A PHYSICS SIMULATION TOOL FOR THE CONTAINER LOADING PROBLEM

António G. Ramos[(a)], João Jacob[(b)], Jorge Justo[(c)] , José F. Oliveira[(d)] , Rui Rodrigues[(e)] , A. Miguel Gomes[(f)]

[(a) (b) (d) (e) (f)] INESC-TEC and Faculty of Engineering, University of Porto
[(a) (c)] CIDEM and School of Engineering, Polytechnic of Porto

[(a)]agr@isep.ipp.pt, [(b)]joajac@fe.up.pt, [(c)]jfj@isep.ipp.pt, [(d)]jfo@fe.up.pt, [(e)] rui.rodrigues@fe.up.pt, [(f)]agomes@fe.up.pt

ABSTRACT
In the Container Loading Problem literature, the cargo dynamic stability constraint has been evaluated by the percentage of boxes with insufficient lateral support. This metric has been used as a proxy for the real-world dynamic stability constraint and has conditioned the algorithms developed for this problem. It has the advantage of not being expensive from a computation perspective. However, guaranteeing that at least three sides of a box are in contact with another box or with the container wall does not necessarily ensure stability during transportation. In this paper we propose a physics simulation tool based on a physics engine that will be used in the evaluation of the dynamic stability constraint. We compare the results of our physics simulation tool with the state-of-the-art simulation engineering software Abaqus Unified FEA, and conclude that our tool is a promising alternative.

Keywords: dynamic stability, physics engine, container loading problem

1. INTRODUCTION

The efficient use of transportation resources is of great relevance in the field of logistics, impacting on operational efficiency, customer satisfaction, and transport safety. The Container Loading Problem (CLP) addresses the optimization of the spatial arrangement of cargo inside containers so that the utilization of the space is maximized. The problem belongs to the wider combinatorial optimization class of Cutting and Packing Problems. According to the typology for cutting and packing problems proposed by Wäscher, Haußner, and Schumann (2007), these can be classified according to dimensionality, assortment of large items, assortment of small items, assignment type and shape of small items. In this paper we will focus on three-dimensional rectangular placement problems. The CLP can have two main variants: the maximization of the value of the cargo loaded when the number of containers is not sufficient to accommodate all the cargo, or the minimization of the value of containers when there are sufficient containers to accommodate all the cargo.

In order to be used in real world scenarios a number of constraints found in practice must be considered when addressing the problem. Cargo stability, weight distribution, cargo positioning or cargo orientation constraints are just some examples (Bortfeldt and Wäscher, 2013).

Stability is considered one of the most important CLP constrains and has received a lot of attention by a large number of authors (Bortfeldt and Wäscher, 2013). Existing approaches to stability can be classified in two main groups, one that only addresses static stability and one that addresses static and dynamic stability. Static stability refers to the ability of each box to maintain the loading position during loading operations, and dynamic stability refers to the ability of each box to maintain the loading position during transportation.

Dynamic stability is usually ensured by placing the boxes with their sides adjacent to other boxes or the container walls. The metric used to evaluate dynamic stability is usually the insufficient lateral support, i.e., the percentage of boxes whose sides are not in contact with other boxes or with the container walls (Bortfeldt and Wäscher, 2013). This approach is used as a proxy of the real-world dynamic stability constraint and has been conditioning the algorithms developed for this problem. However, its effectiveness as a dynamic stability constraint can be easily dismissed. In a wall of boxes, as illustrated in Figure 1, boxes can have 3 sides of lateral support, but in case of acceleration along the *x*-axis they would most likely fall.

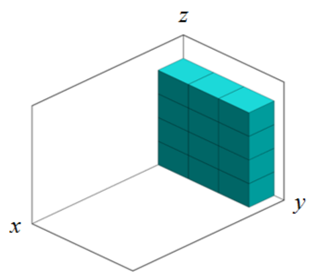


Figure 1: Unstable patterns example

The existing approaches have the benefit of being easy to incorporate in the CLP algorithms without being computationally expensive.

The main objective of this paper is to contribute to narrowing the gap between the real-life dynamic stability constraint and the CLP dynamic stability constraint by developing a physics simulation tool to emulate the interaction between boxes, and between boxes and the container. This approach can be used to validate new dynamic stability indicators applied to solutions generated by CLP algorithms. The results of the developed tool are validated against state-of-the-art simulation engineering software and analytical results.

The remainder of the paper is organized as follows: Section 2 is devoted to reviewing related work. In section 3 the developed physics simulation tool is presented. Section 4 is dedicated to present the test conditions used to compare the two tools and to report computational results. Finally, Section 5 draws some conclusions from the findings and proposes future work.

2. RELATED WORK

A physics engine is a computer software designed to simulate various physical phenomena such as rigid body dynamics, soft body dynamics or fluid dynamics. It manages the forces applied to objects and the interactions between objects by simulating Newtonian physics (Jones, 2011; Seugling and Rolin, 2006). According to Erleben (2002) a physics engine has two main components, collision detection and dynamic simulation. Each one consists of a set of four interacting modules (see Figure 2).

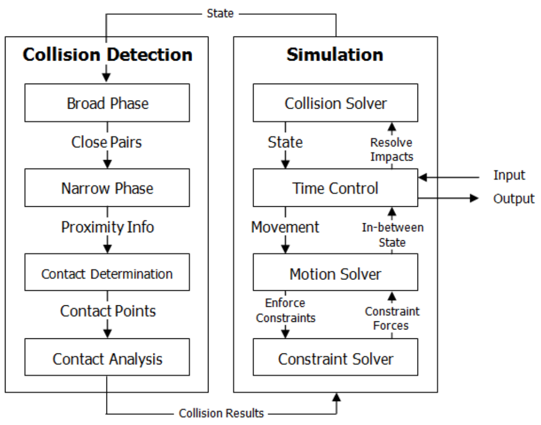


Figure 2: General Purpose Module design (Erleben, 2002)

Their performance is influenced by six essential factors: the simulator paradigm, the integrator, the object representation, the collision detection and contact determination, the material properties and the constraint implementation (Boeing and Bräunl, 2007). These factors are usually developed to address a specific application (Boeing and Bräunl, 2007).

The evaluation or validation of physics engines was addressed by various authors. Seugling and Rolin (2006) and Boeing and Bräunl (2007) evaluate physics engines in a general way, without focus on a particular application, while Hummel et al. (2012) evaluation focused on an interactive application for on-orbit servicing tasks. Pepper et al. (2007) focused on determining and increasing simulation accuracy in urban search and rescue (USAR) robot simulation. The physics engines evaluated or validated in each paper are presented in Table 1.

Table 1: Physics engines evaluated or validated in literature

	(Seugling and Rolin, 2006)	(Boeing and Bräunl, 2007)	(Pepper et al., 2007)	(Hummel et al., 2012)
Open Dynamics Engine	x	x		x
PhysX	x	x		x
Newton Game Dynamics	x	x		x
Tokamak		x		
True Axis		x		
Bullet Physics		x		x
JigLib		x		
Unreal Engine2.0			x	
Havok Physics				x

The evaluation or validation of physics engines was carried out by performing and measuring a set of tests. Seugling and Rolin (2006) developed nine tests intended to evaluate three features: energy preservation, constraint handling and collision detection. Boeing and Bräunl (2007) tested the integrator performance, the material properties, the constraint stability, the collision system and the object stacking. Hummel et al. (2012) focused on collision detection, accuracy of collision, constraint stability and collision and friction of complex geometric objects. Pepper et al. (2007) used a set of tests from the National Institute of Standards and Technology (NIST) standard test methods for USAR robots to compare reality and virtual simulation.

From the performed benchmark tests, Seugling and Rolin (2006) reported that Newton Game Dynamics had the best overall results, Boeing and Bräunl (2007) reported that Bullet Physics had a best overall performance and Hummel et al. (2012) consider that Newton Game Dynamics and PhysX can compete with Bullet Physics.

3. STABLECARGO SIMULATION TOOL

With the goal of further analysing dynamic stability in the CLP, a tool, designated StableCargo, was developed. It consists of a simulator of the physical behaviour of the cargo in a container when different accelerations are applied to the container, much like those that it sustains in real life situations. This tool is based on the CGFLib (a library for computer graphics based on OpenGL, http://paginas.fe.up.pt/~ruirodrig-/pub/sw/cgflib/docs/index.html) and the Bullet physics engine. Both these libraries were chosen as they are cross platform and can run on most hardware. The Bullet physics engine was chosen instead of PhysX engine due to its support of OpenCL that allows improvement in computational speed in parallel architectures (CPU or GPU based).

The results of this tool are meant to improve the development of spatial optimization algorithms that are responsible for creating container layouts. This way, it is possible to test if a given container layout is stable under a given scenario (i.e., a given set of accelerations) or not. Figure 3 presents the tool workflow.

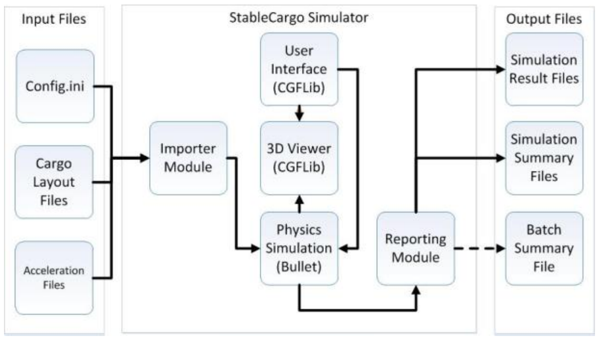


Figure 3: StableCargo's workflow

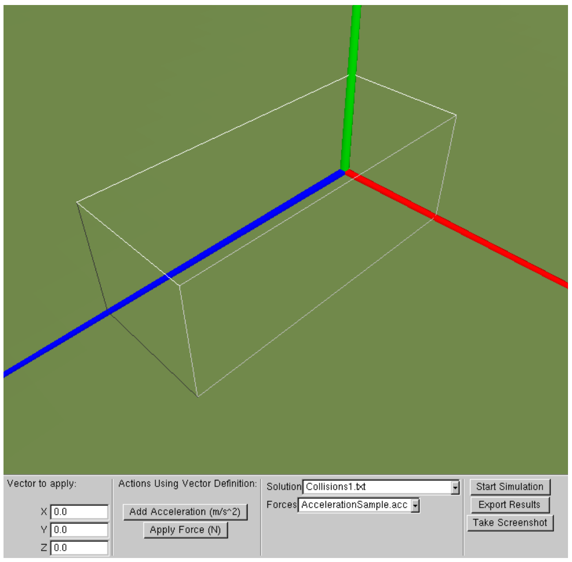
The overall solution consists of three major components, Input files, Output files and the StableCargo Simulator itself.

3.1. The StableCargo Simulator

The StableCargo Simulator was developed in C++. It makes use of the CGFLib, a Computer Graphics library, for 3D rendering and GLUI, for creating a Graphics User Interface to be accessed by the user in order to change simulation parameters in real time.

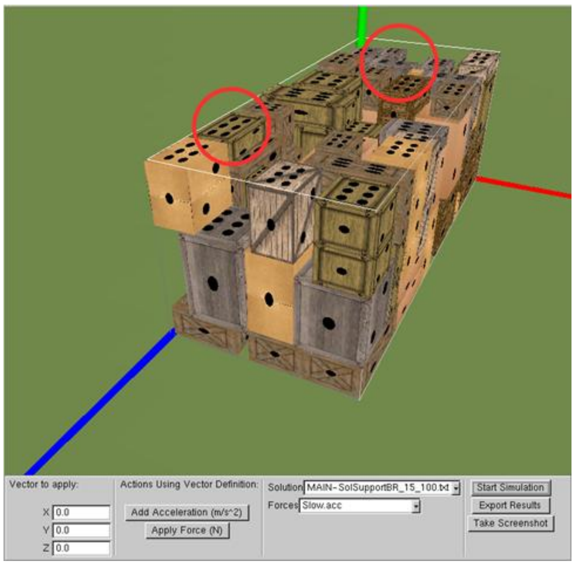
As Figure 4 depicts, the StableCargo Simulator tool allows for some real-time user interaction. Most notably, it allows the user to select what pair of Layout ("Solution") and Acceleration ("Forces") files are to be used at any given time (after being ran through the Importer Module), as well as applying forces or accelerations in real-time. It also allows to export the current results on demand via the "Export Results" button (if the user is interested in only analysing the movement of the container up to a certain moment) and to take a screenshot of the current 3D view of the container ("Take Screenshots" button). The mouse can be used to change the position and rotation of the camera in relation to the container. This way, when the container starts moving, the camera will keep the container framed at the angle defined by the user. The container is drawn using the OpenGL wireframe drawing mode, so as to keep the interior visible at all times. As a physical entity, the container consists of six rigid bodies comprising a compound entity that represents a hollow parallelogram with the dimensions of a standard 20 feet container with a mass of 3700 Kg and friction as specified in the configuration file. In Bullet, friction is declared per physical entity. When two objects are colliding, the friction force is obtained by multiplying the friction coefficients of both objects. This translates to a model compatible with the Newtonian model of a friction coefficient between pairs of objects. Parameterization of the size and mass of a container were considered unnecessary for the scope of this project, as the CPL problem currently under study considers only this type of container. The Reporting Module generates internal statistics concerning each box and each simulation (such as the number of fallen boxes or the maximum kinetic energy of each box), and outputs those statistics to the respective output files and directories.

After selecting "Start Simulation", the application will load the Layout and Acceleration files selected and create each box as a single rigid body (undeformable physical entity) with mass or density as specified by the configuration file (config.ini), with the coordinates of the centre of mass being at the centroid of the respective box. As Figure 5 shows, the visual aspect of the boxes varies, depending on the box type and orientation. Each box type is represented by a unique box image (texture), and the faces are numbered by dots (similar to dice) to show the orientation of the box. This is achieved by blending box textures and markings (dots) using an



X 0.0
Y 0.0
Z 0.0
Solution Collisions1.txt
Forces AccelerationSample.acc

Figure 4: The StableCargo Simulator Tool

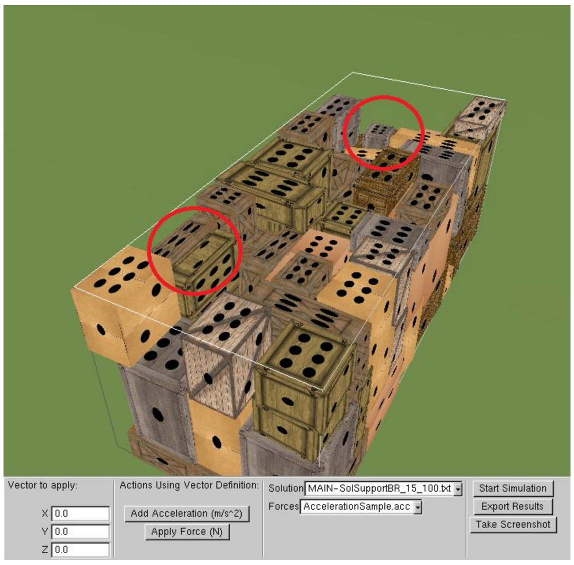


X 0.0
Y 0.0
Z 0.0
Solution MAIN-SolSupportBR_15_100.txt
Forces Slow.acc

Figure 5: Representation of a loaded Layout File

OpenGL Shading Language shader.

The final placement of the boxes is shown after applying the chosen *Acceleration File* (Figure 6). Of particular interest are the boxes highlighted by the red circles, which have moved and fallen. This is easily visible when directly comparing Figure 5 with 6, but when using the tool, it is possible to see the movement of the boxes in real time, or by checking the "black dots" in the face of each box.



X 0.0
Y 0.0
Z 0.0
Solution MAIN-SolSupportBR_15_100.txt
Forces AccelerationSample.acc

Figure 6: Final Layout after stabilization

3.2. Input Files

The input files consist of the *Configuration* (config.ini*)*, *Container Layout* and *Acceleration* files.

The *config.ini* file details several simulation configuration settings that must be set prior to running the tool itself. Parameters consist in:

- [*Physics Engine*]
  - *DensityValue* - represents the density of the boxes in Kg/m^3.
  - *ConstantDensity* – defines whether the *DensityValue* represents the density or the mass of each box.
  - *SimulationStepsPerSecond* – sets how often the simulation is updated.
- [*Renderer Engine*]
  - *DrawAxis* – defines if the Axes are to be drawn by the renderer.
  - *TexturePack* – holds the path of the texture pack, to be used for skinning the boxes with their respective material.
  - *UseRenderer* – used to enable or disable the graphical visualization. It can be disabled to perform batch simulations, without need of visual feedback or interaction.
- [*Export Settings*]
  - *ResultsIntervalInSeconds* – Interval of time, in seconds, for the simulator to sample the statistics of each box.
  - *SavePath* – where the results will be saved.
- [*Simulation Settings*]
  - *StoppingCondition* – defines when the simulation will end. Either by "timeout" (x seconds have passed since the forces have been applied) or by "sleeping" (no box has moved or rotated significantly).
  - *TimeoutInSeconds* – the value in seconds to be used if timeout is the chosen stopping condition.
  - *SleepingThreshold* – the movement threshold to be used in order to consider that the simulation has ended.
  - *BatchSimulation* – specifies if the simulation is a batch simulation, meaning if there will be an attempt to pair all acceleration files and layout files in a folder. This allows for multiple simulations to be done without human intervention.
  - *BatchPath* – the path of the folder containing all acceleration and layout files.
  - *DropThreshold* – how much (in meters) must an object shift its position so that it can be considered to have fallen.
  - *ContainerFriction* – the container friction.
  - *GroundFriction* – the ground friction.
  - *BoxFriction* – the friction of the boxes.

The *Container Layout* file (Figure 7) represents a possible loading scenario of boxes inside a container. It consists of a text file with a one-line header, and multi-line body. Each line of the body represents the position of a box (through the 3D coordinates, in centimetres, of two diagonally opposing corners) and the type of material of that box (identifier of material).

```
26281964     18642852
0    0    0     80      59    106      1
80   0    0     160     59    106      1
80   0    106   160     59    212      1
```

Figure 7: Example of a Layout file

The *Acceleration file* describes the accelerations the container experiences during a period of time. Each line specifies an initial time (in seconds), duration (in seconds) and 3D acceleration vector in m/s^2 (Figure 8).

```
#Initial-Time / Duration / X / Y / Z
0        1        0        0        0
1        0.3      1        0        0
1.3      12       2        0        0
13.3     0.35     2        0        0
```

Figure 8: Example of an Acceleration file

3.3. Output Files

There are three output files: *Solution, Abridged Solution* and *Batch Summary* file.

Solution files, are automatically named with the template [SOLUTION]$LayoutFile$AccelerationFile. The Solution file consists of raw data for each box with multiple readings per box (as specified by the configuration file), extracted from the Bullet physics engine. It contains the following data:

- *Id* – Identifier of the box. It matches the line of the layout file in which the box was declared.
- *Centre of mass position (X,Y,Z) displacement* – displays the container-relative box displacement, e.g. , the difference between the current 3D position and the initial position of the centre of mass, in relation to the position of the centre of mass of the container, in meters.
- *Total Force (X,Y,Z)* – details the vector of external forces (other than collision, gravity and friction) that might have been applied to the box, in Newton.
- *Angular Velocity (X,Y,Z)* – it is the 3D vector that contains the angular velocity of the box in radians per second.
- *Linear Acceleration (X,Y,Z)* – shows the current linear acceleration the box has in relation to the movement of the container, in m/s^2.
- *Elapsed time (s)* – represents each sampling interval time.

Resumed Solution files, are automatically named as [SOLUTION_RESUMED]$LayoutFile$AccelerationFile and contain information extracted from the Solution File data:

- *Number of fallen boxes* – by comparing the vertical displacement each box suffered during the simulation with the configuration file DropThreshold it is possible to estimate the number of fallen boxes.
- *Centre of Mass Displacement* for each box.
- *Kinetic Energy* of each box, in relation to the container.
- *Total Kinetic Energy* – the sum of the kinetic energies of all the boxes.

Batch Summary files, named as [BATCH-RESUMED]$BatchDirectory are only created if the application is set to run in batch mode. They contain:

- *Layout File Name*.
- *Acceleration File Name*.
- *Number of Fallen Boxes*.
- *Total Kinetic Energy*.

4. BENCHMARK TESTS

The purpose of our physics model is to simulate the movement of a set of boxes inside a shipping container, subject to a set of external forces in typical extreme cases such as vehicle full braking, cornering or lane changing.

As friction is a parameter we considered to be one of the most relevant to evaluate the physics engine performance, a set of benchmark tests involving friction as the main parameter was performed. Numerical results obtained using our physics simulation tool and a state-of-the-art engineering simulation software (Abaqus Unified FEA) were then compared with the analytical ones, enabling an assessment of the software packages ability in modelling the friction phenomena.

Abaqus FEA is a software suite for finite element analysis and computer-aided engineering. This software suite consists of five core software products:

- Abaqus/CAE, a software application used for pre-processing and visualizing the finite element analysis result.
- Abaqus/Standard, a general-purpose Finite-Element analyzer that employs implicit integration scheme.
- Abaqus/Explicit, a special-purpose Finite-Element analyzer that employs explicit integration scheme to solve highly nonlinear systems with many complex contacts under transient loads.
- Abaqus/CFD, a Computational Fluid Dynamics software application.
- Abaqus/Electromagnetic, a software application which solves advanced computational electro-magnetic problems.

This product suite is used in academic work as well as in industrial research projects, namely in the aerospace and automotive industry. In the automotive industry engineering, it can be used to analyse sophisticated nonlinear engineering problems, such as impact/crash events, multibody systems, full vehicle loads and dynamic vibration.

For performing the benchmark tests with Abaqus FEA, rigid elements were selected for the boxes as well as for the container floor. Abaqus/Explicit was selected to perform the analysis, with a fixed time increment value of 0.1 ms. The possibility of contact between all surfaces was considered. Displacement, velocity and acceleration values for the centre of gravity of the box were recorded during all the simulations, which made post-processing the results quite simple.

4.1. Friction equations

Friction can be defined as the phenomenon of resistance of a body on another which delays or prevents relative movement between them. The force that expresses this resistance always acts tangent to the contact surface. Being a force between two bodies, it naturally conforms to the principle of action/reaction. The direction of the reaction force on a body that tends to move in a given direction is always opposite to that direction. In general there are two types of friction: the fluid friction where surfaces are interleaved by a fluid layer (e.g. an oil), and the dry friction, where the two bodies are in direct contact.

In dry friction, if a force F acts on a block of weight W that is at rest, it generates reaction forces distributed along the contact surface between the two bodies. These forces have tangential or friction components T, and normal components N (see Figure 9).

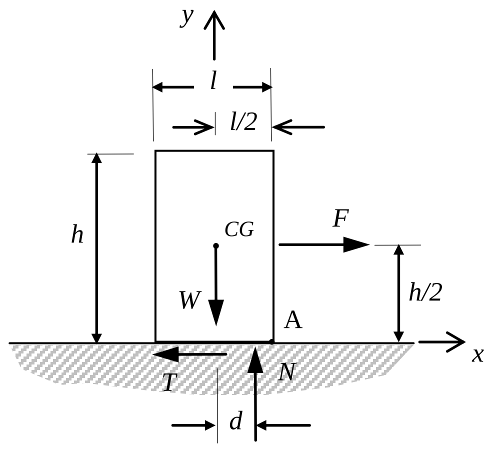


Figure 9: Forces acting on a box

Friction force T is independent of the surface area of contact, but depends directly on the resultant normal force N. The coefficient of static friction μ_s between two surfaces in contact is determined experimentally. The body is considered to be in the imminence of sliding, if condition (1) is met and in the imminence of rolling about A, if condition (2) is met.

$$\begin{cases} T = \mu_s N \\ d < \dfrac{l}{2} \end{cases} \tag{1}$$

$$\begin{cases} T < \mu_s N \\ d = \dfrac{l}{2} \end{cases} \tag{2}$$

To test static friction, two tests are performed. The first has the purpose of testing condition (1), i.e., the imminence of body sliding and the latter has the purpose of testing condition (2), i.e., the imminence of body rolling.

4.2. Sliding Test

In order to test the sliding of a body, one box with dimensions 25 cm × 110 cm × 55 cm was placed in a horizontal plane. A coefficient of static friction between the body and the plane was assigned and the force applied to the box parallel to the plane was incremented until the box started sliding. The acceleration in imminence of (1) in a horizontal plane is equal to (3).

$$a = \mu_s g \tag{3}$$

Figure 10 shows the values of the acceleration in the imminence of sliding for the range of coefficients of static friction 0.1 to 0.8. The analytically calculated value is also represented. Both Abaqus and the StableCargo tool provided results with a high approximation to analytical values.

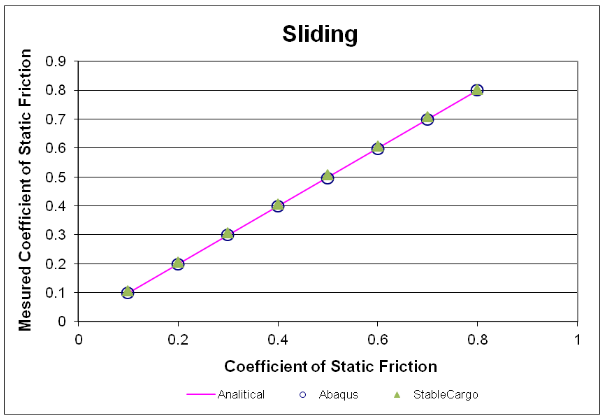


Figure 10: Measured coefficient of static friction

4.3. Body Rolling Test

To analyse the imminence of body rolling, another test was developed. A coefficient of static friction between the body and the plane was assigned and a force, parallel to the plane, was applied to the centre of gravity of the box. This force was incremented until the box started to move. If sliding occurred, then the coefficient of static friction was incremented and the test repeated. When rolling of the body occurred, the coefficient of static friction used was considered the measured coefficient of static friction.

To guarantee that there is no sliding prior to rolling, it can be shown that the height and length ratio of the box must satisfy (4).

$$\frac{l}{h} > \mu_s \tag{4}$$

Figure 11 shows the values of the measured coefficient of static friction in the imminence of rolling for different values of the height and length ratio, ranging from 0.2 to 0.8.

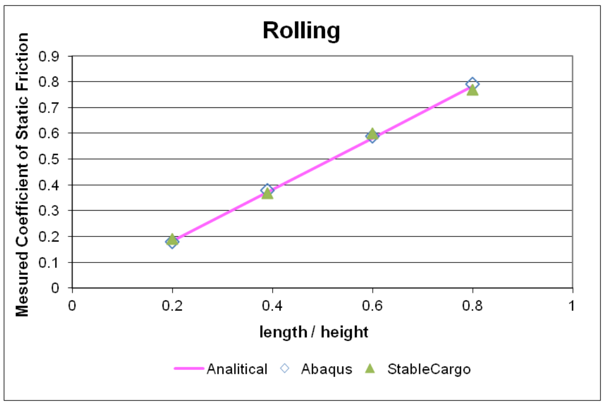


Figure 11: Measured coefficient of static friction for rolling

The analytical values are also represented. Results, obtained with Abaqus and the StableCargo tool, are in very good agreement with the analytical values.

5. CONCLUSIONS AND FUTURE WORK

Friction can be considered one of most relevant parameters when analysing a physics engine. To evaluate our physics simulation tool regarding its ability to model friction, a set of benchmark tests was performed, and the results obtained were compared with analytical values and those obtained using the state-of-the-art engineering simulation software Abaqus FEA.

The results obtained are in agreement with both the analytical values and those obtained using Abaqus FEA, offering good prospects for the use of the tool for evaluating dynamic stability within the CLP.

Future work should concentrate on evaluating our physics simulation tool when modelling other events, like collision and rebound that may occur in a shipping container subject to typical extreme situations, such as vehicle full braking, cornering and lane changing. A set of benchmark tests involving these phenomena should be performed and the results compared to those obtained experimentally and with other simulation engineering software, like Abaqus FEA.

ACKNOWLEDGMENTS

This research was partially supported by ERDF through the Programme COMPETE, by the Portuguese Government through FCT (StableCargo - PTDC/SEN-TRA/121715/2010) and by the Project BEST CASE - SAESCTN-PIIC\&DT/1/2011 which is co-financed by the North Portugal Regional Operational Programme (ON.2 - O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF).

REFERENCES

Boeing, A., & Bräunl, T. (2007). Evaluation of real-time physics simulation systems. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia - GRAPHITE '07* (Vol. 1, p. 281). New York, New York, USA: ACM Press. doi:10.1145/1321261.1321312

Bortfeldt, A., & Wäscher, G. (2013). Constraints in container loading – A state-of-the-art review. *European Journal of Operational Research*, *229*(1), 1–20. doi:10.1016/j.ejor.2012.12.006

Erleben, K. (2002). *Module based design for rigid body simulators* (Vol. 1). Retrieved from http://www.cs.umu.se/kurser/TDBD24/VT05/articles/ErlebenModular2002.pdf

Hummel, J., Wolff, R., & Stein, T. (2012). An evaluation of open source physics engines for use in virtual reality assembly simulations. *Advances in Visual …*, 346–357. doi:10.1007/978-3-642-33191-6_34

Jones, M. T. (2011). *Open source physics engines Building believable worlds with open source* (pp. 1–10).

Pepper, C., Balakirsky, S., & Scrapper, C. (2007). Robot simulation physics validation. In *Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems - PerMIS '07* (pp. 97–104). New York, New York, USA: ACM Press. doi:10.1145/1660877.1660890

Seugling, A., & Rolin, M. (2006). *Evaluation of physics engines and implementation of a physics module in a 3d-authoring tool*. *Umea University*. Umea University. Retrieved from http://www8.cs.umu.se/education/examina/Rapporter/SeuglingRolin.pdf

Wäscher, G., Haußner, H., & Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, *183*(3), 1109–1130. doi:10.1016/j.ejor.2005.12.047

AUTHORS BIOGRAPHY

António Galrão Ramos graduated in Mechanical Engineering from the Faculty of Engineering, University of Porto, Portugal in 1997 and the M.Sc. degree in Logistics by the Porto Business School, University of Porto, Portugal in 2009. He is an Associate Professor with the Department of Mechanical Engineering (DEM), School of Engineering, Polytechnic of Porto (ISEP), where he has been since 2001. He worked in several multinational companies in Project Management, Operations and Logistics Management. He has been the Director of the Laboratory for Manufacturing Systems (DEM/ISEP) since 2010 and a researcher at the Institute for Systems and Computer Engineering of Porto (INES TEC) and at the Center for Research and Development in Mechanical Engineering (CIDEM/ISEP). He participated/participates in national and international research projects.

João Jacob is an Invited Assistant Professor at the Department of Informatics Engineering of the Faculty of Engineering of the University of Porto and also a PhD student focusing on Serious Games, Mobile Computing, Augmented Reality and Location-Based Games. Additionally he is also a researcher at INESC TEC currently working on a project in the area of physics simulation for the container loading problem.

Jorge Fonseca Justo is a graduate (1992) in Mechanical Engineering, with a M.Sc. degree in Mechanical Engineering, specialization in Structural Engineering, and a Ph.D. degree in Mechanical Engineering from the Faculty of Engineering, University of Porto. He is an Associate Professor with the Department of Mechanical Engineering, School of Engineering, Polytechnic of Porto (ISEP), where he has developed his activity since 1994. He has worked in several industrial projects in the Institute of Mechanical Engineering and Industrial

Management (INEGI) in the areas of structural simulation and machine projects. He is a researcher as well as a member of the managing board at the Centre for Research and Development in Mechanical Engineering (CIDEM). He has guided several master's theses, and has been involved in several research projects, both national and international, involving several companies, which resulted in several publications/communications and a patent.

José Fernando Oliveira is Full Professor at the Department of Industrial Engineering and Management of the Faculty of Engineering of the University of Porto and collaborates with the Business School of the University of Porto. His primary research interests are decision and optimization problems, in particular problems related to the efficient use of raw-materials and other resources (cutting and packing problems) and to decision support systems in industry and services. He regularly publishes the results of his research in the main operations research and management science international scientific journals and keeps a constant activity in consultancy with public and private companies with dozens of successful projects completed. During his more than 25 years long academic career he has mainly taught courses on Statistics, Operations Research and Operations Management and Logistics. He has served as Dean of Studies of The Faculty of Engineering, as main editor of the Portuguese Operational Research scientific journal and is now Vice-President of the Association of European Operational Research Societies. He is member of the General Council of the University of Porto.

Rui Rodrigues graduated in Systems and Informatics Engineering at Minho University in 1998. During his PhD he researched in the area of 3D reconstruction from Images divided between Philips Research, Eindhoven, and Minho University, until he concluded in 2006. He worked in the industry in the field of interactive systems, until he joined FEUP in 2009, where he works currently as Assistant Professor, teaching and researching in the areas of Computer Graphics, Interaction and Gaming. He is also a collaborating researcher at INESC TEC/INESC Porto.

António Miguel Gomes is an Assistant Professor at the Department of Industrial Engineering and Management of the Faculty of Engineering of the University of Porto and researcher at INESC-TEC. His main research areas are Cutting and Packing and Computational Geometry. Currently he is one of the ESICUP (EURO Special Interest Group on Cutting and Packing) coordinators.