

ISD – Inferential Structure Determination

A Bayesian Software for NMR Structure calculation, Version 1.1

Wolfgang Rieping^{1,*}, Michael Habeck^{2,*}, Darima Lamazhapova¹

¹Department of Biochemistry, University of Cambridge, 80 Tennis Court Road, Cambridge CB2 1GA, UK

²Department of Protein Evolution, Max-Planck-Institute for Developmental Biology, Spemannstr. 35, 72076 Tübingen, Germany, and Department of Empirical Inference, Max-Planck-Institute for Biological Cybernetics, Spemannstr. 38, 72076 Tübingen, Germany

*Email: wolfgang.riepping@bioc.cam.ac.uk; michael.habeck@tuebingen.mpg.de;

Abstract

Structure determination by NMR is often viewed as less objective than x-ray crystallography. The major reason for this is the lack of an accepted measure of the quality of an NMR structure, and the use of empirical rules for deriving geometrical constraints from the experimental data. The Inferential Structure Determination (ISD) framework can help to alleviate this problem: ISD is an ab initio method and uses Bayesian inference to process the available experimental data, such as assigned NOEs or RDCs, in an optimal way. The result is a probability distribution that represents the unknown structure and its uncertainty. It also determines additional unknowns, such as theory parameters, that previously had to be chosen empirically. The program uses parallel Markov chain Monte Carlo sampling techniques to explore the conformational distribution of a target molecule and to search for probable parameter settings. This manual gives an introduction to the methodology and shows how to setup and analyse a calculation.

Web: <http://www.bioc.cam.ac.uk/isd>

Discussion: <http://groups.google.com/group/isd-discuss>

Contents

1	Introduction	4
2	Inferential structure determination	4
3	Algorithm	7
4	The ISD software package	9
4.1	License	9
4.2	References	10
4.3	Online resources	10
4.4	Installation	10
4.4.1	System requirements	10
4.4.2	Unpacking and testing	12
4.4.3	Password-free SSH access	12
4.4.4	Python installations on master and slave nodes	13
4.5	What's new in version 1.1?	13
5	Program and software library	15
5.1	Command line options	15
5.2	Supported data formats	16
5.3	CCPN data model	16
5.4	Migrating projects from older versions	17
5.5	Replica-exchange calculation	17
5.5.1	Mode of operation	18
5.5.2	Communication protocols	18
6	Supported NMR parameters	19
6.1	NOE intensities	19
6.2	Scalar couplings	20
6.3	Residual dipolar couplings	20
7	Error models	21
7.1	Log-normal	21
7.2	Gaussian	21
7.3	Von Mises	22
8	Setup of a calculation	22
8.1	Creating a new project	23
8.2	General settings	23
8.3	Replica-exchange algorithm	24
8.4	Molecular system	26
8.5	Adding data	27
8.6	Data specific settings	28
8.6.1	NOE intensities	28

8.6.2	Distances	28
8.6.3	Scalar couplings	29
8.6.4	Residual dipolar couplings	29
8.6.5	Dihedral angles	30
8.6.6	Hydrogen bonds	30
8.6.7	Disulfide bridges	30
8.7	Data import/export from a CCPN project	31
8.8	Starting a calculation	32
8.8.1	Restarting a calculation	32
8.8.2	Running a calculation in the background	33
8.9	Runtime commands	33
8.10	Example projects	33
8.10.1	Tudor domain	33
8.10.2	Ubiquitin	34
9	Creating PDB files	34
10	The ISD report	35
10.1	General settings	36
10.2	Quality assessment and structural analyses	36
10.2.1	Quality scores	36
10.2.2	Ramachandran statistics	37
10.2.3	Secondary structure	37
11	Analysing the results	37
11.1	Convergence of a calculation	37
11.1.1	Total energy	38
11.1.2	Rates of exchange	38
11.1.3	High temperature distribution	39
11.1.4	Improving convergence	39
11.2	Data quality	40
11.2.1	Reliability of individual measurements	41
11.3	Structure validation	42
11.3.1	Structural uncertainty	42
11.3.2	Quality scores	43
12	Accessing simulation results using Python	43
12.1	Creating a posterior object	43
12.2	Loading and accessing “ensembles”	44
12.3	Saving conformational samples as PDB files	45

1 Introduction

In an aqueous environment, most proteins fold into thermodynamically stable three-dimensional structures. A detailed understanding of the biological function of proteins or DNA requires knowledge of its molecular structure. Structural knowledge is also crucial in applications such as drug design. High-resolution nuclear magnetic resonance (NMR) spectroscopy has become, along with X-ray crystallography, a routine method for determining biomolecular structures with atomic resolution [1]. In comparison to X-ray crystallography, NMR allows the study of proteins in solution and does not require ordered crystals. It can also provide a dynamical picture of a molecule. But NMR structure determination is far from being straightforward: It requires several manual or semi-automatic preprocessing steps such as spectral analysis, peak picking, and resonance assignment. In the final step, geometrical constraints are derived which are then used to calculate the molecular structure.

Each of these stages requires human intervention, which is why structure determination by NMR is often perceived as being less objective than X-ray crystallography. The major reasons for the “subjectiveness” of NMR structures are: (1) the lack of a generally accepted measure for assessing the quality of an NMR structure, (2) the use of heuristics and rules of thumb in the derivation of geometrical constraints. Strictly speaking, a protein structure determined from experimental data is only useful if it is accompanied by some measure of reliability. Recent works discussing errors in published NMR structures [2] highlight the danger of subjective elements in structure determination procedures.

The aforementioned problems have a common source: Structure determination requires reasoning from incomplete information which is why protein structures necessarily remain uncertain to some degree. Existing methods, however, are based on the concept of structural constraints, and are therefore incapable of taking this uncertainty into account. In essence, ISD relies on Bayesian probabilistic inference that represents any uncertainty through probabilities which are then combined according to the rules of probability calculus. The application of this approach is computationally demanding, and has become feasible only recently due to the development of efficient stochastic sampling algorithms (Markov chain Monte Carlo methods) and increased computational power provided by computer clusters.

2 Inferential structure determination

The principal difficulty in structure determination by NMR is the lack of information required to unambiguously reconstruct a protein structure. Conventional methods view structure determination as a minimisation problem: A so-called “hybrid energy” function combines a pseudo energy term that incorporates the experimental constraints with a force field describing the physical interactions between the atoms. Minimising the hybrid energy is then assumed to answer what the “true” structure of a molecule is. This rule, however, implicitly assumes that there is a unique answer. Repeating the minimisation procedure multiple times, as is standard practice in conventional approaches, does not adequately represent the ambiguity and makes it difficult to judge the validity and precision of NMR structures in an objective way.

We have argued that it is a misconception to use structure calculation methods that are only

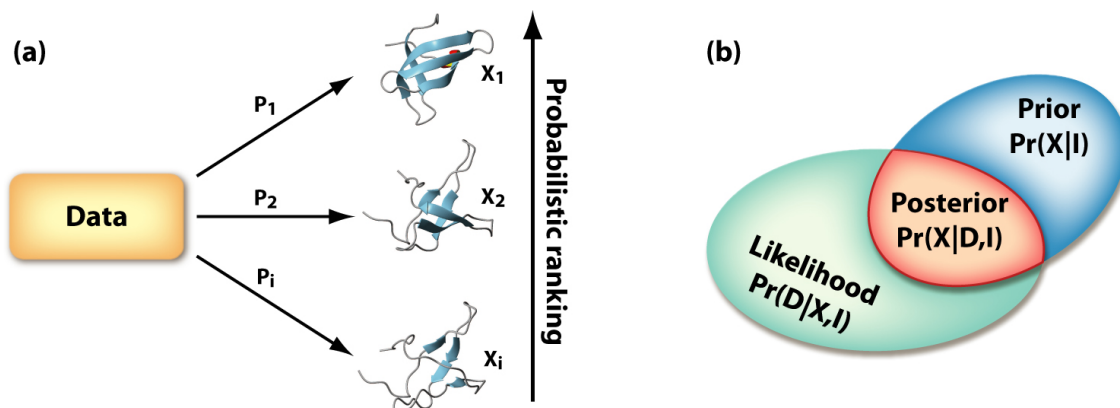


Figure 1: Probabilistic ranking and Bayes' theorem. The experimental data are used to rank every conformation of a protein in terms of a probability (a), i.e. we do not derive geometrical constraints that would completely rule out structures. If of two conformations one has higher probability, then it is more supported by the data. The spread of the probability distribution reflects how well we can determine a structure from the available information. If only a single conformation has non-zero probability, the data uniquely determine the structure. If the probabilities are constant, the available data is uninformative with respect to the structure. Realistic cases lie somewhere in between. Bayes' theorem (b) combines prior information with experimental evidence, represented in terms of a likelihood function, in a consistent way. The posterior distribution represents everything that can be said about the molecular structure given the data and our prior knowledge.

appropriate if the objective is to obtain a unique structure. Instead, we view structure determination as an *inference problem* [3,4], requiring reasoning from incomplete and uncertain information. In contrast to conventional methods, we do not convert the data into geometrical constraints, but use them directly to rank all possible conformations of the molecule. Quantitatively, such a ranking requires us to assign a probability P_i to every protein conformation X_i [5] (figure 1a). We demand the probabilities to be objective in the sense that they should depend only on the data and on relevant prior information (such as the theoretical models to describe the data or knowledge about physical interactions). Thus we are dealing with a conditional probability, $\Pr(X|D, I)$, quantifying how likely a certain conformation X is the correct structure given the data “ D ” and background information “ I ”. Any inferential structure determination is solved by exploring this probability distribution.

But how can we set up $\Pr(X|D, I)$ for a concrete structure determination problem? The answer comes from Bayes' theorem [6] which states that the solution to any structure determination problem is proportional to the product of the likelihood of the data given a structure, $\Pr(D|X, I)$, and the prior probability $\Pr(X|I)$ (figure 1b). That is, once we are able to write down the likelihood and the prior distribution for a particular structure determination problem, we simply use Bayes' theorem to obtain a relative probability for every conformation of the protein, and thus solve the ranking problem. At first glance this seems to complicate things even further, but it turns out that $\Pr(D|X, I)$ and $\Pr(X|I)$ are relatively easy to set up.

An example Let us consider a concrete example. The most informative class of NMR observations are based on the Nuclear Overhauser Effect (NOE) [7]. The NOE is a relaxation effect that leads to an NMR signal with an intensity I roughly proportional to the inverse sixth power of

the distance d between two nuclear spins. However, the model $I \propto d^{-6}$ neglects dynamics [8], indirect magnetisation transfer via spin diffusion [9], and other effects. Due to these theoretical limitations and experimental noise, observed NOE intensities can never be predicted with certainty from a protein structure. In order to deal with deviations between observations and predictions, we introduce an error parameter σ that quantifies how closely our predictions match the observations. Since intensities and distances are positive quantities, we model their deviations with a log-normal distribution [10]. If we have a whole set of intensities I_i with corresponding distances d_i the likelihood of the data is a product of log-normal distributions:

$$\Pr(D|X, \alpha, \sigma, I) = \prod_i \frac{1}{\sqrt{2\pi\sigma I_i}} \exp \left\{ -\frac{1}{2\sigma^2} [\log I_i - \log(\alpha d_i^{-6})]^2 \right\} \quad (1)$$

where α is the unknown proportionality factor, and the distances depend on the protein conformation, i.e. $d_i = d_i(X)$.

This example illustrates two points: First, it is straightforward to write down the likelihood function. Second, one typically needs to introduce auxiliary parameters, such as σ and α . Both are required to describe the measurements, but cannot be determined experimentally. In standard methods, such parameters need to be set empirically, which can bias the results, and adds to the problem of structure validation [11, 12]. In Bayesian theory, such *nuisance parameters* are treated in the same way as the coordinates: They are estimated from the data by applying Bayes' theorem on the joint parameter space.

Bayes' theorem requires that we assign prior probabilities for the conformational degrees of freedom and the nuisance parameters. Using arguments from statistical physics, it turns out that $\Pr(X|I)$ is the canonical ensemble:

$$\Pr(X|I) \propto \exp \{-\beta E(X)\} \quad (2)$$

where $E(X)$ is a molecular force field encoding chemical information on bond lengths, bond angles, etc.; β is the inverse temperature. In the simplest case, the prior probabilities for the nuisance parameters are uniform distributions, and the posterior distribution for *all* unknowns is:

$$\Pr(X, \alpha, \sigma|D, I) \propto \alpha^{-1} \sigma^{-(n+1)} \exp \left\{ -\beta E(X) - \frac{1}{2\sigma^2} \sum_i [\log I_i - \log(\alpha d_i^{-6})]^2 \right\}. \quad (3)$$

Hence, probability calculus formally solves our structure determination problem from NOE data given in the example: The posterior probability distribution represents the complete information on the possible conformations of the molecule, as well as on the values of our nuisance parameters, α and σ in this case.

ISD enables users to infer the 3D coordinates of proteins along with its structural uncertainty using various types of experimental NMR data, such as assigned NOEs or RDCs. To this end, the program sets up the posterior distribution for all unknowns in a similar way as shown above and uses Markov chain Monte Carlo techniques to explore that distribution starting from an extended conformation.

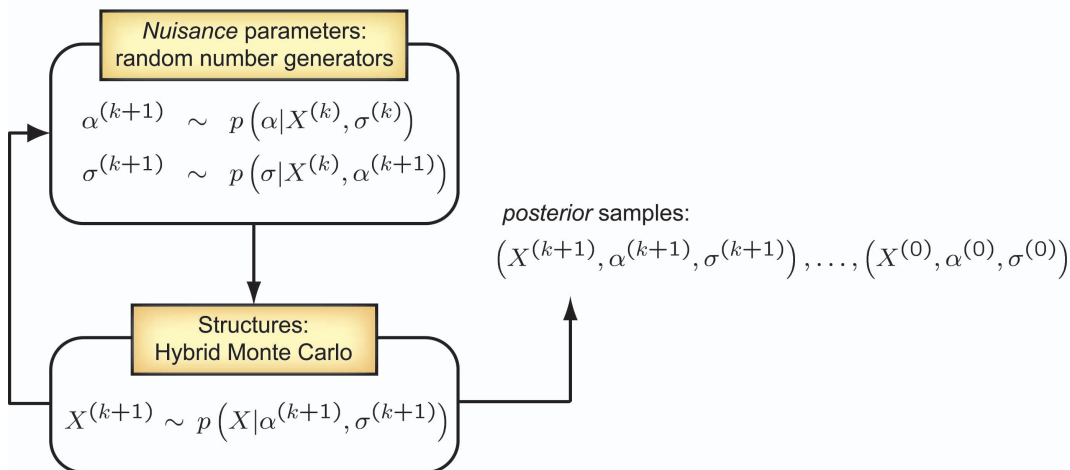


Figure 2: Gibbs sampling scheme used to generate samples from the posterior probability for protein conformation X and nuisance parameters α and σ . Gibbs sampling is an iterative scheme that, upon convergence, produces samples from the full posterior distribution. The nuisance parameters can directly be drawn from their posterior probabilities. To update the conformational degrees of freedom we employ the HMC algorithm. This algorithm uses molecular dynamics [14] to generate a candidate conformation which is accepted according to the Metropolis criterion [15]. The molecular dynamics is defined by the negative log-posterior probability with fixed nuisance parameters.

3 Algorithm

For realistic problems, the posterior probability is a very complex mathematical function. It is defined over a space of typically several hundred dimensions, which makes it impossible to either visualise the probability directly or to analyse it analytically. Therefore, we need to employ numerical methods to analyse the posterior probability. A conceptually simple way to investigate a high dimensional probability is to draw samples from it in such a way that the distribution of samples follows this probability. The samples can then be used to calculate most likely values of parameters, averages, variances, etc. [13]. Therefore in the ISD framework, structure calculation amounts to the generation of random samples of the 3D coordinates and auxiliary parameters from the joint posterior probability, $\Pr(X, \alpha, \sigma|D, I)$ in the example above. This differs fundamentally from conventional structure calculation algorithms because the uncertainty of the structure is explicitly taken into account, and nuisance parameters are not kept fixed.

The stochastic sampling scheme

We generate posterior samples $(X^{(k)}, \alpha^{(k)}, \sigma^{(k)})$ by using a hierarchical ‘‘Gibbs sampling’’ scheme that combines several Markov chain Monte Carlo strategies. The Gibbs sampling procedure [16] facilitates a split-up of the sampling problem into several steps. Each parameter class, X , α , and σ , is sampled sequentially conditioned on the current values of the other parameters (figure 2). In order to apply a Gibbs sampling scheme, we need to be able to simulate the conditional posterior densities for the nuisance parameters and the coordinates. For some distributions, e.g. a normal

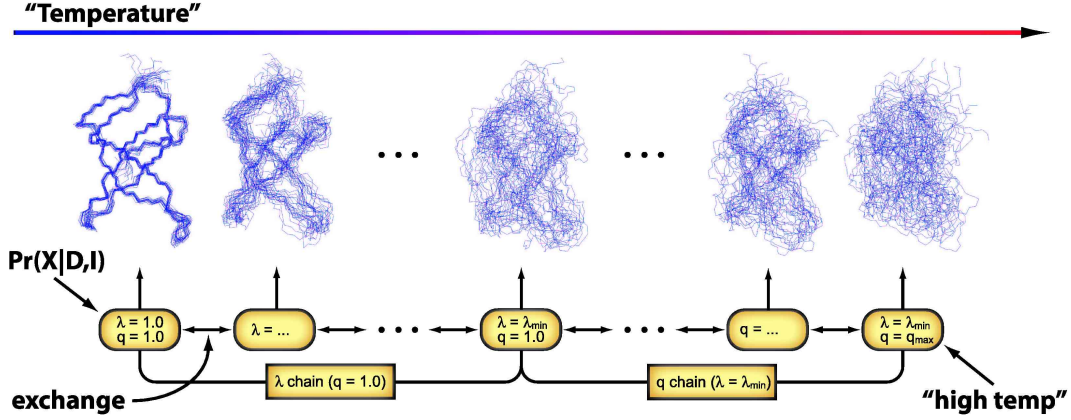


Figure 3: Replica-exchange Monte Carlo algorithm. We embed the Gibbs sampler (figure 2) in a Replica-exchange Monte Carlo scheme which simulates a sequence of “heated” replicas of the system. Two generalized temperatures, λ and q , control the shape of the likelihood function and of the prior distribution, respectively. For $\lambda = 1$ the data are switched on, for $\lambda = 0$ they are switched off. For $q = 1$, the canonical ensemble is restored as prior probability [cf. Eq. (2)]. For $q > 1$ physical interactions are gradually switched off and the prior probability approaches a flat distribution over conformation space. We arrange the replicas in such a way that first the data are switched off (by gradually decreasing λ). In the other half of the arrangement, we additionally switch off the physical interactions by increasing q .

or log-normal distribution, this can be done by using random number generators. To sample the highly correlated 3D coordinates, however, we need to employ more elaborate techniques such as the Hybrid Monte Carlo (HMC) method [17].

Replica-exchange Monte Carlo

For complex systems such as proteins, the sampling scheme described so far is likely to get trapped in one of the modes of the posterior distribution and thus fails to explore the entire parameter space. These modes correspond to protein conformations that all fulfill the data well. Missing a high-probability fold would bias our analysis.

A physical system trapped in a metastable state can be melted by increasing its temperature. For sufficiently high temperatures the system easily explores all regions of the configuration space. The Replica-exchange Monte Carlo method [18] exploits this observation: It considers a composite Markov chain comprising several non-interacting copies of the system, so-called “replicas”, each of which is simulated at a different temperature. Neighbouring replicas are coupled by exchanging configurations after a number of Gibbs sampling steps (“super-transition”), which significantly enhances the mobility of the individual Markov chains.

We have extended this scheme by introducing two generalized “temperatures”, λ and q , to independently control the likelihood function and the prior probability [19] (figure 3). The parameter λ weighs the likelihood function, and thus controls the influence of the data. We further improve the sampling by replacing the canonical ensemble with Tsallis generalized ensemble [20–23]. Tsallis ensemble is based on a non-linear transformation of the force field $E(X)$ and involves a parameter q which controls the strength of the non-linearity. The transformation is chosen such that high energy configurations are no longer exponentially suppressed, but follow a power law. This has

the effect that atoms can exceedingly pass through each other, thus facilitating large conformational changes. During a simulation, states diffuse up and down in the replica arrangement, which guarantees ergodic sampling of the posterior distribution.

Computational considerations

As argued above, solving an inferential structure determination problem boils down to sampling conformations from the conformational posterior distribution. That is, the aim is to explore entire distributions of structures, instead of locating a number of low-energy conformers as done in conventional approaches. Therefore, the ISD approach is computationally more challenging than standard techniques.

However, thanks to the availability of computer clusters, the time required to infer the structure of systems that are accessible to NMR techniques can be kept within reasonable limits. In practise, the calculation time depends on the system size as well as the amount and quality of the data. For example, few data and/or data of poor quality typically result in uncertain structures – as common sense would suggest. That is, the corresponding conformational distributions are severely spread out and thus require more samples (i.e. more computer time) to be adequately represented. In contrast, exploring the distribution of well defined structures can be fast as few samples may be sufficient to describe the structural uncertainty. Though the program works on desktop PCs, we strongly recommend running ISD on a computer cluster. To our experience, the calculation time typically ranges from 1 day to 1 week, when using a Linux cluster with 50 nodes.

4 The ISD software package

The program ISD [24] implements the methodology outlined in the preceding sections. ISD is based on an object-oriented software library written in the programming languages Python [25] and C. Python is an object-oriented language and features an advanced language design. It is open source, offers strong support for integration with other languages, and is easy to learn. C is well known for its performance.

Time critical routines, such as the computation of the energy of physical interactions within a molecule, are written in C for optimal performance. So-called “wrappers” glue the C to the Python world thus enabling the use of C functions seamlessly from within Python. Using this hybrid approach, we benefit from both Python’s advanced language design, and the performance of C.

4.1 License

The ISD distribution comes with a free academic license. If you use ISD you must register at <http://www.bioc.cam.ac.uk/isd> and accept the following license agreement:

The Inferential Structure Determination (ISD) software library. Authors: Wolfgang Rieping and Michael Habeck. Copyright (C) Michael Habeck and Wolfgang Rieping. All rights reserved.

ISD is provided “as is” without warranty of any kind, expressed or implied, including, but not limited to the implied warranties of merchantability and fitness for a particular purpose or a warranty of non-infringement. Distribution of modified versions of any of the modules of ISD is prohibited without the explicit permission of the copyright holders.

The information in this software is subject to change without notice. We do hope, however, to get responses from users, especially when errors have been found.

4.2 References

If you use the program, please quote the following reference(s):

1. Wolfgang Rieping, Michael Habeck, Michael Nilges (2005): Inferential Structure Determination. *Science* 309:303–306

4.3 Online resources

Web site: Please visit our web site at <http://www.bioc.cam.ac.uk/isd>, which contains more information on ISD and related topics.

Discussion group: For those of you who are interested in ISD or who would like to share their experience with other users, please join our ISD user group at <http://groups.google.com/group/isd-discuss>.

4.4 Installation

4.4.1 System requirements

As already mentioned, ISD is computationally more challenging than conventional structure calculation techniques. Though the program runs on desktop PCs, we therefore strongly recommend using a computer cluster in order to keep the calculation time within reasonable limits.

ISD has been tested on different Linux environments (RedHat, Centos, SuSE, Ubuntu, Gentoo and Fedora Core). Memory and disk requirements depend on the model that is used to analyse the data. It also depends on the amount of data and of the size of the system at hand. However, experience suggests that 512 MB of memory per process is sufficient for most applications.

Running ISD requires the following third party software packages:

1. **Python** (version 2.3 or 2.4). Python can be downloaded from the Python homepage¹. The installation should be straightforward for most LINUX systems. Please note that according to some user reports, ISD might hang or crash when used in combination with Python 2.4.x on some SUSE Systems. In this case, please use version 2.3.x instead. ISD has not been tested with Python 2.5 yet.

¹<http://www.python.org>

2. **Numerical Python** (version 21 or higher, except version 23). Numerical Python (not to be confused with the extensions NumPy) is a popular Python module. It can be obtained from SourceForge². Go to **Download Numerical Python** and download the package **Old Numeric**. For installation, please follow the instructions provided with the Python Numeric distribution.

For users who install Numerical Python using the package manager **aptitude**: please make sure you install the module **numeric** as well as **numeric-ext**.

3. **Scientific Python**. The Scientific Python³ (not SciPy!) distribution contains some functionality needed by ISD. For installation, please follow the instruction provided with the distribution.

Additionally, some features of the program require the following optional modules:

1. **Python Remote Objects**. From version 1.1 on, ISD provides an additional communication module based on the *Python Remote Objects* (PyRo) protocol. PyRo is open source and can be downloaded from the PyRo homepage.⁴ Installation of the package does not require special privileges. Please follow the installation instructions provided with the PyRo distribution.
2. **Biggles scientific plotting library**. The *Report* feature of ISD creates a PDF document with summaries and analyses of the calculation results. The feature requires installing the plotting library Biggles⁵ which is freely available.
3. **PDFLatex and PDF/EPS tools**. The report feature also requires the software packages **pdflatex**, **epstopdf**, and **eps2eps**, which are installed on many Linux distributions by default.
4. **CNS**. The program CNS is not required for running ISD. However, we recommend installing the program, as it greatly simplifies the generation of PDB files with the initial coordinates needed to initialise a simulation. Otherwise, users need to provide their own PDB starting structure. Academic users can obtain a free license for CNS from the CNS web site⁶.
5. **CCPN distribution**. Importing and Exporting data from and to a CCPN project, respectively, requires installation of the CCPN distribution⁷. Please download and install the full distribution including the API, the FormatConverter, and Analysis. ISD requires the latest version of some helper modules that are part of Analysis. These modules can be installed after installation of the CCPN distribution by using the CCPN upgrade feature. To do so, simply start Analysis and choose “Upgrade” in menu “File” and follow the instructions.
6. **WhatIf, Procheck, DSSP**. Optionally, ISD calculates a number of quality scores in order to validate the generated structures. The report gives a summary of the validation checks. This requires the programs WhatIf⁸ and Procheck. In addition, ISD calculates probabilities

²<http://sourceforge.net/projects/numeric>

³<http://dirac.cnrs-orleans.fr/ScientificPython>

⁴<http://pyro.sourceforge.net>

⁵<http://biggles.sourceforge.net>

⁶<http://cns.csb.yale.edu>

⁷<http://www.ccpn.ac.uk/downloads/downloads.html>

⁸<http://swift.cmbi.kun.nl/whatif>

for the different secondary structural states using the program DSSP⁹. Please note that some validation/structure scores are available in recent versions of WhatIf only (version 6.0 has been tested).

4.4.2 Unpacking and testing

To install ISD, please unpack the ISD distribution by using the following command:

```
tar xvzf isd-[VERSION_NUMBER].tgz
```

This creates a new directory `isd-[VERSION_NUMBER]`. You find the Python modules in the sub-directory `src/py` of the distribution. XML files that contain force field and topology parameters can be found in the sub-directory `toppar`. Two example projects are located in the sub-directory `examples`. A PDF file with this manual can be found in the sub-directory `manual`.

For convenience we suggest adding the following lines to your `.cshrc` login script:

```
setenv ISD_ROOT <isd-installation-path>
alias isd <Python-executable> -i $ISD_ROOT/isd.py
```

Adding the ISD Python modules to your `PYTHONPATH` is not necessary and might cause problems during runtime. In order to test whether ISD and the Python environment are setup correctly, use the command `isd --check`.

If the installation has been successful, ISD can be started from the the shell with the command `isd`. The program provides a number of options, an overview of which is given by the option `--help`.

4.4.3 Password-free SSH access

Running ISD on a computer cluster requires that you can log into all machines that are used for a calculation via ssh without being prompted for a password. A password-free ssh access can be setup in three steps:

1. Generate a public identity key on the master node by invoking:

```
ssh-keygen -t rsa
```

After invoking this command, you will be asked three questions each of which you answer by pressing the return key. After completion, you should find the key file `id_rsa.pub` in the directory `~/.ssh`.

2. Copy the key file over to the client node (you are still on the master node):

```
scp ~/.ssh/id_rsa.pub <client-node>:/tmp/.
```

3. Now, log into the client node and add the key file to the list of machines that are authorized to do password-free SSH:

⁹<http://swift.cmbi.ru.nl/gv/dssp>

```
cat /tmp/id_rsa.pub >> ~/.ssh/authorized_keys
```

That's it. From the master node, you can now connect to the client node for which you have performed steps 1-3, without being asked for a password.

4.4.4 Python installations on master and slave nodes

ISD supports the use of different Python installations on the master node (from where you run ISD) and the slave nodes used to do the computation. The name of the Python binary that shall be used on the slave nodes can be specified in the project file (s. keyword `python` in section 8.3 for details.)

4.5 What's new in version 1.1?

Infrastructure

Communication ISD 1.1 comes with an additional communication module based on the open source protocol *Python Remote Objects* (PyRo). The module uses PyRo remote services to perform certain tasks in parallel. In the case of ISD, such a service is the implementation of the sampling engine. In the current implementation, all machines that are connected to the same network can be used for a calculation, provided they have the same architecture and operating system.

The new communication module is slightly more demanding in terms of installation and setup. However, compared to the existing method it has the advantage of not requiring a directory that is shared among all machines. It also causes less network traffic and seems to be more reliable.

Project file The layout of a project file has been reorganised slightly. However, project files created with version 1.0 work out of the box and do not need to be updated. Specifically:

- An additional field `data_name` has been added to each sub-section under section "Experimental data". The field specifies the name of a dataset to be used internally. If given, the name will also appear in the PDF report. The new field replaces the variable `data_key` which is now only used as a key to access datasets in ISD XML files or a CCPN project.
- The two variables `molecule.extended` and `molecule.randomize` were collapsed into a new field `molecule.initial_conformation`. The field can be set to `EXTENDED`, `RANDOM` or `AS-IS`. Please see section 8.4 for further information.

Command line options Version 1.1 comes with a number of new command line options. To get a full list of options, use "isd -help". Among other things, it is now possible to save an ensemble of representative or most probable structure as PDB files (option `-write-pdb`). The way in which this is done is controlled by section "pdb_files" in a project file. Please see section 9 for details.

Models

Dihedral angles The existing model to analyse dihedral angle information is based on the assumption, that all dihedral angle measurements have the same (unknown) error. The error is estimated during a calculation. However, the assumption of a common error is not always correct. For example, torsion angle predictions made by the program TALOS assign site-specific error estimates to

each dihedral angle. Therefore, the model has been extended in order to account for this additional information.

Non-bonded interactions To speed-up simulations of larger systems, the calculation of non-bonded interactions involving hydrogen atoms (which contribute significantly to the overall calculation time) can be disabled. To our experience, neglecting these interactions has only very little impact on the quality of a structure but leads to a significant calculation speed-up.

Data handling

CCPN data model ISD now interfaces the CCPN data model (API version 1)¹⁰ to import data from a CCPN project. We support the import of sequences, NOEs, distances, RDCs and dihedral angles. On the export side, we currently support the export of a probabilistic structure ensemble. The project file has been extended and several new command line options were added to the main program `isd` that controls the CCPN import/export feature.

Migrating version 1.0 ensemble and history files A *migration* feature allows to convert version 1.0 ensemble files to the current format. Please see section 5.4 for more details.

Manual and Report

Both manual and report have been extended and revised. The validation module has been extended and now uses the program DSSP to derive the most likely secondary structure assignment of the molecule at hand. The module now also supports the program WhatCheck (the smaller brother of WhatIf) for calculating validation scores. The project file provides a switch to choose between both programs.

Sampling engine

The length of the initial period during which the step size (used by the MD part of the hybrid Monte Carlo engine) is adjusted automatically, can now be set in the project file. After that period, an average step size is used which is kept fixed throughout a calculation.

Miscellaneous

Example Python scripts A couple of example Python scripts illustrate how to use the ISD Python library to manipulate data and analyse results etc. The scripts are accessible on the website.

Minor Extensions and changes In the case of a simulation crash, remote processes terminate automatically, i.e. the system is not clogged up by zombie Python processes. The reader for X-PLOR/CNS `tbl` files now copes with RDC restraints.

Bug fixes Bugs mentioned on the website and in the discussion group (reader for X-PLOR/CNS TBL files, report generation, issues with the "shared" communication module in certain network environments) have been fixed.

¹⁰See <http://www.ccpn.ac.uk>

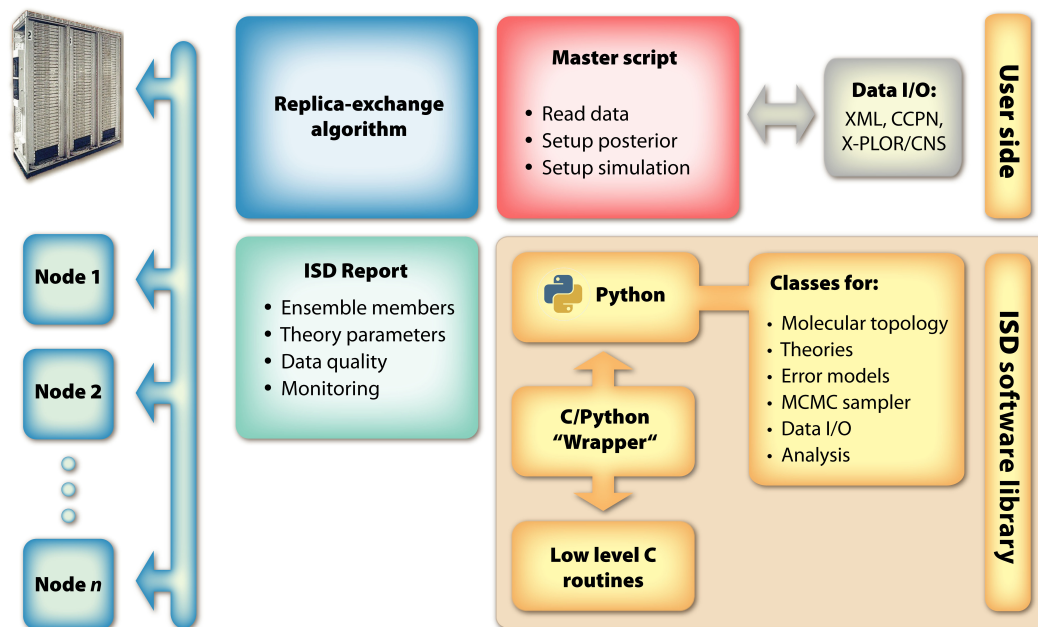


Figure 4: Overview of the architecture of ISD. On the user-side, ISD (red) manages the import of the experimental data (grey), the setup of a replica-exchange calculation (blue), as well as the generation of a report providing analyses of the calculation results (green). The ISD software library (amber) provides the functional basis for performing these steps.

Changes The command line option `--template` to create a new project file has been renamed to `--new-project`. The option `--template` is still available though.

5 Program and software library

Figure 4 shows the principal design of ISD. The object-oriented software library shown in amber forms the heart of the program. It provides the functionality needed for setting up a project, performing a structure calculation, as well as analysing the results. Each of these steps is controlled by a “project file” depicted in red. The project comes with default parameter settings which have proven suitable for inferring biomolecular structures based on various NMR experiments. The experimental data are incorporated into a calculation by setting-up a respective likelihood function, one for every dataset. The program does all these steps fully automatic – the user only needs to specify information on the type and location of the data.

5.1 Command line options

Provided the program has been installed correctly, ISD can be run from the command line by using the command `isd`. The program provides a number of command line options, an overview of which is given by the option `--help`:

- The command line option `--check` verifies that the ISD Python modules can be imported successfully. It also tests your Python environment. If any of these tests fails, please verify

your software setup.

- The option `--new-project` creates a new project file, see chapter 8.1 for details. From version 1.1 on, this option replaces the option `--template` that has been used in version 1.0.
- A PDF report that summarises the simulation results can be generated with the option `--report`. Chapter 10 discusses this feature in detail.
- To generate PDB files with the coordinates of the ensemble members, use the option `--write-pdb` with a project file as argument. Files are stored in the current directory. See section 9 for more details.
- The option `--quit` with a project file as argument can be used to quit a calculation that is running in the background (cf. section 8.8.2).
- Features introduced with a newer version of ISD may require an extended data format for representing the simulation results. The option `--migrate` modifies ensemble and history files that were generated with an older version of ISD in such a way that they are compatible with the current version. See section 5.4 for more details.
- The option `--ccpn-info`, called with a CCPN project as argument, lists the keys of all objects that can be read from the CCPN project. The keys are required by ISD in order to locate a particular object (e.g. a molecular system or restraint list) within a CCPN project.
- The option `--ccpn-export` with a project file as argument exports entities such as structures to the CCPN project specified in the project file. See section 8.7 for details.

5.2 Supported data formats

ISD represents experimental data with a format based on the eXtensible Markup Language (XML) [26]. The XML standard allows the definition of portable and human readable formats for information exchange and facilitates the validation of documents. The program provides XML data formats for describing the molecular topology (following the IUPAC naming system [27]) and the experimental NMR parameters.

In addition to XML, the data can also be specified in X-PLOR/CNS `tbl` format (NOEs/distances, dihedral angles, and RDCs), TALOS format (dihedral angles) and via a CCPN project (s. below). During the initialisation of a calculation, proprietary data formats are converted into ISD XML format and stored in the directory `[WORKING_PATH]/data`.

5.3 CCPN data model

Most NMR software packages use proprietary formats for data storage which need to be inter-converted for transferring data between different applications. This usually requires manual intervention and can lead to a loss of information since the data conversion is often incomplete. The CCPN data model [28] alleviates these problems by defining a storage model that integrates all information emerging in a structure determination project in a common framework. This includes

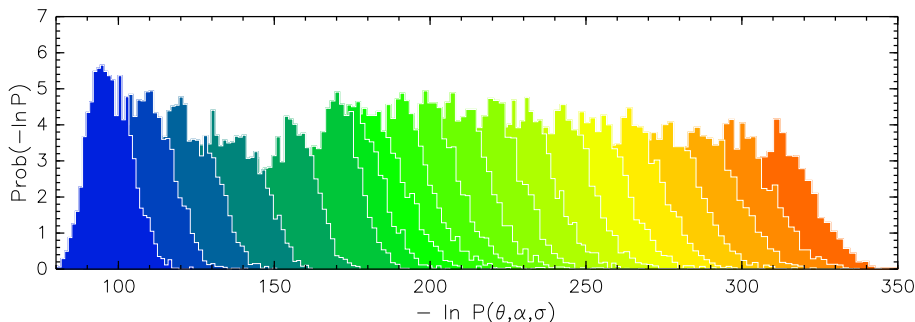


Figure 5: Distributions of the negative log-posterior probabilities of the replicas. The color encodes the position in the replica chain (i.e. “temperature”). Blue: true posterior distribution, red: “high-temperature” posterior distribution.

details on the molecular system and experimental data such as NOEs or residual dipolar couplings. The model can also handle the calculation results, most importantly the three-dimensional coordinates of the structures.

ISD supports the import of data from and the export of the calculation results, most importantly the conformational ensemble, to a CCPN project. This enables users, for example, to run a calculation using restraint lists that were created with other programs that support CCPN, such as the program ARIA [29, 30] for automated NOE assignment, without the need to first convert the data to ISD XML. During the initialisation of a calculation, data read from a CCPN project are converted into ISD XML format and stored in the directory `[WORKING_PATH]/data`.

5.4 Migrating projects from older versions

Features introduced with newer versions of ISD may require an extended data format for adequately representing the simulation results. A project can be migrated from an older version to the current version of the program, by using the command line option `--migrate`. The migration feature modifies the following files:

1. Ensemble files `[FILE_ROOT]_n`, where n depends on the number of replicas;
2. The project history file `[FILE_ROOT]_history`.

Please note: The files above are modified in-place, i.e. no backup files are created. Therefore, please make sure you backup your files before using migrating.

5.5 Replica-exchange calculation

Setting up an ISD replica-exchange simulation only requires specifying the generalised temperature values (λ, q) , as well as the number of Gibbs sampling steps per super-transition. The default settings are suited for standard systems, however, depending on the system size and the amount and quality of the data, “fine-tuning” of the temperatures might be necessary in order to get a reasonable rate of convergence.

```

*** Pyro Name Server ***
Pyro Server Initialized. Using Pyro V3.7
Name server listening on: ('0.0.0.0', 9090)
Broadcast server listening on: ('255.255.255.255', 9090)
URI is: PYRO://192.168.5.123:9090/c0a805bd77d61bc52485597506aace5ae2
URI written to: /tmp/Pyro_NS_URI
Name Server started.

```

Figure 6: Startup of a Python Remote Object name server. If multiple simulations are to be run on the same network, it is sufficient to start the name server only once. The name server listens on port 9090, which needs to be open in case you run a firewall.

5.5.1 Mode of operation

As described in section 3, each super-transition requires a number of Gibbs sampling updates for every replica. Because communication between the replicas is rare (it is only required upon the exchange of states), a replica-exchange scheme is particularly suited to be run on multiple machines in parallel. The setup of a parallel simulation is straightforward: All one needs to do is to specify a list of available machines. ISD then launches the required services on each of these machines using the communication method chosen. During a calculation every remote process receives requests from the master to perform a number of Gibbs sampling steps at a given temperature. ISD then collects the results, attempts an exchange of states, and starts a new super-transition. The parallel setup is completely transparent to the user as the main program runs locally on his desktop machine. The mixing efficiency of the replica algorithm depends on the number of, and the rate of exchange between the replicas. Therefore, the temperatures λ and q need to be chosen carefully as they control the overlap between the posterior distributions and hence the rate of exchange. The default settings typically result in an rate of exchange between 20 and 70 percent (figure 5). For optimal performance, the number of available machines should be equal to the number of replicas (default: 50).

5.5.2 Communication protocols

ISD supports two communication mechanisms for distributing a calculation on a computer cluster. The communication modules provide the means to exchange data between the host on which you run ISD and the remote machines that are used to perform the calculation.

File-based method “shared” The communication method “‘shared’” uses files and a shared directory that is accessible from all machines to pass data between the nodes. The method does not require additional software modules to be installed, and is therefore easy to use, but can become inefficient if the machine that hosts the shared directory is loaded. In some cases, if NFS (used to share the directory) is not setup properly, NFS-related timing issues can cause simulation hang-ups.

Python Remote Objects “pyro” From version 1.1 on, ISD supports a new communication method “‘pyro’” based on the *Python Remote Objects* (“PyRo”) protocol. PyRo uses a *name server* to register remote services (provided by “servers”) that can be used by a “client” in order to perform a certain task remotely. In case of ISD such a service is the implementation of the sampling engine, and the main program “isd” is the client. With the current implementation, all machines that are connected to the same network can be used for a calculation, provided they have the same architecture (e.g. 64-bit PC Linux). Data exchange is done via the TCP/IP protocol and requires several ports to be open. In a nutshell, the name server uses port 9090, the servers (i.e. the

machines which you use for a calculation) listen on ports 7766 to 7865. Therefore, in case your system operates a firewall, please make sure the following ports are open (TCP and UDP):

1. Port 9090 on the machine that runs the name server;
2. Ports 7766-7865 on the machines that are used for a calculation.

For more detailed information, please see the PyRo manual.

PyRo name server Prior to a calculation, the name server needs to be started. This can be done by running the script `Pyro-[version_number]/bin/pyro-ns -k &`. The name or IP address of the machine running the name server needs to be specified in the project file (see section 8.3). Though the command line option `-k` is not required for starting the name server, we strongly recommend using this option as otherwise your name server could accidentally get shut down by other users.¹¹

As mentioned above, the name server is used to register services provided by the servers. To ensure that multiple servers can access the name server simultaneously, please set the environment variable `PYRO_MAXCONNECTIONS` in your `.cshrc` (or similar) to 1000.

Because independent calculations (running on the same network) can share the same name server, it is sufficient to start the name server only once. For example, the name server could be started automatically during boot time on some machine with IP address, say, 192.168.5.1. All users could then use the same name server simply by specifying the IP address 192.168.5.1 in their project files. The output of a successful startup of a PyRo name server is shown in figure 6. In the case a name server is already running on some machine, use the script `Pyro-[version_number]/bin/pyro-nsc listall` to obtain the IP address of that machine.

6 Supported NMR parameters

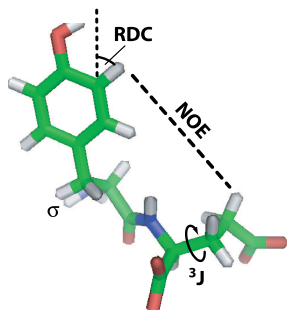
To incorporate experimental data into a calculation, a “theory” is used to calculate the ideal value of a measurand. The ideal value depends on the three-dimensional coordinates of a structure and, depending on the theory, a set of theory parameters. The program also supports geometric parameters directly, such as distances or dihedral angles.

The current version of the program supports most of the commonly used experimental NMR parameters (cf. table 1). The NOE contains information about the spatial vicinity of protons and is the most important source of structural information. It has been utilised to determine virtually all of the NMR structures currently stored in the Protein Data Bank (PDB) [32]. Residual dipolar couplings (RDC) are less informative than the NOE but can be very useful for obtaining information on the global fold of the protein [33] and for validation a structure. Three-bond scalar couplings are routinely measured to obtain precise information on the local conformation of a biomolecule [34]. In addition to experimental parameters, the user can specify structural information directly in the form of distances and torsion angles.

6.1 NOE intensities

ISD uses the isolated spin pair approximation to calculate experimental NOE intensities from the three-dimensional coordinates of a structure. In case the NOE involves atoms that are part of

¹¹This issue is discussed in more detail on the PyRo homepage.



Parameter	Structure	Theory	Error	n [†]
NOE intensity	distance	ISPA	Log-normal	2
Scalar coupling	torsion angle	Karplus curve	Normal	4
RDC	bond orientation	Saupe tensor ¹	Normal	6
Chemical shift	torsion angle	Talos ²	Von-Mises ³	1
Distance	distance	None	Log-normal	2
Hydrogen bond	distance	None	Log-normal	1
Disulfide bridge	distance	None	Log-normal	0
Dihedral angle	torsion angle	None	Von-Mises	1

Table 1: Experimental NMR parameters. Left panel: Relationship between NMR parameter and structural parameter. Right panel: NMR parameters supported by ISD. [†]Number of nuisance parameters to be estimated for a particular dataset. ¹RDCs depend on the orientation of the associated bond angle with respect to an external frame of reference. Experiments establish this reference by using specific media to partially align the molecules. The Saupe tensor describes the alignment. ²Talos [31] is computer program that predicts backbone torsion angles from chemical shifts σ . ³The von-Mises distribution is the “normal distribution for periodic variables”.

equivalent groups (basically methylene and isopropyl groups), the partial NOE volumes are added up, and the observed NOE intensity I_{exp} is calculated as

$$I_{\text{exp}}(X) = \gamma \sum_{i < j} d_{ij}(X)^{-6},$$

where X denotes the three-dimensional coordinates of the structure, $d_{ij}(\cdot)$ the distance between atoms i and j , and γ the scale of the measured intensities. The scale γ is a typical theory parameter: It cannot be determined experimentally but is required in order to match calculated and measured values. During the course of a calculation, the scale is estimated from the data.

6.2 Scalar couplings

The Karplus curve is used to describe the observed three-bond scalar coupling constants 3J in terms of the intervening torsion angle φ :

$$^3J(\varphi) = A \cos^2 \varphi + B \cos \varphi + C.$$

The coefficients A , B , C of the Karplus curve cannot be determined experimentally and, therefore, need to be estimated from the data (see [34] for details).

6.3 Residual dipolar couplings

The Saupe or alignment tensor \mathbf{S} is used to describe the observed dipolar couplings D in terms of the inter-atomic bond vector \mathbf{r} :

$$D = \mathbf{r}^T \mathbf{S} \mathbf{r}. \quad (4)$$

The alignment tensor is symmetric and trace-less, $\mathbf{S}^T = \mathbf{S}$, $\text{tr}[\mathbf{S}] = 0$, and can be parametrized using five independent elements s_1, \dots, s_5 . The explicit parameterization of the alignment tensor

is:

$$\mathbf{S} = \begin{pmatrix} s_1 - s_2 & s_3 & s_4 \\ s_3 & -s_1 - s_2 & s_5 \\ s_4 & s_5 & 2s_2 \end{pmatrix}.$$

The alignment tensor describes the average orientation of the molecule in the alignment medium and also quantifies the degree of alignment. The average orientation can be calculated by diagonalizing the tensor: $\mathbf{S} = \mathbf{U}\mathbf{L}\mathbf{U}^T$. The rotation matrix \mathbf{U} describes the average orientation. The eigenvalues of \mathbf{S} (i.e. the elements of the diagonal matrix \mathbf{L}) can be transformed into an axial and rhombic component of the alignment tensor.

The tensor elements cannot be determined experimentally and, therefore, need to be estimated from the data. Several heuristics such as the histogram method [35] have been developed that allow for an approximate estimation of the axial and rhombic components from the dipolar coupling data alone. When using ISD, such preliminary analyses are superfluous, because the unknown tensor elements are treated as nuisance parameters and estimated during the actual structure calculation [33].

7 Error models

Measured and calculated NMR parameters never match. Deviations of measured from calculated data are the result of experimental noise and, often more important, approximations in the theory used to calculate the data from the three-dimensional coordinates of a structure. For example, most expressions for calculating NMR parameters neglect the dynamics of a molecule, which can lead to systematic deviations of calculated from measured values. The magnitude of these deviations is a priori unknown. In a probabilistic framework, this lack of knowledge is described by an error model. ISD supports various error models, which shall be described in the following paragraphs.

7.1 Log-normal

The lognormal distribution can be considered as the ‘‘Gaussian for positive quantities’’ and is better suited for describing positive measurands (such as distances or NOE intensities) than, for example, a Gaussian or a probability distribution that corresponds to a flat-bottom potential (see [10] for details). The density function of a lognormal distribution is

$$\Pr(A_{\text{exp}}|A_{\text{calc}}(X), \sigma) = \frac{1}{\sqrt{2\pi}\sigma A_{\text{exp}}} \exp \left\{ -\frac{1}{2\sigma^2} \log^2 \frac{A_{\text{exp}}}{A_{\text{calc}}(X)} \right\},$$

where A_{exp} and $A_{\text{calc}}(X)$ denote the observed and calculated NMR parameter, respectively. The error parameter σ relates to the width of the distribution. As mentioned above, σ quantifies the degree to which the data can be explained on the basis of a single structure, and a given theory. However, is a priori unknown and needs to be estimated from the data.

7.2 Gaussian

ISD uses a normal distribution to model the errors of scalar and dipolar coupling constants. The magnitude of these errors is a priori unknown, and is described by an error parameter σ which

relates to the width of the distribution. The density function of the normal distribution is

$$\Pr(A_{\text{exp}}|A_{\text{calc}}(X), \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp \left\{ -\frac{1}{2\sigma^2} (A_{\text{exp}} - A_{\text{calc}}(X))^2 \right\},$$

where A_{exp} and $A_{\text{calc}}(X)$ denote the observed and calculated NMR parameter, respectively.

7.3 Von Mises

The von Mises distribution is the equivalent of the normal distribution for periodic variables. ISD uses this distribution to model the deviations of measured from calculated torsion angles. These deviations are the result of experimental noise and errors in predicted torsion angles (e.g. if they are predicted from chemical shifts). The magnitude of the deviations is a priori unknown, and is described by a shape parameter κ which quantifies the precision of the torsion angle restraints. The density function of the von Mises distribution is

$$\Pr(\varphi_{\text{exp}}|\varphi_{\text{calc}}(X), \kappa) = \frac{1}{2\pi I_0(\kappa)} \exp \{ \kappa \cos(\varphi_{\text{exp}} - \varphi_{\text{calc}}(X)) \},$$

where φ_{exp} and $\varphi_{\text{calc}}(X)$ denote the observed and calculated torsion angle, respectively. I_0 is the Bessel function of the first kind. If the estimated shape parameter κ has a negative sign, this indicates that a common phase of π needs to be added to the torsion angles to obtain the best fit.

If an error estimate of the observed torsion angle is available, it can be used to weight the measurement. The von Mises distribution above is then replaced with a weighted von Mises distribution of the form:

$$\Pr(\{\varphi_{\text{exp}}\}|\{\varphi_{\text{calc}}(X)\}, \kappa, \{w\}) = \frac{1}{\prod_i 2\pi I_0(\kappa w_i)} \exp \left\{ \kappa \sum_i w_i \cos(\varphi_{\text{exp},i} - \varphi_{\text{calc},i}(X)) \right\}$$

for a set of observed and calculated angles $\varphi_{\text{exp},i}$ and $\varphi_{\text{calc},i}$ with weights $w_i \geq 0$. The global shape parameter is still estimated in a way similar to the unweighted von Mises distribution.

8 Setup of a calculation

The purpose of a project setup is to read-in the data and to create the data structures that represent the joint posterior distribution of the structure determination problem. The program also requires some information on the computing environment. The complete parameter set necessary to setup a project is contained in a single file, the *project file*. Technically, a project file is a small Python script that is executed internally in order to create all data structures that are required for running a calculation. The command syntax of Python is clear and simple which makes the project file easy to read and modify. Please note the following characteristics of Python's command syntax:

- Expressions are case sensitive.
- Indentation matters, do not use leading whitespace at the beginning of a line.
- Comments start with a hash symbol (#)

- Strings are defined with either a single (') or double (") quotation mark.
- The decimal point in floating point numbers is a dot, not a comma.
- The value of a boolean variable can be **True** or **False** (case sensitive).
- Python dictionaries (“hash tables”) map keys to values.

Example: `d = {'name': 'mr smith', 'age': 123}`

The project file contains comments which explain each of the settings in more detail. After creation of a project file, you can use your favourite text editor to modify the simulation settings.

8.1 Creating a new project

A new project file with the default settings can be created by calling ISD from the command line:

```
isd --new-project my_project.py
```

This command reads-in the default project file `project_template.py` located in the directory `$ISD_ROOT/src/py/data`, and saves it under a new name (“my_project.py” in this example). The default project file can be modified in order to use certain parameter settings by default.

Please note: project filenames must not start with numbers or special characters.

8.2 General settings

Section “General settings” of the project file summarises general parameters, most of which are mandatory and need to be specified for every simulation (unless these settings are preset in the default project file located in `$ISD_ROOT/src/py/data`):

1. A user-defined name of the simulation can be specified in the field **name**.
2. The variable **working_path** refers to the main directory of your project. ISD uses this directory to store all results, primarily the structure ensembles and a report file (in the sub-directory **working_path/analysis**).
3. The directory **temp_path** is used to store temporary files created during a simulation.
4. Set the variable **shared_temp_path** to **True** if the temporary path is shared via NFS, i.e. if it can be accessed from all machines. Please note that the temporary path has to be shared in case you choose the communication method “shared”.
5. The field **cns_executable** points to the full path of a binary of the program CNS. CNS is required only if a sequence file (**.seq**) is used to define the molecule (cf. Section 8.4), in which case CNS is called to create a PDB file with the coordinates of an extended conformation of your molecule.

6. The `fileroot` is used to compile the name of all output files. For example, if the `fileroot` is set to `'my_protein'`, PDB files are named `'my_protein_1.pdb'` etc.
7. The floating point variable `temperature` defines the physical temperature in Kelvin at which the measurements have been carried out. The default value is 300 K.
8. The parameter `naming.system` specifies the atom name nomenclature convention used in the sequence PDB file and data files. ISD supports the IUPAC standard and the CNS naming convention. Please note that internally, ISD uses the IUPAC naming system. During the startup of a simulation, atom names of sequence and data file are automatically converted to IUPAC.

8.3 Replica-exchange algorithm

As already discussed in Section 3, optimal parameter settings for a replica-exchange calculation depend on many factors, most importantly the size and quality of the data, and the size of the molecule. The section “Replica-exchange Monte Carlo” of the project file summarises the parameters that control the performance of a calculation:

1. The variable `n.replicas` specifies the number of replicas to be simulated in parallel. For most systems, 50 replicas has proven to be sufficient. However, larger systems may require a larger number of replicas in order to converge. The choice of the number of replicas is discussed in more detail in Section 11.1.
2. The number of Monte Carlo samples to be calculated in total can be specified in the field `n.samples`. By default, ISD generates 10000 samples. However, we recommend running ISD *ad infinitum* and regularly check the calculation for convergence by creating a report (see 11.1 for more details).
3. The field `hmc.steps` stores the number of Hybrid Monte Carlo steps that are generated between two attempts to exchange neighbouring states. Larger numbers improve the mixing efficiency, and thus the rate of convergence of a calculation, but reduce its speed. Experience shows that values down to 10 produce good results.
4. The number of molecular dynamics steps (ISD employs torsion angle dynamics) generated during Hybrid Monte Carlo in order to update the molecular coordinates can be specified in the field `hmc.md.steps`. The default values is 250.
5. The stepsize of the leapfrog integration scheme used in the Hybrid Monte Carlo algorithm is a parameter that is critical for the efficiency of conformational sampling. By setting the field `adjust_stepsize` to a boolean value, our automatic stepsize heuristic can be switched on (`True`) or off (`False`). It is advisable to use the heuristic because the optimal stepsize differs for different heat baths and different systems. However, in the production run the stepsize adjustment should be turned off to guarantee proper sampling (detailed balance). If `adjust_stepsize` is set to an integer, the stepsizes will be frozen when the number of super-transitions exceeds the specified number. For every heat bath, the stepsize will be fixed to the median value of the adjusted stepsizes.

6. Associated with each replica is an “ensemble” which represents the information generated in that replica (cf. Section 12). Files with the ensembles are written to disk every `save_interval` samples. The files are named as follows: `[FILE_ROOT]_n` where `n` denotes the index of the respective replica ranging from 0 to `n_replicas-1`.
7. By default, ISD only writes the two ensembles to disk that represent the lowest and highest temperature replica. Set the parameter `full_save` to `True` in case you wish to save the intermediate ensembles as well.
8. If an existing calculation is restarted, ISD normally loads the value of all auxiliary parameters from the ensemble files stored on disk. Therefore, modification of any auxiliary parameter in a project file has no effect as they will be overridden with the values stored on disk. To use the parameter values specified in the project file, set the variable `override_parameters` to `True`.
9. By default, the Monte Carlo engine runs in threaded mode, so that the user can access and control the simulation during runtime by using the interactive Python shell. A calculation can also be run in the background (e.g. by using the UNIX command `nohup`). This is necessary, for example, when ISD shall not be run inside a terminal window. In this case, set the variable `background` to `True`.
10. The name of the Python binary used on the remote machines can be specified in the field `python`. This can come in handy if the Python installation on the remote machines differs from the one on the local machine on which ISD has been started.
11. The priority of the ISD services started on remote machines can be controlled with the parameter `niceness`. The default value is 0, i.e. services are run with the highest priority.
12. The `communication` method can be either `‘shared’` or `‘pyro’` (see 5.5.2 for more details). If set to `‘shared’` the directory used as temporary path needs to be accessible from all machines and the variable `shared_temp_path` (see above) needs to be set to `True`. If the communication method is `‘pyro’`, the field `name_server` (s. below) points to the machines that runs the PyRo name server.
13. The list `host_list` contains the names of all machines that shall be used for a calculation. Machine names or IP addresses need to be given as a list of comma-separated strings, for example

```
sim.replica.host_list = '192.168.0.1', 'localhost', 'machineX'
```

or

```
sim.replica.host_list = ('192.168.0.1', 'localhost', 'machineX')
```

In order for ISD to work properly, each of these machines needs to be configured in such a way, that an SSH connection does not prompt for a password. Please see section 4.4.3 for detailed instructions of how to do that.

For making use of multi-processor machines, simply add the relevant machine names or IP addresses n times to the list, where n denotes the number of processors that shall be used.

14. If the machines do not share a common temporary directory, individual directory names can be listed in the dictionary `temp_paths`. For example, a dictionary for specifying the temporary directories `‘‘/tmp’’` and `‘‘/scratch’’` for the two machines `‘‘localhost’’` and `‘‘node1’’`, respectively, would look as follows:

```
temp_paths = {'localhost': '/tmp', 'node1': '/scratch'}
```

Temperature settings The parameter blocks `replica.weight_schedule` and `replica.prior_schedule`, respectively, contain variables that control the generalised temperatures for the likelihood function and the physical prior distribution. The temperatures T_i are calculated as a function of the replica index i , ranging from 1 to the total number of replicas, using the function

$$T_i = C \exp \{ \text{scale} \cdot i \}.$$

The normalisation constant C is chosen such that the temperatures lie within the limits specified in the fields `initial` and `final` of the respective block. A `scale` quantifies the non-linearity of the temperature curve: Small scales result in an almost linear relationship between replica index and temperature, whereas larger values lead to a curve that is more pronounced at the beginning or end, depending on whether the initial temperature is smaller or larger than the final one.¹²

8.4 Molecular system

The section “Molecular system” summarises the information required to define the sequence of your molecule and the conformation used as a starting structure of a calculation.

1. The field `filename` refers to the file which contains the sequence or the initial conformation of the molecule. A filename is not required if the sequence is read from a CCPN project.
2. The `format` of a sequence file can be PDB, SEQ (default), XML or CCPN. A sequence file in SEQ format contains the sequence in three-letter codes. If the sequence is read from a CCPN project, a SEQ, or an XML file, ISD runs CNS to create an extended starting structure with the correct sequence.
3. In case you work with a SEQ sequence file, the variable `first_residue_number` can be used to change the numbering of the first residue. This can come in handy if the residue numbering of data and sequence have a difference offset.
4. In case you wish to read the sequence from a CCPN project, specify the CCPN key of the molecule in the field `key`. Keys of objects that can be read from a CCPN project can be obtained by using the ISD command line option `--ccpn-info` (see section 8.7 for details). The import of the molecular coordinates from a CCPN project will be supported by a future version of the program.
5. The field `initial_conformation` specifies the conformation that is used as a starting structure. Set to `EXTENDED` or `RANDOM` to start from an extended or random conformation, respectively. Alternatively, you can use the coordinates stored in the PDB file as starting structure. In this case, set the field to `AS.IS`.

¹²If the parameter `initial` is larger than the value specified in the field `final`, ISD uses a negative scale.

```
#####
##                                     ##
## Experimental data                  ##
##                                     ##
#####
...

##
## BLOCK START
##
## data_filename: (string) Name of file that contains the data.
## data_format:   (const) Data format; either XML, TBL; default: XML
## data_key:      (string) Key to identify dataset
## data_name:     (string or None) Unique name for dataset. If set to None,
##               data_key is used as name. Must not contain white space.

data_type      = NOESY
data_filename  = ''
data_format    = TBL
data_key       = '13C'
data_name      = 'NOE'
...

## BLOCK END
```

Figure 7: All sections that relate to the experimental data share a common header. The header contains fields to specify informatio such as the type of a dataset and the location of its data file. “.

6. The field `exclude_hydrogens` controls if non-bonded interactions for hydrogen atoms are taken into account (`True`) or not (`False`). Non-bonded interactions are computationally expensive to evaluate. To our experience, neglecting non-bonded interactions of hydrogen atoms does not result in a serious loss of quality of the structures (RMSD to crystal structure, bumps, etc.) but leads to a significant speed-up of the calculation (about a factor of 1.5 and more depending on the system size).

8.5 Adding data

The section “Experimental data”, an excerpt of which is shown in figure 7, contains fields that need to be filled-in in order to provide information such as the type and location of your data. The lines `## BLOCK START` and `## BLOCK END` indicate the beginning and end, respectively, of the definition of a dataset. Common to all types of data are the following variables:

1. The name stored in the field `data_filename` points to the location of the file that contains for data.
2. The variable `data_format` specifies the format in which the data are stored. The format can be ISD XML (format `XML`) or CCPN (format `CCPN`). Depending on the type of a particular dataset, ISD supports other formats, such as XPLOR/CNS TBL and TALOS format (cf. section 5.2).
3. A `data_key` needs to be specified only if your data are stored in ISD XML format or inside a CCPN project. ISD uses this key to retrieve a datasets from an XML file or a CCPN project. Both data sources can contain multiple datasets. It is not required for the other formats.
4. Assigned to each dataset is a unique name, which can be specified in the field `data_name`. The name can be set to `None` (default) in which case the `data_key` is used as name. In this case, please make sure the key is unique.

The remaining fields depend on the type of a particular dataset and are discussed below. The default settings should work in most situations. To add a dataset, fill in the required parameters in the respective section, and uncomment the line `## sim_data.append(specs)`.

In order to add more than one dataset of the same kind, copy the block of instructions between the lines `## BLOCK START` and `## BLOCK END`. Make sure the line `sim_data.append(specs)` is not commented out for any dataset, i.e. remove any hash (#) symbol, when present.

8.6 Data specific settings

As mentioned above, the project file contains sections with parameters specific to each of the supported data types. The following paragraphs discuss these sections in more detail.

8.6.1 NOE intensities

Parameters required to setup the model that describes assigned NOE data are summarised in section “NOE data”. The section defines the following variables:

1. `theory.scale.update`,
2. `error_model.error.initial`,
3. `error_model.error.update`.

The theory (cf. Section 6.1) ISD uses to calculate NOE intensities introduces a scale in order to match calculated and measured intensities. It cannot be determined experimentally, and needs to be estimated from the data.

The log-normal distribution which models the deviations of calculated from measured intensities involves an error parameter. It depends on various factors and cannot be determined a priori. Like the scale, ISD estimates the error from the data, and can thus provide a measure of the quality of a dataset. Estimation of the quality of a dataset is discussed in more detail in Section 11.2

By default, the estimation of scale and error is activated. It can be turned off by setting the variables `theory.scale.update` and `error_model.error.update`, respectively, to `False`. In case the error shall not be estimated from the data, set the variable `error_model.error.initial` to the relative error of the NOE volumes measured on a distance scale between 0 and 1, i.e. 0.3 corresponds to a relative distance error of 30 %.

8.6.2 Distances

Fill-in the parameters in section “Distance data” to supply ISD with information about inter-atomic distances in your molecule. This section defines the following variables:

1. `theory.scale.update`,
2. `error_model.error.initial`,
3. `error_model.error.update`.

As in case of NOE data, a log-normal distribution is an appropriate error model to describe the deviations of the distances provided and the distances found in the structure. By default, the error σ is estimated from the data. To turn the estimation off, set the field `error_model.error.update` to `False` and `error_model.error.initial` to the desired relative distance error on a scale between 0 and 1, i.e. 0.3 for a relative error of 30 %.

ISD assumes that the overall scale of the inter-atomic distances is correct. If this assumption can be incorrect, the scale can be estimated during the calculation, by setting the variable `theory.scale.update` to `True`.

8.6.3 Scalar couplings

Measurements of scalar couplings can be added in section “Scalar coupling”. This section defines the following variables:

1. `theory.karplus_curve.A`,
2. `theory.karplus_curve.B`,
3. `theory.karplus_curve.C`,
4. `theory.karplus_curve.update`,
5. `error_model.error.initial`,
6. `error_model.error.update`.

Theoretical couplings are calculated from a structure based on the Karplus relationship. By default, the expansion coefficients *A*, *B*, and *C* do not need to be set as they are estimated from the data during a calculation. The estimation can be turned off by setting the field `theory.karplus_curve.update` to `False`. In this case, the variables `theory.karplus_curve.{A,B,C}` need to be set accordingly.

A normal distribution with unknown error describes the deviations of calculated from measured couplings. To turn the error estimation off, set the field `error_model.error.update` to `False` and `error_model.error.initial` to the absolute error in Hz.

8.6.4 Residual dipolar couplings

Measurements of residual dipolar couplings can be added in section “Dipolar couplings”. This section defines the following variables:

1. `theory.saupe_tensor.s1`,
2. `theory.saupe_tensor.s2`,
3. `theory.saupe_tensor.s3`,
4. `theory.saupe_tensor.s4`,
5. `theory.saupe_tensor.s5`,
6. `theory.saupe_tensor.update`,
7. `error_model.error.initial`,
8. `error_model.error.update`.

Theoretical dipolar couplings are calculated from a structure based on the relationship (4). By default, the tensor elements s_1, \dots, s_5 do not need to be set as they are estimated from the data during a calculation. The estimation can be turned off by setting the field `theory.saupe_tensor.update` to `False`. In this case, the variables `theory.saupe_tensor.{s1,s2,s3,s4,s5}` need to be set accordingly.

A normal distribution with unknown error describes the deviations of calculated from measured couplings. To turn the error estimation off, set the field `error_model.error.update` to `False` and `error_model.error.initial` to the absolute error in Hz.

8.6.5 Dihedral angles

Fill-in the parameters in section “Dihedral angles” to supply ISD with information on angles around rotatable bounds of your molecule. Apart from the standard formats XML and XPLOR/CNS TBL, ISD can also handle files containing information on ϕ/ψ dihedral angles predicted by the program Talos. In this case, set the field `data_format` to `TALOS`. Upon simulation startup, Talos data are automatically converted into XML. This section defines the following variables:

1. `error_model.error.update`,
2. `error_model.use_weighting`.

The discrepancies between the dihedral angles and their counterparts in a structure are modelled by a von Mises distribution, the width of which is estimated during a calculation. Optionally, dihedral angles can be weighted by their errors if available (the program Talos, for example, provides such error estimates). Whether to use individual weighting or not, can be specified in project file by setting the field `use_weighting` to `True` or `False`, respectively.

8.6.6 Hydrogen bonds

Measurements of hydrogen bonds can be added in section “Hydrogen bonds”. In the current version of ISD, hydrogen bonds need to be provided as a list of hydrogen-acceptor and donor-acceptor distances. Typically these distances should be set to 1.8 Å for the hydrogen-acceptor, and 2.8 Å for the donor-acceptor distance. The hydrogen bond section defines the following variables:

1. `error_model.error.initial`,
2. `error_model.error.update`.

As for any distance data, a log-normal distribution is an appropriate error model to describe the deviations of the distances provided and the distances found in the structure. By default, the error is estimated from the data. To turn the estimation off, set the field `error_model.error.update` to `False` and `error_model.error.initial` to the desired relative distance error on a scale between 0 and 1, i.e. 0.3 for a relative error of 30 %.

8.6.7 Disulfide bridges

Fill-in the parameters in section “Disulfide bridges” to supply ISD with information on disulfide bridges in your molecule. In the current version of ISD, disulfide bridges are implemented by restraining the distance between the involved sulfur atoms. The section defines the following variables:

1. `distance`
2. `error_model.error.initial`,
3. `error_model.error.update`.

As for any distance data, a log-normal distribution is an appropriate error model to describe the deviations of the distances and the distances between the bridging sulfur atoms found in the structure. By default, the estimation of the error is disabled. To turn the error estimation on, set the field `error_model.error.update` to `True`.

8.7 Data import/export from a CCPN project

ISD supports both the import of data from, and the export of the calculation results to a CCPN project. Section “CCPN data model” in a project file summarises the parameters that control this feature. The CCPN project must exist before you can use the import/export feature. New CCPN projects can be created with CcpNmr Analysis or the FormatConverter.

Import

Each object, such as a molecular system or a list of distance restraints, stored in a CCPN project has a unique key. The keys are used by ISD to access objects within a CCPN project. A list of objects/keys can be obtained by running ISD with the command line option `--ccpn-info` and the name of the CCPN project as argument.

In order to import a particular molecular system or dataset from a CCPN project, you first need to make sure the field `ccpn.project_filename` points to the name of your CCPN project. In a second step, all that needs to be done is to specify the key of the objects you wish to use in the fields `data_set.key` in the respective sections in the project file (or in the variable `molecule.key` in case of a sequence). The variable `data_format` needs to be set to `CCPN`; the field `data_filename` can be left blank. In case you wish to use a nickname for your dataset, simply specify the name in the variable `data_name`. If set, nicknames appear in the PDF summary.

For example, to read a sequence with key `‘mol|A’` as well as an NOE restraint list with key `‘test|1|1’` from a CCPN project, and use “noesy” as a nickname for the NOE restraints, the relevant parts in the project file would look as follows:

```
sim.ccpn.project_filename = '/home/guest/my_ccpn_project.xml'
...
sim.molecule.filename    = ''
sim.molecule.format      = CCPN
sim.molecule.key         = 'mol|A'
...
data_type                 = NOESY
data_filename             = ''
data_format               = CCPN
data_key                  = 'noeC13|1|1'
data_name                 = 'noesy'
...
```

Export

The current implementation supports the export of an ensemble of structures to a CCPN project. As mentioned above, the CCPN project must exist before the export feature can be used. All exported objects are stored in an “NmrProject” (to use CCPN terminology) of a CCPN project, which serve as containers for the exported objects. The name of NmrProject project needs to be specified in the field `nmr_project_name`. ISD creates a new NmrProject if it does not exist yet.

Structures To export an ensemble of structures, set the variable `ensemble.enabled` to `True` (default: `False`). Both the ensemble type (`REPRESENTATIVE` or `MOST_PROBABLE`) as well as the number of structures to be exported can be set in section `pdb_files` (cf. section 9). The name under which the structures are stored (the CCPN “MolSystem”) needs to be specified in the field `molecular_system_name` and must not contain white space.

8.8 Starting a calculation

Once the project file `my_project.py` has been modified and saved, start the calculation by invoking the command:

```
isd my_project.py
```

During the initialization of a calculation, ISD performs the following tasks:

1. Create temporary directory,
2. Create molecule (possibly from a sequence file by using CNS),
3. Read-in experimental data (possibly converting data formatted in proprietary formats into ISD XML format),
4. Start services on cluster,
5. Create initial conformation.

After these tasks have been completed, ISD launches the replica-exchange simulation using parameters and machines specified in the project file. During the runtime of a simulation, the internal data structures of ISD can readily be accessed by using the interactive Python shell. A list of runtime commands is shown below. Ensemble files that contain the Monte Carlo samples are written to disk after a certain number of replica super-transitions. PDB files with representative ensemble members can be generated by creating a report (s. section 10) or by using the command line (s. section 9).

Please note that a “correct” structure ensemble is only obtained after an initial convergence phase (the “burn-in”) of a calculation. A non-converged simulation usually yields structures of sub-optimal quality. Section 10 discusses how to assess the convergence of a calculation. Advanced users may want to write their own scripts in order to analyse the Monte Carlo sample in depth. Section 12 gives some examples of how this can be done.

8.8.1 Restarting a calculation

In case a simulation or one of the machines used for the calculation has crashed, restart the simulation simply by using the same command as for starting a calculation. ISD then automatically resumes with the calculation where it has stopped before.

8.8.2 Running a calculation in the background

We recommend running ISD inside a terminal window as the interactive Python shell gives the user full control over the calculation during runtime. However, ISD can also be run in the background, e.g. detached from a terminal window or by using the UNIX program `nohup`. In order for this to work, the Monte Carlo engine needs to be run in non-threaded mode, which is controlled by the variable `background` in the section “Replica-exchange Monte Carlo” of the project file, which needs to be set to `True`. To quit a calculation that is running in the background, use the command line option `--quit`.

8.9 Runtime commands

ISD supports a number of runtime commands, a list of which can be obtained with the command `help`:

1. The command `save` writes all ensembles to your working directory. If the option `analysis.report.auto` in section “Analyses / report” of the project file is set to `True`, ISD also generates a PDF report containing an analysis of the simulation.
2. Use the command `show` to display the current structure in your PDB viewer. A filename pointing to the binary of your favourite molecular graphics program can be set in section “Analyses / report” of the project file.
3. The command `info` gives a brief summary of the current simulation, such as the number of Monte Carlo samples generated so far.
4. The command `report` creates a PDF report which is stored in the directory `working_path/analysis`. Alternatively, a report can also be generated using `isd` from the command line. The ISD report is discussed in more detail in Section 10
5. ISD can be quit by either using the command `quit` or by exiting the Python interpreter by pressing Ctrl-D. If the calculation is run in the background, use the command line option `--quit` with the name of the project file as argument.

8.10 Example projects

The distribution comes with two example projects which can be found in the sub-directory `./examples`. The first example dataset provided is for the protein Ubiquitin [36], the second one for a Tudor domain¹³ [37,38]. Each of the example directories contains a sub-directory `./pdb` with representative ensemble members, and a sub-directory `report` containing a PDF report.

8.10.1 Tudor domain

The project file for this example can be found in the directory `./examples/tudor`. The sequence (56 amino acids), a list of assigned NOE intensities from two NOESY experiments (¹³C and ¹⁵N) as

¹³We thank Michael Sattler for kindly providing this dataset.

well as scalar coupling constants are contained in the data file `examples/tudor/data/data.xml`, formatted in ISD XML format. By default, the calculation runs on the local machine, and the complete set of nuisance parameters (scales, Karplus coefficients, and errors) is estimated from the data. In order to run the example on a computer cluster, please specify an appropriate list of machines in the project file. Please note that the calculation of quality scores has been turned off. To activate, please modify the project file. If ISD is correctly setup, you can run the example calculation from the command line using

```
isd tudor.py
```

8.10.2 Ubiquitin

The project file for this example can be found in the directory `./examples/ubq`. The sequence (76 amino acids) and lists of nonredundant NOE-based distance restraints, hydrogen bond restraints, scalar and dipolar coupling constants are contained in the data file `examples/ubq/data/data.xml`, formatted in ISD XML format. The NOEs, J couplings and RDCs are part of the original restraint file obtainable at the PDB (ID code: 1D3Z). In addition, dihedral angle restraints predicted with the TALOS program are available (`examples/ubq/data/talos.tab`). By default, the calculation runs on the local machine, and the complete set of nuisance parameters (scales, Karplus coefficients, tensor elements, and errors) is estimated from the data. In order to run the example on a computer cluster, please specify an appropriate list of machines in the project file. Please note that the calculation of quality scores has been turned off. To activate, please modify the project file. If ISD is correctly setup, you can run the example calculation from the command line using

```
isd ubq.py
```

9 Creating PDB files

PDB files with the 3D coordinates of the ensemble members are created automatically whenever you generate a report. The structures are superimposed on the average structure and are stored in the sub-directory `./analysis/structures` of the working directory of the project. Alternatively, PDB files can be generated from the command line using the option `--write-pdb`. In this case, the files are stored in your current directory.

The sub-section “`pdb_files`” of a project file summarises parameters that control the way PDB files are generated:

1. Set the field `ensemble` to `REPRESENTATIVE` or `MOST_PROBABLE` to store PDB files containing representative ensemble members, or most probable structures, respectively. Please note that the variance observed in a “most probable” ensemble (unlike in a “representative” ensemble) under-estimates the true uncertainty of the 3D coordinates.
2. The parameter `n` defines how many ensemble members shall be written to disk.
3. Each PDB file contains the coordinates of residues that are specified in the comma-separated list `residue_list`. If the field is set to `None` (default) the coordinates of all residues are saved.

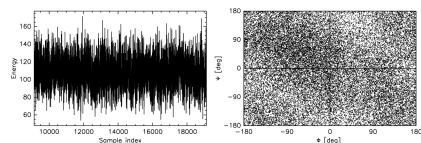


Figure 3: Left: trace of the total energy associated with the target distribution. For converged calculations the energy should scatter around its median (dotted). Right: t-v plot for all residues of the high temperature ensemble. For an appropriate k_{ij} schedule this distribution should be uniform.

to analyse a particular NMR experiment. It also depends, to a lesser extent, on approximations in the background assumptions, such as the form of the force field used to describe physico-chemical interactions between the atoms of a molecule. In others, incomplete data always leads to uncertain structures, as common sense would suggest.

2.1 Uncertainty

It is important to keep in mind that structural uncertainty should always be regarded in the sense of an “error bar” of a structure. This error bar is of statistical nature, that is there is no causal connection to real physical fluctuations of the atom positions. Physical fluctuations can be one of the reasons why a structure is uncertain¹⁴, uncertainty and fluctuations might even correlate on a qualitative level. Quantitatively, however, we cannot infer dynamics from structural uncertainty.

The uncertainty of the present structure, quantified via the median uncertainty in the position of its C_{α} atoms (s. Figure 4) has been calculated on the basis of 100 ensemble members and amounts to $\sigma_{\text{RMSD}} = 1.44 \pm 0.66$ Å. The directory `/WORKINGPATH/analysis/structures` contains 100 members of the structure ensemble (stored in IUPAC PDB format).

2.2 Quality scores

In order to validate the structure further, we calculate a number of quality scores using the programs Whatif and Procheck. The scores are based on our knowledge about general geometric features of proteins, and are derived by comparing the three-dimensional coordinates of the calculated structures with a database of high-resolution x-ray structures. The scores (see table 2) include measures of the packing of a protein, of the local geometry, as well as of the compatibility of the backbone with known protein structures.

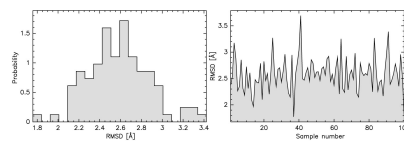


Figure 6: Distribution and trace of the RMSD to the reference structure, calculated for the atoms CA, C, N, O. The most probable structure has an RMSD of 1.76 Å.

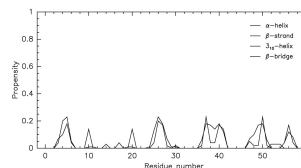


Figure 7: Secondary structure assignment. Shown is the propensity with which a residue adopts the secondary structural states α -helical, β -strand, 3_{10} -helical or isolated β -bridge. Secondary structure assignments were derived with the program DSSP; propensities are calculated on the basis of the conformational ensemble.

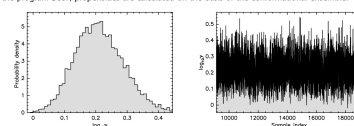


Figure 8: Probability distribution (left) and trace (right) of the scale (‘calibration factor’) γ used in the ISPA to relate measured peak intensities to distances.

Figure 8: Excerpt of an ISD report. A report summarises the calculation results in the form of a PDF document. It provides graphical information on the performance of a calculation, as well as various analyses, such as of the data quality, structure validation scores, and estimates of all theory parameters.

A list of numbers ranging from 1 to (including) n can be created with the Python command `range(1,n+1)`.

10 The ISD report

ISD stores all calculation results in Python’s persistent object format (“Python pickles”)¹⁴. Users familiar with Python can thus access the full information generated during a calculation to perform further analyses. How this is done is discussed in some detail in Section 12. A more convenient way of accessing the simulation results is to create a report which summarises the results in the form of a PDF document (cf. figure 8). An ISD report can be generated during the run-time of a simulation. It is stored in the directory `working_path/analysis`, and can be created by using the following command:

```
isd --report my_project.py
```

The report provides graphical information on the performance of the calculation, analyses of each dataset (for example, an estimate of its quality). It also gives estimates of all theory parameters. Furthermore PDB files are generated that contain the coordinates of representative members of the probabilistic structure ensemble. The ensemble members live in the same Cartesian frame of reference, and thus can directly be visualized in your favourite molecular graphics program without prior superposition. PDB files with the 3D coordinates of the ensemble members are stored in the directory `analysis/structures`.

¹⁴For more information on Python persistent objects, please see the documentation available at www.python.org.

10.1 General settings

Section “Analyses / report” of the project file contains parameters that control the generation of a report:

1. In order to obtain unbiased estimates of a structure, the uncertainty of the 3D coordinates, and auxiliary parameters, ISD needs to make sure it restricts the analyses to the converged part of a calculation only. However, as will be discussed in Section 11.1 it is difficult to determine the point at which a calculation has converged in an automatic fashion. Therefore, the variable `burnin` needs to be specified by the user to define the number of most recent samples to be used during the analyses. The default value is 1000, i.e. the analyses are based on the last 1000 samples.
2. If the variable `report.auto` is set to `True`, a report is generated automatically whenever a simulation is saved to disk. This can come in handy to monitor the results of a calculation on a regular basis.
3. In case you wish to keep the files that were used to generate the PDF report (Latex sources and figures in EPS and PDF format), set the variable `report.keep_sources` to `True`. The sources are stored in the directory `[WORKING_PATH]/analysis/sources`.
4. The setting `pdb_viewer` is unrelated to the report. However, it points the program that is used to visualise structures using the interactive command `show` (cf. Section 8.8).
5. The name of the binary of the utility programs `pdflatex`, `epstopdf`, and `eps2eps` can be specified in the variables `pdf_latex`, `eps_to_pdf`, and `eps_to_eps`, respectively. ISD requires either of these programs for creating a PDF report.

Please note that on SUSE 10.x systems, it has been reported that post-processing of EPS files is required in order for the report to be formatted correctly. Post-processing is turned off by default (`eps_to_eps` is set to `''`).

10.2 Quality assessment and structural analyses

ISD calculates a number of quality scores and performs structural analyses to validate the structures. This requires the programs WhatIf, Procheck, and DSSP. The analyses are carried out for the PDB files that are generated during the report, see section 9 for details.

10.2.1 Quality scores

Scores for the packing quality, the normality of the backbone conformation, etc. are calculated with the program WhatIf. The field `whatif.binary` points to the location of the executable of WhatIf. Disabled by default, calculation of quality scores can be enabled by setting the variable `whatif.enabled` to `True`. If you wish to use the program WhatCheck (which is free of charge, by the time we write this manual) instead of WhatIf, set the variable `whatif.use_whatcheck` to `True`. The quality scores can also be depicted on a per sample basis. The feature is controlled by the variable `whatif.show_traces`.

10.2.2 Ramachandran statistics

The program Procheck is used to calculate the proportion of residues that populate the most favoured and allowed regions of the Ramachandran plot. The field `procheck.binary` points to the location of a binary of Procheck. Disabled by default, use of Procheck can be turned on by setting the variable `procheck.enabled` to `True`. The scores can also be presented on a per sample basis. The feature is controlled by the variable `procheck.show_traces`.

10.2.3 Secondary structure

Secondary structure propensities are calculated with the program DSSP. The propensities (α -helical, β -strand, 3_{10} -helical or isolated β -bridge) are presented in a graphical form on a per residue basis. The field `dssp.binary` points to the location of a binary of DSSP. Disabled by default, calculation of propensities can be enabled by setting the variable `dssp.enabled` to `True`.

11 Analysing the results

An important first step in the analysis of a calculation is to make sure the simulation has converged properly. In case a simulation has not converged, the structures and parameter sets generated were not drawn from the joint posterior distribution. Typically this leads to sub-optimal or, in the worst case, incorrect results.

The second step in the analysis should be devoted to an examination of the quality of each dataset. On the one hand, this can provide another indication of whether a simulation has converged. As the quality of typical datasets usually falls into a narrow range - this is particularly true for NOE data: Overly low data qualities might hint to a general incompatibility of data and structure which is often a result of a non-converged simulation. On the other hand, the data quality can provide valuable information for validating the experimental data.

11.1 Convergence of a calculation

Unfortunately, the current theory of Markov chains does not provide conditions that are sufficient to assess the convergence of a simulation in a mathematically rigorous way. Instead, we need to limit ourselves to the use of heuristic rules. A similar problem arises in conventional structure calculation methods where the difficulty is to distinguish converged from non-converged structures.

In technical terms, if an ISD simulation has converged it means that the sequence of structures and parameter sets, initially drawn from a non-equilibrium distribution¹⁵, has finally become compatible with the joint posterior distribution of an inferential structure determination problem. In other words, for converged simulations one finds that the frequency of occurrence of a particular structure is equal to its probability. Thus, probable structures occur more often in the ensemble than less probable ones.

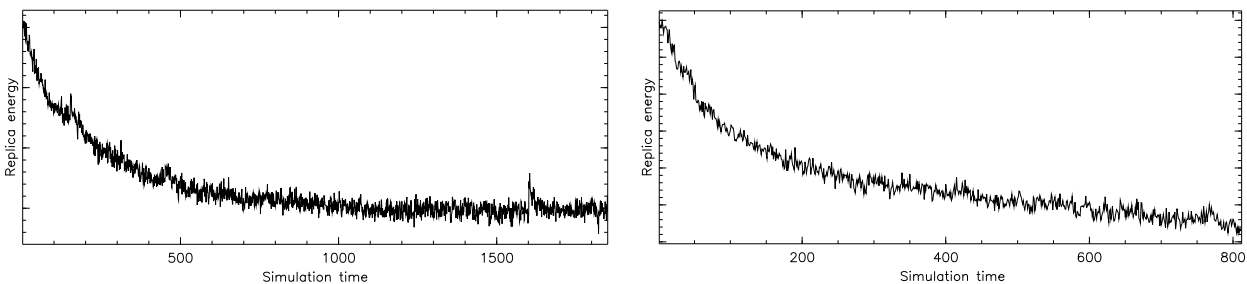


Table 2: Total energy of a calculation. The trace of the total energy can be used to monitor the convergence of a calculation. Ideally, the energy drops quickly in the early phase of a calculation and reaches a plateau. The simulation shown in the right panel has not converged as the energy is not stationary. The left panel shows an example of a converged calculation.

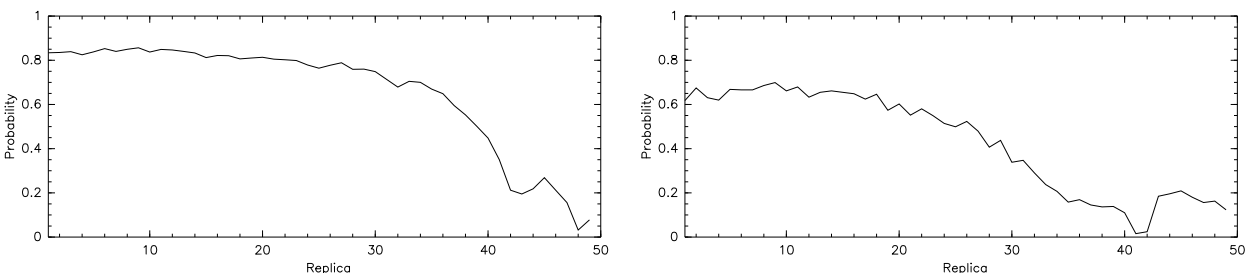


Table 3: Exchange rates. The example on the left shows a simulation which is reasonable well behaved. The exchange rates are homogeneous, and fall off only at the high-temperature end of the replica arrangement. The right panel shows a case in which the temperature scheme has not been chosen adequately. This results in weak mixing and, hence, slower convergence.

11.1.1 Total energy

A suitable quantity to monitor the convergence of a simulation on an empirical level is the “total energy” of a replica calculation. It is defined as the energy associated with each replica, summed over all replicas. During the so-called “burn-in” phase, the total energy typically falls off, and finally reaches an equilibrium value around which it scatters for the rest of a calculation. Figure 2 shows examples of a converged (left) and non-converged simulation (right). Similar figures are contained in an ISD report in Section “Performance”. Unless the total energy is stationary, a calculation should be considered not converged.

11.1.2 Rates of exchange

In order for the total energy to drop quickly, a simulation need to mix sufficiently. This means that conformations need to propagate efficiently up and down in the replica arrangement. As discussed briefly in Section 3, the limiting step for this to happen is the rate by which neighbouring replicas exchange. The rate of exchange is determined by the general temperature settings applied to the likelihood and the prior distribution: Large temperature differences of neighbouring replicas result

¹⁵This is because a calculation starts from a different conformation than the structure we seek to determine.

in lower rates of exchanges. Basically, the reason for this behaviour is that the associated probability distributions are sufficiently dissimilar so that states generated in both replicas are incompatible with each other, hence no exchange.

For a good performance of a calculation it is, therefore, important to choose a suitable temperature scheme. An obvious solution would be to choose the temperatures in such a way that its differences are sufficiently small. However, the spacing cannot be made arbitrarily small as the replicas needs to cover a certain temperature range (the temperatures of the lowest and highest replica are fixed). The effect of the temperature settings on the rate of exchange depends on many factors, mainly the size of datasets and system. The default temperature settings are suitable for standard systems of moderate size. However, in particular for larger systems or datasets, the settings may need to be adjusted.

The section “Protocol” in a report provides some information about the rates of exchange of a calculation. The rates should be homogeneous and not fall below 20 %. Sudden drops, however, especially at the high-temperature end of an replica arrangement, are often unavoidable. Figure 3 shows an example of well and less well behaved simulation. The average rate of exchange of the simulation shown in the right panel is low compared to the well behaved calculation shown on the left. The exchange rate drops to nearly zero for replicas 41 and 42, which leads to very slow convergence as there will be practically no exchange of states.

11.1.3 High temperature distribution

If the temperature schedule as been chosen appropriately, the probability distribution associated with the high-temperature replica (the rightmost distribution in figure 3) is expected to be nearly uniform. If this is not the case, for example because the maximum temperature is not sufficiently high, the simulation cannot escape local modes (or, equivalently, overcome energy barriers) and will converge only slowly. In order to visualise the topography of the this distribution, section “Performance” of the report contains a figure that shows the backbone dihedral angles ϕ/ψ of all sampled conformations. Ideally, the dihedral angles are distributed uniformly, although realistic cases almost always show some residue of structure. Please note that the ϕ/ψ -distribution is not expected to look like a Ramachandran plot.

11.1.4 Improving convergence

If the empirical rules discussed so far suggest that the calculation has not converged yet, one should increase the calculation time. Alternatively, one can adjust the algorithmic settings in order to speed up convergence. This includes:

1. Choosing a larger number of replicas usually increases the average rate of convergence, as it reduces the temperature differences between neighbouring copies. This often solves the problem, but comes at the cost of longer calculation times. However, in particular for larger systems, using more replicas is often the only way to achieve a reasonable rate of convergence.
2. If the ϕ/ψ -distribution associated with the high-temperature replica shows a significant amount of structure, one should increase its temperature. This can either be done by increasing the maximum temperature which transforms the physical prior distribution, or by lowering the minimum value of the likelihood weight. Both parameters are discussed in more detail in Section 8.3.

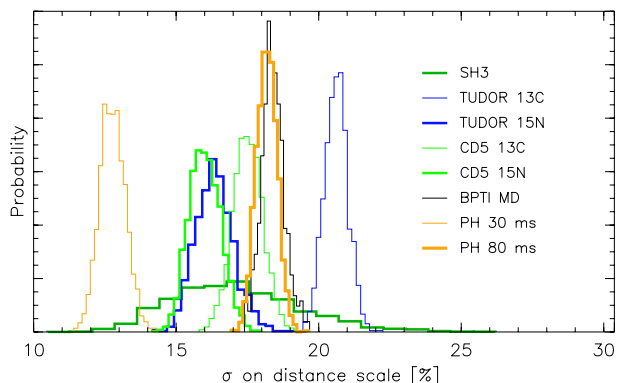


Figure 9: Quality of NOE data. Shown are the distributions for the relative error σ for eight different datasets. When transformed to a distance scale, the errors fall into a narrow range, as long as a dataset does not contain systematic errors. One finds that the discrepancy between measured and calculated ^{15}N data is usually smaller compared to ^{13}C measurements.

3. The average rate of exchange can also be optimized by altering the scale in the functional relationship of temperature and replica index, i.e. the function which assigns a temperature to each replica. Section 8.3 gives brief instructions for how to do that.

If the convergence behaviour has still not improved after taking either of these measures, the only solution to the problem is to run the simulation for a longer time.

11.2 Data quality

An important parameter that is suited to validate the experimental data is the quality of a dataset. Models for NMR observables almost always involve an error-like parameter, called σ here, which ISD estimates individually for every set. We use σ as a measure of the data quality. It is independent of the size of a dataset, and measures the degree to which the measurements can be explained on the basis of a single structure. It thus reflects the consistency of the data, limited, for example, by experimental noise, but also depends on the theory used to analyse the data.

As mentioned before, ISD estimates the error of a dataset during a calculation. The error directly relates to the weight with which the data are incorporated into the joint posterior distribution. This ensures that high quality datasets are assigned higher weights than sets with a lower quality. Furthermore, errors are estimated individually for each set, so that the data are incorporated in a calculation in an optimal manner. The issue of weighting experimental data is discussed in more detail in [39].

In order to quantify the quality of NOE data it is convenient to transform the intensity error to a distance scale. Experience shows that NOE data that do not contain systematic errors are of similar quality. As a rule of thumb, the relative distance error should be less than about 20 %. The report contains estimates for the error of all datasets used during a calculation. One also finds that the discrepancy between measured and calculated ^{15}N data is usually smaller than for ^{13}C measurements. One reason for this behaviour is that hetero-nuclear relaxation, which is not accounted for by the ISPA, affects ^{13}C data more than ^{15}N data.

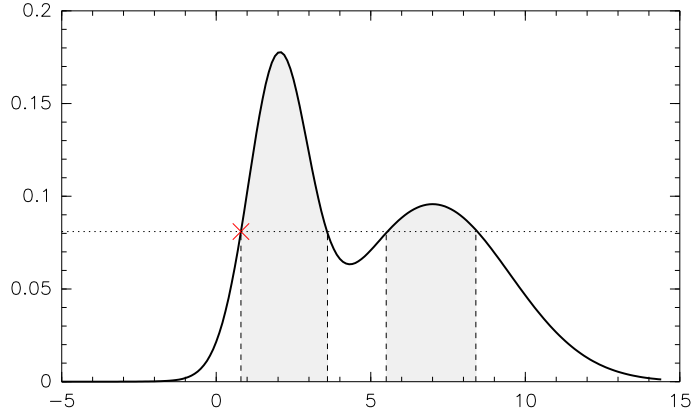


Figure 10: Violation analysis. Shown is a hypothetical predictive distribution (solid line). The red cross indicates the measurement. The probability of this data point to be inconsistent is constructed as follows: (i) draw a line that is parallel to the x axis and passes through the data point (dotted line); (ii) wherever the line intersects with the graph of the predictive distribution, draw a parallel to the y axis (dashed lines); (iii) intersections of the y parallels with the x axis define x intervals, intervals with a probability equal or greater than the probability of the data point define the confidence interval that is associated with the measurement; (iv) the probability mass carried by the confidence interval is considered as the probability that the measurement is inconsistent (sum of the grey areas). That is, measurements that are close to modes of the predictive distribution have only a small associated confidence interval and therefore little probability of being wrong. For measurements falling into the low probability regions of the predictive distribution, the opposite holds.

The report also contains estimates of the error for other datasets, such as three-bond scalar couplings and residual dipolar couplings, for which the error is specified in absolute terms in Hz.

11.2.1 Reliability of individual measurements

The sampled conformations and nuisance parameters can be used to calculate the expected distribution for each measurement under the model assumptions and the experimental data. Such distributions are called *predictive distributions*. Predictive distributions express what we expect for the outcome of a new experiment, given our current knowledge of the structure and nuisance parameters, inferred from the data. Formally, if y is your observable and $f(X, \alpha)$ a theory used to calculate mock data, then the expected distribution of y is

$$\Pr(y|D, I) = \int dX d\alpha d\sigma \Pr(X, \alpha, \sigma|D, I) \Pr(y|f(X, \alpha), \sigma, I).$$

These distributions can be used to assess the consistency of the data with the modelling assumptions and with one another: A measurement is likely to be wrong, if it deviates from its prediction based on the complete dataset. The magnitude of the deviation relates to the probability that the measurement is incorrect.

Predictive distributions can be used to assess the quality of a data point a posteriori. We define the probability associated with the confidence interval that just contains the measurement as the probability that the measurement is inconsistent with the model and the other data. Figure 10 illustrates how this probability is constructed. For a Gaussian model, it is directly related to the Z score of a measurement: a measurement that is one standard deviation off the mean has Z score 1

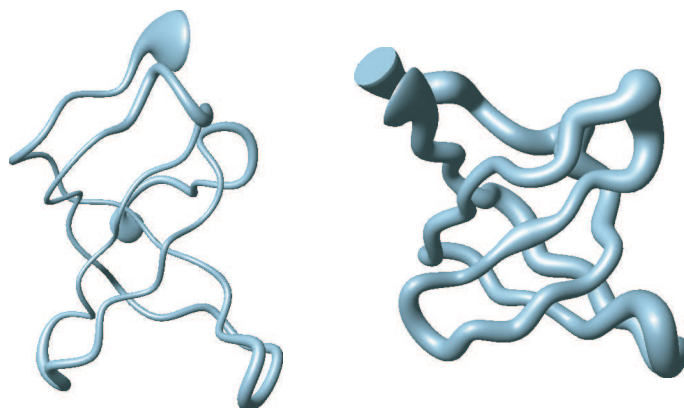


Figure 11: Structural uncertainty. Shown is the structure and uncertainty of a TUDOR domain (left) and an SH3 domain (right). Both datasets are of similar quality (relative distance error of about 20 %). The size of the TUDOR dataset is about ten times larger than the one used for the SH3 domain.

and a probability of 68% of being inconsistent or “wrong”, a measurement with Z score 3 is much less reliable and has a probability of 99% of being wrong.

Similar to a conventional violation analysis, we can now assess the quality of the dataset by defining a threshold above which we consider measurements as incorrect. However, a probabilistic analog of the standard violation analysis is more general than the standard procedure: It takes the uncertainty of the structure, which might vary in different regions of the molecule, fully in account. Furthermore, violation probabilities are not restricted to error models that correspond to flat-bottom potentials¹⁶, but can be calculated for arbitrary error distributions. Here, we assess the quality of the data by defining a threshold above which we consider measurements as incorrect. This threshold is set to 95% by default. Section “Violation analysis” in the report gives a summary of the analysis, and measurements that are likely to be erroneous are explicitly listed in report files stored in the directory `WORKING_PATH/analysis`.

11.3 Structure validation

11.3.1 Structural uncertainty

The uncertainty of the structure is directly represented by the spread of the posterior distribution. Thus, in order to calculate the uncertainty, we need to explore the relevant part of this distribution. This is one of the reasons why ISD employs sampling rather than minimisation techniques. The precision with which we can determine the three-dimensional coordinates of a target structure depends on the amount of information we start with. Hence, datasets of low quality and/or of small size generally lead to less precise structures - just as common sense would suggest (figure 11 shows an example).

The section “Structural analysis” in the report states the overall precision of the structure, quantified via the median uncertainty of the C_{α} positions. It also shows the local uncertainty in graphical form, calculated for the backbone C_{α} atoms. Furthermore, ISD stores PDB files with

¹⁶As has been argued earlier, ISD does not use flat-bottom potentials since they tend to decrease the explanatory power of the data, and furthermore produce less accurate and precise structures compared to proper error distributions

representative members of the conformational ensemble in the directory `analysis/structures`. The ensemble members live in the same Cartesian frame of reference, i.e. the structure ensemble can directly be visualised without prior superposition. The most probable structure is stored in the PDB file `structures/[FILE_ROOT]_most_probable.pdb`.

11.3.2 Quality scores

In order to validate a structure further, ISD calculates a number of quality scores using the programs WhatIf and Procheck. The scores are based on the knowledge we have about general geometric features of proteins, and are derived by comparing the three-dimensional coordinates of the calculated structures with a database of high-resolution x-ray structures. A table listing the scores can be found in section “Structural analysis” in the PDF report. The scores include measures of the packing of a protein (first and second generation packing score), of the local geometry (Ramachandran appearance and percentage coverage of the most favoured region of the Ramachandran plot), as well as of the compatibility of the backbone with known protein structures (backbone normality score).

12 Accessing simulation results using Python

ISD stores the information generated during a calculation in the form of Python persistent objects (“Python pickles”). The data are stored on a per-sample basis, which closely matches the way the data are generated. The Python class `State` represents and stores the information created during one replica-exchange Monte Carlo step. That is, if the simulation has a length of, say, 5000 samples, ISD creates 5000 `State` objects. Each state stores, among other things, information such as the dihedral angles of the respective conformation, and the “energy” of that conformation (defined as the negative logarithm of its probability). A set of samples makes up an ensemble, described by the Python class `Ensemble`. ISD represents the information generated in each replica by an associated ensemble object. A couple of example Python scripts illustrate how to use the ISD Python library to manipulate data and analyse results etc. The scripts can be accessed via the website.

12.1 Creating a posterior object

In order to load an ensemble into memory, we first need to make sure Python has access to the required classes. In case the environment variable `ISD_ROOT` has been set correctly, this is straightforward:

```
import sys, os

modules = os.path.join(os.environ['ISD_ROOT'], 'src/py')
sys.path.insert(0, modules)
```

The next step is to create the Python objects that represent the posterior probability distribution of a simulation. If `my_project.py` is the project file of the simulation, we can create these objects simply by using the following lines:

```
import setup

simulation = setup.create_simulation_from_project('my_project.py')
posterior = simulation.posterior
```

12.2 Loading and accessing “ensembles”

To load the ensemble stored, for example, in the file `my_protein_0` (i.e. the low temperature ensemble), use the following command:

```
from utils import Load

ensemble = Load('my_protein_0')
```

Ensembles behave both like Python dictionaries and lists. A state in an ensemble can be accessed simply as follows:

```
state = ensemble[123]

>>> print state

State(569 torsion angle(s), E=2.40e+03, E_phys=1.02e+02, E_boltzmann=1.02e+02,
E_prior=1.00e+02, E_likelihood=2.30e+03, E_data=2.30e+03, sub_states={'13C':
SubState(name=13C, E=1.08e+03, E_prior=-1.05e+00, E_likelihood=1.08e+03,
E_data=1.08e+03, k=5.21e-01), 'phi': SubState(name=phi, E=-1.20e+02,
E_prior=0.00e+00, E_likelihood=-1.20e+02, E_data=-1.20e+02, k=1.30e+00),
'hbonds': SubState(name=hbonds, E=-1.13e+01, E_prior=1.24e+00,
E_likelihood=-1.26e+01, E_data=-1.26e+01, k=3.45e+00)})
```

The dictionary `sub_states` (see the printout above) is an attribute of `State` and links to information on the different datasets used in a calculation. In the present case these are NOE (“13C”), dihedral angle (“phi”), and hydrogen bond (“hbonds”) data. The dictionary `sub_states` maps the name of a dataset to a `SubState` object which, just like a `State`, contains information on parameters specific to a particular dataset, such as its error σ . For example, the error of dataset “13C” for the 1000th sample can be obtained as follows:

```
from Numeric import sqrt

error = 1./sqrt(ensemble[1000]['13C'].k)
```

Please note that ISD does not store the error directly, but the inverse variance $k = 1/\sigma^2$. The exact set of parameters depends on the theory and error model used to describe the respective data. Alternatively, one can use the dictionary interface of an `Ensemble` to access a whole sequence of sampled parameters of a particular set. For example, the samples for the errors parameter of dataset “13C” can be accessed as follows:

```
from Numeric import sqrt

sub_ensemble = ensemble['13C']
errors = 1./sqrt(sub_ensemble.k)

>>> print len(errors)

1243
```

In this case, the simulation contains 1243 samples in total.

12.3 Saving conformational samples as PDB files

The creation of a PDB file that contains the conformation sampled at a particular time in a simulation consists of two steps. First, one needs to calculate the three-dimensional coordinates from a set of dihedral angles. Second, the coordinates need to be written to a PDB file. This is illustrated by the following lines:

```
molecule = posterior.get_polymer()          ## first get the object that
                                             ## represents the molecule

torsion_angles = ensemble[123].torsion_angles  ## access torsion angles
                                             ## of sample no. 123

molecule.set_torsions(torsion_angles, update=1)

molecule.write_pdb('./conformation_123.pdb')
```

The last two lines first calculate the Cartesian coordinates from the torsion angles generated in sample 123, and afterwards write the coordinates in the form of a PDB file to disk.

References

1. Wüthrich, K. (2003) *Angew. Chem. Int. Ed. Engl.* **42**, 3340–3363.
2. Nabuurs, S. B, Spronk, C. A, Vuister, G. W, & Vriend, G. (2006) *PLoS Comput. Biol.* **2**, e9.
3. Rieping, W, Habeck, M, & Nilges, M. (2005) *Science* **309**, 303–306.
4. Habeck, M, Nilges, M, & Rieping, W. (2005) *Phys. Rev. E* **72**, 031912.
5. Cox, R. T. (1961) *The Algebra of Probable Inference*. (John Hopkins University Press).
6. Jaynes, E. T. (2003) *Probability Theory: The Logic of Science*. (Cambridge University Press, Cambridge UK).
7. Solomon, I. (1955) *Phys. Rev.* **99**, 559–565.
8. Lipari, G & Szabo, A. (1982) *J. Am. Chem. Soc.* **104**, 4546–4558.
9. Macura, S & Ernst, R. R. (1980) *Molecular Physics* **41**, 95–117.
10. Rieping, W, Habeck, M, & Nilges, M. (2005) *J. Am. Chem. Soc.* **27**, 16026–16027.
11. Brünger, A. T. (1992) *Nature* **355**, 472–474.
12. Brünger, A. T, Clore, G. M, Gronenborn, A. M, Saffrich, R, & Nilges, M. (1993) *Science* **261**, 328–331.
13. Chen, M. H, Shao, Q. M, & Ibrahim, J. G. (2002) *Monte Carlo Methods in Bayesian Computation*. (Springer Verlag, Inc., New York).
14. Alder, B & Wainwright, T. (1959) *J. Chem. Phys.* **31**, 459–466.
15. Metropolis, N, Rosenbluth, M, Rosenbluth, A, Teller, A, & Teller, E. (1957) *J. Chem. Phys.* **21**, 1087–1092.
16. Geman, S & Geman, D. (1984) *IEEE Trans. PAMI* **6**, 721–741.
17. Duane, S, Kennedy, A. D, Pendleton, B, & Roweth, D. (1987) *Phys. Lett. B* **195**, 216–222.
18. Swendsen, R. H & Wang, J.-S. (1986) *Phys. Rev. Lett.* **57**, 2607–2609.
19. Habeck, M, Nilges, M, & Rieping, W. (2005) *Phys. Rev. Lett.* **94**, 0181051–0181054.
20. Tsallis, C. (1988) *J. Stat. Phys.* **52**, 479–487.
21. Hansmann, U. H. E & Okamoto, Y. (1997) *Phys. Rev. E* **56**, 2228–2233.
22. Hansmann, U. H. E. (1997) *Chem. Phys. Lett.* **281**, 140–150.
23. Whitfield, T. W, Bu, L, & Straub, J. E. (2002) *Physica A* **305**, 157–171.
24. Rieping, W, Nilges, M, & Habeck, M. (2008) *Bioinformatics* **24**, 1104–1105.

25. van Rossum, G & de Boer, J. (1991) *Linking a stub generator (AIL) to a prototyping language (Python)*. (EurOpen), pp. 229–247.
26. The World Wide Web Consortium. (1999) Extensible Markup Language (XML) 1.0, W3C recommendation (<http://www.w3.org/TR/REC-xml>).
27. Markley, J. L, Bax, A, Arata, Y, Hilbers, C. W, Kaptein, R, Sykes, B. D, Wright, P. E, & Wüthrich, K. (1998) *J. Mol. Biol.* **280**, 933–952.
28. Fogh, R. H, Ionides, J, Ulrich, E, Boucher, W, Vranken, W, Linge, J. P, Habeck, M, Rieping, W, Bhat, T. N, Westbrook, J, Henrick, K, Gilliland, G, Berman, H, Thornton, J, Nilges, M, Markley, J, & Laue, E. (2002) *Nature Struct. Biol.* **9**, 416–418.
29. Nilges, M. (1995) *J. Mol. Biol.* **245**, 645–660.
30. Rieping, W, Habeck, M, Bardiaux, B, Bernard, A, Malliavin, T, & Nilges, M. (2007) *Bioinformatics* **23**, 381–382.
31. Cornilescu, G, Delaglio, F, & Bax, A. (1999) *J. Biomol. NMR* **13**, 289–302.
32. Bernstein, F. C, Koetzle, T. F, Williams, G. J. B, Meyer Jr., E. F, Brice, M. D, Rodgers, J. R, Kennard, O, Shimanouchi, T, & Tasumi, M. (1977) *J. Mol. Biol.* **112**, 535–542.
33. Habeck, M, Nilges, M, & Rieping, W. (2008) *J. Biomol. NMR* **40**, 135–144.
34. Habeck, M, Rieping, W, & Nilges, M. (2005) *J. Magn. Reson.* **177**, 160–165.
35. Clore, G. M, Bax, A, & Gronenborn, A. M. (1998) *J. Magn. Reson.* **133**, 216–221.
36. Cornilescu, G, Marquardt, J. L, Ottiger, M, & Bax, A. (1998) *J. Am. Chem. Soc.* **120**, 6836–6837.
37. Selenko, P, Sprangers, R, Stier, G, Buehler, D, Fischer, U, & Sattler, M. (2001) *Nature Struct. Biol.* **8**, 27–31.
38. Sprangers, R, Groves, M, Sinning, I, & Sattler, M. (2003) *J. Mol. Biol.* **327**, 507–520.
39. Habeck, M, Rieping, W, & Nilges, M. (2006) *Proc. Natl. Acad. Sci. USA* **103**, 1756–1761.