Arc Length Control
by Louie L. Yaw
Walla Walla University
Draft Date August 5, 2017

key words: nonlinear finite element analysis, path following, arc length control, predictor solution

1 Introduction

Sometimes it is necessary to determine the nonlinear load displacement response of a structure. Possible nonlinearities generally arise due to material nonlinearity, geometric nonlinearity, loads that change direction during loading, or a sudden change in boundary conditions. The most common are the first two, material and geometric nonlinearities. However, any of the nonlinearities may occur by themselves or in combination with others. Performing a nonlinear analysis often requires an incremental analysis with Newton-Raphson iterations at the global level of the structure to enforce equilibrium. The load displacement curve generated is then an equilibrium path. In nonlinear finite element analysis such path following is often required to examine the equilibrium characteristics of a structure. One important method for such path following is the arc length method. It is the intent of this paper to present the ingredients and derivations needed to implement the arc length method in a nonlinear finite element analysis.

2 Ingredients of a Quasi-Static Nonlinear Analysis

At the beginning the equilibrium path is not known. The objective is to determine the path by applying loads and displacements to the structure incrementally. A rough overview of the process is provided below to give a sense of how the analysis is carried out. Each increment of the nonlinear analysis is determined by the following steps:

1. Start on the equilibrium path (see Figure 1), point $\mathbf{g}_{converged}$, where $\mathbf{g}$ represents a residual vector, which quantifies the current level of equilibrium error.

2. Begin a new increment of the solution by taking a predictor step (or solution)

3. Check for equilibrium between internal and external loads

4. If equilibrium is not below some tolerance then iterate to improve the predictor step

5. Once iterations improve equilibrium below specified tolerance, stop iterations and make final updates to current solution increment

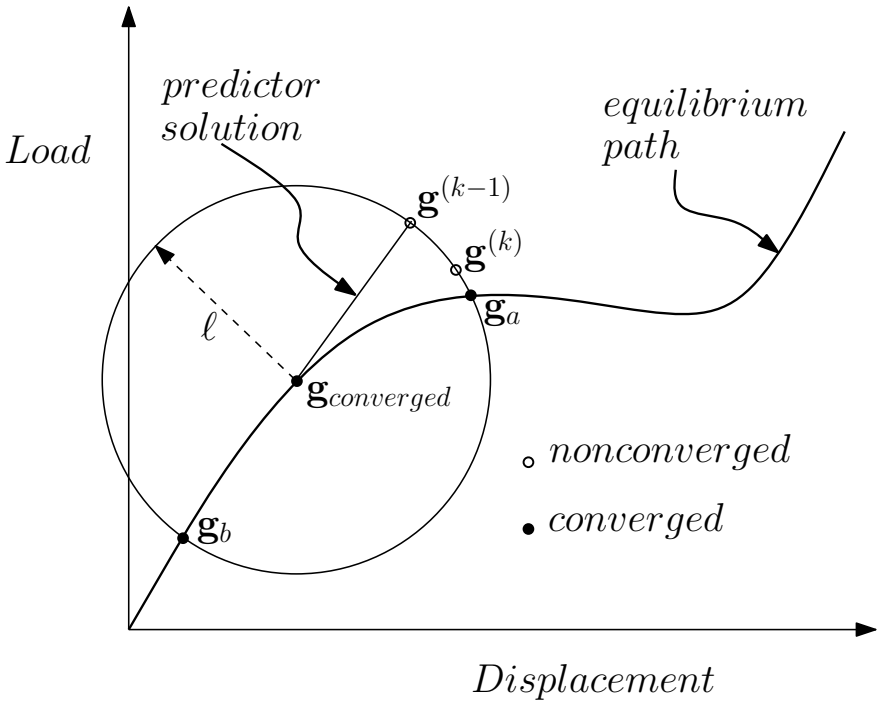


Figure 1: Predictor solution and iterations with arc length constraint

6. Return to step 1 and proceed to a new increment in the solution

Figure 2 serves to further elaborate the pieces of a path following process. Notice the global increments that piece together the equilibrium path. Each increment is due to displacement increment $\Delta\mathbf{u}$ and external load increment $\Delta\lambda\mathbf{q}_e$ (where $\mathbf{q}_e$ is the external load vector). Within each increment iterations are made to improve the increments, these iterative improvements (of which there are as many as necessary to reach equilibrium), are denoted as $\delta\mathbf{u}$ and $\delta\lambda\mathbf{q}_e$. It should be mentioned that the 2D representation shown in Figure 2 is conceptual in that the variables $\mathbf{u}$ and $\mathbf{q}_e$ are vectors. In practice, such plots are created based on a single structure load and a displacement at a single dof of the structure.

2.1 Iterating for Equilibrium with Arc Length Constraint

Assume that an increment of displacement (predictor solution) is taken in a direction that is estimated to follow the equilibrium path. (A scheme for determining a predictor solution is presented in section 2.2.) In general, the predictor solution is not on the equilibrium path. Hence, it is necessary to have a scheme to iteratively improve this estimated displacement until the solution is back on the equilibrium path within some specified tolerance. It is the purpose of this section to derive a method, using Newton-Raphson iterations with arc length constraint, to achieve this goal.

In a nonlinear finite element analysis a vector of external forces, $\mathbf{q}_e$, is specified. The external force vector acts on the structure and contains the relative loads that are applied to the structure incrementally. It is unnecessary to specify the final correct loads, it is only necessary to have the loads in the correct proportion to each other as they act on the

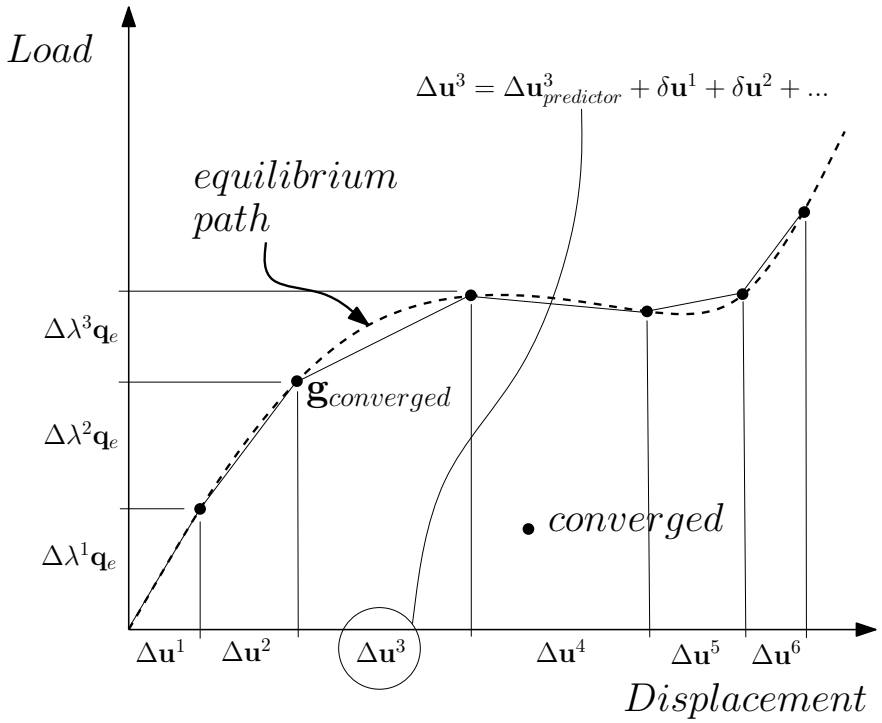


Figure 2: Incrementally, and iteratively, determined equilibrium path

structure degrees of freedom. The nonlinear analysis is performed by adding up incremental effects of the external load vector. Hence, it is really only used to get the load proportions correct. The amount of incremental external force that is applied during a given increment is denoted as $\Delta\lambda\mathbf{q}_e$, where $\Delta\lambda$ is a scalar load parameter determined during the incremental finite element analysis. The total scalar load parameter, λ, is recorded during the incremental finite element analysis and consists of the sum of the incremental scalar load parameters, $\Delta\lambda$. Hence, at the current point on the equilibrium path for the structure, the total external force vector, applied to the structure, is represented by $\lambda\mathbf{q}_e$. The total external force vector and the structure internal force vector, $\mathbf{q}_i$, are compared to each other. The difference between these vectors, known as the residual, $\mathbf{g}$, represents the error in equilibrium between external and internal forces. The goal is to make corrections to λ and the structure displacement vector, $\mathbf{u}$, by iterations to reduce the *norm* of the residual below an acceptable tolerance. If this is done successfully the λ, $\mathbf{u}$ pair found represents a point on the equilibrium path of the structure. It is important to note that the residual, $\mathbf{g}$, is a function of λ and $\mathbf{u}$. The residual or out of balance force vector is written as

$$\mathbf{g}(\mathbf{u}, \lambda) = \mathbf{q}_i(\mathbf{u}) - \lambda\mathbf{q}_e(\mathbf{u}). \tag{2.1}$$

Recall that a basic 1D Taylor series expanded about some point a is

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \dots \tag{2.2}$$

Then write (2.1) as a Taylor series with higher order terms neglected

$$\mathbf{g}^{(k)} = \mathbf{g}^{(k-1)} + \frac{\partial\mathbf{g}}{\partial\mathbf{u}}\delta\mathbf{u} + \frac{\partial\mathbf{g}}{\partial\lambda}\delta\lambda, \tag{2.3}$$

where $\mathbf{g}^{(k-1)}$ is the old residual from the last iteration and $\mathbf{g}^{(k)}$ is the new residual from the current iteration, k. The goal is to reduce the norm of the residual in each successive iteration (see section 2.3 and equation (2.27)).

Next, determine the relevant pieces needed for (2.3). First, consider the expression for $\frac{\partial \mathbf{g}}{\partial \mathbf{u}}$.

$$\frac{\partial \mathbf{g}}{\partial \mathbf{u}} = \frac{\partial \mathbf{q}_i}{\partial \mathbf{u}}\delta\mathbf{u} - \lambda\frac{\partial \mathbf{q}_e}{\partial \mathbf{u}}\delta\mathbf{u} = \left(\frac{\partial \mathbf{q}_i}{\partial \mathbf{u}} - \lambda\frac{\partial \mathbf{q}_e}{\partial \mathbf{u}}\right)\delta\mathbf{u} = \left[(\mathbf{k}_{t1} + \mathbf{k}_{t\sigma}) - \lambda\mathbf{k}_p\right]\delta\mathbf{u} = \mathbf{K}_T\delta\mathbf{u}, \tag{2.4}$$

where,

$$\mathbf{k}_{t1} + \mathbf{k}_{t\sigma} - \lambda\mathbf{k}_p = \mathbf{K}_T. \tag{2.5}$$

Second, observe that

$$\frac{\partial \mathbf{g}}{\partial \lambda} = \mathbf{q}_e. \tag{2.6}$$

Hence, substituting the results of (2.4) and (2.6) into (2.3) yields (setting $\mathbf{g}^{(k)} = 0$ since it is the aim to reach equilibrium)

$$0 = \mathbf{g}^{(k)} = \mathbf{g}^{(k-1)} + \mathbf{K}_T\delta\mathbf{u} - \mathbf{q}_e\delta\lambda. \tag{2.7}$$

In essence, the goal is to find $\delta\mathbf{u}$ and $\delta\lambda$ such that $\mathbf{g}^{(k)} = 0$. In other words, an attempt at equilibrium is sought at some iteration away from the last residual $\mathbf{g}^{(k-1)}$. Each successive iteration attempts to improve the values of $\delta\mathbf{u}$ and $\delta\lambda$. When the residual $\mathbf{g}^{(k)}$ becomes sufficiently close to zero, based on some specified tolerance, then iterations cease and the process continues with the next predictor step.

Continuing with (2.7) yields

$$\begin{aligned} &\mathbf{K}_T^{(k-1)}\delta\mathbf{u}^{(k)} - \delta\lambda^{(k)}\mathbf{q}_e = -\mathbf{g}^{(k-1)} \\ \Rightarrow &\mathbf{K}_T^{(k-1)}\delta\mathbf{u}^{(k)} = -\mathbf{g}^{(k-1)} + \delta\lambda^{(k)}\mathbf{q}_e \\ \Rightarrow &\delta\mathbf{u}^{(k)} = -(\mathbf{K}_T^{(k-1)})^{-1}\mathbf{g}^{(k-1)} + \delta\lambda^{(k)}(\mathbf{K}_T^{(k-1)})^{-1}\mathbf{q}_e. \end{aligned} \tag{2.8}$$

Then the last line of (2.8) is expressed as

$$\delta\mathbf{u}^{(k)} = \delta\mathbf{u}^* + \delta\lambda^{(k)}\delta\bar{\mathbf{u}}, \tag{2.9}$$

where

$$\delta\mathbf{u}^* = -(\mathbf{K}_T^{(k-1)})^{-1}\mathbf{g}^{(k-1)} \tag{2.10}$$

and

$$\delta\bar{\mathbf{u}} = (\mathbf{K}_T^{(k-1)})^{-1}\mathbf{q}_e. \tag{2.11}$$

Notice that (2.9) is the formula for the correction, $\delta\mathbf{u}^{(k)}$. This correction is intended to improve the predictor solution and lead closer and closer to the equilibrium path as iterations continue. Furthermore, the term $\delta\lambda^{(k)}$ is not yet known. It turns out that $\delta\lambda^{(k)}$ comes from the arc length constraint itself. A cylindrical arc length constraint is written as

$$(\Delta\mathbf{u}^{(k)})^T\Delta\mathbf{u}^{(k)} = \ell^2, \tag{2.12}$$

where $\Delta\mathbf{u}^{(k)}$ indicates the current incremental structure displacement vector at iteration k. Then rewriting $\Delta\mathbf{u}^{(k)}$ using (2.9) yields

$$\Delta\mathbf{u}^{(k)} = \Delta\mathbf{u}^{(k-1)} + \delta\mathbf{u}^{(k)} = \Delta\mathbf{u}^{(k-1)} + \delta\mathbf{u}^{*} + \delta\lambda^{(k)}\delta\bar{\mathbf{u}}. \tag{2.13}$$

Next substitute (2.13) into (2.12) to get

$$(\Delta\mathbf{u}^{(k-1)} + \delta\mathbf{u}^{*} + \delta\lambda^{(k)}\delta\bar{\mathbf{u}})^{T}(\Delta\mathbf{u}^{(k-1)} + \delta\mathbf{u}^{*} + \delta\lambda^{(k)}\delta\bar{\mathbf{u}}) = \ell^{2}. \tag{2.14}$$

After some algebra obtain

$$\delta\bar{\mathbf{u}}^{T}\delta\bar{\mathbf{u}}(\delta\lambda^{(k)})^{2} + 2(\Delta\mathbf{u}^{(k-1)} + \delta\mathbf{u}^{*})^{T}\delta\bar{\mathbf{u}}\delta\lambda^{(k)} + (\Delta\mathbf{u}^{(k-1)} + \delta\mathbf{u}^{*})^{T}(\Delta\mathbf{u}^{(k-1)} + \delta\mathbf{u}^{*}) - \ell^{2} = 0. \tag{2.15}$$

Expression (2.15) is further simplified as

$$a(\delta\lambda^{(k)})^{2} + b\delta\lambda^{(k)} + c = 0 \tag{2.16}$$

where

$$\begin{aligned} a &= \delta\bar{\mathbf{u}}^{T}\delta\bar{\mathbf{u}} \\ b &= 2(\Delta\mathbf{u}^{(k-1)} + \delta\mathbf{u}^{*})^{T}\delta\bar{\mathbf{u}} \\ c &= (\Delta\mathbf{u}^{(k-1)} + \delta\mathbf{u}^{*})^{T}(\Delta\mathbf{u}^{(k-1)} + \delta\mathbf{u}^{*}) - \ell^{2}. \end{aligned} \tag{2.17}$$

From the above two roots are possible for $\delta\lambda^{(k)}$. By using the quadratic equation, they are (a 'hat' is used over λ to denote a two solution variable from which a single solution is to be chosen)

$$\delta\hat{\lambda}^{(k)} = \frac{-b \pm \sqrt{b^{2} - 4ac}}{2a}. \tag{2.18}$$

Notice also that all of the variables a, b, c are determined based on known quantities (from the previous iteration). Of the two roots possible, one approach is, choose the one that gives the minimum angle between $\Delta\mathbf{u}^{(k-1)}$ and $\Delta\mathbf{u}^{(k)}$. Another way of saying it is to choose the root that maximizes the dot product between $\Delta\mathbf{u}^{(k-1)}$ and $\Delta\mathbf{u}^{(k)}$. That is (using (2.13))

$$\delta\lambda^{(k)} = \delta\hat{\lambda}^{(k)} \quad that \quad maximizes\{(\Delta\mathbf{u}^{(k-1)} + \delta\mathbf{u}^{*} + \delta\lambda^{(k)}\delta\bar{\mathbf{u}})^{T}\Delta\mathbf{u}^{(k-1)}\} \tag{2.19}$$

The above criteria for choosing the correct root is based on avoiding back tracking along the equilibrium path. An *additional* or alternative criteria, that is easy to implement, quite effective, and necessary when very sharp hairpin turns take place along the equilibrium path, is the same as above except that $\Delta\mathbf{u}^{(k-1)}$ is changed to $\Delta\mathbf{u}_{prev}$. The increment $\Delta\mathbf{u}_{prev}$ is the previously converged displacement increment along the equilibrium path. That is,

$$\delta\lambda^{(k)} = \delta\hat{\lambda}^{(k)} \quad that \quad maximizes\{(\Delta\mathbf{u}^{(k-1)} + \delta\mathbf{u}^{*} + \delta\lambda^{(k)}\delta\bar{\mathbf{u}})^{T}\Delta\mathbf{u}_{prev}\}. \tag{2.20}$$

If (2.18) does not give any real roots a possible solution is to use a smaller arc length size. This often allows difficult spots along the equilibrium curve to be overcome.

Finally, the load factor is updated by the typical iterative update expression

$$\lambda^{(k)} = \lambda^{(k-1)} + \delta\lambda^{(k)}. \tag{2.21}$$

Remarks

1. In (2.4) $\frac{\partial \mathbf{q}_e}{\partial \mathbf{u}}$ often is zero since the external force vector is constant. For the case of follower loads, which are dependent on $\mathbf{u}$, the partial derivative is not zero and must be included. In this paper follower loads are not addressed further.

2. A collection of increments ($\Delta\mathbf{u}$ and $\Delta\lambda$) are obtained during the nonlinear finite element analysis. These increments when pieced together trace the equilibrium path. Iterations within each increment are done as necessary to adjust each increment (using $\delta\mathbf{u}$ and $\delta\lambda$ corrections) so that static equilibrium is maintained before advancing to the next incremental step.

3. Choosing the appropriate arc length ℓ is problem dependent as is the choice for number of increments. Arc length and number of increments are chosen by trial and error after running the problem at hand and adjusting the parameters as necessary to obtain a sufficient equilibrium path. A sufficient equilibrium path depends on the needs of the analyst running the numerical simulation. A constant arc length is often chosen. However, algorithms to automate the choice of arc length during the analysis are possible [3]. Strategies are also possible to automate the number of increments used. For instance, increments are added until displacement at a chosen degree of freedom exceeds a user specified threshold.

4. Understanding the nuances of nonlinear finite element analysis takes experience. It is vital that the analyst is careful to review results and decide if valid results are being obtained. Furthermore, with experience the analyst learns how to overcome various problems, pitfalls, and errors when they arise.

5. Satisfying *both* (2.18) and (2.19) is quite effective. If these fail to provide good results a smaller arc length size very often fixes difficult spots along the equilibrium path. Alternatively, the work of Bathe and Dvorkin [1] advocates constant work increments. This method also leads to two roots, but both roots are guaranteed to be real.

2.2 The Predictor Solution

Consider now the situation in which the current residual is below some specified tolerance. Thus the current solution is *on* the equilibrium path. This situation occurs at the very beginning of the nonlinear analysis before any loads have even been applied and it also occurs at various locations after iterations have returned the solution to the equilibrium path. It is under this condition that it is necessary take an incremental step along the equilibrium path. This step (see Figure 1) is a *predictor solution* and generally (unless in a linear portion of the equilibrium path) results in a solution that is not on the equilibrium path and as a result the iterations that have already been discussed are necessary to return to equilibrium. One way to calculate a predictor solution uses (2.13) with the iteration variable k ignored, since the predictor step takes place prior to iterations. That is, since iterations have not yet commenced for the current solution path increment the term $\Delta\mathbf{u}^{(k-1)} = 0$ because no previous iteration exists and the term $\delta\mathbf{u}^* = 0$ because $\mathbf{g} \approx 0$, before the predictor step takes place. Hence, the predictor step, for the current increment n, is

$$\Delta\mathbf{u}^n = (\Delta\mathbf{u}^{(k-1)} = 0) + (\delta\mathbf{u}^* = 0) + \Delta\lambda^n \delta\bar{\mathbf{u}} = \Delta\lambda^n \delta\bar{\mathbf{u}}, \tag{2.22}$$

where $\Delta\lambda^n$ is used here instead of $\delta\lambda^n$ to indicate a change in the load parameter prior to iterations. Examining the result of (2.22) reveals that $\delta\bar{\mathbf{u}}$, per (2.11), is easy to calculate, but an appropriate $\Delta\lambda^n$ needs to be determined using the constraint (2.12). Therefore, using (2.22) in (2.12) yields

$$(\Delta\lambda^n\delta\bar{\mathbf{u}})^T(\Delta\lambda^n\delta\bar{\mathbf{u}}) = (\Delta\lambda^n)^2\delta\bar{\mathbf{u}}^T\delta\bar{\mathbf{u}} = \ell^2. \tag{2.23}$$

Solving for $\Delta\lambda^n$ results in

$$\Delta\lambda^n = \pm\sqrt{\frac{\ell^2}{\delta\bar{\mathbf{u}}^T\delta\bar{\mathbf{u}}}} = \pm\frac{\ell}{\sqrt{\delta\bar{\mathbf{u}}^T\delta\bar{\mathbf{u}}}}. \tag{2.24}$$

In (2.24), of the two possible signs for $\Delta\lambda^n$, the best sign choice seems to be

$$sign(\Delta\lambda^n) = sign(\Delta\mathbf{u}_{prev}^T\delta\bar{\mathbf{u}}) \tag{2.25}$$

where $\Delta\mathbf{u}_{prev}$ is the previously converged displacement vector increment along the equilibrium path. Hence, finally,

$$\Delta\lambda^n = sign(\Delta\mathbf{u}_{prev}^T\delta\bar{\mathbf{u}})\frac{\ell}{\sqrt{\delta\bar{\mathbf{u}}^T\delta\bar{\mathbf{u}}}}. \tag{2.26}$$

Remarks

1. For the predictor solution, to find $\delta\bar{\mathbf{u}}$ per (2.11), use $\mathbf{K}_T^{(0)}$ which is the most recent tangent stiffness matrix. For a predictor step the most recent tangent stiffness matrix is the one found during the last iteration of the last global displacement increment.

2. The predictor solution (step) is just the initial prediction for the next increment along the equilibrium path. If it does not result in a solution that is in static equilibrium then iterations are made to make corrections. The corrections improve the predictor solution until the norm of the residual is below a user specified tolerance.

3. On the very first increment of the path following analysis $\Delta\mathbf{u}_{prev}$ is created with all entries equal to a constant number such as 1.0 with the sign chosen so that the initial step heads in the intended direction. Other methods are possible, but this method is effective.

2.3 Convergence Criteria

A variety of convergence criteria are possible. However, starting with the residual (2.1) and the external force vector $\mathbf{q}_e$, a simple and effective convergence criteria is

$$\frac{(\mathbf{g}\bullet\mathbf{g})^{1/2}}{(\lambda\mathbf{q}_e\bullet\lambda\mathbf{q}_e)^{1/2}} < tolerance \tag{2.27}$$

Written in another way, but meaning exactly the same thing,

$$\frac{\|\mathbf{g}\|}{\|\lambda\mathbf{q}_e\|} < tolerance \tag{2.28}$$

where $\|\bullet\|$ symbolizes the L^2 norm.

The tolerance for convergence is set by the analyst. The value used depends on the problem at hand and practical considerations. Suggested values range from 10^-3 to 10^-9, yet at times values outside of this range are justified depending on the situation. For example, bigger tolerance values often give satisfactory results and for large problems reduce computation time.

3 The Equilibrium Path

The equilibrium path that the arc length method traces is really a path in a hyperspace of load vector versus displacement vector for the structure. These vectors have a number of components equal to the number of structure nodes times the number of degrees of freedom per node. Clearly, such an equilibrium path cannot be visualized. Yet, this is the equilibrium path that the equations follow. The picture of Figure 1 for instance attempts to figuratively illustrate this hyperspace equilibrium path, but only as a two-dimensional representation. In practice, during the analysis, the analyst records a single external force and the displacement at a single degree of freedom. This data is plotted and provides an equilibrium path for the chosen load and the chosen point on the structure. In this fashion a meaningful two-dimensional load displacement path is generated. If the displacement degree of freedom and external force pair are chosen carefully, useful characteristics and behaviors of the structure are revealed. Often a single load is applied to the structure at a point. This load and the point of application make an obvious pair of data to record for plotting after the analysis. Behavior such as limit points, snap through, and snap back can be handled when the arc length method is working properly.

4 Arc Length Control Algorithm

The following is an arc length control algorithm for performing a co-rotational truss analysis (see [8]). This is an implicit formulation which uses Newton-Raphson iterations at the global level to achieve equilibrium during each incremental arc length step. Material nonlinearities are not presently included in the algorithm. A program implementing this algorithm has been written in MATLAB and some representative results are provided. The algorithm proceeds as follows:

1. Define/initialize variables

   - $\mathbf{F}$ = the total vector of externally applied *relative* global nodal forces
   - $\mathbf{q}_e$ = the total vector of externally applied relative global nodal forces, modified to account for specified displacements (see creation of this vector in algorithm below)
   - $\mathbf{q}_i$ = the current internal force vector
   - $\mathbf{N}$ = the vector of truss axial forces, axial force in truss element i is N_i
   - $\mathbf{u}$ = the vector of global nodal displacements, initially $\mathbf{u} = \mathbf{0}$

- $\Delta\mathbf{u}_{prev}$ = vector to save previous increment of displacement
- $\mathbf{X}$ = the vector of nodal x coordinates in the undeformed configuration
- $\mathbf{Y}$ = the vector of nodal y coordinates in the undeformed configuration
- $\mathbf{L}$ = the vector of truss element lengths based on current $\mathbf{u}$, L for truss i is L_i, save the initial lengths in a vector $\mathbf{L}_o$
- $\mathbf{c}, \mathbf{s}$ = the vectors of cosines and sines for each truss element based on the current $\mathbf{u}$
- $\mathbf{K}_T$ = the assembled global tangent stiffness matrix
- $\mathbf{K}_s$ = the modified global tangent stiffness matrix to account for supports. Rows and columns associated with zero displacement dofs are set to zero and the diagonal position is set to 1.
- λ = load parameter initially set to zero
- $ninc$ = number of arc increments to use to construct the equilibrium path
- ℓ = the specified arc length (in hyperspace) to use during path following, in this algorithm a constant
- n = increment number (sometimes omitted from variables within the iteration phase), note that Δ is used to denote increment changes prior to (or outside of) iterations
- k = iteration variable, note that δ is used to denote change of a variable within iterations

2. **Start Loop** over arch length increments (for $n = 1$ to $ninc$).
   (a) Calculate global stiffness matrix $\mathbf{K}_T$ based on current values of $\mathbf{c}, \mathbf{s}, \mathbf{L}, \mathbf{L_o}$ and $\mathbf{N}$.
   (b) Create $\mathbf{q}_e$, the vector of external relative forces modified to account for specified displacements, using $\mathbf{F}$ and $\mathbf{K}_T$
   (c) Modify $\mathbf{K}_T$, accounting for specified supports, to get $\mathbf{K}_s$.
   (d) Calculate the predictor solution, $\Delta\mathbf{u}^n$, with the following steps:
      (i) $\delta\bar{\mathbf{u}} = \mathbf{K}_s^{-1}\mathbf{q}_e$
      (ii) $\Delta\lambda^n = sign(\Delta\mathbf{u}_{prev}^T\delta\bar{\mathbf{u}})\frac{\ell}{\sqrt{\delta\bar{\mathbf{u}}^T\delta\bar{\mathbf{u}}}}$
      (iii) $\lambda^n = \lambda^{n-1} + \Delta\lambda^n$
      (iv) $\Delta\mathbf{q}_e = \Delta\lambda^n\mathbf{q}_e$
      (v) $\Delta\mathbf{u}^n = \mathbf{K}_s^{-1}\Delta\mathbf{q}_e$
   (e) Update global nodal displacements, $\mathbf{u}^n = \mathbf{u}^{n-1} + \Delta\mathbf{u}^n$
   (f) Update $\mathbf{L}$, $\mathbf{c}$, and $\mathbf{s}$
   (g) Calculate the vector of new internal truss element axial forces $\mathbf{N}^n$. For truss element i the axial force is $N_i^n = (A_iE/L_{oi})\,u_{\ell i}$.
   (h) Construct the vector of internal global forces $\mathbf{q}_i^n$ based on $\mathbf{N}^n$.
   (i) Calculate the residual $\mathbf{R} = \mathbf{q}_i^n - \lambda^n\mathbf{q}_e$

(j) Set all dofs in $\mathbf{R}$ and $\mathbf{q}_e$, that correspond to specified displacement dofs, to zero. (Note that $\mathbf{q}_e$ and specified displacement dofs stay constant during iterations. This is the reason they are all set to zero in $\mathbf{R}$ and $\mathbf{q}_e$.)

(k) Calculate the *scaled* norm of the residual $R = \frac{\|\mathbf{R}\|}{\|\lambda \mathbf{q}_e\|}$

(l) Iterate for equilibrium if necessary. Set up iteration variables.

(i) Set iteration variable, $k = 0$

(ii) Set tolerance below which iterations cease, $tolerance = 10^{-6}$

(iii) Set the maximum number of iterations allowed, $maxiter = 100$

(iv) Set the variable used to store corrections to the predictor solution, $\delta\mathbf{u}^0 = \mathbf{0}$

(v) Set the variable use to store corrections to the load parameter, $\delta\lambda^0 = 0$

(m) **Start Iterations**, while $R > tolerance$ and $k < maxiter$

i. Update iteration counter $k = k + 1$

ii. Calculate the new global stiffness $\mathbf{K}_T$

iii. Modify the global stiffness to account for specified displacements which gives $\mathbf{K}_s$

iv. Calculate the corrections to $\mathbf{u}^n$ and λ^n. (Unlike (2.13) and (2.21), notice they are not updated until *after* iterations are completed (see [3] pages 154-156). Below sometimes superscript n is omitted to not interfere with k)

(A) Calculate $\delta\mathbf{u}^* = -\mathbf{K}_s^{-1}\mathbf{R}$

(B) Calculate $\delta\bar{\mathbf{u}} = \mathbf{K}_s^{-1}\mathbf{q}_e$

(C) Calculate $\Delta\mathbf{u}^{(k-1)} = \Delta\mathbf{u}^n + \delta\mathbf{u}^{(k-1)}$

(D) Set $a = \delta\bar{\mathbf{u}}^T\delta\bar{\mathbf{u}}$

(E) Set $b = 2(\Delta\mathbf{u}^{(k-1)} + \delta\mathbf{u}^*)^T\delta\bar{\mathbf{u}}$

(F) Set $c = (\Delta\mathbf{u}^{(k-1)} + \delta\mathbf{u}^*)^T(\Delta\mathbf{u}^{(k-1)} + \delta\mathbf{u}^*) - \ell^2$

(G) Calculate $\delta\hat{\lambda}_1 = \frac{-b+\sqrt{b^2-4ac}}{2a}$ and $\delta\hat{\lambda}_2 = \frac{-b-\sqrt{b^2-4ac}}{2a}$

(H) Calculate $dotprod1 = (\Delta\mathbf{u}^{(k-1)} + \delta\mathbf{u}^* + \delta\hat{\lambda}_1\delta\bar{\mathbf{u}})^T\Delta\mathbf{u}^{(k-1)}$ and $dotprod2 = (\Delta\mathbf{u}^{(k-1)} + \delta\mathbf{u}^* + \delta\hat{\lambda}_2\delta\bar{\mathbf{u}})^T\Delta\mathbf{u}^{(k-1)}$

(I) If $dotprod1 \geq dotprod2$ then $\delta\lambda^{(k)} = \delta\lambda^{(k-1)} + \delta\hat{\lambda}_1$ and $\delta\mathbf{u}^{(k)} = \delta\mathbf{u}^{(k-1)} + \delta\mathbf{u}^* + \delta\hat{\lambda}_1\delta\bar{\mathbf{u}}$

(J) Else $\delta\lambda^{(k)} = \delta\lambda^{(k-1)} + \delta\hat{\lambda}_2$ and $\delta\mathbf{u}^{(k)} = \delta\mathbf{u}^{(k-1)} + \delta\mathbf{u}^* + \delta\hat{\lambda}_2\delta\bar{\mathbf{u}}$

v. Update $\mathbf{L}$, $\mathbf{c}$, and $\mathbf{s}$ based on current $\mathbf{u}^n + \delta\mathbf{u}^{(k)}$

vi. Calculate the vector of new internal truss element axial forces $\mathbf{N}^{(k)}$. For truss element i the axial force is $(N^{(k)})_i = (A_iE/L_{oi})\, u_{\ell i}$.

vii. Construct the vector of internal global forces $\mathbf{q}_i^{(k)}$ based on $\mathbf{N}^{(k)}$.

viii. Calculate the residual $\mathbf{R} = \mathbf{q}_i^{(k)} - (\lambda^n + \delta\lambda^{(k)})\mathbf{q}_e$ and set all dofs in $\mathbf{R}$, that correspond to specified displacement dofs, to zero.

ix. Calculate $R = \frac{\|\mathbf{R}\|}{\|\lambda \mathbf{q}_e\|}$

(n) **End** of while loop iterations

3. Update variables to their final value for the current increment

   (i) $\mathbf{u}^n_{(final)} = \mathbf{u}^n + \delta\mathbf{u}^{(k)}$

   (ii) $\Delta\mathbf{u}_{prev} = \Delta\mathbf{u}^n + \delta\mathbf{u}^{(k)}$

   (iii) $\lambda^n_{(final)} = \lambda^n + \delta\lambda^{(k)}$

4. **End Loop** over load increments

Remarks

1. In *dotprod*1 and *dotprod*2, one could use the criteria of equation (2.20) instead or use *both* (2.19) and (2.20).

2. When programming the above algorithm for the first time it is often helpful to turn off iterations (set $maxiter = 0$) and make sure the algorithm is functioning reasonably well with iterations turned off. Subsequently, iterations are turned on and verified if working properly.

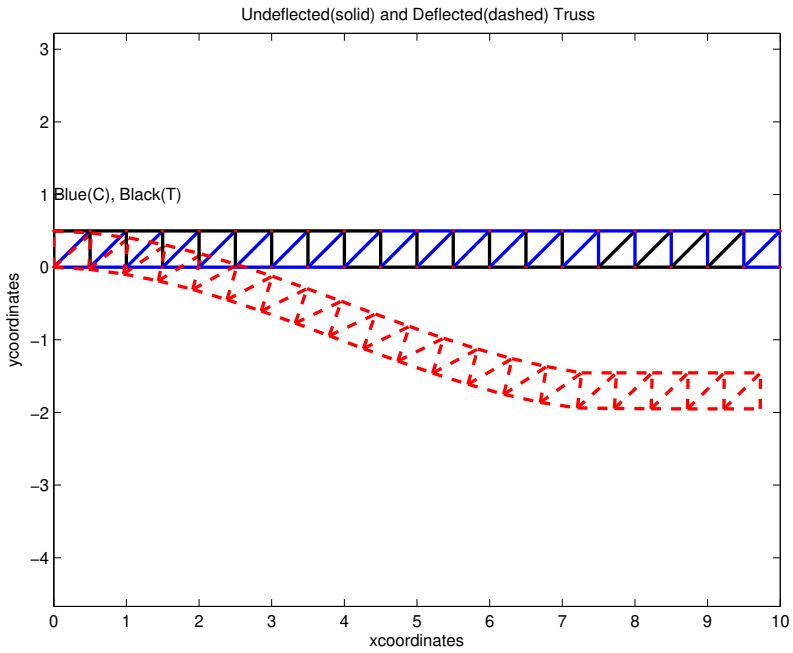
3. Benchmark problems are a key method to determine if the algorithm is working correctly. Experience with nonlinear finite element analysis and behavior is extremely important.

5 Example – Cantilevered Truss (arc length control)

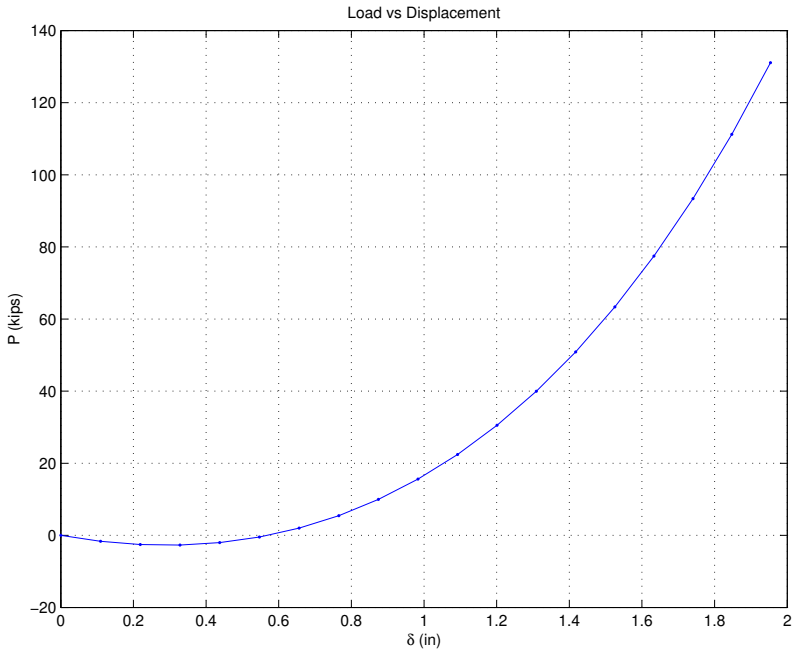
A cantilever truss is deformed by vertical specified displacements at the free end. All six of the top right end nodes are constrained to displace downwards equally. The initial and final configuration of the truss is shown in Figure 3a. The load displacement results are shown in the plot of Figure 3b. The analysis is completed in 18 equal arc length increments. The arc length for this problem is set at 0.5. For this example, all truss members have a cross-sectional area, $A = 0.1$ in^2, and modulus of elasticity, $E = 29000$ ksi. The truss is 10 inches long with 0.5 inch long vertical members and 0.5 inch long horizontal members. As a result of the above dimensions there are 42 nodes and 81 members. For supports, the top node at the left end is supported in the x and y directions, whereas the bottom node at the left only has x support provided. The tolerance used for equilibrium iterations is 10^{-8}.

6 Example – Truss arch with midspan point load (arc length control)

A parabolic arch truss is loaded by a point load at midspan. The initial and final configuration of the truss is shown in Figure 4a. The load displacement results are shown in the plot of Figure 4b. The phenomenon of snap through buckling is illustrated in the load displacement results. The arc length control algorithm successfully traces the load displacement path, whereas using a load control algorithm would not be able to correctly trace the whole path.

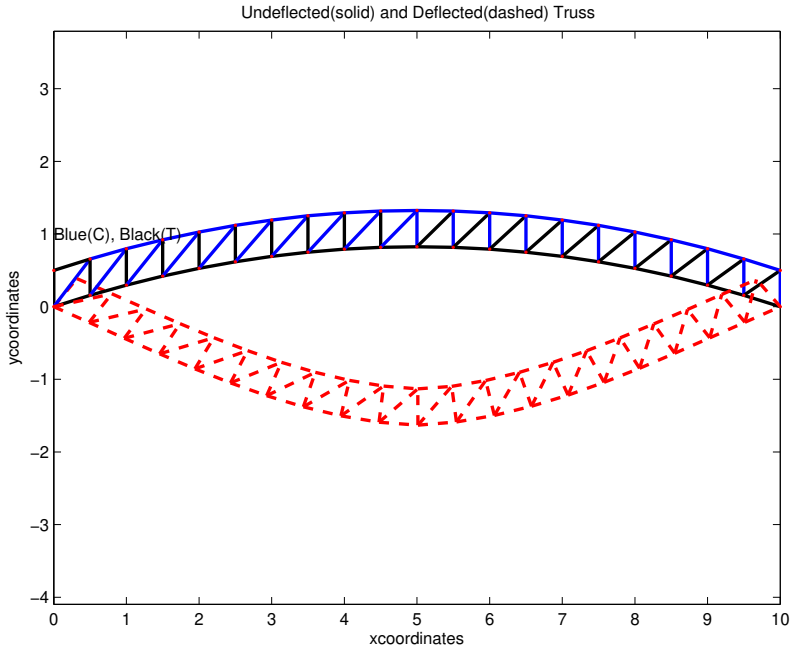


(a)

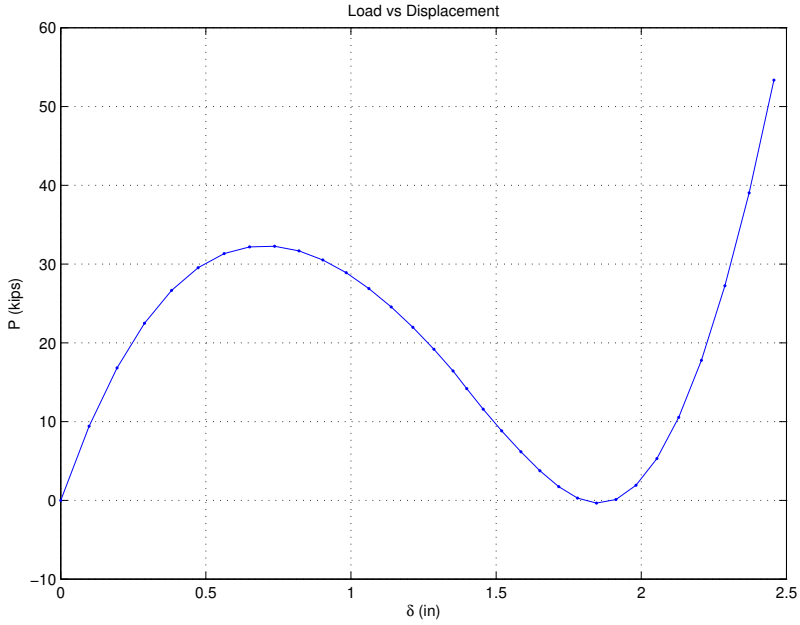


(b)

Figure 3: Cantileverd Truss - Arc Length Control: (a) Truss deflected shape with specified displacements at 6 of the free end top nodes, (b) Load versus free end vertical displacement (load and displacement are positive downwards).



(a)



(b)

Figure 4: Arch Truss With Point Load at Midspan - Arc Length Control: (a) Truss deflected shape, (b) Load versus midpoint displacement (load and displacement are positive downwards).

7 Example – Truss arch with multiple windings in equilibrium path (arc length control)

A parabolic arch truss is loaded at midspan. For some problems the structure geometry (different from previous example) and loading is such that multiple windings exist along the equilibrium path. Arc length control is capable of handling such complicated paths, which include both snap through and snap back. This is one of the reasons that arc length control is popular. Figure 5 illustrates an equilibrium path with multiple windings captured by the arc length formulation.

8 Conclusion

A derivation and explanation is given for the ingredients of arc length control. An algorithm and examples are given within the framework of a 2D corotational truss formulation. Although other path following techniques [2] [6] [7] are possible, arc length control is very effective in nonlinear problems including snap through and snap back.

References

[1] K. J. Bathe and E. N. Dvorkin. On the automatic solution of nonlinear finite element equations. *Computers and Structures*, 17:871–879, 1983.

[2] M. J. Clarke and G. J. Hancock. A study of incremental-iterative strategies for non-linear analyses. *International Journal for Numerical Methods in Engineering*, 29:1365–1391, 1990.

[3] M. A. Crisfield. *Non-linear Finite Element Analysis of Solids and Structures – Vol 1.* John Wiley & Sons Ltd., Chichester, England, 1991.

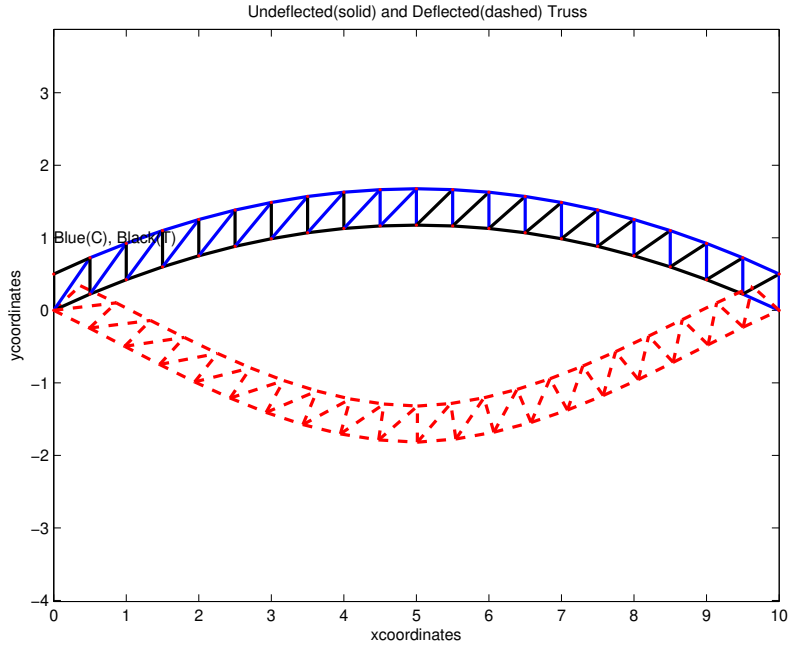
[4] M. A. Crisfield. *Non-linear Finite Element Analysis of Solids and Structures – Vol 2.* John Wiley & Sons Ltd., Chichester, England, 1997.

[5] E.A. de Souza Neto, D. Peric, and D.R.J. Owen. *Computational Methods for Plasticity: Theory and Applications.* Wiley, 2011.

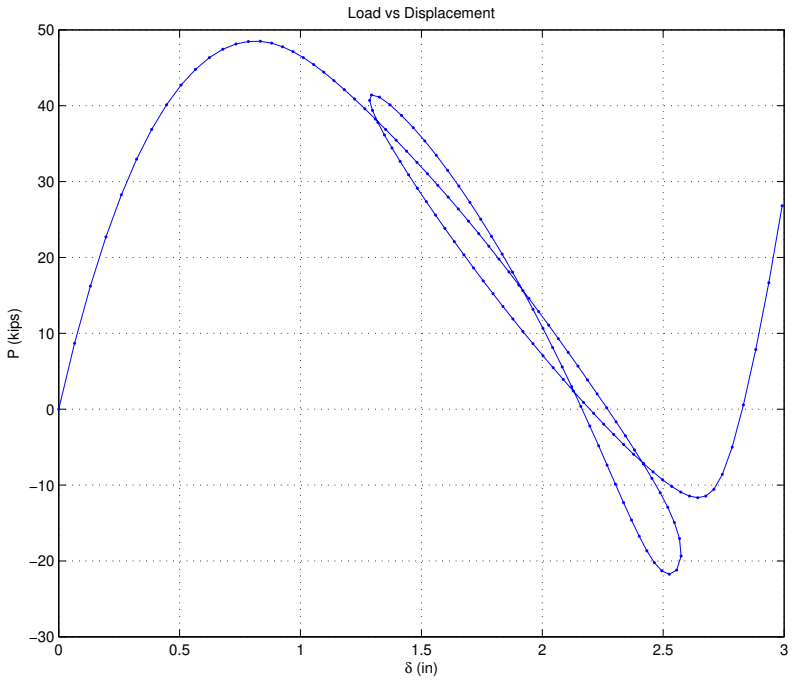
[6] W. McGuire, R. H. Gallagher, and R. D. Ziemian. *Matrix structural analysis.* Wiley, New York, 2nd edition, 2000.

[7] Y. B. Yang, L. J. Leu, and Judy P. Yang. Key considerations in tracing the postbuckling response of structures with multi winding loops. *Mechanics of Advanced Materials and Structures*, 14(3):175–189, March 2007.

[8] L. L. Yaw. 2D co-rotational truss formulation. *Online Web Document,* ***http://people.wallawalla.edu/~louie.yaw/Co-rotational_docs/2Dcorot_truss.pdf***, April 2009.



(a)



(b)

Figure 5: Arch Truss With Point Load at Midspan - Arc Length Control: (a) Truss deflected shape, (b) Load versus midpoint displacement illustrationg multiple windings.