Chapter 1
Methods of Nonlinear Analysis

Keith D. Hjelmstad
University of Illinois at Urbana-Champaign

Engineers have used mathematical models to help estimate the likely response of structures to natural and manmade loads since Galileo first investigated the carrying capacity of a cantilever beam and Bernoulli and Euler investigated elastic deflections and buckling of beams centuries ago. Even though much of the early work in modeling of structures included nonlinear effects, linear analysis eventually became the dominant tool of the structural engineer. Just as the advent of the digital computer made analysis of complex structures possible in the 1960s, the ubiquity of low-cost computers made linear structural analysis routine by the 1980s. During this great period of growth in computational capability, linear structural analysis programs moved to the center of structural engineering practice because of their power, flexibility, and reliability.

During this same period researchers began to explore methods of nonlinear structural analysis of complex structures that could be codified in a manner similar to the way linear structural analysis was (i.e., flexible problem specification, direct assembly of governing equations, and the like). The progress was steady, albeit slow. Because of the variety inherent in nonlinear problems, nonlinear analysis methods took on an ad hoc feel with one method working well for one kind of problem, but poorly for a different one. The development of nonlinear analysis methods suffered under the popularity and success of linear analysis methods because generations of engineers had been taught to think linearly. The tendency toward linear thinking is clearly visible in some of the early attempts at nonlinear structural analysis.

Over the last decade or so, methods of nonlinear structural analysis have matured to the point where we can begin to think of them as useful tools for the practice of structural engineering. With robust nonlinear analysis tools, the important features of structural response can be modeled more accurately and the behavior of structures can be quantified with greater confidence. Better computer simulations of structural response to design loads will forever change the way we do structural design as the need for the traditional specification-based member capacity checks disappears. In some cases the

specification requirements are not even consistent with the data obtained from a nonlinear structural model. We are on the leading edge of a new era in structural engineering, one in which nonlinear analysis will play a central role.

In this monograph we are concerned with the nonlinear analysis of building structures. This focus puts a natural constraint on the types of structural elements and materials under our purview. In some ways, this restriction makes the discussion of nonlinear analysis of structures more manageable. In other ways, the problems and methods of nonlinear analysis manifest in their full glory for buildings structures just as they do in other areas of engineering where nonlinear mathematical models are used.

Throughout this and the next chapter we shall be concerned with solving the equation

$$\mathbf{g}(\mathbf{x}) = \mathbf{0} \tag{1.1}$$

where $\mathbf{x}$ is the vector of unknowns. We will find in Chapter 2 that the equations of equilibrium of a structure can always be put in this form, and that the function $\mathbf{g}$ is simply the difference between the internal resisting forces of the members of the structure and the externally applied loads. In the structural context, the unknowns $\mathbf{x}$ are often the displacements (generally characterized through a finite number of degrees of freedom of the discretized system) but can also include the load level, as in the arc-length control methods, or internal forces (when mixed variational methods are employed). Therefore, establishing the equilibrium of a structural system can be viewed as a problem in finding the roots of a function $\mathbf{g}(\mathbf{x})$. In this chapter we examine general methods for solving Eqn. (1.1) when $\mathbf{g}(\mathbf{x})$ is a nonlinear function.

There are a number of features of nonlinear analysis that make it different from linear analysis. However, at the core of every nonlinear analysis algorithm lies the solution to a linear problem. Indeed, it is instructive to view the solution to a nonlinear problem as a sequence of linear problems. The trick is to figure out what those linear problems should be and how those results should be combined. The solution of $\mathbf{Ax} = \mathbf{b}$, therefore, plays a central role in the solution of nonlinear problems. In the course of solving a nonlinear problem, $\mathbf{Ax} = \mathbf{b}$ will be solved many times and, hence, the premium on efficient solution strategies is even more acute than it is for linear problems.

The simple problem of finding the roots of a univariate function provides a comfortable context in which to warm up to the ideas and algorithms of nonlinear computation. It is here that we start our discussion. In the context of the univariate problem we identify the significance of linearization of the original equation and an iterative strategy for computing the solution to the nonlinear problem to any specified degree of accuracy. Newton's method in n dimensions generalizes the one-dimensional algorithm and introduces the ubiquitous tangent matrix $\mathbf{A}$ and residual $\mathbf{b}$. We discuss some modifications of the basic Newton algorithm for static problems, including alternative methods of evaluation of $\mathbf{A}$ and one-dimensional line searches. We discuss the sources of nonlinearity in building structures and introduce the important notions of nonlinear equilibria and static stability of structures through a simple example. We end the chapter by

presenting a nonlinear theory of truss structures to illustrate the connection between structural theories and the general methods of analysis outlined earlier in the chapter. Introducing the nonlinear truss has two primary merits. First, it is a discrete nonlinear theory—perhaps the only nonlinear structural theory that can be treated as discrete from the outset. As such, it parallels the developments of this chapter and provides a strong tie with linear structural theories where it is also possible to view beams and frames as discrete structures. Second, the truss theory has all of the elements of the more complicated continuum theories of solids and frames. As such, it provides a natural introduction to the developments of Chapter 2.

In Chapter 2 we will bring the focus of the present chapter more directly to bear on the problems structural mechanics in an effort to describe where the function $\mathbf{g}(\mathbf{x})$ comes from and how to set up the Newton iteration. In particular, we introduce the principle of virtual work and the finite element method as the basic framework for setting up the computation. We formulate the general problem of geometric and material nonlinearities and describe continuation (arc-length) methods as an important extension of Newton's method. Finally, the methods of static analysis are extended to dynamic analysis by applying a Newton iteration on top of a Newmark time integrator, thereby illustrating the essential connection between static and dynamic analysis methods.

1.1 Linearity and Nonlinearity

There are a few basic things one needs to know about nonlinearity. As the name implies, "nonlinearity" is an attribute that describes anything that is not linear. Hence, it is simpler to characterize linearity rather than nonlinearity. It seems pretty clear that one cannot fully appreciate the issues surrounding nonlinear analysis without a solid understanding of linear analysis. Therefore, we shall begin this introductory chapter with a fundamental discussion of linearity. Although most readers are familiar with linear structural analysis, having done it for quite some time, it seems prudent to start at the beginning. Many errors in judgement in nonlinear analysis can be attributed to erroneously applying linear thinking. Indeed, most engineers are educated to have a bias toward linear thinking. This introductory section is, in part, an attempt to help the reader break out of the mold of linear thinking before proceeding to the study of the methods of nonlinear analysis of building structures.

1.1.1 Linear functions and linear equations

Linearity is a property that one can ascribe only to a mathematical model of a structure, not to the structure itself. Consider the case where a structure is subjected to forces in an experiment. Both the applied force f and the displacement x are measured at certain discrete levels. The results of the experiment are plotted as circles in Fig. 1.1. The first important observation is that we really only know the response of the structure at a certain number of discrete points in force–displacement space, as indicated in Fig. 1.1(a). The second observation is that linearity is not a property of the data but a property of

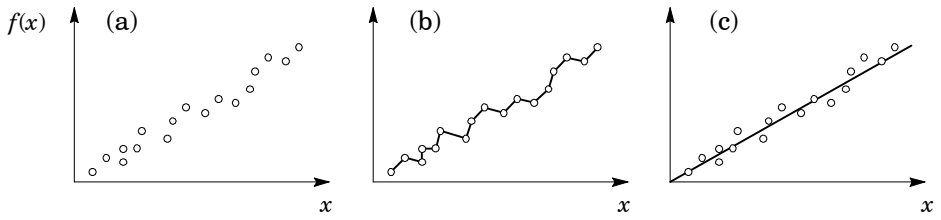

Figure 1.1 Linearity in the realm of measurement (a) measured response, (b) interpolatatory (nonlinear) mathematical model, and (c) linear model.

the mathematical model used to characterize those data. For example, one valid model might be a connect-the-dots type of model that interpolates the data and assumes straight-line behavior between measured points, as shown in Fig. 1.1(b); this model is not linear. Another valid model might be the straight line that splits the difference between the data points, perhaps in a least-squared error sense, as shown in Fig. 1.1(c). The former model assumes that the deviations from linearity can be attributed to the essential behavior of the system while the latter model assumes that the deviations from linearity are not associated with the behavior of the system, but are caused by measurement errors. In both cases, the data are the same.

A real structure can behave linearly only in the sense that its measured response appears to plot along a straight line. Even then, a nonlinearity in the response could be masked by the choice of response quantities plotted. It is often evident, in a qualitative sense, when measured data appear to come from a nonlinear system (i.e., a system that would most appropriately be simulated with a nonlinear mathematical model). Consider, for example, the data shown in Fig. 1.2. Although one might argue that a linear model is appropriate for a limited range of the data, it is clear that these data do not suggest a linear model for the entire range of the data.

A mathematical model of the structure can possess (or fail to possess) linearity in a much more formal sense than can a set of discrete measured data. The following sections define the concept of linearity in the mathematical sense, first for scalar functions

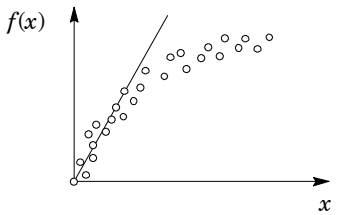

Figure 1.2 When the nonlinearity is strong, one can generally judge when a measured response appears to come from a nonlinear system

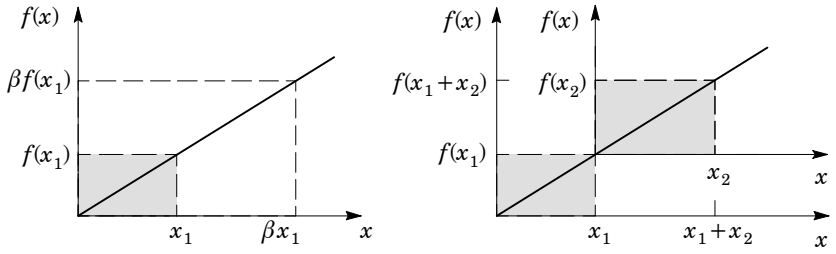

Figure 1.3 Mathematical notion of linearity for univariate functions

of a single variable and then for vector-valued functions of vector-value variables, and discuss the solution of linear equations.

1.1.2 Linearity in one dimension

To solidify our understanding of linearity consider a function that assigns an ordinate $f(x)$ to each abscissa x. The function is *linear* if it has the following properties

$$\boxed{\begin{aligned} f(\beta x_1) &= \beta f(x_1) && \text{(a)} \\ f(x_1+x_2) &= f(x_1) + f(x_2) && \text{(b)} \end{aligned}} \tag{1.2}$$

for any values of β, x_1, and x_2. These properties are shown in Fig. 1.3. Let us view x as the input to the function f and $f(x)$ as the response to that input. Property $(1.2)_{\text{a}}$ suggests that if we scale the input by a certain factor then the output scales by the same factor. Property $(1.2)_{\text{b}}$ suggests that the response to the sum of two inputs is exactly the same as the sum of the responses to the inputs administered separately. This familiar property is often called the *law of superposition*.

It should be clear from Fig. 1.3 that, in order to have these properties, the function $f(x)$ must pass through the origin, that is $f(0) = 0$. This limitation is too restrictive for our purposes, as we shall soon see, and, hence, we must enlarge our understanding of linearity a bit. An *affine* function $g(x)$ is simply a linear function plus a constant, i.e., $g(x) = f(x)+b$. Hence, an affine function does not necessarily pass through the origin. A one-dimensional affine function can be represented by

$$g(x) = ax + b \tag{1.3}$$

where a and b are constants. Figure 1.4 shows a typical affine function in one dimension, as defined by Eq. (1.3). Finding the roots of a univariate affine function is trivial. Setting $g(x) = 0$ we see that the root $x^* = -b/a$, shown in Fig. 1.4, is unique. The only requirement for the existence of a solution is that $a \neq 0$.

Example 1.1. Consider the single-degree-of-freedom system shown in Fig. 1.5. The stiffness of the spring is k and the magnitude of the force is p. The displacement of the system x is measured relative to the unstretched position of the spring.

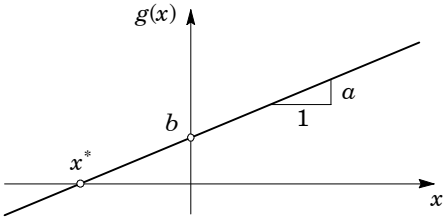

Figure 1.4 A univariate affine function

Consider the case in which the spring exhibits linear response. The internal resisting force that the spring exerts on the box is $f_s = kx$. The system is in equilibrium if the internal resisting force in the spring and the applied force p sum to zero, i.e., if $kx + p = 0$. One can determine the equilibrium configuration of the system to be at the displacement $x^* = -p/k$.

The linear equation $g(x) \equiv kx + p = 0$ has only one root. Therefore, the solution is unique. We can also observe that, if the load p is doubled, then the displacement is doubled. If the stiffness k is doubled, then the displacement is halved. ❑

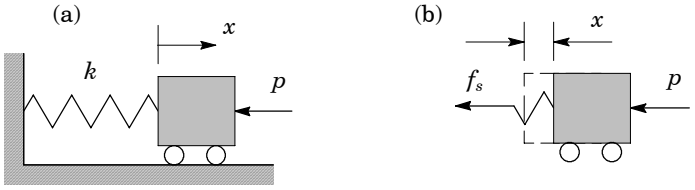

Figure 1.5 Response of a single-degree-of-freedom system (a) description of system, (b) free-body diagram for establishing the equation of equilibrium

1.1.3 Linearity in n dimensions

The concept of linearity carries over exactly to multiple dimensions with a few notational changes. In particular, let $\mathbf{f}(\mathbf{x})$ be a vector valued function of the vector $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}^T$. This function is linear if it satisfies

$$\boxed{\begin{aligned} \mathbf{f}(\beta\mathbf{x}_1) &= \beta\mathbf{f}(\mathbf{x}_1) && \text{(a)} \\ \mathbf{f}(\mathbf{x}_1+\mathbf{x}_2) &= \mathbf{f}(\mathbf{x}_1) + \mathbf{f}(\mathbf{x}_2) && \text{(b)} \end{aligned}} \tag{1.4}$$

for any values of β, $\mathbf{x}_1$, and $\mathbf{x}_2$. The interpretation of linearity in multiple dimensions is exactly the same as in one dimension with the exception that addition of two vectors and multiplication of a vector by a scalar must be done in the sense of matrix algebra. A multi-dimensional affine function can be defined simply as $\mathbf{g}(\mathbf{x}) = \mathbf{f}(\mathbf{x}) + \mathbf{b}$, where

$\mathbf{b} = \{b_1, b_2, \ldots, b_n\}^T$ is a constant vector. Note that $\{\cdot\}^T$ indicates matrix transpose of $\{\cdot\}$. In multiple dimensions, an affine function takes the form

$$\begin{aligned} g_1(\mathbf{x}) &= a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n + b_1 \\ g_2(\mathbf{x}) &= a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n + b_2 \\ &\vdots \qquad\qquad\qquad\qquad\qquad \vdots \\ g_n(\mathbf{x}) &= a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n + b_n \end{aligned} \tag{1.5}$$

This system of linear equations can be compactly represented in matrix notation as

$$\mathbf{g}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b} \tag{1.6}$$

where $\mathbf{b} = \{b_1, b_2, \ldots, b_n\}^T$ is a vector of constants and $\mathbf{x}$ is the independent vector of variables. The matrix $\mathbf{A}$ has components

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix} \tag{1.7}$$

where each a_{ij} is a constant. Note that we adhere to the convention that the first index is the row index and the second index is the column index.

In the context of linear models in structural engineering the equations of linear structural analysis are typically written with the notation

$$\mathbf{K}\mathbf{u} = \mathbf{p} \tag{1.8}$$

where $\mathbf{K}$ is the (constant) stiffness matrix, $\mathbf{u}$ is the vector of unknown nodal displacements and $\mathbf{p}$ is the applied nodal load vector. Hence, we can identify the matrix $\mathbf{A}$ with the linear stiffness matrix, $\mathbf{x}$ with the displacement vector, and $\mathbf{b}$ as the (negative of the) vector of applied nodal loads.

The system of linear equations $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ establishes the requirements of static equilibrium of the system. Hence, the displaced configuration of the structure that satisfies equilibrium is found by solving $\mathbf{A}\mathbf{x} + \mathbf{b} = \mathbf{0}$. To wit,

$$\mathbf{x} = -\mathbf{A}^{-1}\mathbf{b} \tag{1.9}$$

where $(\cdot)^{-1}$ is the matrix inverse of $(\cdot)$.

The linear model gives rise to several important features of response that do not carry over to nonlinear models. In particular,

(a) The solution given by Eq. (1.9), if it exists, is unique. Except in special circumstances (special right-side vectors $\mathbf{b}$), the solution does not exist if the coefficient matrix is not invertible, that is, if $\det \mathbf{A} = 0$.

(b) Let $\mathbf{x}_o$ satisfy the linear system of equations for the right-side vector $\mathbf{b}_o$, that is, $\mathbf{x}_o = -\mathbf{A}^{-1}\mathbf{b}_o$, then $\mathbf{x} = \gamma\mathbf{x}_o$ satisfies the equations for $\mathbf{b} = \gamma\mathbf{b}_o$ for any value of γ.

(c) Let $\mathbf{x}_1$ and $\mathbf{x}_2$ satisfy the linear system of equations for the vectors $\mathbf{b}_1$ and $\mathbf{b}_2$, respectively, that is, $\mathbf{x}_1 = -\mathbf{A}^{-1}\mathbf{b}_1$ and $\mathbf{x}_2 = -\mathbf{A}^{-1}\mathbf{b}_2$. Then the response of the system to $\mathbf{b} = \mathbf{b}_1 + \mathbf{b}_2$ is $\mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2$. This property of linear systems is called the principle of superposition.

In the context of linear structural response, property (a) suggests that, if $\det\mathbf{A} \neq 0$, then there is no other displaced configuration that satisfies the equilibrium equations $\mathbf{g}(\mathbf{x}) = \mathbf{0}$. For a linear structure the condition $\det\mathbf{A} = 0$ implies initial instability—the existence of a mechanism motion that requires no load to develop. Many nonlinear problems also have unique solutions, but uniqueness must be proven on a case by case basis. Property (b) suggests that, if the loads are magnified proportionally by the factor γ, then the responses are also magnified proportionally by the same factor γ. Property (c) suggests that the response to the two loads acting together is the same as the sum of the individual responses under the loads acting independently.

Since nonlinearity is simply the failure to be linear one might expect that there are many equivalent alternative definitions of nonlinearity. A simple and instructive definition is as follows:

1. A nonlinear system is any system for which the principle of superposition does not hold (i.e., systems for which properties (b) and (c) above do not hold).
2. A nonlinear system is any system that cannot be expressed in the canonical linear form $\mathbf{g}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$, where $\mathbf{A}$ and $\mathbf{b}$ do not depend upon $\mathbf{x}$.

1.1.4 Solving a linear system of equations

Symbolically we consider the solution to a linear system of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$ to be $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$. As a practical matter, the inverse of $\mathbf{A}$ is seldom computed explicitly. Rather, the computation associated with Eq. (1.9) is accomplished by any one of several established efficient equation-solving algorithms. Establishing what is the most efficient means of solving a linear system of equations is a rather complicated issue. It depends upon problem size and computer architecture. The two main approaches to solving linear systems of equations are *direct methods* and *iterative methods*.

In a direct method one typically factors the $n \times n$ matrix $\mathbf{A}$ into a product of lower and upper triangular matrices†, i.e., $\mathbf{A} = \mathbf{L}\mathbf{U}$ (Golub and Van Loan 1989, Press et al. 1986). Gaussian elimination is the best known method of carrying out such a factorization. Once the coefficient matrix is factored, the solution can be found by first solving

† An upper-triangular matrix has non-zero values on and above the diagonal and only zeros below the diagonal. A lower-triangular matrix has non-zero values on and below the diagonal and only zeros above the diagonal.

the equations $\mathbf{Ly} = \mathbf{b}$ and subsequently solving $\mathbf{Ux} = \mathbf{y}$. Because the matrices $\mathbf{L}$ and $\mathbf{U}$ are triangular, the two new equations can be solved very quickly. The storage required to save $\mathbf{L}$ and $\mathbf{U}$ is exactly the same as that required to save $\mathbf{A}$ because the profile of the original matrix is preserved by the subsequent operations. The step $\mathbf{Ly} = \mathbf{b}$ is often called *forward reduction* and can be done at the same time the matrix is being factored. The step $\mathbf{Ux} = \mathbf{y}$ is often called *back-substitution*. In linear analysis, where $\mathbf{A}$ is constant, direct factorization methods are popular because, once the matrix is factored, forward reduction and back-substitution can be repeated for many different right-side vectors $\mathbf{b}$. For structural analysis problems with several different load vectors, this opportunity is clearly advantageous. If the matrix $\mathbf{A}$ is symmetric, it can be factored as $\mathbf{A} = \mathbf{LL}^T$, also know as the Cholesky factorization. Matrix factorization takes approximately nh^2 operations, where h is the bandwidth of the matrix. Forward reduction and back-substitution together take approximately nh operations.

An iterative method avoids the factorization of $\mathbf{A}$ and can often be implemented in such a way that even the formation of the global matrix $\mathbf{A}$ can be avoided (for example, in favor of computations of a residual vector at the element level). Iterative methods are competitive with direct methods when the problem size is very large or when parallel computations are possible. There are two basic varieties of iterative equation solvers: fixed-point methods and search direction methods.

A fixed-point iteration method solves the equation $\mathbf{Ax} = \mathbf{b}$ by improving an initial estimate of the solution $\mathbf{x}_o$ through the iterative sequence

$$\mathbf{D}\mathbf{x}_{i+1} = \mathbf{b} - \mathbf{C}\mathbf{x}_i \tag{1.10}$$

where $\mathbf{D}$ and $\mathbf{C}$ can be computed from the coefficient matrix $\mathbf{A}$. Such a method is convenient if the matrices $\mathbf{D}$ and $\mathbf{C}$ can be computed from $\mathbf{A}$ without much effort and the matrix $\mathbf{D}$ can be inverted without much effort (choosing $\mathbf{D}$ to be a diagonal matrix, for example, achieves this goal). The Jacobi method and the Gauss-Seidel method are two of the best known examples of fixed-point iterative methods.

A search direction method seeks to improve an estimate of the solution as

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{d}_i \tag{1.11}$$

where $\mathbf{d}_i$ is the search direction and α_i is the amount of movement in that direction. The most popular search direction method is the method of *conjugate gradients*, generally used with a pre-conditioner. The details of how to compute with these methods can be found in Golub and Van Loan (1989). As with any iterative method one must specify $\mathbf{x}_o$ and the termination tolerance. Iteration terminates when $\| \mathbf{x}_{i+1} - \mathbf{x}_i \| < tol$.

1.2 Nonlinear functions and nonlinear equations

When one opens the door to nonlinear models one finds an endless variety and a raft of computational problems that are not present in linear analysis. Figure 1.6 shows a

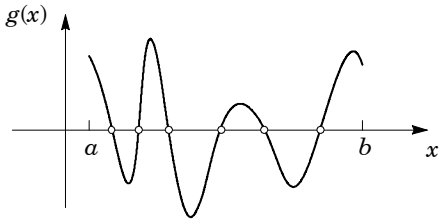

Figure 1.6 A nonlinear function

nonlinear function of a single variable $g(x)$, defined on the region $x \in [a, b]$. In the context of finding the roots of the function by solving the equation $g(x) = 0$, there are two key features of the nonlinear function that are important:

(a) Unlike the linear equation, the solution is not necessarily unique. For the function pictured in Fig. 1.6, there are six roots that satisfy $g(x) = 0$ within the region of definition of the function, and

(b) Unlike the linear equation, there is no computation procedure available to directly find the roots (with a few notable exceptions, like the quadratic formula for solving $ax^2 + bx + c = 0$ or some that are known by inspection such as $\sin x = 0$).

Failure of uniqueness is common in static structural analysis because many structures exhibit limit load response. Consider the response of a structural system (applied force p plotted against displacement u) shown in Fig. 1.7. For the load level p_1 there are two values of displacement, u_1 and u_2, that provide equilibrating configurations. When the solution to a problem is not unique we face several important questions: How many solutions are there? What are those solutions? Do I need to compute all solutions? Are all of the solutions relevant to the current problem or are some of them artifacts of the mathematical model? What particular strategy should I use to find the solution?

The problem of finding a solution to a nonlinear equation presents certain difficulties that the linear equation does not. Since direct solution methods are available only for very special classes of problems we shall abandon that approach here in favor of ap-

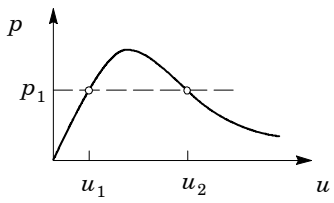

Figure 1.7 An example of lack of uniqueness of solution for structural response

proaches that are more likely to yield solutions in general. It is fair to say that the only general approach to solving nonlinear equations is to use iterative methods.

In an iterative method, one first selects a candidate solution x_o. Having a known state, one can easily check to see if it satisfies the equation (i.e., compute $g(x_o)$ and check to see if it is zero). Rarely will an estimate exactly satisfy the equation, so one must decide if the estimate is close enough. A solution will generally be deemed "close enough" if $\| g(x_o) \| < \epsilon$, where ϵ is a small positive number, selected by the analyst in advance, called the *solution tolerance*.† If the estimated solution is not close enough then an improved estimate is made based on some knowledge of the nonlinear function in the neighborhood of x_o. Clearly, with a new estimate, we can repeat the whole process again. An iterative method is born.

Iterative methods come in great variety. They differ in the starting method (i.e., how to establish x_o), the termination criteria (i.e., how to select ϵ), and, most importantly, the approach to computing an improved estimate of the solution from the existing estimate. In this chapter we will focus on one of these methods, the Newton-Raphson method, as both typical of many available iterative methods and arguably the best among those for problems in nonlinear structural mechanics.

1.2.1 Behavior of nonlinear solution schemes

If you were to suddenly put a load on a real structure that is much greater than its capacity, it would certainly collapse. It might groan and shudder briefly, giving the impression that it is gamely trying to hold up the load, but it would soon fall. Nonlinear structural analysis schemes are somewhat like nature in this regard. If you attempt to impose too great a load too suddenly, the algorithm is likely to fail. It may gamely try to execute a few iterations, giving the impression that it is trying to solve the problem, but it will almost certainly diverge. Contrast this algorithmic behavior with linear structural analysis, where the size of the load applied to the structure has no impact on the solution algorithm.

In nonlinear analysis, the manner in which the sequence of load application is modeled can have a tremendous impact on the success or failure of the solution algorithm. For nonlinear problems, the load generally must be applied in steps small enough to guarantee convergence of the iterative algorithm. Upon convergence in any given load step, the load can be further increased. Load incrementation is continued until the total target load is applied.

If the structure possesses a limit load (a maximum load carrying capacity) a load incrementation scheme, no matter how finely divided, is doomed to failure if the applied load is eventually greater than the capacity of the structure. In such an instance, a non-

† The notation $\| \cdot \|$ indicates a measure of length of the argument. For the scalar function, the absolute value is the appropriate measure. For a vector valued function, the Euclidean norm of the vector is the most widely used measure.

linear solution algorithm is very much like nature itself in that the request is met with an undesirable response. Nonlinear analysis forces us to better understand the response of the structure. Our algorithms must recognize certain natural features of the response and accommodate them. If we were testing a structure that we suspected would exhibit a limit load with unstable post-limit response in a laboratory, we would certainly use displacement-controlled actuators. Likewise in a nonlinear analysis of such a structure, we must use an algorithm that is suited to passing a limit point.

Not all algorithmic failures are associated with natural phenomena, however. The mathematical model may have features that lead to convergence difficulties even in a well behaved structure. In the past, there was a tendency among practitioners of nonlinear analysis to equate algorithmic failure with structural failure. When a nonlinear solution algorithm experiences difficulties it is prudent to suspect that a physical phenomenon may be responsible. However, algorithmic anomalies alone should never be taken as the sole evidence of physical anomalies.

1.3 Linearization of nonlinear functions

The concept of *linearization* plays a key role in nonlinear analysis. As we shall see in subsequent sections, at the heart of a nonlinear analysis algorithm is a linear analysis. Linearization helps us identify what linear problem we should be solving at each step and how to combine that result with the knowledge of the solution obtain up to that point. The mathematical idea of linearization relies on the Taylor series expansion of a function. A one-dimensional nonlinear function $g(x)$ can be expanded as a Taylor series, about the point x_o as

$$g(x) \;=\; g(x_o) \;+\; (x-x_o)g'(x_o) \;+\; \tfrac{1}{2}(x-x_o)^2 g''(x_o) \;+\; \ldots \tag{1.12}$$

where $g(x_o)$, $g'(x_o)$, and $g''(x_o)$ are the function, its first derivative, and its second derivative evaluated at the point x_o. The Taylor series expansion of a function is accurate in the neighborhood of the point x_o and starts to deviate from the function for points remote from x_o. When we linearize a function we simply keep the constant and linear terms of the Taylor series expansion. Thus, we can define† a linear function $\hat{g}(x)$ that is closest to $g(x)$ in the neighborhood of x_o

$$\hat{g}(x) \;\equiv\; g(x_o) \;+\; (x-x_o)g'(x_o) \tag{1.13}$$

The relationship between the original function and the linear function is shown in Fig. 1.8 for a typical univariate function. The linear function is the straight line tangent to the curve at the point x_o. How well the linear function approximates the nonlinear one

† The notation $\equiv$ means "is defined as" rather than "is equal to," which is denoted with an ordinary equal sign (i.e., the symbol $=$). The ordinary equal sign implies that the result can be derived from other principles, whereas the definition equal sign does not imply a derived result. This convention will be used throughout the book.

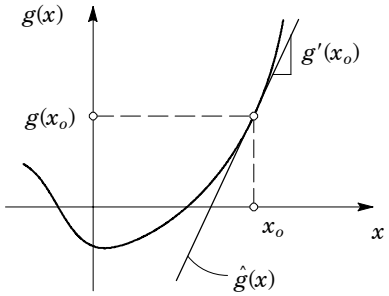

Figure 1.8 Linearization of a univariate function

depends upon the truncated terms in the Taylor series expansion. The next dominant term involves the second derivative of the function $g''(x_o)$. Geometrically, $g''(x_o)$ is roughly proportional to the curvature of $g(x)$. The larger $g''(x_o)$ is, the sooner the nonlinear curve departs from the linear one.

The associated linear function $\hat{g}(x)$ will play a key role in the solution of the equation $g(x) = 0$ as we shall soon see. The primary advantage to having identified an associated linear function is that we can use all the machinery of linear analysis available to us in working with this function.

The idea of linearization is easily extended to n dimension by noting that the Taylor series expansion for a function $\mathbf{g}(\mathbf{x})$ about the point $\mathbf{x}_o$ is given by

$$\mathbf{g}(\mathbf{x}) \;=\; \mathbf{g}(\mathbf{x}_o) \;+\; \nabla\mathbf{g}(\mathbf{x}_o)\big(\mathbf{x}-\mathbf{x}_o\big) \;+\; \ldots \tag{1.14}$$

where $\nabla\mathbf{g}$ is the matrix with components $[\nabla\mathbf{g}]_{ij} \equiv \partial g_i/\partial x_j$, the gradient of the function $\mathbf{g}(\mathbf{x})$. The matrix $\nabla\mathbf{g}$ will appear throughout this chapter. To economize the notation we shall designate it with the symbol $\mathbf{A}$ as follows

$$\mathbf{A}(\mathbf{x}) \;\equiv\; \nabla\mathbf{g}(\mathbf{x}) \tag{1.15}$$

The matrix $\mathbf{A}$ is often called the *tangent matrix* in analogy with the one-dimensional case where $g'(x_o)$ gives the slope of the line tangent to the curve at the point x_o. In the context of displacement-based structural analysis, $\mathbf{A}$ is often called the *tangent stiffness matrix* because, like the ordinary linear stiffness matrix $\mathbf{K}$, it represents the ratio between (an increment in) force and (an increment in) displacement. A linear function associated with $\mathbf{g}(\mathbf{x})$ in the neighborhood of $\mathbf{x}_o$ can be defined as follows

$$\hat{\mathbf{g}}(\mathbf{x}) \;\equiv\; \mathbf{g}(\mathbf{x}_o) \;+\; \mathbf{A}(\mathbf{x}_o)\big(\mathbf{x}-\mathbf{x}_o\big) \tag{1.16}$$

Again, the linear function $\hat{\mathbf{g}}(\mathbf{x})$ is much easier to work with than the original nonlinear function.

1.4 Iterative solution methods—Newton's method†

Except for some fortuitous circumstances, the solution of nonlinear problems is not possible in a closed form calculation. In general, nonlinear computations are done by iterative methods wherein one takes an estimate of the solution and improves on that estimate through solving a subsidiary (linear) problem. The solution is obtained when the current estimate satisfies the given nonlinear equation well enough.

Newton's method is one of the most effective iterative nonlinear solution strategies. The method is based on developing and solving a sequence of linear problems through the process of *linearization* discussed in the previous section. Newton's method is one of several iterative approaches to the solution of nonlinear algebraic equations. By and large, other methods can be viewed as derivative of Newton's. We shall discuss some of the variants of Newton's method in a subsequent section.

1.4.1 Newton's method in one dimension

The basic idea behind Newton's method in one dimension is as follows. Rather than trying to solve the nonlinear equation $g(x) = 0$ we linearize the equation about the point x_o and solve the linear equation $\hat{g}(x) = 0$ instead. From Eq. (1.13) we get

$$x = x_o - \frac{g(x_o)}{g'(x_o)} \tag{1.17}$$

Let us call this solution x_1. This point, which is simply where the linear function crosses the axis, is shown in Fig. 1.9. While x_1 does not satisfy the original nonlinear equation, i.e., $g(x_1) \neq 0$, it comes closer to satisfying it than x_o did. With the new candidate estimate x_1 we can repeat the process of linearization and solving $\hat{g}(x) = 0$ to get a new estimate x_2. The iterative process can be summarized in the equation

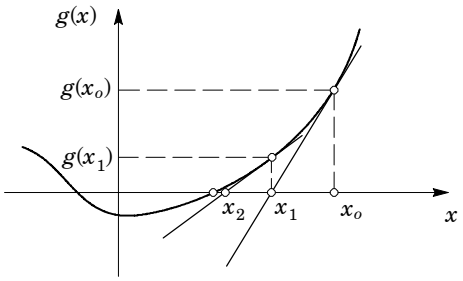

Figure 1.9 Newton's method for a univariate function

† Newton's method is often referred to as the Newton-Raphson method, acknowledging the contribution of Raphson who, by all accounts, was responsible for expressing Newton's method in the form that we use it today. In the interest of keeping the names of things brief we will simply call the method Newton's method.

$$\boxed{x_{i+1} = x_i - \frac{g(x_i)}{g'(x_i)}} \tag{1.18}$$

This iterative formula suggests that we can compute a new estimate x_{i+1} from knowledge of the function and its first derivative evaluated at the point x_i. The starting value x_o must be specified, it cannot be computed. The iteration can be repeated until satisfactory results are obtained. The definition of "satisfactory" must also be specified a priori, it cannot be determined from the computation. Since our aim is to solve $g(x) = 0$ the best criterion for convergence will consider how well that equation is satisfied. We can consider the iteration converged for the state x_n if

$$|\, g(x_n) \,| < tol \tag{1.19}$$

where *tol* is a value specified a priori that quantifies close enough.

An alternative convergence criterion is the Cauchy criterion. If $|\, x_n - x_{n-1} \,| < \epsilon$, where ϵ is a preset tolerance, then the iteration is no longer making progress and x_n can be considered converged. Although this criterion is often used, it is not a very good one. It is possible to have two adjacent iterates close together by accident somewhere in the middle of an iteration sequence. One can reduce the probability of identifying these bogus converged states by insisting that three or more consecutive iterates satisfy the Cauchy criterion, but such ad hoc termination criteria are not fail-safe.

Newton's method converges quite quickly in the neighborhood of the solution. In fact, the convergence is quadratic, meaning that

$$|\, g(x_{i+1}) \,| \approx c\big[g(x_i)\big]^2 \tag{1.20}$$

where c is a constant that is independent of the iteration counter i. Quadratic convergence is evident when the order of magnitude of the residual is halved in each successive iteration.

1.4.2 Behavior of Newton's method

Newton's method will fail at any point where $g'(x) = 0$ because g' appears as a divisor in Eq. (1.18). The algorithm is not particularly attracted to these points, but this feature is a cause for concern. In the neighborhood of such points $g'(x)$ is very small and Eq. (1.18) is likely to predict a large increment $\Delta x = g(x_i)/g'(x_i)$. With a large increment, the next estimate will likely be far from the region of accuracy of the former linear equation and divergence of the iteration may ensue.

There are other conditions in which Newton's method will diverge. One rather harmless looking function that can cause the algorithm to diverge is shown in Fig. 1.10. For the starting point x_o, the iterates bounce around the root x^*, but get increasingly far from it. Fortunately, there are starting points that will converge. The *region of convergence* (also known as the *basin of attraction*) is shown shaded in the figure. Every point within this region converges and every point outside of this region diverges.

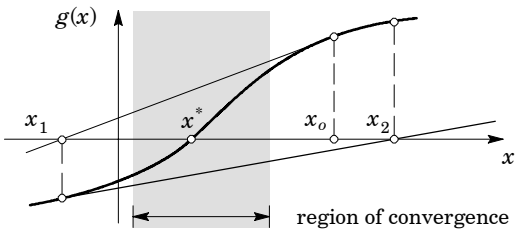

Figure 1.10 Divergence of Newton's method

Divergence of Newton's method can be avoided in a number of ways. The solution is always contained in a basin of attraction. Generally, that basin is a finite region (i.e., contains more than just the solution point). Thus, one strategy for forcing convergence of a Newton iteration is to select the starting point to lie inside the basin of attraction. Although for the general problem of root finding, such a strategy is not really possible, it will be for the problem of finding equilibrium states of structures by load incrementation because we generally start at a known equilibrium point (e.g., the undeformed, unloaded configuration). By selecting small enough increments of load one can virtually guarantee that the next step lies within the basin of attraction. This issue will be discussed further in a subsequent section.

A more general approach to guaranteeing convergence of a Newton iteration is by insisting that the value of the function decrease at each step, i.e.,

$$| g(x_{i+1}) | \;<\; | g(x_i) | \tag{1.21}$$

There are many strategies for achieving descent in every step, although some of them are rather complex.

1.4.3 Newton's method in n-dimensions

While the one-dimensional version of Newton's method provides a good overview of the features and foibles of the method, we are primarily concerned with nonlinear computations in n dimensions. As with the one-dimensional version, we shall solve the associated linear problem $\hat{\mathbf{g}}(\mathbf{x}) = \mathbf{0}$, where the linear function $\hat{\mathbf{g}}(\mathbf{x})$ is given by Eq. (1.16), rather than trying to solve the original nonlinear problem $\mathbf{g}(\mathbf{x}) = \mathbf{0}$. Setting $\hat{\mathbf{g}}(\mathbf{x}) = \mathbf{0}$ and solving for the state $\mathbf{x}$ we find a formula analogous to Eq. (1.17)

$$\mathbf{x} \;=\; \mathbf{x}_o - \left[\mathbf{A}(\mathbf{x}_o)\right]^{-1} \mathbf{g}(\mathbf{x}_o) \tag{1.22}$$

Note that $\mathbf{g}(\mathbf{x}_o)$ and $\mathbf{A}(\mathbf{x}_o)$ represent the function and the tangent matrix evaluated at the point $\mathbf{x}_o$. The new estimate $\mathbf{x}$ should be better than the old one $\mathbf{x}_o$. The function and its partial derivatives can now be computed at the new estimate and a new estimate can be computed from Eq. (1.22). This iterative improvement of the initial guess can be summarized in the equation

$$\boxed{\mathbf{x}_{i+1} = \mathbf{x}_i - \left[\mathbf{A}(\mathbf{x}_i)\right]^{-1}\mathbf{g}(\mathbf{x}_i)} \tag{1.23}$$

where $\mathbf{x}_i$ is the known point from the previous iteration and $\mathbf{x}_{i+1}$ is the next estimate. The iteration can be continued until

$$\| \mathbf{g}(\mathbf{x}_n) \| < tol \tag{1.24}$$

where $\| \mathbf{g} \|$ is the Euclidean norm of the vector $\mathbf{g}$ (square-root of the sum of the squares of all of the components). If Eq. (1.24) is satisfied, then $\mathbf{x}_n$ is the converged state and satisfies the original nonlinear equation to within the tolerance *tol*. Like the one-dimensional method, the convergence is quadratic in the neighborhood of the solution. Also like the one-dimensional version, the method fails if the matrix $\mathbf{A}(\mathbf{x})$ is singular (i.e., cannot be inverted).

Matrix inversion is usually not the most efficient approach to the solution of linear equations. Therefore, the Newton algorithm is more commonly expressed in two steps

$$\boxed{\begin{aligned} \mathbf{A}(\mathbf{x}_i)\Delta\mathbf{x}_i &= -\mathbf{g}(\mathbf{x}_i) \\ \mathbf{x}_{i+1} &= \mathbf{x}_i + \Delta\mathbf{x}_i \end{aligned}} \tag{1.25}$$

The first step is the solution of a linear system of equations, which can be accomplished with an efficient algorithm. The second step is the update.

1.5 Variations and extensions of Newton's method

One of the main computational expenses in Newton's method is the solution of the linear system of equations $\mathbf{A}_i\Delta\mathbf{x}_i + \mathbf{b}_i = \mathbf{0}$ at each iteration, where $\mathbf{A}_i \equiv \mathbf{A}(\mathbf{x}_i)$ and $\mathbf{b}_i \equiv \mathbf{b}(\mathbf{x}_i)$. For a direct solution method, we must form and factor $\mathbf{A}_i$ and form, forward reduce, and back-substitute $\mathbf{b}_i$ at each iteration. The formation and factorization of $\mathbf{A}_i$ is the largest computational task. Methods that aim to save on this cost have been formulated and implemented in nonlinear analysis codes. The basic ideas behind these methods are described in the following paragraphs.

1.5.1 Modified Newton

The simplest modification of Newton's method replaces $\mathbf{A}_i$ with $\mathbf{A}_o$ at each iteration. If the coefficient matrix does not change from one iteration to the next it can be formed and factored only once. Certainly, the elimination of the factorization of each new tangent matrix represents a tremendous computational savings. The idea, cast in one dimension, is shown in Fig. 1.11. Each new version of the linearized function has the same slope while the intercept is modified at each iteration. The hope is that the slope repre-

sented by $\mathbf{A}_o$ is good enough, and that the algorithm will converge in a reasonable number of iterations. One can see from Fig. 1.11 that the rate of convergence of the modified Newton method will be much slower than original Newton. In fact, the rate of convergence is linear rather than quadratic. This change may increase the number of iterations required to be an order of magnitude larger for modified Newton over original Newton. Consequently, much of the savings realized in avoiding the factorizations of $\mathbf{A}_i$ are slowly squandered via forward reductions and back-substitutions in the extra iterations. It is seldom clear which algorithm is fastest.

The modified Newton strategy can be further modified to form and factor a true tangent $\mathbf{A}_i$ every m iterations. This strategy ameliorates a bad tangent associated with a poor starting point $\mathbf{x}_o$ and enhances the overall rate of convergence without going all the way to the level of effort implicit in original Newton. Again, the best value of m is seldom evident.

1.5.2 Quasi-Newton methods

For some problems the tangent matrix $\mathbf{A}_i \equiv \nabla\mathbf{g}(\mathbf{x}_i)$ can be difficult or costly to explicitly compute. Quasi-Newton methods attempt to form an approximation to the tangent matrix by processing information about the function $\mathbf{g}(\mathbf{x})$ itself collected at the iteration points. Indeed, for a linear problem, in which $\mathbf{A}_i$ is constant, one can show that the tangent matrix can be constructed from the function evaluated at n points. The most popular of the methods for constructing the tangent is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method. Let $\mathbf{s}_k \equiv \mathbf{x}_{k+1} - \boldsymbol{x}_k$ be the difference in solution estimates at the k and $k+1$th iterations, and let $\mathbf{y}_k \equiv \mathbf{g}(\mathbf{x}_{k+1}) - \mathbf{g}(\mathbf{x}_k)$ be the difference in the function at those two points. An improved estimate for the tangent matrix for the next iteration can be computed as

$$\mathbf{A}_{k+1} = \mathbf{A}_k - \frac{1}{\alpha}\mathbf{A}_k\mathbf{s}_k\mathbf{s}_k^T\mathbf{A}_k + \frac{1}{\beta}\mathbf{y}_k\mathbf{y}_k^T \tag{1.26}$$

where $\alpha \equiv \mathbf{s}_k^T\mathbf{A}_k\mathbf{s}_k$ and $\beta \equiv \mathbf{y}_k^T\mathbf{s}_k$. This formula is a rank-two update of the matrix $\mathbf{A}_k$.

The quasi-Newton updates can be started with $\mathbf{A}_o = \mathbf{I}$ in the absence of better information. Experience has shown that the BFGS update is most efficiently and robustly

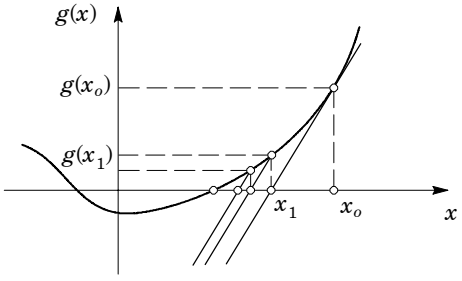

Figure 1.11 Modified Newton's method for a univariate function

implemented as an update of the Cholesky factors of $\mathbf{A}_k$ rather than $\mathbf{A}_k$ itself (Gill, Murray, and Wright 1981). For nonlinear problems it is often a good idea to occasionally restart with a fresh $\mathbf{A}_k$ rather than continue to use an updated version. The reason for the restart is that the updated matrix remembers all of the information it has gathered. Since the exact tangent matrix changes from iteration to iteration, information gathered several iterations ago may have little relevance to the current estimate of the tangent. Restarting can be viewed as purging old information that still influences the estimated tangent matrix.

There are many other methods available that avoid the explicit computation of the tangent matrix. Among those is the nonlinear conjugate gradient method (Luenberger 1984).

1.5.3 Line Searches

The update portion of most iterative strategies for solving nonlinear problems, including Newton's method, can be cast in the form

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k \tag{1.27}$$

The vector $\mathbf{d}_k$ is called the *search direction* and the scalar α_k is the amount to move in the search direction. The search direction for Newton's method is simply the increment $\Delta\mathbf{x}_k$ computed from Eq. (1.25). Therefore, in the original Newton method the search direction is determined as $\mathbf{d}_k = -[\mathbf{A}(\boldsymbol{x}_k)]^{-1}\mathbf{g}(\mathbf{x}_k)$ and we can observe that $\alpha_k = 1$.

The derivation of Newton's method, using the Taylor series expansion, did not suggest changing α_k at each iteration. However, we are also aware that Newton's method does not always converge from an arbitrary starting point. One way to guarantee convergence of Newton's method is to add a *line search*. The idea behind the line search is that, upon determining a good direction $\mathbf{d}_k$ in which to search for the next estimate of our solution point, we still have the freedom to search for the best distance to travel in that direction.

Our goal is to find the solution to $\mathbf{g}(\mathbf{x}) = \mathbf{0}$. Newton's method suggests that the linear estimate $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k$, where $\mathbf{d}_k = -[\mathbf{A}(\boldsymbol{x}_k)]^{-1}\mathbf{g}(\mathbf{x}_k)$, is better than $\mathbf{x}_k$ at satisfying this equation, but does not generally satisfy it. With the direction fixed we can ask the subsidiary question: If $\mathbf{g}(\mathbf{x}_k + \mathbf{d}_k) \neq \mathbf{0}$ is there a value of α_k that minimizes the value of the function in this direction? The associated one-dimensional minimization problem is then

$$\min_{\alpha} \quad \| \mathbf{g}(\mathbf{x}_k + \alpha \mathbf{d}_k) \|^2 \tag{1.28}$$

The solution to this minimization problem is the sought-after α_k. Once this value of the line-search parameter is found, the update implied in Eq. (1.27) can be completed. The minimization problem of Eq. (1.28) is a nonlinear problem itself and can be solved with any of several approaches for minimizing univariate functions (see, for example, Gill et al. 1981). The minimum satisfies the following necessary condition

$$\mathbf{d}_k^T \mathbf{A}(\mathbf{x}_k + \alpha \mathbf{d}_k)\, \mathbf{g}(\mathbf{x}_k + \alpha \mathbf{d}_k) \;=\; 0 \tag{1.29}$$

The minimizing value of α is typically found by an algorithm that brackets the solution and then subdivides the region until a solution is found (Luenberger 1984). The necessary conditions given by Eq. (1.29) are generally not used in the solution because it requires the formation of the tangent matrix for each new estimate of α. Zeroth-order methods for minimizing a scalar function can be applied directly to the problem of Eq. (1.28), thereby requiring only evaluations of the function $\mathbf{g}(\mathbf{x}_k + \alpha \mathbf{d}_k)$ for different values of α.

A popular alternative to minimizing the norm of the residual is to find the value of α for which the residual is most nearly orthogonal to the search direction $\mathbf{d}_k$. This line search can be formulated as (Zienkiewicz and Taylor 1991, page 662)

$$\min_{\alpha} \; \left| \, \mathbf{d}_k^T \mathbf{g}(\mathbf{x}_k + \alpha \mathbf{d}_k) \, \right| \tag{1.30}$$

Again, the minimization can be done with an efficient univariate minimization routine. The modified line search equation requires only that the residual function be evaluated for various values of the line search parameter and does not require the evaluation of the tangent matrix $\mathbf{A}$ at each line-search iteration. Note that some authors (see, for example, Crisfield 1991) suggest that Eq. (1.30) can be stated as $\mathbf{d}_k^T \mathbf{g}(\mathbf{x}_k + \alpha \mathbf{d}_k) = 0$. However, there is no guarantee that there exists a value of α that satisfies this equation.

The line search is a relatively costly operation so most nonlinear codes provide it as an option that one can turn on in times of trouble rather than having it as a standard step in the solution scheme.

1.6 Sources of nonlinearity in structural mechanics

Nonlinearity in structural analysis arises from three basic phenomena: (1) the relationship between stress and strain in the material may be nonlinear, (2) the relationship between strain and displacement may be nonlinear, and (3) there may be unilateral contact between two bodies. The first class of nonlinearities are generally referred to as *material* (or constitutive) *nonlinearities*. There is considerable variety to constitutive nonlinearities, including nonlinear elastic response and inelastic response. For building structures inelastic response is the primary source of material nonlinearity. Inelasticity can be characterized by a yield surface that describes the maximum sustainable elastic states of stress and a flow rule that describes how plastic strains accrue in the yielding process as the material deforms beyond the elastic limit. A general framework for inelasticity will be discussed in the next chapter.

Nonlinear effects that accrue from geometric features of the deformation of a structure are often called *geometric nonlinearities*. These nonlinearities give rise to buckling phenomena and generally arise from two basic considerations. First, when the equations of equilibrium are written in the deformed configuration of the structure, the position of the loads can change with the deformation and give rise to magnification effects

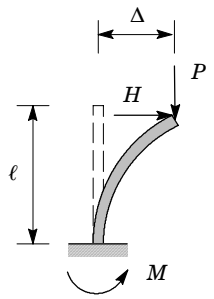

Figure 1.12 Illustration of geometric effects

like those shown in Fig. 1.12. The moment at the base of the structure is $M \approx H\ell + P\Delta$, the second term of which gives the primary geometric effect (also known as the $P-\Delta$ effect). Second, the relationships between strain and displacement can be nonlinear. For example, the net axial strain in a planar beam can be modeled (to a second order approximation) as

$$\epsilon \;=\; u' \,+\, \tfrac{1}{2}\big(w'\big)^2 \tag{1.31}$$

where u and w are the axial and transverse displacements and a prime denotes differentiation with respect to the axial coordinate. The second term is nonlinear and is generally responsible for giving rise to buckling in the model. A careful analysis will reveal that establishing equilibrium in the deformed configuration is not really independent of the nonlinear strain-displacement relationship. They are always related in a consistent theory of structural mechanics. This relationship manifests in the principle of virtual work through the proper expression of the internal virtual work.

In many structures we are concerned with both material and geometric nonlinearities at the same time. Inelastic buckling of steel members is a case in point. As such, the distinction between the two types of nonlinearities is largely of academic interest.

Unilateral contact problems are nonlinear primarily because the contact region generally depends upon the contact mechanism (e.g., the relative compliance of the two bodies in contact), which is not known in advance and must be determined as part of the solution. Unilateral contact is probably the least prevalent nonlinear phenomenon in building structures, but can play an important role, for example, in pounding of adjacent structures, particularly under dynamic excitation, and in the behavior of connections where, after some deformation, two disjoint pieces of the connection come into contact with each other. Crack opening and closure in concrete and masonry (particularly at the joints) are two other important examples where unilateral contact is important. A general treatment of unilateral contact problems is beyond the scope of this book.

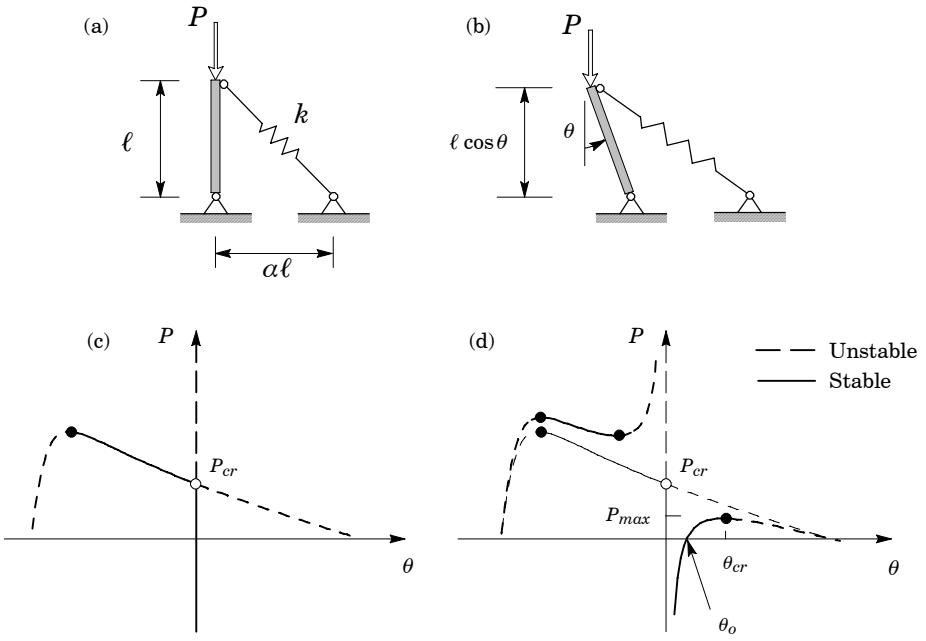

Figure 1.13 Response of rigid column with an elastic guy (a) undeformed geometry of the structure, (b) deformed geometry, (c) response with no initial geometric imperfections, and (d) response with initial geometric imperfection

1.7 Stability of equilibrium

Nonlinear response of a structure is often deeper than a simple departure from linearity. Many unusual things can happen and the analyst must be aware of them. Consider, for example, the rigid column support by the elastic guy shown in Fig. 1.13 with load-deformation response curves both for a system with no initial geometric imperfection and one with an initial imperfection of θ_o. The system is in equilibrium with the applied load for any state designated with a solid or dashed line. As such these lines represent equilibrium paths (load-deformation curves) of the structure. The solid lines represent stable equilibria while the dotted lines represent unstable equilibria.

The structure with no initial geometric imperfections is in equilibrium in the straight configuration for any value of the applied load P. However, the straight configuration is not stable for loads above P_{cr}. The open circle at P_{cr} denotes a *bifurcation point*. A bifurcation point is a state where multiple equilibrium paths cross. The equilibrium path branches or bifurcates at these points. In this case the alternative equilibrium path represent buckling to a bent position (i.e., $\theta \neq 0$). If the structure buckles to the left (positive values of θ) it is unstable and sheds load; to the right (negative values of θ) it is stable (at least for a while) with an ascending post-buckling response. The solid circle on the post-buckling curve indicates a *limit point*. A limit point is a point where

the system goes from stable equilibrium to unstable equilibrium and is a point where the load capacity achieves an extreme value.

If the structure has an initial geometric imperfection then the behavior is substantially different. The response curve shown is for a positive value of θ_o. The structure initially has a stable loading curve, but exhibits a limit load (the solid circle at θ_{cr}) with unstable post-limit response. The maximum capacity of the system P_{max} is substantially lower than the bifurcation load of the perfect system. Since the post-limit equilibrium path is unstable the structure would be inclined to snap to the nearest stable equilibrium configuration. A static nonlinear analysis gives little information relevant to the dynamic snap-through buckling process.

Imperfections can manifest in many ways, including the initial geometry, the loading, and the material properties. Perfect structures do not exist in nature, but are possible in computer simulations with mathematical models. Imperfections can have infinite variety and are often described only in some statistical sense. The response of a perfect structure has interest primarily because it does not require the specification of the poorly understood imperfections. The perfect structure provides a backbone response that can be of some use in engineering. However, as the previous example shows, one must be careful in using the results of an analysis of a perfect system in the estimation of the capacity of a structure (i.e., the critical load P_{cr} can be a poor and unconservative estimate of the capacity P_{max}).

In a computer simulation with a mathematical model one should be aware that most analysis programs do not look for bifurcation points and feel no guilt at following a unstable equilibrium path. For example, if a straight column is subjected to a perfectly placed axial load the simulation will most likely allow increases in the axial load with only shortening of the column taking place. Only if given a small imperfection, will the simulation show buckling behavior.

The simple example serves to illustrate the variety and complexity that one can expect in nonlinear problems. Our goal in static nonlinear analysis is to trace the equilibrium paths of a structure starting from some known state (usually the unloaded configuration). The unusual features described in this section are part of what we wish to discover about the response of the structure. For building structures we are often interested in estimating the capacity of a structure. Hence, we want to find the first limit point that the structure encounters upon loading from the undeformed state. The post-limit response is also of interest because it gives an indication of how quickly the structure will shed load if pushed beyond the limit load.

Many of the features of nonlinear static response are not relevant in a dynamic model. For example, there is no such thing as a limit load in dynamic response because the structure can transfer its energy to inertial and damping forces (El Naschie 1990). For building structures the energy dissipation associated with inelastic response is an important aspect of nonlinear dynamic analysis. Ironically, the procedures for static and

dynamic nonlinear analysis of structures is quite similar, as the next chapter demonstrates.

1.8 Truss structures—A simple nonlinear structural theory

We close this chapter with a simple introduction to nonlinear structural response. The truss is the one structural element that can be treated completely without appealing to differential equations or functionals. All of the equations that govern the response of the truss are discrete, and hence fit naturally with the ideas developed in this chapter. The basic framework that we present here imitates the framework of all nonlinear structural mechanics theories. In that sense, this simple theory can be viewed as a template for most of the more complicated cases. The connection between the truss and the more general problems of solid and structural mechanics should become evident in the next chapter.

Definition of structure. Consider a three dimensional truss structure, like the one shown (albeit as a planar example) in Fig. 1.14(a), comprised of n joints and m members, each of which exhibits a nonlinear force-deformation response when stretched. Let the nodal force applied to joint i of the truss be denoted as $\mathbf{p}_i$ (that load could be zero, of course). Let us assume that the forces are sufficient to significantly deform the struc-

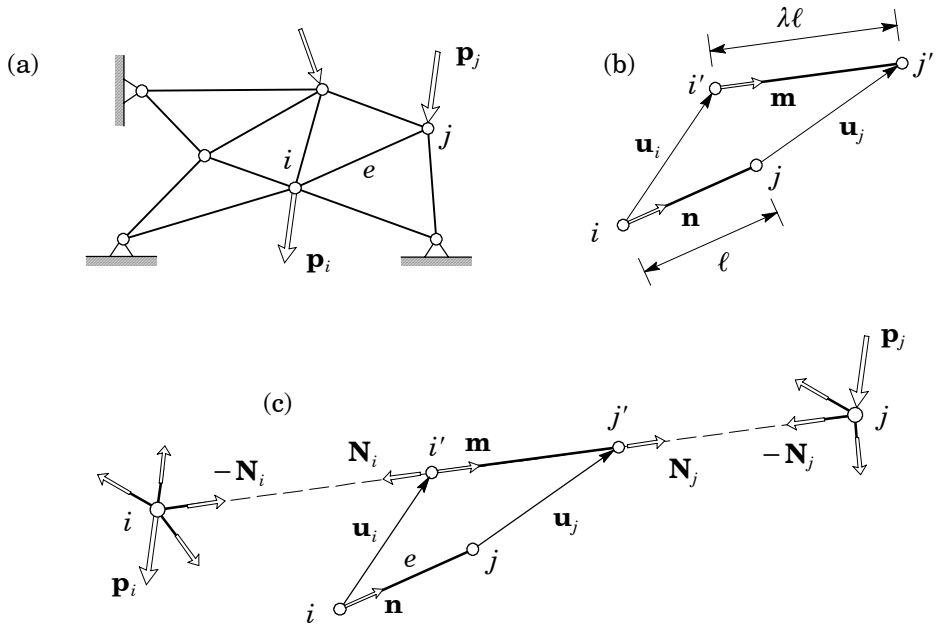

Figure 1.14 Truss structure (a) typical geometry, (b) the displacement and stretching of a typical member e and (c) the force in a typical member e exerted on the joints i and j to which it is attached

ture so that we must consider both material and geometric nonlinearities. These nonlinearities can be captured by combining the nonlinear responses of the individual members through assembly procedures like those typically used in linear structural analysis via the principle of virtual work. Let $\mathbf{u} \equiv \{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n\}$ be the $3n \times 1$ displacement vector for the entire structure. Similarly, let us define the global load vector as $\mathbf{p} \equiv \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n\}$ and the global virtual displacement vector as $\overline{\mathbf{u}} \equiv \{\overline{\mathbf{u}}_1, \overline{\mathbf{u}}_2, ..., \overline{\mathbf{u}}_n\}$.†

Boundary conditions. Certain of the displacements will be prescribed (these are the boundary conditions where the displacement is known). Let us define two Boolean matrices (i.e., a matrix with only ones and zeros—basically useful for picking elements out of other matrices) that extract the prescribed and unprescribed displacements from the complete $3n \times 1$ matrix $\mathbf{u}$ as follows

$$\boldsymbol{\Gamma}_p \mathbf{u} = \mathbf{u}^p, \qquad \boldsymbol{\Gamma}_f \mathbf{u} = \mathbf{u}^f \tag{1.32}$$

where $\mathbf{u}^p$ is an $n_p \times 1$ vector of known displacement values at the n_p prescribed degrees of freedom and $\mathbf{u}^f$ is an $n_f \times 1$ vector of (as yet) unknown displacement values at the n_f unprescribed (free) degrees of freedom. Clearly, the displacement at a certain degree of freedom is either prescribed or not. Therefore, $n_p + n_f = 3n$. The matrix $\boldsymbol{\Gamma}^p$ is $n_p \times 3n$ and $\boldsymbol{\Gamma}^f$ is $n_f \times 3n$. Each row of each of these matrices has a single entry of one with all the rest equal to zero. The action is simply to pick out the relevant displacement component from the global vector. The nature of these transformation is such that

$$\mathbf{u} = \boldsymbol{\Gamma}_p^T \mathbf{u}^p + \boldsymbol{\Gamma}_f^T \mathbf{u}^f \tag{1.33}$$

In other words, the complete displacement vector can always be reconstructed from the prescribed and unprescribed parts using the transposes of those same Boolean matrices. A consequence of Eqs. (1.32) and (1.33) is that $\boldsymbol{\Gamma}_p \boldsymbol{\Gamma}_f^T = \mathbf{0}$ and $\boldsymbol{\Gamma}_f \boldsymbol{\Gamma}_p^T = \mathbf{0}$.

At the places where the displacements are prescribed the corresponding nodal forces are *reaction forces* (i.e., not known a priori). At the places where the displacements are not prescribed the nodal forces are *applied nodal loads*. We can, therefore, partition the load vector $\mathbf{p}$ along the same lines as $\mathbf{u}$. Let $\mathbf{p}^p$ be an $n_p \times 1$ vector of (as yet) unknown reaction forces at the n_p prescribed degrees of freedom and let $\mathbf{p}^f$ be an $n_f \times 1$ vector of known applied loads at the n_f unprescribed (free) degrees of freedom. It should be evident (for the same reasons as the displacements) that

$$\boldsymbol{\Gamma}_p \mathbf{p} = \mathbf{p}^p, \qquad \boldsymbol{\Gamma}_f \mathbf{p} = \mathbf{p}^f, \qquad \mathbf{p} = \boldsymbol{\Gamma}_p^T \mathbf{p}^p + \boldsymbol{\Gamma}_f^T \mathbf{p}^f \tag{1.34}$$

Throughout the development of a method of computation it will be crucial to distinguish between the knowns and unknowns (on both the force side and the displacement

† We engage in a slight abuse of matrix notation here. All $n \times 1$ matrices, which we refer to as "vectors," will be assumed to be column matrices unless otherwise indicated. The abuse of notation is that we should write $\mathbf{a} = [\, \mathbf{b}^T, \mathbf{c}^T, ..., \mathbf{d}^T \,]^T$ to create a column matrix $\mathbf{a}$ by concatenation of shorter column matrices $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$. Where there is no ambiguity we simply write $\mathbf{a} = \{\, \mathbf{b}, \mathbf{c}, ..., \mathbf{d} \,\}$ to imply this operation.

side of the picture). It turns out that the reaction forces $\mathbf{p}^p$ always enter the governing equations linearly and, as a result, it is always possible to put the governing equations in a form that allows the determination of the unknown nodal displacements without solving for the reaction forces at the same time. The reaction forces are then computed as a post-processing step. This fact is exploited in all structural and finite element analysis programs for both linear and nonlinear problems. One should also note that any time a Boolean matrix is used in a mathematical description of a theory there is likely to be a clever computer programming technique that allows the execution of the operation without actually doing the matrix multiplication.

Member strain-displacement relationships. Consider the single member e connected to joints i and j shown in Fig. 1.14(b). The member has an initial length ℓ_e and is initially oriented in the direction $\mathbf{n}_e$ (a unit vector pointing from node i to node j in the undeformed configuration). The bar deforms (uniformly) to a length of $\lambda_e \ell_e$ and has a final orientation in the direction $\mathbf{m}_e$ (a unit vector pointing from node i' to node j' in the deformed configuration). The motion is accomplished by displacing joint i by an amount $\mathbf{u}_i$ and joint j by an amount $\mathbf{u}_j$. Clearly, by vector addition we must have the relationship $\ell_e \mathbf{n}_e + \mathbf{u}_j = \mathbf{u}_i + \lambda_e \ell_e \mathbf{m}_e$ and hence

$$\lambda_e \mathbf{m}_e = \mathbf{n}_e + \frac{1}{\ell_e}\left(\mathbf{u}_j - \mathbf{u}_i\right) \tag{1.35}$$

Since $\left(\lambda_e \mathbf{m}_e\right) \cdot \left(\lambda_e \mathbf{m}_e\right) = \lambda_e^2$, we can compute the Lagrangian strain $E_e \equiv \frac{1}{2}\left(\lambda_e^2 - 1\right)$ in terms of the end displacements as

$$E_e = \frac{1}{\ell_e}\left(\mathbf{u}_j - \mathbf{u}_i\right) \cdot \mathbf{n}_e + \frac{1}{2\ell_e^2} \| \mathbf{u}_j - \mathbf{u}_i \|^2 \tag{1.36}$$

Let us assume that the force in element e, $\hat{N}_e$, depends nonlinearly upon the strain in that bar through a nonlinear constitutive function $N_e(E_e)$ as $\hat{N}_e = \lambda_e N_e(E_e)$. The constitutive function could be determined experimentally from a laboratory test and fit to a suitable curve.

Let us give the structure a virtual displacement $\overline{\mathbf{u}} \equiv \{\overline{\mathbf{u}}_1, \overline{\mathbf{u}}_2, ..., \overline{\mathbf{u}}_n\}$ by moving each of the joints. The virtual strain in member e caused by moving joints i and j by $\overline{\mathbf{u}}_i$ and $\overline{\mathbf{u}}_j$, respectively, can be computed as

$$\overline{E}_e \equiv DE_e \cdot \overline{\mathbf{u}} = \frac{1}{\ell_e}\left[\mathbf{n}_e + \frac{1}{\ell_e}\left(\mathbf{u}_j - \mathbf{u}_i\right)\right] \cdot \left(\overline{\mathbf{u}}_j - \overline{\mathbf{u}}_i\right) \tag{1.37}$$

where the notation $DE \cdot \overline{\mathbf{u}}$ stands for the directional derivative of E in the direction of the virtual displacement $\overline{\mathbf{u}}$. Note that the formal definition of the directional derivative of a function $g(\mathbf{x})$ in the direction $\mathbf{y}$ is

$$Dg(\mathbf{x}) \cdot \mathbf{y} = \frac{d}{d\alpha}\left[g(\mathbf{x} + \alpha \mathbf{y})\right]_{\alpha = 0} \tag{1.38}$$

Let us introduce some notation to help with the statement of virtual work for the entire structure. First, to help us keep track of which members are connected to which (global) nodes, we shall say that member e is attached to nodes i_e and j_e (these numbers are simply the global node numbers associated with local nodes i and j for element e). We keep track of this global to local node mapping in a $m \times 2$ array that lists the global node number attached to the "i" end (column 1) and the "j" end (column 2) of the element. The choice of which end is "i" and which is "j" is arbitrary. Define the $3 \times 3n$ matrix

$$\mathbf{B}_e \equiv \frac{1}{\ell_e}\Big[\begin{array}{ccccccccccc} & & & & i_e & & & & j_e & & \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{array}\Big] \tag{1.39}$$

where $\mathbf{0}$ is a 3×3 zero matrix and $\mathbf{I}$ is a 3×3 identity matrix, so that we can compute the difference in end displacements, divided by the length, for member e as

$$\mathbf{B}_e\mathbf{u} = \frac{1}{\ell_e}\Big[\mathbf{u}_{j_e} - \mathbf{u}_{i_e}\Big] \tag{1.40}$$

One can think of this matrix as a "differencing" operator. It contains information on change in length of the member, but it is really just the difference in displacement divided by the length of the element. It computes the rate of change of displacement over the length of the element. In Chapter 2 we will discuss theories of continuous systems governed by differential equations. The action of differencing will be replaced there with the action of differentiation, but an operator like $\mathbf{B}_e$ will always show up. Note that $\mathbf{B}_e$ does not depend upon the displaced configuration of the structure.

Let us also define the 1×3 matrix $\mathbf{E}_e(\mathbf{u})$ through the relationship

$$\mathbf{E}_e^T(\mathbf{u}) = \mathbf{n}_e + \mathbf{B}_e\mathbf{u} \tag{1.41}$$

where $\mathbf{n}_e$ is the unit vector pointing along the original direction of element e. Note that, with this notation we can compute the real and virtual strain in element e, respectively as

$$E_e = \mathbf{n}_e^T\mathbf{B}_e\mathbf{u} + \tfrac{1}{2}\mathbf{u}^T\mathbf{B}_e^T\mathbf{B}_e\mathbf{u}, \quad \overline{E}_e = \mathbf{E}_e(\mathbf{u})\mathbf{B}_e\overline{\mathbf{u}} \tag{1.42}$$

The principle of virtual work. Consider the virtual work (force times virtual displacement) done by the forces at joints i and j shown in Fig. 1.14(c) when subjected to the virtual displacements $\overline{\mathbf{u}}_i$ and $\overline{\mathbf{u}}_j$, respectively. First, notice that the element forces from element e have the explicit expressions

$$\mathbf{N}_{i_e} = -\hat{N}_e\mathbf{m}_{e=} - \lambda_e N_e(E_e)\mathbf{m}_e, \quad \mathbf{N}_{j_e} = \hat{N}_e\mathbf{m}_e = \lambda_e N_e(E_e)\mathbf{m}_e \tag{1.43}$$

The virtual work done by these forces is, respectively, $\mathbf{N}_{i_e} \cdot \overline{\mathbf{u}}_i$ and $\mathbf{N}_{j_e} \cdot \overline{\mathbf{u}}_j$. Note that the designation i_e and i refer to the same node (i.e., node "i") but the nodal displacements do not depend upon the elements while the element quantities need to be associated with the nodes. The context should always make clear which usage is appropriate.

Let K_i be the set of all members that frame into joint i. When we sum the virtual work done at the n nodes we can observe that the result can be expressed as

$$\overline{W} \;=\; \sum_{i=1}^{n}\Big[\,\mathbf{p}_i + \sum_{k\in K_i} -\mathbf{N}_k\Big]\cdot\overline{\mathbf{u}}_i \;=\; 0 \tag{1.44}$$

The virtual work sums to zero if all nodes are in equilibrium. That is the "principle of virtual work."

The sum on k does not have the element subscript because at a given node either an "i" end or a "j" end could be framing in. Careful inspection of all of the contributions to the sum involving element forces reveals that each element contributes exactly two terms to the double sum over nodes and members connected to those nodes—one at its "i" end and one at its "j" end. Thus, a sum over all nodes of a sum over all elements framing into those nodes can always be rewritten as a sum over elements of a sum over the two ends of each element. To wit,

$$\sum_{i=1}^{n}\sum_{k\in K_i}(\cdot) \;=\; \sum_{e=1}^{M}\sum_{i=1}^{2}(\cdot) \tag{1.45}$$

The structure shown in Fig. 1.14(a), for example, has 7 nodes with {2, 2, 2, 3, 4, 4, 5} members connected to nodes {1, 2, 3, 4, 5, 6, 7}. The sum the number of elements per node over the 7 nodes is 22, which is exactly the number of members (11) times 2. Furthermore, the magnitude of the member force is constant, and the forces that act at opposite ends of the member point in opposite directions, as shown in Eq. (1.43). Therefore, the work equation can also be written as

$$\begin{aligned}
\overline{W} \;&=\; -\sum_{e=1}^{m}\Big[\mathbf{N}_{j_e}\cdot\overline{\mathbf{u}}_{j_e} + \mathbf{N}_{i_e}\cdot\overline{\mathbf{u}}_{i_e}\Big] \;+\; \sum_{i=1}^{n}\mathbf{p}_i\cdot\overline{\mathbf{u}}_i \\
&=\; -\sum_{e=1}^{m}\Big[\lambda_e N_e\mathbf{m}_e\cdot\overline{\mathbf{u}}_{j_e} - \lambda_e N_e\mathbf{m}_e\cdot\overline{\mathbf{u}}_{i_e}\Big] \;+\; \sum_{i=1}^{n}\mathbf{p}_i\cdot\overline{\mathbf{u}}_i \\
&=\; -\sum_{e=1}^{m}N_e(E_e)\Big[\lambda_e\mathbf{m}_e\cdot\big(\overline{\mathbf{u}}_{j_e}-\overline{\mathbf{u}}_{i_e}\big)\Big] \;+\; \sum_{i=1}^{n}\mathbf{p}_i\cdot\overline{\mathbf{u}}_i \\
&=\; -\sum_{e=1}^{m}\ell_e N_e(E_e)\overline{E}_e \;+\; \sum_{i=1}^{n}\mathbf{p}_i\cdot\overline{\mathbf{u}}_i
\end{aligned} \tag{1.46}$$

Finally, we can recognize the first sum in the last line as simply the sum, over the m elements, of the element length ℓ_e times the element constitutive function $N_e(E_e)$ times the element virtual strain, $\overline{E}_e = \mathbf{E}_e(\mathbf{u})\mathbf{B}_e\overline{\mathbf{u}}$. This sum is usually called the *internal virtual work*. The second sum is the *external virtual work* done by the applied nodal forces. Let us define the virtual work function to be the negative of $\overline{W}$. To wit,

$$G(\mathbf{u},\overline{\mathbf{u}}) \;\equiv\; \overline{\mathbf{u}}^T\Big[\sum_{e=1}^{m}\ell_e\mathbf{B}_e^T\mathbf{E}_e^T(\mathbf{u})\,N_e(E_e)\;-\;\mathbf{p}\Big] \tag{1.47}$$

where, as previously noted, $\mathbf{p}\equiv\{\mathbf{p}_1,\,\mathbf{p}_2,\,...,\,\mathbf{p}_n\}$ and $\overline{\mathbf{u}}\equiv\{\overline{\mathbf{u}}_1,\,\overline{\mathbf{u}}_2,\,...,\,\overline{\mathbf{u}}_n\}$. The principle of virtual work states that if $G(\mathbf{u},\overline{\mathbf{u}})=0$ for all virtual displacements $\overline{\mathbf{u}}$ then the system is in equilibrium. Therefore, the nonlinear equilibrium equations are

$$\mathbf{g}(\mathbf{u}) \;\equiv\; \sum_{e=1}^{m}\ell_e\mathbf{B}_e^T\mathbf{E}_e^T(\mathbf{u})N_e(E_e)\;-\;\mathbf{p}\;=\;\mathbf{0} \tag{1.48}$$

This equation is nothing more than an expression that the internal forces contributed to by the members must balance the forces applied at the nodes. The direction and magnitude of the internal member forces have properly accounted for large deformation, through the presence of the term $\mathbf{E}_e(\mathbf{u})$, and material nonlinearity, through the presence of the term $N_e(E_e)$.

Newton's Method. To carry out Newton's method we must compute the gradient of the nonlinear equilibrium function (i.e., the tangent stiffness matrix). The gradient $\nabla\mathbf{g}$ can be computed as

$$\nabla\mathbf{g}(\mathbf{u}) \;\equiv\; \mathbf{A}(\mathbf{u}) \;=\; \sum_{e=1}^{m}\ell_e\mathbf{B}_e^T\Big[\mathbf{E}_e^T(\mathbf{u})\,\frac{\partial N_e}{\partial E_e}\,\mathbf{E}_e(\mathbf{u})\;+\;\mathbf{G}_e\Big]\mathbf{B}_e \tag{1.49}$$

where $\mathbf{G}_e\equiv N_e(E_e)\mathbf{I}$. The first term in the tangent matrix comes from application of the chain rule to $N_e(E_e)$ as $\partial N_e/\partial\mathbf{u}=(\partial N_e/\partial E_e)(\partial E_e/\partial\mathbf{u})$ and noting that $\partial E_e/\partial\mathbf{u}=\mathbf{E}_e(\mathbf{u})\mathbf{B}_e$. The second term comes from the observation that $\partial\mathbf{E}_e(\mathbf{u})/\partial\mathbf{u}=\mathbf{B}_e$.

From Eqs. (1.48) and (1.49) the equilibrium configuration $\mathbf{u}$ of the truss subjected to the loads $\mathbf{p}$ can be found by Newton's method by iteratively solving the system of equations $\mathbf{A}(\mathbf{u}^i)\Delta\mathbf{u}^i=-\mathbf{g}(\mathbf{u}^i)$ for the increment $\Delta\mathbf{u}^i$ and updating as $\mathbf{u}^{i+1}=\mathbf{u}^i+\Delta\mathbf{u}^i$ starting from an initial guess $\mathbf{u}^0$, which can be taken as zero.

Enforcement of Boundary Conditions. Equation (1.48) establishes equilibrium at all nodes in all three directions—the dimension of $\mathbf{g}$ is $3n\times 1$. If we premultiply that equation by $\mathbf{\Gamma}_f$ then we select only those equations associated with the unprescribed displacements. Furthermore, we note that the displacement vector satisfies the relationship $\mathbf{u}=\mathbf{\Gamma}_p^T\mathbf{u}^p+\mathbf{\Gamma}_f^T\mathbf{u}^f$ and the values $\mathbf{u}^p$ are prescribed (and hence not subject to incrementation). Therefore,

$$\frac{\partial E_e(\mathbf{u})}{\partial\mathbf{u}} \;=\; \frac{\partial E_e(\mathbf{u})}{\partial\mathbf{u}_f} \;=\; \mathbf{E}_e(\mathbf{u})\mathbf{B}_e\mathbf{\Gamma}_f^T \tag{1.50}$$

With this observation we can note that

$$\nabla\big(\mathbf{\Gamma}_f\mathbf{g}(\mathbf{u})\big) \;\equiv\; \hat{\mathbf{A}}(\mathbf{u}) \;=\; \sum_{e=1}^{m}\ell_e\hat{\mathbf{B}}_e^T\Big[\mathbf{E}_e^T(\mathbf{u})\,\frac{\partial N_e}{\partial E_e}\,\mathbf{E}_e(\mathbf{u})\;+\;\mathbf{G}_e\Big]\hat{\mathbf{B}}_e \tag{1.51}$$

where $\hat{\mathbf{B}}_e \equiv \mathbf{B}_e \boldsymbol{\Gamma}_f^T$ is a $3 \times n_f$ matrix that executes the restriction of the tangent stiffness matrix have dimension $n_f \times n_f$. Observe that premultiply with With this restriction, we perform the iteration

$$\begin{aligned} \hat{\mathbf{A}}(\mathbf{u}^i)\Delta\mathbf{w}^i &= -\boldsymbol{\Gamma}_f \mathbf{g}(\mathbf{u}^i) \\ \mathbf{w}^{i+1} &= \mathbf{w}^i + \Delta\mathbf{w}^i \\ \mathbf{u}^{i+1} &= \boldsymbol{\Gamma}_f^T \mathbf{w}^{i+1} + \boldsymbol{\Gamma}_p^T \mathbf{u}_p \end{aligned} \tag{1.52}$$

where the $n_f \times 1$ iterate $\mathbf{w} \equiv \mathbf{u}^f$ is simply the "free" part of the displacement vector. Iteration proceeds until the residual reduces to an acceptably small value. Once the final value of the displacement has been found, the reaction forces can be computed directly from Eq. (1.48) and Eq. $(1.34)_{\text{a}}$ as

$$\mathbf{p}^p = \boldsymbol{\Gamma}_p \sum_{e=1}^{m} \ell_e \mathbf{B}_e^T \mathbf{E}_e^T(\mathbf{u}) N_e(E_e) \tag{1.53}$$

The truss example serves to show several of the aspects of nonlinear structural analysis that will be explored in greater detail in the next chapter and show up repeatedly throughout the book. First, there is an orderly way to assemble the equations of equilibrium $\mathbf{g}(\mathbf{u}) = \mathbf{0}$ of the system from member contributions. Second, for any given displacement state $\mathbf{u}$ we can evaluate the strain E_e in each member through Eq. (1.36) and hence we can evaluate the force in each member as $N_e(E_e)$. This organization of the computation is often called a strain-driven problem because we generate, through Newton's method, a sequence of candidate displacement configurations. At a given configuration we can evaluate the state of force in the structure implied by the displaced configuration. Third, the tangent stiffness matrix generally has two parts: (1) the material tangent part that comes from the term with $\partial N/\partial E$ and (2) the geometric part that comes from the term with $\mathbf{G}_e$. For problems with linear geometry, the second term does not appear and the matrix $\mathbf{E}_e(\mathbf{u})$ reduces to $\mathbf{E}_e(\mathbf{0}) = \mathbf{n}_e^T$ (i.e., the matrix of initial direction cosines). For a linear elastic material $\partial N/\partial E$ is constant and equal to the initial axial stiffness of the bar, EA. Finally, this example gives us a hint about the organization of computations for structural analysis. To form the residual and element tangent stiffness we need to isolate those displacements from the global displacement vector that are associated with element e to compute the strain E_e, stress N_e, and the material tangent $\partial N_e/\partial E_e$. Owing to the simple (very sparse) structure of the matrix $\hat{\mathbf{B}}_e$ we never actually use it in a matrix multiplication, but rather use it in the sense of "assembly" of the global tangent stiffness and residual. Assembly procedures for linear analysis are documented in every textbook on structural analysis and will not be further discussed here.

In some sense, the nonlinear truss formulation is prototypical of all nonlinear computations for structural elements. It does, however, leave some room for growth in analyzing nonlinear problems because it is a discrete theory. The next chapter expands on the ideas introduced here.

1.9 References

Gill, P. E., Murray, W. and Wright, M. H. (1981). *Practical optimization*. Academic Press, New York.

Golub, G. H. and C. F. Van Loan (1994). *Matrix computations*. Second edition. The Johns Hopkins University Press, Baltimore.

Crisfield, M. A. (1991). *Non-linear finite element analysis of solids and structures. Volume 1: Essentials.* John Wiley & Sons. New York, New York.

El Naschie, M. S. (1990). *Stress, stability and chaos in structural engineering: an energy approach*. McGraw-Hill. New York, New York.

Hjelmstad, K. D. (1997). *Fundamentals of structural mechanics*. Prentice-Hall, Upper Saddle River, New Jersey.

Hodge, P. G. (1959). *Plastic analysis of structures.* McGraw-Hill. New York, New York.

Hughes, T. J. R. (1987). *The finite element method.* Prentice-Hall. Englewood Cliffs, New Jersey.

Luenberger, D. G. (1984). *Linear and nonlinear programming.* 2nd ed. Addison Wesley, Reading Massachusetts.

McGuire W. and R. H. Gallagher (1979). *Matrix structural analysis*. John Wiley &Sons. New York.

Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling (1986). *Numerical recipes.* Cambridge University Press. Cambridge.

Zienkiewicz, O. C. and Taylor, R. L. (1991). *The finite element method. Volume 2 Solid and fluid mechanics, dynamics and nonlinearity*, 4th ed. McGraw Hill, London.