Elastic-plastic Constraint for Rigid Body Simulation

SIMO NIKULA and AKI MIKKOLA and TIMO BJÖRK
Lappeenranta University of Technology

We introduce simple and efficient method to simulate ductile fracture in existing physics engines. Method is based on common technique of splitting bodies to multiple pieces and joining them with constraints. Constraint behaviour is modelled using body dimensions and material parameters. Sample program with full source code is made available to allow developers already using Bullet Physics to add plasticity into their simulations with minimal effort.

Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Animation*; I.6.3 [**Computer Graphics**]: Simulation and Modelling—*Applications*

Additional Key Words and Phrases: Modeling, Animation, Deformation, Elasticity, Plasticity, Dynamics, Simulation

ACM Reference Format:

Nikula, S., Mikkola, A., Björk, T., G. V. G. 201x. Elastic-plastic Constraint for Rigid Body Simulation. ACM Trans. Graph. VV, NN, Article AN (Mon 201x), nb pages.
DOI = 10.1145/xx.yy
http://doi.acm.org/10.1145/XXXXXXX.YYYYYYY

1. INTRODUCTION

Computational techniques used in modern computer games are able to describe complex dynamic systems such as those found in military and other motorized vehicles with a high level of accuracy while still being able to solve the equations of motion in real time. Film production and the computer game genre of war games are two areas that have greatly benefited from rapidly improving simulation technology. In practice, software for such games is often written using open-source physics engine platforms like ODE - [Smith 2007], Bullet Physics - [Coumans 2016] and Box2D - [Catto 2015].

Computational methods used in physics engines are divided into modules that handle collision detection and contact description and modules that handle solution of equations of motion in real time. The equations of motion that need to be solved can be subdivided into equations related to motion, constraints and collisions. Velocity-based formulation is typically used in constraint based rigid body simulation, [Erleben 2005]. 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 and destruction of various bodies often takes place based on collision or impulse exceeding predefined limit. Nevertheless, breaking of steel or reinforced concrete structures using this approach is not appropriate if the simulation is to look realistic. Theory for handling of plasticity has been presented already in [Terzopoulos and Fleischer 1988]. [Jones et al. 2016] provides extensive listing of related work during past decades. [Müller et al. 2005] present a method for modeling and animating of elastic and plastic bodies in real time using point based animation. This approach has not been widely used in computer games. One major issue is collision handling of deformable bodies. Recently there has been multiple studies which have same target of bringing plasticity into wider usage, [Jones et al. 2016; Patkar et al. 2014; Budsberg et al. 2014]. Suggested methods seem best suited for film production and possible also for high end games. Methodology presented in this study does not require significant software development efforts from game vendors and is thus easily adoptable.

This study will introduce an approach to account for plastic deformation in game applications. In the introduced method, plastic deformation takes place if the force or moment exceeds a predefined limit, deformation absorbs energy and joint breaks if plastic capacity is exceeded. The approach is based on using joint motors to model plasticity. The study extends a method introduced by [Erleben 2005] which was originally proposed for modelling friction in joints. In the introduced method adjacent bodies are connected by motors. Motor power production limits are estimated based on the plastic section modulus. Joint breaking is based on summing plastic deformation and comparing it to a predefined material based limit. The elastic part of deformation is modelled by employing a spring based on modification of an existing constraint in Bullet Physics.

The approach presented in this work can be used in the gaming industry to provide more realistic simulations without significant extra work. For gaming purposes, the presented method works best in scenarios where the connected parts are heavy. This allows a normal integration timestep to be used without stability issues. This methodological approach enbles established structural analysis methods to be combined with modern simulation frameworks.

2. DESCRIPTION OF PLASTICITY IN THE FRAMEWORK OF PHYSICS ENGINES

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

Velocity-based formulation of constraint based rigid body simulation is commonly used by physics based game developers and film production teams. [Erleben 2005] provides reasoning and theoretical details for the popularity of velocity-based formulation

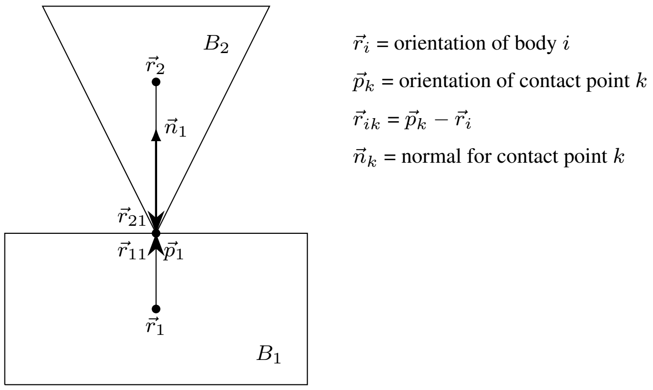

B_2
$\vec{r}_2$
$\vec{n}_1$
$\vec{r}_{21}$
$\vec{r}_{11}$
$\vec{p}_1$
$\vec{r}_1$
B_1
$\vec{r}_i$ = orientation of body i
$\vec{p}_k$ = orientation of contact point k
$\vec{r}_{ik} = \vec{p}_k - \vec{r}_i$
$\vec{n}_k$ = normal for contact point k

Fig. 1. Illustration of nomenclature for equations of motion for collision.

in constraint-based rigid body simulation instead of accelaration based. The main reason is that collision handling can be done without the use of additional procedures.

Work presented by [Erleben 2005] provides the basis for the velocity-based formulation discussed in this work. In the following section, these formulations will be clarified by a simple example using Bullet Physics implementation.

Impulse $\vec{J}$ in the time interval Δt can be written as:

$$\vec{J} = \int_0^{\Delta t} \vec{f}_{true}(t)dt, \tag{1}$$

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

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

$$\int_0^{\Delta t} m\frac{d\vec{v}}{dt}dt = \int_0^{\Delta t} \vec{f}_{true}(t) \tag{2}$$

$$m(\vec{v}^{\Delta t} - \vec{v}^0) = \vec{J}, \tag{3}$$

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 for locations and for velocities respectively as follows:

$$\vec{s}^{t+\Delta t} = \vec{s}^t + \Delta t S \vec{u}^{t+\Delta t} \tag{4}$$

$$\vec{u}^{t+\Delta t} = \vec{u}^t + \Delta t M^{-1}(CN\vec{f}^{t+\Delta t} + \vec{f}_{ext}). \tag{5}$$

The symbols used in Equations 4 and 5 are summarized in Table I. Figure 1 describes the collision of two bodies, B_1 and B_2 where $\vec{r}_i$ is orientation of body i, $\vec{p}_k$ is orientation of contact point k, $\vec{r}_{ik}$ is a vector between the center of gravity of body i and contact point k, and $\vec{n}_k$ is the contact normal for contact point k.

Friction in contacts and joint constraints can be handled in a unified way by refactoring equation 5 as, [Erleben 2005]

$$\vec{u}^{t+\Delta t} = \vec{u}^t + \Delta t M^{-1}(J^T_{contact}\vec{\lambda}_{contact} + J^T_{joint}\vec{\lambda}_{joint} + \vec{f}_{ext}), \tag{6}$$

where Jacobian terms J^T_{joint} for joints are derived by taking time derivatives of the kinematic constraints. Symbols used in Equation 6 are summarized in Table II and Figure 2, where $\vec{r}^i_{anc}$ is used to define at which point joint constraint is applied.

Table I. Nomenclature for equations of motion

Symbol	Description
$\vec{r}_i$	position of center of mass for body i
$\vec{q}_i$	orientation for body i as quaternion $[s_i, x_i, y_i, z_i]^T$
$\vec{p}_h$	contact or joint point k
$\vec{r}_{ki}$	$\vec{p}_k - \vec{r}_i$
$\vec{s}$	$[\vec{r}_1, \vec{q}_1, ..., \vec{r}_n, \vec{q}_n]^T$
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 $N \in \mathbb{R}^{3K \times K}$

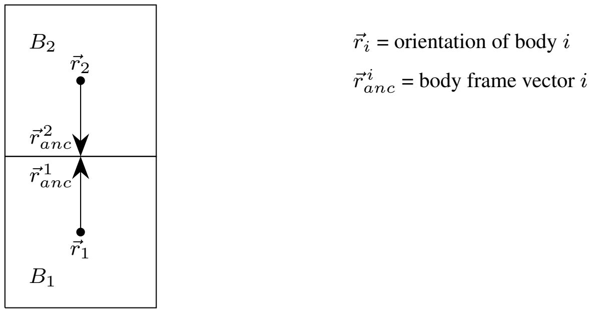

B_2
$\vec{r}_2$
$\vec{r}^2_{anc}$
$\vec{r}^1_{anc}$
$\vec{r}_1$
B_1
$\vec{r}_i$ = orientation of body i
$\vec{r}^i_{anc}$ = body frame vector i

Fig. 2. Illustration of nomenclature for equations of motion for joint.

Table II. Additional terms for generalized equations of motion

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

Constraint processing in Bullet Physics is based on ODE, [Smith 2007]. Joints are also discussed in detail in [Erleben 2005]. Equations 7, 8 and 9 are created for each constraint. Derivation for terms in Equation 7 can be done using the position and orientation of connected bodies e.g. for ball joint formulation is based on both joint points having the same position. In contact cases, formulation is easier if it is done using velocities, [Smith 2002].

$$J_1\vec{v}_1 + \Omega_1\vec{\omega}_1 + J_2\vec{v}_2 + \Omega_2\vec{\omega}_2 = \vec{c} + C\vec{\lambda} \tag{7}$$

Table III. Constraint parameters

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

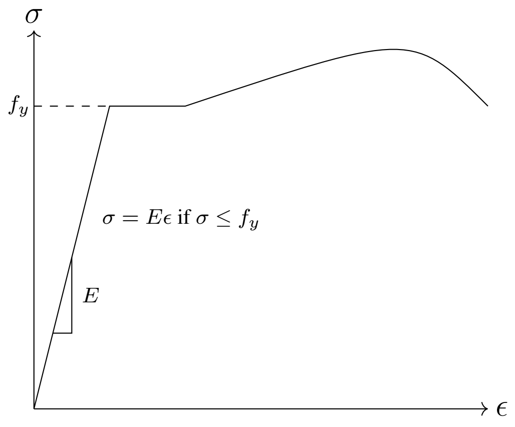


Fig. 3. Engineering stress-strain curve of ductile steel (not to scale).

$$\vec{\lambda} \geq \vec{l} \tag{8}$$

$$\vec{\lambda} \leq \vec{h} \tag{9}$$

In the following section, these equations will be explained by a simple example. The main parameters and corresponding fields in Bullet Physics are given in Table III.

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, a static solution with an assumption of linear strain-displacement relation using displacement based boundary conditions is used. [Bathe et al. 1975] provides a 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 for in simulations by using a suitable coefficient of restitution. This provides a reasonable means to simulate loss of energy in collisions. In this work simulation of breaking of bodies made of ductile material is made more realistic by splitting the rigid body to multiple bodies that are connected by energy absorbing joints. A typical engineering stress-strain curve of ductile steel is shown in Figure 3.

In Figure 3, 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]. The stress-strain curve is not drawn to scale as elastic strain could not be seen as it is typically 0.001 to 0.005 and fracture strain can be 100 times larger.

In this work, an 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 introduced in this work if the deformation is related to ahigher frequency than integration stability would allow. It should be noted that geometry of bodies is not updated during analysis and thus engineering stress-strain properties are used.

In this work, strain hardening is taken into account by assuming that plastic volume in bending expands, [Dowling 2007]. Material that starts to yield first is hardened and as a result of which yielding moves.

The difference between the elastic and plastic section modulus is depicted in Figure 4.

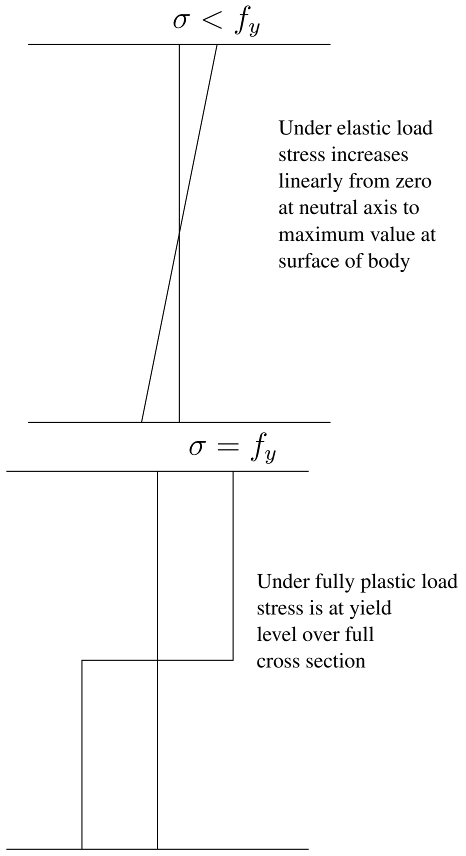

$\sigma < f_y$
Under elastic load stress increases linearly from zero at neutral axis to maximum value at surface of body
$\sigma = f_y$
Under fully plastic load stress is at yield level over full cross section

Fig. 4. Axial stress distribution over a cross section for bending under elastic and fully plastic loads.

As shown in Figure 2.4, if stress is below yield limit f_y, stress and strain are linear within the material. If cross section is fully plastic, stress is assumed to be at yield level over the whole cross section such that the plastic section modulus is higher than the elastic section modulus.

In this work, plasticity is handled by defining maximum forces in Equations 8 and 9 using plastic capacities, which are defined below.

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

Table IV. Maximum forces and moments for rectangular section with width b and height h using constant yield stress f_y

Direction	Maximum value
maximum shear force	$0.5\, b\, h f_y$
maximum normal force	$b\, h\, f_y$
maximum bending moment in direction of h	$0.25\, b\, h^2\, f_y$
maximum bending moment in direction of b	$0.25\, b^2\, h\, f_y$
maximum torque	$\approx 0.19\, b\, h\, \frac{b+h}{2} f_y$

```
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
```

Fig. 5. Calculation of maximum moment around $\vec{r}^{\,i}_{anc}$ using Octave.

$$N_{max} = \int_A f_y. \tag{10}$$

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

$$Q_{max} = \int_A \tau_y. \tag{11}$$

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

$$M^x_{max} = \int_A z f_y, \tag{12}$$

$$M^z_{max} = \int_A x f_y. \tag{13}$$

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

$$M^y_{max} = \int_A d\tau_y. \tag{14}$$

Maximum forces and moments for a rectangular section with width b and height h using constant yield stress are given in Table IV. Yield shear stress is assumed to be $0.5\, f_y$ using the Tresca yield critetion. If the von Mises yield criterion is used 0.5 is replaced by 0.58 ($1/\sqrt{3}$), [Dowling 2007]. These are not exact values in a multiaxial stress state but they should be acceptable in most gaming scenarios.

For torque there is a closed form solution only for circular cross sections. Given approximation is best suited for cases where b and h are similar. Better approximation for any given b and h can be obtained by integrating distance from the center of the joint over cross section and multiplying it with the yield shear stress e.g. using Octave, [Eaton 2016]. An example of calculation of the maximum moment around $\vec{r}^{\,i}_{anc}$ is shown in Figure 5.

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

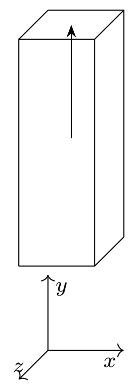


Fig. 6. Single body model for plasticity processing.

Further enhancements were created and tested by forking Bullet Physics source code and adding new constraints, [Nikula 2016b]. Instructions for using windows executable and source code are available, [Nikula 2016a].

3. NUMERICAL EXAMPLES

3.1 Block under tension

In this section, changes to the constraint formulation are exemplified with some simulation steps of non constrained, rigid and elastic-plastic examples using numerical values.

In the first example, the system under investigation has only one dynamic rigid body which is a three meters high block of 0.04 % steel enforced concrete. The cross section is one square meter. A constraint is set so that the connecting frame is at the top of the block. In this case, the frame could be anywhere but if multiple bodies are involved, the connecting frame must be defined so that it reflects the 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 the elastic-plastic case. The yield stress of steel is 200 MPa. Gravity is 10 $\frac{m}{s^2}$ to keep numbers simple. Simulation step is $\frac{1}{60}$ s and 10 iterations are done for a single step. The body is 1.5 meters above rigid ground. In the unconstrained case the body will hit hit the ground at t = 0.548 s ($\sqrt{2 \cdot 1.5/10}$).

A single simulation step is performed using the following substeps. In this work, changes are made only to the constraint setup and the update actions.

(1) Apply gravity to each moveable 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 6 is added by $\{0, -60000, 0\}$.

(2) Predict unconstrained motion for each body controlled by the 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 bodies. Each rigid body is represented by sphere geometry and contact prediction is done if the body moves more than a given threshold value during the simulation step. Continuous collision detection is configured for each body. A 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 the end of the simulation step. In the unconstrained case, contact is detected after t=0.55 s if continuous collision detection is not used and manifold distance gets a 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 the 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 teh setup phase constraints are queried for positional errors for calculation of c in Equation 7 and maximum and minimum impulses in Equations 8 and 9. In the finish phase constraint forces are calculated if requested and velocities of bodies are updated.
(7) Integrate transforms using velocities modified in the previous step.
(8) Update actions (callbacks) are called. In the elastic-plastic case, plasticity is summed and the equilibrium point of the elastic part is updated if the maximum force or moment is exceeded.
(9) Activation state of bodies is updated. To avoid extra calculation bodies are as default put into sleeping state if linear and angular velocities of the body are less than given threshold values (default 0.8 and 1.0) or longer than the set time limit (2.0).

Equation 7 is simplified to 15 in constrained cases.

No rotation takes place. ω_1 and ω_2 are zero.

Constraint force mixing can be ignored.

Only vertical velocity is handled.

The other involved body is rigid and it does not move.

$$mv_y = c \tag{15}$$

Equations 8 and 9 are not active for fixed cases as there are no upper or lower limit for forces. For the elastic-plastic case maximum impulse is set to the product of the yield stress, area of steel enforcement and time step (1330).

Method btSequentialImpulseConstraintSolver::solveGroupCacheFriendlySetup in Bullet Physics is used to get values for the internal variables described in Figure 7.

Actual values for an unconstrained case without continuous collision detection are shown in Table V and location, velocity, posError and velError are depicted in Figure 8. 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 a few steps the location and position stabilize although internally calculation is needed for each time step until a body is deactivated.

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

velError is calculated using velocities and external impulses of the connected bodies.
In constraint cases, the main contibutor is the relative speed of the bodies at the joint point.
In contact cases, the main contributor is the relative speed of the bodies at the 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 provided by the constraint. It is a significant factor in designing stable constraints.
In fixed cases, the value is about 12 times the actual position error. Factor 12 is based on the time step (60) and the default value of error reduction parameter (erp), which has a value of 0.2 in this context.
In elastic plastic cases, the value is set to zero if the impulse would be larger the than maximum impulse or the spring simulation cannot be done in a stable way.
In contact cases, the value is zero if there is no penetration. For penetration cases it is $\frac{-penetration\, erp}{timeStep}$. In contact cases the 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 the mass of the body (6000).
In contact cases, it varies below the mass of the body.

Impulse is impulse applied to the body during a timestep.
In constraint cases, it is obtained from the btJointFeedback structure.
In contact cases, it is obtained by summing the applied impulses from the active manifolds.

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

Fig. 7. Internal variables used in Bullet Physics in constraint solving.

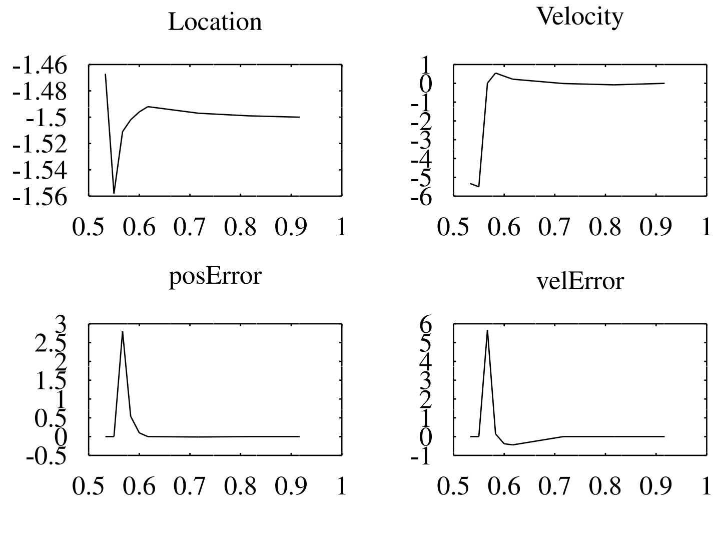


Fig. 8. Graph for location, velocity, positional error and velocity error for the unconstrained case without continuous collision detection.

Table V. Simulation values for the unconstrained case without continuous collision detection. Typical values are shown for internal contact values as the number of contacts and the values at contact points differ.

Time	**Location**	*velError*	*penetration*	*posError*	*rhs*	**Velocity**	**Impulse**
0.017	-0.003					-0.17	0
0.533	-1.467					-5.33	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 VI. Simulation values for the unconstrained case with continuous collision detection.

Time	**Location**	*velError*	*pene-tration*	*rhs*	**Velocity**	**Impulse**
0.017	-0.003				-0.17	0
0.550	-1.500	3.5	0.033	21000	-2	21000
0.567	-1.500	2.17	0	8100	0.01	13000
0.583	-1.500	0.15	0	600	0	937
0.600	-1.500	0.17	0	600	0	1040

Table VII. Constraint parameter values for the fixed constraint

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

Values for a fixed constraint are shown in Table VII. The constraint is activated in the second step and positional error is corrected about 20 % in each step as requested by using an erp value of 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.

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

Values for an elastic-plastic case are shown in Table IX. The body drops freely during first simulation step and gains enough kinetic energy so that higher impulses are needed in the following steps. This causes plastic strain during the next three steps. Positional error is reported as zero at start of simulation because force exceeds maximum force. At later phases positional error is not reported to avoid instability.

Six-dof-spring constraint 2 has an optional feature to avoid unstability by automatically softening the constraint spring. The feature is activated if the current timeStep is overly large for the spring-mass system simulation. The feature is triggered based on the equation below

$$\sqrt{k/m_{min}}dt > 0.25, \tag{16}$$

Table VIII. Selected constraint parameters for elastic-plastic constraint 2

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 IX. Constraint parameter values for elastic-plastic constraint

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

Table X. Constraint parameter values for elastic-plastic constraint 2

Location	*velError*	*rhs*	**Impulse**	**Plastic strain**
0	-0.17	-1000	1000	0

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

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

3.2 Charpy impact test

In a second numerical example, simulation of a Charpy impact test is used to benchmark the approach in a more complicated scenario.

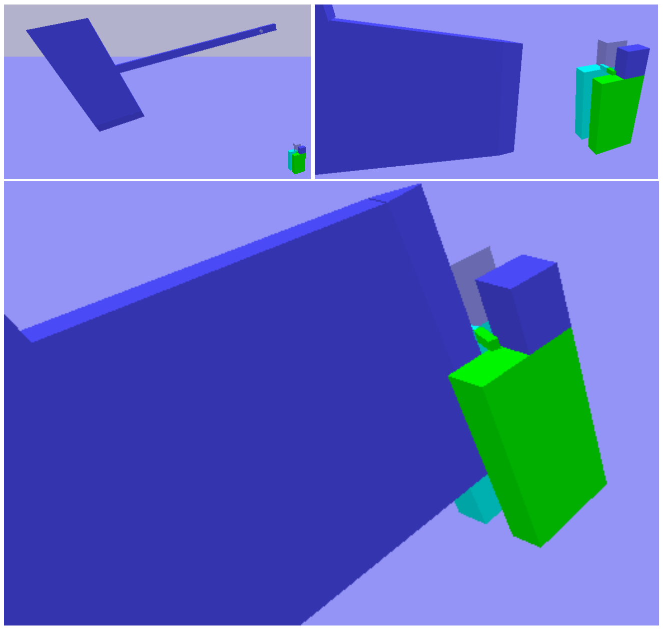


Fig. 9. Detailed pictures from simulation of a Charpy impact test

Table XI. Energy loss for few timesteps

Timestep[ms]	Energy loss [J]	Notes
0.2	170	specimen penetrates hammer
0.1	120	
0.05	150	

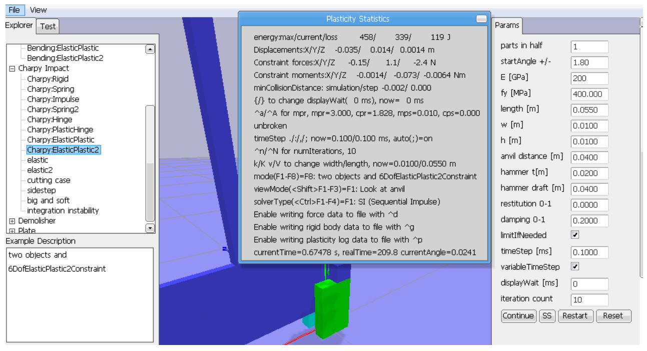
The material is steel and density is 7800 $\frac{kg}{m^3}$. Young's modulus is 200 GPa. Specimen dimensions are 10x10x55 mm with a 2 mm notch in the middle which is taken into account in the calculations. Support anvils initially have 40 mm open space between them and their width is 40 mm. The hammer is 0.5 m wide and 0.25 m high, thickness is 0.02 m and mass is 19.5 kg. The hammer has 40 mm draft. If the specimen bends about 1.9 radians (108 degrees) it can go between the anvils. The expected energy loss is the product of the plastic moment of section, hinge angle needed for the specimen to go through the supports and the yield stress of the specimen. For hinge angle of 1.9 radians and yield stress of 400 MPa, expected energy loss is 122 J.

Bullet Physics simulations are usually done using a fixed time step of 1/60 s, i.e., 16.67 ms. For this case that is too large. The default time step was selected to be 5 ms outside impact and 0.1 ms during impact. An automatic time stepping routine changes the timestep so that at angles higher than 0.2 radians 5 ms time step is used and adjusts it linearly to a selected time step between the angles of 0.2 and 0.05 radians.

Figure 9 shows three detailed pictures from the simulation. The first picture shows early phase of simulation. In the second picture the hammer has not yet collided with the specimen. In the last picture the specimen is bending. After few more time steps specimen goes between anvils in one peace. If maximum rotation is set lower than 1.9 radians, joint is deactivated.

Table XI shows energy loss for a few timesteps to demonstrate the sensitivity of the solution using elastic-plastic constraint 2.

Figure 10 shows an overview from simulation as the specimen passes between the supports.



Example Description

two objects and
6DofElasticPlastic2Constraint

```
energy:max/current/loss        458/       339/       119 J
Displacements:X/Y/Z    -0.035/    0.014/   0.0014 m
Constraint forces:X/Y/Z     -0.15/      1.1/     -2.4 N
Constraint moments:X/Y/Z    -0.0014/   -0.073/  -0.0064 Nm
minCollisionDistance: simulation/step  -0.002/ 0.000
{/} to change displayWait(  0 ms), now=   0 ms
^a/^A for mpr, mpr=3.000, cpr=1.828, mps=0.010, cps=0.000
unbroken
timeStep ./:/,/; now=0.100/0.100 ms, auto(;)=on
^n/^N for numIterations, 10
k/K v/V to change width/length, now=0.0100/0.0550 m
mode(F1-F8)=F8: two objects and 6DofElasticPlastic2Constraint
viewMode(<Shift>F1-F3)=F1: Look at anvil
solverType(<Ctrl>F1-F4)=F1: SI (Sequential Impulse)
Enable writing force data to file with ^d
Enable writing rigid body data to file with ^g
Enable writing plasticity log data to file with ^p
currentTime=0.67478 s, realTime=209.8 currentAngle=0.0241
```

parts in half	1
startAngle +/-	1.80
E [GPa]	200
fy [MPa]	400.000
length [m]	0.0550
w [m]	0.0100
h [m]	0.0100
anvil distance [m]	0.0400
hammer t[m]	0.0200
hammer draft [m]	0.0400
restitution 0-1	0.0000
damping 0-1	0.2000
limitIfNeeded	✓
timeStep [ms]	0.1000
variableTimeStep	✓
displayWait [ms]	0
iteration count	10

Fig. 10. Overview of simulation of Charpy impact test

3.3 Demolisher

The third numerical example introduces the demolisher. The demolisher is a collection of possible scenarios that can be added to games to provide ductile joints between rigid bodies. Rigid concrete bodies are connected by steel enforcement. Concrete density is 2000 $\frac{kg}{m^3}$ and steel density is 7800 $\frac{kg}{m^3}$. Steel is assumed to handle load. 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 a single step. Multiple simulation steps are done for each render if needed.

The simulation can be done in real time on modest hardware, e.g. on laptop having 2.0 GHz Intel i3 CPU with integrated graphics. Figure 11 shows three screenshots from the simulation.

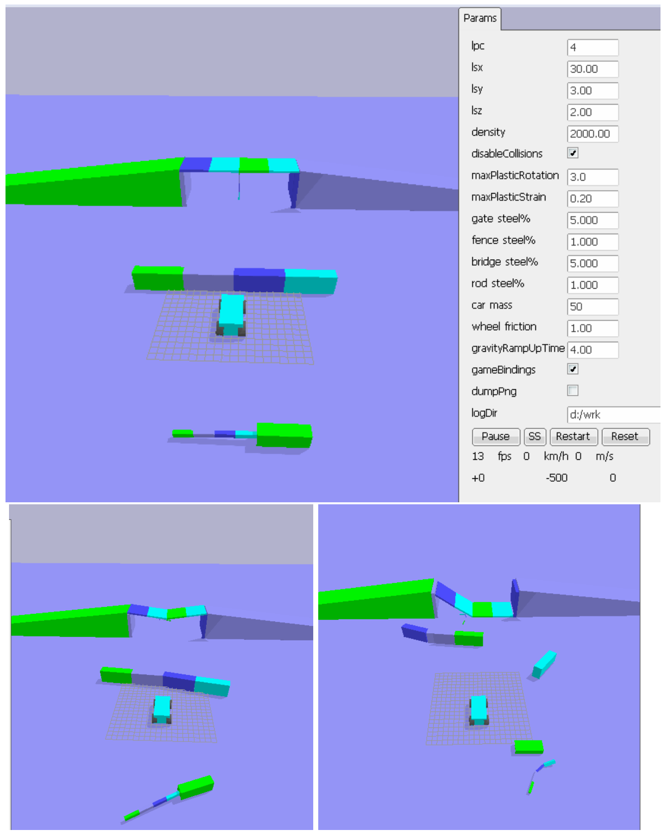
The vehicle is modelled as a single box btRaycastVehicle with four driving wheels. Wheels are rendered for visual feedback. In the left picture, the initial state is shown. The gate in front is modelled to be so slender that it has visible elastic deflection. The gate support is modelled as a heavy box. The bridge has rigid ramps and supports at both ends. Gravity is applied gradually to avoid collapse of bridge at the start of the simulation. In the middle picture, the bridge is shown in a state where a car has driven once over the bridge without stopping and other bodies have taken light damage. In the right picture, two joints have been broken and the bridge has fallen from teh support due to bending.

4. CONCLUSION

This study presents an approach to account for plastic deformation in a velocity based formulation. In the introduced method, the plastic deformation takes place if the force or moment exceeds the given limit, the deformation absorbs energy and the joint breaks if plastic capacity is exceeded. The approach presented in this work can be used in the gaming industry to provide realistic simulations. For gaming purposes the presented method works best in scenarios where the connected parts are heavy. This allows a normal integration timestep to be used without stability issues.

This kind of methodology also has considerable potential for the combination of the structural analysis methods with real-time simulation frameworks. Handling of significant axial and shear forces and moments at the same time is one possible area of further studies.

There is also work to be done in the area of performance optimization. One possible area is the reuse of once calculated values if memory capacity is not limited. This could done, e.g., when dealing with high frequency modes. Another area is usage of the graphics processing unit (GPU) for calculation. Bullet Physics already has an experimental GPU pipeline but most constraint types



lpc	4
lsx	30.00
lsy	3.00
lsz	2.00
density	2000.00
disableCollisions	✓
maxPlasticRotation	3.0
maxPlasticStrain	0.20
gate steel%	5.000
fence steel%	1.000
bridge steel%	5.000
rod steel%	1.000
car mass	50
wheel friction	1.00
gravityRampUpTime	4.00
gameBindings	✓
dumpPng	☐
logDir	d:/wrk

```
13    fps   0    km/h  0    m/s
+0          -500        0
```

Fig. 11. Simulation of Demolisher scenario

are not yet supported. GPU support was one reason for selecting Bullet Physics for use in this study

Integration to 3D modelling and animation software products should also be solved before we can expect to see realistic plasticity in main stream games. Bullet Physics is already integrated into several ones like Blender and it was another significant reason for selecting Bullet Physics to be used in this study.

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, 353–386.

BUDSBERG, J., ZAFAR, N. B., AND ALDÉN, M. 2014. Elastic and plastic deformations with rigid body dynamics. In *ACM SIGGRAPH 2014 Talks*. SIGGRAPH '14. ACM, New York, NY, USA, 52:1–52:1.

CATTO, E. 2007-2015. Box2d a 2d physics engine for games.

COUMANS, E. 2016. Bullet physics library.

DOWLING, N. E. 2007. Mechanical behavior of materials - engineering methods for deformation, fracture, and fatigue, third edition.

EATON, J. W. 2016. Gnu octave.

ERLEBEN, K. 2005. Stable, robust, and versatile multibody dynamics animation. Ph.D. thesis, University of Copenhagen.

JONES, B., THUEREY, N., SHINAR, T., AND BARGTEIL, A. W. 2016. Example-based plastic deformation of rigid bodies. *ACM Trans. Graph. 35,* 4 (July).

MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2005. Meshless deformations based on shape matching. *ACM Transactions on Graphics (TOG) 24,* 3, 471–478.

NIKULA, S. 2016a. Bullet plasticity, working notes.

NIKULA, S. 2016b. Plasticity extension to bullet physics library.

PATKAR, S., AANJANEYA, M., BARTLE, A., LEE, M., AND FEDKIW, R. 2014. Efficient denting and bending of rigid bodies. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '14. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 87–96.

SMITH, R. 2002. How to make new joints in ode.

SMITH, R. 2007. Open dynamics engine.

TERZOPOULOS, D. AND FLEISCHER, K. 1988. Modeling inelastic deformation: Viscolelasticity, plasticity, fracture. *ACM Siggraph Computer Graphics 22,* 4, 269–278.