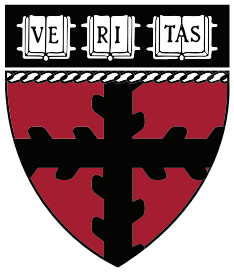



# Nonlinear Analysis of Structures

The Arc Length Method: Formulation, Implementation and Applications

Nikolaos Vasios

PhD Student, Materials Science & Mechanical Engineering

Email Address: vasios@g.harvard.edu

*Fall 2015*

# Contents

# List of Figures

# 1 Introduction

The vast majority of continuum mechanics problems can be formulated as Boundary Value Problems. In such problems, we prescribe conditions in the boundary of our continuum body and we are interested in determining the value as well as the distribution of some quantities inside the body. In structural mechanics problems these quantities are usually displacements, velocities, stresses and strains but in general any other quantity[1] can be assumed to be unknown within the body, as long as the problem is formulated in a way that takes the unknowns into account. The traditional formulation of the Boundary Value Problem (BVP) consists of three distinct equation "families", namely equilibrium, kinematics and constitutive equations. In the following section, we will introduce the form of these equations using the finite deformation theory, in the case of a continuum body in which the primary unknown is the displacement field.

## 1.1 Strong and Weak Formulation of the BVP

Let us consider the spatial configuration of a general continuum body, which initially (at $t = 0$) occupies volume $V_0$ with a mass density $\rho_0$ defining its reference state. The body is then subjected to a combination of body forces $\mathbf{b}$ per unit mass and traction forces $\hat{\mathbf{t}}$ per unit surface. The portion of the body's outer surface subjected to traction forces is called $S_t$. Moreover another portion of the body's surface $S_u$[2] is subjected to known displacements $\hat{\mathbf{u}}$. As a result, after a time period $\Delta t$ the body deforms and occupies volume $V$ with a mass density $\rho$, surrounded by surface $S$. The equilibrium equations[3] are expressed in terms of the Cauchy stress tensor as:

$$\frac{\partial \sigma_{ij}}{\partial x_j} + \rho b_i = 0 \tag{1.1}$$

The kinematic relationships are also introduced as:

$$D_{ij} = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) \tag{1.2}$$

defining the deformation rate tensor $\mathbf{D}$ that corresponds to the velocity field $\boldsymbol{v}$. The deformable material is governed by a general constitutive law of the form:

$$\overset{\nabla}{\boldsymbol{\sigma}} = \overset{\nabla}{\boldsymbol{\sigma}}(\mathbf{L}) \tag{1.3}$$

[1] Other quantities could be the temperature distribution within the body, a given concentration of a phase or a substance, the distribution of electric charge etc.

[2] Note that $S_u \cap S_t = \emptyset$ and $S_u \cup S_t = S$

[3] Assuming that dynamic phenomena are not present and therefore the acceleration field $\mathbf{a}$ vanishes

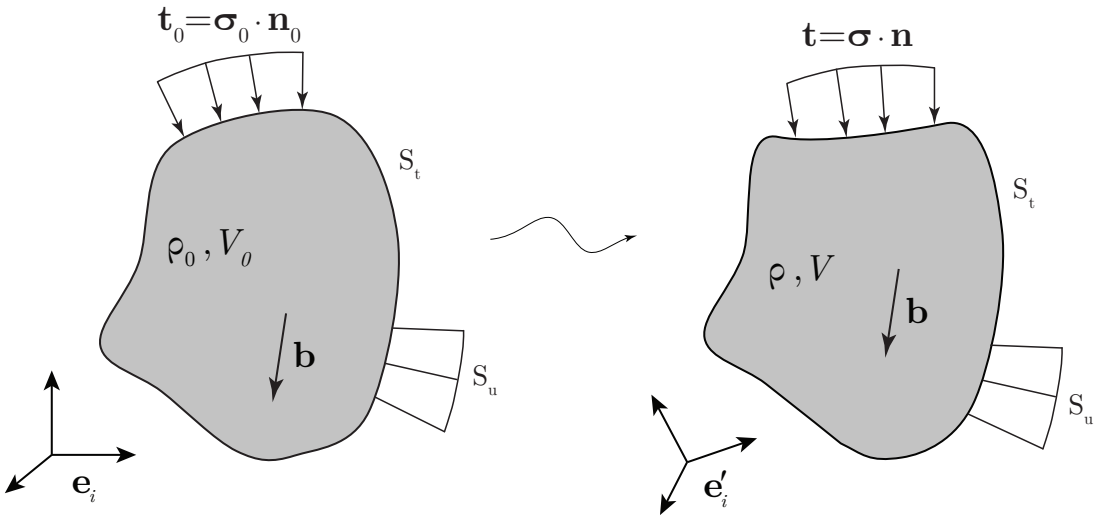



*Figure 1.1: A blob of an arbitrary shape subjected to a stress field and some displacement boundary conditions in its reference and deformed state*

where $\overset{\nabla}{\boldsymbol{\sigma}}$ represents the Jaumann rate of $\boldsymbol{\sigma}$ commonly used to describe the constitutive behavior of solids undergoing finite deformations and rotations and $\mathbf{L}$ is the velocity gradient tensor defined. We also introduce the boundary conditions on $S$:

$$\mathbf{u} = \hat{\mathbf{u}} = \text{known} \quad \text{on } S_u \tag{1.4}$$

$$\hat{\mathbf{t}} = \boldsymbol{\sigma}\cdot\mathbf{n} = \text{known} \quad \text{on } S_t \tag{1.5}$$

Equations (1.1–1.5) constitute the **Strong Formulation** of the Boundary Value problem. The problem consists of 15 unknowns, namely the displacement field ($\mathbf{u}$), the velicity gradient tensor $\mathbf{L}$ and the stress tensor $\boldsymbol{\sigma}$. An alternative formulation can also be introduced as follows. We begin by replacing the three equilibrium equations in (1.1) by a unique scalar equation[4] over the entire body. This equation is obtained by multiplying the differential equations in (1.1) by a virtual (arbitrary but continuous and differentiable) velocity field $\delta\boldsymbol{v}^*$ and then integrating over the entire volume of the continuum body. Hence:

$$\int_{V(t)} \Big[\boldsymbol{\nabla}\cdot\boldsymbol{\sigma} + \rho\mathbf{b}\Big]\cdot\delta\boldsymbol{v}^*\, dV = 0 \tag{1.6}$$

In view of the chain rule we can write:

$$\boldsymbol{\nabla}\cdot(\boldsymbol{\sigma}\cdot\delta\boldsymbol{v}^*) = (\boldsymbol{\nabla}\cdot\boldsymbol{\sigma})\cdot\delta\boldsymbol{v}^* + \boldsymbol{\sigma} : (\boldsymbol{\nabla}\,\delta\boldsymbol{v}^*)$$

and making use of the divergence theorem[5] we can also write:

$$\int_{V(t)} \Big[\boldsymbol{\nabla}\cdot\boldsymbol{\sigma}\Big]\cdot\delta\boldsymbol{v}^*\, dV = \int_{V} \Big[\boldsymbol{\nabla}\cdot(\boldsymbol{\sigma}\cdot\delta\boldsymbol{v}^*) - \boldsymbol{\sigma} : (\boldsymbol{\nabla}\,\delta\boldsymbol{v}^*)\Big]\, dV$$

[4] Note that this replacement does not violate the generality
[5] Also known as Gauss's theorem

$$= \int_{S(t)} \mathbf{n} \cdot \boldsymbol{\sigma} \cdot \delta\boldsymbol{\upsilon}^* \, dS - \int_{V(t)} \boldsymbol{\sigma} : (\boldsymbol{\nabla} \, \delta\boldsymbol{\upsilon}^*) \, dV$$

$$= \int_{S(t)} \hat{\mathbf{t}} \cdot \delta\boldsymbol{\upsilon}^* \, dS - \int_{V} \boldsymbol{\sigma} : \delta\mathbf{L}^* \, dV \tag{1.7}$$

where $\delta\mathbf{L}^*$ is the velocity gradient tensor corresponding to the virtual velocity field $\delta\boldsymbol{\upsilon}^*$. Let us also decompose $\delta\mathbf{L}^*$ into its symmetric $\delta\mathbf{D}^*$ and antisymmetric part $\delta\mathbf{W}^*$ and take advantage of the symmetry of $\boldsymbol{\sigma}$ to write[6]:

$$\boldsymbol{\sigma} : \delta\mathbf{L}^* = \boldsymbol{\sigma} : (\delta\mathbf{D}^* + \delta\mathbf{W}^*) = \boldsymbol{\sigma} : \delta\mathbf{D}^* + \boldsymbol{\sigma} : \delta\mathbf{W}^* = \boldsymbol{\sigma} : \delta\mathbf{D}^* \tag{1.8}$$

Now combining equations (1.6),(1.7) and (1.8) we can express the alternative formulation of the BVP as:

$$\boxed{G(\Delta\mathbf{u}) = \int_{V(t)} \boldsymbol{\sigma} : \delta\mathbf{D}^* \, dV - \int_{S(t)} \hat{\mathbf{t}} \cdot \delta\boldsymbol{\upsilon}^* \, dS - \int_{V(t)} \rho\mathbf{b} \cdot \delta\boldsymbol{\upsilon}^* \, dV = 0} \tag{1.9}$$

The above formulation is also called the **Weak Formulation** of the BVP and provides the basis for the **Finite Element approximation** introduced in the following section.

## 1.2 The Finite Element Approximation

In a finite element setting, the solution is developed incrementally from $t_n$ to $t_{n+1}$ with the primary unknown of the problem being the displacement increment $\Delta\mathbf{u}(\mathbf{x})$. Once $\Delta\mathbf{u}$ is determined, the total displacement field at the end of the current increment at $t = t_{n+1}$ is calculated as:

$$\mathbf{u}_{n+1}(\mathbf{x}) = \mathbf{u}_n(\mathbf{x}) + \Delta\mathbf{u}(\mathbf{x}) \tag{1.10}$$

and consequently the current position of any material point within the continuum body can be directly updated as:

$$\mathbf{x}_{n+1} = \mathbf{X} + \mathbf{u}_{n+1}(\mathbf{x}) \tag{1.11}$$

Discretizing the continuum body into finite elements, we express the unknown displacement increment $\Delta\mathbf{u}$ as a function interpolation[7] within each element as:

$$\underset{3\times1}{\{\Delta u(\mathbf{x})\}} = \underset{3\times n}{[N(\mathbf{x})]} \, \underset{n\times1}{\left\{\Delta u^N\right\}} \tag{1.12}$$

where $[N(\mathbf{x})]$ is the interpolation matrix that consists of user–defined 'shape' functions, whereas $\left\{\Delta u^N\right\}$ is the vector of nodal unknowns. Now recall that the virtual velocity field, $\delta\boldsymbol{\upsilon}^*$, must be compatible with all kinematic constraints. The interpolation introduced in (1.12) however, constrains the displacement to have a certain spatial variation and therefore $\delta\boldsymbol{\upsilon}^*$ must also be defined using the same function interpolation [Abaqus Theory Manual]. Hence,

$$\underset{3\times1}{\{\delta\upsilon^*\}} = \underset{3\times n}{[N(\mathbf{x})]} \, \underset{n\times1}{\left\{\Delta\upsilon^{*N}\right\}} \tag{1.13}$$

[6] Recall that the double dot product of a symmetric and an antisymmetric tensor equals to zero

[7] This approximation was first introduced by Galerkin

The virtual strain rate tensor $\delta\mathbf{D}^*$ is also expressed in array form as:

$$\underset{6\times1}{\{\delta D^*\}} = \underset{6\times n}{[B(\mathbf{x})]}\ \underset{n\times1}{\left\{\Delta\upsilon^{*N}\right\}} \tag{1.14}$$

where $[B(\mathbf{x})]$ is the matrix containing the spatial derivatives of the shape functions $N^\alpha(\mathbf{x})$ in the sense $B_{\alpha\beta} = 1/2(N_{\alpha,\beta} + N_{\beta,\alpha})$. Finally we introduce the array notations of the rest quantities in the Weak form (1.9) as:

$$\boldsymbol{\sigma} \longrightarrow \underset{6\times1}{\{\sigma\}} \qquad \hat{\mathbf{t}} \longrightarrow \underset{3\times1}{\{t\}} \qquad \mathbf{b} \longrightarrow \underset{3\times1}{\{b\}}$$

Substituting each term in (1.9) we derive:

$$\lfloor\Delta\upsilon_e^{*N}\rfloor \underset{e}{\mathrm{A}} \left[ \int\limits_{V_{n+1}^e} \left( [B]_{n+1}^T \{\sigma\}_{n+1} - [N]_{n+1}^T \{b\}_{n+1} \right) dV^e - \int\limits_{S_{n+1}^e} [N]_{n+1}^T \{t\}_{n+1}\, dS^e \right] = 0$$

where $V^e$,$S^e$ are the volume and surface respectively of the finite element under consideration and $\mathrm{A}_e$ represents the Assembly operation. Note however, that since the above expression must hold $\forall\ \lfloor\Delta\upsilon_e^{*N}\rfloor$, we can derive:

$$\underset{e}{\mathrm{A}} \left[ \int\limits_{V_{n+1}^e} [B]_{n+1}^T \{\sigma\}_{n+1}\, dV^e - \int\limits_{S_{n+1}^e} [N]_{n+1}^T \{t\}_{n+1}\, dS^e - \int\limits_{V_{n+1}^e} [N]_{n+1}^T \{b\}_{n+1}\, dV^e \right] = 0$$

Now let's define the external load vector comprising of the traction and body forces as:

$$\{F\}_{n+1}^{\text{ext}} = \underset{e}{\mathrm{A}} \left[ \int\limits_{S_{n+1}^e} [N]_{n+1}^T \{t\}_{n+1}\, dS^e + \int\limits_{V_{n+1}^e} [N]_{n+1}^T \{b\}_{n+1}\, dV^e \right]$$

At this point recall that $\{\sigma\}_{n+1}$ is a non–linear (in general) function of the unknowns $\{\Delta u^N\}$, so that we write:

$$\boxed{\left\{R(\Delta u^N)\right\}_{n+1} \equiv \underset{e}{\mathrm{A}} \int\limits_{V_{n+1}^e} [B]_{n+1}^T \{\sigma\}_{n+1}\, dV^e - \{F\}_{n+1}^{\text{ext}} = \{0\}} \tag{1.15}$$

where $\{R(\Delta u^N)\}_{n+1}$ is the residual vector expressing the difference between the internal $\sigma_{n+1}$ and external $t_{n+1}, b_{n+1}$ forces. The solution of the 'weak' problem is the displacement field $\{\Delta u^N\}$ that satisfies the system of equations in (1.15), or equivalently, the displacement field that at $t = t_{n+1}$ equates the applied loads $\{F\}_{n+1}^{\text{ext}}$ to the internal forces $\{\sigma\}_{n+1}$ yielding a residual smaller than a given tolerance.

The system of equations in (1.15) is a highly nonlinear system with respect to the nodal displacements for two main reasons. First, the displacements are nonlinearly involved in the stresses through the constitutive law in the sense $\boldsymbol{\sigma} = \boldsymbol{\sigma}(\mathbf{D}(\mathbf{u}))$. Secondly, and most importantly in this discussion here, in finite deformation theory where we account for deviations between the reference and the currency geometry, geometric nonlinearities are introduced to the problem through the limits of integration since in general $V^e = V^e(\mathbf{u})$.

# 2 Solving the Finite Element Equations

Recall the system of equations that we derived from the finite element approximation of the BVP (1.15). We restate it below in a much more compact form as:

$$\mathbf{F}^{int}(\mathbf{u}) - \mathbf{F}^{ext} = \mathbf{0} \tag{2.1}$$

where,

$$\mathbf{F}^{int}(\mathbf{u}) = \underset{e}{\mathrm{A}} \int_{V^e} [B]^T \{\sigma(\mathbf{D}(\mathbf{u}))\}\, dV$$

and

$$\mathbf{F}^{ext} = \underset{e}{\mathrm{A}} \left[ \int_{S^e} [N]^T \{t\}\, dS^e + \int_{V^e} [N]^T \{b\}\, dV \right]$$

At this point we remind ourselves that in the case of finite deformations, both $\mathbf{F}^{int}$ and $\mathbf{F}^{ext}$ 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 $\mathbf{u}$[1] in a highly non-linear fashion, $V^e = V^e(\mathbf{u})$, $S^e = S^e(\mathbf{u})$. This suggests that the only way to solve (1.15) is numerically. Two of the most common numerical methods used to solve these equations[2] are **Newton's Method** and the **Arc Length Method**.

## 2.1 Newton's Method

In Newton's method, the incremental loading is expressed as follows. The **external** load vector $\mathbf{F}^{ext}$ is gradually increased from $\mathbf{0}$ 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 $\mathbf{F}^{ext} = \mathbf{q}$ =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 $\lambda$ and express the system as follows

$$\mathbf{R}(\mathbf{u}) = \mathbf{F}^{int}(\mathbf{u}) - \mathbf{F}^{ext} \Rightarrow \boxed{\mathbf{R}(\mathbf{u}) = \mathbf{F}^{int}(\mathbf{u}) - \lambda \mathbf{F}^{ext} = 0} \tag{2.2}$$

Thus by increasing or decreasing $\lambda$ we can control our **load incrementation**. We introduced the term $\mathbf{R}(\mathbf{u})$ because we are interested in the general case were the system of equations is not in equilibrium and

[1] Therefore the limits of integration are also functions of the unknown displacement field

[2] And are implemented in most commercial finite element software packages

therefore the difference expresses the residual vector, which we then use to find corrections to our solution. In this system of equations, we are interested in $\mathbf{u}$ and $\lambda$. At every increment, we change slightly the value of $\lambda$ and try to determine $\mathbf{u}$ so that the system in (2.2) is satisfied. Suppose that the last converged solution is $\{\mathbf{u}_0, \lambda_0\}$. The load increment is initiated by postulating:

$$\lambda' = \lambda_0 + \Delta\lambda$$

where $\Delta\lambda$ is a known predefined incrementation parameter. This variation $\Delta\lambda$ immediately violates equation (2.2) and thus we need to update the displacements $\mathbf{u}_0$ by

$$\mathbf{u}' = \mathbf{u}_0 + \Delta\mathbf{u}$$

so that:

$$\mathbf{R}(\mathbf{u}') = \mathbf{R}(\mathbf{u}_0 + \Delta\mathbf{u}) = \mathbf{0} \Rightarrow \mathbf{F}^{int}(\mathbf{u}_0 + \Delta\mathbf{u}) - (\lambda_0 + \Delta\lambda)\mathbf{q} = \mathbf{0} \tag{2.3}$$

But, $\mathbf{F}^{int}(\mathbf{u}_0 + \Delta\mathbf{u})$ can be expressed in terms of $\mathbf{F}^{int}(\mathbf{u}_0)$ by a Taylor series expansion as:

$$\mathbf{F}^{int}(\mathbf{u}_0 + \Delta\mathbf{u}) = \mathbf{F}^{int}(\mathbf{u}_0) + \left[\frac{\partial \mathbf{F}(\mathbf{u})}{\partial \mathbf{u}}\right]_{\mathbf{u}_0} \cdot \Delta\mathbf{u} = \mathbf{F}^{int}(\mathbf{u}_0) + [K_T]_{\mathbf{u}_0} \cdot \Delta\mathbf{u} \tag{2.4}$$

where $[K_T] = [\partial\mathbf{F}(\mathbf{u})/\partial\mathbf{u}]$ is the "Jacobian" matrix of the system of equations and is commonly referred to as the *Stiffness Matrix*. Now combining equations (2.3) and (2.4), we can solve for $\Delta\mathbf{u}$ as:

$$\mathbf{F}^{int}(\mathbf{u}_0) + [K_T]_{\mathbf{u}_0} \cdot \Delta\mathbf{u} - (\lambda_0 + \Delta\lambda)\mathbf{q} = \mathbf{0} \Rightarrow \overbrace{\mathbf{F}^{int}(\mathbf{u}_0) - \lambda_0\mathbf{q}}^{\mathbf{0}} + [K_T]_{\mathbf{u}_0} \cdot \Delta\mathbf{u} - \Delta\lambda\mathbf{q} = \mathbf{0} \Rightarrow$$

$$\boxed{\Delta\mathbf{u} = [K_T]_{\mathbf{u}_0}^{-1} \cdot (\Delta\lambda\mathbf{q})} \tag{2.5}$$

From equation (2.5) we can calculate the displacement correction $\Delta\mathbf{u}$. In general however, even though we postulated that $\Delta\mathbf{u}$ would be such that (2.3) is satisfied, the linear approximation in Taylor epxansion prevents the immediate achievement (linear-response) of equilibrium. Thus, if we evaluate the system (2.3) at the new point $(\mathbf{u}', \lambda')$ we will obtain a non-zero residual vector $\hat{\mathbf{R}}(\mathbf{u}')$. Using this residual vector, we can calculate a new displacement correction $\delta\mathbf{u}$ as follows:

$$\mathbf{R}(\mathbf{u}_0 + \Delta\mathbf{u} + \delta\mathbf{u}) = \mathbf{0} \Rightarrow \mathbf{F}^{int}(\mathbf{u}_0 + \Delta\mathbf{u}) + [K_T]_{\mathbf{u}'} \cdot \delta\mathbf{u} - (\lambda + \Delta\lambda)\mathbf{q} = \mathbf{0} \Rightarrow$$

$$[K_T]_{\mathbf{u}'} \cdot \delta\mathbf{u} = -(\mathbf{F}^{int}(\mathbf{u}_0 + \Delta\mathbf{u}) - (\lambda + \Delta\lambda)\mathbf{q}) \Rightarrow [K_T]_{\mathbf{u}'} \cdot \delta\mathbf{u} = -\hat{\mathbf{R}}(\mathbf{u}') \Rightarrow$$

$$\boxed{\delta\mathbf{u} = -[K_T]_{\mathbf{u}'}^{-1} \cdot \hat{\mathbf{R}}(\mathbf{u}')} \tag{2.6}$$

Hence, a new displacement correction is determined and evaluating the system (2.3) at the new points $(\mathbf{u}' + \delta\mathbf{u}, \lambda')$ would in general result to a new and smaller residual vector $\hat{\mathbf{R}}(\mathbf{u}'')$. We continue to provide displacement corrections until a norm (usually Euclidean) of the residual vector is less than the specified tolerance. A schematic representation of the Newton-Raphson method is shown in Figure 2.1.

The quadratic convergence rate of Newton method guarantees convergence within few iterations and is the main reason for its universal implementation in almost all commercial FEA software. Fast convergence makes Newton's method ideal when solving large systems of non-linear equations, where each iteration

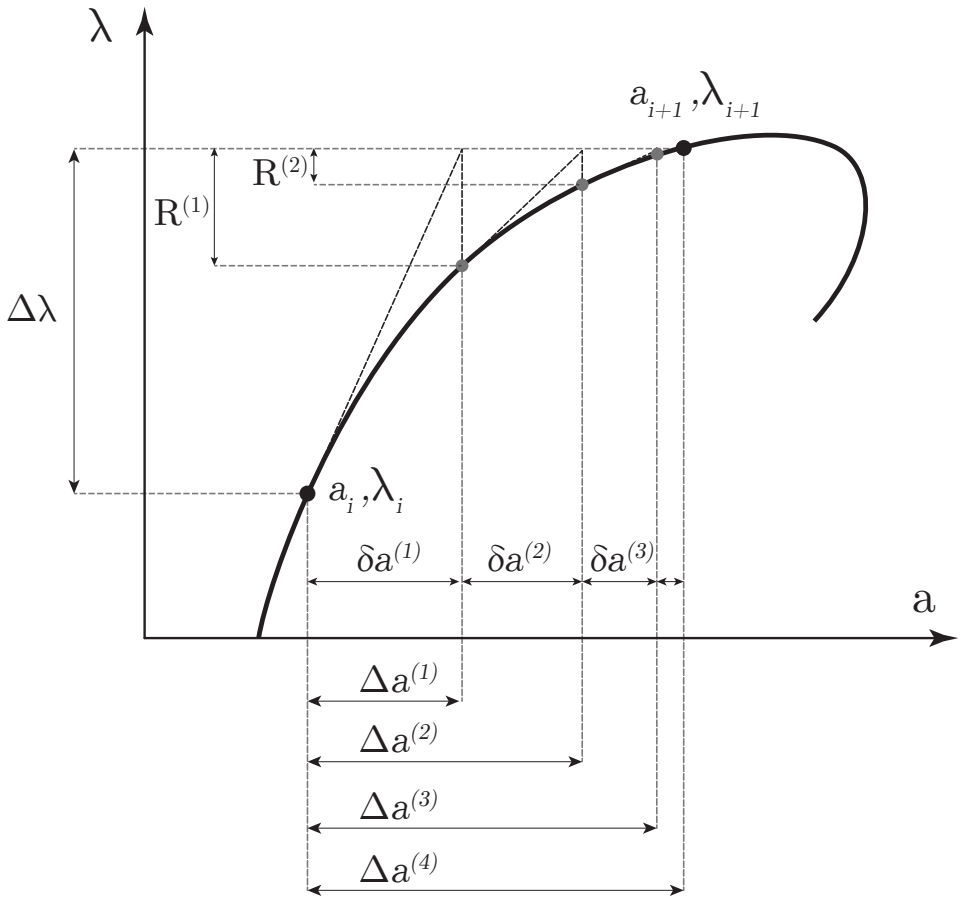



*Figure 2.1: A schematic representation of the Newton iterations. a denotes a normalized displacement whereas λ the load incrementation parameter. The increment is defined by Δλ and the solution is the point of intersection between the equilibrium path and the horizontal line λ + Δλ*

'costs' in terms of computational time. Convergence aside however, Newton's method is also associated with a major drawback. The 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 $\lambda$ 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 $\mathbf{u} - \lambda$ space. This problem can be better conceptualized in Figure 2.2.

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 2.2,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 $\mathbf{u}$ 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 as shown in Figure 2.3. The main problem is, that in most actual applications, the structural response, and therefore the equilibrium path, for the structure under

[3]In most structural mechanics problems for instance, we continuously increase the external loads

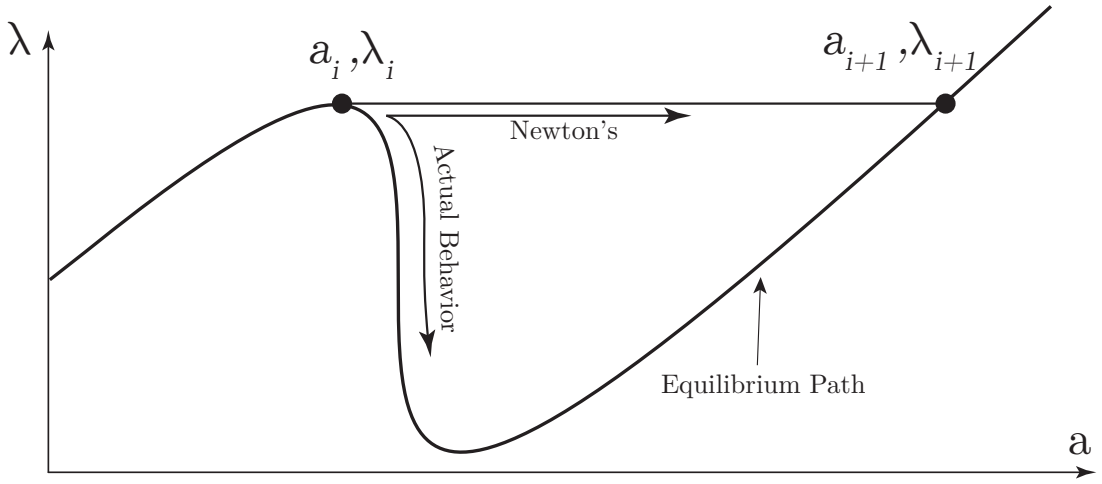



*Figure 2.2: The Newton's method cannot accurately predict the solution after a limit point is reached*

consideration is unknown, and therefore one does not know what type of behavior to expect. 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.

## 2.2 The Arc Length Method

The Arc-Length method [Riks E., 1979] 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.

Recall once again the system of the non-linear (in general) equations that we are interested in solving:

$$\mathbf{F}^{int}(\mathbf{u}) - \mathbf{F}^{ext} = \mathbf{0} \Rightarrow \mathbf{F}^{int}(\mathbf{u}) - \lambda\mathbf{q} = \mathbf{0} \tag{2.7}$$

Suppose that the point $(\mathbf{u}_0, \lambda_0)$ is such to satisfy the system of equations and thus belongs to the 'equilibrium path' that we are trying to identify. Unlike the Newton-Method, the Arc Length method postulates a simultaneous variation in both the displacements $\Delta\mathbf{u}$ and the load vector coefficient $\Delta\lambda$. The main difference is that both $\Delta\mathbf{u}$ and $\Delta\lambda$ are unknowns in contrast to Newton's method where $\Delta\lambda$ was given and we had to iteratively solve for $\Delta\mathbf{u}$. We can write:

$$\mathbf{R}(\mathbf{u}', \lambda') = \mathbf{F}^{int}(\mathbf{u}_0 + \Delta\mathbf{u}) - (\lambda_0 + \Delta\lambda)\mathbf{q} = \mathbf{0} \tag{2.8}$$

If the above equation is satisfied for $(\mathbf{u}_0 + \Delta\mathbf{u}, \lambda_0 + \Delta\lambda)$ then this point also belongs to the 'equilibrium path' and we can successfully update the solution. In most cases however, immediate satisfaction of (2.8)

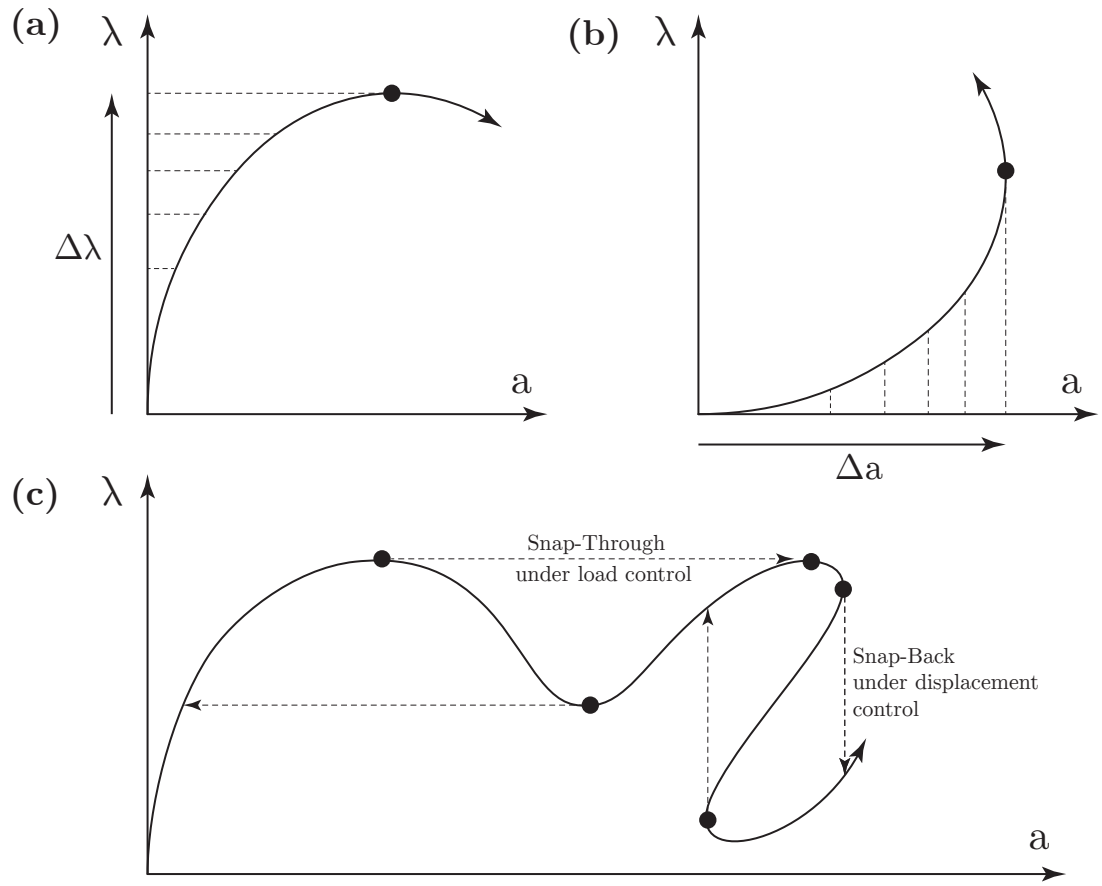



*Figure 2.3: (**a**) A system that is unstable under load control (Snap–Through instability), (**b**) A system that is unstable under displacement control (Snap–Back Instability), (**c**) A system that is unstable under both displacement and load control*

is not achievable. As a result we need to provide the necessary corrections $(\delta\mathbf{u}, \delta\lambda)$, aiming that the new point $(\mathbf{u}_0 + \Delta\mathbf{u} + \delta\mathbf{u}, \lambda_0 + \Delta\lambda + \delta\lambda)$ will satisfy (2.8). Hence,

$$\mathbf{R}(\mathbf{u}'', \lambda'') = \mathbf{F}^{int}(\mathbf{u}_0 + \Delta\mathbf{u} + \delta\mathbf{u}) - (\lambda_0 + \Delta\lambda + \delta\lambda)\mathbf{q} = \mathbf{0} \tag{2.9}$$

Using a Taylor series expansion and retaining only the linear terms, we can rewrite the last equation in the following form:

$$\mathbf{F}^{int}(\mathbf{u}_0 + \Delta\mathbf{u}) + \left[\frac{\partial \mathbf{F}^{int}(\mathbf{u})}{\partial \mathbf{u}}\right]_{\mathbf{u}_0+\Delta\mathbf{u}} \cdot \delta\mathbf{u} - (\lambda_0 + \Delta\lambda + \delta\lambda)\mathbf{q} = \mathbf{0} \tag{2.10}$$

From now on, we will represent the "Jacobian matrix" of the system defined by $[\partial \mathbf{F}^{int}/\partial \mathbf{u}]$ by the quantity $[K_T]$. Implementing the new notation, the system of equations takes the form:

$$\boxed{[K_T]_{\mathbf{u}_0+\Delta\mathbf{u}} \cdot \delta\mathbf{u} - \delta\lambda\mathbf{q} = -[\mathbf{F}^{int}(\mathbf{u}_0 + \Delta\mathbf{u}) - (\lambda_0 + \Delta\lambda)\mathbf{q}] = -\mathbf{R}(\mathbf{u}', \lambda')} \tag{2.11}$$

Recall that $\delta\mathbf{u}$ and $\delta\lambda$ are the unknowns for whom we need to solve. If the $\mathbf{u}$ vector however, has dimensions Nx1 then we have a total of N equations that we need to solve for N+1 unknowns (N unknowns $\delta\mathbf{u}$ and 1 unknown $\delta\lambda$). Equations (2.11) then are not sufficient to determine $\delta\mathbf{u}, \delta\lambda$. The supplementary equation

that completes the system is called the **Arc Length Equation** and has the following form:

$$\boxed{(\Delta\mathbf{u}+\delta\mathbf{u})^T\cdot(\Delta\mathbf{u}+\delta\mathbf{u})+\psi^2(\Delta\lambda+\delta\lambda)^2(\mathbf{q}^T\cdot\mathbf{q})=\Delta l^2} \tag{2.12}$$

where $\psi$ and $\Delta l$ are user defined parameters. In a sense $\Delta l$ 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. Collecting up equations (2.11) and (2.12) we can write the system of equations we need to solve in a much more compact form as:

$$\begin{bmatrix} [K_T] & -\mathbf{q} \\ 2\Delta\mathbf{u}^T & 2\psi^2\Delta\lambda(\mathbf{q}^T\cdot\mathbf{q}) \end{bmatrix}\cdot\begin{bmatrix}\delta\mathbf{u}\\ \delta\lambda\end{bmatrix}=-\begin{bmatrix}\mathbf{R}\\ A\end{bmatrix} \tag{2.13}$$

where,

$$\mathbf{R}=\mathbf{F}^{int}(\mathbf{u}_0+\Delta\mathbf{u})-(\lambda_0+\Delta\lambda)\mathbf{q}$$

$$A=-\left(\Delta\mathbf{u}^T\cdot\Delta\mathbf{u}+\psi^2\Delta\lambda^2(\mathbf{q}^T\cdot\mathbf{q})-\Delta l^2\right)$$

The system of equations in (2.13) is solved for $\delta\mathbf{u},\delta\lambda$ and updates the previous corrections $\Delta\mathbf{u},\Delta\lambda$ to be $\Delta\mathbf{u}'=\Delta\mathbf{u}+\delta\mathbf{u}$ and $\Delta\lambda'=\Delta\lambda+\delta\lambda$ respectively. The method continues to provide such incremental corrections $\delta\mathbf{u},\delta\lambda$ until convergence is achieved in (2.9). When $\psi=1$, the method is also called the ***Spherical Arc-Length Method*** because (2.12) suggests that the points $\Delta\mathbf{u}'$,$\Delta\lambda'$ belong to a circle with radius $\Delta l$. In its most general form for arbitrary $\psi$, equation (2.12) can be geometrically interpreted as a hyper-ellipse in the multidimensional displacement-load space $(\mathbf{u}-\lambda)$. The user decides which value should be assigned to the 'radius' and the next converged point is then obtained as the point of intersection between the equilibrium path and that sphere. This iterative process to determine the next point of intersection is shown below in the 2D $(a-\lambda)$ space where the sphere essentially degenerates into a circle.

A slide–show video that illustrates how the Arc Length method's iterations really work to solve a nonlinear system of equations can be found [here]. The video was made using Python to create png pictures and then Matlab to create a video out of these pictures. All the codes developed and used in the context of this project can be found in the last chapter of this document.

This method is widely proven to cope quite well in problems with a snapping behavior (limit points) and is implemented in most commercial finite element software (i.e. ABAQUS[4]). This way of formulating the Arc–Length method however and in particular the system of equations outlined in (2.13) is not the most efficient one and as a result, most commercial software use a different approach to this method. The reason for this is that expression (2.13) essentially introduces a completely new system of equations to be solved simultaneously for $\delta\mathbf{u}$ and $\delta\lambda$. As a result, the techniques commonly used by finite element software such as ABAQUS to solve the system of equations in all other cases (static analysis with Newton's method, Dynamic Analysis etc) cannot be used in this case where the system of equations is different than the one we introduced in our description of Newton's method ((2.6)). Despite the capabilities of this method in cases where Newton's method fails, this particular formulation obstructed the immediate implementation of the method in such software because sacrificing the solver's efficiency and at the same time having to modify all

[4]Simulia ABAQUS refers to this method as the Riks method, naming it after E.Riks who first introduced the method in [Riks E., 1979]. The particular implementation of the method however is different than the one presented in this section

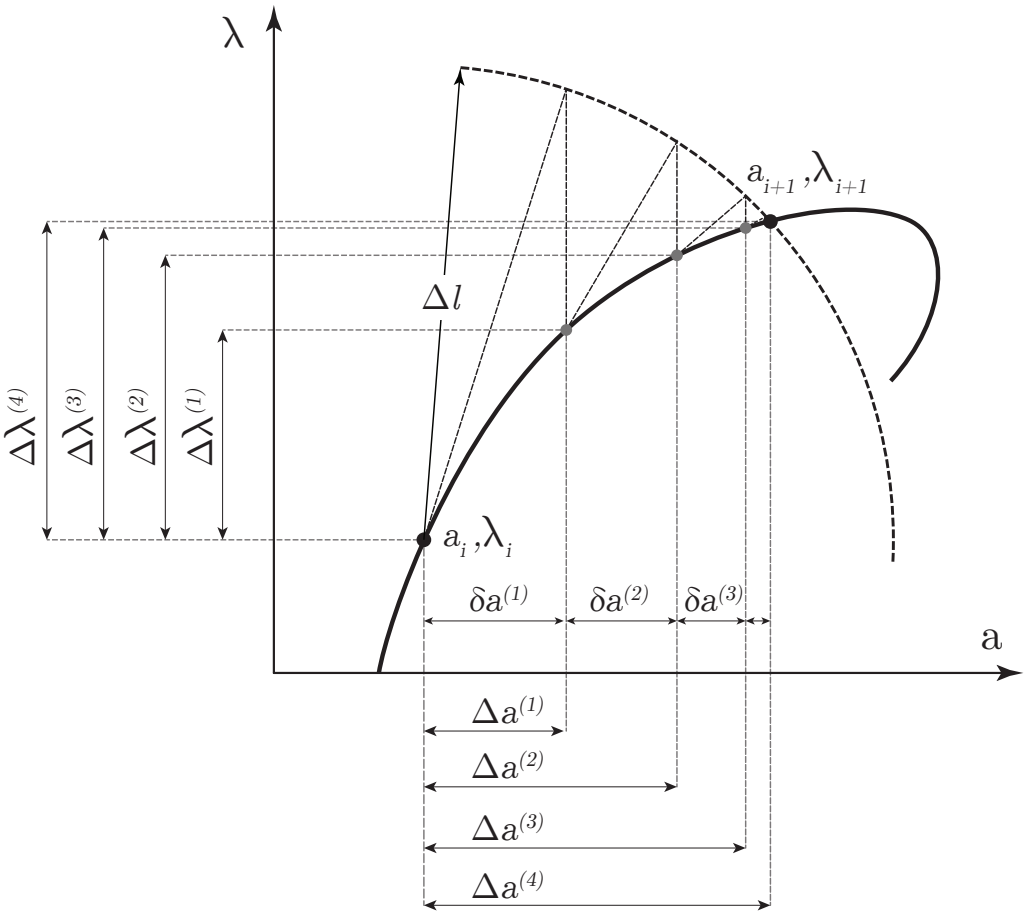



*Figure 2.4: A schematic representation of the Arc-Length method iterations. a denotes a normalized displacement whereas λ the load incrementation parameter. The increment is defined by the radius of the circle Δl and the next point is the point of intersection between the path and the circle.*

the convergence criteria wasn't an option. It was necessary that the implementation of the method would be based on a different formulation that would ideally include no modifications to the system of equations to be solved.

### 2.2.i Crisfield's Formulation

Four years later, in 1983, Crisfield published a paper [Crisfield M.A., 1983] where he presented an alternative formulation for the Arc Length method which could be readily implemented in any commerical finite element software that was able to solve nonlinear problems using Newton's method. In this section we present this formulation, found in [Crisfield M.A., 1983]. Recall the expression (2.11) that need to be solved for the unknowns $\delta\mathbf{u}$ and $\delta\lambda$. Crisfield expressed the equation as:

$$\delta\mathbf{u} = -[K_T]^{-1}_{\mathbf{u}_0+\Delta\mathbf{u}} \cdot \left[\mathbf{F}^{int}(\mathbf{u}_0+\Delta\mathbf{u}) - (\lambda_0+\Delta\lambda)\mathbf{q}\right] + \delta\lambda\left([K_T]^{-1}_{\mathbf{u}_0+\Delta\mathbf{u}} \cdot \mathbf{q}\right) \Rightarrow$$

$$\boxed{\delta\mathbf{u} = \delta\bar{\mathbf{u}} + \delta\lambda\delta\mathbf{u}_t} \tag{2.14}$$

where,

$$\delta\bar{\mathbf{u}} = -[K_T]^{-1}_{\mathbf{u}_0+\Delta\mathbf{u}} \cdot \left[\mathbf{F}^{int}(\mathbf{u}_0+\Delta\mathbf{u}) - (\lambda_0+\Delta\lambda)\mathbf{q}\right]$$

$$\delta\mathbf{u}_t = \left([K_T]^{-1}_{\mathbf{u}_0+\Delta\mathbf{u}} \cdot \mathbf{q}\right)$$

Note that $\delta\hat{\mathbf{u}}$ and $\delta\mathbf{u}_t$ can be calculated immediately since they only require known information. Once the displacement correction is expressed as in (2.14), it can be substituted in the arc-length equation (2.11). Doing so, would ultimately lead to:

$$\boxed{\alpha_1\delta\lambda^2 + \alpha_2\delta\lambda + \alpha_3 = 0} \tag{2.15}$$

where the coefficients $\alpha_1$ $\alpha_2$ and $\alpha_3$ are given by:

$$\alpha_1 = \delta\mathbf{u}^T \cdot \delta\mathbf{u} + \psi^2(\mathbf{q}^T \cdot \mathbf{q})$$

$$\alpha_2 = 2(\Delta\mathbf{u} + \delta\bar{\mathbf{u}})^T \cdot \delta\mathbf{u}_t + 2\psi^2\Delta\lambda(\mathbf{q}^T \cdot \mathbf{q})$$

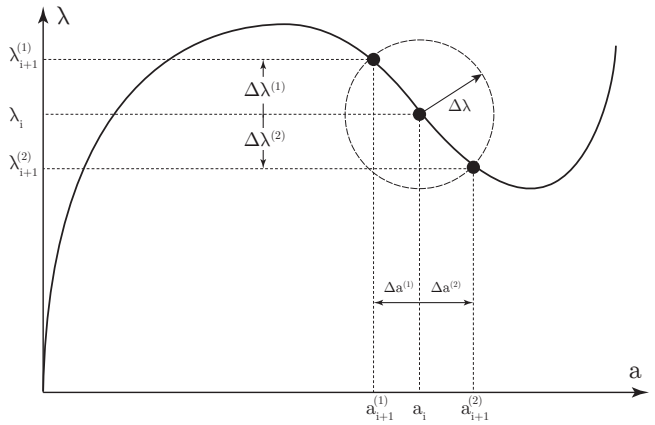
$$\alpha_3 = (\Delta\mathbf{u} + \delta\bar{\mathbf{u}})^T \cdot (\Delta\mathbf{u} + \delta\bar{\mathbf{u}}) + \psi^2\Delta\lambda^2(\mathbf{q}^T \cdot \mathbf{q}) - \Delta l^2$$

Now, with (2.15), we essentially end up with a simple quadratic equation for $\delta\lambda$ which can easily be solve to find $\delta\lambda$. Then, once the $\delta\lambda$ is known, it can be substituted in (2.14) to update the displacement variation and complete the iteration. With this particular formulation, every iteration, the program has to find $\delta\bar{\mathbf{u}}$ and $\delta\mathbf{u}_t$, which can be done by making use of the existing solver since the stiffness matrix involved (Jacobian) is the same as in other methods. Subsequently, one has to make use of $\delta\bar{\mathbf{u}}$ and $\delta\mathbf{u}_t$ to solve the quadratic equation with respect to $\delta\lambda$ and update the variations $\Delta\mathbf{u}$ and $\Delta\lambda$. It is finally a matter of checking of convergence and repeating the aforementioned steps until convergence is achieved and the increment is completed.

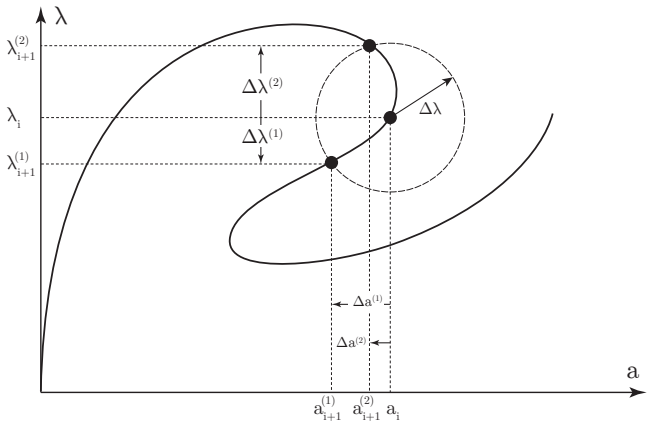
### 2.2.ii The method's drawbacks

Crisfield's implementation however, leads to one of the method's most important drawbacks. The quadratic equation in (2.15) would in general lead to two distinct solutions for $\delta\lambda$ which will in turn lead to two distinct solutions for $\delta\mathbf{u}$. Thus, every iteration, the solver determined two sets of solutions, namely $(\delta\mathbf{u}_1, \delta\lambda_1)$ and $(\delta\mathbf{u}_2, \delta\lambda_2)$. 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 $(\delta\mathbf{u}, \delta\lambda)$ 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'. In an effort to illustrate the difficulty in choosing the correct set of $(\delta\mathbf{u}, \delta\lambda)$, we schematically illustrate the application of the Arc Length method in a snap–through and a snap–back case in Figure 2.5 below. In non-linear problems with a snap–through behavior, it is safe to argue that'next' points in the 'load-displacement' equilibrium path will always cause the displacement




*(a) The Arc Length Method applied on a problem with a Snap–Through behavior. The two solutions that correspond to the candidates for the next point are located bilaterally adjacent to the current point*




*(b) The Arc Length Method applied on a problem with a Snap–Back behavior. The two solutions that correspond to the candidates for the next point are both located at the same region adjacent to the current point*

*Figure 2.5: Formulating a general rule that would indicate which is the direction that the solution evolves 'forwards' is not a straightforward procedure*

to increase, and therefore the correct solution is the one that leads to a positive displacement variation $\Delta\mathbf{u}$. Clearly though, this is not a criterion that could be applied to other cases (i.e. snap–back problems) since there is no general rule that mandates the fact that displacement should always increase. Furthermore, even in snap–through problems, what would be the definition of a positive $\Delta\mathbf{u}$? In a finite element model, there are many diegrees of freedom and every iteration a displacement correction is calculated for every single one of them. It is reasonable then that during deformation, some degrees of freedom experience a positive $\delta u_i$ and others a positive $\delta u_j$. The formulation of a global criterion that would robustly chose the correct solution for every degree of freedom of a general model and at the same time that would be suitable for a wide range of nonlinear problems is a challenging task. New methods and techniques are proposed every now and then, each one making the method more efficient for a particular application but not generally robust.

### 2.2.iii Solution Techniques

An efficient rule to follow in order to choose the next point correctly even when extreme 'snap-back' cases occur is the following. We compute the two displacement corrections $\delta\mathbf{u}_1$ and $\delta\mathbf{u}_2$ corresponding to $\delta\lambda_1$ and $\delta\lambda_2$ respectively. Subsequently we calculate the projections (dot-products) of these generalized correction vectors on the previous corrections. We eventually choose the $\delta\lambda$ and the corresponding $\delta\mathbf{u}$ that lead to the largest value for the dot product and thus form the closest correction to the previous one (hoping that it will be in the right direction). In math form:

$$DOT^{(i)} = (\Delta\mathbf{u} + \delta\mathbf{u}^i, \lambda + \Delta\lambda + \delta\lambda_i) \cdot (\Delta\mathbf{u}, \lambda + \Delta\lambda) \Rightarrow$$

$$DOT^{(i)} = (\Delta\mathbf{u} + \delta\mathbf{u}^i)^T \cdot \Delta\mathbf{u} + \psi^2 \Delta\lambda(\Delta\lambda + \delta\lambda)(\mathbf{q}^T \cdot \mathbf{q}) \quad i = 1, 2 \tag{2.16}$$

Application of this rule leads to a more robust selection of the right correction every time but is again associated with yet another drawback. Since the initial corrections $(\Delta\mathbf{u}, \Delta\lambda)$ are equal to zero at the beginning of each increment (which we will discuss later), the corresponding *DOT* products will be zero as well for both solutions. We have to find a way to initiate the method at the beginning of every increment while still make use of the dot product rule after the first iteration. At every step of the method **apart from the beginning of each increment** we can outline the steps to be followed as:

**a.** Every converged increment store the converged displacement and load corrections as $(\Delta\mathbf{u}_n, \Delta\lambda_n)$.

**b.** Calculate the sign of the product $(\Delta\mathbf{u}_{n+1} + \delta\mathbf{u}^i)^T \cdot \Delta\mathbf{u}_n + \psi^2 \Delta\lambda_n(\Delta\lambda_{n+1} + \delta\lambda)(\mathbf{q}^T \cdot \mathbf{q})$

**c.** We choose the $\delta\lambda$ that leads to the largest DOT product and thus is closer to the previous correction

**d.** In the special case where the two solutions give the same dot products, then choose either one

This method for choosing the correct solution is able to help the solution evolve forwards in most cases and we put it to the test against several problems that exhibit snap–back behavior in the next chapter. By comparing the new correction with the previously converged correction we make sure that the next point does not make the solution evolve backwards. However, applications where this algorithm fails are known to exist and as a general rule, the sharper the transition of the equilibrium path at the onset of a limit point, the harder it is to make sure that the solution evolves forward.

Up until to this point we have postulated that the initial load and displacement variations $\Delta\mathbf{u}$ and $\Delta\lambda$ are given and our only concern is to determine and choose the correct iterative corrections to these variations $\delta\mathbf{u}$ and $\delta\lambda$. However, it is not clear what should be considered at the beginning of every increment with regards to the initial variations $\Delta\mathbf{u}$,$\Delta\lambda$, and as a result it is still unknown how to initiate the method. The Arc-Length method initiation for every increment as well as the iterative loops until convergence is achieved are outlined in the pseudocode that follows:

**1. Initiate Increment**

**2.** Set $\Delta\mathbf{u} = \mathbf{0}$ and $\Delta\lambda = 0$

**3.** $\delta\bar{\mathbf{u}} = \mathbf{0}$ and $\delta\mathbf{u}_t = \left([K_T]_{\mathbf{u}}^{-1} \cdot \mathbf{q}\right)$

**4.** Solve arc length equation for $\delta\lambda_1$ and $\delta\lambda_2$

**5.** Choose the correct solution

**6.** Update $\mathbf{u}, \lambda$ as $\mathbf{u}' = \mathbf{u} + \delta\mathbf{u}$ and $\lambda' = \lambda + \delta\lambda$

**7.** Check for convergence $||\mathbf{R}(\mathbf{u}', \lambda')|| < tol$

**8.** If convergence criteria are met then GOTO Step 10

**9. Initiate Iterations**

- **i.** Set $\Delta\mathbf{u} = \delta\mathbf{u}$ and $\Delta\lambda = \delta\lambda$
- **ii.** Calculate $\delta\bar{\mathbf{u}}$ and $\delta\mathbf{u}_t$
- **iii.** Solve arc length equation for $\delta\lambda_1$ and $\delta\lambda_2$
- **iv.** Choose the correct solution
- **v.** Update $\mathbf{u}, \lambda$ as $\mathbf{u}' = \mathbf{u} + \Delta\mathbf{u} + \delta\mathbf{u}$ and $\lambda' = \lambda + \Delta\lambda + \delta\lambda$
- **vi.** Check for convergence $||\mathbf{R}(\mathbf{u}', \lambda')|| < tol$
- **vii.** If convergence criteria are met then GOTO Step 10
- **viii.** GOTO Step 9

**10.** Proceed to next Increment

Finally, we still need to address a final issue that arises in step **5.** of the previous algorithm. If we set $\Delta\mathbf{u} = \mathbf{0}$ and $\Delta\lambda = 0$ and we do not have any information regarding the last converged increment (i.e. beginning of the analysis) it is impossible to determine the correct solution using the DOT rule since both DOT products will be equal to zero. A way around this issue in such cases is to determine the correct solution based on the sign of the determinant. In particular,

**a.** Calculate the determinant of the Jacobian, namely $[K_T]$, and also it's sign

**b.** Solve the arc length equation for $\delta\lambda_1$ and $\delta\lambda_2$

**c.** Choose the $\delta\lambda_i$ whose sign is the same as the determinants

# 3 Applications

## 3.1 Part I: Structural Mechanics

In this chapter, we examine a number of conceptually easy and fairly simple problems inspired by the fields of structural and continuum mechanics, in an effort to illustrate potential applications where numerical solutions using Newton's method fail and using the Arc Length Method is more appropriate. Our discussion will be initially limited to truss problems that are fairly simple and straightforward to study, mainly because truss structures do not require a spatial discretization in the form of finite elements. Towards the end of this chapter we will also discuss the quasi–static inflation of a hyperplastic spherical membrane and also of a system of interconnected membranes.

### 3.1.i A simple truss problem

We first consider the simplest possible structure comprised of two truss members with initial length $L_0$ and cross section $A_0$ that initially form an angle $\theta_0$ with the horizontal axis as shown in Figure 3.1. The truss members are homogenous and are assumed to be made of an isotropic and linearly elastic material. We also assume that it is impossible for the members to buckle[1] and hence, they can only shrink under compression. Moreover, the fact that trusses can only carry axial forces and can only deform by shrinking or extending, implies that there is no need to discretize this problem[2]. The truss members are connected with a hinge about which they are allowed to rotate, and their lower ends are fixed. A force $P$ is applied at the hinge point subjecting the truss members into compression. With little examination, it is clear that the only degree of freedom is the vertical displacement $u$ of the hinge point and our primary goal in this problem is to determine the relationship between $P$ and $u$.

We will not make any assumptions regarding the magnitude of deformations. The displacement of the hinge $u$ can be arbitrarily large as long as the truss members shrink enough for the displacements to be compatible. A possible deformed configuration for the truss is shown in Figure 3.2. We start from the **equilibrium** equation, which is expressed in terms of force balance between the externally applied force $P$ and the internally developed forces $F_L$, keeping in mind that the equation must be written with respect to

[1] Accounting for buckling in this problem would require to model the trusses as beams and therefore a spatial discretization using finite elements would be necessary. These types of problems can be easily solved using a multipurpose finite element software such as ABAQUS

[2] Trusses in this case can be though as springs

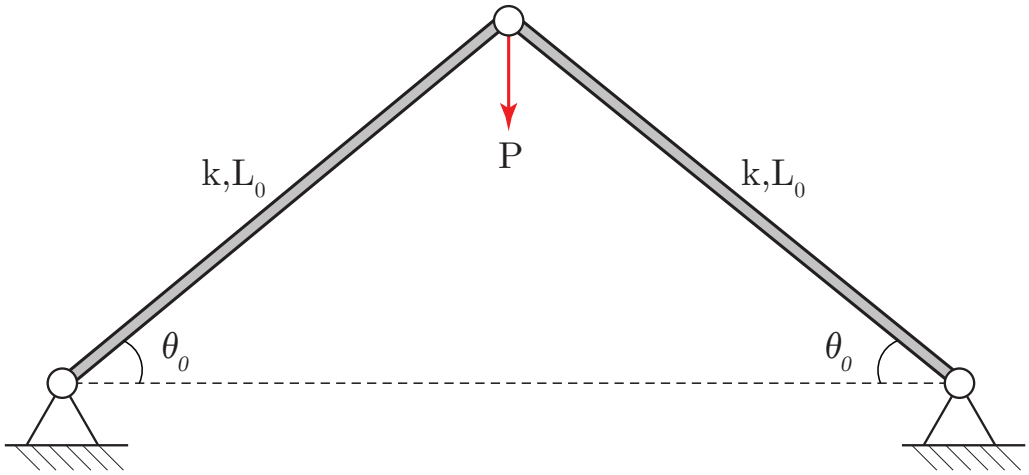



*Figure 3.1: A simple structure consisting of 2 linearly elastic truss members that form an initial angle $\theta_0$ with the horizontal plane. The structure is loaded with a force $P$ that subjects the truss members into compression*

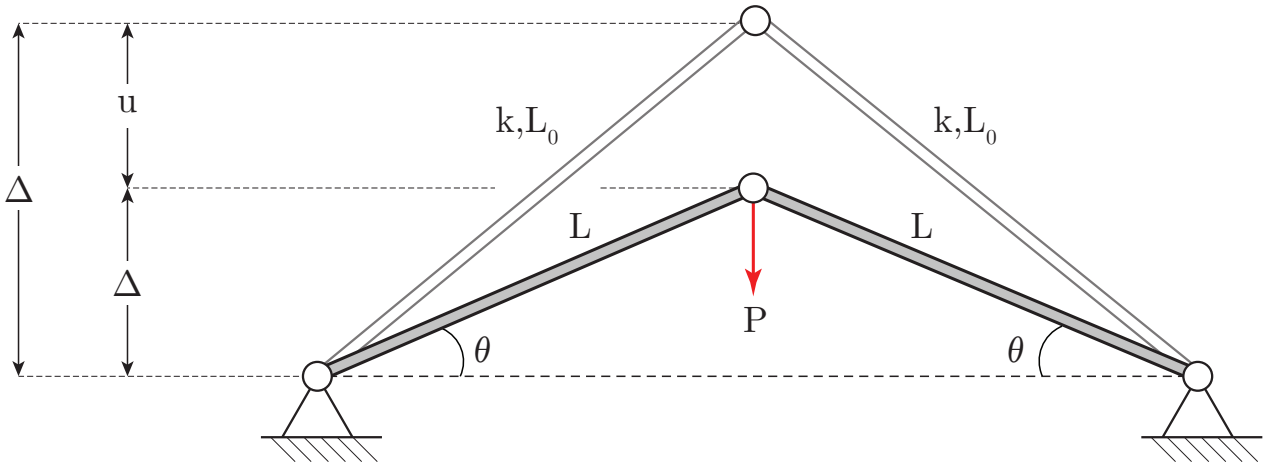



*Figure 3.2: A schematic representation of a possible deformed state for the structure consisting of two truss members subjected into compression*

the deformed state. According to Figure 3.3 we can write:

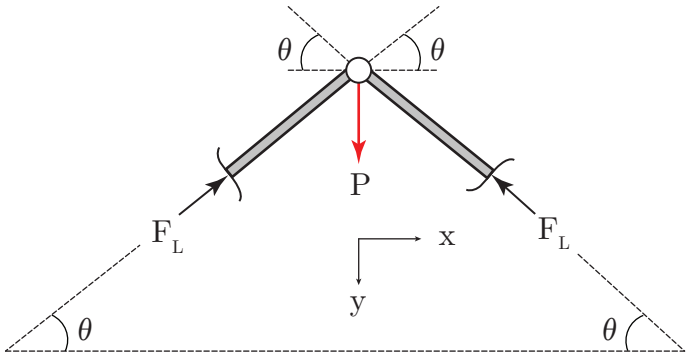



*Figure 3.3: Force balance; the compressive/tensile forces developed internally in the trusses must be in equilibrium with the externally applied force $P$*

$$\mathbf{\Sigma}F = \mathbf{0} \Rightarrow \left\{ \begin{array}{c} 0 = 0 \\ P = 2F_L \sin(\theta) \end{array} \right\} \Rightarrow \boxed{P = 2F_L \sin(\theta)} \tag{3.1}$$

The **constitutive** equation is

$$\sigma = E\,\varepsilon \Rightarrow \frac{F_L}{A_0} = E\frac{L_0 - L}{L_0} \Rightarrow F_L = \frac{\overbrace{E\,A_0}^{k}}{L_0}(L_0 - L) \Rightarrow \boxed{F_L = k(L_0 - L)} \tag{3.2}$$

where $k$ is a measure of each member's stiffness. Now, we only have to determine the **kinematics** equations that would relate the hinge's displacement $u$ with the initial and deformed lengths of the truss members. We expect the kinematics equation to be nonlinear as a result of the preliminary assumption that the displacement $u$ can be arbitrarily large. Based on Figure 3.9, we take the Pythagorean theorem for $(ABC)$ and:

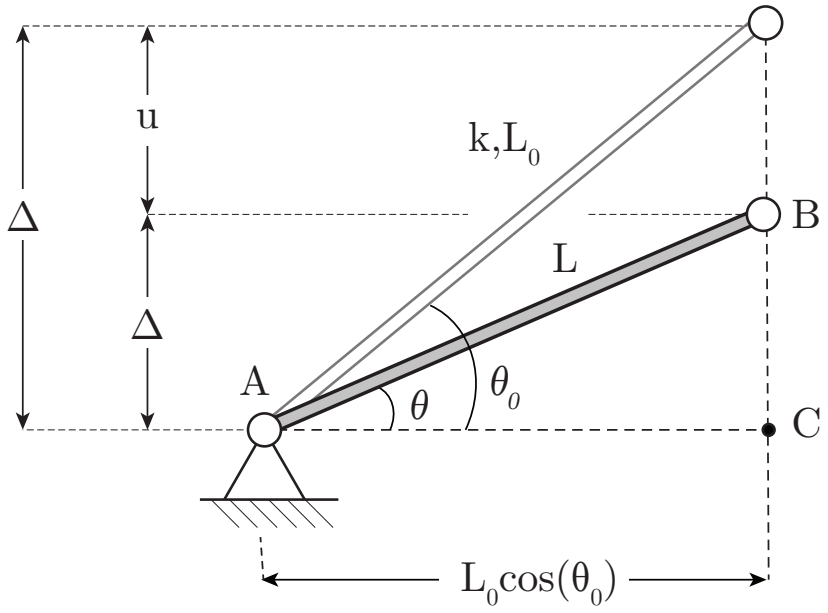



*Figure 3.4: Deformed and undeformed states in the case of finite deformations*

$$L^2 = \left(\Delta'\right)^2 + \left(L_0\cos(\theta_0)\right)^2 = \left(L_0\sin(\theta_0) - u\right)^2 + \left(L_0\cos(\theta_0)\right)^2 = L_0^2 - 2L_0\,u\,sin(\theta_0) + u^2 \Rightarrow$$

$$\boxed{\frac{L}{L_0} = \sqrt{1 - 2\frac{u}{L_0}\sin(\theta_0) + \left(\frac{u}{L_0}\right)^2}} \tag{3.3}$$

Now combining equations (3.1)–(3.3) all together we get:

$$P = 2k(L - L_0)\sin(\theta) = 2k(L - L_0)\left(\sin(\theta_0) - \frac{u}{L_0}\right) \Rightarrow$$

$$\frac{P}{2kL_0} = \left(\frac{L}{L_0} - 1\right)\left(\sin(\theta_0) - \frac{u}{L_0}\right) = \left(\frac{1}{\sqrt{1 - 2\frac{u}{L_0}\sin(\theta_0) + \left(\frac{u}{L_0}\right)^2}} - 1\right)\left(\sin(\theta_0) - \frac{u}{L_0}\right) \Rightarrow$$

$$\boxed{\frac{P}{2kL_0} = \left( \frac{1}{\sqrt{1 - 2\frac{u}{L_0}\sin(\theta_0) + \left(\frac{u}{L_0}\right)^2}} - 1 \right) \left( \sin(\theta_0) - \frac{u}{L_0} \right)} \tag{3.4}$$

We now define the normalized load $\lambda$ and displacement $a$ as follows:

$$\lambda = \frac{P}{2kL_0} \qquad , \qquad a = \frac{u}{L_0}$$

Now, the expression can be written as:

$$\boxed{\lambda(a) = \left( \frac{1}{\sqrt{1 - 2a\sin(\theta_0) + a^2}} - 1 \right) (\sin(\theta_0) - a)} \tag{3.5}$$

Now if we plot this expression we would get a normalized force–displacement curve $\lambda - -a$ that characterizes the structures behavior and a representative plot is shown in Figure 3.5 below.

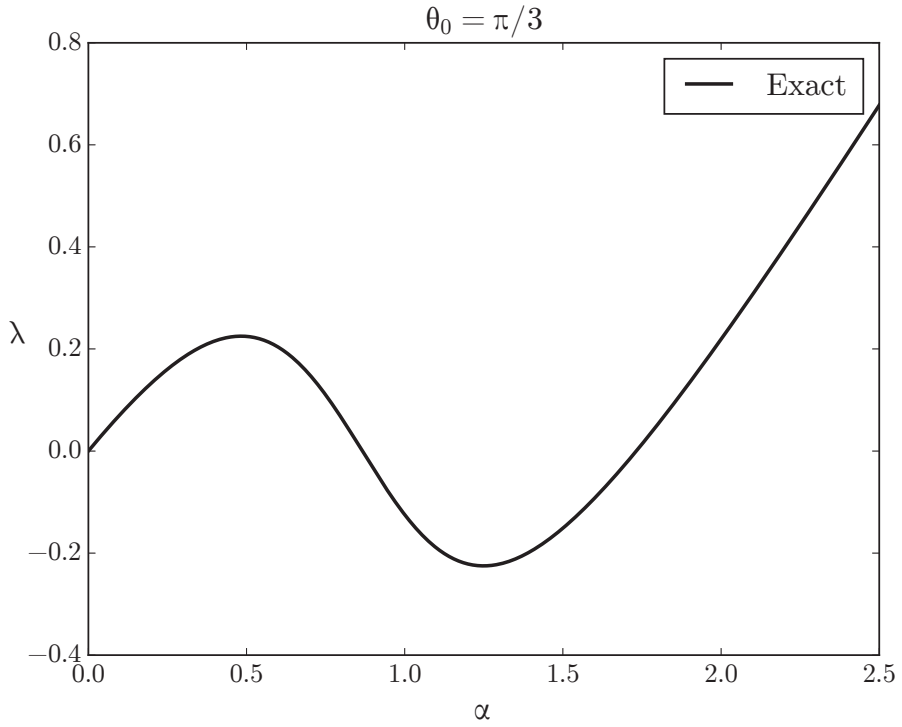



*Figure 3.5: A plot of the normalized force displacement curve for the simple truss problem*

From the structural mechanics point of view it is rather interesting to interpret this type of snap–through behavior under load–control and some typical deformation stages that correspond to curve in Figure 3.5 are shown schematically in Figure 3.6.

For reasons that were extensively described in a previous chapter, it is impossible to capture the structures behavior using Newton's method in solving this problem numerically. An example of the result that we obtain

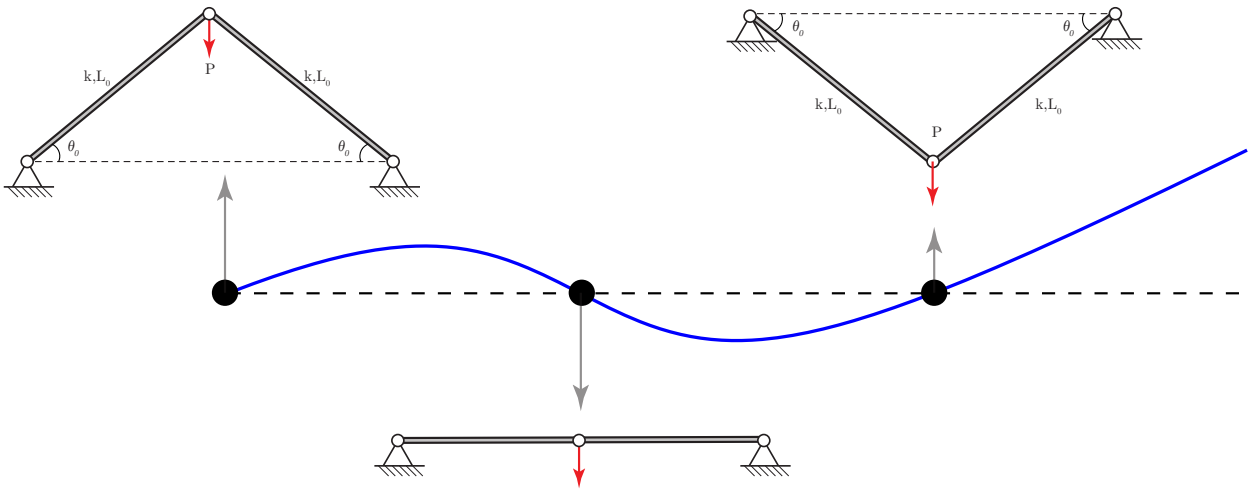



*Figure 3.6: The deformation states that correspond to the three stable configurations during deformation*

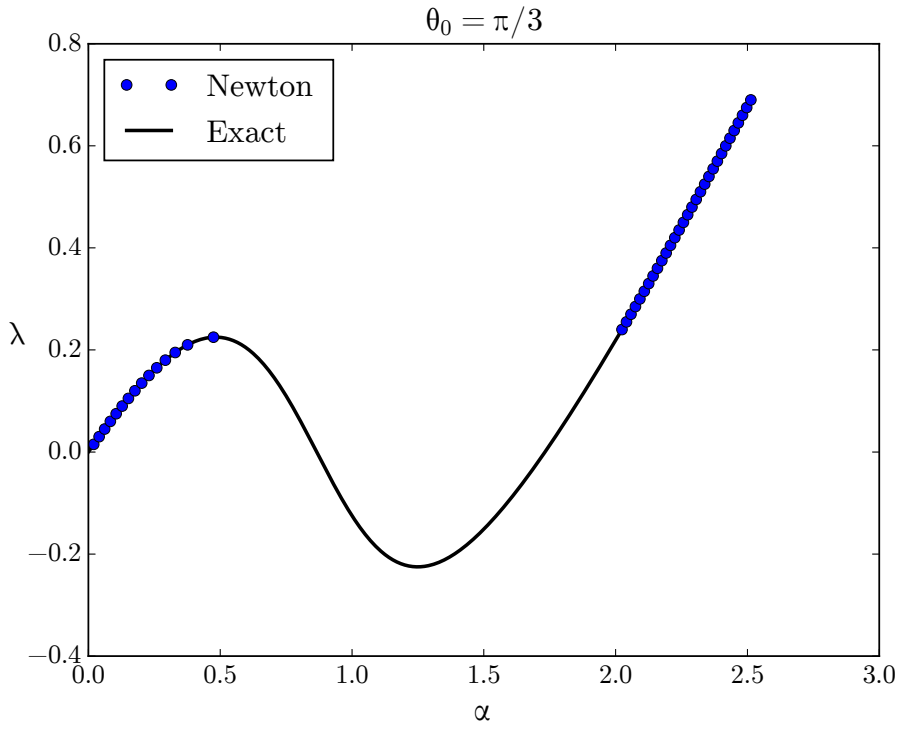



*Figure 3.7: The converged points that one obtains using Newton's method for the simple truss problem. Newton's method is not able to capture the snap–through instability*

if we insist in using Newton's method is shown in Figure 3.7 below. On the other hand, as we would expect, using the Arc–Length method, we are able to capture the actual response in this problem, making the Arc Length method most suitable for such applications. The converged points in the case of the Arc Length method are shown in Figure 3.8.

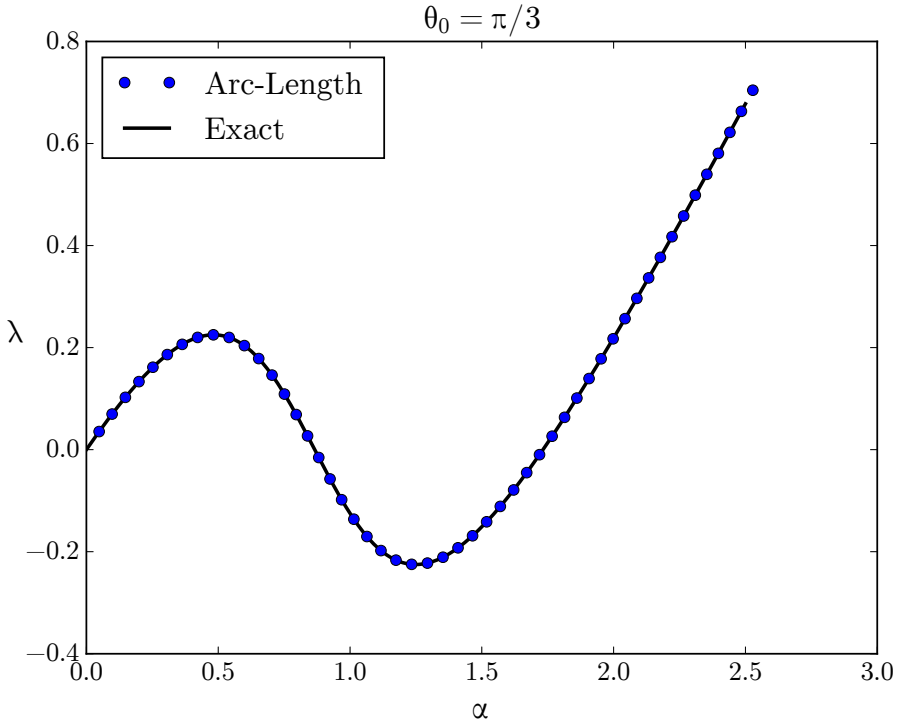



*Figure 3.8: The converged points that one obtains using the Arc Length method for the simple truss problem. The Arc Length method is more suitable for solving numerically such pronblems*

### 3.1.ii A more involved truss problem

Now that we got a first taste on the benefits of the Arc Length method over Newton's method in a very simple problem, we consider a slightly more complicated truss configuration. We have the same truss members arranged just as they were before, but this time we place another truss member vertically and on top of the previous structure. The new truss member has a different undeformed length $l_0$ and cross section $A_0'$. We also assume that in general, the newly introduced truss member is made of a different material and it's stiffness is:

$$\beta = \frac{E'\,A_0'}{l_0}$$

The setup can be better conceptualized in Figure 3.9 that follows. The boundary conditions are the same as before for the lower truss members apart from the the fact that now the force $P$ is applied at the upper part of the structure. Both hinge points are constrained in order to eliminate any displacement in the horizontal direction[3] and thus they are only able to move vertically.

In this case, the problem has 2 degrees of freedom, and we can chose them to be the vertical displacements of the two hinge points. The force subjects the upper member to compression, but the lower part of the top truss is able to move by subjecting the lower truss members into compression as well. Therefore, using the

[3] Although it is not necessary in terms of the mechanics of the problem since it is already guaranteed by symmetry. However, in terms of numerical simulations, it is necessary to impose this boundary conditions in order to get the expected behavior

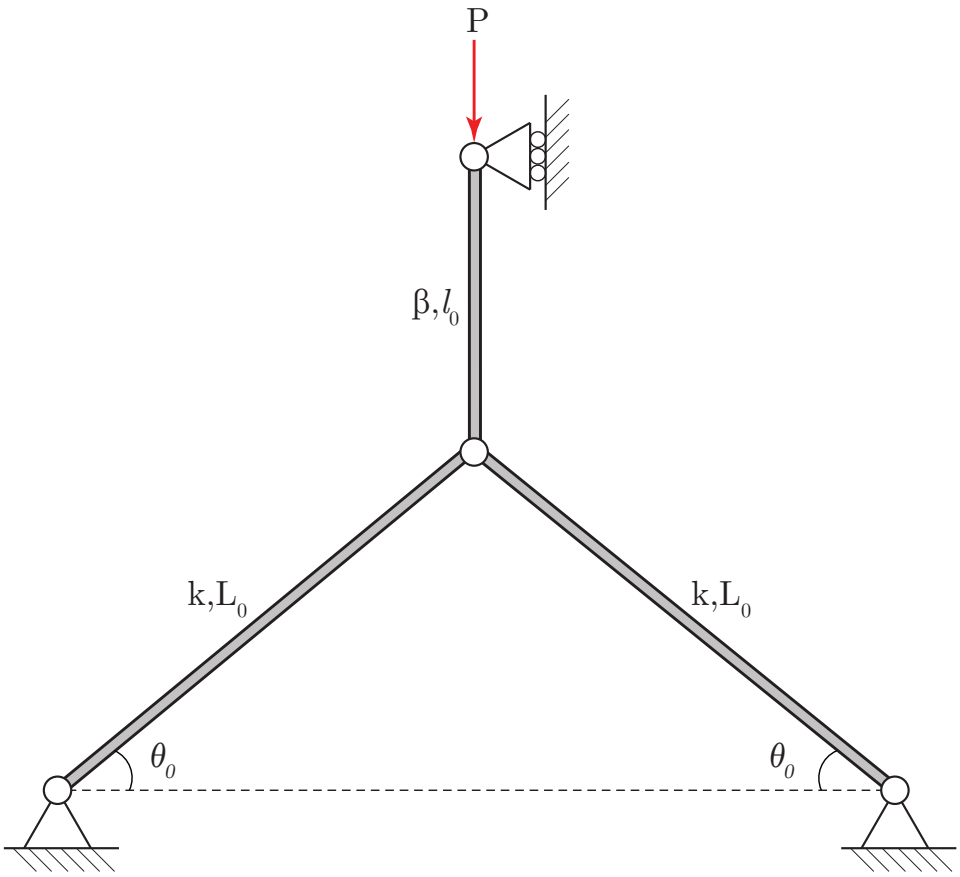



*Figure 3.9: A slight variation to the simple truss structure where we added a linearly elastic vertical member with a different stiffness (in general). The compressive force is now applied at the top of the upper member*

notation $u_1$, $u_2$ depicted in Figure 3.10 and assuming that all truss members have the same length, we can express the deformation of the upper truss member as:

$$e = \frac{u_2 - u_1}{L_0} = a_2 - a_1 \tag{3.6}$$

where recall that $a$ is defined as the displacement normalized with its initial length. Then, the equilibrium and constitutive equations for the upper truss member suggest that:

$$P = \beta L e \Rightarrow P = \beta L(a_2 - a_1) \Rightarrow \frac{P}{\beta L} = a_2 - a_1 \tag{3.7}$$

Now at this point, recall that when we solved the previous problem, we determined a relationship $\lambda(a) \rightarrow P(a)$. This however implies that the displacement $a_2$ depends on $P$[4]. Therefore, we write:

$$\boxed{a_2 = a_1(P) + \frac{P}{\beta L}} \tag{3.8}$$

Unfortunately, it is impossible to invert expression (3.5) due to its complexity and as a result we are unable to determine a closed form solution for this problem. Clearly, as soon as the problem became slightly more

[4]This can become clear if we could invert the expression $P = P(a_2) \Rightarrow a_2 = a_2(P)$

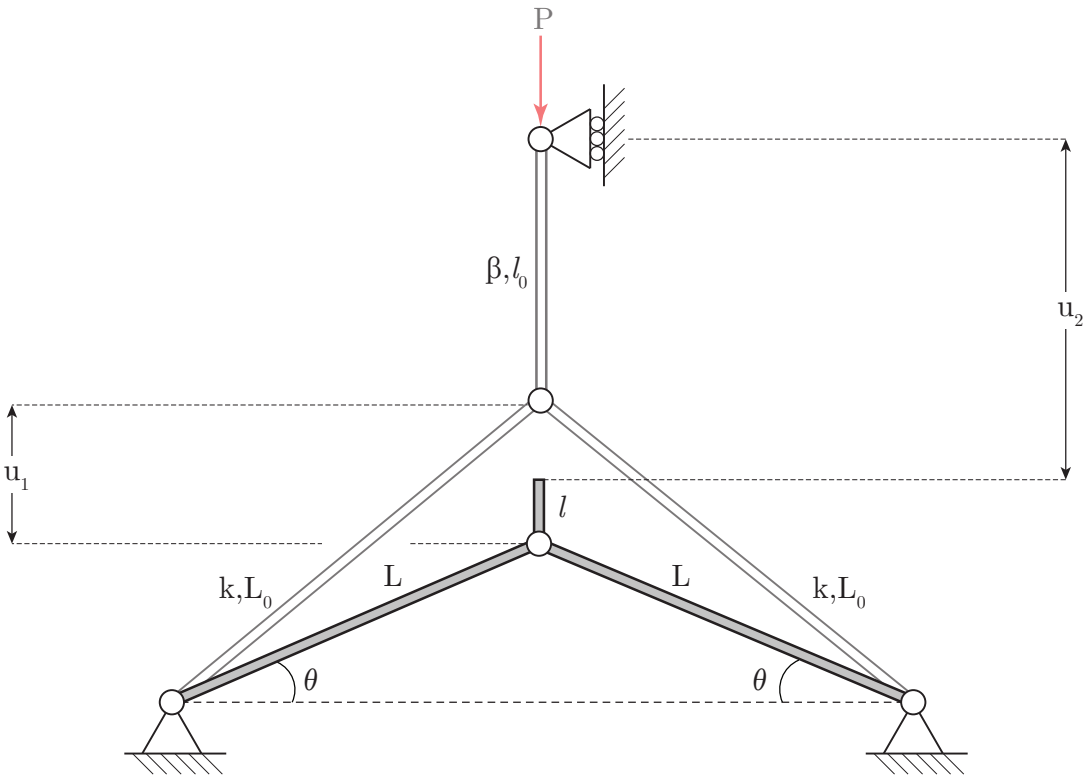



*Figure 3.10: A schematic representation of a possible deformed state for the new structure subjected into compression*

complicated, analytical solutions cannot be determined. However, we are still interested in determining the $P-u$ curve for this problem, and examine the effect of the stiffness ratio of the two truss members, namely:

$$w = \frac{\beta}{k}$$

Our only choice when it comes to solving this problem is to formulate a system of equations and implement a numerical algorithm to solve it incrementally. Now in this problem we won't even try to solve the problem using Newton's method since it is guaranteed to fail, being unable to capture unstable responses as we illustrated in the previous subsection. Hence, we will implement the Arc-Length method. Denoting by $F_B$ the internal force developed in the upper truss member, we can write the constitutive law as:

$$F_B = \beta L(a_2 - a_1) \tag{3.9}$$

Now by demanding the developed forces in the top rod as well as in the hinge point to be in equilibrium we derive:

$$\begin{aligned} &\textit{Hinge Point:} \quad F_B = 2F_L \sin\theta \\ &\textit{Top rod:} \qquad\quad F_B = P \end{aligned} \tag{3.10}$$

Now by recalling equation (3.5) and combining it with equations (3.10) and (3.9) we end up with the

following 2x2 system of equations with respect to the normalized displacements $a_i$ as:

$$\begin{gathered} 2kL\left(\frac{1}{\sqrt{B(a_1,\theta_0)}}-1\right)(\sin\theta_0-a_1)-\beta L(a_2-a_1)=0 \\ \beta L(a_2-a_1)-P=0 \end{gathered} \tag{3.11}$$

where,

$$B(a_1,\theta_0)=1-2a_1\sin\theta_0+a_1^2$$

Now define again the normalized load as $\lambda = P/2kL$, and also the stiffness ratio between the truss members as $w = \beta/k$. Recall that we are already using the normalized displacements as $a_i = u_i/L$. We can now modify the above system and write it in the following normalized form as:

$$\left(\frac{1}{\sqrt{B(a_1,\theta_0)}}-1\right)(\sin\theta_0-a_1)-w(a_2-a_1)=0 \tag{3.12}$$

$$w(a_2-a_1)-\lambda=0 \tag{3.13}$$

The above system has essentially the the general form $\mathbf{F}^{int}(\mathbf{u})-\lambda\mathbf{q}=\mathbf{0}$ hypothesized in (2.7). In order to visualize the similarities we can write the system of equations as:

$$\mathbf{F}^{int}(\mathbf{u})=\mathbf{F}^{int}(a_1,a_2)=\begin{bmatrix} \left(\frac{1}{\sqrt{B(a_1,\theta_0)}}-1\right)(\sin\theta_0-a_1)-w(a_2-a_1) \\ \\ w(a_2-a_1) \end{bmatrix}=\begin{bmatrix} F_1(a_1,a_2) \\ F_2(a_1,a_2) \end{bmatrix}$$

and,

$$\mathbf{q}=\begin{Bmatrix} 0 \\ 1 \end{Bmatrix}$$

and finally:

$$\mathbf{F}^{int}(\mathbf{u})-\lambda\mathbf{q}=\mathbf{0}\Rightarrow\begin{bmatrix} F_1(a_1,a_2) \\ F_2(a_1,a_2) \end{bmatrix}-\lambda\begin{Bmatrix} 0 \\ 1 \end{Bmatrix}=\begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \tag{3.14}$$

The next step is to implement the Arc Length method and solve the system of equations. In Figure 3.11 we plot the normalized load $\lambda$ versus the normalized displacement $a_2$ for various stiffness ratios $w$.

The resulting curves have an interesting interpretation. On the one hand, if the stiffness ratio is very large ($w = 50$), then the upper truss member is much stiffer than the rest of the truss members and as a result it does not deform significantly. As expected then, the force displacement curve in this case is almost identical to the previous problem.

On the other hand, if the decrease the stiffness of the upper truss member with respect to the other members, the curve eventually snaps back and the structure is unstable in both force and displacement control conditions.

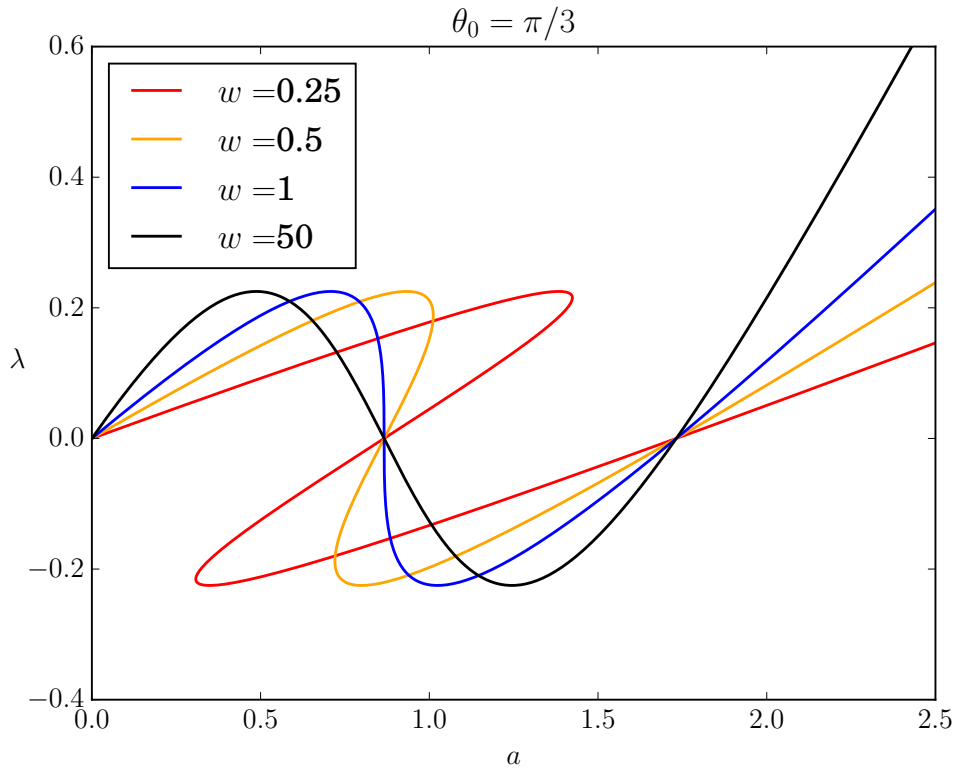



*Figure 3.11: The normalized force–displacement curves (displacement $a_2$) for various stiffness ratios $w = \beta/k$. Extreme values of $w$ can make the structure unstable under loading in both force and displacement control*

## 3.2 Part II: Continuum Mechanics

### 3.2.i Inflation of a Hyperelastic Spherical Membrane

In this problem, we consider the quasi static inflation of a hyperelastic spherical membrane with initial radius and thickness $R$ and $H$ respectively. We assume that $H << R$ so that bending and shear forces do not contribute significantly in the membrane's deformation and can therefore be neglected. When the membrane is subjected to a uniform pressure p, a biaxial state of stress is achieved and the principal stretches are given by

$$\lambda_r = \frac{h}{H} \quad , \quad \lambda_\theta = \lambda_\phi = \frac{r}{R} \tag{3.15}$$

where $r$ and $h$ represent the deformed (current) radius and thickness of the balloon respectively. The stress field that develops into the membrane in order to equate the pressure is taken to be that of a thin–walled spherical container subjected to an internal pressure $p_i = p$ and is of the form:

$$\sigma_r = 0 \quad , \sigma = \sigma_\theta = \sigma_\phi = \frac{p\,r}{2h} \tag{3.16}$$

The fact that $\sigma_r = 0$ arises from the thin–waled assumption that results into a plane stress field for the membrane. Assuming that the membranes we are trying to model are made of an incompressible hyperelastic

material[5] we write:

$$\frac{dV}{dV_0} = 1 \Rightarrow \lambda_r\,\lambda_\theta\,\lambda_\phi = 1 \Rightarrow \lambda_r = \frac{1}{\lambda_\theta\,\lambda\phi} \tag{3.17}$$

and introducing the strain energy function $W$ we can write the constitutive law for the membrane's material as:

$$\sigma = \lambda_\theta \frac{\partial W(\lambda_\theta, \lambda_\phi)}{\partial \lambda_\theta} \tag{3.18}$$

Next, we introduce a specific form for the strain energy function assuming that the material's response can be modelled using the **incompressible Gent model**. Thus:

$$W(\lambda_\theta, \lambda_\phi) = \frac{-\mu J_m}{2} \log\left(1 - \frac{\lambda_\theta^2 + \lambda_\phi^2 + \lambda_\theta^{-2}\lambda_\phi^{-2} - 3}{J_m}\right) \tag{3.19}$$

where $\mu$ is the initial shear modulus and $J_m$ is a constant related to the strain saturation of the material (as the stresses become infinite). Note that the nonlinear pressure-volume response of an inflated spherical membrane depends greatly on the constitutive material model. Although in this study we have focused on a Gent model to achieve a final steep increase in pressure, when using a Varga, neo-Hookean, or three-term Ogden model no strain stiffening is observed upon inflation [Overvelde *et al.*]. Now combining equations (3.15), (3.16) and (3.19) and plugging the result into (3.18) yields:

$$\sigma_\theta = \frac{p\,r}{2\,h} = \mu\,J_m \frac{1 - \lambda_\theta^6}{2\lambda_\theta^6 - \lambda_\theta^4(3 + J_m) + 1} \tag{3.20}$$

Dropping the subscript $\theta$ and expressing $h$ as $h = H\,R^2/r^2$ leads to the following form for expression (3.20):

$$\sigma = \frac{p\,R}{\mu\,H} = \frac{2\,J_m}{\lambda^3} \frac{1 - \lambda^6}{2\lambda^6 - \lambda^4(3 + J_m) + 1} \tag{3.21}$$

The current (deformed) volume inside the membrane is expressed in terms of $\lambda$ as:

$$v = \frac{4\pi}{3}(\lambda\,R^3) \Rightarrow \lambda^3 = \frac{v}{V_i} \tag{3.22}$$

where $V_i$ stands for the initial volume inside the membrane prior to any deformation. In order to simplify expressions we next define the following dimensional variables:

$$\hat{p} = \frac{p\,R}{\mu\,H} \qquad , \qquad \alpha = \lambda^3 \tag{3.23}$$

where $\hat{p}$ is assumed to represent a dimensionless pressure. Let us now write equation (3.21) in the form $F(\lambda, \hat{p}) = 0$, which is a traditional way of introducing nonlinear equations that favors the implementation of numerical techniques. Then:

$$\boxed{\hat{p} = \frac{2\,J_m}{\alpha} \frac{1 - \alpha^2}{2\alpha^2 - \alpha^{4/3}(3 + J_m) + 1}} \tag{3.24}$$

Now if we plot this expression for different values of $J_m$ we would get the curves shown in Figure 3.12.

We note however that we just made the plot of the $p - -\lambda$ curves using the analytical expressions and did not use our numerical technique (Arc Length Method). We are mostly interested in solving problems to which we don't know the answer and one of them is the case where several of these hyperelastic spherical membranes are connected in series and we inflate one of them. We consider this problem in the next subsection.

[5] This is an assumption that we traditionally make when we study hyperelastic rubber like materials, such as balloons

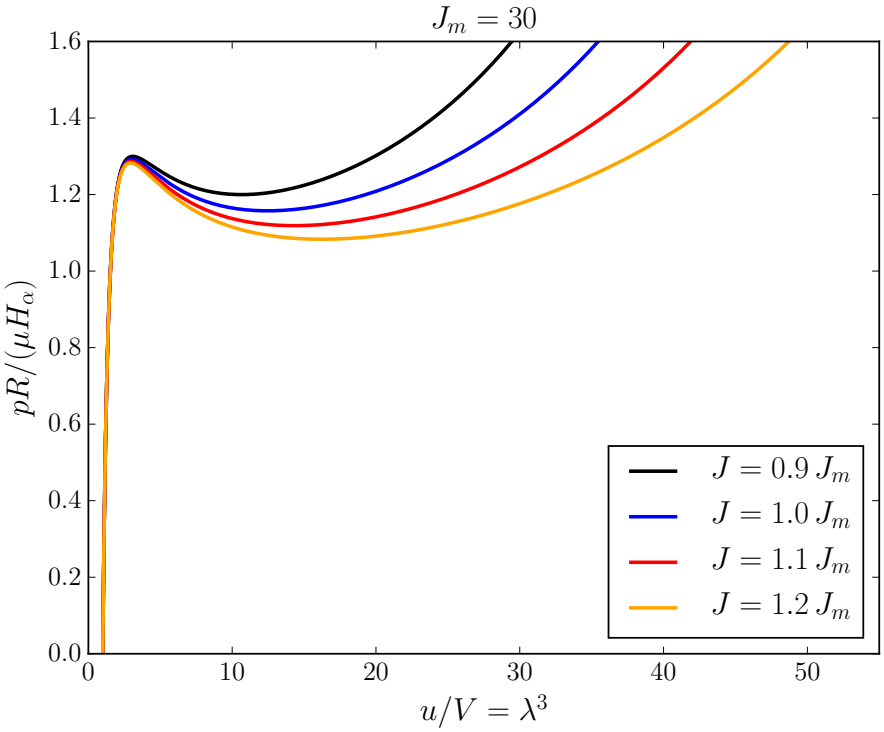



*Figure 3.12: Normalized $p - v$ curves for the inflation of hyperplastic balloons according to the Gent model*

### 3.2.ii Inflating a System of $N$ interconnected hyperelastic spherical membranes

We would like to generalize equation (3.24) in the case of $N$ spherical membranes connected together and which, in general, have different material constants $(H_i, J_i)$. The description of such a system of membranes would result into a system of $N$ equations with respect to $N$ unknowns, namely the $\alpha_i$'s. First, let us introduce the system of equations, considering $N$ membranes with constants $H_i, J_i$ $(i = 1, 2, \ldots, N)$.

$$F_i(\alpha_i, \hat{p}_i) = \frac{2\,J_m}{\alpha_i} \frac{1 - \alpha_i^2}{2\alpha_i^2 - \alpha_i^{4/3}(3 + J_i) + 1} - \hat{p}_i = 0 \tag{3.25}$$

Notice that the dimensionless pressure $\hat{p}_i$ depends on the membrane since its definition involves the constant $H_i$. We could avoid that dependency on the specific membrane however, by introducing the following ratio:

$$q_i = \frac{H_m}{H_i} \tag{3.26}$$

Now using equation (3.26) we can rewrite the equations (3.25) in the form:

$$F_i(\alpha_i, \hat{p}) = \frac{2\,J_i}{\alpha_i} \frac{1 - \alpha_i^2}{2\alpha_i^2 - \alpha_i^{4/3}(3 + J_i) + 1} - q_i\,\hat{p} = 0 \tag{3.27}$$

In the case of interconnected spherical membranes, we can define the total volume within the membranes $v$ as the sum of volumes enclosed by each individual membrane as:

$$v = \sum_{i=1}^{N} v_i \tag{3.28}$$

which can also be readily expressed (Only in the special case where the membranes have initially the same radius $R$) in terms of the $a_i$'s, once we recall the definition in (3.22). Hence:

$$N\,\alpha = \sum_{i=1}^{N} a_i \Rightarrow \alpha_N = N\,\alpha - \sum_{i=1}^{N-1} a_i \tag{3.29}$$

The last equation, serves as a constraint to the system of equations in (3.27). Notice that without this constraint, each equation in (3.27) is autonomous and provides the $p$–$v$ description for each membrane alone. Now however, we have the following, coupled system of equations:

$$\mathbf{F}^{\text{int}}(\alpha_1, \alpha_2, \ldots, \alpha_{N-1}, \alpha) - \mathbf{q}\,p = \mathbf{0} \tag{3.30}$$

where:

$$F_i^{\text{int}} = \frac{2\,J_i}{\alpha_i}\frac{1-\alpha_i^2}{2\alpha_i^2 - \alpha_i^{4/3}(3+J_i)+1} \quad i = 1, 2, \ldots, N-1$$

$$F_N^{\text{int}} = \frac{2\,J_N}{\tilde{\alpha}}\frac{1-\tilde{\alpha}^2}{2\tilde{\alpha}^2 - \tilde{\alpha}^{4/3}(3+J_N)+1} \quad \text{for the last eq.}$$

where

$$\tilde{\alpha} = \alpha - \sum_{i=1}^{N-1} a_i \tag{3.31}$$

Knowing that we are going to solve this system of equations using the Arc Length Method we would like to have an expression for the (Stiffness matrix) Jacobian of the system. The Jacobian $[K]$ is defined as:

$$K_{ij} = \frac{\partial F_i}{\partial \alpha_j} \tag{3.32}$$

in our case however, equations $F_i$ for $i = 1, 2, \ldots, N-1$ depend only on the corresponding $a_i$. In other words they do not depend on $a_j$ for $j \neq i$. That means that matrix $[K]$ will look like a diagonal matrix, since all off diagonal entries will be equal to zero with the only exception being the last row. Schematically:

$$[K] = \begin{bmatrix} K_{11} & 0 & 0 & 0 \\ 0 & K_{11} & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ K_{N1} & K_{N1} & \ldots & K_{NN} \end{bmatrix} \tag{3.33}$$

In the context of this problem, we have created a program in Python that is able to solve the system of equations for a general system of $N$ membranes, and the program is able to build the Jacobian matrix and the system of equations automatically. The only input required by the used is the material parameters associated with each membrane. Below, we present results for a system comprising of 2 interconnected membranes and the 3 cases for the material constants considered are taken from [Overvelde *et al.*] and are: we consider the three possible ways of combining these membranes (a+b,b+c,a+c) and for each case we solve the system of equations with the Arc Length method. The results are presented in the form of $p - v$ curves and are shown below:

| **Membrane** | $J_i$ | $q_i$ |
|---|---|---|
| **a** | $J_a = 30$ | $q_a = 1$ |
| **b** | $J_b = 0.9J_a$ | $q_a = 1.1$ |
| **c** | $J_c = 1.2J_a$ | $q_a = 1.05$ |

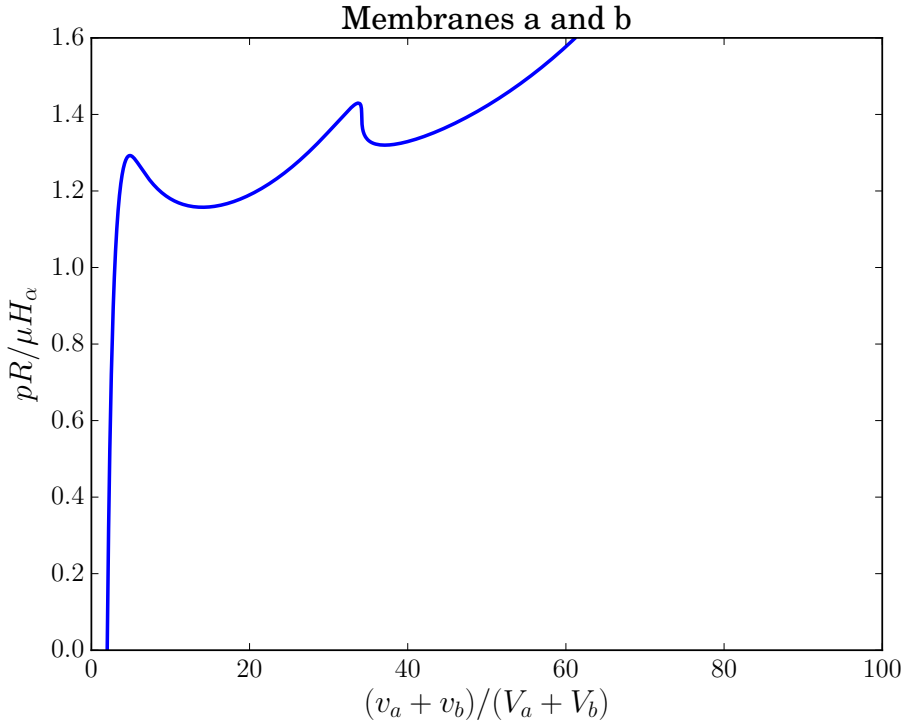



*Figure 3.13: The pressure–total system volume curve in the case of a system comprising of membranes a and b*

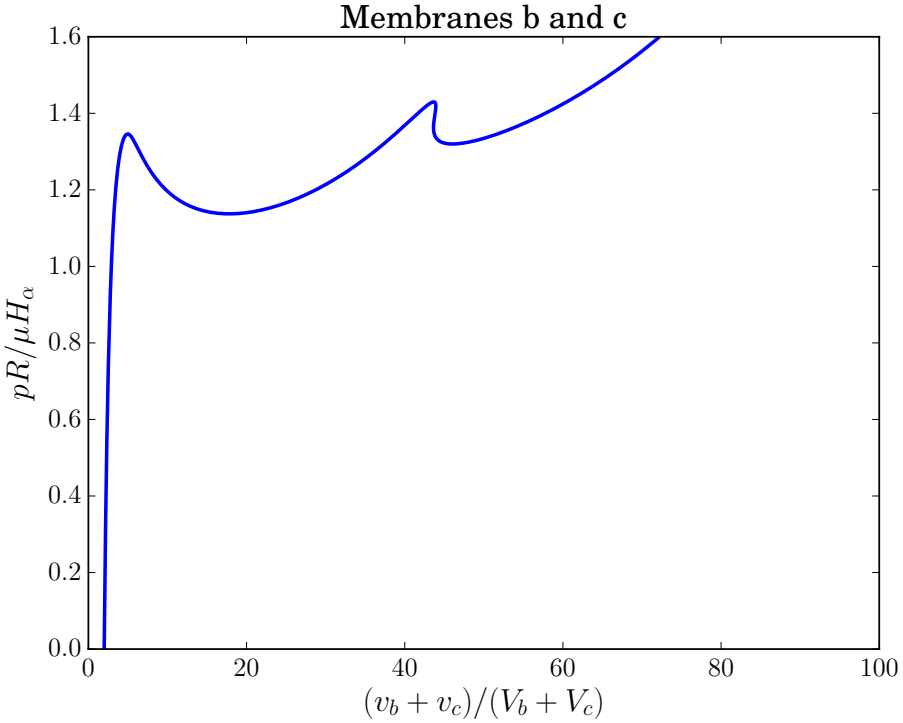



*Figure 3.14: The pressure–total system volume curve in the case of a system comprising of membranes b and c*

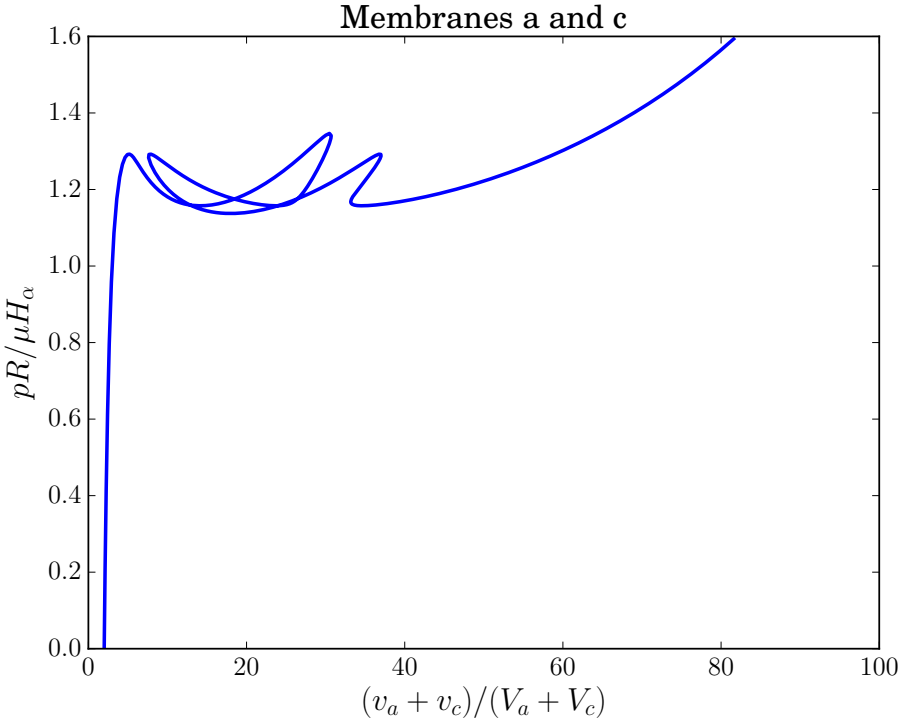



*Figure 3.15: The pressure–total system volume curve in the case of a system comprising of membranes a and c*

## 3.3 Links to Codes, Videos

The programs developed in the context of this project to implement the Newton's method, the Arc Length Method in every application (trusses, membranes etc) as well as codes that developed to produce and make the video to which we provide the link in chapter 2, are available to download through the following links.

- Newton's method for the Simple Truss Problem [Get It Here]
- Arc Length method for the Simple Truss Problem [Get It Here]
- Arc Length method for the 2nd truss problem with 2 d.o.f. [Get It Here]
- Video [Get It Here]

# Bibliography

[Abaqus Theory Manual] *Abaqus Theory Manual, version 6.12, Dassault Systemes.*

[Crisfield M.A., 1983] Crisfield M.A., (1983), 'A fast incremental / iterative solution procedure that handles snap-through', *Computers And Structures*, **13**:55–62

[Overvelde *et al.*] Overvelde JTB, Kloek T, D'haen JJA, Bertoldi K., (2015), 'Amplifying the response of soft actuators by harnessing snap-through instabilities', *The Proceedings of the National Academy of Sciences of the United States of America*,**112**:10863-10868

[Riks E., 1979] Riks E. (1979), 'An incremental approach to the solution to the solution of buckling ans snapping problems', *Int. J. Solids Struct.*, **15**:524–551