Analytical and Numerical Comparison of Position Based Methods versus Physics Based Formulation of Mechanical Systems

Pablo Real Gómez

Tutor: Ignacio García Fernández

Facultat de Matemàtiques
Universitat de València

Analytical and Numerical Comparison of PositionBased Methods versus Physics Based Formulation of Mechanical Systems

Abstract

Mechanical systems simulation for video games and other interactive applications impose important restrictions as stability, flexibility in the scenes and computational complexity. In the last few years several resolution strategies for mechanical systems with constraints have appeared. Some of the most popular ones in the development of video games use only the positions of the particles and a projection algorithm over the manifold defined by the constraints, avoiding manipulation of the system's first derivative (velocities). In this way, a great numerical stability is obtained. The main drawback of these methods is their dependence in non-physical parameters, so is hard to simulate a specific material. In this work we compare the aforementioned methods and apply them in the simulation of materials with different mechanical properties, proposing algorithms to fit the non-physical parameters of the position based algorithms on concrete, significant scenes, following previous work that tried to approximate the overall behaviour instead of specific scenes. We test this procedures with different basic elements used in the literature, such as cubes or tetrahedra, also further detailing the behaviour of the different parameters used on the simulations of these models on the approximation.

Acknowledgements

First and foremost, I'd like to thank Mariam for *everything*, as she has been my whole support this crazy year. Thanks again for making me who I am and a better person everyday.

I'd also like to thank all my teachers: the good ones and the bad ones, as each of them has made me learn on its own way.

I cannot forget Cristóbal Rodero, for joining me on this crazy adventure, never losing the will to work, once more satisfying our ever-expanding curiosity.

To the CoMMlab: thanks for another year of projects and for whatever has to come.

And last, but not least, thanks to Ignacio for his guidance, his never-ending patience with my never-ending questions, and his support.

As always,

Thanks to everybody who has made this journey possible.

Contents

List of Figures

1 | Introduction

Simulation of solid objects, such as rigid bodies, soft bodies, or cloth, has been an active research topic for more than 30 years, to replace real-world experiments as accurately as possible. One of the fields on which efficient simulators are increasingly needed is the surgeries, as medical simulations enable real-time workloads both in interactive and haptic environments, with the obvious benefits of training surgeries without the actual patient [7].

The simulation, in essence, is fairly simple: we capture the regions with medical imaging devices, identify them, translate them into virtual mechanical objects so the computer can simulate their physical behaviour in a realistic way. But the important part of this simulations is the last one: we have to represent realistically how the different soft tissues deform under the virtual forces applied in the interaction.

There are different techniques to create deformable models, with each one of them having its own advantages and disadvantages. We will focus our attention on some of them at chapter 2, as we will present some of the most used methods in literature, which we will also use later as reference when comparing with other methods. Furthermore, we present some criteria to evaluate this methods on their use on computer graphics. Next, we present a relatively new method of simulation, in which we will specially focus: *Position Based Dynamics* (PBD) [20]. We explain it in detail at section 2.3. This method was developed and is greatly used on the scope of videogames, as it has great stability (thus avoiding the players seeing strange artifacts). However, it is not physically based, so it is not physically accurate, although the great stability usually provides visually pleasing results.

As the PBD is not physically based, our objective on this work will be to adapt PBD to other physically based reference models, to better approximate physical models with the PBD. We will develop the methodology used on chapter 3, comparing it to previous work in the area.

We will present the results obtained with our methodology on chapter 4, detailing the behaviour of the stiffness parameters and the error when varying the different physical parameters associated to the models.

At last, in chapter 5 we present a short survey of our own contributions, while we also remark all possible future work following the conclusion of this work.

2 | Simulation methods for elastic materials

An obvious way (and often used) to approach the simulation of a mechanical system is to study the forces in the system, using Newton's second law to obtain a relationship between deformations and forces, as studied in the field of classical dynamics.

The main advantage of this approach is the accuracy, as we translate the real equations and behaviour of the model to our implementation, making it physically based. Another advantage is that these translations have been widely studied over the past decades.

Often found in the literature we can find the Finite Element Method and Mass-Spring Model, which are two of the most used. We will detail these methods on this chapter and use them as reference to compare later with the Position Based Dynamics.

Furthermore, we need some criteria to evaluate these methods:

- *Generality*: we need to be able to simulate a large spectrum of behaviours, such as different geometries or material properties.
- *Robustness*: we also need to handle difficult configurations such as degenerate geometries, large deformations and large time steps.
- *Numerical convergence*: we need to have numerical convergence, as simulation is worthless if there is divergence.
- *Performance*: we need to perform efficiently in order to not lose time testing.
- *Simplicity*: we need to have easily understandable concepts and light code to ease maintenance and adapt to specific needs.

2.1 | Finite Element Method

The Finite Element Method (also written as FEM) is a numerical method used to find approximate solutions to boundary value problems for partial differential equations. Although it is used to solve PDEs in general, when applied to elastic models it creates force based simulation methods where deformations and forces are related, as we are going to use.

There is not a clear date of invention of the FEM: on [2], Bathe traced back its roots to three research groups, which curiously are mathematicians [10], physicists [25] and engineers [12].

We will use elasticity as an example to let us see how this method works in detail. We have that, in a discrete structure, as told in [8], the deformation is defined by a finite number of parameters (the *deformation modes*), which are collected in a vector. If we take a continuous system, we cannot use a finite vector, and we instead take a vectorial function, which is the solution to the differential equation that defines the problem. However, a closed form is not always assured, so the FEM uses the discretization hypothesis, based on:

- The continuum is divided in adjacent disjointed regions, called *finite elements*.
- Finite elements join each other in a finite number of points, called *nodes*.
- Node displacements are the basic variables of the problem, and determine univocally the deformed structure configuration.
- The displacement of an arbitrary point is determined by the displacement of those nodes belonging to the element in which the point is in.

Note that in order to determine the displacement of an arbitrary point from the nodes of the element, we need to define a *shape function*, which has to guarantee the compatibility in boundary elements.

It is both, the shape functions, and the constitutive equations of the material, which define the stress in the element.

This division of a whole domain into simple parts has some advantages:

- We can obtain better representation accuracy when working with a complex geometry.

- We can represent easily the whole solution.
- We can capture easily the local effects.
- We can include dissimilar material properties.

2.1.1 | A DESCRIPTIVE EXAMPLE

In order to understand how the method works we will use it to solve a generic unidimensional differential equation.

Let us consider the following problem:

$$\begin{cases} u''(x) = f(x) & x \in (0,1) \\ u(0) = u(1) = 0 \end{cases} \tag{2.1}$$

In this problem, we suppose that $f(x)$ is some known function. From here we work to obtain the *weak form* of the problem. We consider the expression $u''(x) - f(x)$ and we multiply this expression by an arbitrary function $v(x)$. We note that

$$(u''(x) - f(x))v(x) = 0 \quad \forall v(x) \iff u \text{ is solution of (2.1)}$$

We can integrate this expression on $[0,1]$:

$$\int_0^1 (u''(x) - f(x))v(x)dx = \int_0^1 u''(x)v(x)dx - \int_0^1 f(x)v(x)dx = 0$$

Now, if we suppose that $v(0) = v(1) = 0$, we can apply integration by parts on the first term:

$$\int_0^1 u''(x)v(x)dx = [u'(x)v(x)]_0^1 - \int_0^1 u'(x)v'(x)dx = -\int_0^1 u'(x)v'(x)dx$$

From here, we have:

$$\int_0^1 u'(x)v'(x)dx + \int_0^1 f(x)v(x)dx = 0$$

This is the *weak form* of the problem. Note that with this form, we only need u to be differentiable, not twice differentiable like in (2.1). Now, rather than using $u(x)$, we consider a discretization $\bar{u}(x)$:

$$\bar{u}(x) = \sum_{i=1}^{n} \phi_i(x)\hat{u}_i = (\Phi\hat{u})(x)$$

The functions ϕ_i are part of the *shape function*, which is subject to:

$$\phi_i(x) = \begin{cases} 1 & x = x_i \\ 0 & x = x_j \neq x_i \end{cases} \qquad \sum_{i=1}^{n} \phi_i(x) = 1$$

Therefore, the problem is reduced to:

$$\int_0^1 (\Phi' \hat{u})(x) v'(x) dx = \int_0^1 f(x) v(x) dx$$

From here, we can choose the shape and trial functions so the above equation is easier to solve, and could be even solved numerically. Note that the solution of (2.1) could be easily obtained by means of antiderivatives (should they exist), but this method can be generalized easily, and used even with a multidimensional problem.

2.1.2 | Elasticity in FEM

As we said before, it is our interest to apply the FEM to the particular case of elasticity.

The *elasticity theory* studies the relationship between the forces applied on a body and the consequent deformation on it. Constitutive models are often expressed as a relationship between the strain ε and the stress σ. When considering linear elasticity, this relationship can be expressed as:

$$\sigma = D\varepsilon \tag{2.2}$$

The matrix D expresses the elastic properties of the material. This is known as the *Saint-Venant-Kirchoff* model [13].

The matrix D is given by:

$$D = \frac{E}{(1+\nu)(1-2\nu)} \begin{pmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & 1-2\nu & 0 & 0 \\ 0 & 0 & 0 & 0 & 1-2\nu & 0 \\ 0 & 0 & 0 & 0 & 0 & 1-2\nu \end{pmatrix} \tag{2.3}$$

We have used two parameters that can uniquely determine an homogeneous isotropic linear elastic material: E and ν.

Definition 2.1. The *elastic modulus*, or *Young's modulus*, often denoted by E, is a measure of the stiffness of a solid material. It defines the relationship between stress and strain in a material.

Definition 2.2. *Poisson's ratio*, often denoted by ν, is the signed ratio of transverse strain to axial strain. That is, ν is the amount of transversal expansion divided by the amount of axial compression, for small values of these changes.

The FEM uses a discretization of this model to turn it into a system of linear equations. If we consider that the node i has a deformation u_i, and a force f_i is being applied to it, according to [26] we have that the FEM formulation defines a matrix K such that:

$$f = Ku \tag{2.4}$$

The FEM stiffness matrix for an element has the form:

$$K = \int_V B^T DBdV$$

In this case, we have that V is the volume of the element, matrix B includes the partial derivatives of the shape functions in the element, and D is the matrix defined on (2.3).

2.2 | Mass Spring Model

Mass-Spring Models (from now on, MSMs) are physically based models with simple structure and small computational cost. These methods are suitable for interactive-time and parallel computing, as it is easy to work with topology changes and/or large deformations. It is very common to find these models, in applications like cloth simulation and surgery simulation.

In these models the object is seen as a collection of point masses which replace the whole body, and they are linked by springs, representing the elastic behaviour of the bodies, which is modelled using linear springs. The spring mesh can have many configurations, depending on the geometry of the object and the topology selected to represent the elasticity properties.

When this model interacts with the environment, the length of the springs can change, inducing forces that propagate over the spring mesh. The equilibrium shape of the body is reached when the sum of each of these forces acting over each node is null. It is usual to add a damping factor to the model in order to improve stability of the system.

Generally speaking, MSMs are easy to construct, physically based models. They allow real-time simulations even when the model has to handle user interactions that involve topology changes. Another well-known advantage is their ability to deal with both large displacements and large deformations. However, it is difficult to find methods to tune the springs to achieve the desired behaviour. Furthermore, [23] compared MSM with FEM, concluding that it is not a realistic method in most situations performing a spectral analysis. Furthermore, it is difficult to express certain constraints like incompressibility, anisotropy, volume conservation, or prevention of volume inversion. Also, it has problems with numerical stability, as the solution diverges when far from the equilibrium.

2.3 | Position Based Dynamics

In the previous methods, forces are computed from the velocities and the actual deformation of the given mesh. However, if we want to control the positions in real-time (in case the user moves a particle, or in other cases), the easiest way is to work precisely with the positions. Note that this avoids integration and other problems such as energy gain. This is the idea behind *Position Based Dynamics*, also denoted by PBD.

Some advantages of using PBD are:

- No instability problems.
- Real-time handling of the scene.
- General constraints.
- The solver is easy to understand and implement.

We will see how the PBD works and detail the algorithm used.

The interested reader can read more on PBD on [6], on [4] or [5].

Furthermore, the PBD has been extended in order to be able to simulate more complex scenes in a easier and more robust way, as in [17], in [19] or [1]

2.3.1 | PBD Formulation

We are going to see the method and the algorithm developed in [20]. We will consider a set of n *particles* (or vertexes), each with the attributes described on Table 2.1.

Particle i	
m_i	mass
p_i, x_i	position
v_i	velocity

Table 2.1: Attributes of each particle

The attributes of the particles are straightforward: positions and velocities are vectors with three components each, as we consider a three-dimensional space.

The relationship between these particles is described by a set of m constraints, each with the attributes described on Table 2.2.

Constraint j	
n_j	cardinality
$C_j \colon \mathbb{R}^{3n_j} \to \mathbb{R}$	scalar constraint function
$\{i_1, \ldots, i_{n_k}\}, i_k \in [1, \ldots, n]$	set of indices
$k_j \in [0, 1]$	stiffness parameter
unilateral or bilateral	type of the constraint

Table 2.2: Attributes of each constraint

The attributes of the constraints are not so straightforward. The cardinality n_j of a constraint is the number of variables that the constraint j is applied on. The scalar constraint function C_j is a function, often with some physical meaning, that measures some type of relationship between particles. The set of indices is simply a subset of all the indices, indicating which particles appear on the constraint. The stiffness parameter k_j defines the *strength* of the constraint. We will say that the constraint j is *unilateral*if it is satisfied when $C_j \geq 0$, and is *bilateral* if it is satisfied when $C_j = 0$.

We now take a moment to explain another term that we are going to use: the inverse mass w_i. This is simply $w_i = m_i^{-1}$. We take this variable and will use it as it makes simple to consider unmovable objects, just by setting $w_i = 0$, that is, like if we considered that the particle has infinite mass. This means that, no matter how much force is applied on the particle, it will not move. However, instead of using infinite mass particles, is desirable to use zero inverse mass particles. Note also that we will not have divisions by zero (that is, infinite inverse mass), as an object without mass has no interest on computer animation.

With all these attributes, and the time step Δt, we can simulate the dynamic object using Algorithm 1, which we now further discuss.

Line (2) simply initializes the state variables.

Line (6) sets the new velocity after a time step. Observe that it is a explicit

Algorithm 1 Algorithm of Position Based Dynamics

1: **for all** vertices i **do**
2: $x_i = x_i^0, \quad v_i = v_i^0, \quad w_i = 1/m_i$
3: **end for**
4: **loop**
5: **for all** vertices i **do**
6: $v_i = v_i + \Delta t w_i f_{ext}(x_i)$
7: **end for**
8: dampVelocities$(v_1, \ldots, v_n)$
9: **for all** vertices i **do**
10: $p_i = x_i + \Delta t v_i$
11: **end for**
12: **for all** vertices i **do**
13: GenerateCollisionConstraints(p_i)
14: **end for**
15: **loop**
16: projectConstraints$(C_1, \ldots, C_{m+m_{Coll}}, p_1, \ldots, p_n)$
17: **end loop**
18: **for all** vertices i **do**
19: $v_i = (p_i - x_i)/\Delta t$
20: $x_i = p_i$
21: **end for**
22: velocityUpdate$(v_1, \ldots, v_n)$
23: **end loop**

forward Euler integration step on the velocities, but adding external forces, mainly gravity (but also the user's interactions). It can be used if we cannot convert some force to constraint. Some authors, like [14], pointed out that PBD can be interpreted as a heuristic variant of the variational implicit Euler method taking the inertial term out of the solver, and into the integration step of the simulation.

Line (8) is used to avoid energy gain problems. In real world, energy is reduced due to heat dissipation or contact with other objects, but in this algorithm we simulate these effects by damping the velocities every time we calculate them.

Line (10) is the same as (6). Note that this is not the final position, it is only a prediction, without taking into account the constraints.

Line (13) generates non-permanent external constraints (m_{coll} constraints), such as collision constraints. These constraints change from time step to time step.

Line (16) is the constraint projection loop. It establishes which constraints are not satisfied and corrects one by one the predicted positions. The involved parameters are the number of iterations, n_{it}, the predicted positions, and both types of constraints: fixed and collision constraints. We will explain these solver deeply on subsection 2.3.2.

Line (19) updates the velocities, as the final positions can be different from the predicted ones.

Line (20) updates the positions of the system with the predictions.

Line (22) modifies the velocity of the colliding particles, according to friction and restitution coefficients.

2.3.2 | Solver

The goal of the solver step (9)-(11) is to correct the predicted positions p_i of the particles, such that they satisfy all constraints (supposing that the constraints are not contradictory). The problem that needs to be solved comprises of a set of $m + m_{coll}$ equations for the $3n$ unknown position components. If $m + m_{coll} > 3n$ ($m + m_{coll} < 3n$) the system is overdetermined (underdetermined). In addition, the equations are in general non-linear. The function of a simple distance constraint ($C(p_1, p_2) = \|p_i - p_j\| - d$) yields a non-linear equation. The fact that makes things even more complicated is the fact that collisions produce inequalities rather than equalities. Solving a non-symmetric, non-linear system with equalities and inequalities which can be either overdetermined or underdetermined is a tough problem, as seen in [9].

We will use equalities on all further work. If there is any inequality, it is enough to check if the inequality is satisfied. In this case, we simply don't mind this

constraint. We will not use the cardinality: we will suppose that every constraint is applied over all n particles, even if there is no dependence of the constraint on the particle.

With that in mind, we can express the constraints as:

$$C_j(p) = 0 \qquad j = 1, \ldots, m$$

Note that we have written $C(p_1, \ldots, p_n)$ as $C(p)$. Now we try to guess a first solution, that is, the first iterate of the projection. Each constraint is linearized in the neighbourhood of this solution using Taylor series:

$$C(p + \Delta p) = C(p) + \nabla_{p_i}^T C(p) \cdot \Delta p_i + \mathcal{O}\left(\|\Delta p_i\|^2\right) = 0$$

Here, Δp_i is the correction of the position of the particle.

We would need $C(p + \Delta p) = 0$ in order to accomplish the constraint. Note that this yields a linear system for the global correction vector Δp_i, which is:

$$\nabla_{p_i}^T C_j(p) \cdot \Delta p_i = -C_j(p) \qquad j = 1, \ldots, m$$

Note that $\nabla_{p_i} C_j(p)$ is the column vector containing the $n$derivatives of the function C_j with respect the n coordinates of p_i, and evaluated in p. Both $\nabla_{p_i}^T C_j(p)$ and the right hand side scalars $-C_j(p)$ are constant because they are evaluated at p before the system is solved. When $m = 3n$ (that is, we have a square matrix) and only equalities are present, the system can be solved by any lineal solver (if we suppose that the system is compatible). If the problem is underdetermined but we have full rank, we can use the pseudoinverse to solve it.

We consider the linearization:

$$C_j(p + \Delta p) = C_j(p) + \left(\nabla_{p_i} C_j(p)\right)^T \cdot \Delta p_i + \mathcal{O}\left(\|\Delta p_i\|^2\right) = 0 \tag{2.5}$$

To conserve momentum (we will explain this on subsection 2.3.3) we restrict to:

$$\Delta p_i = -\lambda \nabla_{p_i} C_j(p) \tag{2.6}$$

Substituting Equation 2.6 into Equation 2.5 (without taking in account the term $\mathcal{O}\left(\|\Delta p_i\|^2\right)$), then solving for λ and substituting it back into Equation 2.6 yields:

$$\Delta p_i = -\nabla_{p_i} C_j(p_1, \ldots, p_n) \frac{C_j(p_1, \ldots, p_n)}{\sum_k \|\nabla_{p_k} C_j(p_1, \ldots, p_n)\|^2} \qquad j = 1, \ldots, M \tag{2.7}$$

In case we want to consider particles with distinct masses or non-unitary masses we weight the corrections Δp_i by the inverse mass $w_i = m_i^{-1}$, so the Equation 2.7 turns into:

$$\Delta p_i = -\nabla_{p_i} w_i C_j(p_1, \ldots, p_n) \frac{C_j(p_1, \ldots, p_n)}{\sum_k w_k \|\nabla_{p_k} C_j(p_1, \ldots, p_n)\|^2} \qquad j = 1, \ldots, M$$

The easiest way of incorporating the stiffness parameter is to multiply the corrections Δp by $k \in [0, 1]$. However, k has a non-linear effect for multiple iteration loops of the solver. The remaining error for a single distance constraint after n_s solver iterations is $\Delta p(1-k)^{n_s}$. To get a linear relationship we multiply the corrections not by k directly, but by $k' = 1 - (1-k)^{1/n_s}$. With this transformation the error becomes $\Delta p(1-k')^{n_s} = \Delta p(1-k)$ and thus becomes linearly dependent on k and independent of n_s as desired. However, the resulting material stiffness is still dependent on the time step of the simulation. Real time environments usually use fixed time steps, in which case this dependency is not problematic.

It is a good moment to discuss stability of the PBD. The authors claim that the solver is unconditionally stable, as the integration steps are performed by a semi-implicit forward Euler integration. However, there has not been a thorough stability analysis. Futhermore, the semi-implicit forward Euler is not unconditionally stable in general. Note that the system solved on the projection of the constraints has order one, as the velocities are modified later. However, we cannot conclude that the PBD is unconditionally stable without further work.

2.3.3 | Momentum conservation

Projecting a set of points according to a constraint means moving the points in a way such that they satisfy the constraint. The greatest issue when moving points directly inside a simulation loop is the conservation of linear and angular momentum. Let Δp_i be the displacement of vertex i by the projection. Classical linear momentum is given by:

$$p = \sum_{i=1}^{n} m_i v_i = \sum_{i=1}^{n} \Delta p_i$$

In order to conserve momentum, we want the derivative of the linear momentum being equal to zero. However, due to Newton's second law, the derivative of the linear momentum is precisely the sum of all forces. That is, if we want to conserve momentum, we need:

$$\sum_{i=1}^{n} F_i = 0$$

Note that $\lambda \frac{\partial C}{\partial t}$ is the force of reaction due to the constraint C. Consider now:

$$\Delta p = \lambda \nabla_p C(p) \tag{2.8}$$

On Equation 2.8 we have all the reaction forces. When we add all of the forces, as the system is closed, we have that the sum is precisely zero, as we wanted. Then, the linear momentum is conserved, as we wanted.

Now we see that angular momentum is also conserved. The angular momentum is given by:

$$L = \sum_{i=1}^{n} r_i \times m_i \Delta p_i$$

In this case, r_i is the distance of all the particles to an arbitrary common rotation center. Again we want to settle at 0 the derivative so, using the cross product properties we have

$$\frac{dL}{dt} = \frac{d}{dt}(r \times p) = \frac{d}{dt}(r \times p) + r \times \frac{dp}{dt} = v \times (m \cdot v) + r \times \frac{dp}{dt} = r \times \frac{dp}{dt}$$

And now, by the same argument done on (2.8), the angular momentum is also conserved.

3 | Characterization of Position Based Dynamics for elastic materials

As we have seen in the previous chapters, PBD is only physically inspired, but its parameter sets do not have a correspondence with the standard parameters in elasticity, like Young's modulus and Poisson's ratio. We intend to determine to which extent is PBD capable of reproducing elastic materials.

As PBD depends on this non-physical parameters, as the stiffness parameter of the constraints, and the number of iterations, we will perform an analysis of an elastic element simulated with PBD, trying to approximate the model to another model with known physical parameter sets, as the FEM.

Several works have faced the problem of a parameter fitting for MSM, like [23] or [18]. However, the mechanical properties of PBD models have not been studied systematically yet. A characterization of the dynamics of the PBD elasticity model would determine to what extent they are capable of reproducing elastic materials.

Note that we have talked about XPBD, which uses a set of energy constraints where physically meaningful parameters are used. However, to the best of our knowledge, almost no previous work has been conducted in order to analyze the mechanical properties of PBD elastic materials based on geometric constraints. This problem is analogous to parameter fitting in modelling methodologies such as MSM.

3.1 | Previous work

Note that this problem is analogous to parameter fitting in modelling methodologies like MSM. Although MSM is physically inspired, its stiffness constants of the sets of springs have no relation with the physical parameters describing the systems.

Analytical approaches try to develop expressions that involve both the Mass-Spring parameters and the elastic parameters describing a deformable material. Van Gelder [11] linearized the system of equations of the MSM to find that, in general, its stiffness matrix cannot be directly equated to a linear FEM stiffness matrix. More recently, Lloyd *et al.* [15] derive analytic expressions for triangular (for 2D) and tetrahedral (for 3D) Mass-Spring elements and find that a closed form solution can only be found on equilateral triangles. Other authors develop analytical expressions for the spring stiffness under certain particular deformations, to fit the parameters that best reproduce such deformations [16, 3]. San Vicente *et al.* [24] follow this approach and use the derived analytical expressions to fit the parameters using data from uniaxial tensile deformations.

These approaches were the base for the methodology used on [22], which was to compare the matrices defined by the linear elasticity (recall Equation 2.2), K^{FEM} and K^{PBD}. The idea was that if matrices defined the behaviour of the models, approximating PBD matrix to FEM matrix would also approximate PBD to FEM, following the methodology of [23].

The methodology could be summarized as follows:

1. Choose a Young's modulus E and a Poisson's ratio ν.
2. Build a reference PBD system and an equivalent FEM element.
3. Build the FEM matrix, and set dt.
4. Fit the PBD parameters solving the optimization problem of reducing the distance between the two matrices.

This previous work constitutes the foundation of the methodology we are going to use on this work..

3.2 | Methodology

The objective is simple: we want to approximate the behaviour of the PBD to some reference model with physically based parameters. We are going to further detail how are we going to do this and what are we trying to accomplish.

Now we will detail the main ideas of our methodology:

- *What are we going to use as reference model?*

The model which we are going to use is FEM (discussed previously on section 2.1), as it has physical significance due to obtaining solutions from the physical equations that describe the model.

- *How do we measure if the behaviour of the PBD approximates the reference model?*

  Let us consider p_i^{PBD} and p_i^{FEM} the positions of the particle i in our model. We let time pass until the rest state of the system is achieved, and we obtain the new positions q_i^{PBD} and q_i^{FEM}. From p and q we can obtain the total deformations of each system, namely u_i^{PBD} and u_i^{FEM}, which describe the behaviour of the systems (recall that by Equation 2.4, the effect of a force in the system is described by the deformation of the system). We take D as the set of all indices that have non-zero deformation on FEM, that is, the set of indices of the vertices that do actually move in the system described by FEM. We also define the relative error e_i between PBD and FEM on each index of D as:

  $$e_i = \frac{\|u_i^{PBD} - u_i^{FEM}\|}{\|u_i^{FEM}\|}$$

  Now, we take the following error e as a measure of the approximation of the PBD towards FEM:

  $$e = \frac{\sum_{i \in D} e_i^2}{|D|}$$

  Note that this is simply the mean square error between two sets of vectors (the deformations of each model).

  We have to take some things into consideration: first and foremost, we need the positions when the rest state is achieved, which we will discuss in a moment. Obtaining the deformations of each model is simple, as $u = q - p$. We also use D to describe the actual vertices that have moved, in order to remove the fixed vertices from the error, as they often appear in the scenes simulated with both PBD and FEM. Note that although $u_i^{FEM} = 0$ when $i \notin D$ does not make sense when taking the point of view of a mathematician, we are not adding neither removing error removing the $i \notin D$ on the sum, as for fixed points, both u_i^{PBD} and u_i^{FEM} are null, and so is their difference.

- *How do we obtain the results of PBD and the reference model?*

  To obtain the results of the simulation using PBD, we have used the code provided by Jan Bender on its GitHub repository, found on `https://github.com/InteractiveComputerGraphics/PositionBasedDynamics`. We use a clone of the source code done on 2017-03-23.

To obtain the results of the simulation using FEM, we have used the code for SOFA (Simulation Open-Framework Architecture), developed mainly by INRIA (Institut National de Recherche en Informatique et en Automatique), which can be found on `https://github.com/sofa-framework/sofa`. We use a clone of the source code done also on 2017-03-23.

We used these library and framework as they are open-source, so anybody can reproduce the methodology and experiments on their own.

- *What experimental setups do we use?*

  We use four experimental setups, three being simple scenes and one being a compound scene, consisting on a combination of the three previous scenes.

  On these experimental setups the only force that we will allow is gravity: no other external forces will appear. Our objective is to characterize elastic materials on simple conditions. We do this as the deformations obtained under these conditions are the same that engineers use on their basic tests for elastic materials.

  All basic scenes will start with a main hexahedron, defined thoroughly in the next item, which will be attached to some surface (namely the floor, a wall, or the ceiling). This is the basic foundation of the scene.

  - The first experimental setup is a simple scene called *GravityFloor*, at which the cube is attached to the floor, that is, the $y = 0$ plane.
  - The second experimental setup is a simple scene called *GravityWall*, at which the cube is attached to a wall, that is, the $z = 0$ plane.
  - The third experimental setup is a simple scene called *GravityCeiling*, at which the cube is attached to the ceiling, that is, the $y = 1$ plane.
  - The fourth experimental setup is a compound scene called *Gravity*. This compound scene considers the same parameters for all three previous scenes (GravityFloor, GravityWall, GravityCeiling).

  The intentionality of these experimental setups, as we have stated previously, is to characterize elastic materials on simple conditions, like the ones specified on the three simple experimental setups. However, some questions arise from this line of thinking: each of these scenes will have a set of best parameters in order to approximate the reference model. However, do these *best* parameters coincide (or at least, are somewhat similar) on the different scenes? This introduces naturally the fourth experimental setup, which takes into account the three previous experimental setups, in order to compare their behaviour under the same set of parameters and see if they are somewhat similar.

- *What parameter sets do we use?*

  The parameters of the experimental setups are the following:

  - dt: The size of the time step on the simulations.
  - N: Number of sub-hexahedrons of the main hexahedron, that is, the hexahedron is formed by $N \times N \times N$ smaller, equal-shaped hexahedrons.
  - n_{it}: Number of iterations of the PBD solver on each time step.
  - E: Young modulus
  - ν: Poisson's coefficient
  - k_d: Stiffness of the distance constraint
  - k_v: Stiffness of the volume constraint

  Note that some of these parameters are only used in either PBD (n_{it}, k_d, k_v) or on FEM (E, ν), while the others (dt, N) are used on both methods.

  The main hexahedron will have a total mass of 1, composed of N^3 hexahedrons of mass N^{-3} each. Due to constraints when using SOFA and PBD, each subhexaedron is formed by five tetrahedrons, as using tetrahedrons is far easier than using hexaedrons. The initial position of the main hexahedron will be $[0, 1]^3$.

  For each experimental setup, we will use the same set of parameters. Furthermore, although we will not state it explicitly in the parameter sets, we will use $\nu = \texttt{0.00:0.01:0.49}$.

  The parameter set is the following:

  - $dt = 0.01, N = 1, n_{it} = 1, E = 10^3$. We will consider this as the *base case*, with standard parameters in the literature.
  - $dt = 0.01, 0.005, 0.002, 0.001$, where $N = 1, n_{it} = 1, E = 10^3$ as in the base case. This is intended to see the effect of the decreasing time step on the PBD behaviour.
  - $N = 1, 2, 3, 4$, where $dt = 0.01, n_{it} = 1, E = 10^3$ as in the base case. This is intended to see the effect of subdividing on the PBD behaviour.
  - $n_{it} = 1, 2, 4, 8$, where $dt = 0.01, N = 1, E = 10^3$ as in the base case. This is intended to see the effect of using more iterations per time step on the PBD behaviour.
  - $E = 10^3, 10^4, 10^5, 10^6$, where $dt = 0.01, N = 1, n_{it} = 1$ as in the base case. This is intended to see the effect of the Young modulus on the PBD behaviour.

Note that in all of these cases, as we have said before, we will also see the behaviour due to Poisson's coefficient.

Also note that the base case is repeated many times at the parameter set: this is intentional, in order to display some consistency when representing the results and ease of reading.

- *What methods do we use to adjust PBD to the reference model?*

  We perform a simple optimization. Note that for a certain subset of the parameter set specified previously, the error is $e = e(k_d, k_v)$. We pretend to minimize the error in order to make PBD behave like the reference model. That is, we solve the optimization problem:

$$\min_{(k_d,k_v)} e(k_d, k_v)$$

  We use the *Pattern Search* optimization method, coupled with a preprocessing sweep of some values for k_d and k_v (namely, $k_d, k_v = \texttt{0.00:0.05:1.00}$) to determine a suitable starting point. We perform a full search on the Pattern Search before moving to the next point, and the multiplier for the tolerance is the golden ratio, as it is a good compromise between the area swept by the pattern search and the velocity of convergence.

  Note that Pattern Search does not require the objective function (recall that in our case it is indeed the error E) to be differentiable. We use Pattern Search instead of other optimization methods (like Gradient Descent, or similars) because, even as the error E is differentiable for low values of ν, the jacobian (which can not be computed easily) gets bigger when ν approaches 0.5, and often the results for the optimization get worse than when using differentiable function compatible methods.

Note that there are several differences between our methodology and the one used on [22]. For example, the dt was not free: it depended on the FEM model used. Also, the distance constraints were different, depending on the relative positions of the particles, whereas we consider that distance constraints should always have the same stiffness.

An important difference is the fact that this methodology tried to approximate PBD to FEM on every possible behaviour for a concrete scene, while we only try to approximate concrete behaviours on concrete scenes. However, it is expected that this loss of generality will lead to better results and less error on the approximations.

Also note that although some improvements and expansions have been made to the PBD (like XPBD [17]), we considered that classic PBD deserved a proper analysis, as it is still being used on many fronts, like on [7].

4 | Results

Throughout all this chapter we will see figures describing the results of the experiments, and we will explain the consequences derived form them. Mainly, for each experiment (that is, when we variate a parameter) we will see two figures. The first one shows the error of the fitting for the different parameters. The second one shows the values of k_d and k_v respectively for the different parameters. Note that when we variate a parameter, the others take their default value (the one used on the base case), and then we will not specify it on the caption of the figure.

For all of the figures, the horizontal axis will be the value of ν, going from 0 to 0.49 with steps of size 0.01 (in Matlab notation, `nu=0.00:0.01:0.49`). The vertical axis will be either the error or the possible range of the stiffness parameters (the ks). When representing the stiffness parameters, the one on the top is always k_d, whereas the on at the bottom is k_v.

The results of the present work have been presented in the form of a poster at the *IV Congreso de Jóvenes Investigadores* on Valencia [21].

4.1 | GravityFloor

Recall that the GravityFloor experimental setup is a simple cube glued to the floor. We can observe it on Figure 4.1.

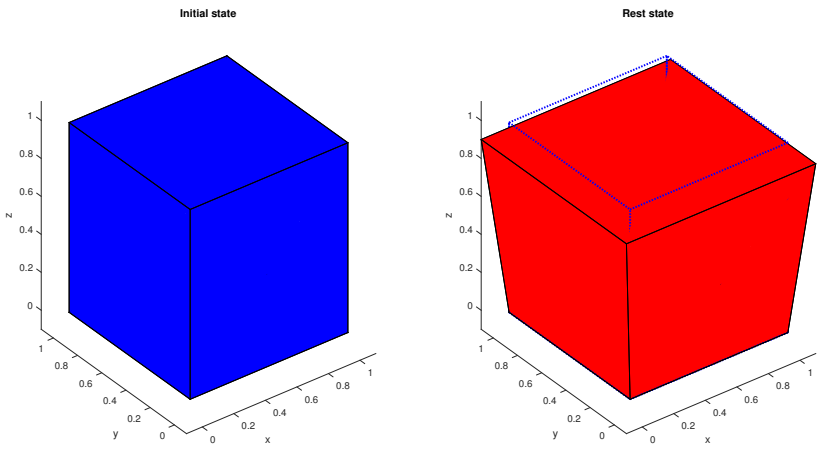

Figure 4.1: Representation of the GravityFloor scene. On the left, the initial state of the scene. On the right, the rest state of the scene.

Base case

We can see both the error and the optimal ks for the base case on Figure 4.2.

We can see that the error is actually small in this experiment, starting at 14%. While $\nu \leq 0.3$, the error is under 20%, and it is only after $\nu > 0.3$ that the error rises significantly up to 75%.

Although we have not said so previously, this increase of error near $\nu = 0.5$ is to be expected, as FEM has a singularity at $\nu = 0.5$, like we can deduce from the expression of the stiffness matrix found on Equation 2.3. This can be applied to all the experiments, so we will not mention it on the following experiments.

We will first observe the behaviour of the stiffness k_d. As ν increases, the stiffness of the distance constraints also increase. Recall that ν is the ratio of transverse strain to axial strain, that is, the ratio at which the element expands on a certain direction if it is compressed on a perpendicular direction. Therefore, a larger ν in this scene means that the cube moves more towards the sides. In order

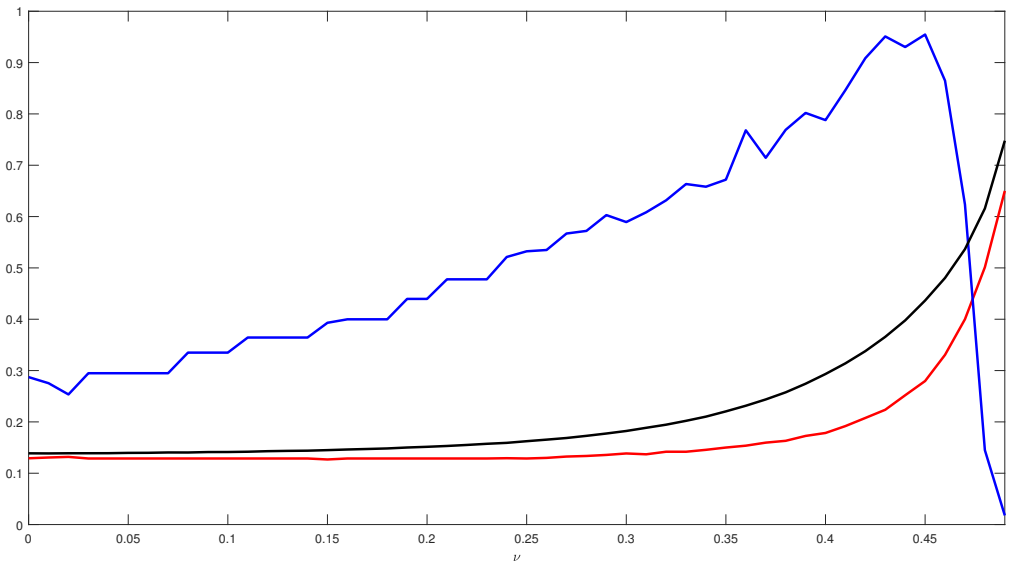

Figure 4.2: Base case for GravityFloor experiment. The parameters used are $dt = 0.01, N = 1, n_{it} = 1$ and $E = 10^3$. We can see in red the optimal k_d for each ν and in blue the optimal k_v. In black we can see the error.

to prevent the cube falling towards the sides, the k_d also increases, as a greater stiffness means less deviation between the particles.

The behaviour of the stiffness of the volume constraints appears somewhat erratic. This is not strange, as there are significantly less volume constraints than distance constraints, and volume constraints affect more vertices that the distance constraints (which only take two vertices), so the overall correction is lesser.

Also, the stiffness of the volume constraints increase as ν does, like before. The argument is the same as the one used with the distance constraint. However, we observe a repentine decrease of k_v when ν is near 0.45. This is due to the fact that k_v steadily increases as ν increases, but as ν approaches 0.45, k_v is almost 1, which is the upper limit for the stiffness. As k_v cannot be greater, k_d increases even more, as if it took k_v's place, so k_v decreases.

Overall, the result is good: we can approximate the behaviour of the reference model with a relatively small error for a simple parameter set, except when ν approaches 0.5, when we already know that the reference model does not behave well.

Variation of dt

First we see the variation of the fitting when dt changes. Recall that dt is the time step, and we are going to decrease it from $dt = 0.01$ to $dt = 0.005, 0.002, 0.001$. Recall that the linear elasticity model supposes that the behaviour is linear, which happens when we take small deformations and/or small forces. Then, reducing the dt should give us also reduced error.

Also recall that the ks depend on the time step, as on each time step the positions are projected in order to prevent constraints not being satisfied. As the time steps become smaller, it is to expect that the constraints keep being satisfied on some time steps after a successful correction. For this, we can expect that when dt decreases, the ks always do.

We can see the results as dt changes on Figure 4.3 and Figure 4.4.

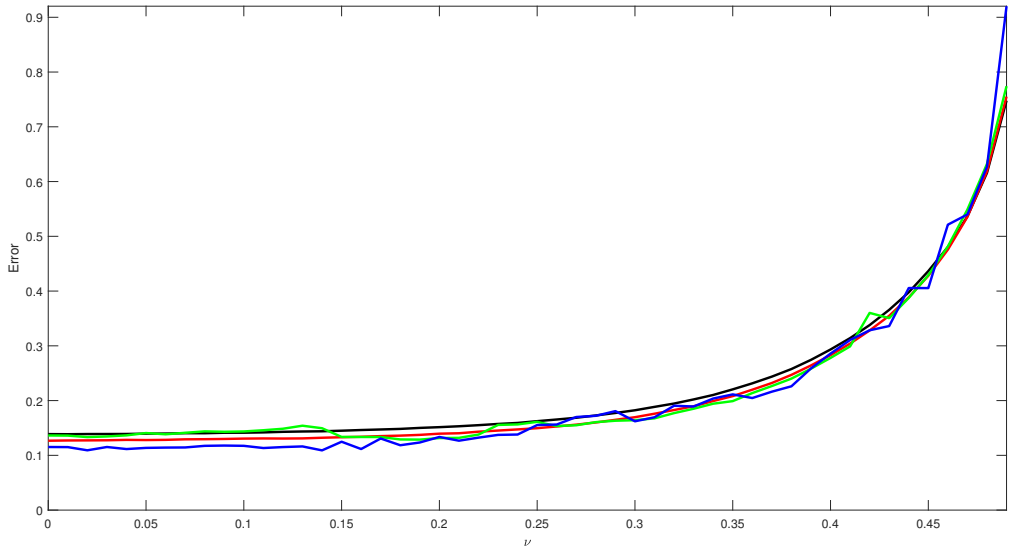

Figure 4.3: Behaviour of error on the GravityFloor experiment when dt changes. In black we have the case $dt = 0.01$, while red represents $dt = 0.005$, green represents $dt = 0.002$ and blue represents $dt = 0.001$.

In these cases the error does not change significatively. The error when using $dt = 0.005$ instead of $dt = 0.01$, the error is reduced, but not greatly. However, comparing with $dt = 0.002$, the error oscillates, presumably due to precision problems. As ν increases, the movements of the vertices to the sides get bigger, and the precision problems decrease. With $dt = 0.001$, the precision problems still appear, but when ν is greater. This makes us think that the error for $dt = 0.001$ for lower ν is actually lower, but the precision errors hide this fact. However, even

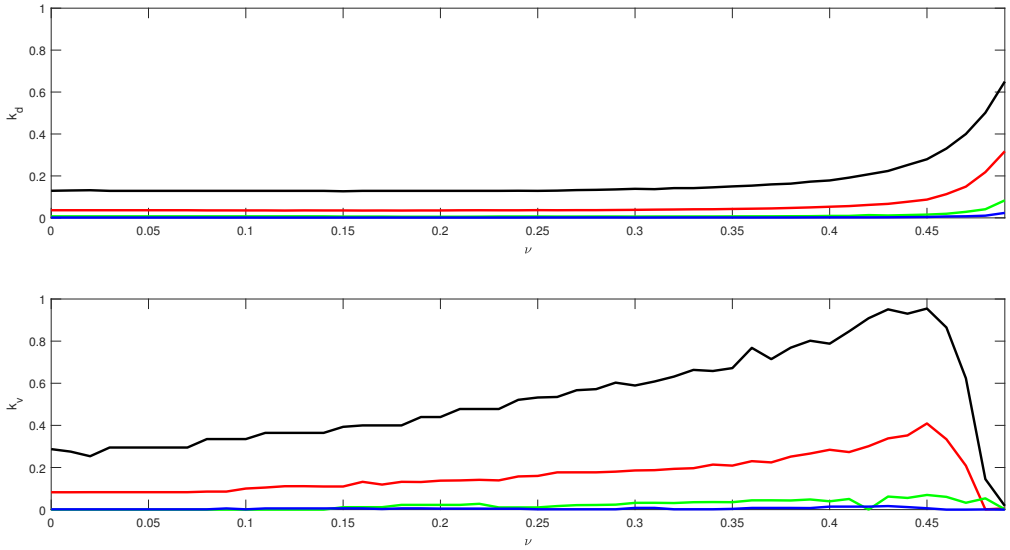

Figure 4.4: Behaviour of the ks on the GravityFloor experiment when dt changes. In black we have the case $dt = 0.01$, while red represents $dt = 0.005$, green represents $dt = 0.002$ and blue represents $dt = 0.001$.

so, we can see that $dt = 0.001$ has a lower error in general. The only exception to this case is when ν approaches 0.5, where the error grows as dt decreases.

The conclusion then is what we could expect: as dt decreases, the elasticity model behaves more linearly, so we can approximate the behaviour better.

Taking into account the ks, we see that they decrease when compared to the base case. However, we could also expect this behaviour, as we know that the PBD stiffness decreases with the time step.

Variation of N

Next we are going to see the behaviour of the experiment when N increases. Recall that N is the number of subdivisions of the base cube, so increasing N means that the respective deformations will be actually smaller for each particle. Therefore, we could expect a similar behaviour to the one observed when dt decreased.

Note that this is not to be expected when considering the ks: previously the same element (be it the cube, or the tetrahedrons in which it is divided) experienced a smaller deformation, whereas now we have a smaller element with the same deformation. Then, we can expect the ks to grow in order to increase rigidity, to prevent great deformations on the smaller elements.

We can see the results as N changes on Figure 4.5 and Figure 4.6.

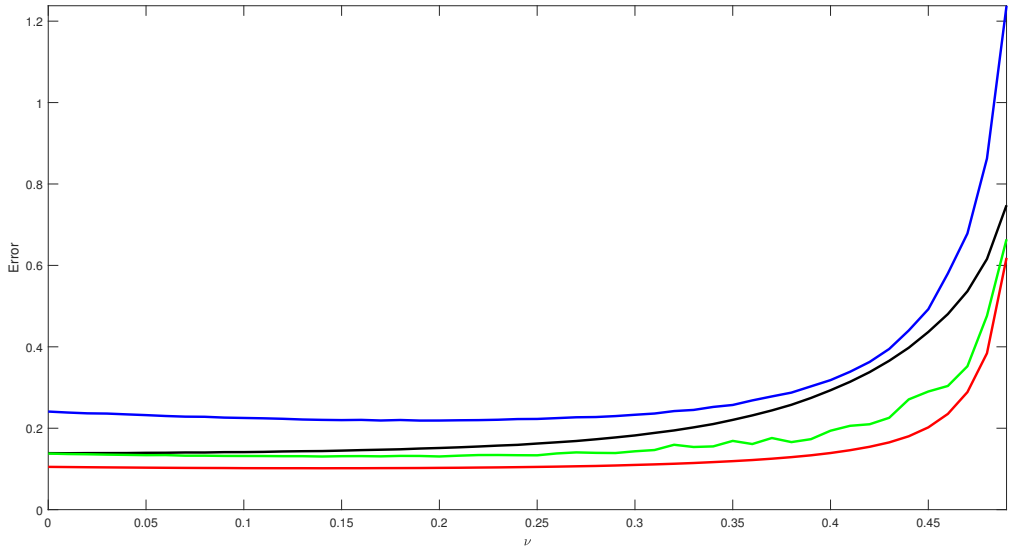

Figure 4.5: Behaviour of the error on the GravityFloor experiment when N changes. In black we have the case $N = 1$, while red represents $N = 2$, green represents $N = 3$ and blue represents $N = 4$.

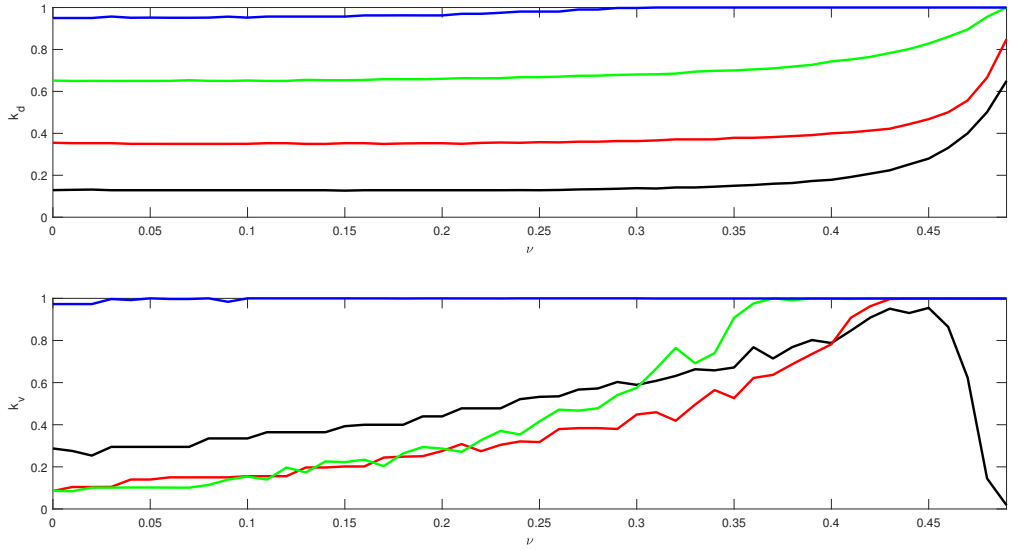

Figure 4.6: Behaviour of the ks on the GravityFloor experiment when N changes. In black we have the case $N = 1$, while red represents $N = 2$, green represents $N = 3$ and blue represents $N = 4$.

The behaviour in this case is similar to the previous case studied, where dt changed. Taking $N = 2$ behaves better that $N = 1$, like we could expect. However,

more subdivisions like when using $N = 3$ (or even worst, $N = 4$) also provide precision errors, so we would have to study in each scene whether subdividing provides better results or actually worsens them due to precision errors.

If we observe the ks, we can see that they increase up to 1 as N gets bigger. As we further subdivide the cube, the movements are actually smaller, and even more near the floor. This is what makes the ks grow to 1, to make the cube more rigid.

Variation of n_{it}

Now we observe the behaviour when n_{it} increases. We do not expect the error to be reduced, as a greater n_{it} means better correction of the possible non-satisfied constraints, but as we are trying to analyze, these constraints do not necessarily have the appropiate physical behaviour.

We can see the results as n_{it} changes on Figure 4.7 and Figure 4.8.

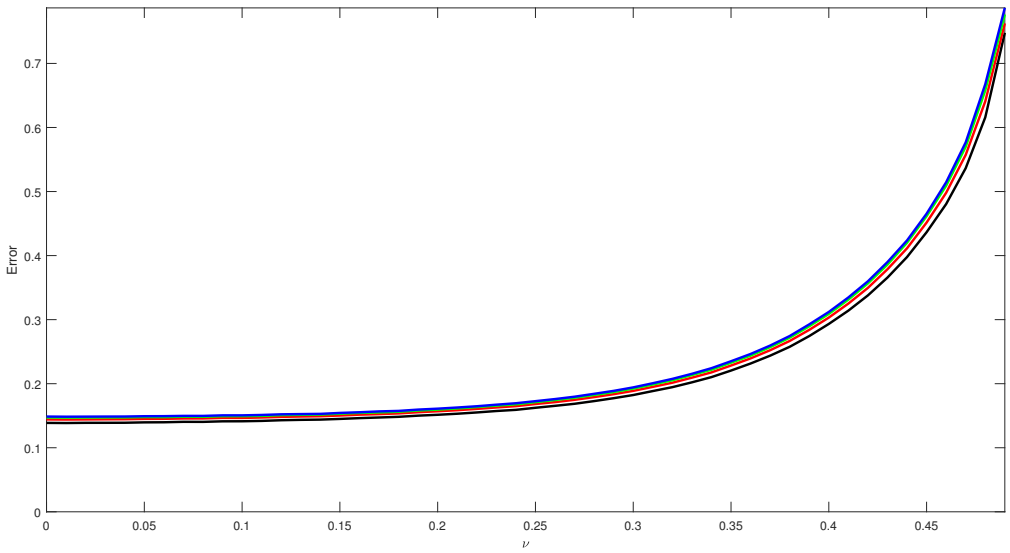

Figure 4.7: Behaviour of the error on the GravityFloor experiment when n_{it} changes. In black we have the case $n_{it} = 1$, while red represents $n_{it} = 2$, green represents $n_{it} = 4$ and blue represents $n_{it} = 8$.

The first thing to take into account is the lack of a significative difference when using different n_{it}. This has a good meaning: the greater n_{it} are often used to further correct the behaviour of PBD. The fact that the error does not change means that our initial approximation was good enough. However, this also means that there is a point at which there is no meaning in increasing n_{it}.

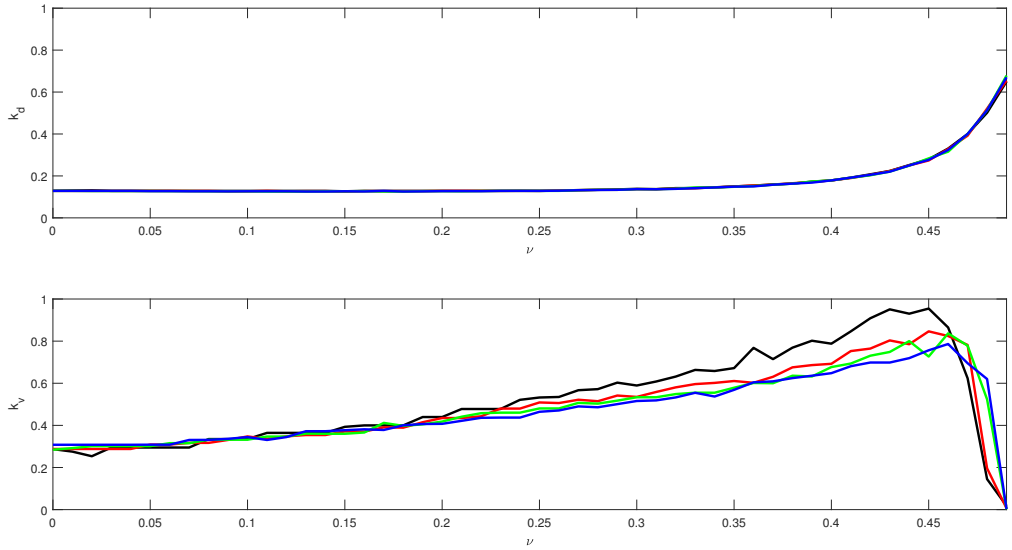

Figure 4.8: Behaviour of the ks on the GravityFloor experiment when n_{it} changes. In black we have the case $n_{it} = 1$, while red represents $n_{it} = 2$, green represents $n_{it} = 4$ and blue represents $n_{it} = 8$.

However, we can see that k_v decreases when increasing n_{it}. As we have mentioned before, the volume constraints affect a greater number of vertices, and their effect on the position projection is consequently lesser. Multiple iterations only further their diminishing effect.

Variation of E

To end with GravityFloor, we see the behaviour as E increases. Recall by the definition of E that it is a measure of stiffness, and as such, as E increases we expect the ks to also grow. In this case, we can actually see that there will be a point when the stiffness on FEM (E) will continue to rise, but on PBD (k_d and k_v) it will reach the maximum limit of 1, so the error will increase greatly.

We can see the results as E changes on Figure 4.9 and Figure 4.10. Note that in this case, the scale of the error is logarithmic. In this case the difference to the original linear axis is not significative, but we present the results with logarithmic scale as in other scenes it will be strictly necessary.

Increasing E also increases the error, as we predicted: a greater E means that the same stress is equivalent to less strain. This is the similar to when N changed: less deformation meant tolerance problems and greater stiffness, which is limited by the PBD. This checks out, as the ks approach the max stiffness 1 as E increases.

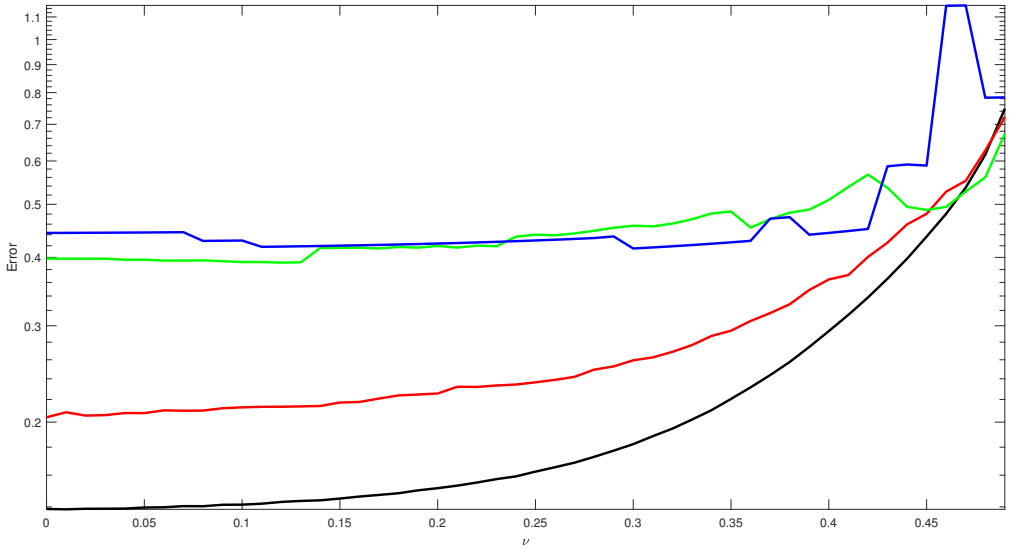

Figure 4.9: Behaviour of the error on the GravityFloor experiment when E changes. In black we have the case $E = 10^3$, while red represents $E = 10^4$, green represents $E = 10^5$ and blue represents $E = 10^6$.

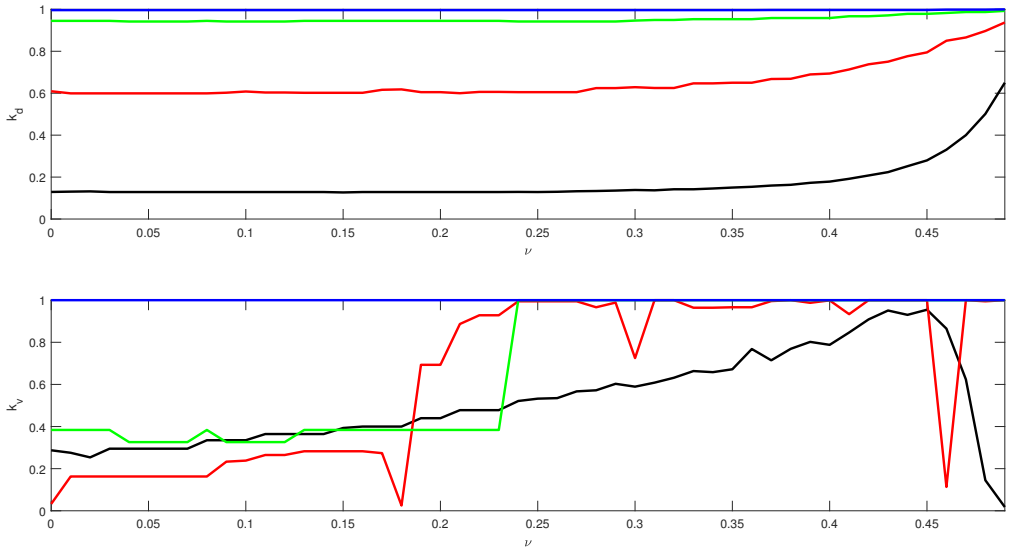

Figure 4.10: Behaviour of the ks on the GravityFloor experiment when E changes. In black we have the case $E = 10^3$, while red represents $E = 10^4$, green represents $E = 10^5$ and blue represents $E = 10^6$.

4.2 | GravityWall

Recall that the GravityWall experimental setup is a simple cube, this time glued to a wall, and falling due to gravity. We can observe it on Figure 4.11.

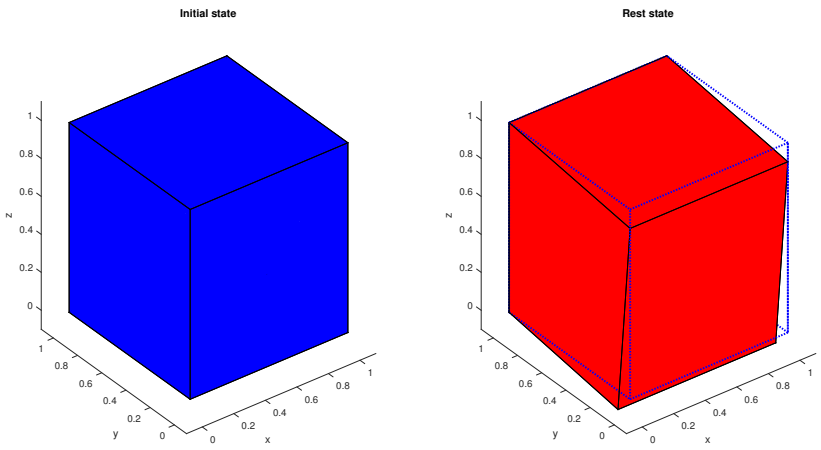

Figure 4.11: Representation of the GravityWall scene. On the left, the initial state of the scene. On the right, the rest state of the scene.

Base case

We can see both the error and the optimal ks for the base case on Figure 4.12.

The first thing that catches our eye is that the error decreases as ν increases, unlike in the GravityFloor scene. We also see that k_v is almost constantly 1, so that the cube doesn't fall or cross the wall. However, k_d also decreases as ν increases, in order to make the cube more deformable.

Note that in this case the error stays under a 12.5%, and decreases almost to the 10% when ν increases, so it seems that the GravityWall scene overall will give a better approximation to the reference model than the GravityFloor scene.

A important note has to be done as the singularity on $\nu = 0.5$ has seemingly vanished on this scene. This is, however, only an impression, as the singularity is still there and, although almost unseen, the error begins to grow when ν is greater than $\nu = 0.47$ approximately. However, the error does grow at a much smaller rate than on the GravityFloor experiment.

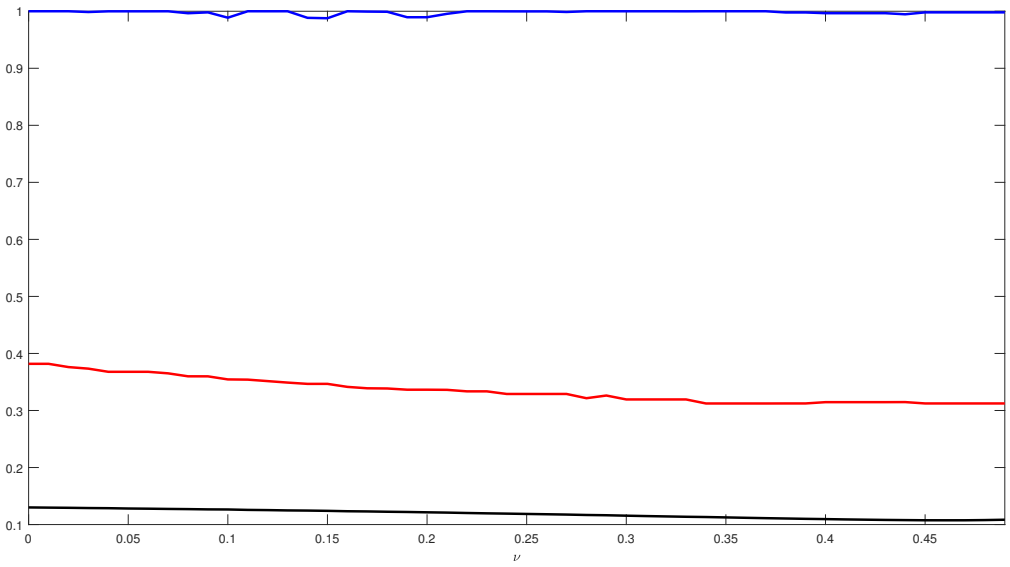

Figure 4.12: Base case for GravityWall experiment. The parameters used are $dt = 0.01, N = 1, n_{it} = 1$ and $E = 10^3$. We can see in red the optimal k_d for each ν and in blue the optimal k_v. In black we can see the error.

Variation of dt

We can see the results as dt changes on Figure 4.13 and Figure 4.14.

These are similar to the ones observed on the GravityFloor experiment, as the error oscillates when the time step is smaller. Observe that k_d also decreases when the time step decreases, as does k_v, due to the same effect that we talked about before.

Note that on this case, the chaotic behavior due to precision errors increases greatly when dt is smaller, although the overall error is almost at the same level, only increasing to a 15% from the original initial 12.5%.

If we observe the ks, note that they decrease too as they did on GravityFloor. However, k_d has almost the same growth on each case, whereas k_d absorbs the chaotic behavior of the precision errors.

Variation of N

We can see the results as N changes on Figure 4.15 and Figure 4.16.

These are also similar to the results observed with this parameter set on the GravityFloor experiment. The case $N = 4$ is even more pronunciated than its

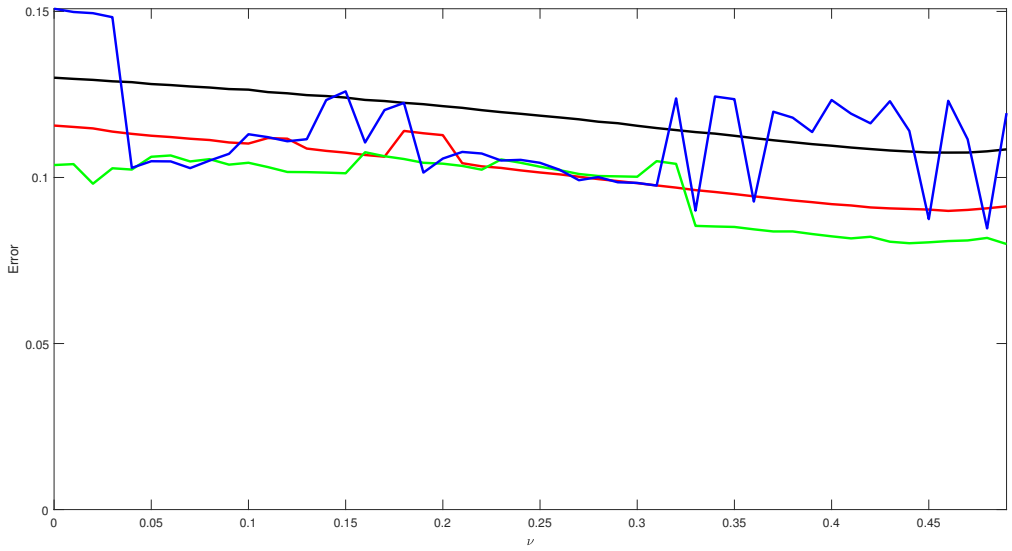

Figure 4.13: Behaviour of error on the GravityWall experiment when dt changes. In black we have the case $dt = 0.01$, while red represents $dt = 0.005$, green represents $dt = 0.002$ and blue represents $dt = 0.001$.

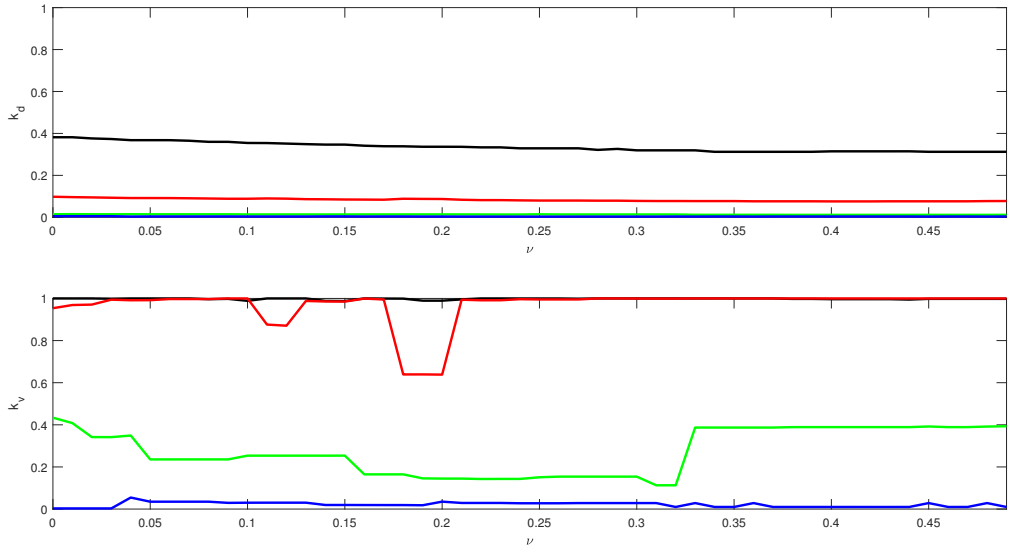

Figure 4.14: Behaviour of the ks on the GravityWall experiment when dt changes. In black we have the case $dt = 0.01$, while red represents $dt = 0.005$, green represents $dt = 0.002$ and blue represents $dt = 0.001$.

counterpart on the GravityFloor experiment. Note that as N grows, the singularity at $\nu = 0.5$ is more and more visible. The chaotic behaviour of the precision errors is also more visible at the $N = 4$ case.

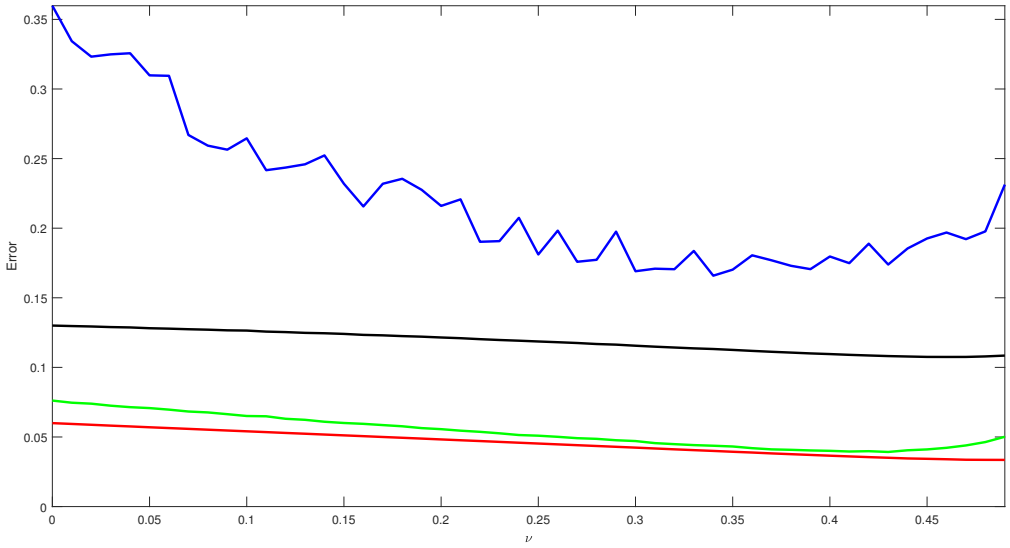

Figure 4.15: Behaviour of the error on the GravityWall experiment when N changes. In black we have the case $N = 1$, while red represents $N = 2$, green represents $N = 3$ and blue represents $N = 4$.

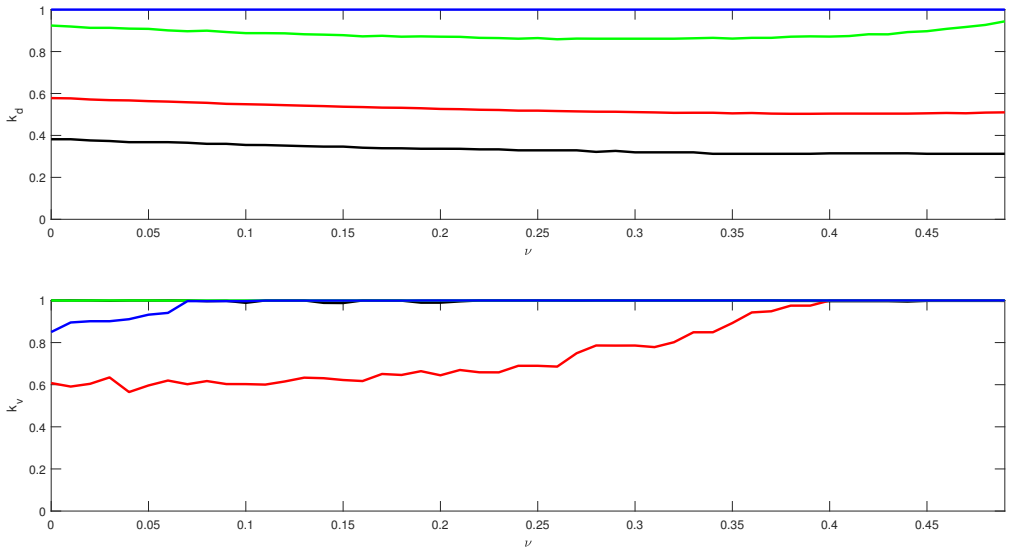

Figure 4.16: Behaviour of the ks on the GravityWall experiment when N changes. In black we have the case $N = 1$, while red represents $N = 2$, green represents $N = 3$ and blue represents $N = 4$.

Note that the stiffness of the distance constraints k_d is consistent with what we observed on the GravityFloor scene, but the behaviour of the k_d is significatively different for $N = 2$. It seems that a smaller size of the elements benefits the

behaviour of the system under the relatively small deformations of the GravityWall scene when compared to the GravityFloor.

Variation of n_{it}

We can see the results as n_{it} changes on Figure 4.17 and Figure 4.18.

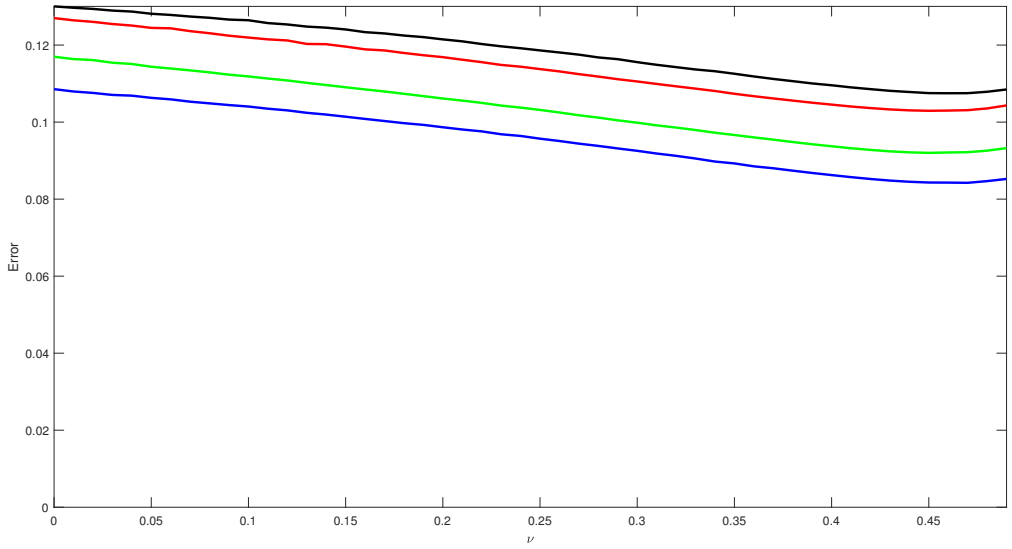

Figure 4.17: Behaviour of the error on the GravityWall experiment when n_{it} changes. In black we have the case $n_{it} = 1$, while red represents $n_{it} = 2$, green represents $n_{it} = 4$ and blue represents $n_{it} = 8$.

This case actually differs from its GravityFloor equivalent. Here, the error decreases as n_{it} increases, as could be expected: in this case, it seems that the constraints are somewhat consistent with the physical behaviour, so more iterations on the positions projection means easily enforced constraints, which means more physical behaviour.. This does not affect the k_v, which we have already seen that is the predominant in this scene. However, as we have mentioned before, a greater n_{it} is equivalent to less stiffness, which we see on the k_d.

Variation of E

We can see the results as E changes on Figure 4.19 and Figure 4.20.

Here we see why previously we used the logaritmic scale for the error. Here we see that the error increases significantly with E, unlike in the GravityFloor scene.

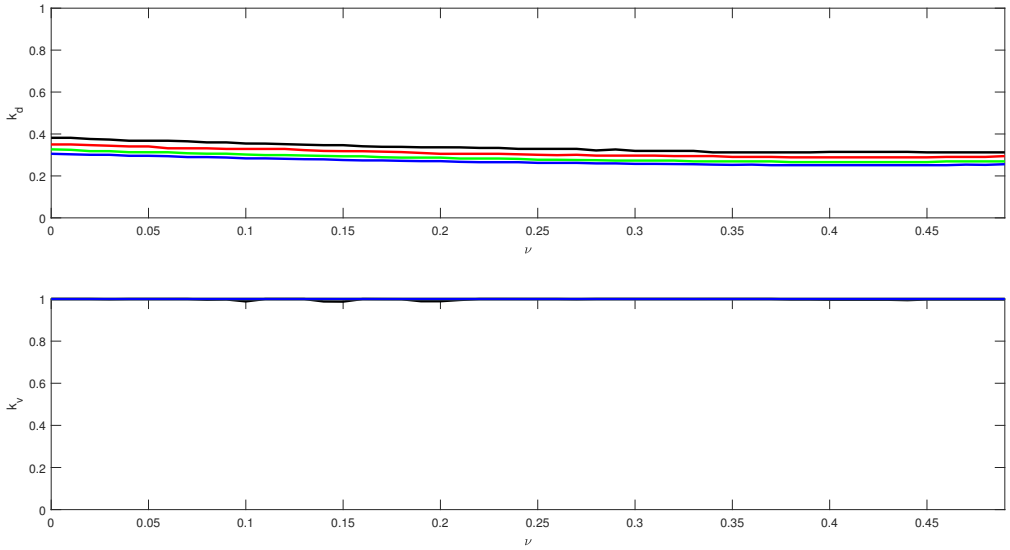

Figure 4.18: Behaviour of the ks on the GravityWall experiment when n_{it} changes. In black we have the case $n_{it} = 1$, while red represents $n_{it} = 2$, green represents $n_{it} = 4$ and blue represents $n_{it} = 8$.

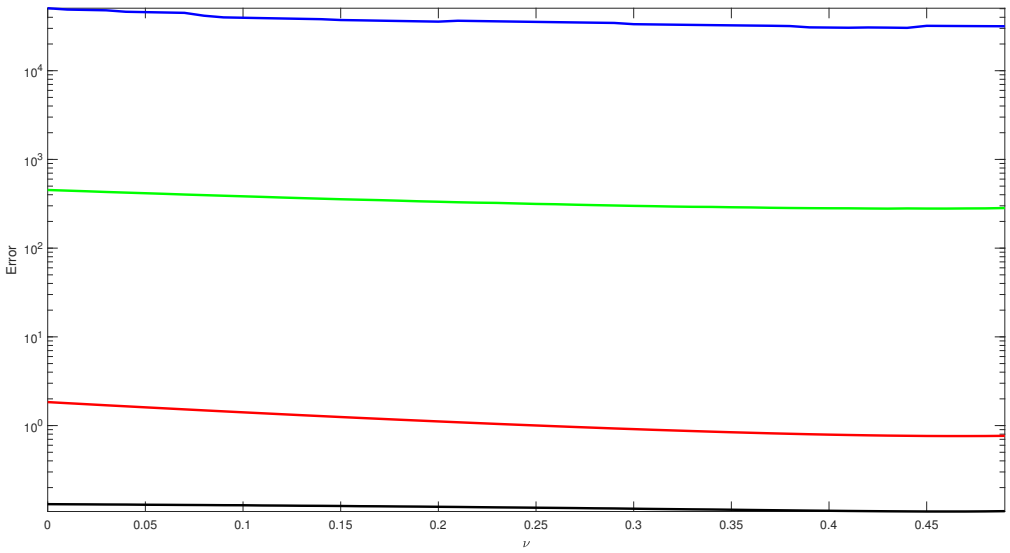

Figure 4.19: Behaviour of the error on the GravityWall experiment when E changes. In black we have the case $E = 10^3$, while red represents $E = 10^4$, green represents $E = 10^5$ and blue represents $E = 10^6$.

This is due to the fact that a greater E actually means less deformation, and PBD has a limit for rigidity (when the ks are 1, as we have said before). We actually see

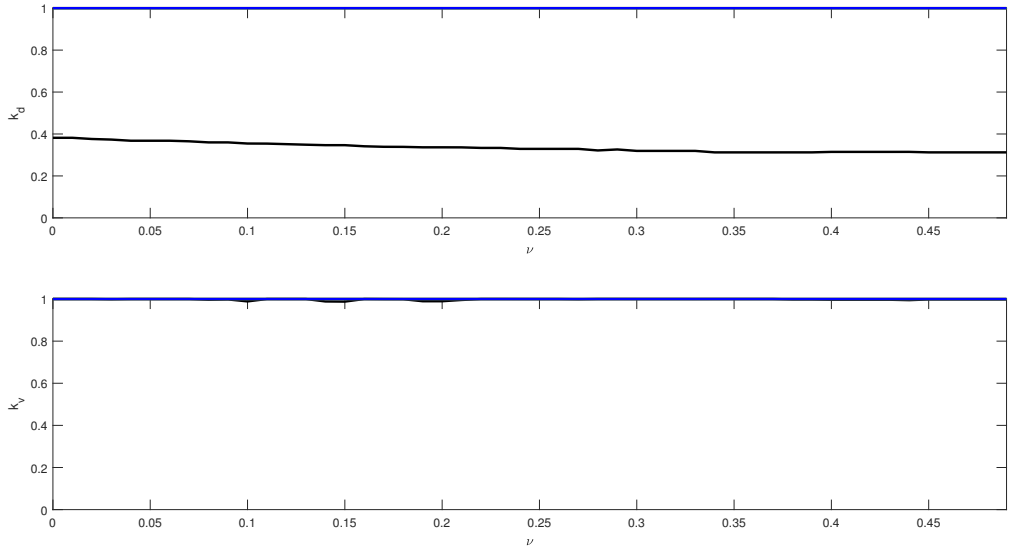

Figure 4.20: Behaviour of the ks on the GravityWall experiment when E changes. In black we have the case $E = 10^3$, while red represents $E = 10^4$, green represents $E = 10^5$ and blue represents $E = 10^6$.

this limit in Figure 4.20, as the ks increase to 1 when $E > 10^3$. Further increasing E only makes the error greater, so we can see that soft materials will be easier to simulate with PBD.

4.3 | GravityCeiling

The last simple scene is GravityCeiling. Recall that this scene is simply a cube glued to the ceiling, under the effect of the gravity. We can see it on Figure 4.21.

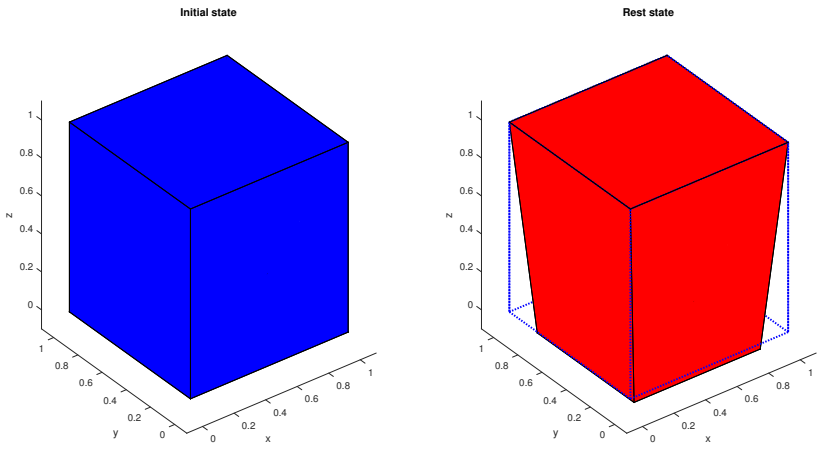

Figure 4.21: Representation of the GravityCeiling scene. On the left, the initial state of the scene. On the right, the rest state of the scene.

Base case

We can see both the error and the optimal ks for the base case on Figure 4.22.

We can see that in this case the error is greater than on the previous two experiments. However, it is not significantly greater: only a 10% more at the start, and around 30% more at the end. The behaviour of k_d and k_v is also similar to the GravityFloor case, with k_d being almost identical and just a little smaller. Looking at k_v we can see that it is also smaller, and grows less, only increasing significantly near $\nu = 0.45$ (when k_v started to decrease on GravityFloor), and staying almost constant until $\nu = 0.49$.

Variation of dt

We can see the results as dt changes on Figure 4.23 and Figure 4.24.

The behaviour of the error as the time step decreases is similar to the GravityWall scene, as the error decreases when the time step decreases, with a few oscillations

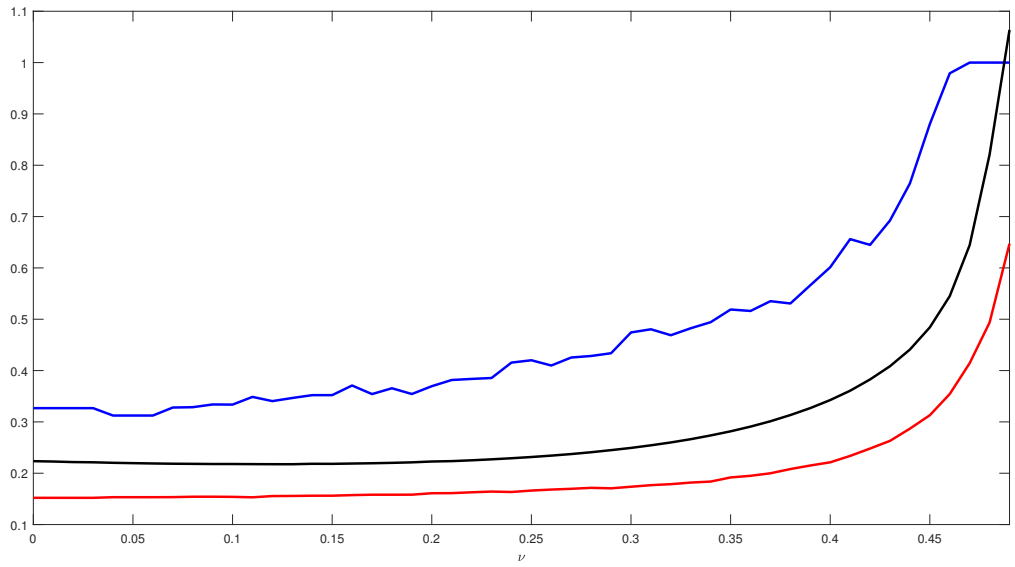

Figure 4.22: Base case for GravityCeiling experiment. The parameters used are $dt = 0.01, N = 1, n_{it} = 1$ and $E = 10^3$. We can see in red the optimal k_d for each ν and in blue the optimal k_v. In black we can see the error.

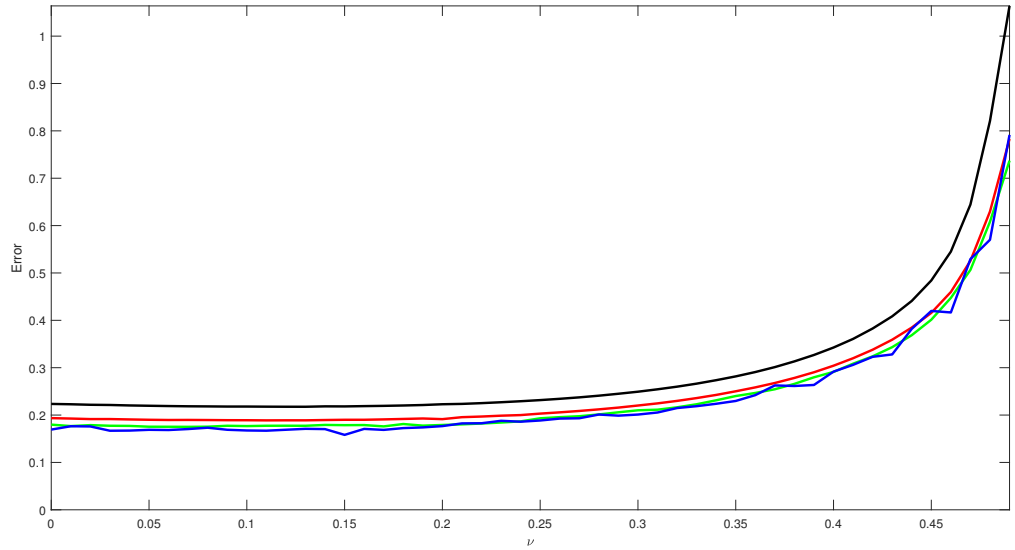

Figure 4.23: Behaviour of error on the GravityCeiling experiment when dt changes. In black we have the case $dt = 0.01$, while red represents $dt = 0.005$, green represents $dt = 0.002$ and blue represents $dt = 0.001$.

due to precision errors. Note that in this case the oscillations are smaller and almost imperceptible until the end. The ks also decrease as the time step decrease,

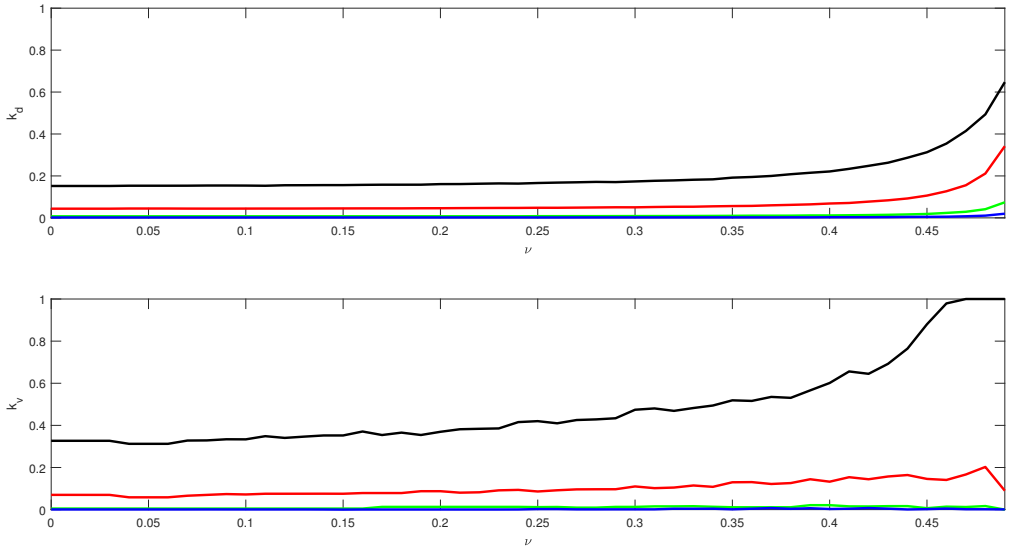

Figure 4.24: Behaviour of the ks on the GravityCeiling experiment when dt changes. In black we have the case $dt = 0.01$, while red represents $dt = 0.005$, green represents $dt = 0.002$ and blue represents $dt = 0.001$.

especially the k_v, as we could expect, due to constraints being satisfied on a greater region for same values of k.

Variation of N

We can see the results as N changes on Figure 4.25 and Figure 4.26.

We can see that in this cases $N = 3$ behaves as expected, while $N = 4$ presents precision errors, whereas in the other cases $N = 2$ was the last case that did not show any precision errors.

The ks grow until the stiffness is at maximum level, as the cube movements are smaller when we further subdivide the cube, making the cube more rigid. Note that the ks are higher as N increases, so a higher N means that it is easier to hit the limit stiffness, as we have seen before.

On Figure 4.26 we can observe clearly how the limit stiffness for k_v also affects the behaviour of k_d, as when k_v gets near 1 on each case, the k_d starts growing.

Variation of n_{it}

We can see the results as n_{it} changes on Figure 4.27 and Figure 4.28.

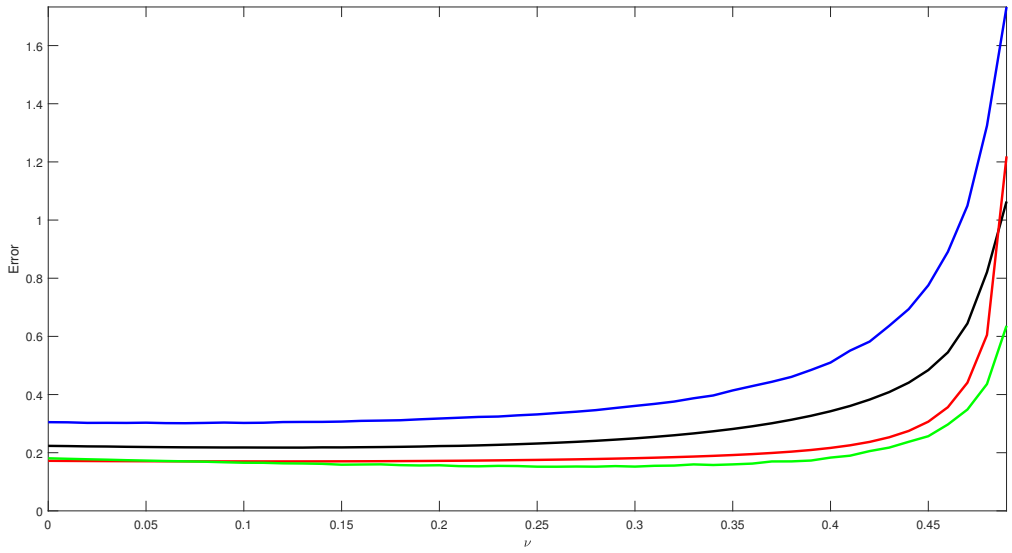

Figure 4.25: Behaviour of the error on the GravityCeiling experiment when N changes. In black we have the case $N = 1$, while red represents $N = 2$, green represents $N = 3$ and blue represents $N = 4$.

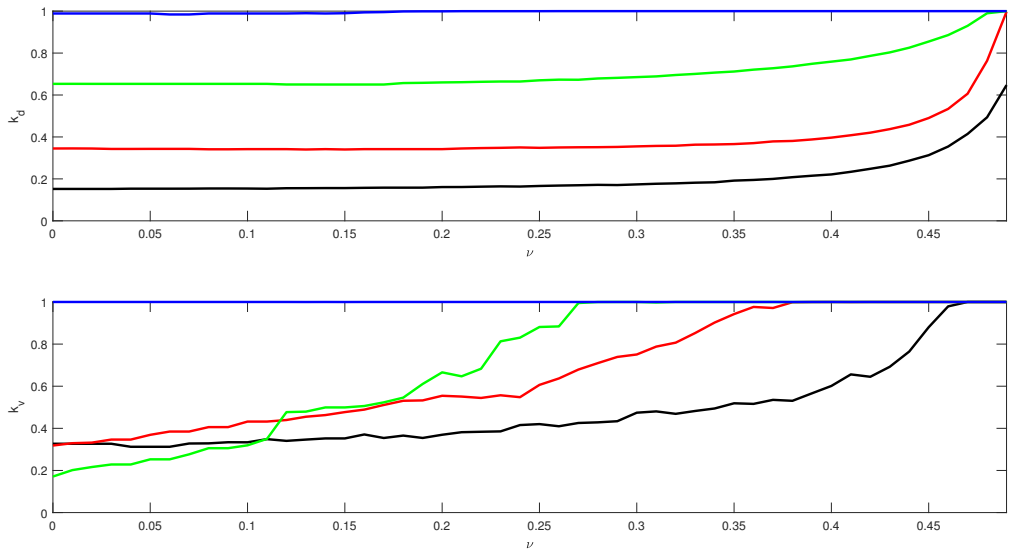

Figure 4.26: Behaviour of the ks on the GravityCeiling experiment when N changes. In black we have the case $N = 1$, while red represents $N = 2$, green represents $N = 3$ and blue represents $N = 4$.

Like on the GravityWall scene, further iterations reduce the error, but not as significantly on this scene. The cause of this behaviour is the same explained on

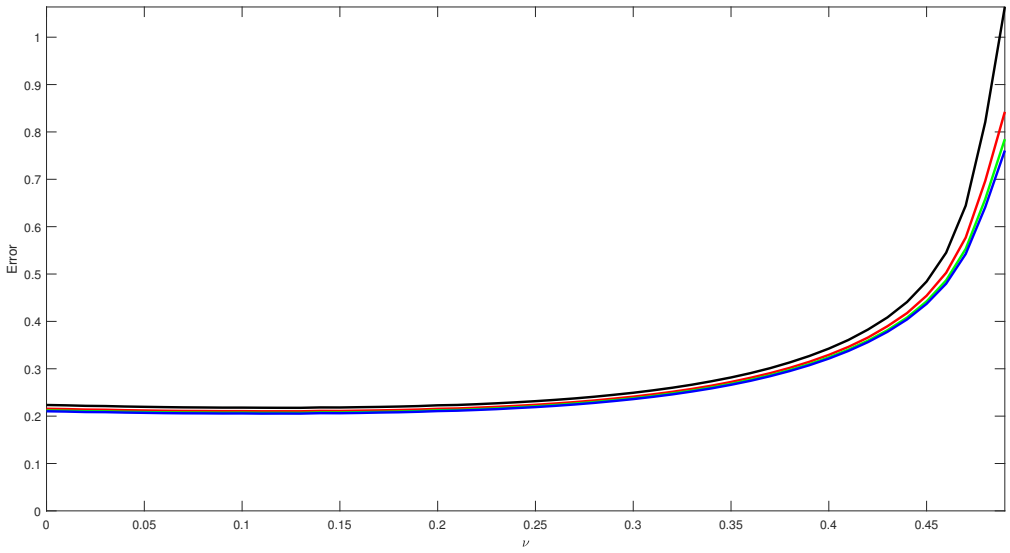

Figure 4.27: Behaviour of the error on the GravityCeiling experiment when n_{it} changes. In black we have the case $n_{it} = 1$, while red represents $n_{it} = 2$, green represents $n_{it} = 4$ and blue represents $n_{it} = 8$.

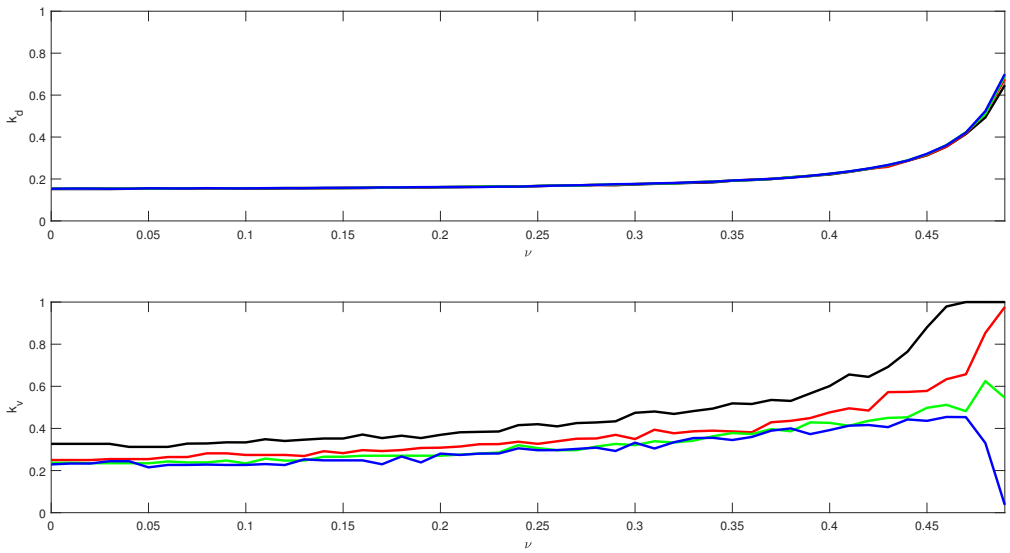

Figure 4.28: Behaviour of the ks on the GravityCeiling experiment when n_{it} changes. In black we have the case $n_{it} = 1$, while red represents $n_{it} = 2$, green represents $n_{it} = 4$ and blue represents $n_{it} = 8$.

the GravityWall experiment.

In fact, the values of ν near $\nu = 0.5$ are the most benefited from a higher n_{it},

as the difference in the respective errors is more noticeable near the singularity.

Note that k_v decreases with more iterations, but k_d is almost constant, and the difference is not significative, only differing at the singularity on $\nu = 0.5$, like the error. This is also similar to the behaviour on the GravityFloor experiment.

Variation of E

We can see the results as E changes on Figure 4.29 and Figure 4.30.

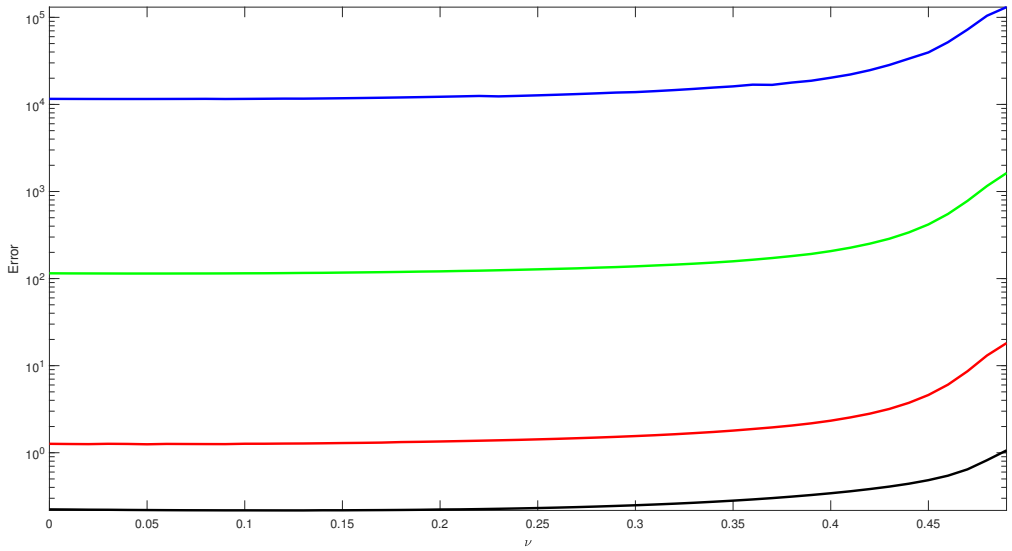

Figure 4.29: Behaviour of the error on the GravityCeiling experiment when E changes. In black we have the case $E = 10^3$, while red represents $E = 10^4$, green represents $E = 10^5$ and blue represents $E = 10^6$.

The error increases greatly as we have also seen on GravityWall. The stiffness k_d increases greatly to prevent the cube falling, and the k_v grows quickly when k_d approaches 1.

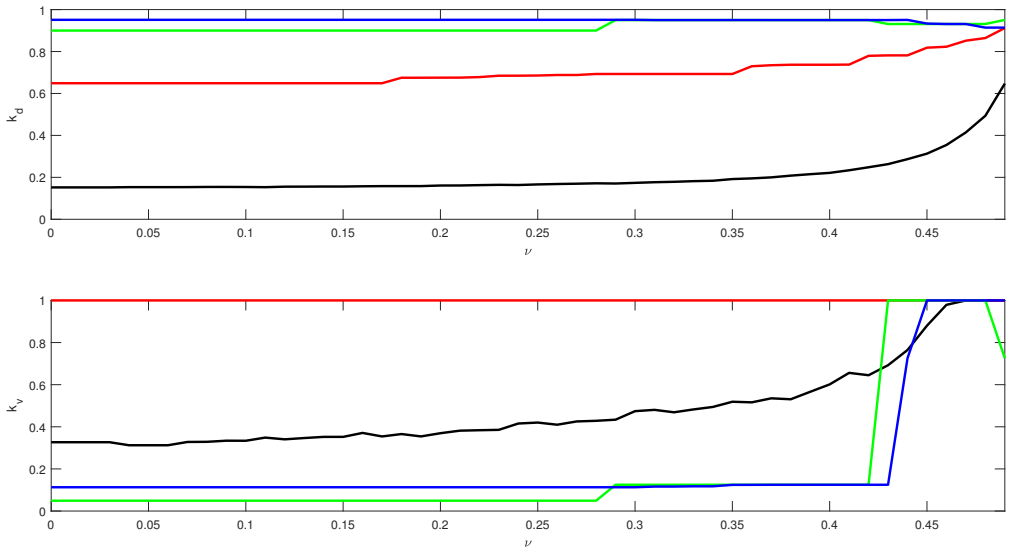

Figure 4.30: Behaviour of the ks on the GravityCeiling experiment when E changes. In black we have the case $E = 10^3$, while red represents $E = 10^4$, green represents $E = 10^5$ and blue represents $E = 10^6$.

4.4 | Gravity

Recall that Gravity is the previous three simple scenes, where we measure the error on all of the scenes for the same parameters, and we take the norm of the three-dimensional vector formed by the respective errors as the total error. This is done in order to reduce the overall error on all three scenes at the same time, not focusing on a concrete scene. Of course, the individual behaviour of each one of the errors of a scene will be also reflected on the overall error.

To further reflect this, when we represent the error for the Gravity experiments, we will draw the proportional part for each of the experiments, with a red fill to represent the GravityFloor experiment, green for the GravityWall experiment, and blue for the GravityCeiling. This fill is not to be confused with the particular cases that will also be represented with red, green, and blue, but without a fill.

Base case

We can see both the error and the optimal ks for the base case on Figure 4.31.

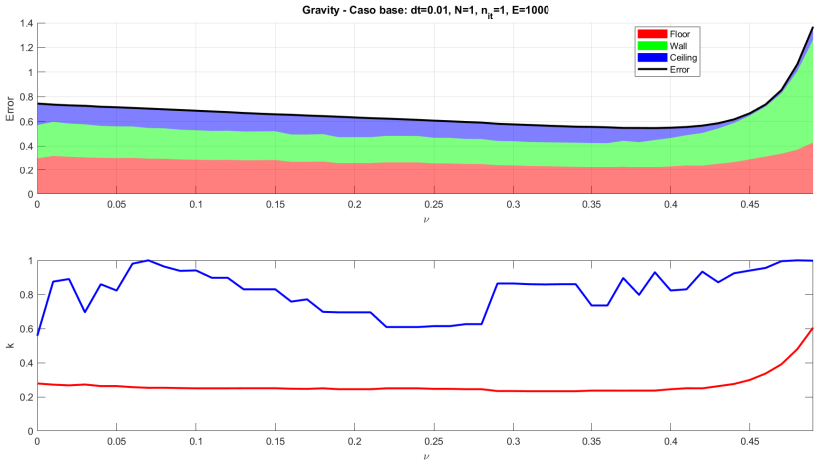

Figure 4.31: Base case for Gravity experiment. The parameters used are $dt = 0.01, N = 1, n_{it} = 1$ and $E = 10^3$. Above we can see the error. Below, we can see in red the optimal k_d and in blue the optimal k_v.

Here we observe a behaviour that we had not expected: the GravityWall experiment had the minor error overall, on almost all cases, especially when ν approached the singularity $\nu = 0.5$. However, in order to minimize the overall error,

it is best to increase the error on the GravityWall experiment when ν increases, being it the one that contributes the most error. This is a big hint to see that the Gravity results are highly dependant on the scenes chosen.

Note also that k_v had erratic behaviour due to precision problems consistently on all three scenes. This is further reflected when using the three scenes at the same time, as we can see. The behaviour of k_d is more consistent with the other experiments.

Variation of dt

We can see the results as dt changes on Figure 4.32 and Figure 4.33.

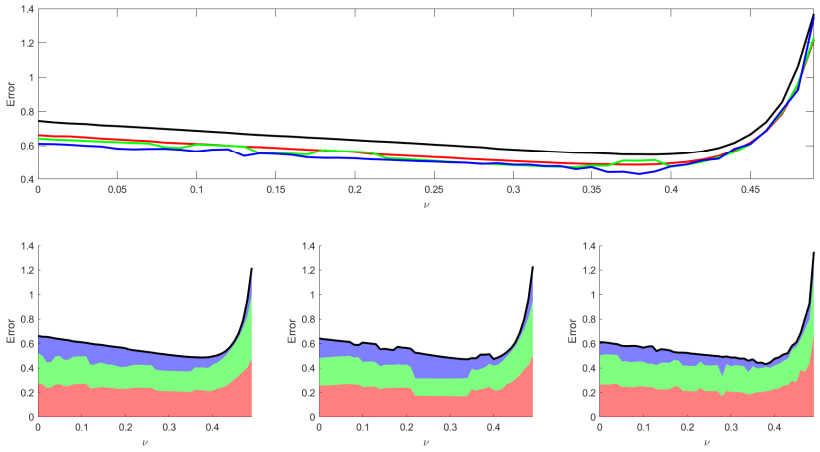

Figure 4.32: Behaviour of error on the Gravity experiment when dt changes. Above, in black we have the case $dt = 0.01$, while red represents $dt = 0.005$, green represents $dt = 0.002$ and blue represents $dt = 0.001$. On the lower part, at the left we have $dt = 0.005$, at the middle we have $dt = 0.002$, and at the right we have $dt = 0.001$

We can see that the overall error decreases when using smaller time steps, as we have seen in GravityWall and GravityCeiling. We can observe that the behaviours are more erratic, as the three previous problems were also erratic. However, the precision problems are minimized (or at least, somewhat softened) when considering the three scenarios at the same time.

We can see that the behaviour of the ks is also more erratic, as we can see on the base case of k_d, as it is not as quasi-constant as on the previous experiments. The k_v also has a less definite form than before, and is also more erratic. However, the behaviour of the ks are similar to what we have observed on the previous scenes.

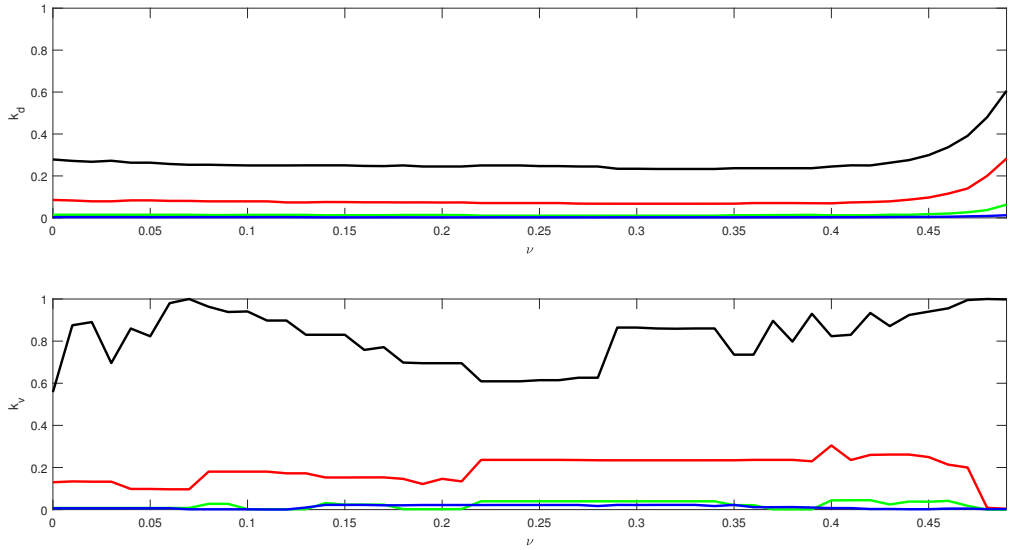

Figure 4.33: Behaviour of the ks on the Gravity experiment when dt changes. In black we have the case $dt = 0.01$ (the base case), while red represents $dt = 0.005$, green represents $dt = 0.002$ and blue represents $dt = 0.001$.

Variation of N

We can see the results as N changes on Figure 4.34 and Figure 4.35.

We can see the same behaviour observed on the previous experiments: a larger N can actually be worse for the error. However, when taking all the scenes into account this effect seems somewhat reduced, as the $N = 4$ case has actually lower error until $\nu \approx 0.37$. That is, although increasing N is not the best solution for each individual scene, it seems that the best overall error is indeed reduced when increasing N. However, we also have to take into account the effect of the precision error, which will grow with N.

If we observe the ks we also can see that $N = 4$ is probably the limit, as in this case $k_d = k_v = 1$, the limit stiffness of PBD.

Variation of n_{it}

We can see the results as n_{it} changes on Figure 4.36 and Figure 4.37.

We can see that the error is almost the same, only reduced on the extreme cases where ν approaches 0.5. The stiffness k_d is also almost the same, with the only real difference being in the less rigid k_v when increasing the number of iterations, fact that we have explained on the previous experiments.

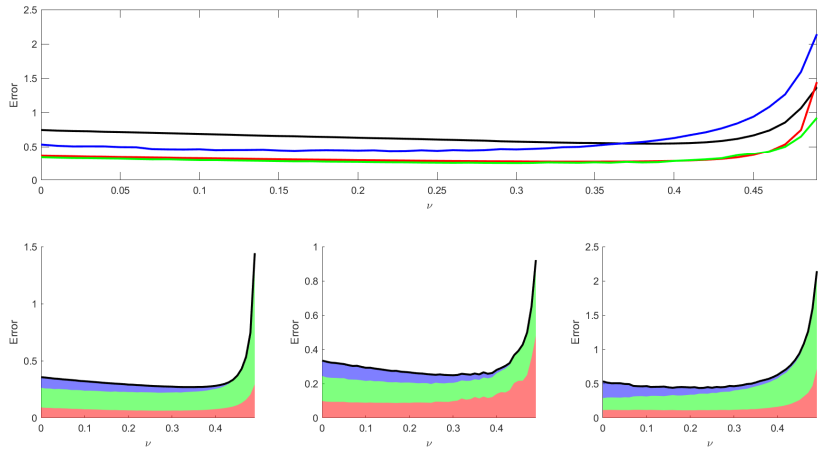

Figure 4.34: Behaviour of the error on the Gravity experiment when N changes. Above, in black we have the case $N = 1$, while red represents $N = 2$, green represents $N = 3$ and blue represents $N = 4$. On the lower part, at the left we have $N = 2$, at the middle we have $N = 3$, and at the right we have $N = 4$

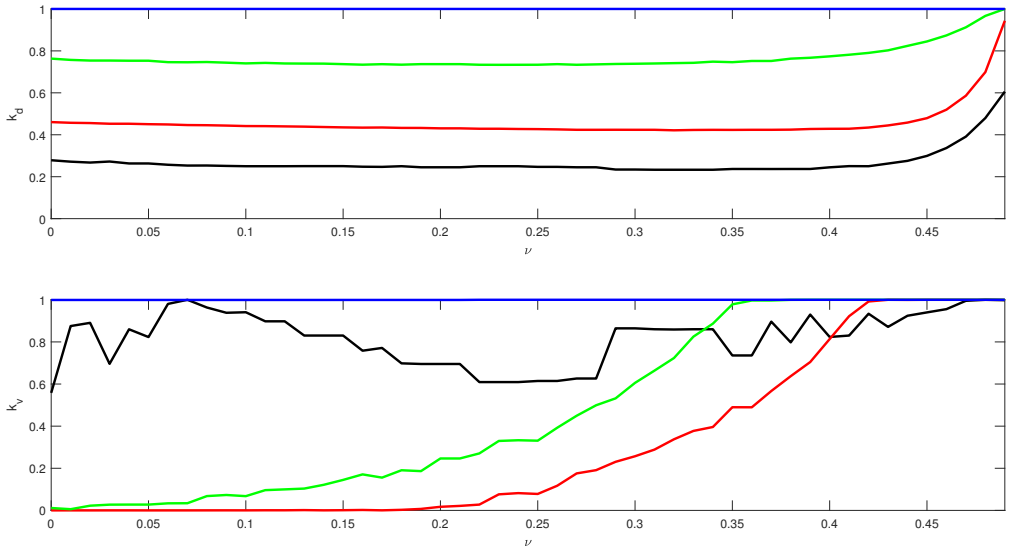

Figure 4.35: Behaviour of the ks on the Gravity experiment when N changes. In black we have the case $N = 1$, while red represents $N = 2$, green represents $N = 3$ and blue represents $N = 4$.

Variation of E

We can see the results as E changes on Figure 4.38 and Figure 4.39.

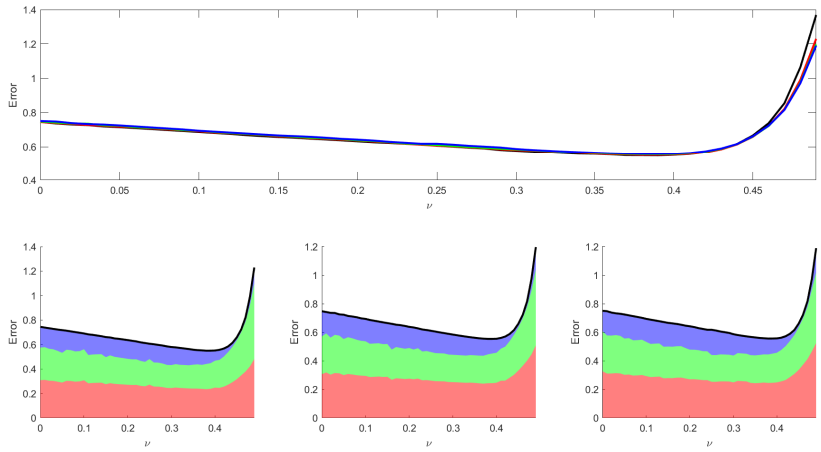

Figure 4.36: Behaviour of the error on the Gravity experiment when n_{it} changes. Above, in black we have the case $n_{it} = 1$, while red represents $n_{it} = 2$, green represents $n_{it} = 4$ and blue represents $n_{it} = 8$. On the lower part, at the left we have $n_{it} = 2$, at the middle we have $n_{it} = 4$, and at the right we have $n_{it} = 8$

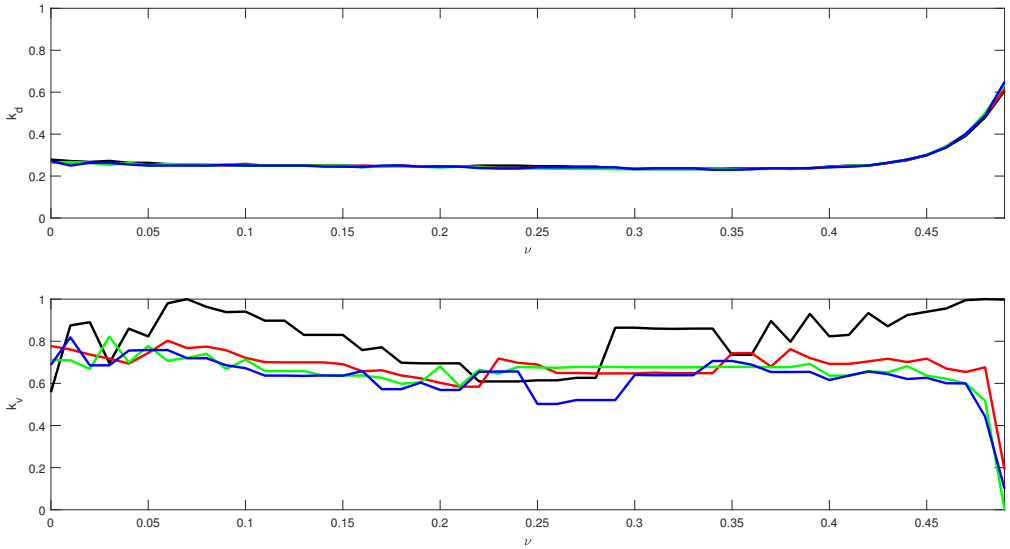

Figure 4.37: Behaviour of the ks on the Gravity experiment when n_{it} changes. In black we have the case $n_{it} = 1$, while red represents $n_{it} = 2$, green represents $n_{it} = 4$ and blue represents $n_{it} = 8$.

Here we can see that the red almost does not appear in the red figures: as we had seen, the GravityFloor is the one less dependant on E. For lower ν, the error is almost all from the GravityWall scene, whereas for greater ν, the error is almost

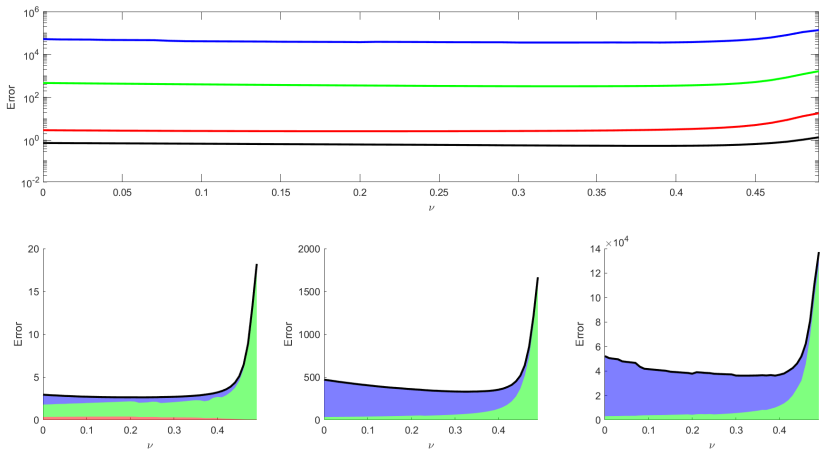

Figure 4.38: Behaviour of the error on the Gravity experiment when E changes. Above, in black we have the case $E = 10^3$, while red represents $E = 10^4$, green represents $E = 10^5$ and blue represents $E = 10^6$. On the lower part, at the left we have $E = 10^4$, at the middle we have $E = 10^5$, and at the right we have $E = 10^6$

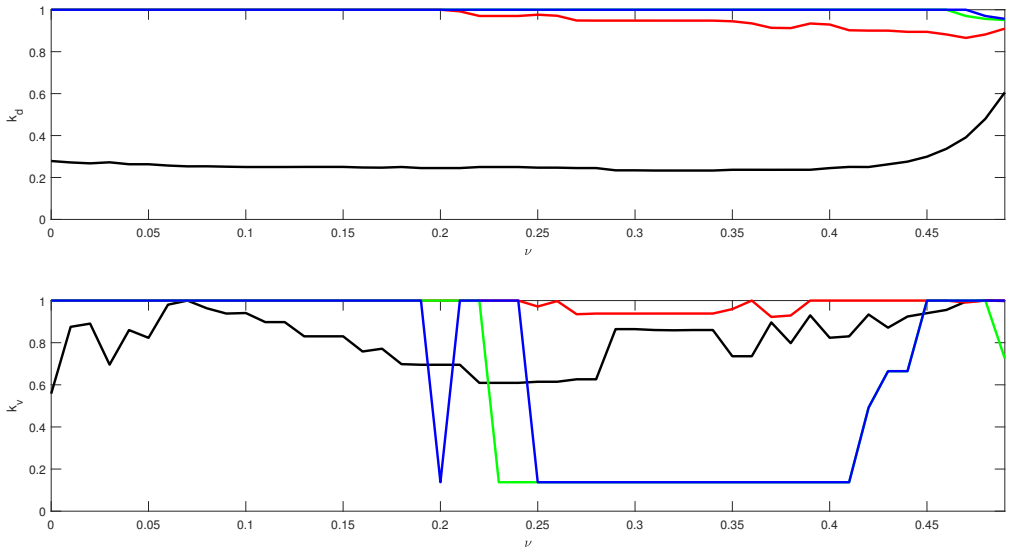

Figure 4.39: Behaviour of the ks on the Gravity experiment when E changes. In black we have the case $E = 10^3$, while red represents $E = 10^4$, green represents $E = 10^5$ and blue represents $E = 10^6$.

all from the GravityCeiling scene. On all cases, a bigger E means a more rigid cube, so the ks grow to 1, as would be expected.

5 | Conclusions and future work

Our objective was to improve PBD by proportioning a set of parameters which could better adapt the physical behaviour of some scenes, and try to adapt different scenes at the same time.

We obtain the best approximations for small E and small ν. We have also studied the improvement (or not) of the error when the main parameters used on the simulations change. Therefore, we can conclude that the PBD can approximate the behaviour of a physically based model, namely the FEM, with a relatively small error. However, we have not been able to replicate the exact behaviour of the FEM, so the PBD will not be physically accurate. We have only been able to make it more visually pleasant and somewhat realistic. A curious effect observed is that, although exaggerating the effects of the parameters does not work well for a specific scene, it is beneficial when fitting overall, as we have found on the Gravity experiments.

Also, we have seen that we cannot find a set of stiffness parameters for the PBD constraints that approximates properly all types of behaviours, but we found that we can emulate with a low error specific types of behaviour.

Possible lines of work are the following:

- Repeat the same work using a more advanced version of PBD, like XPBD.
- Open the possibility of using an unbounded $k > 0$ for the stiffness of the PBD constraints, as often the cause for the lack of improvement of the error is that the PBD has achieved its rigidity internal limit. However, we would have to be careful, as this could result in overshooting the constraints.
- Using other models for elasticity, and not linear elasticity. Likely candidates are neo-hookean models, among other hyperelastic models.
- Using other experiments with usual types of deformation, as shear, traction and torsion. These are the usual experiments performed on elastic materials

by mechanical engineer on their analysis of the properties that characterize a material..

- Using other (possibly irregular) shapes and meshes.
- Use a data-driven model instead of a physically based model, where instead of using simulations (be it FEM, be it another model) as reference model, we would take data from real material with known behaviours and try to approximate the PBD.

Bibliography

[1] Barreiro, H., García-Fernández, I., Alduán, I., and Otaduy, M. A. Conformation constraints for efficient viscoelastic fluid simulation, 2017.

[2] Bathe, K. *Finite Element Procedures.* Prentice Hall, Pearson Educatio, Inc., 2006.

[3] Baudet, V., Beuve, M., Jaillet, F., Shariat, B., and Zara, F. Integrating tensile parameters in mass-spring system for deformable object simulation. In *Int'l Conf. on Computer Graphics, Visalization and Computer Vision (WSCG'09)* (2009), pp. 145–152.

[4] Bender, J., Müller, M., and Macklin, M. Position-based simulation methods in computer graphics. In *Eurographics (Tutorials)* (2015).

[5] Bender, J., Müller, M., and Macklin, M. A survey on position based dynamics, 2017.

[6] Bender, J., Müller, M., Otaduy, M. A., Teschner, M., and Macklin, M. A survey on position-based simulation methods in computer graphics. In *Computer graphics forum* (2014), vol. 33, Wiley Online Library, pp. 228–251.

[7] Berndt, I., Torchelsen, R., and Maciel, A. Efficient surgical cutting with position-based dynamics. *IEEE Computer Graphics and Applications 38*, 3 (2017), 24–31.

[8] Celigüeta, J. *Método de los Elementos Finitos para el Análisis Estructural.* UNICOPIA C.B., 2000.

[9] Conte, S., and De Boor, C. *Elementary Numerical Analysis.* McGraw-Hill Book Company, 1980.

[10] Courant, R. Variational methods for the solution of problems of equilibrium and vibrations. *Bulletin of the American Mathematical Society 49* (1943), 1–23.

[11] Gelder, A. V. Approximate simulation of elastic membranes by triangulated spring meshes. *Journal of Graphics Tools 3*, 2 (1998), 21–41.

[12] Hinton, E., and Irons, B. Least squares smoothing of experimental data using finite elements. *Strain 4* (July 1968), 24–27.

[13] Kikuuwe, R., Tabuchi, H., and Yamamoto, M. An edge-based computationally efficient formulation of saint venant-kirchhoff tetrahedral finite elements. *ACM New York, NY, USA 28*, 1 (2009). doi:10.1145/1477926.1477934.

[14] Liu, T., Bargteil, A., O'Brien, J., and Kavan, L. Fast simulation of mass-spring systems. *ACM Transactions on Graphics (TOG) 32*, 6 (2013).

[15] Lloyd, B., Szekely, G., and Harders, M. Identification of spring parameters for deformable object simulation. *IEEE Transactions on Visualization and Computer Graphics 13*, 5 (2007), 1081–1094.

[16] Maciel, A., Boulic, R., and Thalmann, D. Deformable tissue parameterized by properties of real biological tissue. In *Surgery Simulation and Soft Tissue Modeling* (2003), pp. 74–87.

[17] Macklin, M., Müller, M., and Chentanez, N. Xpbd: Position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games* (2016), ACM, pp. 49–54.

[18] Monteagudo, C., Lozano, M., García-Fernández, I., and Martinez-Gil, F. Phase Space Data-Driven Simulation of Elastic Materials. In *Spanish Computer Graphics Conference (CEIG)* (2016), A. Garcia-Alonso and B. Masia, Eds., The Eurographics Association.

[19] Müller, M., Chentanez, N., Macklin, M., and Jeschke, S. Long range constraints for rigid body simulations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2017), ACM, p. 14.

[20] Müller, M., Heidelberger, B., Hennix, M., and Ratcliff, J. Position based dynamics. In 3^{rd} *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS"* (2006).

[21] Real, P., Rodero, C., and García-Fernández, I. Characterization of the elastic behaviour of some pbd materials. In *4º Congreso de jóvenes investigadores* (2017).

[22] Rodero, C., Real, P., Zuñeda, P., Monteagudo-Mañas, C., Lozano, M., and García-Fernández, I. Characterisation of position based dynamics for elastic materials. In *XXVI Spanish Computer Graphics Conference "CEIG"* (2016).

[23] San Vicente, G. *Designing deformable models of soft tissue for virtual surgery planning and simulation using the Mass-Spring Model.* PhD thesis, School of Engineering, University of Navarra, 2011.

[24] San-Vicente, G., Aguinaga, I., and Celigueta, J. Cubical mass-spring model design based on a tensile deformation test and nonlinear material model. *IEEE Transactions on Visualization and Computer Graphics 18*, 2 (2012), 228–241.

[25] Synge, J. *The hypercircle in mathematical physics.* Cambridge at the University Press, 1957.

[26] Zienkiewicz, O. *The finite element method.* London: McGraw-Hill, 1977.