Sella, an open-source automation-friendly molecular saddle point optimizer

Eric D. Hermes, Khachik Sargsyan, Habib N. Najm, and Judit Zádor*

Combustion Research Facility, Sandia National Laboratories, Livermore, CA 94551-0969 USA

E-mail: jzador@sandia.gov

Abstract

We present a new algorithm for the optimization of molecular structures to saddle points on the potential energy surface using a redundant internal coordinate system. This algorithm automates the procedure of defining the internal coordinate system, including the handling of linear bending angles, e.g. through the addition of dummy atoms. Additionally, the algorithm supports constrained optimization using the null-space sequential quadratic programming formalism. Our algorithm determines the direction of the reaction coordinate through iterative diagonalization of the Hessian matrix, and does not require evaluation of the full Hessian matrix. Geometry optimization steps are chosen using the restricted step partitioned rational function optimization method, and displacements are realized using a high-performance geodesic stepping algorithm. This results in a robust and efficient optimization algorithm suitable for use in automated frameworks. We have implemented our algorithm in Sella, an open source software package designed to optimize atomic systems to saddle point structures. We also introduce a new benchmark test comprising 500 molecular structures that approximate saddle point geometries in order to enable comparison of the performance of different saddle point optimization algorithms.

1 Introduction

The recent and upcoming availability of exascale computational resources has created renewed interest in high-performance automated software frameworks for the development of molecular reaction networks. A crucial step in such frameworks is the determination of saddle point geometries for the calculation of reaction rate coefficients under transition state theory. Automation of saddle point optimization requires robust optimization algorithms that converge quickly and reliably to saddle points. This has typically been a source of difficulty in the development of automated frameworks due to the computational expense involved and the rate of convergence failure many algorithms exhibit. Researchers often have to adjust saddle point initial guess structures manually to avoid pathological configurations and improve convergence of the optimization algorithm. There is therefore great need for development of new saddle point optimization algorithms whose reliability and performance is suitable for deployment in automated frameworks.

We have previously developed a method for saddle point optimization of densely connected systems (such as metals and their oxides in condensed phase) in Cartesian coordinates.[1] Our method was shown to converge in less than half the number of steps as compared to the next best performing code on one of the optbench.org saddle point optimization benchmarks.[2] This method was implemented in Sella,[3] an open source Python software package we have developed that has bindings to more than 20 popular electronic structure theory software packages due to its integration with the ASE software package.[4] However, Cartesian coordinates and the related algorithms are not efficient for saddle point searches on the potential energy landscapes of sparsely connected atomic systems, such as molecules, because of the strong coupling between atomic degrees of freedom. Molecular optimization (both for minima and saddle points) is generally performed in a basis of internal coordinates, which typically comprise bond stretch distances, bending angles, and dihedral angles, as internal coordinates are significantly less energetically correlated than Cartesian coordinates.[5–9] These chemically

relevant features provide a more effective coordinate system for optimization at the cost of additional algorithmic complexity. We have recently developed a new method for realizing displacements in a basis of redundant internal coordinates which dramatically reduces the number of steps required to reach convergence on a minimization benchmark.[10] In this work, we extend this algorithm to optimization of saddle point geometries.

Our optimization algorithm is designed to be used in automated frameworks without the need for manual intervention. However, the complexity of optimization in a basis of internal coordinates frequently frustrates attempts at automation. For example, when a bending angle of a molecule becomes linear or nearly linear during the course of optimization, displacement of the bending angle coordinate becomes ill-defined. In order to solve this problem, it is necessary to redefine the internal coordinate system to replace the linear bending angle with a different coordinate. Occasionally, this may require the addition of fictional "dummy atoms" that act as scaffolding off of which the replacement coordinate can be defined. The process of adding dummy atoms and dealing with the additional degrees of freedom they bring to the optimization problem poses a difficulty for automation efforts.

In order to address these issues and to enable efficient optimization of molecular saddle points, our algorithm defines internal coordinates and replaces pathological linear bending angles with more well-defined coordinates. If necessary, our algorithm will add dummy atoms to the system for the purpose of defining new coordinates to replace linear bending angles. Moreover, if previously well-defined bending angles become linear mid-optimization, our algorithm will replace those coordinates without interrupting the optimization procedure. This resolves the common problem of initially well-defined coordinate systems becoming invalid partway through optimization, resulting in a crash or other error. When adding dummy atoms to the optimization problem, our algorithm also adds constraints to counteract the added Cartesian degrees of freedom generated by the dummy atoms. This enables the optimization of a wide range of molecular geometries in an efficient redundant internal coordinate system. This capability renders

our algorithm a highly effective and robust geometry optimizer for automated workflows that explore reactive potential energy surfaces, such as KinBot.[11]

Further, constrained optimizations, such as relaxed scans, are often used to explore aspects of the potential energy surface and are thus also part of automated workflows. Additionally, the presence of dummy atoms necessitates the addition of constraints, as previously described. However, constrained calculations often fail to converge and are an impediment to automation. In order to address this challenge, we have implemented *constrained* saddle point optimization using sequential quadratic programming (SQP), resulting in reliable and efficient constrained optimization trajectories.[12,13]

In the following, we present our methodology to largely overcome the problems mentioned above. In section 2, we describe our saddle point optimization algorithm. This algorithm features the ability to automatically construct a basis of redundant internal coordinates from the Cartesian molecular structure, including the addition of dummy atoms when necessary. Furthermore, we have extended several methods to a basis of redundant internal coordinates, including iterative diagonalization of the Hessian, SQP-based constrained optimization, and restricted-step partitioned rational function optimization (RS-PRFO) for saddle point structures. In section 3 we introduce a new benchmark, demonstrate the performance of our algorithm, and give guidance about its use. We conclude in section 4.

2 The Optimization Algorithm

This section describes our redundant internal coordinate saddle point optimization algorithm, illustrated in figure 1. Section 2.1 describes the algorithm that is used to define the initial redundant internal coordinates. The addition of dummy atoms and how their associated degrees of freedom are handled is discussed in section 2.1.1. Section 2.2 discusses the construction and transformation of the various projected internal coordinate bases that are used in this work, including the nonredundant internal coordinates, the constrained internal coordinates, and the unconstrained internal coordinates. Section

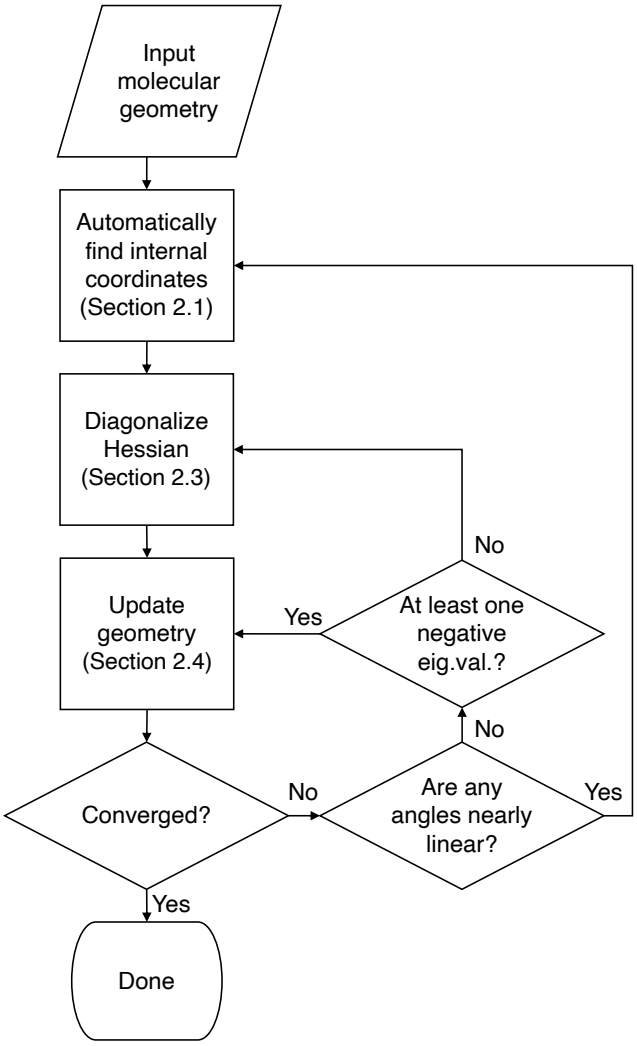


Figure 1: A diagram of our internal coordinate saddle point optimization algorithm as implemented in Sella.

2.3 describes our iterative Hessian diagonalization algorithm, as well as the multi-secant Hessian update formula that is used to construct and update the approximate Hessian matrix. Section 2.4 describes how we have combined the null-space SQP constrained optimization formalism with the RS-PRFO saddle point optimization procedure.

2.1 Automatic Generation of Internal Coordinates

An internal coordinate system describes the geometry of a molecule using the relative positions and orientations of atoms within the molecule, in contrast to the Cartesian coordinate system which describes the geometry using the absolute positions of each atom. The most common types of internal coordinates are the two-atom bond distances, three-atom bending angles, and four-atom dihedral angles.[5,8] A sufficiently large set of these three types of internal coordinates can fully represent all internal degrees of freedom of a molecule. To facilitate automation, our algorithm is capable of automatically determining an appropriate set of internal coordinates. Notably, the algorithm handles systems with linear bending angles by replacing those coordinates with alternate coordinates, adding dummy atoms where necessary. Without this functionality, optimization algorithms in internal coordinates will generally crash due to derivative discontinuities in bending angle degrees of freedom.

There are many ways to construct an internal coordinate system for a molecule. Perhaps the simplest approach is the Z-matrix representation, which defines a *minimal* internal coordinate system. However, the Z-matrix representation necessarily omits chemically relevant features for non-trivial molecules, particularly molecules with rings. This has a deleterious effect on performance, as the excluded chemically relevant features may change drastically after even a small displacement in the Z-matrix coordinate space.[6]

For this reason, our algorithm uses a redundant internal coordinate system that includes all chemically relevant bond stretch, angle bend, and dihedral angle coordinates. Redundant internal coordinate systems increase the complexity of the optimization problem, as the coordinate system is by definition of a higher dimensionality than the

space of all possible molecular configurations. We have described these problems and the methods we developed and implemented in Sella to ameliorate them in a previous work.[10] Our algorithm constructs a complete set of internal coordinates by mapping all possible pairs, triplets, and quadruplets of sequentially-bonded atoms from a molecular graph to bond distance, bending angle, and dihedral angle coordinates, respectively.

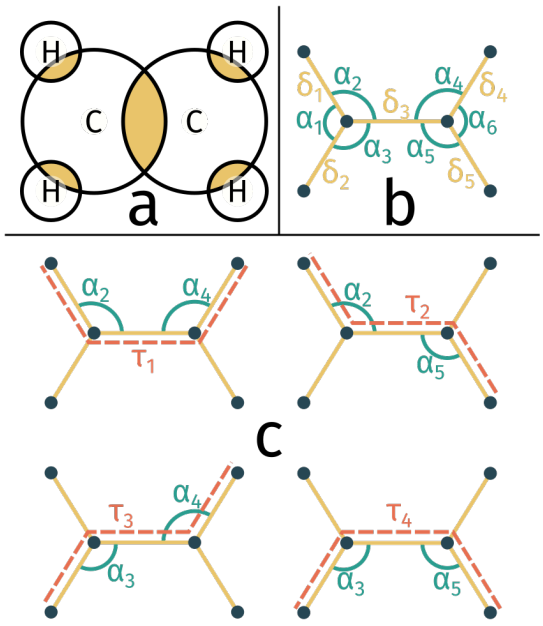


Figure 2: An illustration of our algorithm for identifying internal coordinates. (a) Spheres are drawn around each atom with radius equal to the covalent radius of the element multiplied by a scaling factor. Bonds are added between atoms with overlapping spheres (yellow regions). (b) The molecular graph obtained from this procedure. Nodes of the graph correspond to atoms, while edges (yellow) correspond to bond stretch coordinates. Angle bending coordinates (green) are defined by considering all possible pairs of edges sharing exactly one node. (c) Dihedral angle coordinates (red) are defined by considering all possible pairs of bending angle coordinates sharing exactly one edge, excluding those that would form an improper dihedral.

Our approach for finding internal coordinates, illustrated in figure 2, begins by constructing a molecular graph from the initial Cartesian geometry. This is accomplished

by checking all pairs of atoms, and adding a bond if

$$\|\mathbf{x}_j - \mathbf{x}_i\| < \sigma \left(r_i^{\text{cov}} + r_j^{\text{cov}} \right), \tag{1}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the Cartesian position of atoms i and j, σ is a scalar parameter which in Sella defaults to 1.25, and r_i^{cov} and r_j^{cov} are the covalent radii of atoms i and j.[14] After all pairs of atoms have been checked, a flood fill algorithm is used to count the number of disconnected fragments in the molecular graph. This algorithm works by walking along the graph and noting which atoms belong to the same fragment, then counting the number of distinct fragments. If only a single fragment is found, the algorithm exits; otherwise, σ is increased by 5 % and the above procedure is repeated with the additional rule that only pairs of atoms in different molecular fragments are checked for connectivity. This additional rule helps ensure that geometries with large separation between clusters of atoms do not become densely connected. This procedure continues until only a single molecular fragment is found.

Once the molecular graph has been constructed, the internal coordinates are formed from combinations of the edges of the graph.[8] Each unique graph edge corresponds to a bond stretching coordinate. Pairs of bond stretching coordinates that share exactly one atom are combined to form angle bending coordinates, with the common atom at the center of the angle. Pairs of angle bending coordinates that share exactly one edge are combined to form dihedral angles (pairs of angles that would form improper dihedrals, such as *e.g.* an H(1)-N-H(2)-H(3) dihedral angle in ammonia, are not added automatically).

For systems composed of independent molecules, it may not be chemically meaningful to consider internal coordinates that connect two molecules together. However, the previously described procedure will necessarily add bonds, angles, and dihedrals that span across molecules. This can have a negative impact on performance, as the optimizer will limit the extent to which these chemically irrelevant coordinates can change at each step. To resolve this problem, we also implement as an alternative the

translational and rotation internal coordinates (TRIC) method of Wang and Song.[15] When using the TRIC method, our algorithm will stop adding bonds between atoms after a single iteration of the procedure described above. Each disconnected fragment of the molecular graph is then augmented with translational coordinates of the fragment barycenter and rotational coordinates following the TRIC procedure.

2.1.1 Handling linear angles using dummy atoms

The procedure described above does not account for the possibility of pathological internal coordinates, which can occur when a bending angle becomes nearly linear. This is a problem because linear bending angles do not have a well-defined Cartesian derivative, and dihedral angle coordinates which contain a linear angle are undefined. This problem is typically solved by manually replacing the pathological angle with one or more new coordinates, usually an improper dihedral or in- and out-of-plane bending angles.[8,16] However, if the central atom of the pathological angle is only bonded to two other atoms, then it is not possible to define a chemically meaningful improper dihedral to replace the angle. In that case, it is necessary to define a new, so-called "dummy" atom, which is used to help define the improper dihedral. For this scenario, we have developed an algorithm for automatically adding dummy atoms to a molecule, potentially even mid-optimization. This is a key aspect of our method, as without automatic reconstruction of internal coordinates, some molecular saddle point optimizations will fail due to the coordinate system becoming ill-defined. To the best of our knowledge, no other optimizer implements this functionality.

As dummy atoms are fictitious, the energy of the molecule does not depend on their position, and, therefore, the gradient of the potential energy with respect to the position of dummy atoms is always zero. Because of this, it is possible for the position of the dummy atoms to drift during the course of optimization, reducing the effectiveness of the improper dihedral for which it was added. Our algorithm counteracts this problem by automatically applying three constraints to the position of the dummy atom to ensure it remains in a position that is useful for the optimization. The addition of these

constraints also serves to avoid an increase in the dimensionality of the optimization problem upon addition of dummy atoms, though at the cost of turning an otherwise unconstrained optimization into a constrained optimization problem.

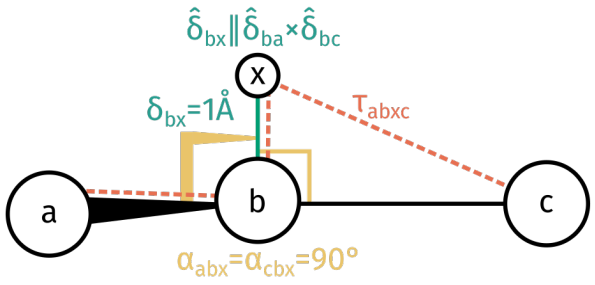


Figure 3: A schematic of how dummy atoms are added to near-linear angles. An angle α_{abc} is considered near-linear when it is above 165°. So long as α_{abc} is not perfectly linear, the dummy atom (x) is added in a direction perpendicular to the plane defined by the angle's three atoms. The bond stretch coordinate δ_{bx} is constrained to 1 Å and the two bending angles α_{abx} and α_{cbx} are constrained to 90°. The nearly-linear angle is replaced by the improper dihedral τ_{abxc}.

Our algorithm automatically places dummy atoms following a precise set of rules that are designed to ensure consistent performance even when a molecule is translated, rotated, or if the atom order is permuted. If a proposed angle α_{abc} is found to be within 15° of linear (0° or 180°) during the construction of internal coordinates, then the angle is not added and an improper dihedral is added instead. If atom b is bonded to at least three atoms, then the improper dihedral τ_{abdc} is added to the coordinate system, where d is chosen to be the bonded atom closest to b excluding a and c. If atom b is bonded only to a and c, a dummy atom is added to the coordinate system, as illustrated in figure 3. The dummy atom x is placed 1 Å away from atom b in the direction of $\hat{\boldsymbol{\delta}}_{ba} \times \hat{\boldsymbol{\delta}}_{bc}$, where $\hat{\boldsymbol{\delta}}$ is the unit bond vector. To counteract the dummy atom's additional Cartesian degrees of freedom, the δ_{bx} bond stretch coordinate is constrained to 1 Å and the α_{abx} and α_{cbx} bending angles are constrained to 90°. The nearly-linear angle α_{abc} is then replaced with the improper dihedral angle τ_{abxc}.

However, if the α_{abc} angle is already too close to linear when the algorithm detects the need for the dummy atom, then this procedure for determining the orientation of the

dummy atom will become numerically unstable. For this reason, if $\|\hat{\boldsymbol{\delta}}_{ba} \times \hat{\boldsymbol{\delta}}_{bc}\|_2 < 10^{-4}$, an alternate procedure is used instead. This procedure places the dummy atom in the direction $\left(\mathbf{I} - \hat{\boldsymbol{\delta}}_{ac}\hat{\boldsymbol{\delta}}_{ac}^T\right)\hat{\mathbf{w}}$, where $\hat{\mathbf{w}}$ is the unit vector of the Cartesian axis that is nearest to being orthogonal with the vector $\hat{\boldsymbol{\delta}}_{ac}$. While this approach loses rotational invariance of the dummy atom positioning, the positioning does maintain translational and permutational invariance.

During the course of an optimization trajectory, if a previously-defined bending angle is found to be within 15° of linear, then the internal coordinate system is discarded and subsequently regenerated by re-running the molecular graph algorithm described in this section.

2.2 Definition and conversion of delocalized coordinate systems

In this work, we extend a method that we have previously developed for saddle point optimization in a Cartesian coordinate basis to a basis of redundant internal coordinates.[1] When using redundant internal coordinates or when optimizing a system with constraints, our algorithm uses a variety of projected internal coordinate bases as well. This section outlines how these projected bases are constructed and the relationships between them.

The Cartesian coordinate vector of a molecule is represented by the $3n$-dimensional vector $\mathbf{x}$, where n is the number of atoms in the molecule. The redundant internal coordinate vector is represented by the m-dimensional vector $\mathbf{q}$, where $m \geq 3n - 6$. Each element of $\mathbf{q}$ corresponds to one of the internal coordinates determined by the algorithm described in section 2.1, and m is the total number of internal coordinates that have been determined. In practice, only the Cartesian coordinate vector $\mathbf{x}$ behaves as an independent variable for the optimization algorithm, and the internal coordinate vector $\mathbf{q}$ is calculated from $\mathbf{x}$. Therefore, all changes to the internal coordinate vector $\mathbf{q}$ are realized as changes to $\mathbf{x}$; this has important implications for the iterative diagonal-

ization algorithm described in section 2.3 and the constrained saddle point optimization algorithm described in section 2.4 below. In this work, all undecorated vector and matrix quantities are defined to be in a basis of redundant internal coordinates with the exception of the Cartesian coordinate vector $\mathbf{x}$.

In addition to the Cartesian and redundant internal coordinate bases, our algorithm also uses a nonredundant basis which is a linear combination of the redundant internal coordinates.[7] We define the nonredundant internal coordinate basis using the singular vectors of the internal–Cartesian Jacobian matrix $\mathbf{J}$ (sometimes called the Wilson B-matrix[17]) as projection matrices,

$$\mathbf{J} = \frac{d\mathbf{q}}{d\mathbf{x}} = \begin{bmatrix} \mathbf{N} & \mathbf{R} \end{bmatrix} \begin{bmatrix} \mathbf{T} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{Z}^T \\ \mathbf{Z}'^T \end{bmatrix}, \tag{2}$$

where $\mathbf{T}$ is the diagonal matrix of non-zero singular values, $\mathbf{N}$ is the $m \times (3n-6)$ orthonormal matrix whose columns span the nonredundant space of $\mathbf{q}$, $\mathbf{R}$ is the $m \times (m-3n+6)$ orthonormal matrix whose columns span the redundant space of $\mathbf{q}$, and $\mathbf{Z}$ and $\mathbf{Z}'$ are the matrices whose columns are the right singular vectors corresponding to $\mathbf{N}$ and $\mathbf{R}$ respectively. We treat singular values of $\mathbf{J}$ below 10^{-6} as zero and assign the corresponding singular vectors to the redundant space.

In this work, vectors and matrices in the nonredundant internal coordinate space will be decorated with a check accent (e.g. $\check{\mathbf{v}}$). Using our definition of the nonredundant internal coordinate system, vectors $\mathbf{v}$ and matrices $\mathbf{M}$ can be transformed between the redundant and nonredundant internal coordinate systems through the relations

$$\check{\mathbf{v}} = \mathbf{N}^T\mathbf{v} \quad \text{and} \quad \mathbf{v} = \mathbf{N}\check{\mathbf{v}} \tag{3}$$

$$\check{\mathbf{M}} = \mathbf{N}^T\mathbf{M}\mathbf{N} \quad \text{and} \quad \mathbf{M} = \mathbf{N}\check{\mathbf{M}}\mathbf{N}^T. \tag{4}$$

The nonredundant internal coordinate system is used primarily in the iterative Hessian diagonalization algorithm described in section 2.3 below.

For constrained optimizations using SQP, our algorithm further subdivides the

nonredundant coordinate system into an unconstrained internal coordinate system and a constrained internal coordinate system. Constraints are represented by the equation

$$\mathbf{c}(\mathbf{q}) = \mathbf{0}, \tag{5}$$

where each element of the vector $\mathbf{c}$ corresponds to an individual constraint expression. The constraints, which may have redundancy, reduce the dimensionality of the optimization problem to $3n-6-p$, where p is less than or equal to the number of constraints. We define the unconstrained and constrained subspaces using the right singular vectors of the constraint–nonredundant Jacobian matrix as projection matrices,

$$\frac{d\mathbf{c}}{d\check{\mathbf{q}}} = \begin{bmatrix} \check{\mathbf{U}} & \check{\mathbf{U}}' \end{bmatrix} \begin{bmatrix} \check{\mathbf{W}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \check{\mathbf{P}}^T \\ \check{\mathbf{Q}}^T \end{bmatrix}, \tag{6}$$

where $\check{\mathbf{W}}$ is the diagonal matrix of non-zero singular values, $\check{\mathbf{P}}$ is the $(3n-6) \times p$ orthonormal matrix whose columns span the constrained subspace of $\check{\mathbf{q}}$, $\check{\mathbf{Q}}$ is the $(3n-6) \times (3n-6-p)$ orthonormal matrix whose columns span the unconstrained subspace of $\check{\mathbf{q}}$, and $\check{\mathbf{U}}$ and $\check{\mathbf{U}}'$ are the matrices whose columns are the left-singular vectors corresponding to $\check{\mathbf{P}}$ and $\check{\mathbf{Q}}$ respectively. The constraint–nonredundant Jacobian can be calculated from the constraint–Cartesian Jacobian and the singular values and right singular vectors of the redundant–Cartesian Jacobian from equation 2,

$$\frac{d\mathbf{c}}{d\check{\mathbf{q}}} = \frac{d\mathbf{c}}{d\mathbf{x}} \left(\frac{d\check{\mathbf{q}}}{d\mathbf{x}} \right)^+ = \frac{d\mathbf{c}}{d\mathbf{x}} \left(\frac{d\mathbf{q}}{d\mathbf{x}} \right)^+ \mathbf{N} = \frac{d\mathbf{c}}{d\mathbf{x}} \mathbf{Z}\mathbf{T}^{-1}\mathbf{N}^T\mathbf{N} = \frac{d\mathbf{c}}{d\mathbf{x}} \mathbf{Z}\mathbf{T}^{-1}, \tag{7}$$

where a superscript plus symbol indicates the Moore–Penrose pseudoinverse.[18,19] As before, we treat singular values below 10^{-6} as zero and assign the corresponding singular vectors to the unconstrained space. Using the projection matrices defined in equations 2 and 6 above, we define projection matrices for transformation directly from the redundant internal coordinates to the constrained and unconstrained internal coordinate

spaces respectively as

$$\mathbf{P} = \mathbf{N}\check{\mathbf{P}} \tag{8}$$

$$\mathbf{Q} = \mathbf{N}\check{\mathbf{Q}}. \tag{9}$$

The projection matrix $\mathbf{N}$ projects from the redundant internal coordinates to the nonredundant internal coordinates, and $\check{\mathbf{P}}$ and $\check{\mathbf{Q}}$ project from the nonredundant internal coordinates to the constrained and unconstrained coordinates, respectively.

In this work, vectors and matrices in unconstrained internal coordinates will be decorated with a tilde accent (e.g. $\tilde{\mathbf{v}}$). Using our definition of the unconstrained internal coordinate system, vectors $\mathbf{v}$ and matrices $\mathbf{M}$ can be converted between redundant internal coordinates and unconstrained internal coordinates through the relations

$$\tilde{\mathbf{v}} = \mathbf{Q}^T\mathbf{v} \quad \text{and} \quad \mathbf{v} = \mathbf{Q}\tilde{\mathbf{v}} \tag{10}$$

$$\tilde{\mathbf{M}} = \mathbf{Q}^T\mathbf{M}\mathbf{Q} \quad \text{and} \quad \mathbf{M} = \mathbf{Q}\tilde{\mathbf{M}}\mathbf{Q}^T. \tag{11}$$

The unconstrained internal coordinate system is used primarily in the geometry optimization algorithm described in section 2.4 below.

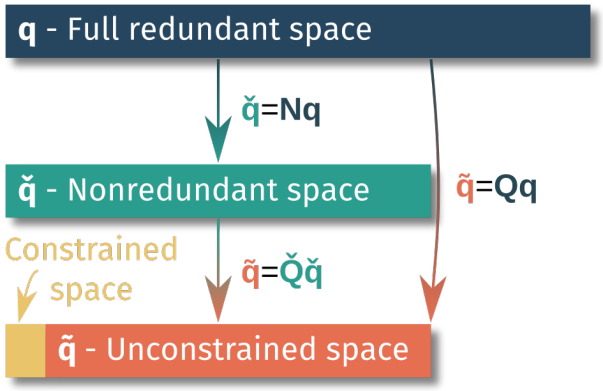


Figure 4: A graphical representation of the relationship between the different internal coordinate systems in this work. The full redundant internal coordinate space $\mathbf{q}$ (dark blue) is projected into the smaller nonredundant space $\check{\mathbf{q}}$ (green) with the projection matrix $\mathbf{N}$. This space is further split into the constrained space (yellow) and the unconstrained space $\tilde{\mathbf{q}}$ (red), which are obtained from $\mathbf{q}$ with the projection matrices $\mathbf{P}$ and $\mathbf{Q}$, respectively.

A summary of the relationships between the various internal coordinate systems is illustrated in figure 4, where the length of the bars correlates with the dimensionality of the coordinate space. As the name implies, the nonredundant internal coordinate space spans only the nonredundant part of the redundant internal coordinates. The constrained and unconstrained spaces together also span the nonredundant space. The redundant internal coordinates can be projected into the nonredundant space, the constrained space, and the unconstrained space with the $\mathbf{N}$, $\mathbf{P}$, and $\mathbf{Q}$ matrices respectively. The nonredundant internal coordinates can be projected into the constrained and unconstrained space with the $\check{\mathbf{P}}$ and $\check{\mathbf{Q}}$ matrices respectively. Note that when performing an unconstrained optimization, the nonredundant and unconstrained spaces are identical and the constrained space is zero-dimensional.

2.3 Iterative Hessian diagonalization in nonredundant internal coordinates

In order to optimize the geometry of a molecule to a first order saddle point, some information about the curvature of the potential energy surface is required. This information is contained within the Hessian matrix, some (or all) of which must be known in order to step towards a saddle point. Optimization to a first order saddle point requires stepping uphill on the potential energy surface in the direction of the reaction coordinate and downhill in all other directions. We define the reaction coordinate as the direction in which the curvature is smallest (most negative or least positive), which is by construction the leftmost eigenvector of the Hessian matrix. Therefore it is at least necessary to know this eigenvector in order to step towards a first order saddle point.

For molecular saddle point optimization, it is common to evaluate the full Hessian matrix at least once at the beginning of optimization. Many electronic structure theory methods such as Hartree Fock and DFT have efficient analytical Hessian implementations. Additionally, transformation of Cartesian Hessians into internal coordinates

is straightforward. When using a level of theory for which efficient analytical second derivatives are not available, the Hessian matrix is usually evaluated in full using finite difference. For computationally expensive levels of theory or for very large molecules, the evaluation of the initial Hessian matrix may become more costly than all subsequent geometry optimization steps. Even when using levels of theory with efficient analytical second derivatives, evaluation of the Hessian matrix will eventually dominate the overall cost of optimization as system sizes grow larger due to the higher computational scaling of Hessian evaluations relative to gradient evaluations. For this reason, saddle point optimization methods which do not require full evaluation of the Hessian matrix may be useful for expensive levels of theory and for large systems.

We have previously developed a saddle point optimization algorithm that efficiently identifies the leftmost eigenvector of the Hessian matrix using iterative diagonalization.[1] Furthermore, this algorithm uses the information obtained by the iterative diagonalization procedure to construct an approximate Hessian that is exact in the subspace searched by the eigensolver. Our method is comparable to that of Sharada, Bell, and Head-Gordon,[20] with the exception that our method incorporates *all* curvature information obtained by the eigensolver into the construction of the approximate Hessian, not just the negative curvature mode. This results in a much more accurate approximate Hessian matrix, and therefore a substantially more efficient optimization algorithm. However, as this method was originally designed only for optimization in a Cartesian coordinate basis, it was not suitable for optimization of *molecular* saddle point geometries. In this work, we extend our iterative diagonalization and approximate Hessian construction methods to optimization in a basis of redundant internal coordinates, potentially with added constraints.

Our optimization algorithm enforces constraints using the SQP formalism, which is based on the method of Lagrange multipliers. The Lagrangian $\mathcal{L}$ is defined as

$$\mathcal{L}\left(\mathbf{q}, \mathbf{w}\right) = \mathcal{E}\left(\mathbf{q}\right) - \mathbf{w}^T \mathbf{c}\left(\mathbf{q}\right), \tag{12}$$

where $\mathcal{E}$ is the potential energy, $\mathbf{c}$ is the vector of constraints, and $\mathbf{w}$ are the corresponding Lagrange multipliers.[21,22]

Algorithm 1 Iterative diagonalization algorithm

procedure RAYLEIGHRITZ
  input: $\mathbf{q}$, $\mathbf{g}$, $\mathbf{B}_k$
  output: $\mathbf{B}_{k+1}$
  if first diagonalization **then**
    $\check{\mathbf{s}}_1 \leftarrow \frac{\check{\mathbf{g}}}{\|\check{\mathbf{g}}\|_2}$
  else
    $\check{\mathbf{s}}_1 \leftarrow$ leftmost eigenvector of $\check{\mathbf{B}}_{\mathcal{L}}$
  $\check{\mathbf{S}}_1 \leftarrow \begin{bmatrix}\check{\mathbf{s}}_1\end{bmatrix}$
  $\check{\mathbf{Y}}_0 \leftarrow []$
  $i \leftarrow 1$
  loop
    evaluate $\check{\mathbf{y}}_i = \check{\mathbf{H}}_{\mathcal{L}}\check{\mathbf{s}}_i$ through forward finite difference (see equation 13)
    $\check{\mathbf{Y}}_i \leftarrow \begin{bmatrix}\check{\mathbf{Y}}_{i-1} & \check{\mathbf{y}}_i\end{bmatrix}$
    diagonalize $\check{\mathbf{Y}}_i^T\check{\mathbf{S}}_i$ to obtain leftmost eigenpair θ_i and $\grave{\mathbf{z}}_i$
    $\check{\mathbf{z}}_i \leftarrow \check{\mathbf{S}}_i\grave{\mathbf{z}}_i$
    $\check{\mathbf{r}}_i \leftarrow \check{\mathbf{Y}}_i\grave{\mathbf{z}}_i - \theta_i\check{\mathbf{z}}_i$
    if $\|\check{\mathbf{r}}_i\|_2 < \gamma|\theta_i|$ or $i = 3n-6$ **then**
      exit and calculate $\mathbf{B}_{k+1}$ using equations 18 through 21
    Solve equation 15 for $\check{\mathbf{t}}_i$
    $\check{\mathbf{t}}_i' \leftarrow \left(\check{\mathbf{I}} - \check{\mathbf{S}}_i\check{\mathbf{S}}_i^T\right)\check{\mathbf{t}}_i$ (in practice, use modified Gram-Schmidt)
    $\check{\mathbf{s}}_{i+1} \leftarrow \frac{\check{\mathbf{t}}_i'}{\|\check{\mathbf{t}}_i'\|_2}$
    $\check{\mathbf{S}}_{i+1} \leftarrow \begin{bmatrix}\check{\mathbf{S}}_i & \check{\mathbf{s}}_{i+1}\end{bmatrix}$
    $i \leftarrow i+1$

The goal of the iterative diagonalization procedure for saddle point optimization, outlined in algorithm 1, is to find the leftmost eigenvector of the Hessian of the potential energy $\mathbf{H}$. For constrained optimization, the goal is instead to find the leftmost eigenvector of the Hessian of the Lagrangian $\mathbf{H}_{\mathcal{L}}$, as this is the Hessian (or approximation thereof) that is used in the optimization procedure (see section 2.4). When performing an optimization in an internal coordinate basis, our algorithm iteratively diagonalizes the Hessian in the *nonredundant* internal coordinate basis $\check{\mathbf{q}}$. One could alternately perform iterative diagonalization of the Hessian in the unconstrained space $\tilde{\mathbf{q}}$. However, it is useful to know the energetic coupling between the constrained and

unconstrained degrees of freedom, particularly for extrapolation of the gradient as in equation 27 below. For this reason, we include constrained degrees of freedom in the iterative Hessian diagonalization procedure.

The iterative diagonalization procedure works by generating an orthonormal matrix $\check{\mathbf{S}}_i$ column-by-column using the Olsen method,[23] which is described below. The columns of the matrix $\check{\mathbf{S}}_i$ are used to define a projected coordinate system, and in this section vectors and matrices in this space are decorated with a grave accent (e.g. $\grave{\mathbf{v}}$). The Hessian of the Lagrangian $\check{\mathbf{H}}_{\mathcal{L}}$ is evaluated in this space using forward finite difference for the energy term, while the constraint term is evaluated exactly,

$$\check{\mathbf{H}}_{\mathcal{L}}\check{\mathbf{s}} \approx \frac{\check{\mathbf{g}}(\check{\mathbf{q}} + \eta\check{\mathbf{s}}) - \check{\mathbf{g}}(\check{\mathbf{q}})}{\eta} - \sum_i w_i \frac{\partial^2 c_i}{\partial \check{\mathbf{q}}^2}\check{\mathbf{s}}, \tag{13}$$

where $\check{\mathbf{s}}$ is one column of $\check{\mathbf{S}}_i$, $\check{\mathbf{g}}$ is the gradient of the potential energy, η is a small number that controls the magnitude of the displacement, and the Lagrange multipliers $\mathbf{w}$ are chosen to be the least-squares solution to the first-order optimality condition,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{q}} = \mathbf{g} - \mathbf{w}^T \frac{\partial \mathbf{c}}{\partial \mathbf{q}} = \mathbf{0}. \tag{14}$$

Note that least-squares solutions to equation 14 will generally have non-zero residual at all points except the final converged geometry. The Hessian-matrix products from equation 13 are stored in the matrix $\check{\mathbf{Y}}_i = \check{\mathbf{H}}_{\mathcal{L}}\check{\mathbf{S}}_i$. Additionally, in order to update the approximate potential energy Hessian matrix $\mathbf{B}$ after the diagonalization algorithm has converged, the *potential energy* Hessian-vector products $\check{\mathbf{H}}\check{\mathbf{S}}_i$ are stored separately in the matrix $\check{\mathbf{Y}}_{\mathcal{E}}$. The iterative diagonalization procedure is considered converged once the space spanned by the columns of $\check{\mathbf{S}}_i$ is found to contain the leftmost eigenvector of $\check{\mathbf{H}}_{\mathcal{L}}$.

As mentioned in section 2.2, displacements in the internal coordinate space must be realized as changes to the Cartesian coordinates $\mathbf{x}$, as these are the independent coordinate variables. In our algorithm, these displacements are realized using a geodesic internal coordinate stepping approach we have previously developed for minimization

in a redundant internal coordinate basis.[10]

At each iteration of the diagonalization algorithm, the part of the Hessian which has already been evaluated $\grave{\mathbf{H}}_i = \check{\mathbf{Y}}_i^T \check{\mathbf{S}}_i$ is exactly diagonalized to find the leftmost eigenvalue θ_i and eigenvector $\grave{\mathbf{z}}_i$. This eigenvector, when projected into the larger nonredundant internal coordinate space through the relation $\check{\mathbf{z}}_i = \check{\mathbf{S}}_i \grave{\mathbf{z}}_i$, approximates the leftmost eigenvector of $\check{\mathbf{H}}_{\mathcal{L}}$. The diagonalization procedure is repeated until the convergence criterion $\|\check{\mathbf{r}}_i\|_2 < \gamma|\theta_i|$ is met, where $\check{\mathbf{r}}_i = \check{\mathbf{Y}}_i \grave{\mathbf{z}}_i - \theta_i \check{\mathbf{z}}_i$ is the residual vector and γ is a unitless convergence parameter which defaults to 0.1 in Sella. This procedure is a particular implementation of the Rayleigh-Ritz method.[24]

Implementations of the Rayleigh-Ritz method differ in how the matrix $\check{\mathbf{S}}_i$ is constructed. Our implementation uses the Olsen method,[23] a method related to the generalized Davidson[25,26] and Jacobi-Davidson[27–29] diagonalization algorithms, which extends the matrix $\check{\mathbf{S}}_i$ with the vector obtained by orthonormalizing the vector $\check{\mathbf{t}}_i$ against $\check{\mathbf{S}}_i$, where $\check{\mathbf{t}}_i$ is the solution to the equation

$$\left(\check{\mathbf{I}} - \check{\mathbf{z}}_i \check{\mathbf{z}}_i^T\right) \left(\check{\mathbf{B}}_{\mathcal{L}} - \theta_i \check{\mathbf{I}}\right) \left(\check{\mathbf{I}} - \check{\mathbf{z}}_i \check{\mathbf{z}}_i^T\right) \check{\mathbf{t}}_i = -\check{\mathbf{r}}_i, \tag{15}$$

where $\check{\mathbf{I}}$ is the identity matrix and $\check{\mathbf{B}}_{\mathcal{L}}$ is the current approximate Hessian of the Lagrangian,

$$\mathbf{B}_{\mathcal{L}} = \mathbf{B} - \sum_i w_i \frac{\partial^2 c_i}{\partial \mathbf{q}^2}, \tag{16}$$

where $\mathbf{B}$ is the approximate Hessian of the potential energy used in the optimization algorithm. In our implementation, the Hessians of the constraints are evaluated in a Cartesian coordinate basis using automatic differentiation and then converted to an internal coordinate basis through the chain rule relation

$$\frac{\partial^2 c_i}{\partial \mathbf{q}^2} = \mathbf{J}^{+T} \left(\frac{\partial^2 c_i}{\partial \mathbf{x}^2} - \sum_j \left[\frac{\partial c_i}{\partial \mathbf{x}} \mathbf{J}^+ \right]_j \frac{\partial^2 q_j}{\partial \mathbf{x}^2} \right) \mathbf{J}^+. \tag{17}$$

Once the iterative diagonalization algorithm has converged, the matrix of displacement vectors $\mathbf{S}$ and the corresponding potential energy Hessian-vector products $\mathbf{Y}_{\mathcal{E}}$

are projected into the full redundant internal coordinate basis and used to update the approximate potential energy Hessian $\mathbf{B}$. This is done with the multi-secant TS-BFGS Hessian update formula,[1,30–33]

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \mathbf{U}\left(\mathbf{Y}_{\mathcal{E}} - \mathbf{B}_k\mathbf{S}\right)^T + \left(\mathbf{Y}_{\mathcal{E}} - \mathbf{B}_k\mathbf{S}\right)\mathbf{U}^T - \mathbf{U}\left(\mathbf{Y}_{\mathcal{E}} - \mathbf{B}_k\mathbf{S}\right)^T\mathbf{S}\mathbf{U}^T \tag{18}$$

$$\mathbf{U} = \mathbf{M}\mathbf{S}\left[\mathbf{S}\mathbf{M}\mathbf{S}\right]^{-1} \tag{19}$$

$$\mathbf{M} = \mathbf{Y}_{\mathcal{E}}\mathbf{Y}_{\mathcal{E}}^T + |\mathbf{B}_k|\mathbf{S}\mathbf{S}^T|\mathbf{B}_k| \tag{20}$$

$$|\mathbf{B}_k| = \sum_i |\lambda_k^{(i)}|\mathbf{v}_k^{(i)}\mathbf{v}_k^{(i)T}, \tag{21}$$

where $\lambda_k^{(i)}$ and $\mathbf{v}_k^{(i)}$ are respectively the ith eigenvalue and eigenvector of $\mathbf{B}_k$. At the beginning of optimization, the approximate potential energy Hessian $\mathbf{B}$ is initialized as a diagonal matrix using the method of Fischer and Almlof.[34]

2.4 Constrained optimization to a saddle point with SQP and RS-PRFO

Constrained optimization of molecular geometries is commonly used in computational chemistry, for example to scan along specific internal coordinates. There are several algorithms for the enforcement of constraints on molecular geometry optimization. RATTLE and SHAKE are two well-known algorithms that rigorously enforce $\mathbf{c}(\mathbf{x}_k) = \mathbf{0}$ for all geometries $\mathbf{x}_k$ visited during the course of the optimization trajectory.[35,36] This is important for molecular dynamics simulations, where conservation of energy requires that all structures of the trajectory lay on the constraint manifold. In contrast, optimization trajectories do not need to conserve energy, and, therefore, only the final converged structure must lie on the constraint manifold. For this reason, we have chosen not to use RATTLE or SHAKE for constrained optimization in our framework. Instead, our algorithm performs constrained optimization using null-space SQP, a method of Lagrange multipliers in which a linear step towards the constraint manifold is made at every iteration of the optimization. Using this approach, intermediate steps

of optimization may not rigorously satisfy the constraint, though the constraint will be satisfied at convergence.

In principle, our algorithm is capable of enforcing any function of the molecular structure as a constraint on the optimization. Currently, Sella supports the fixing of arbitrary atomic distances, bending angles, dihedral angles, and one or more Cartesian components of the barycenter of a cluster of atoms. When applied to a single atom, this last constraint fixes the atom in space.

Algorithm 2 Optimization algorithm

procedure OPTIMIZE
  input: $\mathbf{q}_0$
  output: $\mathbf{q}_{\text{final}}$, the saddle point geometry
  calculate $\mathcal{E}_0, \mathbf{g}_0$ and $\mathbf{c}_0$
  diagonalize $\mathbf{H}_{\mathcal{L}}$ using algorithm 1 to construct $\mathbf{B}_0$
  $k \leftarrow 0$
  repeat
    calculate $\mathbf{s}_P$ by solving equation 25
    if $\|\mathbf{s}_P\|_\infty \geq \epsilon_k$ **then**
      $\mathbf{s}_k \leftarrow \epsilon_k \frac{\mathbf{s}_P}{\|\mathbf{s}_P\|_\infty}$
    else
      calculate $\tilde{\mathbf{s}}_Q$ using equations 30, 31, and 32 with $\alpha = 1$
      if $\|\mathbf{s}_P + \mathbf{s}_Q\|_\infty \leq \epsilon_k$ **then**
        $\mathbf{s}_k \leftarrow \mathbf{s}_P + \mathbf{s}_Q$
      else
        find α such that $\|\mathbf{s}_P + \mathbf{s}_Q(\alpha)\|_\infty = \epsilon_k$
        $\mathbf{s}_k \leftarrow \mathbf{s}_P + \mathbf{s}_Q(\alpha)$
    obtain $\mathbf{q}_{k+1}$ from $\mathbf{q}_k$ and $\mathbf{s}_k$ using geodesic scheme described in reference 10
    calculate $\mathcal{E}_{k+1}, \mathbf{g}_{k+1}$ and $\mathbf{c}_{k+1}$
    update ϵ_k to obtain ϵ_{k+1} using scheme described in reference 1
    obtain $\mathbf{B}_{k+1}$ using geodesic Hessian update scheme described in reference 10
    if $\mathbf{B}_{k+1}$ has no negative eigenvalues **then**
      diagonalize $\mathbf{H}_{\mathcal{L}}$ using algorithm 1 to update $\mathbf{B}_{k+1}$
    $k \leftarrow k + 1$
  until convergence criterion is met, see appendix A

At each iteration of SQP *minimization* algorithms, the coordinates are updated with a vector $\mathbf{s}$ chosen to minimize the quadratic expression[13,37]

$$\mathbf{s}^T\mathbf{g} + \frac{1}{2}\mathbf{s}^T\mathbf{B}_{\mathcal{L}}\mathbf{s} \tag{22}$$

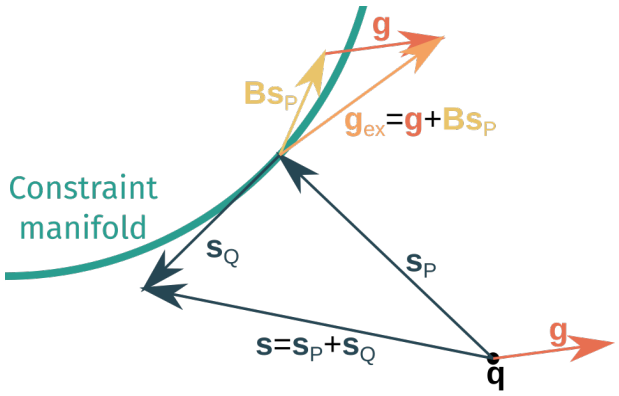


Figure 5: An illustration of the null-space SQP method. The initial point $\mathbf{q}$ has gradient $\mathbf{g}$ (red) and does not lie on the constraint manifold (teal). The constraint-correction displacement vector $\mathbf{s}_P$ leads to the closest point on the constraint manifold. The extrapolated gradient $\mathbf{g}_{\text{ex}}$ (orange) is the sum of the original gradient $\mathbf{g}$ and the approximate gradient $\mathbf{B}\mathbf{s}_P$ (yellow) created by the displacement assuming constant curvature of the potential energy surface. The optimization displacement $\mathbf{s}_Q$ is calculated with this extrapolated gradient in the space tangent to the constraint manifold and orthogonal to $\mathbf{s}_P$. These two displacement vectors are combined into the overall displacement vector $\mathbf{s}$.

subject to the linearized constraint

$$\mathbf{c}\left(\mathbf{q}\right) + \left(\frac{\partial \mathbf{c}}{\partial \mathbf{q}}\right)\mathbf{s} = \mathbf{0}, \tag{23}$$

where $\mathbf{B}_{\mathcal{L}}$ is the approximate Hessian of the Lagrangian defined in equation 16. This minimization problem can be solved in a variety of ways. Our algorithm uses the null-space SQP method,[13,37] illustrated in figure 5, which splits the displacement vector $\mathbf{s}$ into two components,

$$\mathbf{s} = \mathbf{s}_P + \mathbf{s}_Q, \tag{24}$$

where $\mathbf{s}_P$ is the constraint correction displacement vector and $\mathbf{s}_Q$ is the optimization step displacement vector, and $\mathbf{s}_P \perp \mathbf{s}_Q$. This method was chosen as it reduces the dimensionality of the optimization problem by the size of the space spanned by the constraint equations. The vector $\mathbf{s}_P$ is defined as the least-squares solution to equation

23,

$$\left(\frac{\partial \mathbf{c}}{\partial \mathbf{q}}\right) \mathbf{s}_P = -\mathbf{c}(\mathbf{q}). \tag{25}$$

In null-space SQP *minimization* algorithms, the vector $\mathbf{s}_Q$ is found by minimizing the projected quadratic approximation

$$\tilde{\mathbf{s}}_Q^T \tilde{\mathbf{g}}_{\text{ex}} + \frac{1}{2} \tilde{\mathbf{s}}_Q^T \tilde{\mathbf{B}}_{\mathcal{L}} \tilde{\mathbf{s}}_Q, \tag{26}$$

where

$$\tilde{\mathbf{g}}_{\text{ex}} = \tilde{\mathbf{g}} + \mathbf{Q}^T \mathbf{B} \mathbf{s}_P \tag{27}$$

is the gradient which has been first approximately extrapolated to the point $\mathbf{q} + \mathbf{s}_P$ and then projected into the unconstrained space.

Note that equation 26 is the traditional quadratic approximation to the potential energy surface, but with the gradient $\tilde{\mathbf{g}}$ replaced by the extrapolated gradient $\tilde{\mathbf{g}}_{\text{ex}}$ and the approximate Hessian of the potential energy $\tilde{\mathbf{B}}$ replaced by the approximate Hessian of the Lagrangian $\tilde{\mathbf{B}}_{\mathcal{L}}$. Therefore, the procedure for obtaining $\mathbf{s}_Q$ described above is equivalent to standard quasi-Newton minimization with some modifications to the gradient and approximate Hessian. It is possible to perform a constrained minimization using an alternative approach such as the restricted step rational function optimization (RS-RFO) method by making these same substitutions.[38–40] This is useful, as unlike quasi-Newton, RS-RFO guarantees a downhill step even when the approximate Hessian $\tilde{\mathbf{B}}_{\mathcal{L}}$ is indefinite. Furthermore, making these substitutions in the restricted step partitioned rational function optimization (RS-PRFO) method will result in a constrained *saddle point* optimization step.[38–40] Our algorithm implements constrained saddle point optimization by combining null-space SQP with RS-PRFO in this way.

It is illustrative to explain how null-space SQP can be implemented into RS-RFO before explaining how to implement it in the RS-PRFO. In RS-RFO, the vector $\mathbf{s}_Q$ is

chosen to be the minimizer of the rational function

$$\mu = \frac{\tilde{\mathbf{s}}_Q^T \tilde{\mathbf{g}}_{\text{ex}} + \frac{1}{2}\tilde{\mathbf{s}}_Q^T \tilde{\mathbf{B}}_{\mathcal{L}} \tilde{\mathbf{s}}_Q}{1 + \alpha^{-2}\tilde{\mathbf{s}}_Q^T \tilde{\mathbf{s}}_Q}, \tag{28}$$

where α is a parameter that controls the magnitude of the displacement. The value of α is chosen to satisfy our trust region condition, which we describe below. Note that as with the standard SQP minimization method described above, we have replaced the gradient $\tilde{\mathbf{g}}$ with the extrapolated gradient $\tilde{\mathbf{g}}_{\text{ex}}$ and the approximate Hessian of the potential energy $\tilde{\mathbf{B}}$ with the approximate Hessian of the Lagrangian $\tilde{\mathbf{B}}_{\mathcal{L}}$. This will result in a displacement vector $\mathbf{s}_Q$ which is suitable for use in the null-space SQP minimization formalism. The right-hand side of equation 28 can be minimized by solving an eigenvalue problem,[38]

$$\begin{pmatrix} \alpha^2 \tilde{\mathbf{B}}_{\mathcal{L}} & \alpha \tilde{\mathbf{g}}_{\text{ex}} \\ \alpha \tilde{\mathbf{g}}_{\text{ex}}^T & 0 \end{pmatrix} \begin{pmatrix} \alpha^{-1} \tilde{\mathbf{s}}_Q \\ 1 \end{pmatrix} = 2\mu \begin{pmatrix} \alpha^{-1} \tilde{\mathbf{s}}_Q \\ 1 \end{pmatrix}, \tag{29}$$

where a minimization step is obtained from the leftmost solution to equation 29. In theory, choosing to define $\mathbf{s}_Q$ using the jth eigenvector of equation 29 will result in a step towards an order-j saddle point, but in practice this approach often fails to converge.

In the RS-PRFO method,[38–40] the unconstrained space $\tilde{\mathbf{q}}$ is split into maximization and minimization subspaces which are defined using the eigenvectors of $\tilde{\mathbf{B}}_{\mathcal{L}}$. The goal of RS-PRFO is to find a step that maximizes the energy in the maximization subspace while minimizing the energy in the minimization subspace. In order to optimize to an order-j saddle point, the j leftmost eigenvectors of $\tilde{\mathbf{B}}_{\mathcal{L}}$ are assigned to the columns of $\tilde{\mathbf{V}}_{\mathcal{L}}^{(\text{max})}$ and the remaining eigenvectors are assigned to the columns of $\tilde{\mathbf{V}}_{\mathcal{L}}^{(\text{min})}$. In particular, optimization to a first-order saddle point will assign only the leftmost eigenvector of $\tilde{\mathbf{B}}_{\mathcal{L}}$ to $\tilde{\mathbf{V}}_{\mathcal{L}}^{(\text{max})}$. $\tilde{\mathbf{V}}_{\mathcal{L}}^{(\text{max})}$ and $\tilde{\mathbf{V}}_{\mathcal{L}}^{(\text{min})}$ behave as projection matrices from the unconstrained space to the maximization and minimization subspaces, respectively. The step in the *maximization* subspace is found by calculating the *rightmost* eigenvector of the

eigenvalue problem

$$\begin{pmatrix} \alpha^2 \tilde{\mathbf{B}}_{\mathcal{L}}^{(\max)} & \alpha \tilde{\mathbf{g}}_{\text{ex}}^{(\max)} \\ \alpha \tilde{\mathbf{g}}_{\text{ex}}^{(\max)T} & 0 \end{pmatrix} \begin{pmatrix} \alpha^{-1} \tilde{\mathbf{s}}_Q^{(\max)} \\ 1 \end{pmatrix} = 2\mu^{(\max)} \begin{pmatrix} \alpha^{-1} \tilde{\mathbf{s}}_Q^{(\max)} \\ 1 \end{pmatrix}, \tag{30}$$

where $\tilde{\mathbf{B}}_{\mathcal{L}}^{(\max)} = \tilde{\mathbf{V}}_{\mathcal{L}}^{(\max)T} \tilde{\mathbf{B}}_{\mathcal{L}} \tilde{\mathbf{V}}_{\mathcal{L}}^{(\max)}$ and $\tilde{\mathbf{g}}_{\text{ex}}^{(\max)} = \tilde{\mathbf{V}}_{\mathcal{L}}^{(\max)T} \tilde{\mathbf{g}}_{\text{ex}}$ are the approximate Hessian and extrapolated gradient projected into the maximization subspace, respectively. The step in the *minimization* subspace is found by calculating the *leftmost* eigenvector of the eigenvalue problem

$$\begin{pmatrix} \alpha^2 \tilde{\mathbf{B}}_{\mathcal{L}}^{(\min)} & \alpha \tilde{\mathbf{g}}_{\text{ex}}^{(\min)} \\ \alpha \tilde{\mathbf{g}}_{\text{ex}}^{(\min)T} & 0 \end{pmatrix} \begin{pmatrix} \alpha^{-1} \tilde{\mathbf{s}}_Q^{(\min)} \\ 1 \end{pmatrix} = 2\mu^{(\min)} \begin{pmatrix} \alpha^{-1} \tilde{\mathbf{s}}_Q^{(\min)} \\ 1 \end{pmatrix}, \tag{31}$$

where $\tilde{\mathbf{B}}_{\mathcal{L}}^{(\min)} = \tilde{\mathbf{V}}_{\mathcal{L}}^{(\min)T} \tilde{\mathbf{B}}_{\mathcal{L}} \tilde{\mathbf{V}}_{\mathcal{L}}^{(\min)}$ and $\tilde{\mathbf{g}}_{\text{ex}}^{(\min)} = \tilde{\mathbf{V}}_{\mathcal{L}}^{(\min)T} \tilde{\mathbf{g}}_{\text{ex}}$ are the approximate Hessian and extrapolated gradient projected into the minimization subspace, respectively. The final unconstrained displacement vector is calculated as

$$\tilde{\mathbf{s}}_Q = \tilde{\mathbf{V}}_{\mathcal{L}}^{(\max)} \tilde{\mathbf{s}}_Q^{(\max)} + \tilde{\mathbf{V}}_{\mathcal{L}}^{(\min)} \tilde{\mathbf{s}}_Q^{(\min)}, \tag{32}$$

and the overall displacement vector is obtained according to equation 24.

Once a displacement vector $\mathbf{s}$ has been chosen, either for a geometry optimization step or for the iterative Hessian diagonalization scheme described in section 2.3, the molecular geometry must be updated. This process is trivial when using Cartesian coordinates, as the displacement vector can simply be added to the current Cartesian coordinate vector $\mathbf{x}$ to update the molecular geometry. In redundant internal coordinates, however, the point in coordinate space specified by $\mathbf{q} + \mathbf{s}$ may not correspond to a physically possible combination of internal coordinates. This is because the set of all physically possible coordinate vectors forms a $(3n-6)$-dimensional manifold embedded in the larger m-dimensional redundant internal coordinate space. Moreover, this manifold has curvature that arises from the coupling between the internal coordinates. As

a result, even if the starting point $\mathbf{q}$ lies on the manifold and the displacement vector $\mathbf{s}$ lies tangent to the manifold at $\mathbf{q}$, the point $\mathbf{q}+\mathbf{s}$ will generally not lie on the manifold of physically allowed configurations.

In practice, this problem is usually solved by finding the point $\mathbf{x}'$ in Cartesian space which corresponds to the valid coordinate $\mathbf{q}'$ that is closest to $\mathbf{q}+\mathbf{s}$ in redundant internal coordinate space. This can be accomplished by iterating the Newton-Raphson root-finding algorithm until convergence,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{J}_k^{+} \left(\mathbf{q} + \mathbf{s} - \mathbf{q}_k\right), \tag{33}$$

where $\mathbf{q}_k$ and $\mathbf{J}_k$ are the internal coordinate vector and internal–Cartesian Jacobian matrix corresponding to $\mathbf{x}_k$, respectively. While this approach is computationally facile and generally converges in only a few iterations, it does not account for the curvature of the space of valid internal coordinates, which results in poor quality displacements when using highly redundant coordinate systems, for example for molecules with rings. We have previously developed an alternate approach for realizing displacements in redundant internal coordinate systems based on geodesics of the curved internal coordinate manifold.[10] This geodesic approach accounts for the correlation between internal coordinates not only at the endpoints of the displacement, but also throughout the entire displacement trajectory. We have shown that this geodesic displacement strategy dramatically reduces the number of steps required to reach convergence as compared to the traditional Newton displacement method. Following each geometry optimization step, the approximate Hessian is updated using the geodesic secant update, which is also described in our previous work.[10]

As mentioned previously, the parameter α in equation 29 (for minimization) and equations 30 and 31 (for saddle point optimization) is chosen to satisfy a trust region condition. In our trust region algorithm, a displacement vector $\mathbf{s}_k$ is accepted only if it

satisfies the trust region condition,

$$\|\mathbf{s}_k\|_\infty \leq \epsilon_k, \tag{34}$$

where ϵ_k is the size of the trust region at iteration k of the optimization. Note that this differs from our previous work in that the infinity-norm is used in the trust region condition, rather than the 2-norm.[1] This is important for optimization in a redundant internal coordinate basis, as using the 2-norm would result in smaller displacements upon addition of new redundant coordinates to the coordinate system. This would unnecessarily slow optimization in systems with a high degree of redundancy in their internal coordinate system, as these redundant coordinates will result in larger displacement vectors in the 2-norm sense, but not necessarily in the infinity-norm sense.

The means by which equation 34 is enforced depends on the relative magnitudes of $\mathbf{s}_P$ and $\mathbf{s}_Q$. If $\|\mathbf{s}_P\|_\infty \geq \epsilon_k$, then the constraint correction step by itself is too large to fit within the trust region, and thus the displacement step is chosen to be

$$\mathbf{s}_k = \frac{\epsilon_k}{\|\mathbf{s}_P\|_\infty}\mathbf{s}_P, \tag{35}$$

meaning that no optimization step is taken in the unconstrained space. In this scenario, $\mathbf{s}_Q$ is not used or even calculated at this iteration of the optimization algorithm. If $\|\mathbf{s}_P\|_\infty < \epsilon_k$, then $\mathbf{s}_Q$ is calculated with an initial value of $\alpha = 1$. If $\|\mathbf{s}_P + \mathbf{s}_Q\|_\infty \leq \epsilon_k$, then the overall displacement $\mathbf{s}_k = \mathbf{s}_P + \mathbf{s}_Q$ is accepted. Otherwise, a value of α is determined for which $\|\mathbf{s}_P + \mathbf{s}_Q(\alpha)\|_\infty = \epsilon_k$ using the Newton-Raphson root-finding algorithm.

After every optimization step, our algorithm checks to see whether the trust region size ϵ_k needs to be updated. This is accomplished using a scheme we have previously developed for saddle point optimization in a Cartesian coordinate basis.[1]

3 Results

3.1 Sella benchmark

In order to measure the performance of our internal coordinate saddle point optimization algorithm, we have created a benchmark test set of 500 molecular geometries generated by KinBot.[11] This test set consists of small organic molecules with between 7 and 25 atoms in configurations that approximate saddle points for a set of reaction families. Each molecular geometry in the test set was optimized to a saddle point structure in a basis of internal coordinates which were determined automatically by Sella. Calculations were performed with NWChem at the Hartree Fock level of theory with the 3-21G basis for computational expedience.[41,42] Open-shell molecules were calculated at the unrestricted Hartree Fock level of theory instead. The molecular geometry optimization was considered converged once the maximum atomic force in Cartesian coordinates dropped below $0.01\,\mathrm{eV\,\AA^{-1}}$ (see appendix A). Initial molecular geometries and a Python script for performing the optimizations are available in the supplementary material.

The overall performance of our optimization algorithm on the 500-molecule benchmark is summarized in figure 6. The left half of this figure is a scatter plot that represents the total number of gradient evaluations required to converge to a first order saddle point plotted against the number of atoms in each molecule of the 500-molecule test set. The right half of figure 6 is a histogram of the number of gradient evaluations across the entire test set. With our chosen optimization parameters, the molecules in this test set require between 19 and 125 gradient evaluations to reach convergence, with an average of 50 gradient evaluations.

The majority of molecules in our benchmark set (81.6 %, green) do not have any linear bending angles at any point along the optimization trajectory. Of the remaining molecules, most (9.6 %, yellow) have a linear bending angle in the initial geometry, but do not require the addition of a dummy atom. Optimization performance for these molecules is approximately the same as molecules with no linear bending angles,

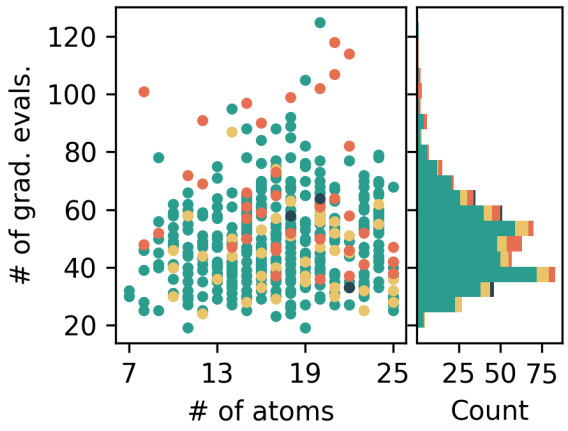
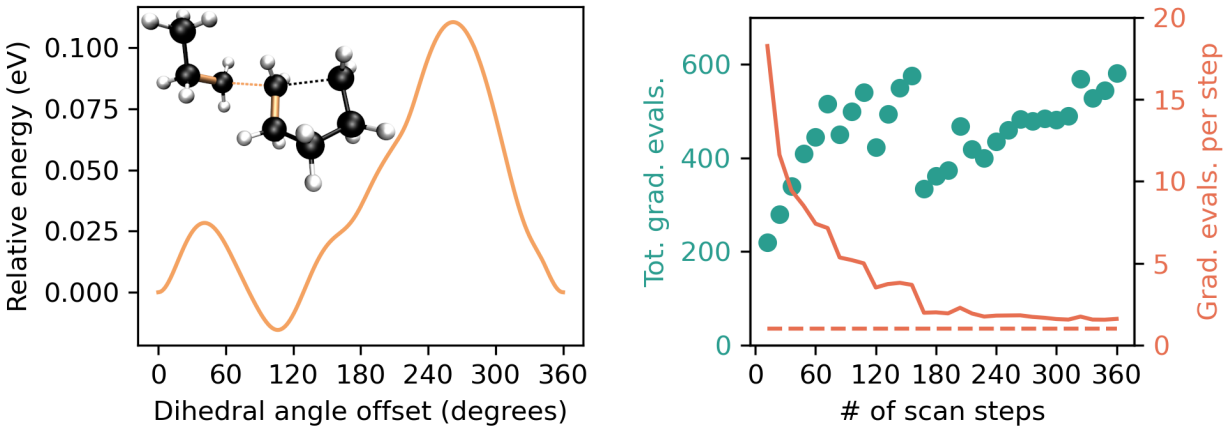


Figure 6: A scatter plot and histogram depicting the number of gradient evaluations required to converge to a saddle point against the number of atoms in each molecule in the 500-molecule benchmark set. Green dots correspond to molecules with no linear bending angles, yellow to molecules with at least one linear angle at the beginning of optimization but without the need for dummy atoms, black dots correspond to molecules with at least one linear angle at the beginning of optimization that required a dummy atom, and red dots correspond to molecules that required a new set of internal coordinates be constructed during optimization due to a previously valid angle becoming linear.

suggesting our algorithm's approach of replacing linear bending angles with improper dihedrals is highly efficient. Only four of the molecules in this test set (0.8 %, black) required the addition of a dummy atom. There is not enough information available in this benchmark set to indicate how optimization performance is affected by the addition of dummy atoms, but the four molecules with dummy atoms in this test set are not outliers. The remaining molecules (8.0 %, red) acquired a linear bending angle partway through optimization, and therefore required the internal coordinate system to be reconstructed from scratch. When our algorithm discards and reconstructs the internal coordinate system, the Hessian matrix must also be re-diagonalized. This results in an increase in the total number of gradient evaluations required to reach convergence on average for these systems, which can be observed in figure 6, though these points do not appear to be outliers.

It is clear from these data that there is no strong correlation between molecule size and the number of gradient evaluations required to reach convergence. This is

perhaps not surprising, considering the small range of molecule sizes included in this study, though it is not at all a guaranteed property. For optimizations in a basis of redundant internal coordinates, it is crucial to use a trust region condition that scales appropriately with the system size, rather than the dimensionality of the basis. If one were to use a 2-norm based trust region condition, the apparent magnitude of any given displacement will increase if additional redundant coordinates are added to the coordinate system. It is for this reason that we choose to use the infinity-norm for the trust region condition, as displacements are limited by only the largest-changing component, and thus the addition of more redundant coordinates will generally not change the displacement magnitude.



(a) (b)

Figure 7: A relaxed dihedral angle scan for a transition state structure. (a) The electronic potential energy of the transition state as a dihedral angle is scanned. A depiction of the molecule is inset with the three bonds that make up the dihedral angle highlighted in orange. (b) The total number of gradient evaluations (green) and the number of gradient evaluations per scan steps (red) versus the number of scan steps. The red dashed line corresponds to theoretical lower bound of 1 gradient evaluations per scan step.

It is common in computational studies of molecules to perform scans over specific internal coordinates. These scans are a series of constrained optimizations, which are straightforward when the goal is to minimize the energy with respect to the unconstrained degrees of freedom, e.g., when the scan is done on a minimum energy structure. However, special care must be taken for saddle point scans to ensure that the scan tra-

jectory preserves the structure of the reacting region of the molecule and does not fall into a different saddle point geometry. Our algorithm's implementation of null-space SQP saddle point optimization provides an easy and effective way to scan coordinates. Figure 7 depicts a full 360° scan of a dihedral angle of a 5-exo-tet cyclization reaction saddle point. The geometry of the saddle point and relative potential energy of the dihedral scan is shown in figure 7a. The orange-highlighted bonds in figure 7a indicate the dihedral angle that is being scanned, which includes several atoms involved in the reaction corresponding to this saddle point. This scan reveals that the structure found by Sella (the structure at zero degrees) is not the lowest-energy conformer for this reaction. This is not surprising, as our saddle point optimization algorithm tends to converge to the saddle point that is geometrically closest to the initial structure, and this may not be the lowest energy conformer.

Figure 7b shows how the total number of gradient evaluations required to perform the entire scan (in green) and the number of gradient evaluations per scan step (in red) scale as a function of the number of scan steps used. For this scan, between 12 and 360 steps were used, which corresponds to a dihedral angle step size of between 30° and 1°. Though the total number of gradient evaluations tends to increase with the number of steps used in the scan, the relationship is not perfectly linear. One possible explanation for this is that larger steps in the scan will tend to perturb the unconstrained degrees of freedom more, therefore requiring more gradient evaluations to converge. The upper limit of this plot indicates that when 360 1° steps are used, only 1.61 gradient evaluations are required per grid point. This is very close to the theoretical lower limit of 1 gradient evaluation per step (the dashed red line), in which each step of the scan converges in a single gradient evaluation. Such performance is only possible due to our use of null-space SQP constrained optimization, which projects the gradient to constraint-corrected geometry before performing any gradient evaluations (see equation 27). The success of this projection approach depends entirely on the accuracy of the approximate Hessian, and so the performance observed in this test system indicates that the initial iterative Hessian diagonalization and TS-BFGS Hessian updates are

Listing 1: A minimal Sella script for optimizing a molecule to a saddle point in internal coordinates.

```
#!/usr/bin/env python3

from ase.io import read
from ase.calculators.nwchem import NWChem
from sella import Sella

atoms = read('input_structure.xyz')
atoms.calc = NWChem()
opt = Sella(atoms, internal=True)
opt.run(fmax=0.01)
```

highly effective.

3.2 Using Sella

Sella is an open source Python software package that is available on both github.com[3] and from the Python Package Index (PyPI). Additionally, the exact version of Sella used to run these calculations is available from Zenodo.[43]

Listing 1 contains a minimal Python script that uses Sella to perform saddle point optimization on a molecule in an internal coordinate basis. An initial geometry contained in `input_structure.xyz` is read using ASE's IO routines.[4] An ASE calculator for NWChem is attached to the resulting ASE Atoms object using all default settings, which will perform a calculation at the HF/3-21G level of theory. This level of theory can be changed by adding arguments to the `NWChem` calculator object constructor. A Sella optimizer object is then created for the ASE Atoms object, with optimization in internal coordinates requested explicitly. Finally, the optimization to a saddle point is run with a convergence criteria of a maximum atomic force of $0.01\,\mathrm{eV\,\AA^{-1}}$.

The optimization algorithm implemented in Sella has several user-tunable hyperparameters that may be passed to the `Sella` class. One important parameter is the iterative diagonalization convergence criterion γ, which may be set with the keyword argument `gamma`. By default, $\gamma = 0.1$, though tighter convergence may be requested by

Listing 2: A more complex Sella script for optimizing a molecule to a saddle point in internal coordinates.

```
#!/usr/bin/env python3

import numpy as np
from ase.io import read
from ase.calculators.nwchem import NWChem
from ase.calculators.socketio import SocketIOCalculator
from sella import Sella

socket_name = 'nwchem'
atoms = read('input_structure.xyz')
nwchem = NWChem(
    driver={'socket': {'unix': socket_name}},
    task='optimize',
)
with SocketIOCalculator(nwchem, unixsocket=socket_name) as calc:
    atoms.calc = calc
    opt = Sella(atoms, internal=True)
    for i, converged in enumerate(opt.irun(fmax=0)):
        print(f'The energy at iteration {i} is {atoms.get_potential_energy()}')
        if atoms.get_distance(0, 1) > 2.0:
            raise RuntimeError('The 0-1 bond broke!')
        if np.abs(opt.pes.get_g()).max() < 1e-2:
            break
```

reducing γ. Reducing γ will result in additional iterative Hessian diagonalization steps, but will result in a more accurate approximate Hessian and therefore better subsequent geometry optimization steps. The initial trust region size ϵ_0 defaults to a value of 0.1 (with units of Å for bond stretch coordinates and radians for angle coordinates). This parameter may be changed by using the keyword argument `delta0`. Increasing the trust region size will result in larger step sizes, though at the cost of increasing the likelihood of a poor step. Sella is also capable of performing minimization or optimization to higher-order saddle points. The desired saddle point order may be specified with the `order` keyword argument. This defaults to 1, corresponding to optimization to a first order saddle point. Minimization can be requrested by setting `order` to 0. Additional keyword arguments are outlined in the Sella manual, available on github.[3]

Listing 2 contains a more complex example Python script that uses Sella to optimize a molecule to a saddle point. First, this example uses the i-PI socket protocol to communicate with the NWChem executable.[44] This speeds up the overall calculation time by reusing the same NWChem processes for all energy and gradient calculations, rather than starting a new instance of NWChem for each single point. Other electronic structure theory codes that support the i-PI socket protocol include Quantum Espresso,[45,46] Siesta,[47,48] Abinit,[49,50] and FHI-aims.[51] Additionally, this example uses the `irun` iterator for running the optimization, rather than the `run` function. The use of `irun` makes it possible to run code at every iteration of the optimization trajectory. This enables users to perform more sophisticated logging, monitor the structure as it is optimizing, or implement their own custom convergence criteria.

4 Conclusion

We have presented a novel algorithm for the automated optimization of molecules to saddle points in a basis of redundant internal coordinates. Our algorithm automatically constructs internal coordinates from the Cartesian positions of the atoms, and automatically replaces pathological linear angles with improper dihedrals. When necessary, our algorithm automatically adds dummy atoms to the system, and adds constraints to ensure that the dummy atom does not drift unnecessarily over the course of optimization. To determine the direction of the reaction coordinates, our algorithm iteratively diagonalizes the Hessian in a basis of internal coordinates. The potential energy surface curvature information obtained during the iterative diagonalization procedure is used to construct a partially-exact Hessian matrix. This Hessian matrix is used in a state-of-the-art constrained partitioned rational function optimization algorithm to step towards a saddle point geometry. Our approach enables efficient high-throughput automated saddle point optimization of molecular species. The performance of our algorithm was tested on a newly created benchmark set of 500 molecular geometries that are geometrically close to reactive saddle points.

Acknowledgement

This work was done within the Exascale Catalytic Chemistry (ECC) Project, which is supported by the U.S. Department of Energy, Office of Science, Basic Energy Sciences, Chemical Sciences, Geosciences and Biosciences Division, as part of the Computational Chemistry Sciences Program.

We thank Dr. Mohan Sarovar for reviewing this manuscript and for providing several helpful comments regarding its presentation.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. The views expressed in the article do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

Supporting Information Available

The 500 initial molecular geometries used in figure 6 and an example Python script for optimizing those structures to a saddle point using Sella are available in the SI.

A Geometry Optimizer Convergence Criterion

For consistency with ASE's built-in optimizer classes, Sella's optimization algorithm uses the maximum atomic force as the default convergence criterion. Using this criterion, convergence is achieved once the largest *Cartesian* force experienced by any atom in the molecule drops below a threshold value,

$$\left\| \frac{d\mathcal{E}}{d\mathbf{x}_i} \right\|_2 \leq f_{\max} \quad \text{for all} \quad i = 1, \ldots, n, \tag{36}$$

where $\mathbf{x}_i$ represents the position of atom i. When using constraints, it is necessary to first project out the components of the gradient in the constraint subspace. This is accomplished by projecting the gradient in the unconstrained space $\tilde{\mathbf{g}}$ back to Cartesian coordinates,

$$\left(\frac{d\mathcal{E}}{d\mathbf{x}}\right)_{\text{proj}} = \mathbf{J}^T\mathbf{Q}\tilde{\mathbf{g}}. \tag{37}$$

References

(1) Hermes, E. D.; Sargsyan, K.; Najm, H. N.; Zádor, J. Accelerated Saddle Point Refinement through Full Exploitation of Partial Hessian Diagonalization. *Journal of Chemical Theory and Computation* **2019**, *15*, 6536–6549.

(2) Chill, S. T.; Stevenson, J.; Ruehle, V.; Shang, C.; Xiao, P.; Farrell, J. D.; Wales, D. J.; Henkelman, G. Benchmarks for Characterization of Minima, Transition States, and Pathways in Atomic, Molecular, and Condensed Matter Systems. *J. Chem. Theory Comput.* **2014**, *10*, 5476–5482.

(3) Sella. 2022; `https://github.com/zadorlab/sella`, original-date: 2019-04-09T15:16:08Z.

(4) Larsen, A. H.; Mortensen, J. J.; Blomqvist, J.; Castelli, I. E.; Christensen, R.; Dułak, M.; Friis, J.; Groves, M. N.; Hammer, B.; Hargus, C.; Hermes, E. D.; Jennings, P. C.; Jensen, P. B.; Kermode, J.; Kitchin, J. R.; Kolsbjerg, E. L.; Kubal, J.; Kaasbjerg, K.; Lysgaard, S.; Maronsson, J. B.; Maxson, T.; Olsen, T.; Pastewka, L.; Peterson, A.; Rostgaard, C.; Schiøtz, J.; Schütt, O.; Strange, M.; Thygesen, K. S.; Vegge, T.; Vilhelmsen, L.; Walter, M.; Zeng, Z.; Jacobsen, K. W. The atomic simulation environment—a Python library for working with atoms. *Journal of Physics: Condensed Matter* **2017**, *29*, 273002.

(5) Schlegel, H. B. A comparison of geometry optimization with internal, cartesian,

and mixed coordinates. *International Journal of Quantum Chemistry* **1992**, *44*, 243–252.

(6) Pulay, P.; Fogarasi, G. Geometry optimization in redundant internal coordinates. *The Journal of Chemical Physics* **1992**, *96*, 2856–2860.

(7) Baker, J.; Kessi, A.; Delley, B. The generation and use of delocalized internal coordinates in geometry optimization. *The Journal of Chemical Physics* **1996**, *105*, 192–212.

(8) Peng, C.; Ayala, P. Y.; Schlegel, H. B.; Frisch, M. J. Using redundant internal coordinates to optimize equilibrium geometries and transition states. *Journal of Computational Chemistry* **1996**, *17*, 49–56.

(9) Bakken, V.; Helgaker, T. The efficient optimization of molecular geometries using redundant internal coordinates. *The Journal of Chemical Physics* **2002**, *117*, 9160–9174, Publisher: American Institute of Physics.

(10) Hermes, E. D.; Sargsyan, K.; Najm, H. N.; Zádor, J. Geometry optimization speedup through a geodesic approach to internal coordinates. *The Journal of Chemical Physics* **2021**, *155*, 094105, Publisher: American Institute of Physics.

(11) Van de Vijver, R.; Zádor, J. KinBot: Automated stationary point search on potential energy surfaces. *Computer Physics Communications* **2020**, *248*, 106947.

(12) Wilson, R. B. A simplicial algorithm for concave programming. *Ph. D. Dissertation, Graduate School of Bussiness Administration* **1963**,

(13) Nocedal, J.; Wright, S. J. In *Numerical Optimization*; Glynn, P., Robinson, S. M., Eds.; Springer Series in Operations Research; Springer-Verlag New York, Inc.: 175 Fifth Avenue, New York, NY 10010, USA, 1999.

(14) Cordero, B.; Gómez, V.; Platero-Prats, A. E.; Revés, M.; Echeverría, J.; Cremades, E.; Barragán, F.; Alvarez, S. Covalent radii revisited. *Dalton Transactions* **2008**, 2832–2838, Publisher: The Royal Society of Chemistry.

(15) Wang, L.-P.; Song, C. Geometry optimization made simple with translation and rotation coordinates. *The Journal of Chemical Physics* **2016**, *144*, 214108, Publisher: American Institute of Physics.

(16) Birkholz, A. B.; Schlegel, H. B. Exploration of some refinements to geometry optimization methods. *Theoretical Chemistry Accounts* **2016**, *135*, 84.

(17) Wilson, E.; Decius, J.; Cross, P. *Molecular Vibrations: The Theory of Infrared and Raman Vibrational Spectra*; Dover Books on Chemistry Series; Dover Publications, 1980.

(18) Moore, E. H. On the reciprocal of the general algebraic matrix. *Bull. Am. Math. Soc.* **1920**, *26*, 394–395.

(19) Penrose, R. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society* **1955**, *51*, 406–413, Publisher: Cambridge University Press.

(20) Sharada, S. M.; Bell, A. T.; Head-Gordon, M. A finite difference Davidson procedure to sidestep full ab initio hessian calculation: Application to characterization of stationary points and transition state searches. *The Journal of Chemical Physics* **2014**, *140*, 164115, Publisher: American Institute of Physics.

(21) Baker, J.; Bergeron, D. Constrained optimization in cartesian coordinates. *Journal of Computational Chemistry* **1993**, *14*, 1339–1346.

(22) Baker, J. Constrained optimization in delocalized internal coordinates. *Journal of Computational Chemistry* **1997**, *18*, 1079–1095.

(23) Olsen, J.; Jørgensen, P.; Simons, J. Passing the one-billion limit in full configuration-interaction (FCI) calculations. *Chemical Physics Letters* **1990**, *169*, 463–472.

(24) Saad, Y.; Lehoucq, R.; Sorensen, D. *Templates for the Solution of Algebraic Eigenvalue Problems*; Software, Environments and Tools; Society for Industrial and Applied Mathematics, 2000; pp 37–44.

(25) Davidson, E. R. The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices. *Journal of Computational Physics* **1975**, *17*, 87–94.

(26) Morgan, R.; Scott, D. Generalizations of Davidson's Method for Computing Eigenvalues of Sparse Symmetric Matrices. *SIAM Journal on Scientific and Statistical Computing* **1986**, *7*, 817–825.

(27) G. Sleijpen, G.; Van der Vorst, H. A Jacobi–Davidson Iteration Method for Linear Eigenvalue Problems. *SIAM Journal on Matrix Analysis and Applications* **1996**, *17*, 401–425.

(28) Sleijpen, G. L. G.; Wubs, F. W. Effective preconditioning techniques for eigenvalue problems. *Preprint 1117* **1999**,

(29) Stathopoulos, A. Nearly Optimal Preconditioned Methods for Hermitian Eigenproblems under Limited Memory. Part I: Seeking One Eigenvalue. *SIAM Journal on Scientific Computing* **2007**, *29*, 481–514.

(30) Schnabel, R. B. Quasi-Newton Methods Using Multiple Secant Equations ; CU-CS-247-83. *Computer Science Technical Reports* **1983**, *224*, 1–40.

(31) Anglada, J. M.; Bofill, J. M. How good is a Broyden–Fletcher–Goldfarb–Shanno-like update Hessian formula to locate transition structures? Specific reformulation of Broyden–Fletcher–Goldfarb–Shanno for optimizing saddle points. *Journal of Computational Chemistry* **1998**, *19*, 349–362.

(32) Bofill, J. M. Remarks on the updated Hessian matrix methods. *International Journal of Quantum Chemistry* **2003**, *94*, 324–332.

(33) Gower, R. M.; Gondzio, J. Action constrained quasi-Newton methods. *arXiv:1412.8045 [cs, math]* **2014**, arXiv: 1412.8045.

(34) Fischer, T. H.; Almlof, J. General methods for geometry and wave function optimization. *The Journal of Physical Chemistry* **1992**, *96*, 9768–9774.

(35) Ryckaert, J.-P.; Ciccotti, G.; Berendsen, H. J. C. Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes. *Journal of Computational Physics* **1977**, *23*, 327–341.

(36) Andersen, H. C. Rattle: A "velocity" version of the shake algorithm for molecular dynamics calculations. *Journal of Computational Physics* **1983**, *52*, 24–34.

(37) Vasantharajan, S.; Biegler, L. T. Large-scale decomposition for successive quadratic programming. *Computers & Chemical Engineering* **1988**, *12*, 1087–1101.

(38) Banerjee, A.; Adams, N.; Simons, J.; Shepard, R. Search for stationary points on surfaces. *The Journal of Physical Chemistry* **1985**, *89*, 52–57.

(39) Anglada, J. M.; Bofill, J. M. A reduced-restricted-quasi-Newton–Raphson method for locating and optimizing energy crossing points between two potential energy surfaces. *Journal of Computational Chemistry* **1997**, *18*, 992–1003.

(40) Besalú, E.; Bofill, J. M. On the automatic restricted-step rational-function-optimization method. *Theoretical Chemistry Accounts* **1998**, *100*, 265–274.

(41) Valiev, M.; Bylaska, E. J.; Govind, N.; Kowalski, K.; Straatsma, T. P.; Van Dam, H. J. J.; Wang, D.; Nieplocha, J.; Apra, E.; Windus, T. L.; de Jong, W. A. NWChem: A comprehensive and scalable open-source solution for large scale molecular simulations. *Computer Physics Communications* **2010**, *181*, 1477–1489.

(42) Aprà, E.; Bylaska, E. J.; de Jong, W. A.; Govind, N.; Kowalski, K.; Straatsma, T. P.; Valiev, M.; van Dam, H. J. J.; Alexeev, Y.; Anchell, J.;

Anisimov, V.; Aquino, F. W.; Atta-Fynn, R.; Autschbach, J.; Bauman, N. P.; Becca, J. C.; Bernholdt, D. E.; Bhaskaran-Nair, K.; Bogatko, S.; Borowski, P.; Boschen, J.; Brabec, J.; Bruner, A.; Cauët, E.; Chen, Y.; Chuev, G. N.; Cramer, C. J.; Daily, J.; Deegan, M. J. O.; Dunning, T. H.; Dupuis, M.; Dyall, K. G.; Fann, G. I.; Fischer, S. A.; Fonari, A.; Früchtl, H.; Gagliardi, L.; Garza, J.; Gawande, N.; Ghosh, S.; Glaesemann, K.; Götz, A. W.; Hammond, J.; Helms, V.; Hermes, E. D.; Hirao, K.; Hirata, S.; Jacquelin, M.; Jensen, L.; Johnson, B. G.; Jónsson, H.; Kendall, R. A.; Klemm, M.; Kobayashi, R.; Konkov, V.; Krishnamoorthy, S.; Krishnan, M.; Lin, Z.; Lins, R. D.; Littlefield, R. J.; Logsdail, A. J.; Lopata, K.; Ma, W.; Marenich, A. V.; Martin del Campo, J.; Mejia-Rodriguez, D.; Moore, J. E.; Mullin, J. M.; Nakajima, T.; Nascimento, D. R.; Nichols, J. A.; Nichols, P. J.; Nieplocha, J.; Otero-de-la Roza, A.; Palmer, B.; Panyala, A.; Pirojsirikul, T.; Peng, B.; Peverati, R.; Pittner, J.; Pollack, L.; Richard, R. M.; Sadayappan, P.; Schatz, G. C.; Shelton, W. A.; Silverstein, D. W.; Smith, D. M. A.; Soares, T. A.; Song, D.; Swart, M.; Taylor, H. L.; Thomas, G. S.; Tipparaju, V.; Truhlar, D. G.; Tsemekhman, K.; Van Voorhis, T.; Vázquez-Mayagoitia, Á.; Verma, P.; Villa, O.; Vishnu, A.; Vogiatzis, K. D.; Wang, D.; Weare, J. H.; Williamson, M. J.; Windus, T. L.; Woliński, K.; Wong, A. T.; Wu, Q.; Yang, C.; Yu, Q.; Zacharias, M.; Zhang, Z.; Zhao, Y.; Harrison, R. J. NWChem: Past, present, and future. *The Journal of Chemical Physics* **2020**, *152*, 184102, Publisher: American Institute of Physics.

(43) Hermes, E. Sella. 2022; `https://doi.org/10.5281/zenodo.6459455`.

(44) Ceriotti, M.; More, J.; Manolopoulos, D. E. i-PI: A Python interface for ab initio path integral molecular dynamics simulations. *Computer Physics Communications* **2014**, *185*, 1019–1026.

(45) Giannozzi, P.; Baroni, S.; Bonini, N.; Calandra, M.; Car, R.; Cavazzoni, C.; Ceresoli, D.; Chiarotti, G. L.; Cococcioni, M.; Dabo, I.; Corso, A. D.; Gironcoli, S. d.; Fabris, S.; Fratesi, G.; Gebauer, R.; Gerstmann, U.; Gougoussis, C.;

Kokalj, A.; Lazzeri, M.; Martin-Samos, L.; Marzari, N.; Mauri, F.; Mazzarello, R.; Paolini, S.; Pasquarello, A.; Paulatto, L.; Sbraccia, C.; Scandolo, S.; Sclauzero, G.; Seitsonen, A. P.; Smogunov, A.; Umari, P.; Wentzcovitch, R. M. QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials. *Journal of Physics: Condensed Matter* **2009**, *21*, 395502.

(46) Giannozzi, P.; Andreussi, O.; Brumme, T.; Bunau, O.; Nardelli, M. B.; Calandra, M.; Car, R.; Cavazzoni, C.; Ceresoli, D.; Cococcioni, M.; Colonna, N.; Carnimeo, I.; Corso, A. D.; Gironcoli, S. d.; Delugas, P.; DiStasio, R. A.; Ferretti, A.; Floris, A.; Fratesi, G.; Fugallo, G.; Gebauer, R.; Gerstmann, U.; Giustino, F.; Gorni, T.; Jia, J.; Kawamura, M.; Ko, H.-Y.; Kokalj, A.; Küçükbenli, E.; Lazzeri, M.; Marsili, M.; Marzari, N.; Mauri, F.; Nguyen, N. L.; Nguyen, H.-V.; Otero-de-la Roza, A.; Paulatto, L.; Poncé, S.; Rocca, D.; Sabatini, R.; Santra, B.; Schlipf, M.; Seitsonen, A. P.; Smogunov, A.; Timrov, I.; Thonhauser, T.; Umari, P.; Vast, N.; Wu, X.; Baroni, S. Advanced capabilities for materials modelling with Quantum ESPRESSO. *Journal of Physics: Condensed Matter* **2017**, *29*, 465901.

(47) Soler, J. M.; Artacho, E.; Gale, J. D.; García, A.; Junquera, J.; Ordejón, P.; Sánchez-Portal, D. The SIESTA method for ab initio order-N materials simulation. *Journal of Physics: Condensed Matter* **2002**, *14*, 2745–2779, Publisher: IOP Publishing.

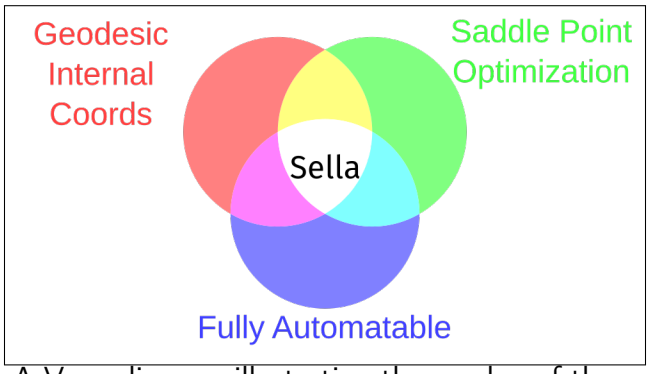
(48) García, A.; Papior, N.; Akhtar, A.; Artacho, E.; Blum, V.; Bosoni, E.; Brandimarte, P.; Brandbyge, M.; Cerdá, J. I.; Corsetti, F.; Cuadrado, R.; Dikan, V.; Ferrer, J.; Gale, J.; García-Fernández, P.; García-Suárez, V. M.; García, S.; Huhs, G.; Illera, S.; Korytár, R.; Koval, P.; Lebedeva, I.; Lin, L.; López-Tarifa, P.; Mayo, S. G.; Mohr, S.; Ordejón, P.; Postnikov, A.; Pouillon, Y.; Pruneda, M.; Robles, R.; Sánchez-Portal, D.; Soler, J. M.; Ullah, R.; Yu, V. W.-z.; Junquera, J. Siesta: Recent developments and applications. *The Journal of Chemical Physics* **2020**, *152*, 204108, Publisher: American Institute of Physics.

(49) Gonze, X.; Amadon, B.; Antonius, G.; Arnardi, F.; Baguet, L.; Beuken, J.-M.; Bieder, J.; Bottin, F.; Bouchet, J.; Bousquet, E.; Brouwer, N.; Bruneval, F.; Brunin, G.; Cavignac, T.; Charraud, J.-B.; Chen, W.; Côté, M.; Cottenier, S.; Denier, J.; Geneste, G.; Ghosez, P.; Giantomassi, M.; Gillet, Y.; Gingras, O.; Hamann, D. R.; Hautier, G.; He, X.; Helbig, N.; Holzwarth, N.; Jia, Y.; Jollet, F.; Lafargue-Dit-Hauret, W.; Lejaeghere, K.; Marques, M. A. L.; Martin, A.; Martins, C.; Miranda, H. P. C.; Naccarato, F.; Persson, K.; Petretto, G.; Planes, V.; Pouillon, Y.; Prokhorenko, S.; Ricci, F.; Rignanese, G.-M.; Romero, A. H.; Schmitt, M. M.; Torrent, M.; van Setten, M. J.; Van Troeye, B.; Verstraete, M. J.; Zérah, G.; Zwanziger, J. W. The Abinit project: Impact, environment and recent developments. *Computer Physics Communications* **2020**, *248*, 107042.

(50) Romero, A. H.; Allan, D. C.; Amadon, B.; Antonius, G.; Applencourt, T.; Baguet, L.; Bieder, J.; Bottin, F.; Bouchet, J.; Bousquet, E.; Bruneval, F.; Brunin, G.; Caliste, D.; Côté, M.; Denier, J.; Dreyer, C.; Ghosez, P.; Giantomassi, M.; Gillet, Y.; Gingras, O.; Hamann, D. R.; Hautier, G.; Jollet, F.; Jomard, G.; Martin, A.; Miranda, H. P. C.; Naccarato, F.; Petretto, G.; Pike, N. A.; Planes, V.; Prokhorenko, S.; Rangel, T.; Ricci, F.; Rignanese, G.-M.; Royo, M.; Stengel, M.; Torrent, M.; van Setten, M. J.; Van Troeye, B.; Verstraete, M. J.; Wiktor, J.; Zwanziger, J. W.; Gonze, X. ABINIT: Overview and focus on selected capabilities. *The Journal of Chemical Physics* **2020**, *152*, 124102, Publisher: American Institute of Physics.

(51) Blum, V.; Gehrke, R.; Hanke, F.; Havu, P.; Havu, V.; Ren, X.; Reuter, K.; Scheffler, M. Ab initio molecular simulations with numeric atom-centered orbitals. *Computer Physics Communications* **2009**, *180*, 2175–2196.

Graphical TOC Entry



A Venn diagram illustrating the overlap of three major design features of Sella: saddle point optimization, geodesic internal coordinates, and full automatability of the code.