SPQR 1.0
A simulation package for the prediction, refinement and simulation of RNA structures.

Simón Poblete

1 Introduction.

SPQR (SPlit and conQueR) is a coarse-grained representation of RNA [1, 2], which is implemented in the present software package. The energy function can be used to score structures, predict three-dimensional structures from a given sequence, explore the conformational space, refine structures and remove topological artifacts.

2 Features.

As a coarse-grained model, SPQR is defined by its degrees of freedom and its energy function. As shown in Figure 1, each nucleoside is represented by an anisotropic particle, while the phosphate group is a point particle.

The base is a triangle with a virtual site attached which stands for the sugar group. This representation allows the introduction of directional interactions and the possibility of forming stacking, canonical and non-canonical base-pairs and base-phosphate interactions. In addition, through the interactions between a nucleoside and its topologically connected phosphate groups, it can specify the state of the glycosidic bond angle and sugar pucker. The relative weight of the interactions considers the number of hydrogen bonds present in the base-pairs while it has been adjusted by fitting a set of X-ray structures for the rest of interactions.

3 Compiling and installing.

To compile, go to the `src/` directory and run

```
autoconf
./configure
make
make install
```

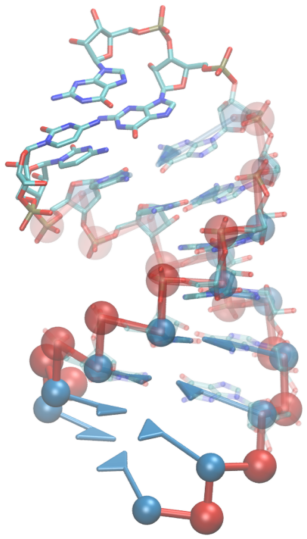

Figure 1: SPQR representation of a RNA duplex.

This will create the binary files in the `bin/` directory and the tabulated interactions file `intrac.btb` in the `interactions/` directory.

In order to run any of these programs one must include the file `params.spqr` in the same directory, plus other files specified in the next section. The following list summarizes the binaries

- `SPQR_MC [ -i <job_index> ] [ -r ]`

  Runs a simple Monte Carlo simulation at constant temperature. Requires the initial condition stored in the `spqr_inits/` directory, which can be created with the scripts described in the Tools section. With the `-r` option, the temperature, time step and random seed are read from the initial condition, in case it is a `.mc` checkpoint file. If not, only the coordinates are read and the rest of the parameters follow their definition from the main parameter file.

- `SPQR_eMC [ -i job_index ]`

  Runs a simple Monte Carlo simulation with the option of including a harmonic restraint on the ERMSD with respect to a target structure. It requires the same files as `SPQR_MC` and the file `ermsd_frags.lst`, which defines the groups over which the ERMSD steering will be applied, their harmonic spring constant and the reference fragments.

- `SPQR_SA [ -i job_index ] [ -r ]`

Runs a simulated annealing. It requires additional parameters in the `params.pms` file, which are described in the next section.

- `SPQR_wMC [ -i job_index ]`

  Runs a Monte Carlo simulation with capped interactions. That is, whenever there is a clash or broken bond, the interactions do not crash but instead push the system towards a more realistic configuration.

- `SPQR_ENERG <pdbfile> <args>`

  A tool for calculating the energy and its contributions of a given structure. In addition, one can perform annotations and detect possible clashes.

4 Input files.

4.1 params.spqr

The most important file is `params.spqr` , which contains the parameters for any type of run. The list is below

- `TEMPERATURE` : temperature of the system in the CG units.
- `PDB_OUTPUT` : A flag (0 or 1) which indicates whether the output trajectory will also be saved in `pdb` format. Use it carefully, since the trajectory files tend to be quite large.
- `RG_COUPL` : Two arguments: the restitution constant and the target radius of gyration. The energy term is proportional to the square of the difference of the radius of gyration and the target value. When the restitution constant is zero, the energy term is not calculated.
- `MC_PH_XYZ` : Monte Carlo step for the translation of phosphate particles.
- `MC_NT_XYZ` : Monte Carlo step for the translation of nucleoside particles.
- `MC_STEPS` : Number of Monte Carlo trials for the simulation. Each of these steps consists of a trial move of each of the nucleotides of the system.
- `MC_TRAJ_STEPS` : Number of steps between each configuration saved in trajectory files.
- `MC_CHKP_STEPS` : Number of steps between checkpoint saving.
- `RANDOM_SEED` : Integer larger than zero. Still, two runs with the same random seed but different `mpi_id` or process id will have internally a different seed and therefore, produce different results.
- `MC_NT_ANGLE` : Monte Carlo parameters for the maximum rotations of the nucleoside, around the base and the sugar group.

- `ENERGS_PATH` : Path leading to the file containing the interaction data, including its name. By default, the name is `intrac.btb`, and it is located in the `interactions/` directory when the package is installed . This file is a binary which contains the information of the whole tables and interaction parameters. Such files can be found in the same directory by default, `tables_src/` . In case a modification to the tables is added, one can generate the corresponding binary file again with the tools included in the same directory.

The order of the input parameters is irrelevant, and the lines can be commented by introducing the character `#` at the beginning of a line.

Additionally, there is a number of parameters needed for running a simulated annealing. All of them have the prefix `SA_` . Considering that the temperature starts with a value T_M and that $T_i = \lambda^i T_M$, where $0 < \lambda < 1$ and $i = 0, 1, 2, ...$, the annealing consists of a set of simulations (using the parameters aforementioned) with temperature T_i, until a minimum value T_0 is reached and the temperature is rounded to zero. In addition, if a maximum number of steps N_m is reached, the simulation will also stop.

- `SA_TINI` : Initial, or maximum temperature T_M, from where the annealing starts. Usually around 20.
- `SA_TMIN` : Minimum temperature, or where the annealing will round it to zero to stop at the next simulation.
- `SA_TFAC` : The value of λ; the prefactor with decreases the temperature.
- `SA_STEP` : Initial step for the run. This must be set to zero if the annealing is not starting from a checkpoint.
- `SA_NT` : The maximum number of annealing steps, N_m.
- `SA_PREENERG` : The energy of the previous annealing step. It must be zero if the annealing is not starting from a checkpoint.
- `SA_SFAC` : A prefactor which decreases the size of the Monte Carlo steps. This follows the procedure of Snow et al. [3], for better convergence.
- `SA_RTIMES` : The number of times the reduction of the Monte Carlo steps has been applied. It must be zero if the simulation does not start from a checkpoint.

All the parameters which are zero unless the simulation starts from a checkpoint can be set manually to specific values. Nevertheless, the same checkpoint contains the information for these values and their setup is automatic. In future versions, their direct manipulation will be removed.

4.2 spqr_inits

The directory `spqr_inits/` contains the initial conditions. The files can be in `.pdb` or `.mc` binary format, which saves the configurations and more information in full precision, allowing to evaluate energies and restarting simulations without complications. Several tools are provided for converting between these formats, modify them and generate them from a sequence or other pdb (see the Tools section). The names of the initial conditions must be `init.p<XX>.<format>` . `XX` corresponds to the index of the simulation. It can also be that only the file `init.<format>` is provided, starting all the simulations from them but with different random seeds.

The SPQR `pdb` format contains some particular features. Each nucleotide is composed of five particles, with atom names `BASE` , `XVEC` , `YVEC` , `SUGR` and `PHOS` . The sugar position is only contained for human-reading purposes, since the SPQR binaries rewrite it anyway. Apart from the atom, residue and chain indexes, it contains the coordinates as any pdb file. However, the `BASE` atoms can contain additional information for a simulation. Starting from the column 56, four parameters `G` , `P` , `L` and `F` can be specified.

- `G` : Corresponds to the glycosidic bond angle state. It can be `A` , `H` or `S` for anti, high-anti and syn. The syn state is not allowed in pyrimidines.
- `P` : Corresponds to the sugar pucker. It can be 3 or 2, for C3' endo and C2' endo states.
- `L` : Specifies if the glycosidic bond angle and sugar pucker states of a nucleotide will be fixed or not during a simulation. It is `A` if both parameters can change, `P` if only the pucker can, `G` if only the glycosidic bond angle is allowed to change, and `N` if both are fixed.
- `F` : Specifies if the position of a nucleotide, or part of it, will be allowed to move during the simulation. It is `A` if the whole nucleotide is movable, `P` if only the phosphate can move, `B` if the nucleoside can move and `N` if the whole nucleotide is frozen.

If any of the previous parameters is wrongly initialized, of absent, **the state of the nucleotide will be set by default to** `A3N3` , which is a nucleotide flexible and free to move, but with glycosidic bond angle in anti conformation and sugar pucker C3' endo, both fixed.

4.3 Interactions

As mentioned before, the `intrac.btb` binary file must be located somewhere, such that the `params.spqr` file knows its path. The interaction tables are written in binary format, which are available in the directory `tables_src`.

5 Output files.

After a simulation is performed, several files are created. Inside the `configs/` directory, the trajectories and the checkpoints are stored. Both of them are in binary format, which is accessible by using the analysis template in the tools directory. The trajectories are stored in the file `confs.pXX.mc` , where `XX` corresponds to the index of the simulation running. The checkpoints, on the other hand, are named `chk.YY.pXX.mc` , with `YY` denoting the Monte Carlo step. These files can also be transformed to pdb format easily with the tool `extract_traj.c` , and modified with similar methods found in the same directory (See the Tools section). The final configuration is also written in pdb format in the local directory under the name `final.pXX.pdb` . The checkpoints contain all the information of the conformation with double precision, and the parameters for the simulated annealing in case this is interrupted.

6 Annotation, energies and secondary structure.

The `SPQR_ENERG` binary allows to calculate the energy of a configuration. Regardless its format (`mc` or `pdb`), it has several options to run as

```
SPQR_ENERG  <filename>  -option
```

The option can be :

- `a` : annotation.
- `s` : secondary structure.
- `t` : total energy (with a detail of its contributions).
- `f` : the total energy.
- `b` : the backbone energy.
- `B` : the backbone energy plus the stacking energy of contiguous nucleotides.
- `n` : non-bonded stacking energy and finally.
- `w` : the base-pairing energy.
- `r` : the radius of gyration.

7 Tools.

The directory tools contains several binaries with their sources for the creation and manipulation of structures and conformation files. The files are listed here:

- `MODIF_MC` : It allows to change the state of a glycosidic bond angle or sugar pucker, to make it fixed or allowed to change during a simulation. Its source is `modif_mc.c` .

- SPQR_ASSEMBLE.py : A python script which allows to generate a simple SPQR pdb file from a sequence. It must be used as

  ```
  python SPQR\_ASSEMBLE.py -s "<seq>"
  [ -t "<secondary\_structure>" ]
  [ -c "<x y z x y z ... >" ] [ -o <output> ] [ -p ]
  ```

  where the options -t and -c are optional. -s specifies the sequence. -t allows to introduce secondary structure constraints (in quotation marks, like this "(((....)))" and -c , the centers of the strands that will be created. The output files are: <output>.pdb (by default, init.pdb) for later initial conditions, and ermsd_frags_<output>.lst which is required for enforcing the contacts given in the arguments.

  The init.pdb file must be in the directory spqr_inits when running the simulation, as shown in the tutorials/single_run directory. The ermsd_frags.lst must be contained in the same directory of the simulation running, which has to be run with the SPQR_eMC or SPQR_eSA binaries.

  Also, note that the ermsd_frags.lst file contains some REMARK lines at the beginning. The first corresponds to the parameters to be used in the simulation: the first line containts the number of groups where the contacts are going to be enforced followed by the cutoff of the ERMSD. The following lines are as many as there are groups to be enforced, and contain the harmonic constant of the ERMSD harmonic potential followed by the residue indexes of the nucleotides belonging to each group. The ATOM coordinates which complete the file are for internal use and constitute the templates used for enforcing the contacts.

  The -p option is for splitting the secondary structure constraint only as base pairs, without taking stacking into consideration. Quite useful when building decoys.

- pdb2spqr.py : Transforms an all-atom pdb file into a SPQR pdb file. It classifies automatically the glycosidic bond angles and sugar puckers of each nucleotide.

- spqr2pc3.py : It converts a SPQR pdb file into a pdb with the positions of the P, C2, C4 and C6 atoms, ready to be used with the Gromacs and Plumed packages for ERMSD pulling.

- SPQR_BBACKMAP.py : A trivial (brute-force) and direct way of backmapping SPQR files into all-atom representations. It simply mounts an atomistic template nucleotide on top of each SPQR base. It uses specific templates depending on the glycosidic and pucker states. Further energy minimization can follow this step for later use in MD simulations. Use like this:

  ```
  python SPQR_BBACKMAP.py [ -i <input> ] [ -o <output> ]
  ```

- SPQR_REFINE : A bash script for minimizing the energy of a pdb structure. It requires the definition (in its 6th line) of the path where SPQR is installed, which is done automatically at the moment of installation. Its syntax is

  ```
  SPQR_REFINE -i <PDBFILE>
  [ -o <OUTPUTNAME> -t <SECONDARYSTRUCTURE> ]
  ```

  This script runs internally several simulations which minimize and remove artifacts in order to provide a suitable structure for simulations. The script will create and store the results in the folder `refSPQR`. Optionally, the name can be specified with the `-o <OUTPUTNAME>` option. The folder will contain a pdb file in SPQR format (`refSPQR<PDBFILE>.pdb`), an all-atom reconstruction (`at_refSPQR<PDBFILE>.pdb`) and a .mc file (`refSPQR<PDBFILE>.mc`), which can be used for further SPQR simulations with full precision (recommended). The all-atom reconstruction is suitable for MD simulations as well. The minimization of energy is a good starting point for evaluating the SPQR energy, which is contained in the .ene file in the working directory. In addition, it removes clashes and fixes broken bonds, which might be of help when dealing with structures generated by fragment assembly methods, such as the ones generated by Ernwin [4]. If the secondary structure elements such as hairpins, stems and internal loops are entangled, these artifacts can also be removed automatically [5]. In this case, the secondary structure must be provided in a fasta file containing three lines : a comment line, the sequence and the secondary structure in Vienna format, without pseudoknots. The option `-t <SECONDARYSTRUCTURE>` takes care of this. Other options can be found in the user's guide and the help message displayed with the -h flag.

- `at.tcl` : A `tcl` script which allows to visualize the SPQR representation of a `pdb` in `vmd`. It must be loaded from `vmd` with the command `source at.tcl` once a molecule is already loaded.

8 An example.

Enforce secondary structure contacts in a GCAA tetraloop. Create the files with

```
python SPQR_ASSEMBLE.py -s GGGCGCAAGCCC -t "((((....))))"
```

Move the `init.pdb` file into the `tutorials/single_run/spqr_inits` directory. Then, move `ermsd_frags.lst` and `SPQR_MC` to the `tutorials/single_run` directory. Finally, run the simulation with `./SPQR_MC` .

Additionally, the simple annealing of the fragment can be run. By running the simulation with the `SPQR_SA` binary, and deleting the `ermsd_frags.lst` file from the directory, the final structure will be a candidate for the native structure. The procedure can be repeated with different random seeds with the option `SPQR_SA -i <random_seed>` .

References

[1] S. Poblete, S. Bottaro and G. Bussi, *A nucleobase-centered coarse-grained representation for structure prediction of RNA motifs*, Nucleic Acids Res. 46, 1674-1683 (2018).

[2] S. Poblete, S. Bottaro and G. Bussi, *Effects and limitations of a nucleobase-driven backmapping procedure for nucleic acids using steered molecular dynamics*, Biochem. Biophys. Res. Comm. 498, 352-358 (2018).

[3] M. E. Snow, *Powerful simulated-annealing algorithm locates global minimum of protein-folding potentials from multiple starting conformations.*, J. Comput. Chem. 13, 579-584 (1991).

[4] P. Kerpedjiev, C. Höner zu Siederdissen and I. L. Hofacker, *Predicting RNA 3D structure using a coarse-grain helix-centered model*. RNA 21(6):1110-1121, (2015)

[5] B. Thiel, G. Bussi, S. Poblete and I. L. Hofacker, https://www.biorxiv.org/content/10.1101/2022.07.02.498583v1