Distributed Replica Sampling Software Suite
Version 2.1.7

Distributed Replica (DR) sampling is a generalized ensemble simulation algorithm wherein sampling occurs amongst a collection of ensembles that are distributed along a predetermined reaction coordinate. During DR, sampling is generated by multiple replicas that each undergo a random walk along this reaction coordinate, weakly coupled by the Distributed Replica potential energy function (DRPE). For a more complete description of DR sampling, see T. Rodinger, P.L. Howell, and R. Pomès, "Distributed Replica Sampling" J. Chem. Theory Comput., 2:725 (2006).

The Distributed Replica Sampling Software Suite (DRSSS) is an implementation of DR sampling. The purpose of this document is the description of the Distributed Replica Sampling Software Suite.

Supervisor: Régis Pomès
pomes@sickkids.ca
www.pomeslab.com
Creator: Tomas Rodinger
tom.rodinger@utoronto.ca
Additional active programmers: Grace Li
fullofgrace88@gmail.com
Program Suite Maintainer: Chris Neale
chris.neale@utoronto.ca
candrewn@gmail.com
www.pomeslab.com/ChrisNeale.htm
416-813-6855
Programmers: Tomas Rodinger, Ching-Hsing Yu, Chris Neale, Grace Li

Distributed Replica Sampling Software Suite
Version 2.1.7

Section 1: Overview

Distributed Replica (DR) sampling is a generalized ensemble simulation algorithm wherein sampling occurs amongst a collection of ensembles that are distributed along a predetermined reaction coordinate. During DR, sampling is generated by multiple replicas that each undergo a random walk along this reaction coordinate, weakly coupled by the Distributed Replica potential energy function (DRPE). For a more complete description of DR sampling, see T. Rodinger, P.L. Howell, and R. Pomès, "Distributed Replica Sampling" J. Chem. Theory Comput., 2:725 (2006).

The Distributed Replica Sampling Software Suite (DRSSS) is an implementation of DR sampling. The purpose of this document is the description of the Distributed Replica Sampling Software Suite.

DRSSS completely abstracts the implementation of the molecular dynamic simulation (MD) or Monte Carlo simulation (MC) from the DR sampling and the DRPE. Users may therefore combine the DRSSS software with any simulations package (for example: GROMACS, CHARMM, or NAMD). While this level of abstraction is useful, it means that the end user is responsible for a significant level of scripting to control the actual MD/MC simulation and to then provide the relevant information in formatted files that will be transmitted back to the server by the DRSSS.

Section 1.1: References and Suggested Reading

Please read and cite the following:

1. T. Rodinger, P.L. Howell, and R. Pomès, "Distributed Replica Sampling" J. Chem. Theory Comput., 2:725 (2006).

Please read the following and cite them if the new ideas introduced therein are utilized in your studies:

2. T. Rodinger, P.L. Howell, R. Pomès "Calculation of absolute protein-ligand binding free energy using distributed replica sampling" The Journal of chemical physics, Vol. 129, No. 15. (21 October 2008)
3. C. Neale, T. Rodinger, R. Pomès, "Equilibrium exchange enhances the convergence rate of umbrella sampling", Chemical Physics Letters 460, 375-381 (2008).
4. R. M. Henry, C-H. Yu, T. Rodinger, R. Pomès, "Functional Hydration and Conformational Gating of Proton Uptake in Cytochrome c Oxidase", J. Mol. Bio., 387, 1165-1185 (2009).
5. S. Rauscher, C. Neale, R. Pomès, "Simulated Tempering Distributed Replica Sampling, Virtual Replica Exchange, and Other Generalized-Ensemble Methods for Conformational Sampling", J. Chem. Theory Comput., 5 (10), 2640-2662 (2009)

For updates of publications utilizing DR sampling, please visit www.pomeslab.com.

Section 1.2: Expectations of the User

1. Competency with your selected MD/MC simulation software
2. Excellent understanding of free energy calculations
3. Competency in any single scripting/programming language (your choice which one).

You will require the above skills in order to configure the DRSSS for your system of interest. The first requirement is simply that you are able to set up the independent runs for your system using your selected simulation software. The second requirement is due to the fact that both the program and

this manual are written for advanced users. Free energy calculations are difficult, and it is essential that you know what you are doing. An explanation of the concepts and potential pitfalls that underpin such calculations is outside the scope of this document. The third requirement is based on the fact that you will need to automate the process of running jobs and post-simulation processing of the data from these jobs in order to obtain the observables along your reaction coordinate that must be provided to the server during the course of your DRSSS run. If you are using CHARMM for your simulations, then expectation 3, above, may be replaced by a competency in scripting directly within a CHARMM input file.

Section 1.3: Terminology

This document is intended as a quick reference guide and remains a work in progress. There may be concepts in this document that are not explicitly defined. We have attempted to define all DR-related concepts well enough for you to use this software. However, omitted terms and ill-defined concepts herein can probably be understood with a quick google search.

Section 1.3.1: Terminology Used in this Document

Simulation: The portion of your run that consists of sampling phase space. This may be *via* MD or MC or any other method that you choose. MC *sampling* is possible and MC is also a type of replica *transition* (see later). It is important to recognize this difference.

Reaction coordinate: An absolutely or partially discretized degree of freedom along which your simulation samples multiple values. The aim of DR is to achieve a random walk along this coordinate or parameter. This degree of freedom may be a function of phase space (i.e. some characteristic of positions or momenta), or may be extraneous to it (e.g. temperature, pressure, a fourth spatial dimension).

Replica: a continuous simulation. That is, a simulation whose atomic trajectories are continuous whereby the starting coordinates of one segment are the exact same as the final coordinates of the previous segment. This does not imply that replicas contain kinetic information; they do not. Metropolis exchanges along any reaction coordinate invalidate any kinetic information.

Nominal position: a defined position along a reaction coordinate at which sampling may occur, or toward which sampling is biased (e.g. harmonically as in umbrella sampling).

Replica vs. Nominal position: A DR run begins with one *replica* at each *nominal position*. During the course of DR sampling, a given *replica* migrates along the reaction coordinate and samples phase space at different *nominal positions*. Likewise, a given *nominal position* is sampled by many *replicas* over the course of the simulation, possibly by multiple replicas concurrently, and is possibly not sampled for periods of time.

Transition: A change of nominal position occupied by a replica. In the replica exchange method, this would be called an *exchange*. However, during DR, there is only one replica moving and hence we refer to it as a replica *transition*.

Sequence Number: each segment of simulation (MD or MC depending on your implementation)

between intervening replica transition attempts. The general progression for each replica is therefore sequence number i -> transition attempt i -> sequence number i+1 -> transition attempt i+1, and so on.

Section 1.4: Units!!!!!!

The DR_server works in energy units of kcal/mol. This is essential for the user to remember. Therefore umbrella sampling force constants and system potential energies must be reported to the server in kcal/mol. Note that many simulation packages utilize kJ/mol and this will require that you make some conversions in your client script. We have considered allowing the user to define their units in the configuration file but decided that this could lead to even larger problems if the value is not set properly. Therefore we have maintained the strict distinction that the DRSSS package must communicate all energy values in kcal/mol and it is the responsibility of your simulation script to make any necessary conversions.

Care about units is required in the following places (at least):

1. When you specify a force constant in the .script file if you are using an umbrella reaction coordinate.
2. In the .force and (especially) the .energy file if you are using a temperature or spatial reaction coordinate.

Section 1.5: Types of Simulations that can be Run

Any DRSSS run has 2 basic options: i) the class of reaction coordinate, and ii) the transition type.

Section 1.5.1: Choice of Reaction Coordinate Type

There are 3 basic classes of reaction coordinate: *Temperature*, *Umbrella*, or *Spatial*. Currently, these reaction coordinate classes are mutually exclusive.

- **TEMPERATURE**: The first option for reaction coordinate type is *temperature*. When using temperature as your reaction coordinate, simulations will be conducted at a variety of temperatures. Each continuous simulation, or replica, is allowed to migrate in temperature space with the intention that a random walk in temperature space results in a random walk in potential energy space (as is directly accomplished in MUCA). This type of simulation is often used to assist the crossing of enthalpic barriers which are unknown or complex, and is generally restricted to systems that do not disassemble quasi-irreversibly at your highest temperature. The resulting data is generally processed using WHAM.

- **UMBRELLA**: The second option for reaction coordinate type is *umbrella*. When utilizing an umbrella reaction coordinate, phase space is divided into a series of overlapping umbrella sampling simulations whose harmonic restraining potentials restrain the sampling at each nominal position to within a region of phase space along the reaction coordinate. The exact nature of the reaction coordinate is up to the user; it could be a dihedral angle, a distance between atoms or centers of mass, or any other physical quantity to which you can apply a harmonic restraint. The resulting data is generally processed using WHAM or, less commonly, FEP. In GROMACS terminology, an umbrella reaction coordinate corresponds to the application of a set of *restraints*. It is essential to understand that the DR_server expects that your harmonic restraining potential uses a force constant preceded by a ½ multiplier. That is, your harmonic restraining potential should be of the form:

$$U(R_i, \xi_{i[0]}) = \frac{1}{2} K_i (\xi_i - \xi_{i[0]})^2 \quad (1)$$

that biases sampling from the current value along the reaction coordinate, ξ_i, toward the center of restraint, $\xi_{i[0]}$, in proportion to a constant K_i, all based on the current system coordinates R_i. While this is just standard umbrella sampling, it is key to note the presence of the ½ multiplier to K_i. This ½ multiplier is utilized in GROMACS, but not in CHARMM. CHARMM users must therefore supply the DRSSS with a value of K_i that is double the value that their script passes to CHARMM.

- **SPATIAL**: The third option for reaction coordinate type is *spatial*. When utilizing a spatial reaction coordinate, phase space is discretized along the selected reaction coordinate by removing that reaction coordinate as a degree of freedom. Sampling at each nominal position represents sampling of a quantized position along the continuous reaction coordinate. The resulting data is generally processed using TI or FEP. In GROMACS terminology, a spatial reaction coordinate corresponds to the application of a set of *constraints*.

The exact nature of spatial and umbrella reaction coordinates is actually not restricted to the regularly existing phase space. For example, a fourth spatial dimension is amenable to this methodology using a modified version of CHARMM, as is the lambda parameter of the double decoupling method implemented in GROMACS. Currently, both of these two examples must utilize a *spatial* reaction coordinate type as opposed to an *umbrella* reaction coordinate type, although that is not a limitation of the DRSSS program itself.

Section 1.5.2: Choice of Transition Method

There are five basic types of transitions currently implemented in this DRSSS package: Monte Carlo (MC), Boltzmann, Continuous, Virtual Replica Exchange (VRE), and No Moves. Currently, these transition types are mutually exclusive.

- **Monte Carlo**: The first option for transition method is *Monte Carlo*. The MC method is basically simulated tempering (ST) with the addition of the Distributed Replica Potential Energy function (DRPE). Details of A-value determination/input will be discussed later. MC transitions with the DRPE are introduced generally in the original DR paper T. Rodinger, P.L. Howell, and R. Pomès, "Distributed Replica Sampling" J. Chem. Theory Comput., 2:725 (2006). MC transitions in temperature are *more formally* introduced as STDR in S. Rauscher, C. Neale, R. Pomès, "Simulated Tempering Distributed Replica Sampling, Virtual Replica Exchange, and Other Generalized-Ensemble Methods for Conformational Sampling", J. Chem. Theory Comput., 5 (10), 2640-2662 (2009)

- **Boltzmann**: The second option for transition method is *Boltzmann*. The Boltzmann method is very similar to the MC method, with one important exception: transitions from the current nominal position, i, are not limited to nominal positions i-1 or i+1. The probability of undergoing a transition from the current nominal position to each of the listed nominal positions is calculated and the transition (or lack thereof) is determined by these normalized probabilities. In this method, the majority of successful transitions from nominal position i will still be to nominal positions i-1 or i+1, but there will exist some transitions that move the sampling

position by a larger amount and this will enhance your overall sampling efficiency. Boltzmann transitions were introduced (as “Boltzmann-weighted jumping”) in equations 7 and 8 of T. Rodinger, P.L. Howell, R. Pomès "Calculation of absolute protein-ligand binding free energy using distributed replica sampling" The Journal of chemical physics, Vol. 129, No. 15. (21 October 2008).

- **Continuous**: The third option for transition method is *continuous*. The continuous method builds on the Boltzmann method by removing the concept of a fixed nominal position. Sampling then occurs at all points along the reaction coordinate. This is probably more efficient than MC or Boltzmann transitions, but reduces the flexibility in data analysis, forcing you to apply TI as there are no other known methods that can utilize this type of data (currently). Continuous transitions are currently unpublished.

- **Virtual Replica Exchange**: The fourth option for transition method is *virtual replica exchange* (VRE). This method is very similar in concept to RE and SREM. The equation itself is basically that of SREM. The difference is that there are no histograms from which a-values are drawn. Instead, a list of virtual reverse-moves is stored and from this list a value is randomly selected during each transition attempt. More specifically, each time a replica completes a sequence number, the final value that is used for the actual transition attempt is also stored in a list for the appropriate nominal position, from which values are drawn subsequently as virtual exchanges. There exists a primary list and a secondary list from which virtual exchange values may be drawn. Values in the primary list are linked to the replica from which they were generated and a value in the primary list is never used in conjunction with the replica from which it was derived. Once a value is used from the primary list it is removed and placed in the secondary list. The intention of the secondary list is to handle cases where the primary list runs out of values. If a large primary list is built up prior to allowing transitions to occur, the secondary list may never be used. VRE was introduced in S. Rauscher, C. Neale, R. Pomès, "Simulated Tempering Distributed Replica Sampling, Virtual Replica Exchange, and Other Generalized-Ensemble Methods for Conformational Sampling", J. Chem. Theory Comput., 5 (10), 2640-2662 (2009). However, the implementation here is somewhat different in that there is a primary and a secondary list of VRE reverse move possibilities in order to avoid reusing a value by the same replica that generated it.

- **No Moves**: The fifth and final option for transition method is *No Moves*. When no moves is selected, transitions are never attempted. This is therefore not DR and is not even an extended ensemble simulation. Advanced users may find this option useful to control a large number of simultaneous simulations on a heterogeneous computing cluster and we have included it only because it might be useful to some users and it comes basically for free since the communication framework is already in place.

Section 1.5.3: Combinations that are Disallowed

It is not possible to combine a spatial reaction coordinate with continuous transitions. This is because the difference in potential energy depending on the reaction coordinate must be a known function. While this is possible for a harmonic restraining potential, it is not possible for an absolute constraint that defines the exact position of the replica along the reaction coordinate.

Section 2: How the DRSSS Program Works

Section 2.1: Server-Client Architecture

The core of the Distributed Replica Sampling software suite (DRSSS) consists of only two programs: a server and a client. The server is the DRSSS program named DR_server. The client, however, is predominantly a shell script that must be written by the user. The reason that this software package does not include a complete client is that we have endeavored to provide an abstract toolkit that is compatible with all molecular dynamics and monte carlo sampling algorithms that now exist or may be developed in the future. We do provide you with a tool named DR_client_comm that can be utilized within your client script to contact the server for the purposes of providing information about the recently finished sampling and to obtain information about the sampling that is to be conducted next. We also provide an example bash script client that can be used as a template for your studies. The benefits of this client abstraction will quickly become obvious to users in groups that utilize multiple simulation packages. In our group, some users employ a bash script as a client to run GROMACS and manage information, while other users simply use a single CHARMM run as the client and take advantage of the fact that CHARMM has built in scripting and shell execution capabilities that may be used to advantage within a large input file.

The server requires very few computational resources, but must be running on a computer that is capable of dispatching the resource intensive clients. If this software is to be run on a single large computing cluster, the server is usually run on the head node where it can submit jobs via your queuing system and generally does not occupy enough resources to bother any other users.

The server is responsible for dispatching jobs, each of which acts as a separate client. The server also instructs each client regarding what sampling is to be done and sends the client the appropriate files necessary for restarting a simulation. The server also stores data generated by the clients in a format that is convenient for later data analysis. The client is responsible for carrying out the actual sampling of phase and for all data manipulation. Communication between client and server is always initiated by the client. There is therefore no need for the server to know the IP address on which clients are running. This unidirectional client -> server initiation of communication also facilitates the use of DR on distributed computing platforms.

Section 2.1.1: Schematic Timeline

1. Generate the package.tar.gz file
2. Start running DR_server.
3. The server inspects the .script file and prepares for a run according to the options that you have specified.
4. The server creates a .forcedatabase file, in which sampling and restart data will be stored.
5. The server dispatches N instances of your client wrapper script, DR_client_wrapper, via the drsub script.
6. Server listens on specified port for connecting clients.
7. One client wrapper script starts execution via your queuing system,
8. DR_client_wrapper calls the get_simulation_package program to obtain generally required files specific to your system, a copy of your DR_client_body script, and a small file that indicates the port on which the server is listening.
9. DR_client_wrapper executes the DR_client_comm program, using the specified port and IP, to contact the server. During this connection, the client sends the server a signal that this is a new client by sending the '**' flag as the third command line argument to DR_client_comm. The

server responds by sending the client a file called setup that defines the replica related variables for the particular sequence number to be simulated. If there are no jobs to run at this time, the setup file indicates that this particular client should shut down and exit.

10. DR_client_wrapper executes the DR_client_body script that was obtained via the previous call to get_simulation_package. If one is running CHARMM in place of GROMACS, then the DR_client_body script would be replaced by a long .inp file that would be passed to CHARMM inside the DR_client_wrapper script and this .inp file would handle all functions here ascribed to the DR_client_body script.
11. The DR_client_body script parses the setup file obtained from the server.
12. The body of the DR_client_body script is a loop that cycles until the job variable equals -1, in which case it will exit. If there are no jobs to run, the server will send a setup file that will set job to -1. Within this loop inside DR_client_body, a number of steps are carried out:
    12.1. First, any template files are modified as prescribed by the setup file. This includes the .mdp file.
    12.2. Second, and only if this is not the very first sequence number for the given replica, the .rst pseudo-restart file (obtained from the server on the previous DR_client_comm call) is unzipped and untarred to yield .trr and .edr files. Note that when using CHARMM in place of GROMACS, this .rst file would be a bona-fide CHARMM restart file. In the case of GROMACS, multiple files are required for a restart and they have been put together into a single zipped tar file names as a .rst in order to abstract the restarting process.
    12.3. Next, grompp is run using the correct .gro file. If this is not the very first sequence number, then .trr and .edr files are used to obtain a correct restart. These files are obtained from the .rst file when required.
    12.4. The mdrun program is used to sample phase space.
    12.5. Output files are checked for integrity.
    12.6. Data is extracted from mdrun output files if it was not produced directly to a .xvg file during mdrun. In any event some general data processing is required here.
    12.7. Data is processed to create .force, .energy, and, optionally, .add# files.
    12.8. The .gro, .trr, and .edr files are tarred and zipped into a .rst file.
    12.9. the “set job -1” flag is appended to the setup file and the DR_client_comm is invoked in order to contact the server. If communication with the server is successful, a new setup file is sent. If client-server communication is unsuccessful, then a new setup file is not sent and the job = -1 condition causes this DR_client_body script to exit its main loop and this client DR_client_wrapper quickly shuts down after a little bookkeeping. During this connection, the client sends the server a signal that this is a running client by sending the job identifier as the third command line argument to DR_client_comm. The server responds by sending the client a file called setup that defines the replica related variables for the particular sequence number to be simulated. If there are no jobs to run at this time, the setup file indicates that this particular client should shut down and exit.
    12.10. Recycle the loop; exit if the job variable equals -1.
    12.11. NOTE: any “set MESSAGE <variable>” values that are put into the setup file by the server for the client to take action on must be handled by the DR_client_body and, possibly, the DR_client_wrapper. Currently, the only messages are related to the *Mobile Server* option.

Section 2.2: Methodologies of Server-Client Communication

The communication between the server and the client is relatively simple, yet there is nothing intuitive about it. Still, understanding what is going on behind the scenes is essential if you are to correctly implement your client side. This sub-section discusses the communication strategies that we apply and the general features of the information that is transmitted. More detailed information on the exact contents of each file type is provided in section entitled File Descriptions, as the requirements for the contents of each file may differ depending on options in your .script file.

When you call DR_client_comm, you give it command line arguments that specify the IP on which the DR_server is running and the port on which the DR_server is listening. You also specify the job variable that enumerates both the replica and the sequence number that has just completed sampling. The '**' job identifier is used to signal the server that this is a new client looking for a job to run. The communication process is broken up into turns. Every call to DR_client_comm goes through each of these turns, listed below.

Turn 1 (client): DR_client_comm attempts to contact DR_server.

Turn 2 (server): DR_server accepts client connection as a new thread.

Turn 3 (client): DR_client_comm sends job identifier to DR_server. If this job identifier is '**', this turn is complete. Otherwise, DR_client_comm looks to see if particular data files exist on the client side, and sends them to the DR_server over the socket if they do exist. These files contain information that the server needs in order to determine the transition probabilities and also information that is to be stored in the forcedatabase by the server for later analysis; they are the .force, .energy, and .add# files as will be discussed later. The DR_client_comm program sends these files in a specific order, but does not complain if files are missing or if it is sending files that it should not be sending. It is up to your client script (DR_client_body or a CHARMM .inp file) to create all of the necessary files and not to create any that are not required. This is the main reason that the user must understand how this communication process works: you need to know what files to create based on the options that you select in the .script file and you need to know exactly what to put in each file. Nevertheless, the server knows what files it expects to receive and will throw an error if it does not get those files. Therefore, if your log file indicates that the server did not receive a specified file, then it is most likely that your DR_client_body did not produce the file (or name it correctly), and it was therefore not sent to the server by DR_client_comm.

Turn 4 (server): DR_server receives files in expected order and checks the integrity of each file in turn. If an expected file is missing, or if an unexpected file is found, the DR_server notices the error and tells the DR_client_comm to instruct its client to quit. If the job identifier was '**' then this step is skipped as there are no files to be sent from the client to the server at this stage.

Turn 5 (server): DR_server sends a restart file to DR_client_comm. If the job identifier was '**' then this step is skipped as there are no restart files existing at this stage.

Turn 6 (server): DR_server sends a file named setup to DR_client_comm. This setup file contains a list of parameters that will be required by the client in order to execute the simulation.

Section 2.3: Limitations

Section 2.3.1: DRSSS Program Limitations

Currently, the initial structure files must either be included in the initial package along with the DR_client_body script so that the client can obtain them via get_simulation_package, or the initial structure files must be simply pulled from a common directory if you are using a single cluster for all clients. This is not optimal as it means that either you are communicating startup coordinates even

when they are no longer necessary (first option above), or that you are using a method that is not entirely amenable to a distributed computing platform. We do intend to fix this at some point in the future.

Section 2.3.2: Computing Cluster-based Complications

One issue that you may experience when running the server on a shared interactive node is a memory limitation if your system is large. This is because the server must keep an entire set of restart information for all replicas in memory at all times. While this is unlikely to overwhelm the entire set of RAM on your head node, it would be inconsiderate to use most of the available RAM on a long-term basis. Further, many clusters limit the memory usage on a per user basis to somewhere around 10%. It is likely that you will run into this issue for large systems. For example, a simulation system composed of 250,000 atoms using 40 replicas, while running GROMACS in single precision, yields a DR_server memory usage around 800 MB, which is >=10% of the RAM available on most head nodes (we generally see 4 GB to 8 GB total RAM). In such cases we have found it may be necessary to contact your system administrator and request that they set up a special queue for you on which you can run your server with special permissions. When you talk with them, let them know that you need unlimited wall clock time, a CPU time limit of one hour or greater, and that they can easily overload this node with other jobs as you only need it for the memory and will be using << 5% of the overall processing power.

Another issue that we have run into is that the CPU time limit on the head node, while unlimited on some clusters, can be as low as 5 minutes on some clusters in order to discourage users from running resource-intensive processes on the interactive login nodes. In this case, you could contact your system administrator, but we have developed a work around for you. The file autorun.sh provided in the scripts/ directory can be used to automate the process of i) starting the server, ii) shutting down the server and saving a system state, iii) restarting the server using the saved system state, and cycling through this procedure. However, if your CPU time-limits are small compared to the resources that your queuing system allocates to your clients, then this script may lead to significant inefficiencies due to the need to re-queue. Note that the *Mobile Server* and Client Cycling have been implemented to offer an efficient method, but that user implementation in the DR_client_body are somewhat complicated for a novice user – we recommend that you learn the DRSSS first without attempting to apply the *Mobile Server*, and then add that part in later.

Section 2.4: Client Cycling

On some clusters, each of your clients will have a limited walltime. Since the server is constantly resubmitting new jobs, it would be a waste to turn away a fresh client and maintain an old client whose walltime allocation is about to expire. Therefore, you can cycle through clients by using the CYCLE_CLIENTS script keyword (see the script file section for more details). This option is usually combined with the mobile server (see below).

Section 2.5: Mobile Server

On some clusters, you may not be allowed to run the DR_server on the headnode and therefore your server will be limited to a specific walltime. In this case, it is desirable to allow the server to take over a relatively new client (you'll probably have one if you allowed client cycling). If server mobility is allowed in your script file, then the server will send special messages to your clients in the setup file. It is necessary for you to add message handling in your clients to accommodate this. There are two

messages that might be sent. After client-server communication, the setup file may contain the BECOME_NEW_SERVER message:

set MESSAGE BECOME_NEW_SERVER <snapshot>

where <snapshot> is the name of the snapshot file that must be loaded when the new server is started. Alternatively, (and mutually exclusively) the setup file may contain the HOLD_AND_CONTACT message:

set MESSAGE HOLD_AND_CONTACT <ip>

where <ip> is the IP address on which the client can find a new server.

Once the server sends out these messages, the regular run on that server has finished: the snapshot has already been written and the forcedatabase file is closed. The server will stay alive until it has been contacted by all living nodes so that all messages are received by the clients. Once that has happened, the original server will exit.

The new server, meanwhile, running on the node that received the BECOME_NEW_SERVER message, should have been started by the DR_client_body (or DR_client_wrapper). The existing clients will contact the new server using the '**' flag and will appear as entirely new clients (although the server will not be aware, unfortunately, that their remaining allotted walltime does not start at its maximum value). New clients will connect to the new server because the package has been remade to indicate the new IP.

To utilize a mobile server, you will need to appropriately set the SERVER_TIMELEFT_ENTER_MOBILE_STATE and ALLOTTED_TIME_FOR_SERVER keywords (and probably also the CYCLE_CLIENTS keyword) in your script file.

Section 3: Installation

Run the source/compileProg script to compile these programs on your machine. This script is not very complex, so you may need to modify it (For example if you don't want to use the g++/gcc compilers). Upon successful installation, your binary files should exist in a new directory, bin/.

Section 4: File Descriptions

Section 4.1: The .script File

Section 4.1.1: The Options Section of the .script File

The script file contains all of the parameters that describe your DRSSS run. The file name must begin with two characters and be followed by .script. The general format of the script file is a KEYWORD which sometimes must appear alone, sometimes must be followed by a value, or in some cases multiple values, associated with that keyword. Comments begin with a double slash, as in c++ programs.

Keyword SIMULATION

This keyword should be followed by two values <reaction coordinate type> and <transition type>

Possible combinations are listed below:

SIMULATION	Spatial	MonteCarlo	
SIMULATION	Spatial	Boltzmann	
SIMULATION	Spatial	NoMoves	
~~SIMULATION~~	~~Spatial~~	~~Continuous~~	Error: not supported
SIMULATION	Temperature	MonteCarlo	
SIMULATION	Temperature	Boltzmann	
SIMULATION	Temperature	Continuous	
SIMULATION	Temperature	NoMoves	
SIMULATION	Umbrella	MonteCarlo	
SIMULATION	Umbrella	Boltzmann	
SIMULATION	Umbrella	Continuous	
SIMULATION	Umbrella	NoMoves	

Keyword REPLICASTEP

This keyword should be followed by one real number. If your move type is MonteCarlo, you must specify the step size in terms of fraction of distance to next replica. The value 1.0 is required for standard WHAM data analysis.

Keyword TEMPERATURE

If your simulation type is 'Spatial' or 'Umbrella', you must specify the temperature at which your simulation will run. The TEMPERATURE keyword should be followed by a real number temperature in Kelvins.

Keyword SUBMITJOBS

For automatic client jobs submission via "drsub working_directory/QueuedJob.run", include the SUBMITJOBS keyword. This keyword should appear alone on a line in the script file (no arguments). The contents of drsub will be discussed later, but basically it must be able to take the script name as a command line argument and submit that script to your queue. Do not include the SUBMITJOBS keyword if you are using DR_tester.

Keyword PORT

Identify the port over which the server will listen for client connections after the Port keyword

Keyword CIRCULAR

If your reaction coordinate is circular, include the CIRCULAR keyword followed by two real numbers. This keyword enables exchange between the upper and lower nominal positions. A 'nominal position' is the position at which a replica begins the simulation. This feature has only been tested for SIMULATION Umbrella MonteCarlo. Let these two numbers be called A and B and the entire length of your coordinate is L (L=B-A). Then A, B, and L must have the following relation: A + L = B. Further, the distance between A and your smallest nominal position should equal the distance between B and your largest nominal position. For example, if your nominal positions range from -180deg to 180deg then you should use CIRCULAR -175.0 185.0. However, if your nominal positions range from 0deg to 350deg then you should use CIRCULAR -5.0 355.0. and if your nominal positions range from 10deg to 360deg then you should use CIRCULAR 5.0 365.0. If a replica moves to a value less than A,

then L is added to the position. If a replica moves to a value more than B, then L is subtracted from the position. Note too that you can 'circularize' some coordinates in interesting ways such as linearly across periodic boundary conditions assuming that your molecular dynamics engine understands what you are asking it to do.

SPECIAL NOTE: There is a coding requirement that the distance between nominal position 0 and nominal position 1 equals the distance between nominal position N-1 and nominal position N equals the distance between nominal position N and nominal position 0 (along the shortest distance when circularized). Other usage may lead to erroneous simulations where sampling migrated away from nominal positions, even during MC transitions.

Keyword NEEDSAMPLEDATA

If you want the client to send a file containing forces back to the server, then include the NEEDSAMPLEDATA keyword. This keyword should appear alone on a line in the script file (no arguments). When the NEEDSAMPLEDATA keyword is specified, your simulation must produce a file called ${job}.force.nni1 that contains as many force values as there are integration steps specified in the 'JOB' section at the end of this file. If you don't know the force value at every integration step, then you can trick the program by specifying a certain number of steps in this script file but a different (larger) number of steps directly in your Molecular Dynamics run file. This keyword is required if you are going to use 'CANCELLATION' because the cancellation profile will be generated based on these values. This is also required if you are going to use 'ADDITIONALDATA' because that keyword overloads NEEDSAMPLEDATA functionality. If you don't want to send force data, then simply comment out 'NEEDSAMPLEDATA' so that it reads '//NEEDSAMPLEDATA'

Keyword NEEDCOORDINATEDATA

If you want the client to send a file containin coordinates back to the server, then include the NEEDCOORDINATEDATA keyword. This keyword should appear alone on a line in the script file (no arguments). This can be useful, but C. Neale is not entirely sure how it works. If you don't want to send coordinates, then simply comment out 'NEEDCOORDINATEDATA' so that it reads '//NEEDCOORDINATEDATA'

Keyword ADDITIONALDATA

If you want to send additional data back to the server, then include the ADDITIONALDATA keyword followed by one integer number. When the ADDITIONALDATA keyword is specified, your simulation must produce a file called ${job}.add#.nni1 that contains as many values as there are integration steps specified in the 'JOB' section at the end of this file. The numbersign in the name of the file ${job}.add#.nni1 should be replaced with an actual integer number. For example, if you include 'ADDITIONALDATA 3' in this script file, then you are expected to produce 3 files named ${job}.add1.nni1, ${job}.add2.nni1, and ${job}.add3.nni1. Each of these files will have the format of the ${job}.force.nni1 file but contain different information. It is not possible to use ADDITIONALDATA without also using NEEDSAMPLEDATA. As an example, while using SIMULATION Umbrella MonteCarlo to determine the free energy profile for rotation about a given dihedral angle, the ${job}.force.nni1 file will contain forces. But in order to actually track the dihedral angle values for use with WHAM, it is necessary to use 'ADDITIONALDATA 1' and then create a file ${job}.add1.nni1 that contains the dihedral angle values sampled. One could also use 'ADDITIONALDATA 2' and create the additional file ${job}.add2.nni1 that contains some measure from an orthoganol degree of freedom if that data is considered useful. This has a compile-time

maximum of 10, but that could easily be changed. Set 'ADDITIONALDATA 0' if you do not want to use this feature.

Keyword N_SAMESYSTEM_UNCOUPLED

In rare cases, one may wish to have each simulation system contain more than one non-interacting replica. For example, C. Neale implemented this for a protein-lipid bilayer system in which one protein is inverted along the bilayer normal to the other in order to reduce the anisotropy of the system. In this case, one would set 'N_SAMESYSTEM_UNCOUPLED 2' then set the 'JOB' definition up normally. The number of full systems that will be run is N_JOB / N_SAMESYSTEM_UNCOUPLED, which must divide with a zero remainder. The systems will be expected to initially be consecutive, i.e. The first system contains replica 1 and 2, the second system contains replica 3 and 4, etc. However, the systems will evolve independently during the run since they are defined to be non-interacting. The default value of this is 1 and the field can be safely commented out for regular runtime behaviour. Note that when you have N_SAMESYSTEM_UNCOUPLED 1 (or commented out) then all of your ${job}.force.nni1, ${job}.energy.nni1, ${job}.add#.nni1, etc, files are named as Number Non-Interacting 1 (.nni1). If you have NNI=2, then you must also have ${job}.force.nni2, ${job}.energy.nni2, ${job}.add#.nni2 files, and so on.

Keyword POTENTIALSCALAR

Specify the POTENTIALSCALAR keyword followed by two real numbers. These numbers are the two constants that are used in the calculation of the distributed replica potential energy function (the DRPE). The first number relates to spread of the replicas and the second number relates to an overall shift of the replicas. In both cases, these numbers are force constants and, as such, larger numbers will further restrict uneven spreading and overall shifting than smaller numbers will. Note that the number of moves is reduced as these force constants are increased so there is a desire to keep them as low as possible and this maximize the number of moves. For general usage, try this 'POTENTIALSCALAR 0.004 0.1'. For use with 'CIRCULAR', the second value must be zero, therefore try 'POTENTIALSCALAR 0.004 0.0'. If you are using 'CANCELLATION' then the above information is incorrect/incomplete. It is essential that you understand how POTENTIALSCALAR relates to CANCELLATION if you are going to apply CANCELLATION.

Keyword CANCELLATION

By invoking the CANCELLATION keyword, you are asking the DRSSS to go through a well-defined procedure. This procedure will be laid out below assuming that you have used 'POTENTIALSCALAR P1 P2' and 'CANCELLATION C1 C2 C3' where P1, P2, C1, and C2 are all real numbers, and C3 is an integer.

1) Run the DRSSS using the P1 and P2 values as force constants for the DRPE until each nominal position has run C3 simulation segments. Note that some nominal positions may run more than C3 if some CPUs are faster than others.
2) Use the values from the ${job}.force.nni1 values to set up an energy profile that will be used for 'cancellation'. Basically this is like the A-values in simulated tempering.
3) Continue the simulation using C1 and C2 as force constants for the DRPE instead of P1 and P2.

In order to effectively use 'CANCELLATION', it is recommended that you use P1=1000.0. This value is large enough to inhibit all moves and sample each nominal position equally. Therefore the following settings are suggested: POTENTIALSCALAR 1000.0 0.0 and CANCELLATION 0.004 0.0 500.

Note that C. Neale thinks that 50 steps should be enough as long as you are sending >20 relatively uncorrelated forces in each ${job}.force.nni1 file and your simulation is relatively simple. You may require significantly more. Note that 'CANCELLATION' can be avoided through the use of 'CANCEL' in the 'JOB' section at the end of this file – in this usage you can specify the A-values / cancellation values yourself. Specify 0 as the third parameter to prevent energy cancellation, or comment-out this command

Keyword VRE_INITIAL_NOMOVES

If you are using Virtual Replica Exchange (VRE) as the transition methodology, then you can specify an initial number of sequence numbers for which no exchanges will be attempted. This is useful if you want to build up the primary VRE lists. CN uses 150 here.

Keyword VRE_INITIAL_NOSAVE

If you are using Virtual Replica Exchange (VRE) as the transition methodology, then you can specify an initial number of sequence numbers for which values will not be saved to the primary (or secondary) VRE list. This is useful if you know that your starting structures are highly non-equilibrium and you want to simply automate the procedure of some initial equilibration. CN uses 100 here.

Keyword VRE_SECONDARY_LIST_LENGTH

If you are using Virtual Replica Exchange (VRE) as the transition methodology, then you can specify the length of your secondary list. CN prefers a shorter list so that older values are overwritten more quickly and the chance of using pre-equilibration values in a post-equilibration phase is less likely. CN uses 500 here.

Keyword NODETIME

Use the NODETIME keyword to specify the approximate time (in seconds) that a node will be occupied before re-queueing. Once the NODETIME has been exceeded, your current iteration will gracefully finish but that node will be released and another job will be queued. The point of this is to avoid hogging a node in a shared environment. It can also be useful to have a smaller number here if you have a few nodes that are significantly slower than the rest and, when all replicas are running, these replicas can get behind the others in terms of sampling.

Keyword REPLICACHANGETIME

Use the REPLICACHANGETIME keyword to indicate that a change of replica running on a node can occur as often as this many seconds. If REPLICACHANGETIME is much larger than the time that a single iteration of MD requires, then many of the jobs on a single node will involve a single replica. That replica may change nominal position, but that particular replica will be followed. This is sub-optimal since some other replicas may not currently be running. Ideally, one would set REPLICACHANGETIME to zero so that when a replica finished a single iteration of MD then the server would assign the CPU to run whichever replica currently has run the least number of iterations. However, a replica-change involves the transmission of new restart files and the overhead is increased. If each segment takes much longer to run than the time required to transmit data, then set this value to zero. However, lower values here are going to increase overhead.

Keyword SNAPSHOTTIME

Use the SNAPSHOTTIME keyword to indicate the time interval between saving state snapshots (seconds). A snapshot file is created by DR_server at each time interval specified by SNAPSHOTTIME. This file contains all relevant information required for completely restarting a DRSSS job. This includes current number of replicas running and their nominal positions, current atom positions and velocities etc... If a DRSSS job needs to be restarted, possibly due to a server crash, this file may be used to resume the job without starting from the beginning. Note that these files become large over time. We therefore suggest that you save them often, but also manually delete them often as there should never be a need to use one that is not the most recently written snapshot.

Keyword TIMEOUT

Use the TIMEOUT keyword to specify the maximum time allowed for one sequence number to finish before we give up and restart it (seconds). This is highly specific to your simulation and CPU speed. You want this to be high enough that no running jobs are terminated but also low enough that the server can reliable resubmit jobs that have died without too much latency. Note that if this value is too low then you will waste a lot of simulation time by submitting each job twice.

Keyword ALLOW_REQUEUE

Use the ALLOW_REQUEUE keyword to allow nodes that have exceeded TIMEOUT to be reused. In some cases this may be inefficient as a node that is damaged will continually be used (perhaps it runs really slowly). However this will stop nodes from being released in the event that they were just preempted by some type of test queue.

Keyword RUNNINGREPLICAS

Use the RUNNINGREPLICAS keyword to manage the suspension of some replicas but not others. Replicas in this range (inclusive) will continue running until all replicas have finished the desired number of iterations. Replicas outside this range will stop if they had individually met the desired number of iterations. Use this with care. This value defaults to the entire range so users are urged to avoid using this keyword at all unless they require a specialized usage. It is very easy to set it to the entire range and then forget to reset this value in your script file when you develop a new, larger, system – and you would then not get the behaviour that you expect.

Keyword STOP_ON_AVERAGE_TIME_EXCEEDED

Add the STOP_ON_AVERAGE_TIME_EXCEEDED flag to indicate that you want the run to terminate when the total number of sampling steps completed exceeds the total number of sampling steps requested. This keyword should appear alone on a line in the script file (no arguments). This is useful when comparing different methodologies, some of which may have severe difficulty when sampling particular nominal positions and the run may continue basically forever.

Keyword CYCLE_CLIENTS

Add the CYCLE_CLIENTS keyword, followed by a real number between 0.0 and 1.0 to allow new clients to kick out old clients when the complete set of replicas is running. On some clusters, each of your clients will have a limited walltime. Since the server is constantly resubmitting new jobs, it would be a waste to turn away a fresh client and maintain an old client whose walltime allocation is about to expire. Therefore, you can cycle through clients by using the CYCLE_CLIENTS script keyword. The real value that follows the keyword specifies the proportion of the old client's .script

assigned NODETIME that must complete before the client is allowed to be kicked out. The purpose of setting a limit here is that it is not efficient to kick out a very new client. *Note that if you do not set the NODETIME to match the walltime limit in your DR_client_wrapper, then this feature may not work as you expect.* If you do not wish to allow client cycling, then either omit this keyword (e.g. comment it out), or set its value less than -0.1.

Keyword DEFINE_STARTING_POSITIONS

Add the DEFINE_STARTING_POSITIONS flag if you want to have your replicas start in some way other than as the nominal positions are setup. You might want to do this if you are switching from another REMD program. If this flag is set, then your working directory must contain a file called switchStart.txt. This file must contain a single column of integers ranging from 0 to N_replicas-1. If you were to set these numbers as 0, 1, 2, ... , N-2, N-1, then the run would be started in the same way as if you had not used this flag. Using a temeprature reactino coordinate as an example and recalling that temperatures are defined from highest (=nominal position 0) to lowest (=nominal position N-1), setting values 0, 0, 2, 3, ... , N-2, N-1 would start two replicas at the highest temperature, none at the second highest temperature, and regularly from the third highest to the lowest temperature. You can use the output header in the .log file to ensure that you are getting what you want -- just remember that the output starting positions are given as beta values (=1/KT).

Keyword SERVER_TIMELEFT_ENTER_MOBILE_STATE

This is a *Mobile Server* option. Add the SERVER_TIMELEFT_ENTER_MOBILE_STATE keyword, followed by an integer to allow the server to kick out a client and move the server to the node on which that client was running. On some clusters, your server will have a limited walltime. Since the server is constantly resubmitting new jobs, it would be a waste to disconnect all of those clients, start a new server, and restart all of the clients. The integer value that follows the keyword specifies that server mobility will be allowed only when there are SERVER_TIMELEFT_ENTER_MOBILE_STATE or fewer seconds remaining in the server's assigned ALLOTTED_TIME_FOR_SERVER (there is considerable overhead to this server node switch so we don't want to do it too often). This option requires that the ALLOTTED_TIME_FOR_SERVER is also defined.

Keyword ALLOTTED_TIME_FOR_SERVER

This is a *Mobile Server* option. Add the ALLOTTED_TIME_FOR_SERVER keyword, followed by an integer to stop the server from running after the defined number of seconds. This may be useful if you are running your server on a compute node in a cluster on which you have a limited walltime. This way, the server will exit cleanly before the allowed walltime is completed. If this is used in combination with SERVER_TIMELEFT_ENTER_MOBILE_STATE, then the ALLOTTED_TIME_FOR_SERVER represents the number of seconds that can be run safely within a single allocation of walltime and expects that your server and clients are all asking for a similar amount of walltime. There is no check that the NODETIME and ALLOTTED_TIME_FOR_SERVER are set the same because there is no direct check on your allocated walltime (which is in your DR_client_wrapper) in any event.

Section 4.1.2: The job definition section of the .script file

This section begins with a header line that specifies the order of the columns that you will define for each nominal position. This header must begin with the keyword COLUMN.

- The 'COLUMN' command is used to specify what the columns in the 'JOB' lines below represent the possibilities are:
- LIGAND1 : the nominal position of the ligand or of ligand 1 in the case of two ligands
- LIGAND2 : the nominal position of the second ligand when a substitution simulation is performed
- FUNNEL : the size (for example the radius) of a restraint in 'funnel simulations'
- FORCE : the force constant for an umbrella (MUST BE IN kcal/mol)
- MOVES : the number of sequence numbers to carry out for the replica
- STEPS : the number of MD or MC steps to perform between transition attempts
- CANCEL : the cancellation energy for this nominal position
- The 'JOB' commands list the parameters for each replica in order as specified by 'COLUMN'

Here is a short example:

COLUMNS	LIGAND1	MOVES	STEPS	FORCE
JOB	.30	2000	101	119.5
JOB	.40	2000	101	119.5
JOB	.50	2000	101	119.5

Section 4.2: Naming the .energy, .force, and .add# Files

These files are all produced by your main client script after each sequence number is completed but before calling DR_client_comm to communicate this information with the server. Naming these files correctly is essential.

First, all of these files must be named as ${job}.energy, ${job}.force, etc. This ${job} identifier is the one sent by the server in the setup file and is of the form <runID>w<replica number>.<sequence number>, with both number lists beginning at zero. Therefore the first replica running its first sequence number based on aa.script would be ${job} = aaw0.0 and the .energy file should be named as aaw0.0.energy.

Importantly, all of these files must also have a .nni# postfix. For most runs, all of these files will have a .nni1 extension, so that the .energy file would be aaw0.0.energy.nni1. This is because it is possible to have multiple noninteracting replicas within a single simulation (see the script file section of this document). In that case, you might have all of these files present as .nni1 files to pass information about the first noninteracting replica and another version of all of these files present as .nni2 files to pass information about the second noninteracting replica.

If you get confused here, the .log file generated by the DR_server should be useful to help you figure out what files you are missing or what files you are sending unexpectedly. This should assit you to figure out the naming conventions.

Section 4.3: The .energy File

The .energy file (actually the .energy.nni# file) contains the information that the server requires in order to determine transition probabilities. *There are a few simple rules:*

1. If your transition type is NoMoves, then this file must not exist.
2. If your coordinate type is Temperature, then this file must contain a single real number representing the potential energy of your simulation system in the final snapshot. **This energy**

must be reported in kcal/mol !!!

3. If your coordinate type is Umbrella, then this file must contain actual position along the reaction coordinate in the final snapshot.
4. If your coordinate type is Spatial and your transition type is MonteCarlo, then this file must contain two real numbers. The first represents the nominal position to which a transition is to be attempted, and the second represents the difference in potential energies for the system in the two states (dE=E_new-E_old).
5. If your coordinate type is Spatial and your transition type is Boltzmann, then this file must contain N real numbers, where N is the total number of nominal positions. Each value represents the dE for moving to one of the N nominal positions. This must be provided in the same order as these nominal positions appear in the .script file.

- *The above rules are summarized in the table below.*

COORDINATE_TYPE	TRANSITION_TYPE	.ENERGY CONTENTS	SIZEOF(.ENERGY)
Spatial	MonteCarlo	new_corr, system dE	2 floats
Spatial	Boltzmann	discrete Es	Nreplicas floats
~~Spatial~~	~~Continuous~~	Error: not supported	
Spatial	NoMoves	None	
Temperature	MonteCarlo	system E	1 float
Temperature	Boltzmann	system E	1 float
Temperature	Continuous	system E	1 float
Temperature	NoMoves	None	
Umbrella	MonteCarlo	position of what umbrella acts on	1 float
Umbrella	Boltzmann	position of what umbrella acts on	1 float
Umbrella	Continuous	position of what umbrella acts on	1 float
Umbrella	NoMoves	None	

Section 4.4: The .force File

The .force file (actually the .force.nni# file) should contain a list of the force acting by the system along the reaction coordinate. It may be possible to access this information directly, although this usually requires a modified version of your simulation package. It might also be possible to output the overall force on your selected atoms and then to process this data post-production to project this force onto your reaction coordinate. It is important to understand here that we are talking about the total system force, excluding the biasing force if you are applying a harmonic restraining potential. To be explicit, the forces contained in this file should represent the system contribution to the sum over LJ, coulombic, bond, angle, dihedral, and improper forces, but not the biasing forces, that act on your selected atoms, all projected onto your reaction coordinate. Since this information is not usually easy to obtain, it is possible to use an approximation to this whose time and ensemble average is correct, although the instantaneous values are not the same. In the case of an umbrella reaction coordinate, we can make use of the fact that some equilibrium position exists at which the force of the harmonic biasing potential exactly cancels out the system force on the selected atoms. Hence we utilize the negative umbrella force. This can be calculated from the current position along the reaction coordinate and the known harmonic biasing potential. Note that this is not entirely rigorous and can not be used to obtain the true PMF along the reaction coordinate. It is however good enough to obtain A-values for use in cancellation, which is all that the force file is ever used for during the run (these forces are also used to develop a preliminary PMF in analyse_force_database, but that is also adequate since we leave it to

the user to run WHAM and obtain the true PMF – and we provide scripts for that purpose).

The data contained within the .force file is information that the server requires in order to determine cancellation values (a-values) during MonteCarlo or Boltzmann transitions. This data is also stored in the forcedatabase to simplify your later analysis. **In order for this to work properly for a temperature coordinate, the values must be in kcal/mol.**

The .force file should only be created if you have included the NEEDSAMPLEDATA keyword in your .script file.

The .force file must contain a predetermined number of real numbers exactly equal to the number of integration steps specified in the 'JOB' section at the end of the script file. If you don't know the force value at every integration step, then you can trick the program by specifying a certain number of steps in this script file but a different (larger) number of steps directly in your Molecular Dynamics run file.

Section 4.5: The .add# File

The .force file (actually the .add#.nni# file) contains information that you want to be stored in the forcedatabase to simplify your later analysis.

The .add# file(s) should only be created if you have included the ADDITIONALDATA keyword in your .script file and set it to a value greater than zero. If you have set ADDITIONALDATA to 3, then you will require .add1, .add2, and .add3 files (all with .nni# extensions).

The format of the .add# files is identical to that of the .force file. It is not possible to utilize .add# files without also utilizing the .force file.

Section 4.6: The setup File

The setup file is created by the server and sent to the client at every call to DR_client_comm. The purpose of the setup file is to transmit information on what run should be done next. The format of the setup file is such that it can be directly sourced by a CHARMM input file. For GROMACS and NAMD users, the DR_client_body script must parse this file to obtain the values. An example setup file from the middle of a VRE Umbrella run is provided below.

```
set force 119.5
set wref 2.200000
set sampNsteps 101
set rnd 1241892664
set iob "50w5.985"
set job "50w5.986"
```

The file above indicates:

1. The force constant is 119.5 (in kcal/mol) that you can parse from this script. It is possible to hard code the force constant into the DR_client_body script, but that becomes more difficult if you utilize different force constants at different nominal positions. Note that GROMACS users would need to convert this force constant into kJ/mol (500.0) before applying it in the .mdp file.
2. Sampling should be harmonically restrained to 2.2 (in whatever units you use for your reaction coordinate, here it is nm).
3. The run should produce 101 values in .force and .add# files. In this case, the run actually samples for 10 ps and only saves one value every 0.1 ps. It was thus necessary to hard-code the

10 ps runtime per sequence number into the DR_client_body script and the sampNsteps 101 is only used to let the server know how many value to expect in data files.

4. A random seed is available for use if you desire it.
5. The last job completed was 50w5.985. This is replica 5, sequence number 985 for a DRSSS run based on 50.script.
6. The next job to work on is 50w5.986

Section 4.7: The .forcedatabase File

The .forcedatabase file contains all of the data that the server has compiled during the DRSSS run. This is a binary file, but you can access it via the analyse_force_database program (note that this program not only generates a bunch of nice plots based on this data, you can also use it to extract a text version -d option to analyse_force_database

Section 4.8: The .snapshot File

The .snapshot file contains all of the data that the server needs to restart an old simulation. When you restart a simulation, you still require the old .forcedatabase file (contains your data) in addition to the .snapshot file (contains restart files and information about the current nominal positions at which each replica is sampling).

Section 5: Description of Programs in the DRSSS

Core programs:

DR_server	The server that dispatches and directs clients.
DR_client_comm	Client side application that handles communication with the server.
DR_commander	Program that can be used to send signals to the server.
get_simulation_package	Used by the client to obtain the main package of files.
DR_tester	Can be used in place of clients in order to test the server program.

Analysis programs:

analyse_force_database	Uses data in .forcedatabase file to make graphs. Can also output the .forcedatabase file in text format.
extractDatabase	Can act on a text version of the .forcedatabase file to store information in a format that is useful for WHAM.
extractDatabase_replica	similar to extractDatabase.
calcMSD	a work in progress; intended to extract information on replica mobility along the reaction coordinate.

Usage information for these programs is given below. This usage information is also available simply by executing the program with no command line arguments.

$ DR_server

```
Error: the script filename was not provided
Error: parseCommandLine() returned non-zero
Usage: DR_server tt.script [-s]
    -s [string] to restart from a snapshot (e.g. tt.283429.snapshot)
```

$ DR_client_comm

```
Usage: DR_client_comm IP-address port replicaIDcode [JID]
OR:    DR_client_comm IP-address port '**'         [JID]
         [JID] is optional and only used for tracking
```

$ DR_commander

```
Usage: DR_commander IP-address port EXIT | SNAPSHOT
```

$ get_simulation_package

```
Usage: get_simulation_package package-location
```

$ DR_tester

```
Usage: DR_tester IP-address script-file [-nsv]
OR:    DR_tester localhost  script-file [-nsv]
     -n [int] include noise (default = 1)
       ( =0) no noise
       (!=0) with noise
     -s [int] microseconds to sleep (default = 100000)
     -v [int] verbose (default = 0)
       ( =0) not verbose
       (!=0) verbose (a debugging feature)
     -r [int] number of replicas to submit (default = -1073756848)
           Negative is a flag for Nreplicas/Nsamesystem_uncoupled
     -e [string] SPECIAL USAGE (no actual test) specify the filename containing
              positions for which the exact solution is desired. A file te.exact
              will be written containing these values.
```

$ analyse_force_database

```
Error: the script filename was not provided
This program creates .ps graphs based on forcedatabase.
Usage: analyse_force_database tt.script [-lcdtfahme] > analysis.ps
     -l [int] sequence-number-limit; negative indicates no limit (default = -1)
     -c [int] plot cancellation data from tt.log file (default = 1)
       ( =0) do not attempt
       (!=0) plot cancellation (values must exist in tt.log file)
     -d [string] text-database-name (default = not created)
     -t [int] text-database-type (default = 1)
       ( =0) only output replica#, sequence#, w
       (!=0) also output all forces and additional_data
     -f [int] do-fitting-convergence-analysis (default = 0)
       ( =0) do not attempt
       (!=0) do fitting (user beware)
     -a [int0 or >=1] which additional_data holds your sampling value; zero indicates not to use (default = 1)
     -h [int 0 or >=1] number of histogram bins for customizable plots; use zero for = Nreplicas (default = 0)
     -m [int] discard data outside JOB range for for customizable plots (default = 1)
       ( =0) do not discard
       (!=0) discard (useful for comparisons using DR_tester)
     -e [real] initial fraction of data to discard (default = 0.000000)
```

$ extractDatabase

```
Usage: extractDatabase <script file> <database text file> <sequence numbers to skip> <max sequence number to use> > wham.input
     - The database text file can be obtained from analyze_force_database -d database.txt -t 1
     - the redirected output (wham.input above) can be used for Alan Grossfield's WHAM
     - to use the full data, set skip=0 and max<0
```

Section 6: Examples

There is an example directory provided here to calculate the free energy profile for rotation of a backbone dihedral of the alanine dipeptide. The authors make no comment on the correctness of this example directory as it was pieced together from one published DRUS application to work with the most recent version of the DRSSS and may now be incorrect. Still, we hope that this gives you an idea about how your scripts should be written. We also have example scripts for CHARMM and an number of different GROMACS applications that may be available on request depending on the publication state of each of these projects.

Section 7: Support

We provide relatively little support via email or phone as the program usage is complex and we do not feel that this is an optimal method for information transfer. If you are interested in assistance learning this program, we suggest that you arrange to visit our lab with R. Pomès. We estimate that such a tutorial will require two to five days, depending on the skill level of the parties involved.

Thank you for your interest in the Distributed Replica sampling software suite
Sincerely, Chris Neale, Tomas Rodinger, and Régis Pomès
May 20 2009.