Load Incrementation Strategies

The previous document on this website says something about how to solve the governing equations $\tilde{\mathbf{F}}_f = \mathbf{F}_f$ in nonlinear structural analysis. That document also alludes to the fact that it is rarely a good idea to put the full external loads on at once; that may lead to convergence problems. Load incrementation strategies address that issue by writing the load as

$$\mathbf{F}_f = \lambda \cdot \mathbf{F}_{ref} \tag{1}$$

where λ is the "load factor" and $\mathbf{F}_{ref}$ are the "reference loads," i.e., the full loads on the structure. In this manner, the loads can be placed on the structure little-by-little, by letting λ vary in steps from zero to unity. Notice the jargon: Each step is called a load *increment* or a load *step*, and Newton-Raphson *iterations* to equilibrium occur at each increment. It is helpful to relate λ to a time-parameter that is "pseudo time" in static analysis and actual time in dynamic analysis:

$$\mathbf{F}_f(t) = \lambda(t) \cdot \mathbf{F}_{ref} \tag{2}$$

That opens up a range of possibilities for applying the reference load, of which some are visualized in Figure 1. An important option in earthquake engineering is to set $\lambda(t) = \ddot{u}_g(t)$ and let the reference load be expressed as $-\mathbf{M}\cdot\boldsymbol{\Gamma}$, the latter described in the document on MDOF dynamics.

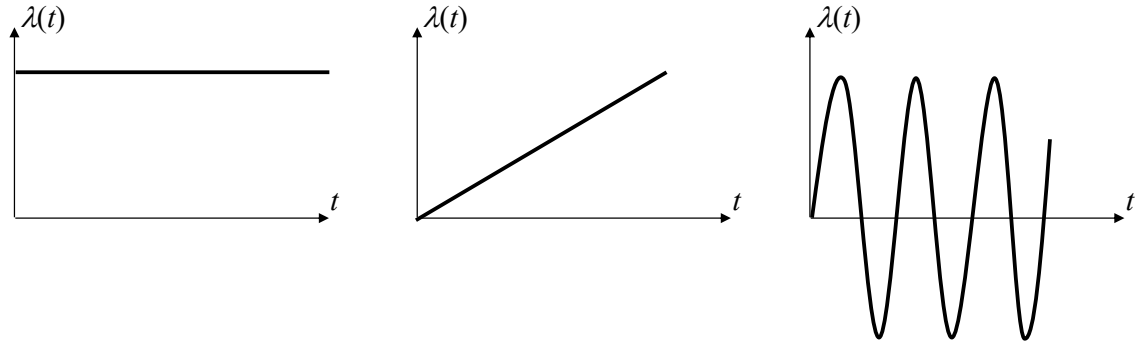

Figure 1: Time series.

In static analysis, where t represents pseudo time, there is freedom in how $\lambda(t)$ and $\mathbf{F}_{ref}$ are defined relative to each other. Suppose a 5kN load is to be applied gradually to the structure, according to the linear graph at the middle of Figure 1. One option is then to set $\mathbf{F}_{ref}$ =5kN and $\lambda(t)=t$, reaching the full load at pseudo time t=1. Another option is to set $\mathbf{F}_{ref}$ =5kN and $\lambda(t)=0.01t$, reaching the full load at time t=100. Yet another option is to set $\mathbf{F}_{ref}$ =1kN and $\lambda(t)=t$, reaching the full load at time t=5. Finally, another option is to set $\mathbf{F}_{ref}$ =1kN and $\lambda(t)=0.1t$, reaching the full load at time t=50. Clearly, an infinite number of other options exist. Regardless, Δt is selected so that the desirable number of increments, i.e., load steps are analyzed. In terms of jargon, the functions $\lambda(t)$ in Figure 1 are referred to as *time series*, while the reference load, $\mathbf{F}_{ref}$, is referred to as a *load pattern*. The total load on a structure can be expressed by any number of time series & load pattern pairs:

$$\mathbf{F}_f(t) = \sum_{m=1}^{M} \lambda_m(t) \cdot \mathbf{F}_{ref,m} \tag{3}$$

where *M*=number of load patterns. What is described above is the simplest form of "load control" analysis. At each increment, the loads on the structure are directly controlled by the load factor, λ, which is incremented according to a pre-defined time series, $\lambda(t)$. The implementation of a load control analysis in Python is demonstrated in the "Nonlinear 2-DOF Load Control" example posted on this website.

Continuation Methods

A problem with the approach described above is that the restoring forces in the structure may not be able to reach the applied load level. In other words, the load control analysis described above is unable to determine post-peak response. That is what continuation methods are intended for and some of those are described below. First, a generic derivation is provided in order to allow for the load factor, λ, to vary within an increment, i.e., during the Newton-Raphson iterations. To that end, it is first recalled that the symbol *n* represents the increment number, while *i* is the iteration number. Also, the subscript *f* for the Final structural configuration is dropped for brevity of notation. The load factor at an increment can be expressed in terms of the previous increment as

$$\lambda_n = \lambda_{n-1} + \Delta\lambda \tag{4}$$

That means the applied loads at increment *n* are

$$\mathbf{F}_n = \mathbf{F}_{n-1} + \Delta\lambda \cdot \mathbf{F}_{ref} \tag{5}$$

In turn, that means the residual at iteration *i* reads

$$\mathbf{R}_i = \tilde{\mathbf{F}}_i - \mathbf{F}_n = \tilde{\mathbf{F}}_i - \mathbf{F}_{n-1} - \Delta\lambda \cdot \mathbf{F}_{ref} \tag{6}$$

Next, we make a big move, by allowing the load factor to vary from one iteration to the next within an increment. As a result, the terms in Eq. (6) that relates to the applied loads are given the index *i*:

$$\mathbf{R}_i = \tilde{\mathbf{F}}_i - \mathbf{F}_{i-1} - \Delta\lambda_i \cdot \mathbf{F}_{ref} \tag{7}$$

It is stressed that now $\Delta\lambda$ has the index *i*, allowing it to vary within an increment. At the first Newton-Raphson iteration within an increment, the load vector $\mathbf{F}_{i-1}$ is the load vector at the end of the previous increment; thereafter it is the load vector at the previous iteration. The steadily accumulating total trial displacements are

$$\mathbf{u}_i = \mathbf{u}_{i-1} + \Delta\mathbf{u}_i \tag{8}$$

The Newton-Raphson algorithm determines the change in the trial displacements, i.e., the displacement increment, $\Delta\mathbf{u}_i$, by solving the system of equations

$$\mathbf{K}\,\Delta\mathbf{u}_i = -\mathbf{R}_i \tag{9}$$

Following Filippou & Fenves' 2004 chapter in Bozorgnia and Bertero's book Methods of Analysis for Earthquake-Resistant Structures, Eq. (7) is now substituted into Eq. (9), which gives

$$\mathbf{K}\,\Delta\mathbf{u}_i = \mathbf{F}_{i-1} + \Delta\lambda_i \cdot \mathbf{F}_{ref} - \tilde{\mathbf{F}}_i \tag{10}$$

Next, the displacement increment is split into two parts, each matching a portion of the right-hand side of Eq. (10). Adopting the notation $\Delta\mathbf{u}_i = \Delta\mathbf{u}_{\mathrm{R},i} + \Delta\mathbf{u}_{\mathrm{T},i}$ (R for residual, T for tangent) the two equations emanating from the split of Eq. (10) are

$$\mathbf{K}\,\Delta\mathbf{u}_{\mathrm{R},i} = \mathbf{F}_{i-1} - \tilde{\mathbf{F}}_i \tag{11}$$

and

$$\mathbf{K}\,\Delta\mathbf{u}_{\mathrm{T},i} = \Delta\lambda_i \cdot \mathbf{F}_{ref} \tag{12}$$

In regards to Eq. (11), it is noted that the applied loads can be written in terms of the load factor at the previous iteration, implying that Eq. (11) can be written

$$\mathbf{K}\,\Delta\mathbf{u}_{\mathrm{R},i} = \lambda_{i-1} \cdot \mathbf{F}_{ref} - \tilde{\mathbf{F}}_i \tag{13}$$

In regards to Eq. (12), it is observed that $\Delta\mathbf{u}_{\mathrm{T},i}$ can be written in terms of a reference displacement, $\mathbf{u}_{\mathrm{T}}$, which remains constant throughout the iterations at each increment:

$$\Delta\mathbf{u}_{\mathrm{T},i} = \Delta\lambda_i \cdot \Delta\mathbf{u}_{\mathrm{T}} \tag{14}$$

where $\mathbf{u}_{\mathrm{T}}$ is obtained by solving the system of equations

$$\mathbf{K}\,\mathbf{u}_{\mathrm{T}} = \mathbf{F}_{ref} \tag{15}$$

In short, the displacement increment is written

$$\Delta\mathbf{u}_i = \Delta\mathbf{u}_{\mathrm{R},i} + \Delta\lambda_i \cdot \mathbf{u}_{\mathrm{T}} \tag{16}$$

The main contribution of the formulation presented above is opening up the possibility of making the load factor increment, $\Delta\lambda$, a variable within an increment. That facilitates the calculation of the post-peak response. Several methods are available for that purpose, as described in the following.

Displacement Control

In this approach, one degree of freedom is selected to be a "control DOF," sometimes casually referred to as a control node. At that DOF the displacement increment is imposed, meaning that it is not unknown, thereby giving us a left-over equation from which we determine the load factor. In other words, we take advantage of the developments above, allowing the load factor to vary during the Newton-Raphson iterations. In the following derivations it is convenient to define a "selection vector," $\mathbf{s}$, of dimension equal to the number of DOFs. $\mathbf{s}$ has zeros everywhere, except unity in the position of the control DOF. As an illustration, $\mathbf{s}^{\mathrm{T}}\mathbf{u}$ is the displacement at the control DOF picked from the displacement vector $\mathbf{u}$. Applied to Eq. (16), the displacement increment at the control DOF is

$$\mathbf{s}^{\mathrm{T}}\Delta\mathbf{u}_i = \mathbf{s}^{\mathrm{T}}\Delta\mathbf{u}_{\mathrm{R},i} + \Delta\lambda_i \cdot \mathbf{s}^{\mathrm{T}}\mathbf{u}_{\mathrm{T}} \tag{17}$$

Except at the first iteration, which is addressed below, we wish the change in the displacement at the control node to be zero. Setting Eq. (17) equal to zero and solving it for the load factor increment yields

$$\Delta\lambda_i = -\frac{\mathbf{s}^{\mathrm{T}}\Delta\mathbf{u}_{\mathrm{R},i}}{\mathbf{s}^{\mathrm{T}}\mathbf{u}_{\mathrm{T}}} \tag{18}$$

That is the essence of the displacement control algorithm. However, a special consideration is needed at the first iteration, i.e., for i=1. It is in that iteration that we impose the user-defined displacement increment at the control DOF. That is done by setting Eq. (17) equal to the user-defined Δu_o and solving for the load increment while recognizing that the "residual" displacement is zero at the first iteration:

$$\Delta\lambda_1 = \frac{\Delta u_o}{\mathbf{s}^{\mathrm{T}}\mathbf{u}_{\mathrm{T}}} \tag{19}$$

The implementation of a displacement control analysis in Python is demonstrated in the "Nonlinear 2-DOF Displacement Control" example posted on this website.

Other Continuation Methods

(To be written.)