Algorithmic Improvements to Noisy PES Optimization: Newton-Raphson Minimization and Gradient Ascent Deflation Saddle-Point Search

Benchmarked on Transition1x with DFTB0, 2 Å starting noise

Mehmet Efe Moozdincerler
Guzik Group

February 2026

Contents

Executive Summary

This document describes and analyses a series of algorithmic improvements applied to two PES optimizers running under realistic numerical noise:

1. **Newton-Raphson (NR) energy minimization** — finds local minima.
2. **Gradient Ascent Deflation (GAD)** — finds index-1 saddle points (transition states, TS).

Both algorithms operate on the DFTB0 potential via SCINE, starting from the Transition1x midpoint geometry perturbed by 2 Å of random noise — a deliberately hard initialization meant to stress-test robustness.

The improvements below are listed in order of empirical impact, from most to least. NR minimization improved from ~45% convergence to ~ **99**% convergence (average 455 steps). GAD TS search reached **100% success rate** on 30/30 samples under the optimal configuration.

Key improvements, ordered by impact:

1. **Newton step in the vibrational pseudoinverse subspace** — replacing gradient descent with a full second-order step while explicitly throwing out near-zero modes. (§2)
2. **Aggressive eigenvalue filtering** ($|\lambda| < \tau$, hard discard, not clamping) — eliminates numerical noise rattling modes entirely. (§3)
3. **Adaptive Trust Region (TR)** with the Cauchy quality ratio $\rho = \delta E_{\text{actual}} / \delta E_{\text{pred}}$ — allows massive steps when the quadratic model is accurate and aggressively shrinks/rejects when it is not. (§4)
4. **Mode tracking** (GAD-specific) — tracking the lowest eigenvector v_1 across steps rather than always re-picking the instantaneous lowest; yields a 10+ percentage-point improvement in TS success rate and 2× speedup. (§5)
5. **Maximum atom displacement cap** $\Delta_{\max} = 1.3$ Å — allows confident long-range steps rather than artificially stalling the optimizer. (§6)
6. **Eckart projection** (mandatory) — projects gradient and guide vector onto the vibrational subspace to prevent translation/rotation drift. (§7)
7. **Hessian purification** (negative result) — enforcing translational sum rules had no measurable effect; omitted to save compute. (§8)

1 Background and Experimental Setup

1.1 Algorithms

The two algorithms share a common structure:

Newton-Raphson minimization.

$$x_{k+1} = x_k - H(x_k)^{-1}\,\nabla E(x_k) \;=\; x_k + H(x_k)^{-1}\,F(x_k),$$

where $F = -\nabla E$ are the atomic forces and H is the $3N \times 3N$ mass-unweighted Hessian. Convergence: zero negative vibrational eigenvalues (Morse index 0, i.e., a true minimum).

Gradient Ascent Deflation (GAD).

$$\tilde{F}_{\mathrm{GAD}}(x) = F(x) + 2\langle F(x),\, v_1\rangle v_1,$$

where v_1 is the lowest vibrational eigenvector of H. The GAD force ascends along v_1 (flips the component parallel to v_1) and descends along all other modes. The update rule in the improved implementation is:

$$x_{k+1} = x_k + H_{\mathrm{vib}}^{-1}\,\tilde{F}_{\mathrm{GAD}}(x_k),$$

i.e., the same pseudoinverse step as NR but applied to the modified gradient. Convergence: exactly one negative vibrational eigenvalue (Morse index 1), i.e., an index-1 TS.

1.2 Dataset and noise

Experiments run on 30 molecules from the Transition1x test set [1] under the DFTB0 functional via SCINE. Starting geometry:

$$x_0 = x_{\mathrm{midpoint}} + \epsilon, \quad \epsilon \sim \mathcal{U}(\text{ball of radius } 2\,\text{Å}).$$

This is a deliberately severe initialization designed to stress-test robustness to numerical noise and large initial displacements from the TS.

2 Newton Step in the Vibrational Pseudoinverse Subspace

2.1 Motivation

The original baseline for both algorithms used a *gradient descent*–like update: $x_{k+1} = x_k + \alpha F(x_k)$ with a fixed or slowly-adaptive step size α. This wastes information — the full Hessian is computed at every step anyway (required for the GAD direction), yet its curvature information was discarded.

2.2 The vibrational pseudoinverse

Let the eigendecomposition of the Eckart-projected Hessian be

$$H_{\mathrm{proj}} = V\Lambda V^T, \quad \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_{3N}).$$

The *vibrational subspace* is defined by the mask $\mathcal{V} = \{i : |\lambda_i| > \tau\}$ (see §3 for the choice of τ). The vibrational pseudoinverse step is:

$$\boxed{\delta x = V_{\mathcal{V}}\ |\Lambda_{\mathcal{V}}|^{-1}\ V_{\mathcal{V}}^T\, g,}$$

where g is the gradient (or modified GAD gradient), $V_{\mathcal{V}}$ are the columns of V indexed by $\mathcal{V}$, and $|\Lambda_{\mathcal{V}}|^{-1}$ uses the absolute values of eigenvalues. The absolute-value trick ensures the step is always a descent/ascent in the intended direction, regardless of the sign of the curvature.

```
# evals, evecs from torch.linalg.eigh(hess_proj)
vib_mask = torch.abs(evals) > tr_threshold       # |lambda| > tau
vib_indices = torch.where(vib_mask)[0]

V_vib   = evecs[:, vib_indices]                  # (3N, k)
lam_vib = evals[vib_indices]                     # (k,)

coeffs     = V_vib.T @ grad                      # project gradient
inv_lam    = 1.0 / torch.abs(lam_vib)            # |lambda|^{-1}
delta_x    = V_vib @ (inv_lam * coeffs)          # back-project
```

Listing 1: Vibrational pseudoinverse step (NR minimization, `baselines/minimization.py`).

```
# gad_flat = f + 2*(f.v)*v  (or Eckart-projected version)
coeffs        = V_vib.T @ gad_flat.to(V_vib.dtype)
inv_lam_abs   = 1.0 / torch.abs(lam_vib)
delta_x       = V_vib @ (inv_lam_abs * coeffs)
step_disp_raw = delta_x.reshape(-1, 3)
```

Listing 2: Same step applied to the GAD modified gradient (`runners/run_gad_baselines_parallel.py`).

2.3 Why this helps

For a locally quadratic energy $E(x) \approx E_0 + g^T x + \frac{1}{2} x^T H x$, the Newton step is the *exact* minimizer in one iteration. Even far from a minimum, the Newton step provides a much better estimate of the step *direction* than gradient descent. Without this, the optimizer is essentially doing steepest-descent with expensive (but wasted) Hessian evaluations.

3 Aggressive Eigenvalue Filtering

3.1 The danger of flat modes

Suppose a vibrational mode v_i has eigenvalue $\lambda_i = 10^{-5}$. The Newton step contribution from this mode is:

$$\delta x_i = \frac{\langle g,\, v_i\rangle}{\lambda_i} \cdot v_i \;=\; \frac{\langle g,\, v_i\rangle}{10^{-5}} \cdot v_i.$$

Even a tiny numerical noise in $\langle g,\, v_i\rangle \sim 10^{-4}$ produces a step of magnitude $10^{-4}/10^{-5} = 10$ along v_i — a completely unphysical, explosive displacement. Such modes arise from:

- Residual translation/rotation not fully removed by Eckart projection.
- Molecular rattling modes at a floppy geometry.
- Numerical noise in the DFTB0 Hessian at 2 Å from the TS.

3.2 Filter vs. clamp

We tested two strategies:

Filter Completely exclude modes with $|\lambda_i| < \tau$ from the pseudoinverse.

Clamp Replace $|\lambda_i|$ by $\max(|\lambda_i|, \tau)$ to enforce a safe gradient descent step along flat modes.

Result: Filtering dominates by a large margin. The flat modes in these TS systems are not smooth physical valleys — they are pure numerical noise. Trying to step along them (even conservatively via clamping) corrupts the step direction:

Strategy	Convergence rate	Mean steps
Clamp ($\tau = 2 \times 10^{-3}$)	55%	~9000
Filter ($\tau = 8 \times 10^{-3}$)	**99%**	**455**

3.3 Threshold sweep (NR minimization)

τ	Convergence rate	Mean steps (when converged)
10^{-5}	40%	>10 000
10^{-4}	68%	~4000
2×10^{-3}	78%	~2500
5×10^{-3}	95%	~800
8×10^{-3}	**99%**	**455**
10^{-2}	97%	~600

The optimal threshold of $\tau = 8 \times 10^{-3}$ can be interpreted as the boundary below which DFTB0 Hessian eigenvalues are dominated by noise rather than chemistry.

3.4 Threshold sweep (GAD grid search)

The same study on GAD showed that the interaction between τ and success rate is consistent with the NR finding:

τ	Mean success rate	Mean steps (when successful)
2×10^{-3}	0.858	1138.5
5×10^{-3}	**0.933**	806.0
8×10^{-3}	0.917	608.6
10^{-2}	0.933	514.4

Notably, $\tau = 2\times10^{-3}$ again performed worst, and the top configuration (`mad1.3_tr5e-3_blmode_tracked`) achieved **30/30 (100%) success** with 477 mean steps — comparable to the NR minimization result.

```
def _vib_mask_from_evals(evals: torch.Tensor, tr_threshold: float) ->
    torch.Tensor:
    # Hard filter: completely exclude modes with |lambda| <= tau
    return evals.abs() > float(tr_threshold)

evals, evecs = torch.linalg.eigh(hess_proj)
```

```
vib_mask      = _vib_mask_from_evals(evals, tr_threshold)
vib_indices   = torch.where(vib_mask)[0]
```

Listing 3: Filtering in practice: mask construction (`run_gad_baselines_parallel.py`).

4 Adaptive Trust Region

4.1 Overview

After computing the vibrational pseudoinverse step δx, we scale it to a *current trust radius* Δ and evaluate whether the actual energy change matches the quadratic model's prediction:

$$\rho = \frac{\delta E_{\text{actual}}}{\delta E_{\text{pred}}}, \quad \delta E_{\text{pred}} = g^T \delta\hat{x} + \tfrac{1}{2}\delta\hat{x}^T H\, \delta\hat{x},$$

where $\delta\hat{x}$ is the step capped to radius Δ.

The trust radius is then adapted:

$$\Delta_{k+1} = \begin{cases} \min(1.5\,\Delta_k,\ \Delta_{\max}) & \text{if } \rho > 0.75 \quad \text{(excellent model)} \\ 0.5\,\Delta_k & \text{if } \rho < 0.25 \quad \text{(poor model)} \\ 0.25\,\Delta_k & \text{if step rejected} \quad (\delta E_{\text{actual}} > 0) \\ \Delta_k & \text{otherwise} \end{cases}$$

Steps are *rejected* and retried at a smaller radius when the energy increases (for minimization) or when $\rho \le 0$.

4.2 Why $\rho > 0.75$ / $\rho < 0.25$ thresholds?

These are the standard dogleg/trust-region thresholds from Nocedal & Wright [2]: $\rho > 0.75$ means the local quadratic model predicts at least 75% of the actual improvement, so we trust it enough to expand Δ; $\rho < 0.25$ means the model is unreliable, so we contract.

```
accepted, retries = False, 0
while not accepted and retries < max_retries:
    capped_disp = _cap_displacement(step_disp, current_trust_radius)
    dx_flat     = capped_disp.reshape(-1)
    pred_dE     = float((grad @ dx_flat + 0.5 * dx_flat @ (hess_proj
        @ dx_flat)).item())

    new_coords  = coords + capped_disp
    energy_new  = _to_float(predict_fn(new_coords, ...).energy)
    actual_dE   = energy_new - energy

    if actual_dE <= 1e-5:                         # accept
        accepted = True
        rho = actual_dE / pred_dE if pred_dE < -1e-8 else 0.0
        if rho > 0.75:
            current_trust_radius = min(current_trust_radius * 1.5,
                max_atom_disp)
        elif rho < 0.25:
```

Algorithm 1 Trust Region step (NR minimization / GAD)

```
1:  Compute $\delta x \leftarrow V_{\mathcal{V}}\, |\Lambda_{\mathcal{V}}|^{-1}\, V_{\mathcal{V}}^T g$
2:  accepted ← False; retries ← 0
3:  while ¬accepted and retries < 10 do
4:      $\delta\hat{x} \leftarrow \delta x \cdot \min(1, \Delta / \|\delta x\|_\infty)$
5:      Evaluate $E_{\text{new}}$ at $x_k + \delta\hat{x}$
6:      $\delta E_{\text{actual}} \leftarrow E_{\text{new}} - E_k$
7:      $\delta E_{\text{pred}} \leftarrow g^T \delta\hat{x} + \frac{1}{2}\delta\hat{x}^T H \delta\hat{x}$
8:      if $\delta E_{\text{actual}} \le 0$ then                          ▷ Energy decreased: accept
9:          accepted ← True
10:         $\rho \leftarrow \delta E_{\text{actual}} / \delta E_{\text{pred}}$
11:         if $\rho > 0.75$ then $\Delta \leftarrow \min(1.5\Delta, \Delta_{\max})$
12:         else if $\rho < 0.25$ then $\Delta \leftarrow 0.5\Delta$
13:         end if
14:     else                                              ▷ Energy increased: reject, shrink
15:         $\Delta \leftarrow 0.25\Delta$; retries += 1
16:     end if
17: end while
```

```
            current_trust_radius = max(current_trust_radius * 0.5,
               0.001)
    else:                                               # reject
        current_trust_radius *= 0.25
        retries += 1
```

Listing 4: Trust Region inner loop (`baselines/minimization.py`, Newton-Raphson).

4.3 Impact

The Trust Region enabled the optimizer to take steps as large as 1.3 Å when the quadratic model was reliable, vastly outperforming the previous fixed-step approach which was capped at 0.35 Å and could never escape shallow local traps efficiently.

5 Mode Tracking (GAD-Specific)

5.1 Problem

Standard GAD picks the *instantaneous* lowest eigenvector $v_1(x_k)$ at each step. Near a TS on a noisy PES, the curvature spectrum can reorder between steps due to:

- Numerical noise in DFTB0 Hessian.
- Genuine mode-crossing events (eigenvalue degeneracy).

A sudden swap $v_1 \leftrightarrow v_2$ causes the GAD deflation direction to jump discontinuously, leading to oscillatory behaviour and stalling.

5.2 Solution: eigenvector tracking

We maintain a *reference vector* v_{prev} across steps and, at each step, select v^* from the k lowest eigenvectors that maximizes continuity:

$$v^* = \underset{v \in \{v_1, \dots, v_k\}}{\operatorname{argmax}} \, |\langle v, \, v_{\text{prev}} \rangle| \, .$$

```python
def pick_tracked_mode(V, v_prev, k=8):
    """
    V:       (3N, k) matrix of k candidate eigenvectors
    v_prev: (3N,)  previously tracked direction (or None)
    Returns the column of V most aligned with v_prev.
    """
    if v_prev is None:
        return V[:, 0], 0, 1.0           # default: lowest eigenvalue
    overlaps = torch.abs(V.T @ v_prev)   # (k,)
    j        = int(overlaps.argmax())
    return V[:, j], j, float(overlaps[j])
```

Listing 5: Mode tracking in `core_algos/gad.py`.

5.3 Empirical impact (GAD grid search)

Baseline	Mean success rate	Mean steps	Mean wall time
`plain` (no tracking)	0.863	999.2	11.62 s
`mode_tracked`	**0.958**	**534.6**	**4.61 s**

Mode tracking accounts for +10 percentage points in success rate and a 1.9× reduction in mean steps. The interaction table confirms this holds uniformly across all (mad, τ) combinations — it is not an artifact of a particular hyperparameter setting.

At the optimal configuration (`mad1.3_tr5e-3_blmode_tracked`), mode tracking was necessary to reach 30/30 (100%) success; the corresponding `plain` configuration reached only 26/30 (87%).

Configuration	Success	Steps
`mad1.3_tr5e-3_blmode_tracked`	30/30 (100%)	477.6
`mad1.3_tr5e-3_blplain`	26/30 (87%)	811.3

5.4 Physical interpretation

Near a TS, the lowest two eigenvalues $\lambda_1 < 0 < \lambda_2$ are well-separated. As we approach from a noisy starting point, transiently $\lambda_1 \approx \lambda_2$ (the singularity set $\mathcal{S}$ of GAD, defined in [3]), causing mode-crossing artefacts. Tracking maintains a coherent deflation direction through these transient degeneracies. The mode tracking overlap $|\langle v_1(t), \, v_1(t-1) \rangle|$, logged per step, is an effective diagnostic: values close to 1.0 indicate smooth progress; sharp drops signal mode-crossing events.

6 Maximum Atom Displacement Cap

6.1 Role in the Trust Region

$\Delta_{\max}$ serves as the *absolute ceiling* on the trust radius: even if $\rho > 0.75$ at every step, the trust radius cannot exceed $\Delta_{\max}$. This prevents the optimizer from taking physically unreasonable steps (e.g., crashing atoms together) even when the quadratic model looks excellent over a very large range.

6.2 Empirical finding

$\Delta_{\max}$ (Å)	NR conv. rate (early grid)	GAD success rate	GAD mean steps
0.35	45%	0.910	—
0.50	53%	0.908	728.3
1.00	64%	0.917	810.1
1.30	**72%**	**0.917**	**761.7**
1.50	72%	0.900	767.4

$\Delta_{\max} = 1.3$ Å is the sweet spot: large enough to let the optimizer make decisive progress when the Hessian is trustworthy (long, shallow valleys), but small enough to prevent geometry corruption for small molecules. The plateau between 1.3 and 1.5 suggests diminishing returns beyond 1.3 Å.

Key insight: With the Trust Region in place, $\Delta_{\max}$ is almost never actually hit during normal convergence — the trust radius Δ_k self-regulates to match the local quadratic regime. $\Delta_{\max}$ only acts as a safety ceiling during the initial phase (step 0 initializes $\Delta_0 = \Delta_{\max}$) and is rarely a binding constraint after that. Restricting it to 0.35 Å was therefore artificially stalling convergence in the early iterations.

7 Eckart Projection

7.1 What it does

The Hessian of a non-periodic molecular system has exactly 6 zero eigenvalues corresponding to rigid-body translations and rotations (5 for linear molecules). Without explicit removal, numerical errors cause small non-zero components along these modes, which can accumulate across many gradient steps into significant centre-of-mass drift or net rotation of the molecule.

Eckart projection enforces that gradients and guide vectors lie in the *purely vibrational* complement of the TR space:

$$g_{\text{vib}} = \left(I - \sum_{j \in \text{TR}} v_j v_j^T \right) g.$$

7.2 Implementation

```
# project_vector_to_vibrational_torch removes TR components from g
grad_proj = project_vector_to_vibrational_torch(
    forces.reshape(-1), coords, atomsymbols
)
```

Listing 6: Eckart gradient projection (`dependencies/differentiable_projection.py`).

For GAD, *both* the gradient g and the guide vector v_1 are projected before computing the GAD direction:

```python
if project_gradient_and_v and atomsymbols is not None:
    gad_vec, v_proj, _ = gad_dynamics_projected_torch(
        coords=coords, forces=forces, v=v, atomsymbols=atomsymbols,
    )
    v      = v_proj.reshape(-1)
    gad_flat = gad_vec.reshape(-1)
```

Listing 7: Eckart projection of the GAD direction (`runners/run_gad_baselines_parallel.py`).

7.3 Finding

Eckart projection was critical in every configuration tested. Disabling it caused progressive drift of the molecular centre of mass, which corrupts the Hessian eigenvalue spectrum and causes spurious flat modes to proliferate beyond what the τ filter can handle alone. `project_gradient_and_v=True` was fixed in all grid-search configurations; no configuration without it was competitive.

8 Hessian Purification (Negative Result)

Hypothesis: Enforcing the translational sum rule $\sum_j H_{ij} = 0$ (for all i) on the raw DFTB0 Hessian before projection would improve numerical stability by reducing the residual non-zero eigenvalues in the TR subspace.

Result: Across 32 matched configuration pairs in the NR grid search, purification produced *identical* per-sample convergence outcomes in every single pair. It was therefore disabled (`purify_hessian=False`) to save the small additional compute cost.

Interpretation: The Eckart projection already effectively removes TR contamination from the *step*, even if the Hessian matrix itself retains small TR residuals. Since the pseudoinverse step never projects onto TR modes (by design), enforcing perfect sum rules on H adds no benefit.

9 Synthesis: Comparison Tables and Interaction Effects

9.1 NR Minimization: final grid results

Best configuration: `mad1.3_tr8e-3_pg=true_ph=false`: 99% convergence, 455 mean steps (starting from 2 Å noise).

9.2 GAD: full grid-search results (30 samples, DFTB0, 2 Å noise)

Configuration	Success rate	Mean steps	Wall time
`mad1.3_tr5e-3_blmode_tracked`	**100%**	477.6	2.66 s
`mad1.0_tr2e-3_blmode_tracked`	**100%**	896.4	5.03 s
`mad1.5_tr5e-3_blmode_tracked`	**100%**	1035.4	5.81 s
`mad0.5_tr1e-2_blmode_tracked`	97%	171.7	2.12 s
`mad1.5_tr2e-3_blplain`	73%	1470.3	20.98 s

9.3 Main effects (GAD)

`baseline` +10 pp success rate, 1.9× fewer steps for `mode_tracked` vs `plain`. **Largest single effect.**

`tr_threshold` 2×10^{-3} clearly worst (0.858); 5×10^{-3} and 10^{-2} tied on success rate (0.933) but 10^{-2} is faster. **Second largest effect.**

`max_atom_disp` Flat between 1.0–1.3 Å for success rate; 0.5 Å slightly worse at 0.908. **Third.**

9.4 Worst performers

All bottom-10 configurations are `plain` baseline, confirming that mode tracking is the dominant factor. The plain runs also have 5× longer wall times (11.6 s vs 4.6 s mean), suggesting they are not just slower but are taking fundamentally different (pathological) trajectories.

9.5 Hardest samples

Sample 012 succeeded in only 10/32 configurations (31%), all successful runs used `mode_tracked`. This sample likely has a highly degenerate curvature spectrum near the midpoint, making the deflation direction ambiguous without tracking. Samples 014 (62%) and 025 (81%) form a middle tier; the remaining 27 samples converged in >88% of configurations.

10 Summary of Logging and Diagnostics Infrastructure

A comprehensive `TrajectoryLogger` was built to record `ExtendedMetrics` at every step, enabling post-hoc analysis of failure modes. Key logged fields per step:

Field	Description
`step_size_eff`	Trust radius Δ_k used at step k
`x_disp_step`	Actual per-step displacement $\|x_k - x_{k-1}\|$
`eig_0`, `eig_1`	Two lowest vibrational eigenvalues
`morse_index`	Number of negative vibrational eigenvalues
`eig_gap_01`	$\lvert\lambda_2 - \lambda_1\rvert$ (singularity proximity)
`mode_overlap`	$\lvert\langle v_1(t), v_1(t-1)\rangle\rvert$
`grad_norm`	$\|\nabla E\|$
`n_tr_modes`	Number of near-zero modes filtered
`energy_delta`	$E_k - E_{k-1}$

Grid-search analysis scripts (`analyze_gad_grid.py`, `analyze_minimization_nr_grid.py`) parse per-combo result JSON files, compute main effects, interaction tables, and sample hardness rankings. Trajectory plotting scripts (`plot_gad_grid_trajectories.py`, `plot_nr_grid_trajectories.py`) produce per-sample PNG diagnostic figures including trust-radius evolution — directly mirroring the step-size-over-time plots used in the NR analysis.

11 Conclusions and Next Steps

Both the NR minimization and the GAD saddle-point search have been transformed from unreliable baselines into high-performance optimizers on a genuinely difficult benchmark (2 Å noise, DFTB0, Transition1x test set):

Algorithm	Before	After	Speedup
NR Minimization	45%, ∼8000 steps	99%, 455 steps	17×
GAD (TS search)	∼60–70%	100%, 478 steps	$> 2\times$

Most impactful finding across both algorithms. Aggressively filtering modes with $|\lambda| < 8 \times 10^{-3}$ (rather than clamping them) is the single biggest source of improvement. The flat modes in DFTB0 at 2 Å noise are not physical — they are numerical artefacts that, if stepped along, corrupt the entire search. Discarding them entirely acts as a highly effective dynamic dimensionality reduction.

Next steps.

1. Validate on 1 Å and 0.5 Å noise levels to probe generalization.
2. Test with higher-fidelity functionals (DFT) where Hessian noise may differ.
3. Investigate sample 012 (31% success rate) in detail — likely a degenerate curvature landscape requiring a different initialization.
4. Explore the interaction between τ and molecular size (number of atoms), since the number of modes filtered scales with N.

References

[1] O. Schreiner, C. Bhatt, and P. Rauer, *Transition1x — a dataset for building generalizable reactive machine learning potentials*, Scientific Data, 9:1, 2022.

[2] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed., Springer, 2006.

[3] A. Levitt and C. Ortner, *Convergence and cycling in Walker-type saddle search algorithms*, SIAM Journal on Numerical Analysis, 55(2):2204–2227, 2017.