CE394M: FEM solvers and errors

Krishna Kumar

University of Texas at Austin

krishnak@utexas.edu

February 11, 2019

Overview

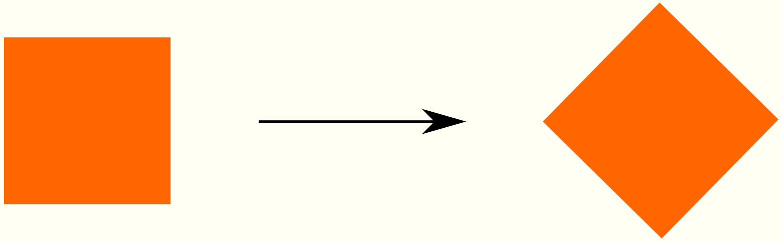
1. Solving non-linear problems
   - Iterative solvers
   - Tangent stiffness
   - Newton Raphson
2. Arc length method

Linear and non-linear problems

- **Linear problems**
  - The response can only be approximated as linear if its deformations/motions are small.
  - In linear analyses, the response to individual load cases can be scaled and added to the results from other linear analyses, which is the principle of superposition.
- **Non-linear problems**
  - Superposition is invalid.
  - The solution is an incremental/iterative process.
  - An iteration is the solution of a system of equations linearised about the current state of the nonlinear physical problem.

Non-linearity in geotechnical engineering

- **Material non-linearity**
  - plasticity
- **Contact**
  - discontinuous source of non-linearity
- **Large deformations and motions of a geotechnical structures**
  - rotation, rigid body motion
  - often ignored, still in the research area.



No strains if linear strain-displacement relations are used.

Linear solvers

To solve a system of linear equations of the form $\mathbf{Ka} = \mathbf{b}$, there are two families of methods of solvers that can be used: direct and iterative.

- Direct solvers solve a system of linear equations in a predefined number of steps.
- Methods are based on Gauss elimination, with the most common method being LU decomposition.
- The time required to solve a linear system increases with the number of computer operations performed.
- For a direct linear solver applied to a dense system of size $n \times n$: CPU time $= Cn^3$.
- If the number of degrees of freedom in a finite element simulation is doubled, the time required to solve the system will increase by a factor of 8!

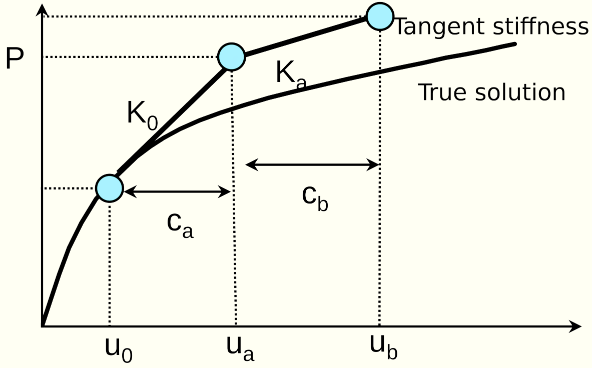
Iterative solvers

The system of equations that we derived from the finite element approximation of the BVP:

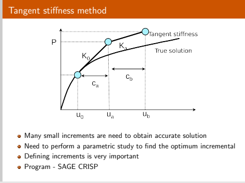
$$F_{\mathrm{int}}(u) - F_{\mathrm{ext}} = 0$$

At this point we remind ourselves that in the case of finite deformations, both F_{int} and F_{ext} are in general very nonlinear terms.
The integration has to be carried over the current volume and surface (of the finite element under consideration) that may in general depend on u in a highly non-linear fashion.

Tangent stiffness method



- Many small increments are need to obtain accurate solution
- Need to perform a parametric study to find the optimum incremental
- Defining increments is very important
- Program - SAGE CRISP



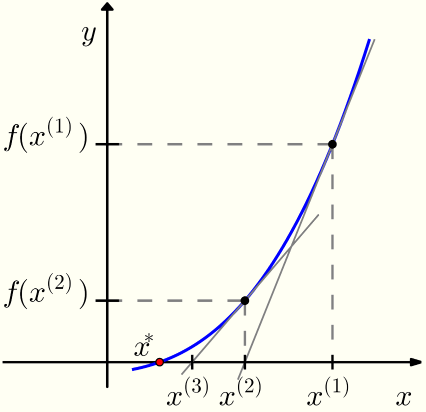
Tangent stiffness method

- Many small increments are need to obtain accurate solution
- Need to perform a parametric study to find the optimum incremental
- Defining increments is very important
- Program - SAGE CRISP

In the incremental solution we divide the load into increments $\Delta P = \lambda P$, where λ is also known as a load factor and apply a repeated solution of $\Delta u = K^{-1} \Delta P$.

Basically we divide the load into substeps, and treat each as linear - but that is usually not accurate enough and inefficient.

Newton Raphson method



$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

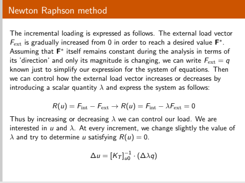
Newton Raphson method

The incremental loading is expressed as follows. The external load vector F_{ext} is gradually increased from 0 in order to reach a desired value $\mathbf{F}^*$. Assuming that $\mathbf{F}^*$ itself remains constant during the analysis in terms of its 'direction' and only its magnitude is changing, we can write $F_{\text{ext}} = q$ known just to simplify our expression for the system of equations. Then we can control how the external load vector increases or decreases by introducing a scalar quantity λ and express the system as follows:

$$R(u) = F_{\text{int}} - F_{\text{ext}} \to R(u) = F_{\text{int}} - \lambda F_{\text{ext}} = 0$$

Thus by increasing or decreasing λ we can control our load. We are interested in u and λ. At every increment, we change slightly the value of λ and try to determine u satisfying $R(u) = 0$.

$$\Delta u = [K_T]_{u0}^{-1} \cdot (\Delta \lambda q)$$



Newton Raphson method

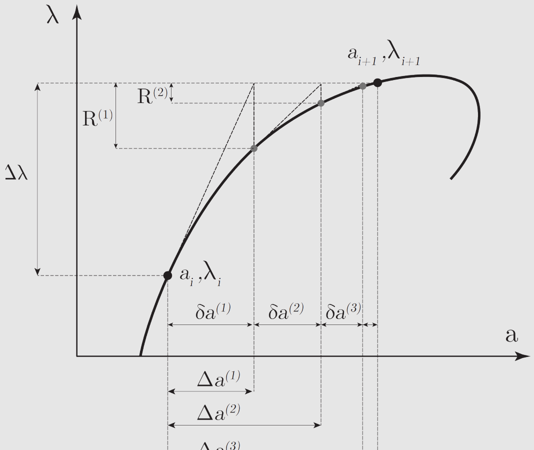
The incremental loading is expressed as follows. The external load vector F_{ext} is gradually increased from 0 in order to reach a desired value $\mathbf{F}^*$. Assuming that $\mathbf{F}^*$ itself remains constant during the analysis in terms of its 'direction' and only its magnitude is changing, we can write $F_{\text{ext}} = q$ known just to simplify our expression for the system of equations. Then we can control how the external load vector increases or decreases by introducing a scalar quantity λ and express the system as follows:

$$R(u) = F_{\text{int}} - F_{\text{ext}} \to R(u) = F_{\text{int}} - \lambda F_{\text{ext}} = 0$$

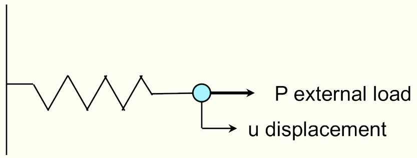
Thus by increasing or decreasing λ we can control our load. We are interested in u and λ. At every increment, we change slightly the value of λ and try to determine u satisfying $R(u) = 0$.

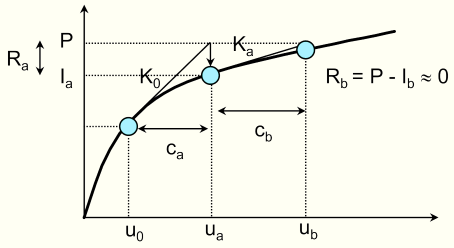
$$\Delta u = [K_T]_{u0}^{-1} \cdot (\Delta \lambda q)$$

where $[K^T] = [\partial F(u)/\partial u]$ is the "Jacobian" matrix of the system of equations and is commonly referred to as the Stiffness Matrix.

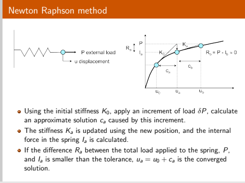


Newton Raphson method

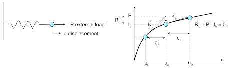




- Using the initial stiffness K_0, apply an increment of load δP, calculate an approximate solution c_a caused by this increment.
- The stiffness K_a is updated using the new position, and the internal force in the spring I_a is calculated.
- If the difference R_a between the total load applied to the spring, P, and I_a is smaller than the tolerance, $u_a = u_0 + c_a$ is the converged solution.



Newton Raphson method



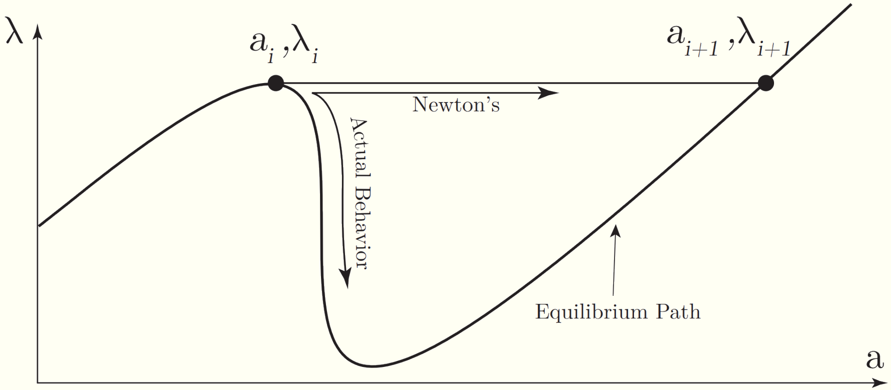
- Using the initial stiffness K_0, apply an increment of load δP, calculate an approximate solution c_a caused by this increment.
- The stiffness K_a is updated using the new position, and the internal force in the spring I_a is calculated.
- If the difference R_a between the total load applied to the spring, P, and I_a is smaller than the tolerance, $u_a = u_0 + c_a$ is the converged solution.

- If R_a is not small, a new displacement correction c_b is calculated by solving $c_b = R_a/K_a$
- The new displacement u_b is updated, and the internal force I_b in the updated configuration is calculated.
- The new force residual R_b is obtained. If $R_b < \mathit{tolerance}$, the solution is converged. If not, continue the iteration.

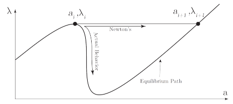
Newton Raphson method: Tolerance and Convergence

- Program - ABAQUS
- Newton-Raphson is the most standard method to solve nonlinear problems in FE.
- A large error in the initial estimate can contribute to non-convergence of the algorithm.
- **Tolerance**
  - must be small enough to ensure that the approximate solution is close to the exact mathematical solution.
  - must be large enough so that reasonable number of iterations are performed.
- **Quadratic convergence**
  - If the tangent stiffness is calculated correctly, R should reduce quadratically from one iteration to the next.

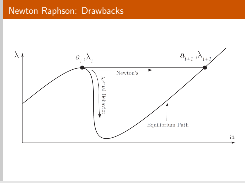
Newton Raphson: Drawbacks



Newton Raphson: Drawbacks

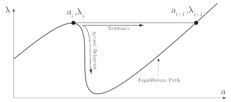


Newton's method fails to accurately follow the 'equilibrium' path once the tangent stiffness reaches zero. That happens due to the formulation of Newton's method, and in particular that it restricts the parameter λ to change monotonically every increment 3 . The definition of a limit point then (saddle points excluded), suggests that in order to remain on the equilibrium path you need to change your loading pattern depending on whether the limit point is a local maximum or maximum in the $u-\lambda$ space.



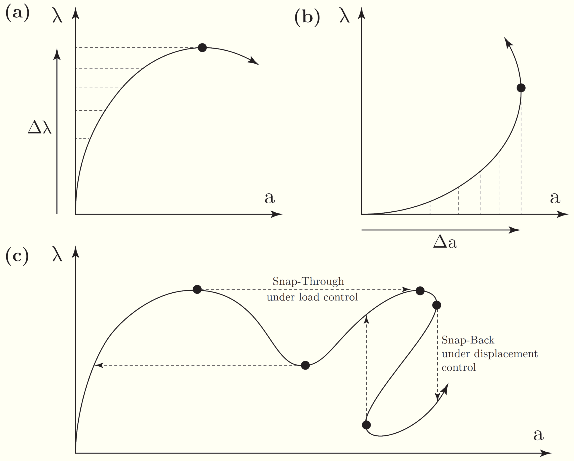
In terms of mechanical systems then, this method is able to solve any non-linear system of equations very efficiently but only up to the critical point (if any). In the case shown in figure, Newton's method fails in load–control. Now in many cases, one way to circumvent problems like these is to use displacement control, where you can continuously increase the displacements u and still remain on the equilibrium curve. In general however, apart from Snap–Through behaviors under load control, a problem may exhibit Snap– Back behaviors under displacement control or even both. The main problem is, that in most actual applications, the structural response, and therefore the equilibrium path, for the structure under consideration is unknown, and therefore one does not know what type of behavior to expect.

Newton Raphson: Drawbacks



As a general rule, if the problem under consideration requires information after its critical/failure points then Newton's method is not a good choice. Buckling analysis and non-linear materials that exhibit work softening are just two example problems that cannot be solved using Newton's method. Furthermore, very often, strong nonlinearities that arise in finite deformation problems may eventually lead to such behaviors and thus it is necessary to introduce a numerical technique to solve such problem with strongly nonlinear behaviors.

Newton Raphson: Drawbacks


(a)
λ
Δλ
a
(b)
λ
a
Δa
(c)
λ
Snap-Through under load control
Snap-Back under displacement control
a

(a) A system that is unstable under load control (Snap–Through instability), (b) A system that is unstable under displacement control (Snap–Back Instability), (c) A system that is unstable under both displacement and load control

Arc length method

System of the non-linear (in general) equations to solve:

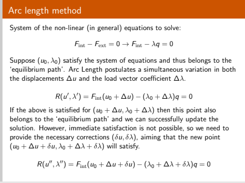
$$F_{\text{int}} - F_{\text{ext}} = 0 \rightarrow F_{\text{int}} - \lambda q = 0$$

Suppose (u_0, λ_0) satisfy the system of equations and thus belongs to the 'equilibrium path'. Arc Length postulates a simultaneous variation in both the displacements Δu and the load vector coefficient $\Delta\lambda$.

$$R(u', \lambda') = F_{\text{int}}(u_0 + \Delta u) - (\lambda_0 + \Delta\lambda)q = 0$$

If the above is satisfied for $(u_0 + \Delta u, \lambda_0 + \Delta\lambda)$ then this point also belongs to the 'equilibrium path' and we can successfully update the solution. However, immediate satisfaction is not possible, so we need to provide the necessary corrections $(\delta u, \delta\lambda)$, aiming that the new point $(u_0 + \Delta u + \delta u, \lambda_0 + \Delta\lambda + \delta\lambda)$ will satisfy.

$$R(u'', \lambda'') = F_{\text{int}}(u_0 + \Delta u + \delta u) - (\lambda_0 + \Delta\lambda + \delta\lambda)q = 0$$


Arc length method

System of the non-linear (in general) equations to solve:

$$F_{\text{int}} - F_{\text{ext}} = 0 \rightarrow F_{\text{int}} - \lambda q = 0$$

Suppose (u_0, λ_0) satisfy the system of equations and thus belongs to the 'equilibrium path'. Arc Length postulates a simultaneous variation in both the displacements Δu and the load vector coefficient $\Delta\lambda$.

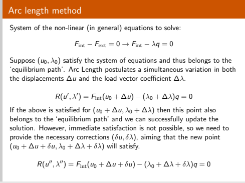
$$R(u', \lambda') = F_{\text{int}}(u_0 + \Delta u) - (\lambda_0 + \Delta\lambda)q = 0$$

If the above is satisfied for $(u_0 + \Delta u, \lambda_0 + \Delta\lambda)$ then this point also belongs to the 'equilibrium path' and we can successfully update the solution. However, immediate satisfaction is not possible, so we need to provide the necessary corrections $(\delta u, \delta\lambda)$, aiming that the new point $(u_0 + \Delta u + \delta u, \lambda_0 + \Delta\lambda + \delta\lambda)$ will satisfy.

$$R(u'', \lambda'') = F_{\text{int}}(u_0 + \Delta u + \delta u) - (\lambda_0 + \Delta\lambda + \delta\lambda)q = 0$$

Arc length is a very efficient method in solving non-linear systems of equations when the problem under consideration exhibits one or more critical points. In terms of a simple mechanical loading-unloading problem, a critical point could be interpreted as the point at which the loaded body cannot support an increase of the external forces and an instability occurs.

Unlike the Newton-Method, the Arc Length method postulates a simultaneous variation in both the displacements Δu and the load vector coefficient $\Delta lambda$. The main difference is that both Δu and $\Delta\lambda$ are unknowns in contrast to Newton's method where $\Delta lambda$ was given and we had to iteratively solve for Δu.



Arc length method

System of the non-linear (in general) equations to solve:

$$F_{\text{int}} - F_{\text{ext}} = 0 \rightarrow F_{\text{int}} - \lambda q = 0$$

Suppose (u_0, λ_0) satisfy the system of equations and thus belongs to the 'equilibrium path'. Arc Length postulates a simultaneous variation in both the displacements Δu and the load vector coefficient $\Delta\lambda$.

$$R(u', \lambda') = F_{\text{int}}(u_0 + \Delta u) - (\lambda_0 + \Delta\lambda)q = 0$$

If the above is satisfied for $(u_0 + \Delta u, \lambda_0 + \Delta\lambda)$ then this point also belongs to the 'equilibrium path' and we can successfully update the solution. However, immediate satisfaction is not possible, so we need to provide the necessary corrections $(\delta u, \delta\lambda)$, aiming that the new point $(u_0 + \Delta u + \delta u, \lambda_0 + \Delta\lambda + \delta\lambda)$ will satisfy.

$$R(u'', \lambda'') = F_{\text{int}}(u_0 + \Delta u + \delta u) - (\lambda_0 + \Delta\lambda + \delta\lambda)q = 0$$

The system of equations takes the form:

$$\left[K^T\right]_{u_0+\Delta u} \cdot \delta u - \delta\lambda q = -\left[F_{\text{int}}(u_0 + \Delta u) - (\lambda_0 + \Delta\lambda)q\right] = -R(u', \lambda')$$

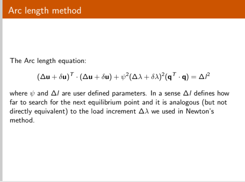
δu and $\delta lambda$ are the unknowns for whom we need to solve. If the u vector however, has dimensions $N \times 1$ then we have a total of N equations that we need to solve for $N + 1$ unknowns (N unknowns δu and 1 unknown $\delta\lambda$). Equations then are not sufficient to determine $\delta u, \delta\lambda$. The supplementary equation that completes the system is called the Arc Length Equation.

Arc length method

The Arc length equation:

$$(\Delta \mathbf{u} + \delta \mathbf{u})^T \cdot (\Delta \mathbf{u} + \delta \mathbf{u}) + \psi^2 (\Delta\lambda + \delta\lambda)^2 (\mathbf{q}^T \cdot \mathbf{q}) = \Delta l^2$$

where ψ and Δl are user defined parameters. In a sense Δl defines how far to search for the next equilibrium point and it is analogous (but not directly equivalent) to the load increment $\Delta\lambda$ we used in Newton's method.


Arc length method

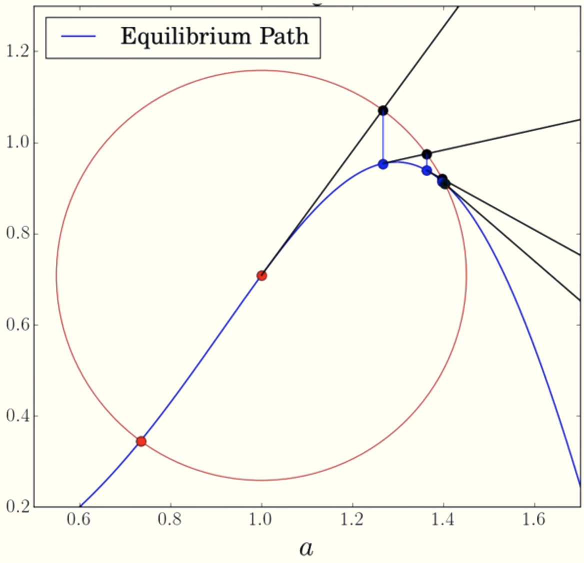
The Arc length equation:

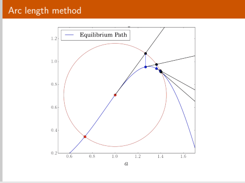
$$(\Delta \mathbf{u} + \delta \mathbf{u})^T \cdot (\Delta \mathbf{u} + \delta \mathbf{u}) + \psi^2 (\Delta\lambda + \delta\lambda)^2 (\mathbf{q}^T \cdot \mathbf{q}) = \Delta l^2$$

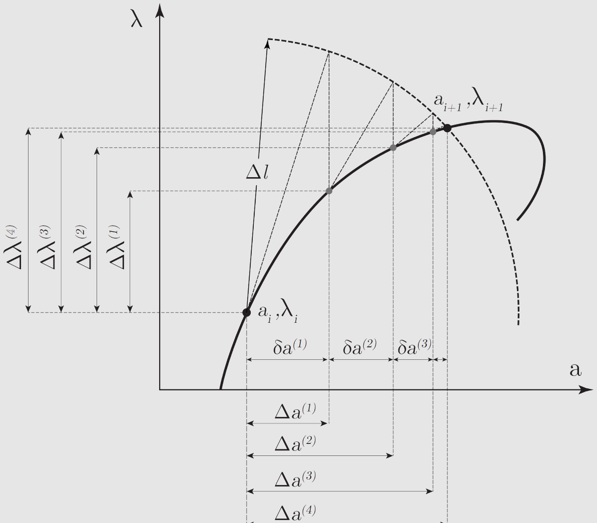
where ψ and Δl are user defined parameters. In a sense Δl defines how far to search for the next equilibrium point and it is analogous (but not directly equivalent) to the load increment $\Delta\lambda$ we used in Newton's method.

The system of equations is solved for $\delta u, \delta\lambda$ and updates the previous corrections $\Delta u, \Delta\lambda$ to be $\Delta u' = \Delta u + \delta u$ and $\Delta\lambda' = \Delta\lambda + \delta\lambda$ respectively. The method continues to provide such incremental corrections $\delta u, \delta\lambda$ until convergence is achieved. When $\psi = 1$, the method is also called the **Spherical Arc-Length Method** because the points $\Delta u', \Delta\lambda'$ belong to a circle with radius Δl. In its most general form for arbitrary ψ can be geometrically interpreted as a hyper-ellipse in the multidimensional displacement-load space $(u - \lambda)$.

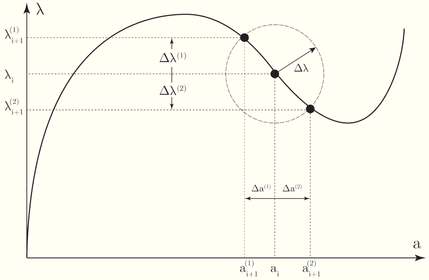
Arc length method

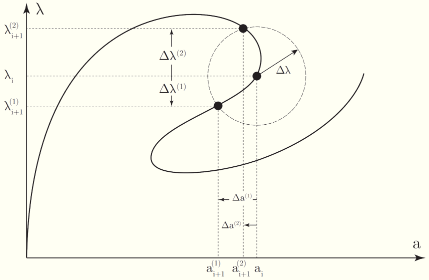


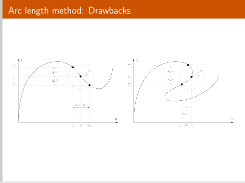

Arc length method



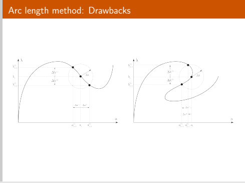
Arc length method: Drawbacks






Arc length method: Drawbacks

Crisfield's [1983] implementation of Arc-length however, leads to one of the method's most important drawbacks. The quadratic equation would in general lead to two distinct solutions for $\delta\lambda$ which will in turn lead to two distinct solutions for δu. Thus, every iteration, the solver determined two sets of solutions, namely $(\delta u_1, \delta\lambda_1)$ and $(\delta u_1, \delta\lambda_1)$. This is no surprise since a circle (or a hyperellipse) would always intersect a curve in two points if its center is located on the curve.



The issue that arises then, is to develop a robust algorithm that would be able to accurately determine the correct set of $(\delta u, \delta\lambda)$ to update the solution. In general, we would like to choose the set, so that the solution 'evolves forwards'. This term 'forward evolution' is commonly used in the context of this method since choosing the wrong set would make the solution move back towards a previously converged point, and not in the desired (forward) direction. It is interesting to note, that an effective solution to this problem that works for all applications is yet to be found and as a result, many times programs like ABAQUS fail to converge to the correct solution or fail to 'evolve forwards'.