Real-time Simulation of Physically Realistic Global Deformation*

Yan Zhuang† John Canny‡
Computer Science Department, University of California, Berkeley, CA 94720-1776

Abstract

Real-time simulation and animation of global deformation of 3D objects, using finite element method (*FEM*), is difficult due to the following 3 fundamental problems: (1) The linear elastic model is inappropriate for simulating large motion and large deformations (unacceptable distortion will occur); (2) The time step for dynamic integration has to be drastically reduced to simulate collisions; (3) The size of the problem (the number of elements in the FEM mesh) is one order of magnitude larger than a 2D problem.

In this paper, we present a novel approach to counter these 3 difficulties: (1) using quadratic strain instead of the popular linear strain to simulate arbitrarily large motion and global deformation of a 3D object; (2) applying an implicit simplified impulse to a decoupled system, which makes an integration step for collision as cheap as a regular dynamic integration step; (3) using a graded mesh instead of a uniform mesh, which reduces the asymptotic complexity of a 3D problem to that of a 2D problem.

1 Introduction

Physically realistic modeling and manipulation of deformable objects has been the bottleneck of many applications, such as human tissue modeling, character animation, surgical simulation, etc. Among the potential applications, a virtual surgical training system is the most demanding for the real-time performance because of the real-time interaction with virtual human tissue.

So far real time simulation and animation of deformation has only been achieved in two special cases: 2D problems such as cloth simulation [2], and small or local deformations for 3D objects [4].

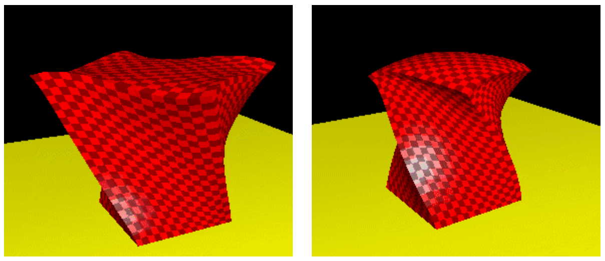

Figure 1: The bottom of the object is fixed and its top is twisted. The top in the left image is distorted (grown bigger) because it is simulated using linear elasticity. The right image shows that the same distortion does not occur with nonlinear elasticity.

In this paper, we address the bottleneck problem of real-time simulation of physically realistic large *global deformations* of 3D objects. In particular we apply the finite element method to model such deformation. By *global deformation*, we mean deformations,

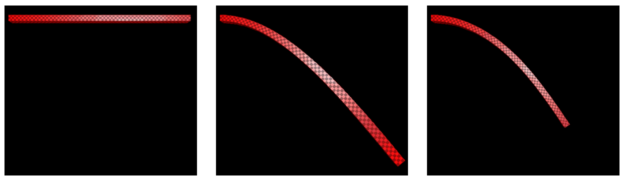

Figure 2: The left image shows a beam at its initial configuration with a fixed left end and a free right end. The middle image shows the *distorted* deformation under gravity, using linear strain. The right image shows the *undistorted* deformation, under the same gravitational force, using quadratic strain (equation (5) and (6)).

such as large twisting or bending of an object, which involve the entire body, in contrast to poking and squeezing, which involves a relatively small region of the deformable object.

First, we point out that the application of linear elastic finite element methods to simulating large global deformation leads to unacceptable distortions (Figure 1 and 2). In section 3 we propose to simulate global deformations using nonlinear finite element methods.

Secondly, we propose an efficient collision handling method for FEM in section 4. Simulating deformable object collisions using a penalty method [12] requires tiny time steps to generate visually satisfactory animations. A general impulse collision [1] is considered more efficient and accurate but still requires more computational power than collision-free dynamics. In section 4 we present an extremely simple and efficient collision time integration scheme, which makes the time integration of collision dynamics as cheap as that of collision-free dynamics.

Finally, we observe that simulation of 3D deformation is at least one order of magnitude more difficult than a similar 2D problem because the size of the problem (the number of elements in its mesh) is one order higher. To counter this problem we propose a *graded* mesh that reduces the complexity of the 3D problem by one order of magnitude asymptotically.

2 Related Work

Our work of modeling and simulating a deformable object falls into the realm of physically based modeling. Witkin *et al*[15] summarizes the methods and principles of physically based modeling, which has emerged as an important new approach to computer animation and computer graphics modeling.

In general, there are two different approaches to modeling deformable objects: the mass spring model and the finite element model.

The mass spring model has good success in creating visually satisfactory animations. Waters [14] uses a spring model to create a realistic 3D facial expression. Provot *et al*[10] describes a 2D model for animating cloth, using double cross springs. Promayon *et al*[9] presents a mass-spring model of 3D deformable objects and develops some control techniques.

Despite the success in some animation applications, the mass

*Supported by a Multi-Disciplinary Research Initiative grant for 3D Visualization, sponsored by BMDO with support from ONR.

†yzhuang@cs.berkeley.edu
‡jfc@cs.berkeley.edu

spring models do not model the underlining physics accurately, which makes it unsuitable for simulation that requires more accuracy. The structure of the mass spring is often application dependent and hard to interpret. The animation results often vary dramatically with different spring structures. The distribution of the mass to nodes is somehow (if not completely) arbitrary. Despite its inaccuracy, it does not have visual distortion and it is computationally cheap to integrate over time because the system is, by its very nature, a set of independent algebraic equations, which require no matrix inversions to solve.

As an alternative, finite element methods (*FEM*) model the continuum much more accurately and its underlining mathematics is well studied and developed. Another similar method is the finite difference method, which is less accurate but simple and appropriate to some applications. Indeed a linear finite difference method over a uniform mesh is just a special case of FEM. Its accuracy and mathematical rigorousness makes FEM a better choice for applications such as surgical simulations.

Terzopoulos *et al*[12, 11, 13] applies both finite difference and finite element methods in modeling elastically deformable objects. Celniker *et al*[6] applies FEM to generate primitives that build continuous deformable shapes designed to support a new free-form modeling paradigm. Pieper *et al*[8] applies FEM to computer-aided plastic surgery. Chen [3] animates human muscle using a 20 node hexahedral FEM mesh. Keeve *et al*[5] develops a static anatomy-based facial tissue model for surgical simulation using the FEM. Most recently, Cotin *et al*[4] presents real-time elastic deformation of soft tissues for surgery simulation, which only simulates the static deformation.

Out work differs from the previous work by either one or all of the following: (1) we simulate large global deformation instead of small local deformation; (2) we simulate the dynamic behavior of soft objects rather than the static deformation.

3 Global Deformation Using Nonlinear FEM

By *global deformation*, we mean deformations that involve the entire body, such as large bending and twisting. We apply the displacement based finite element method (*FEM*) to model the dynamics of such deformation of 3D elastic objects. Essentially this requires solving the following system of differential equations

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{D}\dot{\mathbf{u}} + \mathbf{R}(\mathbf{u}) = \mathbf{F} \tag{1}$$

where $\mathbf{u}$ is the $3n$-dimensional nodal displacement vector; $\dot{\mathbf{u}}$ and $\ddot{\mathbf{u}}$, the respective velocity and acceleration vectors; $\mathbf{F}$, the external force vector; $\mathbf{M}$, the $3n \times 3n$ mass matrix; $\mathbf{D}$, the damping matrix; and $\mathbf{R}(\mathbf{u})$, the internal force vectors due to deformation. n is the number of nodes in the FEM model ([16]). It is worth pointing out that the mass matrix $\mathbf{M}$ and damping matrix $\mathbf{D}$ usually remain constant. At each time step, we just have to compute the external force $\mathbf{F}$ and the internal force $\mathbf{R}(\mathbf{u})$, and solve the system (1)[1].

All previous works ([8],[3],[5],[4]) define internal force using linear strain as following:

$$\epsilon_x = \frac{\partial u}{\partial x} \tag{2}$$

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \tag{3}$$

[1]Note that the static equilibrium equation is just a special case of equation (1), where $\ddot{\mathbf{u}}$ and $\dot{\mathbf{u}}$ are set to zeros.

where x, y and z are the independent variables of the cartesian frame, and u, v and w are the corresponding displacement variables at the given point. Other terms of the strain at point (x, y, z), ϵ_y, ϵ_z, γ_{yz} and γ_{zx}, are defined similarly.

This linear strain makes the internal force vector linear with respect to nodal displacement vector. Namely it simplifies equation (1) to the following *linear* system:

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{D}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{F} \tag{4}$$

This allows a preprocessing step that computes the constant stiffness matrix $\mathbf{K}$ and its LU factorization. This preprocessing step has been the key to real-time performance in previous works such as [4], which animates deformations using a sequence of static equilibrium.

However this approach is only appropriate for simulating local deformation, such as poking and compression, and small bending and twisting. Application of this linear strain to modeling large global deformation will lead to unacceptable distortion (Figure 1 and 2). The distortion is due to the fact that this linear strain models rigid body motions as *differential* motions. If we subject an undeformed object to a large (instead of differential) rigid body rotation, the linear strain (2) and (3) will give a non-zero strain, while the body has no deformation at all. Therefore the object will deform itself to balance this *artificial* strain.

Large global deformations are crucial to many applications, such as surgical simulation and character animation. This tempts us to model the strain differently. To simulate global deformations and large motions, we apply quadratic strain to model $\mathbf{R}(\mathbf{u})$ as following:

$$\epsilon_x = \frac{\partial u}{\partial x} + \frac{1}{2}\left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial x}\right)^2\right] \tag{5}$$

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} + \left[\frac{\partial u}{\partial x}\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\frac{\partial v}{\partial y} + \frac{\partial w}{\partial x}\frac{\partial w}{\partial y}\right] \tag{6}$$

Notice that this quadratic strain handles arbitrarily large rigid body motion correctly. Namely no artificial strain will be introduced when we subject the object to a rigid body motion.

This quadratic strain makes (1) a nonlinear system, in which the internal force $\mathbf{R}(\mathbf{u})$ is no longer a linear term of nodal displacements. If we solve this nonlinear system using an implicit integration scheme as [2], real time simulation is impossible for reasonably large meshes.

We observe that a soft material such as live tissue has small stiffness in all directions (not necessarily isotropic). This makes explicit time integration schemes appropriate because we can take large time steps. We apply the explicit Newmark scheme to equation (1), which leads to the following equations:

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \dot{\mathbf{u}}_n \triangle t_n + \frac{1}{2}\ddot{\mathbf{u}}_n \triangle t_n^2 \tag{7}$$

$$\dot{\mathbf{u}}_{n+1} = \dot{\mathbf{u}}_n + \frac{1}{2}(\ddot{\mathbf{u}}_n + \ddot{\mathbf{u}}_{n+1})\triangle t_n \tag{8}$$

$$(\mathbf{M} + \frac{1}{2}\triangle t_n \mathbf{D})\ddot{\mathbf{u}}_{n+1} = \mathbf{F}_{n+1} - \mathbf{R}(\mathbf{u}_{n+1}) - \mathbf{D}(\dot{\mathbf{u}}_n + \frac{1}{2}\ddot{\mathbf{u}}_n \triangle t_n) \tag{9}$$

The order of updating is (7), (9) and then (8). The bottleneck is equation (9), which is a nonlinear system of equations. If we apply a general method, such as Newton's method, to solve this equation, it requires inverting a large sparse matrix $\mathbf{M} + \frac{1}{2}\triangle t_n \mathbf{D}$ at each time step. Note that the time step $\triangle t_n$ is, in general, not a constant, therefore it is impossible to preprocess the system by computing the inversion of this large sparse matrix. Inverting a large matrix at each integration step makes real-time simulation impossible for any problems of reasonable size.

To achieve real-time performance, we approximate the mass matrix $\mathbf{M}$ by a diagonal matrix, which is obtained by lumping its rows ([16]). If we then apply Rayleigh damping $\mathbf{D} = \alpha\mathbf{M} + \beta\mathbf{K}$, with $\beta = 0$, the matrix $\mathbf{M} + \frac{1}{2}\triangle t_n \mathbf{D}$ becomes a diagonal matrix. This simplifies the nonlinear system of equations (9) into a set of independent *algebraic* equations as following:

$$q^i \ddot{u}^i_{n+1} = f^i_{n+1} - r^i_{n+1} - d^i_{n+1} \tag{10}$$

where q^i is the $i-th$ component of the diagonalized $\mathbf{M} + \frac{1}{2}\triangle t_n \mathbf{D}$; $\ddot{u}^i_{n+1}$, f^i_{n+1}, r^i_{n+1} and d^i_{n+1} are the $i-th$ component of $\mathbf{u}_{n+1}$, $\mathbf{F}_{n+1}$, $\mathbf{R}(\mathbf{u}_{n+1})$ and $\mathbf{D}(\dot{\mathbf{u}}_n + \frac{1}{2}\ddot{\mathbf{u}}_n \triangle t_n)$ respectively. Solving this system of equations requires no matrix inversion.

The diagonalization also makes the enforcement of the all types of boundary conditions very simple. For natural boundary condition, we specify the force and compute u^i_{n+1}. For essential boundary conditions, we simply ignore equation (10) and set explicitly the corresponding displacement and velocity to the given values.

It is worth pointing out that the critical time step for an explicit integration scheme is dictated by the largest stiffness in the material. This is why an explicit integration scheme is appropriate for soft tissues, which is "soft" in all directions (although not necessarily isotropic), while it is not appropriate for cloth simulation [2].

4 Collision Integration Scheme

For deformable object collisions, the collision time can be assumed finite (unlike the instantaneous collision of rigid bodies). This allows a larger time step for integration.

The popular penalty methods for collision handling [12, 11, 13] did not take advantage of this. A penalty method models the collision by adding an artificial spring of large stiffness at the point of collision. This stiff spring requires tiny time steps to stably simulate a collision. Various experiments show that the ratio between a collision free time step and that of a penalty collision is on the order of hundreds if not more.

This tempts us to develop new collision-handling methods that avoid adding extra artificial stiffness into the system. We will illustrate our collision-handling method, using a special case: collision between a moving deformable body and a stationary rigid body (figure 3). Later in this section, we will show that it is straightforward to extend this method to handle general deformable object collisions.

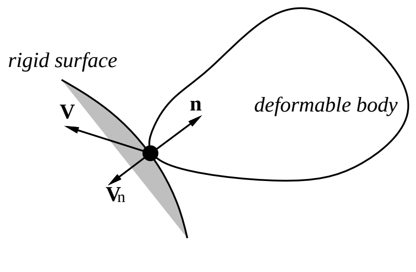

Figure 3: A flexible body collides with a rigid body.

Consider the collision between a deformable body and a stationary rigid body (figure 3). Assume that at time t_n, the node p, with velocity $\hat{\mathbf{v}}(p)_n$, is colliding with a rigid surface of outward normal $\hat{\mathbf{n}}$. The non-penetration constraint requires that the normal component of the velocity of point p drops to zero at the moment of collision. Unlike a rigid body collision, the flexible body will maintain the contact with the stationary rigid body for a nonzero period of time. We enforce the non-penetration constraint at node p by setting the normal component of $\hat{\mathbf{v}}(p)_{n+1}$ to zero as following:

$$\hat{\mathbf{v}}(p)_{n+1} = \hat{\mathbf{v}}(p)_n + (\hat{\mathbf{v}}(p)_n \cdot \hat{\mathbf{n}})\hat{\mathbf{n}} \tag{11}$$

By equation (8), we get

$$\hat{\mathbf{a}}(p)_{n+1} = \frac{2\hat{\mathbf{v}}(p)_{n+1}}{\triangle t_n} - \frac{2\hat{\mathbf{v}}(p)_n}{\triangle t_n} - \hat{\mathbf{a}}(p)_n \tag{12}$$

If we choose $\triangle t_{n+1} = \triangle {t_n}^2$, by equation (7), we have

$$\hat{\mathbf{u}}_{n+2} \cdot \hat{\mathbf{n}} = \hat{\mathbf{u}}_n \cdot \hat{\mathbf{n}} \tag{13}$$

This shows that the non-penetration constraint is enforced after two time steps, because there is no relative motion of the flexible body normal to the surface of the stationary rigid body.

This collision-handling integration scheme can be considered a special case of impulse [1]. However unlike a general impulse, the impulse is never explicitly computed for a frictionless collision.

When friction at collision needs to be considered, we can compute the Coulomb friction easily. By plugging equation (11) into equation (8), we can compute the equivalent acceleration at point p. The we can use equation (12) to compute the equivalent impulse at point p, which is a force normal to the collision surface. This enables us to compute the Coulomb friction and simulate a frictional collision. Note that no matrix inversion is ever needed for impulse computation because the system is diagonalized.

This collision integration scheme can be generalized to general collisions. A general collision involves multiple point contact. Since the system is decoupled, such a collision is modeled as a set of simultaneous *independent* single point collision. When both objects are deformable and moving, we simply set the normal component of the velocity at the point of collision such that the relative normal velocity at the point is zero.

Unlike a general impulse, we do not have to distinguish between the colliding contact and the resting contact [1]. A deformable object's resting on a surface is handled exactly the same as collision, with no additional computational cost.

5 Graded Mesh

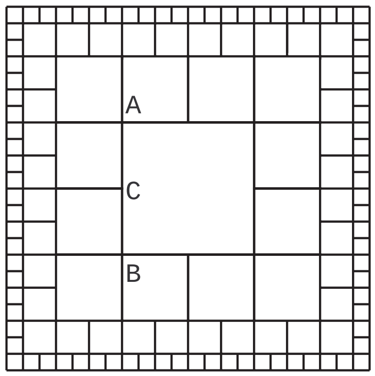

Figure 4: A 2D example of graded mesh.

[2]This does not mean that the entire simulation has to use a constant time step. Indeed the simulation can still use variable time step. This constraint (choice) is only enforced at collision time.

While 2D FEM has great success in achieving real time performance in computer graphics applications, the computational cost is much higher for 3D applications, mainly due to the increase in the number of elements in the mesh. In a roughly uniform 2D finite element mesh, the number of elements is about $O(n^2)$, where n is the average number of elements in each principle direction. However a similar 3D mesh would have $O(n^3)$ elements, which leads to a much larger system of equations.

A mesh has to be fine enough to capture the relevant modes. To illustrate, let us consider using FEM to simulate a wave in a soft material, which is one special kind of deformation. When the spatial frequency of the wave increases, we need finer elements to simulate the wave. When the spatial frequency decreases, we need fewer elements. Modal analysis shows that deformation can be approximated by model composition, where each mode is corresponding to a deformation of a fixed spatial frequency.

Nicolson [7] shows that the cutoff spatial frequency of an object in response to external loads decreases faster than $1/d$ in terms of the distance d away from the surface. This means that for a given error bound, we need finer elements on the surface and coarser elements away from the surface. Furthermore Nicolson's [7] result suggests that if the size of the element increases proportionally to d, we will lose little accuracy with respect to static forces exerted on the surface. Based on this observation, we propose using a graded mesh to model 3D objects. A 2D example of such mesh is shown in figure 4. Extension of this 2D example to a 3D hexahedra mesh is straightforward. Such a mesh reduces the complexity of 3D problems from $O(n^3)$ to $O(n^2)$, while losing little accuracy.

In general, a graded mesh has a severe drawback despite its reduction of the problem size. It is computationally costly to enforce the compatibility at element interface. To simplify the presentation, we discuss this using the 2D example in figure 4. However, note that the discussion applies to 3D hexahedra mesh as well.

For a mesh of linear quadralateral elements, the edge of each element is always a straight line. Therefore the node such as C in figure 4 is constrained by node A and B. Namely the displacement of C has to be such that A, B and C are always co-linear. We refer to a node such a C as a *geometrically constrained* node. The general solution is to use a Lagrangian multiplier, which expands the system and therefore adds extra computational cost to the simulation.

However since we have a diagonalized system (10), the compatibility at element interface can be easily enforced without a Lagrangian multiplier. For unconstrained nodes, such as A and B, we proceed with the computation as presented in section 3. For a constrained node, such as node C, we simply set its displacement to the average of A and B. This explicitly enforces the compatibility without any additional computational cost.

6 Experiments

We implemented both statics and dynamics of elastic objects, using FEM with a hexahedral mesh. On a 400MHz Pentium II PC, a uniform mesh of 1331 elements needs about 0.11 seconds per time step. The graded mesh with the same accuracy needs only 0.06 seconds per step. Also there is no visual slowdown during a collision.

7 Conclusion

We argued that it is important to simulate large global deformation using non-linear strain. A nonlinear strain leads to a nonlinear FEM formulation, which is in general expensive to solve in real time. In order to achieve real-time performance, we diagonalize the mass matrix and the damping matrix.

In some sense this approach combines the best of the linear FEM model and mass-spring model. A mass-spring model is inaccurate in its mathematical formulation, however it is cheaper to solve because it is a diagonal system from the very beginning, and it does not introduce any geometric distortion. Linear FEM model is more accurate in its mathematical formulation of material behavior, but expensive to solve and has distortion for large motion and deformation. A diagonalized nonlinear FEM approach models the material behavior with more accuracy than a linear model and it is still cheap to solve and has no distortion.

The nonlinear FEM model (section 3) and collision-handling integration scheme (section 4) apply to any type of mesh. Although we presented a graded mesh in terms of a hexahedral mesh, the asymptotic argument applies to tetrahedral meshes as well. Indeed it becomes simpler with a tetrahedral mesh because the issue of geometric compatibility at the element interface does not arise.

Despite the many advantages of the hexahedral mesh, it is difficult to generate the mesh for complex geometry. Because of this, we are in the process of extending our system to tetrahedral meshes so that we can simulate deformations of more complex geometry.

8 Acknowledgement

We thank Panayiotis Papadopoulos for sharing with us his FEM expertise and Jonathan Shewchuk for his insight into 3D meshing. We thank David Forsyth and Ken Goldberg for their valuable feedback.

References

[1] David Baraff and Andrew Witkin. Dynamic simulation of non-penetrating flexible bodies. In *Computer Graphics: Proceedings of SIGGRAPH*, pages 303–308. ACM, 1992.

[2] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Computer Graphics: Proceedings of SIGGRAPH*, pages 303–308. ACM, 1998.

[3] David Chen. *Pump It Up: Computer Animation of a Biomechanically Basded Model of Muscle Using the Finite Element Method*. PhD thesis, MIT, 1992.

[4] Stéphane Cotin, Hervé Delingette, and Nicholas Ayache. Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transcation on Visualization and Computer Graphics*, 5(1):62–73, January-March 1999.

[5] E. Keeve, S. Girod, P. Pfeifle, and B. Girod. Anatomy-based facial tissue modeling using the finite element method. *IEEE Visualization*, 1996.

[6] G. Celniker nad G. Gossard. Deformable curve and surface finite elements for free form shage design. *Computer Graphics*, 25(4), 1991.

[7] Edward John Nicolson. *Tactile Sensing and Control of a Planar Manipulator*. PhD thesis, EECS, University of California, Berkeley, 1987.

[8] S. Peiper, J Rosen, and D. Zeltzer. Interactive graphics for plastic surgery: A task-level analysis and implementation. In *Symposium on Interactive 3D Graphics*, 1992.

[9] E. Promayon, P. Baconnier, and C. Puech. Physically-based deformations constrained in displacements and volume. In *EUROGRAPHICS*, 1996.

[10] X. Provot. Deformation constrains in a mass-spring model to describe rigid cloth behavior. *Computer Interface*, 1995.

[11] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. *Computer Graphics*, 22, August 1988.

[12] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *Computer Graphics*, 21, July 1987.

[13] D. Terzopoulos and K. Waters. Physically-based facial modeling, analysis and animation. *Journal of Visualization and Computer Animation*, 1990.

[14] K. Waters. A muscle model for animating three-dimensional facial expression. *Computer Graphics*, 21(4), July 1987.

[15] A. Witkin and *et al*. *An Introduction to Physically Based Modeling*. 1993. Course Notes.

[16] O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method: Solid and Fluid Mechanics Dynamics and Non-Linearity*, volume 2. McGraw-Hill Book Company, 4th edition, 1989.