NAME

surgen.in, − General input file for **surfgen**.

SYNOPSIS

This input file is used by coupled potential surface fitting program **surfgen** and its potential evaluation library. For a detailed description of the program and the evaluation library, see surfgen(1), potlib(1)

Place **surfgen.in** in the same directory where **surfgen** is being called.

DESCRIPTION

surgen.in is a Fortran 90 namelist input file. This file contains input parameters that controls the behavior of **surfgen**. This includes the **GENERAL** namelist input that specify the job type and, along with other input files, defines the molecular system, the electronic states, their symmetry, the internal coordinate basis and the expansion of Hd. Most job types will have another namelist that defines parameters that are specific to a job, except for *jobtype=-1*.

The evaluator library also use the general namelist to define the expansion. It also use a POTLIB namelist to define behaviors such as geometry logging and error estimation.

Developed by Yarkony group, Johns Hopkins University 2010-2013.

NAMELIST INPUT

GENERAL

General input that controls the expansion ansatz of the quasi-diabatic Hamiltonian (Hd), and the job to be performed by **surfgen**. Input files *connect.in*, *coord.in*, and *irrep.in* also contain detailed parameters about the definition of the ansatz.

List of Input Parameters

jobtype **INTEGER [0]** Specifies the type of job to be performed. Currently the following Types are implemented:

- -1 Use the molecule and connectivity definitions to generate all feasible symmetry permutations and print them to the standard output.
- 0 Do nothing during execution. During initialization the program will construct the expansion and read the coefficients from Hd storage file. Used by potential evaluation libraries only. Requires namelist: *POTLIB*
- 1 Construct fit Hd from ab initio data. Required namelist: *MAKESURF*
- 2 Find minima or conical intersections on the ab initio surface corresponding to a fit Hd. Requires namelist: *MINMEX*

natoms **INTEGER [0]** Number of atoms in the molecular system. It is usually unwise to use the default value of this parameter.

order **INTEGER [2]** Maximum order of the Hd expansion. Order here means the number of basis functions that are multiplied together, should they be same type of coordinate or not, in the form of coordinates defined in *coord.in.*

CpOrder **INTEGER []** Maximum order for the off-diagonal coupling blocks. When this option is not specified, lower than 0, or larger than **order**, the value of **order** is used as default.

atmgrp **INTEGER,dimension(natoms)** Array that specify the equivalency of the atoms. Must contain *natoms* elements, each of which specify the group index of an atom. Atoms in the same group are considered equivalent by the program and the permutations among them will be generated as symmetry operations.

nGrp **INTEGER** Number of electronic state groups. Each group contains a set of electronic states that carry an irreducible representation. For example, for OH radical the lowest 3 states are contained in 2 groups, an E group with 2 states and A group with 1 state.

nSymLineUps **INTEGER [1]** Total number of symmetry lineups. The program will generate basis matrices for all the symmetry line-ups, then combine them to yield the basis. Any basis that result from multiple line-ups will be merged to remove any possible redundancy. This detection procedure only examines the expansion coefficients of the basis matrices and is based on the assumption that each coordinate is an independent degree of freedom.

groupsym **INTEGER,dimension(ngrp,nSymLineUps)** Index of permutational irreducible representations carried by each of the state groups. For each irrep, representation matrices for every permutation must be supplied in input file irrep.in Irreps are indexed as they appear in `irrep.in` If *nSymLineUps* is specified, then each column should contain a complete set of state symmetries that defines one symmetry line-up. Note that even if two symmetry line-ups have different symmetry definitions, some block still may possess the same symmetry so adding symmetry

may not generate more basis for all the blocks. Using non-pure symmetry also allows the Hamiltonian to experience constant rotations, which is in most cases removed by symmetry near a symmetric geometry. Use *enfDiab* if you find it a problem.

groupprty **INTEGER,dimension(ngrp,nSymLineUps)** Inversion symmetry (parity) of each of the state group for each of the symmetry line-ups. Use -1 for anti-symmetric and 1 for symmetric.

printlvl **INTEGER [1]** Controls the level of infomation printed to standard output. 0 is lowest and 5 is highest level.

inputfl **CHARACTER(255) ['hd.data']** Name of Hd expansion coefficient input file. Hd will be initialized using these coefficients. When empty or file not exist, uncoupled surface with constant (but not identical) energies will be used to initialize Hd.

MAKESURF

This namelist contains parameters that control the surface fitting procedure. These are only used for *jobtype=1*.

Input File Specifications These options tell the program where to look for *ab initio* data. The data should be divided into groups of data points, and the information of each group is stored in a separate directory. The program will search for data in each of these directories to determine which piece of data is available, and append all available data to fitting set. This is a new functionality in 2.1 and is intended to allow the program to use results from calculations where not all data are obtained. For example, calculation of only energy, a fewer number of states, or where states are well separated and couplings are not calculated. It is also intended to make data storage more managable and transparent by grouping different data in separate directories.

The name of input files can be specified for each directory. The filenames of gradients and coupling data are expected to contain one and two wildcard characters '$' which acts as a placeholder for the indices of states.

SearchPath **CHARACTER(255),dimension(100) ['.',99∗'']** Paths of directories where the program should look for sets of input file. Each path should contain all the input information of a group of points. Including geometries, energies, and optionally energy gradients and couplings. '.' has to be included for the program to search the current path.

noteptn **CHARACTER(255),dimension(100) [100∗'note']** Specifies the filename of the optional file where a brief note is kept to explain where the geometries in the current path are and what data are available. The first line of this file will also be printed to standard output when the program reads this directory. No wildcard allowed. Optional.

gmfptn **CHARACTER(255),dimension(100) [100∗'geom.all']** Name of geometry input file. No wildcard characters allowed. Required.

enfptn **CHARACTER(255),dimension(100) [100∗'energy.all']** Name of energy input file. No wildcard characters allowed. Required. If the file does not contain all the state, please add in the first line of the file
STATES *st1 st2*
where *st1* and *st2* are the lowest and highest state of which the energy is included in this file. For example, if the directory contains data from hessian calculation on state 3, then the line should be
STATES 3 3

grdfptn **CHARACTER(255),dimension(100) [100∗'cartgrd.drt1.state$.all']** Pattern for energy gradient input file. Has 1 wildcard character which holds the slot for the index of the state of which the gradients are calculated. Optional. The program will search all the states that has an energy data in energy input file.

cpfptn **CHARACTER(255),dimension(100) [100∗'cartgrd.drt1.state$.drt1.state$.all']** Pattern for derivative coupling input file. Has 2 wildcard characters which holds the slot for the indices of the pair of states between which the couplings are calculated. Optional. The program will search all the pairs of states that both has an energy data in energy input file.

Data Selection and Weighing

npoints **INTEGER [0]** Number of points to be fit. Note that if the program cannot find the specified number of points, the variable will be adjusted to the actual number of data points read from files. If there are more data than specified, the program will only use the first *npoints* data points.

eshift **DOUBLE PRECISION [.0]** A uniform shift applied to all *ab initio* energies.

gcutoff **DOUBLE PRECISION [1D-14]** The threshold below which gradients will be considered vanished and treated as exactly 0.

usefij **LOGICAL [.true.]** Specifies if the derivative couplings instead of derivative coupling times energy differences will be used as coupling input. Derivative couplings approach infinity at intersections while coupling times energy difference remain well behaved everywhere.

w_energy **DOUBLE PRECISION [1.]** Weight factor for energy equations. This factor is multiplied with point weights and high energy scaling weights to yield the final weight of equations.

w_grad **DOUBLE PRECISION [1.]** Weight factor for energy gradient equations. This factor is multiplied with point weights and high energy scaling weights to yield the final weight of equations.

w_fij **DOUBLE PRECISION [1.]** Weight factor for coupling equations. This factor is multiplied with point weights and high energy scaling weights to yield the final weight of equations.

energyT **DOUBLE PRECISION,dimension(10) [1D30]**

highEScale **DOUBLE PRECISION,dimension(10) [1.]** *energyT* specifies a series of thresholds for the downscaling of equations when the ab initio energy of an electronic state is very high. When *E>energyT(i)*, weight **highEScale(i)** is applied to the energy, gradient and derivative coupling equations that involve that state. For couplings, the higher state is used to determine the weight. The highest possible energy bracket (with lowest weight) will be used.

ediffcutoff **DOUBLE PRECISION [20.]**

nrmediff **DOUBLE PRECISION [2D4]** The weight for derivative coupling equations is weighed down by factor **nrmediff/(ΔE+ediffcutoff)**. This weighing procedure is due to the fact that coupling times energy difference is being fit instead of the coupling itself, which is singular near intersections. Increasing the weight according to energy difference ensures that residue couplings are properly minimized, and the cutoff term prevents problematic singular behavior. This prevents the mathematical complexity of directly taking derivatives of the couplings with respect to fitting coefficients, which will give rise to term that correspond to change in energy difference.

ediffcutoff2 **DOUBLE PRECISION [1.]**

nrmediff2 **DOUBLE PRECISION [100.]** Similar to the above case, the energy equations are weighed up by factor **nrmediff2/(ΔE+ediffcutoff2)** if this value is greater than 1. This is to ensure that energy differences are properly reproduced for points that are close to degeneracy.

Fitting Algorithm and Acceleration

maxiter **INTEGER [3]** Maximum number of iterations for the fitting algorithm.

toler **DOUBLE PRECISION [1D-3]** Convergence tolerance for change in expansion coefficient.

maxd **DOUBLE PRECISION [1D0]** Maximum allowed change in Hd expansion coefficients between iterations.

dfstart **INTEGER [0]** Iteration at which differential convergence will be started. The normal equations will be constructed for the *change* of coefficients instead of expansion coefficients themselves. This will usually result in better fit and allows dumping while lifting the flattening term to very small value. However, this convergence mode has more tendency to experience oscillations and should not be enabled if the fit is qualitatively incorrect.
It is recommended that when differential convergence is enabled, set *DijScale=1*

exactTol **DOUBLE PRECISION [1D-12]** Eigenvalue cutoff when solving constrained normal equations. This parameter dictates how accurate the exact equations will be reproduced.

LSETol **DOUBLE PRECISION [1D-7]** Diagonal shift on the normal equations when solving linear equations. Larger value leads to more stable but usually slower convergence.

flattening **DOUBLE PRECISION [1D-8]** Flattening term that will be included in the objective function. In differential convergence mode, this option will remove contributions that have very small contributions to the quality of fit. As opposed to `LSETol`, which only changes the convergence procedure but does not affect the converged results, `flattening` changes the Lagrangian and thus will result in a different converged Hd.

ndiis **INTEGER [10]** Maximum dimensionality of DIIS interpolation space

ndstart **INTEGER [10]** The number of iterations to start DIIS interpolation.

linSteps **INTEGER [0]** Number of linear steps to perform. When greater than 0, the program will break the predicted change into **linSteps** smaller steps and try to find the step length that yields the smallest gradient for the Lagrangian. Step sizes are automatically shrinked when the norm of the gradient increases.

linNegSteps **INTEGER [0]** Number of linear steps to be taken to the opposite direction of the predicted change but with the same size. This should only be used when the normal equations fail to give the correct direction of changes and the linear steps towards the positive direction encounter an immediate increase in the norm of Lagrangian.

DijScale **DOUBLE PRECISION [1.]** This option controls the multiplier of the derivative of eigenvectors with respect to the fitting coefficients during construction of the normal equations. When set to 0, the dependency of eigenvectors on fitting coefficients are ignored. When set to 1, the first order response of eigenvectors with respect to the change in fitting coefficients is fully implemented. It is recommended to have *DijScale=1.0* in most cases. It only needs to be turned down when eigenvectors are changing too rapidly and gives oscillations.

scaleEx **DOUBLE PRECISION [1.]** Uniformly scale all exact equations. Since there is no weight for exact equations, this is done through scaling the gradient of the Lagrangian with respect to Lagrange multipliers. This option normally does not need to be changed. Only use it when convergence problems occur.

Eigenvector Ordering and Phasing

enfDiab **INTEGER [0]** Specify a point where diabatic and adiabatic representation will be forced to coincide. Every iteration the program will force the eigenvector of this point to be unit vectors. The off-diagonal element will be fit to 0 and the derivative of eigenvectors at this point (Dij) will also be 0 under all conditions.
The adiabatic-diabatic transformation is subject to a globally constant transformation. Since such transformation does not affect the Hamiltonian in any manner, it cannot be determined

from the fitting procedure itself. When states have different symmetry, such degree of freedom can be removed through the use of correct symmetry. When some states carry the same symmetry, this option is used to eliminate the extra degree of freedom.

gorder **DOUBLE PRECISION [1D-3]** Threshold for energy difference below which the states will by ordered by gradients instead of absolute energy. This option is ignored when *followPrev=.true.*

ckl_input **CHARACTER(255) ['']** Input file that contains the initial guess of eigenvectors at selected data points. Each line of the file contains the index of a point and the eigenvector of that point. Repeating input will be overiden but the one that is last encountered in the file. Any points that are not specified in this file will use the diagonalization of initial Hd to generate initial eigenvectors. When left empty or file not exist, all the eigenvectors are initialized by diagonalizing initial Hd.

ckl_output **CHARACTER(255) ['ckl.out']** Output file that contains the final eigenvectors at each data point.

guide **CHARACTER(255) ['']** Input file that contains reference wavefunctions at a certain set of points that will be used to determine the ordering of states. This serves as a weaker guiding tool than ‘ckl_input‘ because the wave functions and their signs are still generated by diagonalization and matching of couplings. As a result, approximate vectors such as unit vectors can be used as ‘guide‘.

followPrev **LOGICAL [.false.]** Whether the new eigenvectors will be ordered and phased to match the vectors from the previous iteration. This allows a more consistent and smoother convergence but may increase the tendency to match the states in a non-optimal way.

maxRot **DOUBLE PRECISION [.0]** When set greater than 0, the eigenvectors rotation between rotations are monitored and when the rotation is larger than this parameter the rotation is dumped to this value. **HAVE NOT BEEN TESTED FOR MORE THAN 2 STATES. DO NOT USE IT IF YOU HAVE >2 STATES!**

Local Coordinate Construction

useIntGrad **LOGICAL [.true.]** Specifies wether the gradients and derivative couplings will be fit using Cartesian components or a transformed coordinate constructed at each point that removes the null equations (translations, rotations, relative motion of dissociated fragments and symmetry zeros). This coordinate is constructed by obtaining the eigenvectors of matrix BˆT.B, where B is the Wilson's B matrix.

intGradT **DOUBLE PRECISION [1D-3]** Threshold for eigenvalue cut off of BˆT.B matrix. When an eigenvalue is lower than *intGradT*, the coordinate is considered non-internal and removed from the fitting equaitons.

intGradS **DOUBLE PRECISION [1D-1]** Threshold for diminished weights. New coordinates that correspond to eigenvalues lower than *intGradS* will be weighed by factor **ev/intGradS**, where **ev** is the eigenvalue.

gScaleMode **INTEGER [2]** Controls how the gradients and couplings will be weighed according to *intGradS*. Available scaling methods are :

=0 Do not scale

>0 Scale all coordinates

<0 Scale couplings only

deg_cap **DOUBLE PRECISION [1D-5]** Threshold for energy difference below which the states will be considered degenerate. Intersection adapted coordinate will be used for these electronic states. *Degeneracy for more than 2 states is coded but never tested.*

Removal of Null Space

TBas **DOUBLE PRECISION [1D-6]** Theshold for eigenvalue cutoff of the primitive basis overlap matrix. This controls the degree of linear dependency that will be allowed in the basis constructed for the fit.

ecutoff **DOUBLE PRECISION [1.]** Energy threshold in *hartree* above which the energy data will not be considered in null space removal procedure. This is used to prevent the equations that are irrelevant from introducing extra degrees of freedom.

egcutoff **DOUBLE PRECISION [0.6]** The gradients and couplings data of a point will not be considered in null space removal procedure when the ab initio energy of the lowest state is higher than this value. Similar to **ecutoff**, this parameter is used to prevent irrelevant high energy data points from introducing unnecessary degrees of freedom.

Input and Output

restartdir **CHARACTER(255) ['']** When not empty, the program will store eigenvectors and coefficients of Hd in this directory.

outputfl **CHARACTER(255) ['']** Name of the output file that will store the fit surface.

flheader **CHARACTER(255) ['----']** Header that will be printed into the description field of Hd storage file.

rmsexcl **INTEGER [0]** This parameter controls if low weight points will be included in the RMS analysis. Points with weight lower than **-1/rmsexcl** will be excluded when *rmsexcl<0*. No effect when *rmsexcl>=0*

printErr **LOGICAL [.false.]** This option specifies if geometry and error information will be generated for error analysis procedure in 'libsurfgen'. Geometries will be stored in file 'refgeom' and error info will be stored in 'error.log'.

POTLIB

Parameters that control the behavior of the potential evaluation library.

Parsing evaluations The evaluation subroutines has the capability of performing analysis of the geometries of the point of evaluation, including the minimum distance to a set of reference points where ab initio data are available, an estimation of eror of energies using the fitting error of gradients at the nearest reference point. These informatio are stored along with the time of evaluation, the geometry, energies of each electronic states and the surface the trajectory is currently moving on.

These information will be stored in files 'trajdata$x.csv'. Each file contains information of one complete trajectory. The data are stored in comma separated values(CSV) format. **General parsing options**

parsing **LOGICAL [.false.]** This option determines if evaluation geometries will be recorded and analyzed.

timeeval **LOGICAL [.false.]** This option determines if the program will try to time each of the individual task performed during evaluation and print out the time cost.

Distance evaluations To help determine if trajectories are exporing areas that are not populated by existing data points, the evaluation program has the capability to evaluate the minimum distance to all data points, referred to as reference points. The distances are calculated in a subset of internal coordinates natively defined by 'coord.in'. With each reference point, the program will perform all the symmetry operations to find the minimum distance among symmetry related points. Although this implies a significantly large amount of distance calculations for every evaluation, the time required for such analysis is in fact negligible in most situations as a result of the method used by the program to identify the potential low distance points. The program maintains the lower and upper bound of distances at each point of evaluation and updates these lists using triangular inequalities using the size of displacement when a new evaluation is performed. In most cases, the evaluations are made at a series of adjacent points. As a result, the bounds of distances change only slightly per evaluation and the small subset of reference points that can potentially have the smallest distance, usually only 2 or 3 points, can be deteremined with little cost. Exact distances are only evaluated at these few reference points and the bounds are tighted for them.

calcmind **LOGICAL [.false.]** This option controls if the minimum distance to the reference geometries will be calculated.

mindcutoff **DOUBLE PRECISION [1D-5]** Desired precision for minimum distances. When set to nonzero, this allows the program to skip reevaluation of distances when the inequality relations along are enough to determine the minimal distances at the new geometry. Also when

the distances from two reference geometries are estimated to be within this range, or more precisely when the lower bound of distance to reference point A is not lower than the upper bound of distance to reference point B by more than ‘mindcutoff‘, point A will be eliminated from the list of potential minimal distance points because its distance will be at best almost the same as point B. Increasing this value will slightly decrease the accuracy of the distance, but significantly reduce the number of evaluation needed. The default value is found to yield reliable distance and at the same time only require a very small number of distance evaluations. Thus it is not recommended to change this value by significant amount.

gflname **CHARACTER(255) ['refgeom']** This file stores the geometries of all the reference points. It can be generated by enabling option ‘printError‘ during fitting.

nrpts **INTEGER [20]** The number of reference points contained in reference geometry input file. Note that the default will almost always have to be changed.

ndcoord **INTEGER [3]** Number of internal coordinates that will be used to define the distances. No more than 200 coordinates can be used.

dcoordls **INTEGER,dimension(ndcoord)** Index of internal coordinates used to define distances. Note that the index of full internal coordinate list is used, which includes coordinates generated by symmetry operations. Please refer to the list in standard output at the beginning of fitting procedure for the definition of these coordinates. The program does not require or try to determine the set of coordinate used is symmetric or linear independent.

Energy error estimation The evaluation program use the fitting error of energy and energy gradients at the most adjacent reference point, along with the displacement to that reference point, to estimate the error of fit energy at the current evaluation point, using a first order approximation. This value is only meaningful when the reference point is sufficiently nearby, and is therefore not a good criteria for deciding if new points should be generated. Rather, this is used to monitor if the trajectory enters region that have been populated by has large fitting error.

calcErr **LOGICAL [.false.]** This option controls if the first order estimation of fitting error will be estimated.

errflname **CHARACTER(255) ['error.log']** File that contains fitting error to energy and energy gradients. This file can be generated by enabling option ‘printError‘ during the fitting procedure.

Molden output The evaluation subroutine can take the geometries of evaluatoin and construct a molden file to enable visualization of trajectories. Each trajectory will generate a file named ‘molden$x.all‘, where $x is the index of the trajectory. Geometries are converted to Angstroms in this file.

atomLabels **CHARACTER(3),dimension(20) ['']** Atom labels for each group of atoms. Atom groups of each atoms are defined by option ‘atmgrp‘ in the GENERAL namelist. These names are only used to mark atoms in the molden file.

molden_p **INTEGER [100]** One geometry will be included in the file every ‘molden_p‘ evaluations. This is useful because trajectories usually compose of a very large number of geometries that have very small displacements. This option thus allows to generate a molden file that contains a reasonable number of geometries with noticable distances.

m_start **INTEGER [100]** The number of evaluations before the program starts to store geometries to the molden file.

Potential modification The evaluation subroutine allows diagonal corrections with very simple forms to allow users to make slight modifications of the surface which will not change the descriptions of the conical intersections. Possible causes may be to correct size inconsistency problems with the ab initio data, or to shift the energies by a fixed amount. This functionality is currently very limited and is still in construction

eshift **DOUBLE PRECISION [0d0]** Flat shift of all energies.

nfterms Number of Morse-potential type diagonal shifts. The shift takes the forms of
D∗(1-w/w0)^2
D usually specifies the change in dissociation energy, which is defined by options ‘fcoef‘. w is an internal coordinate defined in ‘coord.in‘, the index of which is defined by ‘fterm‘.

Parameter ‘w0‘ determines where this correction will vanish, which is determined by option ‘forig‘. Analytical gradients are also corrected with the gradients of this shift term. The same shift will be applied to all diagonal blocks. Note that even though positions of conical intersections will not move, the derivative coupling vectors may experience small changes.

fterm **INTEGER,dimension(:) [0]** Which one of the internal coordinates will be used to perform the shift. The internal coordinates are defined by ‘coord.in‘. Note that the indexing includes coordinates generated by symmetry. Please refer to the output of surfgen fitting program to the definition of these coordinates. The program does not assume or check if the set of coordinates used are symmetric.

fcoef **DOUBLE PRECISION,dimension(:) [0d0]** Expansion coefficients of each Morse form corrections, shown as D in the above equations. Normally this has to be a small number.

forig **DOUBLE PRECISION,dimension(:) [1d0]** Origin where the correction will vanish. Usually the value of the above defined coordinate at the equilibrium geometries of the bound structure is used. Small changes usually make no significant differences.

SEE ALSO

`connect.in`(1) `coords.in`(1), `irrep.in`(1), `hd.data`(1), `points.in`(1), `potlib`(1), `surfgen`(1),

BUGS

Please send bug reports to Xiaolei Zhu ⟨virtualzx@gmail.com⟩