Kaplan: System Verification and Validation Plan

Jen Garner

October 22, 2018

1 Revision History

Date	Version	Notes
Saturday 20 October, 2018	1.0	Write initial draft (this time with a spell check for Texstudio...)

2 Symbols, Abbreviations and Acronyms

See the SRS document for Kaplan for other abbreviations and symbols, particularly see Section 2 (Reference Material).

symbol	description
T	Test
FT	Functional Test
NFT	Non-Functional Test
QCM	Quantum Chemical Method
BS	basis set
SRS	Software Requirements Specification

[symbols, abbreviations or acronyms – you can simply reference the SRS tables, if appropriate —SS]

Contents

List of Tables

List of Figures

The system verification and validation (systVnV) plan document is designed to outline the tests for Kaplan and show how the software will be satisfying its requirements. Verification means that the software was built correctly and validation means that the correct software was built (for the intended problem). The two important sections to this document include the planning section (4) and the testing section (5). In Section 4, a brief overview as to how to verify the SRS, Design, and Implementation of Kaplan will be given. Furthermore, software validation techniques will be discussed. In Section 5, specific test inputs will be provided, and the expected outputs for these tests will be described. The functional tests (FT) are designed to satisfy functional requirements (R). The non-functional tests (NFT) are designed to satisfy some of the non-functional requirements. All requirements are listed in the SRS documentation. Section 5.3 gives traceability for these requirements. A usability questionnaire is given in Appendix 7.2.

[provide an introductory blurb and roadmap of the Verification and Validation plan —SS]

3 General Information

3.1 Summary

Kaplan is designed to search the potential energy space of molecules for possible conformational isomers. The inputs will be some geometry specification for the molecule(s) of interest, a quantum chemical method (QCM), and a basis set (BS). The output will be a geometry specification and a list of relevant energies.

[Say what software is being tested. Give its name and a brief overview of its general functions. —SS]

3.2 Objectives

These tests should ensure that the input geometry can be converted to z-matrix format, such that the dihedral angles can be extracted for manipulation. The tests should also accurately procure a molecule's energy (which confirms that the input geometry converges with the given basis set and method). Once the conformational isomers have been found, the tests should certify that they are distinct from one another (and not merely a translation

or rotation in space). The output geometries should be parsable by other quantum chemical software.

Kaplan intends to be simple in its use and upkeep. Any error messages should be verbose and steps to fix the issues will be given as suggestions. The maintainability is important, since the code must interact with many other libraries. If the code is not maintainable and the related libraries change, then it will be difficult and time consuming to perform updates.

The tests should prove that Kaplan is robust with respect to a wide range of inputs, and (in passing) they should convince the user that their returned structures are chemically meaningful. The performance of Kaplan should be easily tracked and it should be evident to the user if their method and basis set are too computationally expensive for the input molecule.

[State what is intended to be accomplished. The objective will be around the qualities that are most important for your project. You might have something like: "build confidence in the software correctness," "demonstrate adequate usability." etc. You won't list all of the qualities, just those that are most important. —SS]

3.3 Relevant Documentation

The Software Requirements Specification (SRS) document included in the Kaplan repository should be referenced for any definitions. It may be necessary to reference the documentation for external software such as Psi4 Parrish et al. (2017), Openbabel O'Boyle et al. (2011), oba (2018), or Gaussian Frisch et al. (2016).

[Reference relevant documentation. This will definitely include your SRS —SS]

4 Plan

4.1 Verification and Validation Team

Mulder, c'est moi.

[Probably just you. :-) —SS]

4.2 SRS Verification Plan

As stated in the SRS, two likely changes are present for Kaplan. First, LC1 states that the fitness function may not be a linear combination of the sum of energies and the sum of RMSD values. Once the code has been written and tested, then the fitness function can be modified to determine if a more efficient and/or accurate function exists. The modification will depend on the time available; the optimal fitness function is not expected to be found, but rather to be improved upon.

Second, LC2 highlights assumption A11, which states that the conformer space is independent of the ordering of atoms. If n_a atoms are in molecule M, then $n_a - 3$ dihedral angles are needed. There are a total of n_a choose $n_a - 3$ ways to select dihedral angles (simplified below in Equation 1). The question to answer will be: does the set of chosen atoms change the conformational isomers that are found or the time needed to find them?

$$\binom{n_a}{n_a - 3} = \frac{n_a!}{3!(n_a - 3)!} = \frac{n_a(n_a - 1)(n_a - 2)}{6} \; ; \; n_a > 3 \tag{1}$$

This question is relatively easy to test, as the input molecular geometry can be re-ordered such that different subsets of atoms are chosen to construct the z-matrix dihedral angles. It should be noted here that this test should be run for M with as little symmetry as possible (C_1 point group), to contract the search space and avoid comparing mirror images. The time to solution/energies (output) will be compared for M. If the results are not equal (when using the same random seed), then there is either a bug in the code, or the choice of dihedral angles matters. A different time to solution suggests that the energy calculations are somewhat dependent on the input configuration. A different energy implies that some dihedral angles are more important than others. The next question here would be - can a pattern be found to minimize the time to solution or find lower energy conformers? For example, does the set of dihedral angles, D_i, with heavier atoms converge more rapidly? Does D_i outperform when atoms are chosen with more bonded atoms?

As for the SRS document itself, the author will review it again once the rest of the program has been written to see if its requirements have been met. The author can ask her supervisor to review the paper as well as a few lab members. Brooks MacLachlan has already raised some github issues for the SRS document that the author will address by the end of this semester (Fall

2018). Professor Spencer Smith will also review the SRS and give feedback.

[List any approaches you intend to use for SRS verification. This may just be ad hoc feedback from reviewers, like your classmates, or you may have something more rigorous/systematic in mind.. —SS]

4.3 Design Verification Plan

The design will be reviewed during a meeting with the author's supervisor.

[Plans for design verification —SS]

4.4 Implementation Verification Plan

This document will represent the system testing. Another document will be written to outline the unit tests to be performed. The code will be written in Python, and nosetests will be used to run the tests. Various linters will be used, such as pylint, pydocstyle, and pycodestyle, to ensure that standards are being upheld for the code formatting and documentation (PEP8, numpy docstrings). The implementation will also be tested by classmate, Malavika Srinivasan, and Professor Spencer Smith.

[You should at least point to the tests listed in this document and the unit testing plan. —SS]

4.5 Software Validation Plan

Journal articles that explore the conformer space of a given molecule will be used as a reference, for example see Herrebout et al. (1995). Using the same input molecule, determine whether Kaplan can find the same conformers.

[If there is any external data that can be used for validation, you should point to it here. If there are no plans for validation, you should state that here. —SS]

5 System Test Description

5.1 Tests for Functional Requirements

[Subsets of the tests may be in related, so this section is divided into different areas. If there are no identifiable subsets for the tests, this level of document structure can be removed. —SS]

5.1.1 FT1 - Data Input and Output

These tests will ensure that all data conversion and formatting is handled correctly by Kaplan. As per functional requirement R1, the user must provide 6 items to the program: n_G, C_E, C_{RMSD}, BS, QCM, and a molecular geometry.

1. Non-Geometry Input Type Check

   Description: test the inputs and check that they are of the correct type and have sensical bounds.

   Control: Automatic

   Initial State: empty, no molecule input.

   Input:

Input Variable	Special Cases	Usual Case	Error Cases
n_G	0, 1	5*	'a', "bloop", 1.0, 5.43, -5
C_E	0	0.5, 5	'a', "blip", -10
C_{RMSD}	0	0.7, 2	'a', "banana", -13
BS	BS from file*	"cc-pVTZ", "STO-6G"	'o', "not-a-basis-set", unavailable basis (for atoms), unavailable basis (for external program), 10, 0.3
QCM	N/A	"HF", "CCSD"	'o', "not-a-method", unavailable method, 7, 0.25

Table 1: Test cases for the input variables.

* May need to issue a warning if the user provides a value for n_G that is larger than 20 (or do stress testing to find the upper bound for n_G).
** Depending on the external program used for energy calculations, a user-specified basis set may not be supported. Here is a link to a

tutorial for how to make a basis set for Gaussian (using gen keyword) Barroso (2011), Frisch et al. (2016).

Output: Expect the program to raise an input error (type error) for the error cases. The coefficients (C_E and C_{RMSD}) can be either a floating point or an integer value, but must not be negative. The special cases occur when there is a reduction of function calls (i.e. have $n_G = 0$ means returning 0 rather than calculating S_E or S_{RMSD}). The BS and QCM chosen must be available in the external program (as used for energy calculations). Ensuring that the BS is applicable to the input molecule, and ensuring energetic convergence for the input molecule, is not covered by this test - see subsequent Test FT1-2.

How test will be performed: Provide inputs from each category as per Table 1 and ensure that they provide errors where expected and minimize computation by following protocol for special cases.

2. Test Input Geometry

   Description: This test ensures that a molecule can be input in various ways (SMILES, xyz, z-matrix) and still produce the same geometry. Furthermore, it checks that the geometry is compatible with the chosen BS and QCM. If the geometry can converge during the optimization using that BS and QCM, then the inputs pass.

   Control: Automatic

   Initial State: N/A

   Input: set of various test molecules (as SMILES, xyz, and z-matrix). Provide geometries that are purposefully bad (for example, a flat molecule that is supposed to be 3D). These geometries should not converge, no matter how the dihedral angles are manipulated. Provide a geometry for which the energy calculation will not converge, but can converge provided the dihedral angles are changed. Here, an example could be to superimpose two atoms that can otherwise be separated by free rotation. Note: depending on how robust the chosen external program is for energy calculations, this type of geometry may be hard to find. A last part of this test will be to input a geometry for which the chosen BS does not include one or more of the molecule's atoms. A good resource to check for this availability is the Basis Set Exchange Feller (1996), Schuchardt et al. (2007), bas (2018). Some atoms, especially

the heavier elements, do not have as many BS available as others (they simply have not been calculated yet).

Output: Errors are expected when all energies are calculated as zero (which will be the default value for non-converging calculations) - i.e. $S_E = 0$ for $n_G > 0$. Note: non-converging here does not mean that the external program fails (or runs out of computational time or resources, etc. - those are separate errors). Errors should also be raised when the BS is available, but there is an atom in the geometry for which the BS has not been parametrized. Given the same random seed and molecule, the results should be equal (within a small tolerance), regardless of whether the geometry is initialized as a SMILES string, set of xyz coordinates, or z-matrix. Note: this may be changed later if the SMILES strings give very different geometries to known experimental structures of molecules.

How test will be performed: This test will require running the program for the testing molecules, as described in the input section. It is only intended to satisfy the first part of R4, which requires that the energy calculations converge.

3. Test Output Generation

   Description: as per R5, the program should generate a geometry from D_i and the starting geometry.

   Control: automatic

   Initial State: an optimised input molecule versus an unoptimised input molecule.

   Input: list of dihedral angles, D_i, for each conformer, alongside the starting geometry for that molecule.

   Output: xyz coordinates for each conformer.

   How test will be performed: Kaplan will generate a new z-matrix file using the original geometry and the new dihedral angles. Then, it will convert this z-matrix to xyz coordinates using Openbabel and compare the outputs. For the optimised molecule, test that the geometry is different from the input. For the non-optimal molecule, test that the geometries are equal.

5.1.2 FT2 - Calculations

1. Calculate Fit_G

   Description: This test will be a simple check on the formula for Fit_G. The C_{RMSD} and C_E coefficients will be tested such that their limits will be sufficiently covered.

   Control: Automatic

   Initial State: n_G = 0, 1, 2, 7, -10, 1.6

   Input: For each initial state, calculate Fit_G for the parameters in Table 2.

Variable	Values			
C_{RMSD}	-1.4	0	1	0.5
C_E	-4	1	0	0.5
S_E	-120	500.24	23.4	400.3
S_{RMSD}	-1.5	4.3	2	8.3
Result Fit_G	Error	500.24	2	204.3

Table 2: Example input and output of Fit_G calculations for some values of coefficients and summations.

   Output: The value for n_G (number of conformers being optimised) starts at 0, representing the base case where no conformers are being searched for. In this case, the result should be zero, and no functions need to be called. This test ensures no divide-by-zero errors occur. The second value ($n_G = 1$) assumes one conformer. The expected behaviour is calculate Fit_G and assert that the value is equal to E_i (the energy of the conformer). Note: some errors may arise if the S_{RMSD} value is not correctly handled (i.e. index-out-of-range errors), since there is only one molecule and the RMSD expects $n_G > 1$.

   The next two values for n_G should be calculated as usual (one odd and on even value). No errors are expected. The last two values are invalid inputs for n_G and should raise errors (n_G cannot be a float and it cannot be negative).

For each value of n_G, 3 sets of coefficients will be tested. The chosen values attempt to represent the limits of what should be calculated.

How test will be performed: Run all input combinations with a test script and pass to nosetests.

2. Calculate S_E

   Description: Psi4 is the chosen quantum chemistry package that will be used to run most energy calculations. More packages will be added later-on in the project. In case updates to Psi4 change the output energies, these tests will confirm that the same answer is being returned for the same input calculation. The results can be compared to literature values (within error tolerance).

   Control: Automatic

   Initial State: $n_G = 5$

   Input: Set of input molecules (previously calculated using Gaussian), BS, and QCM.

   Output: S_E

   How test will be performed: use Psi4 to calculate the energies of a set of geometries. The geometries will have two types: (1) they are all equivalent, and (2) they have been minimally randomized based on an initial convergent geometry. Calculate the S_E. Ensure that the result is positive and (if applicable) compare to previous test results. Compare the results to the Gaussian output for the same BS and QCM and ensure that value is within a tolerance.

3. Calculate S_{RMSD}

   Description: The rmsd repository made by user charnley
   (https://github.com/charnley/rmsd)
   will be used to calculate the RMSD between two molecular geometries. This test should ensure that the S_{RMSD} value is calculated correctly.

   Control: automatic

   Initial state: $n_G = 0$, 1, 2, 6

   Input: given molecule, M, rotate this molecule 90°, called M_r. Translate M 2Å, called M_t.

Output: S_{RMSD}

How test will be performed: use rmsd repository to calculate the RMSD between geometries. As with the energy calculations, there should be a test done on equivalent geometries to ensure the RMSD is 0. The RMSD should also be 0 for M_r and M_t when compared to M. The second set of tests should be on geometries that have been changed by a small, random amount (+/-0.2Å). Since such a change should not impact the rotation matrix, the RMSD can be calculated directly in a spreadsheet or other tool for validation.

5.2 Tests for Nonfunctional Requirements

5.2.1 NFT1 - External Package Compliance Tests

This section of testing ensures that Kaplan satisfies the non-functional requirement of working well with other quantum chemistry packages. These tests will show that the chosen quantum chemistry packages work as intended and within quantifiable limits.

1. Testing Formatting Conversion Error

   Description: This test will determine the change in geometry that occurs when the molecule undergoes a formatting change. The conversion is performed using Openbabel software. The test will confirm that Openbabel is a suitable library to use for the program and quantify the rounding/floating point error associated with n conversions. This test is a form of stress testing. The author is open to suggestions as to how many conversions should be performed before considering the conversion to be stable. If this test passes, then the program will not need to keep re-reading the input geometry from a file nor maintain the original geometry as a variable in memory. The program will therefore be more robust and reliable upon quantifying the formatting change error.

   Type: functional (automatic running, manual inspection)

   Initial State: N/A

   Input/Condition:
   SMILES string for caffeine CN1C=NC2=C1C(=O)N(C(=O)N2C)C
   Number of conversions $\rightarrow n = 1, 10, 20, 30..10000$.

Output/Result: coordinates for caffeine after n conversions. Check D_i each conversion and atom ordering. R5 should be satisfied here, in that the geometry can be regenerated from D_i.

How test will be performed: Starting with the SMILES string for caffeine, use Openbabel to convert this format to Cartesian coordinates. Then, convert the xyz coordinates to a z-matrix with Openbabel. One conversion counts as converting to and from a z-matrix. Complete n conversions and calculate the RMSD error between the initial and final xyz geometries. Plot the results with Matplotlib (Python library) for all n. Ensure that there is a tolerance for the number of conversions that Kaplan executes. Note: it may be necessary to test this tolerance for molecules of differing sizes. This test should also ensure that, after each conversion, the dihedral angles list D_i and the atom ordering remain constant.

2. Testing Error between Quantum Chemical Packages

   Description: In order to determine which packages will to interface with Kaplan, their energy calculation results will be compared for the same BS and QCM. The energies should be similar within a tolerance, and this tolerance should not impact the results of the conformer search.

   Type: functional

   Initial State: N/A

   Input/Condition: set of molecules, BS, and QCM.

   Output/Result: E_i for each molecule.

   How test will be performed: Gaussian and Psi4 are the first two programs that will interface with Kaplan. The difference in energy for the same calculation shall be documented. If there is a significant difference in results for any given BS or QCM, then these inputs may be removed as options from Kaplan (at least until the case can be determined).

5.2.2 NFT2 - Robustness Tests for Fit_G

1. Testing different Fit_G functions.

   Description: Since the form of Fit_G is likely to change (and will be changed for research purposes), the program should be able to handle a change in form for this function.

Type: functional

Initial state: N/A

Input/condition: run all tests for Kaplan (that are dependent on the calculation of Fit_G) such that the Fit_G is evaluated as:

(a) $\frac{C_E S_E}{C_{RMSD} S_{RMSD}}$

(b) $\frac{C_{RMSD} S_{RMSD}}{C_E S_E}$

(c) $C_E C_E^2 + C_{RMSD} \sqrt{S_{RMSD}}$

(d) $C_{RMSD} C_{RMSD}^2 + C_E \sqrt{S_E}$

(e) $C_E \ln S_E + C_{RMSD} e^{S_{RMSD}}$

(f) $C_{RMSD} \ln S_{RMSD} + C_E e^{S_E}$

Output/result: returns the results of the tests involving Fit_G. If any tests failed, these tests should be investigated and fixed such that they pass for the new function for Fit_G.

How test will be performed: write the new Fit_G functions in an external file and use a test to call them. It might be easier to add the Fit_G function choice as an input parameter to the program (however this choice may not be readily visible to the user).

5.2.3 NFT3 - Parallelizability and HPC Testing

1. Testing for n cores.

   Description: this test will quantify the performance increase that occurs when the number of available computing cores is increased.

   Type: late-stage implementation (functional, manual observation)

   Initial state: no optimization has been performed on molecule, M.

   Input/condition: use n cores to optimize M, where n = 1, 2, 4, 8.

   Output/result: return the running time for each optimization.

   How test will be performed: Plot the time to solution (y) with increasing n (x) using Matplotlib. Inspect this plot to ensure that the slope of this plot is not negative (i.e. increasing cores increases performance). This will likely be part of a later phase of implementation and testing.

2. Testing on HPC cluster.

   Description: the program will be used primarily on a HPC cluster (for example, Compute Canada's cluster, Graham). This test should require the user to install the program on a HPC cluster and run the other tests.

   Type: manual installation, automatic running

   Input/condition: tests for Kaplan.

   Output/result: result of tests.

   How test will be performed: author will manually install the program on Graham and run the tests.

5.3 Traceability Between Test Cases and Requirements

[Provide a table that shows which test cases are supporting which requirements. —SS]

	R1	R2	R3	R4	R5
FT1-1	X				
FT1-2	X				
FT1-3					X
FT2-1		X		X	
FT2-2		X	X	X	
FT2-3		X	X		

Table 3: traceability table for functional requirements and tests.

	NFR1	NFR2	NFR3	NFR4	NFR5
NFT1-1					X
NFT1-2					X
NFT2-1	X				
NFT3-1			X		
NFT3-2			X		

Table 4: traceability table for non-functional requirements and tests.

6 Static Verification Techniques

[In this section give the details of any plans for static verification of the implementation. Potential techniques include code walkthroughs, code inspection, static analyzers, etc. —SS]

It is assumed that any external programs that are used by Kaplan have their own tests and that these tests will be run (and hopefully pass) on the applicable systems.

Planned external tests for these programs (note: cannot test Gaussian as it is not open-source):

- Openbabel: https://openbabel.org/docs/dev/Contributing/Testing.html
- Psi4: http://www.psicode.org/psi4manual/1.2/testsuite.html
- rmsd: https://github.com/charnley/rmsd/blob/master/tests.py

The author will provide a code review for this project at a lab meeting. The other lab members will answer the usability survey questions as outlined in the appendix section 7.2.

Travis will be used to verify the build when a commit is made to the Kaplan repository. Travis will attempt to install Kaplan as a Conda package inside a Conda environment.

Here are some important style links:

- https://www.python.org/dev/peps/pep-0008/
- https://pypi.org/project/pycodestyle/
- https://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_numpy.html

References

Basis set exchange, 2018. URL https://bse.pnl.gov/bse/portal.

The open babel package, version 2.3.1, 2018. URL http://openbabel.org.

Joaquin Barroso, 2011. URL https://joaquinbarroso.com/2011/11/02/gen-keyword-gaussian/.

D. Feller. The role of databases in support of computational chemistry calculations. *J. Comp. Chem.*, 17:1571–1586, 1996.

M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, G. A. Petersson, H. Nakatsuji, X. Li, M. Caricato, A. V. Marenich, J. Bloino, B. G. Janesko, R. Gomperts, B. Mennucci, H. P. Hratchian, J. V. Ortiz, A. F. Izmaylov, J. L. Sonnenberg, D. Williams-Young, F. Ding, F. Lipparini, F. Egidi, J. Goings, B. Peng, A. Petrone, T. Henderson, D. Ranasinghe, V. G. Zakrzewski, J. Gao, N. Rega, G. Zheng, W. Liang, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, K. Throssell, J. A. Montgomery, Jr., J. E. Peralta, F. Ogliaro, M. J. Bearpark, J. J. Heyd, E. N. Brothers, K. N. Kudin, V. N. Staroverov, T. A. Keith, R. Kobayashi, J. Normand, K. Raghavachari, A. P. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, J. M. Millam, M. Klene, C. Adamo, R. Cammi, J. W. Ochterski, R. L. Martin, K. Morokuma, O. Farkas, J. B. Foresman, and D. J. Fox. Gaussian16 Revision B.01, 2016. Gaussian Inc. Wallingford CT.

WA Herrebout, BJ Van der Veken, Aiying Wang, and JR Durig. Enthalpy difference between conformers of n-butane and the potential function governing conformational interchange. *The Journal of Physical Chemistry*, 99 (2):578–585, 1995.

N M O'Boyle, M. Banck, James C. A, C. Morley, T. Vandermeersch, and G. R. Hutchison. Open babel: An open chemical toolbox. *J. Cheminf.*, 3, 2011. doi: 10.1186/1758-2946-3-33.

R. M. Parrish, L. A. Burns, D. G. A. Smith, A. C. Simmonett, A. E. DePrince III, E. G. Hohenstein, U. Bozkaya, A. Yu. Sokolov, R. Di Remigio,

R. M. Richard, J. F. Gonthier, A. M. James, H. R. McAlexander, A. Kumar, M. Saitow, X. Wang, B. P. Pritchard, P. Verma, H. F. Schaefer III, K. Patkowski, R. A. King, E. F. Valeev, F. A. Evangelista, J. M. Turney, T. D. Crawford, and C. D. Sherrill. Psi4 1.1: An open-source electronic structure program emphasizing automation, advanced libraries, and interoperability. *J. Chem. Theory Comput.*, 13:3185–3197, 2017.

K.L. Schuchardt, B.T. Didier, T. Elsethagen, L. Sun, V. Gurumoorthi, J. Chase, J. Li, and T.L. Windus. Basis set exchange: A community database for computational sciences. *J. Chem. Inf. Model.*, 47:1045–1052, 2007. doi: 10.1021/ci600510j.

7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

7.2 Usability Survey Questions

[This is a section that would be appropriate for some projects. —SS]

1. Were you able to install the program using the instructions alone, or did you require additional researching/help? If any steps were unclear, please explain how they might be improved.
2. Did the program accept your molecular input?
3. Did the program return the same number of conformer geometries and energies as specified by your choice of input for n_G?
4. When you view these conformers (chimera, VMD, pymol, etc.), do their geometries seem appreciably different from one another?
5. Did your selection of BS or QCM result in any convergence errors?
6. What did you choose for your coefficients? Do you think that the results can be improved with a different choice of C_E and/or C_{RMSD}?
7. Was the output delivered more quickly or more slowly than expected?