Rigid Origami Simulator

Carl Osborne
Dept. of Computer Science
osbo@mit.edu

Ekanem Okeke
Dept. of Mechanical Engineering
eokeke@mit.edu

Hayashi Layers
Dept. of Computer Science
hayastl@mit.edu

***Abstract*—We present a computational tool for simulating the folding of rigid origami patterns using kinematic nodal analysis. In rigid origami, faces remain planar and deformation occurs only at crease lines. We model the system state using the dihedral angles at every fold, governed by geometric closure constraints around each vertex. To solve this highly non-linear system, we implemented a Damped Newton-Raphson solver. We address the computational bottleneck of Jacobian assembly and inversion by implementing a Sparse Matrix formulation. We demonstrate that for large grids (degrees of freedom > 1000), our sparse solver achieves near-linear time scaling compared to the cubic scaling of dense solvers, enabling real-time interactive simulation of complex patterns like the Waterbomb and Miura-ori tessellations.**

I. Introduction & Motivations

Rigid origami—folding structures where panels remain undeformed—has significant applications in engineering, ranging from deployable space solar arrays [1] to medical stents and self-assembling robots [2], [3]. We address this by finding the valid kinematic configurations of a crease pattern. This is useful to engineers who need to verify if a theoretical crease pattern can reach a specific folded state geometrically without tearing or bending the panels.

While existing tools like TreeMaker [4] focus on crease pattern design, our project focuses on resolving the geometric constraints of the folding process. Rather than simulating dynamic motion over time, our approach iteratively solves for the dihedral angles that satisfy closure constraints, allowing us to generate valid configurations that progress from an initial condition toward a folded shape. This reduces physical prototyping costs for deployable structures. Similar techniques have been applied in robotics by Balkcom [5], though our approach focuses on the general nodal analysis of arbitrary vertex constraints.

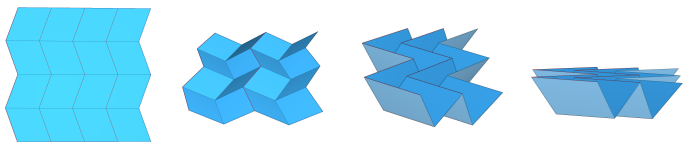

Fig. 1: Folding sequence of the Miura-ori tessellation generated by our simulator, satisfying closure constraints at every step [1].

II. Problem Formulation

We analyze the problem using a nodal analysis framework adapted from Tachi [2]. The origami pattern is treated as a graph where vertices are nodes and creases are edges.

State Vector (x): The state consists of the fold angles for all internal, non-driven edges. These are the unknowns solved for by the numerical method.

$$x = [\rho_1, \rho_2, \dots, \rho_{N_{free}}]^T \tag{1}$$

Fig. 2: Example use of rigid origami: deployable solar arrays.

Parameters (p): The parameters define the static topology and geometry of the mesh.

$$p = [\mathcal{V}^0, \mathcal{E}, \{\alpha_{ij}\}, \{\ell_{ij}\}]^T \tag{2}$$

where $\mathcal{V}^0 \in \mathbb{R}^{N_v \times 3}$ represents the reference vertex configuration, $\mathcal{E}$ represents the edge connectivity set, α_{ij} are the pre-computed sector angles between creases on a flat face, and ℓ_{ij} are crease lengths.

Input (u): The input vector contains the fold angles of the "driven" creases, which are controlled by the user or an actuator over time t.

$$u(t) = [\rho_{driven,1}(t), \rho_{driven,2}(t), \dots]^T \tag{3}$$

Governing Equation (f): For the structure to be geometrically closed, the product of rotation matrices around any internal vertex must equal the identity matrix I_3. This forms our residual function $f(x, p, u)$.

For a vertex k with incident creases $j = 1 \dots M$ ordered counter-clockwise, the closure constraint is:

$$f_k(x, p, u) = \left(\prod_{j=1}^{M} R_{local}(\rho_j, \alpha_j) \right) - I_3 = 0 \tag{4}$$

To explicitly distinguish between state variables and driving inputs, the rotation term R_{local} is defined piecewise:

$$R_{local}(\rho_j, \alpha_j) = \begin{cases} R_{\hat{x}}(\rho_j \in u) R_{\hat{z}}(\alpha_j) & \text{if edge } j \text{ is driven} \\ R_{\hat{x}}(\rho_j \in x) R_{\hat{z}}(\alpha_j) & \text{if edge } j \text{ is free} \end{cases} \tag{5}$$

The system is in a valid configuration when $f(x, p, u) = 0$.

Output (y): While the solver computes angles, the physical quantity of interest is the 3D shape. The output y consists of the Cartesian coordinates of the vertices, derived from the state x via a geometric reconstruction function $g(x, p)$ (implemented as a Breadth-First Search traversal from a pinned root face).

$$y = g(x, p) = [V_{x1}, V_{y1}, V_{z1}, \dots, V_{zN}]^T \tag{6}$$

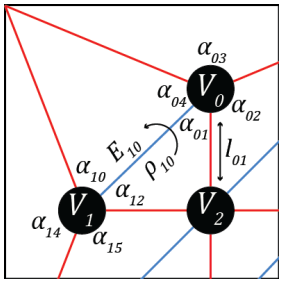

Fig. 3: Top left corner of a bird base, labeling vertices (V_i), angles (a_{ij}), and edges (E_{ij}).

III. Fundamental Numerical Methods

To solve the governing equation $f(x, p, u) = 0$, we utilized a Damped Newton-Raphson solver. Standard Newton iteration is often unstable for this application because the "flat sheet" configuration represents a singular point where the Jacobian becomes ill-conditioned. This leads to bifurcation, where the solver might jump between "mountain" and "valley" solutions, breaking frame-to-frame coherence. To resolve this, we implemented a step-limited damping scheme (formally a Trust Region approach, as described in Alg. 1) [6]. This ensures that the iterative updates remain within the local basin of attraction of the previous time step, preserving geometric validity.

Algorithm 1 Damped Newton-Raphson Solver

Require: Initial guess $x^{(0)}$, Input u, Tol ϵ
Ensure: State x satisfying $||f(x,u)||_\infty < \epsilon$
  function NewtonSolver($x^{(0)}, u$)
    $k \leftarrow 0$
    $r \leftarrow f(x^{(k)}, u)$
    while $||r||_\infty > \epsilon$ **and** $k < k_{max}$ **do**
      $J \leftarrow$ SparseJacobian($x^{(k)}$) ▷ See Sec. IV
      $\Delta x \leftarrow \text{LSMR}(J, -r)$ ▷ Iterative Solve
      $\delta_{max} \leftarrow ||\Delta x||_\infty$
      $\lambda \leftarrow 1.0$
      if $\delta_{max} > \Delta_{max}$ **then**
        $\lambda \leftarrow \Delta_{max}/\delta_{max}$ ▷ Damping
      $x^{(k+1)} \leftarrow x^{(k)} + \lambda \Delta x$
      $r \leftarrow f(x^{(k+1)}, u)$
      $k \leftarrow k + 1$
    return $x^{(k)}$

IV. The Technical Challenge

Our specific technical challenge was the efficient assembly of the Jacobian matrix for large-scale tessellations.

In rigid origami, the geometric constraints are topologically local: a fold angle at edge i only affects the closure constraints of the vertices it touches. Consequently, the Jacobian matrix $J \in \mathbb{R}^{3N_v \times N_e}$ is over 98% sparse. A naive dense implementation requires $O(N^2)$ memory and time to assemble, which becomes prohibitive for grids larger than 12×12.

To address this, we implemented a custom **Sparse Coordinate (COO) Assembly** routine [7]. Instead of iterating over matrix rows/columns, we iterate only over the geometric constraints. For each constraint, we compute the local gradients analytically by differentiating the product of rotation matrices using Rodrigues' rotation formula [8]. This allows us to construct the sparse matrix in $O(N)$ time. Furthermore, this structure allows for massive parallelization (using Numba JIT), as each constraint calculation is independent. The assembly logic is detailed in Alg. 2.

Algorithm 2 Sparse Jacobian Assembly (COO Format)

Require: State x, Constraints set C
Ensure: Arrays Val, Row, Col for Sparse Matrix J
  function SparseJacobian(x, C)
    $k \leftarrow 0$ ▷ Global index pointer
    for each constraint $c_i \in C$ **do** ▷ Parallelizable
      $Edges \leftarrow c_i.edges$
      $Angles \leftarrow c_i.sector\_angles$
      for $j \in 0 \dots \text{len}(Edges)$ **do**
        $e_{idx} \leftarrow Edges[j]$
        $\alpha \leftarrow Angles[j]$
        $\rho \leftarrow x[e_{idx}]$
        $\partial R/\partial \rho \leftarrow \text{ChainRuleDeriv}(\rho, \alpha)$
        $Val[k:k+3] \leftarrow \partial R[0:3]$
        $Row[k:k+3] \leftarrow (i \times 3) + [0,1,2]$
        $Col[k:k+3] \leftarrow e_{idx}$
        $k \leftarrow k + 3$
    $J \leftarrow$ ConstructSparse(Val, Row, Col)
    return J

V. Results

The simulation results were generated using Python 3.10 and the NumPy and SciPy libraries [9], [10]. All experiments were conducted on a laptop CPU (Apple M3 Max).

A. Convergence Experiment

We executed four different solvers—Forward Euler, Undamped Newton-Raphson, Homotopy Continuation, and our Damped Newton-Raphson method—on a fixed 6×6 Waterbomb tessellation ($N = 345$ DOFs) with diagonal folds driven to an angle of 30° [11], [12]. Fig. 4 plots the norm of the residual vector $||f(x)||$ against the iteration count for each method.

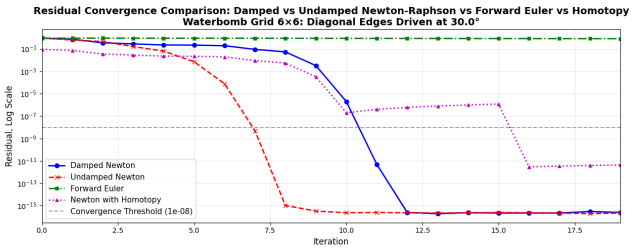

Fig. 4: Residual history for three solver methods on a 6x6 Waterbomb grid. The plot displays the log-scale residual norm vs. iteration count.

B. Scalability Experiment

We measured the time-to-convergence for two different Jacobian implementations—Dense and Sparse—across a range of grid sizes from 2×2 (43 DOFs) to 20×20 (3670 DOFs). Fig. 5 plots the execution time in seconds against the number of edges (Degrees of Freedom) for both implementations. A crossover in performance occurs between the 4×4 grid (158 DOFs) and the 5×5 grid (242 DOFs).

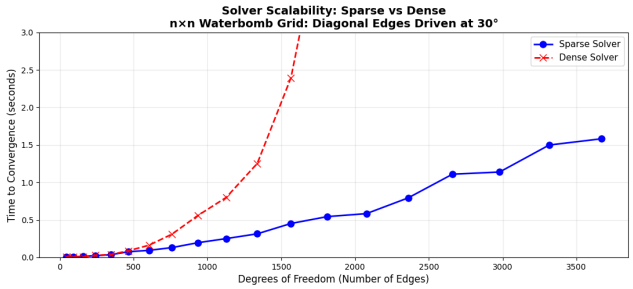

Fig. 5: Wall-clock execution time vs. Degrees of Freedom (Edges). Data points represent grid sizes increasing from 2×2 to 20×20.

C. Failure Analysis

To quantify the solver limitations, we tested it against the traditional "Flapping Bird" model, which, unlike the Miura-Ori or Waterbomb patterns, requires the paper to bend to be solved. Table I compares the convergence performance across these models. While the solver successfully converged for the rigid tessellations (rows 1-3) with residuals below 10^{-12}, it failed to converge for the Flapping Bird (row 4). The solver stagnated at a residual of 4.02×10^{-1}, correctly identifying that no geometric state exists where all rigid constraints are satisfied.

TABLE I: Convergence Time Comparison & Failure Analysis

System	DOFs	Time (s)	Iters	Residual	Status
1×1 Miura-Ori	12	0.0029	3	5.81e-17	Success
6×6 Miura-Ori	312	0.0023	3	3.42e-16	Success
6×6 Waterbomb	345	0.0231	11	3.72e-12	Success
Flapping Bird*	59	0.0369	74	4.02e-01	Failed

*Model is not rigidly solvable (requires deformation).

VI. Technical Discussion

A. Numerical Stability Analysis

As illustrated in Fig. 4, the Forward Euler baseline (Green) proved ill-formed for this nonlinear problem, hardly decreasing the residual at all over the 30-iteration window.

In contrast, the Undamped Newton-Raphson method (Red) achieved the fastest algebraic convergence, reducing the residual below 10^{-8} in the fewest iterations. However, despite this rapid convergence, practical testing revealed that the method failed to maintain frame-to-frame coherence. Near the "flat sheet" singularity, the unconstrained step sizes caused the solver to jump between bifurcation branches (e.g., inverting "mountain" and "valley" assignments) from one frame to the next.

The Damped Newton method (Blue) resolved this issue, as the solver was constrained to converge to a solution within the basin of attraction of the previous frame. Although this damping necessitated a higher iteration count compared to the undamped method, it was essential for preserving the geometric validity of the folding sequence.

Finally, we benchmarked Homotopy Continuation (Purple). The plot displays its characteristic "sawtooth" pattern as it traverses the parameter space. While theoretically robust, it required significantly more iterations to deliver the same stability achieved by our Damped Newton method. Consequently, we determined that the computational overhead of Homotopy was unjustified.

B. Computational Scalability

The scaling trends in Fig. 5 quantify the precise cost-benefit threshold of sparse matrix operations. The crossover point between 158 and 242 DOFs indicates the threshold where the overhead of sparse indexing is outweighed by the $O(N^3)$ factorization cost of the dense solver. For the 6×6 grid, the Jacobian is over 98% sparse. The near-linear $O(N^{1.2})$ scaling of the sparse implementation confirms that the LSMR solver [13], selected for its superior convergence monotonicity over LSQR, successfully exploits this topological locality, avoiding the cubic bottleneck of direct factorization.

C. Implementation Optimizations

The absolute performance values in Fig. 5 (sub-second convergence for > 1000 DOFs) reflect the impact of JIT compilation. While the algorithmic scaling is determined by the solver choice (Dense vs. Sparse), the magnitude of the execution time is dominated by the assembly overhead.

VII. Ethics & Limitations

There are also multiple limitations within the solver that users should be aware of. While this solver is designed for mechanical applications, the solver represents an idealized system, where each face is perfectly rigid, and has 0 thickness. Additionally, as shown with the Flapping Bird, it cannot model non-rigid origami. The solver also does not account for any internal stresses or deformation and as such structures created in the solver are not guaranteed to have any level of structural integrity. Rigid origami can also be applied within a wide range of industries and disciplines, including biomedical and aerospace engineering [14]. As a result, while this tool does not directly create harmful technologies, it could facilitate the development of dangerous technologies. We encourage users to remain conscious of the broader impact of their work.

Lastly, there is an artistic, cultural and creative nature to origami, that is also important to consider and value. The authorship and creative traditions associated with origami should not be overshadowed by computational modeling. Crease patterns often exist in a gray area with respect to intellectual property protection, and we do not seek to enable the appropriation or uncredited replication of creative works [15].

VIII. Conclusions

Ultimately, this work demonstrates that rigid origami can be modeled as a low-latency, real-time simulation using a Newton–Raphson–based solver. When compared to explicit methods such as Forward-Euler, which failed to reduce residuals significantly over 30 iterations, the Newton–Raphson approach handles the nonlinear constraints effectively. Among the implicit solvers evaluated, the Undamped Newton method achieved the fastest algebraic convergence. However, the Damped Newton variant, despite requiring slightly more iterations, proved necessary to prevent bifurcation near the singular flat-sheet state, making it the only reliable choice for interactive use.

Regarding computational scalability, our Sparse Coordinate (COO) assembly routine successfully bypassed the $O(N^3)$ bottleneck of dense factorization. We identified a performance crossover point between 158 and 242 Degrees of Freedom (DOFs); beyond this threshold, the sparse solver exhibits near-linear scaling following an empirical power law of approximately $O(N^{1.2})$. This efficiency enables sub-second convergence for complex tessellations, validating the system's suitability for real-time, interactive design applications.

REFERENCES

[1] K. Miura, "Method of packaging and deployment of large membranes in space," 1985.

[2] T. Tachi, "Simulation of rigid origami," *Origami 4: Fourth International Meeting of Origami Science, Mathematics, and Education*, vol. 4, 08 2009.

[3] M. Schenk and S. D. Guest, "Geometry of miura-folded metamaterials," *Proceedings of the National Academy of Sciences*, vol. 110, no. 9, pp. 3276–3281, 2013.

[4] R. J. Lang, "Treemaker," 2015. Accessed: 2025-12-10.

[5] D. J. Balkcom, *Robotic Origami Folding*. PhD thesis, Carnegie Mellon University, Robotics Institute, Pittsburgh, PA, USA, August 2004. Ph.D. thesis / Technical Report.

[6] J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Philadelphia, PA: SIAM, classics ed., 1996. See Section 6.4: The Trust Region Approach.

[7] Y. Saad, *Iterative Methods for Sparse Linear Systems*. SIAM, 2nd ed., 2003.

[8] O. Rodrigues, "Des lois géométriques qui régissent les déplacements d'un système solide dans l'espace, et de la variation des coordonnées provenant de ces déplacements considérés indépendamment des causes qui peuvent les produire," *Journal de Mathématiques Pures et Appliquées*, vol. 5, pp. 380–440, 1840.

[9] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[10] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[11] L. Daniel, "2.091/6.7300/16.910 Introduction to Numerical Simulation." MIT, Graduate Course, Fall 2025.

[12] J. L. Silverberg, A. A. Evans, L. McLeod, R. C. Hayward, T. Hull, C. D. Santangelo, and I. Cohen, "Using origami design principles to fold reprogrammable mechanical metamaterials," *Science*, vol. 345, no. 6197, pp. 647–650, 2014.

[13] D. C.-L. Fong and M. A. Saunders, "Lsmr: An iterative algorithm for sparse least-squares problems," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2950–2971, 2011.

[14] M. Meloni, J. Cai, Q. Zhang, D. Sang-Hoon Lee, M. Li, R. Ma, T. E. Parashkevov, and J. Feng, "Engineering origami: A comprehensive review of recent applications, design methods, and tools," *Advanced Science*, vol. 8, no. 13, p. 2000636, 2021.

[15] C. S. Kim, "The art and craft of mathematical expression: Computational origami and the politics of creativity," *Osiris*, vol. 38, 2023.