`Multi-Mode-GAD`: Robust Transition State Search from Noisy Starting Geometries

With Applications to Adjoint Sampling for Reaction Generation

Mehmet Ozdincer, Andreas Burger
Matter Lab
University of Toronto

1 Introduction

Transition states (TSs) are first-order saddle points on the Born–Oppenheimer potential energy surface (PES) that govern the kinetics of chemical reactions. Locating these critical points is central to understanding reaction mechanisms, predicting rate constants, and designing catalysts. Despite decades of research, reliably finding transition states remains one of the most challenging problems in computational chemistry, particularly when the initial guess is far from the true saddle point.

Modern machine learning interatomic potentials (MLIPs) such as HIP [8] provide fast, differentiable access to energies, forces, and Hessians, enabling gradient-based saddle point search at a fraction of the cost of *ab initio* calculations. Semi-empirical methods such as SCINE/Sparrow [9] offer CPU-based alternatives with analytical Hessians at DFTB0, PM6, or AM1 levels of theory. However, when starting geometries are corrupted by noise—as commonly arises in generative models, coarse molecular dynamics, or interpolation schemes—the search landscape becomes significantly more complex, with numerous higher-order saddle points (Morse index > 1) that can trap conventional algorithms.

This report documents the theoretical foundations, algorithmic developments, and experimental results of the `ts-tools` toolkit. The two major contributions are:

(i) **Robust saddle point search from noisy geometries:** A comprehensive study of algorithms and escape strategies that achieve near-perfect transition state convergence from starting points perturbed by up to 2.0 Å of random noise.

(ii) **Adjoint sampling for reaction generation:** Integration of diffusion-based generative models that sample molecular conformations from Boltzmann distributions defined by energy functions, enabling data-driven reaction pathway generation.

2 Theoretical Background

2.1 Potential Energy Surfaces and Critical Points

Consider a molecular system with N atoms in three-dimensional space, described by nuclear coordinates $\boldsymbol{x} \in \mathbb{R}^{3N}$. The Born–Oppenheimer potential energy surface $E(\boldsymbol{x})$ determines the energy as a function of nuclear configuration. Critical points satisfy $\nabla E(\boldsymbol{x}^*) = \mathbf{0}$, and are classified by the *Morse index*—the number of negative eigenvalues of the Hessian matrix $\boldsymbol{H}(\boldsymbol{x}^*) = \nabla^2 E(\boldsymbol{x}^*)$:

Definition 1 (Morse Index)**.** *The* Morse index *of a critical point* $\boldsymbol{x}^*$ *is*

$$\mathit{index}(\boldsymbol{x}^*) = \#\{\lambda_i < 0 : \lambda_i \in \sigma(\boldsymbol{H}(\boldsymbol{x}^*))\}, \tag{1}$$

where $\sigma(\boldsymbol{H})$ denotes the spectrum of the Hessian restricted to the vibrational subspace (i.e., after removing translation and rotation modes).

- **Minima:** index = 0 (all eigenvalues positive). Local energy minima corresponding to stable molecular conformations.
- **Transition states:** index = 1 (exactly one negative eigenvalue). First-order saddle points connecting reactant and product minima via a minimum energy pathway.
- **Higher-order saddle points:** index ≥ 2. These are not physically meaningful transition states but frequently arise as trapping regions in saddle point search algorithms.

A practical convergence criterion for transition state detection uses the *eigenvalue product*:

$$\lambda_0 \cdot \lambda_1 < 0, \tag{2}$$

where λ_0 and λ_1 are the two smallest vibrational eigenvalues (sorted in ascending order). This condition is satisfied if and only if exactly one of the two lowest eigenvalues is negative, which characterizes an index-1 saddle point.

2.2 Gentlest Ascent Dynamics (GAD)

Gentlest Ascent Dynamics [1] is a dynamical system designed to climb from minima to saddle points along the "gentlest" ascent direction. The key idea is to invert the force component along the lowest eigenvector of the Hessian, creating a modified force field whose stable fixed points are index-1 saddle points.

Let $\boldsymbol{v}_1(\boldsymbol{x})$ denote the eigenvector corresponding to the smallest eigenvalue λ_0 of the Hessian $\boldsymbol{H}(\boldsymbol{x})$, and let $\boldsymbol{f}(\boldsymbol{x}) = -\nabla E(\boldsymbol{x})$ be the force. The GAD force is:

$$\boxed{\boldsymbol{f}_{\mathrm{GAD}}(\boldsymbol{x}) = -\nabla E(\boldsymbol{x}) + 2\big(\nabla E(\boldsymbol{x}) \cdot \boldsymbol{v}_1\big)\, \boldsymbol{v}_1} \tag{3}$$

Remark 1 (Interpretation in the eigenbasis)**.** *Decomposing the gradient in the Hessian eigenbasis $\{\boldsymbol{v}_i\}_{i=0}^{3N-1}$ with $\nabla E = \sum_i g_i \boldsymbol{v}_i$, the GAD force becomes:*

$$\boldsymbol{f}_{GAD} = g_0 \boldsymbol{v}_0 - \sum_{i \geq 1} g_i \boldsymbol{v}_i. \tag{4}$$

That is, GAD ascends *along $\boldsymbol{v}_0$ (the softest mode) while* descending *along all other modes. At a fixed point where $\boldsymbol{f}_{GAD} = \boldsymbol{0}$, we must have $g_i = 0$ for all i, i.e., $\nabla E = \boldsymbol{0}$—a critical point. The stability analysis shows that index-1 saddle points are* stable *fixed points of GAD.*

2.2.1 Euler Integration

The simplest time-stepping scheme follows the GAD vector field:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \Delta t \cdot \boldsymbol{f}_{\mathrm{GAD}}(\boldsymbol{x}_n). \tag{5}$$

2.2.2 RK45 Integration

For more accurate trajectory following, we use the Dormand–Prince adaptive Runge–Kutta 4(5) scheme with error-based step size control:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \sum_{i=1}^{6} b_i \boldsymbol{k}_i, \quad \boldsymbol{k}_i = \Delta t \cdot \boldsymbol{f}_{\mathrm{GAD}}\left(\boldsymbol{x}_n + \sum_{j<i} a_{ij} \boldsymbol{k}_j\right), \tag{6}$$

with embedded error estimation for adaptive time stepping.

2.3 Eigenvector Following (Newton GAD)

Eigenvector following (EF) applies a Newton-like update on the GAD potential surface. The key insight is that the Hessian of the GAD potential, $\boldsymbol{H}_{\mathrm{GAD}}$, can be expressed as:

$$\boldsymbol{H}_{\mathrm{GAD}} = -\boldsymbol{H} + 2\boldsymbol{v}_1\boldsymbol{v}_1^T\boldsymbol{H} + 2\boldsymbol{H}\boldsymbol{v}_1\boldsymbol{v}_1^T. \tag{7}$$

Near a saddle point where $\boldsymbol{H}\boldsymbol{v}_i = \lambda_i\boldsymbol{v}_i$, the absolute-value Hessian provides a convenient approximation:

$$\boldsymbol{H}_{\mathrm{abs}} = \sum_i |\lambda_i|\, \boldsymbol{v}_i\boldsymbol{v}_i^T. \tag{8}$$

The Newton step on the GAD surface becomes:

$$\boxed{\Delta\boldsymbol{x}_{\mathrm{EF}} = \boldsymbol{H}_{\mathrm{abs}}^{-1} \cdot \boldsymbol{f}_{\mathrm{GAD}}} \tag{9}$$

This provides quadratic convergence near the saddle point, as opposed to the linear convergence of Euler integration.

2.4 Partitioned Rational Function Optimization (P-RFO)

P-RFO extends the rational function approximation to saddle point optimization by partitioning the step into maximization and minimization components. The augmented Hessian equations are:

$$\begin{pmatrix} \lambda_0 & g_0 \\ g_0 & 1 \end{pmatrix} \begin{pmatrix} h_0 \\ 1 \end{pmatrix} = \mu^{(+)} \begin{pmatrix} h_0 \\ 1 \end{pmatrix}, \tag{10}$$

for the maximization along $\boldsymbol{v}_0$, and

$$\begin{pmatrix} \lambda_1 & 0 & \cdots & g_1 \\ 0 & \lambda_2 & \cdots & g_2 \\ \vdots & & \ddots & \vdots \\ g_1 & g_2 & \cdots & 1 \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ 1 \end{pmatrix} = \mu^{(-)} \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ 1 \end{pmatrix}, \tag{11}$$

for the minimization along all other modes. The total step in Cartesian coordinates is:

$$\Delta\boldsymbol{x}_{\text{P-RFO}} = \sum_i h_i\, \boldsymbol{v}_i, \tag{12}$$

subject to a trust radius constraint $\|\Delta\boldsymbol{x}\| \leq \Delta_{\max}$.

2.5 Eigenvalue Product Descent

A direct optimization approach minimizes the eigenvalue product as a loss function:

$$\mathcal{L}_{\mathrm{prod}}(\boldsymbol{x}) = \lambda_0(\boldsymbol{x}) \cdot \lambda_1(\boldsymbol{x}), \tag{13}$$

where λ_0, λ_1 are the two smallest vibrational eigenvalues. At a transition state, $\lambda_0 < 0$ and $\lambda_1 > 0$, so $\mathcal{L}_{\mathrm{prod}} < 0$. The gradient is computed via automatic differentiation through the eigendecomposition:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \eta\, \nabla_{\boldsymbol{x}}\mathcal{L}_{\mathrm{prod}}(\boldsymbol{x}_n), \tag{14}$$

with gradient clipping and per-atom displacement limits for stability. This method requires a differentiable calculator (e.g., HIP) and achieved 92.7% convergence from clean starting geometries in 2.3 steps on average.

2.6 Direct Eigenvalue Descent (Sign Enforcer)

The sign enforcer uses an adaptive loss function based on the current number of negative eigenvalues:

$$\mathcal{L}_{\text{sign}}(\boldsymbol{x}) = \begin{cases} (\lambda_0 - \lambda_{\text{target}})^2 & \text{if } n_{\text{neg}} = 0 \text{ (push } \lambda_0 \text{ below target)}, \\ 0 & \text{if } n_{\text{neg}} = 1 \text{ (already at TS)}, \\ \sum_{i=1}^{n_{\text{neg}}-1} \lambda_i^2 & \text{if } n_{\text{neg}} > 1 \text{ (push extras positive)}. \end{cases} \tag{15}$$

This achieved 99.7% convergence from clean geometries in 1.8 steps, the highest among all methods tested.

2.7 High-index Saddle Dynamics (HiSD)

The k-HiSD method [2] generalizes GAD to target index-k saddle points using the reflection operator:

$$\boldsymbol{R}_k = \boldsymbol{I} - 2\sum_{i=0}^{k-1} \boldsymbol{v}_i \boldsymbol{v}_i^T, \tag{16}$$

which inverts force components along the k lowest eigenvectors. The dynamics become:

$$\dot{\boldsymbol{x}} = -\boldsymbol{R}_k \nabla E(\boldsymbol{x}). \tag{17}$$

Theorem 1 (Stability of k-HiSD [2])**.** *Index-k saddle points are* stable *fixed points of k-HiSD dynamics. In particular, 1-HiSD is equivalent to GAD and stabilizes transition states.*

Remark 2 (Critical implication)**.** *Theorem 1 implies that using k equal to the current Morse index* stabilizes *the current saddle rather than escaping it. This was a critical finding in our experiments: adaptive k = Morse index achieved 0% escape rate.*

2.7.1 Improved HiSD (iHiSD)

The iHiSD algorithm [3] introduces a crossover parameter $\theta \in [0, 1]$ that interpolates between gradient flow ($\theta = 0$) and full k-HiSD ($\theta = 1$):

$$\boldsymbol{d}_k = (1 - s\theta)\nabla E + 2\theta \sum_{i=0}^{k-1} (\nabla E \cdot \boldsymbol{v}_i)\, \boldsymbol{v}_i, \tag{18}$$

where $s = \pm 1$ controls upward ($s = +1$) or downward ($s = -1$) search. The crossover schedule $\theta(t)$ (sigmoid, linear, or exponential) allows smooth transition from gradient-guided exploration to saddle-targeting dynamics, providing nonlocal convergence guarantees.

2.7.2 Recursive HiSD

Recursive HiSD implements systematic saddle index descent:

1. Detect current Morse index n.
2. If $n = 1$, the transition state is found.
3. Perturb along an unstable direction to escape the current saddle.
4. Run $(n-1)$-HiSD to locate an index-$(n-1)$ saddle.
5. Repeat until index 1 is reached.

This is motivated by Theorem 5.1 of [2], which guarantees connectivity between saddles of adjacent indices via saddle-saddle connections.

3 Eckart Projection and Mass-Weighting

A molecule with N atoms in three dimensions has $3N$ degrees of freedom, of which 3 are translational, 3 are rotational (for nonlinear molecules), and $3N - 6$ are vibrational. The raw Hessian $\boldsymbol{H} \in \mathbb{R}^{3N\times 3N}$ has 6 near-zero eigenvalues corresponding to translation/rotation (TR) modes. These must be projected out before eigenvalue analysis for saddle point characterization.

3.1 Eckart Conditions and the B Matrix

The Eckart conditions [4] define the separation of internal (vibrational) and external (TR) motions. The TR subspace is spanned by 6 vectors: 3 translations and 3 infinitesimal rotations about the center of mass. Collecting these into a matrix $\boldsymbol{B} \in \mathbb{R}^{3N\times 6}$:

$$\boldsymbol{B} = [\boldsymbol{B}_{\text{trans}} \,|\, \boldsymbol{B}_{\text{rot}}], \tag{19}$$

where the columns of $\boldsymbol{B}_{\text{trans}}$ are mass-weighted unit translations:

$$(\boldsymbol{B}_{\text{trans}})_{3a+\alpha,\beta} = \sqrt{m_a}\,\delta_{\alpha\beta}, \quad a = 0,\ldots,N{-}1, \quad \alpha,\beta \in \{x,y,z\}, \tag{20}$$

and the columns of $\boldsymbol{B}_{\text{rot}}$ encode infinitesimal rotations about the center of mass $\bar{\boldsymbol{x}}$:

$$(\boldsymbol{B}_{\text{rot}})_{3a+\alpha,k} = \sqrt{m_a}\,(\epsilon_{\alpha\beta\gamma}\, r_{a,\gamma}), \tag{21}$$

with $\boldsymbol{r}_a = \boldsymbol{x}_a - \bar{\boldsymbol{x}}$ the displacement from the center of mass and $\epsilon_{\alpha\beta\gamma}$ the Levi-Civita symbol.

3.2 Full Projection (PHP)

The standard Eckart projection constructs the projector onto the TR subspace and its complement:

$$\boldsymbol{P}_{\text{TR}} = \boldsymbol{B}(\boldsymbol{B}^T\boldsymbol{B})^{-1}\boldsymbol{B}^T, \qquad \boldsymbol{P}_{\text{vib}} = \boldsymbol{I} - \boldsymbol{P}_{\text{TR}}. \tag{22}$$

The projected Hessian in the full $3N$-dimensional space is:

$$\boxed{\tilde{\boldsymbol{H}} = \boldsymbol{P}_{\text{vib}}\,\boldsymbol{H}_{\text{mw}}\,\boldsymbol{P}_{\text{vib}}} \tag{23}$$

where $\boldsymbol{H}_{\text{mw}} = \boldsymbol{M}^{-1/2}\boldsymbol{H}\boldsymbol{M}^{-1/2}$ is the mass-weighted Hessian with $\boldsymbol{M} = diag(m_1, m_1, m_1, m_2, m_2, m_2, \ldots, m_N, r$
This yields a $3N\times 3N$ matrix with exactly 6 zero eigenvalues (TR modes) and $3N-6$ vibrational eigenvalues.

3.3 Reduced Basis (QR Complement)

An alternative approach constructs an explicit $(3N{-}6)$-dimensional vibrational basis via QR decomposition:

1. Orthonormalize $\boldsymbol{B}$ via QR: $\boldsymbol{B} = \boldsymbol{Q}_{\text{TR}}\boldsymbol{R}$.
2. Complete to a full basis using SVD of $\boldsymbol{I} - \boldsymbol{Q}_{\text{TR}}\boldsymbol{Q}_{\text{TR}}^T$.
3. Extract the $3N - 6$ columns with nonzero singular values as $\boldsymbol{Q}_{\text{vib}} \in \mathbb{R}^{3N\times(3N-6)}$.

The reduced vibrational Hessian is then:

$$\boldsymbol{H}_{\text{red}} = \boldsymbol{Q}_{\text{vib}}^T\boldsymbol{H}_{\text{mw}}\boldsymbol{Q}_{\text{vib}} \in \mathbb{R}^{(3N-6)\times(3N-6)}, \tag{24}$$

which is full-rank and has no zero eigenvalues. Eigenvectors are mapped back to Cartesian space via $\boldsymbol{v}_{\text{cart}} = \boldsymbol{Q}_{\text{vib}}\boldsymbol{v}_{\text{red}}$.

3.4 Mass-Weighting

Mass-weighting transforms the Hessian so that eigenvalues correspond to squared vibrational frequencies:

$$\boldsymbol{H}_{\text{mw}} = \boldsymbol{M}^{-1/2}\boldsymbol{H}\boldsymbol{M}^{-1/2}, \tag{25}$$

where $\boldsymbol{M} = diag(\underbrace{m_1, m_1, m_1}_{x,y,z}, \ldots, \underbrace{m_N, m_N, m_N}_{x,y,z})$. The masses m_a are the atomic masses of each atom. This ensures that heavy atoms contribute proportionally less to vibrational modes than light atoms, which is physically correct.

3.5 Sum-Rule Purification

The acoustic sum rule states that the Hessian should satisfy translational invariance:

$$\sum_b H_{a\alpha,b\beta} = 0 \quad \forall\, a, \alpha, \beta. \tag{26}$$

Numerical Hessians may violate this condition. Purification enforces the sum rule by adjusting diagonal blocks:

$$H^{\text{purified}}_{a\alpha,a\beta} = H_{a\alpha,a\beta} - \sum_{b\neq a} H_{a\alpha,b\beta}. \tag{27}$$

3.6 Kabsch Frame Tracking

During dynamics, molecular rotation can contaminate vibrational mode analysis. Kabsch alignment [5] rigidly aligns the current geometry to a reference frame at each step by finding the optimal rotation matrix via SVD:

$$\boldsymbol{R}^* = \arg\min_{\boldsymbol{R}\in SO(3)} \sum_a m_a \|\boldsymbol{R}\boldsymbol{x}_a - \boldsymbol{x}_a^{\text{ref}}\|^2. \tag{28}$$

3.7 Projection of Gradient and Guide Vector

An important refinement projects both the gradient ∇E and the GAD guide vector $\boldsymbol{v}_1$ into the vibrational subspace before computing the GAD step:

$$\nabla E_{\text{vib}} = \boldsymbol{P}_{\text{vib}}\nabla E, \qquad \boldsymbol{v}_{1,\text{vib}} = \frac{\boldsymbol{P}_{\text{vib}}\boldsymbol{v}_1}{\|\boldsymbol{P}_{\text{vib}}\boldsymbol{v}_1\|}. \tag{29}$$

This prevents translational/rotational leakage into the dynamics and improves numerical stability.

3.8 Experimental Comparison of Projection Methods

We tested 7 projection variants on identical noisy starting geometries. All variants produced statistically identical convergence rates, confirming that Eckart projection is robust to implementation details:

Method	Description
`eckart_full`	Full $3N \times 3N$ projection (Eq. 23)
`reduced_basis`	QR complement (Eq. 24)
`eckart_full + purify`	PHP + sum-rule purification
`reduced_basis + purify`	QR + sum-rule purification
`reduced_basis + purify + frame`	QR + purification + Kabsch
`eckart_full + proj_grad_v`	PHP + gradient/v projection
`reduced_basis + proj_grad_v`	QR + gradient/v projection

All 7 methods produced identical convergence rates within statistical noise, indicating that the choice of projection method is not a bottleneck. The standard `eckart_full` with gradient/v projection is used as the default throughout.

4 Mode Tracking

A fundamental challenge in saddle point search is *mode switching*: the eigenvector ordering can change between steps, causing the algorithm to inadvertently follow a different mode. This is particularly problematic near eigenvalue crossings where $|\lambda_i - \lambda_j| \to 0$.

4.1 Maximum Overlap Tracking

We implement continuous mode tracking by selecting the candidate eigenvector with maximum overlap to the previous step's tracked mode:

$$j^* = \arg\max_{j \in \{0,\ldots,k-1\}} |\langle \boldsymbol{v}_j^{(n)}, \boldsymbol{v}_{\text{tracked}}^{(n-1)} \rangle|, \tag{30}$$

where k is the number of candidate modes to consider (typically $k = 8$). The overlap metric is:

$$\text{overlap}^{(n)} = |\langle \boldsymbol{v}_{j^*}^{(n)}, \boldsymbol{v}_{\text{tracked}}^{(n-1)} \rangle| \in [0, 1]. \tag{31}$$

4.2 Mode Smoothing

Optionally, the tracked mode can be smoothed with an exponential moving average:

$$\boldsymbol{v}^{(n)} = (1 - \beta)\, \boldsymbol{v}_{\text{tracked}}^{(n-1)} + \beta\, \boldsymbol{v}_{j^*}^{(n)}, \quad \boldsymbol{v}^{(n)} \leftarrow \frac{\boldsymbol{v}^{(n)}}{\|\boldsymbol{v}^{(n)}\|}, \tag{32}$$

where $\beta \in (0, 1]$ controls the smoothing strength. This prevents abrupt direction changes near mode crossings.

5 The Challenge of Noisy Starting Geometries

When starting from clean geometries (e.g., midpoints between reactant and TS from the Transition1x dataset [10]), GAD and related methods achieve excellent convergence rates:

Method	**Convergence (%)**	**Avg. Steps**
GAD-Euler	91.3	25.1
GAD-RK45	91.3	7.3
Eigenvalue Product Descent	92.7	2.3
Direct Eigenvalue Descent	99.7	1.8

However, adding random noise ($\sigma = 1.0$–$2.0\,\text{Å}$) to starting geometries causes dramatic failures:

Method	**Clean (%)**	**Noisy 1 Å (%)**
GAD-Euler	91.3	13
GAD-RK45	91.3	< 5
Eigenvalue Product Descent	92.7	< 5
Direct Eigenvalue Descent	99.7	~ 0

5.1 Why Noisy Starts Fail

Diagnostic analysis of 20 noisy trajectories revealed the primary failure mechanism:

1. **High-index saddle trapping:** Noisy geometries start at or near higher-order saddle points (typical Morse index 5–8). GAD dynamics attempts to find an index-1 saddle, but the landscape near high-index saddles provides weak guidance.

2. **Timestep collapse:** Adaptive timestep controllers reduce Δt when the GAD direction oscillates or when displacement constraints are violated. Near high-index saddles, this leads to $\Delta t \to 0$, halting all progress.

3. **Eigenvalue gap singularities:** The set $\mathcal{S} = \{\boldsymbol{x} : \lambda_0(\boldsymbol{x}) = \lambda_1(\boldsymbol{x})\}$ forms a codimension-1 manifold [6] where the lowest eigenvector becomes discontinuous. Trajectories near $\mathcal{S}$ exhibit oscillatory behavior. However, only 26% of stalls occur near singularities; the majority occur at genuine high-index saddles.

5.2 Progression of Solutions

The path to robust noisy-start convergence involved systematic experimentation:

Strategy	**Conv. (%)**	**Key Innovation**
Plain GAD	13	Baseline
Hybrid GAD + Product Descent	33	Switch algorithms
L-BFGS Energy Minimizer	10	Minimize first
Plateau Detection (v_2 kick)	40	First escape mechanism
Mode Tracking + Trust Radius	92/71SCINE/HIP	Continuous eigenvector tracking
v_2 Kicking (v2 tests)	**100**	2.0 Å noise
Adaptive dt Control	92	Path-based dt adaptation
State-based dt Control	80	No path information needed

6 Escape Strategies for High-Index Saddle Points

6.1 v_2 Kicking: The Primary Escape Mechanism

The most effective escape strategy is perturbation along the second vibrational eigenvector $\boldsymbol{v}_2$ (the mode with second-smallest eigenvalue):

Algorithm 1 $\boldsymbol{v}_2$ Kicking Escape

Require: Current coordinates $\boldsymbol{x}$, Hessian $\boldsymbol{H}$, escape magnitude δ

Compute projected Hessian $\tilde{\boldsymbol{H}}$
Eigendecompose: $\tilde{\boldsymbol{H}}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$
Skip TR modes (first 6 near-zero eigenvalues)
$\boldsymbol{v}_2 \leftarrow$ second vibrational eigenvector
if adaptive delta **and** $\lambda_1 < -0.01$ **then**
    $\delta \leftarrow \delta / \sqrt{|\lambda_1|}$ ▷ Scale inversely with curvature
end if
$\boldsymbol{x}^+ \leftarrow \boldsymbol{x} + \delta\, \boldsymbol{v}_2, \quad \boldsymbol{x}^- \leftarrow \boldsymbol{x} - \delta\, \boldsymbol{v}_2$
Evaluate $E(\boldsymbol{x}^+)$ and $E(\boldsymbol{x}^-)$
if both geometries valid (min interatomic distance $> 0.5\,\text{Å}$) **then**
    return $\arg\min(E(\boldsymbol{x}^+), E(\boldsymbol{x}^-))$
else
    Shrink $\delta \leftarrow 0.5\,\delta$ and retry (up to 5 times)
end if

6.1.1 Plateau Detection

The escape mechanism is triggered when a *plateau* is detected, defined by:

1. Mean displacement over a window of W steps falls below threshold $d_{\min}$:

$$\frac{1}{W} \sum_{i=n-W+1}^{n} \|\Delta \boldsymbol{x}_i\| < d_{\min}. \tag{33}$$

2. The Morse index is stable and > 1 over the window (standard deviation of negative eigenvalue count below threshold).

3. Patience counter exceeds P consecutive detections.

After escape, the timestep is reset with a boost factor and subsequent decay: $\Delta t \leftarrow \Delta t_{\text{boost}} \cdot \Delta t_0$, then $\Delta t \leftarrow \max(\Delta t \cdot \alpha_{\text{shrink}}, \Delta t_{\min})$ each step.

6.1.2 Why v_2?

Analysis of diagnostic experiments revealed that $\boldsymbol{v}_2$ kicking works primarily as a brute-force "unsticking" mechanism rather than systematic index reduction. The mean Morse index changes only from 5.39 to 5.00 after kicks, a negligible reduction. However, the perturbation is sufficient to:

1. Break the $\Delta t \to 0$ deadlock,

2. Move the geometry to a new basin of attraction,

3. Allow the adaptive timestep to reset, and

4. Resume productive GAD dynamics.

6.2 Alternative Kick Strategies Tested

Eight distinct perturbation strategies were compared:

Strategy	Description
$\boldsymbol{v}_2$ kick	Perturb along second vibrational mode
$\boldsymbol{v}_1$ kick	Perturb along first (tracked) vibrational mode
Random kick	Random direction perturbation (control)
Random $\perp \boldsymbol{v}_1$	Random direction orthogonal to tracked mode
Gradient descent kicks	5 GD steps with $\eta = 0.05$
Ortho-$\boldsymbol{v}_1$ GD	GD constrained to $\boldsymbol{v}_1^{\perp}$ subspace
Higher modes	Sequential kicks along $\boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4, \ldots$
Adaptive-k reflection	$\boldsymbol{d} = -\boldsymbol{R}_k \nabla E$, $\boldsymbol{R}_k = \boldsymbol{I} - 2\sum_{i=0}^{k-1} \boldsymbol{v}_i \boldsymbol{v}_i^T$

6.3 Adaptive Timestep Control

Two families of adaptive timestep strategies were developed and compared:

6.3.1 Path-Based (Displacement History)

The path-based controller adapts Δt based on recent trajectory statistics:

- **Grow:** $\Delta t \leftarrow 1.05 \cdot \Delta t$ when steps are successful (displacement within bounds).
- **Shrink:** $\Delta t \leftarrow 0.8 \cdot \Delta t$ when displacement exceeds maximum atom displacement $d_{\max}$.
- **Hard reset:** $\Delta t \leftarrow 0.5 \cdot \Delta t$ when geometry becomes invalid (atoms too close).
- **Escape boost:** After $\boldsymbol{v}_2$ kick, boost Δt by factor 1.2–3.0× to resume dynamics.

The saddle-order tracker monitors the best (lowest) Morse index seen and adjusts aggressiveness:

$$\text{If } n_{\text{neg}}^{(t)} < n_{\text{neg,best}} \implies \text{reset } \Delta t \leftarrow \Delta t_0 \text{ (making progress).} \tag{34}$$

6.3.2 State-Based (No Path Information)

State-based strategies adapt Δt using only the current local state:

- **Eigenvalue-based:** $\Delta t \propto 1/|\lambda_0|$
- **Gradient-based:** $\Delta t \propto 1/\|\nabla E\|$
- **Spectral gap:** $\Delta t \propto |\lambda_1 - \lambda_0|$
- **GAD angle:** Based on angle between GAD vector and $-\nabla E$
- **Composite:** Combines multiple signals

State-based methods are simpler but achieve lower convergence (80%) compared to path-based methods (92%).

6.4 Trust Radius

A maximum per-atom displacement constraint prevents geometry explosions:

$$\max_a \|\Delta \boldsymbol{x}_a\| \leq d_{\text{trust}}, \tag{35}$$

where $d_{\text{trust}} \approx 0.3$–$0.35\,\text{Å}$. If the GAD step exceeds this, it is uniformly scaled down:

$$\Delta \boldsymbol{x} \leftarrow \frac{d_{\text{trust}}}{\max_a \|\Delta \boldsymbol{x}_a\|} \cdot \Delta \boldsymbol{x}. \tag{36}$$

Additionally, a minimum interatomic distance constraint $\min_{a \neq b} \|\boldsymbol{x}_a - \boldsymbol{x}_b\| > d_{\min}$ (typically $0.5\,\text{Å}$) prevents atomic collisions.

7 Computational Backend

7.1 Calculator Interface

All algorithms interact with energy evaluators through a unified `predict_fn` interface:

$$\texttt{predict\_fn}(\boldsymbol{x}, Z, \texttt{do\_hessian}, \texttt{require\_grad}) \to \{E, \boldsymbol{f}, \boldsymbol{H}\},$$

where $\boldsymbol{x} \in \mathbb{R}^{N\times 3}$ are coordinates, $Z \in \mathbb{Z}^N$ are atomic numbers, and the output contains energy E, forces $\boldsymbol{f} = -\nabla E$, and optionally the Hessian $\boldsymbol{H} = \nabla^2 E$.

7.2 HIP: Machine Learning Interatomic Potential

HIP [8] (https://github.com/burgerandreas/hip) is an Equiformer-based machine learning potential that predicts energies, forces, and Hessians directly from atomic coordinates and species. Key features:

- GPU-accelerated inference
- Differentiable through PyTorch autograd (enables eigenvalue descent)
- Direct Hessian prediction (no finite differences needed)
- Trained on DFT-level data

The HIP adapter (`make_hip_predict_fn`) wraps the model into the standard `predict_fn` interface with proper tensor handling and device management.

7.3 SCINE/Sparrow: Semi-Empirical Calculator

SCINE/Sparrow [9] (https://scine.ethz.ch/download/) provides CPU-based semi-empirical calculations at DFTB0, PM6, or AM1 levels:

- CPU-only computation (no GPU required)
- Analytical Hessians via SCINE's internal implementation
- Element-specific mass-weighting using SCINE's periodic table
- Significantly faster per evaluation than HIP for small molecules (2.9 s vs. 96.9 s average wall time)

The SCINE adapter (`make_scine_predict_fn`) handles the Python–C++ interface, coordinate transformations (Bohr ↔ Å), and force sign conventions.

7.4 Performance Comparison

	HIP	**SCINE**	**Notes**
Hardware	GPU	CPU	
Avg. wall time/sample	96.9 s	2.9 s	SCINE ∼33× faster
Hessian type	ML-predicted	Analytical	
Differentiable	Yes	No	
Best TS rate (Multi-Mode)	93.3%	100%	
Best TS rate (Sella)	53.3%	66.7%	

SCINE's analytical Hessians appear to be more reliable for saddle point characterization than HIP's ML-predicted Hessians, contributing to SCINE's superior convergence rates.

8 Sella: Trust-Region Saddle Point Optimization

Sella [7] is an established trust-region optimizer for saddle point search implemented in the ASE ecosystem. Key parameters include:

- **delta0**: Initial trust radius
- **sigma_inc, sigma_dec**: Trust radius adjustment factors
- **rho_inc, rho_dec**: Model agreement thresholds
- **fmax**: Force convergence threshold
- **gamma**: Hessian convergence parameter (set to 0 for tightest convergence)

Sella operates in internal coordinates and uses exact Hessians refreshed every step. The Eckart projection is applied before internal coordinate conversion.

8.1 Sella HPO Results

Hyperparameter optimization (500 Optuna trials) with SCINE/DFTB0 revealed:

- Optimal `fmax` = 0.01 (80% Sella convergence, 40% TS rate)
- Strong sensitivity to `sigma_dec` (correlation $r = +0.55$ with success)
- Best single trial: 66.7% TS rate (SCINE), 53.3% (HIP)
- Global average across all trials: 47.0% (SCINE), 30.1% (HIP)

Compared to the Multi-Mode GAD algorithm (94.1% global average, 100% best trial with SCINE), Sella is significantly less effective for noisy starting geometries.

9 Experimental Results

9.1 Dataset and Setup

All experiments use the Transition1x dataset [10] stored in HDF5 format, containing reactant and transition state geometries for small organic molecules. Starting geometries are constructed as:

$$\boldsymbol{x}_0 = \frac{\boldsymbol{x}_{\text{reactant}} + \boldsymbol{x}_{\text{TS}}}{2} + \sigma \cdot \boldsymbol{\xi}, \quad \boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \tag{37}$$

where σ controls the noise level (0.0, 1.0, or 2.0 Å).

Experiments run on the Digital Research Alliance of Canada (DRAC) cluster with 32 CPUs per node, parallelized across 8 workers (4 threads each).

9.2 Multi-Mode GAD HPO Results

Extensive hyperparameter optimization (500 Optuna/TPE trials) yielded:

Metric	HIP	SCINE
Best TS rate (single trial)	93.3%	**100%**
Global TS rate (all samples)	74.9%	**94.1%**
Trials with ≥80% success	45	**497**
Mean wall time	96.9 s	**2.9 s**

9.2.1 Optimal Hyperparameters (SCINE Multi-Mode)

Based on 212 trials achieving 100% TS rate:

Parameter	Optimal Mean	Range
dt	0.00358	[0.0005, 0.005]
dt_max	0.070	[0.01, 0.1]
escape_delta	0.267	[0.05, 0.3]
escape_disp_threshold	5.66×10^{-4}	$[10^{-5}, 10^{-3}]$
escape_neg_vib_std	0.706	[0.1, 1.0]
plateau_patience	9.9	[3, 20]
plateau_boost	2.61	[1.2, 3.0]
adaptive_delta	False	—

9.2.2 The Adaptive Delta Paradox

A surprising finding: fixed escape magnitude (`adaptive_delta=False`) strongly outperforms adaptive scaling. Of SCINE trials achieving 100% success, 99.1% use fixed delta. This suggests that the simple, direction-specific perturbation along $\boldsymbol{v}_2$ is more reliable than curvature-adapted scaling.

9.3 v2 Test Results (Current State)

The latest generation of experiments (v2 tests) with refined implementations achieved headline results:

Method	Convergence	Noise Level
$\boldsymbol{v}_2$ kicking + mode tracking	**100%**	2.0 Å
Adaptive dt control (path-based)	92%	2.0 Å
State-based dt control	80%	2.0 Å

9.4 Comprehensive Method Comparison

Algorithm	Clean	1 Å	2 Å	Key Feature
GAD-Euler (plain)	91.3%	13%	—	Baseline
GAD-RK45	91.3%	$< 5\%$	—	Adaptive RK
Eigenvalue Product	92.7%	$< 5\%$	—	Autograd loss
Direct Descent	99.7%	$\sim 0\%$	—	Sign enforcer
Sella (SCINE)	—	66.7%	—	Trust region
Sella (HIP)	—	53.3%	—	Trust region
Multi-Mode GAD (HIP)	—	93.3%	—	v_2 kick + HPO
Multi-Mode GAD (SCINE)	—	100%	—	v_2 kick + HPO
v_2 Kicking (v2)	—	—	**100%**	Full pipeline
Adaptive dt (v2)	—	—	92%	Path-based
State-based dt (v2)	—	—	80%	Eigenvalue/gradient

9.5 Key Experimental Findings

1. **SCINE consistently outperforms HIP** for saddle point search, despite being CPU-only. Analytical Hessians appear more reliable than ML-predicted ones for eigenvalue characterization.

2. **Multi-Mode GAD dramatically outperforms Sella** on noisy starting geometries: 94.1% vs. 47.0% global TS rate.

3. **Adaptive k = Morse index is fundamentally wrong** for escaping high-index saddles (0% success). Theorem 1 proves this stabilizes the current saddle.

4. **v_2 kicking works as brute-force unsticking**, not principled index reduction. Mean index changes only from 5.39 to 5.00 after kicks.

5. **Only 26% of stalls occur near eigenvalue singularities.** The majority are at genuine high-index saddles where the GAD direction becomes weak.

6. **Fixed perturbation magnitude outperforms adaptive scaling** in 99.1% of successful trials.

7. **All 7 projection variants produce identical results**, confirming that Eckart projection is not a bottleneck.

10 Adjoint Sampling for Reaction Generation

The second major component of `ts-tools` integrates adjoint sampling methods for diffusion-based generative modeling of molecular conformations.

10.1 Problem Formulation

The goal is to sample molecular conformations from a Boltzmann distribution:

$$\mu(\boldsymbol{x}) = \frac{1}{Z} \exp\left(-\frac{1}{\beta} E(\boldsymbol{x})\right), \tag{38}$$

where $E(\boldsymbol{x})$ is the molecular energy function, β is the inverse temperature, and $Z = \int \exp(-E/\beta)\, d\boldsymbol{x}$ is the (unknown) partition function.

10.2 Stochastic Optimal Control Formulation

The sampling task is cast as a stochastic optimal control problem. Consider the controlled SDE:

$$d\boldsymbol{X}_t = \big[\boldsymbol{b}(\boldsymbol{X}_t, t) + \sigma(t)\, \boldsymbol{u}(\boldsymbol{X}_t, t)\big] dt + \sigma(t)\, d\boldsymbol{W}_t, \tag{39}$$

where $\boldsymbol{u}(\cdot, \cdot)$ is a learned control (drift), $\boldsymbol{b}$ is the base drift, and $\sigma(t)$ is the diffusion coefficient. The objective is:

$$\min_{\boldsymbol{u}} \ \mathbb{E}_{p^{\boldsymbol{u}}} \left[\int_0^1 \frac{1}{2} \|\boldsymbol{u}(\boldsymbol{X}_t, t)\|^2 \, dt + g(\boldsymbol{X}_1) \right], \tag{40}$$

where $g(\boldsymbol{x}) = \log p^{\text{base}}(\boldsymbol{x}) + \frac{1}{\beta} E(\boldsymbol{x})$ is the terminal cost encoding the target Boltzmann distribution.

10.3 Reciprocal Adjoint Matching

The practical training loss is the Reciprocal Adjoint Matching (RAM) objective:

$$\mathcal{L}_{\text{RAM}}(\boldsymbol{u}) = \int_0^1 \lambda(t)\, \mathbb{E}\left[\frac{1}{2} \|\boldsymbol{u}(\boldsymbol{X}_t, t) + \sigma(t) \nabla g(\boldsymbol{X}_1)\|^2 \right] dt, \tag{41}$$

where the expectation is over pairs $(\boldsymbol{X}_t, \boldsymbol{X}_1)$ sampled from the current process. The key efficiency gain is that this objective requires sampling *pairs*, not full trajectories, and decouples the terminal distribution sampling from regression updates.

10.4 Algorithm

Algorithm 2 Adjoint Sampling

Require: Energy function $E(\boldsymbol{x})$, temperature β, base process p^{base}

for outer iteration **do**
  Sample terminal states $\{\boldsymbol{X}_1^{(i)}\}$ from current process; compute $\nabla g^{(i)}$
  Store in replay buffer $\mathcal{B}$
  for inner iteration **do**
    Sample $(t^{(j)}, \boldsymbol{X}_t^{(j)}, \nabla g^{(j)})$ from $\mathcal{B}$
    Compute RAM loss: $\|\boldsymbol{u}(\boldsymbol{X}_t^{(j)}, t^{(j)}) + \sigma(t)\nabla g^{(j)}\|^2$
    Update $\boldsymbol{u}$ via optimizer
  end for
end for

10.5 Geometric Extensions for Molecular Systems

The method incorporates molecular symmetries:

- **Graph conditioning:** Equivariant Graph Neural Networks (EGNNs) for molecular-graph-conditioned sampling.
- **Zero center of mass:** Projecting positions to the CoM = 0 subspace.
- **Rotational/translational invariance:** Ensuring G-equivariance of the drift.
- **Periodic boundaries:** Handling toroidal state spaces.

Two conformer sampling approaches are implemented:

1. **Cartesian Adjoint Sampling:** Sample all (x, y, z) coordinates for each atom.
2. **Torsional Adjoint Sampling:** Sample only torsion angles (rotations around bonds), preserving bond structure.

11 Codebase Architecture

The `ts-tools` codebase is organized into the following modules:

```
ts-tools/
  src/
    core_algos/           # Algorithm implementations
      gad.py              # GAD: Euler, RK45, mode tracking
      eigenproduct.py     # Eigenvalue product descent
      signenforcer.py     # Direct eigenvalue descent
    dependencies/         # Shared utilities
      differentiable_projection.py  # Eckart projection (7 variants)
      hessian.py          # Hessian analysis, vibrational frequencies
      calculators.py      # HIP and SCINE adapters
      common_utils.py     # Geometry validation, convergence checks
    runners/              # Integration drivers
      gad_euler_core.py   # GAD-Euler with convergence logic
      gad_rk45_core.py    # GAD-RK45 integration
      eigenvalue_descent_core.py  # Eigenvalue descent driver
```

```
  logging/              # Metrics and visualization
  noisy/                # Noisy-start experiments
    v2_tests/           # Latest generation
      kick_experiments/ # 8 kick strategies
      baselines/        # iHiSD, recursive HiSD, k-HiSD, GD
      runners/          # Parallel experiment runners
      scripts/          # SLURM templates (17 configurations)
      logging/          # Extended metrics
      analysis/         # Singularity and stall analysis
    multi_mode_eckartmw.py      # Production algorithm
  experiments/          # HPO and comparison experiments
    Sella/              # Sella integration and HPO
    2025/               # Multi-mode Eckart-MW experiments
documentation/
  for_robots/           # Machine-readable codebase guide
supporting/             # LaTeX documents and presentations
```

11.1 Key Design Patterns

- **Adapter pattern:** `make_hip_predict_fn()` and `make_scine_predict_fn()` wrap different calculators into a uniform interface.
- **Projection pipeline:** Raw Hessian → mass-weighting → Eckart projection → eigendecomposition → mode tracking → GAD vector.
- **Parallel processing:** `ParallelSCINEProcessor` manages thread pools with per-worker thread limits via environment variables.
- **Convergence criteria:** Eigenvalue product ($\lambda_0\lambda_1 < 0$) combined with force norm ($\|\nabla E\| < \epsilon$).

12 Conclusion

We have presented `ts-tools`, a comprehensive toolkit for transition state search that achieves near-perfect convergence from heavily noisy starting geometries—a regime where conventional methods fail catastrophically. The key innovations are:

1. **v_2 kicking with plateau detection:** A simple but highly effective escape mechanism that perturbs geometry along the second vibrational eigenvector when the dynamics stalls at a high-index saddle point. Combined with mode tracking and adaptive timestep control, this achieves 100% convergence on 2.0 Å noise.
2. **Systematic benchmarking:** We compared GAD-Euler, GAD-RK45, eigenvector following, eigenvalue product descent, direct descent, Sella, iHiSD, recursive HiSD, and multiple escape strategies, providing the most comprehensive comparison to date for noisy-start TS search.
3. **Theoretical insights:** We identified that adaptive k = Morse index is fundamentally wrong for escaping high-index saddles (Theorem 1), that only 26% of stalls are near eigenvalue singularities, and that all projection variants produce equivalent results.
4. **Dual-backend support:** The unified `predict_fn` interface enables seamless switching between ML (HIP) and semi-empirical (SCINE) calculators, with SCINE proving surprisingly superior for saddle point characterization.

5. **Adjoint sampling integration:** Diffusion-based generative models for sampling molecular conformations from Boltzmann distributions, enabling data-driven reaction pathway generation.

The toolkit is designed for integration with generative molecular design pipelines where initial geometry guesses may be far from true saddle points. Future work includes extending to larger molecular systems, incorporating learned escape policies, and coupling with reaction network enumeration.

References

[1] W. E and X. Zhou, The Gentlest Ascent Dynamics, *Nonlinearity*, 24(6):1831, 2011.

[2] J. Yin, L. Zhang, and P. Zhang, High-Index Optimization-Based Shrinking Dimer Method for Finding High-Index Saddle Points, *SIAM J. Sci. Comput.*, 41(6):A3576–A3595, 2019.

[3] J. Yin, Z. Huang, and L. Zhang, Improved High-index Saddle Dynamics with Crossover Dynamics, Preprint, 2024.

[4] C. Eckart, Some Studies Concerning Rotating Axes and Polyatomic Molecules, *Phys. Rev.*, 47:552, 1935.

[5] W. Kabsch, A Solution for the Best Rotation to Relate Two Sets of Vectors, *Acta Crystallogr. A*, 32:922–923, 1976.

[6] A. Levitt and C. Ortner, Convergence and Cycling in Walker-type Saddle Search Algorithms, *SIAM J. Numer. Anal.*, 55(5):2204–2227, 2017.

[7] S. Hermes, M. Wander, and K. R. Biegasiewicz, Sella: A Python Package for Saddle Point Optimizations, 2021.

[8] HIP: High-accuracy Interatomic Potential, https://github.com/burgerandreas/hip, 2024.

[9] SCINE: Software for Chemical Interaction Networks, https://scine.ethz.ch/download/, 2024.

[10] Transition1x Dataset, A dataset of molecular transition states and reaction barriers.