Energy Embedded Gauss-Seidel Iteration for Soft Body Simulations

Ozan Cetinaslan
Instituto de Telecomunicações &
Faculdade de Ciências,
Universidade do Porto, Portugal
Email: ozan.cetinaslan@gmail.com

Rafael Oliveira Chaves
Federal University of Pará &
State University of Pará, Brazil
Email: rafael.ufpa@gmail.com

***Abstract*—In this paper, we present our novel energy embedded Gauss-Seidel iteration to simulate soft objects. Our algorithm is inspired by the equality of the kinetic and potential energy changes, and employs the extended Position-Based Dynamics algorithm (XPBD) as the base algorithm. The proposed method does not aim to conserve the total energy of the system, only alters the position constraints based on the kinetic and potential energy balances within the Gauss-Seidel process of the XPBD algorithm. Our algorithm provides an implicit solution for relatively better visual results during the simulation of soft bodies. Since we apply our solution within the Gauss-Seidel iteration, it is not dependent to both simulation step-size and integration methods. We demonstrate the benefits of our method with many position constraints such as geometric deformation constraints, strain based constraints and continuous materials.**

I. Introduction

In recent years, Position-Based Dynamics (PBD) [1] has been a widely used method in games, movies and medical applications. This can be explained with its simplicity, stability and performance. Traditional techniques formulate the physical phenomena by using forces and velocities based on the mass-spring systems. On the other hand, PBD method is based on the projected constraints which provide a large spectrum to simulate dynamic behaviors from simple stretch to nonlinear continuous materials.

PBD generalizes the Verlet integration approach of [2] with velocity updates. The core of the algorithm is a constraint projection schema for updating the vertex positions by employing a non-linear Gauss-Seidel type iteration. Furthermore, a recent extension to PBD, which is called XPBD [3], improves the original algorithm by applying a total Lagrange multiplier for general material stiffness. Unfortunately, XPBD based simulation of deformable objects suffers from extreme soft material behavior and unintended mesh degeneration while the Gauss-Seidel iterations are too few. This is more obvious during the simulation of continuous materials. Since the stiffness of the materials is dependent on the number of iterations and the compliance matrix entries, altering those values is not practical during the interactive simulations.

The proposed algorithm is inspired by the idea of conserved energy differences from [4], and utilizes it within the Gauss-Seidel iterations by assuming the projected constraints as the potential energy functions of the materials. Furthermore, the velocities are kept in their implicit state similar to [2]. Therefore the position updates do not suffer from the possible extreme values of velocities due to the step-size division. During the interactive simulations, XPBD may produce non-smooth mesh degeneration either under constant force (such as gravity) or direct user interaction. In our algorithm, we address this unintended effect, and provide relatively smoother results during the simulations.

The proposed algorithm is inspired from the total energy conservation principle. However, the simulation scheme dissipates the overall energy of the system. This is due to the fact that PBD/XPBD algorithms employ the Verlet integration scheme with the known energy dissipation properties. Our algorithm works in the same direction as XPBD, nevertheless Gauss-Seidel loop takes advantage of the overall energy of the system. In order to test our method, we have implemented many different constraint types such as stretching, shearing, bending, area and volume preservations, strain based dynamics constraints [5] and many continuous material constraints [6]. Since our algorithm is based on XPBD method, they share the common features. However, our version of Gauss-Seidel algorithm have more computation steps than XPBD; therefore, a slight overhead can be observed. We keep the proposed algorithm as generic as possible, it is straightforward, and it can be easily adapted to the existing position-based frameworks.

II. Related Work

Foundations of Position-Based Dynamics can be seen in [7] for rigid body animation and in [8] for deformable objects. After, Jakobsen [2] described the Verlet integration scheme with direct position manipulation by using projected constraints for rigid and deformable body dynamics. After Müller et. al. [1] generalized the idea with corrected velocities, and introduced the formal PBD algorithm. Goldenthal et al. [9] also used constraints and offered a fast projection algorithm. Besides, in a different framework, Stam [10] introduced Nucleus which is a unified constrained based solver for Autodesk Maya. More recently, Tournier et al. [11] addressed the instabilities by applying second order derivatives on the constraints. Recently, based on [12], Macklin et al. [3] improved the PBD algorithm with a total Lagrange multiplier to address the iteration count and simulation step-size dependencies. In this paper, we use

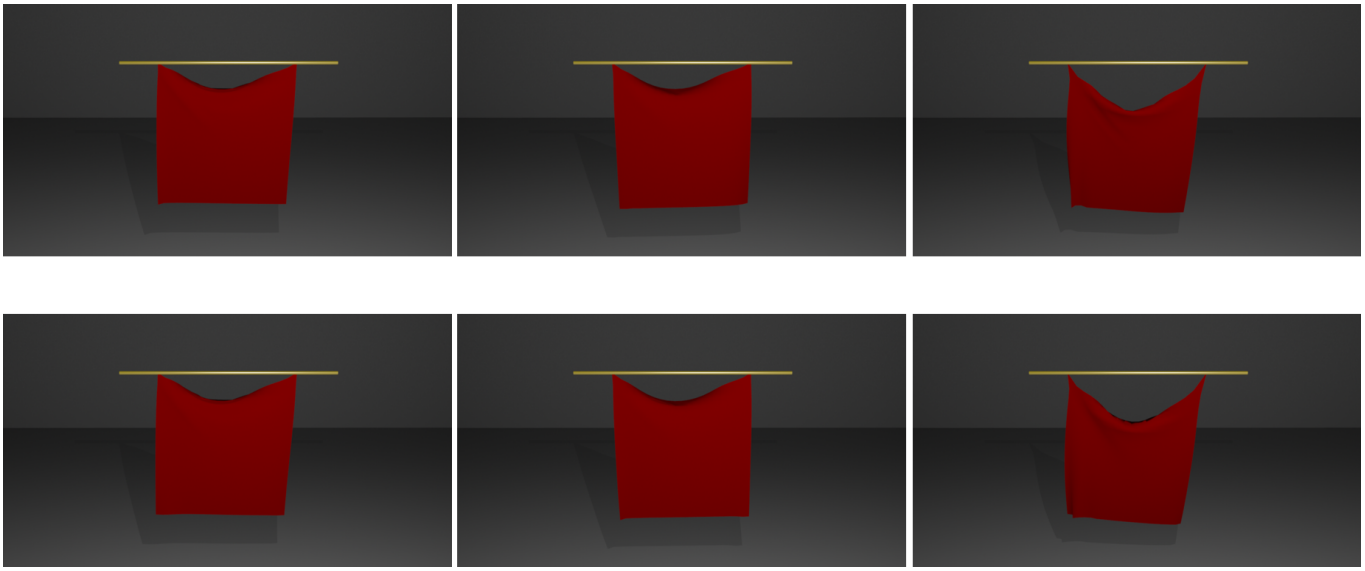

Fig. 1. A piece of cloth hangs under constant gravitational force. Left column is stretch and shear constraints attached together. Middle column is 2D Green strain tensor constraint. Right column is the 2D anisotropic continuous material constraint. All simulations are executed under the same conditions, such as $\alpha = 0.0001$, iteration count $= 20$, damping coef $= 0.3$, step-size $= 1/24$. Top row: Our method. Bottom row: XPBD.

the same base algorithm. For more detailed information on PBD, we refer the reader to [13].

Due to simplicity and efficiency of PBD algorithm, many researchers used PBD method for solid, fluid, rigid body, cloth and skinning simulations. Müller and Chentanez introduced wrinkle meshes [14] for cloth and skin deformations, and later on they proposed oriented particles [15] for rapid simulation of various solid objects. Kim et al. [16] offered long range attachments to simulate inextensible clothing. Kelager et al. [17] presented a novel triangle bending algorithm as an alternative to dihedral bending constraint for better performance in cloth simulations. Müller et al. [18] simulate hair and fur, after Macklin and Müller [19] simulate fluids with position-based approach. Furthermore, Müller et al. [5] and Bender et al. [6] used PBD to simulate continuous material models. PBD algorithm also incorporates with facial animation frameworks. Fratarcangeli [20] offered a facial dynamics model including muscle and skinning system based on a PBD method. Cetinaslan and Orválho [21] proposed a framework for local contact deformations on facial models that has employed the constrained dynamics. The character skinning and plausible skin deformations have been handled by using PBD. Deul and Bender [22] employed constrained projections for collision handling during simulating the elastic behavior of the skin. Abu Rumman and Fratarcangeli [23] incorporated the constraint projections and linear blend skinning for the articulated characters. Elastic rod simulations have been taking the advantage of the PBD method. Umetani et al. [24] proposed a novel algorithm to simulate complex bending and twisting behavior of elastic rods. Recently, Deul et al. [25] presented a novel direct solver for stiff rod simulations with a modified constraint solver. Their technique provides a notable speed-up during the simulations.

Energy conservation (or Hamiltonian system) is studied along with the symplectic numerical integrators which is the subtopic of geometric numerical integrators [26]. Several papers such as [27], [28] studied symplectic integration systems for long-time energy conservation. Bathe [29] proposed a direct implicit time integration which conserves energy and momentum. Their algorithm has provided the large deformations in long time durations. Recent work by Dinev et al. [30] presented a hybrid integration method by blending implicit and explicit Euler integrators to obtain high stability and energy conservation properties. Similar to the Projective Dynamics method, their method provides a local/global solution. On the other hand, Su et al. [4] applied the energy conservation to the mass-spring systems independent from the integration method. They took the advantage of energy conservation approach and presented an energy budgeting method.

Although, the proposed method inspired from the similar concepts of the conserved total energy principles, we have computed the energy differences equality concept within the implicit formulation of the Gauss-Seidel algorithm without violating the original position-based dynamics method. Therefore, our algorithm does not provide the total energy conservation, but embeds the energy differences to Gauss-Seidel iteration to enhance the existing XPBD algorithm.

III. Background

This section briefly reviews the foundations of PBD/XPBD methods and we refer to the tutorial by Bender et al. [13] for further details. PBD [1] accepts the input model as a system of interconnected particles with a given set of position constraints. The algorithm works in three steps: As a first step, Verlet integration scheme predicts positions of the particles ($x_i \in \mathbb{R}^3$). In the second step, the bilateral constraint set

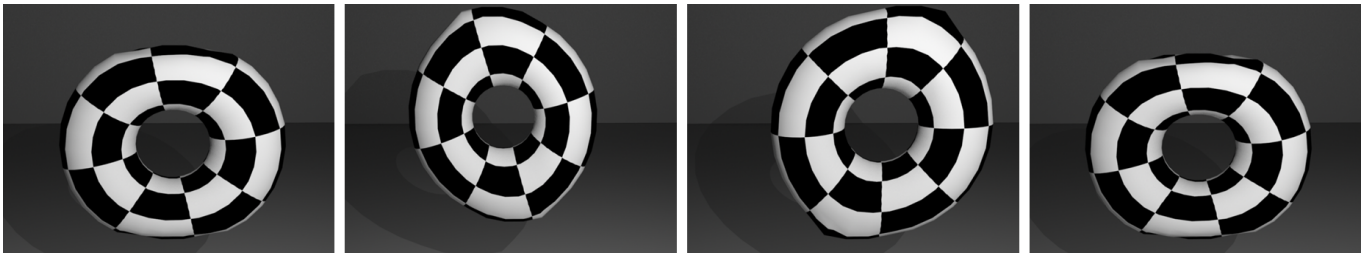

Fig. 2. Strain based torus model is stretched and compressed by using our method. Many different material models are simulated, First column: Linear elasticity constraint, Second column: 3D Green strain tensor constraint, Third column: St. Venant - Kirchhoff material constraint, Fourth column: Neo - Hookean material constraint. All simulations are executed under the same conditions, such as $\alpha = 0.0001$, iteration count $= 5$, damping coef. $= 0.3$, step-size $= 1/24$.

is solved by employing a Gauss-Seidel solver. Lastly, the velocities are updated according to the new particle positions.

The second step of PBD is the core of the algorithm and determines the position correction of each particle (Δx_i) such that bilateral behavior is conserved ($C(x + \Delta x) = 0$). The displacements of the particles are calculated with the first order Taylor approximation: $C(x + \Delta x) \approx C(x) + \nabla_x C(x) \cdot \Delta x = 0$. The directions of the Δx are limited to the directions of the $\nabla_x C(x)$ to conserve both linear and angular momentums implicitly. Thus, with the consideration of the mass (m) of each particle i, Δx is computed as in Equation (1):

$$\Delta x_i = w_i \nabla_{x_i} C(x) \lambda_i \tag{1}$$

where $w_i = \frac{1}{m}$ and λ_i stands for the Lagrange multiplier which is obtained by substituting Equation (1) in the first order Taylor approximation:

$$\lambda_i = -\frac{C(x)}{\sum_i w_i |\nabla_{x_i} C(x)|^2} \tag{2}$$

XPBD [3] improves the original algorithm with total Lagrange multiplier to address the dependency of simulation step-size and iteration count. XPBD replaces λ_i from Equation (1) with $\Delta\lambda_i$ based on the constraint derivation of [12]. Furthermore, $\Delta\lambda_i$ is extended with the compliance parameter α which is the inverse stiffness. Therefore, the new position correction is computed as in Equation (3):

$$\Delta x_i = w_i \nabla_{x_i} C(x) \Delta\lambda_i \tag{3}$$

and $\Delta\lambda_i$ is computed as in Equation (4):

$$\Delta\lambda_i = -\frac{C(x) + \tilde{\alpha}\lambda_i}{(\sum_i w_i |\nabla_{x_i} C(x)|^2) + \tilde{\alpha}} \tag{4}$$

where $\tilde{\alpha} = \alpha/h^2$, $\lambda_{i+1} = \lambda_i + \Delta\lambda_i$ that is computed in each iteration, and h represents the simulation step-size. Therefore, the λ_i value is updated incrementally for each constraint value at the instant iteration.

IV. Energy Embedded Gauss-Seidel Iteration

The proposed algorithm works under two necessary assumptions. First, we utilize the implicit velocity vectors similar to [2]. Second, we consider that the projected constraints are the potential energies of mass-spring systems or continuous material models based on [6].

According to our proposed method, the Gauss-Seidel iteration should satisfy $\Delta KE = -\Delta PE$ where KE is the kinetic energy and PE is the potential energy. When we simulate with PBD or XPBD, this condition is not satisfied due to energy dissipation. Furthermore, it is not possible to directly apply energy budgeting from [4] due to the fact that XPBD computes velocities based on the particle positions. However, it is possible to directly apply energy budgeting within the Gauss-Seidel iteration. The Gauss-Seidel algorithm updates the positions of particles as many times as the iteration count. This allows us to track the new particle positions and their corresponding velocities during each iteration. Therefore, the kinetic energy and potential energy differences can be defined as:

$$\Delta KE = \frac{1}{2} m[(V_i^{k+1})^T V_i^{k+1} - (V_i^k)^T V_i^k] \tag{5}$$

$$\Delta PE = C(x^n)^k - C(x^{n-1}) \tag{6}$$

where V_i^{k+1} and V_i^k are the velocities for each particle i at iteration $k+1$ and k respectively. That is simply $V_i^{k+1} = (x_i^n)^{k+1} - x_i^{n-1}$ and $V_i^k = (x_i^n)^k - x_i^{n-1}$ at time t^n. $C(x)^k$ is the constraint value at iteration k. It should be noted that there exists a direct relation between the velocity changes and the position corrections. In order to maintain an energy balance within the simulation loop, $\Delta KE + \Delta PE = 0$ should be satisfied while iterating the Gauss-Seidel algorithm. However, after some iterations, the system starts to lose that equality due to dissipation which causes the energy residuals (ϵ). In [4], these energy residuals are computed in the last iteration, and corrected before moving to the next simulation step. Alternative to that, we compute $\Delta KE + \Delta PE = \epsilon$ for each Gauss-Seidel iteration, and alter the constraint values with the energy residual (ϵ) in each iteration. Therefore, the constraint value becomes $C(x^n)^k = C(x^n)^k + \epsilon$. Moreover, it is possible to define the current constraint as in Equation (7):

$$C(x^n)^k = C(x^{n-1}) - \Delta KE \tag{7}$$

Either by using equation (7) or by updating $C(x)^k$ with ϵ, our approach guarantees an implicit energy balance within

Algorithm 1 Energy Embedded XPBD Simulation Loop

for `each particle i` **do**
  initialize $x_i \leftarrow x_i^0, V_i^k \leftarrow 0, V_i^{k+1} \leftarrow 0, w_i \leftarrow 1/m$
end for
loop
  Verlet Integ.: $x_i^{n+1} \leftarrow x_i^n + (x_i^n - x_i^{n-1}) + h^2 w_i f_{ext}(x_i^n)$
  initialize Total Lagrange Multiplier $\lambda_i^0 \leftarrow 0$
  while `k < iterationCount` **do**
    for `each Constraint` **do**
      compute $C(x^{n-1})$
      compute $C(x^n)$
      compute ΔPE using Eq (6)
      compute ΔKE using Eq (5)
      $\epsilon \leftarrow \Delta KE + \Delta PE$
      $C(x^n) \leftarrow C(x^n) + \epsilon$
      compute $\Delta\lambda_i$ using Eq (4)
      compute Δx_i using Eq (3)
      $V_i^k \leftarrow (x_i^n)^k - x_i^{n-1}$
      update $\lambda_i^k \leftarrow \lambda_i^k + \Delta\lambda_i$
      update $(x_i^n)^{k+1} \leftarrow (x_i^n)^k + \Delta x_i$
      $V_i^{k+1} \leftarrow (x_i^n)^{k+1} - x_i^{n-1}$
    end for
    $k \leftarrow k + 1$
  end while
  update positions $x_i^{n+1} \leftarrow (x_i^n)^k$
end loop

each iteration with a slight computation cost. Besides, ϵ value provides information about the energy residual and determines the current state of the constraint. In each iteration, it may load or relax the constraint according to its value. Algorithm 1 presents our proposed technique.

V. Constraints

XPBD provides the simulation of many position constraints. Extending the original algorithm with our method does not violate its generality. In this section we define the constraint functions which we use to simulate the deformable models and cloth.

A. Geometric Constraints

Geometric constraints are the oldest and the most employed constraint types of position-based frameworks. They are easy to implement, fast and robust (Figure 1 - first column). These constraints are mostly employed to simulate cloth. The most popular constraint is the stretch [2] (or distance) in this class. Furthermore, visual quality of cloth simulation can be improved by adding shear [10] and dihedral bending [1] constraints with a slight performance loss. The mathematical expressions of these constraints are:

$$C_{stretch}(x_1, x_2) = |x_2 - x_1| - l_{12} \tag{8}$$

$$C_{shear}(x_1, x_2, x_3) = \cos^{-1}(M_{12} \cdot M_{13}) - \gamma_{123} \tag{9}$$

$$C_{bend}(x_1, x_2, x_3, x_4) = \cos^{-1}(N_{123} \cdot N_{124}) - \theta_{1234} \tag{10}$$

where l_{12} is the rest length of each edge between the particles, γ_{123} is the rest angle between edges and θ_{1234} is the rest angle between each face primitive. Besides, M and N values are the shear and bending angles:

$$\begin{aligned} M_{12} &= \frac{x_2 - x_1}{|x_2 - x_1|} \\ M_{13} &= \frac{x_3 - x_1}{|x_3 - x_1|} \end{aligned} \tag{11}$$

$$\begin{aligned} N_{123} &= \frac{(x_2 - x_1) \times (x_3 - x_1)}{|(x_2 - x_1) \times (x_3 - x_1)|} \\ N_{124} &= \frac{(x_2 - x_1) \times (x_4 - x_1)}{|(x_2 - x_1) \times (x_4 - x_1)|} \end{aligned} \tag{12}$$

Derivations of the gradients of these constraints can be found in [31]. Besides, geometrically motivated area and volume preservation constraints can be added to this list; however, strain based alternatives of these constraints behave more strongly and accurately which we cover in the next section.

B. Strain Based Constraints

Instead of applying geometric constraints, we can simulate the deformable objects by using strain tensor constraints from [5]. Although, our method accepts triangle meshes as input, we provide the behavior of tetrahedral meshes. We compute the model's center of mass for each iteration, and use that point as the tip of each "tetrahedron". We perform this operation for the simulation of volumetric meshes. Green strain tensor is computed as in Equation (13):

$$G = \frac{1}{2}(F^T F - I) \tag{13}$$

where I is 3×3 identity matrix, and $F \in \mathbb{R}^{3x3}$ is the deformation gradient: $F = D_s D_m^{-1}$ where D_s is the deformed shape matrix ($D_s = (x_1 - x_0, x_2 - x_0, x_3 - x_0)$), and D_m is the initial material matrix ($D_m = (X_1 - X_0, X_2 - X_0, X_3 - X_0)$) for each tetrahedron. The diagonal entries of G from equation (13) determines the stretch and non-diagonal entries are employed for shear. Therefore, strain based constraints are:

$$C_{stretch}(x_0, x_1, x_2, x_3) = S_{ii} - 1 \tag{14}$$

$$C_{shear}(x_0, x_1, x_2, x_3) = S_{ij}, i < j \tag{15}$$

where $S = F^T F$ and x_0 is the center of mass. Although, those constraints are associated with stiffness parameters in [5], XPBD algorithm does not require any stiffness parameters for those constraints (Figure 2 - second column).

For cloth simulation, two dimensional triangle elements of the mesh are processed (Figure 1 - second column). Due to the fact that D_s and D_m are not square it is not possible

to invert D_m. To overcome that problem, we follow the procedure explained in the appendix of [5] and employ the texture coordinates to define D_m in two dimensional space. After we compute the deformation gradient for each triangle ($F^{tri} \in \mathbb{R}^{2x2}$), the constraint functions to simulate cloth are similar to equations (14) and (15) without a center of mass point.

Green strain tensor does not provide volume and area preservation. During the simulations, it is desired to preserve the volume of solids and area of cloth for visually pleasing results. Therefore, we attach these constraints ($C_{volume} = det(D_s) - det(D_m), C_{area} = |x_2 \times x_3|^2 - |X_2 \times X_3|^2$) as an addition to Green strain constraint for strong volume and area preservation. Although over constraining the simulation may cause a modest performance cost, those constraints are easy to implement and provide an effective outcome for the volume and area preservation.

C. Continuous Materials

While explaining the energy embedded Gauss-Seidel iteration, we assume that the constraints are the potential energies of the model. In this section, we accept that constraints are actually the potential energies which are stored in a single tetrahedron according to Equation (16):

$$C(x_0, x_1, x_2, x_3) = \int_\Omega \Psi(F) dX = V\Psi(F) \qquad (16)$$

where V is the initial (undeformed) volume, Ψ is the energy density function which determines the character of the material. In the scope of this paper, we apply three different elastic materials to simulate solid objects. Those are linear elastic material, St. Venant - Kirchhoff material and Neo - Hookean material.

Linear elasticity is a simple and practical constitutive model compared to others (Figure 2 - first column). The energy density function is computed as in Equation (17):

$$\Psi_{LinElas}(F) = \mu\xi : \xi + \frac{\Lambda}{2} tr^2(\xi) \qquad (17)$$

where ξ is the small strain tensor and expressed as $\xi = \frac{1}{2}(F + F^T) - I$. This strain tensor is more suitable for small motions (or deformations). It may not give pleasing results for large deformation cases. μ and Λ are the Lamé coefficients which are associated with Young's modulus and Poisson ratio.

St. Venant - Kirchhoff material is an improved constitutive model based on Green strain tensor (G) from Equation (13) (Figure 2 - third column). Unlike linear elastic material, St. Venant - Kirchhoff material can be applicable to large deformations. The energy density function is computed as in Equation (18):

$$\Psi_{STVK}(F) = \mu G : G + \frac{\Lambda}{2} tr^2(G) \qquad (18)$$

Neo - Hookean material is defined by employing isotropic invariants which are $I_1 = tr(F^T F)$ and $I_3 = det(F^T F)$ (Figure 2 - fourth column). The energy density function of Neo - Hookean material is computed as in Equation (19):

Fig. 3. Duck Lifebuoy (Bob) is simulated with St. Venant - Kirchhoff (left) and Neo - Hookean (right) materials and collision handling is tested. Left: Sphere collision constraint. Right: Convex objects collision constraint.

$$\Psi_{NeoH}(F) = \frac{\mu}{2}(I_1 - log(I_3) - 3) + \frac{\Lambda}{8} log^2(I_3) \qquad (19)$$

In order to obtain the position corrections from Equation (3), the gradients of the constraints have to be computed. The gradients of the continuous material constraints are associated with the first Piola-Kirchhoff stress tensor ($P(F) = \partial\psi(F)/\partial F$). The foundations and detailed derivations of the first Piola-Kirchhoff stress tensors for each mentioned material can be found in [32].

Regarding the cloth simulation, the energy constraint is similar to Equation (16) except the potential energy is stored in triangle elements: $C(x_1, x_2, x_3) = A\Psi(F)$, where A is the initial (undeformed) area. For computing the deformation gradient (F), we follow the same procedure which is discussed in strain based constraints, and the anisotropic elasticity tensor is adapted from [6] (Figure 1 - third column).

D. Collision and Damping

Collision: Two different unilateral collision constraints are employed: *sphere collision* and *convex objects collision* (Figure 3). Sphere collision is satisfied by $C(x) = R - |x - S_{cen}| \geq 0$ where R is the sphere radius and S_{cen} is the sphere center point. Convex objects collision is adapted from [33] with a slight modification. The nearest collision point p is checked with a surface normal n_s for the colliding particle x. This collision surface has to satisfy the constraint $C(x) = (x - p) \cdot n_s \geq 0$. This list can be improved with more advanced collision types.

Damping: Damping is a critical factor for pleasing results. Our damping model is analogous to [3] in which Rayleigh dissipation function is employed: $R = \frac{1}{2}\dot{C}(x)^T \beta \dot{C}(x)$ where $C(x)$ is the constraint function and β is the diagonal matrix of damping coefficients. This damping model is derived directly to the Total Lagrange Multiplier and performs damping locally for each position correction within the Gauss-Seidel iterations. Besides, we have implemented the damping model from [1] and it does not have any conflict with our method.

VI. Implementation and Results

We have implemented our method as a plugin for Autodesk Maya by using C++ as a single-threaded CPU implementation. All test scenarios presented within this paper have been performed on a 4-core Intel i7-2600 3.4 GHz machine with 8

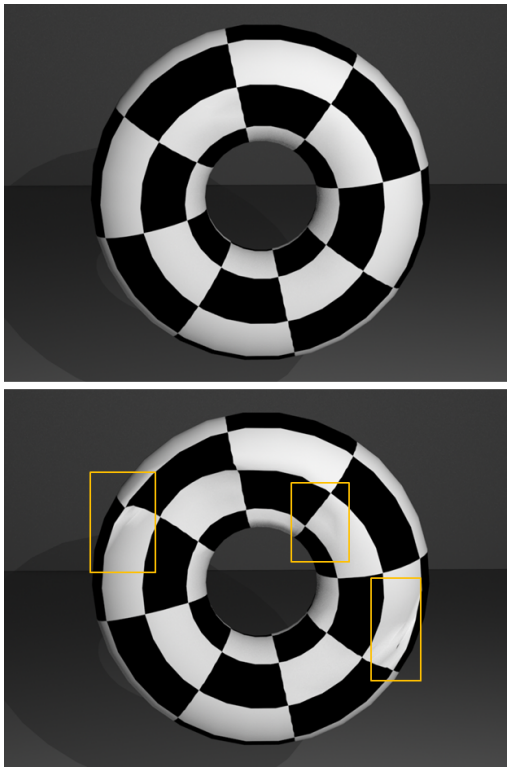

Fig. 4. Neo - Hookean material is simulated. Top: Our Method. Bottom: XPBD. Our method behaves more resistant than XPBD under constant manual stretching and compression. Many unintended buckling effects are obtained during XPBD simulation which can be observed left, upper and right sides of the torus.

GB of RAM and an nVidia GTX 570 GPU. Our experiment setups use only triangle meshes to compare our algorithm 1 with XPBD [3] method. The details of the models and simulation performances are listed in Table I.

We have designed three comparison test scenarios, and we have mostly relied on visual inspection. The major purpose is to evaluate the gain achieved in stability and visual aesthetics over XPBD. In Figure 1, we simulate a falling cloth under the constant gravitational force. Both methods show similar response during falling. This is due to the fact that the difference of the potential and kinetic energy changes is not a notable value that causes a dramatic visual distinction. During our observations, this has not been a surprising fact, because the XPBD approximates the geometric constraints as if they are the actual potentials of a mass-spring system. Furthermore, while simulating the cloth with strain based constraint, strain tensor enforces the elastic behavior. This can be explained by the effect of x-stretch, y-stretch and shear simultaneously to the vertices of the model. On the other hand, while simulating anisotropic material constraint, our method behaves relatively more aesthetic than XPBD under the exact same parametrization. XPBD produces a slight degeneration in the top folded part of the cloth. This can be seen in Figure 6.

Our method and XPBD work well with continuous material

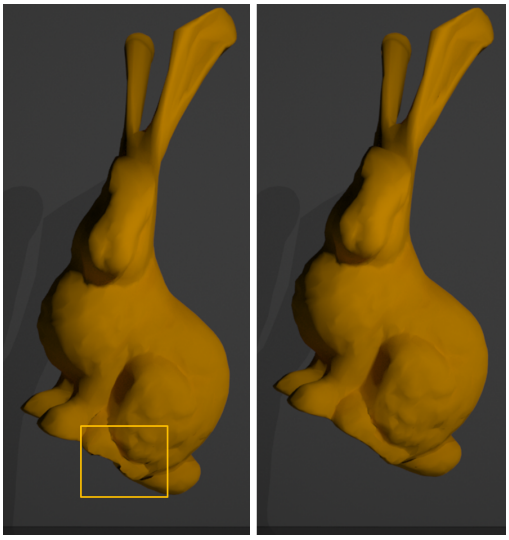

Fig. 5. Bunny model is hanged under constant gravity. St. Venant - Kirchhoff material is simulated. Left: XPBD. Right: Our method. Although both methods behaves identical, our method produces slightly more aesthetic result than XPBD. The difference can be observed on the face and bottom parts of the model.

models. Figures 2 and 4 simulate the simple torus without any constant external forces. We fix the top and bottom vertices, and manually compress and stretch the torus. When we apply the continuous material constraints to the model (St. Venant - Kirchhoff material and Neo - Hookean material), we observe more jiggling in XPBD than in our method. Furthermore, XPBD produces an unexpected but modest buckling during the simulation which is not observed with our method (in Figure 4). This observation motivates our next test scenario. We hang the bunny model under gravity by using the same constraints from the previous test (in Figure 5). During falling, our method and XPBD behave almost identical. However, when the model hangs, our method preserves the shape better than XPBD. During the hang phase, both methods produce some noise in the ears and undesired mesh degeneration in the bottom of the model. This peculiarity is more obvious in XPBD than our method. The main reason is the fact that we plug the energy residual to the constraint in each Gauss-Seidel iteration, and this allows the constraint to correct the corresponding vertex positions partially dependent to the previous iterations. This dependency provides visually more plausible results.

The plots in Figure 7 illustrate the relative error between the proposed method and XPBD. Both algorithms behave similar, and produce some error that is not dramatically different than each other. However, our method produces smaller relative errors. During the cloth simulation (top row of Figure 7), the obtained errors are close to each other. Especially while the cloth falls, both algorithms behave almost identical. This is not unexpected by the fact that our algorithm is an extension to XPBD. On the other hand, when the cloth stabilizes, XPBD produces a slightly larger error than our method. Furthermore, the same case is applied to the bunny model (bottom row of Figure 7). The obtained results are not considerably differ-

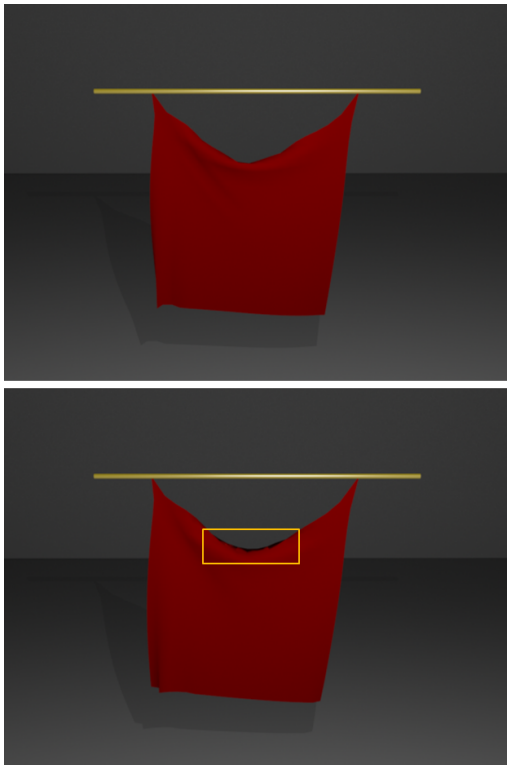

Fig. 6. A piece of cloth is hanged under constant gravity. Anisotropic continuous material is simulated. Top: Our Method. Bottom: XPBD. When both cloths remain stable, our method produces relatively more smooth surface than XPBD. This can be seen in the top folded part of the cloth.

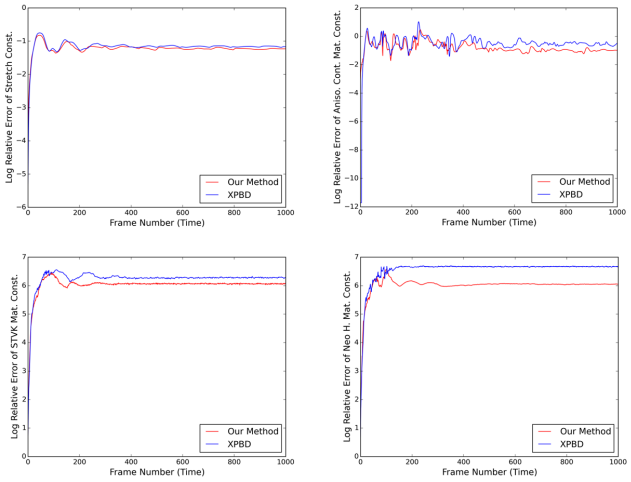

Fig. 7. The relative error of our method is compared with XPBD. Top row: Simulation of falling cloth. Bottom row: Simulation of falling bunny. The relative error values are defined as, $err = log(\frac{C}{I.V.})$, where C represents the corresponding constraint, and $I.V.$ represents the initial value of each corresponding primitive (edge, triangle or tetrahedron).

ent from the cloth simulation case. However, our algorithm stabilizes the bunny with smaller error values. This can be explained as the constraint values benefit from their previous values by using our proposed energy embedded formulation in each Gauss-Seidel iteration. This type of dependency is not

Details of the models with Constraints				Frame Rates (FPS)	
Model	# vertices	# faces (tri.)	# iters	XPBD	Our Method
Cloth (with St. + Sh.)	441	800	20	55	50
Cloth (with 2D Strain)	441	800	20	64	60
Cloth (with Ani. Cont. Mat.)	441	800	20	70	65
Torus (with Lin. Elast.)	900	1800	5	50	46
Torus (with StVK)	900	1800	5	48	45
Torus (with NeoH.)	900	1800	5	46	43
Torus (with 3D Strain)	900	1800	5	38	34
Bunny (with StVK)	4098	8192	10	24	19
Bunny (with NeoH.)	4098	8192	10	22	18
Duck Lifebuoy (Bob with StVK)	3087	6174	10	23	19
Duck Lifebuoy (Bob with NeoH.)	3087	6174	10	21	17

TABLE I

DETAILS OF THE MODELS AND SIMULATION PERFORMANCE RATES FOR OUR EXAMPLES. *St. + Sh.* = STRETCH AND SHEAR CONSTRAINTS TOGETHER, *2D Strain* 2D GREEN STRAIN TENSOR CONSTRAINT, *Ani. Cont. Mat.* = ANISOTROPIC CONTINUOUS MATERIAL CONSTRAINT, *Lin. Elas.* = LINEAR ELASTIC MATERIAL CONSTRAINT, *StVK* = ST. VENANT - KIRCHHOFF MATERIAL CONSTRAINT, *NeoH* = NEO - HOOKEAN MATERIAL CONSTRAINT, *3D Strain* = 3D GREEN STRAIN TENSOR CONSTRAINT. FRAME RATES ARE OBTAINED FROM THE DEFAULT INTERFACE OF MAYA.

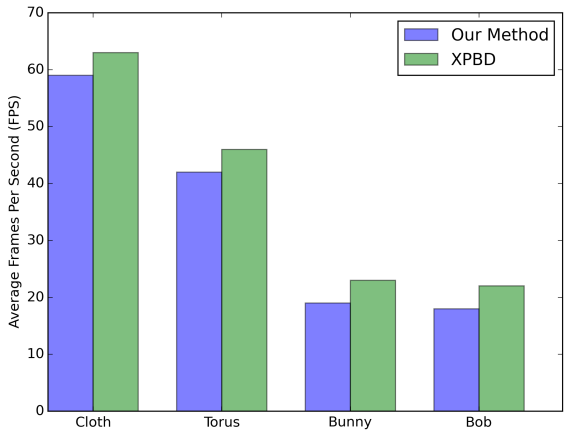

Fig. 8. This plot illustrates the average performance comparison of our method and XPBD. Due to additional energy computations within the Gauss-Seidel process, our method performs 4 FPS in average slower than XPBD. However, this slight performance difference can be considered within the tolerable limits. The details can be seen in Table I.

unrealistic, because Gauss-Seidel iterates the constraints based on their previous states where the proposed algorithm enforces this fact.

In terms of performance rates, XPBD performs slightly more efficient than our proposed method. This is not an unexpected fact due to additional computation steps that are required in our algorithm. In average, the difference is four frames per second which can be seen in Figure 8. Nevertheless, this overhead can be considered within the tolerable limits since both methods perform over the standard interactive rates. Therefore, we consider that the obtained marginal difference is the cost for enhancing the visual quality.

VII. CONCLUSIONS

We have presented a new method that embeds the energy to the Gauss-Seidel iteration process of PBD/XPBD to simulate soft bodies. Our method takes advantage of the energy budgeting principle from [4] and utilizes it within the XPBD method.

It is not limited to any integration technique, simulation step-size and damping approach. We have demonstrated the usability of our method with a large spectrum of constraints that vary from simple stretch to continuous material models. Energy embedded Gauss-Seidel algorithm is straightforward, stable and easy to adapt to the existing frameworks.

A. Limitations and Future Work:

Currently, our method supports only simple static object collisions. In the future, we would like to improve our collision handling with more advanced methods such as [34]. Furthermore, we pursue to adapt different numerical integration methods to our method. Finally, we would like to take the advantage of parallelism on GPU for faster simulation results similar to [35].

ACKNOWLEDGMENT

Authors are grateful to Marco Fratarcangeli for insightful discussions, Keenan Crane for sharing "Duck Lifebuoy (Bob)" model and Christopher Harrison for proofreading. Bunny model is courtesy of Stanford University. This work is a result of the project UID/EEA/50008/2019, funded by the applicable financial framework (FCT/MCTES) (PIDDAC). Furthermore, this work was supported in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, in part by the University of Pará (UFPA), in part by the National Council of Technological and Scientific Development (CNPq), and in part by the Carlos Chagas Filho Research Support Foundation (FAPERJ).

REFERENCES

[1] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff, "Position based dynamics," *Journal of Visual Communication and Image Representation*, vol. 18, no. 2, pp. 109–118, 2007.

[2] T. Jakobsen, "Advanced character physics," in *In Proceedings of the Game Developers Conference*, 2001, pp. 383–401.

[3] M. Macklin, M. Müller, and N. Chentanez, "Xpbd: Position-based simulation of compliant constrained dynamics," in *Proceedings of the 9th International Conference on Motion in Games*, ser. MIG '16. ACM, 2016, pp. 49–54.

[4] J. Su, R. Sheth, and R. Fedkiw, "Energy conservation for the simulation of deformable bodies," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 2, pp. 189–200, feb 2013.

[5] M. Müller, N. Chentanez, T.-Y. Kim, and M. Macklin, "Strain based dynamics," in *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*, 2014.

[6] J. Bender, D. Koschier, P. Charrier, and D. Weber, "Position-based simulation of continuous materials," *Computers & Graphics*, vol. 44, no. 0, pp. 1 – 10, 2014.

[7] J.-D. Gascuel and M.-P. Gascuel, "Displacement constraints for interactive modeling and animation of articulated structures," *The Visual Computer*, vol. 10, no. 4, pp. 191–204, Apr 1994.

[8] F. Faure, "Interactive solid animation using linearized displacement constraints," in *Computer Animation and Simulation '98*. Vienna: Springer Vienna, 1999, pp. 61–72.

[9] R. Goldenthal, D. Harmon, R. Fattal, M. Bercovier, and E. Grinspun, "Efficient simulation of inextensible cloth," *ACM Trans. Graph.*, vol. 26, no. 3, p. 49, 2007.

[10] J. Stam, "Nucleus: Towards a unified dynamics solver for computer graphics," in *Computer-Aided Design and Computer Graphics, 2009. CAD/Graphics '09. 11th IEEE International Conference on*, 2009, pp. 1–11.

[11] M. Tournier, M. Nesme, B. Gilles, and F. Faure, "Stable constrained dynamics," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 132:1–132:10, jul 2015.

[12] M. Servin, C. Lacoursiere, and N. Melin, "Interactive simulation of elastic deformable materials," in *In proceedings of Sigrad Conference*, 2006, pp. 22–32.

[13] J. Bender, M. Müller, and M. Macklin, "A survey on position based dynamics, 2017," in *EUROGRAPHICS 2017 Tutorials*, 2017.

[14] M. Müller and N. Chentanez, "Wrinkle meshes," in *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2010, pp. 85–92.

[15] ——, "Solid simulation with oriented particles," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 92:1–92:10, 2011.

[16] T.-Y. Kim, N. Chentanez, and M. Müller-Fischer, "Long range attachments - a method to simulate inextensible clothing in computer games," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2012, pp. 305–310.

[17] M. Kelager, S. Niebe, and K. Erleben, "A triangle bending constraint model for position-based dynamics," in *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2010)*, 2010.

[18] M. Müller, T.-Y. Kim, and N. Chentanez, "Fast Simulation of Inextensible Hair and Fur," in *Workshop on Virtual Reality Interaction and Physical Simulation*. The Eurographics Association, 2012.

[19] M. Macklin and M. Müller, "Position based fluids," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 104:1–104:12, jul 2013.

[20] F. Marco, "Position based facial animation synthesis," *Computer Animation and Virtual Worlds*, vol. 23, no. 3-4, pp. 457–466, 2012.

[21] O. Cetinaslan and V. Orvalho, "Localized verlet integration framework for facial models," in *Articulated Motion and Deformable Objects - 9th International Conference, AMDO, Proceedings*, ser. Lecture Notes in Computer Science, vol. 9756. Springer, 2016, pp. 1–15.

[22] C. Deul and J. Bender, "Physically-based character skinning," in *Virtual Reality Interactions and Physical Simulations (VRIPhys)*, nov 2013.

[23] N. A. Rumman and M. Fratarcangeli, "Position-based skinning for soft articulated characters," *Computer Graphics Forum*, vol. 34, no. 6, pp. 240–250, 2015.

[24] N. Umetani, R. Schmidt, and J. Stam, "Position-based elastic rods," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '14, 2014, pp. 21–30.

[25] C. Deul, T. Kugelstadt, M. Weiler, and J. Bender, "Direct position-based solver for stiff rods," *Computer Graphics Forum*, vol. 37, no. 6, pp. 313–324, 2018.

[26] A. Stern and M. Desbrun, "Discrete geometric mechanics for variational time integrators," in *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*. New York: ACM Press, 2006, pp. 75–80.

[27] E. Hairer and C. Lubich, "Long-time energy conservation of numerical methods for oscillatory differential equations," *SIAM Journal on Numerical Analysis*, vol. 38, no. 2, pp. 414–441, 2000.

[28] A. Lew, J. E. Marsden, M. Ortiz, and M. West, "Variational time integrators," *International Journal for Numerical Methods in Engineering*, vol. 60, no. 1, pp. 153–212, 2004.

[29] K.-J. Bathe, "Conserving energy and momentum in nonlinear dynamics: A simple implicit time integration scheme," *Comput. Struct.*, vol. 85, no. 7-8, pp. 437–445, 2007.

[30] D. Dinev, T. Liu, and L. Kavan, "Stabilizing integrators for real-time physics," *ACM Trans. Graph.*, vol. 37, no. 1, pp. 9:1–9:19, jan 2018.

[31] O. Cetinaslan, "Localized constraint based deformation framework for triangle meshes," *Entertainment Computing*, vol. 26, pp. 78–87, 2018.

[32] E. Sifakis and J. Barbic, "Fem simulation of 3d deformable solids: A practitioner's guide to theory, discretization and model reduction," in *ACM SIGGRAPH 2012 Courses*, ser. SIGGRAPH '12, 2012, pp. 20:1–20:50.

[33] S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly, "Projective dynamics: Fusing constraint projections for fast simulation," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 154:1–154:11, jul 2014.

[34] M. Müller, N. Chentanez, T.-Y. Kim, and M. Macklin, "Air meshes for robust collision handling," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 133:1–133:9, jul 2015.

[35] M. Fratarcangeli, V. Tibaldo, and F. Pellacini, "Vivace: A practical gauss-seidel method for stable soft body dynamics," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 214:1–214:9, Nov. 2016.