Crazy Fast Physics! Augmented Vertex Block Descent in Action!

Chris Giles
Independent
San Mateo, CA, USA
cgiles17@gmail.com

Elie Diaz
University of Utah
Salt Lake City, UT, USA
elie.diaz@utah.edu

Cem Yuksel
University of Utah
Salt Lake City, UT, USA
cem@cemyuksel.com

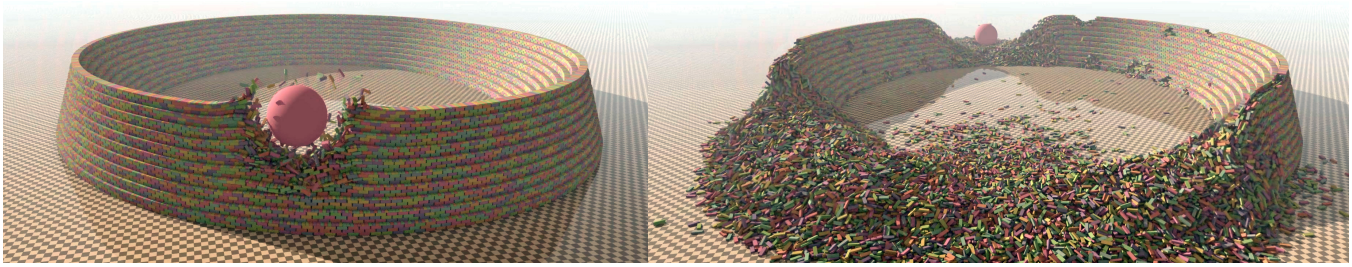

Figure 1: *Augmented vertex block descent allows faster physics simulations than ever before! It offers an efficient way of handling hard constraints, which is critical for simulating contacts and stacking. In this example, a sphere smashes a pile of 110,000 blocks, simulated using only 4 iterations per frame, though the pile has 40 levels! It is balanced by frictional contacts alone, so the whole structure begins to collapse after the initial impact. It only takes 3.5 ms (9.8 ms including collision detection) per frame to fully simulate this scene on an NVIDIA RTX 4090 GPU.*

ACM Reference Format:
Chris Giles, Elie Diaz, and Cem Yuksel. 2025. Crazy Fast Physics! Augmented Vertex Block Descent in Action!. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Real-Time Live! (SIGGRAPH Real-Time Live! '25), August 10–14, 2025, Vancouver, BC, Canada.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3721243.3735982

1 Introduction

We demonstrate the power of the *Augmented Vertex Block Descent* (AVBD) [Giles et al. 2025] method for physics simulation with unprecedented performance. AVBD is designed to handle a large number of objects (or degrees of freedom) connected with various constraints, including articulated joints, complex collisions, friction, and stacking. It not only delivers unconditional numerical stability but also superior computational performance and substantially improved numerical accuracy, as compared to prior methods.

AVBD extends the *vertex block descent* (VBD) [Chen et al. 2024] method, which presented superior performance than prior methods and unconditional stability in the context of soft body simulation. Our Augmented VBD method includes an Augmented Lagrangian formulation for handling hard constraints with VBD. These hard constraints are essential for properly handling contacts and stacking scenarios, which are essential for rigid-body simulations.

Our demos include large scenes with over a million objects, interacting with each other via joint constraints and frictional contacts, all simulated via AVBD at real-time frame rates on the GPU. We also show scenes that include long articulated chains and objects with large mass ratios interacting with each other, which are particularly challenging cases to simulate using prior methods.

SIGGRAPH Real-Time Live! '25, Vancouver, BC, Canada

ACM ISBN 979-8-4007-1545-7/25/08
https://doi.org/10.1145/3721243.3735982

2 Vertex Block Descent

VBD provides an efficient and numerically stable solution for an optimization problem that provides the solution of an implicit Euler time integration. More specifically, VBD solves the variational form of implicit Euler via position updates. VBD computes the positions at the end of the time step $\mathbf{x}^{t+\Delta t}$ by solving

$$\mathbf{x}^{t+\Delta t} = \underset{\mathbf{x}}{\operatorname{argmin}} \frac{1}{2\,\Delta t^2}\|\mathbf{x}-\mathbf{y}\|_M^2 + E(\mathbf{x})\,, \tag{1}$$

where Δt is the timestep size, $\mathbf{y}$ is the inertial positions, $\|\cdot\|_M$ represent mass-weighted norm, and $E(\mathbf{x})$ is the *total potential energy* evaluated at positions $\mathbf{x}$. The inertial positions $\mathbf{y}$ are calculated using the positions $\mathbf{x}^t$ and velocities $\mathbf{v}^t$ at the beginning of the timestep, such that

$$\mathbf{y} = \mathbf{x}^t + \Delta t\mathbf{v}^t + \Delta t^2\mathbf{a}_{\text{ext}}\,, \tag{2}$$

where $\mathbf{a}_{\text{ext}}$ represent the external acceleration, such as gravity. The total potential energy $E(\mathbf{x})$ is the sum of the energies of all force elements, including material deformations and collisions.

VBD solves this optimization problem using Gauss-Seidel iterations. Each iteration, updates $\mathbf{x}$ by modifying the position $\mathbf{x}_i$ of one vertex i at a time, keeping all other vertices of a deformable object fixed at their latest positions. When no other vertex moves, the minimization in Equation 1 is equivalent to a local minimization that only modifies $\mathbf{x}_i$, such that

$$\mathbf{x}_i \leftarrow \underset{\mathbf{x}_i}{\operatorname{argmin}} \frac{1}{2\,\Delta t^2}\|\mathbf{x}_i-\mathbf{y}_i\|_{\mathbf{M}_i}^2 + \sum_{j\in\mathcal{F}_i} E_j(\mathbf{x})\,, \tag{3}$$

where $\mathbf{M}_i$ is the mass matrix and E_j is the energy of a force element j in the set $\mathcal{F}_i$ of all force elements that use $\mathbf{x}_i$. All other force elements can be ignored here, since all other vertex positions are fixed while updating the position $\mathbf{x}_i$.

VBD solves Equation 3 using a single Newton iteration, which corresponds to solving the linear system

$$\mathbf{H}_i \, \Delta\mathbf{x}_i = \mathbf{f}_i \,, \tag{4}$$

where Δx_i is the position change between iterations, $\mathbf{f}_i$ is the force

$$\mathbf{f}_i = -\frac{1}{\Delta t^2}\mathbf{M}_i(\mathbf{x}_i - \mathbf{y}_i) + \sum_{J \in \mathcal{F}_i} \mathbf{f}_{ij} \,, \tag{5}$$

calculated using $\mathbf{f}_{ij} = -\partial E_j(\mathbf{x})/\partial \mathbf{x}_i$, and $\mathbf{H}_i$ is the Hessian

$$\mathbf{H}_i = \frac{\mathbf{M}_i}{\Delta t^2} + \sum_{J \in \mathcal{F}_i} \mathbf{H}_{ij} \,, \tag{6}$$

such that $\mathbf{H}_{ij} = \partial^2 E_j(x)/\partial \mathbf{x}_i^2$ is the Hessian of each force element j acting on vertex i.

Since vertices have 3 degrees of freedom (DOF), $\mathbf{H}_i$ and $\mathbf{M}_i = m_i\mathbf{I}$ are 3×3 matrices, where m_i is the mass of the vertex and $\mathbf{I}$ is the identity matrix. The same goes for simulating particles with 3 DOF.

When simulating rigid bodies with 6 DOF, $\mathbf{x}_i = [\mathbf{p}_i \;\; \mathbf{q}_{i,\mathrm{v}}]^T$ includes both the positions $\mathbf{p}_i$ and quaternions $\mathbf{q}_i = (q_{i,\mathrm{s}}, \mathbf{q}_{i,\mathrm{v}})$ that represent the orientations, where $q_{i,\mathrm{s}}$ and $\mathbf{q}_{i,\mathrm{v}}$ are the scalar and vector components. Since $\mathbf{q}_i$ is a unit quaternion, $q_{i,\mathbf{s}}$ can be safely omitted and the 6D subtraction operation can be defined as

$$\mathbf{x}_i - \mathbf{x}_j \coloneqq \begin{bmatrix} \mathbf{p}_i - \mathbf{p}_j \\ (2\mathbf{q}_i\mathbf{q}_j^{-1})_{\mathrm{v}} \end{bmatrix} . \tag{7}$$

In this case, $\mathbf{H}_i$ and $\mathbf{M}_i$ are 6×6 matrices, such that

$$\mathbf{M}_i = \begin{bmatrix} m_i\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_i \end{bmatrix} , \tag{8}$$

where the 3×3 matrices $\mathbf{I}$, $\mathbf{0}$, and $\mathbf{I}_i$ represent the identity matrix, zero matrix, and the rotated moment of the rigid body, respectively. Similarly, $\Delta\mathbf{x}_i = [\Delta\mathbf{p}_i \;\; \Delta\mathbf{w}_i]^T$ contains a position $\Delta\mathbf{p}_i$ and an orientation $\Delta\mathbf{w}_i$ component and the resulting update is defined as

$$\mathbf{x}_i + \Delta\mathbf{x}_i \coloneqq \begin{bmatrix} \mathbf{p}_i + \Delta\mathbf{p}_j \\ \mathrm{normalize}\left(\mathbf{q}_i + \frac{1}{2}(0, \Delta\mathbf{w}_i)\mathbf{q}_i\right) \end{bmatrix} . \tag{9}$$

3 Augmented Vertex Block Descent

In VBD, constraints can be modeled by a quadratic energy potential

$$E_j(\mathbf{x}) = \frac{1}{2}k_j\Big(C_j(\mathbf{x})\Big)^2 , \tag{10}$$

where k_j and C_j are the stiffness and the constraint error for constraint j, respectively. However, this does not work for hard constraints with infinite stiffness. Therefore, AVBD uses augmented Lagrangian to represent the energy for each iteration n as

$$E_j(\mathbf{x}) = \frac{1}{2}k_j^{(n)}\Big(C_j(\mathbf{x})\Big)^2 + \lambda_j^{(n)}C_j(\mathbf{x}) \,, \tag{11}$$

where $k_j^{(n)}$ is the finite stiffness and $\lambda_j^{(n)}$ the *dual variable* of the constraint, which are updated after each iteration, starting with

$$k_j^{(0)} = k_{\mathrm{start}} \qquad \text{and} \qquad \lambda_j^{(0)} = 0 \,, \tag{12}$$

where $k_{\mathrm{start}} > 0$ is an initial stiffness parameter, and using

$$\lambda_j^{(n+1)} = k_j^{(n)}C_j(\mathbf{x}) + \lambda_j^{(n)} \tag{13}$$

$$k_j^{(n+1)} = k_j^{(n)} + \beta\,|C_j(\mathbf{x})| \,, \tag{14}$$

where β is a parameter that controls the stiffness increment rate.

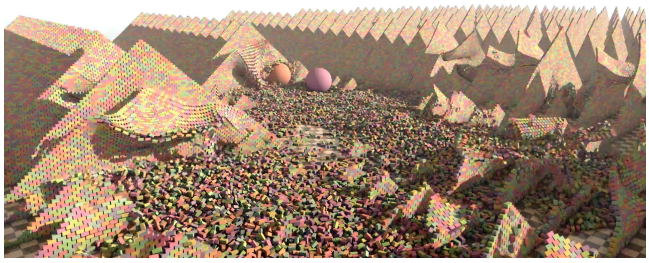

Figure 2: *510,000 blocks smashed by two spheres, simulated with AVBD using 4 iterations per frame (10.3 ms on NVIDIA RTX 4090).*

Using hard constrains, the Hessian $\mathbf{H}_i$ can easily become non-invertible. To avoid this, AVBD approximates the Hessians using

$$\mathbf{H}_{ij} \approx k_j^{(n)}\left(\frac{\partial C_j(\mathbf{x})}{\partial \mathbf{x}_i}\right)^T\frac{\partial C_j(\mathbf{x})}{\partial \mathbf{x}_i} + \tilde{\mathbf{G}}_{ij} \tag{15}$$

where the second term $\tilde{\mathbf{G}}_{ij}$ is a diagonal matrix that approximates

$$\mathbf{G}_{ij}(\mathbf{x}) = \left(k_j^{(n)}C_j(\mathbf{x}) + \lambda_j^{(n)}\right)\frac{\partial^2 C_j(\mathbf{x})}{\partial \mathbf{x}_i^2} \,, \tag{16}$$

such that each diagonal element $g_{ij,c}$ of $\tilde{\mathbf{G}}_{ij}$ is the norm of the vector $\mathbf{G}_{ij,c}$ that forms the column c of $\mathbf{G}_{ij}$, such that $g_{ij,c} = \|\mathbf{G}_{ij,c}\|$.

In real-time simulations, it is typical to use a limited number of iterations, instead of iterating until the constraint error is removed. Unfortunately, with hard constraints, the resulting error can inject an arbitrary amount of energy into the system due to position correction. AVBD limits this using $C_j(\mathbf{x}) = C_j^*(\mathbf{x}) - \alpha\, C_j^*(\mathbf{x}^t)$, where C_j^* is the original constraint function and $\alpha \in [0, 1]$ is the user-specified regularization parameter.

The convergence rate of AVBD can be significantly improved by warm starting the stiffness and dual variables using

$$k_j^{(0)} = \max\left(\gamma\, k_j^t,\; k_{\mathrm{start}}\right) \qquad \text{and} \qquad \lambda_j^{(0)} = \alpha\gamma\,\lambda_j^t \tag{17}$$

where k_j^t and λ_j^t are the ones computed after the last iteration of the previous frame, and $\gamma \in [0, 1)$ is the scaling parameter.

The user-defined parameters of AVBD do not need to be fine-tuned. We use $\alpha = 0.95$, $\beta = 10$, and $\gamma = 0.99$ for all examples.

4 Conclusion

AVBD offers real-time physics simulation with an unprecedented level of performance (Figures 1 & 2). It inherits the unconditional stability and the parallel computing power of VBD. It allows extreme mass variations and, unlike VBD, it can handle extreme stiffness ratios. AVBD's formulation is essential for properly resolving frictional contacts and stacking, in addition to simulations of articulated objects with arbitrary joints and attachment constraints. Most notably, AVBD can simulate stable stacking of large piles and long articulated chains with only a few iterations per frame.

References

Anka He Chen, Ziheng Liu, Yin Yang, and Cem Yuksel. 2024. Vertex Block Descent. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2024)* 43, 4, Article 116 (07 2024), 16 pages. doi:10.1145/3658179

Chris Giles, Elie Diaz, and Cem Yuksel. 2025. Augmented Vertex Block Descent. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2025)* 44, 4 (08 2025), 12 pages. doi:10.1145/3731195