\section{Background}

The previously developed stability analysis for the 3D block printing system was only capable of utilizing 1x1, 1x2 and 2x2 blocks. This limitation was imposed due the complexity of the chosen strategy where every friction and normal force that could appear in a block had to be manually defined. Nonetheless, it was a great starting point to verify the usability of the proposed mechanical analysis to substitute the prior heuristic based one.

\subsection{Forces position and bounds}

The algorithm works like that: First, after loading the 3D model, the analysis must define the position of every single force the entire block model. To do that, it needs to verify, for each block, if there is another block in contact in the X, Y or Z axis and how they are touching each other. Then, it sets the lower and upper bounds for every force previously defined. This is needed to the later linear programing problem. Friction forces must be nonnegative. Normal forces in each axis are defined in a certain direction and must be positive and doesn’t have a maximum value. The normal forces internal of each knob does not have a correct direction and can have any value.

\subsection{Linear programing problem}

After the definition of the forces position and bounds, it is necessary to determine if the given structure is stable or not. This is a challenging problem to solve because it is very hard to know the exact value of each friction and normal force in complex systems. To solve this, it was proposed a linear programing problem that search for the best value of each force while respecting the static stability constrains. Such a problem is defined in equation \ref{eq:linprog }. Where the first equation is the liner inequality, the second is the linear equality and the last one is the bounds of the solution.

\begin{equation}

min f(x) \text{ such as}\left\{\begin{matrix}

A\cdot x \leq b\\

Aeq \cdot x = beq\\

lb \leq x \leq ub

\end{matrix}\right.

\label{eq:linprog}

\end{equation}

Here \(x\) is the force vector. The linear equality is the static balance of forces in each block, that is, the sum of forces and momentum in each block must be equal to zero. The linear inequality is defined as the sum of every friction force in each block connection, and it must be less than or equal the maximum friction value. The maximum friction value minus that sum is defined as capacity. The objective function for the linear programing problem is defined to maximize the value of the minimum capacity in the entire model. That way the stability analysis will try to redistribute all the forces, respecting the constrains, searching for a configuration that maximize the capacity of the weakest point of the structure.

\subsection{Insertion force}

It is necessary to define which block will be pushed by the insertion force when the robot place each block in the assembling process. Because this force can be seen as a disturbance in the force and momentum static balance, it was simply added the force and torques that appear to the linear equality equation of each block disturbed.

\section{Progress}

\subsection{New strategy for force definition}

It was completely changed how the position of each force is set in the input block model. The previous strategy required manual definition, what would require increasingly bigger effort to program as the block increases. It was implemented an automatic system that get each block information in the 3D space and given the relative position to all other adjacent block calculates all forces position automatically. Now the stability analysis can accept any \(1xn\) and \(2xn\) format, where \(1 <= n <= 9\).

With that change and other optimizations in the problem definition, it is now possible to utilize much more block types in the stability judge program, enabling a higher variety of 3D input models with a smaller execution time. The code documentation was also completely redone in English with every single function having the explanation of its functionality, input, and output. Making it easier to any new student to understand how it all work.

\subsection{Experimental constants value}

The program also needs the definition of a few experimental values. Those are: the maximum friction force a connection can handle before snaping out, the mass of each type of block and the value of gravity. The later was defined as \(g = 9.8 m/s^2\). For the masses, it was done an experiment using a precision scale to measure each available block in the laboratory. The results are in the table \ref{tab:mass}.

\begin{table}[]

\begin{tabular}{|c|c|}

\hline

\textbf{Block type} & \textbf{Mass (g)} \\ \hline

1x1 & 0.0380 \\ \hline

1x2 & 0.0695 \\ \hline

1x3 & 0.0971 \\ \hline

1x4 & 0.1288 \\ \hline

2x2 & 0.1266 \\ \hline

2x4 & 0.2458 \\ \hline

2x8 & 0.4583 \\ \hline

\end{tabular}

\caption{Mass experiment results.}

\label{tab:mass}

\end{table}

For the maximum friction value, it was done an experiment as in figure XX. Here, it was changed the hanged mass weight, number of connection points, and block type. It was considered also that every friction force in each connection has the same value. The free body diagram of one experiment can be seen in XX. For each experiment it is possible to calculate the static momentum balance to find the friction force \(F\_{fi}\). If a setting is not stable, that is, in the experiment the block snapped out of the base, it is assumed that this friction force greater than the maximum possible. In the other hand, if the block does not snap out, it is considered that the value calculated is lower than the maximum.

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX friction force photo

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX free body diagram

For the \(1xn\) block types, the maximum friction force is between 150.788 and 175.458. For the \(2xn\) block types, it was between 282.616 and 301.115. Values are in gram force. One explanation for this difference is the internal difference between these block types, where the \(2xn\) has less connection points in the center.

\section{Next steps}

Now that the stability analysis is complete, it is necessary to integrate it with the 3D block printing system. Before reassembling the physical setup and implementing the assembly process with the real robot, it is necessary verify the correct functionality in the simulated environment. For that, it is still necessary to study how to integrate the MATLAB code with the rest of the system and how to utilize and define the environment in the simulator.