POOL
User Manual
Version 1.0

Chittaranjan Tripathy and Bruce R. Donald

Contents

1 Introduction **3**

2 License Information **3**

3 Citation Requirements **4**

4 Installation **5**

5 Configurations, Input and Output **6**
5.1 File Organization of RDC-ANALYTIC 6
5.2 Input Format 9
5.3 Output Format 10

6 Examples **10**

7 Utilities **11**

1 Introduction

POOL is a suite of programs for protein loop backbone structure determination from residual dipolar couplings (RDCs) (only two RDCs per residue are required) in one alignment medium. Additional experimental data, e.g., TALOS dihedral restraints and unambiguous backbone NOEs can be used to filter the candidate loop conformations. POOL is a part of RDC-ANALYTIC suite of programs for high-resolution protein backbone fold determination from RDCs. RDC-ANALYTIC/POOL is developed in the lab of Prof. Bruce R. Donald at Duke University.

RDC-ANALYTIC/POOL is free software and can be redistributed and/or modified under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (optionally) any later version. RDC-ANALYTIC is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. For full licensing details, including citation requirements for the software, please refer to Section 2 and Section 3, respectively. This information can also be found in the document `PoolLicense.pdf` enclosed with this package distribution.

POOL is designed to compute high-quality protein loop backbone conformations from RDCs in one alignment medium. The POOL algorithm exploits the interplay between protein backbone kinematics and the global orientational restraints derived from RDC data to naturally discretize the conformation space by polynomial-root solutions, and represents the candidate conformations using a tree. A systematic depth-first search of the conformation tree is used to enumerate all possible loop conformations that are consistent with the data. POOL uses efficient pruning strategies capable of pruning the majority of the conformations that are provably not part of a valid loop, thereby achieving a huge reduction in the search space. POOL requires only two RDCs per residue in one alignment medium, specifically, one RDC type from $\{C^{\alpha}\text{-}H^{\alpha}, C^{\alpha}\text{-}C'\}$ and one RDC type from $\{N\text{-}H^{N}, C'\text{-}N\}$. Therefore, for a loop with n residues, while POOL requires about $2n$ RDCs in total to match with the number of degrees of freedom of the loop, it performs well when only 80-90% of (the $2n$) RDCs are present.

For the remaining degrees of freedom, for which RDC data is not available, POOL employs a finite-resolution uniform sampling of the Ramachandran map is used for that dihedral (degree of freedom). For the cases with less RDC data, POOL may take longer to compute the loop conformations, since dihedral angles for which RDCs are missing, are sampled uniformly, thereby increases the time complexity of the tree search. Also, in this case the solutions can be less accurate. For loops with moderate amount of dynamics, POOL can be used to compute ensembles of loop conformations from RDCs.

This document contains license information, citations required upon using the software, and the details of how to install and use POOL.

2 License Information

The source header below must be included in any modification or extension of the source code of RDC-ANALYTIC.

Source Header

```
This file is part of RDC-ANALYTIC.
```

Contact Info:
    Bruce R. Donald
    Duke University
    Department of Computer Science
    Levine Science Research Center (LSRC)
    Durham, NC 27708-0129
    USA
    email: www.cs.duke.edu/brd/

3 Citation Requirements

Any publications, grant applications, or patents that use RDC-ANALYTIC/POOL must state that RDC-ANALYTIC/POOL was used, with a sentence such as "We used the open-source RDC-ANALYTIC/POOL software [Ref] to compute..."

In addition, you are required to cite our papers in any publications that use this code. The primary citation corresponding to this software is [1]. The papers that can be cited based-on or

related-to this software are listed below.

[1] Chittaranjan Tripathy, Jianyang Zeng, Pei Zhou, and Bruce Randall Donald. Protein loop closure using orientational restraints from NMR data. *Proteins: Structure, Function, and Bioinformatics*, 80(2):433–453, 2012.

[2] Chittaranjan Tripathy, Jianyang Zeng, Pei Zhou, and Bruce Randall Donald. Protein loop closure using orientational restraints from NMR data. In Vineet Bafna and S. Sahinalp, editors, *Proceedings of the 15th Annual International Conference on Research in Computational Molecular Biology (RECOMB), Vancouver, BC Canada*, volume 6577 of *Lecture Notes in Computer Science*, pages 483–498. Springer Berlin / Heidelberg, 2011.

[3] Anna Yershova, Chittaranjan Tripathy, Pei Zhou, and Bruce Randall Donald. Algorithms and Analytic Solutions using Sparse Residual Dipolar Couplings for High-Resolution Automated Protein Backbone Structure Determination by NMR. *The Ninth International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 68:355–372, 2010.

[4] Jianyang Zeng, Jeffrey Boyles, Chittaranjan Tripathy, Lincong Wang, Anthony Yan, Pei Zhou, and Bruce Randall Donald. High-resolution protein structure determination starting with a global fold calculated from exact solutions to the RDC equations. *Journal of Biomolecular NMR*, 45(3):265–281, 2009.

[5] Bruce R. Donald and Jeffrey Martin. Automated NMR Assignment and Protein Structure Determination using Sparse Dipolar Coupling Constraints. *Progress in Nuclear Magnetic Resonance Spectroscopy*, 55(2):101–127, 2009.

[6] Lincong Wang, Ramgopal R. Mettu, and Bruce R. Donald. A Polynomial-Time Algorithm for *De Novo* Protein Backbone Structure Determination from NMR Data. *Journal of Computational Biology*, 13(7):1276–1288, 2006.

[7] Lincong Wang and Bruce Randall Donald. Analysis of a Systematic Search-Based Algorithm for Determining Protein Backbone Structure from a Minimal Number of Residual Dipolar Couplings. In *Proceedings of the 2004 IEEE Computational Systems Bioinformatics Conference (CSB04), Stanford CA*, pages 319–330, 2004.

4 Installation

Since POOL is written in Java, it requires JDK 1.7. Henceforth, it is assumed that JDK 1.7 has already been installed. To install POOL

1. Unpack the tar file in a directory of your choice. Then go to the (sub)directory that contains the directory structure shown in Figure 1.

2. The Java files are in the directory `./src/analytic/`, and the class files (after compilation) will be in the following directory: `./analytic/`. To compile the Java files type the following two commands:

```
javac -d .  -classpath ./javax/vecmath:./Jampack/Jampack:.  ./src/analytic/*.java
javac -d .  -classpath ./javax/vecmath:./Jampack/Jampack:.  ./src/utilities/*.java
```

For convenience, we have provided a shell script with the name `compile.sh`. Executing this script will compile the Java files.

3. This completes installation, and POOL is ready for use.

4. To run the program type the following command:

   ```
   java analytic/Pool <arguments>
   ```

   The `<arguments>` that are supplied to POOL can be found by typing

   ```
   java analytic/Pool -help
   ```

5 Configurations, Input and Output

The inputs to RDC-ANALYTIC are (1) one RDC type from {N-H^N, C'-N } and one RDC type from {C^α-H^α, C^α-C' } measured in one alignment medium; (2) the core, that is, the SSEs of the NMR structures with no loops on it; (3) the alignment tensor computed from the core of the respective NMR structures and the experimental RDCs using singular value decomposition (SVD); and (4) the primary sequence of the loop to instantiate the appropriate residue-specific Ramachandran map. Additional experimental data, e.g., TALOS dihedral restraints and unambiguous backbone NOEs can also be used by POOL.

5.1 File Organization of RDC-ANALYTIC

The directory structure of RDC-ANALYTIC is shown in Figure 1.

Henceforth, we denote the main working directory (`mainDirectory` in Figure 1) for POOL by a period (`.`). The Java source files are located in the folders `./src/analytic/` and `./src/utilities/`. The Java binary class files are in the folders `./analytic/` and `./utilities/`.

To run POOL on input data set to compute loops, we create a directory (call it *master directory*), inside which the input files, the output files and the log files are organized. To illustrate this, let us consider an example of computing the loop VAL17-ILE23 of ubiquitin. Let the master directory be ./EXPERIMENTS/1d3z_17_23. The master directory contains the file dirArch.txt (or a file with any name that contains the directory architecture information and passed as argument to `analytic/Pool`). This file contains the information about all the input, output and log file names. Therefore, the name of the files and directories under the master directory can be changed to (or specified as) any valid directory and file names by the user. In our example we used the following names as supplied in `dirArch.txt`:

```
inputDirectory: input_files_1d3z
// e.g., input directory: ./input_files/

inputDataDirectory: input_data_files
//e.g., Input Data Directory: ./input_files/input_data_files/

inputParameterDirectory: input_parameter_files
//e.g., Input Parameter Directory: ./input_files/input_parameter_files/
```

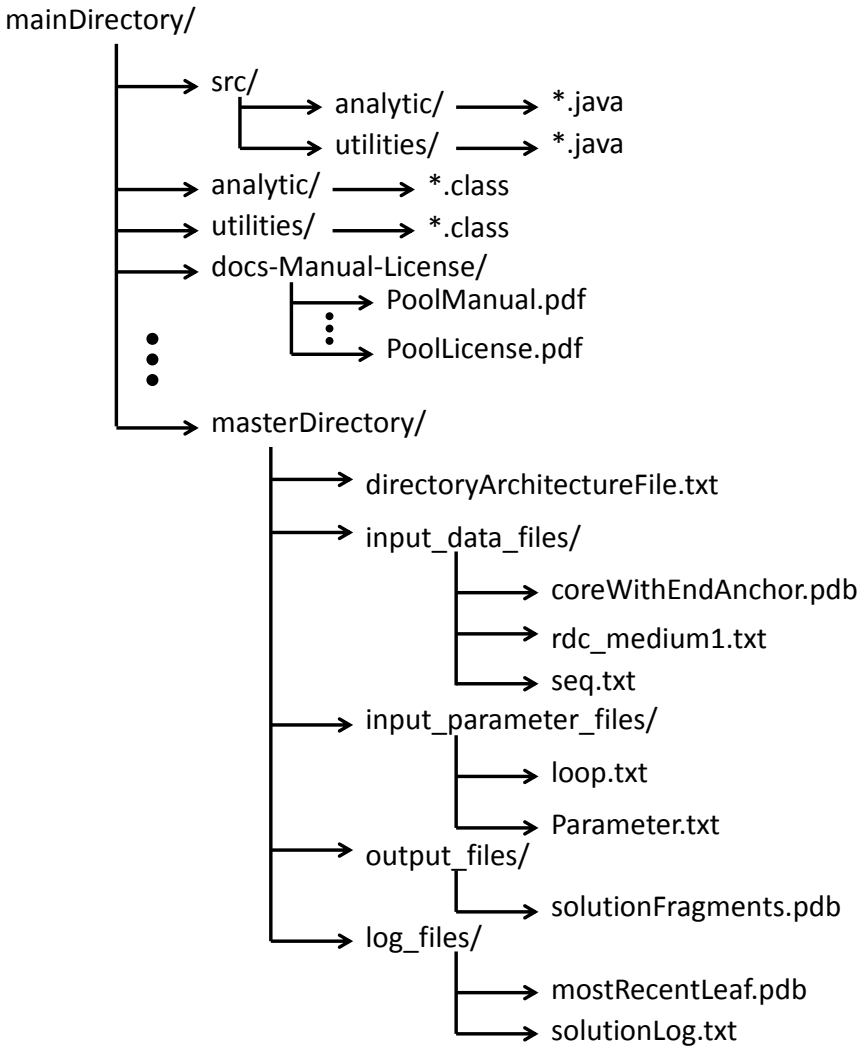

Figure 1: Directory structure of POOL. Each directory and file name in the directory `masterDirectory/`, including the master directory `masterDirectory/`, can be customized (defined or renamed) by the user.

```
outputDirectory: output_files

logDirectory: log_files


// input data directory must contain RDC, NOE information, dihedral angle
// restraints from TALOS+ etc.

//program parameter directory must contain:
//    (1) the set of parameters to be used for the program, and
//    (2) the RDC scaling factor information (i.e., how your RDCs are mutually comparable)

RDC and CSA Files in Medium1: rdc_medium1.txt
RDC and CSA Files in Medium2: null
//RDC and CSA Files in Medium2: rdc_medium2.txt

programParameterFile: parameter.txt

SSeInfoFile: sse.txt

sequenceFile: seq.txt

loopConfigurationFile: loop.txt

dihedralRestraintFile: talos.txt

noeRestraintFile: noe.txt


// **** Warning:  not recommended to change the settings below **** //

bestFragmentPdbFile: solutionFragments.pdb

solutionTreeLogFile: solutionLog.txt

mostRecentFragmentCorrespondingToALeaf: mostRecentLeaf.pdb
```

Note that in the above specifications of file and directory names, the part left to the colon (:) symbol is a tag used by the program, and the part right to the colon symbol is the user-supplied name. For example, above the input directory is specified to be `input_files_1d3z`. It has two sub-directories, namely, `input_data_files` and `input_parameter_files`. `input_parameter_files` contains the input data files and the core of the protein. The directory `input_parameter_files` contains two files: `loop.txt` and `parameter.txt`.

The file `loop.txt` contains the specification of the loop as shown below:

```
// Format is @loop(beginResidueNumber, endResidueNumber,
```

```
            closureDistanceThreshold, gridResolutionForPhiPsi,
            DepthAtWhichStericCheckerTurnsOn, pdbFileContainingGlobalFold,
            phiTypeRdcRmsdThreshold, psiTypeRdcRmsdThreshold, Syy, Szz,
            numberOfSearchTrees);

@loop(17, 23, 1.0, 5.0, 6, 1D3ZBB_CAHA_NHN_POF.pdb, 2.0, 1.0,
      15.457906239256296, 24.715711838306476, 1000)

@computeNow(17, 23)
```

Note that here the tag `@computeNow`, if not specified the loop anchors must be specified as command line arguments while invoking `analytic/Pool`.

The file `parameter.txt` looks like the following:

```
// **RDC Scaling Factors**
// The flag scaleRdcTo can be set to one of the values from the following set:
// {scaled, CA_HA, N_HN}. It sets the values of the prefactors (Dmax) for
// the different types of RDCs measured. If the data has already been scaled,
// then use the flag scaled. If the data are to be scaled wrt. CA_HA then set
// the flag to CA_HA, and if the data are to be scaled wrt. N_HN then set the
// flag to N_HN. We recommend to use scaled RDCs for our program or to scale the
// RDCs wrt. CA_HA.

@scaleRdcTo CA_HA

// Alignment media name for RDCs mest be medium1, medium2, etc.
@rdcMediumName medium1

// Only two RDCs in the same alignment medium required
@typesOfRdcsToBeUsedForAnalyticSolutions CA_HA, N_HN

// Can be used to filter the solutions further
@typeOfRdcsToBeUsedForRefinement CA_HA, N_HN
```

The relative scaling of the RDCs must be specified for correct interpretation of RDC data. Also, the RDCs to be used can be specified here. For this release `@typeOfRdcsToBeUsedForRefinement` must be specified to have the same types as `@typeOfRdcsToBeUsedForRefinement`.

The formats of the input files are described below.

5.2 Input Format

noe.txt

Only a sparse set of unambiguous backbone NOEs can be used by POOL at this time. These NOEs can be obtained from chemical shift analysis of small proteins, or Isoleucine-Leucine-Valine methyl labeling strategies used for larger proteins. XPLOR format for NOEs is used, e.g.,

```
// example NOE
```

```
assign ((resid 5 and name HA)) ((resid 67 and name HN)) 3.555 3.555 0.876 !
```

The program requires that the interacting proton names conform with the latest PDB naming convention. A line comment in the file `noe.txt` starts with `//` as shown above.

rdc_medium1.txt

This file contains the RDCs in XPLOR format, e.g.,

```
assign ( resid 500 and name OO)
       ( resid 500 and name Z)
       ( resid 500 and name X)
       ( resid 500 and name Y)
       ( resid 15 and name N)
       ( resid 15 and name HN) -10.5000 0.0000 0.0000
```

The RDCs are read from this file.

seq.txt

This file specifies the amino acid sequence of the protein in the format

`residueNumber  threeLetterIUPACAminoAcidName`.

TALOS+ Dihedral Restraints in talos.txt

We use TALOS + dihedral restraint format to specify dihedral restraints predicted from the analysis of the chemical shift information.

5.3 Output Format

The output directory, is automatically created when POOL executes, and the ensemble of loop conformations are written into `solutionFragments.pdb`, the file specified in `dirArch.txt` against the tag `bestFragmentPdbFile`.

In addition, a directory is created to keep the log files for the execution of POOL. The log file `solutionLog.txt` and `mostRecentLeaf.pdb` respectively hold the progress of the execution of POOL, and the fragment corresponding to the most recently evaluated leaf node of a solution tree.

6 Examples

This distribution comes with seven examples of how to prepare the input files and run POOL on proteins. For ubiquitin (two examples), experimental NMR data is used. For the rest five proteins, since no NMR data is available, simulated RDCs are used. These loops vary in size (i.e., the number of residues in the loop) from 6 to 12. Specifically, the lengths of the seven loops are 6, 7, 8, 12, 12, 12 and 12.

To invoke POOL for each of these loops, the following commands can be issued (in parallel without any problem) in any order:

```
java analytic/Pool -masterdir EXPERIMENTS/1d3z_17_23/
```

```
java analytic/Pool -masterdir EXPERIMENTS/1d3z_7_12/

java analytic/Pool -masterdir EXPERIMENTS/1ds1/

java analytic/Pool -masterdir EXPERIMENTS/1dqz/

java analytic/Pool -masterdir EXPERIMENTS/1cnv

java analytic/Pool -masterdir EXPERIMENTS/1dysA/

java analytic/Pool -masterdir EXPERIMENTS/1oyc/
```

7 Utilities

For convenience, we provide some basic utilities that comes with RDC-ANALYTIC/POOL. To facilitate faster learning of the use of these utilities, we provide examples in the directory `./EXPERIMENTS/experimentsToTestUtilities/`.

- To align two structures and extract information such as backbone RMSD for ranges of residues (with or without alignment), and compute the magnitudes rotation and translation for alignment, we provide the utility `StructureAligner`.

  To learn more about the command the options type:

  `java utilities/StructureAligner -help`.

  For example, to run the `StructureAligner` utility, you can use the following command (or something similar for your set up):

  ```
  java utilities/StructureAligner
  -pdbfile EXPERIMENTS/experimentsToTestUtilities/1ghh/1GHHModel1.pdb
  -pdbfile EXPERIMENTS/experimentsToTestUtilities/1ghh/1GHHModel2.pdb
  -atomtypes N CA C -ranges 2 12 72 80
  ```

- To simulate RDCs using a structure and an alignment tensor, we provide the utility `RDCSimulator`.

  To learn more about the command the options type:

  `java utilities/RDCSimulator -help`

  For example, to run the `RDCSimulator` utility, you can use the following command (or something similar for your set up):

  To run the RDCSimulator utility you can use the following command (something similar for your set up):

  ```
  java utilities/RDCSimulator
  -masterdir EXPERIMENTS/experimentsToTestUtilities/1oyc/
  -pdbfile EXPERIMENTS/experimentsToTestUtilities/1oyc/1oyc_afh.pdb
  -Szz 8.8465e-04 -Sxx_yy -1.2187e-03 -Sxy -6.5320e-04 -Sxz 6.0936e-05 -Syz 3.4005e-04
  -rdctypes N_HN CA_C
  ```

- To test the fit of RDC data with structural coordinates, we provide the utility `OrientationalRestraintAndStructureFitter`.

  To learn more about the command the options type:

  ```
  java utilities/OrientationalRestraintAndStructureFitter -help
  ```

  For example, to run the `OrientationalRestraintAndStructureFitter` utility, you can use the following command (or something similar for your set up):

  ```
  java utilities/OrientationalRestraintAndStructureFitter
  -masterdir EXPERIMENTS/experimentsToTestUtilities/1ghh
  -pdbfile EXPERIMENTS/experimentsToTestUtilities/1ghh/1GHHModel1And2.pdb
  -rdctypes N_HN CA_C CA_HA
  -printbackcomputedrdcsincsrosettaformat -printbackcomputedrdcsinxplorformat
  -ranges 20 30 -printbackboneinpof -printbackbonedihedrals
  ```