15th Canadian Masonry Symposium
Ottawa, Canada
June 2-5, 2025

Videogame-Inspired Out-of-Plane Collapse Analysis of Dry-Joint Masonry Structures

Anna Wang[i], Bora Pulatsu[ii], Sheldon Andrews[iii], and Daniele Malomo[iv]

ABSTRACT

Numerical modelling is a critical part of structural and seismic evaluations, particularly for existing unreinforced masonry (URM) structures built without mortar or exhibiting mortar-loss (i.e., dry-joint). Discontinuum methods are typically used for simulating the failure and collapse behaviours of dry-joint URM; however, such refined computational solutions often require excessive analysis times. An underexplored alternative for structural analysis of dry-joint URM is the use of physics engines, computational tools that present surprising conceptual similarities with DEM but are primarily used in animation and videogame industries for visually credible simulations. While these techniques feature exceptional computational speed when simulating rigid body collisions (i.e., contact, separation, and re-contact), they have yet to be rigorously scrutinized for URM structural analysis. This study explores the capabilities of PyBullet, a Python-based module operating the well-known, open-source Bullet Physics engine, in replicating the out-of-plane (OOP) collapse behaviour of dry-joint URM assemblies and full-scale constructions. Preliminary results indicate that PyBullet models can accurately predict the typical failure and collapse modes observed during experimental testing. However, the implicit Coulomb friction cone model utilized for simulating joint slip underestimates the angle of collapse during OOP tilting. Response predictions obtained using PyBullet are overall in agreement with previous experimental and traditional discontinuum results, but require significantly less time to complete, making them a promising alternative for complex URM discontinuum analysis.

KEYWORDS

Discontinuum analysis, unreinforced masonry, physics engines, PyBullet, dry-joint, collapse

[i] PhD student, McGill University, Montréal, Canada, anna.h.wang@mail.mcgill.ca
[ii] Assistant Professor, Carleton University, Ottawa, Canada, bora.pulatsu@carleton.ca
[iii] Associate Professor, École de Technologie Supérieure (ÉTS), Montréal, Canada, sheldon.andrews@etsmtl.ca
[iv] Assistant Professor, McGill University, Montréal, Canada, daniele.malomo@mcgill.ca

INTRODUCTION

Assessing seismic collapse mechanisms and post-failure consequences of unreinforced masonry (URM) structures is essential for predicting loss assessment due to natural hazards (e.g., earthquakes floods, landslides) and advising disaster preparedness (e.g., assessing road accessibility for emergency vehicles). Seismic collapse analysis of URM [1] is typically accomplished using numerical modelling, notably discontinuum methods (e.g., the Distinct Element Method (DEM) [2], the Applied Element Method (AEM) [3], and Non-Smooth Contact Dynamics (NSCD) [4]) which are now regarded as the leading analysis techniques for URM [5]. These methods are distinguished in their ability for simulating large displacements, contact, separation, and re-contact of discrete units (i.e., masonry). Consequently, these methods are capable of comprehensive representation of out-of-plane (OOP) URM collapse phenomena but necessitate impractical analysis times [6] and technical prerequisites that are unsuitable for practical engineering and research settings.

An emerging discontinuum method is the use of physics engines – computer graphics simulation platforms developed for visually credible 3D animation applications (e.g., videogames, movies). Physics engines employ a contact mechanics algorithm that shares key features with that of discontinuum methods (e.g., point-contact constitutive models, collision detection processes, time-stepping scheme). However, their more efficient formulation and implementation of distinctive numerical solutions (e.g., use of direct solvers, GPU computing resources, compliance with kinematic and geometrical constraints) allows for unrivaled computational speed and unconditional stability – key for fast, real-time simulations that are presently unachievable with standard discontinuum methods. Current applications of physics engines for URM analysis are limited in quantitative validations of mechanical outputs (i.e., numerical comparisons of forces and displacements) [7] but demonstrate proof of concept (i.e., exhibiting realistic crack propagation and damage patterns) for rigorous structural analysis of complex URM assemblies (e.g., walls, arches, full-scale structures).

To further comprehend and examine the capabilities of physics engines for URM analysis, we explore the novel application of PyBullet, a Python module operating the open-source Bullet Physics engine [8], in reproducing the OOP collapse mechanisms of dry-joint URM assemblies and buildings. Dry-joint URM structures are representative of existing building stock experiencing mortar degradation or historical constructions – thus, the accurate simulation of these constructions is critical for informing structural assessment and retrofit. OOP collapse mechanisms are of particular interest as they are often the "first-mode damage mechanism" in seismic collapse of historical (i.e., dry-joint) URM – in-plane (IP) failures are common as "second mode damage mechanisms" [9]. Discontinuum analysis of dry-joint constructions typically feature meso-scale representations of URM assemblies with Mohr-Coulomb no-tension/infinite compression relationships at the contact interfaces [10]. Numerous studies have validated such simulations using DEM and NSCD (to the authors' knowledge, no implementations of AEM for exclusively dry-joint URM are found in literature). DEM, despite its sensitivity to time-step size and conditional stability, has effectively replicated experimental OOP overturning collapse mechanisms observed during quasi-static tilting tests [11]. Similarly, NSCD – a non-smooth model featuring Signorini's impenetrability condition (comparable to hard contact models in physics engines) – has reproduced OOP "rotation-like" mechanisms during seismic simulations of a full-scale URM church [12] (albeit with loss of geometric fidelity in exchange for computational cost). These studies provide an existing suite of trusted discontinuum models to validate our unconventional use of PyBullet for URM analysis. PyBullet was the selected physics engine for its accessibility with engineers (i.e., Python is a high-level coding language with extensive documentation) – contact formulations used in PyBullet and similar engines are discussed in Section 2. The outcomes from this study (Section 3) will support the potential of calibrated and improved physics engines as a rigorously validated numerical modelling approach that can address the current shortcomings of discontinuum techniques.

PHYSICS ENGINE CONTACT MECHANICS

This section presents a high-level overview of rigid body contact mechanics in physics engines, highlighting their unique numerical parameters and their influence on mechanical output. For appropriate comparisons with discontinuum modelling, readers are referred to Malomo and Pulatsu (2024) [13] for respective contact formulations and applications thereof. The following equations and notation are adopted from Andrews et al. (2022) [14].

Time discretization in physics engines is typically represented as frames per second, FPS, the rate at which the system can generate frames (i.e., proceed to the next time increment). This metric is typically manually specified (adaptive time-stepping is favoured in DEM, in contrast) and dependent on the preferred visual fidelity of the simulation – a faster (up to 60 fps) FPS for real-time, first-person videogames and a slower (< 30 fps) FPS for 3D animations. Time increments can also be divided further into n_s sub-steps, which are not visually rendered but contribute to calculating the actual time-step, Δt (Eqn. **Error! Reference source not found.**), used in the time integration scheme. Physics engines also typically prefer implicit time-stepping (opposed to explicit integration schemes in DEM) for their numerical stability at larger (magnitude $10^{-2}\ s$) time-steps (favoured for high-speed, real-time simulations that require unconditional stability).

(1) $\Delta t = \frac{1}{n_s \cdot FPS}$

Collision detection processes are run at each time-step, h, to identify rigid bodies that are currently in contact. For efficiency, physics engines will separate the collision detection pipeline into two phases: (i) broad phase, where simple shapes (e.g., boxes, spheres) are used to eliminate objects that are definitely not in contact, and (ii) narrow-phase, where possible collisions are reduced using more complex shapes (e.g., convex hull, mesh-based geometries). Techniques such as using a collision margin (buffers around the specified collision shape) and continuous collision detection (CCD) can also be implemented to catch edge cases (e.g., edge-edge intersections) and prevent artificial tunneling (i.e., fast moving objects passing through each other). Critical outputs from the collision detection phase that are passed to subsequent stages (i.e., equations of motion) include: discrete contact points and their global position, $\boldsymbol{q}$, a contact normal direction, $\hat{\boldsymbol{n}}$, to inform how bodies should move to avoid further inter-penetration, and a penetration (i.e., gap) measure, $\psi(\boldsymbol{q})$, describing the gap or penetration between two bodies.

Once contact is established, physics engines then proceed with implementing the Newton-Euler equations of motion – typically solved at the velocity-level (compared to the displacement-level in DEM). These equations of motion are presented as a second-order ordinary differential equation (ODE), with the following system inputs: the system masses $\boldsymbol{M} \in \mathbb{R}^{n \times n}$, their respective velocities $\boldsymbol{u} \in \mathbb{R}^{n}$, and a function $\boldsymbol{F}$ that defines the applied forces on the system for each degree of freedom n. Using a first-order Taylor expansion of the implicit velocities, $\boldsymbol{u}^{+} \approx \boldsymbol{u} + \Delta t \dot{\boldsymbol{u}}$ (the superscript + denotes implicit quantities), the rigid body kinematics of the system can be represented as a linear equation (Eqn. **Error! Reference source not found.**).

(2) $\boldsymbol{M}\boldsymbol{u}^{+} = \boldsymbol{M}\boldsymbol{u} + \boldsymbol{F}\Delta t$

Boundary conditions and non-interpenetration requirements in physics engines are implemented through the application of constraint equations, $\psi(\boldsymbol{q}) \in \mathbb{R}^{m}$, where m is the number of constraints. The forces needed to impose these constraints have a magnitude λ and a direction determined by the constraint gradient, $\boldsymbol{J} \in \mathbb{R}^{m \times n}$ (assumed to be constant throughout the time-step) (Eqn. **Error! Reference source not found.**).

(3) $\boldsymbol{J} = \frac{\partial \psi(\boldsymbol{u})}{\partial \boldsymbol{u}}$

The contact (i.e., constraint) force in the normal direction is applied as a non-interpenetration impulse, where the magnitude $\lambda_{\hat{n}}{}^{+} \in \mathbb{R}^{m}$ is defined by a "push-only" spring-dashpot applied at the contact point with contact stiffness k_n and damping c. The implicit formulation of this impulse (Eqn. **Error! Reference source not found.** has a linear relationship with the relative displacement between the bodies (ψ^{+}). As shown in Figure 1, the magnitude of the normal contact impulse is also dependent on the relative normal velocity between the bodies, $v_{\hat{n}}{}^{+}$. A greater contact impulse is required to prevent further overlap if the bodies are expected to continue penetrating in the next time step (i.e., $v_{\hat{n}}{}^{+} < 0$) and a smaller or zero-magnitude impulse is applied if the bodies are in constant contact (i.e., $v_{\hat{n}}{}^{+} = 0$) or moving away from each other (i.e., $v_{\hat{n}}{}^{+} > 0$). No contact normal impulse is applied if the bodies are not in contact (i.e., zero tensile strength, $f_t = 0$).

(4) $\lambda_{\hat{n}}{}^{+} = -k_n \psi^{+} - c v_{\hat{n}}{}^{+}$

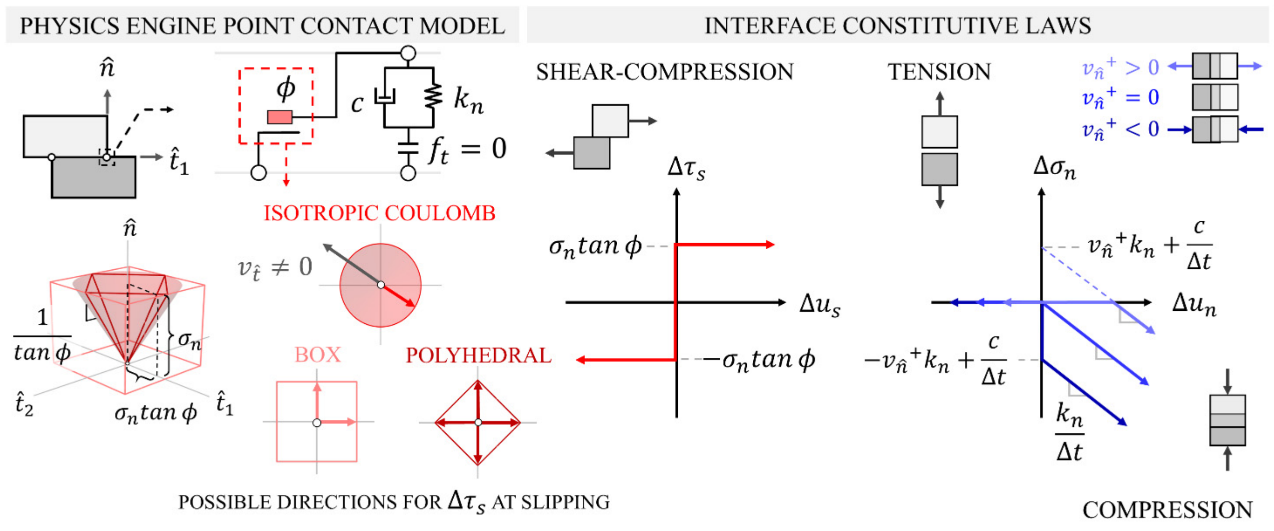

Figure 1: Typical physics engine point contact model and interface constitutive laws (expressed in the stress-displacement domain) for shear-compression and normal tension and compression

Shear contact impulses, $\lambda_{\hat{t}}$, follow Coulomb assumptions of planar dry friction such that the maximum shear impulse magnitude at slipping (i.e., non-zero relative shear displacement) is capped at $\lambda_{\hat{t}}{}^{+} = \lambda_{\hat{n}}\, tan\, \phi$ (where ϕ is the friction angle). Assuming perfectly isotropic Coulomb friction (which can be visually represented with a "friction cone") (Figure 1), the direction of the shear impulse ensures that energy is maximally dissipated. Numerical errors (if present) in the friction model can be attributed to their coupling with contact normal impulses (which is dependent on contact stiffness and damping) and/or linearized approximation of the friction problem (for numerical efficiency, e.g., polyhedral cone approximation [15], [16], box approximation [14]) (Figure 1).

Once contact impulses have been determined, the equations of motion for the complete multibody system can be expressed as follows:

$$(5)\ \begin{bmatrix} \boldsymbol{M} & -\boldsymbol{J}^{\boldsymbol{T}} \\ \boldsymbol{J} & \boldsymbol{\Sigma} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{u}}^{+} \\ \boldsymbol{\lambda}^{+} \end{bmatrix} = \begin{bmatrix} \boldsymbol{M}\dot{\boldsymbol{u}} + \Delta t \boldsymbol{f} \\ -\boldsymbol{\Upsilon}\frac{\psi}{\Delta t} - \mathbf{E}\boldsymbol{J}\dot{\boldsymbol{u}} \end{bmatrix}$$

Matrices $\boldsymbol{\Sigma}$ and $\boldsymbol{\Upsilon}$ contain the constraint force mixing (CFM) and error reduction parameters (ERP) for each constraint m. The CFM introduces artificial compliance to the system (i.e., allowing flexibility for violating constraints), and the ERP determines fraction of the constraint error (ψ^{+}) that is resolved in the next time-step. These constraint stabilization parameters are a product of the spring-dashpot non-interpenetrative contact model and can be tuned using Δt, k_n, and c (or sometimes directly specified as an input). The matrix $\mathbf{E}$ contains the coefficients of restitution (which are accounted for while solving for kinematic constraints – typically 0). The physics engine's chosen solver – e.g., pivoting methods (which can find an exact solution using a direct solver), iterative methods (which find an approximate solution using iterations) – is then used to find the system velocities at the next time-step, $\boldsymbol{u}^{+}$ from Eqn. **Error! Reference source not found.**). Bullet Physics implements the Projected Gauss-Seidel iterative solver. The rigid body positions, $\boldsymbol{q}$, are then updated using $\boldsymbol{q}^{+} = \boldsymbol{q} + \Delta t \boldsymbol{S}\boldsymbol{u}^{+}$, where $\boldsymbol{S}$ maps the angular velocities in $\boldsymbol{u}^{+}$ to their respective quaternion orientations.

OUT-OF-PLANE COLLAPSE ANALYSIS

The analyses presented in this section were run using Python version 3.12 and PyBullet version 3.2.6 [8] on a Dell Precision 7865 Tower PC equipped with an AMD Ryzen Threadripper CPU. Select constructions from experimental OOP tilting tests by Restrepo-Vélez et al. (2014) [9] that were modelled using DEM by Bui et al. (2017) [11] were replicated in the PyBullet simulation environment (naming conventions from Restrepo-Velez et al. are used herein). Experimental assemblies were constructed of masonry blocks with dimensions 80 mm × 40 mm × 30 mm and density 2680 kg/m^3 and wood lintels/joists (if present) with a density of 160 kg/m^3. Specimens included C-walls of varying aspect ratios and a two-story building – exact configurations and dimensions can be found in Restrepo-Vélez et al. (2014) [9].

The tilting apparatus in PyBullet was constructed by using the *p.createMultiBody* function to join two rigid plates of dimensions 1200 mm × 800 mm × 10 mm at the edge with a hinge constraint (i.e., rotation OOP is only degree of freedom). All rigid bodies, including the masonry units, in the simulation used the box collision shape. A narrow support half the height of a single URM unit was fixed with respect to the top tilting plate to prevent the structure from sliding off. Like in the DEM simulations, a friction coefficient of 0.67 – the minimum experimentally found coefficient – was used between the masonry joints (a coefficient of 0.4 was used between wood and masonry). Through preliminary simulations, the authors observed that variation in the friction coefficient had a minimal effect on tilting angle at collapse (±1°). Gravity was set to -9.8 m/s^2, and the simulation was manually stepped using *p.stepSimulation*. Each model commenced with at least 2 seconds of purely gravity loads to ensure equilibrium. For quasi-static tilting, the apparatus was rotated at a rate of 1 degree/s until near collapse (1-2 degrees prior to expected collapse), where the rotation rate was slowed to the experimental rate of 0.045 degree/s. Additional physics engine parameters used are shown in

Table 1. While the CFM parameter was not exposed in the PyBullet user interface, the ERP could be specified on a scale from 0 to 1 – the fraction of the positional error to be corrected in the next time-step.

Table 1: PyBullet simulation parameters

Time-step	Sub-steps*	Solver iterations	Contact stiffness	Contact damping	ERP	Coefficient of restitution
[*s*]	[–]	[–]	[*N/m*]	[*Ns/m*]	[–]	[–]
0.001	0	10	10^9	10	0.95	0

*S42 construction used 3 sub-steps for stability

Assemblies under out-of-plane tilting

Assemblies simulated under OOP tilting can be grouped into three categories: interior collapse C-walls (S1-3, 5, and 6), exterior collapse C-walls (S7, 8-9, 10), and C-walls with an interior partition. The observed failure mechanisms and collapse progression can be shown in Figure 2, Figure 3, and Figure 4 for each category, respectively. As expected from DEM models and experimental results, the PyBullet simulations of the interior collapse C-walls exhibited collapse mechanism G (see [17] for detailed descriptions of collapse mechanisms referenced herein) – characterized as a central trapezoidal portion of the façade displacing and rotating outwards. Exterior collapse C-walls reproduced collapse mechanism A (i.e., overturning of the main wall) with the overturning portion hinging around the center of the wall. Specimen 22, C-wall with openings and an interior partition, exhibited a B2 collapse mechanism (i.e., overturning of the façade and portions of the orthogonal walls). Mechanisms A and B2 also feature diagonal cracking on the orthogonal walls (secondary IP shear-compression failure).

Collapse was herein defined as a local or global severe damage limit state or post-failure condition when (i) components lose their ability to withstand vertical load and (ii) units or components/assemblies dislocate and start interacting dynamically [1]. The critical angle at collapse (φ_{cr}) was defined as the rotational displacement where block behaviour fit the aforementioned definition of collapse – this was identified through visual observation of the collapse progression. While φ_{cr} was underestimated in PyBullet (as much as -40%) for all specimens, there was an overall decreasing trend of φ_{cr} while the length/height aspect ratio increased (Table 2), which is in agreement with experimental and DEM results. This underestimation is likely attributed to excessive block slipping which can be a product of the coupling between contact normal and friction impulses. This is especially evident in S22, where the lintels lose frictional resistance and lead to IP failure in the orthogonal walls and loss of connection with the façade. Unexpected rotations of blocks near the structure base are also observed, which can be attributed to the engine attempting to adhere to the non-interpenetration constraints. Conversely, DEM typically overestimated experimental critical angles (up to +15.32%) for interior walls and underestimated (up to -9.25%) for exterior walls.

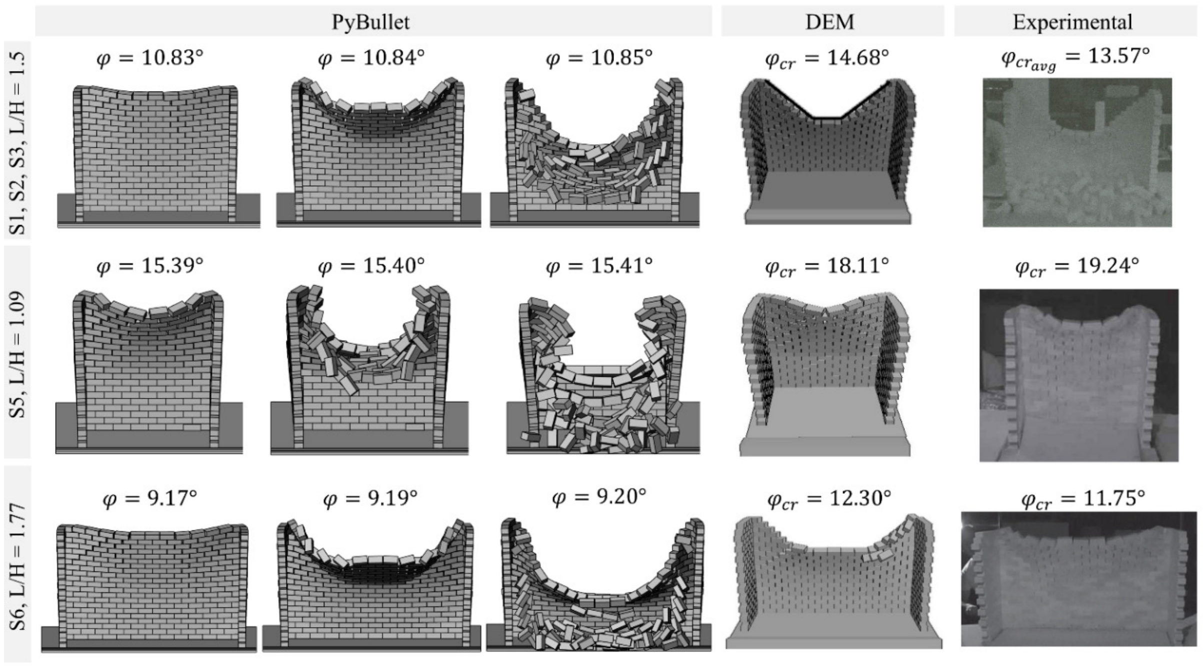

Figure 2: Observed collapse mechanisms in PyBullet, DEM [11]**, and experimental tests** [9] **for interior collapse C-walls (specimens S1, 2, 3, 5, 6)**

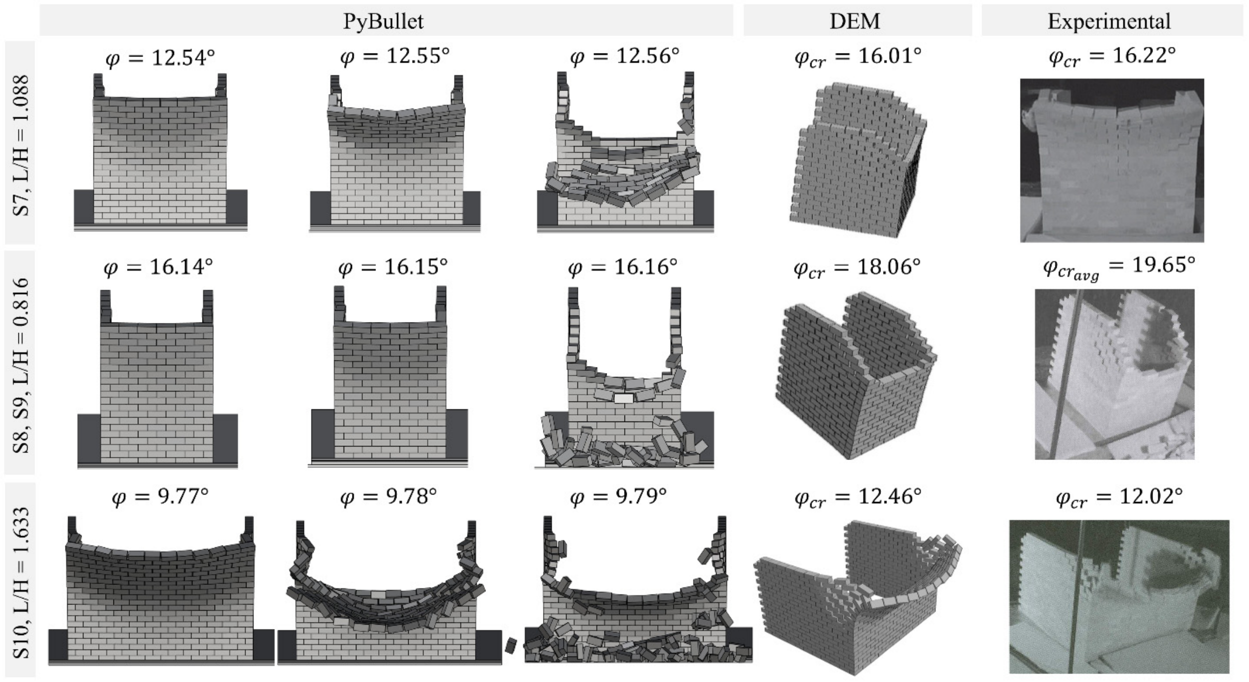

Figure 3: Observed collapse mechanisms in PyBullet, DEM [11]**, and experimental tests** [9] **for the exterior collapse C-walls (specimens S7, 8, 9, 10)**

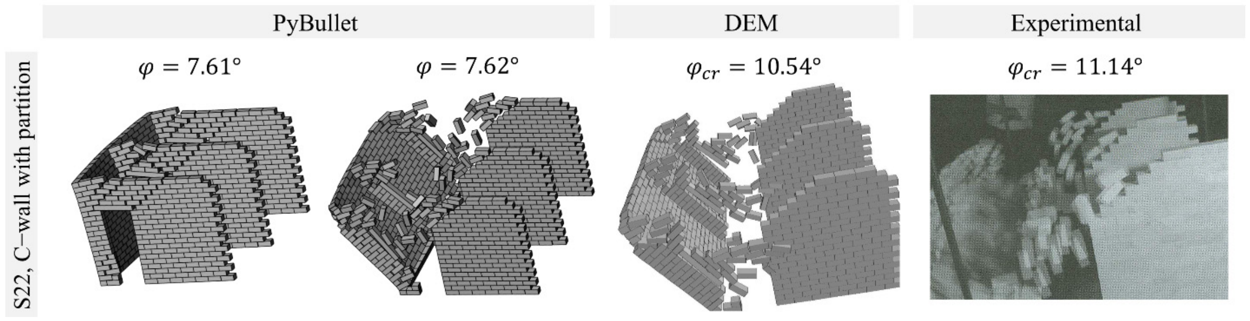

Figure 4: Observed collapse mechanisms in PyBullet, DEM [11]**, and experimental tests** [9] **for specimen S22 (C-wall with partition)**

Table 2: Comparison of critical angle for PyBullet, DEM [11]**, and experimental tests** [9]

	Specimen	**S1**	**S2**	**S3**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S22**	**S42**
	Aspect Ratio, L/H	1.5	1.5	1.5	1.09	1.77	1.088	0.816	0.816	1.633	-	-
Critical Angle φ_{cr} [°]	**PyBullet**	10.85	-	-	15.41	9.20	12.56	16.16	-	9.79	7.62	9.07
	DEM	14.68	-	-	18.11	12.30	16.01	18.06	-	12.46	10.54	11.09
	Experimental	14.25	12.73	13.71	19.24	11.75	16.22	19.90	19.39	12.02	11.14	13.28

Buildings under out-of-plane tilting

A two-story URM construction (S42) of dimensions 800 mm × 1040 mm × 1260 mm was also tested under OOP tilting (Figure 5). A set of wood joists were used to simulate the slabs with vertical reactions of 16.46 N applied at the end of each joist in the form of an equivalent density block. Eight openings are present (four on each story), each with a lintel on top. Due to the scale of the simulation, the number of sub-steps was increased to 3 for numerical stability (fewer sub-steps led to erroneous large displacements). Similar to specimen S22, this specimen exhibited the B2 collapse mechanism however with hinging at the height midpoint instead of the base (this contrasts with the experimental test which exhibited asymmetrical rotation of the façade). As the joists were free to move axially, shear resistance in the orthogonal direction was similarly governed by lintels – which can contribute to an underestimation (-34%) of the collapse angle. Nevertheless, diagonal shear cracking near the openings was still observed across both numerical models prior to collapse.

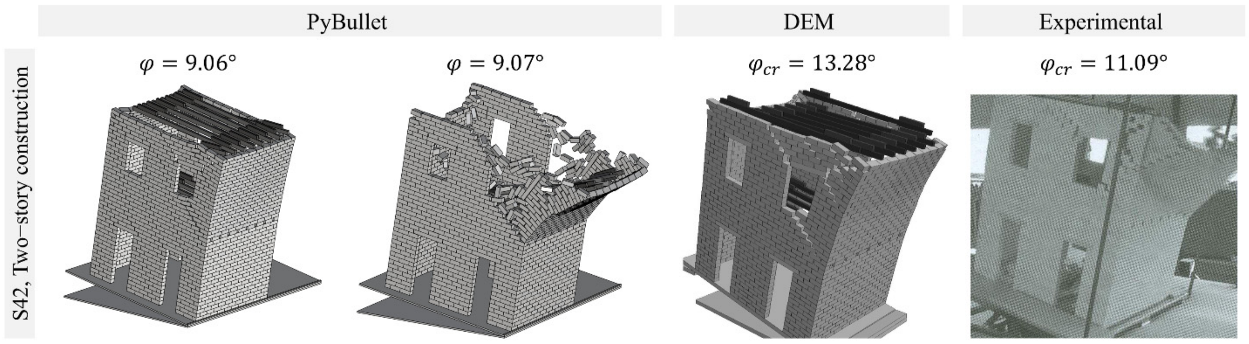

Figure 5: Observed collapse mechanisms in PyBullet, DEM [11]**, and experimental tests** [9] **for specimen S42**

Computational speeds for all simulations are shown in Table 3. As expected, the execution time increased as the number of rigid bodies in the system increased – similarly observed by Hamano et al. (2016) [18] for simulations using Bullet CPU. Increasing the tilting rate to 1 deg/s (for pre-collapse angles) greatly improved execution times. Although computational times for URM assemblies did not exceed 20 minutes, specimen S42 required 5.25 hours. However, it's acknowledged that this model had significantly more time-steps and over double the amount of rigid bodies. Computational difficulties may also arise from the high stiffness of the system (i.e., numerous rigid bodies stacked on top of each other) leading to issues with the numerical solver. Although respective DEM computational times are not reported, comparably sized models (with URM assemblies) can take up to 30 minutes to complete (based on authors' own experience). While further investigation of the model's sensitivity (regarding numerical efficiency and mechanical output) to selected inputs is warranted, this study confirms PyBullet's capability to simulate OOP behaviour beyond near collapse at a meso-scale discretization with practical execution times.

Table 3: PyBullet execution times

Specimen	**S1,2,3**	**S5**	**S6**	**S7**	**S8,9**	**S10**	**S22**	**S42**
Aspect Ratio, L/H	1.5	1.09	1.77	1.088	0.816	1.633	-	-
No. blocks	387	324	429	450	408	660	846	1826
Execution Time (min)	6.61	7.11	10.06	15.42	16.1	11.77	19.51	315.47

CONCLUSIONS

Initial exercises of quasi-static OOP tilting using PyBullet support the feasibility of physics engines for quantitative URM structural analysis. Although critical angles were underestimated, expected collapse mechanisms G, A, and B2 were largely reproduced – modifications to the friction and shear contact models in future studies can address conservative predictions of critical angle. Additional tests of different specimens (e.g., aspect ratio, number of blocks) and loading conditions (e.g., IP shear-compression, settlements, seismic loading) are necessary to further uncover capabilities and limitations of physics engines. Key considerations from this study include:

- The default isotropic Coulomb friction model embedded in PyBullet underestimates the shear contact forces leading to premature slipping in the joints.
- Additional studies into the direct effects of the physics engine parameters on mechanical behaviour are needed to fully understand the simulation's contact model.

- Loading schemes such as dynamic and cyclic testing have yet to be explored using PyBullet – findings therein can confirm constitutive laws under loading and un-loading conditions.
- Computational times are practical (under 20 minutes) for URM assemblies but can be improved using GPU computing resources or more efficient numerical solvers in the contact formulation.

ACKNOWLEDGEMENTS

The authors acknowledge funding from the Government of Canada's New Frontiers in Research Fund (NFRF) [Ref. No. NFRFE-2022-00312]. The work of the first author was also supported by the Fonds de recherche du Québec Nature et Technologies (FRQNT), through the B2X scholarship program.

REFERENCES

[1] D. N. Grant *et al.*, "Explicit modelling of collapse for Dutch unreinforced masonry building typology fragility functions," *Bull. Earthq. Eng.*, vol. 19, no. 15, pp. 6497–6519, Dec. 2021, doi: 10.1007/s10518-020-00923-y.

[2] P. A. Cundall, "Formulation of a three-dimensional distinct element model-Part I. A scheme to detect and represent contacts in a system composed of many polyhedral blocks," *Int. J. Rock Mech. Min. Sci.*, vol. 25, no. 3, pp. 107–116, 1988, doi: 10.1016/0148-9062(88)92293-0.

[3] H. Tagel-Din, "A new efficient method for nonlinear, large deformation and collapse analysis of structures," *Ph. D. thesis, Civil Eng. Dept., The University of Tokyo*. 東京大学, 1998. [Online]. Available: http://ci.nii.ac.jp/naid/10004542840/

[4] M. Jean, "The non-smooth contact dynamics method," *Comput. Methods Appl. Mech. Eng.*, vol. 177, no. 3–4, pp. 235–257, 1999, doi: 10.1016/S0045-7825(98)00383-1.

[5] A. M. D'Altri *et al.*, "Modeling Strategies for the Computational Analysis of Unreinforced Masonry Structures: Review and Classification," *Arch. Comput. Methods Eng.*, vol. 27, no. 4, pp. 1153–1185, 2020, doi: 10.1007/s11831-019-09351-x.

[6] F. Galvez, D. Dizhur, and J. M. Ingham, "Correction to: Adjacent interacting masonry structures: shake table test blind prediction discrete element method simulation (Bulletin of Earthquake Engineering, (2023), 10.1007/s10518-023-01640-y)," *Bull. Earthq. Eng.*, pp. 1–27, 2023, doi: 10.1007/s10518-023-01655-5.

[7] R. A. Bello, M. Günaydin, and A. C. Altunişik, "Structural Collapse Visualization Using Blender and BCB," in *Sustainable Civil Infrastructures*, A. S. Mosallam, B. El Bhiri, V. M. Karbhari, and S. Saadeh, Eds., Cham: Springer Nature Switzerland, 2023, pp. 163–172. doi: 10.1007/978-3-031-47428-6_13.

[8] E. Coumans and Y. Bai, "PyBullet Quickstart Guide," *PyBullet*, vol. 1, no. 1. PyBullet Quickstart Guide. https://docs. google. com/document/u/1/d …, pp. 1–84, 2023.

[9] L. F. Restrepo Vélez, G. Magenes, and M. C. Griffith, "Dry stone masonry walls in bending-Part I: Static tests," *Int. J. Archit. Herit.*, vol. 8, no. 1, pp. 1–28, Jan. 2014, doi: 10.1080/15583058.2012.663059.

[10] B. Pulatsu, S. Gonen, E. Erdogmus, P. B. Lourenço, J. V. Lemos, and R. Prakash, "In-plane structural performance of dry-joint stone masonry Walls: A spatial and non-spatial stochastic discontinuum analysis," *Eng. Struct.*, vol. 242, p. 112620, 2021, doi: 10.1016/j.engstruct.2021.112620.

[11] T. T. Bui, A. Limam, V. Sarhosis, and M. Hjiaj, "Discrete element modelling of the in-plane and out-of-plane behaviour of dry-joint masonry wall constructions," *Eng. Struct.*, vol. 136, pp. 277–294, Oct. 2017, doi: 10.1016/j.engstruct.2017.01.020.

[12] G. Lancioni, S. Lenci, Q. Piattoni, and E. Quagliarini, "Dynamics and failure mechanisms of ancient masonry churches subjected to seismic actions by using the NSCD method: The case of the medieval church of S. Maria in Portuno," *Eng. Struct.*, vol. 56, pp. 1527–1546, 2013, doi: 10.1016/j.engstruct.2013.07.027.

[13] D. Malomo and B. Pulatsu, "Discontinuum models for the structural and seismic assessment of

unreinforced masonry structures: a critical appraisal," *Structures*, vol. 62, 2024, doi: 10.1016/j.istruc.2024.106108.

[14] S. Andrews, K. Erleben, and Z. Ferguson, "Contact and friction simulation for computer graphics," in *Proceedings - SIGGRAPH 2022 Courses*, Association for Computing Machinery, Inc, Aug. 2022. doi: 10.1145/3532720.3535640.

[15] M. Anitescu and F. A. Potra, "Formulating Dynamic Multi-Rigid-Body Contact Problems with Friction as Solvable Linear Complementarity Problems," *Nonlinear Dyn.*, vol. 14, no. 3, pp. 231–247, 1997, doi: 10.1023/A:1008292328909.

[16] D. E. Stewart and J. C. Trinkle, "An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction," *Int. J. Numer. Methods Eng.*, vol. 39, no. 15, pp. 2673–2691, 1996, doi: 10.1002/(SICI)1097-0207(19960815)39:15<2673::AID-NME972>3.0.CO;2-I.

[17] D. D'Ayala and E. Speranza, "Definition of Collapse Mechanisms and Seismic Vulnerability of Historic Masonry Buildings," *Earthq. Spectra*, vol. 19, no. 3, pp. 479–509, Aug. 2003, doi: 10.1193/1.1599896.

[18] T. Hamano, M. Onosato, and F. Tanaka, "Performance comparison of physics engines to accelerate house-collapsing simulations," in *SSRR 2016 - International Symposium on Safety, Security and Rescue Robotics*, 2016, pp. 358–363. doi: 10.1109/SSRR.2016.7784327.