Governing Equations & Solution Algorithms

Nonlinear analysis on the computer is an extension of the stiffness method. From another document on this website, the governing equilibrium equations for the computational stiffness method at the structural level are

$$\mathbf{K}_f \mathbf{u}_f = \mathbf{F}_f \tag{1}$$

where $\mathbf{K}_f$=stiffness matrix in the Final configuration, $\mathbf{u}_f$=degrees of freedom (DOFs), and $\mathbf{F}_f$=load vector containing externally applied loads on the structure. As described in the introductory document on the stiffness method, the left-hand side of Eq. (1), **Ku**, resists forces applied along the DOFs. In nonlinear analysis those resisting forces must be written differently, because the resisting forces are no longer proportional to **u**. Instead, the resisting forces are, in general, a nonlinear function of **u**. That nonlinearity is schematically visualized in the force-displacement diagram in Figure 1. As the lateral force on the structure increases beyond some level, an increase in the displacement is no longer proportional to the increase in the force; it is greater. As described in separate documents, the nonlinearity can come from two sources: geometrical and material nonlinearity.

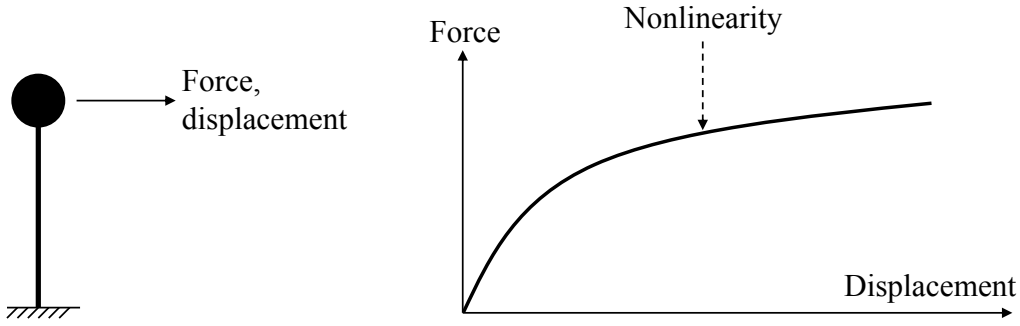

Figure 1: Nonlinear force-displacement relationship.

The notation adopted for the nonlinear resisting forces employs the tilde symbol; hence, the governing equilibrium equations for nonlinear static structural analysis are

$$\tilde{\mathbf{F}}_f(\mathbf{u}_f) = \mathbf{F}_f \tag{2}$$

where $\tilde{\mathbf{F}}_f$=resisting forces in the Final configuration. As mentioned in the document on the computational stiffness method, there are several versions of the vector **F**. Having now introduced $\tilde{\mathbf{F}}_f$, the following summary is given:

- $\tilde{\mathbf{F}}_f$, $\tilde{\mathbf{F}}_a$, $\tilde{\mathbf{F}}_g$, $\tilde{\mathbf{F}}_l$, $\tilde{\mathbf{F}}_b$ = Resisting force vector in any of the DOF configurations
- $\grave{\mathbf{F}}_f$, $\grave{\mathbf{F}}_a$ = Nodal load vector at the structural level, i.e., point loads along DOFs
- $\bar{\mathbf{F}}_f$, $\bar{\mathbf{F}}_a$, $\bar{\mathbf{F}}_g$, $\bar{\mathbf{F}}_l$, $\bar{\mathbf{F}}_b$ = Clamping forces, addressing element loads
- $\mathbf{F}_f$, $\mathbf{F}_a$ = Total load vector at the structural level, containing both previous cases
- $\mathbf{F}_g$, $\mathbf{F}_l$, $\mathbf{F}_b$ = Force vector at the element level containing member-end forces.

Two comments can be added to that list. First, in addition to the Basic, Local, Global, All, and Final configurations, two additional configurations appear in nonlinear analysis: Section and Material. The indices "s" and "m" are employed for those, and they appear in other documents posted on this website. Second, in the implementation of nonlinear analysis on the computer, the element loads are sometimes made part of the resisting force vector. That is convenient because the elements usually return their resisting forces and the element forces at the same time. Symbolically, this means that Eq. (2) remains valid but $\bar{\mathbf{F}}_f$ is part of $\tilde{\mathbf{F}}_f$ instead of $\mathbf{F}_f$. Nothing changes at the element level, where the element force vector is determined, for example at the Basic level, by

$$\mathbf{F}_b = \tilde{\mathbf{F}}_b(\mathbf{u}_b) + \bar{\mathbf{F}}_b \tag{3}$$

Specifics on the relationship between the deformations, $\mathbf{u}$, and the resisting forces, $\tilde{\mathbf{F}}$, is the subject of other documents on this website.

Newton-Raphson Algorithm

While the governing equations in linear analysis, $\mathbf{Ku}=\mathbf{F}$, are straightforwardly solved for $\mathbf{u}$ by linear solvers, other algorithms are needed to solve $\tilde{\mathbf{F}}(\mathbf{u}) = \mathbf{F}$ for $\mathbf{u}$. As a starting point, Eq. (2) is written on residual form, now omitting subscripts because it is clear we are at the structural level:

$$\tilde{\mathbf{F}}(\mathbf{u}) - \mathbf{F} = \mathbf{R} = \mathbf{0} \tag{4}$$

where $\mathbf{R}$=residual force vector. This is a root-finding problem; we seek the value of $\mathbf{u}$ that gives zero residual along all DOFs. The document on 1D optimization algorithms explains the Newton-Raphson algorithm for root-finding, named after Isaac Newton (1643-1727) and Joseph Raphson (1648–1715), for the problem

$$f(x) = 0 \tag{5}$$

leading to the recursive formula

$$x_{i+1} = x_i - \frac{1}{f'(x_i)} \cdot f(x_i) \tag{6}$$

where i=iteration number. Considering the multivariate case, the Taylor series expansion of the residual about a displacement level $\mathbf{u}_i$ reads

$$\mathbf{R}(\mathbf{u}) = \mathbf{R}(\mathbf{u}_i) + \frac{\partial \mathbf{R}(\mathbf{u}_i)}{\partial \mathbf{u}}(\mathbf{u} - \mathbf{u}_i) + \frac{1}{2}\frac{\partial^2 \mathbf{R}(\mathbf{u}_i)}{\partial \mathbf{u}^2}(\mathbf{u} - \mathbf{u}_i)^2 + \cdots \tag{7}$$

Ignoring the second-order term and higher-order terms leads to the following linear system of equations when the residual is set to zero:

$$\mathbf{R}(\mathbf{u}_i) + \frac{\partial \mathbf{R}(\mathbf{u}_i)}{\partial \mathbf{u}}(\mathbf{u} - \mathbf{u}_i) = \mathbf{0} \tag{8}$$

A conceptual, but not practical way of deriving the Newton-Raphson algorithm for the multivariate case is to solve Eq. (8) for $\mathbf{u}$:

$$\mathbf{u}_{i+1} = \mathbf{u}_i - \left[\frac{d\mathbf{R}(\mathbf{u}_i)}{d\mathbf{u}}\right]^{-1} \cdot \mathbf{R}(\mathbf{u}_i) \tag{9}$$

In practical implementations, we do not calculate the inverse of the derivative of the residual. That is why Eq. (9) is conceptual. Instead, we define $\Delta\mathbf{u}=\mathbf{u}-\mathbf{u}_i$ and solve the linear system of equations shown in Eq. (8) for that displacement increment, using some solver that is more efficient than calculating the inverse shown in Eq. (9):

$$\frac{\partial \mathbf{R}(\mathbf{u}_i)}{\partial \mathbf{u}} \Delta\mathbf{u} = -\mathbf{R}(\mathbf{u}_i) \tag{10}$$

so that the Newton-Raphson algorithm reads

$$\mathbf{u}_{i+1} = \mathbf{u}_i + \Delta\mathbf{u} \tag{11}$$

Assuming that the load vector, $\mathbf{F}$, does not depend on the deformations $\mathbf{u}$, the derivative appearing above is

$$\frac{\partial \mathbf{R}(\mathbf{u}_i)}{\partial \mathbf{u}} = \frac{\partial \tilde{\boldsymbol{F}}(\mathbf{u}_i)}{\partial \mathbf{u}} = \mathbf{K}_{tangent} \tag{12}$$

where the "tangent stiffness" has been defined. That stiffness is informally illustrated in Figure 2 together with the "initial stiffness."

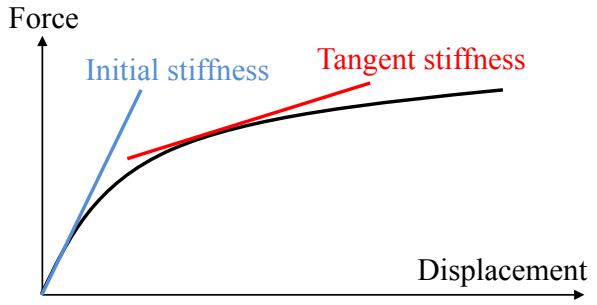

Figure 2: The concepts of initial and tangent stiffness.

Depending on which stiffness is used, the following two algorithms emerge, again using the conceptual notation described above:

- Newton-Raphson algorithm: $\mathbf{u}_{i+1} = \mathbf{u}_i - \mathbf{K}_{tangent}^{-1} \cdot \mathbf{R}(\mathbf{u}_i)$
- Modified Newton-Raphson algorithm: $\mathbf{u}_{i+1} = \mathbf{u}_i - \mathbf{K}_{initial}^{-1} \cdot \mathbf{R}(\mathbf{u}_i)$

(13)

Figure 3 illustrates the two versions of the Newton-Raphson algorithm. One observation is that the modified Newton-Raphson algorithm requires more iterations than the version that uses the tangent stiffness. On other hand, there is cost associated with the implementation and repeated recalculation of the tangent stiffness. For that reason, both algorithms are viable in practical applications.

A pseudo-code for one load step with the Newton-Raphson algorithm is, starting at the displacement $\mathbf{u}=\mathbf{0}$:

1. Calculate the initial stiffness, $\mathbf{K}_{\text{initial}}$
2. Calculate the trial displacement, $\mathbf{u}_{i+1}$, using Eq. (11)

3. State determination:
    a. Calculate trial strains in materials corresponding to the trial displacement
    b. Calculate stress (and material stiffness?) corresponding to the trial strains
    c. Propagate back up to get the resisting forces, $\tilde{\mathbf{F}}$ (and tangent stiffness?)
4. Calculate the residual $\mathbf{R} = \tilde{\mathbf{F}} - \mathbf{F}$
5. If the residual is non-zero, perhaps measured by its norm, then return to Step 2 and use the tangent stiffness if that was calculated in Step 3

This algorithm is incomplete because it places the full external load on the structure at once. That is rarely a good idea for real problems, which exhibit multiple yielding and potential local unloading events on the way to the final load level. Convergence problems would likely occur. This is addressed by "load incrementation" strategies, addressed in another document, essentially controlling how the loads are applied.

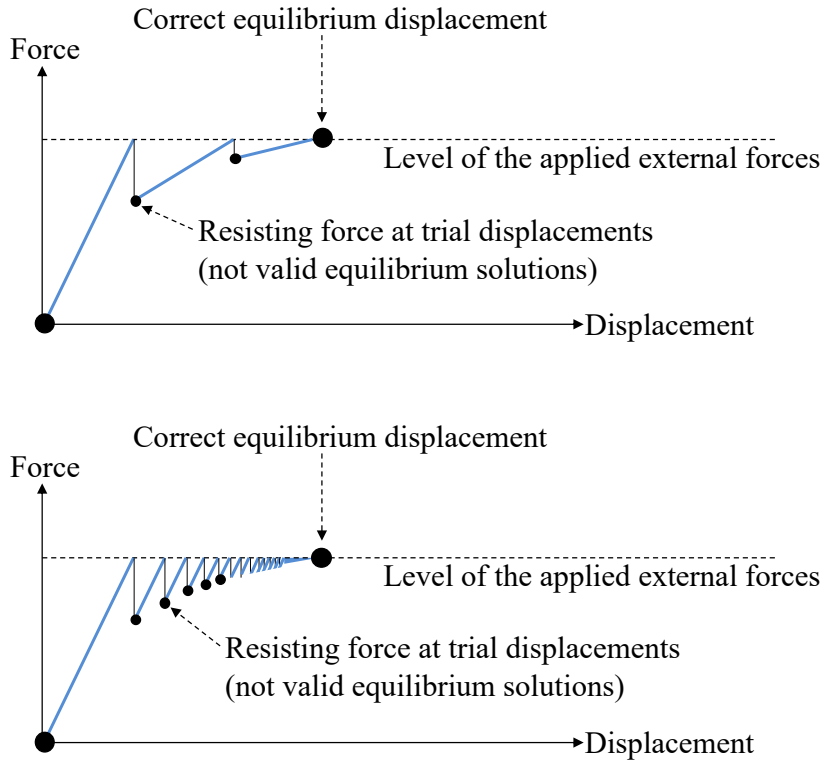

Figure 3: Newton-Raphson (top) and Modified Newton-Raphson (bottom).