

Supporting Information for "Out-of-Distribution Inverse Design of Elastic Networks with Differentiable Graph Neural Network Molecular Dynamics"

Sergey A. Shteingolts¹, Salman N. Salman¹, Ron Levie², and Dan Mendels^{*1}

¹The Wolfson Department of Chemical Engineering, Technion - Israel Institute of Technology, Haifa, 32000, Israel

²Faculty of Mathematics, Technion – Israel Institute of Technology, Haifa 32000, Israel

*Email: danmendels@technion.ac.il

1 Static Property Predictor

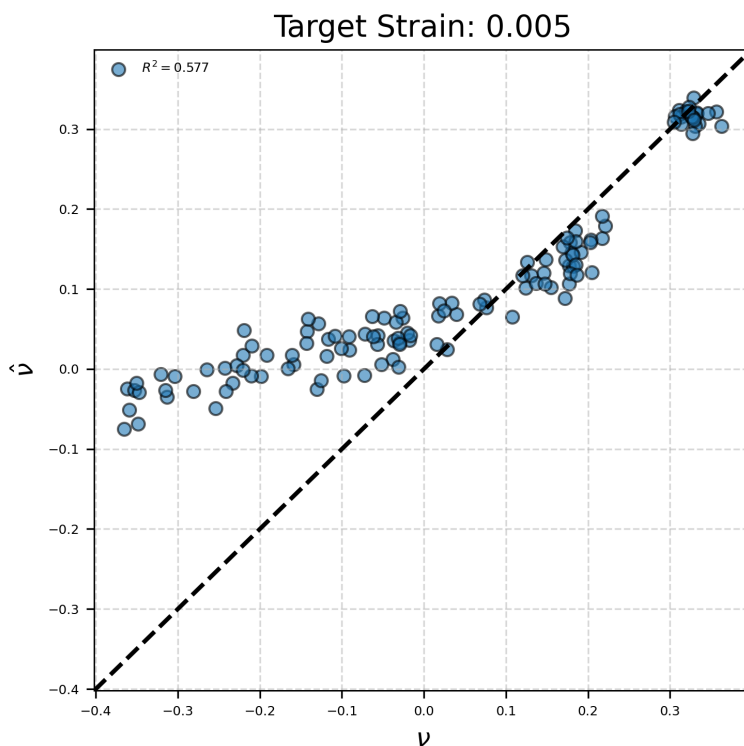


Figure S1: Performance of a static structure-property model trained on data with $\nu \geq 0.1$.

2 Disordered Elastic Networks and Data Generation

2.1 Elastic network generation

Disordered elastic networks (DENs) were constructed in a two-step process.[1] First, N soft disks of four different types and diameters were randomly placed in equal proportions within a periodic simulation box and relaxed to a local energy minimum using a standard jamming algorithm.[2] The contact force between two disks i and j , with respective radii R_i and R_j , within the contact distance $d = R_i + R_j$ is given by

$$\mathbf{F} = (k_n \delta \mathbf{n}_{ij} - m_{eff} \gamma_n \mathbf{v}_n) - (k_t \Delta \mathbf{s}_t + m_{eff} \gamma_t \mathbf{v}_t), \quad (1)$$

where δ is the overlap distance; $\mathbf{n}_{ij}$ is the unit vector along the line connecting the disk centers; k_n and k_t are the normal and tangential elastic constants, respectively; and γ_n and γ_t are the corresponding viscoelastic constants. Furthermore, $m_{eff} = \frac{M_i M_j}{M_i + M_j}$ is the effective mass of disks i and j , whose individual masses are M_i and M_j ; $\mathbf{v}_n$ and $\mathbf{v}_t$ are the normal and tangential components of their relative velocity; and $\Delta \mathbf{s}_t$ is the tangential displacement vector between the disks. Here, k_t , γ_n , and γ_t were set to zero, whereas k_n was set to 1.0. The resulting force between the two disks therefore reduces to

$$F = \delta \mathbf{n}_{ij}. \quad (2)$$

During energy minimization, the system was gradually cooled from $T_{start} = 5 \times 10^{-6}$ to $T_{end} = 1 \times 10^{-6}$. Subsequently, a bead was placed at the center of each disk, and harmonic bonds were introduced between beads corresponding to disks in contact, $r_{ij} \leq d$. Each harmonic bond was assigned an elastic energy of the form

$$E(r_{ij}) = \frac{1}{2} k_{ij} (l - l^0)^2, \quad (3)$$

where k_{ij} is the harmonic bond stiffness, with units of *energy/distance*² (dimensionless with $k_B = 1$, $\sigma = \bar{l}^0$ and $\epsilon = \bar{k}_{ij} \sigma^2$); l^0 is the bond rest length; and l is the distance between the two beads. For each bond, k was set to $1/l^0$. The resulting network was then scanned for “dangling” beads, defined as beads having fewer than three connections to neighboring beads. These beads were removed, and the scanning and removal procedure was repeated iteratively until none remained.

For one of the datasets, Lennard–Jones-type nonbonded interactions were additionally introduced into the elastic networks for both bonded and nonbonded beads. The interaction energy between any two beads i and j , separated by a distance r_{ij} , was defined using the standard Lennard–Jones potential:

$$V_{LJ}(r_{ij}) = 4\epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^6 \right] \quad (4)$$

for $r_{ij} \leq r_c$, and zero otherwise. Here, ϵ denotes the depth of the potential well, σ is the finite separation at which the interparticle potential vanishes, and r_c is the cutoff distance beyond which the interaction is truncated. Following several parameter-testing iterations, the values producing the most interesting compression dynamics were $\epsilon = 0.01$, $\sigma = 1.0$, and $r_c = 1.122$. The latter approximates $2^{1/6}\sigma$, corresponding to the position of the potential minimum and thereby yielding an effectively repulsive interaction.

2.2 Uniaxial compression simulation

Compression simulations of the 2D DENs were carried out using the LAMMPS[3] molecular dynamics simulator. The systems were simulated within the NPT ensemble with anisotropic pressure control (applied to the simulation box in the direction perpendicular to compression), with constant temperature $T = 1 \times 10^{-10}$, $k_B = 1$, damping parameter of 7.1×10^{-3} and time step $\tau = 7.13 \times 10^{-6}$ controlled by the Langevin thermostat.[4] The compression was simulated using the `fix deform` command of LAMMPS with a constant engineering strain rate $s_r = 1 \times 10^{-8}$ and the maximum strain of 5% along the compressed axis. Constant zero-pressure along the non-compressed axis was applied on the simulation box perpendicular to the direction of compression using the Parrinello–Rahman barostat.[5]

2.3 Dataset generation

2.3.1 Global node displacement optimization

Auxetic DENs were generated using a global node optimization strategy.[6] This strategy is based on stochastic gradient descent and seeks to minimize Poisson’s ratio, ν , by modifying the positions of individual nodes. At each optimization epoch n , each coordinate of node i , denoted by $\mathbf{r}_i$, is perturbed by 5×10^{-4} along an axis $\eta \in x, y$. The bond lengths and stiffnesses are subsequently recalculated to maintain the system in its equilibrium configuration, and the gradient of the objective function is evaluated as

$$\frac{\partial(\nu + \mathcal{L}_1 + \mathcal{L}_2)}{\partial r_{i,\eta}}, \quad (5)$$

where $r_{i,\eta}$ denotes the coordinate of node i along axis η , and $\mathcal{L}_1$ and $\mathcal{L}_2$ are constraints on the internode distances and bond angles, respectively. These constraints are defined as

$$\mathcal{L}_1 = 0.1 \sum_{i,j} H(r_{\min} - r_{ij})(r_{ij} - r_{\min})^2, \quad (6)$$

$$\mathcal{L}_2 = 0.01 \sum_j H(\theta_{\min} - \theta_j)(\theta_j - \theta_{\min})^6, \quad (7)$$

where H is the Heaviside step function, while $r_{ij,\min} = 0.3r_{ij}^0$ and $\theta_{\min} = 15^\circ$ are empirically selected minimum values for the harmonic bond distances and bond angles, respectively. Both $\mathcal{L}_1$ and $\mathcal{L}_2$ restrict the accessible optimization space, thereby preventing the network beads from overlapping and limiting the formation of intersecting bonds. To avoid excessively large structural changes and improve the stability of the optimization, the absolute value of each gradient component was clipped at 0.01. Finally, the node positions were updated using the $N \times 2$ gradient matrix obtained in the preceding step:

$$r_{i,\eta}^{n+1} = r_{i,\eta}^n - \lambda \frac{\partial(\nu + \mathcal{L}_1 + \mathcal{L}_2)}{\partial r_{i,\eta}}, \quad (8)$$

where λ is the learning rate. This procedure was repeated until either (a) the desired target value of ν was reached or (b) the number of iterations exceeded a predefined maximum.

2.3.2 Harmonic bond stiffness tuning

Using the minimal differentiable PyTorch-based MD engine described in Section 5, we investigated two distinct optimization schemes, referred to as **continuous** and **discrete** optimization. In both schemes, the learnable parameters are adjustments to the individual bond stiffnesses k_{ij} . These adjustments are denoted by Δk_{ij} , where the directed graph edge represents a harmonic bond between the two network beads i and j .

In the **continuous optimization** scheme, each initial harmonic bond stiffness k_{ij}^0 is adjusted continuously according to

$$k'_{ij} = k_{ij}^0 (1 + C \cdot \tanh(\Delta k_{ij})), \quad (9)$$

where C is a predefined constant that limits the maximum relative change in stiffness.

Conversely, the **discrete optimization** scheme, inspired by a procedure reported elsewhere,[7] restricts the allowed changes in bond stiffness in a manner resembling discrete bond pruning. This is achieved using an edge mask g_{ij} :

$$g_{ij} = \sigma(\beta \Delta k_{ij}^{\text{sym}}), \quad (10)$$

$$k'_{ij} = k_{ij}^0 g_{ij}, \quad (11)$$

where σ is the sigmoid function and β is a hyperparameter controlling its steepness. Due to the undirected nature of the graph, the before each update the symmetric stiffness adjustment $\Delta k_{ij}^{\text{sym}}$ is computed:

$$\Delta k_{ij}^{\text{sym}} = \frac{\Delta k_{ij} + \Delta k_{ji}}{2} \quad (12)$$

At each optimization epoch, a DEN with the updated stiffnesses k' is compressed for a fixed number of MD steps. The Poisson's ratio ν of the network is calculated directly from the changes in the periodic box dimensions between the initial ($t = 0$) and final (t) simulation steps:

$$\nu = -\frac{(L_y^t - L_y^0)/L_y^0}{(L_x^t - L_x^0)/L_x^0}. \quad (13)$$

The optimization objective is to minimize the mean squared error (MSE) between the measured Poisson's ratio and its target value. The total loss function $\mathcal{L}$ is defined as

$$\mathcal{L} = (\nu - \nu_{\text{target}})^2 + \lambda \frac{1}{M} \sum_{ij} (1 - g_{ij}), \quad (14)$$

where M is the total number of unique edges. The second term is a bond removal penalty term used exclusively in the discrete mode, and λ is its corresponding weight. For the continuous mode, λ is set to zero.

Optimization was performed using stochastic gradient descent (SGD) with momentum. Before each optimizer step, the gradients with respect to the stiffness-adjustment parameters Δk_{ij} were clipped to a maximum norm of 1.0. The procedure was continued for a predefined number of epochs or terminated early if the simulated network achieved the target Poisson's ratio ν_{target} .

3 GNN MD Simulator

3.1 Model architecture

The model adopts an Encoder–Processor–Decoder architecture.[8, 9] The encoder maps the input node and edge features of a graph G to latent representations. The temporal history of the node features $\mathbf{x}_i$ is first separated into its spatial components (x and y). Each component is processed using the same axis-shared MLP before the resulting representations are flattened and linearly projected onto a hidden space of dimension 128. The edge features $\mathbf{e}_i$ are mapped directly to the 128-dimensional latent space using an independent MLP, denoted by ε^e .

The processor consists of k message-passing (MP) layers with identical architectures but independent sets of learnable parameters. These layers are applied sequentially, with each layer operating on the output of the preceding layer and incorporating a residual connection. The node embeddings are updated according to

$$\mathbf{x}_i^{(k)} = \mathbf{x}_i^{(k-1)} + \psi^{(k)} \left(\mathbf{x}_i^{(k-1)}, \sum_{j \in \mathcal{N}(i)} \phi^{(k)}(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{ij}) \right), \quad (15)$$

where ψ and ϕ are learnable MLPs employing layer normalization. Here, $\mathbf{x}_i^{(k-1)}$ denotes the embedding of node i in the $(k-1)$ th processor layer, and the summation is taken over all nodes j bonded to node i .

The processor output is mapped by the decoder to a two-dimensional output $\hat{\mathbf{a}}_i \in \mathbb{R}^{N \times 2}$ representing the predicted node accelerations. The decoder is implemented using an additional MLP denoted by δ^x . All MLPs in the architecture, including ε^x , ε^e , ψ , ϕ , and δ^x , were implemented as feed-forward networks with hidden dimensions of 128 and ReLU activation functions.

3.2 Implementation and training details

The model was implemented using PyTorch Geometric 2.7.0[10, 11] and PyTorch 2.11.0.[12, 13] The datasets were generated from uniaxial compression trajectories of DENs spanning a range of Poisson's ratios $\nu \in [-0.6, 0.6]$.

The input data, comprising the node and edge features, and the output data were normalized online and independently during training to have zero mean and unit variance. Specifically, at each forward step, the data were normalized using statistics accumulated from all values observed by the model up to that point.[14, 15] To stabilize the later stages of training, the normalizer statistics were frozen after a predefined epoch,

typically epoch 5. All learnable weights and biases were randomly initialized from a normal distribution with zero mean and a standard deviation of $1/\sqrt{n}$, where n is the number of weights in the corresponding layer. We found that initialization using $\mathcal{N}(0, 1/\sqrt{n})$ generally produced better performance than initialization using the uniform distribution $\mathcal{U}(-1/\sqrt{n}, 1/\sqrt{n})$.

Optimization was performed using the Adam optimizer with zero weight decay. The learning rate was reduced after each epoch using an exponential scheduler with $\gamma = 0.995$. Gradient clipping with a maximum norm of 1.0 was applied together with gradient accumulation. The model was trained in a supervised manner by minimizing the Huber loss[16] between the predicted accelerations $\hat{\mathbf{a}}$ and the normalized ground-truth accelerations $\mathbf{a}$:

$$L_\delta(\mathbf{a}, \hat{\mathbf{a}}) = \begin{cases} \frac{1}{2}(\mathbf{a} - \hat{\mathbf{a}})^2 & \text{if } |\mathbf{a} - \hat{\mathbf{a}}| \leq \delta \\ \delta|\mathbf{a} - \hat{\mathbf{a}}| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases} \quad (16)$$

where δ is the threshold separating the quadratic and linear regimes of the loss.

Training was performed using a multi-step supervision (MST) strategy.[17] For each training sample during the forward pass the simulator model performs a short, continuous rollout. To maintain the training stability, the rollout length increases as a function of training epochs, starting at a single step and scaling up to 10 steps in later epochs. The autoregressive training loop is dynamically coupled with the barostat algorithm.[17] At each rollout step, the simulation box dimensions L_x and L_y are updated to maintain the target pressure in the transverse direction $P_{yy} = 0.0$. Edge attributes and per-particle forces are iteratively recomputed based on the evolving box dimensions before being fed back into the model for the subsequent prediction step.

4 Simulator-coupled barostat

4.1 Implementation

At each timestep t , the periodic box algorithm computes the instantaneous transverse virial P_y^V and kinetic P_y^K pressure from the predicted node positions and pairwise forces. The kinetic term is computed using the ideal gas approximation as follows:

$$P_y^K = \frac{1}{2} \frac{k_B N T}{A}, \quad (17)$$

The virial contribution is calculated by summing the pairwise force contributions from all active interaction edges:

$$P_y^V = \frac{1}{2A} \sum f_{ij,y} r_{ij,y}, \quad (18)$$

where $A = L_x L_y$ is the box area and $f_{ij,y}$ and $r_{ij,y}$ are the y -components of the pairwise force and minimum-image separation vector, respectively. The factor of 1/2 corrects for the bidirectional representation of each physical interaction.

For a harmonic bond, the pairwise force is

$$\mathbf{f}_{ij}^{\text{harm}} = -k_{ij}(r_{ij} - r_{ij}^0), \quad (19)$$

where k_{ij} is the bond stiffness, $r_{ij} = |\mathbf{r}_{ij}|$ is the current bond length, and r_{ij}^0 is its rest length. When nonbonded Lennard–Jones interactions are present, the force associated with an LJ interaction edge is

$$\mathbf{f}_{ij}^{\text{LJ}} = \frac{24\epsilon}{r_{ij}} \left[2 \left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^6 \right], \quad (20)$$

and is zero for $r_{ij} \geq r_c$. Here, ϵ and σ are the Lennard–Jones parameters and r_c is the interaction cutoff. The force used in the virial calculation is therefore computed according to the present interaction types. For networks without nonbonded interactions the virial contains only harmonic contributions.

The difference between the internal pressure P_y and the target pressure P_t generates a thermodynamic driving force F_d . We define a total force F_{total} and apply a frictional damping coefficient γ to prevent

aggressive oscillations:

$$F_d = (P_y - P_t) \cdot L_x \quad (21)$$

$$F_{\text{total}} = F_d - \gamma \cdot v_y^t \quad (22)$$

where v_y^t is the box velocity along the y axis. Box acceleration a_y^t can then be computed using the piston mass W_y :

$$\mathbf{a}_y^t = \frac{F_{\text{total}}}{W_y} \quad (23)$$

Finally, the box velocity is updated and the next-step L_y is calculated as follows:

$$v_y^{t+1} = v_y^t + a_y^t \cdot \Delta t \quad (24)$$

$$L_y^{t+1} = L_y^t + v_y^{t+1} \Delta t \quad (25)$$

4.2 Empirical parameter tuning

Because the periodic box adjustment procedure operates on coarse-grained dynamics (a single GNN step corresponds to 200 MD timesteps), the physical constants typically used for piston mass W_y and damping coefficient γ in classical barostats are not directly applicable. Consequently, the specific values of both γ and W_y must be empirically tuned.

To this end, we utilize the Optuna hyperparameter optimization framework. [18] The piston mass and damping coefficient are defined as functions of the number of particles N and the coarse-grained time stride Δt :

$$W_y = C_{\text{coupling}} \cdot N \cdot (\Delta t)^2 \quad (26)$$

$$\gamma = C_{\text{damping}} \cdot N \cdot \Delta t \quad (27)$$

where C_{coupling} and C_{damping} are the dimensionless scalar factors to be optimized.

To find the optimal values for C_{coupling} and C_{damping} , we minimize a cumulative objective function. The objective function calculates the discrepancy between the predicted transverse box dimension L_y^{pred} and the ground-truth dimension L_y^{GT} using a Huber loss. During the optimization procedure, we utilize ground-truth particle positions at each step to isolate the box adjustment from GNN prediction errors. In our testing, 200 optimization trials were enough to fully converge on optimal values. The search space for C_{coupling} was defined as $[1.0, 15.0]$ and for C_{damping} as $[0.5, 10.0]$. The resulting best parameters were then fixed for all subsequent rollout experiments and MST training regimes.

5 Minimal differentiable MD simulator

We implemented a compact molecular dynamics (MD) engine natively in PyTorch. The simulator performs uniaxial compression along the x -axis while allowing the transverse box dimension L_y to fluctuate dynamically in response to the internal pressure. The simulation is differentiable with respect to continuous model parameters for a fixed interaction graph and realization of the stochastic noise.

Compression along the x -direction is imposed deterministically at a constant engineering strain rate $\dot{\epsilon}$. At simulation step n , corresponding to the physical time $t_n = n\Delta t$, the longitudinal box dimension is defined as

$$L_x(t_n) = L_x(0)(1 - \dot{\epsilon}t_n). \quad (28)$$

Particle coordinates along the x -axis are affinely rescaled by the ratio of the new and previous box dimensions. In addition to this affine deformation, the particle positions are propagated using their half-step velocities, thereby allowing nonaffine motion along both spatial directions.

The transverse box dimension is coupled to a dynamic barostat that seeks to maintain a target pressure P_{target} . The instantaneous transverse pressure is evaluated as

$$P_{yy} = \frac{1}{V} \left(\sum_{i=1}^N m v_{i,y}^2 + \frac{1}{2} \sum_{(i,j) \in \mathcal{E}_{\text{dir}}} F_{ij,y} r_{ij,y} \right), \quad (29)$$

where $V = hL_xL_y$ is the effective simulation volume, with a fixed out-of-plane thickness $h = 0.2$. The second sum is evaluated over the directed interaction edges $\mathcal{E}_{\text{dir}}$. The factor of $1/2$ removes the double counting that arises when each physical interaction is represented by two oppositely directed edges. Here, m is the particle mass, $v_{i,y}$ is the transverse particle velocity, and $F_{ij,y}$ and $r_{ij,y}$ are the y -components of the pairwise force and minimum-image separation vectors, respectively.

The barostat variable $v_{b,y}$ represents the logarithmic strain rate of the transverse box rather than the ordinary velocity of its boundary. Its acceleration is defined as:

$$a_{b,y} = \frac{(P_{yy} - P_{\text{target}})V}{W_y}, \quad (30)$$

where the piston inertia is defined as

$$W_y = (2N - 3)k_B T_b \tau_b^2. \quad (31)$$

Here, α is an inertia prefactor, T_b is the barostat temperature, and τ_b is the barostat damping time. The piston degree of freedom is coupled to a three-variable Nosé–Hoover chain with mass parameter $Q_b = k_B T_b \tau_b^2$. The barostat and thermostat-chain variables are integrated using five subcycles during each half-step. The accumulated barostat scaling is applied to the transverse particle velocities, and the box dimension is propagated according to

$$L_y(t + \Delta t) = L_y(t) \exp\left(v_{b,y}^{1/2} \Delta t\right), \quad (32)$$

where $v_{b,y}^{1/2}$ denotes the barostat rate obtained after the predictor half-step. The particle equations of motion are integrated using a stochastic velocity-Verlet-like scheme with a default timestep $\Delta t = 0.01$.

A complete integration step proceeds as follows:

1. The deterministic forces and virial pressure are evaluated using the current particle coordinates, box dimensions, and interaction graph.
2. The barostat velocity and its three Nosé–Hoover-chain variables are advanced through five predictor subcycles using the current pressure difference. The resulting metric scaling factor is accumulated over these subcycles.
3. A Gaussian random force is sampled, and the particle velocities are advanced by $\Delta t/2$ using the deterministic forces, Langevin drag, stochastic force, and transverse metric scaling.
4. The new longitudinal box length is evaluated from the prescribed engineering-strain schedule, while the transverse box length is propagated exponentially using the half-step barostat rate. Particle coordinates in both directions are updated by combining affine box rescaling with displacement according to the half-step velocities.
5. The deterministic forces and transverse pressure are recalculated using the updated coordinates and box dimensions.
6. The barostat and Nosé–Hoover-chain variables are advanced through five corrector subcycles using the updated pressure.
7. Particle velocities are advanced through the final $\Delta t/2$ using the updated deterministic forces and Langevin drag, together with the same random-force realization used in the predictor step.

After each full integration step, the graph representation is refreshed. Bond stiffnesses are retained, whereas distance-dependent edge attributes and, when applicable, nonbonded interaction edges are updated. Every 100 MD steps, both the center-of-mass translational velocity and the net rigid-body rotational velocity are removed.

6 Inference-Time Physics-based Optimization

6.1 Physics constraints

The net force on each particle $\mathbf{F}_i$ is derived from the harmonic bond stiffness $k_{ij} = 1/l_{ij}^0$:

$$\mathbf{F}_i = \sum_{j \in \mathcal{N}(i)} -k_{ij}(l_{ij} - l_{ij}^0) \frac{\mathbf{r}_{ij}}{l}, \quad (33)$$

where l_{ij} and l_{ij}^0 are the current and rest bond lengths between the beads i and j , while $\mathbf{r}_{ij}$ is the bond vector. When nonbonded Lennard–Jones interactions are present, the total force is the sum of the harmonic and nonbonded contributions:

$$\mathbf{F}_i = \mathbf{F}_i^{\text{harm}} + \mathbf{F}_i^{\text{LJ}} \quad (34)$$

The Lennard–Jones contribution is given by

$$\mathbf{F}_i^{\text{LJ}} = \sum_{j \in \mathcal{N}_{\text{LJ}}(i)} \frac{24\epsilon}{r_{ij}} \left[2 \left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^6 \right] \quad (35)$$

and is zero for $r_{ij} \geq r_c$. Here, $r_{ij} = |\mathbf{r}_{ij}|$, ϵ and σ are the Lennard–Jones parameters, r_c is the interaction cutoff, and $\mathcal{N}_{\text{LJ}}(i)$ denotes the set of nonbonded neighbors of particle i within the cutoff r_c .

The force loss component penalizes any non-zero forces along the transverse axis y across the N particles and is defined as:

$$\mathcal{L}_{\text{force}} = \frac{1}{N} \sum (F_{i,y})^2 \quad (36)$$

The potential energy constraint $\mathcal{L}_U$ is defined as a sum of individual harmonic bond energies $E(\mathbf{r}_{ij})$ (Equation 3):

$$\mathcal{L}_U = \sum E(\mathbf{r}_{ij}) \quad (37)$$

The pressure loss term is defined as the squared virial stress:

$$\mathcal{L}_{\text{pressure}} = (\sigma_{yy})^2 \quad (38)$$

where σ_{yy} is computed using the pairwise harmonic forces $\mathbf{f}_{ij,y}$ and edge vectors $\mathbf{r}_{ij,y}$ over area A :

$$\sigma_{yy} = \frac{1}{2A} \sum f_{ij,y} r_{ij,y} \quad (39)$$

6.2 Empirical weights tuning

Maximizing ITPO performance requires carefully tuning the individual weights α_i of the physics constraints:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{anchor}} + \sum_i \alpha_i \mathcal{L}_i^{\text{phys}}, \quad (40)$$

This presents a practical challenge. Because our objective is OOD generalization to unseen dynamics (typically to $\nu < 0.1$, while the training data is characterized by $\nu \geq 0.1$), ITPO weights cannot be found using these data. However, if the GMDS has already been trained using the MST strategy on data extending down to $\nu = 0.1$, it can sometimes perform *too well*. Because of this, the hyperparameter search converges on virtually zero physics weights $\alpha_i \approx 0$, as the GMDS predictions are already plenty accurate. To ensure the physics constraints remain active, we utilize a staggered training and tuning pipeline:

1. First, a GMDS model is trained strictly on *in-distribution* data characterized by $\nu > 0.2$ using the MST strategy.
2. Next, a search for ITPO physics constraint weights α_i is performed using the intermediate finetuning data with $\nu \in [0.1, 0.2]$. Because the GMDS is not familiar with this dynamics, optimizer relies on the physics constraints to minimize the ITPO objective function, yielding non-zero constraint weights α_i .

3. Finally, the model used for the search is further trained on the intermediate data with $\nu \in [0.1, 0.2]$ using a MST strategy.

To find individual physics constraint weights α_i we employ the Optuna hyperparameter optimization framework.[18] The objective function is defined through the coefficient of determination (R^2) between the predicted and ground-truth Poisson’s ratios ν over a 50-step rollout. In total, up to 5 different parameters can be optimized:

1. Refinement iterations N_{iter} : The number of gradient descent steps performed at each rollout timestep, sampled uniformly in the range $[5, 40]$.
2. Learning rate η for the Adam optimizer, sampled on a log-uniform scale between 10^{-8} and 10^{-3} .
3. The scalar coefficients for the force α_{force} , potential energy α_{energy} , and virial pressure α_{pressure} terms. These are sampled log-uniformly between 10^{-7} and 10^{-1} to account for the varying orders of magnitude of the underlying physical quantities.

In practice, we found it beneficial to leave the number of refinement iterations N_{iter} constant, since it plays a similar role to the learning rate η . Depending on the dataset, N_{iter} was set to 20 or 30, which presents a compromise between the refinement accuracy and the computational penalty incurred due to ITPO. Overall, 500 Optuna optimization trials were performed per dataset.

6.3 Gradient computation via the Implicit Function Theorem

A significant challenge for the proposed end-to-end differentiable inverse design framework is backpropagating the macroscopic property gradients through the unrolled inner ITPO loop. Directly tracking gradients through inner optimization steps at every rollout timestep is computationally intractable. To circumvent this, we utilize the Implicit Function Theorem (IFT) to analytically compute the approximate implicit gradients at the local minimum of the ITPO loss function.

Assuming the ITPO reaches a stable minimum at $\mathbf{a}_{\text{nn}}^*$, the gradient of the total loss with respect to the initial GNN prediction $\hat{\mathbf{a}}_{\text{nn}}$ can be evaluated implicitly. Let $H = \nabla_{\mathbf{a}_{\text{nn}}^*}^2 \mathcal{L}_{\text{total}}$ represent the Hessian matrix of the loss evaluated at the refined state. Applying IFT requires calculating the product of the inverse Hessian and the upstream gradient vector, $\mathbf{v}$. We employ a matrix-free Hessian-vector product (HVP):

$$\text{HVP}(\mathbf{v}) = \nabla_{\mathbf{a}_{\text{nn}}^*} ([\nabla_{\mathbf{a}_{\text{nn}}^*} \mathcal{L}_{\text{total}}] \cdot \mathbf{v}) \quad (41)$$

To solve the linear system $H\mathbf{x} = \mathbf{v}$ for the inverse HVP, we utilize a conjugate gradient solver. Finally, the implicit gradient must account for the mixed partial derivative between the refined and initial accelerations. Given that the anchor loss is defined as a mean squared error:

$$\mathcal{L}_{\text{anchor}} = \frac{1}{N} \|\mathbf{a}_{\text{nn}}^* - \hat{\mathbf{a}}_{\text{nn}}\|^2 \quad (42)$$

where N is the total number of particles, the mixed derivative simplifies to a scaled identity operator. Consequently, the analytical gradient mapped back to the simulator model output is computed as:

$$\nabla_{\hat{\mathbf{a}}_{\text{nn}}} \mathcal{L} = \frac{2}{N} (H^{-1} \mathbf{v}) \quad (43)$$

References

- (1) Rocks, J. W.; Pashine, N.; Bischofberger, I.; Goodrich, C. P.; Liu, A. J.; Nagel, S. R. Designing allostery-inspired response in mechanical networks. *Proceedings of the National Academy of Sciences* **2017**, *114*, 2520–2525.
- (2) Liu, A. J.; Nagel, S. R. The jamming transition and the marginally jammed solid. *Annual Review of Condensed Matter Physics* **2010**, *1*, 347–369.

- (3) Thompson, A. P.; Aktulga, H. M.; Berger, R.; Bolintineanu, D. S.; Brown, W. M.; Crozier, P. S.; in 't Veld, P. J.; Kohlmeyer, A.; Moore, S. G.; Nguyen, T. D.; Shan, R.; Stevens, M. J.; Tranchida, J.; Trott, C.; Plimpton, S. J. LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Computer Physics Communications* **2022**, *271*, 108171.
- (4) Schneider, T.; Stoll, E. Molecular-dynamics study of a three-dimensional one-component model for distortive phase transitions. *Physical Review B* **1978**, *17*, 1302–1322.
- (5) Parrinello, M.; Rahman, A. Polymorphic transitions in single crystals: A new molecular dynamics method. *Journal of Applied Physics* **1981**, *52*, 7182–7190.
- (6) Shen, M.; Reyes-Martinez, M. A.; Powell, L. A.; Iadicola, M. A.; Sharma, A.; Byléhn, F.; Pashine, N.; Chan, E. P.; Soles, C. L.; Jaeger, H. M.; de Pablo, J. J. An autonomous design algorithm to experimentally realize three-dimensionally isotropic auxetic network structures without compromising density. *npj Computational Materials* **2024**, *10*, 113.
- (7) Reid, D. R.; Pashine, N.; Wozniak, J. M.; Jaeger, H. M.; Liu, A. J.; Nagel, S. R.; de Pablo, J. J. Auxetic metamaterials from disordered networks. *Proceedings of the National Academy of Sciences* **2018**, *115*, E1384–E1390.
- (8) Hamrick, J. B.; Allen, K. R.; Bapst, V.; Zhu, T.; McKee, K. R.; Tenenbaum, J. B.; Battaglia, P. W. Relational inductive bias for physical construction in humans and machines. **2018**.
- (9) Pfaff, T.; Fortunato, M.; Sanchez-Gonzalez, A.; Battaglia, P. W. Learning Mesh-Based Simulation With Graph Networks. *ICLR 2021 - 9th International Conference on Learning Representations* **2021**, 1–18.
- (10) Fey, M.; Lenssen, J. E. Fast Graph Representation Learning with PyTorch Geometric. **2019**, 1–9.
- (11) Fey, M.; Sunil, J.; Nitta, A.; Puri, R.; Shah, M.; Stojanović, B.; Bendias, R.; Barghi, A.; Kocijan, V.; Zhang, Z.; He, X.; Lenssen, J. E.; Leskovec, J. PyG 2.0: Scalable Learning on Real World Graphs. *Proceedings of Temporal Graph Learning Workshop, SIGKDD International Conference on Knowledge Discovery and Data Mining 2025 (TGL Workshop, KDD 2025)* **2025**, *1*.
- (12) Paszke, A. et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*. *NeurIPS* **2019**, 8026–8037.
- (13) Ansel, J. et al. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ACM: New York, NY, USA, 2024; Vol. 2, pp 929–947.
- (14) Sanchez-Gonzalez, A.; Godwin, J.; Pfaff, T.; Ying, R.; Leskovec, J.; Battaglia, P. W. Learning to Simulate Complex Physics with Graph Networks. *37th International Conference on Machine Learning, ICML 2020* **2020**, *PartF16814*, 8428–8437.
- (15) Kumar, K.; Vantassel, J. GNS: A generalizable Graph Neural Network-based simulator for particulate and fluid modeling. **2022**.
- (16) Huber, P. J. In *Breakthroughs in Statistics: Methodology and Distribution*, Kotz, S., Johnson, N. L., Eds.; Springer New York: 1992, pp 492–518.
- (17) Shteingolts, S. A.; Salman, S. N.; Mendels, D. Enabling Structure-Only Initialization and Out-of-Distribution Generalization in GNN-based Molecular Dynamics Simulators. **2026**.
- (18) Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; Koyama, M. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM: New York, NY, USA, 2019, pp 2623–2631.