

VM2
Version 2.8.2

User's Manual

VeraChem LLC



Copyright (c) 2015-2019, VeraChem LLC, Germantown, MD, USA. All rights reserved.

VeraChem has been issued a patent (**USPTO Patent No. 8,140,268**) for the VM2 method.

Contact:

For information regarding VM2 software package licensing contact VeraChem LLC at sales@verachem.com

For technical support contact VeraChem LLC at support@verachem.com

For general enquiries contact VeraChem LLC at info@verachem.com

Contents

I. Introduction

1. VM2 background and theory
 - 1.1 Purpose of VM2
 - 1.2. Binding free energy
 - 1.3. Enthalpy-entropy compensation
 - 1.4. Computational free energy methods
 - 1.5. Mining minima approximation
 - 1.6. Molecular coordinate systems
 - 1.7. Conformational searching
 - 1.8. Configuration integrals
 - 1.9. Filtering of conformer repeats
 - 1.10. VM2 algorithm
 - 1.11. Molecular system partitioning
2. VM2 package modules
 - 2.1. VM2 : VeraChem second-generation mining minima
 - 2.2. Vconf : Conformational search
 - 2.3. VfreeE : Configuration integration
 - 2.4. Vstereochem : Stereochemistry
 - 2.5. Vfilter : Filtering of repeat conformers
 - 2.6. Vrmsd : Multiple conformer RMSD
 - 2.7. Vhessian : Potential energy 2nd-derivative calculation
 - 2.8. Vgeomopt : Geometry optimization
 - 2.9. Vpotential : Potential energy and 1st-derivative

II. Energy Potentials

1. Molecular mechanics potentials
 - 1.1. Force field support
 - 1.1.1. Basic potential energy form
 - 1.1.2. Bond and angle energy terms
 - 1.1.3. Torsion energy terms
 - 1.1.4. Coulomb nonbonded energy
 - 1.1.5. Van der Waals energy term
 - 1.1.6. CHARMM parameter sets
 - 1.1.7. OPLS parameter sets
 - 1.1.8. Amber parameter sets
 - 1.1.9. Dreiding parameter set
 - 1.2. The VeraChem topology and force field parameter file (.top)
 - 1.2.1. Identification of force field
 - 1.2.2. Atom types, charges, and van der Waals parameters

- 1.2.3. Bonds and force constants
- 1.2.4. Angles and angle force constants
- 1.2.5. Proper dihedrals and corresponding parameters
- 1.2.6. Improper dihedrals and corresponding parameters
- 1.2.7. Nonbonded 'fixes'
- 1.2.8. Final count of atoms, bonds, angles, proper and improper dihedrals
- 1.2.9. !NDON
- 1.3. Solvation models
 - 1.3.1. Generalized Born (GB)
 - 1.3.2. Constant dielectric (CD)
 - 1.3.3. Distance dependent (DD)
 - 1.3.4. Poisson Boltzmann Surface Area (PBSA)
- 2. Quantum mechanics (not supported in this version)

III. Installation

- 1. Obtaining package, and package choices
 - 1.1. Commercial licensing
 - 1.2. Trial license
 - 1.3. Academic licensing
 - 1.4. Package choices
- 2. Operating systems and hardware
 - 2.1. Linux workstations
 - 2.2. Linux desktop
 - 2.3. Linux clusters
 - 2.4. Linux workstations and clusters with NVIDIA GPU accelerators
 - 2.5. OSX
 - 2.6. MS Windows
- 3. Installation procedure
 - 3.1. Download the VM2 package
 - 3.2. License file
 - 3.3. Environment variables for installation
 - 3.4. Requirements for installation
 - 3.5. Installation script
- 4. Installed VM2 package structure
 - 4.1. Helper tools
 - 4.1.1. Vcharge : assignment of partial atomic charges
 - 4.1.2. Vconf : 2D to 3D and small molecule conformational search
 - 4.1.3. prm2top : AMBER formatted input data files to VM2 input data files
 - 4.1.4. psf2top : CHARMM formatted input data files to VM2 input data files
 - 4.1.5. mmo2top : Schrodinger mmo file to VM2 input data files
 - 4.2. VM2 executables
 - 4.3. Supplied libraries

5. Environment variables for running validation calculations
6. Installation validation tests
 - 6.1. Helper tools validation tests
 - 6.1.1. Maestro/Macromodel pathway
 - 6.1.2. AmberTools pathway
 - 6.1.3. Biovia Discovery Studio Visualizer (DSV) pathway
 - 6.1.4. VCharge validation test
 - 6.1.5. VConf validation test
 - 6.2. VM2 validation tests

IV. Parallel Processing

1. VM2 serial calculation bottlenecks
2. Parallelization strategy
3. MPI coarse-grained parallelization
 - 3.1. Coupled MPI conformational search
 - 3.2. Uncoupled MPI conformational search: introduction of diversity
 - 3.3. Non-blocking MPI communication
4. Fine-grained parallelization
 - 4.1. OpenMP
 - 4.1.1. OpenMP fine-grained parallelization of GB energy-gradient
 - 4.1.2. OpenMP parallelization of Hessian transformation and diagonalization
 - 4.1.3 OpenMP parallelization of PBSA
 - 4.2. CUDA
 - 4.2.1. CUDA based fine-grained parallelization of the GB energy-gradient
 - 4.2.2. CUDA parallelization of Hessian transformation and diagonalization
5. Combined coarse grained-fine grained parallelization
 - 5.1. MPI-OpenMP parallelization
 - 5.2. MPI-CUDA and MPI-OpenMP-CUDA parallelization

V. Molecular system and input data file preparation

1. Molecular systems
 - 1.1. Ligands
 - 1.2. Proteins
 - 1.3. Protein+ligand complexes
 - 1.4. Host+ligand complexes
2. Molecular system data sources
 - 2.1. The Protein Data Bank (PDB)
 - 2.2. The Cambridge Crystallographic Data Center (CCDC)
 - 2.3. The Binding Data Base (BindingDB)
 - 2.4. Chemical components in the PDB (PDBeChem)
 - 2.5. ZINC: a free database of commercially available compounds

3. Molecular system preparation and computational model choices
 - 3.1. Proteins
 - 3.1.1. Missing side chains
 - 3.1.2. Non-standard amino acids
 - 3.1.3. Chain breaks
 - 3.1.4. Hydrogen addition
 - 3.1.5. Stereochemistry
 - 3.1.6. Metal centers
 - 3.1.7. Solvent and ions
 - 3.1.8. Specific residue protonation states
 - 3.1.9. Residue mutations
 - 3.1.10 Choice of protein real/live set
 - 3.1.11. Protein atom typing and parameters
 - 3.2. Host molecules and ligands
 - 3.2.1. Hydrogen addition
 - 3.2.2. Protonation states
 - 3.2.3. Bond order recognition
 - 3.2.4. Stereochemistry
 - 3.2.5. Generalized atom typing and parameters
4. Mandatory formatted molecular data file generation
5. System and data file preparation routes
 - 5.1. Route 1: Conversion of Amber style input files
 - 5.1.1. AmberTools
 - 5.1.2. UCSF Chimera
 - 5.1.3. OpenMM
 - 5.2. Route 2: Conversion of CHARMM style input files
 - 5.2.1. Discovery Studio Visualizer
 - 5.2.2. CHARMMing
 - 5.2.3. CHARMM-GUI
 - 5.3. Route 3: Maestro/Macromodel (OPLS2005)
 - 5.3.1. System preparation and generation of MMO files
 - 5.3.2. Conversion of MMO files to .crd, .top, and .mol/.sdf
 - 5.4. Route 4: VeraChem preparation tools

VI. Running VM2 calculations

1. Mandatory input file (.inp)
2. Mandatory data files
 - 2.1. Card (.crd) file
 - 2.2. Topology/parameter (.top) file
 - 2.3. Molecular data (.mol/.sdf) file
3. Optional data files

- 3.1 Formatted file defining atoms/points in space used for automatic generation of protein real/live sets
- 3.2. Formatted text file that explicitly defines fixed and mobile atoms
- 3.3. Formatted file containing atoms to be constrained
- 3.4. Formatted file containing atoms to be excluded from conformational searches
- 3.5. Standard format files containing coordinates of previously generated molecular conformers
 - 3.5.1 Ability to read in multiple conformers to initiate runs
- 4. Environment variables
 - 4.1. Placeholder
 - 4.2. Placeholder
 - 4.3. Placeholder
 - 4.4. Placeholder
- 5. Run scripts
 - 5.1. Bash shell scripts
 - 5.1.1. Example 8 MPI process run
 - 5.1.2. Example 8 MPI processes, 2 OpenMP threads per MPI process, 1 GPU per MPI process
 - 5.2. C-shell scripts
 - 5.2.1. Example 8 MPI process run using C-shell
 - 5.2.2. Example 8 MPI processes, 2 OpenMP threads per MPI process, 1 GPU per MPI process, using C-shell
 - 5.3. PBS batch queue scripts
 - 5.3.1. Example 8 MPI process PBS run
 - 5.3.2. Example 8 MPI processes, 2 OpenMP threads per MPI process, 1 GPU per MPI process, using PBS
 - 5.4. SLURM batch queue scripts
 - 5.4.1. Example 8 MPI process SLURM run
 - 5.4.2. Example 8 MPI processes, 2 OpenMP threads per MPI process, 1 GPU per MPI process, using SLURM
 - 5.5. LSF batch queue scripts
 - 5.5.1. Example 8 MPI process LSF run
 - 5.5.2. Example 8 MPI processes, 2 OpenMP threads per MPI process, 1 GPU per MPI process, using LSF
- 6. CloudVM2 – running VM2 on Amazon Web Services (AWS) cloud environment
 - 6.1. Outline of CloudVM2 operation
 - 6.2. Architecture and economics
 - 6.3. CloudVM2 GUI
 - 6.3.1. Main menu
 - 6.3.2. Start menu
 - 6.3.3. Check menu
 - 6.3.4. Retrieve menu

7. Front end workflow
 - 7.1. General scheme for ligand series and receptor binding
 - 7.2. Local clusters
 - 7.3. CloudVM2

VII. Output files

1. Standard output files
 - 1.1. Verbose output file (.out) contains detailed description of all steps of the calculation
 - 1.2. Summary output file (.summary.out) contains a basic summary of the run, including energy tables at the end
 - 1.3. Binary restart file (.vcbin)
2. Optional output files
 - 2.1. Structural data
 - 2.2. Energy data
3. Back end workflow
 - 3.1. Generation of binding affinity tables
 - 3.2. Placeholder

VIII. Input file run options reference

1. Choice of system type and calculation type and other top-level control
 - 1.1. molSystemType : set molecular system type
 - 1.2. calcnType : set calculation type
 - 1.3. timeLimit : set calculation wall clock time limit
 - 1.4. readInConfs : read in previously generated molecular conformers
 - 1.5. ligandConfsToCrd : control the placement of read-in molecular conformers
 - 1.6. useCrdAsTemplate : controls template used when constructing complexes
 - 1.7. useCrdAsConf : when constructing conformers also use .crd as a conformer
 - 1.8. outputFormats : control formatted molecular data files to output
 - 1.9. fullEnergyBreakdown : controls level of detail in energy breakdown output
 - 1.10. splitOutputFormats : controls output of separate receptor/ligand data files
 - 1.11. limitConfsToOutput : limit the number of conformers output
 - 1.12. atomsToOutput : controls whether all atoms, real, or just live atoms output
 - 1.13. binaryFileRestart : option to restart a calculation from binary check point file
 - 1.14. Example usage 1
2. Molecular system definition options for protein macromolecules
 - 2.1. inputProtein
 - 2.2. setChainIds
 - 2.3. constructLiveReal

- 2.4. realCutoffDist
- 2.5. liveCutoffDist
- 2.6. symmetrizeRealSet
- 2.7. symmetrizeLiveSet
- 2.8. Example usage 2
- 2.9. Example usage 3
- 3. Molecular system definition options for host molecules
 - 3.1. inputHost
 - 3.2. Example usage 4
- 4. Molecular system definition options for ligand molecules
 - 4.1. inputLigand
 - 4.2. placeLigandMethod
 - 4.3. doSnapTemplatePairs
 - 4.4. snapTemplatePairsFC
 - 4.5. Example usage 5
- 5. Math related options e.g. control of random seed generation
 - 5.1. randomSeedsMethod
 - 5.2. setRandomSeeds
 - 5.3. Example usage 6
- 6. VeraChem mining minima (VM2) calculation options
 - 6.1. convTolVm2
 - 6.2. maxVm2Iters
 - 6.3. Example usage 7
- 7. General conformational search control options
 - 7.1. convTolConfsearch
 - 7.2. maxConfsearchIters
 - 7.3. confSearchStyle
 - 7.4. maxSearches
 - 7.5. modeRotnMax
 - 7.6. switchToRandomRotnMax
 - 7.7. numRlsearch
 - 7.8. ligandTranMax
 - 7.9. ligandRotnMax
 - 7.10. excludeBackBone
 - 7.11. excludeSideChains
 - 7.12. excludedAtomsFile
 - 7.13. forceConstCutoff
 - 7.14. deltaLevel1Cutoff
 - 7.15. nonBlockingUpdate
 - 7.16. doLoadBalance
 - 7.17. mixSearchBasis
 - 7.18. mixSearchIters
 - 7.19. mixSearchPicks
 - 7.20. doClusterBy

- 7.21. poolSize
- 7.22. relaxNonDriverAtoms
- 7.23. Example usage 8
- 8. Custom conformational search options
 - 8.1. Search
 - 8.2. modeSearch
 - 8.3. mode
 - 8.4. focusedSearch
 - 8.5. ndrivers
 - 8.6. drivers
 - 8.7. binRandomPairs
 - 8.8. modeDistMaxE
 - 8.9. ligandSearch
 - 8.10. sligandSearch
 - 8.11. rligandSearch
 - 8.12. ligandDistMaxE
 - 8.13. Example usage 9
- 9. Options and control of spatial boundary based conformer rejection
 - 9.1. boxedAtoms
 - 9.2. atomBoxSize
 - 9.3. ligandBoxSize
 - 9.4. Example usage 10
- 10. Options for free energy processing of conformers
 - 10.1. modeScanning
 - 10.2. temperature
 - 10.3. freeEnergyPreFactor
 - 10.4. Example usage 11
- 11. Stereochemistry checking and enforcement control
 - 11.1. maintainCisTrans
 - 11.2. maintainParity
 - 11.3. maintainProteinPepBonds
 - 11.4. Example usage 12
- 12. Control of filtering out conformer repeats
 - 12.1. preFilterCalcType
 - 12.2. pairCutoff1
 - 12.3. pairCutoff2
 - 12.4. pairRmsdCutoff1
 - 12.5. pairRmsdCutoff2
 - 12.6. firstConfCullE
 - 12.7. ConfCullE
 - 12.8. displaceCurrentConfs
 - 12.9. Example usage 13
- 13. Options for molecular alignment and RMSD calculation

- 13.1. preRmsdCalcType
- 13.2. preRmsdFilter
- 13.3. rmsdAllPairsMethod
- 13.4. confAlignment
- 13.5. numAlignAtoms
- 13.6. atomsToAlign
- 13.7. Example usage 14
- 14. Geometry optimization options and control, including constraints
 - 14.1. maxAtomGrad
 - 14.2. maxAtomGradLoose
 - 14.3. doPreoptSteps
 - 14.4. preoptMethod
 - 14.5. maxPreoptSteps
 - 14.6. geomoptMethod
 - 14.7. maxGeomoptSteps
 - 14.8. batchEnergyCutoff
 - 14.9. tetheredAtoms
 - 14.10. tetherForceConstant
 - 14.11. tetherScalingFactor
 - 14.12. tetherDistance
 - 14.13. tetherOrder
 - 14.14. nfreezeAtoms
 - 14.15. freezeAtoms
 - 14.16. Example usage 15
- 15. Molecular mechanics potential energy calculation: methods and usage control
 - 15.1. level1mmMethod
 - 15.2. level2mmMethod
 - 15.3. allowZeroWaterLJ
 - 15.4. allowZeroLJ
 - 15.5. mmAddFxdFxdConst
 - 15.6. Example usage 16
- 16. Molecular mechanics Generalized Born (GB) solvation model
 - 16.1. gbSolvationModel
 - 16.2. still97ParamSet
 - 16.3. gbDielectricExt
 - 16.4. gbDielectricInt
 - 16.5. gbCavityRadii
- 17. Molecular mechanics constant dielectric (CD) solvation model
 - 17.1. cdSolventDielectric
- 18. Molecular mechanics distance dependent (DD) dielectric solvation model
 - 17.1. ddCoefficient
- 19. Molecular mechanics Poisson Boltzmann Surface Area (PBSA) solvation model
 - 19.1. pbDielectricExt

- 19.2. pbDielectricInt
- 19.3. pbsaCavityRadii
- 19.4. sasaProbeRadius

IX. Ligand example

1. CHARMM pathway using Discovery Studio Visualizer (DSV)
 - 1.1. Get mol2 data file for chosen molecule: ibuprofen
 - 1.2. Load molecule into DSV
2. CHARMM pathway using the web user interface CHARMMing
 - 2.1. Get mol2 data file for chosen molecule: ibuprofen

X. Protein-ligand example: HIV-1 protease and 38 inhibitors

1. Setup
 - 1.1. Protein setup
 - 1.1.1. Remove all hetatoms and water atoms except atom 1580
 - 1.1.2. Extract the co-crystallized ligand
 - 1.1.3. Prepare the PDB file for tleap
 - 1.1.4. Run tleap to assign parameters
 - 1.1.5. Convert .prmtop and .inpcrd to .crd, .top, and .mol files
 - 1.2. Ligand Setup
 - 1.2.1. Initial 2D structures
 - 1.2.2. 2D to 3D conversion
 - 1.2.3. Generate partial charges and assign parameters to the ligands
 - 1.3. Define fixed and mobile protein atoms
 - 1.3.1. Generate co-crystallized ligand based AD-81 conformation
 - 1.3.2. Relax all hydrogen atoms in the system
 - 1.3.3. Distance based generation of real/live set
2. Run Calculations
 - 2.1. Generation of Ligand Starting Conformations
 - 2.1.1. Example run
 - 2.1.2. Options available for building conformer generation directories
 - 2.1.3. Options available for running conformer generation
 - 2.2. Protein-ligand calculations
 - 2.2.1. Example run
 - 2.2.2. Options available for building VM2 directories
 - 2.2.2. Options available for running VM2 calculations
3. Results Collection
 - 3.1. Generate binding free energy spreadsheets and collect conformer files
 - 3.2. Results generation options

XI. Host-guest example: Sampl6 Octa-acids and guests

1.Setup

- 1.1. Host Setup
 - 1.1.1. Source files
 - 1.1.2. Generate partial charges and assign parameters
- 1.2. Ligand Setup
 - 1.2.1. Source files
 - 1.2.2. Generate partial charges and assign parameters

2. Run Calculations

- 2.1. Generation of Ligand Starting Conformations
 - 2.1.1. Example run
 - 2.1.2. Options available for building conformer generation directories
 - 2.1.3. Options available for running conformer generation
- 2.2. Host-guest calculations
 - 2.2.1. Example run
 - 2.2.2. Options available for building VM2 directories
 - 2.2.3. Options available for running VM2 calculations

3. Results Collection

- 3.1. Generate binding free energy spreadsheets and collect conformer files
- 3.2. Results generation options

XII. VeraChem file formats

- 1. VeraChem's topology and force field parameter (.top) file format example
- 2. Definition of protein real/live atom sets (real_live_set.txt)
- 3. Identify constrained (tethered) atom sets (tethered_atoms.txt)
- 4. Identify atoms to be excluded in conformation search drivers (excluded_atoms.txt)

XIII. References

XIV. Index

I. Introduction

1. VM2 background and theory

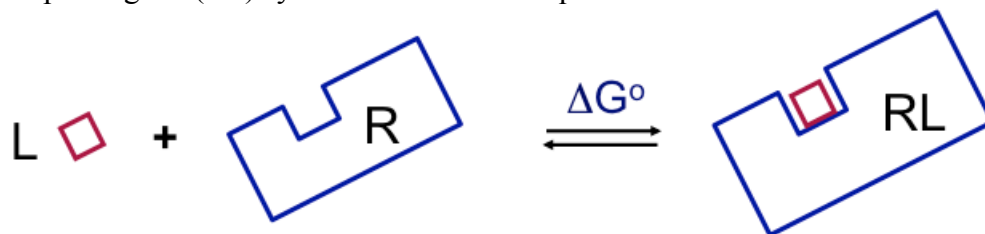
1.1 Purpose of VM2

The main purpose of VM2 is to compute the binding affinities of small molecule ligands to proteins and other types of receptors.(1, 2) VM2 was developed for use by researchers in the pharmaceutical industry to aid in drug development, as well as researchers in academia and government laboratories for the purposes of drug discovery research and fundamental studies of the driving forces for molecular binding.(3) Additional applications of the VM2 technology are in the chemical and agricultural industries, where prediction of molecular binding affinities can also aid in product research and development.

VM2 is grounded in rigorous statistical mechanics theory of binding affinities, (4, 5) and is suitable for application to condensed phase systems, where commonly there are multiple degenerate and/or near-degenerate low energy molecular conformations that have significant weightings in the Boltzmann distribution.

1.2. Binding free energy

For a receptor-ligand (RL) system in solution at equilibrium



the free energy of binding, ΔG° , may be determined via the experimental equilibrium constant K_{bind}

$$K_{\text{bind}} \equiv \left(\frac{C_{RL} C^0}{C_R C_L} \right)_{\text{Equilibrium}}$$

$$\Delta G^\circ = -RT \ln(K_{\text{bind}})$$

where C^0 is the standard concentration (usually 1 molar), and C_{RL} , C_R , and C_L are the receptor-ligand complex concentration, the receptor concentration, and the ligand concentration, respectively, and R and T are the ideal gas constant and temperature, respectively.

Isothermal titration calorimetry experiments can directly measure binding free energies as well as the constituent enthalpy and entropy terms

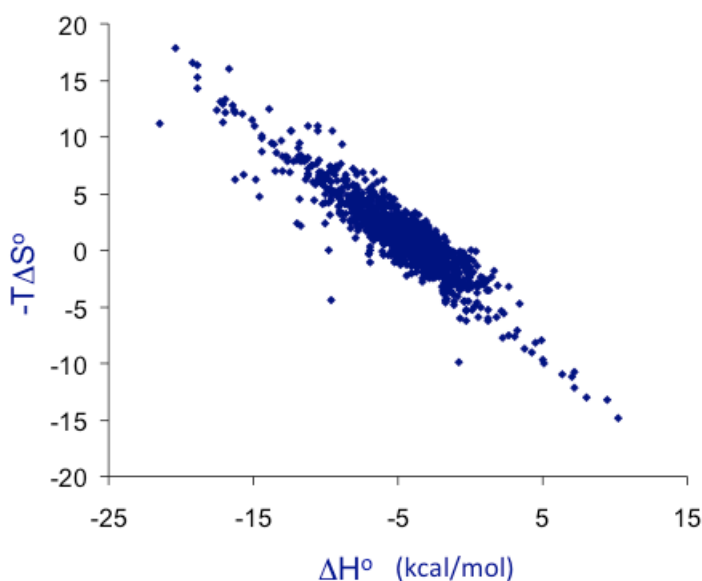
$$\Delta G^\circ = \Delta H^\circ - T\Delta S^\circ$$

1.3. Enthalpy-entropy compensation

Examination of the constituent enthalpy and entropy terms of the free energy of binding can provide useful insight and direction when designing ligands to strongly bind to a given receptor.

One challenge in the design of strongly binding ligands is the so-called enthalpy-entropy compensation effect: when a ligand makes favorable interactions that lower the overall potential energy/enthalpy, the often-resulting increase in rigidity of the ligand (and possibly parts of the receptor) leads to a loss of entropy. The net effect, then, can be little or no increase in the binding affinity of a ligand, even though favorable interactions with respect to potential energy are present.

Enthalpy-entropy compensation has been widely observed both experimentally (6, 7) and in modeling studies that adequately account for entropic effects. The following graph is typical of host-guest data. (8)



An additional key finding is that the compensation is not in general exact, for example the data in the above graph is roughly linear but has a width of 5 Kcal/mol or more in parts. This means that the enthalpy alone will not necessarily indicate the relative strengths of binding of a series of ligands to a receptor. It is, therefore, very often necessary to include entropy effects in computational modeling of receptor-ligand binding affinities to consistently predict correctly the ordering of ligands with respect to their binding affinity strength.

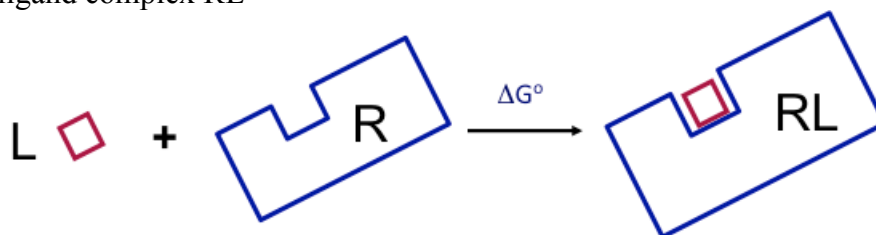
Based on the above findings, to achieve very high binding affinities, one strategy is to push the enthalpy-entropy compensation non-linearity further by designing ligands that fall considerably outside the normal quasi-linear range. (9) These ligands are able to achieve favorable interactions with respect to potential energy lowering, but do not pay a

large entropy penalty for doing so. Computational free energy methods can be a useful tool when pursuing this strategy.

1.4. Computational free energy methods

A common approach for computational prediction of relative strengths of receptor-ligand binding is docking and scoring. While docking and scoring methodology has been found to be effective in terms of enrichment, its use of ad hoc scoring functions means it is unable to reliably rank ligands with respect to their binding affinities;(10) therefore, R&D scientists are looking to more rigorous free energy calculations to provide the accuracy they need in the context of drug development and development of other products dependent on molecular binding properties.

VM2 is a so-called end-point method where receptor-ligand binding free energies are determined from the standard chemical potentials of the receptor R, ligand L, and receptor-ligand complex RL



$$\Delta G^0 = \mu_{RL}^0 - \mu_R^0 - \mu_L^0 \quad \text{where } \mu^0 = \text{Standard Chemical Potential}$$

The standard chemical potential is defined by classical statistical mechanics as an integral over all phase space

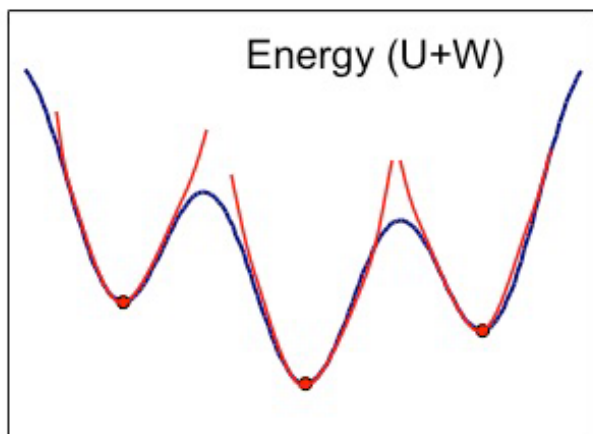
$$\mu^0 = -RT \ln \left(\frac{8\pi^2}{C^0} \int e^{-(U+W)/RT} dr_{\text{int}} \right)$$

where R is the gas constant, C^0 is the standard concentration, and the term $8\pi^2/C^0$ is integrated external translational/rotational degrees of freedom. T is temperature, U is the potential energy, W is solvation energy (U and W are currently molecular mechanics (MM) based), and r_{int} represents internal coordinate phase space.

The integral over all phase space is difficult to compute and is a major reason for the computational expense of the application of rigorous statistical mechanics to binding affinity modeling of condensed phase systems such as protein-ligand complexes.

1.5. Mining minima approximation

The VM2 algorithm approximates μ^0 using the 2nd-generation mining minima approach, (11, 12) which replaces the integral over all phase space shown above with a sum of integrals over local energy minima of the system. The following shows an example three minima system. (Note that actual calculations involve tens to many hundreds of minima.)



$$\mu^0 = -RT \ln \left(\frac{8\pi^2}{C^0} \sum_i^3 Z_i \right)$$

$$Z_i = \int_{\text{well } i} e^{-\beta(U(r)+W(r))} dr_{\text{int}}$$

where Z_i are local configuration integrals. This is a good approximation as long as the dominant low energy minima of the system are found.

1.6. Molecular coordinate systems

The VM2 software uses both Cartesian and internal molecular coordinate systems.

1.6.1 Cartesian coordinates

Standard Cartesian coordinates are used for calculation of force field non-bonded pair energies and gradients as well as solvation pair energies and gradients where required. While the force field bonded term energies are calculated directly by the force field defined bond, angle, and dihedral expressions, the bonded gradient terms although calculated via bond, angle, and dihedral expressions are projected onto the Cartesian coordinates to provide total gradients in terms of Cartesian coordinates.

1.6.2 Anchored Cartesian coordinates

If a Cartesian coordinate frame is chosen so that it rotates and translates with the molecule, in classical statistical mechanics rotational and translational degrees of freedom can be formally separated. (13) For systems where all atoms are mobile this enables calculation of a Hessian matrix with no rotational/translational contaminants.

1.6.3 Bond-Angle-Torsion (BAT) internal coordinates

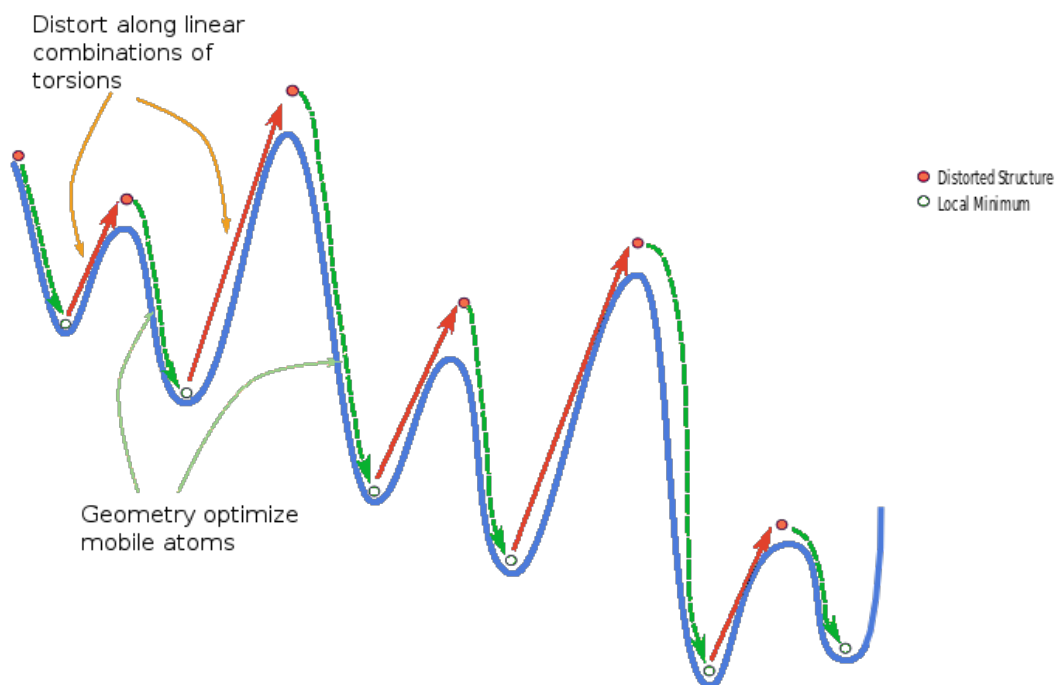
On reading in molecular systems, the VM2 software automatically generates a set of bond-angle-torsion (BAT) internal coordinates. This allows transformation of Cartesian coordinate Hessian matrices into internal coordinates, and subsequent determination of modes that are linear combinations of bonds stretches, angle bends, and torsional

rotations. These modes are useful in VM2's conformational search algorithms (see Section 1.7) as well as its treatment of anharmonic effects via numerical integration along BAT mode distortions (see Section 1.8).

1.7. Conformational searching

The VM2 algorithm requires that the low energy minima/conformers of a molecular system be found. Two basic types of conformational search are available in the VM2 package: a distort-minimization scheme, where distortions along torsional modes are carried out followed by energy minimization of the distorted structure, and a rigid body translation-rotation of ligands in receptor binding pockets. These two types of search may be used separately or in conjunction.

The distort-minimization scheme, an enhanced version of the Tork algorithm (11), starts with calculation of the matrix of the energy 2nd derivatives (Hessian) in Cartesian coordinates, followed by transformation to internal (BAT) coordinates, removal of rows and columns corresponding to bond and angle distortions, followed by diagonalization to produce purely torsional modes comprising linear combinations of dihedrals. The system is then distorted along these torsional modes (or search drivers). The distortion is broken up into steps and between each step some relaxation of atoms not significantly weighted in the driver occurs, while keeping the driver atoms fixed. This allows a larger distortion before encountering steric clashes. On completion of the distortion, the resulting structure is geometry optimized. The basic idea of the technique is to drive structures over energy barriers and with subsequent geometry optimization to find new and lower energy minima.



1.8. Configuration integrals

The local configuration integrals

$$Z_i = \int_{\text{well } i} e^{-\beta(U(r)+W(r))} dr_{\text{int}}$$

are calculated using the Harmonic Approximation with Mode Scanning method (HAMS),(12) when using molecular mechanics (MM) energy potentials. In this method the Hessian (matrix of the energy 2nd derivative) is calculated in the anchored Cartesian coordinate system, transformed to bond-angle-torsion (BAT) internal coordinates, (13) and diagonalized, with the inverse square root of the eigenvalues providing a harmonic approximation to the integral. Anharmonic effects are checked for the low energy modes via numerical integrals (mode scanning). The following equation summarizes the HAMS approach:

$$Z_i \cong b_2^2 \prod_{j=3}^N b_j \sin \theta_j \prod_k^{N_{\text{scan}}} S_k \prod_l^{N_{\text{harm}}} \left[\sqrt{\frac{2\pi kT}{K_l}} \text{erf} \left(\frac{w_l}{\sqrt{\frac{2kT}{K_l}}} \right) \right]$$

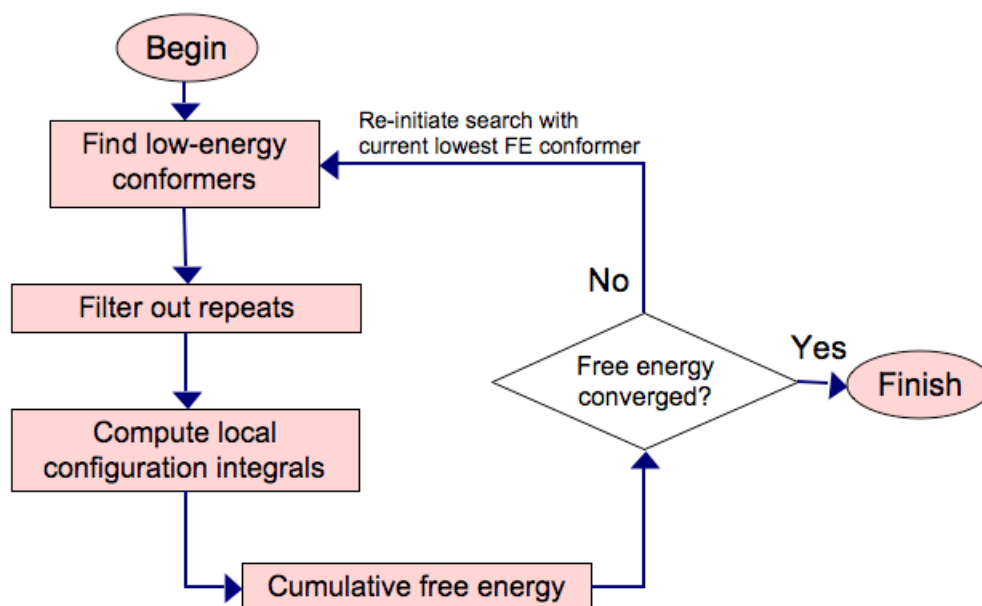
where b and θ are bond and angle internal coordinates (Jacobian determinant contributions), S_k are numerical integrals, K_l are eigenvalues of the internal (BAT) coordinate Hessian matrix, w_l is the integration range of mode l , erf is the error function, T is the temperature, and k is Boltzmann's constant. For a more detailed theory description see reference (12).

1.9. Filtering of conformer repeats

Conformational searches inevitably produce conformer repeats. In order to avoid double counting of conformers in the Boltzmann averaging, the conformers produced during VM2 calculations are checked against each other and any determined to be repeats are discarded. This process requires determination of conformer pair RMSDs that is symmetry aware. (14) For hosts, ligands, and host-ligand complexes the symmetry recognition algorithm involves traversing the molecule from a starting atom and building up a molecule name based upon the names of the atoms encountered along the traversal. Additional molecule names are generated from other starting atoms, and name-name matches are identified as corresponding to symmetry operations. The method detects not only global symmetries but also local symmetries associated with bond rotations as well as symmetries that are only apparent when alternate resonance forms are considered. For protein-involved systems, a simpler check of between conformer amino acid side chains that have rotational symmetries is performed.

1.10. VM2 algorithm

The VM2 algorithm is iterative (see schematic representation below), with the calculated free energy deemed converged when it is no longer changing within a given tolerance.



As the above diagram indicates, the low energy conformer search can be biased toward lowering the free energy by seeding it each iteration with conformers that have low free energy as opposed to low potential energy.

1.11. Molecular system partitioning

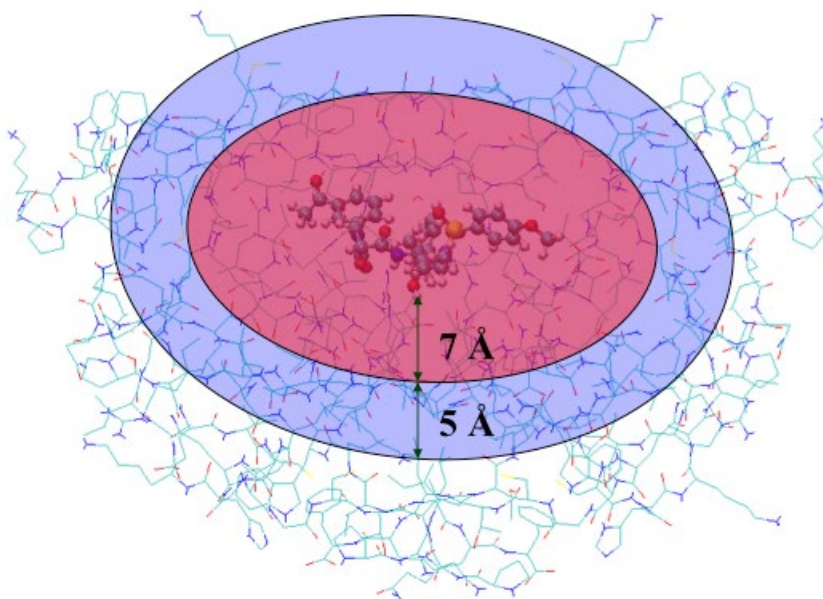
For small systems such as ligand and host molecules (600 atoms or less) all atoms in the system are always defined as 'real' and 'live'. The term 'real' means the atom will be included in any calculation of the energy; the term 'live' means the atom is also treated as mobile in the system, though the user can choose atoms to constrain as they wish.

For larger systems such as protein receptors, to make calculations more tractable the system is partitioned into totally excluded atoms and a 'real' set of atoms. The 'real' set of atoms contains a subset of atoms that are 'live' i.e. mobile, the rest are fixed in space. In other words all live atoms are in the 'real' set, but not all 'real' atoms are in the 'live' set.

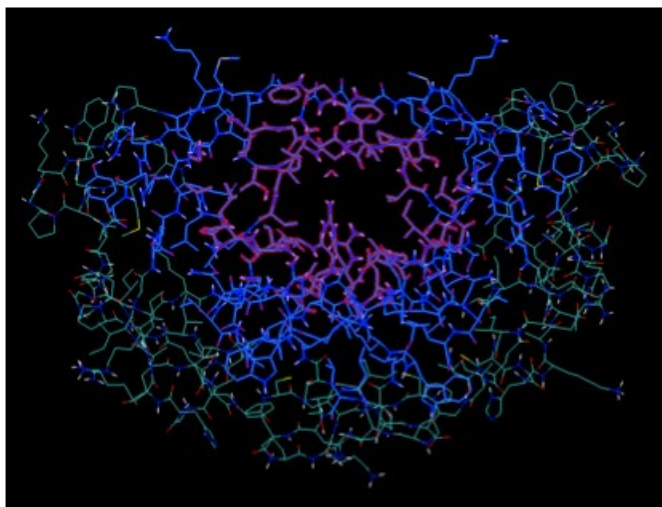
Usually the 'live' set are atoms in the active site of a protein; however, the user can choose as 'live' any part of the protein that they have an interest in exploring structural conformations and associated energetics e.g. specific loops or flaps in the case of 1HVR. For protein-ligand complex calculations, the ligand atoms are treated as 'live' by default, but again the user can choose ligand atoms to constrain.

The following figure shows schematically how a real/live set of atoms may be chosen based on the position of a co-crystallized ligand in a protein active site; in this case the 'real' set distance cutoff is 12 Å and the 'live' set cutoff distance is 7 Å. The 'live' set is usually chosen by atom-based distances from the ligand; whereas, the 'real' set is amino

acid residue based, i.e., if a residue's atom is within the cutoff distance, the whole residue is included in the 'real' set.



This translates to the real/live set shown in the following figure, where purple signifies 'live' atoms and blue signifies 'real' but fixed in space atoms.



2. VM2 package modules

2.1. VM2 : Second-generation mining minima

The VM2 module makes use of all other modules in the package. As such, it initializes and reads user input for all package modules at the start of a run. A completed VM2 calculation provides the free energy of the molecular system, which is a Boltzmann average over the conformers found. Average energy and entropy terms are also output. In

addition, an energy breakdown is provided for all individual conformers found as well as the conformer geometry data in standard formatted files such as sdf, xyz, and mol2.

2.2. Vconf : Conformational search

The Vconf module takes one or multiple starting conformations and attempts to find lower energy conformers. It is used by the VM2 module, for which it provides conformations for subsequent free energy evaluation. When used by the VM2 module it is biased toward lowering the free energy as the VM2 module feeds it starting conformers that have the lowest free energy as opposed to lowest potential energy. It can also be used independently to find low energy conformers based only on potential energy, as well as provide a diverse set of ligand conformations for subsequent placement into a binding site as a starting point for a receptor-ligand calculation.

2.3. VfreeE : Configuration integration

The VfreeE module takes one or more conformations. It is used by the VM2 module, which feeds it previously geometry-optimized conformers. It returns individual conformer free energies and component energies, as well as Boltzmann averages over all conformers found. It can also be used independently, processing conformers read-in by the package, which may have been generated previously by the VM2 package itself or by 3rd-party software.

2.4. Vstereochem : Stereochemistry

The Vstereochem module identifies the stereochemistry of starting conformers as well as conformers generated during the course of calculations. It is used by the VM2, Vconf, and Vgeomopt modules. It cannot currently be run independently.

2.5. Vfilter : Filtering of repeat conformers

The Vfilter module checks for and removes repeat conformers. It will check both energies and RMSDs (see below) between conformers. The VFilter module is used by the VM2, Vconf, and VfreeE modules; it can also be used independently to filter conformers generated previously and read-in.

2.6. Vrmsd : Conformer RMSD

The Vrmsd module determines RMSDs of conformer pairs. It is symmetry aware (14), so correctly determines zero RMSDs when conformer subgroups of atoms are rotated. The Vrmsd module is used by the Vfilter module and can also be used independently to determine RMSDs of groups conformers generated previously and read-in.

2.7. Vhessian : Hessian calculation

The Vhessian module, supplied one or more conformers, calculates the matrix of the energy 2nd-derivatives. This module is used by the VM2, Vconf, and VFreeE modules. The Vhessian module cannot currently be used independently.

2.8. Vgeomopt : Geometry optimization

The Vgeomopt module, supplied one or more conformers, optimizes their geometries, driving down the energy-gradient RMSD below a designated tolerance. Available geometry optimization methods are conjugate gradient and quasi-Newton. The Vgeomopt module is used by the VM2, Vconf, Vfilter, Vrmsd, and Vhessian modules, but can also be used independently for previously generated conformers that are read-in.

2.9. Vpotential : Potential energy and potential energy 1st derivative

The Vpotential module, supplied one or more conformers, calculates the potential energy(s) or potential energy plus gradient(s). The Solvation energy model(s) used depend(s) on user input selections. The Vpotential module is always initialized as it is utilized by all other modules. The Vpotential module can be used independently for previously generated conformers that are read-in.

II. Energy Potentials

In principle, the VM2 method can incorporate any type of molecular energy potential or combination of molecular potentials. The current production versions are, however, limited to standard molecular mechanics potentials.

1. Molecular mechanics potentials

1.1. Force field support

The VM2 package supports the forms of the most commonly used classical molecular mechanics force fields, namely CHARMM (15, 16), OPLS (17), and AMBER (18, 19). The force field parameters are supplied to the VM2 through a VeraChem formatted topology file (see section 1.2.); therefore, the standard published parameter sets for proteins/nucleic acids can be used as is, but can also be modified as desired by simple text file editing. Generalized parameter sets e.g. GAFF, (20) CGenFF, (21) and Dreiding (22) are read in via a text file of the same format.

1.1.1. Basic potential energy form

The basic classical force field potential energy can be written in terms of bonded and nonbonded terms

$$U_{\text{forcefield}} = U_{\text{bonded}} + U_{\text{nonbonded}}$$

where

$$U_{\text{bonded}} = U_{\text{bond}} + U_{\text{angle}} + U_{\text{dihedral}} + U_{\text{improper}}$$

$$U_{\text{nonbonded}} = U_{\text{Coulomb}} + U_{\text{van der Waals}}$$

1.1.2. Bond and angle energy terms

The forms of the bond and angle terms are

$$U_{\text{bond}} = \sum_b^{\text{bonds}} K_b (l - l_0)^2$$

$$U_{\text{angle}} = \sum_a^{\text{angles}} K_a (\theta - \theta_0)^2$$

where K_b and K_a are bond and angle force constants, respectively, l and θ are the current bond length and angle, respectively, and l_0 and θ_0 are the reference bond length and angle, respectively.

1.1.3. Torsion energy terms

The forms of the dihedral and improper dihedral for the CHARMM force field are

$$U_{\text{dihedral}} = \sum_{\varphi}^{\text{dihedrals}} K_{\varphi} (1 + \cos(n\varphi - \delta))$$

$$U_{\text{improper}} = \sum_{\omega}^{\text{impropers}} K_{\omega} (\omega - \omega_0)^2$$

where K_{φ} and K_{ω} are dihedral and improper torsion force constants, respectively, φ is the dihedral angle, n and δ are the dihedral multiplicity and phase, and ω_0 and ω are the reference improper torsion angle and current improper torsion angle, respectively.

The form of both the dihedral and improper dihedral for the AMBER force field takes the same form as the CHARMM dihedral term but with a factor of a half

$$U_{\text{torsion}} = \sum_{\varphi}^{\text{torsions}} \frac{1}{2} K_{\varphi} (1 + \cos(n\varphi - \delta))$$

The OPLS torsion energy, dihedral and improper, appears in the literature as an explicit truncated Fourier series with the coefficients V_1 , V_2 , and V_3 , and phase angles f_1 , f_2 , and f_3 ,

$$U_{\text{torsion}} = \sum_{\varphi}^{\text{torsions}} \left(\frac{V_1}{2} [1 + \cos(\varphi - f_1)] + \frac{V_2}{2} [1 + \cos(2\varphi - f_2)] + \frac{V_3}{2} [1 + \cos(3\varphi - f_3)] \right)$$

1.1.4. Coulomb nonbonded energy terms

The form of the CHARMM non-bonded electronic (Coulomb) term is

$$U_{\text{Coulomb}} = \sum_{ij}^{\text{nonb pairs}} \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}}$$

where i and j are the indexes of a unique non-bonded atom pair at least three bonds apart, q_i is the partial charge for atom i , r_{ij} is the distance between atoms i and j , and ϵ_0 is the permittivity of free space. For AMBER and OPLS interactions involving atoms four bonds apart (so-called 1-4 interactions) are scaled by a factor of a half

$$U_{\text{Coulomb}} = \sum_{ij}^{\text{nonb pairs}} f_{ij} \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}}$$

where $f_{ij} = 0.5$ for 1-4 interactions and $f_{ij} = 1.0$ for 1-N interactions, where $N > 4$. Literature descriptions of the OPLS Coulomb term also include the factor e^2 in the numerator for conversion from elementary charges to energy units. This conversion factor is assumed in CHARMM and AMBER descriptions. In the VM2 package the factor used for conversion from elementary charges to Kcal/mol is 332.054.

1.1.5. Van der Waals nonbonded energy terms

The CHARMM form of the Lennard-Jones (LJ) van der Waals non-bonded energy term is

$$U_{\text{van der Waals}} = \sum_{ij}^{\text{nonb pairs}} \varepsilon_{ij} \left[\left(\frac{r_{ij}^{\text{min}}}{r_{ij}} \right)^{12} - 2 \left(\frac{r_{ij}^{\text{min}}}{r_{ij}} \right)^6 \right]$$

The force field LJ parameters ε_i and r_i^{min} are associated with the well depth and the minimum energy distance, respectively. The well depth and minimum energy distance values used in the van der Waals expression for atom pairs are calculated as the geometric mean $\varepsilon_{ij} = \sqrt{\varepsilon_i \varepsilon_j}$ and the arithmetic mean $r_{ij}^{\text{min}} = (r_i^{\text{min}} + r_j^{\text{min}})/2$. As is the case for the Coulomb term, the AMBER and OPLS van der Waals expression is the same except that it includes a scaling factor $f_{ij} = 0.5$ for 1-4 interactions and $f_{ij} = 1.0$ for 1-N interactions, where $N > 4$.

In practice, and is the case for the VM2 package, for reasons of computational efficiency the van der Waals expression may be re-expressed as follows

$$U_{\text{van der Waals}} = \sum_{ij}^{\text{nonb pairs}} 4\varepsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^6 \right]$$

where

$$\sigma_{ij} = 2^{-1/6} r_{ij}^{\text{min}}$$

1.1.6. CHARMM parameter sets

The CHARMM (15, 16) parameter sets for macromolecular systems as well as the generalized set CGenFF (21) are freely available for download here http://mackerell.umaryland.edu/charmm_ff.shtml.

1.1.7. OPLS parameter sets

Except for the most recent propriety sets, the OPLS parameter sets are available in the literature. (17, 23-27)

1.1.8. AMBER parameter sets

The AMBER parameter sets are provided with the ambertools download (<http://ambermd.org>).

1.1.9. Dreiding parameter set

The Dreiding parameter set is published in the literature. (22)

1.2. The VeraChem topology and force field parameter file (.top)

The force field parameters discussed above are read in at the start of each VM2 calculation from a formatted text file with the extension **.top**. The format of this file and the correspondence with the force field parameters identified above are now summarized. A specific example of a **.top** file can be found in **Section XII** of this manual.

Each new section in the **.top** file starts with an exclamation point plus keyword.

1.2.1. Identification of force field

The first line in the VeraChem parameter file identifies the force field to be used. For example

```
!NTITLE 3
```

indicates that the AMBER force field will be used. The available values 1, 2, 3, and 4 correspond to CHARMM, Dreiding, AMBER, and OPLS, respectively.

1.2.2. Atom types, charges, and van der Waals parameters

The next section starts with a line with the atom keyword and the total atom count and subsequent lines contain six columns of data e.g.

```
!NATOM 3137
1 N3      14.01000   -0.20200   -0.17000   1.82400
2 H       1.00800    0.31200   -0.01570   0.60000
3 H       1.00800    0.31200   -0.01570   0.60000
4 CT      12.01000   -0.01200   -0.10940   1.90800
```

The column numbers and corresponding data are:

- 1: atom number
- 2: force field atom type
- 3: atomic mass
- 4: partial atomic charge
- 5: well depth parameter ϵ_i
- 6: minimum energy distance parameter r_i^{min}

Note that for CHARMM parameters an additional two columns can be present containing ϵ_i and r_i^{min} specifically for 1-4 atom pairs.

1.2.3. Bonds and bond force constants

The bond section first line contains the bond keyword and bond count; subsequent lines contain bond data in six columns e.g.

```
!NBOND: 3164
1      4      367.000      1.47100 N3      CT
4      7      310.000      1.52600 CT      CT
7     10      310.000      1.52600 CT      CT
1     13      367.000      1.47100 N3      CT
```

The column numbers and corresponding data are:

- 1: atom number i
- 2: atom number j
- 3: force constant K_b associated with bond between atom numbers i and j
- 4: the reference (equilibrium) bond length l_0 between atom numbers i and j
- 5: force field atom type of atom number i
- 6: force field atom type of atom number j

1.2.4. Angles and angle force constants

The angle section first line contains the angle keyword and angle count; subsequent lines contain angle data in eight columns e.g.

```
!NTHETA: 5795
2      1      3      35.000      1.911136 H      N3      H
2      1      4      50.000      1.911136 H      N3      CT
3      1      4      50.000      1.911136 H      N3      CT
2      1     13      50.000      1.911136 H      N3      CT
```

The column numbers and corresponding data are:

- 1: atom number i
- 2: atom number j
- 3: atom number k
- 4: force constant K_a associated with angle between atom numbers i, j , and k
- 5: the reference (equilibrium) angle θ_0 , in radians, between atom numbers i, j , and k
- 6: force field atom type of atom number i
- 7: force field atom type of atom number j
- 8: force field atom type of atom number k

1.2.5. Proper dihedrals and corresponding parameters

The proper dihedral section first line contains the proper dihedral keyword and unique proper dihedral count; subsequent lines contain dihedral data in twelve columns e.g.

```
!NPDI: 8474
16     15     17     18      2.0000      1.0000      0.0000 1 O      C      N      H
16     15     17     18      2.5000      2.0000      3.1416 1 O      C      N      H
16     15     17     19      2.5000      2.0000      3.1416 1 O      C      N      CT
13     15     17     18      2.5000      2.0000      3.1416 1 CT     C      N      H
13     15     17     19      2.5000      2.0000      3.1416 1 CT     C      N      CT
```

14	13	15	16	0.0000	2.0000	0.0000	1	HP	CT	C	O
14	13	15	17	0.0000	2.0000	0.0000	1	HP	CT	C	N

The column numbers and corresponding data are:

- 1: atom number i
- 2: atom number j
- 3: atom number k
- 4: atom number l
- 5: force constant K_ϕ associated with torsion between atom numbers i, j, k , and l
- 6: the multiplicity n associated with torsion between atom numbers i, j, k , and l
- 7: the phase δ for the current term
- 8: integer setting force constant sign
- 9: force field atom type of atom number i
- 10: force field atom type of atom number j
- 11: force field atom type of atom number k
- 12: force field atom type of atom number l

1.2.6. Improper dihedrals and corresponding force constants

The improper dihedral section first line contains the improper dihedral keyword and improper dihedral count; subsequent lines contain improper dihedral data in eleven columns e.g.

```

!NIMPDI: 550
13    17    15    16    10.50    3.14    2.0 CT    N    C    O
15    19    17    18     1.10    3.14    2.0 C     CT    N    H
24    29    27    28    10.50    3.14    2.0 CT    N    C    O
27    30    29    31     1.00    3.14    2.0 C     H    N    H
19    34    32    33    10.50    3.14    2.0 CT    N    C    O

```

The column numbers and corresponding data are:

- 1: atom number i
- 2: atom number j
- 3: atom number k
- 4: atom number l
- 5: force constant K_ϕ associated with torsion between atom numbers i, j, k , and l
- 6: the phase δ for the current term
- 7: the multiplicity n associated with torsion between atom numbers i, j, k , and l
- 8: force field atom type of atom number i
- 9: force field atom type of atom number j
- 10: force field atom type of atom number k
- 11: force field atom type of atom number l

1.2.7. Nonbonded ‘fixes’

This section is only relevant for the CHARMM force field and allows van der Waals parameters for specific pairs of atom types to be modified e.g.

```

!NBFIX: 1
H O -0.30 1.50 -0.15 1.50

```

(Note: the VM2 code currently expects ϵ_{ij} and r_{ij}^{min} , not the uncombined parameters ϵ_i , r_i^{min} , ϵ_j , and r_j^{min} , as is the case for the CHARMM standard parameter file)

1.2.8. Final count of atoms, bonds, angles, proper and improper dihedrals

This section provides the atom, bond, angle, proper dihedral, and improper dihedral counts. Note that here the proper dihedral count includes all dihedrals in possible series expansions. The last column containing 9999 is not used.

```
!NFINAL: 6
3137      3164      5795      11874      550 9999
```

1.2.9. !NDON

This section if present will be ignored.

1.3. Solvation models

A solvation treatment is essential for any quantitative prediction of condensed phase binding affinities. The VM2 package supports continuum methods for modeling of bulk solvation effects. Solvent molecules that play a direct role in ligand binding can be explicitly included if desired.

1.3.1. Generalized Born (GB)

The Generalized Born solvation model (28) is available for energy and energy 1st and 2nd derivative calculations. The GB electrostatic polarization solvation energy is given by

$$W_{\text{el}}^{\text{GB}} = -166.027 \left(\frac{1}{\epsilon_{\text{int}}} - \frac{1}{\epsilon_{\text{ext}}} \right) \sum_{i=1}^n \sum_{j=1}^n \frac{q_i q_j}{(r_{ij}^2 + \alpha_{ij}^2 e^{-D_{ij}})^{0.5}}$$

$$\text{where } \alpha_{ij} = (\alpha_i \alpha_j)^{0.5} \quad \text{and} \quad D_{ij} = r_{ij}^2 / (2\alpha_{ij})^2$$

The indices i and j run over atoms, ϵ_{int} is the dielectric constant of the internal (solute) medium (1.0 for gas phase), ϵ_{ext} is the dielectric constant of the external medium (80.0 for water), q are the partial atomic charges, r_{ij} is the distance between atoms i and j , and α_i is the Born radius of atom i (defined below). The total potential energy with GB solvation energy is then given by addition to the force field potential energy

$$E_{\text{total}}^{\text{GB}} = U_{\text{forcefield}} + W_{\text{el}}^{\text{GB}}$$

Currently only the Still method (29) for calculation of the required Born radii is supported. In this approach approximate Born radii are calculated analytically by the expression

$$\alpha_j = -(-166.027/W_{el,j}^{GB'})$$

where

$$W_{el,j}^{GB'} = \frac{W_{el,j}^{GB}}{1 - \frac{1}{\epsilon}} = \frac{-166.027}{R_{vdW-i} + \phi + P_1} + \sum_i^{\text{stretch}} \frac{P_2 V_j}{r_{ij}^4} + \sum_i^{\text{bend}} \frac{P_3 V_j}{r_{ij}^4} + \sum_i^{\text{nonbonded}} \frac{P_4 V_j \text{CCF}}{r_{ij}^4}$$

and $W_{el,j}^{GB}$ is the polarization energy of atom j in Kcal/mol, ϕ is a dielectric offset, V_j is the volume of atom j , R_{vdW-i} is the van der Waals radius (or solvation cavity radius) of atom i , P_1 is the single atom scale factor, P_2 is the bonded 1,2 atom pair scale factor, P_3 is the angle 1,3 atom pair scale factor, and P_4 is the nonbonded 1, \geq 4. CCF is the close contact function for nonbonded 1, \geq 4 interactions and is given by

$$\text{CCF} = 1.0 \quad \text{if} \quad \left(\frac{r_{ij}}{R_{vdW-i} - R_{vdW-j}} \right)^2 > \frac{1}{P_5}$$

else

$$\text{CCF} = \left\{ 0.5 \left[1.0 - \cos \left\{ \left(\frac{r_{ij}}{R_{vdW-i} - R_{vdW-j}} \right)^2 P_5 \pi \right\} \right] \right\}^2$$

The default solvation cavity radii used in VM2 is $r_{ij}^{min}/2$, with the exceptions of hydrogen atoms bonded to a hetero atom and covalently bound F atoms, which are set to 1.15 and 2.00 Angstroms, respectively. The user can choose from a range of alternative solvation cavity radii including $\sigma/2$, bondi,(30) and mbondi. (31)

The default scaling factors P_1 - P_5 are those from the original fit in reference (29). An alternative set of scaling factors, which were fit for a single protein-ligand system (HIV-1 protease with inhibitor KNI-272) are also available. (32)

1.3.2. Constant dielectric (CD)

A constant dielectric solvation model is available for energies, and energy 1st and 2nd derivatives. This very basic damping solvation model divides the Coulomb energy term by a dielectric constant. The default is the dielectric of water $\epsilon = 80.0$.

$$U_{\text{CD-Coulomb}} = \sum_{ij}^{\text{nonb pairs}} \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}} \cdot \frac{1}{\epsilon}$$

The VM2 package calculates a “solvation energy” due to this model by simply backing out the vacuum Coulomb energy i.e.

$$W^{CD} = \sum_{ij}^{\text{nonb pairs}} \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}} \cdot \left(\frac{1}{\epsilon} - 1\right)$$

The total potential energy with CD solvation can then be written as

$$E_{\text{total}}^{CD} = U_{\text{forcefield}} + W^{CD}$$

1.3.3. Distance dependent dielectric (DD)

A distance dependent dielectric solvation model is available for energies, and energy 1st and 2nd derivatives. This is also a very basic model that varies the dielectric with distance between atom pairs. The default is to divide the Coulomb energy term by $4r_{ij}$

$$U_{\text{CD-Coulomb}} = \sum_{ij}^{\text{nonb pairs}} \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}} \cdot \frac{1}{4r_{ij}}$$

Again, the VM2 package calculates a “solvation energy” by simply backing out the vacuum Coulomb energy i.e.

$$W^{DD} = \sum_{ij}^{\text{nonb pairs}} \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}} \cdot \left(\frac{1}{4r_{ij}} - 1\right)$$

The total potential energy with DD solvation can then be written as

$$E_{\text{total}}^{DD} = U_{\text{forcefield}} + W^{DD}$$

1.3.4. Poisson Boltzmann Surface Area (PBSA)

The Poisson Boltzmann Surface Area solvation model is available for energy calculations. As in the GB case, the PBSA potential energy (or potential of mean force (PMF), given it is an average over solvent degrees of freedom for a particular solute conformation \mathbf{r}_p) is added to the force field potential energy to give a total potential energy

$$E(\mathbf{r}_p) = U_{\text{forcefield}} + W(\mathbf{r}_p)^{\text{el}} + W(\mathbf{r}_p)^{\text{np}}$$

where $W(\mathbf{r}_p)^{\text{el}}$ is the electrostatic solvent polarization term and $W(\mathbf{r}_p)^{\text{np}}$ is the non-polar term which is proportional to the accessible surface area of the solute. The electrostatic term is calculated via the solvent induced electrostatic potential ϕ_j at each solute atom j , and its partial atomic charge q_j

$$W(\mathbf{r}_p)^{\text{el}} = \frac{1}{2} \sum_j q_j \phi_j$$

The electrostatic potential ϕ_j at atomic charge q_j is determined by solving the Poisson-Boltzmann equation

$$\nabla \epsilon(\mathbf{r}) \nabla \phi(\mathbf{r}) = 4\pi \rho(\mathbf{r}) - 4\pi \sum_i z_i c_i e^{[-z_i \phi(\mathbf{r})/k_B T]}$$

where $\epsilon(\mathbf{r})$ is the dielectric constant, $\phi(\mathbf{r})$ is the electrostatic potential, $\rho(\mathbf{r})$ is the solute charge, z_i is the charge of ion type i , k_B is the Boltzmann constant, T is the temperature, c_i is the bulk density of ion type i far from the solute.

The PBSA implementation in the VM2 package solves the linearized PB equation through finite differencing on a grid. (33-35) A nonlinear solver is not currently available.

The accessible surface area of the solute used for calculating the non-polar energy term is determined

Boundary condition calculation, modified incomplete Cholesky conjugate gradient (ICCG) solver, and energy-from-dielectric-boundary calculation

2. Quantum mechanics

Quantum mechanics potentials are not supported in this version of the VM2 package.

III. Installation

1. Obtaining the VM2 package, and package choices

1.1. Commercial licensing

To obtain the VM2 package for commercial use contact sales@verachem.com.

Commercial licensing available includes one and two year licenses as well as a perpetual license. Multi-site licenses are available.

1.2 Trial license

To obtain a trial license for the VM2 package contact sales@verachem.com

Free three-month licenses are available for users to trial the fully functional parallel processor enabled VM2 package.

1.3 Academic licensing

To obtain the VM2 package for academic use contact info@verachem.com

Provide your name, position, and institution, and outline in general terms your intended use of the software.

1.4 Package choices

A number of choices are available, which range in capability from ligand only calculations in serial processor mode to protein-ligand binding affinity calculations run in parallel processor modes. The following table shows the various packages available and their capabilities:

VM2 Package	Parallelization	Maximum atom count	
		Real atoms	Live atoms
Ligand only	Serial, MPI	-	200
Host+ligand	Serial, MPI	-	600
Protein+ligand	Serial, MPI, MPI+OpenMP, MPI+CUDA, MPI+OpenMP+CUDA	10000	3000
Full suite	Serial, MPI, MPI+OpenMP, MPI+CUDA, MPI+OpenMP+CUDA	10000	3000
Full suite - large	Serial, MPI, MPI+OpenMP, MPI+CUDA, MPI+OpenMP+CUDA	10000	5000

2. Operating systems and hardware

The VM2 package currently runs on Linux desktops, workstations, and clusters. It can also take advantage of GPU acceleration.

2.1. Linux workstations

The serial, MPI, and MPI-OpenMP VM2 packages can be installed any workstation with Intel CPU(s) and two gigabytes of RAM per compute core or more available, which is running Linux kernel 2.6.32 or later e.g. CentOS 6.9+, Ubuntu 14.04+, etc. It is recommended that a minimum of 8 CPU cores is available for computation.

2.2. Linux desktops

These VM2 packages can also run on commodity desktop Intel PCs running Linux kernel 2.6.32 or later that have adequate memory, though recommended use would be for smaller calculations (ligand, hosts, host-ligand complexes), with dedicated workstations more suitable for the more computationally demanding protein and protein-ligand complex calculations.

2.3. Linux clusters

The MPI and MPI-OpenMP VM2 packages can run across clusters of workstations (or clusters of commodity machines in the case of Beowulf clusters). Given that the MPI parallelization schemes are not communication bound slower Ethernet interconnects are adequate, though parallel MPI also works with the faster InfiniBand interconnects if present.

2.4. Linux workstations and clusters with NVIDIA GPU acceleration

The MPI-CUDA and MPI-OpenMP-CUDA VM2 packages can take advantage of NVIDIA GPUs (Fermi and Kepler architectures) for acceleration of parts of its algorithm. This includes use of multi-GPU workstations and clusters of workstations each with multiple GPUs.

2.5. OSX

VM2 is not currently available for OSX.

2.6. MS Windows

VM2 is not currently available for MS Windows.

3. Installation procedure

3.1. Download the VM2 package

After downloading the VM2 package and the example set

```
vcCompChem_<version>.tar.bz2
```

```
vcCompChem_<version>_examples.tar.bz2
```

where version is the major, minor, and sub-minor version numbers. (e.g. 2_7_050),
uncompress and untar them in location of your choice, e.g.

```
tar xvf vcCompChem_<version>.tar.bz2
```

```
tar xvf vcCompChem_<version>_examples.tar.bz2
```

will create the directories

```
vcCompChem_<version>/
```

```
vcCompChem_<version>_examples/
```

in the directory you are currently in.

3.2. License file

Copy your license file, named vm2_license.LIC, into the vcCompChem_<version>/exe directory.

3.3. Environment variables for installation

These installation instructions assume the bash shell is being used. Place the following shell commands and environment variable settings in your **.bashrc** file, which should then be sourced prior to running the installation script. You may use another default shell as you wish, as long as the equivalent command/same environment variables are set.

Modify the variable **VCHOME** to reflect the location of the directory resulting from the tar file extraction above.

```
-----  
ulimit -s unlimited  
export VCHOME=/home/<my_user_name>/vcCompChem_<version>  
export VM2HOME=$VCHOME  
export VCPYTHON=$VCHOME/exe/vc_python  
export VM2PYTHON=$VCPYTHON  
-----
```

3.4 Requirements for installation

It may be necessary, depending on the Linux flavor being used, to install packages such as tsh and g77.

zlib-devel.x86_64 might be required to compile python and gcc-c++.x86_64 for the extensions. In most cases these packages will already be installed on the system.

To check for already installed libraries:

CentOS, RHEL:

```
yum list zlib-devel
yum list gcc
yum list g++
```

Debian, Ubuntu:

```
dpkg -l zlib-devel
dpkg -l gcc
```

3.5 Installation script

The following sequence of commands should complete the installation.

```
cd vcCompChem_<version>
cd build
./install_vcCompChem.sh
```

The installation will take several minutes. At the conclusion of the installation steps an automated test set will run, which may also take several minutes to complete. If any of the automated tests fail, relevant information will be found in the log files they generate in the vcCompChem_<version>/tests directory. One common issue is that the VCHOME and/or VCPYTHON environment variable(s) are not set or set incorrectly. Check this by typing:

```
echo $VCHOME

echo $VCPYTHON
```

Please contact VeraChem for support at support@verachem.com if you have difficulties with installation.

4. Installed VM2 package structure

The installed VM2 package directories of interest are:

```
$VCHOME/documentation
$VCHOME/exe
$VCHOME/lib
$VCHOME/tests
```

The documentation directory contains a PDF of the package manual and a text file containing the installation directions. The exe directory contains helper software tools and the VM2 executables themselves.

4.1. Helper tools

A set of helper command line software tools are present in the \$VCHOME/exe directory. Currently, the most useful of these are:

4.1.1. VCharge : assignment of partial atomic charges

[VCharge](#) provides fast, easy access to accurate partial charges for virtually any drug-like compound. As input it requires an sdf/mol file. In addition to the Linux command line version supplied with this package, a [GUI version](#) is available.

4.1.2. VConf : 2D to 3D and small molecule conformational search

[VConf](#) is a standalone conformational search application, which processes an SD file of drug-like compounds containing an initial 2D or 3D conformation of each molecule. In addition to the Linux command line version supplied with this package, a [GUI version](#) is available.

4.1.3. prm2top : AMBER formatted input data files to VM2 input data files

This tool given AMBER format .prmtop and .inpcrd files, outputs VM2 input data files – see [Section V. 5.1.](#)

4.1.4. psf2top : CHARMM formatted input data files to VM2 input data files

This tool given a CHARMM format .psf file and .sdf/.mol file, outputs VM2 input data files – see [Section V. 5.2.](#)

4.1.5. mmo2top : Schrodinger mmo file to VM2 input data files

This tool given a Schrodinger .mmo file, output VM2 data files – see [Section V. 5.3.](#)

4.2. VM2 executables

The VM2 executables present in the \$VCHOME/exe directory depends on the licensing level – see Section 1.4 above.

Ligand only:	VC_CompChemPackage_serial.exe VC_CompChemPackage_mpi.exe
Host+ligand:	VC_CompChemPackage_serial.exe VC_CompChemPackage_mpi.exe
Protein+ligand:	VC_CompChemPackage_serial.exe VC_CompChemPackage_mpi.exe VC_CompChemPackage_mpi_openmp.exe VC_CompChemPackage_mpi_openmp_cuda.exe

Full suite: VC_CompChemPackage_serial.exe
 VC_CompChemPackage_mpi.exe
 VC_CompChemPackage_mpi_openmp.exe
 VC_CompChemPackage_mpi_openmp_cuda.exe

Full suite - large: VC_CompChemPackage_serial.exe
 VC_CompChemPackage_mpi.exe
 VC_CompChemPackage_mpi_openmp.exe
 VC_CompChemPackage_mpi_openmp_cuda.exe

4.3. Supplied libraries

The following run time libraries are supplied in \$VCHOME/lib :

/intel Required Intel math, linear algebra, and parallel processing
 libraries (MPI, OpenMP.)

/cuda Required Nvidia CUDA libraries for running on GPUs.

/magma Required linear algebra libraries for running on GPUs.

5. Environment variables for running validation calculations

The following environment variables must be set before running a calculation. They can either be set in the user's **.bashrc** or, preferably, within a script used to launch the calculation. Note that the actual values of OMP_NUM_THREADS and MKL_NUM_THREADS will depend on the type of parallel run being requested. See **Sections VI 4.** and **VI 5.** below for examples of different runs and alternatives to **bash** shell scripts e.g C-shell, PBS, SLURM.

```
-----  
ulimit -s unlimited  
  
INTEL_LIBS=$VCHOME/lib/intel  
INTEL_MKL_LIBS=$INTEL_LIBS/mkl  
INTEL_MPI_LIBS=$INTEL_LIBS/mpi  
  
CUDA_LIBS=$VCHOME/lib/cuda:$VCHOME/lib/magma  
  
LD_LIBRARY_PATH=$INTEL_LIBS:$INTEL_MKL_LIBS:$INTEL_MPI_LIBS:$CUDA_LIBS  
export LD_LIBRARY_PATH  
  
PATH=$INTEL_MPI_LIBS:$PATH  
export PATH  
  
export OMP_NUM_THREADS=1  
export MKL_NUM_THREADS=1  
export I_MPI_PIN_DOMAIN=omp  
export KMP_STACKSIZE=16m  
-----
```

Since other software besides VM2 may depend on existing MPI and CUDA configurations, care should be taken when setting the variables to ensure that they only affect the environment in which VM2 software is being run.

6. Validation tests

A set of validation tests is available in the `vcCompChem_<version>/tests` directory. A subset of these tests is run automatically after installation, as alluded to above. These tests are a basic confirmation of installation. It is recommended that the user run all the tests appropriate to their intended use of the package (set up pathway type and hardware configurations) to confirm correct installation.

A python script that automates the full set of tests is provided:

```
run_all_tests.py
```

This script will run the entire verachem test suite and validate the results. The command line argument `-py` will only run python helper tools tests, `-vm2` will only run vm2 tests, `-c` will add cluster tests, and `-g` will add gpu tests. The default with no arguments will run the python helper tools tests, then vm2 tests, but no cluster or gpu tests.

To run the subset of tests run automatically after installation use:

```
run_install_tests.py
```

6.1. Helper tools validation tests

The supplied helper tool validation tests check that the file format conversions for the Maestro/Macromodel, AmberTools, and Biovia Discovery Studio set up pathways (see [Section V.5.](#)) are functioning correctly.

6.1.1. Maestro/Macromodel pathway

This test is run automatically after installation. To run manually, carry out the following commands, monitor for error messages, and examine log.out for differences with reference values:

```
cd vcCompChem_<version>/tests/mmo2top/ligand_08

./run.sh

./verify.sh
```

6.1.2. AmberTools pathway

This test is run automatically after installation. To run manually, carry out the following commands, monitor for error messages, and examine log.out for differences with reference values:

```
cd vcCompChem_<version>/tests/prm2top/1ke5/ligand
```

```
./run.sh
```

```
./verify.sh
```

```
cd vcCompChem_<version>/tests/prm2top/1ke5/protein
```

```
./run.sh
```

```
./verify.sh
```

6.1.3. Biovia Discovery Studio Visualizer (DSV) pathway

To run this test the variable VCDSPATH must be set to the location of the CHARMM forcefield files from your installation of Discovery Studio Visualizer. (For the 2016 version on the PC this is DiscoveryStudio_2016/share/forcefield/CHARMM.)

Carry out the following commands, monitor for error messages, and examine log.out for differences with reference values:

```
cd vcCompChem_<version>/tests/psf2top/1ke5/ligand
```

```
./run.sh
```

```
./verify.sh
```

```
cd vcCompChem_<version>/tests/prm2top/ psf2top/1ke5/protein
```

```
./run.sh
```

```
./verify.sh
```

6.1.4. VCharge validation test

This test is run automatically after installation. To run manually, carry out the following commands, monitor for error messages, and examine log.out for differences with reference values:

```
cd vcCompChem_<version>/tests/vcharge
```

```
./run.sh
```

```
./verify.sh
```

6.1.5. VConf validation test

This test is run automatically after installation. To run manually, carry out the following commands, monitor for error messages, and examine log.out for differences with reference values:

```
cd vcCompChem_<version>/tests/vconf
```

```
./run.sh
```

```
./verify.sh
```

6.2. VM2 validation tests

The tests for vm2 are located in vcCompChem_<version>/tests/vm2 . The test mpi_4 is run automatically after installation.

```
mpi_16/  
mpi_4/  
mpi_8/  
mpi_cuda/  
mpi_openmp_8_2/  
mpi_openmp_8_4/  
mpi_openmp_cuda/
```

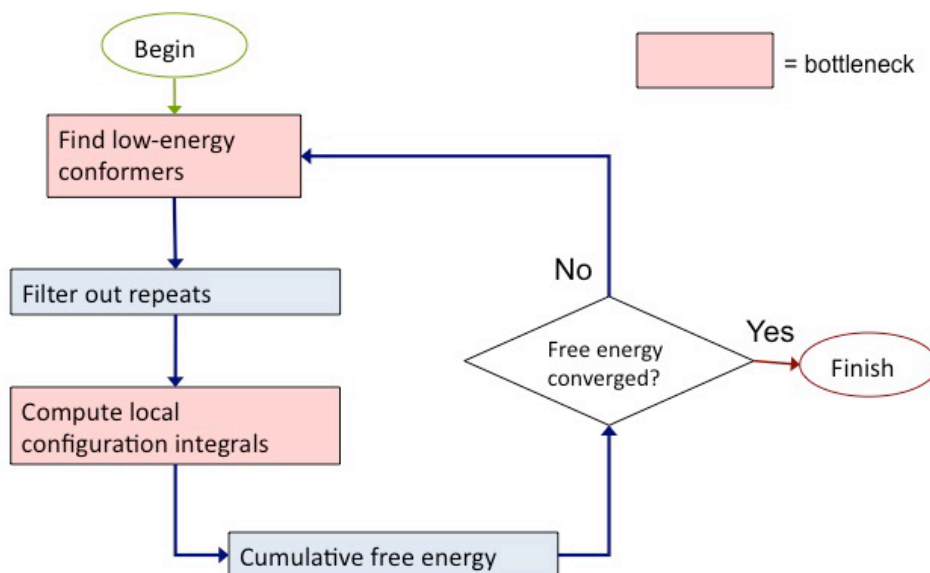
Each test is named for a different configuration of mpi, openmp, and cuda. Most tests include scripts for use with PBS / Torque and when running interactively. The PBS scripts will need to be modified to match your computing environment, queue names, run time limits, etc.

Example output is provided in the reference subdirectory of each test. If you open either .out file, the time required for the test on our hardware will be found at the bottom of the file.

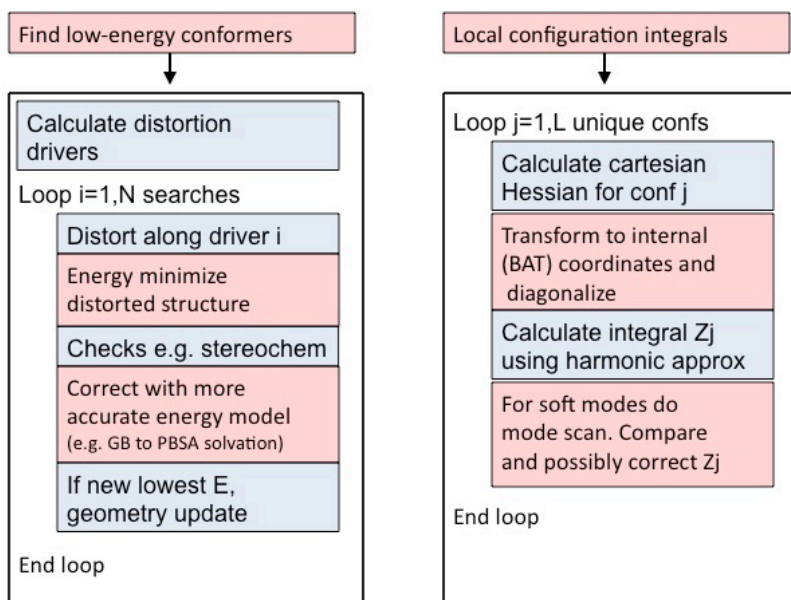
IV. Parallel processing

1. VM2 serial calculation bottlenecks

As described in Section I, a typical VM2 calculation is iterative and during each iteration a search for the system's low energy minima occurs (conformational search) followed by computation of local configuration integrals for this new batch of minima (after filtering out repeats). The cumulative free energy is then calculated and if it is no longer changing within a given tolerance the calculation is deemed converged. Timings for typical VM2 iterations show that the conformational search and the computation of configuration integrals are bottlenecks (see red blocks in following diagram) and so are targeted for parallelization by distribution across multiple computer processors.



These bottleneck steps can be further broken down into constituent tasks:



2. Parallelization strategy

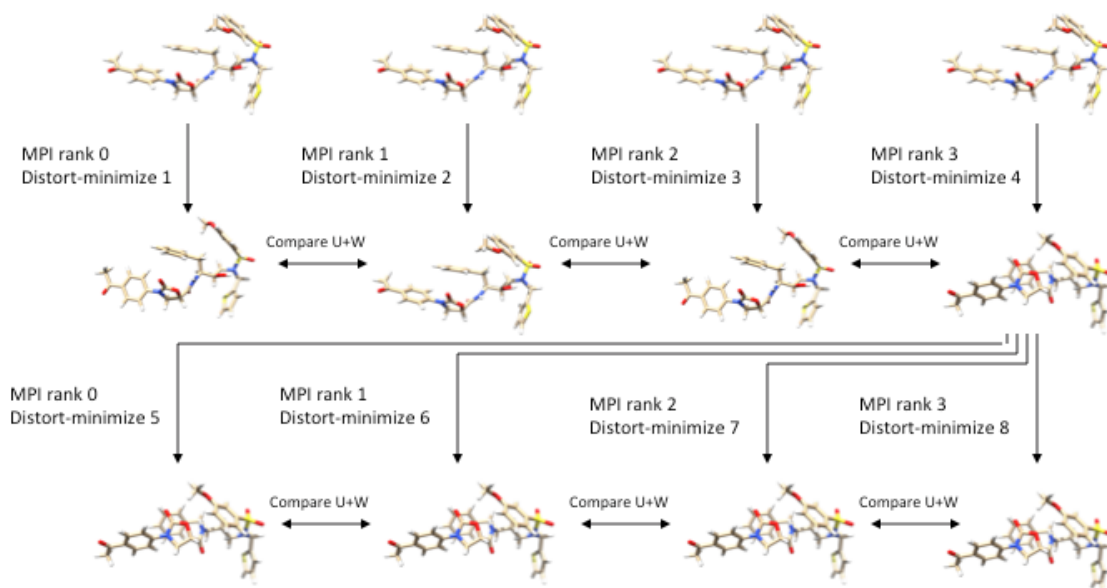
VM2 uses two parallelization approaches to address the identified bottlenecks above. The first, a coarse-grained parallelization, distributes the loops “ $i=1,N$ searches” and “ $j=1,L$ unique confs”, across Message Passing Interface (MPI) processes.⁽³⁶⁾ The MPI distributed tasks are each relatively large and require replicated memory. The second, a fine-grained parallelization approach, distributes smaller constituent tasks such as calculation of an energy-gradient (used in energy minimization of structures) or Hessian transformation and diagonalization (required for configuration integrals), across compute cores. This finer grained distribution across cores is most efficient with a shared memory approach, such as OpenMP.⁽³⁷⁾

We have pursued both coarse and fine-grained approaches, as each can be applied independently or combined to take best advantage of available hardware with respect to system size. For example, for systems where more than ~ 1000 atoms are allowed to be mobile, a purely coarse-grained replicated memory approach can lead to oversubscription to resources such as main memory and cache. If the two approaches are combined, however, the larger coarse-grained tasks are themselves parallelized using shared memory, thereby using less resources and using them more cooperatively.

3. MPI coarse-grained parallelization

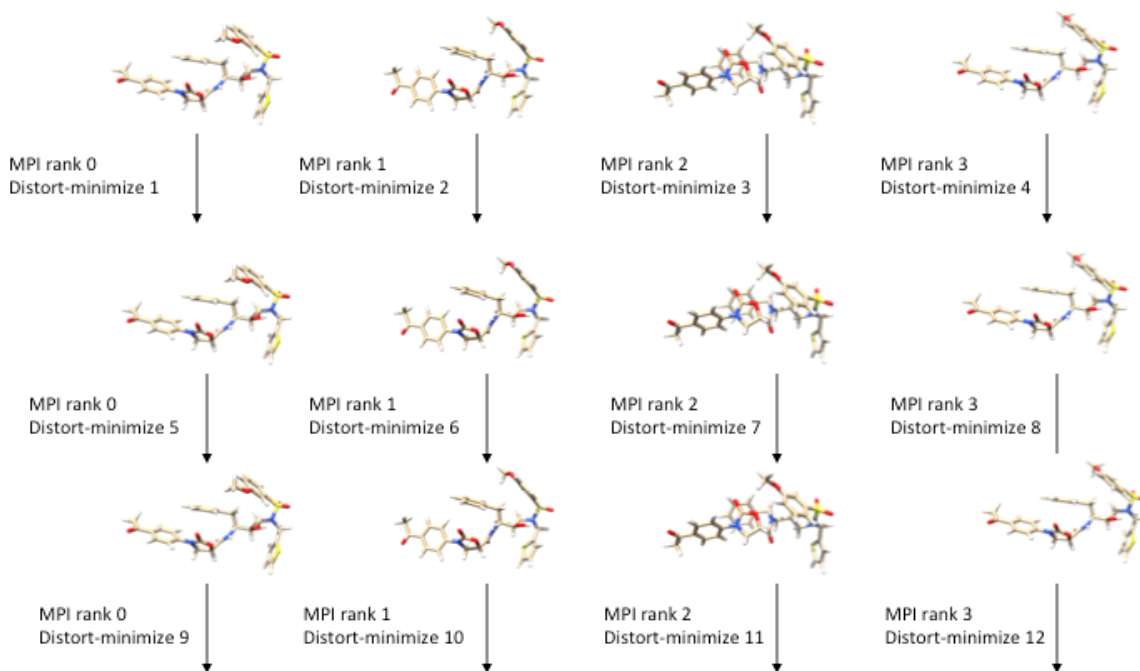
3.1. Coupled MPI conformational search

The “coupled” MPI implementation of VM2’s conformational search, in which processes continually compare the energies they have found, can exhibit super-linear speed up with respect to conformer throughput with the number of MPI processes. This is because the algorithm can sometimes, in effect, look ahead compared to the serial algorithm as a particular MPI process may happen to find a considerably lower energy conformer, and other MPI processes can then switch to searching from that structure.



3.2. Uncoupled MPI conformational search: introduction of diversity

The coupled MPI conformational search procedure always communicates amongst the MPI processes which process had found the lowest energy conformer so all processes can then use it as a basis for the ongoing search. It was found that occasionally this approach led the search to stall before lower energy conformers were found. This problem was addressed by injecting conformer diversity through periodically assigning structurally different conformers to each MPI process and allowing them to be searched on independently. This is referred to as an “uncoupled” MPI conformational search in subsequent sections of this documentation.



This combination of all processes working on the same conformers (coupled) and then periodically working on structurally diverse conformers independently (uncoupled) has been found to be very effective at finding low energy conformers. Various ways of assigning sets of conformers for coupled and uncoupled searches to the available MPI processes at the start of each VM2 conformational search step have been implemented .(See Section VIII 7.)

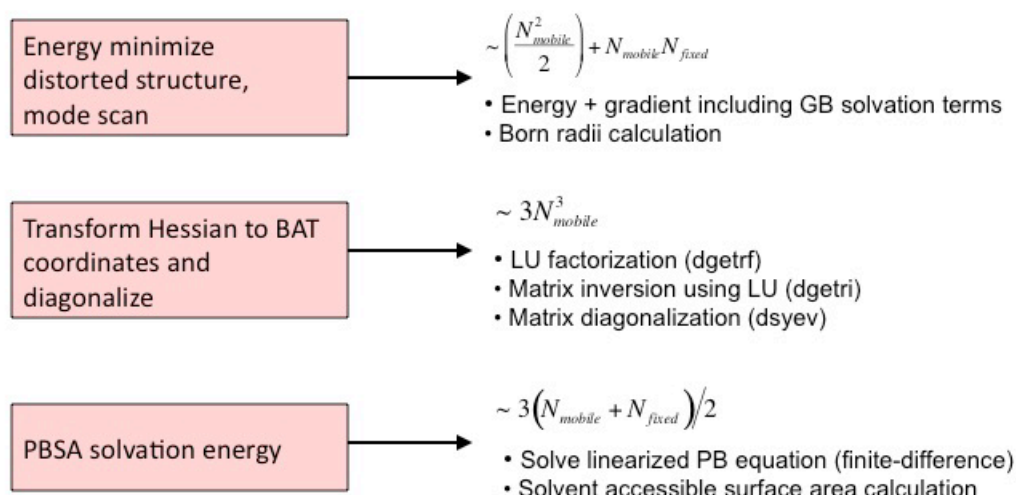
3.3. Non-blocking MPI communication

The coupled MPI implementation of the conformational search requires periodic communication of the lowest energy structure found between processes by use of MPI collective communication routines (e.g. MPI_ALLREDUCE), which results in a synchronization event between all MPI processes. For MPI VM2 protein-ligand calculations this leads to processes falling idle for significant amount of time while

waiting for other processes to “catch up” before the collective communication can occur. Therefore, an approach using non-blocking MPI calls was implemented, which allows processes to communicate their lowest energy conformers with other processes without a synchronization event.

4. Fine-grained parallelization

The figure below summarizes tasks in the VM2 algorithm targeted, based on profiling, for fine-grained parallelization. Scaling with respect to the number of atoms included in the calculation (mobile as well as those fixed in space) is also shown.



Shared-memory fine-grained parallelization of the routines associated with these tasks for both standard CPU multicore architectures using OpenMP and for GPU multicores using CUDA (<https://developer.nvidia.com/cuda-zone>) has been carried out. Energy minimization (i.e. geometry optimization) and mode scanning tasks are sped up by calls to shared-memory parallelized molecular mechanics plus Generalized Born solvation related energy and energy-gradient routines, transformation and diagonalization of the Hessian is addressed by use of calls to fine-grained parallelized linear algebra routines, and the PBSA energy correction is sped up by fine-grained parallelization of routines specified below (Section 4.1.3.).

4.1. OpenMP

4.1.1. OpenMP fine-grained parallelization of GB energy-gradient

OpenMP parallelization of energy-gradient routines, including those related to Generalized Born solvation terms, have been implemented by distribution of the inner loop of the atom-pair loop across cores. At the same time the inner loop takes advantage of cache blocking and vectorization via use of highly optimized vector math libraries e.g.

<https://software.intel.com/en-us/mkl-developer-reference-c-vector-mathematical-functions> .

4.1.2 OpenMP parallelization of Hessian transformation and diagonalization

OpenMP fine-grained parallelization of the transformation of the Cartesian coordinate Hessian to matrix to bond-angle-torsion (BAT) coordinates and diagonalization is achieved by calls to appropriate parallelized linear algebra routines. <https://software.intel.com/en-us/mkl-windows-developer-guide-improving-performance-with-threading>

4.1.3 OpenMP parallelization of PBSA

Profiling of the PBSA energy calculations carried out during VM2 calculations revealed that the boundary condition, modified incomplete Cholesky conjugate gradient (ICCG) solver, and energy-from-dielectric-boundary routines are where the majority of time is spent. OpenMP parallelization of the boundary condition and energy-from-dielectric-boundary routines has been carried out. Work is ongoing to OpenMP parallelize the solver step.

4.2. CUDA

4.2.1. CUDA based fine-grained parallelization of the GB energy-gradient

CUDA parallelization of energy-gradient routines, including those related to Generalized Born solvation terms, has been implemented and integrated with the VM2 quasi-Newton and conjugate gradient geometry optimization algorithms.

4.2.2. CUDA parallelization of Hessian transformation and diagonalization

CUDA parallelization of the transformation of the Cartesian coordinate Hessian to matrix to bond-angle-torsion (BAT) coordinates and diagonalization is implemented via calls to the Matrix Algebra on GPU and Multicore Architectures library. <http://icl.cs.utk.edu/magma/>

5. Combined coarse grained – fine grained parallelization of VM2

The parallel processor capabilities described above, namely a coarse-grained MPI implementation of VM2 and fine-grained parallelization of various VM2 constituent methods provide the infrastructure for a combined coarse grained-fine grained VM2 capability that can make optimal use of computer clusters regardless of the calculation size i.e. atom count.

5.1. MPI-OpenMP

The MPI-OpenMP combined package is suitable for running calculations on workstations and clusters of workstations.

5.2. MPI-CUDA and MPI-OpenMP-CUDA

The MPI-CUDA and MPI-OpenMP-CUDA packages are suitable for running calculations on workstations and clusters of GPUs with multiple GPUs.

V. Molecular system and input data file preparation

1. Molecular systems

VM2 can, in principle, be applied to any molecular system - size restrictions notwithstanding. The restriction in the currently available packages to standard classical molecular mechanics force fields does, however, preclude its use to describe bond breaking or excited state molecular processes.

The VM2 package suite has been tested for, and applied to, ligand molecules, protein macromolecules, protein-ligand complexes, host molecules, and host-ligand complexes.

1.1. Ligands

In the context of VM2 calculations, ligands are molecules ranging from a few atoms to usually no more than one hundred atoms, and with a current enforced limit of three hundred atoms. All license levels of the VM2 software package are ligand calculation capable.

VM2 offers very exhaustive conformational sampling for ligand molecules, which results in the reliable determination of global minima for the particular molecular mechanics energy potential in use. It also provides a Boltzmann distribution of ligand conformations in solvent. For large “floppy” ligands this is essential to accurately determine loss of entropy on binding.

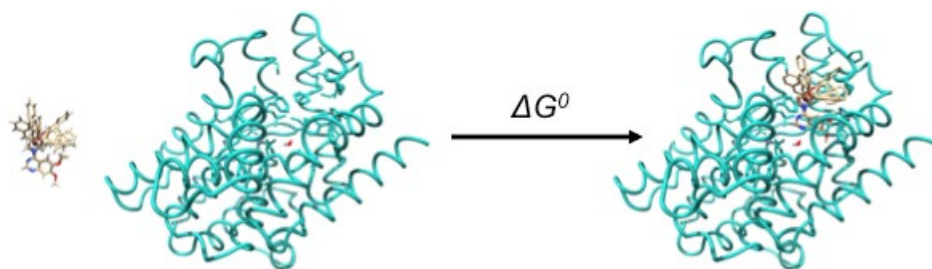
Figure.

1.2. Proteins

Protein macromolecules range from hundreds to tens of thousands of atoms. Unlike most software packages for calculation of molecular properties of proteins, VM2 does not use molecular dynamics methodology. Instead, as outlined in Section I, VM2 is an end-point based method and therefore employs very robust conformational searching of mobile protein atoms. This can be useful in various scenarios, in addition to the main focus of binding energy calculations, including full relaxation of chains that have been grafted on to incomplete structures.

1.3. Protein-ligand complexes

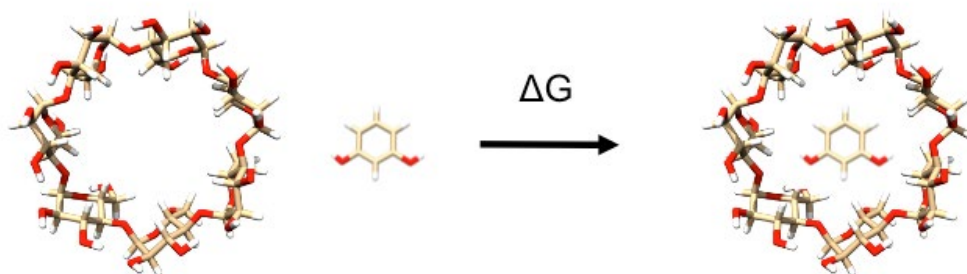
The accurate prediction of relative binding affinities of small molecules (ligands) to protein active sites is the central goal of the VM2 software package.



1.4. Host-ligand complexes

Host molecules such as cyclodextrin or cucurbituril can provide useful models to study binding affinity models. Due to their relatively small size, the conformational sampling burden, while still present, is less than for protein systems leading to much faster turnaround of calculations. As such, repeated rounds of calculations to study effects of various aspects of energy potentials can be carried out with modest computational cost.

In addition, the accurate prediction of binding affinities of ligand (or guest) molecules to hosts is useful in many applied research areas: drug discovery, enantiomeric separation science, chemical pollutant removal, and scavengers for chemical warfare agent removal, are examples.



2. Molecular system data sources

There are numerous sources that provide molecular data that can be a starting point for computational studies or research projects using the VM2 package. The following are commonly used.

2.1. The Protein Data Bank (PDB)

The PDB (<http://www.rcsb.org/pdb/home/home.do>) is repository of over one hundred thousand biological macromolecular structures. Many of the macromolecular structures contain co-crystallized ligands, providing an indication of, for example, a particular protein's active site.

The PDB provides structural data files for download in the PDB and CIF formats.

2.2. The Cambridge Crystallographic Data Centre (CCDC)

The CCDC (<http://www.ccdc.cam.ac.uk/pages/Home.aspx>) is a leading provider of small molecule crystal structure data. Over 875,000 fully curated entries are available for download in CIF format.

2.3. The Binding Data Base (BindingDB)

From the website the BindingDB (<https://www.bindingdb.org/bind/index.jsp>) is “a public, web-accessible database of measured binding affinities, focusing chiefly on the interactions of protein considered to be drug-targets with small, drug-like molecules.” It currently contains over 1,346,000 entries, for ~7,100 protein targets and ~ 600,000 small molecules.

Structural data as well as binding affinity data are available for download in various formats. This includes computationally docked conformations of congeneric series of ligands with their targets https://www.bindingdb.org/bind/surflex_entry.jsp.

2.4. Chemical components in the PDB (PDBeChem)

From the website the PDBeChem (<http://www.ebi.ac.uk/pdbe-srv/pdbechem/>) is a dictionary of chemical components – ligands, small molecules, and monomers – referred to in PDB entries. There are currently over 24,000 entries. A comprehensive search facility is provided.

The entries provide downloads of the component in multiple formats e.g. .mol, .mol with hydrogen atoms added (idealized structure), .pdb (ideal representation), mmCIF, CML, SMILES, etc.

2.5. ZINC: free database of commercially available compounds

From the website, <http://zinc15.docking.org>, ZINC “contains over 100 million purchasable compounds in ready-to-dock, 3D formats.”

The ZINC database (38) integrates and curates biological activity, chemical property, and commercial availability data for small molecules from public sources. In addition, calculated properties are added into a chemistry-aware relational database. It is an easy to use GUI for database interrogation and 3D structures for all molecules may be downloaded in mol2 and sdf formats, with biologically relevant tautomers and protonation states available for each compound.

3. Molecular system preparation steps and computational model choices

Once molecular data from one or more of the above sources is obtained, further preparation steps are required. These steps may include manipulation; for example, removal of some atoms and addition of others, to achieve chemistry related preferences such as particular protonation states. Specific step-by-step examples are described in Sections IX through XI: the following gives a general overview of potential issues and choices to be made.

3.1. Proteins

Often the starting point for a protein model preparation will be a PDB file downloaded from the Protein Data Bank (see **Section 2.1** above). A first step is to visualize and examine the protein using one of the many available graphical viewers that support the PDB format. Examples include UCSF Chimera, (39)VMD, (40) Discovery Studio Visualizer, and Maestro. If a co-crystallized ligand is present this provides the location of the binding site, which is useful information for set up of VM2 calculations of ligand binding affinities with this target. Visual examination may also indicate obvious problems such as chain breaks; other more subtle issues may be present such as incorrect stereochemistry of peptide bonds, which will require the use of interrogation methods these visualizers have available.

PDB submissions are not designed nor curated with facilitation of molecular modeling as a goal. As such, no attempts are made to “correct” issues that arise when areas of electron density cannot be resolved into structure. Therefore, issues such as missing side chains, no hydrogen atoms, incorrect residue stereochemistry etc. must be explicitly considered and addressed by the VM2 user. Furthermore, certain user decisions will affect how well the model will ultimately perform in binding free energy calculations. Example decisions include how to treat metal centers, which, if any, of the solvent molecules or ions present in the PDB file to retain in the calculation, what residue protonation states to choose, which atoms to make mobile and which atoms to include in the calculation but fix in space (see **Section I 1.11**), what force field to use, and how to deal with non-standard protein residues. Further details of potential problems and user decisions now follow. The graphical viewers mentioned above have tools that can aid users in this process; an additional non-visual package is [PDBFixer](#).

3.1.1. Missing side chains

If residue side chains are found to be missing they can either be added in their ideal geometry, or through rotamer libraries, to the system before typing and parameter assignment. If, however, they are very far away from the binding site or potentially will not even be present in the real/live set, they can be capped as GLY. The idealized geometry of any added side chain should be relaxed during the setup process.

3.1.2. Non-standard amino acids

If non-standard amino acid residues are present if parameters exist they should be typed and parameterized accordingly, if parameters do not exist and the residue is far from the active site it may be simply swapped out for a standard residue. If the residue is close to the active site it should be typed and charged using a generalized scheme.

3.1.3. Chain breaks

If a chain break occurs near the binding site or in a loop that plays an important role then the missing residues should be added. As for missing side chains, if the chain break is very far away from the binding site and likely to be fixed in space or even not present in the real/live set then the termini of the break can be capped.

3.1.4. Hydrogen addition

Most PDB files do not contain hydrogen atoms and they must be added appropriately. They may be added in idealized positions with idealized bond lengths etc. and relaxed during the setup process. Note that the PDB standard hydrogen atom-naming scheme is not identical to the naming scheme used by some preparation and typing tools.

3.1.5. Stereochemistry

Stereochemistry errors that can occasionally occur are *cis* peptide bond arrangements and incorrect alpha-carbon parity. It is preferable these issues are detected and fixed during preparation stage.

3.1.6. Metal centers

If metal centers, e.g. zinc, magnesium, iron, copper, occur far away from the binding site or even fall outside the ‘real’ set then minimal effort with respect to parameterization is required as they can be assigned unit charges e.g. Zn^{2+} , Mg^{2+} and possibly zero force constant bonds. However, if metal centers are nearby the binding site it is highly recommended that if parameters have not already been established for the particular binding motif of the metal, that efforts through electronic structure calculations etc. are made to assign appropriate charges, and force constants.

3.1.7. Solvent and ions

Protein structures in the PDB often include water with the water oxygen atoms indicating their positions. Ions can also be present. For VM2 binding affinity calculations the current recommendation is to remove all ions. With respect to water, the vast majority of water molecules should always be removed and often all the water molecules are removed. However, if it is recognized that certain water molecules in the binding pocket are playing a role in the binding of the co-crystallized ligand then such water molecules should be included in the VM2 calculation as part of the protein ‘live’ set. Users may want to utilize 3rd-party software that may algorithmically identify important water molecules in the binding site.

3.1.8. Specific residue protonation states

These are five ionizable protein residue types: Asp, Glu, His, Tyr, Lys. Their states are known at standard physiological pH 7.4, but the local pH in a binding pocket can vary according to the local structure. Also, it is possible that a ligand on binding will induce a change in the protonation state of binding pocket residues. Users may want to employ 3rd-party software that attempts to predict protein system protonation states e.g. Propka. (41, 42)

3.1.9. Residue mutations

For studies of the effect of residue mutations on protein-ligand binding affinities during the protein preparation removal of the current residue and replacement with the mutant residue is required with the associated atom renumbering etc.

3.1.10. Choice of protein real/live set

The choice of protein atoms to include in the calculation (real atoms) and which of these should be mobile (live atoms) should be taken with care. Enough protein atoms should be live so the protein environment can adjust appropriately when the binding ligand is present. Furthermore, if the system under consideration contains a flexible loop that can affect the active site, as it the case for kinase systems, this loop should be mobile. Of course, the larger the real/live set is, the more computationally demanding a VM2 run will be, so choosing a real/live set that is larger than needed should also be avoided. Arriving at the correct balance in this regard sometimes requires a trial and error process.

3.1.11. Protein atom typing and parameters

As described in Section II VM2 supports various force fields types: AMBER, CHARMM, OPLS, and Dreiding. Tools for protein typing and parameter assignment include AmberTools,(43) UCSF Chimera,(39) VMD,(40) Discovery Studio Visualizer, OpenMM,(44) and Maestro. Additionally, VeraChem provides a tool for typing and assigning parameters for proteins. Several web-based tools also exist e.g. CHARMM-GUI (45) and CHARMMing. (46)

3.2. Host molecules and ligands

Host molecules and especially ligand molecules provide further challenges with respect to preparation, typing, and parameter assignment, as in contrast to protein systems, where constituent amino acids are specifically characterized and preparation tools can take advantage of this pre-knowledge, generalized approaches must be used. Generalized preparation steps include hydrogen addition, protonation state assignment, bond order recognition, assignment of stereochemistry, and generalized typing and parameter assignment.

3.2.1. Hydrogen addition

Often source files do not contain information on hydrogen atoms for host and ligand molecules, in this case they must be added.

3.2.2. Protonation states

When adding hydrogen atoms, an important consideration is the possible protonation states as the choice of protonation state can have a large effect on predicted binding affinities. If pKa values are known for the ligand under consideration (or for a similar chemical motif) then this may be adequate for good choices. If no pKa values are available, the use of third-party software for prediction of protonation states can be considered.

3.2.3. Bond order recognition

It is important to assign correct bond orders as this affects force field parameter assignment, and in addition the VM2 algorithms make use of bond order information when performing conformational searches and other procedures.

3.2.4. Stereochemistry

Care should be taken that the stereochemistry of the host and/or ligand molecule prepared is correct, as the VM2 algorithms will recognize the stereochemistry as supplied, and, if requested, maintain that stereochemistry throughout the calculation.

3.2.5. General atom typing and parameters

Atom typing and parameterization for general molecules remains a considerable challenge. There are now available packages for typing and parameter assignment of general molecules. A commonly used tool is antechamber, distributed with AmberTools, which can type and assign GAFF parameters for general molecules. The CGenFF program types and assigns parameters from the CGenFF parameter set.

4. Mandatory formatted molecular data file generation

Once the molecular systems are prepared as described above, the mandatory formatted molecular data files **.crd**, **.top**, and **.mol/.sdf** must be generated (see Section VI.2.). Usually the tools employed for the molecular system preparation will provide other formatted files such as **.psf** or **.prmtop**, which must then be converted to the required formats by VeraChem supplied scripts. There follows outlines of preparation/conversion routes currently available.

5. System and data file preparation routes

Currently, there are four established routes for molecular system preparation and subsequent VM2 input data file preparation. Three routes use VeraChem scripts to convert third-party software produced data files to VeraChem format input files, the remaining route is through VeraChem's own preparation and typing tools.

Route 1

AmberTools, UCSF Chimera, OpenMM

- Molecule prep
- Protein: typing/charges for Amber
- Ligand: GAFF typing; AM1-bcc charges
- Output **.prmtop** and **.inpcrd** files

VeraChem Conversion Script



.top, .mol, .crd

Route 2

Biovia's Discovery Studio Visualizer, CHARMMing, CHARMM-GUI

- Molecule prep
- Output **.mol** and **.psf** files

VeraChem Conversion Script



.top, .mol, .crd

Route 3

Schrodinger's Maestro/Macromodel

- Molecule prep using Maestro
- Macromodel single-point energy and output **.mmo** file

VeraChem Conversion Script



.top, .mol, .crd

Route 4 (development ongoing)

VeraChem's own prep and typing tools

- Basic molecule prep
- Protein: typing for CHARMM, Amber, OPLS
- Ligands: GAFF and Dreiding typing; Vcharge

Produced Directly



.top, .mol, .crd

The basic steps for each route are now described. For step-by-step specific examples see Sections IX – XI.

5.1. Route 1: Conversion of Amber style input files

For protein molecules any preparation and typing software tools that produce the Amber **.prmtop** and **.inpcrd** files are suitable; examples, which will be discussed in more detail below are AmberTools software, UCSF Chimera, and OpenMM. These Amber formatted files are used as input for VeraChem's conversion tool. The following is an example of usage:

```
$VCPYTHON $VCHOME/exe/prm2top.pyc -prm protein.prmtop -crd protein.inpcrd -protein >& log.out
```

The **.top**, **.mol**, and **.crd** files required to run VM2 are written out and any warnings appear in **log.out**.

For ligands/host molecules if a high quality **.mol2** file is already available it can be immediately used in conjunction with the tools mentioned above, namely AmberTools, UCSF Chimera, and OpenMM, which can type and assign parameters via Antechamber and output **.prmtop** and **.inpcrd** files for the ligand.

However, if, for example, the user is starting only with a chemical formula and is sketching a 2-D structure, or is starting with a SMILES string or 2-D .sdf/.mol file, a reliable conversion to a 3-D structure, with addition of hydrogen atoms, assignment of bond orders and stereochemistry, and generation of an associated .mol2 formatted file, is required before proceeding with the .prmtop and .inpcrd file generation.

Software that can be used to sketch 2D structures include Software that can read in SMILES strings and convert to 3D include Software that can convert 2D mol/sdf files to 3D include

Once the .prmtop, .inpcrd, and .mol2 files are available as for the protein case they are used as input for VeraChem's conversion tool to produce the required .top, .mol, and .crd files:

```
$VCPYTHON $VCHOME/exe/prm2top.pyc -prm ligand.prmtop -crd ligand.inpcrd -  
mol2 ligand.mol2 >& log.out
```

5.1.1. AmberTools

See ambertools test directory for 1ke5... look at the run.sh scripts in each of the subdirectories.. (also note that the ligand can be in mol format as well, not just pdb as is the case in the write up).

For proteins:

```
tleap -s -f protein_leap.in >& log.out
```

For ligands the sequence is

```
ligand_antechamber  
antechamber -i ligand.mol -fi mdl -o ligand.mol2 -fo mol2 -c bcc -s 2 -df 2 -nc 0 -j 5 -dr  
no >& log.out
```

```
ligand_parmchk  
parmchk -i ligand.mol2 -f mol2 -o frmod >& log.out
```

```
ligand_tleap  
tleap -s -f ligand_leap.in >& log.out
```

Where ligand_leap.in is:

```
verbosity 1  
source leaprc.gaff  
mods = loadamberparams frmod  
MOL = loadmol2 <ligand>.mol2  
list  
saveamberparm MOL <ligand>.prmtop <ligand>.inpcrd
```

quit

and <ligand> must be replaced with the actual name of the ligand. For example
umass_1_ad_23, ad_81, etc.

The prepareLigands.pyc script automates this process to allow the preparation of an entire set of ligands contained in a single sdf file. The steps above are executed for each ligand in the file in order. It is also possible to specify a range of ligands within the sdf to process. It is important that the ligands in the sdf file have all hydrogens present, stereochemistry defined with parity values, and correct formal charges. The complete Protein – ligand example that is included later in this document provides an example of its use.

5.1.2. UCSF Chimera

The visualization and analysis package UCSF Chimera (39)

5.1.3. OpenMM

The open source toolkit OpenMM (44)

5.2. Route 2: Conversion of CHARMM style input files

protein

```
$VCPYTHON $VCHOME/exe/psf2top.pyc -psf protein.psf -crd protein.sd >& log.out
```

Ligand

```
$VCPYTHON $VCHOME/exe/psf2top.pyc -psf ligand.psf -crd ligand.sd >& log.out
```

5.2.1. Discovery Studio Visualizer

The BIOVIA Discovery Studio Visualizer is a freely available graphical visualization tool: <http://accelrys.com/products/collaborative-science/biovia-discovery-studio/visualization.html>. Discovery Studio Visualizer (DSV) supports the import and export of common molecular data file formats e.g. PDB, crd, mol2, sdf etc., and it also provides tools for building and manipulating small molecules and macromolecules. DSV provides a convenient path for building/loading/editing molecules and outputting data files that can then be read by VeraChem's conversion scripts, which then output the required input files to run VM2 calculations.

The following provides a brief step-by-step description of how to generate VM2 input data files using the DSV pathway.

Step 1: Import or use DSV to build your target molecular system e.g. small molecule or protein.

Step 2: Use DSV to visualize and edit your molecular system as required e.g. correct bond orders, add missing side-chains etc.

Step 3: For small molecules, e.g. ligands and/or host molecules, if required, refine the structure using the **Clean Geometry** option in DSV under the **Structure** top menu item.

Step 4: Under the **Tools** top menu item select **Simulation** and then in the resulting sub-menu select **Change Forcefield**. This will bring up the following force field options in the far left panel: **Forcefield**, **Forcefield Status**, and **Forcefield Customization**. Under the **Forcefield** option select a **CHARMm** or **charmm** option and then press the **Apply Forcefield** button. The **Forcefield Status** should then be indicated as typed.

Step5: Save the required formatted files: .crd, .psf, .sd, and .mol2. To do this, go to the **File** top menu item, select the option **Save As**, and then select **CHARMm Simulation Files** as the **Save as Type**. The extension you provide will determine format of the file; i.e., if you specify 1hvr.crd it will output a crd file, and if you specify 1hvr.psf, a psf will be written even though the **Save as Type** remains the same.

Step 6: Check the .sd file to make sure that it contains a ffml data block. If no parameters had to be estimated it will be empty, e.g.

```
-----  
> <ForcefieldFFML>  
<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE ffml SYSTEM "Ffml.dtd">  
<ffml version="1.0">  
<forcefield name="1HVR-CHARMm" derivedFrom="CHARMm" base="CHARMm">  
<atomTypes>  
</atomTypes>  
<parameters>  
</parameters>  
</forcefield>  
</ffml>  
  
$$$$  
-----
```

If there are estimated parameters they will appear as in the following example:

```
-----  
> <ForcefieldFFML>  
<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE ffml SYSTEM "Ffml.dtd">  
<ffml version="1.0">  
<forcefield name="1HVR-CHARMm" derivedFrom="CHARMm" base="CHARMm">  
<atomTypes>  
</atomTypes>  
<parameters>  
<bond atom1="SO1" atom2="OT">  
<quadratic  
refValue="1.55"  
-----
```

```

    forceConstant="220"/>
</bond>
<angle atom1="CT" atom2="SO1" atom3="OT">
<quadratic
    refValue="107"
    forceConstant="66"/>
</angle>
<angle atom1="SO1" atom2="OT" atom3="HO">
<quadratic
    refValue="109.147"
    forceConstant="51.3667"/>
</angle>
<torsion atom1="CT" atom2="SO1" atom3="OT" atom4="HO">
<multiplecos
    v1="0"
    v2="0"
    v3="0.2"
    v4="0"
    v5="0"
    v6="0"
    gamma1="0"
    gamma2="0"
    gamma3="0"
    gamma4="0"
    gamma5="0"
    gamma6="0"/>
</torsion>
</parameters>
</forcefield>
</ffml>

```

\$\$\$\$

If this block is not present save an ffml file with the information directly by selecting the **More** option under **Forcefield Status** in the far left panel.

Step 7: Conversion to VM2 input files via VeraChem scripts. First, the environment variable VCDSPATH must be set to the location of the CHARMM forcefield files from your installation of Discovery Studio Visualizer. (For the 2016 version on the PC this is DiscoveryStudio_2016/share/forcefield/CHARMM.) Then invoke the conversion scripts as follows:

Ligand

```
$VCPYTHON $VCHOME/exe/psf2top.pyc -psf ligand.psf -crd ligand.sd >& log.out
```

protein

```
$VCPYTHON $VCHOME/exe/psf2top.pyc -psf protein.psf -crd protein.sd >& log.out
```

The seven step process just described will produce the .crd, .top, and .mol files required to run VM2 calculations.

5.2.2. CHARMMing

The CHARMMing website interface ...

5.2.3. CHARMM-GUI

An alternative web-based tool is the CHARMM-GUI ...

5.3. Route 3: Maestro/Macromodel (OPLS2005)

VeraChem provides a python script that parses a Schrodinger Inc. formatted .mmo data file and outputs the .crd, .top, and .mol/.sdf files required to run VM2 calculations.

Access to the 3rd-party software Maestro and Macromodel is required. The Maestro graphical user interface is used to prepare molecules e.g. hydrogen addition etc. and MacroModel is required for the final generation of an .mmo file for conversion.

5.3.1. System Preparation and Generation of MMO files

(1) From the Maestro tool bar, select Workflows -> Protein Preparation Wizard.

In the wizard:

- Load structure to workspace;

- Use all default values for ligands; for protein, select "Cap termini";

- Click "Preprocess";

- On Page "Review and Modify", delete unwanted waters and Hets from the lists for proteins.

(2) From the tool bar, select Application -> MacroModel -> Current Energy

On page "Potential":

- Select force field as OPLS2005;

- Set solvent to "none";

- Set cutoff to "none" for ligands and to "normal" for proteins;

On page "Ecalc":

- Set energy list to "complete";

- Click "Start", give a name for the MMO file to output.

5.3.2. Conversion of MMO files to .crd, .top, and .mol/.sdf

Then you would run

```
$VCPYTHON $VCHOME/exe/mmo2top.pyc ligand.mmo >& log.out
```

```
$VCPYTHON $VCHOME/exe/mmo2top.pyc protein.mmo >& log.out
```

To generate the crd, top, and 'mol' files

5.4. VeraChem preparation tools

VeraChem supplies its own for PDB/CIF file parsing, protein preparation, protein atom typing and parameter assignment, and .crd, .top, and .mol file generation.

VI. Running VM2 calculations

To run VM2 package calculations an input file (**.inp**), which sets calculation options, three molecular/force field data files (**.crd**, **.top**, **.mol/.sdf**), and a run script are always required. Additional data files may be required depending on the chosen molecular system type as well as other specific user choices. The following provides a summary of basic requirements, which are covered in more detail further below.

Files		Information taken
<u>Mandatory</u>		
receptor.top and ligand.top	→	Force field parameters. (VeraChem topology files)
receptor.crd and ligand.crd	→	Standard format. Starting coords, residue names etc.
receptor.mol and ligand.mol	→	Standard format. Bond orders and atom parities.
receptor_ligand_vm2.inp	→	Input text file. Calculation control options.
<u>Optional</u>		
co_xtal_ligand.pdb, .sdf, ...	→	Standard formats. Coords to define binding pocket.
real_live_atoms.txt	→	Mobile and fixed atoms. (protein atoms only)
constrained_atoms.txt	→	Atoms to constrain. (harmonic/flat bottomed well)
excluded_driver_atoms.txt	→	Drivers that contain these atoms excluded from search distortions.

All files are, except where explicitly stated, ASCII text files. A description of their roles and content now follows.

1. Mandatory input file (.inp)

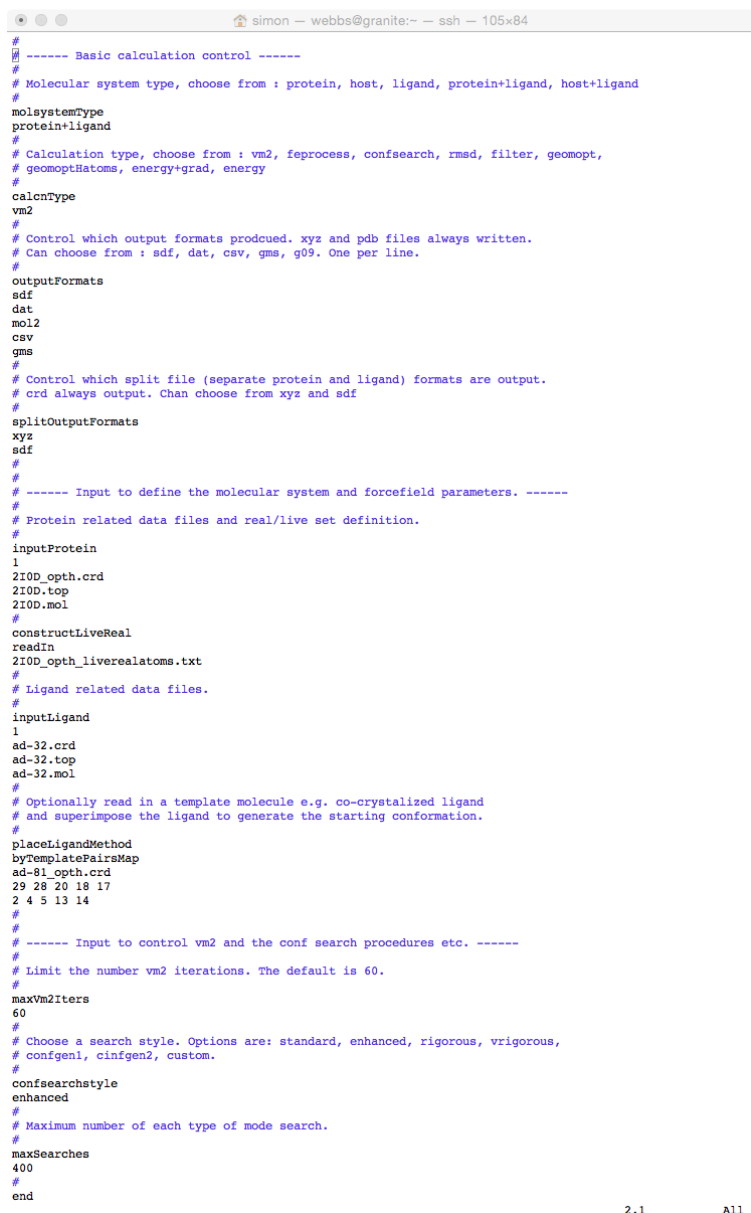
The mandatory input file must have the **.inp** suffix. It contains keywords and associated options that control the calculation. The following is a very simple example for a ligand VM2 calculation that uses only program defaults. The only keywords/options shown are those user is required to supply. The hash sign '#' tells the parser not to read that particular line. It can be used to add comments, but is also required at the transition between keywords. See **Section VIII** for a full list of input keywords/options and example usage.

```
#
molSystemType
ligand
#
calcnType
vm2
#
inputLigand
```

1

```
~/path/ligand_name.crd  
~/path/ligand_name.top  
~/path/ligand_name.mol  
#  
end
```

The following is a screen shot of a more complex .inp file for a protein-ligand VM2 calculation.



```
#  
# ----- Basic calculation control -----  
#  
# Molecular system type, choose from : protein, host, ligand, protein+ligand, host+ligand  
#  
molSystemType  
protein+ligand  
#  
# Calculation type, choose from : vm2, feprocess, confsearch, rmsd, filter, geomopt,  
# geomoptHatoms, energy+grad, energy  
#  
calcType  
vm2  
#  
# Control which output formats produced. xyz and pdb files always written.  
# Can choose from : sdf, dat, csv, gms, g09. One per line.  
#  
outputFormats  
sdf  
dat  
mol2  
csv  
gms  
#  
# Control which split file (separate protein and ligand) formats are output.  
# crd always output. Can choose from xyz and sdf  
#  
splitOutputFormats  
xyz  
sdf  
#  
# ----- Input to define the molecular system and forcefield parameters. -----  
#  
# Protein related data files and real/live set definition.  
#  
inputProtein  
1  
2I0D_oph.crd  
2I0D.top  
2I0D.mol  
#  
constructLiveReal  
readIn  
2I0D_oph_livereatoms.txt  
#  
# Ligand related data files.  
#  
inputLigand  
1  
ad-32.crd  
ad-32.top  
ad-32.mol  
#  
# Optionally read in a template molecule e.g. co-crystallized ligand  
# and superimpose the ligand to generate the starting conformation.  
#  
placeLigandMethod  
byTemplatePairsMap  
ad-81_oph.crd  
29 28 20 18 17  
2 4 5 13 14  
#  
# ----- Input to control vm2 and the conf search procedures etc. -----  
#  
# Limit the number vm2 iterations. The default is 60.  
#  
maxVm2Iters  
60  
#  
# Choose a search style. Options are: standard, enhanced, rigorous, vigorous,  
# confgen1, cinfgn2, custom.  
#  
confsearchstyle  
enhanced  
#  
# Maximum number of each type of mode search.  
#  
maxSearches  
400  
#  
end
```

2,1

All

2. Mandatory data files

The three mandatory data files .crd, .top, .mol supply molecular data and force field data to the program.

2.1. .crd file

This is the standard CHARMM format card file. See the following link for format details

<https://www.charmm.org/charmm/documentation/by-version/c40b1/params/doc/io/#Top>

It is used to supply VM2 the initial molecular geometry, residue names and IDs, and IUPAC protein atom names. (47) For host and ligand molecules reasonable atom names are sufficient.

2.2. .top file

This is a VeraChem formatted file. This file provides atomic masses, atomic partial charges, the Lennard-Jones parameters r^{\min} and ϵ . Additionally, it defines the molecule's topology (bonds, angles, proper and improper dihedrals) and provides the associated force constant parameters. See section II for the **.top** file format specification and Section XII for a specific example.

2.3. .mol/.sdf file

This is a standard and widely used format originally developed at Molecular Design Limited. (48) This file provides atom parities, bond orders, and stereochemistry to the VM2 package. This information use in VM2 includes determination and maintenance of stereochemistry, reduction of torsional search space, and correct output of other descriptive molecular formats.

3. Optional data files

3.1 Formatted file defining atoms/points in space used for automatic generation of protein real/live sets

The user supplies a formatted file that defines atoms or points in space – these atoms/points are usually within the protein binding pocket, and could, for example, represent the co-crystallized ligand. In the **.inp** file, the user also supplies cutoff distances relative to these atom positions/points in space that are used to determine which atoms are in the real/live atom regions. The software outputs an associated formatted text file (see 3.2 below) that defines these regions, which can then be read in for any subsequent calculations so importantly the exact same real/live protein atom set can reproduced for a series of protein-ligand complexes. Allowed formats are **.crd**, **.xyz**, **.sdf**, **.mol**, **.pdb**, and Macromodel **.dat**.

3.2. Formatted text file that explicitly defines fixed and mobile atoms

The user can generate this VeraChem formatted file or can use one programmatically generated - see 3.1 above. An example name could be

real_live_atoms.txt

See section XII 2. for the “real_live_atoms.txt” file format specification.

3.3. Formatted file containing atoms to be constrained

A VeraChem formatted file that lists atoms that will have energy constraints applied to them. An example name could be

constrained_atoms.txt

See section XII 3. for the “constrained_atoms.txt” file format specification.

3.4. Formatted file containing atoms to be excluded from conformational searches

A file that lists atoms that if found in a mode based search driver cause that driver to be excluded from the set used in the Vconf conformational search procedures. An example name could be

excluded_driver_atoms.txt

See section XII 4. for the “excluded_atoms.txt” file format specification.

3.5. Standard format files containing coordinates of previously generated molecular conformers

The standard formatted file types .xyz, .sdf, .mol, Macromodel .dat, and .crd containing previously generated conformers may be read in to provide a starting point for a new calculation.

3.5.1 Ability to read in multiple conformers to initiate runs

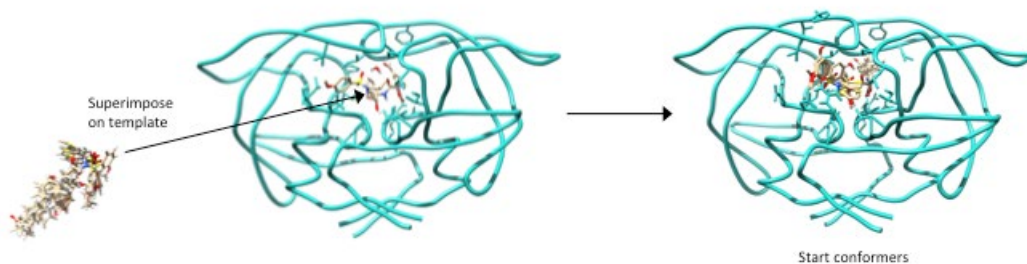
As described earlier, the VM2 algorithm relies on the generation of low energy conformers of the molecular system. Determining the lowest energy conformations of a system can sometimes require a multistep process. To this end, VM2 has the capability to read in sets of conformers via formatted text files, so conformers from previous VM2 runs, or indeed conformers generated by third-party software, can be used to initiate new runs. These conformers can, if desired, be simply processed to provide a free energy value or may serve as a starting point for a full VM2 iterative calculation.

For protein-ligand molecular systems, the user can choose to read in only ligand or only protein conformers, and the software will construct protein-ligand conformations, by pairing with the basic set up protein and each ligand structure, respectively.

An example use of this is for congeneric series of ligands where the ligand scaffold position in the active site is known and is kept fixed in position, conformers are then generated that sample different R-group positions.

Confgen figure here ...

These separately generated ligand conformations are then read in for generation of protein-ligand starting conformations, speeding up the search for low energy protein-ligand conformations.



Another example is the case where nothing is known about a ligand pose in the binding site. In this case, a conformer generation setting can be used to generate ~ 20 ligand conformers, which sample R-group positions, and then randomly rotate these conformers about their center of geometry to provide an ensemble of 1000 conformers that can be used to generate protein-ligand starting conformations for a VM2 run.

Confgen4 figure here ...

4. Environment variables

4.1. Placeholder

4.2. Placeholder

4.3. Placeholder

4.4. Placeholder

5. Run scripts

Example shell scripts for running VM2 calculations are presented below. In addition, as most computer clusters, whether at pharmaceutical companies, universities, or government laboratories require that calculations be submitted via a batch queue system, we also provide examples for common batch queue run scripts: Portable Batch System (PBS), Simple Linux Utility for Resource Management (SLURM), and Platform Load Sharing Facility (LSF).

The run script examples assume the environment variable VCHOME is set elsewhere; other required environment variables are set in the scripts. For example, the OMP_NUM_THREADS, MKL_NUM_THREADS and I_MPI_PIN_DOMAIN environment variables are always set in the MPI and mixed MPI/OpenMP examples below. For runs with the purely serial VM2 executable, these variables would not be required. The required environment variables can also be set in the user's **.bashrc** or **.cshrc** file if preferred.

5.1. Bash shell scripts

To submit a calculation, where the bash script is called my_bash_script.bsh, issue the command

```
./my_bash_script.bsh >& my_bash_script.log &
```

5.1.1. Example 8 MPI process run

```
-----  
#!/bin/bash  
#  
# Bash-shell script  
  
ulimit -s unlimited  
  
# Set locations and environment for Intel  
INTEL_LIBS=$VCHOME/lib/intel  
INTEL_MKL_LIBS=$INTEL_LIBS/mkl  
INTEL_MPI_LIBS=$INTEL_LIBS/mpi  
  
LD_LIBRARY_PATH=$INTEL_LIBS:$INTEL_MKL_LIBS:$INTEL_MPI_LIBS  
export LD_LIBRARY_PATH
```

```

PATH=$INTEL_MPI_LIBS:$PATH
export PATH

export OMP_NUM_THREADS=1
export MKL_NUM_THREADS=1
export I_MPI_PIN_DOMAIN=omp
export KMP_STACKSIZE=16m

# Set VM2 executable to use
VC_FORTRAN_EXE=$VCHOME/exe/VC_CompChemPackage_mpi.exe

# Set VM2 input and output file names
VC_IN_FILE=protein_ligand_vm2.inp
VC_OUT_FILE=protein_ligand_vm2.out

# Will run 8 MPI processes
nohup mpirun -n 8 $VC_FORTRAN_EXE $VC_IN_FILE $VC_OUT_FILE

```

5.1.2. Example 8 MPI processes, 2 OpenMP threads per MPI process, 1 GPU per MPI process

```

#!/bin/bash
#
# Bash-shell script

ulimit -s unlimited

# Designed to run on a node equipped with dual xeon 8 core processors
# and 4 K80 Tesla cards (8 GPUs)

# Set locations and environment for Intel and CUDA libraries
INTEL_LIBS=$VCHOME/lib/intel
INTEL_MKL_LIBS=$INTEL_LIBS/mkl
INTEL_MPI_LIBS=$INTEL_LIBS/mpi

CUDA_LIBS=$VCHOME/lib/cuda:$VCHOME/lib/magma

LD_LIBRARY_PATH=$INTEL_LIBS:$INTEL_MKL_LIBS:$INTEL_MPI_LIBS:$CUDA_LIBS
export LD_LIBRARY_PATH

PATH=$INTEL_MPI_LIBS:$PATH
export PATH

# Set number of OpenMP threads to run per MPI process
export OMP_NUM_THREADS=2
export MKL_NUM_THREADS=2
export I_MPI_PIN_DOMAIN=omp
export KMP_STACKSIZE=16m

# Set VM2 executable to use
VC_FORTRAN_EXE=$VCHOME/exe/VC_CompChemPackage_mpi_openmp_cuda.exe

# Set VM2 input and output file names
VC_IN_FILE=protein_ligand_vm2.inp
VC_OUT_FILE=protein_ligand_vm2.out

```

```
# Will run 8 MPI processes and 2 OpenMP threads per process.
# Also uses 8 GPUS.
nohup mpirun -n 8 $VC_FORTRAN_EXE $VC_IN_FILE $VC_OUT_FILE
```

5.2. C-shell scripts

The following are the C-shell equivalents of the bash scripts in 5.1. above. To submit a calculation, where the C-shell script is called my_csh_script.csh, issue the command

```
./my_csh_script.csh >& my_csh_script.log &
```

5.2.1. Example 8 MPI process run using C-shell

```
#!/bin/csh
#
# C-shell script

limit stacksize unlimited

# Set locations and environment for Intel
setenv INTEL_LIBS $VCHOME/lib/intel
setenv INTEL_MKL_LIBS $INTEL_LIBS/mkl
setenv INTEL_MPI_LIBS $INTEL_LIBS/mpi

LD_LIBRARY_PATH=$INTEL_LIBS\: $INTEL_MKL_LIBS\: $INTEL_MPI_LIBS
setenv LD_LIBRARY_PATH

PATH=$INTEL_MPI_LIBS\: $PATH
setenv PATH

setenv OMP_NUM_THREADS 1
setenv MKL_NUM_THREADS 1
setenv I_MPI_PIN_DOMAIN omp
setenv KMP_STACKSIZE 16m

# Set VM2 executable to use
set VC_FORTRAN_EXE=$VCHOME/exe/VC_CompChemPackage_mpi.exe

# Set VM2 input and output file names
set VC_IN_FILE=protein_ligand_vm2.inp
set VC_OUT_FILE=protein_ligand_vm2.out

# Will run 8 MPI processes
nohup mpirun -n 8 $VC_FORTRAN_EXE $VC_IN_FILE $VC_OUT_FILE
```

5.2.2. Example 8 MPI processes, 2 OpenMP threads per MPI process, 1 GPU per MPI process, using C-shell

```
#!/bin/csh
#
```

```

# C-shell script

limit stacksize unlimited

# Designed to run on a node equipped with dual xeon 8 core processors
# and 4 K80 Tesla cards (8 GPUs)

# Set locations and environment for Intel and CUDA libraries
setenv INTEL_LIBS $VCHOME/lib/intel
setenv INTEL_MKL_LIBS $INTEL_LIBS/mkl
setenv INTEL_MPI_LIBS $INTEL_LIBS/mpi

CUDA_LIBS=$VCHOME/lib/cuda\: $VCHOME/lib/magma

LD_LIBRARY_PATH=$INTEL_LIBS\: $INTEL_MKL_LIBS\: $INTEL_MPI_LIBS\
: $CUDA_LIBS
setenv LD_LIBRARY_PATH

PATH=$INTEL_MPI_LIBS\: $PATH
setenv PATH

# Set number of OpenMP threads to run per MPI process
setenv OMP_NUM_THREADS 2
setenv MKL_NUM_THREADS 2
setenv I_MPI_PIN_DOMAIN omp
setenv KMP_STACKSIZE 16m

# Set VM2 executable to use
set VC_FORTRAN_EXE=$VCHOME/exe/VC_CompChemPackage_mpi_omp_cuda.exe

# Set VM2 input and output file names
set VC_IN_FILE=protein_ligand_vm2.inp
set VC_OUT_FILE=protein_ligand_vm2.out

# Will run 8 MPI processes and 2 OpenMP threads per process.
# Also uses 8 GPUS.
nohup mpirun -n 8 $VC_FORTRAN_EXE $VC_IN_FILE $VC_OUT_FILE
-----

```

5.3. PBS batch queue scripts

The following are PBS batch queue scripts initiate the equivalent runs to those the bash scripts in 5.1. initiate. To submit a calculation, where the PBS script is called my_pbs_script.pbs, issue the command

```
qsub my_pbs_script.pbs
```

5.3.1. Example 8 MPI process PBS run

```

-----
#!/bin/bash
#PBS -q default
#PBS -N test
#PBS -l nodes=1:ppn=8
#PBS -o test.out

```

```

#PBS -e test.err

ulimit -s unlimited

# Set location and environment for Intel libraries
INTEL_LIBS=$VCHOME/lib/intel
INTEL_MKL_LIBS=$INTEL_LIBS/mkl
INTEL_MPI_LIBS=$INTEL_LIBS/mpi

LD_LIBRARY_PATH=$INTEL_LIBS:$INTEL_MKL_LIBS:$INTEL_MPI_LIBS
export LD_LIBRARY_PATH

PATH=$INTEL_MPI_LIBS:$PATH
export PATH

# Set number of OpenMP threads to run per MPI process
export OMP_NUM_THREADS=1
export MKL_NUM_THREADS=1
export I_MPI_PIN_DOMAIN=omp
export KMP_STACKSIZE=16m

# Set VM2 executable to use
VC_FORTRAN_EXE=$VCHOME/exe/VC_CompChemPackage_mpi.exe

# Set VM2 input and output file names
VC_IN_FILE=protein_ligand_vm2.inp
VC_OUT_FILE=protein_ligand_vm2.out

cd $PBS_O_WORKDIR

# Will run a total of 8 MPI processes on a single node
mpirun -n 8 $VC_FORTRAN_EXE $VC_IN_FILE $VC_OUT_FILE

```

5.3.2. Example 8 MPI processes, 2 OpenMP threads per MPI process, 1 GPU per MPI process, using PBS

```

#!/bin/bash
# Modify to match local setup
# This script has not been tested
#PBS -q default
#PBS -N test
#PBS -l nodes=1:ppn=16
#PBS -o test.out
#PBS -e test.err

ulimit -s unlimited

# Designed to run on a node equipped with dual xeon 8 core processors
# and 4 K80 Tesla cards (8 GPUs)

# Set location and environment for Intel and CUDA libraries
INTEL_LIBS=$VCHOME/lib/intel
INTEL_MKL_LIBS=$INTEL_LIBS/mkl
INTEL_MPI_LIBS=$INTEL_LIBS/mpi

```



```

CUDA_LIBS=$VCHOME/lib/cuda:$VCHOME/lib/magma

LD_LIBRARY_PATH=$INTEL_LIBS:$INTEL_MKL_LIBS:$INTEL_MPI_LIBS:$CUDA_LIBS
export LD_LIBRARY_PATH

PATH=$INTEL_MPI_LIBS:$PATH
export PATH

# Set number of OpenMP threads to run per MPI process
export OMP_NUM_THREADS=2
export MKL_NUM_THREADS=2
export I_MPI_PIN_DOMAIN=omp
export KMP_STACKSIZE=16m

# Set VM2 executable to use
VC_FORTRAN_EXE=$VCHOME/exe/VC_CompChemPackage_mpi_openmp_cuda.exe

# Set VM2 input and output file names
VC_IN_FILE=protein_ligand_vm2.inp
VC_OUT_FILE=protein_ligand_vm2.out

cd $PBS_O_WORKDIR

# Runs 8 MPI processes with 2 OpenMP threads per processes.
# Also uses 8 GPUs, all on a single node.
mpirun -n 8 $VC_FORTRAN_EXE $VC_IN_FILE $VC_OUT_FILE

```

5.4. SLURM batch queue scripts

Below are SLURM batch queue scripts equivalent to the PBS scripts in 5.3. above. To submit a calculation, where the SLURM script is called `my_slurm_script.sh`, issue the command

```
sbatch my_slurm_script.sh
```

5.4.1. Example 8 MPI process SLURM run

```

#!/bin/bash
# Modify to match local setup
# This script has not been tested
#SBATCH --partition=default
#SBATCH --job-name=test
#SBATCH --ntasks=8
#SBATCH --ntasks-per-node=8
#SBATCH --output=test.out
#SBATCH --error=test.err
#SBATCH --export=NONE

ulimit -s unlimited

# Set location and environment for Intel libraries
INTEL_LIBS=$VCHOME/lib/intel

```

```

INTEL_MKL_LIBS=$INTEL_LIBS/mkl
INTEL_MPI_LIBS=$INTEL_LIBS/mpi

LD_LIBRARY_PATH=$INTEL_LIBS:$INTEL_MKL_LIBS:$INTEL_MPI_LIBS
export LD_LIBRARY_PATH

PATH=$INTEL_MPI_LIBS:$PATH
export PATH

# Set number of OpenMP threads to run per MPI process
export OMP_NUM_THREADS=1
export MKL_NUM_THREADS=1
export I_MPI_PIN_DOMAIN=omp
export KMP_STACKSIZE=16m

# Set VM2 executable to use
VC_FORTRAN_EXE=$VCHOME/exe/VC_CompChemPackage_mpi.exe

# Set VM2 input and output file names
VC_IN_FILE=protein_ligand_vm2.inp
VC_OUT_FILE=protein_ligand_vm2.out

cd $PBS_O_WORKDIR

# Will run a total of 8 MPI processes on a single node
mpirun -n 8 $VC_FORTRAN_EXE $VC_IN_FILE $VC_OUT_FILE

```

5.4.2. Example 8 MPI processes, 2 OpenMP threads per MPI process, 1 GPU per MPI process, using SLURM

```

#!/bin/bash
# Modify to match local setup
# This script has not been tested
#SBATCH --partition=default
#SBATCH --job-name=test
#SBATCH --ntasks=16
#SBATCH --ntasks-per-node=16
#SBATCH --output=test.out
#SBATCH --error=test.err
#SBATCH --export=NONE

ulimit -s unlimited

# Designed to run on a node equipped with dual xeon 8 core processors
# and 4 K80 Tesla cards (8 GPUs)

# Set location and environment for Intel and CUDA libraries
INTEL_LIBS=$VCHOME/lib/intel
INTEL_MKL_LIBS=$INTEL_LIBS/mkl
INTEL_MPI_LIBS=$INTEL_LIBS/mpi

CUDA_LIBS=$VCHOME/lib/cuda:$VCHOME/lib/magma

LD_LIBRARY_PATH=$INTEL_LIBS:$INTEL_MKL_LIBS:$INTEL_MPI_LIBS:$CUDA_LIBS
export LD_LIBRARY_PATH

PATH=$INTEL_MPI_LIBS:$PATH

```

```

export PATH

# Set number of OpenMP threads to run per MPI process
export OMP_NUM_THREADS=2
export MKL_NUM_THREADS=2
export I_MPI_PIN_DOMAIN=omp
export KMP_STACKSIZE=16m

# Set VM2 executable to use
VC_FORTRAN_EXE=$VCHOME/exe/VC_CompChemPackage_mpi_openmp_cuda.exe

# Set VM2 input and output file names
VC_IN_FILE=protein_ligand_vm2.inp
VC_OUT_FILE=protein_ligand_vm2.out

# Runs 8 MPI processes with 2 OpenMP threads per processes.
# Also uses 8 GPUs, all on a single node.
mpirun -n 8 $VC_FORTRAN_EXE $VC_IN_FILE $VC_OUT_FILE
-----

```

5.5. LSF batch queue scripts

Below are LSF batch queue scripts equivalent to the PBS scripts in 5.3. above. To submit a calculation, where the LSF script is called my_lsf_script.sh, issue the command

```
bsub < my_lsf_script.sh
```

5.5.1. Example 8 MPI process LSF run

```

-----
#!/bin/bash
#
# LSF batch script to run an MPI application
#
#BSUB -P project_code      # project code
#BSUB -W 48:00             # wall-clock time (hrs:mins)
#BSUB -n 8                 # number of tasks in job
#BSUB -R "span[ptile=1]"   # run 8 MPI tasks per node
#BSUB -J job_name          # job name
#BSUB -o job_name.%J.out   # output file name in which %J is replaced
#                          by the job ID
#BSUB -e job_name.%J.err   # error file name in which %J is replaced
#                          by the job ID
#BSUB -q queue_name        # queue

ulimit -s unlimited

# Set location and environment for Intel libraries
INTEL_LIBS=$VCHOME/lib/intel
INTEL_MKL_LIBS=$INTEL_LIBS/mkl
INTEL_MPI_LIBS=$INTEL_LIBS/mpi

LD_LIBRARY_PATH=$INTEL_LIBS:$INTEL_MKL_LIBS:$INTEL_MPI_LIBS
export LD_LIBRARY_PATH

PATH=$INTEL_MPI_LIBS:$PATH
export PATH

```

```

# Set number of OpenMP threads to run per MPI process
export OMP_NUM_THREADS=1
export MKL_NUM_THREADS=1
export I_MPI_PIN_DOMAIN=omp
export KMP_STACKSIZE=16m

# Set VM2 executable to use
VC_FORTRAN_EXE=$VCHOME/exe/VC_CompChemPackage_mpi.exe

# Set VM2 input and output file names
VC_IN_FILE=protein_ligand_vm2.inp
VC_OUT_FILE=protein_ligand_vm2.out

# Will run a total of 8 MPI processes on a single node
mpirun -n 8 $VC_FORTRAN_EXE $VC_IN_FILE $VC_OUT_FILE

```

5.5.2. Example 8 MPI processes, 2 OpenMP threads per MPI process, 1 GPU per MPI process, using LSF

```

#!/bin/bash
#
#BSUB -a poe                                # set parallel operating environment
#BSUB -P project_code                       # project code
#BSUB -J hybrid_job_name                   # job name
#BSUB -W 48:00                             # wall-clock time (hrs:mins)
#BSUB -n 16                               # number of tasks in job
#BSUB -R "span[ptile=8]"                  # run eight MPI tasks per node
#BSUB -q regular                          # queue
#BSUB -e errors.%J.hybrid                 # error file name in which %J is replaced
by the job ID
#BSUB -o output.%J.hybrid                 # output file name in which %J is
replaced by the job ID

ulimit -s unlimited

# Designed to run on a node equipped with dual xeon 8 core processors
# and 4 K80 Tesla cards (8 GPUs)

# Set location and environment for Intel and CUDA libraries
INTEL_LIBS=$VCHOME/lib/intel
INTEL_MKL_LIBS=$INTEL_LIBS/mkl
INTEL_MPI_LIBS=$INTEL_LIBS/mpi

CUDA_LIBS=$VCHOME/lib/cuda:$VCHOME/lib/magma

LD_LIBRARY_PATH=$INTEL_LIBS:$INTEL_MKL_LIBS:$INTEL_MPI_LIBS:$CUDA_LIBS
export LD_LIBRARY_PATH

PATH=$INTEL_MPI_LIBS:$PATH
export PATH

# Set number of OpenMP threads to run per MPI process
export OMP_NUM_THREADS=2
export MKL_NUM_THREADS=2
export I_MPI_PIN_DOMAIN=omp
export KMP_STACKSIZE=16m

```

```

export MP_TASK_AFFINITY=core:$OMP_NUM_THREADS

# Set VM2 executable to use
VC_FORTRAN_EXE=$VCHOME/exe/VC_CompChemPackage_mpi_openmp_cuda.exe

# Set VM2 input and output file names
VC_IN_FILE=protein_ligand_vm2.inp
VC_OUT_FILE=protein_ligand_vm2.out

# Runs 8 MPI processes with 2 OpenMP threads per processes.
# Also uses 8 GPUs, all on a single node.
mpirun -n 8 $VC_FORTRAN_EXE $VC_IN_FILE $VC_OUT_FILE

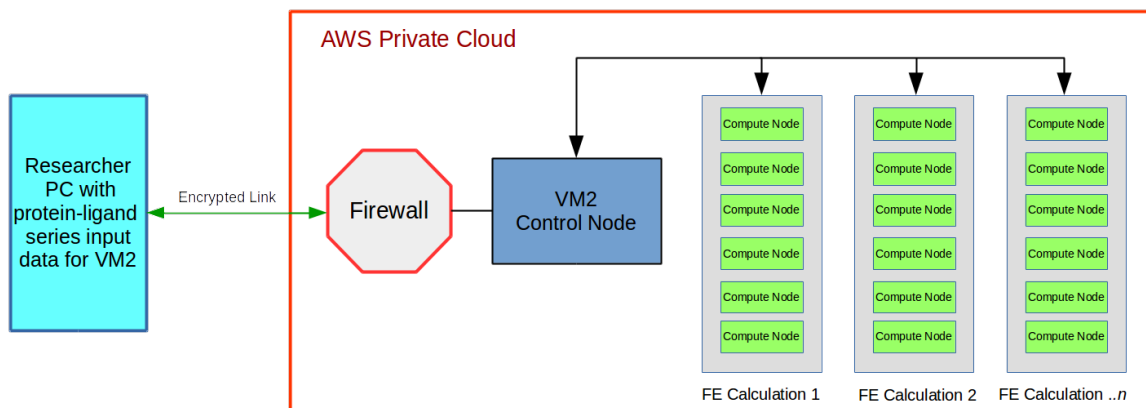
```

6. CloudVM2 – running VM2 on Amazon Web Services (AWS) cloud environment

CloudVM2 has three main components: a GUI front-end program which can be installed on the user's machine or in the AWS cloud, a private cloud network with firewall on the AWS platform, and compute nodes running VM2 calculations. The front-end program starts the private cloud network and the compute nodes and then distributes calculations to them.

6.1. Outline of CloudVM2 operation

The user will have already selected a receptor (e.g. protein or host molecule) and one or possibly a whole series of ligands for which they need to calculate the binding free energy. They will run the CloudVM2 front-end program, which will then contact the Amazon Web Services cloud, and create a firewalled private cloud. The program will then temporarily transmit the data files to Amazon's S3 data service. Once data has been uploaded the GUI will start a compute node for each receptor, ligand, and each receptor-ligand pair. Once the node is operational it will download the appropriate data files and delete them from temporary storage. The node will then start the calculations. Upon completion of the calculations, the node uploads the results back to S3 and is then terminated, stopping costs accruing for that node. The user can then download the results at their leisure via the GUI.



6.2. Architecture and economics

CloudVM2 can take advantage of Amazon's 'spot instances', which represent spare capacity on the AWS cloud that is sold to the highest bidder. The spot instances achieve an average savings of 75% (summer 2017) compared to normal 'on-demand' instances, the drawback to this is that AWS reserves the right to terminate a spot instance with 2 minutes warning, so that they may sell it to another customer who is willing to pay the full retail price. The CloudVM2 architecture is built to accommodate this with minimal loss of efficiency. CloudVM2 continually monitors the running instance for the termination signal, and when received it uploads the last calculation checkpoint file to fast local storage (S3) so that the calculation can be restarted. Automatic calculation restarts are planned for the next release of CloudVM2.

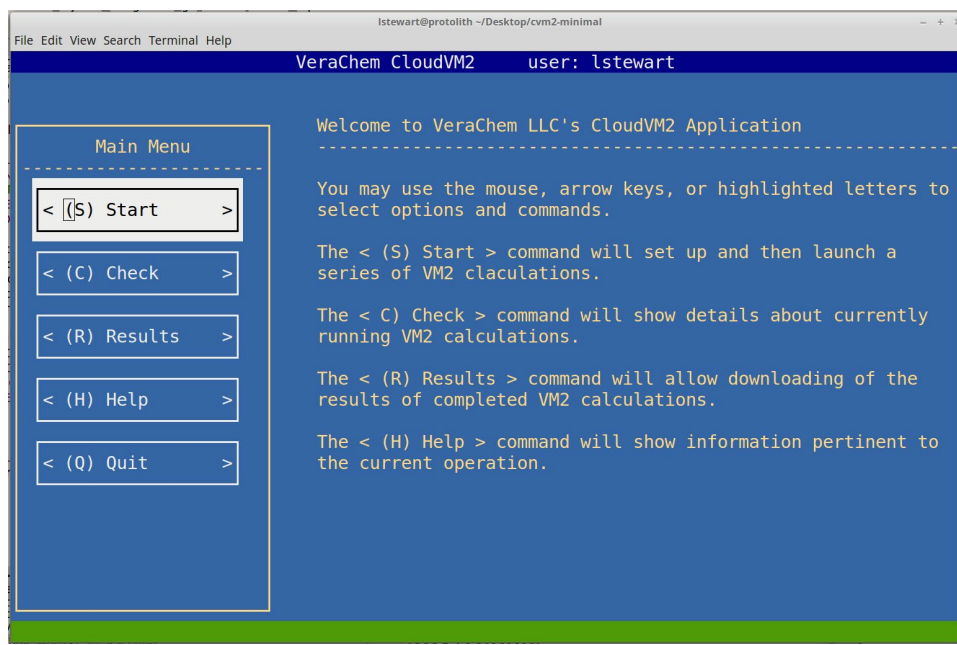
Amazon Web Services is built on a worldwide infrastructure, with 15 major datacenters (increasing rapidly), each of which is divided into two or more 'Availability Zones'. Each datacenter and each zone have different spot prices for each instance type, and these prices fluctuate continuously. CloudVM2 is aware of Amazon's worldwide infrastructure, and scans all datacenters and all zones for current and historic spot prices, and will then launch computations in the datacenter and zone with the most economical predicted total cost. Data storage for each calculation is also located in the same datacenter for the quickest and most economical operation.

6.3. CloudVM2 GUI

The CloudVM2 GUI can be run locally on a system with Python 2.7 installed, or it can be hosted on a small, low cost instance in the AWS cloud. The CloudVM2 GUI has three main functions: start a series of calculations, check status of compute nodes, and retrieve results.

6.3.1. Main Menu

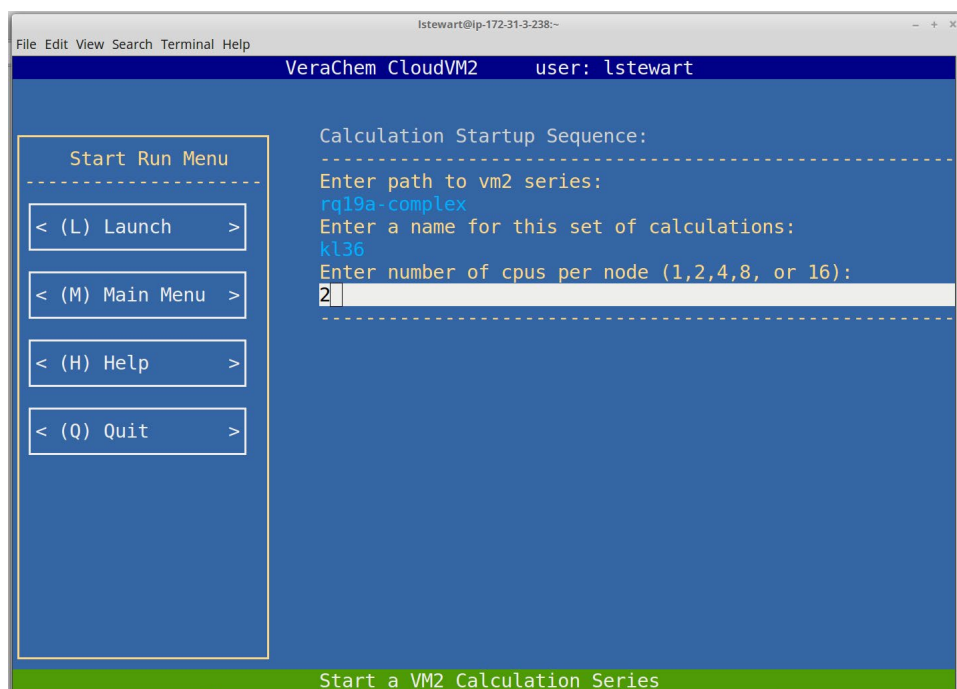
The opening screen in CloudVM2 displays some helpful information and the main menu. Other than help info, no functionality is accessible from this screen. Depending upon the



user's platform and terminal emulation application, CloudVM2 menu selections and entries can be accessed by 'hotkeys' (single letter shortcuts), mouse clicks, or arrow buttons. Not all terminal programs support all methods, but at the very least the 'hotkeys' should work.

6.3.2. Start Menu

The start menu allows the user to launch a VM2 calculation series on AWS. The user must supply the path to the top-level directory of their series, relative to the user's home directory. The user must also select a name for the series, and what size of node to run each calculation on.



After selecting <(L) Launch>, CloudVM2 will download price data from AWS (this may take a couple minutes) and then start launching computational nodes sequentially. Information on the status of the launch will scroll down in the right hand data column.

```

File Edit View Search Terminal Help
VeraChem CloudVM2 user: lstewart

Start Run Menu
-----
< (L) Launch >
< (M) Main Menu >
< (H) Help >
< (Q) Quit >

rq19a_pd56a results will be saved in the collection:
lstewart-kl36-cpx-07-13-2017-1851-ap-northeast-2
job upload successful
Starting Cluster: rq19a_pd56a

Spot request `sir-2gr8nj2j` fulfilled for instance
rq19a_pd56a
With instance: i-0bbf001ad8c5ea9be

Waiting for instance rq19a_pd56a to be running...
rq19a_pd56a Compute Node Launched

rq19a_pd59 Will be Launched in Zone: ap-northeast-2c
current: $0.0339 bid: $0.1695
rq19a_pd59 results will be saved in the collection:
lstewart-kl36-cpx-07-13-2017-1851-ap-northeast-2
job upload successful
Starting Cluster: rq19a_pd59

Spot request `sir-yfb8mbwh` fulfilled for instance
rq19a_pd59
With instance: i-0d62380bedbb0b4c6

Waiting for instance rq19a_pd59 to be running...
Start a VM2 Calculation Series

```

6.3.3. Check Menu

The check menu shows the user the current state of that user's calculation nodes. Functionality to shutdown nodes launched in error will be included in the next release of CloudVM2.

```

File Edit View Search Terminal Help
VeraChem CloudVM2 user: lstewart

Instances Menu
-----
< (M) Main Menu >
< (H) Help >
< (Q) Quit >

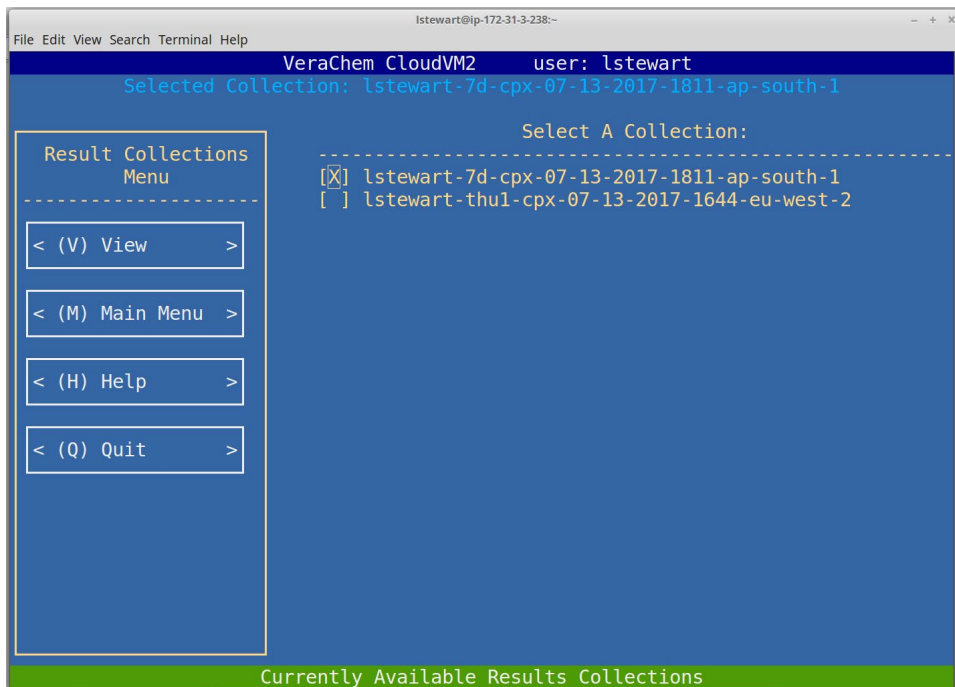
Virginia Datacenter
Name      Type      up(hrs)    load(% cpu)
South Korea Datacenter
Name      Type      up(hrs)    load(% cpu)
[ ] rq19a_pd56a    c4.xlarge    3.22      100.1
[ ] rq19a_pd58    c4.xlarge    3.20      100.2
[ ] rq19a_pd59    c4.xlarge    3.21      100.1
India Datacenter
Name      Type      up(hrs)    load(% cpu)
[ ] rq17_pp5      c4.large     3.87      100.2
[ ] rq17_pp4      c4.large     3.87      100.1
[ ] rq17_pp7b     c4.large     3.88      100.0

Details of Currently Running Calculations

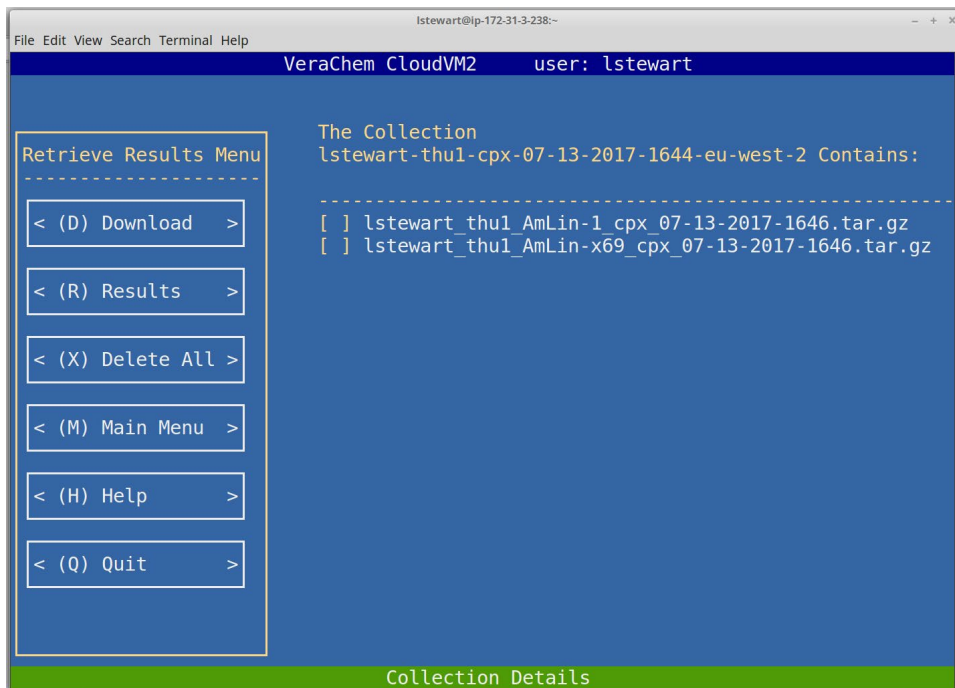
```

6.3.4. Retrieve Menu

The Results menu allows a user to inspect, download, and delete the results of their calculation series. After selecting a collection with the mouse, space, or enter keys, the user may select <View> to see the contents of the collection. From there the user may



download the collection or delete it.



7. Front end workflow

7.1. General scheme for ligand series and receptor binding

7.2. Local clusters

7.3. CloudVM2

VII. VM2 output files

The VM2 method produces a wealth of information regarding the chemical system under study. In addition to the free energy G (Boltzmann averaged over all conformers), and its constituent energy $\langle E \rangle$ and entropic terms $-TS$, it provides a breakdown of $\langle E \rangle$ into internal potential energy $\langle U \rangle$ and solvation energy $\langle W \rangle$ terms, with $\langle U \rangle$ further broken down into molecular mechanics terms (bond, angle, torsion, vdW, and nonbonded energies). Furthermore, values for each individual conformer are available as well as molecular coordinates.

1. VeraChem standard output files

1.1. Verbose output file (.out)

Contains detailed description of all steps of the calculation.

```
simon -- webbs@granite:~/umass_run_examples/complexes/ad32_complex_movetoxtalad81 -- ssh -- ...
-----
Single-mode search number      64
-----

Process Rank:      7

Unique Conformer ID:      64

Attempt a driver rotation of 180.0 degrees.

Driver Information : Number      32
                  : ID          20
                  : Direction    1
                  : Force Constant 36.2

Distortion Information : Energy before -3104.8516135149 kcal/mol
                     : Energy after  -3050.7225451215 kcal/mol
                     : Steps          10
                     : Size          180.0 Degrees

Done with distortion. Now carry out geometry optimization ...

GB Quasi-Newton Geometry Optimization.
-----

In Vacuo energy      : -1713.3682920693 Kcal/mol
GB Solvation energy  : -1386.6922858270 Kcal/mol
GB Energy            : -3100.0605778964 Kcal/mol

RMS gradient         : 0.0012792440 Kcal/mol/Angs
Max gradient         : 0.0098695853 Kcal/mol/Angs
Total geometry steps : 2356

Structure accepted. Taking a closer look.

-----
Single-point energy.
-----

GB energy of this structure -3100.0605778964 kcal/mol
Difference w.r.t current lowest energy 4.7910356185 kcal/mol

Tighten geometry convergence criterion and re-minimize ...

GB Quasi-Newton Geometry Optimization.
-----

In Vacuo energy      : -1713.6124770655 Kcal/mol
GB Solvation energy  : -1386.3884447156 Kcal/mol
GB Energy            : -3100.0009217811 Kcal/mol

RMS gradient         : 0.0001600908 Kcal/mol/Angs
Max gradient         : 0.0009501323 Kcal/mol/Angs
Total geometry steps : 782

Structure accepted. Taking a closer look.

-----
Single-point energy.
-----

GB energy of this structure -3100.0009217811 kcal/mol
Difference w.r.t current lowest energy 4.8506917337 kcal/mol

FBSA energy of this structure -3077.0301396668 kcal/mol
Difference w.r.t current lowest energy -2.0005470316 kcal/mol

... found new lowest energy minimum.

-----
Single-mode search number      65
-----

Process Rank:      0

Unique Conformer ID:      65

Attempt a driver rotation of -180.0 degrees.
```

4624,0-1 36%

1.2. Summary output file (.summary.out)

Contains a basic summary of the run, including energy tables.

```
simon -- webbs@granite:~/umass_run_examples/complexes/ad32_complex_movetotxlad81 -- ssh -- ...

Final VM2 Free Energy is -1291.419351 Kcal/mol.

-----
Final Individual Conformer Data.
-----

List data for conformers with probability of at least : 0.01 %
Rankings are with respect to : Free Energy (G)

-----
Free Energy: G = E - TS
-----
```

Rank	ID	ln(Zx)	Prob(%)	Sum Prob(%)	Del G	G	E	-TS
1	163	745.52045	63.16	63.16	0.00	-1291.15	-2427.25	1136.11
2	80	744.47510	22.21	85.37	0.62	-1290.52	-2427.02	1136.50
3	39	743.96542	13.34	98.71	0.93	-1290.22	-2424.97	1134.75
4	85	740.67480	0.50	99.21	2.89	-1288.26	-2423.00	1134.74
5	35	740.27189	0.33	99.54	3.13	-1288.02	-2423.33	1135.31
6	157	739.82707	0.21	99.75	3.39	-1287.75	-2423.99	1136.24
7	145	739.65651	0.18	99.93	3.50	-1287.65	-2425.47	1137.82
8	17	737.89165	0.03	99.96	4.55	-1286.60	-2423.95	1137.36
9	129	737.02705	0.01	99.97	5.06	-1286.08	-2425.51	1139.43
10	177	736.99942	0.01	99.99	5.08	-1286.07	-2422.50	1136.43
11	84	736.81050	0.01	100.00	5.19	-1285.95	-2421.02	1135.07

```
-----
Potential Energy: E = (U+W) + H.O. Equipartion energy
-----
```

Rank	ID	E	U+W	U	W(PBSA)	PB	SA
1	163	-2427.25	-3077.37	-1704.94	-1372.42	-1416.87	44.45
2	80	-2427.02	-3077.14	-1713.63	-1363.48	-1407.88	44.40
3	39	-2424.97	-3075.08	-1709.16	-1365.92	-1410.59	44.66
4	85	-2423.00	-3073.11	-1697.62	-1375.49	-1420.10	44.60
5	35	-2423.33	-3073.44	-1707.89	-1365.55	-1410.22	44.67
6	157	-2423.99	-3074.11	-1711.97	-1362.13	-1406.70	44.56
7	145	-2425.47	-3075.58	-1730.92	-1344.66	-1389.05	44.39
8	17	-2423.95	-3074.07	-1714.06	-1360.01	-1404.55	44.54
9	129	-2425.51	-3075.63	-1717.86	-1357.77	-1402.14	44.37
10	177	-2422.50	-3072.61	-1731.79	-1340.85	-1385.23	44.38
11	84	-2421.02	-3071.13	-1706.48	-1364.65	-1409.31	44.66

```
-----
Gas Phase Internal Potential Energy: U
-----
```

Rank	ID	U	Valence	Coulomb	VdW	VdW6	VdW12
1	163	-1704.94	603.87	-1893.29	-415.51	-1523.93	1108.42
2	80	-1713.63	602.43	-1898.24	-417.82	-1525.18	1107.36
3	39	-1709.16	591.09	-1889.54	-410.70	-1516.26	1105.36
4	85	-1697.62	592.39	-1881.00	-409.01	-1515.31	1106.30
5	35	-1707.89	591.18	-1890.54	-408.52	-1516.31	1107.78
6	157	-1711.97	594.15	-1897.16	-408.96	-1517.91	1108.95
7	145	-1730.92	602.62	-1920.76	-412.78	-1526.05	1113.27
8	17	-1714.06	592.91	-1896.51	-410.46	-1526.08	1115.61
9	129	-1717.86	603.83	-1903.08	-418.62	-1535.18	1116.56
10	177	-1731.79	604.66	-1918.68	-417.77	-1526.60	1108.82
11	84	-1706.48	592.76	-1888.98	-410.26	-1515.95	1105.69

```
-----
Bonded Energy Terms: Valence
-----
```

Rank	ID	Valence	Bond	Angle	Pdih	Idih	Tether
1	163	603.87	25.85	76.74	496.90	4.37	0.00
2	80	602.43	26.05	77.03	494.96	4.40	0.00
3	39	591.09	26.09	77.15	484.47	3.39	0.00
4	85	592.39	25.88	76.77	486.38	3.36	0.00
5	35	591.18	26.00	76.55	485.21	3.41	0.00
6	157	594.15	26.15	77.45	487.16	3.38	0.00

241,11 648

1.3. Binary restart file (.vcbn)

At the conclusion of each VM2 iteration a binary restart file is written out to disk.

Occasional crashes or downtime for maintenance can unexpectedly interrupt running calculations and waste large compute resources already devoted to a particular run. In addition, sometimes a run may not converge within the default maximum iterations and will need to be restarted at the last iteration carried out. To handle these fairly common situations, during Phase II we implemented a binary file restart capability for the parallel VM2 software. For each VM2 iteration a binary file is updated with energy and molecular coordinate data. If a run unexpectedly stops the binary file can be read in and

the calculation restarted at the last performed VM2 iteration. If a run does not converge the binary file may be read in and additional iterations requested; again the calculation starts at the last performed iteration in the original run.

An option to read in a binary file from a VM2 run (either incomplete or complete) and output the users choice of formatted text data files (see previous sections) was also implemented.

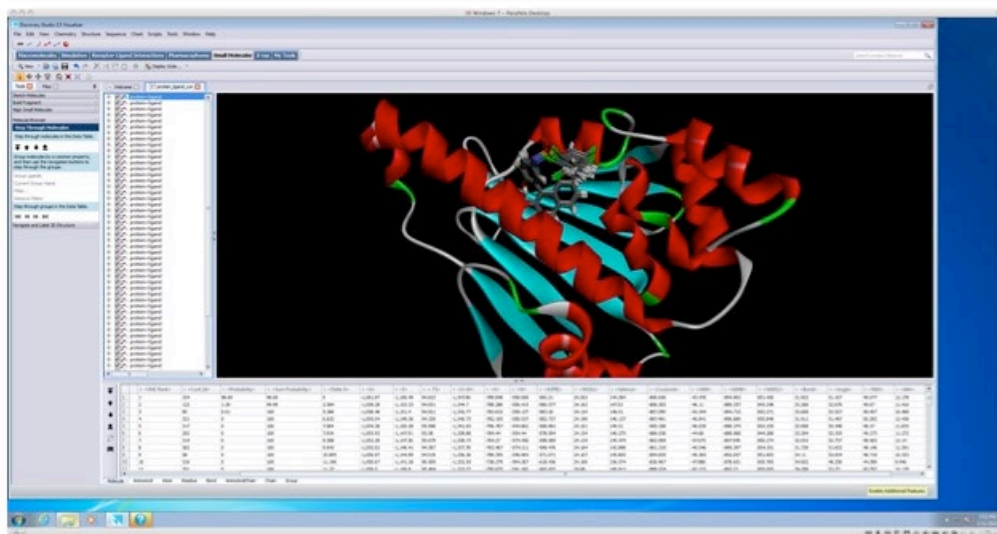
2. Formatted output files

In order that users can explore this data conveniently using the many available (free and commercial) molecular visualization packages, we have continued to expand the range of formatted output files our VM2 software can output.

2.1. Structural data

- .xyz Standard cartesian coordinate file
- .pdb Protein Data Bank format
- .crd Standard CHARMM card format
- .dat Macromodel Structure File format
- .sdf Structure Data File format
- .mol2 Tripos molecular data file format
- .gms Template GAMESS input file for each conformer
- .g09 Template GAUSSIAN09 input file for each conformer

The .mol2 and .sdf formats include data for molecular visualization as well as energy data. For example, a .mol2 file read into the freely available Discovery Studio Visualizer provides a way to look at a set of VM2 generated conformers as well as their energy breakdowns:



[illegible]

.csv Comma separated values file containing calculated energy data

		A		B		C		D		E		F		G		H		I		J		K		L		M		N		O		P		Q		R		S		T		U		V		W		X		Y		Z																																																																																																																																																																																																																										
		1		2		3		4		5		6		7		8		9		10		11		12		13		14		15		16		17		18		19		20		21		22		23		24		25		26		27		28		29		30		31		32		33		34		35		36		37		38		39		40		41		42		43		44		45		46		47		48		49		50		51		52		53		54		55		56		57		58		59		60		61		62		63		64		65		66		67		68		69		70		71		72		73		74		75		76		77		78		79		80		81		82		83		84		85		86		87		88		89		90		91		92		93		94		95		96		97		98		99		100		101		102		103		104		105		106		107		108		109		110		111		112		113		114		115		116		117		118		119		120		121		122		123		124		125		126		127		128		129		130		131		132		133		134		135

3. Back end workflow

3.1. Generation of binding affinity tables

3.2. Placeholder

VIII. Input file run options reference

VeraChem Computational Chemistry Package Input Options.

Sections are as follows:

1. Choice of system type and calculation type and other top-level control.
 2. Molecular system definition options for protein macromolecules.
 3. Molecular system definition options for host molecules e.g. cyclodextrins.
 4. Molecular system definition options for ligand molecules.
 5. Math related options e.g. control of random seed generation.
 6. VeraChem mining minima (VM2) calculation options.
 7. General conformational search control options.
 8. Custom conformational search options.
 9. Options and control of spatial boundary based conformer rejection.
 10. Options for free energy processing of conformers.
 11. Stereochemistry checking and enforcement control.
 12. Control of filtering out conformer repeats.
 13. Options for molecular alignment and RMSD calculation.
 14. Geometry optimization options and control, including constraints.
 15. Molecular mechanics potential energy calculation: methods and usage control.
 16. Generalized Born (GB) solvation model control.
 17. Constant dielectric (CD) solvation model control.
 18. Distance dependent (DD) dielectric solvation model control.
 19. Poisson Boltzmann Surface Area (PBSA) solvation model control.
-

1. Choice of System Type and Calculation Type and Other Top Level Control.

molSystemType

Choose the type of molecular system. There is no default; this option must be given. See below for additional input required dependent on this choice.

‘protein’	Protein receptor calculation (could include explicit water, ions, etc.). Part of the system must be fixed in space (see Section 2).
‘host’	Host molecule calculation. These should be ‘small’ receptor systems of a few hundred atoms or less e.g. cyclodextrins.
‘ligand’	Ligand calculation; for example, a ‘drug like’ small molecule.
‘protein+ligand’	Protein-ligand complex.
‘host+ligand’	Host-guest complex.

calcnType

Choose type of calculation to be carried out. There is no default; this option must be given. All calculation types can be initiated with one or multiple input conformers.

‘vm2’	VeraChem Second-generation mining minima (VM2) free energy calculation.
‘feprocess’	Free energy processing of one or multiple conformers supplied by the user.
‘confsearch’	Conformational search (potential energy only).
‘rmsd’	Structural comparison of read-in conformers.
‘filter’	Filter out repeats contained in read-in conformers.
‘geomopt’	Geometry optimization.
‘geomoptHatoms’	Optimize positions of just hydrogen atoms. Only allowed for molSystemType ‘protein’ and ‘protein+ligand’.
‘energy+grad’	Single-point energy and gradient.

‘energy’ Single-point energy.

timeLimit

Time limit for calculations given in wall clock hours. Currently only relevant for calcnType ‘vm2’. The program terminates cleanly and outputs all data files when the limit is projected to be reached in the next phase of a calculation. The default is 96.0 hours.

readInConfs

Optionally read in molecular conformations (one or more) from a text file or multiple text files to initiate a calculation. The text file formats may be **.xyz**, **.sdf**, Macromodel **.dat**, or **.crd**. This option may be used, for example, to read in a previously generated ensemble of ligand conformations to generate initial protein-ligand conformations, or simply to read in previously generated ensemble of protein-ligand conformations. If this option is not used a single starting conformation is taken from the input **.crd** coordinates – see Sections 2-4.

The readInConfs option may be given up to a maximum of **three** times, providing multiple types of conformer ensembles. For each instance of readInConfs multiple conformer source files may be read in. The program automatically makes appropriate combinations of conformer types read-in. For example, if molSystemType is ‘protein+ligand’ and if ‘complex’, ‘protein’, and ‘ligand’ conformer ensembles are read-in, the ‘complex’ conformers are taken as is and all unique combinations of the ‘protein’ and ‘ligand’ ensembles make additional ‘protein+ligand’ start conformers. The maximum number of start conformations is 1000. The program makes sensible truncations if the conformer files provided result in more.

‘complex’	Formatted file(s) containing protein-ligand or host-guest conformers.
‘protein’	Formatted file(s) containing only protein conformers.
‘host’	Formatted file(s) containing only host molecule conformers.
‘ligand’	Formatted file(s) containing only ligand conformers.

ligandConfsToCrd

Only relevant when using the readInConfs option to read in ‘ligand’ conformers. Controls how, if at all, read-in ligand conformers are superimposed on the ligand input **.crd** coordinates. (Note that the input **.crd** coordinates themselves can be moved *prior* to this by superimposition on template coordinates – see Section 4.)

'no'	Use the coordinates of the ligand conformers as read-in. This is the default .
'byConf1COG'	Translate the center of geometry (COG) of the first ligand conformer read-in to the COG of the ligand .crd . Apply the same translation to all subsequent ligand conformers read-in.
'byConfsCOG'	Translate the COG of each ligand conformer read-in to the COG of the ligand .crd .
'byConf1All'	Carry out a rotation/translation superposition of all heavy atoms (non hydrogens) of the first ligand conformer read-in on the corresponding ligand .crd atom positions. Apply the same rotation/translation to all subsequent ligand conformers read-in.
'byConfsAll'	Carry out a rotation/translation superposition of all heavy atoms (non hydrogens) of the each ligand conformer read-in on the corresponding ligand .crd atom positions.
'byConf1PairsMap'	Carry out a rotation/translation superposition of the first ligand conformer read-in with the ligand .crd coordinates using the atom indexes provided on the very next line. Apply the same rotation/translation to all subsequent ligand conformers read-in e.g. byConf1PairsMap 3 5 18 21 22 23
'byConfPairsMap'	Carry out a rotation/translation superposition of each ligand conformer read-in with the ligand .crd coordinates using the atom indexes provided on the very next line e.g. byConfsPairsMap 3 5 18 21 22 23

useCrdAsTemplate

Only relevant when using the readInConfs option to read in 'complex' conformers plus another type of conformer (e.g. 'protein', 'host', or 'ligand') and molSystemType is protein+ligand or host+ligand (i.e. a complex). Controls whether to use the **.crd** input coordinates (see Sections 2-4) as a template for generation of complex conformers ('yes') or whether to use the coordinates of the first 'complex' conformer read-in as a template ('no').

'yes'

'no' This is the **default**.

useCrdAsConf

Only relevant when using the readInConfs option. Controls whether to use the **.crd** input coordinates (see Sections 2-4) as a starting conformation in addition to the ones generated through readInConfs. Note that if readInConfs option is not used the **.crd** coordinates are *always* used to define a single starting conformation.

‘yes’ This is the **default**.

‘no’

outputFormats

Choose any number of the following file formats. Currently **.xyz** and **.pdb** formats are always output in addition to those chosen. Place one per line directly following the keyword with no blank lines.

‘sdf’ A structure-data file (SDfile) with standard V2000 or V3000 molfile formatting.

‘mol2’ Tripos mol2 file.

‘dat’ Macromodel data file.

‘csv’ Comma-separated-values file containing energy data.

‘gms’ Basic template input files for the GAMESS electronic structure software package.

‘g09’ Basic template input files for the Gaussian09 software package.

fullEnergyBreakdown

Requests that for output of **.sdf** and **.csv** files a full breakdown of the energy into constituent terms is written out. If ‘no’ is selected a limited number of constituent energy terms are output.

‘yes’ This is the **default**.

‘no’

splitOutputFormats

Mostly relevant for molSystemType ‘protein+ligand’ and ‘host+ligand’. The same as outputFormats above, but a separate formatted file is output for each of the molecules comprising the complex. Currently **.crd** files are always output in addition to those chosen, even for non-complexes. The base-name for the split output files is taken from the input **.crd** file names; a descriptor is added based on the calculation type e.g. xxxxx.vm2.sdf, xxxxx.vm2_rank1.crd. Place one output format type per line directly following the keyword with no blank lines.

‘sdf’ A structure-data file (SDfile) with standard V2000 or V3000 molfile formatting.

‘xyz’ Standard xyz file format.

limitConfsToOutput

The way that the number of conformers written to the formatted output files is limited can be chosen using this keyword.

‘byCount’ The user sets the maximum number of conformers to be output. Follow this line directly with an integer. This is the **default** with a maximum number of conformers set as **1000**.

‘byPopulation’ The user sets the maximum cumulative conformer population that limits the number of conformers output. Follow this line with a percentage value e.g. 99.9. Note that this option only makes sense for calcnType’s ‘vm2’ and ‘feprocess’.

atomsToOutput

This is relevant for systems that include proteins as not all the atoms are required to be present in calculations, and not all atoms present are mobile.

‘all’ All atoms are included in the formatted output. This is the **default**.

‘real’ Only ‘real’ atoms are included in the formatted output. (Real atoms are those atoms that are included in the energy calculation; however, they are not necessarily free to move.)

‘live’ Only live (flexible) atoms are included in the formatted output.

binaryFileRestart

Restart a calculation from a VeraChem binary data file. The binary file has the suffix **.vcbin**. The program expects the base name of the binary restart file to have the same base name of the **.inp** file.

'crashed'	Use when calculation quits unexpectedly. This option is currently only available for calcnType 'vm2'.
'extendRun'	Use for carrying out additional iterations of a calculation that finished, but, for example, did not converge. This option is currently only available for calcnType 'vm2'.
'reprocess'	Uses the conformations produced from a prior run as a starting point, but reprocesses them for energies, carrying out a geometry optimizations as necessary, and proceeds with the requested calculation. The user can change the energy potential (e.g. different solvation model) from the original run if desired. This option is currently only available for calcnType 'vm2'.
'textOutput'	Read a VeraChem binary data file and output the data as formatted text files (see outputFormats above.) This option is currently only available for calcnType 'vm2'.

Example usage 1

```
#
molSystemType
protein+ligand
#
calcnType
vm2
#
timeLimit
48.0
#
readInConfs
ligand
ligand_confs.xyz
#
outputFormats
sdf
csv
#
limitConfsToOutput
byPopulation
99.9
```

#

2. Molecular System Definition Options for Protein Macromolecules

Relevant for molSystemType 'protein' and 'protein+ligand'.

inputProtein	Names of input files containing protein system data and real/live set definition related data. They are mandatory and must be given in order with no blank lines.
1	Signifies protein molecule one. A single protein molecule is the current limit.
~/path/protein_name.crd	Starting coordinates, atom names, residue names etc. Files must conform to standard .crd format (regular or extended).
~/path/protein_name.top	Topology and molecular mechanics parameters. See Section XII for format specification.
~/path/protein_name.mol	Provides protein molecule bond orders and stereocenter information. File must be standard V2000 or V3000 mol format.
setChainIds	<p>If present controls relabeling of protein chain and residue Ids given in the .crd file. Requires that the very next line contain an integer, or integers, corresponding to the count(s) of the last residue of each newly defined chain. Optionally the next line can provide the new chain Ids. If this second line is not present the defaults are A, B, C, ... and so on. E.g.</p> <pre>setChainIds 99 198 199 A B C</pre>
constructLiveReal	Controls how the protein real/live set is defined i.e. the protein atoms that are included in the energy calculation (real), and which atoms are also allowed to move in the calculation (live). The live set is a subset of the real set. This keyword is mandatory .

'readIn'	Read in a formatted text file that defines the protein real/live set. See Section XII for format specification. The name of the file must be provided on the very next line e.g. readIn ~/path/protein_real_live.txt
'byTemplateCOGs'	Read in a template molecule's atomic coordinates, from a .crd , .xyz , .sdf , .mol , .pdb , or Macromodel .dat formatted file, distances to this molecules center of geometry (COG) will define the protein real/live set. For example, use co-crystalized ligand coordinates. The name of the file must be provided on the very next line e.g. byTemplateCOG ~/path/template_real_live.crd
'byTemplateAtoms'	Read in a template molecule's atomic coordinates, from a .crd , .xyz , .sdf , .mol , .pdb , or Macromodel .dat formatted file, distances to which will define the protein real/live set. For example, use co-crystalized ligand coordinates. The name of the file must be provided on the very next line e.g. byTemplateAtoms ~/path/template_real_live.crd
'byXYZ'	Cartesian coordinates to be used as a reference point to define the protein real/live set. The coordinates must be provided on the very next line e.g. byXYZ 3.2345 5.7941 9.7745

The following are relevant for the constructLiveReal choices 'byTemplateCOG', 'byTemplateAtoms', and 'byXYZ'

realCutoffDist	The default is 9.0 Angstroms. This cutoff is residue based. The distance is from any protein atom to any template molecule atom for option 'byTemplate' or to a single user defined point for option 'byXYZ'. Any residue with an atom within this distance is 'real' i.e. its atoms are included in the energy calculation, but are not necessarily mobile.
liveCutoffDist	The default is 7.0 Angstroms. This cutoff is atom based. The distance is from any protein atom to any template molecule atom for option 'byTemplate' or to a single user defined for option 'byXYZ'. Any atoms within this

distance are 'live' i.e. mobile. They are subset of the 'real' set.

symmetrizeRealSet

If 'yes' multiple chains are present and are symmetric, based on exact matching of residue and atom names between chains, residues will be added to real set as necessary to make it symmetric.

'yes'

'no' This is the **default**.

symmetrizeLiveSet

If multiple chains are present and are symmetric, based on exact matching of residue and atom names between chains, atoms will be added to live set as necessary to make it symmetric.

'yes'

'no' This is the **default**.

Example usage 2

```
-----  
#  
inputProtein  
1  
~/path/protein_name.crd  
~/path/protein_name.top  
~/path/protein_name.mol  
#  
constructLiveReal  
readIn  
~/path/protein_real_live.txt  
#  
-----
```

Example usage 3

```
-----  
#  
inputProtein  
1  
~/path/protein_name.crd  
~/path/protein_name.top  
~/path/protein_name.mol
```

```
#
constructLiveReal
byTemplateAtoms
~/path/template_real_live.crd
#
realCutoffDist
8.0
#
liveCutoffDist
6.0
#
```

3. Molecular System Definition Options for Host Molecules

Relevant or molsystemType 'host' and 'host+ligand'.

inputHost	Names of input files containing host molecule data. They are mandatory and must be given in order with no blank lines. The program checks they are present by examination of their suffixes.
1	Signifies that names of formatted data files for host molecule 1 will follow. Currently, one 'molecule' is the limit; however, a system comprising two hosts could still be run by including the data for both host molecules in each file.
~/path/host_name.crd	Starting coordinates, atom names, etc. Files must conform to standard .crd format (regular or extended).
~/path/host_name.top	Topology and molecular mechanics parameters. See Section XII for format specification.
~/path/host_name.mol	Provides host molecule bond orders and stereocenter information. File must be standard V2000 or V3000 mol format.

Example usage 4

```
#
inputHost
1
~/path/host_name.crd
~/path/host_name.top
~/path/host_name.mol
```

#

=====

4. Molecular System Definition Options for Ligand Molecules

Relevant or molSystemType 'protein+ligand' and 'host+ligand' and 'ligand'.

inputLigand	Names of input files containing host molecule data. They are mandatory and must be given in order with no blank lines.
1	Signifies that names of formatted data files for ligand molecule 1 will follow. Currently, one ligand molecule is the limit.
~/path/ligand_name.crd	Starting coordinates, atom names, etc. Files must conform to standard .crd format (regular or extended).
~/path/ligand_name.top	Topology and molecular mechanics parameters. See Section XII for format specification.
~/path/ligand_name.mol	Provides ligand molecule bond orders and stereocenter information. File must be standard V2000 or V3000 .mol format.
placeLigandMethod	Controls how, if at all, the ligand will be moved from the .crd starting coordinates given above before the start of a calculation by placement relative to a user supplied position in space or template set of coordinates. (Note: Calculation of center of geometry (COG) excludes hydrogen atoms, as does the least squares fit for superpositions.) The moved ligand coordinates then redefine what the 'input' .crd coordinates are.
'none'	The ligand is not moved from the starting coordinates defined in .crd above. This is the default .
'byReceptorCOG'	Only relevant for molSystemType's 'protein+ligand' and 'host+ligand'. The receptor's (protein or host) center of geometry (COG) is used as a reference point that the ligand COG is translated to.

'byXYZ'	<p>Cartesian coordinates to be used as a reference point that the ligand center of geometry (COG) is translated to, and the <i>very</i> next line after that must contain the Cartesian coordinates, e.g.</p>
.	<pre>byXYZ 3.2745 5.7654 9.7653</pre>
'byTemplateCOG'	<p>Read in a template molecule, .crd, .xyz, .sdf, .mol, .pdb, or Macromodel .dat format, and use its center of geometry (COG) as a reference point that the ligand COG is translated to. For this option the <i>very</i> next line must contain the name of a formatted file containing the template e.g.</p>
	<pre>byTemplateCOG ~/path/template_molecule.xyz</pre>
'byTemplateAll'	<p>Read in a template molecule, .crd, .xyz, .sdf, .mol, .pdb, or Macromodel .dat format, and superimpose all heavy atoms of the template onto the ligand atoms. The template should be a conformer of the same ligand defined by the starting coordinate .crd file above, with atoms in the same order. For this option the <i>very</i> next line must contain the name of a formatted file containing the template e.g.</p>
	<pre>byTemplateAll ~/path/template_conformer.sdf</pre>
'byTemplatePairsMap'	<p>Read in a template molecule, .crd, .xyz, .sdf, .mol, .pdb, or Macromodel .dat format, and superimpose the ligand by chosen pairs of atoms to map onto each other. For this option the <i>very</i> next line must contain the name of a formatted file containing the template, the following line must contain the template atom indexes for use in superposition, and the subsequent line must contain the corresponding ligand atom indexes e.g.</p>
	<pre>byTemplatePairsMap ~/path/template_molecule.crd 7 8 9 10 11 12 13 3 5 11 15 19 20 21</pre>
doSnapTemplatePairs	<p>If 'yes' a harmonic potential (see below) is applied to the ligand atoms defined by the 'byTemplatePairsMap' setting above, but at the position of the template atoms. This</p>

guides/snaps the chosen ligand atoms to the template positions during conformational searches/geometry optimizations. Only relevant when placeLigandMethod option 'byTemplatePairsMap' is used.

'yes'

'no'

This is the **default**.

snapTemplatePairsFC

Relevant when doSnapTemplatePairs is 'yes'. Sets the harmonic potential force constant. The default value is 2.0 Kcal/mol/Angs.

Example usage 5

```
-----
#
inputLigand
1
~/path/ligand_name.crd
~/path/ligand_name.top
~/path/ligand_name.mol
#
placeLigandMethod
byTemplateCOG
~/path/template_molecule.xyz
#
=====
```

5. Math Related Options.

randomSeedsMethod

Choose method to generate seeds for the KISS random number generator. Random number generation is required for various stochastic algorithms in the VeraChem computational chemistry package.

'byWallClock'

Uses wall clock timing data combined with process ID data to automatically generate a different set of seeds every run. Note that for parallel runs a different seeds are produced for each process, but only the master process's set is written to output files. This is the **default**.

'byUser'

The seeds are supplied by the user (see below). This option must be used if deterministic parallel processor runs are required.

setRandomSeeds

For 'byUser' option above include this keyword and supply four integers in the following four lines.

Example usage 6

```
-----  
#  
randomSeedsMethod  
byUser  
#  
setRandomSeeds  
9759  
9850  
7072  
203  
#  
-----
```

6. VeraChem Mining Minima VM2 Calculation Options.

Relevant for calcnType 'vm2'.

convTolVm2

Specifies the free energy difference between VM2 iterations that signifies convergence. At least 3 iterations must have been carried out and the free energy must have gone down compared to the last 2 iterations. The **default** is 0.01 Kcal/mol.

maxVm2Iters

Specifies the maximum number of VM2 iterations to be carried out before quitting whether converged or not. The **default** is 60.

Example usage 7

```
-----  
#  
convTolVm2  
0.001  
#
```

maxVm2Iters

30

#

7. General Conformational Search Control Options.

Relevant for calcnType 'vm2' and 'confsearch'.

The VeraChem conformational search capability comprises various vibrational mode-distort-minimize types as well as rigid body translation-rotation distort-minimize algorithms. The 'canned' search styles use various combinations of these algorithms suitable for specific chemical system-based search demands. For fine control of these algorithms a 'custom' search may be requested (see Section 9).

Iteration and convergence control: only relevant for calcnType option 'confsearch'.

convTolConfsearch

Specifies the potential energy difference between confsearch iterations that signifies convergence. At least 3 iterations must have been carried out and the potential energy must have gone down compared to the last 2 iterations. The **default** is 0.01 Kcal/mol.

maxConfsearchIters

Specifies the maximum number of confsearch iterations to be carried out before quitting whether converged or not. The **default** is 60.

Search methods control: relevant for calcnType options 'vm2' and 'confsearch'.

confSearchStyle

Specifies the style of conformational search to be carried out. **Note:** See Section 9 for default ligand box constraint settings associated with confSearchStyle settings.

- | | |
|------------|---|
| 'standard' | Requests the standard single-mode based sampling of conformational space. The quickest 'canned' search style, but will not consistently find the lowest energy conformers of a system, so use with caution. |
| 'enhanced' | Requests an enhanced sampling of conformational space. In addition to the single-mode based sampling, search drivers built from random combinations of pairs of single modes |

are used. Usually appropriate when the approximate pose/position of the ligand is known – for example by superposition on a ligand with the same scaffold that was co-crystallized with the receptor. This is the **default**.

‘rigorous’	Requests a rigorous sampling of conformational space. Useful when the active/binding site is known, but the receptor and/or ligand itself may be quite flexible with large R groups etc. As well as single-mode and random-pair-modes searches, it includes searches using focused drivers where fewer torsions are included in each driver, but distortions tend to be more pronounced.
‘vrigorous’	Requests a very rigorous sampling of conformational space. Useful when the active/binding site is known, but nothing is known about the pose and position of the ligand in the active/binding site. Large translations and rotations are included in the search as well as mode distortions.
‘confgen1’	This setting is designed solely to generate a diverse set of conformations for starting points in other calculations. It carries out only one vm2/confsearch iteration and uses stricter than default filtering and expanded energy cutoff to achieve diversity of structures as opposed to energy convergence.
‘confgen2’	Relevant for molSystemType ‘ligand’ only. The same process as ‘congen1’ above, but in addition the resulting conformers are rotated about their 3 principal axes 180 degrees. The 4-fold expanded set of conformers then have some orientational as well conformational diversity.
‘confgen3’	Placeholder – ongoing implementation.
‘confgen4’	Relevant for molSystemType ‘ligand’ only. The same process as ‘congen1’ above, but in addition a maximum of 20 of the resulting conformers are randomly rotated about their 3 principal axes between 0 and 360 degrees to generate 1000 final conformations. This provides large orientational diversity. For use when no information on the ligand pose is known.
‘custom’	All search methods and parameters can be finely controlled according to the user’s choice. Combinations of the many available conformational search options can be employed. Recommended for expert users who want detailed control of the search procedures. See custom search control parameters in Section 8 below.

confGenLengthSort

Only relevant for molSystemType 'ligand' calculations with confSearchStyle 'confgen1', 'confgen2', and 'confgen3'. If 'yes' ligand conformers are sorted according to their length (longest first) before any rotomers are generated and conformers output.

'yes' This is the **default**.

'no'

maxSearches

The maximum number of searches for each mode-distort-minimize search type strung together to form the search style. The **default** is 400. This may be automatically adjusted downwards for small systems. It may also be automatically adjusted for MPI parallel runs for load balancing.

modeRotnMax

The maximum rotation angle for a mode distortion.
The **default** is 180.0 (degrees).

switchToRandomRotnMax

The 'vm2' or 'confsearch' iteration at which the maximum rotation angle for mode distortions is randomly chosen from the range modeRotnMax/2 to modeRotnMax. The **default** is 7.

numRlsearch

The number of random ligand fixed-body translation-rotation searches to be carried out. Only relevant when a 'vrigorous' search style is requested or when a random ligand rotation/translation search is requested through the custom search option. The **default** is 24.

ligandTranMax

The maximum ligand fixed-body translation distortion length.
The **default** is 2.0 (Angstroms).

ligandRotnMax

The maximum angle for ligand fixed-body rotation distortions.
The **default** is 180.0 (degrees).

excludeBackBone

Only relevant for systemType 'protein' and 'protein+ligand'. If 'yes' the protein

backbone atoms are excluded from drivers for conformational searches; if 'no' the protein backbone atoms are included in mode-distort conformational searching. Note that regardless, live (mobile) backbone atoms are always included in geometry optimizations after mode distortions.

'yes' This is the **default**.

'no'

excludeSideChains

Only relevant for systemType 'protein' and 'protein+ligand'. If 'yes' the protein sidechain atoms are excluded from drivers for conformational searches; if 'no' the protein sidechain atoms are included in mode-distort conformational searching. Note that regardless, live (mobile) sidechain atoms are always included in geometry optimizations after mode distortions.

'yes'

'no' This is the **default**.

excludedAtomsFile

Optionally specify a text file that provides a list of atoms to be excluded from drivers for conformational searches. See Section XII for format.

~/path/file_name_excluded_atoms.txt

forceConstCutoff

Mode drivers with force constants larger than this cutoff are excluded from the mode search. The **default** is 5000.0.

deltaLevel1Cutoff

Relevant when there is a level 2 correction to the level 1 energy e.g. single -point energy with PBSA solvation model at geometry determined with GB solvation model. For level 1 energy differences between the lowest energy conformer and the conformer just found that are greater than this cutoff, the level 2 energy correction is skipped and the current conformer discarded. The **default** is 20.0 Kcal/mol.

nonBlockingUpdate

This keyword is only relevant for MPI multi-processor runs. If 'yes', non-blocking sends and receives are used to communicate low energy structures between MPI processes every 'vm2' or 'confsearch' iteration; if 'no', blocking collective operations are used, which can result in large latencies.

‘yes’	This is the default for systemType ‘protein’, ‘protein+ligand’, ‘host’, and ‘host+ligand’.
‘no’	This is the default for systemType ‘ligand’.

doLoadBalance

This keyword is only relevant for MPI multi-processor runs. If ‘yes’, the MPI process that finishes its assignment of searches first in each ‘*vm2*’ or ‘*confsearch*’ iteration signals all other processes to proceed when their current mode distort-minimize is complete. This results in some skipped searches, but improves load balancing considerably.

‘yes’	This is the default for systemType ‘protein’, ‘protein+ligand’, ‘host’, and ‘host+ligand’.
‘no’	This is the default for systemType ‘ligand’.

mixSearchBasis

This keyword and the following four related ones are only relevant for MPI multi-processor runs. Periodically, multiple conformers are used as a basis for independent (i.e. decoupled) conformational searching, with no communication between MPI processes. This adds diversity to the conformational search. The number of conformer starting structures equals the number of MPI processes. (see mixSearchPicks below).

Integer 0, 1 to 4	0	sets this option as off
	1	Use multiple conformers every call to the conformational search i.e. every <i>vm2</i> or <i>confsearch</i> iteration.
	2	Use multiple conformers every second <i>vm2/confsearch</i> iteration. This is the default .
	3	Use multiple conformers every third <i>vm2/confsearch</i> iteration.
	4	Use multiple conformers every fourth <i>vm2/confsearch</i> iteration.

mixSearchIters

Relevant if concurrent conformer searching is on (i.e. if mixSearchBasis above is not 0). Sets the *vm2/confsearch* iteration above which concurrent searching is completely switched off. The **default** is 20.

mixSearchPicks

Controls how the group of conformers is selected for the ‘mixSearchBasis’ approach.

‘inorder’	Select N conformers in order of their free energy as the set of conformers to search on, where N is the number of MPI processes.
‘random1’	Select the first N/2 conformers in order, then pick an additional N/2 at random from all the remaining conformers.
‘random2’	Select the first N/2 conformers in order, then pick an additional N/2 at random from the next poolSize – N/2 conformers in order of their free energy. See below for poolSize. This is the default .
‘cluster’	Select the first N/2 conformers in order, then cluster the remaining conformers starting at N/2 + 1 with an RMSD cutoff of 0.5 Angstroms. Pick the lowest energy conformer of each cluster up to N MPI processes. If not enough clusters present select from the lowest energy conformer up again (to double search the low energy conformers).

doClusterBy

Controls whether clustering (mixSearchPicks ‘cluster’ option) is based on RMSDs of the whole molecule system or a component. For example, for a protein+ligand complex the clustering can be set as based solely on the ligand RMSDs.

‘complex’	The default if molSystemType is ‘protein+ligand’ or ‘host+ligand’.
‘receptor’	The only option if molSystemType is ‘protein’ or ‘host’. Can also be selected for ‘protein+ligand’ or ‘host+ligand’ runs.
‘ligand’	The only option if molSystemType is ‘ligand’. Can also be selected for ‘protein+ligand’ or ‘host+ligand’ runs.

poolSize

For mixSearchPicks option ‘random2’ option, sets the size of the pool of conformers that are picked from at random. The **default** is 64. For the first iteration of a VM2 run when starting conformers are read in (see Section 1.) the default is quadrupled to allow a more diverse search basis. For ‘random1’ and ‘cluster’ options it is hardwired as all available conformers; for option ‘inorder’ it

is hardwired as the number of MPI processes.

relaxNonDriverAtoms

If 'yes', when carrying out distortions along drivers, non-driver atoms are allowed to relax after each distortion step via a few geometry optimization cycles (driver atoms are kept fixed during these cycles). If 'no' is selected all non-driver atoms are kept fixed in space during distortions. Note that enforcing rigidity during driver distortions will speed up the search, but will invariably result in extremely high energies for small driver distortions limiting the conformational space sampled.

'yes' This is the **default**.

'no'

Example usage 8

```
-----  
#  
confSearchStyle  
vrigorous  
#  
maxSearches  
200  
#  
numRlsearch  
48  
#  
excludedAtomsFile  
~/path/file_name_excluded_atoms.txt  
#  
mixSearchBasis  
2  
#  
mixSearchPicks  
random2  
#  
-----  
  
=====
```

8. Custom Conformational Search Options.

Relevant for calcnType 'vm2' and 'confsearch'.

Use these options when keyword confSearchStyle is set to 'custom'.

Search

Choose the type of search to be carried out.

- 'mode' Initiates a search using distortions along mode based drivers followed by geometry optimization. The nature of the mode-based search can be further controlled by the options below. This is the **default**.
- 'ligand' Initiates a ligand based search where the ligand is translated, and/or rotated followed by a geometry optimization of the system. The ligand based search can be further controlled by the options described below.
- 'combined1' Requests a mode based search followed immediately by a ligand based search.
- 'combined2' Requests a ligand based search followed immediately by a mode based search.

modeSearch

Choose the type of mode search to be carried out.

- 'normal' A standard mode search with distortions along drivers weighted according to mode coefficients. This is the **default**.
- 'focused' A more robust mode search with more focused and larger distortions. This style of mode search cannot be applied to ligand only systems.
- 'combined1' Requests a standard mode search directly followed by a robust mode search i.e. 'normal' then 'focused'.
- 'combined2' Requests a robust mode search directly followed by a standard mode search i.e. 'focused' then 'normal'.

mode

For a 'normal' search (see above), choose how to determine geometry displacements i.e. drivers.

- 'single' Use individual modes only. This is the **default**.
- 'pair' Use a linear combination of randomly chosen pairs of modes (generated on the fly).
- 'combined1' Carry out a 'single' mode search directly followed by a 'pair' mode search.

‘combined2’ Carry out a ‘pair’ mode search directly followed by a ‘single’ mode search.

focusedSearch

For a ‘focused’ search (see above), choose ligand driven, receptor driven, or a combination of the two.

‘ligand’ Ligand driven focused search only. All receptor atom and any small ligand mode coefficients are zeroed out. Distortions are then focused on small groups of ligand atoms.

‘receptor’ Receptor driven focused search only. All ligand atom and any small receptor mode coefficients are zeroed out. Distortions are then focused on small groups of receptor atoms.

‘combined1’ Carry out a ‘ligand’ driven focused search directly followed by a ‘receptor’ driven focused search. This is the **default**.

‘combined2’ Carry out a ‘receptor’ driven focused search directly followed by a ‘ligand’ driven focused search.

ndrivers N Number of drivers N to select from the total available (only applicable to ‘single’ mode generated drivers).

-1 Select all available drivers i.e. N is set equal the total number of drivers generated. This the **default**.

drivers

Determines how the drivers are chosen or ordered.

‘largest’ Pick N drivers in order of the largest number of coefficients $> |0.1|$. This is the **default**.

‘random’ Randomly pick N drivers.

‘bottom’ Pick the N drivers with the smallest eigenvalues.

‘middle’ Pick N drivers from the middle range of eigenvalues.

‘top’ Pick the N drivers with the largest eigenvectors.

binRandomPairs

For searches with random pairs of modes if ‘yes’ the possible pair combination

are binned and the algorithm will pick equally from all the bins; if ‘no’ totally random pair combinations are used.

‘yes’ This is the **default** for host involved systems and ligand only systems.

‘no’ This is the **default** for protein involved systems.

modeDistMaxE

Specify the energy change cutoff for mode distortions. The **default** is 2000.0 (kcal/mol).

ligandSearch

Choose the type of ligand search to be carried out.

‘systematic’ Requests a systematic ligand search. Rotations of +/- ligandRotnMax/4, ligandRotnMax/2, and ligandRotnMax degrees (see ligandRotnMax, Section 7) and translations of +/- ligandTranMax/4, ligandTranMax/2, and ligandTranMax Angstroms (see ligandTranMax, Section 7) of the ligand about and along its principal axes are carried out in small steps. Between each step a few geometry relaxation steps are carried out for the receptor. Combined translation-rotations are also carried out giving a total of 80 searches per dimension searched. The number of dimensions searched is controlled by rligandSearch (see below). The preceding distances and angles are limits, and the rotation or translation is stopped at any step that results in an energy change greater than ligandDistMaxE (see below). After stopping each rotation or translation, a full geometry optimization is carried out.

‘random’ Requests a search involving random translations and rotations of the ligand along and about its principal axes. Rotation limits are +/- ligandRotnMax and translation limits are +/- ligandTranMax. The number of dimensions searched is controlled by rligandSearch (see below). Again, distortions are stopped if an energy change greater than ligandDistMaxE occurs. A geometry optimization is carried out after each distortion. The number of searches is controlled by numRlsearch (see Section 7 above).

‘combined1’ Requests a systematic ligand search directly followed by a random ligand search.

‘combined2’ Requests a random ligand search directly followed by a systematic ligand search.

sligandSearch

Number of dimensions in which to carry the systematic ligand search.

- ‘1d’ Rotation about the principal axis with the smallest principal moment of inertia, followed by full geometry optimization. Then translation along the same axis again followed by geometry optimization. Then translation-rotation along the same axis followed by geometry optimization. This is the **default**.
- ‘2d’ Carry out ‘1d’ rotations as above, then do the same for the axis with the second largest principal moment of inertia. Then move onto the translations, then onto translation-rotations.
- ‘3d’ All principal axes are tried in the same manner as above.

rligandSearch

Number of dimensions in which to carry the random ligand search plus control of the procedure.

- ‘1d’ Random translations and rotations along and about the principal axis with the smallest principal moment of inertia, followed by full geometry optimization. This is the **default**.
- ‘2d’ Carry out ‘1d’ as above, then do the same for the axis with the second largest principal moment of inertia i.e. separate geometry optimization for each axis trans/rots.
- ‘3d’ All principal axes are tried in the same manner as above.
- ‘comb2d’ Combines the random translations and rotations along and about two principal axes *before* the geometry relaxation step.
- ‘comb3d’ Combines the random translations and rotations along all principal axes *before* the geometry relaxation step.

ligandDistMaxE

Specify the energy change cutoff for ligand rotation/translation distortions. The **default** is 10000.0 (kcal/mol).

Example usage 9

Custom search settings that reproduce the confSearchStyle setting ‘vrigorous’ described above in Section 7.

```
-----  
#  
Search  
combined1  
#
```

```

modeSearch
combined1
#
mode
combined1
#
sdriver
1
#
ndrivers
-1
#
drivers
bottom
#
modeDistMaxE
2000.0
#
ligandSearch
combined1
#
sligandSearch
3d
#
rligandSearch
comb3d
#
ligandDistMaxE
10000.0
#

```

9. Options and Control of Spatial Boundary Based Conformer Rejection.

Relevant for calcnType 'vm2' and 'confsearch'.

These options allow conformers that do not fit the users predetermined geometric criteria to be discarded during a conformational search. They allow, for example, protein-ligand conformations where the ligand may have left the region of the known binding pocket to be discarded, or for conformers in which explicit water molecules that move too far away from a known crystallographic position to be discarded. These region-based exclusions can be used in conjunction with or be replaced by energy-based constraints applied during geometry optimizations (see Section 14).

boxedAtoms

integer1 integer2 integer3

An integer or list of integers that specifies an atom or atoms (other than ligand atoms) to apply a spherical boundary to; for example, an explicit water molecule oxygen atom. The center of geometry of the atoms in the list is only allowed to move in a sphere of specified dimension (see below), if it moves outside the sphere the conformation is rejected. Atoms on the list are also fixed in space during mode distortions. The reference center is defined by the input **.crd** coordinates of specified atoms. This option may be given up to twenty times i.e. the spherical box ‘constraint’ may be applied to twenty separate groups of atoms. Each spherical box may apply to a maximum of 200 atoms.

atomBoxSize

Specify the radius of the sphere that the ‘boxedAtoms’ center of coordinates must remain in. The **default** is 1.0 (Angstroms). If the ‘boxedAtoms’ center of coordinates moves outside this sphere the conformation is rejected.

ligandBoxSize

Specify the radius of the sphere in Angstroms that the ligand center of coordinates must remain in. If the ligand center of coordinates moves outside this spherical box the conformer is rejected. The reference center is defined by the input **.crd** coordinates of the ligand. To turn this filter off set as -1.0. The **default** is -1.0 (off) for molSystemType ‘host+ligand’. For all other molSystemTypes, the **default** radius depends on the confSearchStyle: for ‘custom’, ‘standard’, and ‘enhanced’ it is 1.0 Angstroms; for ‘rigorous’ it is 2.0 Angstroms; for ‘vrigorous’ it is 4.0 Angstroms.

Example usage 10

```
-----  
#  
boxedAtoms  
32 35  
#  
atomBoxSize  
2.0  
#  
ligandBoxSize  
2.0  
#  
-----  
  
=====
```

10. Options for Free Energy Processing of Conformers.

Relevant for calcnType 'vm2' and 'feprocess'.

modeScanning Allows the mode scanning step in the calculation of the configuration integral to be turned on or off.

 'on' This is the **default**.

 'off'

temperature Temperature in Kelvin used in the calculation of configurational integrals. The **default** is 300.00.

freeEnergyPreFactor

 Control which atoms are used in the calculation of the free energy prefactor. Only relevant for protein involved calculations.

 'useLiveAtoms' Use only the 'live' atoms.

 'useRealAtoms' Use all 'real' atoms. This is the **default**.

Example usage 11

```
-----  
#  
modeScanning  
off  
#  
temperature  
273.15  
#  
-----  
  
=====
```

11. Stereochemistry Checking and Enforcement Control.

Relevant for calcnType 'vm2', 'confsearch', 'feprocess', and 'geomopt'.

maintainCisTrans

 If 'yes' cis/trans arrangements across double bonds are enforced by rejecting conformers where isomerization has occurred; if set as 'no' cis/trans isomerization is allowed. Double bonds are as identified by the bond orders given in the input mol/sdf file; Cis/trans arrangements across double bonds are identified automatically.

‘yes’ This is the **default**.

‘no’

maintainParity

If ‘yes’ R/S stereocenters are enforced by rejecting conformers where stereoisomerization has occurred. If set as ‘no’ stereoisomerization is allowed. R/S stereocenters are as defined in the input mol/sdf file.

‘yes’ This is the **default**.

‘no’

maintainProteinPepBonds

Control the stereochemistry of protein peptide bonds by rejecting generated conformers that violate the chosen option.

‘asInput’ The stereochemistry of protein peptide bonds are maintained as they are in the user provided input structure. This is the **default**.

‘asTrans’ An attempt will be made to flip any cis protein peptide bonds found in the input structure and all peptide bonds will then be maintained as trans. **This option is not yet functional.**

‘no’ Protein peptide bond isomerization is allowed.

Example usage 12

```
-----  
#  
maintainCisTrans  
yes  
#  
MaintainParity  
yes  
#  
MaintainProteinPepBonds  
asInput  
#  
-----
```

12. Control of Filtering Out Conformer Repeats.

Relevant for calcnType 'vm2', 'confsearch', 'feprocess', 'rmsd', and 'filter'.

These parameters set energy difference cutoffs and geometry RMSD cutoffs that control how similar two conformers have to be for one of them to be designated a repeat and discarded. Additionally, energy parameters that control the culling of 'high energy' conformers can be set.

preFilterCalcnType

Choose type of calculation to be carried out prior to filtering. Only relevant for calcnType 'filter'.

'geomopt'	Geometry optimization. This is the default .
'energy+grad'	Single-point energy and gradient.
'energy'	Single-point energy.
'none'	No calculation before filtering.

pairCutoff1

Used in the filtering conformers either read in or resulting from a conformational search that *have not* undergone free energy processing. It is the bonded-term-energy difference below which a pair of conformers will be geometrically compared. The **default** for calcnType 'vm2' is 0.5 Kcal/mol; for calcnType's 'filter', 'rmsd', 'confsearch', the **default** is 2.0 Kcal/mol.

pairCutoff2

Used in the filtering conformers either read in or resulting from a conformational search that *have* undergone free energy processing (relevant for calcnType's 'vm2' and 'feprocess'). It is the bonded-term-energy difference below which a pair of conformers will be geometrically compared. The **default** is 1.0 Kcal/mol.

pairRmsdCutoff1

Used in the filtering conformers either read in or resulting from a conformational search that *have not* undergone free energy processing. It is the geometric RMSD lower than which the conformer with the higher potential energy is discarded. The **default** for calcnType 'vm2' is 0.2 Angstroms; for calcnType's 'filter', 'rmsd', and 'confsearch' the **default** is 0.3 Angstroms.

pairRmsdCutoff2	Used in the filtering conformers either read in or resulting from a conformational search that <i>have</i> undergone free energy processing. It is the geometric RMSD lower than which the conformer with the higher free energy is discarded. The default is 0.3 Angstroms.
firstConfCullE	Energy cutoff used for initial culls e.g. the first 2 VM2 iterations. Depending on the calculation type and status, it is the conformer potential energy or free energy relative to the current lowest energy conformer at which all higher energy conformers are discarded. The default is 20.0 Kcal/mol except for calcnType's 'filter' and 'rmsd' when the default is 100.0 Kcal/mol.
ConfCullE	Standard energy cutoff used for culling high energy conformers. Depending on the calculation type and status, it is the conformer potential energy or free energy relative to the current lowest energy conformer at which all higher energy conformers are discarded. The default is 10.0 Kcal/mol except for calcnType's 'filter' and 'rmsd' when the default is 100.0 Kcal/mol.
displaceCurrentConfs	<p>Only relevant for the molSystemType's 'protein' and 'protein+ligand'. If 'yes' during the filtering process a newly generated conformer found to be a repeat of a currently established conformer, which also has a lower energy (this energy difference will always be very small i.e. a fraction of a kcal/mol) will displace the currently established conformer. In some cases with this will lead to very small energy fluctuations between iterations and therefore very slow convergence, therefore the default is set as 'no'.</p> <p>'yes'</p> <p>'no' This is the default.</p>

Example usage 13

```

#
pairCutoff1
0.2
#
pairCutoff2
0.3
#
firstConfCullE

```

30.0

#

ConfCulle

20.0

#

13. Options for Molecular Alignment and RMSD Calculation.

Relevant for calcnType 'vm2', 'confsearch', 'feprocess', 'rmsd', 'filter', and 'geomopt'. For calcnType 'rmsd' a set of conformers must be read-in via the readInConfs keyword - see Section 1.

Currently alignment options are only relevant for molssystemType 'ligand', 'host', and 'host+ligand'. For molssystemType 'protein' and 'protein+ligand' no alignment will be carried out regardless of user input as protein real-fixed atoms are already exactly aligned and provide the reference position and orientation for the whole system.

The alignment options allow the conformations produced during the course of a particular calculation to be superimposed on the input conformation for output. The default for the molssystemType's listed above is for alignment to be turned on. Unless the user wants to specify the specific atoms to align, e.g. when there is a suitable ligand scaffold, the defaults picked by the program are usually appropriate.

preRmsdCalcnType

Choose type of calculation to be carried out prior to RMSD calculation. Only relevant for calcnType 'rmsd'.

'geomopt'	Geometry optimization. This is the default .
'energy+grad'	Single-point energy and gradient.
'energy'	Single-point energy.
'none'	No calculation before filtering.

preRmsdFilter

If 'yes' filter the read-in conformers before calculation of RMSD. Only relevant for calcnType 'rmsd'.

'yes'

‘no’ This is the **default**.

rmsdAllPairsMethod

Choose symmetry aware method to calculate and output the RMSD between *all* pairs of conformers that remain after any filtering. Only relevant for calcnType ‘rmsd’.

‘symaware1’ Basic fast symmetry aware algorithm. This is the **default**.

‘symaware2’ More sophisticated and expensive symmetry aware algorithm – see *J. Chem. Inf. Comput. Sci.* **44**, 1301-1313 (2004). Not available for molSystem ‘protein’ and ‘protein+ligand’

‘none’ Only RMSDs between the Rank 1 conformer and the rest are calculated using the basic symmetry aware method.

confAlignment

‘none’ Turn alignment off.

‘receptor’ The **default** for molSystemType ‘host’ and ‘host+ligand’ runs.

‘ligand’ The **default** for molSystemType ‘ligand’ runs.

‘selectatoms’ Indicates that the user will provide specific atoms to use for alignment.

numAlignAtoms Number of atoms the user will provide for alignment.

N

atomsToAlign Integers identifying which atoms to align.

integer1 integer2 integer3 interger4 ...

Example usage 14

```
#
confAlignment
selectatoms
#
numAlignAtoms
11
#
```

atomsToAlign

10 16 21 18 20 12 19 17 15 7 24

#

14. Geometry Optimization Options and Control, Including Constraints.

Relevant for calcnType 'vm2', 'confsearch', 'feprocess', and 'geomopt'.

The following control convergence criteria, geometry optimization methods, and maximum allowed geometry steps to achieve convergence.

maxAtomGrad	Standard convergence criterion. Used, for example, for calcnType 'geomopt' or 'feprocess' runs or for final geometries after mode distortion. It is the maximum absolute value gradient allowed of any individual mobile atom in the system. A second criterion is that the whole mobile system gradient RMSD must also be less than 1/3 of this parameter. The default is 0.001 (Kcal/mol)/Angstrom.
maxAtomGradLoose	Loose convergence criterion. Used, for example, for an initial geometry optimization after a mode distortion. It is the maximum absolute value gradient allowed of any individual mobile atom in the system. As above, the whole mobile system gradient RMSD must also be less than 1/3 of this parameter. The default is 0.01 (Kcal/mol)/Angstrom.
doPreoptSteps	Do some initial geometry steps before a first full geometry optimization is attempted. During pre-optimizations steps any atom gradients above 100.0 Kcal/mol/Angstrom or below -100.0 Kcal/mol/Angstrom are set to +/- 100.0 Kcal/mol/Angstrom are damped. This is useful for initial starting structures where there may be close contacts.
'yes'	Turn this option on. This is the default .
'no'	Turn this option off.
preoptMethod	Method to use for the pre-optimization geometry steps.
'1'	Quasi-Newton geometry optimization algorithm.
'2'	Conjugate-gradient geometry optimization algorithm. This is the default .

maxPreoptSteps	Maximum number of pre-optimization geometry steps. The default is 100.
geomoptMethod	Method to use for geometry optimization.
'1'	Quasi-Newton geometry optimization algorithm. This is the default .
'2'	Conjugate-gradient geometry optimization algorithm.
maxGeomoptSteps	Maximum number of geometry steps allowed for a geometry optimization. The default is 5000.
batchEnergyCutoff	This energy cutoff overrides the ConfCullE cutoffs in Section 12. The default is large so when the user supplies a wide range of conformers for geometry optimization less are discarded and can be examined via formatted output files. The default is 10000.0 Kcal/mol.

The following apply constraints to selected atoms in the system so they do not move far away from a desired position during a geometry optimization.

tetheredAtoms File that identifies atoms in the system that will be tethered. Multiple groups can be defined with each group being subject to different constraints defined by the harmonic and polynomial tether related keywords that follow below. The file name is arbitrary. See Section XII for format specification.

~/path/tethered_atoms_file.txt

tetherForceConstant

Specify a force constant if a harmonic constraint is required.

To specify a polynomial constraint the following three options with no blank lines are required to give the polynomial function $E(dr) = A*(dr/R)**n$.

tetherScalingFactor

Real number A

tetherDistance

Real number R

tetherOrder

Real number n

nfreezeAtoms Number of 'live' atoms to freeze in space during a geometry optimization by simply zeroing out their gradient. Currently, it is recommended that this option is not used for calcnType 'vm2' or 'feprocess'.

freezeAtoms List of integers that identify which atoms to freeze.

integer1 integer2 integer3 integer4

Example usage 15

maxAtomGrad
0.001

maxAtomGradLoose
0.01

doPreoptSteps
yes

preoptMethod
2

maxPreoptSteps
400

geomoptMethod
1

maxGeomoptSteps
10000

tetheredAtoms
~/path/tethered_atoms_file.txt

Constrained Group 1

tetherScalingFactor

```

100.0
tetherDistance
0.25
tetherOrder
12.0
#
# Constrained Group 2
#
tetherScalingFactor
1.0
tetherDistance
0.5
teherOrder
12.0
#
-----
=====

```

15. Molecular mechanics potential energy calculation: methods and usage control

level1mmMethod

Choose the method to treat mm solvation for energy derivative based calculations i.e. energy+grad calculations, geometry optimizations, and hessian calculations. Currently, straightforward use of the defaults is suggested. Control and selection of parameters for the methods themselves is described in Sections 16-19 below.

- ‘gb’ This is the **default**. Use a Generalized Born solvation method.
- ‘cd’ Use a constant dielectric solvation model.
- ‘dd’ Use distant dependent dielectric solvation model.

level2mmMethod

Choose the method to treat mm solvation for single-point energy corrections applied to, for example, any molecular geometries determined using level1mmMethod. For calcnType ‘energy’ and ‘energy+grad’ this single-point energy will be applied to the input structure(s). Control and selection of parameters for the methods themselves is described in Sections 16-19 below.

- ‘pbsa’ This is the **default**. Use the Poisson-Boltzmann Surface-Area (PBSA) solvation model.
- ‘none’ The PBSA energy correction will not be carried out. Only level 1 energies will be used.

allowZeroWaterLJ	Controls whether Lennard-Jones parameters for water hydrogen atoms will be allowed to be zero – as they are in OPLS.
‘yes’	Zero value parameters are allowed.
‘no’	Zero value parameters are not allowed and are replaced with TIP3P parameters. This is the default .
allowZeroLJ	Controls whether Lennard-Jones parameters for non-water hydrogen atoms will be allowed to be zero – as they are in OPLS for polar hydrogens.
‘yes’	Zero value parameters are allowed.
‘no’	Zero value parameters are not allowed and are replaced with: CHARMM/Dreiding: $\epsilon_i = -0.046$ $r_j^{min}/2 = 0.2245$ AMBER/GAFF: $\epsilon_i = -0.0157$ $r_j^{min}/2 = 0.6$ OPLS: $\epsilon_i = -0.03$ $r_j^{min}/2 = 0.2806$

This is the **default**.

mmAddFxdFxdConst

Controls whether the fixed-fixed real atom constant energy terms e.g. bond, angle, dihedral, improper, vdW, pure Coulomb (not GB solvation pairs) are calculated once at the start of a calculation and added as corrective constants throughout the calculation. Addition of these terms may facilitate energy comparisons with other programs.

‘yes’	Calculate the fixed-fixed constant energy terms. This is the default .
‘no’	Do not calculate the fixed-fixed terms.

Example usage 16

```
-----
#
level1mmMethod
gb
#
level2mmMethod
pbsa
#
```

16. Molecular mechanics Generalized Born (GB) solvation model control

gbSolvationModel

Choose the particular GB model used.

- ‘still97’ Use Still’s analytical method for calculating the approximate Born radii for use in the GB solvation energy expression. See Qiu, Hollinger, and Still, *J. Phys. Chem. A* **1997**, 101, 3005-3014. This is the **default**.
- ‘hawkins96’ Currently disabled due to ongoing reimplementation work.

still97ParamSet

Choose the P1-P5 scaling parameters for still97 GB solvation energy calculations.

- ‘still’ Use the original scaling parameters from *J. Phys. Chem. A* **1997**, 101, 3005-3014. This is the **default**.
- ‘gilson’ Use an alternative set of scaling parameters. See David, Luo, and Gilson, *J. Comput. Chem.* **2000**, 21, 295-309.

gbDielectricExt

External solvent dielectric used in the GB solvation model. The **default** value is 80.0, modeling bulk water.

gbDielectricInt

Internal (i.e. solute) dielectric used in the GB solvation model. The **default** value is 1.0.

gbCavityRadii

Choose the atomic cavity radii to use in the GB solvation model.

- ‘halfRmin’ Use $R_{min}/2$, where R_{min} is the force field Lennard-Jones parameter, except for hydrogen atoms bonded to hetero atoms, which are set to 1.15 Å, and covalently bound fluorine atoms, which are set to 2.00 Å. This is the **default**, with the only exception being CHARMM combined with ‘still97’ and still97ParamSet option ‘gilson’ (see ‘legacy’ option below).

'halfSigma'	Use $\sigma/2$, where σ is the force field Lennard-Jones parameter, except for hydrogen atoms bonded to hetero atoms, which are set to 1.15 Å, and covalently bound fluorine atoms, which are set to 2.00 Å.
'bondi'	Use the Bondi van der Waals radii. See Bondi, A., <i>JPC</i> 1964 , 68, 441.
'mbondi'	Use the modified Bondi radii. See Rizzo, Aynechi, Case and Kuntz, <i>J. Chem. Theory Comput.</i> 2006 , 2, 128-139.
'legacy'	Use $R_{\min}/2$, where R_{\min} is the force field Lennard-Jones parameter, except for hydrogen atom radii, which are all set to 1.20 Å. This is the default for gbSolvationModel 'still97' and still97ParamSet 'gilson'. Note: These are the radii used in all preceding versions of the VM2 software package i.e. version 2.1 and earlier, regardless of the force field and model.

Example usage 17

```
-----
#
gbSolvationModel
still97
#
still97ParamSet
still
#
gbCavityRadii
legacy
#
-----
```

17. Molecular mechanics constant (CD) dielectric solvation model control

cdSolventDielectric

Solvent dielectric constant used in the constant dielectric solvation model 'mm-cd'. The **default** value is 80.0.

18. Molecular mechanics distance dependent (DD) dielectric solvation model control

ddCoefficient

Coefficient used in the distance dependent dielectric solvation model ‘mm-dd’. The **default** value is 4.0 resulting in the so-called 1/4r method.

19. Molecular mechanics Poisson Boltzmann Surface Area (PBSA) solvation model control

pbDielectricExt

External solvent dielectric used in the PBSA solvation model. The **default** value is 80.0 modeling bulk water.

pbDielectricInt

Internal (i.e. solute) dielectric used in the PBSA solvation model. The **default** value is 1.0.

pbsaCavityRadii

Choose the atomic cavity radii to use in the PBSA solvation model. Currently the same radii are used for calculation of the electrostatic solvation energy (PB) and the non-polar solvation energy (SA). **Note:** If the ‘still97’/‘gilson’ GB solvation model is being used, to match GB and PBSA cavity radii the ‘legacy’ option below must be explicitly selected.

- | | |
|-------------|---|
| ‘halfRmin’ | Use $R_{min}/2$, where R_{min} is the force field Lennard-Jones parameter, except for hydrogen atoms bonded to hetero atoms, which are set to 1.15 Å, and covalently bound fluorine atoms, which are set to 2.00 Å. This is the default . |
| ‘halfSigma’ | Use $\sigma/2$, where σ is the force field Lennard-Jones parameter, except for hydrogen atoms bonded to hetero atoms, which are set to 1.15 Å, and covalently bound fluorine atoms, which are set to 2.00 Å. |
| ‘fitted’ | Use atomic cavity radii fitted to reproduce solvation energies determined using explicit TIP3P water molecules and the AMBER force field. See Tan, Yang, and Luo, <i>J. Phys. Chem. B</i> 2006 , <i>110</i> , 18680-18687. For GAFF atoms i.e. non-peptide atoms, ‘mbondi’ radii are used. |
| ‘bondi’ | Use the Bondi van der Waals radii. See Bondi, A., <i>JPC</i> 1964 , <i>68</i> , 441. |
| ‘mbondi’ | Use the modified Bondi radii. See Rizzo, Aynechi, Case and Kuntz, <i>J. Chem. Theory Comput.</i> 2006 , <i>2</i> , 128-139. |

'legacy' Use $R_{\text{min}}/2$, where R_{min} is the force field Lennard-Jones parameter, except for hydrogen atom radii, which are all set to 1.20 Å.
Note: These are the radii used in all preceding versions of the VM2 software package i.e. version 2.1 and earlier, regardless of the force field and model.

sasaProbeRadius

Set the solvent accessible surface area (SASA) probe radius. The **default** value is 1.4 Angstroms.

=====

IX. Ligand example

1. CHARMM pathway using Discovery Studio Visualizer (DSV)

1.1. Get mol2 data file for chosen molecule: ibuprofen

Step 1: Go to, for example, the ZINC database website <http://zinc15.docking.org> and perform a search for 'ibuprofen'.

Step 2: Placeholder

1.2. Load molecule into DSV

Step 1: Placeholder

2. CHARMM pathway using the web user interface CHARMMing

2.1. Get mol2 data file for chosen molecule: ibuprofen

Step 1: Go to, for example, the ZINC database website <http://zinc15.docking.org> and perform a search for 'ibuprofen'.

Step 2: Placeholder

2.2. Load the molecule

Step 1: Placeholder

X. Protein-ligand example: HIV-1 protease and 38 inhibitors

This is a full example of setup, execution of calculations, and collection of binding affinity results for a protein plus ligand series: the target protein is human HIV-1 protease and there are 38 ligands in the inhibitor series. (49)

NOTE: You will need a working installation of AmberTools with the \$AMBERHOME environment variable set to carry out the full procedure as described below. Please see <http://ambermd.org/> to download AmberTools and for its documentation.

To proceed, first, untar the examples file `vcCompChem_2_8_2_examples.tar.bz2`, which is provided with the package:

```
tar xvf vcCompChem_2_8_2_examples.tar.bz2
```

The main directory for this example is:

```
vcCompChem_2_8_2_examples/protein_ligand/hiv1_protease_series_1/
```

it contains a readme file: `README.hiv1p`, which describes the overall process, stepping through the following three directories in turn

```
hiv1_protease_series_1/setup
hiv1_protease_series_1/run
hiv1_protease_series_1/results
```

An outline of each step now follows. You can skip the setup section by going straight to [Section 2](#), and making use of the “-d reference” option, described in [Sections 2.1.2.](#) and [2.2.2.](#)

1. Setup

The procedure starts with setup, namely structure preparation, typing, charge assignment of the protein target molecule and ligand inhibitors, and assignment of mobile and fixed protein atoms.

1.1. Protein setup

The basis for this setup is the crystal structure of HIV-1 protease and the co-crystallized inhibitor AD-81. The PDB access code for this structure is 2I0D. The multiple aspects to consider when preparing a protein for molecular mechanics calculations starting from PDB coordinates are described in [Section V 3.1.](#) of this manual. Furthermore, the AMBER reference manual, available through the link given above, provides detailed advice for the use of AmberTools in this process - see the section titled “Preparing PDB Files”.

The files used for the following steps are found in the following subdirectory:

```
hiv1_protease_series_1/setup/protein
```

1.1.1. Remove all hetatoms and water atoms except atom 1580

For this particular receptor and set of inhibitors, it is important to explicitly include one of the water molecules (atom number 1580) present in the 2I0D crystal structure. Therefore, edit the `pdh` file `2i0d.pdb` deleting everything prior to the first ATOM entry, all HETATOM entries except for that of atom 1580, and everything except the END record after HETATOM 1580. Name the resulting file `2i0d_1580.pdb`.

1.1.2. Extract the co-crystallized ligand

The co-crystallized ligand in 2I0D is used as a reference structure, so copy and edit the original 2i0d.pdb file, deleting all atoms except the AD-81 ligand atoms, and rename the file ad_81_from_2i0d.pdb .

1.1.3. Prepare the PDB file for tleap

Prepare the pdb file for tleap by running the script run_pdb4amber_1.sh, i.e.

```
./run_pdb4amber_1.sh >& run_pdb4amber_1.log &
```

This will produce the file 2i0d_1580_p4a.pdb as well as other files required by tleap.

1.1.4. Run tleap to assign parameters

Run tleap to assign parameters using the script run_tleap_2.sh.

```
./run_tleap_2.sh >& run_tleap_2.log &
```

This will produce .incpcrd, .prmtop, .mol2, and .pdb files. These will be named 2i0d_1580_p4a_tleap.*

1.1.5. Convert .prmtop and .incpcrd to .crd, .top, and .mol files

Run the VeraChem amber pathway conversion tool prm2top.pyc using the script run_prm2top_3.sh, i.e.

```
./run_prm2top_3.sh >& run_prm2top_3.log &
```

This will produce the files 2i0d_1580_p4a_tleap_vm2.[crd,top,mol] These are the files that will be used to run the VM2 calculations.

Compare your results with those provided in the ./reference subdirectory to ensure that the procedure was successful.

1.2. Ligand Setup

Some remaining protein setup steps require that the AD-81 ligand be already setup, so next, the full set of ligands are prepared and parameterized. The relevant subdirectories are:

```
hiv1_protease_series_1/setup/ligands/source_files  
hiv1_protease_series_1/setup/ligands/vconf  
hiv1_protease_series_1/setup/ligands/prepare_ligands
```

1.2.1. Initial 2D structures

Processing with AmberTools requires an input sdf file containing the ligands in 3D, with all hydrogens present and stereochemistry properly defined with parity values. For this

example, the ligands were first drawn in 2D by a chemical draw program referencing figures from the published experimental binding affinity article.(49) A 2D mol file was saved for each ligand.

These 2D structures can be found in the ./source_files subdirectory of ligands/. A simple python script (mol_2_sdf.py) is used to assemble them into a single sdf file called umass_1.sdf.

```
python mol_2_sdf.py -o umass_1.sdf
```

To process only a chosen subset of the prepared 2D structures a key file can be used that contains the names of the ligands, one on each line, to be processed e.g.

```
python mol_2_sdf.py -o umass_1.sdf -k ligand_key_5.txt
```

1.2.2. 2D to 3D conversion

VeraChem's Vconf program is used to convert these 2D structures to 3D. The relevant files are found in the vconf/ subdirectory. First, copy over the umass_1.sdf file generated by the last step, and then execute the run_vconf.sh script to carry out the conversion:

```
./run_vconf.sh &
```

The resulting 3D structures can be found in the file

```
hiv1_protease_series_1/setup/ligands/vconf/umass_1_vconf.sdf
```

You can compare your results against those provided in the reference/ subdirectory.

1.2.3. Generate partial charges and assign parameters to the ligands

Ambertools is used to assign bond, angle, torsion, and non-bonded Lennard-Jones parameters, while atom partial charges can be generated either by VeraChem's VCharge method or by AM1-BCC through AmberTools. The resulting prmtop and inpcrd files are then converted to the [crd,top,mol] file set used by VM2.

The prepareLigands.pyc script automates this process. First, go to the prepare_ligands directory

```
hiv1_protease_series_1/setup/ligands/prepare_ligands
```

then copy over the 3D sdf file

```
cp ../source_files/umass_1.sdf .
```

Then, to execute the script choosing VCharge partial atomic charges type:

```
./run_prepareLigands_vcharge.sh &
```

and to assign charge using AM1-BCC type:

```
./run_prepareLigands_am1-bcc.sh &
```

While VCharge takes less than a minute for the set of 28 ligands, generation of AM1-BCC partial charges requires a QM calculation, which can take a considerable amount of time, e.g., approximately 3 hours on a Xeon E5-2667, 3.2GHz cpu.

You can compare your results against those in the reference subdirectories.

1.3. Define fixed and mobile protein atoms

The choice of the included mobile and fixed protein atoms can have a significant impact on the final binding energy predictions produced by the VM2 method. VeraChem recommends inclusion of enough mobile atoms to capture relevant aspects such as loop movement on binding, while avoiding inclusion of large numbers of atoms as mobile, which are effectively spectators, so as to keep calculations manageable with respect to turnover times, and also minimize the occurrence of spurious minima that sometimes occur due to force field inadequacies.

A process for defining mobile and fixed atoms for subsequent free energy calculations is now described.

1.3.1. Generate co-crystallized ligand based AD-81 conformation

First, go to the directory

```
setup/define_fixed_and_mobile_atoms/1_gen_coxtal_ligand_conf
```

Next, generate a conformation of the co-crystallized ligand AD-81 to use as the reference coordinates to carve out the mobile and fixed atoms in subsequent steps. This is achieved by ['snapping'](#) scaffold atoms from the AD-81 structure generated in Step 2 above, to the corresponding positions of the co-Xtal AD-81 scaffold atoms in the 2I0D PDB file i.e. scaffold atoms in the file `ad_81_from_2i0d.pdb` generated in Step 1.2.2

The required files are:

<code>ad_81_pdbsnap_confs.inp</code>	: VM2 input file
<code>ad_81.crd</code>	: coordinate file generated in Section 1.2.3.
<code>ad_81.top</code>	: topology/parameter file fin Section 1.2.3.
<code>ad_81.mol</code>	: mol file generated in Section 1.2.3.
<code>ad_81_from_2i0d.pdb</code>	: reference ad_81 coordinates from Section 1.1.2.

Generate the AD-81 conformations by typing:

```
./runvm2.bsh >& runvm2.log
```

The output of interest is the file:

```
ad_81.confsearch_rank1.crd
```

which contains the coordinates of lowest energy AD-81 conformer ‘snapped’ to the co-crystallized ligand scaffold atoms. The coordinate file is used in the next step.

1.3.2. Relax all hydrogen atoms in the system

To relieve close contacts that can occur on hydrogen atom placement, all hydrogen atom positions in the protein and AD-81 ligand are optimized according to the force field energy function.

Go to the directory

```
setup/define_fixed_and_mobile_atoms/2_opt_all_protein_h
```

then copy the file required from last step and rename it:

```
cp ../1_gen_coxtal_ligand_conf/ad_81.confsearch_rank1.crd ad_81_snap2pdb.crd
```

The required files for this step are:

2i0d_1580_p4a_tleap_hopt.inp	: VM2 package input file for H atom optimization
ad_81_from_2i0d.pdb	: reference ad_81 coordinates from Section 1.1.2.

2i0d_1580_p4a_tleap_vm2.crd	Protein coordinates, parameters etc.
2i0d_1580_p4a_tleap_vm2.top	<-- generated by Section 1.1. above.
2i0d_1580_p4a_tleap_vm2.mol	Copied directly from ./protein

ad_81_snap2pdb.crd	ad_81_snap2pdb.crd is the just generated
ad_81.top	<--- ad_81.confsearch_rank1.crd copied and
ad_81.mol	renamed. The top and mol files are as in 1.3.1.

Relax all hydrogen atom positions by typing:

```
./runvm2.bsh >& runvm2.log
```

The outputs of interest are the files

```
2i0d_1580_p4a_tleap_vm2.geomopt_rank1.crd
ad_81_snap2pdb.geomopt_rank1.crd
```

which contain the lowest energy coordinates of the protein and ligand AD-81 after hydrogen atom optimization. These coordinates are used in the next step.

1.3.3. Distance based generation of real/live set

Carve out a mobile and fixed set of protein atoms. VM2 uses so-called [real and live](#) sets, where the 'real' set are all the atoms included in the calculation (mobile and fixed) and the 'live' set is the subset of the 'real' set that is mobile. In this step, the VM2 package is used to carve out a 'real' set that comprises all residues that have an atom within 7 Angstroms any atom of the supplied AD-81 ligand coordinates, and a 'live' set of all protein atoms within 5 Angstroms of any atom of the supplied AD-81 ligand coordinates.

Go to the directory

```
setup/define_fixed_and_mobile_atoms/ 3_dist_based_real_live_set
```

then copy and rename the required files from the last step:

```
cp ../2_opt_all_protein_h/2i0d_1580_p4a_tleap_vm2.geomopt_rank1.crd
2i0d_1580_p4a_tleap_vm2_opth.crd
```

```
cp ../2_opt_all_protein_h/ad_81_snap2pdb.geomopt_rank1.crd
ad_81_snap2pdb_opth.crd
```

The required files for this step are:

```
2i0d_1580_p4a_tleap_genlivereal.inp <--- VM2 package input file for generation of
                                         'real' atom set of all atoms within 7
                                         Angstroms of any atom in the supplied AD-
                                         81 ligand crd, and a 'live' atom set within 5
                                         Angstroms.
```

```
2i0d_1580_p4a_tleap_vm2_opth.crd    | The crd file is the just generated
2i0d_1580_p4a_tleap_vm2.top          <--| 2i0d_1580_p4a_tleap_vm2.geomopt_rank1.crd
2i0d_1580_p4a_tleap_vm2.mol          | renamed. The top and mol are unchanged.
```

```
ad_81_snap2pdb_opth.crd              | ad_81_snap2pdb_opth.crd is the just generated
ad_81.top                            <---| ad_81_snap2pdb.geomopt_rank1.crd from above
ad_81.mol                            | renamed. The top and mol files are unchanged.
```

Generate the real and live sets by typing:

```
./runvm2.bsh >& runvm2.log
```

The following output files allow you to visualize the 'live' set produced:

```
2i0d_1580_p4a_tleap_genlivereal.mol2 <--Load into visualizer to see live set produced.
2i0d_1580_p4a_tleap_genlivereal.pdb
2i0d_1580_p4a_tleap_genlivereal.sdf
```

To see the 'real' set of atoms defined in by these distance cutoffs, run the same calculation with the input file 2i0d_1580_p4a_tleap_genlivereal.inp changed to output 'real' atoms:

```
#
atomsToOutput
real
#
```

Generated output files required for running VM2:

```
2i0d_1580_p4a_tleap_vm2_opth_liverealatoms.txt <--- This file contains the atom
                                                    numbers of the live and real
                                                    atoms generated by the
                                                    applied distance cutoffs.
```

Once you are happy with the defined real/live sets copy the protein data files required for VM2 runs directly into the directory `define_fixed_and_mobile_atoms/` i.e.

```
cp 2i0d_1580_p4a_tleap_vm2.mol ../.
cp 2i0d_1580_p4a_tleap_vm2_opth.crd ../.
cp 2i0d_1580_p4a_tleap_vm2.top ../.
cp 2i0d_1580_p4a_tleap_vm2_opth_liverealatoms.txt ../2i0d_5_7_live_real.txt
```

NOTE: mandatory renaming of `2i0d_1580_p4a_tleap_vm2_opth_liverealatoms.txt` to include the text “live_real”

The setup stage is now complete.

2. Run Calculations

The next step is to run the protein-ligand, protein, and ligand, free energy calculations. The relevant directories and readme file are:

```
hiv1_protease_series_1/run/1_ligand_confgen
hiv1_protease_series_1/run/2_vm2_runs
hiv1_protease_series_1/run/README.runvm2
```

Optionally, ligand conformations can be pre-generated in `/1_ligand_confgen` and used to seed the VM2 calculations in `/2_vm2_runs`.

2.1. Generation of Ligand Starting Conformations

Two types of pre-generated ligand conformations can be utilized in this example. One is ‘snapped’ conformations, where atoms in each ligand common to a, for example, co-crystallized ligand are, with an applied guiding force, superimposed, while conformational space of the remaining atoms is sampled. The other is randomly orientated conformations of the ligand, suitable for when no pose information is known, only the location of the binding site.

2.1.1. Example run

Go to the directory

```
run/1_ligand_confgen
```

This directory contains a python script to generate run directories for conformer generation, and a python script to run the conformer generation calculations. Example usage is as follows:

```
python build_ligand_start_conf_dirs.py -t ad_81_from_2i0d.pdb
```

will first populate the directories

```
1_ligand_confgen/gen_ligand_start_confs_snap
```

```
1_ligand_confgen/gen_ligand_start_confs_rndm
```

with the required subdirectories, input files, and data files to run. Then the following command

```
python run_ligand_confs_gen.py -r slurm
```

will step through all these subdirectories, generating slurm scripts, and submitting the calculations to the batch queue. See Section 2.1.3 below for additional submission options through the -r flag.

Note: Requirements for this example run are:

ad_81_from_2i0d.pdb	<---	must be present in /setup/ligands/prepareLigands
scaffold_mapping_wkey.txt	<---	must be present in the current directory and contain the mapping of each ligand onto the reference ligand

2.1.2. Options available for building conformer generation directories

The python script `build_ligand_start_conf_dirs.py` can take a number of arguments for non-default control the source of the system data etc.:

-d or --data	reference	: Populate 'input_data' directory using the data in the setup 'reference' directories e.g. /setup/ligands/prepareLigands/reference, and subsequently build the run directories with this data.
	new	: Populate 'input_data' directory using the new data in the setup directories e.g. /setup/ligands/prepareLigands, and subsequently build the run directories with this data. (Default behavior.)

- reuse : Reuse the data from an already populated 'input_data' directory.
- s or --startconfs random : Make a run directory for each ligand in the series for generation of ligand conformers in random orientations and with their center of geometry (COG) placed at a template ligand's COG.
- snap : Make a run directory for each ligand in the series for generation of ligand conformers where scaffold atoms are 'snapped' to corresponding template ligand scaffold atoms (via applied harmonic potentials).
- all : Make both of the above run directories. (Default behavior.)
- t or --template 'template_filename' : Name of file containing template ligand coordinates e.g. co-xtal ligand or previously docked ligand. Required unless '-d reuse' option set.
- c or --clear input : Delete the contents of 'input_data' directory.
- rundirs : Delete the contents of the run directories 'gen_ligand_start_confs_rndm' and 'gen_ligand_start_confs_snap'.
- all : Delete content from the 'input_data' directory and the run directories.

Example usage:

```
python build_ligand_start_conf_dirs.py -c rundirs -d reuse
```

This will clear the contents of previously generated run directories and use the data already present in ./input_data to regenerate the run directories i.e. data will not be taken from the setup directories in this case.

2.1.3. Options available for running conformer generation

The python script run_ligand_confs_gen.py can take a number of arguments:

<code>-s</code> or <code>--startconfs</code>	<code>random</code>	: Step through each ligand directory in <code>/gen_ligand_start_confs_rndm</code> and submit a calculation for generation of ligand conformers in random orientations and with their center of geometry (COG) placed at a template ligand's COG.
	<code>snap</code>	: Step through each ligand directory in <code>gen_ligand_start_confs_snap</code> and submit a calculation for generation of ligand conformers where scaffold atoms are 'snapped' to corresponding template ligand scaffold atoms (via applied harmonic potentials).
	<code>all</code>	: Carry out both sets of calculations. (Default behavior.)
<code>-r</code> or <code>--runscript</code>	<code>bsh</code>	: Generate and use bash shell scripts for submission of each calculation. (Default behavior.)
	<code>csh</code>	: Generate and use c-shell scripts for submission of each calculation.
	<code>pbs</code>	: Generate a pbs script for submission of each calculation to a queue.
	<code>slurm</code>	: Generate a slurm script for submission of each calculation to a queue.
<code>-q</code> or <code>--partition</code>	<code>'queue name'</code>	: For pbs and slurm run scripts, the name of the queue or partition if the default queue is not being used.
<code>-p</code> or <code>--prepmode</code>		: If present the run scripts are generated and placed in every directory, but the calculations are not submitted.

2.2. Protein-ligand calculations

Two main types of VM2 protein-ligand free energy calculation are available. One is regular VM2, which carries out iterative rounds of conformational searching until convergence; the other type carries out geometry optimizations of protein-ligand conformations constructed from ligand conformers read-in and processes them for free energy. The latter is much faster, but much less exhaustive in terms of sampling conformational space. In combination, there are three ways to seed these two VM2

calculation types with ligand conformers: multiple conformers with selected atoms ‘snapped’ to a reference ligand – see Section 2.1. above; multiple conformers randomly orientated in space, but placed at the location of the binding site – see Section 2.1. above, and a single conformer, based on the position and geometry in which it was prepared originally. This provides for six different overall VM2 calculation schemes, which cover various types of use scenarios.

2.2.1. Example run

Go to the directory

```
run/2_vm2_runs
```

This directory contains a python script to generate run directories for protein-ligand VM2 free energy calculations, and a python script to step through the directories and run the calculations. Example usage is as follows:

```
python build_vm2_run_dirs.py -t ad_81_from_2i0d.pdb
```

will first populate the following six directories, which cover the calculation types described above, with the required subdirectories, input files, and data files to run.

```
/2_vm2_runs/fast_vm2_snap  
/2_vm2_runs/fast_vm2_rndm  
/2_vm2_runs/fast_vm2_single  
/2_vm2_runs/vm2_snap  
/2_vm2_runs/vm2_rndm  
/2_vm2_runs/vm2_single
```

Note: For “_snap” and “_rndm” types, the corresponding pre-generation of ligand conformers – Section 2.1. - must already have occurred.

Then the following command:

```
python run_vm2_calculations.py -s snap -v fast -r slurm
```

will step through the subdirectories of /2_vm2_runs/fast_vm2_snap, generating slurm scripts, and submitting the calculations to the batch queue. Similarly, any of the other five calculations types may be run by setting the appropriate flags – see Section 2.2.2 below. See Section 2.2.3 below for additional submission options through the -r flag.

2.2.2. Options available for building VM2 directories

The python script build_vm2_run_dirs.py can take a number of arguments for non-default control of the source of the system data etc.:

```
-d or --data    reference    : Populate 'input_data' directory using the  
                             data in the setup 'reference' directories  
                             e.g. /setup/ligands/prepareLigands/reference and
```

/setup/define_fixed_and_mobile_atoms/reference,
 and the ligand start conformer generation
 reference directory /run/1_ligand_confgen/reference
 and subsequently build the run directories
 with this data.

new : Populate 'input_data' directory using the new data in the
 setup directories e.g. /setup/ligands/prepareLigands and
 /setup/define_fixed_and_mobile_atoms/
 and the ligand start conformer generation directories
 /run/1_ligand_confgen/gen_ligand_start_confs_rndm
 and /run/1_ligand_confgen/gen_ligand_start_confs_snap
 and subsequently build the run directories
 with this data. (Default behavior.)

reuse : Reuse the data from an already populated
 'input_data' directory.

-s or --startconfs random : Requests run directory set up for VM2 free energy
 calculations where randomly oriented ligand conformers
 are placed in the active site and are used to generate
 starting protein-ligand conformations.

snap : Requests run directory set up for VM2 free energy
 calculations where ligand conformers in which scaffold
 atoms have been 'snapped' to corresponding scaffold
 atoms of a template ligand (e.g. co-xtal ligand) are
 used to generate starting protein-ligand conformations.

single : Requests run directory set up for VM2 free energy
 calculations where a single ligand starting conformation
 and placement is used based on the supplied ligand .crd
 file coordinates. The placement can be adjusted if a
 template ligand is supplied and the place ligand flag set;
 see -t, --template and -p, --placelig below. Only used a
 non-adjusted ligand .crd if you prepared the ligand in a
 very good placement and pose in the receptor binding site.

all : Requests both types of directory to be set up.
 (Default behavior.)

-t or --template 'template_filename' : Name of file containing template ligand
 coordinates e.g. co-xtal ligand or
 previously docked ligand. Could simply be
 coordinates that signify the location of
 the binding site. Not required unless
 random start conformers are in use or the
 place ligand option just below is set.

- p or --placelig tcog : Place ligand .crd coordinates center of geometry at template ligand's center of geometry.

- c or --clear input : Delete the contents of 'input_data' directory.

- rundirs : Delete the contents of the run directories.

- all : Delete content from the 'input_data' directory and the run directories.

- v or --vm2type regular : Requests run directory set up for regular VM2 protein-ligand free energy calculations, which carry out extensive conformational searching.

- fast : Requests run directory set up for fast VM2 protein-ligand free energy calculations, which calculate free energies via geometry optimizing protein-ligand conformations generated from read-in ligand conformers previously snapped to a template scaffold.

- all : Requests set up for both types of VM2 calculation.

- k or --keyfile 'ligand_key_filename' : Name of text file containing the subset of ligands in the series - one on each line (see ligand_key_5.txt.)

2.2.3. Options available for running VM2 calculations

The python script run_ligand_confs_gen.py can take a number of arguments:

- s or --startconfs random : Requests that VM2 free energy calculations are run for the series where randomly oriented ligand conformers are placed in the active site and are used to generate starting protein-ligand conformations.

- snap : Requests that VM2 free energy calculations are run for the series where ligand conformers in which scaffold atoms have been 'snapped' to corresponding scaffold atoms of a template ligand (e.g. co-xtal ligand) are used to generate starting protein-ligand conformations. (Default behavior.)

- single : Requests run directory set up for VM2 free energy calculations where a single ligand starting conformation

and placement is used based on the supplied ligand .crd file coordinates. (See above.)

- all : Requests all types of run be carried out.
- r or --runscript bsh : Generate and use bash shell scripts for submission of each calculation. (Default behavior.)
- csh : Generate and use c-shell scripts for submission of each calculation.
- pbs : Generate a pbs script for submission of each calculation to a queue.
- slurm : Generate a slurm script for submission of each calculation to a queue.
- q or --partition 'queue name' : For pbs and slurm run scripts, the name of the queue or partition if the default queue is not being used.
- p or --prepmode : If present the run scripts are generated and placed in every directory, but the calculations are not submitted.
- v or --vm2type regular : Requests regular VM2 protein-ligand free energy calculations for the series, which carry out extensive conformational searching.
- fast : Requests fast VM2 VM2 protein-ligand free energy calculations for the series, which calculate free energies via geometry optimizing protein-ligand conformations generated from read-in ligand conformers snapped to a template scaffold. (Default behavior.)
- all : Requests both types of VM2 calculation are run for the series.
- i or --mpiprocs n (integer) : Sets the number of MPI processes to run. Currently all processes must run on the same node - though hand editing of run scripts can remove this restriction. The default is 8.
- g or --gpu : If present requests use of CUDA enabled VM2 executable.

<code>-o</code> or <code>--ompthreads</code>	1	:	If <code>-g</code> not set results in MPI parallelism only. Enforced for ligand only runs.
	2	:	If set will result in MPI+OpenMP run (8 MPI processes (default), 2 OpenMP threads per process). If <code>-g</code> also set will result in MPI+OpenMP+CUDA parallelism.
	4	:	Same as previous, but 4 OpenMP threads.

<code>-m</code> or <code>--molsystems</code>	complexes+ligands		
	complexes+protein		
	protein+ligand		
	complexes		
	ligands		
	protein		
	all	:	Default. Run ligands, complexes, and protein.

----> Run subset of the molecular system types.

Example usage:

```
nohup python run_vm2_calculations.py -g -o 2
```

Run default fast-snap set of calculations (fast_vm2_snap directory) with 8 MPI process calculations for ligand calculations, but MPI+OpenMP+CUDA calculations for the complexes and the protein.

This run utilizes 8 MPI processes with 1 GPU per MPI process and 2 OpenMP threads per MPI process. It therefore requires 16 compute cores and 8 GPUs.

3. Results Collection

When the protein-ligand, protein, and ligand VM2 free energy calculations for the complete ligand series have completed, the binding free energies may then be calculated, and the formatted files, e.g., .mol2, .pdb, .sdf, containing the associated molecular structures collected.

The relevant directories and readme file are:

```
hiv1_protease_series_1/results
```

```
hiv1_protease_series_1/results/conformers
hiv1_protease_series_1/results/README.results
```

3.1. Generate binding free energy spreadsheets and collect conformer files

Go to the directory

```
hiv1_protease_series_1/results
```

To generate spreadsheets and collect molecule conformer files for the “fast_vm2_snap” calculations from Section 2.2.1 type:

```
python create_vm2_summaries.py -c fast_vm2_snap -n 2i0d -l ad_81
```

Requirements:

File containing experimental data: experimental_data.csv

The filename must contain “experimental_data”.

The format is <proteinname_ligandname>, <value> e.g.

```
2i0d_ad_12,-9.367
2i0d_ad_17,-14.203
2i0d_ad_23,-11.559
2i0d_ad_24,-10.126
2i0d_ad_32,-10.337
2i0d_ad_33,-12.458
:
```

Output spreadsheets:

```
results/2i0d_fast_vm2_snap_complex.csv
results/2i0d_fast_vm2_snap_protein.csv
results/fast_vm2_snap_ligand.csv
results/2i0d_fast_vm2_snap_SUMMARY.csv
```

The last of these contains the binding free energies.

Output conformer files:

For the protein, each ligand, and each protein-ligand complex, formatted files (e.g. mol2, pdb, sdf, xyz) containing the lowest energy conformer, and the eight lowest energy conformers are written to:

```
results/conformers/fast_vm2_rndm/complexes
results/conformers/fast_vm2_rndm/ligands
results/conformers/fast_vm2_rndm/protein
```

3.2. Results generation options

For the script `create_vm2_summaries.py` the following two commandline arguments are mandatory with the following options:

<code>-c</code> or <code>--calctype</code>	<code>fast_vm2_snap</code>	: Identify the calculation type to collect and summarize run data for.
	<code>fast_vm2_rndm</code>	

fast vm2 single

vm2_snap

vm2 rndm

vm2 single

-n or --receptorname : Provide the name of the receptor
e.g. for this case the protein
is named “2i0d”

There are two additional non mandatory arguments:

-l or --refligand : Provide the name of the reference ligand to be used in relative binding affinity calculation i.e. for Delta(DeltaG)
The default is no reference.

-g or --getconfs <number of confs> : The number of conformers to keep in the extracted formatted conformer files e.g. .sdf, .mol2, .pdb. The default is 8 plus a set of formatted files each with the lowest energy conformer.

XI. Host-guest example: Sampl6 Octa-acids and guests

This is a full example of setup, execution of calculations, and collection of binding affinity results for the host molecules octa-acid (OA) and methylated octa-acid (TEMOA) and series of eight guests (ligands), for a total of sixteen complexes. The data sets – starting SD files and experimental binding affinities - are taken from the Sampl6 challenge [repository](#). (50)

NOTE: You will need a working installation of AmberTools with the \$AMBERHOME environment variable set to carry out the full procedure as described below. Please see <http://ambermd.org/> to download AmberTools and for its documentation.

To proceed, first, untar the examples file vcCompChem_2_8_2_examples.tar.bz2, which is provided with the package:

```
tar xvf vcCompChem_2_8_2_examples.tar.bz2
```

The main directory for this example is:

```
vcCompChem_2_8_2_examples/host_guest/Sampl6/oa_gaff_vcharge
```

it contains a readme file: README.sampl6.oa , which describes the overall process, stepping through the following three directories in turn

```
Sampl6/oa_gaff_vcharge/setup  
Sampl6/oa_gaff_vcharge/run  
Sampl6/oa_gaff_vcharge/results
```

An outline of each step now follows. You can skip the setup section by going straight to Section 2. and making use of the “-d reference” option, described in Sections [2.1.2.](#) and [2.2.2.](#)

1. Setup

The procedure starts with setup, namely structure preparation, typing, and charge assignment of the host and guest molecules. A step-by-step description of the setup process now follows. Also, see:

```
Sampl6/oa_gaff_vcharge/setup/README.setup
```

1.1. Host Setup

The relevant subdirectories are:

```
Sampl6/oa_gaff_vcharge/setup/hosts/source_files  
Sampl6/oa_gaff_vcharge/setup/hosts/prepareHosts
```

1.1.1. Source files

The /source_files directory contains .sdf, .mol2, and .pdb files for the host molecules octa-acid (OA) and methylated octa-acid (TEMOA) taken from the Sampl6 challenge repository. It also contains .mol files, derived from the .sdf files, along with a script mol_2_sdf.py to combine these .mol files into a single SD file, oa_hosts.sdf, for processing in /prepareHosts.

```
python mol_2_sdf.py oa_hosts.sdf
```

1.1.2. Generate partial charges and assign parameters

Ambertools is used to assign bond, angle, torsion, and non-bonded Lennard-Jones parameters, while atom partial charges can be generated either by VeraChem's VCharge method or by AM1-BCC through AmberTools – for this example VCharge will be used. The resulting prmtop and inpcrd files are then converted to the [crd,top,mol] file set used by VM2.

The prepareLigands.pyc script (it can be used for host molecules as well as ligands) automates this process. First, go to the prepareHosts directory

```
Sampl6/oa_gaff_vcharge/setup/hosts/prepareHosts
```

then copy over the host sdf file just generated

```
cp ../source_files/oa_hosts.sdf .
```

Then, to execute the script choosing VCharge partial atomic charges type:

```
./run_prepareHosts.sh &
```

This script contains the command line:

```
$VCHOME/exe/vc_python $VCHOME/exe/prepareLigands.pyc -charge_method  
vcharge oa_hosts.sdf >& run_prepareHosts.out &
```

To assign charge using AM1-BCC instead remove the charge method argument:

```
$VCHOME/exe/vc_python $VCHOME/exe/prepareLigands.pyc oa_hosts.sdf >&  
run_prepareHosts.out &
```

You can compare your results against those in the reference subdirectories.

1.2. Ligand Setup

The relevant subdirectories are:

```
Sampl6/oa_gaff_vcharge/setup/ligands/source_files
```

Sampl6/oa_gaff_vcharge/setup/ligands/prepareLigands

The steps basically mirror those just described for the host molecules.

1.2.1. Source files

The /source_files directory contains .sdf and .mol2 files for the ligand molecules OA-G0 to OA-G7 taken from the Sampl6 challenge repository. It also contains a script combine_sdfs.py to combine the SD files into a single SD file, oa_ligands.sdf, for processing in /prepareLigands.

```
python combine_sdfs.py oa_ligands.sdf
```

1.2.2. Generate partial charges and assign parameters

Ambertools is used to assign bond, angle, torsion, and non-bonded Lennard-Jones parameters, while atom partial charges can be generated either by VeraChem's VCharge method or by AM1-BCC through AmberTools – for this example VCharge will be used. The resulting prmtop and inpcrd files are then converted to the [crd,top,mol] file set used by VM2.

The prepareLigands.pyc script automates this process. First, go to the prepareLigands directory

Sampl6/oa_gaff_vcharge/setup/hosts/prepareLigands

then copy over the ligand sdf file just generated

```
cp ../source_files/oa_ligands.sdf .
```

Then, to execute the script choosing VCharge partial atomic charges type:

```
./run_prepareLigands.sh &
```

This script contains the command line:

```
$VCHOME/exe/vc_python $VCHOME/exe/prepareLigands.pyc -charge_method  
vcharge oa_ligands.sdf >& run_prepareLigands.out &
```

To assign charge using AM1-BCC instead remove the charge method argument:

```
$VCHOME/exe/vc_python $VCHOME/exe/prepareLigands.pyc oa_ligands.sdf  
>& run_prepareLigands.out &
```

You can compare your results against those in the reference subdirectories.

The setup stage is now complete.

2. Run Calculations

The next step is to run the host-guest, host, and ligand, free energy calculations. The relevant directories and readme file are:

```
Sampl6/oa_gaff_vcharge/run/1_ligand_confgen  
Sampl6/oa_gaff_vcharge/run/2_vm2_runs  
Sampl6/oa_gaff_vcharge/run/README.runvm2
```

Ligand conformations can be pre-generated in /1_ligand_confgen and used to seed the VM2 calculations in /2_vm2_runs.

2.1. Generation of Ligand Starting Conformations

Randomly orientated conformations of the ligand are generated, which are read-in to seed the actual host-guest VM2 free energy calculations.

2.1.1. Example run

Go to the directory

```
run/1_ligand_confgen
```

This directory contains a python script to generate run directories for conformer generation, and a python script to run the conformer generation calculations. Example usage is as follows:

```
python build_ligand_start_conf_dirs.py
```

will first populate the directory

```
1_ligand_confgen/gen_ligand_start_confs_rndm
```

with the required subdirectories, input files, and data files to run. Then the following command

```
python run_ligand_confs_gen.py -r slurm
```

will step through all these subdirectories, generating slurm scripts, and submitting the calculations to the batch queue. See Section 2.1.3 below for additional submission options through the -r flag.

2.1.2. Options available for building conformer generation directories

The python script build_ligand_start_conf_dirs.py can take a number of arguments for non-default control the source of the system data etc.:

```
-d or --data    reference    : Populate 'input_data' directory using the  
                           data in the setup 'reference' directories
```


e.g. /setup/ligands/prepareLigands/reference,
and subsequently build the run directories
with this data.

new	:	Populate 'input_data' directory using the new data in the setup directories e.g. /setup/ligands/prepareLigands, and subsequently build the run directories with this data. (Default behavior.)
reuse	:	Reuse the data from an already populated 'input_data' directory.
-c or --clear	input	: Delete the contents of 'input_data' directory.
	rundirs	: Delete the contents of the run directories 'gen_ligand_start_confs_rndm' and 'gen_ligand_start_confs_snap'.
	all	: Delete content from the 'input_data' directory and the run directories.

Example usage:

```
python build_ligand_start_conf_dirs.py -c rundirs -d reuse
```

This will clear the contents of previously generated run directories and use the data already present in ./input_data to regenerate the run directories i.e. data will not be taken from the setup directories in this case.

2.1.3. Options available for running conformer generation

The python script run_ligand_confs_gen.py can take a number of arguments:

-r or --runscript	bsh	: Generate and use bash shell scripts for submission of each calculation. (Default behavior.)
	csh	: Generate and use c-shell scripts for submission of each calculation.
	pbs	: Generate a pbs script for submission of each calculation to a queue.
	slurm	: Generate a slurm script for submission of each calculation to a queue.
-q or --partition	'queue name'	: For pbs and slurm run scripts, the name of the queue or partition if the default queue is not being used.

-p or --prepmode : If present the run scripts are generated and placed in every directory, but the calculations are not submitted.

2.2. Host-guest calculations

Two main types of VM2 host-guest free energy calculation are available. One is regular VM2, which carries out iterative rounds of conformational searching until convergence; the other type carries out geometry optimizations of host-guest conformations constructed from ligand conformers read-in and processes them for free energy. The latter is much faster, but much less exhaustive in terms of sampling conformational space. In combination, there are two ways to seed these two VM2 calculation types with ligand conformers: multiple conformers randomly orientated in space, but placed at the center of geometry of the host – see Section 2.1. above, and a single conformer, based on the geometry in which it was prepared originally, and also placed at the center of geometry of the host. This provides for four different overall VM2 calculation schemes, which cover various types of use scenarios.

2.2.1. Example run

Go to the directory

```
run/2_vm2_runs
```

This directory contains a python script to generate run directories for host-guest VM2 free energy calculations, and a python script to step through the directories and run the calculations. Example usage is as follows:

```
python build_vm2_run_dirs.py
```

will first populate the following four directories, which cover the calculation types described above, with the required subdirectories, input files, and data files to run.

```
/2_vm2_runs/fast_vm2_rndm  
/2_vm2_runs/fast_vm2_single  
/2_vm2_runs/vm2_rndm  
/2_vm2_runs/vm2_single
```

Note: For “_rndm” types, the corresponding pre-generation of ligand conformers – Section 2.1. - must already have occurred.

Then the following command:

```
python run_vm2_calculations.py -s random -v fast -r slurm
```

will step through the subdirectories of /2_vm2_runs/fast_vm2_snap, generating slurm scripts, and submitting the calculations to the batch queue. Similarly, any of the other

three calculations types may be run by setting the appropriate flags – see Section 2.2.2 below. See Section 2.2.3 below for additional submission options through the -r flag.

2.2.2. Options available for building VM2 directories

The python script `build_vm2_run_dirs.py` can take a number of arguments for non-default control of the source of the system data etc.:

- | | | |
|--------------------|-----------|--|
| -d or --data | reference | : Populate 'input_data' directory using the data in the setup 'reference' directories e.g. <code>/setup/ligands/prepareLigands/reference</code> , and the ligand start conformer generation reference directory <code>/run/1_ligand_confgen/reference</code> and subsequently build the run directories with this data. |
| | new | : Populate 'input_data' directory using the new data in the setup directories e.g. <code>/setup/ligands/prepareLigands</code> and the ligand start conformer generation directory <code>/run/1_ligand_confgen/gen_ligand_start_confs_rndm</code>

and subsequently build the run directories with this data. (Default behavior.) |
| | reuse | : Reuse the data from an already populated 'input_data' directory. |
| -s or --startconfs | random | : Requests run directory set up for VM2 free energy calculations where randomly oriented ligand conformers are placed at the host center of geometry and are used to generate starting host-guest conformations. |
| | single | : Requests run directory set up for VM2 free energy calculations where a single ligand starting conformation is used based on the supplied ligand <code>.crd</code> file coordinates. The placement is set as the center of geometry of the host molecule. |
| | all | : Requests both types of directory to be set up. (Default behavior.) |
| -c or --clear | input | : Delete the contents of 'input_data' directory. |
| | rundirs | : Delete the contents of the run directories. |
| | all | : Delete content from the 'input_data' directory and the run directories. |

- v or --vm2type regular : Requests run directory set up for regular VM2 host-guest free energy calculations, which carry out extensive conformational searching.
- fast : Requests run directory set up for fast VM2 host-guest free energy calculations, which calculate free energies via geometry optimizing host-guest conformations generated from read-in ligand conformers previously generated.
- all : Requests set up for both types of VM2 calculation.
- k or --keyfile 'ligand_key_filename' : Name of text file containing the subset of ligands in the series - one on each line (see ligand_key_5.txt.)

2.2.3. Options available for running VM2 calculations

The python script run_ligand_confs_gen.py can take a number of arguments:

- s or --startconfs random : Requests that VM2 free energy calculations are run for the series where randomly oriented ligand conformers are placed in the active site and are used to generate starting protein-ligand conformations. (Default behavior.)
- single : Requests that VM2 free energy calculations are run for the series where a single ligand/guest conformation is placed at the host's center of geometry generating a single starting host-guest conformation.
- all : Requests both types of run be carried out.
- r or --runscript bsh : Generate and use bash shell scripts for submission of each calculation. (Default behavior.)
- csh : Generate and use c-shell scripts for submission of each calculation.
- pbs : Generate a pbs script for submission of each calculation to a queue.
- slurm : Generate a slurm script for submission of each calculation to a queue.

-q or --partition	'queue name'	: For pbs and slurm run scripts, the name of the queue or partition if the default queue is not being used.
-p or --prepmode		: If present the run scripts are generated and placed in every directory, but the calculations are not submitted.
-v or --vm2type	regular	: Requests regular VM2 protein-ligand free energy calculations for the series, which carry out extensive conformational searching.
	fast	: Requests fast VM2 VM2 protein-ligand free energy calculations for the series, which calculate free energies via geometry optimizing protein-ligand conformations generated from read-in ligand conformers snapped to a template scaffold. (Default behavior.)
	all	: Requests both types of VM2 calculation are run for the series.
-g or --gpu		: If present requests use of CUDA enabled VM2 executable.
-o or --ompthreads	1	: If -g not set results in MPI parallelism only. Enforced for ligand only runs.
	2	: If set will result in MPI+OpenMP run (8 MPI processes (default), 2 OpenMP threads per process). If -g also set will result in MPI+OpenMP+CUDA parallelism.
-m or --molsystems	complexes+ligands	 ----> Run subset of the molecular system types.
	complexes+hosts	
	hosts+ligand	
	complexes	
	ligands	
	hosts	

all : Default. Run ligands, complexes, and hosts.

Example usage:

```
nohup python run_vm2_calculations.py -g -o 2
```

Run default fast-random set of calculations (fast_vm2_randm directory) with 8 MPI process calculations for ligand calculations, but MPI+OpenMP+CUDA calculations for the complexes and the hosts.

This run utilizes 8 MPI processes with 1 GPU per MPI process and 2 OpenMP threads per MPI process. It therefore requires 16 compute cores and 8 GPUs.

3. Results Collection

When the host-guest (ligand), host, and ligand VM2 free energy calculations for the complete ligand series have completed, the binding free energies may then be calculated, and the formatted files, e.g., .mol2, .pdb, .sdf, containing the associated molecular structures collected.

The relevant directories and readme file are:

```
Sampl6/oa_gaff_vcharge /results
Sampl6/oa_gaff_vcharge /results/conformers
Sampl6/oa_gaff_vcharge /results/README.results
```

3.1. Generate binding free energy spreadsheets and collect conformer files

Go to the directory

```
Sampl6/oa_gaff_vcharge /results
```

To generate spreadsheets and collect molecule conformer files for the “fast_vm2_rndm” calculations from Section 2.2.1 type:

```
python create_vm2_summaries.py -c fast_vm2_rndm -l OA-G0
```

Requirements:

File containing experimental data: sampl6_oa_experimental_data.txt

The filename must contain the text “experimental_data”.

The format is <hostname_ligandname>, <value> e.g.

```
OA_OA-G0, -5.68
OA_OA-G1, -4.65
OA_OA-G2, -8.38
```

OA_OA-G3, -5.18
OA_OA-G4, -7.11
:

Output spreadsheets:

```
results/OA_TEMOA_fast_vm2_rndm_complex.csv  
results/OA_TEMOA_fast_vm2_snap_host.csv  
results/fast_vm2_rndm_ligand.csv  
results/OA_TEMOA_fast_vm2_rndm_SUMMARY.csv
```

The last of these contains the binding free energies.

Output conformer files:

For the protein, each ligand, and each host-ligand complex, formatted files (e.g. mol2, pdb, sdf, xyz) containing the lowest energy conformer, and the eight lowest energy conformers are written to:

```
results/conformers/fast_vm2_rndm/complexes  
results/conformers/fast_vm2_rndm/ligands  
results/conformers/fast_vm2_rndm/hosts
```

3.2. Results generation options

For the script `create_vm2_summaries.py` the following commandline argument is mandatory with the following options:

<code>-c</code> or <code>--calctype</code>	<code>fast_vm2_rndm</code>	: Identify the calculation type to collect and summarize run data for.
	<code>fast_vm2_single</code>	
	<code>vm2_rndm</code>	
	<code>vm2_single</code>	

There are three additional non mandatory arguments:

<code>-n</code> or <code>--receptorname</code>	: Provide the name of the receptor e.g. for this case the hosts are named "OA" and "TEMOA" This is useful if more than one host and separate summary files are required for each host or if you want the results files labeled with the host name.
<code>-l</code> or <code>--refligand</code>	: Provide the name of the reference ligand to be used in relative binding

affinity calculation i.e. for Δ (Δ G)
The default is no reference.

-g or --getconfs <number of confs> : The number of conformers to keep in the extracted formatted conformer files e.g. .sdf, .mol2, .pdb. The default is 8 plus a set of formatted files each with the lowest energy conformer.

XII. VeraChem file formats

1. VeraChem's topology/parameter file (.top) format examples

The .top file format specification is described in detail in Section II. The following is a specific example for a small (ligand) molecule and the CHARMM force field – note the columns 8 and 9 in the atom block, which, specific to CHARMM, contain van der Waals parameters for 1-4 interactions.

```
!NTITLE 1
!NATOM: 23
  1 C6R 12.01100 -0.11100 -0.05000 2.04000 -0.10000 1.76000
  2 C6R 12.01100 -0.11100 -0.05000 2.04000 -0.10000 1.76000
  3 C6R 12.01100 -0.11300 -0.05000 2.04000 -0.10000 1.76000
  4 C6R 12.01100 -0.11300 -0.05000 2.04000 -0.10000 1.76000
  5 C6R 12.01100 -0.01000 -0.05000 2.04000 -0.10000 1.76000
  6 C6R 12.01100 0.08800 -0.05000 2.04000 -0.10000 1.76000
  7 C 12.01100 0.59800 -0.14100 1.87000
  8 CT 12.01100 -0.25800 -0.09030 1.80000 -0.10000 1.75000
  9 CT 12.01100 0.03700 -0.09030 1.80000 -0.10000 1.75000
 10 NP 14.00670 -0.69000 -0.09000 1.83000 -0.10000 1.63000
 11 O 15.99940 -0.51600 -0.15910 1.55000 -0.20000 1.36000
 12 OS 15.99940 -0.35100 -0.15910 1.60000 -0.20000 1.36000
 13 HA 1.00800 0.10900 -0.04200 1.33000
 14 HA 1.00800 0.10900 -0.04200 1.33000
 15 HA 1.00800 0.10900 -0.04200 1.33000
 16 HA 1.00800 0.10900 -0.04200 1.33000
 17 HA 1.00800 0.09100 -0.04200 1.33000
 18 HA 1.00800 0.09100 -0.04200 1.33000
 19 HA 1.00800 0.09100 -0.04200 1.33000
 20 HA 1.00800 0.08700 -0.04200 1.33000
 21 HA 1.00800 0.08700 -0.04200 1.33000
 22 H 1.00800 0.33400 -0.04980 0.80000
 23 H 1.00800 0.33400 -0.04980 0.80000
!NBOND: 23
  1 3 880.000 1.38300 C6R C6R
  1 5 880.000 1.38300 C6R C6R
  1 13 740.000 1.08000 C6R HA
  2 4 880.000 1.38300 C6R C6R
  2 5 880.000 1.38300 C6R C6R
  2 14 740.000 1.08000 C6R HA
  3 6 880.000 1.38300 C6R C6R
  3 15 740.000 1.08000 C6R HA
  4 6 880.000 1.38300 C6R C6R
  4 16 740.000 1.08000 C6R HA
  5 7 772.000 1.46000 C6R C
  6 10 780.000 1.35500 C6R NP
  7 11 1280.000 1.22500 C O
  7 12 700.000 1.31900 C OS
  8 9 536.000 1.52900 CT CT
  8 17 680.000 1.09000 CT HA
  8 18 680.000 1.09000 CT HA
  8 19 680.000 1.09000 CT HA
  9 12 786.000 1.42000 CT OS
  9 20 680.000 1.09000 CT HA
  9 21 680.000 1.09000 CT HA
 10 22 931.200 1.00000 NP H
 10 23 931.200 1.00000 NP H
!NTHETA: 37
  3 1 5 140.000 2.094395 C6R C6R C6R
  3 1 13 62.000 2.094395 C6R C6R HA
  5 1 13 62.000 2.094395 C6R C6R HA
```

4	2	5	140.000	2.094395	C6R	C6R	C6R		
4	2	14	62.000	2.094395	C6R	C6R	HA		
5	2	14	62.000	2.094395	C6R	C6R	HA		
1	3	6	140.000	2.094395	C6R	C6R	C6R		
1	3	15	62.000	2.094395	C6R	C6R	HA		
6	3	15	62.000	2.094395	C6R	C6R	HA		
2	4	6	140.000	2.094395	C6R	C6R	C6R		
2	4	16	62.000	2.094395	C6R	C6R	HA		
6	4	16	62.000	2.094395	C6R	C6R	HA		
1	5	2	140.000	2.094395	C6R	C6R	C6R		
1	5	7	140.000	2.094395	C6R	C6R	C		
2	5	7	140.000	2.094395	C6R	C6R	C		
3	6	4	140.000	2.094395	C6R	C6R	C6R		
3	6	10	130.000	2.094395	C6R	C6R	NP		
4	6	10	130.000	2.094395	C6R	C6R	NP		
5	7	11	172.000	2.216568	C6R	C	O		
5	7	12	120.000	1.919862	C6R	C	OS		
11	7	12	162.000	2.171190	O	C	OS		
9	8	17	75.000	1.932079	CT	CT	HA		
9	8	18	75.000	1.932079	CT	CT	HA		
9	8	19	75.000	1.932079	CT	CT	HA		
17	8	18	66.000	1.881465	HA	CT	HA		
17	8	19	66.000	1.881465	HA	CT	HA		
18	8	19	66.000	1.881465	HA	CT	HA		
8	9	12	160.000	1.910612	CT	CT	OS		
8	9	20	75.000	1.932079	CT	CT	HA		
8	9	21	75.000	1.932079	CT	CT	HA		
12	9	20	118.000	1.889319	OS	CT	HA		
12	9	21	118.000	1.889319	OS	CT	HA		
20	9	21	66.000	1.881465	HA	CT	HA		
6	10	22	60.000	2.094395	C6R	NP	H		
6	10	23	60.000	2.094395	C6R	NP	H		
22	10	23	36.000	2.052507	H	NP	H		
7	12	9	166.000	2.022837	C	OS	CT		
!NPHI: 46									
5	1	3	6	2.800	2.000	3.142	C6R	C6R	C6R
5	1	3	15	3.000	2.000	3.142	C6R	C6R	HA
13	1	3	6	3.000	2.000	3.142	C6R	C6R	HA
13	1	3	15	2.500	2.000	3.142	HA	C6R	HA
3	1	5	2	2.800	2.000	3.142	C6R	C6R	C6R
3	1	5	7	3.100	2.000	3.142	X	C6R	X
13	1	5	2	3.000	2.000	3.142	C6R	C6R	HA
13	1	5	7	3.100	2.000	3.142	X	C6R	X
5	2	4	6	2.800	2.000	3.142	C6R	C6R	C6R
5	2	4	16	3.000	2.000	3.142	C6R	C6R	HA
14	2	4	6	3.000	2.000	3.142	C6R	C6R	HA
14	2	4	16	2.500	2.000	3.142	HA	C6R	HA
4	2	5	1	2.800	2.000	3.142	C6R	C6R	C6R
4	2	5	7	3.100	2.000	3.142	X	C6R	X
14	2	5	1	3.000	2.000	3.142	C6R	C6R	HA
14	2	5	7	3.100	2.000	3.142	X	C6R	X
1	3	6	4	2.800	2.000	3.142	C6R	C6R	C6R
1	3	6	10	3.100	2.000	3.142	X	C6R	X
15	3	6	4	3.000	2.000	3.142	C6R	C6R	HA
15	3	6	10	3.100	2.000	3.142	X	C6R	X
2	4	6	3	2.800	2.000	3.142	C6R	C6R	C6R
2	4	6	10	3.100	2.000	3.142	X	C6R	X
16	4	6	3	3.000	2.000	3.142	C6R	C6R	HA
16	4	6	10	3.100	2.000	3.142	X	C6R	X
1	5	7	11	1.300	2.000	3.142	O	C	C6R
1	5	7	12	0.500	2.000	3.142	X	C	C6R
2	5	7	11	1.300	2.000	3.142	O	C	C6R
2	5	7	12	0.500	2.000	3.142	X	C	C6R
3	6	10	22	0.500	2.000	3.142	X	C6R	NP
3	6	10	23	0.500	2.000	3.142	X	C6R	NP
4	6	10	22	0.500	2.000	3.142	X	C6R	NP

```

4      6      10      23      0.500      2.000      3.142 X      C6R      NP      X
5      7      12      9      2.500      2.000      3.142 X      C      OS      X
11     7      12      9      2.500      2.000      3.142 X      C      OS      X
17     8      9      12      0.150      3.000      0.000 X      CT      CT      X
17     8      9      20      0.150      3.000      0.000 X      CT      CT      X
17     8      9      21      0.150      3.000      0.000 X      CT      CT      X
18     8      9      12      0.150      3.000      0.000 X      CT      CT      X
18     8      9      20      0.150      3.000      0.000 X      CT      CT      X
18     8      9      21      0.150      3.000      0.000 X      CT      CT      X
19     8      9      12      0.150      3.000      0.000 X      CT      CT      X
19     8      9      20      0.150      3.000      0.000 X      CT      CT      X
19     8      9      21      0.150      3.000      0.000 X      CT      CT      X
8      9      12      7      0.100      3.000      0.000 CT      CT      OS      C
20     9      12      7      0.330      3.000      3.142 X      CT      OS      X
21     9      12      7      0.330      3.000      3.142 X      CT      OS      X
!NIMPHI: 8
1      5      13      3      150.000      0.000      3.142 HA      X      X      C6R
2      4      14      5      150.000      0.000      3.142 HA      X      X      C6R
3      6      15      1      150.000      0.000      3.142 HA      X      X      C6R
4      2      16      6      150.000      0.000      3.142 HA      X      X      C6R
5      1      7      2      200.000      0.000      3.142 C      X      X      C6R
6      3      10      4      180.000      0.000      3.142 C6R      X      X      NP
7      5      12      11     294.000      0.000      3.142 C      X      X      O
10     22     23      6      180.000      0.000      3.142 C6R      X      X      NP
!NBFIX: 0
!NFINAL: 6
23      23      37      46      8 9999
!NDON:

```

2. Definition of protein real/live atom sets

The following provides the format for the file to identify the atoms to include in the calculation (real atoms), and which of these are mobile (live atoms). See Section VIII 2.

```

#total no. of atoms in the protein
numberOfAtoms
3137
#no of flexible atoms
numberOfLiveAtoms
645
#list of flexible atoms
listOfLiveAtoms
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
158
378

```

```

379
380
382
:
:
2943
2944
2955
3135
3136
3137
#number of real atoms
numberOfRealAtoms
2027
#list of real atoms
listOfRealAtoms
67
68
69
70
71
72
73
74
75
76
:
:
3096
3097
3098
3099
3135
3136
3137
#end
end

```

3. Identify constrained (tethered) atoms sets

The following provides the format for the file to identify sets of atoms to which constraints are applied. In case shown to sets of atoms are identified, one is a large set of 503 atoms, the other a small set of 4 atoms. The type and strength of the constraints applied are defined in the .inp file (see Section VIII 14.)

```

#Constrained atoms information
#numProtein
1
#numLigand
1

#proteinid
1

#setid
1

```

```

#numTetheredAtoms
503
#atomList
122
123
124
125
126
127
128
129
130
131
132
133
134
135
267
268
269
270
271
:
:
2654
2655
2656

#setid
2
#numTetheredAtoms
4
#atomList
2686
2689
2692
2695

#ligandid
1
#numTetheredAtoms
0
#atomList

#end
end

```

4. Identify atoms to exclude search drivers

The following provides the format for the file to identify atoms which if present in a particular search driver results in the exclusion of that driver in the conformational search. (see Section VIII 7.)

```

#Excluded Atoms information
#numProtein
1

```

```
#numLigand
1

#proteinid
1
#numExcludedAtoms
1066
#atomList
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
:
:
2682
2683
2686
2689
2692
2695

#ligandid
1
#numExcludedAtoms
0
#atomList

#end
end
```

XIII. References

1. W. Chen, M. K. Gilson, S. P. Webb, M. J. Potter, Modeling Protein–Ligand Binding by Mining Minima. *J. Chem. Theory Comput.* **6**, 3540-3557 (2010).
2. C.-E. Chang, M. K. Gilson, Free Energy, Entropy, and Induced Fit in Host-Guest Recognition: Calculations with the Second-Generation Mining Minima Algorithm *J. Am. Chem. Soc.* **126**, 13156-13164 (2004).
3. Y.-m. H. W. C. M. P. C.-e. Chang, W. Chen, M. J. Potter, C.-e. A. Chang, Insights from Free-Energy Calculations: Protein Conformational Equilibrium, Driving Forces, and Ligand-Binding Modes. *Biophysj* **103**, 342-351 (2012).
4. M. K. Gilson, J. A. Given, B. L. Bush, J. A. McCammon, The statistical-thermodynamic basis for computation of binding affinities: A critical review. *Biophys. J.* **72**, 1047-1069 (1997).
5. M. Mihailescu, M. K. Gilson, On the theory of noncovalent binding. *Biophys. J.* **87**, 23-26 (2004).
6. T. Liu, Y. Lin, X. Wen, R. N. Jorissen, M. K. Gilson, BindingDB: a web-accessible database of experimentally determined protein-ligand binding affinities. *Nucl. Acid Res.* **35**, D198-D201 (2007).
7. K. N. Houk, A. G. Leach, S. P. Kim, X. Y. Zhang, Binding affinities of host-guest, protein-ligand, and protein-transition-state complexes. *Ang. Chem. Int. Ed.* **42**, 4872-4897 (2003).
8. M. V. Rekharsky, Y. Inoue, Complexation thermodynamics of cyclodextrins. *Chem. Rev.* **98**, 1875-1917 (1998).
9. M. V. Rekharsky *et al.*, A synthetic host-guest system achieves avidin-biotin affinity by overcoming enthalpy-entropy compensation. *Proceedings of the National Academy of Sciences of the United States of America* **104**, 20737-20742 (2007).
10. G. L. Warren *et al.*, A critical assessment of docking programs and scoring functions. *Journal of Medicinal Chemistry* **49**, 5912-5931 (2006).
11. C.-E. Chang, M. K. Gilson, Tork:Conformational analysis method for molecules and complexes. *J. Comput. Chem.* **24**, 1987-1998 (2003).
12. C.-E. Chang, M. J. Potter, M. K. Gilson, Calculation of molecular configuration integrals. *J. Phys. Chem. B* **107**, 1048-1055 (2003).
13. M. J. Potter, M. K. Gilson, Coordinate Systems and the Calculation of Molecular Properties. *J. Phys. Chem. A* **106**, 563-566 (2002).
14. W. Chen, J. Huang, M. K. Gilson, Identification of symmetries in molecules and complexes. *J. Chem. Inf. Comput. Sci.* **44**, 1301-1313 (2004).
15. B. R. Brooks *et al.*, CHARMM: A Program for Macromolecular Energy, Minimization and Dynamics Calculations. *J. Comput. Chem.* **4**, 187-217 (1983).
16. B. R. Brooks *et al.*, CHARMM: The biomolecular simulation program. *Journal of Computational Chemistry* **30**, 1545-1614 (2009).
17. W. L. Jorgensen, J. Tirado-Rives, The OPLS Potential Function for Proteins. Energy Minimizations for Crystals of Cyclic Peptides and Crambin. *J. Am. Chem. Soc.* **110**, 1657-1666 (1988).
18. D. A. Pearlman *et al.*, Amber, A Package of Computer-programs for Applying Molecular Mechanics, Normal-Mode Analysis, Molecular-Dynamics and Free-

- Energy Calculations to Simulate the Structural and Energetic Properties of Molecules. *Comput. Phys. Commun.* **91**, 1-41 (1995).
19. S. J. Weiner *et al.*, A new force-field for molecular mechanical simulation of nucleic-acids and proteins. *Journal of the American Chemical Society* **106**, 765-784 (1984).
 20. J. M. Wang, R. M. Wolf, J. W. Caldwell, P. A. Kollman, D. A. Case, Development and testing of a general amber force field. *Journal of Computational Chemistry* **25**, 1157-1174 (2004).
 21. K. Vanommeslaeghe *et al.*, CHARMM General Force Field: A Force Field for Drug-Like Molecules Compatible with the CHARMM All-Atom Additive Biological Force Fields. *Journal of Computational Chemistry* **31**, 671-690 (2010).
 22. S. L. Mayo, B. D. Olafson, W. A. Goddard III, DREIDING: A generic force field for molecular simulations. *J. Phys. Chem.* **94**, 8897-8909 (1990).
 23. W. L. Jorgensen, D. S. Maxwell, J. TiradoRives, Development and testing of the OPLS all-atom force field on conformational energetics and properties of organic liquids. *Journal of the American Chemical Society* **118**, 11225-11236 (1996).
 24. W. Damm, A. Frontera, J. TiradoRives, W. L. Jorgensen, OPLS all-atom force field for carbohydrates. *Journal of Computational Chemistry* **18**, 1955-1970 (1997).
 25. W. L. Jorgensen, N. A. McDonald, Development of an all-atom force field for heterocycles. Properties of liquid pyridine and diazenes. *Theochem-Journal of Molecular Structure* **424**, 145-155 (1998).
 26. R. C. Rizzo, W. L. Jorgensen, OPLS all-atom model for amines: Resolution of the amine hydration problem. *Journal of the American Chemical Society* **121**, 4827-4836 (1999).
 27. G. A. Kaminski, R. A. Friesner, J. Tirado-Rives, W. L. Jorgensen, Evaluation and reparametrization of the OPLS-AA force field for proteins via comparison with accurate quantum chemical calculations on peptides. *Journal of Physical Chemistry B* **105**, 6474-6487 (2001).
 28. W. C. Still, A. Tempczyk, R. C. Hawley, T. Hendrickson, Semianalytical Treatment of Solvation for Molecular Mechanics and Dynamics. *J. Am. Chem. Soc.* **112**, 6127-6129 (1990).
 29. D. Qiu, P. S. Shenkin, F. P. Hollinger, W. C. Still, The GB/SA continuum model for solvation. a fast analytical method for the calculation of approximate born radii. *J. Phys. Chem.* **101**, 3005-3014 (1997).
 30. A. Bondi, in *The Journal of Physical Chemistry*. (1964), vol. 68, pp. 441-451.
 31. R. C. Rizzo, T. Aynechi, D. A. Case, I. D. Kuntz, Estimation of absolute free energies of hydration using continuum methods: Accuracy of partial, charge models and optimization of nonpolar contributions. *J. Chem. Theory Comput.* **2**, 128-139 (2006).
 32. L. David, R. Luo, M. K. Gilson, Comparison of the generalized Born and Poisson models of electrostatics: Energetics and dynamics of the HIV-1 protease. *J. Comput. Chem.* **21**, 295-309 (2000).
 33. A. Nicholls, B. Honig, A rapid finite difference algorithm, utilizing successive over-relaxation to solve the Poisson-Boltzmann equation. *J. Comput. Chem.* **12**, 435-445 (1991).
 34. R. Luo, L. David, M. K. Gilson, Accelerated Poisson-Boltzmann calculations for static and dynamic systems. *Journal of Computational Chemistry* **23**, 1244-1253 (2002).

35. Q. Lu, R. Luo, A Poisson-Boltzmann dynamics method with nonperiodic boundary condition. *Journal of Chemical Physics* **119**, 11035-11047 (2003).
36. P. S. Pacheco, *Parallel Programming with MPI*. (Morgan Kaufmann Publishers, Inc., San Francisco, California, 1997).
37. B. Chapman, G. Jost, R. Van Der Pas, *Using OpenMP*. (MIT Press, Cambridge, Massachusetts, 2008), pp. 353.
38. T. Sterling, J. J. Irwin, ZINC 15 – Ligand Discovery for Everyone. *J. Chem. Inf. Model.* **55**, 2324-2337 (2015).
39. E. F. Pettersen *et al.*, UCSF chimera - A visualization system for exploratory research and analysis. *Journal of Computational Chemistry* **25**, 1605-1612 (2004).
40. W. Humphrey, A. Dalke, K. Schulten, VMD: Visual molecular dynamics. *Journal of Molecular Graphics & Modelling* **14**, 33-38 (1996).
41. M. H. M. Olsson, C. R. Sondergaard, M. Rostkowski, J. H. Jensen, PROPKA3: Consistent Treatment of Internal and Surface Residues in Empirical pK(a) Predictions. *J. Chem. Theory Comput.* **7**, 525-537 (2011).
42. C. R. Sondergaard, M. H. M. Olsson, M. Rostkowski, J. H. Jensen, Improved Treatment of Ligands and Coupling Effects in Empirical Calculation and Rationalization of pK(a) Values. *J. Chem. Theory Comput.* **7**, 2284-2295 (2011).
43. R. Salomon-Ferrer, D. A. Case, R. C. Walker, An overview of the Amber biomolecular simulation package. *Wiley Interdiscip. Rev.-Comput. Mol. Sci.* **3**, 198-210 (2013).
44. P. Eastman *et al.*, OpenMM 4: A Reusable, Extensible, Hardware Independent Library for High Performance Molecular Simulation. *J. Chem. Theory Comput.* **9**, 461-469 (2013).
45. S. Jo *et al.*, CHARMM-GUI 10 years for biomolecular modeling and simulation. *Journal of Computational Chemistry* **38**, 1114-1124 (2017).
46. B. T. Miller *et al.*, CHARMMing: A new, flexible web portal for CHARMM. *Journal of Chemical Information and Modeling* **48**, 1920-1929 (2008).
47. J. L. Markley *et al.*, Recommendations for the presentation of NMR structures of proteins and nucleic acids - (IUPAC Recommendations 1998). *Pure and Applied Chemistry* **70**, 117-142 (1998).
48. A. Dalby *et al.*, Description of several chemical-structure file formats used by computer-programs developed at Molecular Design Limited. *J. Chem. Inf. Comput. Sci.* **32**, 244-255 (1992).
49. A. Ali *et al.*, Discovery of HIV-1 protease inhibitors with picomolar affinities incorporating N-aryl-oxazolidinone-5-carboxamides as novel P2 ligands. *J. Med. Chem.* **49**, 7342-7356 (2006).
50. A. Rizzi *et al.*, Overview of the SAMPL6 host-guest binding affinity prediction challenge. *J Comput Aided Mol Des* **32**, 937-963 (2018).

XIV. Index