

# Computational Issues in Non-linear Structural Analysis Using a Physics Engine

## Author:

Kirk Martini, University of Virginia, Department of Architecture, PO Box 400122,  
Charlottesville VA 22904, martini@virginia.edu

## ABSTRACT

The paper describes issues involved in a novel computational approach to non-linear structural analysis. The method is called a physics engine, which has been widely used in computer games and other visually-oriented simulations, but little used in structural engineering. The paper describes an implementation which models a structure as a collection of point masses in a time-step simulation. The position and velocity of each particle are known at the beginning of the time step, and the method calculates forces on the particles resulting from element deformation, gravity, damping, and other sources. The forces for each degree of freedom are then divided by the corresponding mass quantity to calculate acceleration. The acceleration is used in solving the differential equations of motion to determine the position and velocity at the end of the time step, and the process repeats. This procedure has the following distinguishing characteristics: all calculations are done with respect to the deformed structural geometry; there is no global stiffness matrix; mass must be assigned to all degrees of freedom; the method is extremely computationally intensive. The large displacement characteristic means that the method is well suited to modeling phenomena such as post-buckling behavior. The lack of a global stiffness matrix means that the method can easily analyze unstable structures and mechanisms. The combination of these characteristics makes the method well suited to modeling collapse-related phenomena resulting from removing members. The requirement to assign mass to all degrees of freedom raises questions about the proper calculation of rotational mass, which the paper discusses. Because the method is computationally intensive, it is limited to relatively small scale problems. It has been implemented in a program called *Arcade*, and has been primarily used for teaching, since small-scale models are effective for teaching. The paper briefly presents some teaching applications.

## OVERVIEW

To achieve visual realism, the fields of computer graphics and computer games have developed methods to model the physics of moving objects, both rigid and deformable [Bourg]. Many games incorporate commercially available *physics engines*, programming libraries that included algorithms to model rigid and deformable bodies with a variety of physical constraints in a time-step simulation, solving the Newtonian equations of motion

to simulate realistic physical motion [Hecker, Witkin]. Examples of such packages include *Havok Physics* [Havok], *Ageia PhysX* [Ageia], and the freely available *Open Dynamics Engine* [Smith]. An example application would be a game involving race cars, where the game would calculate the movement of each car based on engine forces, wheel friction, collision forces, gravity, and the mass properties of each car. In games, the physical simulation runs in real time, so the on-screen model responds immediately to input from the game controller. Dozens of commercially popular games incorporate physical simulation in some form [Havok].

The physics engine approach has not been widely applied in modeling structural frameworks, and it has the potential to introduce a new kind of program for non-linear, dynamic, time history analysis of structures. Using this approach, a structure can be modeled as a collection of nodal masses connected by elements (in the computer game literature, nodes are commonly called particles and elements are commonly called springs).

This approach has been implemented in a program called *Arcade*, which performs analysis in real time with game-like interactivity; it has been used in structural design courses in the department of architecture and the department of civil engineering at the University of Virginia [Martini, ref. 8 and 9]. Because of its ability to model large displacements and unstable structures, the program can model a range of non-linear dynamic phenomena, including post-buckling behavior of frames, the behavior of cables under changing loads, the motion of free-floating bodies subjected to forces, and progressive collapse mechanisms. One of the characteristics of the *Arcade* implementation that distinguishes it from methods used in games is that *Arcade* emphasizes reasonable engineering accuracy. Implementations used in games typically emphasize computational speed, visual realism, and robust numeric stability with little emphasis on numeric accuracy.

While earlier papers [Martini, ref. 8 and 9] have focused on the use of the program in teaching, this paper focuses on the computational aspects of the physics engine approach in its application to structural engineering. The discussion begins with the organization of the computational simulation loop, and then addresses particular aspects of element formulation.

## THE COMPUTATION LOOP

### Overall Organization

Figure 1 illustrates the organization of the computational loop. The process begins at the box labeled *A*, representing the beginning of a time step. At that point the position and velocity corresponding to each degree of freedom are known. For the two-dimensional *Arcade* program, the degrees of freedom include two translations and one rotation for each node. The positions and velocities are then used to calculate forces and moments acting on the nodes, box *B* in the figure. Forces and moments can result from damping applied directly to the nodes, or from elastic or damping forces exerted by elements attached to the nodes. Once the force quantities are known for each degree of freedom, each is divided by the corresponding mass quantity to get the acceleration at the beginning of the time step, corresponding to box *C* in the figure. The acceleration at the beginning of the time step is then used to predict the position and velocity at the end of

the time step, using conventional numeric methods for differential equations. The Arcade program uses the 4th-order Runge-Kutta method with a constant time step.

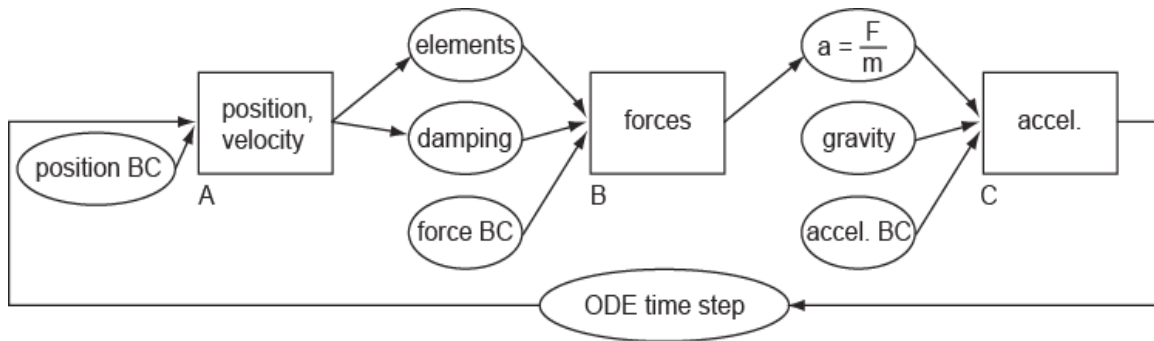


FIGURE 1  
THE COMPUTATIONAL LOOP

Types of boundary condition include the following:

- **Position:** These include locking degrees of freedom to model conventional supports or imposing a changing position over time, such as a support settlement. Position boundary conditions are imposed in box *A* in figure 1.
- **Force:** These include imposed forces on nodal degrees of freedom according to a specified time history. Force boundary conditions are applied in box *B* in figure 1.
- **Acceleration:** These include imposed accelerations on nodal degrees of freedom according to a specified time history, and can be used to model the effects of earthquakes. Acceleration boundary conditions are applied in box *C* in figure 1.

In real-time animation, the program compares the simulation time (the sum of all time steps since the simulation began) with real time (that actual time elapsed since the simulation began) at the beginning of each cycle of the loop. If the simulation time exceeds real time (i.e. the simulation is proceeding faster than the actual physical event would), then the program renders the current graphic state of the model to the screen. The time required for the rendering interrupts the calculation and brings the simulation time and real time toward synchronization. The *frame rate* of the animation is the number of times per second that the program graphically renders the model. A minimum frame rate of about 18 frames per second is needed for smooth-looking animation.

The ability of the program to render a model in real time depends on the number of nodes, the number and complexity of the elements, and the time step. If the model is too large or the time step too small, then the calculation will be slower than real time. In that case, the program offers the option to set a scale factor which effectively runs a model in slow motion, so that one second of simulation may correspond to multiple seconds of real time.

The size of the time step required for numeric stability strongly depends on the mass and stiffness properties of the model. Models with low mass and high stiffness require smaller time steps because of their high frequencies of vibration. Experience has shown that conventional building-scale structural frameworks can usually be modeled with a time step in the range of 0.5 to 0.1 milliseconds. Testing on a desktop computer with 3.4 GHz processor has found that two-dimensional models with up to 30 nodes and 40

elements can run in real time at a frame rate of about 30 frames per second. Models of this scale have limited application in commercial practice, but have wide application in teaching.

Note that using the 4<sup>th</sup>-order Runge-Kutta solution method requires that the state of the model be evaluated four times per time step, so that a time step of 0.1 milliseconds requires 40,000 model evaluations per second. Although this level of computational intensity was overwhelming during the era when computer-based structural analysis methods were first being developed, it is now feasible to perform such calculations in real time on conventional personal computers for small models.

### Computational Characteristics

The following are aspects of the computation process that distinguish it from conventional stiffness-based matrix structural analysis:

- **The method does not use global displacements:** The method keeps track of the absolute position, velocity, and acceleration of each node, but does not use the displacement (that is, the distance from the undeformed position to the deformed position). All computation is done with respect to the deformed position of the structure, without reference to the undeformed geometry.
- **Element stiffness matrices must be formulated in terms of deformations, rather than displacements:** Because the method does not use global displacements, the elastic forces in an element must be calculated based on the absolute positions of its nodes. Since the changes in node position may include large rigid body displacements, the element forces must be calculated by first determining the element deformations based on the node positions, and then determining the element forces based on those deformations. This process is discussed in more detail below for a two-dimensional frame element and a triangular membrane element.
- **All degrees of freedom must have mass:** Since the acceleration for each degree of freedom is calculated by dividing the force by the mass (or for rotational degrees of freedom, dividing the moment by the mass moment-of-inertia), it is necessary for each degree of freedom to have mass. The Arcade program enforces this requirement by assigning a portion of each element's self-mass to each of its nodes. If a node is not connected to an element, or the weight density of the element's material has been set to zero, the program issues a warning and assigns a small arbitrary mass to each degree of freedom of the node.
- **There is no global stiffness matrix:** Since the method does not use global displacements, it does not use a global stiffness matrix. The method does not employ a means to solve a system of simultaneous equations, so there is no special procedure required if a degree of freedom has zero stiffness. This means the method can easily model unstable structures.

### ELEMENT FORMULATION

The requirement to formulate element stiffness matrices in terms of element deformations is trivial for a truss element; the deformation is calculated as the current distance between nodes minus the at-rest length of the element. The formulation, however, becomes more

complex for other element types. The following discussion describes the formulation used in Arcade for a two-dimensional beam-column element and a constant-strain triangular membrane element.

### Displacement DOF Formulation For a Beam Element

For a conventional two-dimensional beam element, there are six displacement degrees of freedom: two translations and one rotation for each end node. When the stiffness is formulated in terms of the element deformation, the three rigid-body modes of deformation are removed, reducing the total degrees of freedom to three. Figure 2 shows the degrees of freedom used in the Arcade implementation, with forces denoted as  $R_i$  and displacements as  $r_i$ .



FIGURE 2

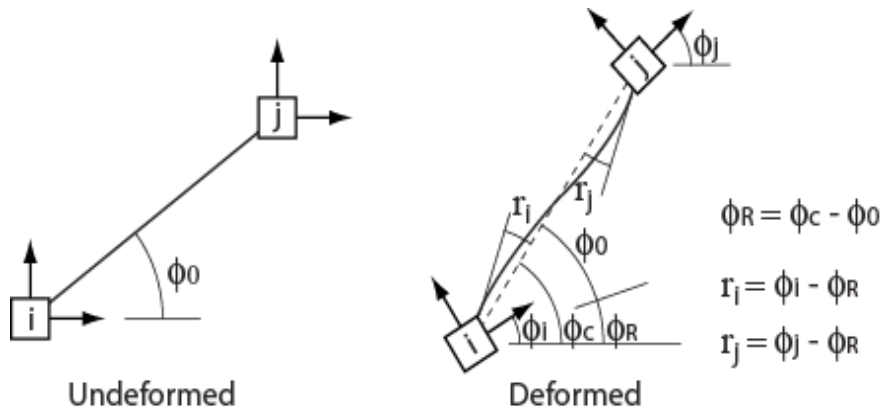
DEFORMATION DEGREES OF FREEDOM FOR A TWO-DIMENSIONAL BEAM-COLUMN ELEMENT

The deformations  $r_1$  and  $r_2$  correspond to the angle between the member tangent and the chord connecting the two nodes. The stiffness matrix relating the forces and displacements comes from elementary matrix theory [Gutkowski], as shown in (1) where  $E$  denotes the elastic modulus of the material,  $I$  denotes the moment of inertia of the cross section,  $A$  denotes the cross section area, and  $L$  denotes the length of the element:

$$\begin{bmatrix} R_1 \\ R_2 \\ R_3 \end{bmatrix} = \begin{bmatrix} \frac{4EI}{L} & \frac{2EI}{L} & 0 \\ \frac{2EI}{L} & \frac{4EI}{L} & 0 \\ 0 & 0 & \frac{AE}{L} \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} \quad (1)$$

The computation method requires that the deformation vector  $\mathbf{r}$  be calculated based on the current positions of the end nodes. The deformation vector is then multiplied by the stiffness matrix to obtain the force vector  $\mathbf{R}$ . The remaining element forces (i.e. the shears at each node, and the axial force at node  $i$ ) are then computed by statics. The opposites of the element end forces are then transformed to the global coordinate system and applied to the nodes, corresponding to box  $B$  of the flow chart in figure 1.

The most unusual step in this process involves calculating the rotational deformations based on the node positions, recognizing that those positions may include very large rigid body motions, including rotations exceeding 360 degrees. Figure 3 shows how the Arcade implementation calculates flexural deformations as relative rotations from the chord connecting the end nodes.



- $\phi_0$ : Chord angle for the undeformed structure.
- $\phi_c$ : Total rotation of the element chord
- $\phi_R$ : Rigid body rotation of the element chord.
- $\phi_i$ : Global rotation of the i node
- $\phi_j$ : Global rotation of the j node
- $r_i$ : Rotation of the i node relative to the element chord.
- $r_j$ : Rotation of the j node relative to the element chord.

FIGURE 3

COMPUTING ELEMENT ROTATION DEFORMATION BASED ON NODE POSITION

The rotation of the chord is determined by calculating its change during each time step and maintaining a running total, making it possible to account for chord rotations greater than 180 degrees in either direction.

This element formulation can be characterized as a small-strain, large-displacement model. The computation of element forces from element deformations is linear elastic, and does not account for interaction of axial and flexural stiffness. Accounting for such interaction can be achieved by using multiple elements, as demonstrated in the example shown in figure 4. In this textbook example [Salmon, 903] a portal frame braced against sidesway is modeled using 4 elements for each column and 2 elements for the girder. The model includes position boundary conditions modeling pin supports at the column bases and a vertical roller at the upper left corner of the frame. The nodes of the columns are arranged to produce a parabolic out-of-straightness of  $L/1000$ . The graph in the figure was generated by the program, with an added annotation marking the frame's theoretical buckling load of 10,600 kips (47.1 MN). The graph's vertical axis corresponds to the vertical reaction at the base of the left column, while the horizontal axis corresponds to the vertical displacement at the top of that column. The graph shows that the theoretical buckling load closely corresponds to the point where the load-displacement curve deviates from linear behavior.

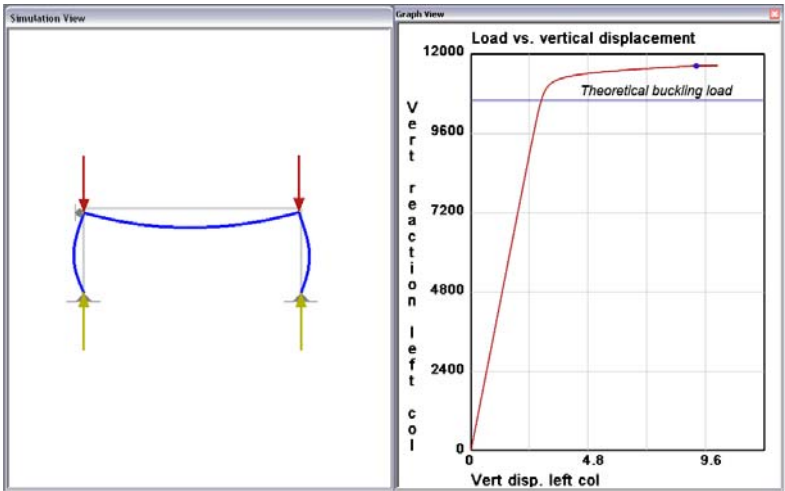


FIGURE 4  
BUCKLING ANALYSIS OF PORTAL FRAME BRACED AGAINST SIDESWAY

Figure 5 shows an example from the same textbook [Salmon, 906], where the same frame is not braced against sidesway. The nodes of the columns are arranged to be out of plumb by  $L/1000$ . For the graph in this case, the vertical axis corresponds to the sum of the two vertical reactions and the horizontal axis corresponds to the average of the vertical reactions. These more global measures of force and displacements are used because the vertical loads resisted by the two columns become different as the frame sways to the right as a result of the moment generated by the  $P-\Delta$  effect. The graph shows that the theoretical buckling load corresponds to the flat plateau of the load-displacement curve.

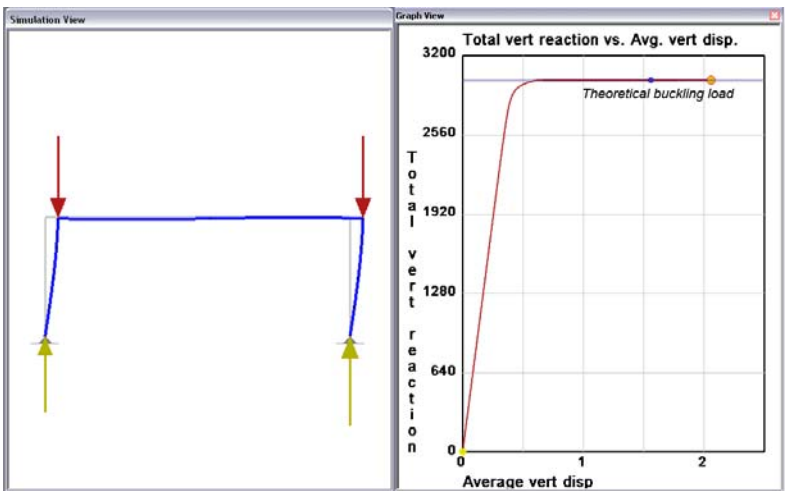


FIGURE 5  
BUCKLING ANALYSIS OF PORTAL FRAME NOT BRACED AGAINST SIDESWAY

These examples demonstrate that the computation method is able to model buckling and post-buckling response using extremely simple elements. The beam elements are the

simplest possible elastic elements, but the computation method is able to capture stability phenomena associated with large displacements.

### Displacement DOF Formulation for a Triangular Membrane Element

In two dimensions, a conventional constant-strain triangular membrane element has six displacement degrees of freedom, corresponding to two translation degrees of freedom for each node. When the three rigid-body modes are eliminated, three deformation degrees of freedom remain. Figure 6 shows the configuration used in Arcade for the three deformation degrees of freedom with respect to the local element coordinate system. At any time step, that coordinate system is defined so that the origin corresponds to the location of node  $i$ , and the  $x$  axis is oriented toward node  $k$ , and the  $y$  axis is oriented according to the right-hand rule with the  $z$  axis pointing out of the plane. With this definition, the local  $x$  and  $y$  coordinates of node  $i$  are always zero, as well as the  $y$  coordinate of node  $k$ . The remaining degrees of freedom are shown in the figure, with  $r_i$  corresponding to displacements and  $R_i$  corresponding to forces. The superscripts in the figure denote the value of the time step, with 0 being the initial value.

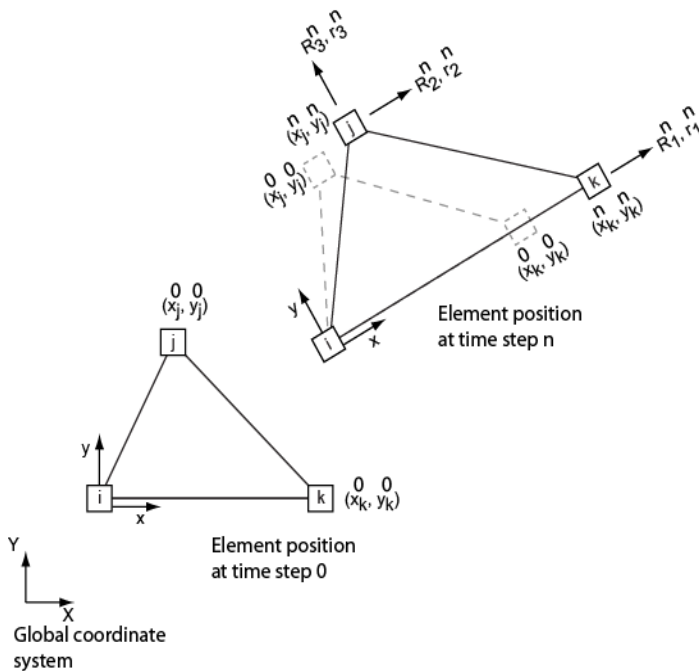


FIGURE 6

DEFORMATION DEGREES OF FREEDOM FOR TRIANGULAR MEMBRANE ELEMENT

The algorithm for calculating an element's deformation vector  $\mathbf{r}^n$  is as follows:

- Before the analysis begins, determine a transformation from global coordinates to element coordinates at time step 0:  $\mathbf{T}_{EG}^0$ .
- Also before the analysis begins, apply the above transformation to nodes  $j$  and  $k$  to determine the coordinates corresponding to the deformation degrees of freedom at time step 0:  $\mathbf{x}^0 = [x_k^0, x_j^0, y_j^0]$ . Store this vector for use during the analysis.

- At time step  $n$  during the analysis, use the current global coordinates of node  $i$  and node  $k$  to calculate the transformation from global coordinates to element coordinates:  $\mathbf{T}_{EG}^n$ .
- Apply the above transformation to nodes  $j$  and  $k$  to determine the coordinates corresponding to the deformation degrees of freedom at time step  $n$ :  $\mathbf{x}^n = [x_k^n, x_j^n, y_j^n]$ .
- Calculate the deformation vector at time step  $n$  as  $\mathbf{r}^n = \mathbf{x}^n - \mathbf{x}^0$ , where  $\mathbf{r}^n = [r_1^n, r_2^n, r_3^n]$

Once the deformation vector is known, the force vector  $\mathbf{R}^n$  corresponding to the forces for the deformation degrees of freedom can be calculated using the element stiffness matrix and the relationship  $\mathbf{R}^n = \mathbf{K}\mathbf{r}^n$ , shown in (2) where  $E$  is the elastic modulus of the material,  $t$  is the element thickness,  $\nu$  is Poisson's ratio and  $\Delta$  is as shown in (3).

$$\begin{bmatrix} R_1^n \\ R_2^n \\ R_3^n \end{bmatrix} = \frac{E|\Delta|t}{1-\nu^2} \begin{bmatrix} \frac{(1-\nu)}{2(y_j^0)^2} & 0 & \frac{(1-\nu)x_j^0}{4\Delta y_j^0} \\ 0 & \frac{1}{(y_j^0)^2} & \frac{-\nu}{2\Delta} \\ \frac{(1-\nu)x_j^0}{4\Delta y_j^0} & \frac{-\nu}{2\Delta} & \frac{(1-\nu)(x_j^0)^2 + 2(y_j^0)^2}{8\Delta^2} \end{bmatrix} \begin{bmatrix} r_1^n \\ r_2^n \\ r_3^n \end{bmatrix} \quad (2)$$

$$\Delta = \frac{-y_j x_k}{2} \quad (3)$$

The stiffness matrix is derived using fundamental finite element theory [Zienkiewicz, 20-30, 93-103], adapted to account for the three deformation degrees of freedom rather than the conventional six. The forces for the remaining degrees of freedom (e.g.  $x$  and  $y$  at node  $i$ , and  $y$  at node  $k$ ) are computed by statics.

As with the frame element, the computation of deformations is based on the element positions, from box  $A$  of the flow chart in figure 1. The computation of element forces corresponds to box  $B$  in that figure.

Finally, for the purpose of reporting element results, the stress vector  $\boldsymbol{\sigma}^n = [\sigma_x^n, \sigma_y^n, \tau_{xy}^n]$  for an element is determined using a stress recovery matrix as shown in (4).

$$\begin{bmatrix} \sigma_x^n \\ \sigma_y^n \\ \tau_{xy}^n \end{bmatrix} = \frac{E}{1-\nu^2} \begin{bmatrix} 0 & \frac{\nu}{y_j^0} & \frac{-y_j^0}{2\Delta} \\ 0 & \frac{1}{y_j^0} & \frac{-\nu y_j^0}{2\Delta} \\ \frac{(1-\nu)}{2y_j^0} & 0 & \frac{(1-\nu)x_j^0}{4\Delta} \end{bmatrix} \begin{bmatrix} r_1^n \\ r_2^n \\ r_3^n \end{bmatrix} \quad (4)$$

As with the stiffness matrix, this relationship is derived from fundamental theory, adapted to account for the three-degree-of-freedom formulation.

Figure 7 shows a simple test of the large displacement capabilities of the element. The model shows an unstable structure composed of four nodes and two triangle elements with a pin support at the node labeled *n1*. Under load, the structure swings like a pendulum about *n1*, and settles at a position rotated 90-degrees clockwise after damping halts the initial oscillation. The model does not include the effect of gravity. The properties of the structure are as follows: E-modulus 3,000 ksi (20.7 GPa), Poisson’s ratio 0.3, thickness 6 in. (0.152 m), width 10 feet (3.03 m) from *n2* to *n4*, length 20 feet (6.10 m) from *n1* to *n3*, weight density 150 lb/ft<sup>3</sup> (23.5 kN/m<sup>3</sup>). In its final position under load, the state of stress in each element is 2.78 ksi (19.2 MPa) normal stress on the global *y* axis, with effectively zero normal stress on the *x* axis and zero shear stress. The computed *y*-axis normal stress corresponds to the 1000 kip (4,450 kN) load divided by the average cross sectional area perpendicular to the axis through *n1* and *n3*. In the structure’s final position, the program calculates an elongation of 0.22 in (5.59 mm) from *n1* to *n3*, which results in a value of normal strain consistent with the values of stress and E-modulus. The program also calculates a shortening of 0.034 in (0.864 mm) from *n2* to *n4*, which is consistent with the value of Poisson’s ratio.

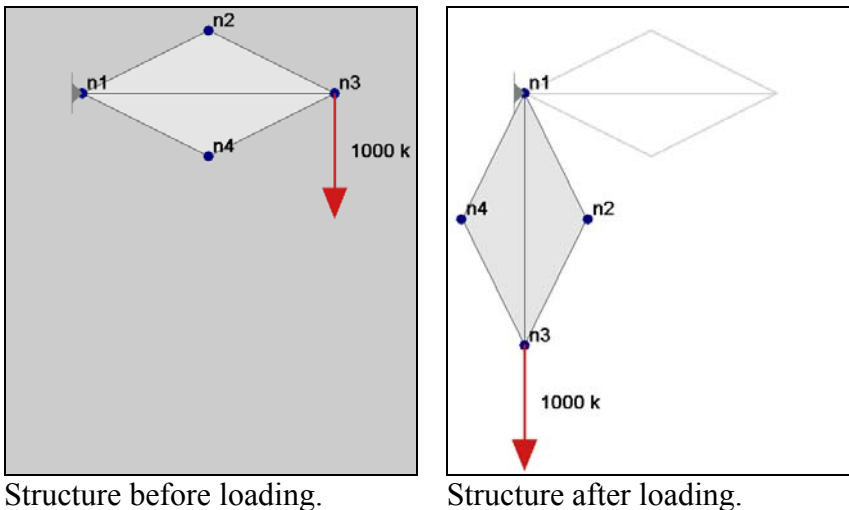


FIGURE 7  
SIMPLE TEST OF LARGE DISPLACEMENT CAPABILITIES OF CONSTRAINT-STRAIN TRIANGLE ELEMENT

**Rotational Mass and Frame Elements**

As described above, one of the distinguishing characteristics of the computation method is that it requires all degrees of freedom to have associated mass, so that accelerations can be determined from forces. While the computation of a lumped mass approximation is straightforward for translational degrees of freedom, it is more complex for rotational degrees of freedom. In conventional analysis, the mass associated with rotational degrees of freedom is commonly set to zero because it has little influence on the behavior of practical structures [Chopra, 324]. In this computation method, however, zero mass leads to an unacceptable division by zero. It is therefore necessary to assign a rotational mass moment of inertia to each node that reasonably models the inertia effects of the distributed mass of the structure. Figure 8 illustrates the fundamental problem of modeling the distributed mass of a beam using a lumped-mass frame element.

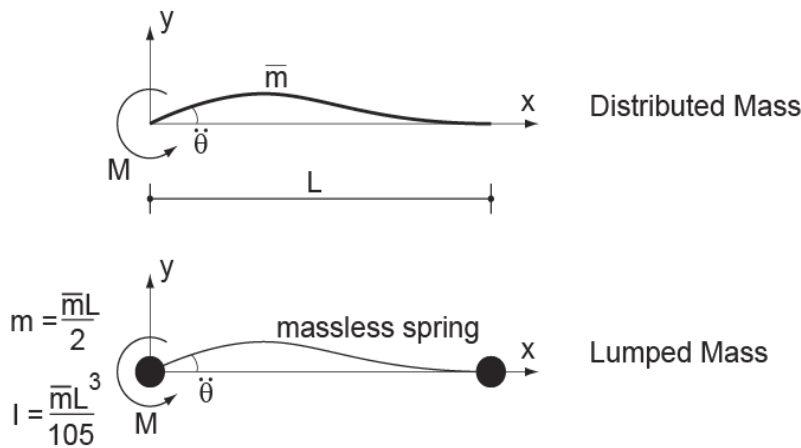


FIGURE 8  
MODELING ROTATIONAL INERTIA OF DISTRIBUTED MASS WITH A LUMPED MASS MODEL

The computation of the mass moment of inertia for each node is derived as follows:

- Assume the beam is subjected to a moment at one end which imposes a rotation while all other displacement are held at zero.
- Assume the deflection follows the corresponding Hermitian polynomial  $h(x)$ . With this assumption, the vertical displacement  $y(x)$  at any point along the element is equal to the end rotation  $\theta$  times  $h(x)$ :  $y(x) = \theta h(x)$  where  $h(x)$  is shown in (5).

$$h(x) = \frac{x^3}{L^2} - \frac{2x^2}{L} + x \tag{5}$$

- Impose a virtual displacement  $\delta\theta$  on the rotational degree of freedom.
- Set the virtual work done by the moment,  $M\delta\theta$ , equal to the virtual work done by the inertial forces of the distributed mass, which is the integral of the mass of a differential element, times the real acceleration at that point, times the virtual displacement at that point, as shown in (6) and figure 9.

$$M\delta\theta = \int_0^L (\bar{m}dx)\ddot{y}(x)\delta y(x) \quad (6)$$

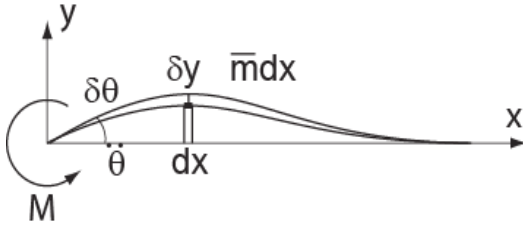


FIGURE 9

MODELING ROTATIONAL INERTIA OF DISTRIBUTED MASS WITH A LUMPED MASS MODEL

- Substituting for  $\ddot{y}(x) = \ddot{\theta}h(x)$  and  $\delta y(x) = \delta\theta h(x)$  into (6) yields (7)

$$M = \ddot{\theta} \left( \bar{m} \int_0^L h^2(x) dx \right) \quad (7)$$

- Substituting (5) for  $h(x)$  and solving yields (8), which produces the value for the rotational moment of inertia.

$$M = \ddot{\theta} \left( \frac{\bar{m}L^3}{105} \right) \quad (8)$$

For each frame element, the program adds the mass moment of inertia calculated in (8) to each of its nodes.

The accuracy of this approximation for a practical problem was tested by comparing the fundamental frequency of vibration of three models of a simply supported beam with varying numbers of elements to the theoretical value calculated in a textbook example [Englekirk, 225]. The properties of the example beam are as follows: span = 30 ft (9.14 m), E-modulus = 29,000 ksi (200 GPa), moment of inertia = 1090 in<sup>4</sup> (454x10<sup>6</sup> mm<sup>4</sup>), total distributed weight = 11.81 kips (52.4 kN). The computed natural frequency of vibration is 7.39 Hz.

The frequency of the computer models was determined by subjecting each model to an impulse load to set it in free vibration, and then using the Arcade program's graphing function to measure the time required to complete 5 cycles of vibration; this time was divided by five and then inverted to get the frequency. Note that a method of this type is required to determine the frequency since the computation method is limited to time history analysis, it cannot compute modal frequencies. The dynamic impulse load is applied as a total of 2000 kips, using point loads at each node to approximate a distributed triangular load, which is maximum at midspan, and zero at the ends. The impulse starts at zero at time zero, ramps linearly to a maximum at 1 millisecond, and back to zero at 2 milliseconds. The impulse produces a maximum midspan displacement

and end rotations ranging from 4.7 inches (119 mm) and 2.4 degrees for the 6-element model, to 5.6 inches (142 mm) and 3.0 degrees for the 2-element model. Figure 10 shows the model and graph for the two-element model, and figure 11 shows the same for the 6-element model. The results for the 2-, 4-, and 6-element models are summarized in table 1.

The results show that the computer models are reasonably close to the theoretical result. The two-element model predicts the frequency with an error of approximately 3 percent, while the error in the 4- and 6-element models is less than 1 percent. These results show that the approximation of distributed mass as lumped rotational mass does not introduce significant error for this practical problem, which is characterized by relatively high frequencies and rotations.

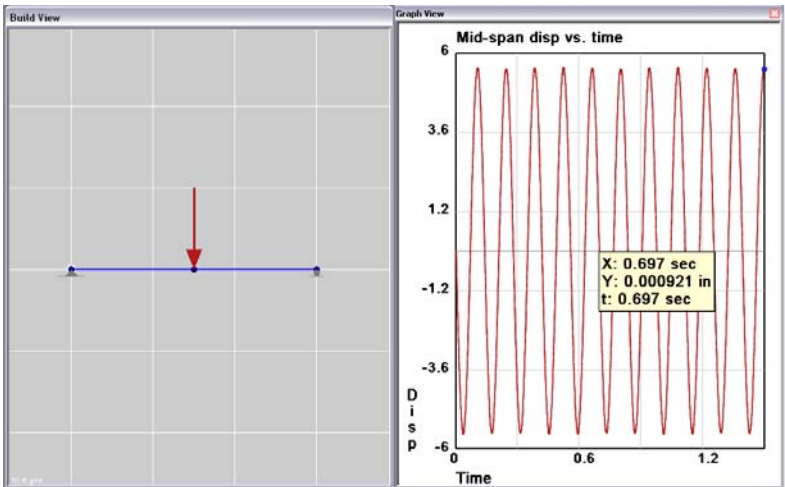


FIGURE 10  
MODEL AND GRAPH FOR 2-ELEMENT MODEL OF DYNAMIC BEAM, VALUES SHOWN AT END OF FIFTH CYCLE

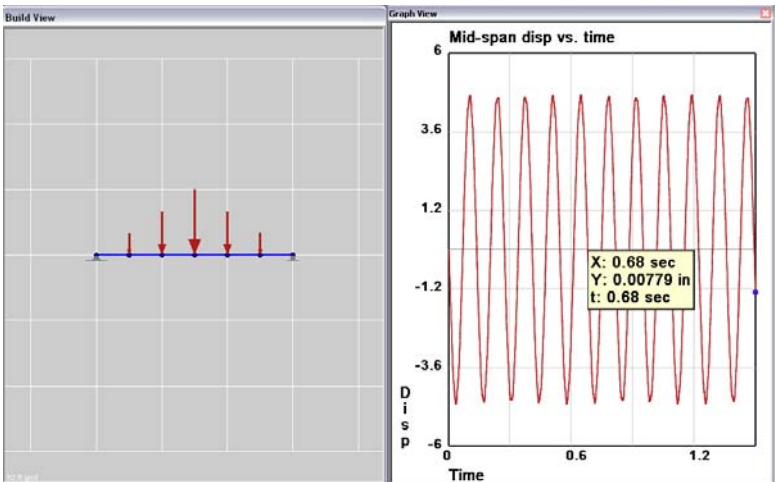


FIGURE 11  
MODEL AND GRAPH FOR 6-ELEMENT MODEL OF DYNAMIC BEAM, VALUES SHOWN AT END OF FIFTH CYCLE

No. Elements	Time for 5 cycles (sec)	Frequency (sec <sup>-1</sup> )	% Error
2	0.697	7.17	3.0
4	0.682	7.33	0.79
6	0.680	7.36	0.50

TABLE 1

MEASURED FREQUENCIES OF COMPUTER MODELS AND ERROR RELATIVE TO THEORETICAL RESULT

## TEACHING EXAMPLES

The examples of frame buckling described above clearly illustrate potential applications of the program in teaching non-linear behavior of frames. Indeed, the program was initially conceived to teach such aspects of structural behavior. Application of the program in practice revealed, unexpectedly, that the program was useful in teaching more elementary topics, including introductory statics. Figure 12 shows an introductory statics example concerning zero-force members in trusses. The figure uses a stop-motion rendering of the animated motion that occurs when a “zero-force” member is removed from a truss, leading to its collapse. The truss members are rendered so that the line thickness is proportional to the member force and the color represents the sense of tension or compression. The member is removed by clicking on it with an element-removal tool (graphically represented as a bomb), and the structure responds instantly on screen.

The truss collapses because the removed member served as a brace for two compression members. The lecture demonstration shows that when a zero-force member braces compression members its removal leads to collapse, but when it braces tension members it does not lead to collapse. This approach allows concepts of second-order behavior to be introduced in the earliest stages of the structures curriculum.

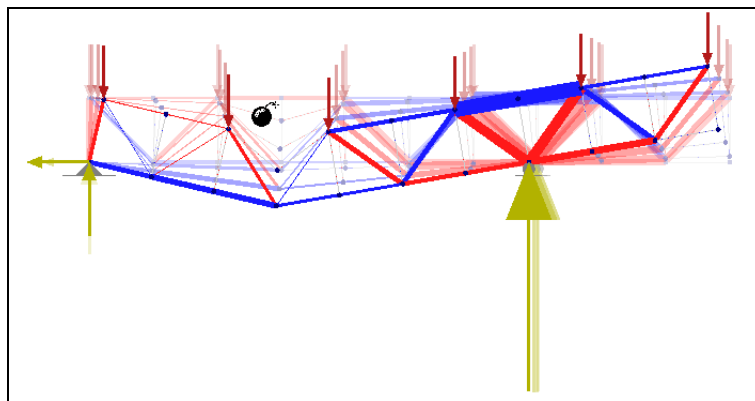


FIGURE 12

TRUSS EXAMPLE FROM AN INTRODUCTORY STATICS COURSE, ILLUSTRATING “ZERO FORCE” MEMBERS

Figure 13 shows another example from an introductory statics course. In this homework problem, students are given a body with a force system which is rotationally unbalanced. The assignment is to add exactly two forces that put the structure in equilibrium. The problem is unusual in that there are an infinite number of correct

answers, since there are infinite configurations of balancing couples. If a student's answer is incorrect, then the body rotates.

Experience has shown that with this program, non-linear, dynamic, time-history analysis can be introduced in the earliest stages of the undergraduate structures curriculum, and used throughout.

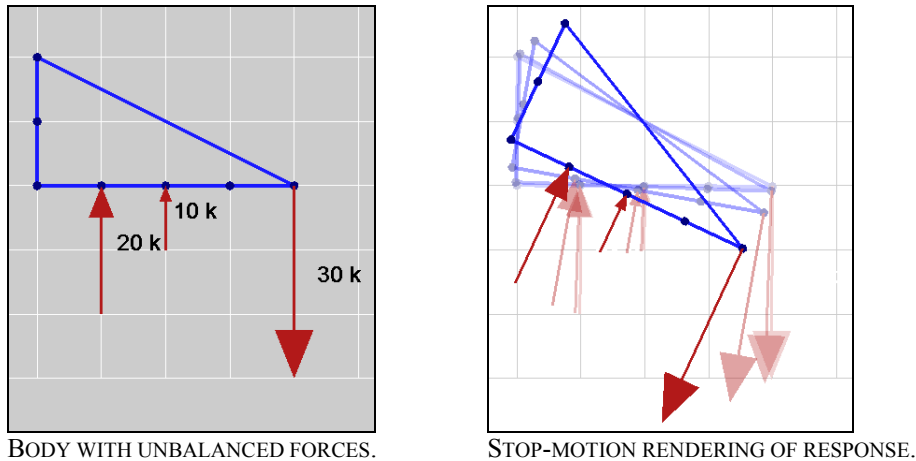


FIGURE 13  
STATICS PROBLEM BASED ON A BODY WITH A ROTATIONALLY UNBALANCED FORCE SYSTEM

## CLOSING

Although the physics engine approach has been widely used in computer games and graphics, it has received much less application in the modeling of structural frameworks. The paper has explained some of the computational issues involved in adapting and implementing the method to model structural engineering phenomena. The method makes it possible to model very large displacement phenomena such as post-buckling behavior using computationally simple elements.

The computational approach imposes the following restrictions:

- The element stiffness matrix must be formulated with respect to the element deformation degrees of freedom, so that no rigid body modes are present.
- All degrees of freedom must have associated mass.

The program's ability to model unstable structures and large displacements using a computationally simple approach has proven especially useful in teaching, where the program has been applied in a wide range of topics, beginning with elementary statics and ranging through advanced aspects of frame behavior. The Arcade program can be freely downloaded from the project web site [Martini, ref. 10].

## ACKNOWLEDGEMENT

This material is based upon work supported in part by the National Science Foundation under Grant No. 0230573. That support is gratefully acknowledged. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

## REFERENCES

- [1] Ageia Technologies Inc., *Ageia Corporate Web Site*, Mountain View, December 2005, <[www.ageia.com](http://www.ageia.com)>
- [2] Bourg, D, M, *Physics for Game Developers*, O'Reilly, 2002.
- [3] Chopra, A, K, *Dynamics of Structures*, Prentice Hall, 1995.
- [4] Englekirk, R *Steel Structures: Controlling Behavior Through Design*, J. Wiley, 1994.
- [5] Gutkowsky, R M, *Structures: Fundamental Theory and Behavior*, Van Nostrand Reinhold, 1980.
- [6] Havok.com Inc, *Havok Corporate Web Site*, Dublin, Ireland, December 2005, <[www.havok.com](http://www.havok.com)>
- [7] Hecker, C, "Physics, The Next Frontier", *Game Developer*, October/November, 1996, p.12-20.
- [8] Martini, K, "Real-time, Non-linear, Dynamic Simulation in Teaching Structures: Elementary to Advanced" *Proceedings of the 2005 American Society for Engineering Education Annual Conference*, 2005, CD-ROM publication.
- [9] Martini, K, "Teaching Structural Behavior with a Physics Engine", *Proceedings of 2005 Structures Congress*, American Society of Civil Engineers, 2005, CD-ROM publication.
- [10] Martini, K *Arcade: Interactive Non-linear Structural Analysis and Animation*, University of Virginia, 2005, <<http://www.arch.virginia.edu/arcade>>
- [11] Salmon, C, G, Johnson, J, E, *Steel Structures: Design and Behavior*, Addison Wesley Longman, 1996.
- [12] Smith, Russell, *Open Dynamics Engine*, December 2005, <[ode.org](http://ode.org)>
- [13] Witkin, A, Baraff, D, *Particle System Dynamics lecture notes for SIGGRAPH 1997 Course Physically Based Modeling: Principles and Practice*. Carnegie Mellon University <<http://www.cs.cmu.edu/afs/cs/user/baraff/www/sigcourse/notesc.pdf>>
- [14] Zienkiewicz, O, C, *The Finite Element Method*, 3<sup>rd</sup> ed., McGraw Hill, 1977.