Handling of plasticity in physics engine

September 24, 2016

1 Introduction

The application of modern computer game techniques enables the description of complex dynamic systems such as military vehicles with a high level of detail while still solving the equations in real-time. Film production and war games, in particular, is a key area that have benefited from simulation technology. In practice, games are often accomplished using an open-source platform such like ODE - Smith (2001-2007), Bullet Physics - Coumans (2003-2016) and Box2D - Catto (2007-2015).

Computational methods used in physics engines are divided to modules that handle collision detection and contact description and modules that handle solution of equations in real-time. Equations need to be solved can further be subdivided to be associated to motion, constraints and collisions. Velocity-based formulation is typically used in constraint based rigid body simulation. Friction is typically taken into account and mechanical joints are handled by constraint equations, Erleben (2005).

Plasticity is not typically taken into account in gaming solutions. Breaking of various objects typically takes place based on collision or impulse. Nevertheless, breaking of steel or reinforced concrete structures using this approach is not appropriate making a simulation to look unrealistic. Theory for handling of plasticity has been presented already in Terzopoulos and Fleischer (1988). Müller et al. (2004) and Müller et al. (2005) present a method for modeling and animating of elastic and plastic objects in real-time using point based animation. This approach is not been widely used in simulation applications. On major issue is collision handling of deformable objects.

This study will introduce an approach to account plastic deformation in game applications. In the introduced method, the plastic deformation takes place if force or moment exceeds given limit, deformation absorbs energy and joint breaks if plastic capacity is exceeded. The approach is based on using joint motors to model plasticity. Erleben (2005, p. 90) suggests similar method for modelling friction in joints. Adjacent objects are connected by motors. Motor power production limits are estimated based on plastic section modulus. Joint breaking is accounted by summing plastic deformation and comparing it to predefined material based limit. Elastic part of deformation is modelled by employing spring description which is based on modification of existing constraint in Bullet Physics.

Approach presented in this work can be used in gaming industry to provide more realistic simulations without significant extra work. For gaming purposes presented method works best in scenarios where connected parts are relatively heavy. This allows normal integration timestep to be used without stability issues. This kind of metodology also opens large area of combining old structural analysis methods to modern simulation frameworks.

2 Description of plasticity in the framework of physics engine

In this section, key concepts related to the introduced model are explained. Main differences between traditional structural analysis and physics engines are reviewed and discussed.

Velocity-based formulation is popular within physics based game developers and film production teams. Erleben (2005, p. 45) provides reasoning and theoretical details on why velocity-based formulation is popular in constraint-based rigid body simulation. Main reason is that collision handling can be done without additional procedures.

Background for velocity based formulation shown here is based on Erleben (2005, p. 45-50). In following section, these formulations will be clarified by simple example using Bullet Physics implementation. Impulse $\vec{J}$ in the time interval Δt can we written as

$$\vec{J} = \int_0^{\Delta t} \vec{f}_{true}(t)dt \tag{2.1}$$

where $\vec{f}_{true}(t)$ is force.

Using Newton's second law of motion $\vec{F} = m\vec{a}$ one can solve for the velocity, $\vec{v}^{\Delta t}$

$$\int_0^{\Delta t} m\frac{d\vec{v}}{dt}dt = \int_0^{\Delta t} \vec{f}_{true}(t) \tag{2.2}$$

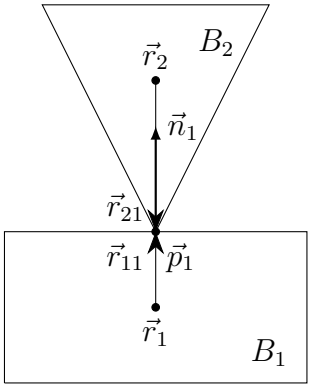
$$m(\vec{v}^{\Delta t} - \vec{v}^0) = \vec{J} \tag{2.3}$$

where superscripts denote time, i.e. $\vec{v}^{\Delta t} = \vec{v}(\Delta t)$. Next position can be found by integrating the velocity. Updates after each step can be summarized as Equation 2.4 for locations and Equation 2.5 for velocities.

$$\vec{s}^{t+\Delta t} = \vec{s}^t + \Delta t S\vec{u}^{t+\Delta t} \tag{2.4}$$

$$\vec{u}^{t+\Delta t} = \vec{u}^t + \Delta t M^{-1}(CN\vec{f}^{t+\Delta t} + \vec{f}_{ext}) \tag{2.5}$$

Symbols are summarized in Table 2.1 and Figure 2.1. Figure 2.1 describes collision of two bodies B_1 and B_2.



$\vec{r}_i$ = orientation of body i
$\vec{p}_k$ = orientation of contact point k
$\vec{r}_{ik} = \vec{p}_k - \vec{r}_i$
$\vec{n}_k$ = normal for contact point k

Figure 2.1: Illustration of nomenclature for equations of motion for contact.

$\vec{r}_i$ is orientation of body i. $\vec{p}_k$ orientation of contact point k. $\vec{r}_{ik}$ is vector between center of gravity of body i and contact point k. $\vec{n}_k$ is contact normal for contact point k. With this formulation, simplified scenario can be simulated.

Friction in contacts and joint constraints can be handled in unified way by refactoring equation 2.5 to get 2.6, Erleben (2005, p. 66-67)

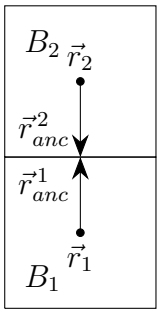
$$\vec{u}^{t+\Delta t} = \vec{u}^t + \Delta t M^{-1}(J^T_{contact}\vec{\lambda}_{contact} + J^T_{joint}\vec{\lambda}_{joint} + \vec{f}_{ext}) \tag{2.6}$$

Symbol	Description
$\vec{r}_i$	position of center of mass for body i
$\vec{q}_i$	orientation for body i as quaternion $[s_i, x_i, y_i, z_i]^T$
$\vec{p}_i$	contact or joint point i
$\vec{r}_{ki}$	$\vec{p}_k - \vec{r}_i$
$\vec{s}$	$[\vec{r}_1, \vec{q}_1, ..., \vec{r}_n, \vec{q}_n]^T$
Q_i	rotation of quaternion $\vec{q}_i$ as matrix where $\frac{1}{2}\vec{\omega}_i\vec{q}_i = Q_i\vec{\omega}_i$ $\frac{1}{2}\begin{bmatrix} -x_i & -y_i & -z_i \\ s_i & z_i & -y_i \\ -z_i & s_i & x_i \\ y_i & -x_i & s_i \end{bmatrix}$
S	generalized transformation matrix $S \in \mathbb{R}^{7n \times 6n}$ $\begin{bmatrix} 1 & & & 0 \\ & Q_i & & \\ & & \ddots & \\ & & & 1 \\ 0 & & & Q_n \end{bmatrix}$
$\vec{v}_i$	linear velocity of center of mass for body i
$\vec{\omega}_i$	angular velocity of center of mass for body i
$\vec{u}$	$[\vec{v}_1, \vec{\omega}_1, ..., \vec{v}_n, \vec{\omega}_n]^T$
M	generalized mass matrix $M \in \mathbb{R}^{6n \times 6n}$ $\begin{bmatrix} m_i 1 & & & 0 \\ & I_1 & & \\ & & \ddots & \\ & & & m_n 1 \\ 0 & & & I_n \end{bmatrix}$
I_i	inertia tensor for body i
C	contact condition matrix $C \in \mathbb{R}^{6n \times 3K}$
N	contact normal matrix $N \in \mathbb{R}^{3K \times K}$

Table 2.1: Nomenclature for equations of motion

where jacobian terms $J^T_{contact}$ for joints are derived by taking time derivates of kinematic constraints.

Added symbols are summarized in Table 2.2 and Figure 2.2. Figure 2.2 shows terms needed for joint processing.



$\vec{r}_i$ = orientation of body i
$\vec{r}^{\,i}_{anc}$ = body frame vector i

Figure 2.2: Illustration of nomenclature for equations of motion for joint.

$\vec{r}^{\,i}_{anc}$ is used to define at which point joint constraint is applied.

Constraint processing in Bullet Physics is based on ODE, Smith (2001-2007). Mathematical background and detailed examples are available by Smith (2002). Joints are also discussed in detail in Erleben (2005, p. 60-90). Equations 2.7, 2.8 and 2.9 are created for each constraint.

Symbol	Description
$J_{contact}$	Jacobian matrix for contacts
$\lambda_{contact}$	vector of lagrange multipliers for contacts
J_{joint}	Jacobian matrix for joints
λ_{joint}	vector of lagrange multipliers for joints

Table 2.2: Additional terms for generalized equations of motion

$$J_1\vec{v}_1 + \Omega_1\vec{\omega}_1 + J_2\vec{v}_2 + \Omega_2\vec{\omega}_2 = \vec{c} + C\vec{\lambda} \tag{2.7}$$

$$\vec{\lambda} \geq \vec{l} \tag{2.8}$$

$$\vec{\lambda} \leq \vec{h} \tag{2.9}$$

In following section, these equations will be clarified by simple example. Main parameters and corresponding fields in Bullet Physics are described in Table 2.3.

Parameter	Description	btConstraintInfo2 pointer
J_1, Ω_1 J_2, Ω_2	jacobian	m_J1linearAxis, m_J1angularAxis m_J2linearAxis, m_J2angularAxis
$\vec{v}$	linear velocity	
$\vec{\omega}$	angular velocity	
$\vec{c}$	right side vector	m_constraintError
C	constraint force mixing	cfm
$\vec{\lambda}$	constraint force	
$\vec{l}$	low limit for constraint force	m_lowerLimit
$\vec{h}$	high limit for constraint force	m_upperLimit

Table 2.3: Constraint parameters

In structural analysis, a formulation and associated numerical solution procedure are selected based on needed features. Often, finite element method is used. In most cases, static solution with assumption of linear strain-displacement relation using displacement based boundary conditions is used. Bathe et al. (1975) provides description for handling of various nonlinearities. In large displacement analysis, formulation may be based on updated formulation (Eulerian) or Lagrangian formulation where initial configuration is used. Further enhancements are material nonlinearity and dynamic analysis. Physics engine provides dynamic analysis with large reference translations and rotations while assuming bodies to be undeformable.

Material plasticity can be accounted in games by using suitable coefficient of restitution. This provides reasonable means to simulate loss of energy in collisions. Simulation of breaking of objects made of ductile material can be made more realistic by splitting rigid bodies to multiple bodies which are connected by energy absorbing joints. Typical engineering stress-strain curve of ductile steel is shown in Figure 2.3.

where σ is stress, E is Youngs modulus and f_y is yield stress. Engineering stress and strain mean that original dimensions are used in stress calculation, Dowling (2007, p. 108). Stress-strain curve is not drawn to scale as elastic strain could not be seen as it is typically 0.001 to 0.005.

In this work elastic-fully plastic material model is used in most scenarios. Having elastic part allows elastic displacements for slender structures. Elastic material behavior is ignored in approach

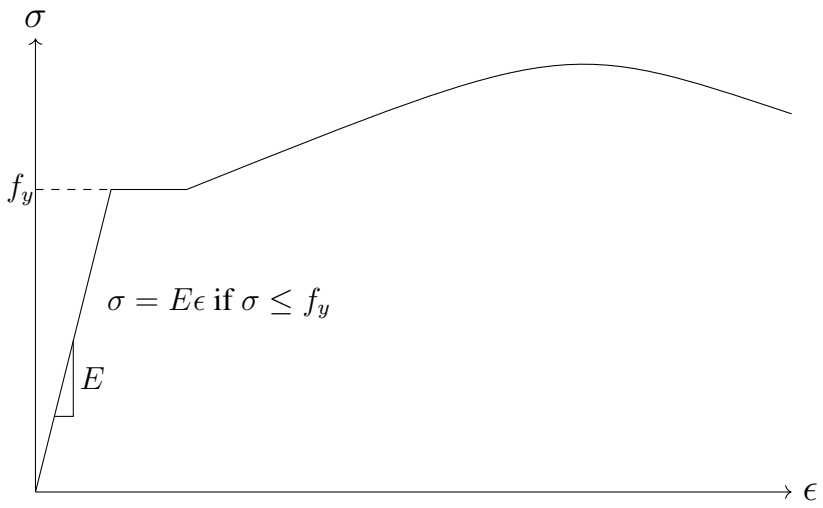


Figure 2.3: Engineering stress-strain curve of ductile steel (not to scale).

introduced in this work provided that deformation is related to higher frequency than integration stability would allow. It should be noted that geometry of objects is not updated during analysis and thus engineering stress-strain properties should be used.

Strain hardening is taken into account in this work mainly by assuming that plasticity in bending expands, Dowling (2007, p. 672). Material that starts to yield first is hardened and yielding moves slightly. This can be seen e.g. by bending paperclip. It does not break at low angles but can take few full bends.

Difference between elastic and plastic section modulus is shown in Figure 2.4.

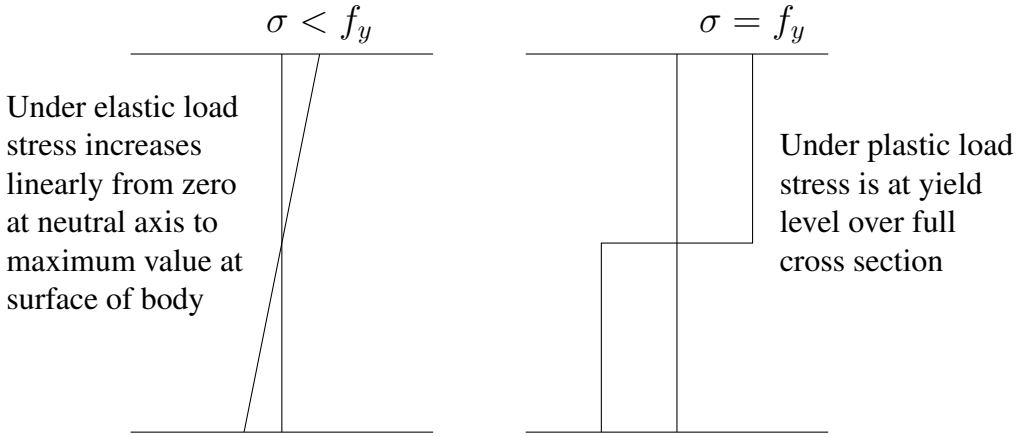


Figure 2.4: Axial stress distribution over cross section for bending under elastic and plastic loads.

If stress is below yield limit f_y, stress and strain are linear within material. If cross section is fully plastic, stress is assumed to be at yield level over whole cross section and so plastic section modulus is higher than elastic section modulus.

Plasticity is handled by defining maximum forces in Equations 2.8 and 2.9 using plastic capasities as maximum forces.

Maximum force acting in a direction of $\vec{r}_{anc}^{\,i}$ is product of area and yield stress as follows

$$N_{max} = \int_A f_y \tag{2.10}$$

Maximum forces acting perpendicular to $\vec{r}_{anc}^{\,i}$ are product of area and shear yield stress τ_y as follows

$$Q_{max} = \int_A \tau_y \tag{2.11}$$

Maximum moments acting around axis perpendicular to $\vec{r}^{\,i}_{anc}$ are integrals of perpdendicular distance and yield stress f_y as given for the moment around x-axis and, correspondingly, moment around z-axis.

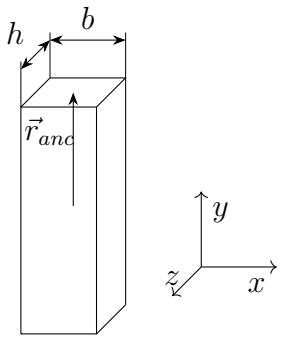
$$M^x_{max} = \int_A z f_y \tag{2.12}$$

$$M^z_{max} = \int_A x f_y \tag{2.13}$$

Maximum moment around $\vec{r}^{\,i}_{anc}$ is integral of distance d from joint point and shear yield stress τ_y as

$$M^y_{max} = \int_A d\tau_y \tag{2.14}$$

Maximum forces and moments for rectangular section using constant yield stress are summarized in Figure 2.5. Yield shear stress is assumed to be $0.5\, f_y$ using Tresca yield critetion. If von Mises yield criterion is used 0.5 is replaced by 0.58 ($1/\sqrt{3}$). These are not exact values in multiaxial stress state but they should be quite usable in most cases.



maximum force in x and z directions = $0.5\, b\, h f_y$
maximum force in y direction = $b\, h\, f_y$

maximum moment around x-axis = $0.25\, b\, h^2\, f_y$
maximum moment around y-axis $\approx 0.19\, b\, h\, \frac{b+h}{2} f_y$
maximum moment around z-axis = $0.25\, b^2\, h\, f_y$

Figure 2.5: Maximum forces in plastic joint for rectangular cross section.

For rotation around y-axis there is closed form solution only for circular cross sections. Given approximation is best suited for cases where b and h are quite similar. Better approximation for any given b and h can be obtained by integrating distance from center of joint over cross section and multiplying it with yield shear stress e.g. using octave. Example of calculation of maximum moment around $\vec{r}^{\,i}_{anc}$ is shown in Figure 2.6

```
b=0.01; h=0.01; fy=200e6;
wpy=fy/2*dblquad(@(x,z) sqrt(x.*x+z.*z),-b/2,b/2,-h/2,h/2)
38.2
```

Figure 2.6: Calculation of maximum moment around $\vec{r}^{\,i}_{anc}$.

Basic idea in this work can be tested with any framework having motors and hinge constraints. This can be done by setting target velocity of motor to zero and limiting maximum motor impulse to plastic moment multiplied by timestep.

Further enhancements were created and tested by forking Bullet Physics source code and adding new constraints Nikula (2014-2016).

3 Numerical examples

In this section, changes to constraint formulation are clarified by doing few simulation steps of non constrained, rigid and elastic-plastic examples using numerical values.

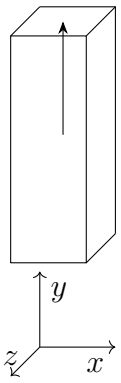


Figure 3.1: Single body model for plasticy processing demonstration.

System has only one dynamic rigid body which is three meters high block of 0.04 % steel enforced concrete. Cross section is one square meter. Constraint is set so that connecting frame is at top of block. In this case frame could be anywhere but if multiple bodies are involved, connecting frame must be defined so that it reflects scenario under investigation. Concrete density is $2000\frac{kg}{m^3}$ and steel density is $7800\frac{kg}{m^3}$. Steel is assumed to handle load in elastic-plastic case. Yield stress of steel is 200 MPa. Gravity is $10\frac{m}{s^2}$. Simulation step is $\frac{1}{60}s$ and 10 iterations are done for single step. Body is 1.5 meters above rigid ground. In unconstrained case body will hit hit ground at t=0.548 s ($\sqrt{2 \cdot 1.5/10}$).

Single simulation step is done using following substeps. In this work, changes are made only to constraint setup and update actions.

1. Apply gravity to each non static body. This step makes programming easier as otherwise each program should add forces due to gravity in each step. In this case, $\vec{f}_{ext}$ in Equation 2.6 gets added by $\{0, -60000, 0\}$.

2. Predict unconstrained motion for each body controlled by physics simulation. Static and kinematic bodies are not processed in this step. Prediction is done based on current linear and angular velocities of the body.

3. Predict contacts. In this phase, continuous collision detection is done based on simplified objects. Each rigid body is represented by sphere geometry and contact prediction is done if body moves more than given threshold value during simulation step. Continuous collision detection is configured for each body. Typical scenario that requires continuous collision detection is fast moving bodies that would otherwise go through walls.

4. Perform discrete collision detection. All overlapping bodies are processed and manifoldPoints are created for each detected contact at end of simulation step. In unconstrained case, contact is detected after t=0.55 s if continuous collision detection is not used and manifolds distance gets negative value (-0.058 m).

5. Calculate simulation islands(groups). Bodies that are near each other based on contact prediction or discrete collision detection or connected with constraints are grouped in same group.

6. Solve constraints. Both contact and other constraints are processed in this step. This step is subdivided to setup, iterations and finish phases. In setup phase constraints are queried for positional errors for calculation of c in Equation 2.7 and maximum and minimum impulses in Equations 2.8 and 2.9. In finish phase constraint forces are calculated if requested and velocities of bodies are updated.

7. Integrate transforms using velocities modified in previous step.

8. Update actions (callbacks) are called. In elastic-plastic case, plasticity is summed and equilibrium point of elastic part is updated if maximum force or moment is exceeded.

9. Activation state of bodies is updated. To avoid extra calculation bodies are as default put to sleeping state if linear and angular velocities of body are less than given threshold values (default 0.8 and 1.0) longer than set time limit (2.0).

Equation 2.7 is simplified to 3.1 in constrained cases.

- No rotation takes place. ω_1 and ω_2 are zeros.
- Constraint force mixing can be ignored.
- Only vertical velocity is handled.
- Other involved body is rigid and it does not move.

$$mv_y = c \tag{3.1}$$

Equations 2.8 and 2.9 are not active for fixed case. For elastic-plastic case maximum impulse is set to product of yield stress, area of steel enforcement and time step (1330).

Method btSequentialImpulseConstraintSolver::solveGroupCacheFriendlySetup in Bullet Physics was used to pick up values for internal variables described in Figure 3.2.

Actual values for unconstrained case without continuous collision detection are shown in table 3.1. Penetration is detected at time 0.567 s when *velError* is 5.67 (5.5+0.167) and *posError* is 2.8 (0.058*0.8/0.0167). Impulse is 34000 and contact force is thus about 2 MN (34000/0.0167). After few steps location and position stabilize although internally calculation is needed for each time step until body is deactivated.

Time	**Location**	*velError*	*penetration*	*posError*	*rhs*	**Velocity**	**Impulse**
0.017	0					-0.17	0
0.550	-1.558					-5.5	0
0.567	-1.511	5.67	-0.058	2.8	21270	0.01	34000
0.583	-1.502	0.14	-0.011	0.54	2570	0.55	420
0.600	-1.496	-0.38	-0.002	0.1	-1000	0.38	0
0.617	-1.492	-0.44	0.004	0	-1600	0.22	0
0.717	-1.497	0.004-0.08	-0.0003-0.001	0-0.01	10-400	-0.01	400
0.817	-1.499					-0.08	700
0.917	-1.500					-0.001	1000

Table 3.1: Simulation values for unconstrained case. For internal contact values typical values are shown as number of contacts and detailed values differ.

Actual values for unconstrained case with continuous collision detection (ccd) using 1.5 as radius of ccd sphere and 0.001 as ccd motion threshold are shown in 3.2. Collision is detected at time 0.550 s when *velError* is 3.5 (5.34+0.167)-0.033/0.0167 and *posError* is 0 as collision is detected before penetration. It should be noted that in general ccd sphere should not extend actual body as premature contacts are created if collision takes place in those regions.

Values for fixed constraint are shown in table 3.3. Constraint is activated in second step and positional error is corrected about 20 % in each step as requested by using erp value 0.2.

There are currently two alternative six-dof-spring constraint implementations in Bullet Physics and in this work elastic-plastic versions of both of them are developed.

velError is calculated using velocities and external impulses of connected bodies.

In constraint cases, main contibutor is bodies relative speed at joint point.

In contact case, main contributor is bodies relative speed at point of contact. Bodies are not allowed to penetrate each other. Restitution increases velError. If contact is not penetrating velError is reduced by $penetration * timeStep$.

posError is calculated by constraint. It is significant factor in designing stable constraints.

In fixed case, value is about 12 times actual position error. Factor 12 is based on time step (60) and default value of error reduction parameter (erp) which has value of 0.2 in this context.

In elastic plastic case, value is set to zero if impulse would be larger than maximum impulse or spring simulation cannot be done in stable way.

In contact case, value is zero if there is no penetration. For penetration cases it is $\frac{-penetration\,erp}{timeStep}$. In contact cases default value for erp is 0.8.

rhs (c) is calculated by velError jInv + posError jInv

jInv is calculated using masses and inertias of connected bodies and constraint geometry.

In constraint cases, it is mass of body (6000).

In contact case, it varies below mass of body.

Impulse is impulse applied to body during timestep.

In constraint cases, it is obtained from btJointFeedback structure.

In contact case, it is obtained by summing applied impulses from active manifolds.

erp Error reduction parameter (0...1) is used to handle numerical issues e.q. object drifting. Setting erp to 1 would in theory eliminate error in one step but in practice value of 0.2 - 0.8 is used in most cases.

Figure 3.2: Internal variables used in Bullet Physics in constraint solving.

Time	**Location**	*velError*	*penetration*	*posError*	*rhs*	**Velocity**	**Impulse**
0.017	0					-0.17	0
0.550	-1.500	3.5	0.033	0	21000	-2	21000
0.567	-1.500	2.17	0	0	8100	0.01	13000
0.583	-1.500	0.15	0	0	600	0	937
0.600	-1.500	0.17	0	0	600	0	1040

Table 3.2: Simulation values for unconstrained case with continuous collision detection.

Table 3.4 summarizes most significant parameters for this study. There are also e.g. parameters for controlling joint motors. Additional equation is created if for each additional constraint. Enabled springs and motors add one row and limits add one if upper and lower are same and two if both upper and lower limits are defined.

Values for elastic-plastic case are shown in tables 3.5. Body drops freely during first simulation step and gains enough kinetic energy so that higher impulses are needed in few following steps. This causes plastic strain during next three steps.

Six-dof-spring constraint 2 has optional feature to avoid unstability by automatically softening constraint spring. Feature is activated if current timeStep is too large for spring-mass system simula-

Time	**Location**	*velError*	*posError*	*rhs*	**Velocity**	**Impulse**
0.017	-0.0028				-0.17	0
0.033	-0.0022	0.33	-0.033	-2200	0.033	2200
0.050	-0.0018	-0.13	-0.027	-960	0.027	960
0.067	-0.0014	-0.14	-0.021	-970	0.021	970
0.35...	0	-0.17	≈ 0	-1000	0.0	1000

Table 3.3: Constraint parameter values for fixed constraint

Parameter	**Description**
lowerLimit	minimum allowed translation or rotation
upperLimit	maximum allowed translation or rotation
springStiffness	elastic spring stiffness
enableSpring	defines if spring is active
springStiffnessLimited	should elastic behaviour be tuned to avoid instability
equilibriumPoint	should elastic behaviour be tuned to avoid instability
currentLimit	describes state of constraint 0: not limited 3: loLimit=hiLimit 4: current value is between loLimit and HiLimit

Table 3.4: Selected constraint parameters for elastic-plastic constraint 2

Time	**Location**	*velError*	*posError*	*rhs*	*velocity*	**Impulse**	**Plastic strain**
0.017	-0.0028	-0.17	0	-1000	-0.17	0	0
0.033	-0.0046	-0.33	0	-2000	-0.11	1330	0.001
0.050	-0.0056	-0.28	0	-1670	-0.056	1330	0.003
0.067	-0.0056	-0.22	0	-1340	-0.001	1330	0.004
0.083...	-0.0056	-0.17	0	-1000	0.0	1000	0.004

Table 3.5: Constraint parameter values for elastic-plastic constraint

tion. The feature is trigged based on the equation below

$$\sqrt{k/m_{min}}dt > 0.25 \tag{3.2}$$

where k is elastic spring stiffness between bodies, m_{min} is smaller of connected masses or inertias and dt is integration timestep.

If this feature is active for elastic-plastic constraint 2 constraint it does not report positional error but it allows full plastic capacity to be used for correcting velocity based error instead of force provided by spring. In this scenario object behaves as depicted in table 3.6 i.e. it does not move at all.

Time	**Location**	*velError*	*posError*	*rhs*	**Velocity**	**Impulse**	**Plastic strain**
0.017...	0	-0.17	0	-1000	0	1000	0

Table 3.6: Constraint parameter values for elastic-plastic constraint 2

In following sections, results for few selected scenarios are presented. Simulation of Charpy impact test shows that plausible results can be obtained even for this quite difficult scenario.

4 Conclusion

This study has presented an approach to account plastic deformation in velocity based formulation. In the introduced method, the plastic deformation takes place if force or moment exceeds given limit, deformation absorbs energy and joint breaks if plastic capacity is exceeded. Approach presented in this work can be used in gaming industry to provide more realistic simulations without significant extra work. For gaming purposes presented method works best in scenarios where connected parts are relatively heavy. This allows normal integration timestep to be used without stability issues. This kind of metodology also opens large area of combining old structural analysis methods to modern simulation frameworks. As already noted in Terzopoulos and Fleischer (1988), the modeling of inelastic deformation remains open for further exploration in the context of computer graphics.

References

Bathe, K.J., Ramm, E., and Wilson, E.L. (1975). Finite element formulations for large deformation dynamic analysis. *International Journal for Numerical Methods in Engineering*, 9(2), pp. 353–386. ISSN 1097-0207, url: goo.gl/G12ZNa.

Catto, E. (2007-2015). *Box2D A 2D Physics Engine for Games*. url: box2d.org.

Coumans, E. (2003-2016). *Bullet Physics Library*. url: bulletphysics.org.

Dowling, N.E. (2007). *Mechanical Behavior of Materials - Engineering Methods for Deformation, Fracture, and Fatigue, third edition*. Prentice-Hall, New Jersey. ISBN 0-13-186312-6.

Erleben, K. (2005). *Stable, Robust, and Versatile Multibody Dynamics Animation*. Ph.D. thesis. University of Copenhagen.

Müller, M., Heidelberger, B., Teschner, M., and Gross, M. (2005). Meshless deformations based on shape matching. In: *ACM Transactions on Graphics (TOG)*, vol. 24, 3, pp. 471–478. url: goo.gl/RlNckJ.

Müller, M., et al. (2004). Point based animation of elastic, plastic and melting objects. In: *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 141–151. url: goo.gl/uMEgQF.

Nikula, S. (2014-2016). *Plasticity extension to Bullet Physics Library*. url: goo.gl/AxRABG.

Smith, R. (2001-2007). *Open Dynamics Engine*. url: ode.org.

Smith, R. (2002). *How to make new joints in ODE*. url: ode.org/joints.pdf.

Terzopoulos, D. and Fleischer, K. (1988). Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In: *ACM Siggraph Computer Graphics*, vol. 22, 4, pp. 269–278. url: goo.gl/0ihJzJ.