DL-FIND: An Open-Source Geometry Optimizer for Atomistic Simulations[†]

Johannes Kästner,[*,‡,§] Joanne M. Carr,[‡] Thomas W. Keal,[‡,||] Walter Thiel,[||] Adrian Wander,[‡] and Paul Sherwood[‡]

Computational Science and Engineering Department, STFC Daresbury Laboratory, Daresbury, Warrington WA4 4AD, United Kingdom, and Max-Planck-Institut für Kohlenforschung, Kaiser-Wilhelm-Platz 1, D-45470 Mülheim an der Ruhr, Germany

Received: March 31, 2009; Revised Manuscript Received: June 12, 2009

Geometry optimization, including searching for transition states, accounts for most of the CPU time spent in quantum chemistry, computational surface science, and solid-state physics, and also plays an important role in simulations employing classical force fields. We have implemented a geometry optimizer, called DL-FIND, to be included in atomistic simulation codes. It can optimize structures in Cartesian coordinates, redundant internal coordinates, hybrid-delocalized internal coordinates, and also functions of more variables independent of atomic structures. The implementation of the optimization algorithms is independent of the coordinate transformation used. Steepest descent, conjugate gradient, quasi-Newton, and L-BFGS algorithms as well as damped molecular dynamics are available as minimization methods. The partitioned rational function optimization algorithm, a modified version of the dimer method and the nudged elastic band approach provide capabilities for transition-state search. Penalty function, gradient projection, and Lagrange−Newton methods are implemented for conical intersection optimizations. Various stochastic search methods, including a genetic algorithm, are available for global or local minimization and can be run as parallel algorithms. The code is released under the open-source GNU LGPL license. Some selected applications of DL-FIND are surveyed.

I. Introduction

The identification of stationary points, such as minima and transition states, on a potential energy surface (PES) is a central task in atomistic simulations and is therefore vital to fields such as computational chemistry, theoretical surface science, and solid state physics. A stationary point on the potential energy surface $E(\boldsymbol{x})$ is a point, $\boldsymbol{x}$, where the gradient of E vanishes: $\boldsymbol{g}(\boldsymbol{x}) \equiv \nabla E = \boldsymbol{0}$. Various types of stationary points are of interest in atomistic simulations, including minima, transition states, and, in quantum chemical simulations, conical intersections.

Local minima are characterized by the Hessian $\mathbf{H}(\boldsymbol{x})$, the matrix of second derivatives with respect to the atomic coordinates, having only positive eigenvalues. Hence, they correspond to chemically stable or metastable structures, as a small displacement in any direction raises the potential energy. The minimum of lowest energy on a PES is denoted the global minimum and corresponds to the most stable structure at 0 K.

Transition states represent the highest point(s) on the minimum-energy path connecting two stationary points (usually minima). They can be defined geometrically[1] as stationary points characterized by a single negative eigenvalue of the Hessian matrix. Steepest-descent paths leaving the transition state forward and backward along the corresponding eigenvector will lead to the two connected stationary points. The barrier height—the difference in energy between a minimum and a transition state—is related to the reaction rate by transition state theory.

Excited state surfaces can be characterized by the optimization of conical intersections, which occur when two electronic states I and J are degenerate. The degenerate region is normally a seam (hyperline) with $a - 2$ dimensions, where a is the number of degrees of freedom. The degeneracy is broken by moving in the remaining plane (known as a branching plane) defined by the gradient difference vector $\boldsymbol{g}_{IJ}$ and the gradient of the interstate coupling $\boldsymbol{h}_{IJ}$. These two vectors play an important role in conical intersection optimization algorithms. The energetically lowest point on a conical intersection (the minimum energy crossing point, MECP) is the point of the most likely transition between the two electronic states and is therefore somewhat analogous to the transition state in single state reactions.

A wide variety of algorithms are available to locate the stationary points described above. Historically most methods rely on a sequential optimization cycle where the gradient and sometimes an analytical or approximate Hessian are used to progressively improve a guess for the optimized geometry. More recently parallel search algorithms have been developed to exploit the resources of parallel computers and clusters. Serial algorithms can also be used in these environments by parallelizing the energy and gradient evaluation, but this approach is generally limited by the methods used to calculate the energy. Intrinsically parallel search algorithms, which use the information from many points on the potential energy surface at once, can lead to a higher efficiency in finding stationary points, in terms of the effective time-to-solution.

The efficiency with which stationary points are located often depends on the coordinate system used. Algorithms which follow the negative gradient toward a minimum generally converge faster when the degrees of freedom are less coupled. This coupling depends on the choice of the coordinate system. Often Cartesian coordinates are more coupled than internal coordinates, which describe the system via bond lengths, angles,

[†] Part of the “Walter Thiel Festschrift”.
[*] Corresponding author. E-mail address: kaestner@theochem.uni-stuttgart.de.
[‡] STFC Daresbury Laboratory.
[§] Current address: Institute for Theoretical Chemistry, University of Stuttgart, Pfaffenwaldring 55, 70569 Stuttgart, Germany.
[||] Max-Planck-Institut für Kohlenforschung.

and dihedral angles, and are therefore more aligned with the way the system can move.

In this paper, we present the algorithms and implementation of the geometry optimization methods and coordinate systems available in the DL-FIND program. In section II. we describe the general features of the DL-FIND library, including the structure of the code and the interface to the program that performs the energy and gradient evaluation (the calling program). Other general features include the choice of coordinate system (which is independent of the search algorithm), the trust region definition, and convergence criteria. In section III we describe the specific optimization tasks that can be performed under the categories of local minimization, global (parallel) minimization, transition state optimization, and conical intersection optimization. Finally, example optimizations are given in section IV.

II. General Features

II.A. Structure of the DL-FIND Code. DL-FIND is written as a library which can be linked to codes (referred to here as the calling program) that provide the potential energy and its derivatives for a given atomic structure. The energy can be obtained from electronic structure calculations, classical force fields, or any other method. In principle any objective function of multiple variables can be used, although the coordinate transformation algorithms and associated constraints are only meaningful in relation to atomic coordinates.

The internal structure of the code is modular. A main module controls the overall progress of the optimization cycles. It also calls the interface routines which pass the atomic geometry to the calling program and obtain the energy and gradient. These calculations are performed in Cartesian coordinates so that the calling program does not have to understand the coordinate systems used by DL-FIND. Another module is responsible for the coordinate transformations to and from Cartesian coordinates. This sequence of coordinate transformation and energy evaluation can loop over multiple sets of structures, which is needed for optimization methods which require more than one gradient evaluation per optimization step, like the nudged elastic band method or the dimer method.

The coordinate systems available are independent of the optimization algorithms, which operate on an unspecified set of internal coordinates. The algorithms are therefore also contained in separate modules, which are given coordinate and gradient vectors by the coordinate transformation module. The optimization module contains algorithms to follow the negative gradient to a minimum, to find transition states, or to perform a constrained minimization to a conical intersection. It obtains a coordinate vector and a gradient vector and returns a step direction and, depending on the algorithm, also a step length. It also includes the global (parallel) optimization algorithms, which necessarily have a distinct implementation and mode of operation (for example, they do not return a step vector). The next module takes care of the line search or trust radius algorithm. Finally, there are modules providing utility functions, such as bookkeeping of the overall storage allocation, timing measurement, or wrappers for calls to linear algebra libraries and parallel communication routines.

This modular arrangement allows the combination of different optimization algorithms with different coordinate systems. To our knowledge we were the first to apply the dimer method in any type of internal coordinate description, which turned out to accelerate convergence to transition states significantly. The modularity of the code also allows new optimization algorithms or coordinate systems to be implemented in a straightforward manner.

Restart information can be written to disk at specified intervals. The assumption has been made that most of the computation time is spent calculating the energy and gradient. Therefore, restart information is written after an energy evaluation.

The DL-FIND library is written in standard Fortran 95, including the technical report TR 15581, which means that allocatable arrays are used as dummy arguments and in derived types. The code is released under the open-source GNU Lesser General Public License (LGPL)[2] and can be downloaded online.[3]

II.B. Parallelism. Optimization techniques that involve more than one independent configuration at each iteration (such as the genetic algorithm, random search, and nudged elastic band) are amenable to parallelization whereby the energy evaluations for the individual structures occur simultaneously on different processors. Such a "task-farming" parallelism is implemented in DL-FIND using MPI (message passing interface[4]). Each task farm (or workgroup, i.e., subset of processors) calculates the energy and required gradients for a nonoverlapping selection of the structures at the current iteration. The load balancing is static, and the number of task farms must be a factor of the total number of processors for a given run. If the number of task farms requested is less than the total number of processors, then the single-point energy and gradient calculations for each individual can also be parallelized within a task farm, subject to the availability of this functionality in the calling program. The task-farming parallelism does not affect the operation of DL-FIND compiled as a serial library. A parallel-enabled version of the DL-FIND library is obtained at compile time using a build option, as described in the user manual supplied with the source code.

II.C. Interface to the Calling Program. DL-FIND presents a well-defined interface for integration with the calling program. The calling program first calls the main DL-FIND routine, which does not return until the optimization is complete. When DL-FIND requires information from the calling program during the optimization, an interface routine is called. These routines must be supplied by the calling program. A basic set of routines is required for serial calculations while an additional set is necessary to handle parallelization.

In the serial case, the first interface call is used to get the input parameters from the calling program. These consist of the starting structure for the optimization and the DL-FIND options that the user has requested. Following this, the optimization begins and a further call is made each time an energy or energy derivative is required. DL-FIND provides a set of Cartesian coordinates to the calling program and receives back the required values. The main interface routine here requests the energy and gradient. A second routine requests the Hessian if the optimization method requires one and the user has specified that it should be calculated by the calling program rather than calculated numerically by DL-FIND. There is also a third routine specific to conical intersection optimizations, which requests an energy and gradient for both of the electronic states involved, and if necessary the interstate coupling gradient. The latter two routines only have to be implemented by the calling program if these methods are desired. At the end of each cycle (and at the end of the optimization), another routine passes the current set of Cartesian coordinates back to the calling program.

Two further interface calls are made in special circumstances. The first is called after a reset of the optimization algorithm. This gives the calling program an opportunity to make updates that would result in a discontinuity in the potential energy surface (e.g., an update of the molecular mechanics neighbor lists). The second routine is called in the event of an error and simply returns control to the calling program.

The interface strategy concerning parallelism allows for either DL-FIND or the calling program to set up and obtain parameters for the MPI communications, as is most appropriate. In this way, the task-farming parallelism in DL-FIND may be employed with calling programs that lack parallel capability in isolation. A standard DL-FIND routine for direct or indirect initialization and setup of the global communicator must be called from the calling program in either case. This call should be located either in the usual place for MPI initialization (as close to the start of the program as possible) if DL-FIND is responsible for the parallelism or between the MPI setup in the calling program and the first call to a DL-FIND routine, otherwise. DL-FIND will then automatically determine whether it or the calling program controls the global MPI initialization. If the calling program is not parallelized, then a standard DL-FIND routine to shut down and finalize the MPI communications must also be called from an appropriate place.

Furthermore, unit numbers for input/output to disk that are set in the calling program can, if desired, be made known to DL-FIND via an interface routine called from the main program. This subroutine can also be used by DL-FIND to disentangle the output from different processors in a parallel run. Interface routines to supply information on the global and the task-farming communications (e.g., processor identity, number of processors, communicator handle) from either the calling program to DL-FIND or vice versa, as appropriate, must also be provided. A variable that indicates to DL-FIND whether it or the calling program will setup the task-farming communications should be included with the input parameters passed to DL-FIND via the first serial interface routine discussed above.

Interfaces to DL-FIND have been implemented in GAMESS-UK[5,6] and ChemShell.[7,8] An interface to CRYSTAL[9] is also under development. The interface to ChemShell in turn allows other programs to be employed with DL-FIND using the standard ChemShell drivers. These drivers have been extended to allow multiple-state calculations with MNDO,[10] Gaussian,[11] and MOLPRO,[12] which enables the conical intersection algorithms to be used with these codes. Currently, the GAMESS-UK and CRYSTAL interfaces can allow task-farming parallelism. Such parallelism in conjunction with ChemShell is intended as a future development.

II.D. Coordinate Systems. The optimization algorithms in DL-FIND operate with a set of working coordinates, $\boldsymbol{x}'$, which may be defined in a number of ways as follows.

II.D.1. Cartesian Coordinates. In the simplest case, the optimization is carried out in Cartesian coordinates without constraints. Then, no coordinate transformation is done, $\boldsymbol{x}' = \boldsymbol{x}$. Even in Cartesian coordinates, constraints can be used: atoms can be frozen (not optimized) or Cartesian components of atoms can be constrained. These are mapped out of the coordinate and gradient vectors. Additionally, optimization in mass-weighted Cartesian coordinates is implemented, where $\boldsymbol{x}' = \mathbf{m}^{1/2}\boldsymbol{x}$, with $\mathbf{m}^{1/2}$ being a diagonal matrix with the square roots of the masses of the atoms in the diagonal.

II.D.2. Internal Coordinates. Internal coordinates describe a system in terms of bond lengths, bond angles, and dihedral angles, summarized as $\boldsymbol{q}(\boldsymbol{x})$. The number of these "primitive internals" is generally larger than the number of degrees of freedom they could describe ($3N - 6$ in general, where N is the number of atoms, as global rotation and translation are not described in internals). Thus, it is a redundant set of coordinates. This cannot directly be used for optimization because of forbidden step directions. For example, the optimization algorithm could attempt to increase all three angles of a triangle. Therefore, a set of nonredundant coordinates, delocalized internal coordinates (DLC), is constructed by a linear transformation: $\boldsymbol{x}' = \mathbf{U}\cdot\boldsymbol{q}(\boldsymbol{x})$.[13−15] In contrast to other implementations,[13,14] we calculate this transformation once for one geometry, $\boldsymbol{x}_0$, and then perform the optimization in the resulting set of nonredundant internal coordinates.[16] The alternative is to use the complete redundant set in the optimization algorithm and restrict the optimization step to allowed directions by an equivalent linear transformation. If the geometry $\boldsymbol{x}$ deviates strongly from the geometry $\boldsymbol{x}_0$ where the transformation was calculated, the back-transformation from $\boldsymbol{x}'$ to Cartesian coordinates $\boldsymbol{x}$ becomes unstable. In this case, we construct a new transformation matrix $\mathbf{U}$. In such cases, the optimization algorithm has to be reset as its history is inconsistent with the new set of coordinates. This is one of the few cases where an interaction between the coordinate transformation routines and the optimization algorithm is necessary.

In some cases, the use of all distances between pairs of atoms as $\boldsymbol{q}(\boldsymbol{x})$, i.e., a total-connection scheme, is more appropriate than primitive internals. This is especially the case for structures close to a transition state where the automatic definition of bonds is difficult. In solid-state systems, the definition of bonds is often impossible. Cartesian coordinates are more appropriate for these systems.

The effort of the coordinate transformation from Cartesian to nonredundant internal coordinates scales as $\mathcal{O}(N^3)$. Thus, for large systems, it can take up a significant portion of the overall computer time used. For these cases, a linear-scaling algorithm for using internal coordinates is implemented. In so-called hybrid delocalized internal coordinates (HDLC),[16] the system is partitioned into fragments. Primitive internals are used within each fragment. To couple the fragments together, the Cartesian coordinates of each fragment are added to the redundant set of coordinates $\boldsymbol{q}$ when constructing the transformation matrix $\mathbf{U}$. This still leads to weak coupling between the nonredundant coordinates because the resulting weight of the Cartesians is low but also to an algorithm which scales linearly with system size. As for DLC, the total connection scheme can be used instead of primitive internals.

II.D.3. Constraints. Coordinate-based constraints are available to the user. They are implemented in a simple manner, whereby only constraints that are components of the coordinate system can be specified. For example, in Cartesian coordinates only Cartesian components can be fixed. In internal coordinates based on primitive internals, bond lengths, angles, and/or dihedral angles can be fixed. These constraints are handled by mapping the corresponding components out of the coordinate vector when forming the set of coordinates $\boldsymbol{x}'$ used by the optimization algorithm.

Conical intersection optimizations also involve the use of constraints. These are automatically generated and specific to each algorithm, and are discussed in section III.D.

II.E. Trust Radius Approaches. Optimization algorithms generally provide the direction $\boldsymbol{s}$ of the next optimization step. Linear algorithms, like steepest descent or conjugate gradient, do not provide a step length. In these cases, and also in some cases where step lengths are provided by the optimization algorithm, a line search or trust radius approach can be used to restrict the step length to the region where the extrapolation of the energy by using the gradient is expected to hold.

The simplest possible approach is a constant trust radius, i.e., a maximum length of the total step or maximum change in each coordinate value. The latter is implemented here.

For sequential energy minimizations, a variable trust radius based on the change of the target function, the energy, can be used.[16] The trust radius is increased by a factor of 2 whenever the energy decreases and the so-called Wolfe conditions are fulfilled. The first of these is the Armijo condition, which tests whether the energy decreases by more than $\alpha \boldsymbol{s}\cdot\boldsymbol{g}_1$, with $\boldsymbol{s}$ being the step vector and $\boldsymbol{g}_1$ being the gradient before the step is taken. The constant α can be freely chosen. A value of 10^{-4} was found to be useful. The second Wolfe condition is the curvature condition. It is fulfilled if $|\boldsymbol{s}\cdot\boldsymbol{g}_2| \leq \beta|\boldsymbol{s}\cdot\boldsymbol{g}_1|$, with $\boldsymbol{g}_2$ being the gradient after the step $\boldsymbol{s}$ is taken. The factor β is chosen to be 0.99. Whenever these conditions are fulfilled, the trust radius is scaled up by a factor of 2, subject to a maximum value which is user-definable. If the energy decreases, but the Wolfe conditions are not fulfilled, the step is accepted, but the trust radius is kept unchanged. If the energy increased, it is a sign that the last step taken was too long. In this case, the step is rejected. The new trial position is obtained by adding half the step to the previous position. The new trust radius is half the previous trust radius or half the step length, whichever is smaller.

For transition state searches, and all tasks where no target function can be defined (as in the nudged elastic band approach), a variable trust radius based on the projection of the gradient on the step vector can be used. The projection of $\boldsymbol{g}_1$ on the step vector $\boldsymbol{s}$ is calculated before ($p_1 = \boldsymbol{s}\cdot\boldsymbol{g}_1$) and after the step was performed ($p_2 = \boldsymbol{s}\cdot\boldsymbol{g}_2$). Ideally, p_2 should be zero. If p_2 is positive, then the step was too small, if p_2 is negative, then the step was too large. Whenever p_2 is positive, the point of zero projection is extrapolated from p_1 and p_2 and the trust radius is scaled up to this point. Thereby, the scaling factor is bound to a maximum of 2. Whenever p_2 is negative, the point of zero projection is interpolated and the trust radius is calculated from the resulting step length. In this case, the last step is rejected and a smaller step with the new step length is used. This leads to a good convergence behavior for conjugate gradient and steepest descent optimizations whenever no objective function is available, as is the case in the dimer method and the nudged elastic band method.

II.F. Convergence Criteria. An optimization is considered converged when all of the following five criteria are fulfilled (all values in atomic units, a.u.). (i) The maximum absolute gradient component has to be lower than $c = 0.00045$; this value can be changed by the user; (ii) the root-mean-square (rms) of the gradient must be lower than $(2/3)c$, (iii) the maximum absolute component of the step vector must be smaller than $4c$, (iv) the rms of the step vector must be less than $(8/3)c$, and (v) the last change in energy has to be smaller than 10^{-6}, a value which can be changed by the user independently of c.

III. Specific Optimization Tasks

III.A. Local Minimization. Local minimization aims at finding the minimum in the vicinity of the starting structure following the negative gradient downhill in a more-or-less direct fashion.

III.A.1. Steepest Descent. Steepest descent is the most primitive minimization method which uses a gradient: the step direction is $\boldsymbol{s} = -\boldsymbol{g}$. It converges very slowly, even for quadratic optimization problems with a full line search.

III.A.2. Conjugate Gradient. The conjugate gradient scheme following Polak−Ribière[17] is implemented. With

$$\gamma = \frac{(\boldsymbol{g}_2 - \boldsymbol{g}_1)\cdot\boldsymbol{g}_2}{\boldsymbol{g}_1\cdot\boldsymbol{g}_1} \tag{1}$$

the step direction is obtained from $\boldsymbol{s} = -\boldsymbol{g}_2 + \gamma\boldsymbol{s}_1$, with $\boldsymbol{s}_1$ being the direction of the last step. The scheme is reset every 10 steps by setting $\gamma = 0$. This improves convergence for optimization problems that are not strictly quadratic.

III.A.3. Damped Molecular Dynamics. Integrating Newton's equations of motion with friction leads to an energy minimum. Here, the Verlet algorithm[18] is used with a damping factor α. The limiting cases are $\alpha = 0$, which results in undamped dynamics, and $\alpha = 1$, which results in the steepest descent direction

$$\boldsymbol{s} = \frac{(1-\alpha)\boldsymbol{s}_1 - \boldsymbol{g}_2\dfrac{(\Delta t)^2}{\boldsymbol{m}}}{1+\alpha} \tag{2}$$

Initially, a high damping factor of 0.3 is used, which is decreased by a factor of 0.95 in each step while the energy decreases. Whenever the energy increases, $\alpha = 0.3$ is used again to stop the system from progressing uphill. Defaults are a time step Δt of 1 au and masses of 1 au for all particles, $\boldsymbol{m} = \mathbf{1}$. Realistic masses are not required, as the aim is to find a stationary point rather than simulate the correct dynamics. All these values can be changed by the user.

III.A.4. Newton−Raphson/Quasi-Newton. The traditional Newton−Raphson algorithm for minimization is implemented, which uses a quadratic model for the optimization and therefore requires an explicit calculation of the Hessian matrix. The step is given by

$$\boldsymbol{s} = -\mathbf{H}^{-1}\boldsymbol{g} \tag{3}$$

The Hessian $\mathbf{H}$ may be calculated by either one- or two-point finite difference within DL-FIND or analytically by the calling program (via an interface call). The Hessian is then inverted to calculate the step.

Quasi-Newton methods are also available in which the expensive Hessian calculation is replaced by an approximate update algorithm. The standard Broyden−Fletcher−Goldfarb−Shanno[19−22] (BFGS) algorithm should be selected to update the Hessian for energy minimizations. The initial Hessian can be specified as a unit matrix or a diagonal matrix (calculated numerically from a single displaced geometry), or alternatively a full initial Hessian can be calculated using the same methods as for the Newton−Raphson algorithm.

In principle the Newton−Raphson method may be used to locate transition states as well as energy minima. The quasi-Newton method can also be used if a Bofill[23] or Powell[24] Hessian update is selected (BFGS is unsuitable as it enforces a positive definite Hessian). However, these approaches are less robust than the specialized methods described in section III.C.

The quasi-Newton algorithm is the recommended method for minimizing small- and medium-sized systems where the Hessian matrix is relatively small. For larger systems it is not efficient to keep the full Hessian matrix in memory, and so the limited memory L-BFGS algorithm should be used instead.

III.A.5. L-BFGS. The limited-memory variant[25,26] of the BFGS update of the inverse Hessian is implemented (L-BFGS). It starts out from a unit matrix for the inverse Hessian and updates it with the BFGS scheme for the last M steps of the geometry optimization (M being user-definable). The search direction $\boldsymbol{s} = -\mathbf{H}^{-1}\boldsymbol{g}$ is thereby calculated without ever calculating or storing the full Hessian, as described elsewhere[25,26] in detail. Therefore this algorithm scales linearly in memory and CPU time requirement with the number of degrees of freedom. It provides a step direction as well as a step length. To make sure the algorithm converges to a minimum rather than a saddle point, the resulting step direction is reversed whenever the angle between $\boldsymbol{s}$ and $\boldsymbol{g}$ is <90°, i.e., $\boldsymbol{s}\cdot\boldsymbol{g} > 0$. The L-BFGS algorithm is implemented in an object-oriented way. This allows multiple independent instances to be used, e.g., for the rotation and translation in the dimer method.

III.B. Global Minimization. ***III.B.1. Genetic Algorithm.*** Genetic algorithms are biologically inspired global optimization methods in which populations of individuals (structures) evolve by genetic operations and natural selection according to a fitness function, which in this case is the potential energy.[27,28]

The optimization is seeded with a single input structure from which a pool of n_{init} individuals are created by adding random, uniformly distributed displacements on the interval $[-r,r]$ to the optimizable coordinates. The n structures of lowest energy are then taken as the initial population. The values of n, n_{init}, and r are all determined via input options. At each generation, the "mating" process, by which a number of the most unfit individuals are replaced by "offspring", proceeds via a crossover scheme with blending at the randomly selected crossover point.[29] The number of structures to be replaced is calculated from the chosen fractional replacement rate per population. For each pair of offspring required, two different parents are chosen from the viable section of the population using a cost-weighted (i.e., energy-weighted) random approach (sometimes referred to as a roulette-wheel scheme[28]). One component of the coordinate vector is then chosen as the crossover point, according to a uniform random distribution. The coordinate vector of the first offspring is constructed as follows. The components of lower index than the crossover point are taken directly from the first parent identified, and those of higher index come from the second parent. The new coordinate at the crossover point is chosen via linear interpolation from parent 1 to parent 2 with a uniformly distributed random number on the interval [0,1] for the "blending" factor. The second offspring is generated similarly but with the roles of the two parents reversed. Random mutations (coordinate perturbations) are also applied to increase the diversity of the population, within an elitist strategy[28,30] whereby the lowest-energy individual survives unmutated.[29] The fraction of the set of optimizable coordinates in the population to be mutated is an input parameter that is constant throughout the run. The population is periodically reset,[31] using the scheme by which the original population was created but with a smaller maximum displacement, after a chosen number of elapsed generations to avoid structural stagnation. Population resetting also occurs when the diversity of the current working population is insufficient for efficient exploration of the potential energy surface (i.e., fewer than three members of the breeding section of the population plus the individual of next-highest energy are structurally distinct).

The algorithm terminates on completion of a selected number of generations. Convergence to a minimum is assessed using a chosen tolerance on the largest component of the absolute gradient vector. At the end of a successful run, a requested number of the minima of lowest energy found are reported.

The parallelization strategy employed is a single-population master–slave model in which one processor performs the genetic operations and the energy evaluations for the individuals are distributed among all the available processors.[32] For optimal efficiency under task-farming parallelization (section II.B) where all the energy calculations in a particular generation are of similar duration, the two population sizes should be multiples of the number of task farms.

III.B.2. Random (Stochastic) Search. Optimization via a random search was probably first suggested by Anderson,[33] with early practical algorithms described and discussed by Brooks.[34] The optimization is seeded with a single input structure from which the first generation (cycle) of individuals (structures) is created by adding random, uniformly distributed displacements to the optimizable coordinates. At each subsequent generation, the new population is derived from the known individual of lowest energy by distributing structures in configuration space, according to the basic principle of the "creeping random" method.[34] The fixed population size is an input parameter. As the individuals within a population are independent, their energies (and gradients) may be evaluated simultaneously[34] using the task-farming parallelism described in section II.B.

In the current implementation, a new population can be created according to three different schemes, the latter two of which are probably novel:

(1) uniform sampling[35] (the default scheme). The displacements are random and uniformly distributed on the interval $[-r,r]$.

(2) force-direction-biased. The magnitude of each displacement is drawn at random with uniform distribution from the range $[0,r]$, and the sign of each is the opposite of that of the corresponding component of the gradient vector for the lowest-energy structure. Therefore, only "downhill" displacements are allowed (although the resulting displacement vector may not actually lead to a structure of lower energy if the value of r is sufficiently large that, for instance, the bottom of the basin of attraction is overshot).

(3) force-biased. The displacement in each direction is calculated as in the force-direction-biased scheme but is then scaled by a factor of the product of an input parameter and the magnitude of the corresponding component of the gradient vector for the lowest-energy structure. The input parameter should be chosen such that the resulting displacements are small for a structure that is nearly converged and that generally the full range of displacements is permitted for a configuration that is far from convergence.

The initial value of the search radius, r, is an input parameter. As the generations evolve, r is reduced by a chosen factor between cycles, in order to explore a shrinking region of configuration space and focus in on the stationary point. Termination tests are applied after every generation. Convergence is deemed to have been reached when the largest component of the absolute gradient vector falls below a chosen threshold value. If the new value of the search radius lies below a selected cutoff, then the run is terminated unsuccessfully as configuration space can no longer be explored effectively. A maximum number of cycles can also be set.

The two force-biased schemes for creating a new population are designed for local optimization, as the use of the gradient vector renders them inappropriate for global optimization when the PES under consideration supports multiple minima. For optimal efficiency under task-farming parallelization where all the energy calculations in a particular cycle are of similar duration, the population size should be a multiple of the number of task farms, as in the genetic algorithm (section III.B.1).

III.C. Transition State Optimization Methods. Various methods for transition state optimization are implemented in

DL-FIND. In this section, we only summarize the methods and highlight differences from the original versions.

III.C.1. P-RFO. The partitioned rational function optimization method[36−39] converges to first-order saddle points, i.e., transition states. It uses the Hessian matrix to find a step which points downhill in all directions except the one that corresponds to the negative Hessian eigenvalue. The step points uphill in this one direction. The algorithm itself is described elsewhere.[36−39]

When the Hessian contains some noise and is, thus, not exact, the resulting small eigenvalues can cause convergence problems in P-RFO. In our implementation, we allow the user to specify a criterion below which an eigenvalue is considered to belong to a "soft mode", which is ignored in calculating the step. This accelerates convergence for a noisy Hessian but impedes exact convergence for a good Hessian.

The explicit calculation of the Hessian at each time step is usually prohibitively expensive. For transition state searches using the P-RFO method, it is recommended to calculate the Hessian once at the start of the optimization and then update it. The update schemes by Powell[24] and Murtagh−Sargent,[40] and a combination of those two by Bofill[23] are implemented. Powell or Bofill updates are recommended. For steps that are too small, noise in the gradient can lead to significant errors in the Hessian update. In these cases, no update is performed. The more often the Hessian is updated, the more it deviates from the true Hessian. To this end, the user can specify an interval to recalculate the Hessian. The recalculation can be done either completely or partially. In the latter case, only soft modes (i.e., modes with an eigenvalue of the Hessian below some criterion) are recalculated by finite-differencing in the direction of these modes. This saves computation time if the Hessian has to be recalculated by finite difference.

III.C.2. Nudged Elastic Band Method. We implemented the nudged elastic band (NEB) method to characterize and optimize reaction paths in terms of minimum-energy paths. NEB maps a path in configuration space by means of a number (typically 10−20) of discrete images (replicas of the system). In the original version of NEB[41,42] these images are connected by springs to distribute them uniformly along the path. The improved-tangent NEB formulation[43] avoids the formation of kinks in flat areas of the potential energy surface and also avoids corners being cut off. We implemented the improved-tangent NEB with a climbing image formulation. The climbing image is free of spring forces, and the force in the tangent direction is reversed. Thus, the climbing image is maximized in the direction of the reaction path, but minimized in all other directions, and therefore converges to a transition state. In contrast to the original method, we introduce an additional image once the path is roughly converged, rather than turning one existing image into a climbing image. This reduces the distances between the climbing image and its neighboring images and improves the accuracy of the tangent and, thus, the convergence. Toward the end of the NEB optimization, individual images can be frozen if their force perpendicular to the path becomes smaller than a user-defined criterion.

The modular design of DL-FIND allows the NEB path to be optimized with any optimization algorithm implemented. The recommended one is the L-BFGS algorithm. Besides Cartesian coordinates, DLC or HDLC coordinates can be used. While the latter two tend to be less stable than Cartesian coordinates, they can help in defining the initial path. If the linear transit guess to construct the images initially is performed in Cartesian coordinates, this sometimes leads to unphysical structures and atom clashes. Often, those can be avoided by doing the interpolation in one of the two choices for internal coordinates, while optimizing the path in Cartesian coordinates.

Large systems can be handled by (a) freezing part of the system, and/or (b) defining atom-dependent weights. Weights are used in the form of a diagonal matrix **W** with entries for each coordinate component. A weight of zero causes the respective atoms to be minimized—independently of the other images. A weight of one causes normal NEB behavior. This allows restriction of the transition path search to the relevant region of a large system while the rest of the system is minimized. Our implementation allows for three different treatments of the end points of the path: they may be freely minimized, frozen, or restricted to minimization perpendicular to the path. A more detailed description of the implementation is given elsewhere.[44]

While it is rather involved to search for an exact transition state with NEB, the method is very useful for mapping out an approximate reaction path and finding a starting geometry for a transition state search with the dimer method.

III.C.3. Dimer Method for Transition State Search. The dimer method[45−47] turns a transition state search in a degrees of freedom into a minimization problem in $2a$ degrees of freedom. The dimer method permits the location of transition states without calculating the Hessian, and is thus useful for large systems. The gradients of two close-lying points (the dimer) on the PES are calculated. The dimer vector is defined as pointing from one end point of the dimer to the other end point. First, the dimer is rotated around its midpoint along the forces, i.e., against the gradients. This rotation eventually aligns the dimer vector along the eigenmode of the Hessian with the lowest eigenvalue: the softest vibrational mode. Equivalently, it minimizes the sum of the energies of the dimer end points and the curvature of the energy surface along the dimer vector. After convergence of the rotation, the dimer is translated to maximize the energy in the direction of the dimer vector and to minimize the energy in all directions perpendicular to it.

As with the other methods, the modular implementation of DL-FIND allows the use of different coordinate systems and different optimization algorithms in a straightforward manner. Generally, the L-BFGS algorithm is used for the rotation and the translation in two separate instances. The transition state search tends to converge faster in internal coordinates (DLC or HDLC) than in Cartesians. Once the transition state is found, it is possible to rotate (realign) the dimer in mass-weighted internal coordinates without further translation, which provides the vibrational mode corresponding to the transition mode, and its imaginary vibrational frequency.

As with the NEB method, atom-dependent weights can be defined. They restrict the dimer vector (and, thus, the transition mode) to atoms with nonzero weight. Atoms with a weight of zero are minimized. This was found useful for large systems.

More details of the implementation of the dimer method can be found elsewhere.[48]

III.D. Optimization Algorithms for Conical Intersections. The three conical intersection optimization algorithms in DL-FIND are essentially the same as those previously implemented in the semiempirical MNDO program.[49] This section describes implementation details specific to DL-FIND.

III.D.1. Penalty Function Method. The penalty function method is defined as

$$f(\boldsymbol{r}) = \frac{E_\mathrm{I} + E_\mathrm{J}}{2} + c_1 c_2{}^2 \ln\left[1 + \left(\frac{E_\mathrm{J} - E_\mathrm{I}}{c_2}\right)^2\right] \qquad (4)$$

where E_I is the energy of the lower state I and E_J is the energy of the upper state J. This method therefore requires only the energies and gradients of the two electronic states involved, not the interstate coupling gradient.

In principle the penalty function method can be used with any optimizer which can minimize an objective function. In DL-FIND the recommended choice for small- to medium-sized systems is the quasi-Newton method with BFGS updating and an initial diagonal Hessian. For larger systems the L-BFGS optimizer may be more appropriate, but as multiple-state calculations are usually relatively expensive, the cost of the electronic structure calculation will likely dominate up to very large systems.

The penalty function method can be used with any of the coordinate systems supported by DL-FIND. In principle the coordinate transformation can be performed either on the individual state gradients or on the gradient of the objective function as the result is mathematically equivalent. In practice the objective function gradient is built first in Cartesians and then the coordinate transformation is applied.

The default values for the two parameters in the penalty function expression are the same as in the MNDO implementation, i.e., $c_1 = 5$ (kcal mol^{-1})$^{-1}$ and $c_2 = 5$ kcal mol^{-1}.

III.D.2 Gradient Projection Method. The gradient for the gradient projection method is defined as

$$\boldsymbol{g} = c_3[c_4\boldsymbol{f}_1 + (1 - c_4)\boldsymbol{f}_2] \tag{5}$$

with

$$\boldsymbol{f}_1 = 2(E_I - E_J)\frac{\boldsymbol{g}_{IJ}}{|\boldsymbol{g}_{IJ}|} \tag{6}$$

$$\boldsymbol{f}_2 = (\mathbf{I} - \tilde{\boldsymbol{g}}_{IJ} \otimes \tilde{\boldsymbol{g}}_{IJ} - \tilde{\boldsymbol{h}}_{IJ} \otimes \tilde{\boldsymbol{h}}_{IJ})\frac{\partial E_J}{\partial \boldsymbol{q}} \tag{7}$$

where $\tilde{\boldsymbol{g}}_{IJ}$ and $\tilde{\boldsymbol{h}}_{IJ}$ represent the gradient difference vector and interstate coupling gradient vector after orthonormalization. Unlike the penalty function method, the gradient projection method requires the evaluation of the interstate coupling gradient at each step in the optimization.

A peculiarity of the gradient projection method is that the overall gradient is a nonlinear function of the state gradients and interstate coupling gradient. This is problematic because the coordinate transformation from, for example, Cartesian to DLCs is not in general an orthogonal transformation. Therefore if the overall gradient is constructed in Cartesians before being transformed, it will be different than if the transformation is applied to the individual state gradients before constructing the overall gradient. The only point at which the two approaches must agree is at the converged minimum, where the gradient should be zero under any coordinate system. In DL-FIND the overall gradient is built first in Cartesian coordinates before the coordinate transformation, as this was found in tests to give faster convergence.

The quasi-Newton optimization method is again recommended for the gradient projection algorithm, at least for small systems. The defaults for the parameters in the expression for the gradient remain the same as in the MNDO implementation, i.e., $c_3 = 1.0$ and $c_4 = 0.9$.

III.D.3. Lagrange–Newton Method. The Lagrangian function minimized by the Lagrange–Newton method is

$$L_{IJ} = \frac{E_I + E_J}{2} + \xi_1(E_I - E_J) + \xi_2 H_{IJ} \tag{8}$$

where ξ_1 and ξ_2 are Lagrange multipliers. As H_{IJ}, the interstate coupling, is not available via the multiple-state interface (or readily available in the QM codes themselves), this function is never actually built. Expanding it to second order gives

$$\begin{bmatrix} \nabla\nabla L_{IJ} & \boldsymbol{g}_{IJ} & \boldsymbol{h}_{IJ} \\ \boldsymbol{g}_{IJ}^{\dagger} & 0 & 0 \\ \boldsymbol{h}_{IJ}^{\dagger} & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{q} \\ \delta\xi_1 \\ \delta\xi_2 \end{bmatrix} = -\begin{bmatrix} \nabla L_{IJ} \\ E_I - E_J \\ 0 \end{bmatrix} \tag{9}$$

with the gradient of the Lagrangian

$$\nabla L_{IJ} = \frac{\boldsymbol{g}_I + \boldsymbol{g}_J}{2} + \xi_1\boldsymbol{g}_{IJ} + \xi_2\boldsymbol{h}_{IJ} \tag{10}$$

Unfortunately it is not possible simply to iterate eq 9 to convergence. The vectors $\boldsymbol{g}_{IJ}$ and $\boldsymbol{h}_{IJ}$ are not slowly varying near the conical intersection seam and so they must be orthogonalized to give slowly varying quantities. The orthogonalization procedure in DL-FIND is slightly different from that described in ref 49. In the original procedure, following Yarkony,[50] a rotation matrix is applied to the vectors to orthogonalize them

$$\begin{bmatrix} \frac{1}{2}\boldsymbol{g}_{IJ,\theta} \\ \boldsymbol{h}_{IJ,\theta} \end{bmatrix} = \begin{bmatrix} \cos 2\theta & \sin 2\theta \\ -\sin 2\theta & \cos 2\theta \end{bmatrix} \begin{bmatrix} \frac{1}{2}\boldsymbol{g}_{IJ} \\ \boldsymbol{h}_{IJ} \end{bmatrix} \tag{11}$$

where the value of θ for which $(1/2)\boldsymbol{g}_{IJ,\theta}$ and $\boldsymbol{h}_{IJ,\theta}$ are orthogonal is given by

$$\tan 4\theta = -\frac{\boldsymbol{h}_{IJ} \cdot \boldsymbol{g}_{IJ}}{(\boldsymbol{h}_{IJ} \cdot \boldsymbol{h}_{IJ}) - \frac{1}{4}(\boldsymbol{g}_{IJ} \cdot \boldsymbol{g}_{IJ})} \tag{12}$$

As the value of θ is not unique, the resulting vectors must be checked for transpositions and sign changes and corrected back if these have occurred.

This orthogonalization procedure is only valid on the conical intersection seam (where $E_I - E_J = 0$), when the constraints on the two vectors become equivalent and they lose their meaning as independent variables. In reality the energy gap is never exactly zero, and so a threshold t_1 is defined below which the geometry is considered to be on the seam and the orthogonalization procedure is applied. In MNDO this has a default value of $t_1 = 10^{-4}$ kcal mol^{-1}.

The question remains, however, of what to do with the residual energy difference $E_I - E_J$ on the right-hand side of eq 9. If this is ignored, the errors in the constraints caused by applying the rotation matrix will gradually build up and convergence will be prevented. In ref 49 an empirical correction was used, whereby the residual energy difference had its sign changed and was transposed/halved along with the vectors. A more justifiable approach has been developed for the DL-FIND implementation (and subsequently back-ported to MNDO). In this method the rotation matrix is applied to the right-hand side of the constraint equation as well, i.e.

$$\begin{bmatrix} \cos 2\theta & \sin 2\theta \\ -\sin 2\theta & \cos 2\theta \end{bmatrix} \begin{bmatrix} \frac{1}{2}(E_I - E_J) \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\cos 2\theta \cdot (E_I - E_J) \\ -\frac{1}{2}\sin 2\theta \cdot (E_I - E_J) \end{bmatrix} \quad (13)$$

(Note that $E_I - E_J$ is halved before the orthogonalization because $\boldsymbol{g}_{IJ}$ was also halved). Because there is more than one solution for θ, sign changes and transpositions can occur when the rotation matrix is applied. With this scheme it is natural that sign changes and transpositions on the right-hand side should be corrected as well as on the left, because the rotation matrix has been applied to both sides. Both sides of the constraint corresponding to $\boldsymbol{g}_{IJ}$ must then be doubled again before insertion into eq 9 (or alternatively the constraint could be reformulated to include the factor of 1/2 throughout).

By applying the rotation matrix to both sides of the constraint equations, there is in principle no longer any need to have a threshold for the orthogonalization procedure, because the equations now hold for any value of $E_I - E_J$. However, there is one other factor to consider. In the MNDO implementation, the Hessian of the Lagrangian $\nabla\nabla L_{IJ}$ is updated by a quasi-Newton method, e.g., BFGS. The gradient difference and interstate coupling gradient terms are only included in the update when orthogonalization is switched on, because otherwise they are not slowly varying. With the revised method, these terms can be included from the start as they are always orthogonalized. However, this does not lead to an improvement in convergence properties. Including these terms from the start forces the optimization toward a conical intersection seam too quickly, at a detriment to minimizing the overall energy. It was therefore found necessary to retain the orthogonalization thresholds t_1 and t_2, with the same default values as in MNDO, i.e., $t_1 = 10^{-4}$ kcal mol^{-1}, $t_2 = 1$ kcal mol^{-1}.

Unlike in MNDO, geometrical constraints have not been implemented in the DL-FIND version of the Lagrange–Newton method. This is because they essentially duplicate the functionality of the restraints available in ChemShell.

Unlike the other two approaches, the Lagrange–Newton method requires a purpose-built optimizer. This is similar to the Newton–Raphson optimizer, but with two extra coordinates corresponding to the constraint terms of the Lagrangian (along with the two extra gradients and two extra rows/columns in the Hessian). To avoid numerical problems associated with inversion of the Lagrange–Newton Hessian, the inversion step was replaced by diagonalization and projection of the Hessian, as in the MNDO implementation. As with the gradient projection method, the interstate coupling is required at each step.

It does not matter whether the coordinate transformation is applied before or after the Lagrange–Newton gradient is constructed as the results are mathematically equivalent (unlike the case of the gradient projection method). However, as the Hessian is constructed in internal coordinates, it is in fact necessary to apply the coordinate transformation beforehand so that the individual $\boldsymbol{g}_{IJ}$ and $\boldsymbol{h}_{IJ}$ vectors are available for insertion into the Hessian.

III.D.4. General Considerations about Conical Intersection Optimizations. The Lagrange–Newton method is the most efficient of the implemented conical intersection optimization methods[49] and is recommended as the first choice when a reliable interstate coupling gradient is available. The gradient projection method is usually less efficient but can be used for confirming the results of Lagrange–Newton. The penalty function method is only recommended when the interstate coupling gradient is unavailable.

For all three algorithms a constant trust radius is recommended with a size of 0.1 Å. For the gradient projection and Lagrange–Newton methods an energy-based trust radius is not available as no objective function is computed. For the penalty function method it is possible to use an energy-based trust radius but in tests this has been found to hinder convergence. This is because the objective function of the penalty function method is much more likely to increase in value during the normal course of an optimization than an ordinary energy value, and an energy-based trust radius restricts this unnecessarily.

It is highly recommended for conical intersection optimizations with the quasi-Newton optimizer to do a BFGS update of the Hessian at every step. This is particularly important for the Lagrange–Newton method where the rows/columns of the Hessian corresponding to the constraints must be updated in order to drive the system to convergence.

If the starting geometry is known to be close to the optimized geometry, the initial diagonal Hessian may be more accurate if a small value of the finite-difference displacement (e.g., 0.001 au) is used.

IV. Applications

IV.A. Survey of Applications Using DL-FIND. DL-FIND contains a variety of commonly available methods, such as local minimizers, and less standard approaches which increase the versatility of the program and provide extra functionality. Although DL-FIND has only been recently implemented, its special features have already found use in a number of applications, some of which are mentioned in this section.

Hybrid delocalized coordinates are particularly suitable for the study of large systems. These coordinates were first implemented in the HDLCopt optimizer[16] which has been ported to DL-FIND and has been employed successfully in many ChemShell applications to enzymatic systems. Two recent reviews[51,52] give an overview over such work, for example on extensive quantum mechanical-molecular mechanical (QM/MM) studies of the cytochrome P450cam enzyme.[51] Other recent applications of this kind include QM/MM geometry optimizations of phosphoglucore isomerase,[53] cysteine protease,[54] and flavin photoreceptors.[55]

The dimer method in DL-FIND has been used to find the transition state of the rate-limiting oxygenation reaction in the catalytic cycle of the enzyme *p*-hydroxybenzoate hydroxylase.[48] In this case, as many as 18762 degrees of freedom had to be optimized in a transition-state search. The dimer method was found to have a larger radius of convergence and required less energy and gradient calculations than the P-RFO method.

The nudged elastic band method has been employed to find transition paths in astrochemical reactions, such as the adsorption of water[44] and dihydrogen[56] on an olivine surface. It has also been an essential tool in a QM/MM study of the reductive half-reaction of xanthine oxidase,[57] where some of the steps involve a reorientation of the substrate on a relatively flat potential energy surface; in these cases, the NEB method in DL-FIND has proved helpful to locate the transition states whereas other approaches failed.

In a study on the small protonated Schiff base penta-3,5-dieniminium,[58] DL-FIND was used to find critical points on the ground-state and excited-state energy surfaces using the semiempirical OM2/GUGA-CI method from MNDO and the ab initio CASSCF and CASPT2 methods from MOLPRO. The

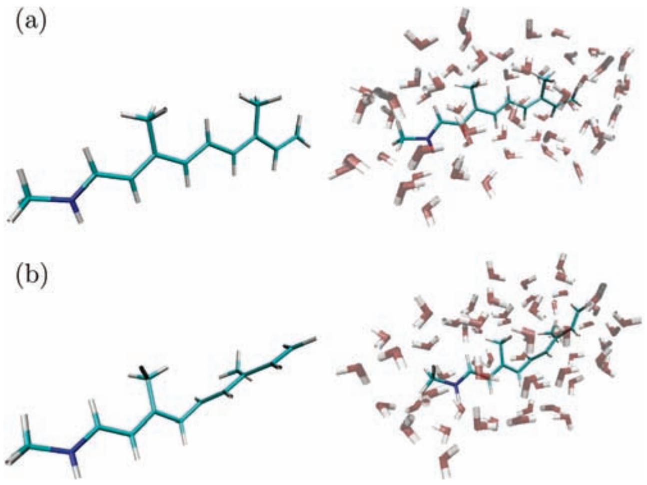

Figure 1. The gas phase and microsolvated (a) ground state and (b) $C_{11}-C_{12}$ twisted minimum energy crossing point of the all-trans retinal model used in this example.

OM2/GUGA-CI excited-state minimum energy path was calculated using the NEB algorithm, and CASSCF and CASPT2 reaction paths were determined along the torsional reaction coordinate.

The NEB method in DL-FIND has further been used to elucidate the mechanism of flavin adduct formation after excitation of the LOV1 domain of the blue-light receptor protein phototropin.[59] Minimum energy paths on the lowest triplet energy surface were calculated in order to characterize the transition state and to calculate the relative energies of the triplet minimum and singlet−triplet intersystem crossing point.

IV.B. QM/MM Conical Intersection Optimization. One particular feature of DL-FIND is conical intersection optimization. The original MNDO implementation[49] has already been successfully used in semiempirical QM investigations on 9*H*-adenine,[60] guanine,[61] and the pyrimidine nucleobases uracil, thymine, and cytosine.[62] By reimplementing the algorithms in DL-FIND, the range of possible applications has been extended to cover both other electronic structure methods (e.g., ab initio QM methods as in ref 58) and combined QM/MM approaches. This is made possible by linking DL-FIND with ChemShell, which has been extended to handle QM/MM energies and gradients for multiple electronic state calculations. ChemShell interfaces with external programs or its own routines to obtain the separate QM and MM energies and gradients and then provides the combined QM/MM energies and gradients to DL-FIND. ChemShell is responsible for adding the MM gradient to each QM state gradient in turn, and it resolves link atom forces. Note that no manipulation of the QM interstate coupling gradients is required as there is no MM contribution to them.

In the QM code separate gradient contributions from the point charges should be calculated for each electronic state and for the interstate coupling gradient. This is done when using MNDO as the QM code in QM/MM studies with ChemShell; the interstate coupling gradient is not treated in any special way, although the gradient contributions from the MM point charges are very small (and have been neglected in previous QM/MM work[63]).

The QM/MM implementation with MNDO was validated by comparison with the microsolvation study of an all-trans retinal model reported in ref 63. In this study the retinal model is optimized first in the gas phase and then again surrounded by 57 water molecules placed in favorable hydrogen bonding arrangements (Figure 1). This is not expected to be the global minimum but rather representative of a geometry that might be sampled in such a cluster. In the current study the starting coordinates were taken from the Supporting Information of ref 63.

TABLE 1: Optimized Energies in Electronvolts Relative to the S_0 Minimum for the Gas Phase Retinal Model

method	S_1 energy at S_0 min	S_0/S_1 MECP
FOMO AM1/CISDT[63]	2.22	1.31
OM2/GUGA-CISD	3.02	1.96
SA-2-CASSCF[69]	3.51	2.30

TABLE 2: Change in Energy (in Electronvolts) on Reoptimization following Addition of Solvent Molecules

method	S_1 energy at S_0 min	S_0/S_1 MECP
FOMO AM1/CISDT[63]	+0.65	+0.32
OM2/GUGA-CISD	+0.49	+0.52

The required force field parameters were specified directly in the ChemShell script (via the DL_POLY interface). The MM force field was defined following ref 63. The solvent molecules were modeled using the flexible SPC representation of water.[64] The van der Waals parameters for the retinal model were taken from the OPLS-UA force field of Weiner et al.[65] (using the closest available atom types). There are some differences in the semiempirical QM methods used here and in the previous work.[63] In the latter case, orbitals were generated using the floating occupation molecular orbital (FOMO) SCF procedure (which is not available in MNDO), and the relevant states were computed at the AM1/CISDT level with an eight orbital active space.[63] By comparison, we evaluate the performance of the OM2 semiempirical Hamiltonian[66,67] with GUGA configuration interaction (singles and doubles),[68] using ROHF molecular orbitals with an active space of four orbitals (one doubly occupied, two singly occupied, one unoccupied). In addition, we compare our gas-phase results with SA-2-CASSCF results (obtained with a 10 orbital active space).[69]

Table 1 presents energies for the vertical excitation to the first excited (S_1) state and the optimized conical intersection (MECP) relative to the ground-state (S_0) minimum. The OM2/GUGA-CISD vertical excitation and MECP energies are intermediate between the AM1/CISDT and CASSCF values. The latter are expected to bracket the true values, because CASSCF does not include dynamic correlation and AM1/CISDT overcorrects.[69] The OM2/GUGA results are thus realistic which is particularly gratifying in view of the fact that only a rather small active space was used.

Table 2 presents the change in energy on addition of the solvent molecules and reoptimization: For both methods the ground state is preferentially stabilized by the solvent molecules. This is as anticipated, since the chromophore is more polar in the ground state with the charge more localized around the N atom.

In an overall assessment, the present and the previous[69] semiempirical QM/MM study on a microsolvated retinal model thus give results that are consistent with each other (considering the differences in the QM treatment). Further such QM/MM investigations on conical intersections in a condensed-phase environment are in progress using DL-FIND, e.g., for adenine in aqueous solution.[70]

V. Conclusion

The DL-FIND package[3] provides a powerful and flexible open source library of optimization methods for use with atomistic simulation codes. In this article the wide range of optimization algorithms supported by the code have been outlined, including standard local minimization routines and parallel minimization, as well as methods for the optimization of transition states and conical intersections. Each of these methods may be used with either Cartesian, delocalized, or total connection coordinates. Large systems can be efficiently treated using hybrid delocalized coordinates and the low-memory L-BFGS optimizer. All methods are fully restartable.

In recent studies DL-FIND has been shown to be a versatile optimizer for transition state problems (using the dimer and nudged elastic band methods) and quantum mechanical conical intersection optimizations. In a further application presented here, DL-FIND was used in conjunction with ChemShell to optimize the ground state and the conical intersection of a retinal model in a microsolvated environment using a QM/MM approach.

In addition to ChemShell, interfaces have been developed to GAMESS-UK and CRYSTAL. For ease of integration, a well-defined interface is presented to external programs, which should facilitate its inclusion into other codes. The modular design of DL-FIND should also make it straightforward to add further optimization algorithms in the future.

Acknowledgment. T.W.K. was supported by the Deutsche Forschungsgemeinschaft (SFB 663). J.M.C. is grateful to the EPSRC for financial support.

References and Notes

(1) Murrell, J. N.; Laidler, K. J. *J. Chem. Soc., Faraday Trans.* **1968**, *64*, 371.
(2) http://www.gnu.org/licenses/lgpl.html.
(3) http://ccpforge.cse.rl.ac.uk/projects/dl-find.
(4) http://www.mpi-forum.org.
(5) Guest, M. F.; Bush, I. J.; van Dam, H. J. J.; Sherwood, P.; Thomas, J. M. H.; van Lenthe, J. H.; Havenith, R. W. A.; Kendrick, J. *Mol. Phys.* **2005**, *103*, 719.
(6) GAMESS-UK is a package of ab initio programs. See: http://www.cfs.dl.ac.uk/gamess-uk/index.shtml and http://ccpforge.cse.rl.ac.uk/projects/gamess-uk/.
(7) ChemShell, a Computational Chemistry Shell, see www.chemshell.org.
(8) Sherwood, P.; de Vries, A. H.; Guest, M. F.; Schreckenbach, G.; Catlow, C. R. A.; French, S. A.; Sokol, A. A.; Bromley, S. T.; Thiel, W.; Turner, A. J.; Billeter, S.; Terstegen, F.; Thiel, S.; Kendrick, J.; Rogers, S. C.; Casci, J.; Watson, M.; King, F.; Karlsen, E.; Sjøvoll, M.; Fahmi, A.; Schäfer, A.; Lennartz, C. *J. Mol. Struct. (THEOCHEM)* **2003**, *632*, 1.
(9) Dovesi, R.; Saunders, V. R.; Roetti, C.; Orlando, R.; Zicovich-Wilson, C. M.; Pascale, F.; Civalleri, B.; Doll, K.; Harrison, N. M.; Bush, I. J.; D'Arco, Ph.; Llunell, M.; Crystal06 user's manual; http://www.crystal.unito.it/, 2006.
(10) Thiel, W.; MNDO program, version 6.1, Mülheim, 2008.
(11) Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Scuseria, G. E.; Robb, M. A.; Cheeseman, J. R.; Montgomery, J. A., Jr.; Vreven, T.; Kudin, K. N.; Burant, J. C.; Millam, J. M.; Iyengar, S. S.; Tomasi, J.; Barone, V.; Mennucci, B.; Cossi, M.; Scalmani, G.; Rega, N.; Petersson, G. A.; Nakatsuji, H.; Hada, M.; Ehara, M.; Toyota, K.; Fukuda, R.; Hasegawa, J.; Ishida, M.; Nakajima, T.; Honda, Y.; Kitao, O.; Nakai, H.; Klene, M.; Li, X.; Knox, J. E.; Hratchian, H. P.; Cross, J. B.; Bakken, V.; Adamo, C.; Jaramillo, J.; Gomperts, R.; Stratmann, R. E.; Yazyev, O.; Austin, A. J.; Cammi, R.; Pomelli, C.; Ochterski, J. W.; Ayala, P. Y.; Morokuma, K.; Voth, G. A.; Salvador, P.; Dannenberg, J. J.; Zakrzewski, V. G.; Dapprich, S.; Daniels, A. D.; Strain, M. C.; Farkas, O.; Malick, D. K.; Rabuck, A. D.; Raghavachari, K.; Foresman, J. B.; Ortiz, J. V.; Cui, Q.; Baboul, A. G.; Clifford, S.; Cioslowski, J.; Stefanov, B. B.; Liu, G.; Liashenko, A.; Piskorz, P.; Komaromi, I.; Martin, R. L.; Fox, D. J.; Keith, T.; Al-Laham, M. A.; Peng, C. Y.; Nanayakkara, A.; Challacombe, M.; Gill, P. M. W.; Johnson, B.; Chen, W.; Wong, M. W.; Gonzalez, C.; Pople, J. A.; *Gaussian 03, Revision C.02*; Gaussian, Inc.: Wallingford, CT, 2004.
(12) Werner, H.-J.; Knowles, P. J.; Lindh, R.; Manby, F. R.; Schütz, M.; et al. MOLPRO, version 2006.1, a package of ab initio programs; see http://www.molpro.net.
(13) Peng, C.; Ayala, P. Y.; Schlegel, H. B.; Frisch, M. J. *J. Comput. Chem.* **1996**, *17*, 49.
(14) Pulay, P.; Fogarasi, G. *J. Chem. Phys.* **1992**, *96*, 2856.
(15) Baker, J.; Kessi, A.; Delley, B. *J. Chem. Phys.* **1996**, *105*, 192.
(16) Billeter, S. R.; Turner, A. J.; Thiel, W. *Phys. Chem. Chem. Phys.* **2000**, *2*, 2177.
(17) Polak, E.; Ribière, G. *Ref. Fra. Inf. Rech. Op.* **1969**, *3*, 35.
(18) Verlet, L. *Phys. Rev.* **1967**, *159*, 98.
(19) Broyden, C. G. *IMA J. Appl. Math.* **1970**, *6*, 76.
(20) Fletcher, R. *Comput. J.* **1970**, *13*, 317.
(21) Goldfarb, D. *Math. Comp.* **1970**, *24*, 23.
(22) Shanno, D. F. *Math. Comp.* **1970**, *24*, 647.
(23) Bofill, J. M. *J. Comput. Chem.* **1994**, *15*, 1.
(24) Powell, M. J. D. *Math. Prog.* **1971**, *26*, 1.
(25) Liu, D. C.; Nocedal, J. *Math. Program.* **1989**, *45*, 503.
(26) Nocedal, J. *Math. Comp.* **1980**, *35*, 773.
(27) Holland, J. H. *Adaptation in Natural and Artificial Systems*; University of Michigan Press: Ann Arbor, MI, 1975.
(28) Goldberg, D. *Genetic Algorithms in Search, Optimization, and Machine Learning*; Addison-Wesley: New York, 1989.
(29) Haupt, R. L.; Haupt, S. E. *Practical genetic algorithms*; Wiley: New York, 1998.
(30) De Jong, K. A. An analysis of the behavior of a class of genetic adaptive systems; PhD thesis, University of Michigan, 1975.
(31) Goldberg, D. E. Sizing populations for serial and parallel genetic algorithms. In *Proceedings of the third international conference on Genetic algorithms*; Morgan Kaufmann Publishers, Inc.: San Mateo, CA, 1989; p 70.
(32) Grefenstette, J. J. *Parallel adaptive algorithms for function optimization*; Tech. rep. no. cs-81-19; Vanderbilt University, Computer Science Department, Nashville, TN, 1981.
(33) Anderson, R. L. *J. Am. Stat. Assoc.* **1953**, *48*, 789.
(34) Brooks, S. H. *Operations Res.* **1957**, *6*, 244.
(35) Luus, R.; Jaakola, T. H. I. *AIChE J.* **1973**, *19*, 760.
(36) Cerjan, C. J.; Miller, W. H. *J. Chem. Phys.* **1981**, *75*, 2800.
(37) Simons, J.; Jørgensen, P.; Taylor, H.; Ozment, J. *J. Phys. Chem.* **1983**, *87*, 2745.
(38) Banerjee, A.; Adams, N.; Simons, J.; Shepard, R. *J. Phys. Chem.* **1985**, *89*, 52.
(39) Baker, J. *J. Comput. Chem.* **1986**, *7*, 385.
(40) Murtagh, B. A.; Sargent, R. W. H. *Comput. J.* **1970**, *13*, 185.
(41) Jónsson, H.; Mills, G.; Jacobsen, K. W. Nudged Elastic Band Method for Finding Minimum Energy Paths of Transitions. In *Classical and Quantum Dynamics in Condensed Phase Simulations*; World Scientific: Singapore, 1998; Chapter 16, p 385.
(42) Henkelman, G.; Uberuaga, B. P.; Jónsson, H. *J. Chem. Phys.* **2000**, *113*, 9901.
(43) Henkelman, G.; Jónsson, H. *J. Chem. Phys.* **2000**, *113*, 9978.
(44) Goumans, T. P. M.; Catlow, C. R. A.; Brown, W. A.; Kästner, J.; Sherwood, P. *Phys. Chem. Chem. Phys.* **2009**, *11*, 5431.
(45) Henkelman, G.; Jónsson, H. *J. Chem. Phys.* **1999**, *111*, 7010.
(46) Olsen, R. A.; Kroes, G. J.; Henkelman, G.; Arnaldsson, A.; Jónsson, H. *J. Chem. Phys.* **2004**, *121*, 9776.
(47) Heyden, A.; Bell, A. T.; Keil, F. J. *J. Chem. Phys.* **2005**, *123*, 224101.
(48) Kästner, J.; Sherwood, P. *J. Chem. Phys.* **2008**, *128*, 014106.
(49) Keal, T. W.; Koslowski, A.; Thiel, W. *Theor. Chem. Acc.* **2007**, *118*, 837.
(50) Yarkony, D. R. *J. Phys. Chem. A* **2004**, *108*, 3200.
(51) Senn, H. M.; Thiel, W. *Angew. Chem.* **2009**, *48*, 1198.
(52) Senn, H. M.; Thiel, W. *Top. Curr. Chem.* **2007**, *268*, 173.
(53) Wu, R. B.; Xie, H. J.; Cao, Z. X.; Mo, Y. R. *J. Am. Chem. Soc.* **2008**, *130*, 7022.
(54) Mladenovic, M.; Fink, R. F.; Thiel, W.; Schirmeister, T.; Engels, B. *J. Am. Chem. Soc.* **2008**, *130*, 8696.
(55) Sadeghian, K.; Bocola, M.; Schütz, M. *J. Am. Chem. Soc.* **2008**, *130*, 12501.
(56) Goumans, T. P. M.; Catlow, C. R. A.; Brown, W. A. *Mon. Not. R. Astron. Soc.* **2009**, *393*, 1403.
(57) Metz, S.; Thiel, W. To be published.
(58) Keal, T. W.; Wanko, M.; Thiel, W. *Theor. Chem. Acc.* **2009**, *123*, 145.
(59) Silva-Junior., M. R.; Thiel, W. To be published.
(60) Fabiano, E.; Thiel, W. *J. Phys. Chem. A* **2008**, *112*, 6859.
(61) Lan, Z.; Fabiano, E.; Thiel, W. *ChemPhysChem* **2009**, *10*, 1225.
(62) Lan, Z.; Fabiano, E.; Thiel, W. *J. Phys. Chem. B* **2009**, *113*, 3548.
(63) Toniolo, A.; Granucci, G.; Martínez, T. J. *J. Phys. Chem. A* **2003**, *107*, 3822.
(64) Dang, L. X.; Pettitt, B. M. *J. Phys. Chem.* **1987**, *91*, 3349.
(65) Weiner, S. J.; Kollman, P. A.; Case, D. A.; Singh, U. C.; Ghio, C.; Alagona, G.; Profeta, S., Jr.; Weiner, P. *J. Am. Chem. Soc.* **1984**, *106*, 765.
(66) Weber, W. PhD thesis, Universität Zürich, 1996.
(67) Weber, W.; Thiel, W. *Theor. Chem. Acc.* **2000**, *103*, 495.
(68) Koslowski, A.; Beck, M. E.; Thiel, W. *J. Comput. Chem.* **2003**, *24*, 714.
(69) Toniolo, A.; Ben-Nun, M.; Martínez, T. J. *J. Phys. Chem. A* **2002**, *106*, 4679.
(70) Lan, Z.; Fabiano, E.; Thiel, W. To be published.

JP9028968