Software development for assistance in the learning of structural analysis

Pedro Lopes

DECivil, Instituto Superior Técnico, University of Lisbon

November 2015

Abstract

A new teaching-oriented computer software capable of simulating the collision of particles and linear and non-linear behaviours of truss structures is presented in this paper. A physics engine based on a finite particle method (FPM) specially developed for this program is used. This engine uses the Newton's Laws of motion to describe the movement of single particles or sets of particles over time, which may be used to simulate particles collisions and structures (trusses) that may be discretized into sets of nodes/particles subjected to nodal forces. The explicit second order Taylor's method for the time integration is used to solve the equations of motion. The FPM does not require structures to be in equilibrium, instead, applies external forces to the corresponding particle elements and calculates the internal forces derived from the deformation of connection elements (rods) at the current time step. Alongside this engine, the program extensively uses a rendering engine and an event handler. This technique, commonly used by video games, is useful to create an environment in which physical phenomena can be simulated while giving the program's users the possibility to easily interact with the physical system in real time without restarting the simulation, intuitively understanding the laws at play. The comparison of the simulation results with the theoretical values in a set of specific tests showed small errors, demonstrating that the precision of this program, although not being on the levels usually required by engineering and scientific analysis, is enough to be used in a teaching environment.

Keywords: Finite particle method; Structural mechanics; Real time structural analysis; Non-linear analysis; Particle collisions; Interactive teaching tools.

1 Introduction

The teaching of the basics of structural mechanics in various engineering courses, namely Civil Engineering, is traditionally based on lectures, tutorials and written material. Yet, it is a field that has much to gain by exploiting the more recent educational advances, particularly those related to technological developments. Thus, a new computer program focused on the teaching of mechanics and structural analysis is developed in this work.

Using the principles behind the construction of video games, this software is developed as an interactive program capable of simulating a set of physical phenomena while displaying its results in real time to its users in an intuitive, graphical manner. As such, the program, similarly to video games, uses a physics engine, a rendering engine and event handler at its core. These engines communicate with each other in order to make the program work, that is, for the simulation results to be graphically represented and for the user input to be registered and the corresponding orders issued. With this combination, the program is capable of displaying an animation corresponding to the simulation in real time, and simultaneously accept new input that can change the physical system's properties.

In general, physics engines present in video games focus on simulating collisions and rigid body dynamics (Gregory, 2009). It is not to say that soft (deformable) bodies are not deserving of video game engineers' attention, as games that aim for higher levels of realism cannot ignore it. This has led to the development of engines capable of simulating brittle and ductile materials (O'Brien & Hodgins, 1999), (O'Brien, Bargteil, & Hodgins, 2002) and super-realistic cloth materials (Kim et al., 2013), among many others (Nealen, Müller, Keiser, Boxerman, & Carlson, 2006).

An altogether different approach in the field of structural mechanics software is that of the finite element method (FEM). This method is extensively used in many engineering analysis software, however, it loses some of its strength when faced with progressive failure due to strong nonlinearities and discontinuities (Yu, Paulino, & Luo, 2010). FEM variations have arisen that are capable of filling these gaps, successfully simulating the impact collapse of framed structures (Isobe & Lynn, 2004) and dynamic fracture and crack microbranching processes (Paulino, Zhang, & Celes, 2008). On the other hand, the less common finite particle method (FPM) seems to be better suited for large displacements simulations associated with failure mechanisms. It is a more computationally expensive method than FEM, and thus not very popular within the engineering community, yet, the tremendous increase in computational processing power over time is turning those concerns obsolete (Martini, 2001).

Usually, in FPMs, structures are composed of particles without volume but with mass inherited from the surrounding structure, being the Newton's Laws used to describe their motion (Yu et al., 2010). With it, the displacements of a structure over time, even when outside the scope of small displacements, can be easily described – something that FEM struggles with, rendering it less viable to analyse geometrical and material non-linear behaviours that involve large displacements.

The capabilities of a new teaching-oriented software by taking advantage of the characteristics of the FPM are presented in this paper.

2 Physics Engine

The physics engine is capable of collision detection and simulation, and of truss structures linear and non-linear behaviour simulations. The collision of particles is not based on the calculation of impulses, instead it simply calculates the new velocities of each particle after impact by enforcing the conservation of momentum and the variation of kinetic energy considering a coefficient of restitution (which can be defined by the users). It is also limited to the collision of two particles, at any given time, as multiple simultaneous collisions would require further analysis. In terms of truss structures, this engine simulates their linear elastic behaviour under static loads. It also simulates the yielding of structural elements and is capable of calculating large displacements and take them into account when calculating structural forces. These systems are modelled as a system of particles, that can be interconnected, and the corresponding physical phenomena is calculated based on a FPM.

2.1 Finite Particle Method

This method models the mentioned physical systems (colliding particles and truss structures) using two types of elements: particles and rods. Particles are elements with volume and an assigned mass to which the Newton's Laws of motion are applied. Their position over time is approximately calculated by solving the time step order two equation (1) .

$$\overrightarrow{x_{n+1}} = \overrightarrow{x_n} + \overrightarrow{v_n}\Delta t + \overrightarrow{a_n}\frac{\Delta t^2}{2} \tag{1}$$

The acceleration of particles is always a consequence of applied forces and the mass of the particle. If free, particles have an assigned mass that is independent of that of any other particles. If in a structure, however, their mass must be representative of the surrounding elements. In the latter case it is assumed that the mass of the structure concentrates at its nodes, the particles, whereby each particle is assigned half the mass corresponding to each attached rod.

When particles are not restrained, the forces applied to them are simply equal to the total external force. Otherwise, it is necessary to take into account the existence of internal forces. Internal forces appear as a consequence of the connected rods' deformation. These massless elements (particles inherit the mass of the structure) are responsible for interconnecting the particles that compose the structure and restrain their movements by applying their axial internal forces. The internal force within a rod is calculated based on the length at the current time, on the original resting position and the plastic deformation, as expressed by Hooke's Law equation (2).

$$F_{H_n} = EA(\varepsilon_n - \varepsilon_{p_n}) \tag{2}$$

Where F_{H_n} is the internal force at time t_n; E is the rod's elastic modulus; A is the cross-section area of the rod; and ε_n and ε_{p_n} are, respectively, the total strain and the plastic strain at time t_n.

Whenever this force is higher than the yielding limit, the plastic deformation of the force is updated (equation (3)).

$$\begin{cases} \varepsilon_{p_n} = \varepsilon_n - \varepsilon^{tension}_{elastic\ limit\ range}\ if\ \varepsilon > \varepsilon^{tension}_{elastic\ limit} \\ \varepsilon_{p_n} = \varepsilon_n - \varepsilon^{compression}_{elastic\ limit\ range}\ if\ \varepsilon \leq \varepsilon^{compression}_{elastic\ limit} \end{cases} \quad (3)$$

Where $\varepsilon^{tension}_{elastic\ limit\ range}$ and $\varepsilon^{compression}_{elastic\ limit\ range}$ are respectively the range of elastic deformation under tension and under compression and are calculated by equation (4). The elastic limit deformation in tension and in compression at time step n given by equation (5).

$$\begin{cases} \varepsilon^{tension}_{elastic\ limit\ range} = \dfrac{F^{tension}_{yielding}}{EA} \\ \varepsilon^{compression}_{elastic\ limit\ range} = \dfrac{F^{compression}_{yielding}}{EA} \end{cases} \quad (4)$$

$$\begin{cases} \varepsilon^{tension}_{elastic\ limit_n} = \varepsilon^{tension}_{elastic\ limit\ range} + \varepsilon_{pl_n} \\ \varepsilon^{compression}_{elastic\ limit\ \ n} = \varepsilon^{compression}_{elastic\ limit\ range} + \varepsilon_{pl_n} \end{cases} \quad (5)$$

Where $F^{compression}_{yielding}$ and $F^{tension}_{yielding}$ stand for the compressive and tensile strength, respectively.

Using these equations, the linear elastic – perfectly plastic model present in Figure 1 can be simulated.

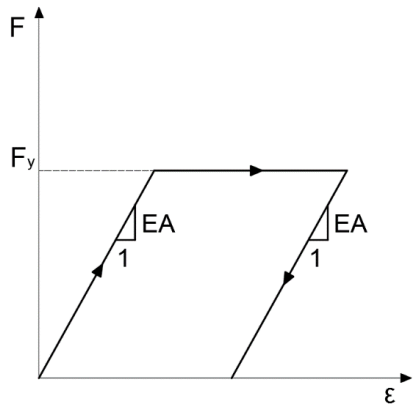

Figure 1 - Linear elastic – plastic material's force-strain relation. Loading and unloading. F_y stands for yielding force

The rod elements are also responsible for calculating the damping force and applying it to the particles. This force opposes movement and is calculated by equation (6).

$$F_d = -C \times v_n \quad (6)$$

Where F_d is the damping force; C the damping coefficient; and v_n the relative axial velocity of both end nodes (the particles).

The simulation of particle collisions is done in two phases (inside one same time step). First, it detects if a particle is overlapping another one and, if so, separates them by changing their position as expressed by equation (7). Then, changes the velocity over the collision axis of each particle in accordance to equations (8) and (9).

$$pos_{1_{n+1}} = pos_{1_n} \pm \frac{overlapping}{2} \tag{7}$$

$$v_{1_{n+1}} = \frac{C_r m_2 (v_{2_n} - v_{1_n}) + m_1 v_{1_n} + m_2 v_{2_n}}{m_1 + m_2} \tag{8}$$

$$v_{2_{n+1}} = \frac{C_r m_1 (v_{1_n} - v_{2_n}) + m_1 v_{1_n} + m_2 v_{2_n}}{m_1 + m_2} \tag{9}$$

3 Examples of application and model verification

3.1 Particle collisions

As stated, only collisions of two particles at a time are implemented. The verification of the simulation was done by analysing the obtained results to the ones theoretically expected in six different tests. These were: (*i*) perfectly plastic collision with equal masses; (*ii*) non-perfectly plastic collision with equal masses; (*iii*) elastic collision of two particles with the same mass; (*iv*) elastic collision with the resting particle's mass much higher than the moving one; (*v*) elastic collision with the moving particle many times heavier than the resting particle; (*vi*) non-head on elastic collision of two particles with the same mass. In each one of these cases particle number 2 is resting before collision and the other has a velocity of 5 m/s. Every collision is head-on and over the horizontal axis, except for the sixth. The obtained and the expected results for case *i* to *v* are presented in Table 1. Table 2 presents the obtained and the expected results for case *vi*.

Table 1 – Expected and obtained velocities comparison. Cases (i) to (v)

Collision	***Cr***	***Particle 1 Mass [kg]***	***Particle 2 Mass [kg]***	***Particle 1 Expected Velocity x [cm/s]***	***Particle 2 Expected Velocity x [cm/s]***	***Particle 1 Obtained Velocity x [cm/s]***	***Particle 2 Obtained Velocity x [cm/s]***
i	0	1	1	250.00	250.00	250.00	250.00
ii	0.5	1	1	125.00	375.00	125.00	375.00
iii	1	1	1	0.00	500.00	0.00	500.00
iv	1	1	10000	-499.90	0.10	-499.90	0.10
v	1	10000	1	499.90	999.90	499.90	999.90

Table 2 – Expected and obtained velocities comparison. Case (vi)

Collision	*Angle [rad]*	*Particle 1 Mass [kg]*	*Particle 2 Mass [kg]*	*Particle 1 Expected Velocity x [cm/s]*	*Particle 1 Expected Velocity y [cm/s]*	*Particle 2 Expected Velocity x [cm/s]*	*Particle 2 Expected Velocity y [cm/s]*
vi	0.524	1	1	125.22	-216.63	374.78	216.63
				***Particle 1 Obtained* Velocity x [cm/s]**	***Particle 1 Obtained* Velocity *y* [cm/s]**	***Particle 2 Obtained* Velocity x [cm/s]**	***Particle 2 Obtained* Velocity y [cm/s]**
				125.22	-216.63	374.78	216.63

3.2 Hyperstatic Structure

The program lets users quickly build systems as the one represented in Figure 2. The program's interface permits to easily change the properties of particles and rods (Figure 3), being that for the structure under analysis these are represented in Table 3. When the simulation is started, the nodes are moved depending on the forces applied to them, deforming the rods and, consequently, the internal forces of the system. This process is animated in real time, eventually leading to a stable solution (although in some cases the results might diverge), as presented in Figure 4.

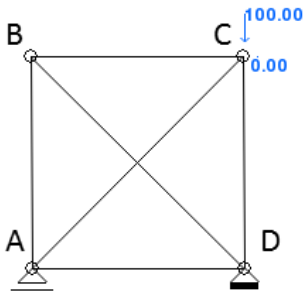

Figure 2 - Hyperstatic structure (interior indeterminacy) subjected to a nodal load

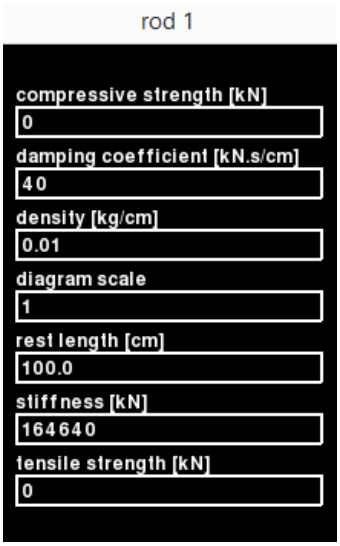

Figure 3 – Properties of a rod that are accessible to the user

Table 3 – Hyperstatic structure's properties

Rod	Rest Length [cm]	Stiffness [kN]	Tensile Strength [kN]	Compressive Strength [kN]
AB	1.50	164640	215	215
BC	1.50			
CD	1.50			
DA	1.50			
AC	2.12			
BD	2.12			

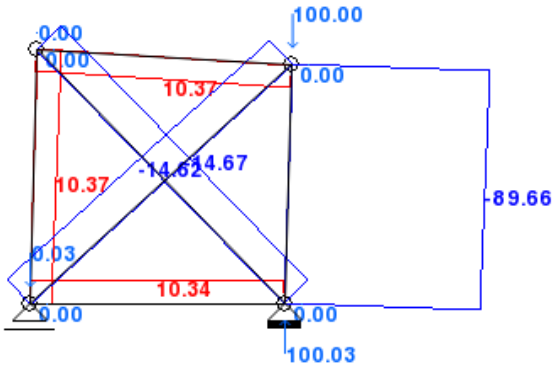

Figure 4 – Hyperstatic structure's loads, reaction forces and axial forces while subjected to a nodal load. Scale factor 100

Figure 4 shows that if the load is increased, element *CD* is the first element to yield. Thus, increasing the structure's load to 260 kN should lead to the yielding of rod *CD*. As this is a very interesting case to be studied, the program is prepared to highlight yielding rods as visible in Figure 5. Since the original structure is internally hyperstatic (1 degree of indeterminacy), no mechanism is formed. Note that, as expected, the axial forces and displacements of the structure are no longer proportional to the total load. If the load is further increased, the yielding of rods *BD* and *AC* follow, leading to the formation of a mechanism (Figure 6).

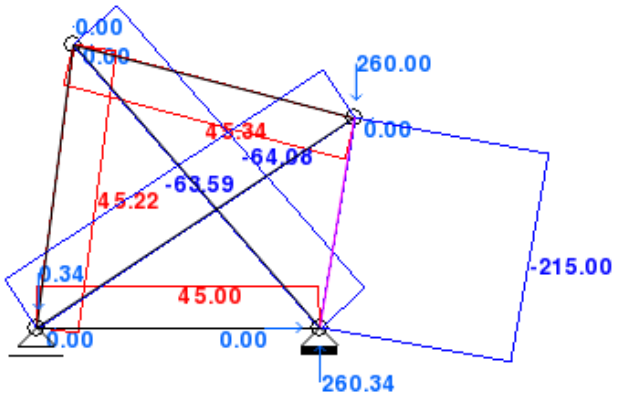

Figure 5 – Hyperstatic structure with a yielding rod (CD) and the rod's axial forces. No mechanism is formed. Scale factor 100

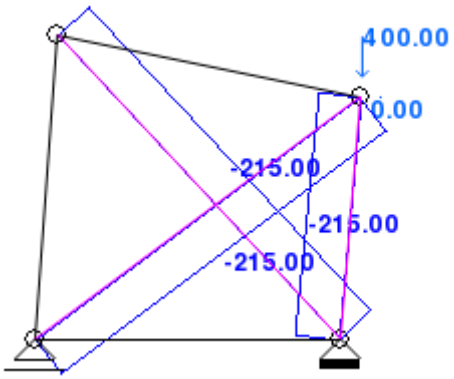

Figure 6 – Formation of a mechanism after the yielding of more than one rod

Demonstrating the force method is one of the most valuable capabilities included in this program. The force method is used to find the statically indeterminacies of hyperstatic structures through the use of the superposition principle (only applicable to linear domains). Thus, this program includes an adapted method that is simpler to implement but visually similar to the original. Users can apply it to find one (and only one) static indeterminacy, being visually led through the three stages of the method: (*0*) the original structure subjected to the original load; (*1*) a similar structure, but without the chosen static indeterminacy, (hereinafter called the base structure) subjected to the original load; (*2*) the base structure subjected to a single (for constraints) or a pair (for rods) of unit forces in place of the chosen indeterminacy. When using this adaptation of the force method, the user is presented with these three systems, being able to visualize the deformed structure, its forces and the value of the displacement corresponding to the chosen indeterminacy (the value used by the adapted method, not the original one). Finally, equation (10) is solved and the chosen indeterminacy is found and applied to the base structure. Figure 7 shows the three mentioned stages, being that the value of the indeterminate force has already been calculated and applied to system (2). The superposition principle can be verified as the forces of stage (*0*) match the sum of the forces from stage (*1*) and (2).

$$\Delta_0 = \Delta_1 + \Delta_2 p \qquad (10)$$

Where Δ_0, Δ_1 and Δ_2 stand for the displacements associated to the chosen indeterminacy in each one of three system.

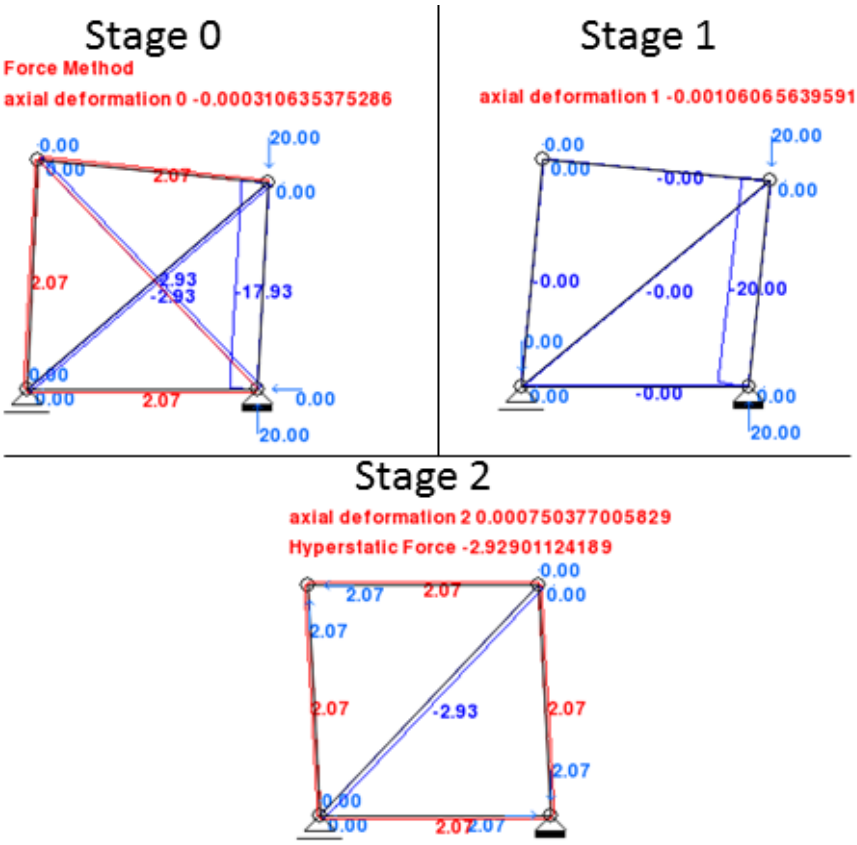

Figure 7 – The three stages of the (adapted) force method. The last stage already has the indeterminate force calculated

4 Common Errors

In what concerns the current program, there are two big categories into which most common errors fit: incompatible mechanical properties and numerical errors. To the first group belong errors that cannot be solved by reducing the time step while the latter are strongly associated with the time step. When stiffness and damping coefficients are too low, displacements might be too large for the simulation to converge to the expected solution. These are typically associated with particle displacements that significantly change the geometry of the structure or that yield rods that are not meant to yield for the given applied forces. It is possible for these problems to persist no matter how small the time step is.

On the other hand some problems are primarily caused by big time steps that are incompatible with not only the existing external forces but mainly with the stiffness and the damping coefficients of the rods. In these cases, where the last two mentioned properties cannot be altered, the best solution is to reduce the time step.

5 Conclusions

This work is a push to the development of new, more user-friendly, practical and intuitive didactic tools in the field of structural engineering. Using the fundamental aspects long used by the video games industry, this teaching-oriented software has been created with the purpose of contributing to increasing

the quality of structural engineering courses and giving students a means to autonomously refine and test their knowledge on this field.

The physical phenomena simulation, the displaying of the results in an intuitive manner and the possibility to interact with users, are treated as the three main pillars of the program. To implement these, the most common technique in the video games industry was used. That being, these aspects have been separated into a physics engine, a rendering engine and an event handler.

To implement the physics engine a finite particle method was developed in this work. This method was advantageous in relation to traditional methods, as it made it possible to simulate physical phenomena related to the collision of particles and to the behaviour of simple truss structures while displaying its results in real time in an animated fashion. The simulations carried out produced low relative error results for determinate and indeterminate truss problems under elastic and plastic situations and for the collision of two particles in a variety of situations.

6 References

Gregory, J. (2009). *Game engine architecture* (1st ed.). A K Peters, Ltd.

Isobe, D., & Lynn, K. M. (2004). A finite element code for structural collapse analyses of framed structures under impact loads. *European Congress on Computational Methods in Applied Sciences and Engineering*, *4*(July).

Kim, D., Koh, W., Narain, R., Fatahalian, K., Treuille, A., & O'Brien, J. F. (2013). Near-exhaustive precomputation of secondary cloth effects. *ACM Transactions on Graphics*, *32*(4), 87. http://doi.org/10.1145/2461912.2462020

Martini, K. (2001). Non-linear Structural Analysis as Real-Time Animation Borrowing from the Arcade. In *Computer Aided Architectural Design Futures 2001* (pp. 643–656). Springer.

Nealen, A., Müller, M., Keiser, R., Boxerman, E., & Carlson, M. (2006). Physically based deformable models in computer graphics. *Computer Graphics Forum*, *25*(4), 809–836. http://doi.org/10.1111/j.1467-8659.2006.01000.x

O'Brien, J. F., Bargteil, A. W., & Hodgins, J. K. (2002). Graphical modeling and animation of ductile fracture. In *ACM transactions on graphics (TOG)* (Vol. 21, pp. 291–294). http://doi.org/10.1145/566654.566579

O'Brien, J. F., & Hodgins, J. K. (1999). Graphical modeling and animation of brittle fracture. *ACM Transactions on Graphics*, *21*(3), 137 – 146. http://doi.org/10.1145/566654.566579

Paulino, G. H., Zhang, Z., & Celes, W. (2008). Dynamic Failure , Branching and Fragmentation Using Cohesive Zone Modeling. *XVIII Convegno Nazionale Gruppo Italiano Frattura*.

Yu, Y., Paulino, G. H., & Luo, Y. (2010). Finite Particle Method for Progressive Failure Simulation of Truss Structures. *Journal of Strucural Engineering*, *137*(10), 1168–1181. http://doi.org/10.1061/(ASCE)ST.1943-541X.0000321