the Constitutive Relation of Materials With Microstructure Using a Potential-Based Cohesive Model for Interface Interaction," Eng. Fract. Mech., **77**(7), pp. 1153–1174.
[88] Yang, Z., and Chen, J., 2004, "Fully Automatic Modelling of Cohesive Discrete Crack Propagation in Concrete Beams Using Local Arc-Length Methods," Int. J. Solids Struct., **41**(3–4), pp. 801–826.
[89] Gu, H., and Chattopadhyay, A., 1996, "Delamination Buckling and Postbuckling of Composite Cylindrical Shells," AIAA J., **34**(6), pp. 1279–1286.
[90] Alfano, G., and Crisfield, M. A., 2001, "Finite Element Interface Models for the Delamination Analysis of Laminated Composites: Mechanical and Computational Issues," Int. J. Numer. Methods Eng., **50**(7), pp. 1701–1736.
[91] Mallardo, V., and Allesandri, C., 2004, "Arc-Length Procedures With BEM in Physically Nonlinear Problems," Eng. Anal. Boundary Elem, **28**(6), pp. 547–559.
[92] Mukherjee, S., and Chandra, A., 1984, "Boundary Element Formulations for Large Strain-Large Deformation Problems of Plasticity and Viscoplasticity," *Developments in Boundary Element Methods-3*, P. K. Banerjee and S. Mukherjee, eds. Elsevier, Barking, Essex, UK, Chap. 2, pp. 27–58.
[93] Chandra, A., and Mukherjee, S., 1983, "Applications of the Boundary Element Method to Large Strain Large Deformation Problems of Viscoplasticity," J. Strain Anal. Eng. Design, **18**(4), pp. 261–270.
[94] Chandra, A., and Mukherjee, S., 1984, "Boundary Element Formulations for Large Strain-Large Deformation Problems of Viscoplasticity," Int. J. Solids Struct., **20**(1), pp. 41–53.
[95] Paulino, G. H., and Liu, Y., 2001, "Implicit Consistent and Continuum Tangent Operators in Elastoplastic Boundary Element Formulations," Comput. Methods Appl. Mech. Eng., **190**(15–17), pp. 2157–2179.
[96] Bathe, K. J., and Dvorkin, E. N., 1983, "On the Automatic Solution of Nonlinear Finite Element Equations," Comput. Struct., **17**(5–6), pp. 871–879.
[97] Lin, T. W., Yang, Y.-B., and Shiau, H. T., 1993, "A Work Weighted State Vector Control Method for Geometrically Nonlinear Analysis," Comput. Struct., **46**(4), pp. 689–694.
[98] Chen, H., and Blandford, G. E., 1993, "Work-Increment-Control Method for Non-Linear Analysis," Int. J. Numer. Methods Eng., **36**(6), pp. 909–930.
[99] Kouhia, R., 2008, "Stabilized Forms of Orthogonal Residual and Constant Incremental Work Control Path Following Methods," Comput. Methods Appl. Mech. Eng., **197**(13–16), pp. 1389–1396.
[100] Yang, Y.-B., Lin, T. J., Leu, L. J., and Huang, C., 2008, "Inelastic Postbuckling Response of Steel Trusses Under Thermal Loadings," J. Constr. Steel Res., **64**(12), pp. 1394–1407.
[101] Thai, H.-T., and Kim, S.-E., 2009, "Large Deflection Inelastic Analysis of Space Trusses Using Generalized Displacement Control Method," J. Constr. Steel Res., **65**(10–11), pp. 1987–1994.
[102] Liew, J. Y. R., Punniyakotty, N. M., and Shanmugam, N. E., 1997, "Advanced Analysis and Design of Spatial Structures," J. Constr. Steel Res., **42**(1), pp. 21–48.
[103] Lakshmi, B., and Shanmugam, N. E., 2002, "Nonlinear Analysis of In-Filled Steel-Concrete Composite Columns," J. Struct. Eng., **128**(7), pp. 922–933.
[104] Li, C., and Chou, T.-W., 2003, "Multiscale Modeling of Carbon Nanotube Reinforced Polymer Composites," J. Nanosci. Nanotechnol., **3**(5), pp. 423–430.
[105] Li, C., and Chou, T.-W., 2003, "Elastic Moduli of Multi-Walled Carbon Nanotubes and the Effect of van der Waals Forces," Compos. Sci. Technol., **63**(11), pp. 1517–1524.
[106] Krenk, S., 1995, "An Orthogonal Residual Procedure for Non-Linear Finite Element Equations," Int. J. Numer. Methods Eng., **38**(5), pp. 823–839.
[107] Asferg, J. L., Poulsen, P. N., and Nielsen, L. O., 2007, "A Consistent Partly Cracked XFEM Element for Cohesive Crack Growth," Int. J. Numer. Methods Eng., **72**(4), pp. 464–485.
[108] Mougaard, J. F., Poulsen, P. N., and Nielsen, L. O., 2007, "An Enhanced Cohesive Crack Element for XFEM Using a Double Enriched Displacement Field," Proceedings of the 6th International Conference on Fracture Mechanics of Concrete and Concrete Structures, pp. 139–146.
[109] Krenk, S., and Hededal, O., 1995, "A Dual Orthogonality Procedure for Non-Linear Finite Element Equations," Comput. Methods Appl. Mech. Eng., **123**(1–4), pp. 95–107.
[110] Poulsen, P. N., and Damkilde, L., 1996, "A Flat Triangular Shell Element With Loof Nodes," Int. J. Numer. Methods Eng., **39**(22), pp. 3867–3887.
[111] Renard, Y., and Pommier, J., 2011, "Getfem++ Web Page," http://download.gna.org/getfem/html/homepage/
[112] Schenk, O., and Gartner, K., 2004, "Solving Unsymmetric Sparse Systems of Linear Equations With PARDISO," J. Future Gener. Comput. Syst., **20**(3), pp. 475–487.
[113] Schenk, O., and Gartner, K., 2006, "On Fast Factorization Pivoting Methods for Symmetric Indefinite Systems," Electron. Trans. Numer. Anal., **23**, pp. 158–179.
[114] Davis, T. A., 2004, "A Column Pre-Ordering Strategy for the Unsymmetric-Pattern Multifrontal Method," ACM Trans. Math. Softw., **30**(2), pp. 165–195.
[115] Mello, L. A. M., De Sturler, E., Paulino, G. H., and Silva, E. C. N., 2010, "Recycling Krylov Subspaces for Efficient Large-Scale Electrical Impedance Tomography," Comput. Methods Appl. Mech. Eng., **199**(49–52), pp. 3101–3110.
[116] Parks, M. L., De Sturler, E., Mackey, G., Johnson, D. D., and Maiti, S., 2006, "Recycling Krylov Subspaces for Sequences of Linear Systems," SIAM J. Sci. Comput., **28**(5), pp. 1651–1674.
[117] Pai, F., 2007, *Highly Flexible Structures: Modeling, Computation, and Experimentation*, American Institute of Aeronautics and Astronautics, Reston, VA.
[118] Doyle, J. F., 2010, *Nonlinear Analysis of Thin-Walled Structures: Statics, Dynamics, and Stability*, Springer-Verlag, New York.
[119] Palazaotto, A. N., and Dennis, S. T., 1992, *Nonlinear Analysis of Shell Structures*, American Institute of Aeronautics and Astronautics, Reston, VA.
[120] Stanciulescu, I., 2009, private communication.
[121] Bergan, P. G., 1980, "Solution Algorithms for Nonlinear Structural Problems," Comput. Struct., **12**(4), pp. 497–509.
[122] Bazant, Z. P., and Cedolin, L., 1991, *Stability of Structures: Elastic, Inelastic, Fracture and Damage Theories*, Oxford University Press, New York.
[123] Leon, S., 2010, "A Unified Library of Nonlinear Solutions Schemes: An Excursion Into Nonlinear Computational Mechanics," M.S. thesis, University of Illinois at Urbana-Champaign.
[124] Simo, J., and Hughes, T. J. R., 1998, *Computational Inelasticity*, Vol. 7., Springer-Verlag, New York.
[125] Yang, Y.-B., and Leu, L.-J., 1991, "Constitutive Laws and Force Recovery Procedures in Nonlinear Analysis of Trusses," Comput. Methods Appl. Mech. Eng., **92**(1), pp. 121–131.
[126] Yang, Y.-B., Leu, L.-J., and Yang, J. P., 2007, "Key Considerations in Tracing the Postbuckling Response of Structures With Multi Winding Loops," Mech. Adv. Mater. Struct., **14**(3), pp. 175–189.
[127] Lee, S. L., Manuel, F. S., and Rossow, E. C., 1968, "Large Deflections and Stability of Elastic Frames," ASCE Proc. J. Eng. Mech. Div., **94**(2), pp. 521–548.
[128] Schweizerhof, K. H., and Wriggers, P., 1986, "Consistent Linearization for Path Following Methods in Nonlinear FE Analysis," Comput. Methods Appl. Mech. Eng., **59**(3), pp. 261–279.
[129] Parente, E., and Vaz, L. E., 2001, "Improvement of Semi-Analytical Design Sensitivities of Non-Linear Structures Using Equilibrium Relations," Int. J. Numer. Methods Eng., **50**(9), pp. 2127–2142.
[130] Celes, W., Paulino, G. H., and Espinha, R., 2005, "A Compact Adjacency-Based Topological Data Structure for Finite Element Mesh Representation," Int. J. Numer. Methods Eng., **64**(11), pp. 1529–1556.
[131] Heath, M. T., 2002, *Scientific Computing: An Introductory Survey*, McGraw-Hill, New York.
[132] Wang, S., De Sturler, E., and Paulino, G. H., 2007, "Large-Scale Topology Optimization Using Preconditioned Krylov Subspace Methods With Recycling," Int. J. Numer. Methods Eng., **69**(12), pp. 2441–2468.
[133] van der Vorst, H. A., 2003, *Iterative Krylov Methods for Large Linear Systems*, Cambridge University Press, Cambridge, UK.