Physics simulator

Gabriel Vallat

September 2023

1 Introduction

This document summarizes the important aspects of the physical simulator used during the training of the algorithm.

2 Physics

The complete simulator has to handle two tasks: the collision detection and the static equilibrium calculation. The task of collision detection is to make sure that no blocks (or grounds) are intersecting one another, and can be trivially solved using the grid. On the other hand, the static equilibrium computation is more subtle, so this section of the thesis describe precisely how it is computed.

A structure is in static equilibrium when it is not accelerating and is in a state of balance. According to Newton's first law, this is the case when the net forces and moments are zero.

The simulator only considers the statics of the structure, and although most physics engines (such as PhysX or pybullet) and optimization libraries (Gekko, Gurobi, ...) are able to handle this easily, these programs are designed for more complex tasks, such as multi-body dynamics, and the time required to send messages describing the structure to the external solver often exceeds the time required to actually calculate the stability of the structure. Therefore, it was decided to calculate the physical equilibrium directly using linear programming.

2.0.1 Static equilibrium

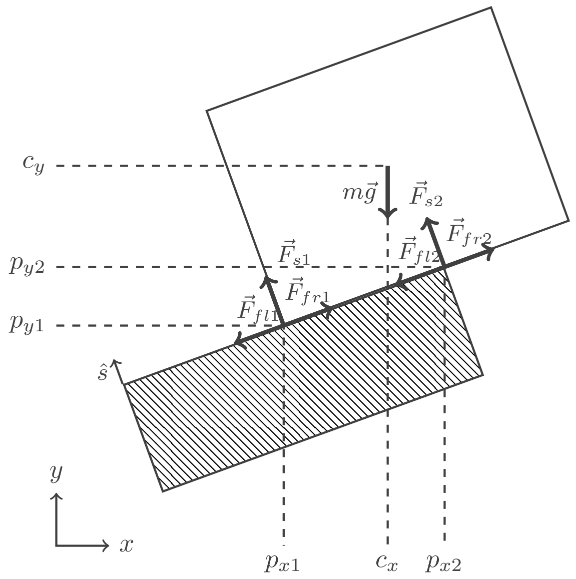


Figure 1: A single block on the ground: in this simplest case, 6 distinct forces are needed to compensate the weight of the block

As mentioned earlier, the static equilibrium conditions require that the sums of the forces and moments acting on each block equal 0. When the block has only one interface oriented with a direction $\hat{s}$, as in the Figure 1, the equilibrium can be written as $A_{eq}^{(s)}\mathbf{x}^{(s)} = \mathbf{b}_{eq}^{(s)}$, where $\mathbf{x}$ is the amplitude of all the reaction forces, and where the first two rows of $A_{eq}^{(s)}$ are respectively tasked to calculate the sum of the reaction forces along the x- and y-directions, and the third (and last) row is tasked to calculate the sum of the moments. This last step justifies the choice of two separate sets of forces applied at each end of the interface: by combining them linearly, the resulting force can be applied at any point along the interface without the need for a nonlinear operation at any time. Finally, the vector $\mathbf{b}_{eq}^{block}$ is simply composed of the weight of the block and its induced moment:

$$A_{eq}^{(s)\mathrm{T}} = \begin{pmatrix} s_x & s_y & p_{y1}s_{xi} + p_{x1}s_{yi} \\ -s_y & s_x & p_{y1}s_{yi} + p_{x1}s_{xi} \\ s_y & -s_x & -p_{y1}s_{yi} - p_{x1}s_{xi} \\ s_x & s_y & p_{y2}s_{xi} + p_{x2}s_{yi} \\ -s_y & s_x & p_{y2}s_{yi} + p_{x2}s_{xi} \\ s_y & -s_x & -p_{y2}s_{yi} - p_{x2}s_{xi} \end{pmatrix} \mathbf{x}^{(s)} = \begin{pmatrix} F_{s1} \\ F_{fl1} \\ F_{fr1} \\ F_{s2} \\ F_{fl2} \\ F_{fr2} \end{pmatrix} \quad \mathbf{b}_{eq}^{block} = \begin{pmatrix} 0 \\ mg \\ c_x mg \end{pmatrix},$$

where the component of the matrix A_{eq}^{s} are the x and y component of the vector normal to the surface $\hat{s}$ and the coordinate of the application point of each force (as seen on Figure 1).

2.0.2 Compression constraints

An important factor in our setup is that no mortar is used between the blocks, requiring that all of the contact forces are compressive. Doing so creates a non-negativity constraints on the support forces:

$$F_{s1} \geq 0,\ F_{s2} \geq 0$$

To simplify the calculation, the frictional forces were also constrained to be non-negative, but it can be seen that the combination of the two forces acting in opposite directions already allows them to bypass this constraint. One can then use the formulation

$$\mathbf{x}^{(s)} \geq \mathbf{0}$$

2.0.3 Friction constraints

In solid physics, the maximum friction force that can be applied is equal to a coefficient μ_c times the support force (in the static case), i.e $F_{fr1} + F_{fl1} \leq \mu_c F_{s1}$ and $F_{fr2} + F_{fl2} \leq \mu_c F_{s2}$. This constraint can then be written in matrix form, $A^{inter}\mathbf{x} \leq 0$, by using

$$A^{(s)} = \begin{pmatrix} -\mu_c & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\mu_c & 1 & 1 \end{pmatrix}.$$

This coefficient depends on the material of both the block and the ground, as well as other factors such as the roughness of the surface. Its value in real building is generally around 0.7, but with the prototype blocks used in the appendix **??** to validate the model, a value of 0.5 better matched the reality.

2.0.4 Combination of several interfaces

In a real construction, each block is generally not in contact with only one other, but with several (as in Figure 2)

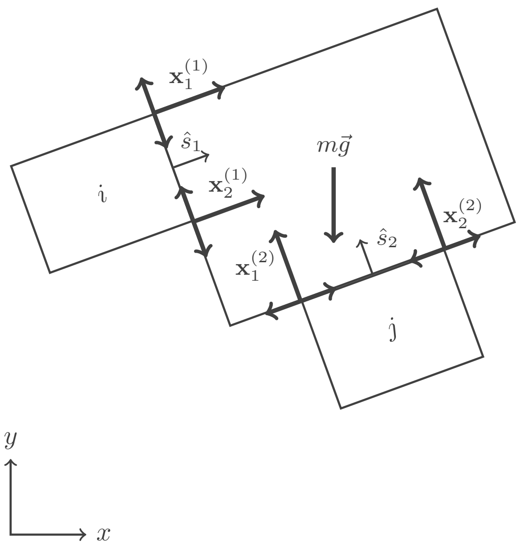


Figure 2: A block added to an existing structure. It shares its interface (1) with the block i and its interface (2) with the block j

In this case, the matrices defined above can simply be stacked together, creating the system of equations and inequation

$$\begin{aligned} A_{eq}^{block}\mathbf{x}^{block} &= \mathbf{b}^{block} \\ A^{block}\mathbf{x}^{block} &\leq \mathbf{0} \\ \mathbf{x}^{block} &\geq \mathbf{0} \end{aligned}$$

If the block has n_s different interfaces, the composition of each matrix and vector is:

$$\begin{aligned} A_{eq}^{block} &= \begin{pmatrix} A_{eq}^{(s_1)} & A_{eq}^{(s_2)} & ... & A_{eq}^{(s_{n_s})} \end{pmatrix} \\ A^{block} &= \begin{pmatrix} A^{(s_1)} & \mathbb{0} & \cdots & \mathbb{0} \\ \mathbb{0} & A^{(s_2)} & \cdots & \mathbb{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{0} & \mathbb{0} & \cdots & F^{(s_{n_s})} \end{pmatrix} \\ \mathbf{x}^{block} &= \begin{pmatrix} \mathbf{x}^{(s_1)} \\ \mathbf{x}^{(s_2)} \\ \vdots \\ \mathbf{x}^{(s_{n_s})} \end{pmatrix}, \end{aligned}$$

where $\mathbb{0}$ is filler matrix composed exclusively of zeros.

2.0.5 Combination of several blocks

These constraints do not need to be satisfied only for a block, but for each block present in the structure. The complete constraint system of a structure can then be built iteratively by adding a block at each time step:

$$\begin{aligned}
A_{eq}^{(t)} &= \begin{pmatrix} A_{eq}^{(t-1)} & A_{eq}^{col_t} \\ \mathbb{0} & A_{eq}^{block_t} \end{pmatrix} \\
A^{(t)} &= \begin{pmatrix} A^{(t-1)} & \mathbb{0} \\ \mathbb{0} & A^{block_t} \end{pmatrix} \\
\mathbf{x}^{(t)} &= \begin{pmatrix} \mathbf{x}^{(t-1)} \\ \mathbf{x}^{block_t} \end{pmatrix} \\
\mathbf{b}_{eq}^{(t)} &= \begin{pmatrix} \mathbf{b}_{eq}^{(t-1)} \\ \mathbf{b}_{eq}^{block_t} \end{pmatrix} \\
\mathbf{b}^{(t)} &= \begin{pmatrix} \mathbf{b}^{(t-1)} \\ \mathbf{0} \end{pmatrix},
\end{aligned}$$

where $A_{eq}^{col_t}$ is taken care of the reactive forces according to Newton's third law by placing the opposite of the matrix $A_{eq}^{(s_i)}$ in the rows corresponding to the support block:

$$A_{eq\ jx,jy,jm}^{col_t} = \begin{cases} \mathbb{0} & \text{if the block j does not share an interface with the new block} \\ -A_{eq}^{(s_i)} & \text{if the block j shares the interface i with the new block} \end{cases},$$

where the indices jx, jy, jm represent the three lines of the matrix $A_{eq}^{col_t}$ that compute the equilibrium of the block j.

2.0.6 Robot forces

The problem must also deal with a robot holding a block. This is done by using the initial matrices and vector $A^{(0)}$, $\mathbf{b}^{(0)}$,$\mathbf{x}^{(0)}$, (while $A_{eq}^{(0)}$ and $\mathbf{b}_{eq}^{0}$ are initialised empty). The vector

$$\mathbf{x}^{(0)} = \begin{pmatrix} \mathbf{F}_0 \\ \vdots \\ \mathbf{F}_{N-1} \end{pmatrix}$$

contains the forces applied by each robot (numbered from 0 to $N-1$). Each F_n, $n = 0, \ldots, N-1$, contains 6 components

$$\mathbf{F}_n = \begin{pmatrix} F_{x+} \\ F_{x-} \\ F_{y+} \\ F_{y-} \\ M_+ \\ M_- \end{pmatrix},$$

each representing the force (or moment) acting in a particular direction. If the robot n holds the block i at a certain time during the simulation, A_{eq} is changed in the following way:

$$A_{eq\ (ix,iy,im)\times\mathbf{n}} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ -p_{ry} & p_{ry} & p_{rx} & -p_{rx} & 1 & -1 \end{pmatrix},$$

where the indices $(ix, iy, im) \times \mathbf{n}$ represent the three lines corresponding to the block i and the six columns corresponding to the robot n.

2.0.7 LP formulation of the physics module

A linear program (LP) has a linear objective and several linear equality and inequality constraints. In matrix form, the problem can be formulated as follows:

$$\begin{aligned} &\min_{\mathbf{x}}(\mathbf{c}\cdot\mathbf{x}) \\ &s.t.\ A_{eq}\mathbf{x} = \mathbf{b}_{eq}, \\ &\qquad A\mathbf{x} \leq \mathbf{b}, \\ &\qquad \mathbf{x} \geq \mathbf{0}, \end{aligned}$$

As can be seen, the constraints defined above can be used directly in the LP, and only the cost vector $\mathbf{c}$ remains to be defined. As no preference is given to any of the forces, all forces were then given a constant weight of 1.

2.0.8 Soft constraints

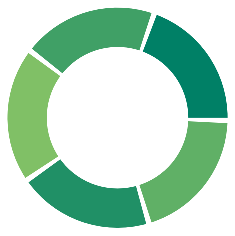
In the next section, the forces applied to each blocks are added to the structures. Another helpful aid that can be added is the failure points of the structures, allowing to know whether a collapse originates from a robot that is not able to sustain the force, a friction that is too large or an out of balance block. To find and indicate these failure points, the LP is extended with unbounded sticking forces that would act on the mortar or binding if some was used:

$$\begin{aligned}
A^{(t)}_{eqsoft} &= \begin{pmatrix} A^{(t)}_{eq} & -A^{(t)}_{eq} \end{pmatrix} \\
A^{(t)}_{soft} &= \begin{pmatrix} A^{(t)} & \mathbb{0} \end{pmatrix} \\
\mathbf{x}^{(t)}_{soft} &= \begin{pmatrix} \mathbf{x}^{(t)} \\ \mathbf{x}_{stick} \end{pmatrix} \\
\mathbf{b}^{(t)}_{eqsoft} &= \mathbf{b}^{(t)}_{eq} \\
\mathbf{b}^{(t)}_{soft} &= \mathbf{b}^{(t)} \\
\mathbf{c}^{(t)}_{soft} &= \begin{pmatrix} \mathbf{c}^{(t)} \\ \beta\mathbf{c}^{(t)} \end{pmatrix}
\end{aligned}$$

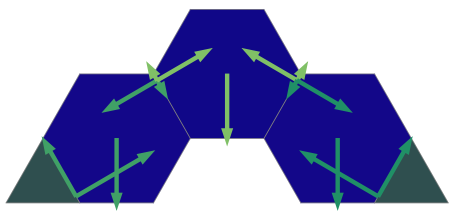
where β is an arbitrarily large constant (fixed at 10^6 in the experiments). This softening of the equality constraints allows a solution to always be found, and the nature of the linear cost tends to minimize the number of soft constraints violated rather than the magnitude of those violations, leading to fewer failure points and an easier-to-understand failure mode: The collapse of a structure is often caused by a single frictional force that is too large or could be fixed by using mortar in only a few locations. The result of using such soft constraints can be seen on the second line of Figure 4, where the red arrows show the failures points.

2.0.9 Visualisation

To illustrate the resulting forces when used in the setup, the forces acting on the corners of the same interface are added together, and only the resulting force is then displayed. The application point of the resulting forces is computed so the moment it creates equals the one created by all corner forces[1].



(a) Colors used sequentially to draw the arrows



(b) Example: without using different colors, it would not be possible to determine whether the friction forces applied on the top block are pointing upward or downward

Figure 3: Colors used to draw the arrows when no robot is holding the structure

The color rule used to visualize forces is shown in Figure 3, except for forces exerted by any robot. Whenever the arrows were drawn, the force exerted by a robot was represented in its corresponding color. With this notation, the first scenario in Figure **??** looks like the first scenario in Figure 4.

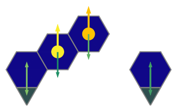
Thick red arrows are used to highlight failure points when soft constraints are not met, as shown at the end of the second scenario in Figure 4

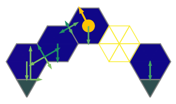
To increase visibility when a larger number of block are used, the amplitude of each force is normalized with respect to the largest one. Note that this information is still qualitatively visible by the color of the blocks, as in Figure 5.

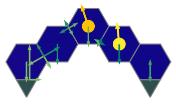
2.0.10 Safety kernel

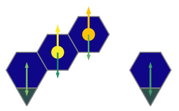
To give the physics solver a sense of stability, the notion of a safety kernel is introduced. The concept of safety kernel is to modify directly the application points of the forces (p_x and p_y on Figure 1), to create a safe region in which the forces can be applied. This method allows to eliminate some unstable equilibria that occur when forces are applied too close to the corners of the block (see

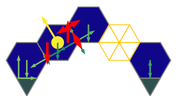
[1]This position can easily be computed by taking an amplitude-weighted average of the application points of each force


(a) Initial situation


(b) The robot 0 wants to place a new block


(c) Result


(d) Initial situation


(e) The robot 1 wants to place a new block

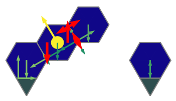
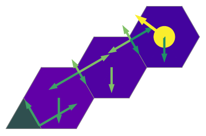

(f) The action cannot be performed

Figure 4: Same sequences as in figure ??, with the force arrows replacing the colored grid when a robot is holding a block and a red arrows at the failure points of the structure: In the second scenario, the robot would need to apply a torque and the third block would need to be stuck to the second one.


(a) When only three blocks need to be held, the force received by each is quite small compared to the maximum force of the robot, so the color remains uniform.

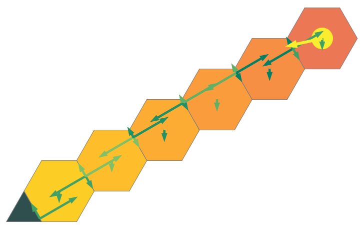
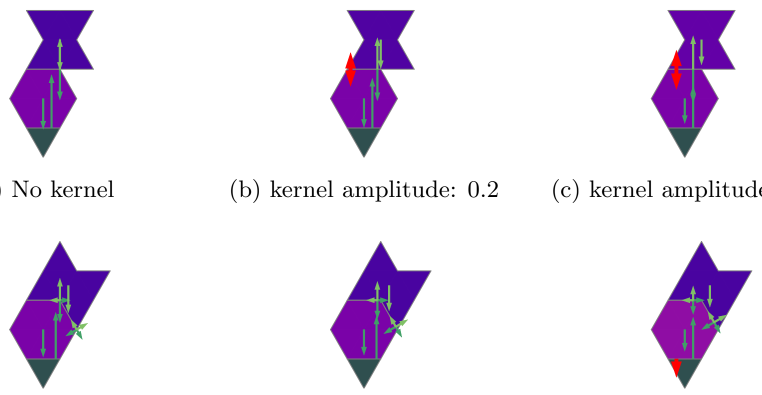

(b) While the actual force is greater, the size of the oblique arrows does not change when more blocks are used. However, the color of the blocks is modified to indicate the increase in force, and the weight are relatively smaller

Figure 5: To hold the structure together, the robot needs to apply a force whose amplitude and direction changes with the number of blocks.

Figure 6a: in practice, the stability of this structure is unreliable, as a slight shift of the top block to the left would be enough to make it collapse). Instead, the point of application of each force is shifted toward the center of the side by a fraction of its length. This simple trick eliminates most unrealistic equilibria, and the resulting structures could always be tested with real blocks (see the appendix ??). Note that the implementation of this safety does not requires to

modify the equation used above, but rather modify the geometry of the blocks.



(a) No kernel

(b) kernel amplitude: 0.2

(c) kernel amplitude: 0.5

(d) No kernel

(e) kernel amplitude: 0.2

(f) kernel amplitude: 0.5

Figure 6: On the top line, a small deviation from the position would result in a collapse of the structure, and the introduction of a kernel makes this structure always unstable, leading to safer and more realistic constructions. The structure on the bottom line was consistently stable in the prototype, justifying a kernel size of 0.2.

3 Validation

To validate the results, sometimes counter-intuitive, of the simulator, wooden pieces were laser-cut (see Figure 7). They were then placed against an inclined plane, and held in place using magnets. These prototypes were first used to estimate the friction coefficient between the blocks. As one can see on Figure 8, the friction coefficient of these experimental blocks is between 0.5 and 0.7. After thorough testing, all structures that are considered as stable in the simulator were possible to build using the real blocks. Some of the structures that broke the constraints of the simulator were still possible to build (see Figure 9), relying on 2 different reasons that were neglected by the simulator:

- Corner to corner forces: These forces were not considered in the simulator, as they are not reliable enough in real life. Some of them, however, were able to maintain the structure (see Figure 9a)
- Torque in the robot: Due to technical issues with real robots, the torque that they are applying is hard to control. To mitigate this effect, the simulator set a maximum torque of 0. The prototype, however, was perfectly able to hold of the rotation of the held blocks(see Figure 9b).

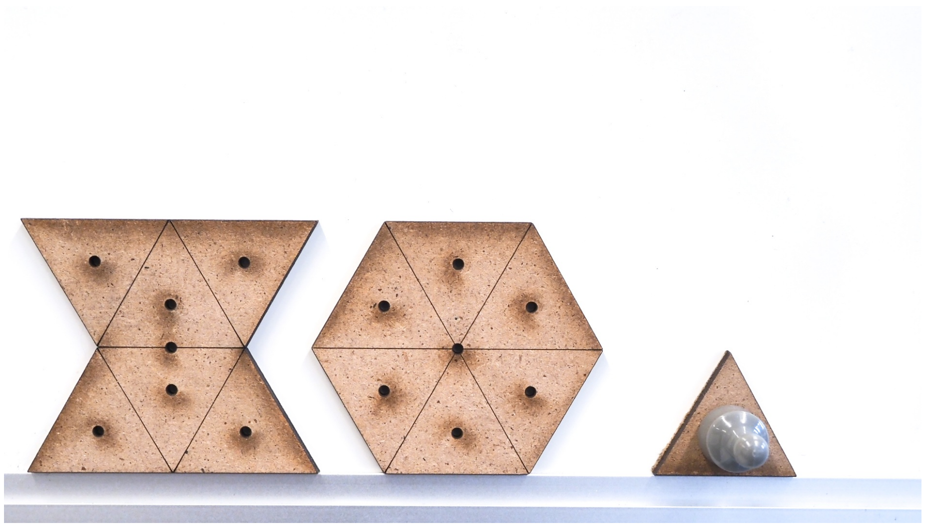


Figure 7: Prototype of blocks

These conservative approaches used in the simulator can be justified by the fact that a real block would have its sides not as well defined as a laser cut piece, thus relying on them would be inconvenient in the long run.

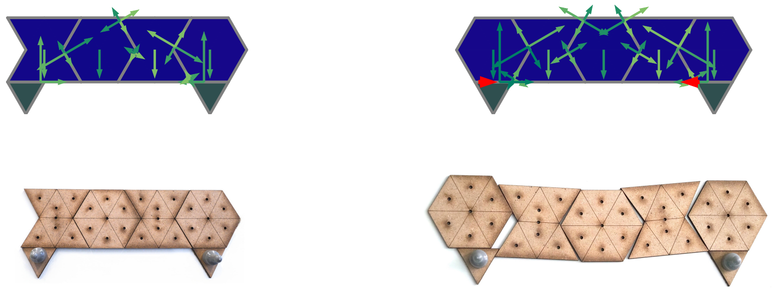
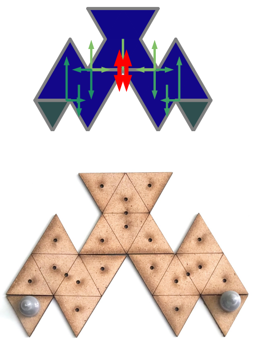
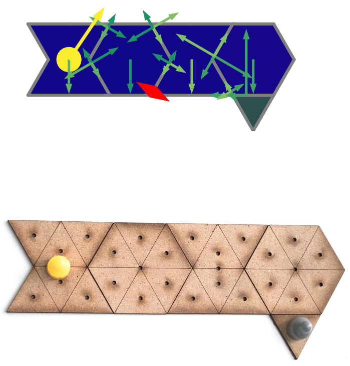


Figure 8: On the top line, equilibrium calculations from the simulator. On the bottom line, reproduction of the structures using the prototype blocks. Note that in the bottom right image, the blocks get a stable configuration by breaking the alignment



(a) Structure holding by using corner to corner forces



(b) Structure holding by applying a torque

Figure 9: On the top line, equilibrium calculations from the simulator. On the bottom line, reproduction of the structures using the prototype blocks. Note how despite being judged as unstable by the simulator, the real structures hold correctly