URA Report for Computer Simulation of Many Object Collisions

Kevin Wan

Winter 2022

Contents

§1 Introduction

Modelling the simultaneous collision of many objects both sufficiently efficient and accurate is still an open problem. The object of study for this research is the General Reflections (GR) model proposed by [2] using the codebase provided here: (https://github.com/breannansmith/scisim).

The core idea of the proposed model revolves around an accurate and efficient solution to the Linear Complementarity Problem (LCP). The implementation used for the LCP solver in the provided codebase is transforms the problem into a quadratic program and solves it with the IPOPT library.

The purpose of this research is to examine both the consistency and efficiency of an alternative solution to the LCP in comparison to the available IPOPT implementation. This alternative solution is nearly a direct implementation of the Policy Iteration (PI) method described in [1].

Any additions made to the codebase for the purpose of the research, as well as all generators for discussed test cases can be accessed here: (https://github.com/Lyestria/scisim).

§2 Initial Setup

In this section, the setup of the original codebase will be outlined, as well as issues encountered with the given codebase.

- The project is set up and run on the Ubuntu 20.04 operating system.

- The old Eigen library in the given codebase is no longer functional and was replaced
- In addition to following the instructions given in `readme_ipopt.md`, MKL BLAS libraries must first be installed. This can be done with `sudo apt install intel-mkl`

  The reference library for `-with-blas` was set to `/usr/lib/x86_64-linux-gnu`
- The compilation of the project is done on g++9. For some reason, the IPOPT library couldn't be installed with g++11.
- All other dependencies for both IPOPT and the project must also be installed

§3 Implementation of Policy Iteration (PI)

The policy iteration method used to solve is the implementation described in [1]. This implementation can be specified in the input XML file by selecting the `policy_iteration` solver for the impact operator. This is done with the following line:

```
<solver name="policy_iteration" max_iters="100" tol="1.0e-6"/>
```

where `max_iters` specifies the maximum number of iterations we will attempt before returning an error and `tol` specifies the tolerance for the solution.

The implementation relies on the use of Eigen's implementation of the conjugate gradient method for finding the solution to the symmetric linear system.

§4 Analysis

For the analysis, the program was altered so that each time an LCP needed to be solved, both the IPOPT and PI solver would be run, and so their performance could be directly compared. This behaviour is provided by the `both` branch in the repository and is used when the PI solver is specified. For the purposes of this experiment, the relative tolerance for both solvers was set to 10^{-6}. In addition to the total time used for each solver, both the size of the LCP, as well as the number of iterations used by the PI solver is recorded.

For each of the test case scenarios, a generator was also created to generate test cases of various sizes for both the IPOPT and PI solvers.

§4.1 2D Test Cases

The program was first tested on the following 2D test cases with balls with identical mass.

- `balls_in_box`: This configuration is a larger and more densely packed version of a given configuration. It consists of a diagonally oriented box with a number of balls placed within it in a near-earth gravity setting.
- `triangle`: This configuration consists of an arrangement of balls in the form of an equilateral triangle with an incoming ball about to collide with one of the corners of the triangle.

In the `balls_in_box` configuration with 10000 balls, collisions initially started with the bottom-most balls before slowly propagating to all of the other balls. This eventually led up to LCP solves

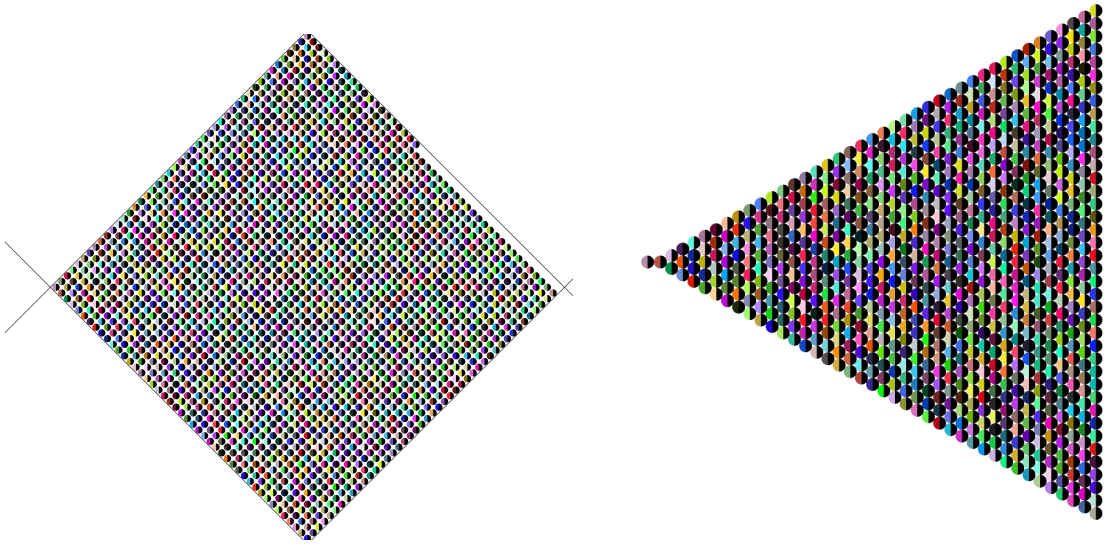

Figure 1: Structure of the 2D configurations. Note that these figures show a smaller number of balls than the used test cases for the purpose of clarity.

involving matrices of size 6000. After symmetry was broken, LCP solves continued to involve large matrices of size between 6000 and 12000 as a result of time-discretization making many collisions simultaneous. On the other hand, the `triangle` configuration generated LCP solves of a much lower size. For the configuration with a sidelength of 150 balls, the largest LCP matrices was only of size 318. Even the configuration with 1000 balls all had LCP matrices of size smaller than 2300. It was also noted that the GR algorithm got stuck in an infinite loop in this case when the tolerance was set to 10^{-6}. Changing the tolerance to 10^{-9} seemed to fix the issue. Furthermore, this behaviour occured when using both PI and IPOPT. This seems to suggest that the issue is not caused by our PI solver but instead due to GR not being able to deal with tolerances of this size.

While running on `balls_in_box`, the PI solver occasionally failed to converge after the initial symmetry of the configuration is broken. In such cases, it is observed that the resulting relative error can range from 2 to 100000 implying that it could be arbitrarily large in the worst case. No such issues were observed with the `triangle` configuration.

To gather more information about the conditions that lead to the PI solver failing to converge, additional information such as whether or not the matrix is an M matrix or diagonally dominant was recorded. With respect to be results, it seems that most of the M matrices that appeared were either of size smaller than 20 or part of the initial symmetrical collisions. This seems to suggest that in the absense of any specific setup, the matrices encountered are not M matrices. A similar observation was made for the diagonally dominant matrices only that the limit to their sizes (excluding the initial symmetrical collisions) is somewhat larger. Since the vast majority of LCP solves involved matrices that are neither M matrices or diagonally dominant, the results do not suggest that these factors and the convergence of the PI solver are causally related. Additional information about how much the matrices deviated from being an M-matrix (in both diagonal and non-diagonal entries) or being diagonally dominant (the number of rows breaking the conditions and how much they deviated) was also recorded but did not provide any further insight into this issue.

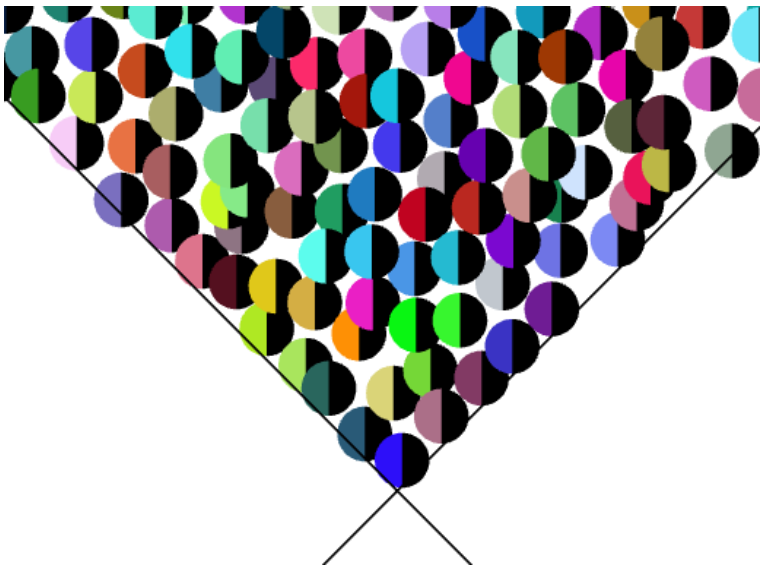

Figure 2: The balls intersect when the configuration gets too large

However, it was noted that during the simulation that the balls started to intersect with each other and the bounding box. This was suspected to be due to time-discretization leaving the algorithm unable to find a next position where the balls didn't intersect. It was believed that this may have been the cause of the instability of the PI solver.

To further investigate this issue, two more test cases were created based off of the `balls_in_box` configuration. These were `big_balls_in_box` which increased the size of the balls and `dt_balls_in_box` which increased the size of the discrete timestep. Both these tests produced similar simulations and aimed to reduce the possibility or signifigance of the balls intersecting. With configurations with the same number of balls, it was observed that the LCP matrices were much smaller due to a fewer number of collisions being considered simultaneous. To account for this, cases with 160000 balls were used, which resulted in LCP matrices of sizes up to 60000. However, no convergence issues were observed even with matrices of this size, furthering the belief that the cause of the lack of convergence is due to the balls significantly intersecting.

The ratio of the running time of the PI solver to the IPOPT solver was analyzed, producing Figure 3.

Note that the topmost points on the plot represent the LCP solves where the PI solver failed to converge. Hence, all 100 iterations are used and so the runtime for PI is correspondingly abysmal. In the other cases, it can still be noticed that the PI solver was significantly faster than the IPOPT solver where the matrix size is small and increasingly less so as the matrix size increases. From Figure 3, it can be seen that the PI solver was more than 2 times faster for matrices of size 2000 or less and so should be preferred then. It is inconclusive whether the PI solver can still be competitive for larger matrices with similar behaviour to `balls_in_box`.

It was also noted that the initial symmetrical collisions produced the much better runtimes for PI for matrices of sizes 200 to 6000 that can be seen in the figure. This seems to suggest that the later collisions that cause PI to not converge also seem negatively impact the performance of the PI solver even in the case where it is convergent. This was initially suspected to be because such cases were more likely to require more iterations before converging. The number of iterations used was plotted in Figure 4 which does show the initial symmetric collions to use a fewer number of iterations.

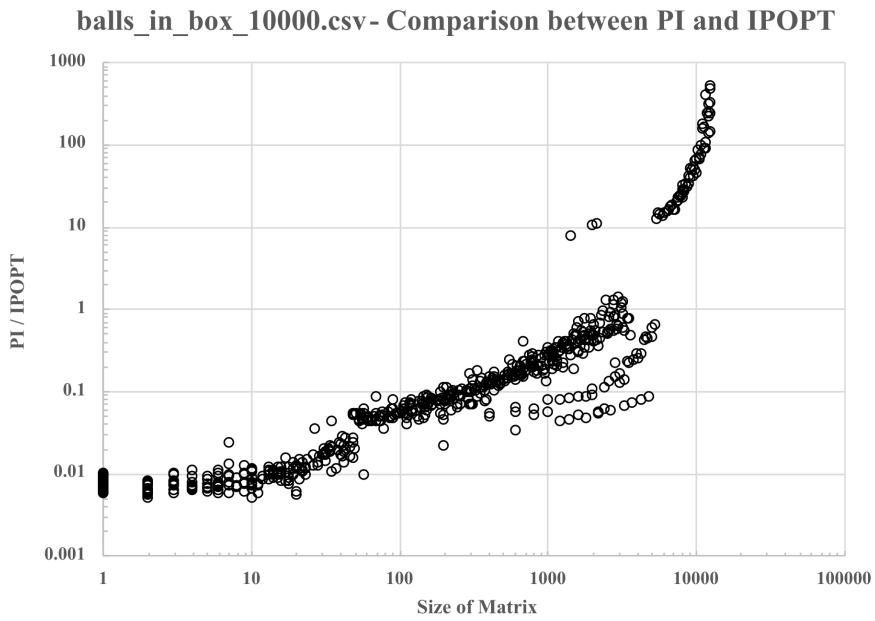

Figure 3: The ratio of the running time of the PI solver to the IPOPT solver. Note the log scale on both axes.

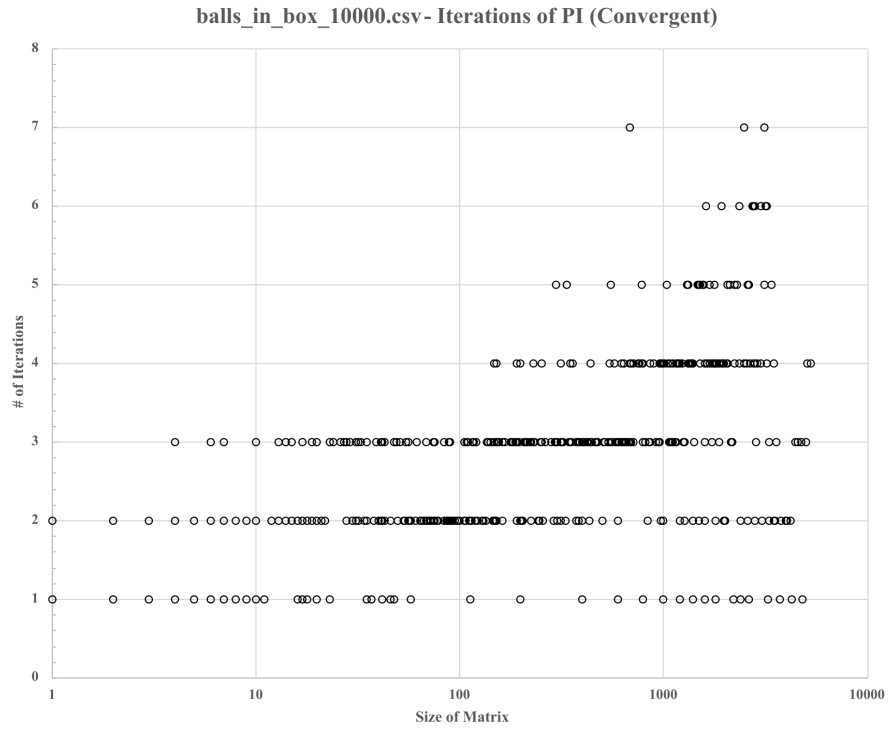

Figure 4: The number of iterations used by PI before converging.

Additionally, it can also be seen that the number of iterations is consistently less than 10 which allow us to safely conclude that the solves using more than 100 iterations are truly not convergent. The number of iterations used also does seem to to be positively correlated with size of the matrix, but this is not enough to fully explain the increased runtime of PI as the LCP size increases.

§4.2 3D Test Cases

The program was also tested on various 3D cases to see how much it impacted our results. Similar to in the 2D test cases, the balls are of equal size and mass. Furthermore, note that we track the same indicator variables were tracked as in the 2D test cases.

- `balls_in_box_3d`: This case consists of a closed box with a number of balls placed in a lattice inside, each with a random velocity.
- `gravity_balls_in_box_3d`: Similar to `balls_in_box_3d`, but with gravity acting on the balls and the box being less full.
- `small_balls_in_box_3d`: Similar to `gravity_balls_in_box_3d`, but with balls of a smaller radius.
- `overlap_balls_in_box_3d`: Similar to `small_balls_in_box_3d`, but with the balls overlapping initially.
- `pyramid`: Similar to `triangle` only it consists of a sphere hitting a square-based pyramid
- `falling_pyramid`: A square-based pyramid is dropped onto a static surface under the influence of near-earth gravity.
- `rfalling_pyramid`: Similar to `falling_pyramid` but with the pyramid rotated facing down.

Of the seven configurations, the configurations that had issues with PI convergence were `small_balls_in_box_3d`, `falling_pyramid`, and `overlapping_balls_in_box_3d`. Note that `small_balls_in_box_3d` was only observed to have issues with PI convergence for configurations of size 40 (containing 44800 balls) and that that `falling_pyramid` was only observed to have issues with PI convergence for configurations of size 50 (containing approximately 40000 balls). On the other hand, `overlapping_balls_in_box_3d` was observed to have issues with PI convergence for cases of size 3 containing only 18 balls.

The result from `small_balls_in_box_3d` could somewhat be expected mostly due its similarity to the 2D `balls_in_box` case. However, the lack of convergence was still not observed at size 20 (containing 5510 balls), and so does not provide any small non-convergent LCP problems. In comparison to `gravity_balls_in_box_3d`, this configuration reduced the size of the balls so that intersections of spheres would hopefully be more frequent or impactful.

The result from `falling_pyramid` was slightly more surprising considering its counterpart, `rfalling_pyramid`, did not run into any PI convergence issues. However, similar to the above case, the pyramid hitting the static plane starting with the base would necessarily leave balls near the center of the base less room to move, possibly increasing the number of impactful collisions between balls.

Unfortunately, no information was gathered in this research over whether or not significant intersections of spheres were actually occured in the program. If this was the case however, it would

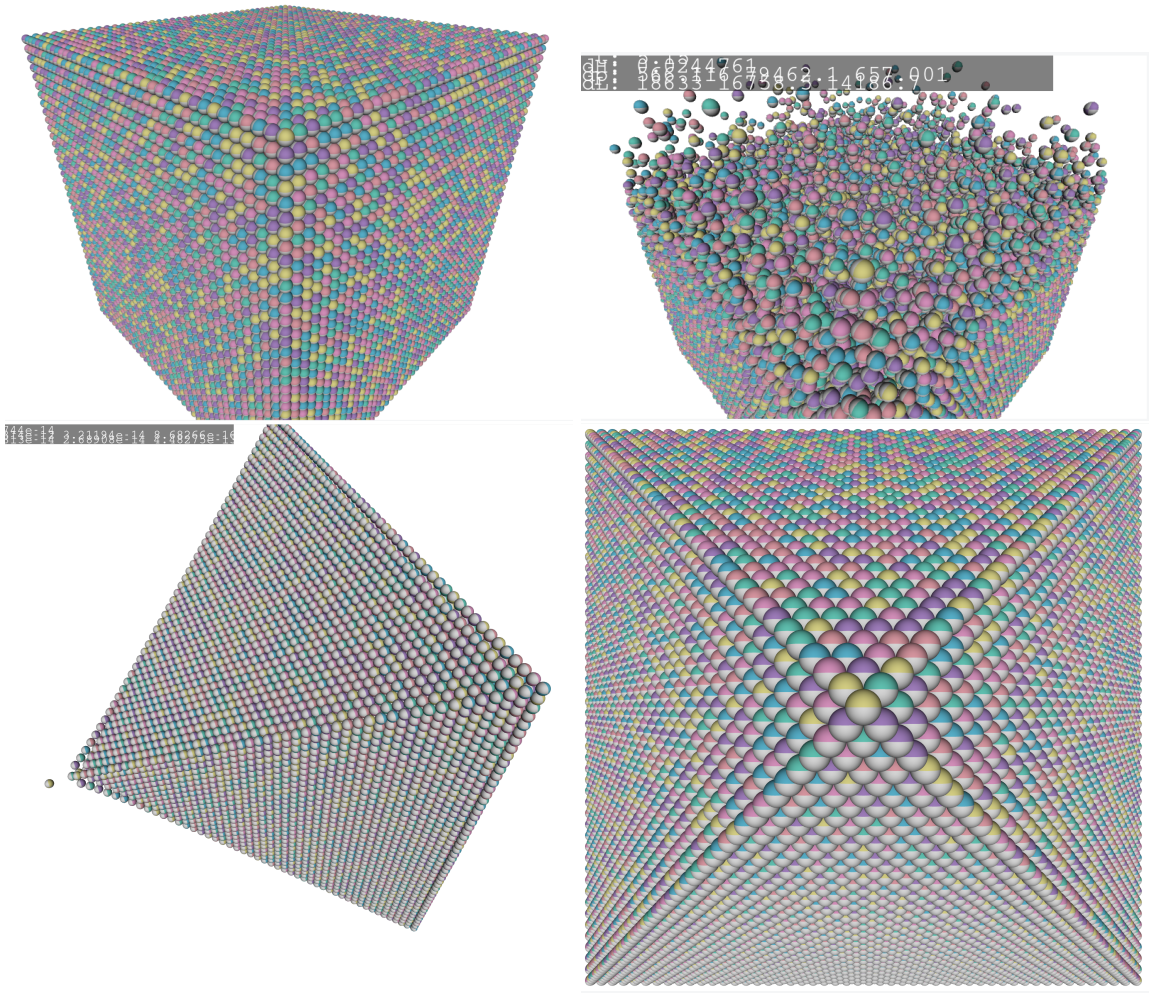

Figure 5: Top left: Initial arrangement for `balls_in_box_3d`.
Top right: Intermediate state for `gravity_balls_in_box_3d` (`small_balls_in_box_3d` is similar).
Bottom left: Initial arrangement for `pyramid`.
Bottom right: Initial arrangement for both `falling_pyramid` and `rfalling_pyramid`.

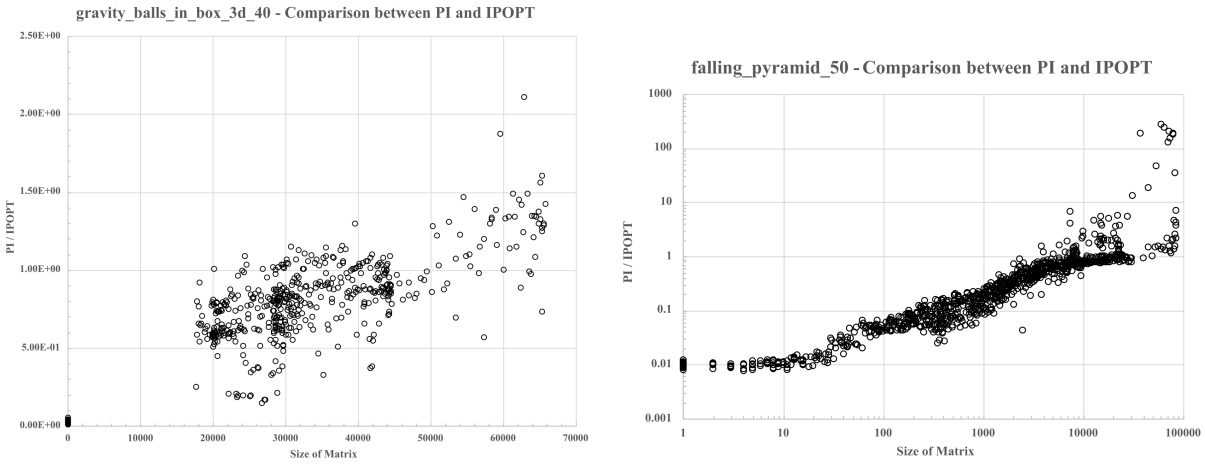

Figure 6: The ratio of the runtimes of PI and IPOPT. Note that linear scale for `gravity_balls_in_box_3d` and the log scale for `falling_pyramid`.

strengthen the argument that invalid configurations such as the overlap of balls is what causes the PI solver to not converge.

The `overlap_balls_in_box_3d` configuration was created intentionally to have massive overlap regions between the balls in the starting configuration, and so its results were expected. Fortunately, this also creates non-convergent LCP matrices that are small enough that we can compute their condition number in a reasonable amount of time. In this case, the condition number of the LCP matrices were found to all be in excess of 10^{15}. On the other hand, convergent LCP matrices found in `balls_in_box_3d` that are also unlikely to have intersecting balls were found to have condition numbers between 7 and 20. This suggests that a large number of significant intersections of balls could cause LCP matrices to have large condition numbers which are known to be problematic for PI. However, the amount of information we have on this correlation is still minimal and more data on the condition numbers is needed to make a more informed conclusion.

Finally, the ratio of the running time of the PI solver to the IPOPT solver was also plotted for both the `gravity_balls_in_box_3d` case and the `falling_pyramid` case, producing Figure 6.

Note that even though PI always converges `gravity_balls_in_box_3d` and sometimes doesn't converge for `falling_pyramid`, it appears that the ratio of PI to IPOPT is similar in both plots, and is also similar to the plot of `balls_in_box`. As such, this suggests that the weakening performance of PI as IPOPT increases is independent to both the intersection of balls during the simulation or whether or not PI always converges (assuming we only look at the remaining convergent LCP solves).

References

[1] Junyu Lai. Fast and scalable solvers for the fluid pressure equations with separating solid boundary conditions. Master's thesis, 2021.

[2] Breannan Smith, Danny M. Kaufman, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. Reflections on simultaneous impact. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2012)*, 31(4):106:1–106:12, 2012.