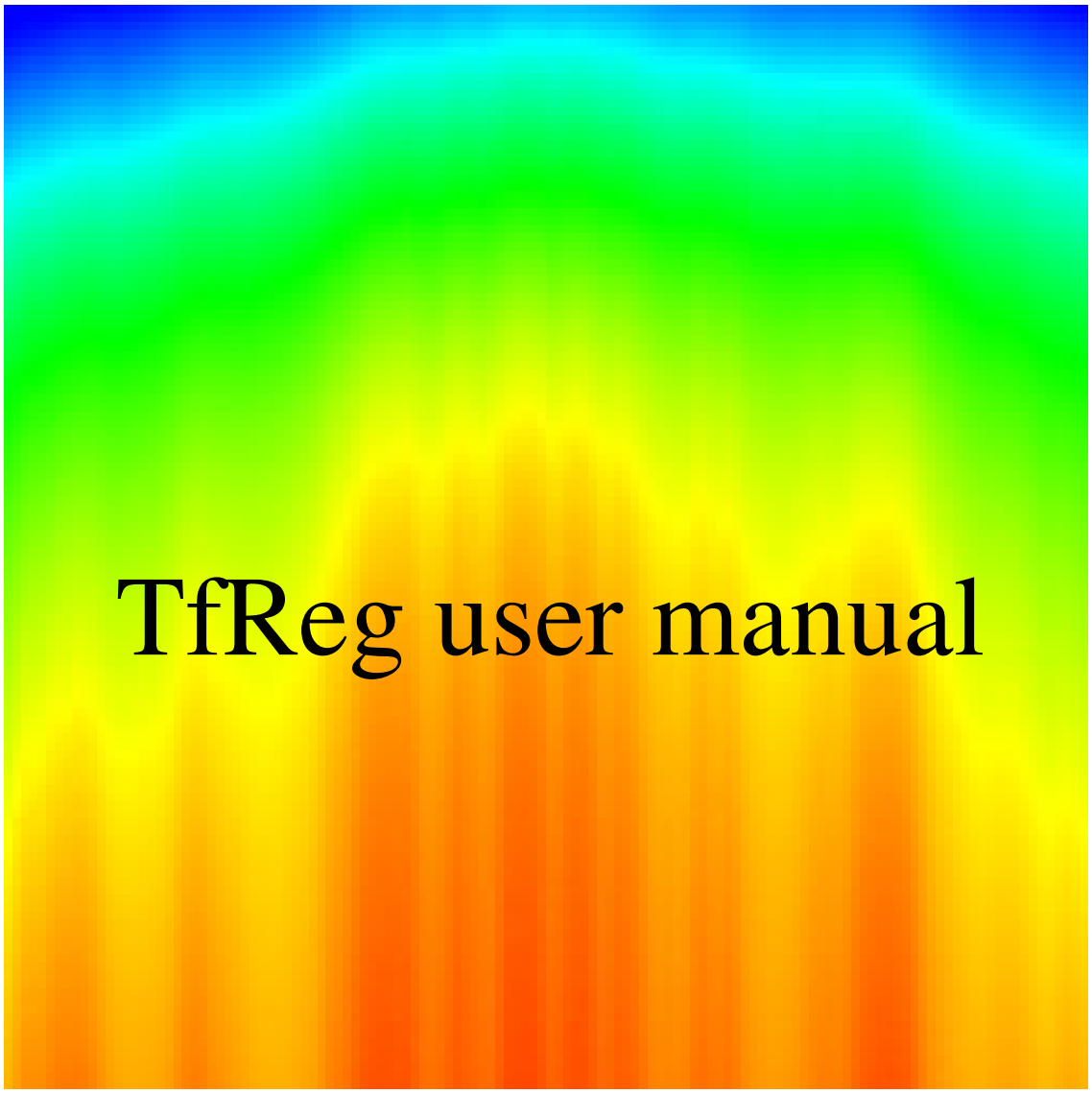

TfReg user manual

TfReg Version 7.0 user manual

Gerald Weber gweberbh@gmail.com
Departamento de Física, Universidade Federal de Minas Gerais, Brazil

September 13, 2022

Contents

1 Introduction

TfReg implements the calculation of Peyrard-Bishop [1] style Hamiltonians to obtain some physical properties of DNA and RNA duplexes. The method uses the transfer matrix technique for the calculation of the classical partition function. Also, TfReg calculates the regression of experimental versus predicted melting temperatures using the equivalent melting index [2].

What will this software do for you? Given a set of experimental melting temperatures and a set of model parameters you will be able to calculate the regression parameters which will allow you predict melting temperatures of any DNA or RNA sequences. Alternatively, you may use one of the calculated regression sets which are provided and start calculating melting temperatures right away. You may chose between four different "flavours" of Hamiltonians if you wish to investigate the effect of different model parameters. If you do have basic programming skills in C++ you should be able to add new types of model Hamiltonians, as long as they fit within the 1D framework of the original Peyrard-Bishop model [1].

Evidently, this is work in progress. I hope to add further programs in the near future as well as increase the number of parameters for other types of oligunucleotides.

I would find it truly helpful indeed if you would let me know if this software is of any use to you. Showing a list of interested users to funding agencies often helps to secure the necessary resources to keeping such projects running. So, please, if you find this software useful let me know and if you use it for your scientific work please cite the appropriate papers which are listed at the end of this manual.

I wish you all the best in using TfReg

Gerald

Belo Horizonte, September 13, 2022

2 Changes

Version 7.1, September 2022

-
- New parameters and data files for Mg^{2+} in DNA [3]

Version 7.0, February 2022

- New appendix A for system messages.
- `libboost_iostreams` is now required for compilation.
- New models `-model=pcla`, `-model=pclj` and `-model=trmf`.
- New option `-printusedpar` (4.1.21).
- New parameters for metal mediated DNA [4]

Version 6.0, April 2021

- New parameters and data files for LNA [5]
- The first line of the parameter file can now be used to add comments, see section 6.
- Fixed bug that make the program hang when a parameter line was wider than 1024 characters. It can now have any length.
- Missing arguments to options that require it now generates an error and terminates the program.
- Fixing bug for generic parameters for hybrid context dependent such as `TT^j_CG` for the case of parameter fallback.
- Added missing files for `dna_tpb_*.par`.

Version 5.3, November 2020

- New example scripts.
- New parameters for RNA with varying sodium concentrations [6].
- New feature for context parameters to fall back to non-context, see section 6.4 for details.
- New `-res=sprediction` which runs predictions using less memory.
- Updating file `rna-dna-pb-1000.par` with the correct parameters published in Ref. [7].

Version 5.2, July 2020

- New parameters and data files for DNA+LNA/DNA (DLD) and DNA+LNA/RNA (DLR) duplexes [8]
- New 3D model, see section 5.6 [9]
- For options with a limited set of values, the program now stops if unknown values are given. For example if you misspell a model name.

Version 5.1, March 2020

- New parameters and data files for mismatches [10]

Version 5, February 2020

- New parameters and data files
  - Added terminal related parameters, files of type `dna_tpb_*.par` [11].
  - Adapted sequence interpretation for TNA with new parameters for TNA-DNA hybrids [12].
- New features
  - Multiple files in `-data` (4.1.4)
  - New regression option `-pm=-2` (4.1.20), see section 4.1.20
- Corrections
  - Fixing bug that was using always `-pm=2` (4.1.20) for predicting single sequences.
  - Fixing bug that was giving wrong trimer counts with `-res=nncheck` (4.1.6).

Version 4.0, September 2018

- For DNA/RNA hybrids: new melting temperature data file `rna-dna-1000.dat` from references 13–15 and parameters `rna-dna-pb-1000.par` [7].
- Now prints out CPU time after finishing calculations.
- New optimization in matrix calculations to reduce CPU time.
- Fixed error that would generate an infinite loop if matrix elements were either infinite or NaN.
- Fixed error in file `TestModel.h` that would prevent users to implement their own model.

Version 3.2, July 2017

- New parameters for Cy3 and Cy5 terminally attached to DNA [16] with new files:

  ```
  dna_pb_cy3.par dna_pb_cy5.par
  dna_pb_cy3.reg dna_pb_cy5.reg
  moreira15cy3.dat moreira15cy5.dat
  ```

- A warning about missing parameters was reclassified as an error and now the program will terminate. Previsously the pogram would continue with a value of zero for those missing parameters.
- Fixed bug for `-res=averagey` (4.1.6) which, under certain very specific circumstances, would erroneously apply periodic boundary conditions as if `-pbc=1` (4.1.12) had been set. Symptoms were missing end-fraying, for example for DNA terminating in AT no end-fraying would be observed.
- documentation updates.
- removed obsolete files:

  ```
  JobControl.h JobControl.cpp
  ```

Version 3.1, April 2017

- New parameters for single type I bulges in RNA [17] with new files:

  ```
  rna_pb_bulge_group1.par
  rna_bulge_group1_adenosine_adj200.dat rna_bulge_group1_cytosine_adj200.dat
  rna_bulge_group1_guanosine_adj200.dat rna_bulge_group1_uridine_adj200.dat
  rna_pb_bulge_group1_adenosine.reg rna_pb_bulge_group1_cytosine.reg
  rna_pb_bulge_group1_guanosine.reg rna_pb_bulge_group1_uridine.reg
  ```

Version 3.0, January 2016

- Fixed important bug concerning matrix multiplications which affects the average opening profiles, especially for symmetric sequences. This may also affect, although very slightly, the melting index and predicted temperatures. Differences in both cases are around 0.01% or less compared to previous versions of tfreg.
- Fixed bug from version 2.0 which was generating empty files for option `-res=prediction` (4.1.6) with single sequences.
- Added new base pair representation of type `XY^z` which introduces context-dependent base pairs. See section 6.2.1.
- Added optimized parameters for RNA GU [18].
- Linking to the library `libboost_regex` is now required.
- Code clean up, removed `gbc_exp` from C++ name space.
- Adding version identification, see first printed line when running tfreg.
- You can now add easily your own model potentials, see section 5.10.
- Example scripts were revised for consistency, in particular they no longer accept the data folder as argument. Alternative data folders should be provided by setting the `PREFIX` command line variable.
- Add new option `-mar` (4.1.18).

Version 2.0, December 2014

- Corrected harmless bug which was causing the last line of a parameter to be read twice.
- Added a new model `-model=mes` (4.1.7).
- Added new option `-res=nncheck` (4.1.6), which shows how the sequences are analysed in terms of base pairs and nearest-neighbours by TfReg.
- Added optimized parameters for deoxyinosine [19], see section 9.5.
- `.ver` files now have an additional column providing ΔT_{RMS}.

Version 1.2, November 2013

- Corrected bug which would fail to predict temperatures with `-pm=-1` (4.1.20) from reg files also generated with `-pm=-1` (4.1.20).
- Removed generation of TEX files.
- Adding new option `-dict` (4.1.9) which allows you to add new characters to represent nucleotides.

Version 1.1, February 2013

- We corrected a software bug which recalculated unecessarily the Gauss-Lengendre quadrature weights. As a result TfReg runs considerably faster.
- We added a script for adjusting the regression coefficients to other salt concentrations. See section 9.3.

Version 1.0, November 2012

First release of TfReg.

3 Installation

3.1 Download and direct install

You can find the compiled binary package, for many Linux distributions, as well as source code, at these locations:

- http://bioinf.fisica.ufmg.br/software
- https://download.opensuse.org/repositories/home:/drgweber/

browse to the desired version of the software and then download the appropriate package for your Linux distribution. If your distribution is not covered then download the source code and compile manually, see below for instructions.

3.2 What will be installed?

Typically, there will be at least a binary executable file

`/usr/bin/tfreg`

and further files, such as model parameter files, pre-calculated regression parameters as located in

`/usr/share/TfReg/data or /usr/share/tfreg/data`

note that these folder may vary depending on your Linux distribution. Example scripts which help to understand how to run TfReg are to be found in

`/usr/share/TfReg/example or /usr/share/tfreg/example`

The documentation (which you are reading right now) should be located at

`/usr/share/doc/packages/TfReg/tfreg-user-manual.pdf or`
`/usr/share/doc/packages/tfreg/tfreg-user-manual.pdf`

3.3 Specific instructions

3.3.1 OpenSUSE

Installing via repository Using the graphical interface Yast2 or the command line zypper add the following repository URL

`https://download.opensuse.org/repositories/home:/drgweber/openSUSE_Leap_15.1/`

then search for the package TfReg and select install. If you have an older OpenSUSE then change the last number to your installed version accordingly. The installation via repository has the advantage that you may simply update for future versions instead of repeating the whole installation procedure.

Command line download/install Download the appropriate package for your system from

https://download.opensuse.org/repositories/home:/drgweber/openSUSE_Leap_15.1/

for example if your system is 64bits, you may download the package

`http://download.opensuse.org/repositories/home:/drgweber/openSUSE_Leap_15.1/x86_64/TfReg-7.0-1.21.1.x86_64.rpm`

note: version numbers may vary from this example. Then install

`zypper install TfReg-7.0-1.21.1.x86_64.rpm`

3.4 Compiling the source files

Please read this section if you are unable to find the packages for your specific Linux distribution or if you are interested in modifying the source code.

Download the source package from or from my personal webpage

https://bioinf.fisica.ufmg.br/software/tfreg-7.0

Typically the package is called something like `tfreg-7.0.tar.bz2`After unpacking the `tar` package

```
tar -xvjf tfreg-7.0.tar.bz2
```

change into the unpacked folder

```
cd TfReg-7.0
```

if all necessary packages are available you should try to compile using the `make` command

```
make
```

If the compilation is successful, you should see something like

```
g++ -O3 -o tfreg -Isrc src/Options.cpp src/JobControl.cpp src/Nucleotide.cpp src/tfreg.cpp -lz
-lgsl -llapack -lgslcblas -lm -lboost_filesystem  -lboost_system -lboost_regex
```

and nothing else, that is it! This generates the binary file `tfreg`, which you may copy to your main installation at `/usr/bin` (you will need root permission) or into your local folder `/home/user/bin` (replace `user` with you actual user name).

3.4.1 Libraries needed for compiling the source code

This software was developed and tested under OpenSUSE Linux 15.1 and depends on some libraries to function properly:

1. `libboost_filesystem libboost_system libboost_regex libboost_iostreams` http://www.boost.org
2. `gsl` http://www.gnu.org/software/gsl/
3. `lapack gslcblas` http://www.netlib.org/lapack/
4. `gcc-fortran` http://gcc.gnu.org/

which means that you will need at *least* these specialized packages in addition to the usual gcc and g++ compiler. If you already installed pre-compiled binaries then probably you will already have all the required packages installed in your Linux system. The Intel®[1] C++ compiler `icc` was also tested and works well.

3.5 If things go wrong

Most problems will come from missing library packages or from erroneous usage of your system. It is not possible for me to cover everything that may go wrong, so please feel free to contact me. Please include a detailed description of error messages, which system you are using and a step by step description of what you tried to do. Please understand that I will need as much information as possible. I can do nothing with messages saying simply "TfReg is not working on Ubuntu".

3.5.1 License

This software is published under the GNU General Public License version 3 (GPLv3), the complete text of this license can be found in the documentation folder. If you wish to use this software under a different license or wish to make changes to the software without distributing it under the GPLv3 please contact me so that we can arrange for a specific license.

[1] All trademark mentioned in this manual displayed with sign ® are registered by its respective companies.

4 Usage

TfReg take all its program option from the command line, therefore you should invoke the binary `tfreg` with some of the arguments which are described below. Example shell scripts are provided and I recommend you study them as they are the best source to illustrate how to use TfReg. The following section details every available program option for TfReg.

4.1 Program options and configurations

4.1.1 -o=<basename>

Specifies the output basename, that is, all files which are generated start with `basename`

4.1.2 -reg=<filename>

specifies the input regression file name. Use this option together with `-res=prediction` or `-res=sprediction`, in this case the regression coefficients are taken from this file. See section 4.1.20 regarding the prediction and regression options.

4.1.3 -par=<filename or comma-separated list of filenames>

Specifies the input parameter file name. This can be a list of files `-par=file1.par,file2.par,file3.par`. In case of multiple specifications of the same model parameter, the last one supersedes previous parameters. For example if `file1.par` has the parameter `AT:morse.D 0.05` and `file3.par` has `AT:morse.D 0.03`, the final value for `AT:morse.D` will be 0.03.

4.1.4 -data=<filename or comma-separated list of filenames>

specifies the input files containing nucleotide sequences and melting temperatures. Multiple files in a comma-separated list are accepted since version 5.0. See section 7 for more information regarding preparing those files.

Note on memory limits: TfReg was not created with very large datasets in mind, and all sequences are loaded into memory. A safe limit is of the order of 5000 sequences which will require about 1–2 GB of memory to run. If you are only interested in running predictions and are running out of memory, see `-res=spredictions` (4.1.6).

4.1.5 -matrix=<new directory>

If specified this will create a directory where the calculated matrices will be located. Once calculated these matrices will be reloaded the next time the program is run again. This speeds up the calculations, however if you do change parameters or the model from one run to the next you should not use this option as you will load matrices which are incorrect. Also, the Gauss-Legendre quadrature weights are saved into this directory if set, these weights are totally independent of any model parameters.

4.1.6 -res=<resulttype>

which type of result we wish to obtain.

(default) `-res=regression` given a set of parameters and a set o melting temperatures, calculates the regression parameters (stored in a file with extension `.reg`) and also calculated the prediction of temperatures.

`-res=prediction` predicts temperatures for a given set of parameters, requires a pre-calculated regression file passed through the `-reg` (4.1.2) option. Some calculated regression files can be found in the `data` folder, see 10.1. For very large sequence files see next option.

`-res=spredcition` same as `-res=prediction` but reads the sequences from file without loading into memory. Use this for files with many sequences if you are running out of memory.

`-res=nncheck` only checks consistency of the nearest-neighbour decomposition of the nucleotide sequences. This will display the sequence analysis and exit immediately without doing any calculations.

`-res=averagey` calculates the average opening $\langle y \rangle$, results are given in Ångstrom. For a single sequence use with `-seq` (4.1.22), and optionally with `-cseq` (4.1.23), and the result will be a file with $\langle y \rangle$ arranged column-wise. For larger number of sequences arrange all sequences in one or more files with `-data` (4.1.4). Note that for short sequences the temperature needs to be unrealistically low, see example 9.6.

`-res=zyfy` calculates the configurational part of the partition function Z_y (see for instance Eq. 28 of Ref. [20]) and its total Helmholtz free energy $-kT \ln Z_y$.

4.1.7 `-model=<model acronym>`

Selects the Peyrard-Bishop model which should be used. Each model requires specific parameters which should be passed via the `-par=<filename>` option.

(default) `-model=pb` original Peyrard-Bishop model [1], see section 5.1

`-model=dpb` the anharmonic variant [21], see section 5.2

`-model=hms` PB model with added solvent potential [22], see section 5.3

`-model=jb` finite enthalpy model [23–27], see section 5.4

`-model=mes` Morse-exact stacking model [28], see section 5.5

`-model=pb3DA` approximated 3D Peyrard-Bishop model [9], see section 5.6

`-model=pcla` entropic y^3 barrier from [29], see section 5.7

`-model=pclj` hump potential from [30], see section 5.8

`-model=trmf` Gaussian barrier from [31], see section 5.9

`-model=test` Reserved for *your* test models, please see section 5.10 for instructions. Please do not use without first implementing the changes described in 5.10 and recompiling the source code.

4.1.8 `-duplextype=<DNA or RNA>`

Selects the type of duplex we should expect, this is important for selecting the base pair complementarity. Note that IUPAC codes cannot be used since we need to know which nucleotide parameters to use, that is, we cannot use N for example since we would not know which potentials to use. If yo need to define other types of nucleotides see option `-dict` (4.1.9).

(default) `-duplextype=DNA` , will expect A, C, G and T nucleotides and will consider A complementary to T and C complementary to G.

`-duplextype=RNA` , same as for DNA but considers A and U as complementaries, and removes T from the dictionary. [32].

4.1.9 -dict=<list of nucleotides>

When using non-canonical nucleotides, for example inosine, you must tell TfReg which characters to use in addition to its usual dictionary of A, C, G, T or U. You should also tell TfReg if there is a complementary pair to this new nucleotide. If there is none, simply repeat the character. In the following example we are adding the letter I for inosine

`-dict=I:I`

here we are saying: consider I and its complementary I as new letters. If you wish to add two or more new letters simply make a comma-separated list, but do not add blank spaces.

4.1.10 -expand=<nearest neighbours>

Selects which nearest neighbours to expand.

(default) `-expand=CG_CG` for DNA or RNA

`-expand=dCrG_dCrG` for DNA-RNA hybrids [7].

`-expand=dCtG_dCtG` for DNA-TNA hybrids [12].

4.1.11 -cutoff=<integer number>

This controls the truncation P in Eq. (22) of Ref. 20, that is, you will be using only the first P eigenvalues of the diagonalized matrix. All subsequent matrix multiplications will be $P \times P$, using a small P will make TfReg run considerably faster.

(default) `-cutoff=0` no cutoff. While this is the default, it is not recommended and not necessary. If you wish a lot of precision $P = 80$ is more than enough.

(recommended for T_m) `-cutoff=10` a cutoff $P = 10$ gives quite good results and reduces the computational cost for melting tenperatures.

(recommended for $\langle y \rangle$) `-cutoff=80` a cutoff $P = 80$ gives quite good results for average openings.

4.1.12 -pbc=<0 or 1>

Controls the type of boundary conditions.

(default) `-pbc=0` open boundary conditions, this is what you normally would have and usually shows end fraying

`-pbc=1` periodic boundary conditions, this would be the case for a circular sequence.

4.1.13 -t=<temperature>

Selects the temperature in kelvin for which the calculation of the matrices is carried out. Please note: this temperature is completely unrelated to the melting temperatures.

(default) `-t=370`

4.1.14 -int=<range>

Range of integration, this specifies the limits of the integral shown in Eq. (14) of Ref. 20 and the size M of the matrices. There are two distict notations for ranges, one is of type `start:end:increment`, for example `-1:30:2` starts at -1 Å, ends at 30 Å and each step increments by 2 Å. The second type is `start:end/steps`, for example `-1:30/100` divides the interval starting at -1 and ending at 30 Å in 100 steps.

Important: for `-model=pb3DA` (4.1.7), which uses a radius r instead of a displacement y, this range should always start at 0 Å, example `0:200/100`.

(default) `-int=-1:30/100` integrates from $y = -1$ to $y = 30$ Å and uses matrices of size 100

(recommended) `-int=-1:200/400` integrates from $y = -1$ to $y = 200$ Å and uses matrices of size 400. This gives very accurate results without too much computational cost.

4.1.15 -ee=<experimental error>

This sets the experimental error (in °C) of the melting temperature set. If given, the data set (provided through `-data` (4.1.4)) with be modified by small positive or negative amounts such that the standard deviation is close to the experimental error.

(example) `-ee=0.5` will modify the dataset to within 0.5 °C

4.1.16 -rs=<seed>

Sets the seed of the random number generator (C function srand) which is used to modify the dataset. If you use the same seed you will get exactly the same random modifications. Only make sense to be used together with `-ee` (4.1.15).

4.1.17 -mlr=<sequence length>

Sets the minimal sequence length to be considered in calculating regression parameters.

(example) `-mlr=6` only considers sequences with 6 bp and above

4.1.18 -mar=<number of sequences>

Sets the minimal number of sequences to be considered in calculating regression parameters. Note that this depends on how the sequences are grouped together, see `-pm` (4.1.20) which defines the groups.

(default) `-mar=3` does not attempt regressions with less than 3 sequences for any group

(example) `-mar=2` accepts 2 sequences for any group. Note: `-mar=2` is not recommended at all except in very special circumstances.

4.1.19 -pg=<prediction groups>

This option is used in conjunction with `-pm=-2` (4.1.20) and defines what type of grouping should be used.

(default) `-pg=ct` groups all T_m data by the species concentration C_t.

`-pg=salt` groups by salt concentration as found in the data file given in `-data` (4.1.4).

`-pg=key` groups by an arbitrary key which is the *first* word that follows the comment sign `#` (see 7.2).

4.1.20 -pm=<prediction method number>

Selects the prediction/regression method and how melting temperatures and sequences are grouped.

(default) `-pm=2` sequences are grouped by length N and by salt concentration $[Na^+]$, and considers two equations for regression

$$T_p = a_0(N, [Na^+]) + a_1(N, [Na^+])\tau, \tag{4.1}$$

there will be one equation for each length N and for each salt concentration $[Na^+]$ and each coefficient a_k is calculated as

$$a_k(N, [Na^+]) = b_{0,k}([Na^+]) + b_{1,k}([Na^+])N^{1/2}, \tag{4.2}$$

where τ is the adimensional equivalence index. Note that there need to be at least 3 different sequences for each length N (this number can be changed with the option `-mar`). **Important:** if the dataset has non-uniform salt concentrations, there will be separate regressions for each salt concentrations. For example, say that the dataset mixes measurements at 119 mM and 220 mM, there will be one regression Eq. (4.1) for 119 mM and another for 220 mM. That is the reason why the coefficients are given as a function of N and $[Na^+]$.

`-pm=3` like `-pm=2`, in addition of the two previous equations, this considers a third equation for the case where there are two or more salt concentrations in the same data file

$$b_{j,k}([\mathrm{Na}^+]) = c_{0,j,k} + c_{1,j,k} \log[\mathrm{Na}^+]. \tag{4.3}$$

Note that if there is only one salt concentration $[\mathrm{Na}^+]$ in the data file this reverts automatically to option `-pm=2`.

`-pm=-1` forms a single groups with all melting temperatures and considers only one regression

$$T_p = a_0 + a_1 \tau, \tag{4.4}$$

which is useful if the dataset has only a few sequences, or if all sequences are of the same length or if you have otherwise trouble in getting good linear regression coefficients for Eq. (4.2). The last situation may happen if your model parameters are very far from the optimized values. This option was first introduced for Ref. [32].

`-pm=-2` this considers separate regressions of type `-pm=-1` per group, where the group type is specified with the option `-pg` (4.1.19).

4.1.21 -printusedpar=<0 or 1>

Controls the printing of a file with extension `.usedpar.` that will output all parameters as they were read from files given in `-par` (4.1.3). See section 8.5 for the output format.

(default) `-printusedpar=0` does not generate a file

`-printusedpar=1` generates a file with extension `.usedpar.`, taking the base name from `-o` (4.1.1)

4.1.22 -seq=<nucleotide sequence>

Instead of providing a file with your sequences you can give them on the command line. This is useful if you want to see a melting for just one sequence. You should give the main strand from $5'$ to $3'$, the complementary sequence will be worked out automatically.

(example) `-seq=ACGTTGAATT`

4.1.23 -cseq=<nucleotide sequence>

You should provide a $3' \to 5'$ sequence if your sequence is not perfectly complementary, say like in a sequence with nucleotide mismatches.

(example) `-cseq=TGCTACTTAA` from $3' \to 5'$

4.1.24 -salt=<salt concentration>

In the case where a `-reg` (4.1.2) file contains several salt concentration, this will select the one that should be used for the calculations. Otherwise, this option is silently ignored. *If you need to calculate melting temperatures for different salt concentrations please see section 9.3.*

4.1.25 -v=<0 or 1>

Controls the printing to `stdout` of options given.

(default) `-v=0` prints only the options provided by the user

`-v=1` prints all options, regardless if they were provided or not.

4.1.26 -debug=<predefined flags>

`TfReg` has several debug flags which print further information to `stderr`. Debug flags are currently sparsely documented, are of limited interest to the use and are intended mainly for the developer. If critically needed you may consult the contents of the source file `ErrorCodes.h`. Another possibility is to simply turn all flags on by using `-debug=ALL`, but note that this will generate a huge amount of information and it is recommended to redirect `stderr` to a file. See section A.2 for available debug flags. Sometimes it is useful to activate debug flags using the environment variable, before running `TfReg`, for example

```
export TFREG="-debug=DOPTSHEQ"
```

this helps debugging without actually touching your scripts.

5 Models

There are several variants of the PB model, each of which requires different model parameters. In this section you will find which models TfReg currently supports and which model parameters are needed. Example files with model parameters are provided and typically you will find these in `/usr/share/TfReg/data` (or `/usr/share/tfreg/data`) with file extension `.par`, see also section 10.1. Note that the parameters are specified per nucleotide type and follow a very specific notation described in section 6.

The configurational part of the Model Hamiltonian has the general form

$$U_{i,i-1} = V(y_i) + w(y_i, y_{i-1}) \tag{5.1}$$

where V is a potential for the base-pair at site i, and w a nearest-neighbour potential for sites $i-1$ and its neighbour i.

5.1 Peyrard-Bishop (`-model=pb`)

This is the original Peyrard-Bishop model proposed in Ref. [1] which uses a Morse potential for modelling the hydrogen bonds

$$V_{\text{Morse}}(y_i) = D\left(e^{-y_i/\lambda} - 1\right)^2, \tag{5.2}$$

where y_i is the relative displacement of the bases and λ a potential width factor. Note that strictly speaking, there is a factor of $\sqrt{2}$ that is not explicit, that is,

$$V_{\text{Morse}}(y_i) = D\left(e^{-\sqrt{2}y_i/\lambda'} - 1\right)^2, \tag{5.3}$$

these were present in the early articles [1, 33] but were later factored into the λ, such that

$$\lambda = \frac{\lambda'}{\sqrt{2}} \tag{5.4}$$

It is important to bear this in mind in case one wishes to reproduce the results in Ref. [1, 33]. The nearest-neighbour stacking interaction is described as a harmonic oscillator

$$w_{\text{harm.}}(y_i, y_{i-1}) = \frac{k}{2}\left(y_i - y_{i-1}\right)^2. \tag{5.5}$$

because of the divergence of the partition function [34] we modified this to

$$w_{\text{harm.}}(y_i, y_{i-1}) = \frac{k}{2}\left(y_i^2 - 2y_i y_{i-1}\cos\theta + y_{i-1}^2\right), \tag{5.6}$$

see [22] for details on the parameter θ. The configurational part of the Hamiltonian is

$$U_{i,i-1} = V_{\text{Morse}}(y_i) + w_{\text{harm.}}(y_i, y_{i-1}) \tag{5.7}$$

See Tab. 5.1 for the description of program parameters. An example of parameter file

data/dna_pb_119.par

```
dna-pb-119
#weber09b doi: 10.1038/nphys1371
#revised 23/07/2021
Na+:concentration 119
*:harmonic.theta 0.01
AT:morse.D 0.0319629
AT:morse.lambda 0.359734
AT_AT:harmonic.k 0.0221774
AT_CG:harmonic.k 0.0257994
AT_GC:harmonic.k 0.0231292
```

```
AT_TA:harmonic.k 0.0160271
CG:morse.D 0.0732641
CG:morse.lambda 0.106465
CG_AT:harmonic.k 0.0351931
CG_CG:harmonic.k 0.020606
CG_GC:harmonic.k 0.0274629
GC_AT:harmonic.k 0.0280368
GC_CG:harmonic.k 0.0336531
TA_AT:harmonic.k 0.025482
```

5.2 The Dauxois variant (`-model=dpb`)

In 1993 Dauxois, Peyrard and Bishop (DPB or PBD) introduced an anharmonicity term to account for sharp transitions in the original PB model [21, 35],

$$w_{\text{an.}}(y_i, y_{i-1}) = \left[1 + \rho e^{-\alpha(y_i + y_{i-1})}\right] w_{\text{harm.}}(y_i, y_{i-1}), \tag{5.8}$$

where $w_{\text{harm.}}$ is from Eq. (5.6). Set the parameter `harmonic.theta` to zero ($\theta = 0$), to reproduce results from the original papers. The configurational part of the Hamiltonian is

$$U_{i,i-1} = V_{\text{Morse}}(y_i) + w_{\text{an.}}(y_i, y_{i-1}) \tag{5.9}$$

An example parameter file using the arguments of Tab. 5.2.

data/weber06-1.par

```
weber06
AT:AU:morse.D         0.05
CG:morse.D          0.08
AT:AU:morse.lambda    0.33333
CG:morse.lambda     0.125
*:harmonic.theta    0.01
*:harmonic.k        0.025
*:anharmonic.alpha  0.35
*:anharmonic.rho    2.0
```

5.3 PB model with added solvent potential (`-model=hms`)

A solvent term was added to the harmonic PB model [22]

$$V_{\text{solvent}}(y_i) = -f_s D \left[\tanh[(y_i + y_e)/\lambda_s] + s\right], \tag{5.10}$$

which is an adaptation of the solvent term Eq. (7) of Ref. [36] and Eq. (6) of Ref. [37]

$$-\frac{1}{4} v_0 \left[\tanh[\beta(r + r_H^*)] + 1\right] \tag{5.11}$$

which corresponds to $f_s = 1/4$, $v_0 = D$, $\beta = 1/\lambda$ and the radial distance is adapted as $r \to y$. For $s = 1$ and $y_e = 0$ takes the particular form[1]

$$V_{\text{solvent}}(y_i) = -f_s D \left[\tanh(y_i/\lambda_s) + 1\right] \tag{5.12}$$

The configurational part of the Hamiltonian is

$$U_{i,i-1} = V_{\text{Morse}}(y_i) + V_{\text{solvent}}(y_i) + w_{\text{harm.}}(y_i, y_{i-1}) \tag{5.13}$$

data/weber06b-1.par

```
weber06b
AT:morse.D          0.05
CG:morse.D          0.08
AT:morse.lambda     0.33333
CG:morse.lambda     0.125
*:solvent.eq_sol    0.0
*:solvent.sign_sol  1.0
*:harmonic.theta    0.01
*:harmonic.k        0.025
*:solvent.lambda    1.0
*:solvent.f_s        0.1
```

[1] Note that in Eq. (2) of Ref. [22] the solvent term was mistyped as $s = -1$, however all results in the article are for $s = +1$.

Hamiltonian	model parameter	program parameter	units	type
$D\left(e^{-y_i/\lambda}-1\right)^2$	D	`morse.D`	eV	BP
	λ	`morse.lambda`	Å	BP
$\frac{k}{2}\left(y_i^2-2y_iy_{i-1}\cos\theta+y_{i-1}^2\right)$	k	`harmonic.k`	eV/Å^2	NN
	$\theta^\dagger$	`harmonic.theta`	rad	NN

†Parameter introduced in [22], not part of the original model. Set $\theta = 0$ to reproduce results from the original papers, but be aware that you will have divergence issues discussed in [34].

Table 5.1
Parameters for `-model=pb` [1].

Hamiltonian	model parameter	program parameter	units	type
$D\left(e^{-y_i/\lambda}-1\right)^2$	D	`morse.D`	eV	BP
	λ	`morse.lambda`	Å	BP
$\left[1+\rho e^{-\alpha(y_i+y_{i-1})}\right]$	k	`harmonic.k`	eV/Å^2	NN
$\times\frac{k}{2}\left(y_i^2-2y_iy_{i-1}\cos\theta+y_{i-1}^2\right)$	$\theta^\dagger$	`harmonic.theta`	rad	NN
	ρ	`anharmonic.rho`	adimensional	NN
	α	`anharmonic.alpha`	Å^{-1}	NN

†Parameter introduced in [22], not part of the original model. Set $\theta = 0$ to reproduce results from the original papers.

Table 5.2
Parameters for `-model=dpb` [21].

Hamiltonian	model parameter	program parameter	units	type
$D\left(e^{-y_i/\lambda}-1\right)^2$	$D^\dagger$	`morse.D`	eV	BP
	λ	`morse.lambda`	Å	BP
$-f_sD\left[\tanh[(y_i+y_e)/\lambda_s]+s\right]$	f_s	`solvent.f_s`	adimensional	BP
	λ_s	`solvent.lambda`	Å	BP
	y_e	`solvent.eq_sol`	Å	BP
	s	`solvent.sign_sol`	adimensional	BP
$\frac{k}{2}\left(y_i^2-2y_iy_{i-1}\cos\theta+y_{i-1}^2\right)$	k	`harmonic.k`	eV/Å^2	NN
	θ	`harmonic.theta`	rad	NN

Table 5.3
Parameters for `-model=hms` [22]. Setting $f_s = 0$ reproduces `-model=pb` (4.1.7).

5.4 The Joyeux and Buyukdagli model (-model=jb)

The model by Joyeux and Buyukdagli [23–26] introduces a finite stacking enthalphy

$$w_{\text{fin.}}(y_i, y_{i-1}) = \frac{\Delta H}{C}\left[1 - e^{-b(y_i - y_{i-1})^2}\right] + \frac{K_b}{2}(y_i - y_{i-1})^2 . \tag{5.14}$$

The configurational part of the Hamiltonian is

$$U_{i,i-1} = V_{\text{Morse}}(y_i) + w_{\text{fin.}}(y_i, y_{i-1}) \tag{5.15}$$

data/jb.par

```
var_jb_owczarzy04_init2
AT:morse.D          0.041
CG:morse.D          0.054
AT:morse.lambda     0.1667
CG:morse.lambda     0.1667
*:finite_enthaphy.C 4.0
*:finite_enthalpy.DeltaH 0.409
*:finite_enthalpy.b  0.80
*:finite_enthalpy.kb 4.0e-4
```

5.5 The PB model including rise step h (-model=mes)

This Hamiltonian [28] rewrites the torsional 3D Hamiltonian [38] in the notation of the original PB model [1] by setting the angles to zero ($\phi_i = 0$ and $\theta_0 = 0$), we obtain

$$V_{\text{Morse}} = D(e^{-y_i/\lambda} - 1)^2 \tag{5.16}$$

and

$$w_{\text{exact}}(y_i, y_{i-1}) = +k\left(\sqrt{h^2 + \frac{1}{2}(y_i - y_{i-1})^2} - h\right)^2 \tag{5.17}$$

where h is stacking distance (rise) between base pairs. Unlike the PB model where the stacking factor Eq. (5.5) is approximate, this model evaluates the stacking exactly. The acronym `mes` stands for "Morse Exact Stacking". The configurational part of the Hamiltonian is

$$U_{i,i-1} = V_{\text{Morse}} +_{\text{exact}} (y_i, y_{i-1}) \tag{5.18}$$

5.6 The approximated 3D model (-model=pb3DA)

This model uses a plain 3D framework [9] and works out several approximations to take advantage of the 1D framework of PB-type models. For the first order expansion of Eq. (10) from Ref. 9

$$U(r_n, r_{n-1}) = D\left[e^{-(r-R_0)\lambda} - 1\right]^2 + \frac{k}{8J_0^2}\left[(r_n - r_{n-1})^2 + \omega^2 r_n r_{n-1}\right]^2 \tag{5.19}$$

here instead of the relative displacement y of the PB-type models we use a radius r defined in polar cylindrical coordinates, and ω is a twist angle, R_0 is an equilibrium distance. The second order expansion of Eq. (10) results in

$$\begin{aligned} U(r_n, r_{n-1}) \quad = \quad & D\left[e^{-(r-R_0)\lambda} - 1\right]^2 + \frac{k}{8J_0^2}\left[(r_n - r_{n-1})^2 + \omega^2 r_n r_{n-1}\right]^2 \\ & - \frac{k}{16J_0^4}\left[(r_n - r_{n-1})^2 + \omega^2 r_n r_{n-1}\right]^3 \\ & + \frac{5k}{128J_0^6}\left[(r_n - r_{n-1})^2 + \omega^2 r_n r_{n-1}\right]^4 \end{aligned} \tag{5.20}$$

see table 5.6 for the program parameters.

Hamiltonian	model parameter	program parameter	units	type
$D\left(e^{-y_i/\lambda}-1\right)^2$	D	`morse.D`	eV	BP
	λ	`morse.lambda`	Å	BP
$\frac{\Delta H}{C}\left[1-e^{-b(y_i-y_{i-1})^2}\right]$	ΔH	`finite_enthalpy.DeltaH`	eV	NN
	C	`finite_enthalpy.C`	adimensional	NN
	b	`finite_enthalpy.b`	adimensional	NN
$\frac{K_b}{2}\left(y_i-y_{i-1}\right)^2$	K_b	`finite_enthalpy.Kb`	eV/Å^2	NN

Table 5.4
Parameters for `-model=jb` [23–26].

Hamiltonian	model parameter	program parameter	units	type
$D\left(e^{-y_i/\lambda}-1\right)^2$	D	`morse.D`	eV	BP
	λ	`morse.lambda`	Å	BP
$k\left(\sqrt{h^2+\frac{1}{2}(y_i-y_{i-1})^2}-h\right)^2$	k	`harmonic.k`	eV/Å^2	NN
	h	`exactstack.h`	Å	NN

Table 5.5
Parameters for `-model=mes` [28].

Hamiltonian	model parameter	program parameter	units	type
$D\left[e^{-(r-R_0)\lambda}-1\right]^2$	D	`morse3D.D`	eV	BP
	λ	`morse3D.lambda`	Å	BP
	R_0	`morse3D.R0`	Å	BP
First order Hamiltonian	Eq. (5.19)	`harmonic3DA.J_order=1`		
Second order Hamiltonian	Eq. (5.20)	`harmonic3DA.J_order=2`		
$\frac{k}{8J_0^2}\left[(r_n-r_{n-1})^2+\omega^2 r_n r_{n-1}\right]^2$	k	`harmonic3DA.k`	eV/Å^2	NN
	J_0	`harmonic3DA.J0`	Å	NN
	ω	`harmonic3DA.omega`	Å	NN

Table 5.6
Parameters for `-model=pb3DA` [9].

5.7 Morse potential with y^3 entropic barrier (-model=pcla)

Introduced by Peyrard, Cuesta-Lopez and Angelov (PCLA) [29] it adds an entropic barrier to the Morse potential

$$V_{\text{barrier}}(y_i) = \Theta(y) \frac{b y_i^3}{\cosh^2\left[c(\alpha y_i - d \ln 2)\right]} \tag{5.21}$$

This model is also used in [39]. The configurational part of the Hamiltonian is

$$U_{i,i-1} = V_{\text{Morse}}(y_i) + V_{\text{barrier}}(y_i) + w_{\text{an.}}(y_i, y_{i-1}) \tag{5.22}$$

To use it see the parameters in Tab. 5.7. An example parameter file, that was used to generate Fig. 5.1 is shown here

data/peyrard09.par

```
peyrard09
#for the pcla model doi:10.1088/0953-8984/21/3/034103
AT:morse.D 0.09900
AT:morse.lambda 0.333333333333 #1/lambda=alpha=3.0
AT:barrier_y3.b 4.00
AT:barrier_y3.c 0.74
AT:barrier_y3.d 0.20
AT:barrier_y3.alpha 3.0
*:anharmonic.rho 25
*:anharmonic.alpha 0.8 #called \delta in the article
*:harmonic.theta 0
CG:morse.D 0.09900
CG:morse.lambda 0.294117647059 #1/lambda=alpha=3.4
CG:barrier_y3.b 6.00
CG:barrier_y3.c 0.74
CG:barrier_y3.d 0.20
CG:barrier_y3.alpha 3.4
AT_TA:harmonic.k 0.00017 #A-T
AT_AT:TA_TA:harmonic.k 0.00418 #A-A T-T
GC_TA:harmonic.k 0.00480 #G-T this will be ignored as the program will read only AT_CG
AT_CG:harmonic.k 0.00462 #A-C should in principle be the same as GC_TA
TA_AT:harmonic.k 0.00506 #T-A
GC_AT:TA_CG:harmonic.k 0.00546 #G-A T-C
CG_CG:GC_GC:harmonic.k 0.00810 #C-C G-G
GC_CG:harmonic.k 0.00865 #G-C
CG_TA:AT_GC:harmonic.k 0.00865 #C-T A-G
CG_AT:TA_GC:harmonic.k 0.001140 #C-A T-G
CG_GC:harmonic.k 0.01690 #C-G
```

5.8 Morse 'hump' potential (-model=pclj)

Introduced by Peyrard, Cuesta-Lopez and James (PCLJ) [30] it adds an entropic barrier to the Morse potential, which is nicknamed 'hump' by some authors [40]

$$V_{\text{hump}}(y_i) = \begin{cases} A\left(e^{y/\lambda} - 1\right)^2 & y < 0 \\ ay^2 + by^3 + cy^4 & 0 \le y \le 1 \text{ Å} \\ D + Ee^{-y/\gamma}\left(y + \gamma\right) & y > 1 \text{ Å} \end{cases} \tag{5.23}$$

where we replaced $\alpha = 1/\lambda$ and $\beta = 1/\gamma$ in regard to [30,41], to be consistent with our previous notation of the Morse potential Eq. (5.2). The configurational part of the Hamiltonian is

$$U_{i,i-1} = V_{\text{hump}}(y_i) + w_{\text{an.}}(y_i, y_{i-1}) \tag{5.24}$$

Note that V_{hump} replaces completely V_{Morse};. Program parameters are shown in Tab. 5.8, and some coefficients are deduced from the continuity of the first and second derivatives at $y = 0$ and $y = 1$ Å,

$$\begin{aligned} A &= a\lambda^2 \\ a &= 6D + \frac{1}{2}Ee^{-1/\gamma}\left(\gamma^{-2} + 5\gamma^{-1} + 12\gamma + 12\right) \\ b &= -8D - Ee^{-1/\gamma}\left(\gamma^{-2} + 4\gamma^{-1} + 8\gamma + 8\right) \\ c &= -3D\frac{1}{2}Ee^{-1/\gamma}\left(\gamma^{-2} + 3\gamma^{-1} + 6\gamma + 6\right) \end{aligned} \tag{5.25}$$

see [42] for details. The factors of Eq. 5.25 are not configurable and are not printed in the log files. If you are interested to see their actual values select -debug=DMOP_HCF (4.1.26).

An example of input parameters that were used to calculate Fig. 5.2

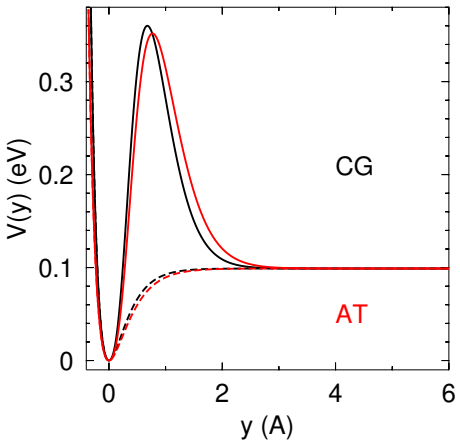

Figure 5.1
Base-pair potential of the entropic barrier model calculated for Eq. (5.21). Parameters are the same as of Ref. [29], dashed curve was calculated for $b = 0$ reproduces that of Fig. 7 of Ref. [30].

Hamiltonian	model parameter	program parameter	units	type
$D\left(e^{-y_i/\lambda}-1\right)^2$	D	`morse.D`	eV	BP
	λ	`morse.lambda`	Å	BP
$by_i^3/\cosh^2\left[c(\alpha y_i - d\ln 2)\right]$	b	`barrier_y3.b`	eV/Å^{3*}	BP
	c	`barrier_y3.c`	adimensional*	BP
	d	`barrier_y3.d`	adimensional	BP
	$\alpha^\dagger$	`barrier_y3.alpha`	Å^{-1}	BP
$\left[1+\rho e^{-\alpha(y_i+y_{i-1})}\right]$	k	`harmonic.k`	eV/Å^2	NN
$\times\frac{k}{2}\left(y_i^2-2y_iy_{i-1}\cos\theta+y_{i-1}^2\right)$	θ	`harmonic.theta`	rad	NN
	ρ	`anharmonic.rho`	adimensional	NN
	$\alpha^\dagger$	`anharmonic.alpha`	Å^{-1}	NN

†In the papers [29, 39] the α parameters are the same, however in our program implementation they can assume different values.

*Apparently, the units used in [29, 39] are not dimensionally consistent, since the argument to cosh needs to be adimensional, therefore c also needs to be adimensional. The result of cosh being adimensional, the term by^3 can only be eV if b is eV/Å^3. Please contact me if you have a better explanation for this.

Table 5.7
Parameters for `-model=pcla` [29]. Setting $b = 0$ disables this additional barrier term which then becomes the same as `-model=dpb`.

data/peyrard09b.par

```
peyrard09b
#for the pclj model doi:10.1007/s10867-009-9127-2
*:hump.D 0.0857
*:hump.lambda 0.25 #1/lambda=alpha=4.0
*:hump.gamma 0.25 #1/gamma=beta=4.0
*:hump.E 4.0
*:harmonic.k 0.01
*:harmonic.theta 0
*:anharmonic.rho 3.0
*:anharmonic.alpha 0.8 #called \delta in the article
```

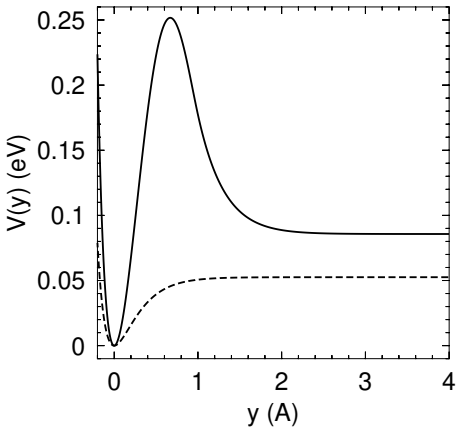

Figure 5.2
Base-pair potential of the hump model calculated for Eq. (5.23). Parameters are the same as of Ref. [30], dashed curve was calculated for `-model=dpb` (4.1.7) and this calculation is a reproduces that of Fig. 7 of Ref. [30].

Hamiltonian term	program parameter	units	type	
$A\left(e^{y/\lambda}-1\right)$	A	calculated from Eq. (5.25)		
	λ	`hump.lambda`	Å	BP
$ay^2+by^3+c^4$	a	calculated from Eq. (5.25)		
	b	calculated from Eq. (5.25)		
	c	calculated from Eq. (5.25)		
$D+Ee^{-y/\gamma}\left(y+\gamma\right)$	D	`hump.D`	eV	BP
	E	`hump.E`	eV/Å*	BP
	γ	Å	BP	
$\left[1+\rho e^{-\alpha(y_i+y_{i-1})}\right]$	k	`harmonic.k`	eV/Å^2	NN
$\times\frac{k}{2}\left(y_i^2-2y_iy_{i-1}\cos\theta+y_{i-1}^2\right)$	θ	`harmonic.theta`	rad	NN
	ρ	`anharmonic.rho`	adimensional	NN
	$\alpha^\dagger$	`anharmonic.alpha`	Å^{-1}	NN

*In Ref. [30] E appears in several places as having units of 1/Å which is probably a typing mistake.

Table 5.8
Parameters for `-model=pclj` [30, 41].

5.9 Morse potential with Gaussian barrier (`-model=trmf`)

Introduced by Tapia-Rojo, Mazo, and Falo (TRMF) [31] it adds an Gaussian barrier to the Morse potential

$$V_{\text{Gaussian}}(y_i) = Ge^{-(y_i-y_0)^2/b} \tag{5.26}$$

The configurational part of the Hamiltonian is

$$U_{i,i-1} = V_{\text{Morse}}(y_i) + V_{\text{Gaussian}}(y_i) + w_{\text{an.}}(y_i, y_{i-1}) \tag{5.27}$$

To use it see the parameters in Tab. 5.9. An example of input parameters that were used to calculate Fig. 5.3

data/tapiarojo10.par

```
tapiarojo10
#for the trmf model Fig 1 doi:10.1103/PhysRevE.82.031916
AT:morse.D 0.05185
AT:morse.lambda 0.25    #alpha=4
AT:gaussian.G 0.15555   #G=3*D
AT:gaussian.y0 0.5      #2/alpha=2*lambda
AT:gaussian.b 0.03125   #1/2*alpha^2=0.5*lambda^2
CG:morse.D 0.077775       #1.5*AT:morse.D
CG:morse.lambda 0.166666666667 #alpha=6
CG:gaussian.G 0.233325   #G=3*D
CG:gaussian.y0 0.333333333334   #2/alpha=2*lambda
CG:gaussian.b 0.0138888888889   #1/2*alpha^2=0.5*lambda^2
#the following values are not used for plotting Fig 1
*:harmonic.k 0.03
*:harmonic.theta 0
*:anharmonic.rho 3.0
*:anharmonic.alpha 0.8 #called \delta in the article
```

Hamiltonian term	program parameter	units	type	
$D\left(e^{-y_i/\lambda}-1\right)^2$	D	`morse.D`	eV	BP
	λ	`morse.lambda`	Å	BP
$Ge^{-(y_i-y_0)^2/b}$	G	`gaussian.G`	eV	BP
	y_0	`gaussian.y0`	Å	BP
	b	`gaussian.b`	Å	BP
$\left[1+\rho e^{-\alpha(y_i+y_{i-1})}\right]$	k	`harmonic.k`	eV/Å^2	NN
$\times\frac{k}{2}\left(y_i^2-2y_iy_{i-1}\cos\theta+y_{i-1}^2\right)$	θ	`harmonic.theta`	rad	NN
	ρ	`anharmonic.rho`	adimensional	NN
	$\alpha^\dagger$	`anharmonic.alpha`	Å^{-1}	NN

Table 5.9
Parameters for `-model=trmf` [31]. Setting $G=0$ turns off the potential barrier and becomes equivalent to `-model=dpb` (4.1.7).

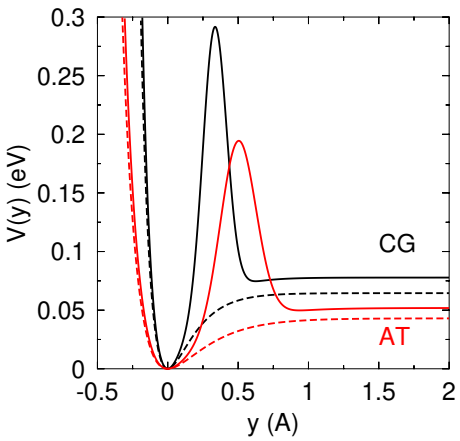

Figure 5.3
Base-pair potential of the Gaussian barrier model calculated for Eq. (5.26). Parameters `tapiarojo10.par` are the same as of Ref. [31], dashed curve was calculated for $G=0$ (`tapiarojo10-nobarrier.par`) reproduces that of Fig. 1 of Ref. [31].

5.10 Build your own model (`-model=test`)

We have been contacted by some users who wanted to modify tfreg for their own models. Here we are trying to make this as simple as possible, but please be warned that recompilation of the code is required. Instructions for downloading and compiling the source code are given in section 3.4.

No knowledge C++ is required to make the modifications to the code described in this section. The only changes you will make require a tiny bit of knowledge of how to write a simple equation in standard C.

If you are interested in testing different potentials see the next section. If you would like to try different combinations of existing potentials, without introducing new potentials, see section 5.10.2.

5.10.1 Changing the model potentials

Parameters We reserved 8 parameters for each potential. For the $V(x)$ potential those are

```
test_vx.a test_vx.b test_vx.c test_vx.d test_vx.e test_vx.f test_vx.g test_vx.h
```

and for $w(x, y)$

```
test_wxy.a test_wxy.b test_wxy.c test_wxy.d test_wxy.e test_wxy.f test_wxy.g test_wxy.h
```

your potentials will have to be written considering those available parameters.

Prepare your model First work out what the required parameters of your model will be. Typically you will have two potentials, one is the base-pair potential $V(x)$ and the second is the stacking potential $w(x, y)$.

Lets suppose that your $V(x)$ potential is $V = x(a + b^2)$, and your stacking potential $w(x, y) = ax + by^2$, then your parameter file should look more or less like this

```
identification
AT:test_vx.a 0.2
AT:test_vx.b 0.3
AT_AT:test_wxy.a 1.2
AT_AT:test_vxy.b 3.3
```

where the first line is an arbitrary identification string.

Change the source code Using a program file editor, open the file `TestModel.h`, and locate the line

```
class TestVx: public BasePotential<_Tp>
```

and scroll down until you find

```
return x*(a+b+c+d+e+f+g+h); // MODIFY HERE
```

for our example, $V = x(a + b^2)$ we would change this to

```
return x*(a+pow(b,2));
```

where we used simple C math formatting.

Now locate

```
class TestWxy: public BasePotential<_Tp>
```

and scroll down until you find

```
return x*y*(a+b+c+d+e+f+g+h); // MODIFY HERE
```

and for the example $w = ax + by^2$ change this to

```
return a*x+b*pow(y,2);
```

Now recompile the code, which mostly boils down to issue the `make` command, see section 3.4. To run use `-model=test`.

5.10.2 Changing the Hamiltonian

If you simply want a different combination of existing model potentials, then you only have to change the main Hamiltonian. Open the file `TestModel.h` and locate the line

```
class TestHamiltonian: public Hamiltonian<TestVx<_Tp>,TestWxy<_Tp> > // Here specify the how the....
```

suppose that you would like to use a Hamiltonian where the V potential is a simple Morse-solvent potential but with anharmonic stacking, you should then replace `TestVx` with `MorseSolvent` and `TestWxy` with `AnharmonicStacking`. The above line would then look like

```
class TestHamiltonian: public Hamiltonian<MorseSolvent<_Tp>,AnharmonicStacking<_Tp> >
```

also change in the exact same way the line

```
typedef Hamiltonian<TestVx<_Tp>,TestWxy<_Tp> > base_type;// Here specify the how the Hamiltonian
```

to

```
typedef Hamiltonian<MorseSolvent<_Tp>,AnharmonicStacking<_Tp> > base_type;
```

Now recompile the code, which mostly boils down to issue the `make` command, see section 3.4. To run use `-model=test`.

6 Syntax of parameter files

This chapter explains how to write and use a parameter file.

Let's start with a simple parameter file

data/dna_pb_69.par

```
dna-pb-69 weber09b
#weber09b doi: 10.1038/nphys1371
#revised 23/07/2021
Na+:concentration 69
*:harmonic.theta 0.01
AT:morse.D 0.0324604
AT:morse.lambda 0.362944
AT_AT:harmonic.k 0.0240653
AT_CG:harmonic.k 0.0256472
AT_GC:harmonic.k 0.0224831
AT_TA:harmonic.k 0.0183875
CG:morse.D 0.0733641
CG:morse.lambda 0.10156
CG_AT:harmonic.k 0.0344013
CG_CG:harmonic.k 0.0205701
CG_GC:harmonic.k 0.0273539
GC_AT:harmonic.k 0.0280203
GC_CG:harmonic.k 0.0335629
TA_AT:harmonic.k 0.0241575
```

The first line holds a simple identifier and comments. The identifier is a single word and stops at the first occurrence of space or newline. This identifier is used when you specify the `-matrix` option to identify the files, therefore, make sure it does not contain characters that clashes with file system characters such as a forward slash.

(Since version 6.0) The rest of the line after the identifier is ignored, you may use it to write a note about the set of parameters.

Second to last lines contain the actual parameters to be read. Each line should start with the base pair or nearest-neighbour configuration, followed by one or more spaces, and a number containing the actual value of the parameter in the units defined in chapter 5. The remaining content of the line is ignored and may be used for annotations.

In the next sections we will explain the syntax of the generic, base pair and nearest neighbours configurations.

6.1 Generic * parameters

The generic base `*` specification can be used when a given parameter should be applied to any base pair or any nearest-neighbours. Its typical use is as a fallback when specific parameters are not available.

data/generic-bp.par

```
initial
*:harmonic.theta   0.01
*:harmonic.k   0.025
*:morse.D 0.0324083
```

However, if the parameter should applied to a specific base pair you should specify either in BP or NN form (see next section).

6.2 BP parameters

Base-pair parameters are those which generally do not depend on the context, that is, it is not relevant which are the neighbours of the given base pair. For DNA we use Watson-Crick base pairs AT=TA and CG=GC, but may also use mismatched base pairs such as GT or AA. Switching the nucleotides, eg. GT to TG, makes

no difference for BP parameters. The reason for this is that they usually represent properties of the hydrogen bond.

The format which needs to be specified in the parameter file is the base pair code, followed by the model parameter and its value, like in the following example

BP parameter example 1

```
AT:morse.D 0.5
```

You can specify multiple base pair with the same parameters as in the following example

BP parameter example 2

```
AT:CG:morse.D 0.5
```

which is equivalent to

BP parameter example 3

```
AT:morse.D 0.5
CG:morse.D 0.5
```

Hybrid DNA-RNA and DNA-TNA an additional character is required, `d` for the DNA strand, `r` for RNA strand and `t` for TNA strand. For example `dCrG` means C on the DNA strand and G on the RNA strand. `TfReg` should be able to read a construct such as `dCdG` for DNA-DNA or `rCrG` for RNA-RNA, however this is generally unnecessary.

BP example for DNA-RNA

```
dArU:morse.D 0.027758
dCrG:morse.D 0.073671
dGrC:morse.D 0.062591
dTrA:morse.D 0.040300
```

6.2.1 Context dependence for BP parameters

There are situations where you may need different BP parameters for the same base pairs. Such a situations arises for example for the GU wobble pair in RNA [18]. In this case we will denote the base pair by an arbitrary superscript like GU^a or GU^b. First however we need to add context rules for these base pairs. This is all done in the parameter file as illustrated here:

Context BP parameter example

```
+GU^a AGU/UUG,GUA/UGU
+GU^b AGG/UUU,AUU/UGG,GGA/UUU,UGG/AUU
GU^a:morse.D 0.50
GU^b:morse.D 0.50
```

where we are saying that any GU in the context `AGU/UUG` or `GUA/UGU` is given the specific name `GU^a`, and that all further parameters use this name like `GU^b:morse.D`. NN parameter specification (see next section) are equally affected by this new specification.

Note that the actual parameter set for RNA GU mismatches is more complicated than the above example. You can find the complete optimized parameters in file `rna_pb_GU.par` [18].

6.3 NN parameters

Nearest neighbour (NN) parameters follow the same conventions as usually found in linear regression models [43, 44]. The convention used is of type AB_CD, where AB is the first base pair and CD the second base pair. Note that DC_BA is equivalent to AB_CD, for example, AT_AT is the same as TA_TA. When specifying a NN sequence always specify in lexical ordering, that is write AT_AT and not TA_TA. For hybrid DNA-RNA an example would be `dArU_dCrG`.

The following example shows all 10 irreducible N parameters for `finite_enthalpy.DeltaH`.

Examples of NN parameters

```
AT_AT:finite_enthalpy.DeltaH 0.42085
AT_CG:finite_enthalpy.DeltaH 0.416718
AT_GC:finite_enthalpy.DeltaH 0.400366
AT_TA:finite_enthalpy.DeltaH 0.330006
CG_AT:finite_enthalpy.DeltaH 0.445258
CG_CG:finite_enthalpy.DeltaH 0.385034
CG_GC:finite_enthalpy.DeltaH 0.411902
```

```
GC_AT:finite_enthalpy.DeltaH 0.44666
GC_CG:finite_enthalpy.DeltaH 0.497106
TA_AT:finite_enthalpy.DeltaH 0.431154
```

6.3.1 Context dependence for NN

Note that if you are using context-dependent BP parameters, see section 6.2.1, you should use the superscript as normal

Examples of NN parameters with context dependent BP

```
GC_UG^j:harmonic.k 0.025
GU^a_UG^a:harmonic.k 0.025
```

For DNA-RNA or DNA-TNA hybrids it is necessary to use the d, r or t prefixes as in the following example for DNA-RNA

Examples of NN parameters for DNA-RNA

```
dArU_dArU:harmonic.k 0.009002
dGrC_dTrA:harmonic.k 0.042731
```

6.4 Context dependence fall back

(Introduced in version 5.3) If you are using context dependence BP and NN parameters, you may find yourself in a situation where the context-dependent parameter is not available. In this case TfReg will fall back to the equivalent non-context parameters and show a `WUBPG` warning. Consider the example where two BP contexts were introduced, say `GU^a` and `GU^b`, and your sequence would have NN contexts like `GU^a_GU^b` yet no such context parameters is available. However, if a non-context `GU_GU` exists it will be used in place of `GU^a_GU^b` and a warning will be printed. Note: this is an experimental feature introduced in version 5.3 and may be revised in future versions.

6.5 Parameter precedence

Since the program can read more than one parameter file, the last parameter read is the final value. There may also be several specification in the same file as well. Consider the following example

Precedence example 1

```
*:morse.D 0.3
AT:morse.D 0.5
```

the first line says that all base pairs should a `morse.D` value of 0.3. The second line however says that AT base pairs should use 0.5. In this case CG base pairs for instance will use 0.3 since nothing different was specified.

However, a generic base pair `*` does not supersedes a specific base pair as in the following example

Precedence example 2

```
AT:morse.D 0.5
*:morse.D 0.3
```

in this case AT base pairs will continue using 0.5, not 0.3. You should understand the generic base pair `*` as: *if nothing else matches, use this value.*

6.6 Documentation parameters

Some parameters are given only for documentation purposes, they are not actually used by TfReg. They are useful for the identification of the parameter files.

- `Na+:concentration` Na^+ concentration in mM
- `Ct:ln_group_f` logarithmic grouping factor introduced in Ref. [6]

7 Formatting the sequence data file

A file containing all sequences can be used to make bulk calculations. The sequence file names should be passed through the option `-data`, see section 4.1.4.

7.1 DNA or RNA duplexes

The file format is composed of columns

data/owczarzy04-69.dat

```
temperature
ATCAATCATA TAGTTAGTAT 21.3 69 2
TTGTAGTCAT AACATCAGTA 24.7 69 2
GAAATGAAAG CTTTACTTTC 22.1 69 2
```

The first line should always start with the word `temperature`, the remaining content of this line is ignored and can be used for annotations. The first column is the sequence (from $5' \rightarrow 3'$) and the second column is the secondary strand (from $3' \rightarrow 5'$). The third column is the melting temperature in °C, the fourth column is the salt concentration (in mM). The last column the species concentration (in μM) which is currently not used. The secondary strand does not necessarily need to be the complementary of the main strand as long as there are parameters for the mismatched pairs. If TfReg fails to find parameters for your sequences it will complain loudly.

7.2 Adding comments to your files

Since version 5.0 it is possible to add comments after the `#` sign for all sequences, as exemplified next

Example data with comments

```
temperature #sequences from schoning00
d(AAAAAAAAAAAAAAAA) t(TTTTTTTTTTTTTTTT) 32 1010 10 #DNA-TNA poly-A and poly-T
d(AAAATTTATATTATTA) t(TTTTAAATATAATAAT) 47 1010 10 #DNA-TNA
```

7.2.1 Special use of comments to group temperatures

There might be some interest in running separate melting temperature regressions by some arbitrary key. Consider the following example

Example data with comments used for grouping

```
XACGATCGTV VTGCTAGCAX  48.3 1000 1 #Cy3
XCTGATCAGV VGACTAGTCX  44.1 1000 1 #Cy3
YACGATCGTV VTGCTAGCAY 48.0 1000 1 #Cy5
YCTGATCAGV VGACTAGTCY 44.9 1000 1 #Cy5
```

the first two sequences are for Cy3 and last two for Cy5. Normally, one would have to split this in two files and run the regression separately for each file. However, an alternative is to use the file as it is, and with `-pm` (4.1.20)-2 together with `-pg` (4.1.19)key run a single regression. What will happen is that TfReg will read the first word after `#` (without spaces) and use this as index. Please note that this is case-sensitive.

7.3 Hybrid DNA-RNA duplexes

For hybrid DNA-RNA, all sequences should contain the additional `d` and `r` characters to distinguish if they are of DNA or RNA type [7]. The sequence should be enclosed as show in the following example.

Example data files for DNA-RNA

```
temperature
r(UUUGUAUCCAAU) d(AAACATAGGTTA) 45.6 1000 100
d(GTTGGTTGGTTG) r(CAACCAACCAAC) 60.0 1000 8.3
```

7.4 Hybrid DNA-TNA duplexes

In a very similar way as for DNA-RNA, you need the addition `d` and `t` characters as in the example below

Example data files for DNA-TNA

```
temperature
t(GCCGTGAG) d(CGGCACTC) 39.9 1010 10
t(ACGTCATTCCTC) d(TGCAGTAAGGAG) 44.6 1010 10
```

Note on strand concentrations: TfReg makes no adjustments to strand concentration C_t. Ideally the sequences should be adjusted beforehand to the same strand concentration that was used to calculate the parameters.

7.5 RNA single bulges

Single bulges occur when one strand is shorter than the other, but otherwise they are complementary to each other. To mark the position of the bulge (if known) we mark this with an X so that both strands become of equal length [17], as in the following example.

Example data file for RNA single bulges

```
temperature
GGCGACUCG CCGCXGAGC 57.6588010149076 1000 200
GAGCAGGUC CUCGXCCAG 52.8854893430268 1000 200
```

for a full example see the files

`rna_bulge_group1_adenosine_adj200.dat rna_bulge_group1_cytosine_adj200.dat`
`rna_bulge_group1_guanosine_adj200.dat rna_bulge_group1_uridine_adj200.dat`

included with TfReg.

7.6 Cy3 or Cy5 attached to DNA

When Cy3 or Cy5 is attached through a flexible linker to the 5′ end of DNA it behaves like an additional base pair [45, 46]. We use this to represent them as a XV (Cy3) or YV (Cy5) base pair, where X or Y are linked to the 5′ end. The following example shows how the sequences are formatted for the case of Cy3 [16].

Example data file for Cy3 attached to DNA

```
temperature
XACGATCGTV VTGCTAGCAX  48.3 1000 1
XCTGATCAGV VGACTAGTCX  44.1 1000 1
```

for a full example see files `moreira15cy3.dat` and `moreira15cy5.dat` included with TfReg.

7.7 LNA+DNA/DNA duplexes

LNA analogues of DNA bases are usually preceded with a plus sign, such as +A for LNA modified adenine, or with a L superscript as in A^L. However, currently TfReg only accepts single characters to represent a base. Therefore, for LNA we typically use some different characters to indicate LNA modifications. For instance in

data/you06.dat

```
temperature #you06 conversion of LNA W=+A R=+T N=+G Z=+C
GGTCCTTAZTTGGTG CCAGGAATGAACCAC  63.6 1020 2 #you06
GGTCCTTTZTTGGTG CCAGGAAAGAACCAC  64.4 1020 2 #you06
GGTCCTTCZTTGGTG CCAGGAAGGAACCAC  65.6 1020 2 #you06
```

we used the conversion W=+A R=+T N=+G Z=+C. Note that you will also need to add the option `-dict=R:R,N:N,W:W,Z:Z`.

8 Result files

TfReg produces many result files, here we will make an attempt to describe them in some detail. Some result files can be used again as input file such as the file with extension `.reg`. All files have a base name given by `-o` (4.1.1) to which an extension such as `.reg` is added.

8.1 .reg regression parameters

The regression parameters will be stored in a file with extension `.reg`

examples/verify/epl2011-69.reg

```
1  0.5 0.5 2
2  b
3  69 -86.7907 48.7773 15.3198 -6.02399
4  a
5  69
6  10 -39.4893 30.041
7  15 -26.6021 25.2989
8  20 -16.7027 21.2912
9  25 -10.8378 18.784
10 30 -3.52253 16.0363
```

Lets start from line 4 which contains the single letter `a`, this flags the start of all regression parameters which are length and salt concentration dependent. Line 5 holds the first salt concentration 69 mM in this case. The first regression equation starts at line 6 for all sequences of length 10 bp, and the next two numbers are the a_0 and a_1 coefficients of Eq. (4.1), that is

$$T_p(N = 10, [\mathrm{Na}^+] = 69) = -39.4819 + 30.0373\tau, \tag{8.1}$$

At line 2 we see the letter `b` which flags the start of the regression parameters which are length dependent. There will be one line for each salt concentration. At line 3 we see the coefficients for 69 mM corresponding to $b_{0,0}, b_{1,0}, b_{1,0}$ and $b_{1,1}$ of Eq. (4.2),

$$a_0([\mathrm{Na}^+] = 69) = -86.7785 + 48.7698N^{1/2}, \tag{8.2}$$
$$a_1([\mathrm{Na}^+] = 69) = 15.3185 - 6.02284N^{1/2}. \tag{8.3}$$

these equations are necessary to calculate the a_0 and a_1 coefficients for sequence length which are not in the regression file.

8.2 .dat melting temperatures and melting index results

One of the main result files has the extension `.dat` and typically contains the melting temperatures and melting index calculated for each sequence. Here is an example (output of 9.2.2).

examples/verify/predict-2.dat

```
1 Main/Complementary alpha salt_concentration species_concentration temperature.measured ...
2 TACTAACATTAACTA/ATGATTGTAATTGAT 4 69 2 0 0 34.222 0 0 0 0 0 0 2 6.006 2.45071 0 ...
3 ATACTTACTGATTAG/TATGAATGACTAATC 4 69 2 0 0 35.8269 0 0 0 0 0 0 2 6.318 2.51356 0 ...
4 GTACACTGTCTTATA/CATGTGACAGAATAT 4 69 2 0 0 39.712 0 0 0 0 0 0 2 7.106 2.66571 0 ...
```

The first line (which is too long to be shown completely) identifies each column. Importantly, the 7th column is the predicted temperature and the last column is the melting index τ. The 8th column is Z_y and and the 9th column is $-kT \ln Z_y$, in this specific case they are zero as those are not calculated with `-res=prediction`.

8.3 .dat average opening if used with -res=averagey

8.3.1 Single sequence

If a single sequence is specified using `-seq` (4.1.22) and optionally with `-cseq` (4.1.23) the file will be arranged column-wise

open-1.dat

```
0 1.30847
1 0.875882
2 0.848514
3 0.591862
4 0.635131
```

Shown are the first few lines of the result from example 9.6. The first column is the base pair position and the second column is the average opening $\langle y \rangle$, results are given in Ångstrom.

8.3.2 Multiple sequences

Multiple sequences should be arranged in one or more files, and the file names should be passed through the `-data` (4.1.4) option. In this case the average opening will be arranged row-wise.

8.4 .ver quality of the prediction

The file with extension `.ver` is a short file intended to show how close the predicted melting temperatures are when compared to the experimental melting temperatures. This file only make sense if a data file with the experimental melting temperatures was given.

examples/verify/np2009-1-69.ver

```
average diff_deviation sqr_diff sqrt_diff2 relative_sqr_diff N
Tm 0.811472 0.771594 115.354 1.11975 0.0806365 92
 prediction method=2
```

where the first number average difference in melting temperature prediction

$$\langle \Delta T \rangle = \frac{1}{N} \sum_{i=1}^{N} |\Delta T_i| \,, \tag{8.4}$$

the second is the standard deviation of ΔT_i,

$$\delta(\Delta T) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} [\Delta T_i - \langle \Delta T \rangle]^2} \tag{8.5}$$

the third is

$$\chi^2 = \sum_{i=1}^{N} [T_i - T_i'(P_k)] \,, \tag{8.6}$$

and the 4th is

$$\langle \Delta T \rangle_{RMS} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} [\Delta T_i]^2} = \sqrt{\frac{\chi^2}{N}}.' \tag{8.7}$$

and the last the relative squared difference

$$\left\langle \frac{\Delta T}{T} \right\rangle_{\text{RMS}} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \frac{[\Delta T_i]^2}{T_i}}, \tag{8.8}$$

8.5 .usedpar summary of parameters read and used

If `-printusedpar=1` (4.1.21) is given, TfReg will generate an additional file with extension `.usedpar` which shows which parameters were read from which file, and how many times they were used.

The rules for files of type `.usedpar` are summarized here

```
parameter value [* value:FILE3 <> value:FILE2 = value:FILE1] (N)
```

The files are shown in reverse order from how they were read, that is, they were read in order `FILE1`, `FILE2` and last `FILE3`. The specific value of the parameter is shown in front of the file name `value:FILE3`. The signs indicate if the values were the same as of the previous file (`=`) or different (`<>`), and (`*`) means that the parameter was changed internally, after reading all files. The number of times this parameter was called internally is given by `N`, although this reflects only the number of times it was searched in the database and not the number of times it was employed in equations. However, it is accurate when displaying zero meaning it was really never used.

File of type .usedpar

```
parameters
*:harmonic.theta 0.01 [0.010000:dna_mm_ci_pb_60.par = 0.010000:dna_pb_60.par] (136)
AA:morse.D 0.00654159 [0.006542:dna_mm_ci_pb_60.par] (17)
TA_AT:harmonic.k 0.019949 [0.019949:dna_mm_ci_pb_60.par <> 0.024157:dna_pb_69.par] (1)
```

In this example it is shown that the parameter `*:harmonic.theta` was first read from file `dna_pb_60.par`, and afterwards `dna_mm_ci_pb_60.par`, the equal sign indicates that the parameter was the same in both files, this parameter was used 136 times. Parameter `AA:morse.D` was read from `dna_mm_ci_pb_60.par` and used 17 times. For parameter `A_AT:harmonic.k` it was read first in `dna_pb_69.par` and again in `dna_mm_ci_pb_60.par` but with a different value as indicated by the `<>` sign

8.6 Matrix files if used with -matrix=

If you specify a matrix directory say, `-matrix=open-1` (4.1.5), the program will create files containing all matrices used in the calculation:

Matrix files

```
dna-pb-69-CG_CG-270-A.AT_AT dna-pb-69-CG_CG-270-A.AT_CG dna-pb-69-CG_CG-270-A.AT_GC
dna-pb-69-CG_CG-270-A.AT_TA dna-pb-69-CG_CG-270-A.CG_AT dna-pb-69-CG_CG-270-A.CG_CG
dna-pb-69-CG_CG-270-A.CG_GC dna-pb-69-CG_CG-270-A.GC_AT dna-pb-69-CG_CG-270-A.GC_CG
dna-pb-69-CG_CG-270-A.TA_AT dna-pb-69-CG_CG-270-C.AT_AT dna-pb-69-CG_CG-270-C.AT_CG
dna-pb-69-CG_CG-270-C.AT_GC dna-pb-69-CG_CG-270-C.AT_TA dna-pb-69-CG_CG-270-C.CG_AT
dna-pb-69-CG_CG-270-C.CG_CG dna-pb-69-CG_CG-270-C.CG_GC dna-pb-69-CG_CG-270-C.GC_AT
dna-pb-69-CG_CG-270-C.GC_CG dna-pb-69-CG_CG-270-C.TA_AT dna-pb-69-CG_CG-270-eigenvalues
dna-pb-69-CG_CG-270-eigenvectors dna-pb-69-CG_CG-270-Y
```

The first part of the matrix file name `dna-pb-69` is the parameter identifier which you will find in the first line of the file `dna-pb-69.par`. The next field `CG_CG` refers to the base pair used as expansion basis (see option `-expand`). Next comes the temperature at which the calculation was performed (in this case 270 K). The matrix type is given by the last letter or identification and can be either `A` [Eq. (64) of Ref. 34] or `C` [Eq. (56) of Ref. 34]. The extensions like `AT_AT` refer to the nearest neighbour base pairs, there will be one for each type of nearest neighbour present in the sequence. The matrices labeled `eigenvalues` and `eigenvectors` are self-explanatory and represent the result of the diagonalization. Finally the matrix ending in `Y` is that of Eq. (60) of Ref. 34.

9 Examples

The example scripts are located in /usr/share/TfReg/examples (or /usr/share/tfreg/examples). In /usr/share/TfReg/examples/verify (or /usr/share/tfreg/examples/verify) you will find the output of some of these script which will allow you to check if your installed version of TfReg is working properly.

9.1 Task: given a set of melting temperatures, find the regression parameters

This scenario appears when you have some parameters for the hydrogen bond and perhaps stacking interaction and you wish to know how close these may get to experimental melting temperatures. First you will need to calculate the regression parameters which you may later use to calculate melting temperatures for untested sequences.

What you will need:

1. A set of experimental melting temperatures (for example data/owczarzy04-69.dat)
2. A set of parameters (for example dna_pb_69.par)

9.1.1 Results of Ref. 47

In this example we take the parameters which were calculated by the minimization procedure, stored in file dna_pb_X.par (X being 69, 119, 220, 621 or 1020), and calculate the regression parameters which are going to be stored in file np2009-X.

examples/np2009-1.sh

```
tfreg -cutoff=10 -int=-1:200/400 -m=pb -pbc=0 -pm=2 -res=regression -t=370 -v=1 ...
-par=../data/dna_pb_69.par -data=../data/owczarzy04-69.dat

tfreg -cutoff=10 -int=-1:200/400 -m=pb -pbc=0 -pm=2 -res=regression -t=370 -v=1 ...
-par=../data/dna_pb_119.par -data=../data/owczarzy04-119.dat

tfreg -cutoff=10 -int=-1:200/400 -m=pb -pbc=0 -pm=2 -res=regression -t=370 -v=1 ...
-par=../data/dna_pb_220.par -data=../data/owczarzy04-220.dat

tfreg -cutoff=10 -int=-1:200/400 -m=pb -pbc=0 -pm=2 -res=regression -t=370 -v=1 ...
-par=../data/dna_pb_621.par -data=../data/owczarzy04-621.dat

tfreg -cutoff=10 -int=-1:200/400 -m=pb -pbc=0 -pm=2 -res=regression -t=370 -v=1 ...
-par=../data/dna_pb_1020.par -data=../data/owczarzy04-1020.dat

```

9.1.2 Results of Ref. 27

In this example we take the parameters which were calculated by the minimization procedure, stored in file stat_jb_owczarzy04-69.par, and calculate the regression parameters which are going to be stored in file epl2011-69. The file var_jb2_owczarzy04_init.par contains the initial parameters which were used in the minimization procedure, most parameters will be superseded by the contents of stat_jb_owczarzy04-69.par.

examples/epl2011-69.sh

```
tfreg -cutoff=10 -int=-1:200/400 -m=jb -o=epl2011-69 \
-par=../data/dna_jb_69.par -data=../data/owczarzy04-69.dat \
-pbc=0 -pm=2 -res=regression -t=370 -v=1
```

9.2 Task: prediction of DNA melting temperatures

You would like to predict melting temperatures of some nucleotide sequence for which you have no experimental data.

What you will need:

1. A set of parameters (for example `data/dna_pb_69.par`)
2. A file with calculated regression parameters (for example `data/np2009-1-69.reg` for salt concentration of 69 mM)

9.2.1 Single sequence example

If you want to predict the melting temperature of just one sequence the easiest is to specify the sequence as a command argument `-seq` as in the following example.

examples/predict-1.sh

```
tfreg -cutoff=10 -int=-1:200/400 -m=pb -pbc=0 -pm=2 -t=370 -v=1 -o=predict-1 \
-par=../data/dna_pb_69.par \
-reg=../data/np2009-1-69.reg -salt=69 \
-res=prediction \
-seq=ACAGCGAATGGACCTACGTGGCCTT
```

9.2.2 Multiple sequence example

If you want to predict the melting temperature of many sequences it is advisable to edit a simple file like this:

data/example2.dat

```
temperature
TACTAACATTAACTA ATGATTGTAATTGAT 0 69 0
ATACTTACTGATTAG TATGAATGACTAATC 0 69 0
GTACACTGTCTTATA CATGTGACAGAATAT 0 69 0
```

the 69 refers to the salt concentration at which you want to predict these temperatures. Note that the second column is the complementary sequence of the first column. The way to run this example is as follows:

examples/predict-2.sh

```
tfreg -cutoff=10 -int=-1:200/400 -m=pb -pbc=0 -pm=2 -res=prediction -t=370 -v=1 ...
-par=../data/dna_pb_69.par \
-reg=../data/np2009-1-69.reg \
-res=prediction \
-data=../data/example2.dat
```

The result file is this:

examples/verify/predict-2.dat

```
Main/Complementary alpha salt_concentration species_concentration temperature.measured ...
TACTAACATTAACTA/ATGATTGTAATTGAT 4 69 2 0 0 34.222 0 0 0 0 0 0 2 6.006 2.45071 0 ...
ATACTTACTGATTAG/TATGAATGACTAATC 4 69 2 0 0 35.8269 0 0 0 0 0 0 2 6.318 2.51356 0 ...
GTACACTGTCTTATA/CATGTGACAGAATAT 4 69 2 0 0 39.712 0 0 0 0 0 0 2 7.106 2.66571 0 ...
```

where the 7th column is the predicted temperature and the 8 column is the melting index τ.

9.3 Task: prediction of DNA melting temperatures with different salt concentrations

If you need to predict salt concentrations which differ from the ones currently provided you will need to generate a new *regression file*. To ease this task we provided a Perl script which does this regression for you called `tfreg-salt-regression.pl`. To use it you will need to provide some existing regression files for different salt concentrations which will use the regression equation (4.3). Here is an practical example where we use 5 files containing different salt concentration for which we generate a new regression file for a salt concentration of 50 mM.

Example usage of tfreg-salt-regression.pl

```
tfreg-salt-regression.pl 50 \
"np2009-1-69.reg,np2009-1-119.reg,np2009-1-220.reg,np2009-1-621.reg,np2009-1-1020.reg" \
new50.reg
```

the first argument is the new salt concentration, then a list of files comma-separated (the order is unimportant), and the last is the name of the new file. And here is an example script showing how to use the new regression file.

examples/predict-salt50.sh

```
tfreg -cutoff=10 -int=-1:200/400 -m=pb -pbc=0 -pm=2 -t=370 -v=1 -o=predict-salt50 ...
-par=../data/dna_pb_69.par \
-reg=../data/new50.reg -salt=50 \
-res=prediction \
-seq=ACAGCGAATGGACCTACGTGGCCTT
```

Select the model parameters closest to the salt concentration you need. In the example above, we used 69 mM. The model parameters do not vary much with salt concentration, in fact, the Morse potentials hardly change (see 47).

9.4 Task: prediction of RNA melting temperatures

This is very much the same as predicting the melting temperatures for DNA described in the previous sections. The main difference is that you need to use the additional command parameter `-duplextype=RNA` (4.1.8).

What you will need:

1. A set of parameters for RNA (for example `data/rna_pb.par` from Ref. [32])
2. A file with calculated regression parameters (for example `data/reg_pb_xia98-t1.reg` for salt concentration of 1000 mM, this was published as supplementary tables IV and V of Ref. [32])

In the following example we calculate the melting temperatures of all sequences from Ref. 48.

examples/predict-rna-1.sh

```
tfreg -cutoff=10 -int=-1:200/400 -m=pb -pbc=0 -pm=2 -res=prediction -t=370 -v=1 ...
-duplextype=RNA \
-par=../data/rna_pb.par \
-reg=../data/reg_pb_xia98-t1.reg \
-data=../data/xia98-t1.dat
```

9.5 Task: prediction of melting temperatures DNA containing inosine mismatches

Optimized parameters were calculated in Ref. 19, and pre-calculated regression parameters are given in files

`deoxyinosine_pb_ia.reg deoxyinosine_pb_ic.reg deoxyinosine_pb_ig.reg`
`deoxyinosine_pb_ii.reg deoxyinosine_pb_it.reg`

for IA, IC, IG, II and IT inosine mismatches. The parameter file is given in `deoxyinosine_pb.par` and the sequence data from [49] are given in files

`watkins05ia.dat watkins05ic.dat watkins05ig.dat`
`watkins05ii.dat watkins05it.dat`

Please note that you will need to add the letter 'I' to the list of recognized characters with

`-dict=I:I`

see section 4.1.9 for further information.

9.6 Task: calculating the average opening $\langle y \rangle$

Using the option `-res=averagey` you will obtain the average opening $\langle y \rangle$ as a function of nucleotide position (see Ref. 34). Please note that the temperatures used here are unrelated to the predicted melting temperatures of section 9.2, and that for short sequences those temperatures may be unrealistically small. Typically you will want to use this for a qualitative study of localized helix opening.

What you will need:

1. A set of parameters (for example `data/dna_pb_69.par`)

9.6.1 Calculating $\langle y \rangle$ at a given temperature

examples/open-1.sh

```
tfreg -matrix=open-1 -cutoff=10 -int=-1:200/400 -m=pb -pbc=0 -t=270 -v=1 \
-o=open-1 \
-par=../data/dna_pb_69.par -salt=69 -res=averagey \
-seq=ACAGCGAATGGACCTACGTGGCCTT
```

the results will be in the file with extension `.dat`.

9.6.2 Calculating $\langle y \rangle$ for a range of temperatures

Here we use a simple bash loop to loop over the temperatures, each results is stored in a file `open-2-T.dat` where `T` is the temperature.

examples/open-2.sh

```
for T in {250..300}
do
tfreg -cutoff=10 -int=-1:200/400 -m=pb -pbc=0 -v=1 -salt=69 -res=averagey \
-o=open-2-$T -t=$T \
-par=../data/dna_pb_69.par -seq=ACAGCGAATGGACCTACGTGGCCTT
done
```

This example takes about 6 min to run (intel i7-2620M 2.70GHz).

9.6.3 Using Perl scripts and parallel processing

Sometimes, doing something more complicated with shell scripts can be very cumbersome. For example, a simple `for` loop with non-integer values can be very complicated. In this case, I would recommend spending a few moments to learn the basics of Perl. In the following example we run the average opening for RNA in increments of 0.5 K. Also each instance is send into background to be processed in a sort of poor man's parallel processing.

examples/open-rna-3.pl

```
#!/usr/bin/perl
my $prefix='../data';
$prefix=$ARGV[0] if (exists $ARGV[0]);
system("mkdir -p open-rna-3");
my $common='-cutoff=80 -int=-1:30/400 -m=pb -pbc=0 -v=1  -res=averagey -matrix=open-rna-3';

for (my $T=280; $T < 320; $T += 0.5)
  {
  my $fT=sprintf("%.1f",$T); #here we format $T with 1 decimal place
  my $com="tfreg $common -o=open-rna-3/open-rna-3-$fT  -t=$fT -duplextype=RNA -par=$prefix/rna_pb.par";
  my $seq='-seq=GUGCCCAUUUAGGGUAUAUAUGGCCGAGUGAGCGAGCAGGAUCUCCAUUUUGACCGCAAAUUUGAACG';
  system("$com $seq");# . ' < /dev/null &> open-rna-3/open-rna-3-$T.echo &');
  }
```

Note in this example a `-cutoff=80` is used, that is, all matrix multiplications will be 80×80. You may reduce the processing time by using a smaller value such as `-cutoff=10` but for higher temperatures the loss of precision is considerable.

10 Files included in the package

10.1 Sequences, melting temperatures and model parameters

Folder /usr/share/data/

This folder contains the input data such as parameter files (extension `.par`) and melting temperature data (extension `.dat`)

Melting temperatures

`chen12_1000.dat` RNA sequences with GU mismatches from Ref. [50]

`moreira15cy3.dat` DNA with Cy3 from Ref. [46]

`moreira15cy5.dat` DNA with Cy5 from Ref. [46]

`tna-tm.dat` DNA-TNA hybrid sequences from Refs. 12, 51, 52

`owczarzy04-*.dat` DNA at *=69, 119, 220, 621 and 1020 mM [Na^+] from Ref. [53]

`rna-dna-1000.dat` DNA-RNA from Refs. [13, 15, 54]

`watkins05*.dat` for DNA with deoxyinosine, *=ia,ic,ig,ii and it from Ref. [49]

`xia98-t1.dat` RNA from table 1 of Ref. [48]

`dna_mm_60_tm.dat` DNA mismatches at 60 mM [10]

`ferreira19_*_f1.dat` RNA with logarithmic grouping factor $f = 1$, [Na^+] *=71, 121, 221, and 621 mM, adapted from Ref. [55].

`owczarzy12-Mg-*.dat` DNA at *=0.5, 1.5, 3, 10, 20 and 125 mM [Mg^{2+}] from Ref. [56]

`owczarzy12-MgK-*.dat` DNA at 50 mM [K] and *=0.5, 1.5, 10 and 20 M [Mg^{2+}] from Ref. [56]

`bustos-Mg-*.dat` DNA at *=0.5, 1.5, 3, 10, 20 and 125 mM [Mg^{2+}] from Ref. [3]

`fakhfakh15-t1.dat` LNA (non-mismatched) from table 1 of [57], W=+A R=+T N=+G Z=+C.

`fakhfakh15-t2.dat` LNA (non-mismatched) from table 2 of [57], W=+A R=+T N=+G Z=+C.

`you06.dat` LNA (non-mismatched) [58], W=+A R=+T N=+G Z=+C.

`mctigue04.dat` LNA (non-mismatched) [59],W=+A R=+T N=+G Z=+C.

`silva22-cc-ag.dat` DNA with C-Ag^+-C from Refs. [60–69]

`silva22-cc-ctrl.dat` DNA control sequences related to `dna-c-ag-c.dat` from Refs. [60–69]

`silva22-tt-hg.dat` DNA with T-Hg^{2+}-T from Refs. [62, 65–68, 70–72, 72–76]

`silva22-tt-ctrl.dat` DNA control sequences related to `dna-t-hg-t.dat` from Refs. [62, 65–68, 70–72, 72–76]

Model parameters

`-model=pb` PB model parameters

- `lna-dna-pb-1021.par` for DNA+LNA/DNA (DLD) duplexes at 1021 mM of Na^+ [5]
- `dna_pb_mg-*.par` for DNA at *=0.5, 1.5, 3, 10, 20 and 125 mM [Ma^{2+}] [3]
- `dna_pb_mgk-*.par` for DNA at 50 mM [K] and *=0.5, 1.5, 10 and 20 M [3]
- `domljanovic20-dld.par` for DNA+LNA/DNA (DLD) duplexes [8]
- `domljanovic20-dlr.par` for DNA+LNA/RNA (DLR) duplexes [8]
- `tna-1000.par` for RNA [12]
- `rna-dna-pb-1000.par` for DNA/RNA [7]
- `rna_pb_bulge_group1.par` type I RNA bulges [17]
- `dna_pb_cy3.par` for DNA with Cy3 [16]
- `dna_pb_cy5.par` for DNA with Cy5 [16]
- `dna_tpb_*.par` DNA with terminal parameters at *=69, 119, 220, 621 and 1020 mM [Na^+] [11]
- `deoxyinosine_pb.par` for deoxyinosine [19]
- `dna_pb_*.par` DNA at *=69, 119, 220, 621 and 1020 mM [Na^+] [47]
- `rna_pb.par` RNA at 1000 mM [32].
- `rna_pb_*_un_f1.par` parameters of type uniform (UN) for RNA with logarithmic grouping factor $f = 1$, at [Na^+] *=71, 121, 221, and 621 mM [55].
- `rna_pb_*_un.par` parameters of type uniform (UN) for RNA with logarithmic grouping factor $f = 1$, at [Na^+] 1021 mM [55].
- `rna_pb_*_ti_f1.par` parameters of type terminal/internal (T/I) for RNA with logarithmic grouping factor $f = 1$, at [Na^+] *=71, 121, 221, and 621 mM [55].
- `rna_pb_*_ti.par` parameters of type terminal/internal (T/I) for RNA with logarithmic grouping factor $f = 1$, at [Na^+] 1021 mM [55].
- `dna_pb_tt_ag.par` parameters for metal mediated T-Hg^{2+}-T mismatches [4].
- `dna_pb_cc_ctrl.par` parameters from control sequences related to `dna-t-hg-t.par` [4].
- `dna_pb_cc_ag.par` parameters for metal mediated C-Ag^+-C mismatches [4].
- `dna_pb_cc_ctrl.par` parameters from control sequences related to `dna-c-ag-c.par` [4].

`-model=dpb` DPB model parameters

- `peyrard09b-dbp.par` parameters used in Fig. 5.2 of this manual and Fig. 7 of Ref. [30].

`-model=jb` JB model parameters

- `dna_jb_*.par` DNA at *=69, 119, 220, 621 and 1020 mM [Na^+] [27]

`-model=pcla` PCLA model parameters

- `peyrard09.par` parameters used in Fig. 5.2 of this manual and Fig. 4 of Ref. [29].
- `peyrard09-b0.par` parameters setting $b = 0$ used in Fig. 5.2 of this manual and Fig. 4 of Ref. [29].

`-model=pclj` PCLJ model parameters

- `peyrard09b.par` parameters used in Fig. 5.2 of this manual and Fig. 7 of Ref. [30].

`-model=trmf` TRMF model parameters

- `tapiarojo10.par` parameters used in Fig. 5.3 of this manual Fig. 1 of Ref. [31].
- `tapiarojo10-nobarrier.par` parameters with $G = 0$ used in Fig. 5.3 of this manual Fig. 1 of Ref. [31].

A System messages

The documentation of system messages is still work in progress. Therefore, during normal operations of `TfReg` you may encounter many that are not documented here, however most of these messages are hopefully self-explanatory.

Messages are separated in groups, see sections below, and are identified by a unique uppercase code starting and ending with a colon as in the following example

```
:INFORPF:Reading parameters from init_s96.par
:WNSEEV: Warning  no specific Enthalpy or Entropy values, falling back to generic values
:INFOSRSC: Setting reference species concentration to 100 from sequence CCGG
```

The format of the message code should allow you to parse output files with system utilities like `grep` as well as locate sections in the source code where these messages occur.

A.1 Informational messages `INFO`

These type of messages are sent to `stdout`, most of these messages are self-explanatory.

- `INFOAEDF` informs that all debug flags were activated.
- `INFOEDF` informs that a specific debug flag was activated.
- `INFODFREG` informs that a debug flag was recognized as a regular expression.
- `INFODFNF` the given debug flag does not exists.
- `INFOREDENV` shows options that were read from the environment variable `TFREG`.
- `INFORPF` shows the parameter file being read, `-par` (4.1.3).
- `INFOSBSCT` reports the first salt concentration read from file, `-data` (4.1.4).
- `INFOSPSCT` shows that a parameter of type `Na+:concentration` was read.
- `INFOSRSC` reports the first total strand concentration read from file, `-data` (4.1.4).
- `INFOTRRF` informs how many trimer rules were read.

A.2 Debug messages `D`

Debug messages are only shown if explicitly set with the `-debug` (4.1.26) option which are printed to `stderr`. Multiple debug flags can be set with regular expressions, for example `-debug=DINN.*` (4.1.26) activates all flags that start with `DINN`.

- `ALL` activate all existing debug flags, make sure you redirect `stderr` to a file. Shows a message of type `INFOAEDF`.
- `DMOP_HCF` shows additional factors of Eq. 5.25.
- `DOPT_SHEQ` shows how options are interpreted in the command line or environment variable.
- `DREG_FSRCRNAN` informs about problems in the linear regression algorithm.

A.3 Warning messages W

Warning messages are sent to **stderr**, pay attention to these alerts, they were not considered errors but may affect the outcome.

- **WRIIWR** an internal representation of a nucleotide was changed as a result of using **-dict** (4.1.9).
- **WROVNV** a parameter was read again from some file with a different value, **-par** (4.1.3).
- **WUBPG** could not find a parameter for a context-dependent base-pair, but found a context-independent parameter and is using it. See section 6.4.

A.4 Error messages ERR

Error messages are printed to **stderr** after which the program terminates, that is, these are conditions that prevent the **TfReg** from continuing. Also printed is the location in the source file where this error occurred, which may be used for debugging. An error message looks typically like this:

```
:ERRNAPSLA:../../base/options/Options.cpp@126: Error No arguments provided,
showing list of available arguments and exiting
Program terminating
```

- **ERRNAPSLA** No options were read from command line. In this case **TfReg** simply prints all available options and exits.
- **ERRMOCYP** A required 'mandatory' option is missing.

  General cause: usually you will have just forgotten or misspelled an option, please check the documentation or study some examples.

  Special situation: Options for **TfReg** can be 'optional' or 'mandatory'. Certain options are always 'mandatory', however certain options may switch from 'optional' to 'mandatory' depending which previous options were given. Sounds confusing? Lets see an example. Consider two hypothetical option **-x** and **-y**, both are optional but one depends on the other. Therefore, if you do not provide any of them the program will not complain. But if you **do** provide one, you also **need** to provide the other. Meaning: if you specify **-x** then you also need to specify **-y**. In which case **-y** becomes 'mandatory', and if you do not provide **-y** ends with **ERRMOCYP**.
- **ERRCDINAN** diagonalization can not be performed if any of the values are **NaN** (not-a-number) or **inf** (infinite). This typically happens when the Hamiltonian becomes negative, in which case the exponential grows and may become **inf**. One way to correct this is to chose your parameters more carefully such that the Hamiltonian does not get negative or choose smaller interaction limits **-int** (4.1.14) and see what happens
- **ERRNOSEQR** no information was found in the sequence datafiles that were provided with **-data** (4.1.4) and **TfReg** was left with no sequence data to work with. See section 7 on how to format this type of file.
- **ERRNPF** after reading in all sequences, the program was unable to find a parameter, not even a generic one. In this case you typically need to revise the parameter files given in **-par** (4.1.3) and check that they contain the all parameters needed. If your parameter files contain everything you think you need and you still get this messages, we suggest revising your sequences which may have been mistyped (a very common occurrence, unfortunately). In this case we suggest running **-res=nncheck** (4.1.6) and check that resulting base-pairs and nearest neighbours are consistent.
- **ERRUTPL** sequence data file given in **-data** (4.1.4) is in a format that cannot be interpreted correctly. See section 7 for details on data formats.

A.5 Internal error messages IERR

These type of message has a common **IERR** prefix and they are not user correctable, that is, they are not a matter of choosing different options or missing files. They usually happen when the program finds itself in a condition that should never have occurred, regardless of what the user did. Typical situations where these errors may occur if when changes in the C++ compiler cause the program logic to change. For example, if the source was compiled with different compiler options, or with a different compiler. If any of these occur please contact us.

Bibliography

[1] M. Peyrard, A. R. Bishop, Statistical mechanics of a nonlinear model for DNA denaturation, Phys. Rev. Lett. 62 (23) (1989) 2755–2757. doi:10.1103/PhysRevLett.62.2755.

[2] G. Weber, N. Haslam, N. Whiteford, A. Prügel-Bennett, J. W. Essex, C. Neylon, Thermal equivalence of DNA duplexes without melting temperature calculation, Nat. Phys. 2 (2006) 55–59. doi:10.1038/nphys189.

[3] M. I. Muniz, A. H. Bustos, S. Slott, K. Astakhova, G. Weber, Cation valence dependence of hydrogen bond and stacking potentials in DNA mesoscopic models, unpublished (2022).

[4] L. G. Silva, G. Weber, Mesoscopic model confirms strong hydrogen bonding metal mediation for T-Hg^{2+}-T and weaker for C-Ag^{+}-C, unpublished (2022).

[5] I. Ferreira, S. Slott, K. Astakhova, G. Weber, Complete mesoscopic parametrization of single LNA modifications in DNA applied to oncogene probe design, J. Chem. Inf. Model. 61 (2021) 3615–3624. doi:10.1021/acs.jcim.1c00470.

[6] I. Ferreira, T. D. Amarante, G. Weber, Salt dependent mesoscopic model for RNA with multiple strand concentrations, Biophys. Chem. 271 (2020) 106551. doi:10.1016/j.bpc.2021.106551.

[7] E. d. O. Martins, V. B. Barbosa, G. Weber, DNA/RNA hybrid mesoscopic model shows strong stability dependence with deoxypyrimidine content and stacking interactions similar to RNA/RNA, Chem. Phys. Lett. 715C (2019) 14–19. doi:10.1016/j.cplett.2018.11.015.

[8] I. Domljanovic, M. Taskova, P. Miranda, G. Weber, K. Astakhova, Nucleic acid probes – optical and theoretical study reveals new details on strand recognition, Commun. Chem. 3 (2020) 111. doi:10.1038/s42004-020-00362-5.
URL https://www.nature.com/articles/s42004-020-00362-5

[9] M. Rodrigues Leal, G. Weber, Sharp DNA denaturation in a helicoidal mesoscopic model, Chem. Phys. Lett. 755 (2020) 137781. doi:10.1016/j.cplett.2020.137781.
URL https://www.sciencedirect.com/science/article/pii/S00092614203069651

[10] L. M. Oliveira, A. S. Long, T. Brown, K. R. Fox, G. Weber, Melting temperature measurement and mesoscopic evaluation of single, double and triple DNA mismatches, Chem. Sci. 11 (2020) 8273–8287. doi:10.1039/d0sc01700k.
URL https://pubs.rsc.org/en/content/articlelanding/2020/SC/D0SC01700K

[11] I. Ferreira, T. D. Amarante, G. Weber, DNA terminal base pairs have weaker hydrogen bonds especially for AT under low salt concentration, J. Chem. Phys. 143 (2015) 175101. doi:10.1063/1.4934783.

[12] M. I. Muniz, H. H. Lackey, J. M. Heemstra, G. Weber, DNA/TNA mesoscopic modeling of melting temperatures suggest weaker hydrogen bonding of CG than in DNA/RNA, Chem. Phys. Lett. (2020) 137413doi:10.1016/j.cplett.2020.137413.
URL https://www.sciencedirect.com/science/article/abs/pii/S0009261420303286

[13] N. Sugimoto, S.-i. Nakano, M. Katoh, A. Matsumura, H. Nakamuta, T. Ohmichi, M. Yoneyama, M. Sasaki, Thermodynamic parameters to predict stability of RNA/DNA hybrid duplexes, Biochem. 34 (35) (1995) 11211–11216. doi:10.1021/bi00035a029.

[14] P. Wu, S. Nakano, N. Sugimoto, Temperature dependence of thermodynamic properties for DNA/DNA and RNA/DNA duplex formation, Eur. J. Biochem. 269 (12) (2002) 2821–2830.
URL http://www.blackwell-synergy.com/doi/abs/10.1046/j.1432-1033.2002.02970.x

[15] B. Rauzan, E. McMichael, R. Cave, L. R. Sevcik, K. Ostrosky, E. Whitman, R. Stegemann, A. L. Sinclair, M. J. Serra, A. A. Deckert, Kinetics and thermodynamics of DNA, RNA, and hybrid duplex formation, Biochem. 52 (5) (2013) 765–772. doi:10.1021/bi3013005.

[16] P. Miranda, L. M. Oliveira, G. Weber, Mesoscopic modelling of Cy3 and Cy5 dyes attached to DNA duplexes, Biophys. Chem. 230C (2017) 62–67. doi:10.1016/j.bpc.2017.08.007.
URL http://www.sciencedirect.com/science/article/pii/S0301462217302831

[17] E. d. O. Martins, G. Weber, An asymmetric mesoscopic model for single bulges in RNA, J. Chem. Phys. 147 (2017) 155102. doi:10.1063/1.5006948.

[18] T. D. Amarante, G. Weber, Evaluating hydrogen bonds and base stackings of single, tandem and terminal GU in RNA mismatches with a mesoscopic model, J. Chem. Inf. Model. 56 (1) (2016) 101–109. arXiv:http://dx.doi.org/10.1021/acs.jcim.5b00571, doi:10.1021/acs.jcim.5b00571.
URL http://dx.doi.org/10.1021/acs.jcim.5b00571

[19] R. V. Maximiano, G. Weber, Deoxyinosine mismatch parameters calculated with a mesoscopic model result in uniform hydrogen bonding and strongly variable stacking interactions, Chem. Phys. Lett. 631–632 (2015) 87–91. doi:10.1016/j.cplett.2015.04.045.

[20] G. Weber, N. Haslam, J. W. Essex, C. Neylon, Thermal equivalence of DNA duplexes for probe design, J. Phys.: Condens. Matter 21 (2009) 034106. doi:10.1088/0953-8984/21/3/034106.

[21] T. Dauxois, M. Peyrard, A. R. Bishop, Entropy-driven DNA denaturation, Phys. Rev. E 47 (1) (1993) R44–R47. doi:10.1103/PhysRevE.47.R44.

[22] G. Weber, Sharp DNA denaturation due to solvent interaction, Europhys. Lett. 73 (5) (2006) 806–811. doi:10.1209/epl/i2005-10466-6.

[23] M. Joyeux, S. Buyukdagli, Dynamical model based on finite stacking enthalpies for homogeneous and inhomogeneous DNA thermal denaturation, Phys. Rev. E 72 (2005) 051902. doi:10.1103/PhysRevE.72.051902.

[24] S. Buyukdagli, M. Joyeux, Scaling laws at the phase transition of systems with divergent order parameter and/or internal length: The example of DNA denaturation, Phys. Rev. E 73 (5) (2006) 51910. doi:10.1103/PhysRevE.73.051910.

[25] S. Buyukdagli, M. Joyeux, Theoretical investigation of finite size effects at DNA melting, Phys. Rev. E 76 (2) (2007) 021917. doi:10.1103/PhysRevE.76.021917.

[26] S. Buyukdagli, M. Joyeux, Statistical physics of the melting of inhomogeneous DNA, Phys. Rev. E 77 (3) (2008) 031903. doi:10.1103/PhysRevE.77.031903.

[27] G. Weber, Finite enthalpy model parameters from DNA melting temperatures, Europhys. Lett. 96 (2011) 68001. doi:10.1209/0295-5075/96/68001.
URL http://iopscience.iop.org/0295-5075/96/6/68001

[28] T. D. Amarante, G. Weber, Analysing DNA structural parameters using a mesoscopic model, J. Phys.: Conf. Ser. 490 (1) (2014) 012203. doi:10.1088/1742-6596/490/1/012203.
URL http://iopscience.iop.org/1742-6596/490/1/012203

[29] M. Peyrard, S. Cuesta-López, D. Angelov, Experimental and theoretical studies of sequence effects on the fluctuation and melting of short DNA molecules, J. Phys.: Condens. Matter 21 (2009) 034103. doi:10.1088/0953-8984/21/3/034103.

[30] M. Peyrard, S. Cuesta-Lopez, G. James, Nonlinear analysis of the dynamics of DNA breathing, J. Biol. Phys. 35 (1) (2009) 73–89. doi:10.1007/s10867-009-9127-2.

[31] R. Tapia-Rojo, J. J. Mazo, F. Falo, Thermal and mechanical properties of a DNA model with solvation barrier, Phys. Rev. E 82 (3) (2010) 031916. doi:10.1103/PhysRevE.82.031916.

[32] G. Weber, Mesoscopic model parametrization of hydrogen bonds and stacking interactions of RNA from melting temperatures, Nucleic Acids Res. 41 (2013) e30. doi:10.1093/nar/gks964.
URL http://nar.oxfordjournals.org/content/41/1/e30

[33] T. Dauxois, Dynamics of breather modes in a nonlinear "helicoidal" model of DNA, Phys. Lett. A 159 (8-9) (1991) 390–395.

[34] Y.-L. Zhang, W.-M. Zheng, J.-X. Liu, Y. Z. Chen, Theory of DNA melting based on the Peyrard-Bishop model, Phys. Rev. E 56 (6) (1997) 7100–7115. doi:10.1103/PhysRevE.56.7100.

[35] T. Dauxois, M. Peyrard, Entropy-driven transition in a one-dimensional system, Phys. Rev. E 51 (5) (1995) 4027–4040.

[36] K. Drukker, G. Wu, G. C. Schatz, Model simulations of DNA denaturation dynamics, J. Chem. Phys. 114 (1) (2001) 579–590.

[37] F. Zhang, M. A. Collins, Model simulation of DNA dynamics, Phys. Rev. E 52 (4) (1995) 4217–4224.

[38] M. Barbi, S. Lepri, M. Peyrard, N. Theodorakopoulos, Thermal denaturation of a helicoidal DNA model, Phys. Rev. E 68 (2003) 061909. doi:10.1103/PhysRevE.68.061909.

[39] M. Marty-Roda, O. Dahlen, T. S. van Erp, S. Cuesta-López, Improving the mesoscopic modeling of DNA denaturation dynamics, Phys. Biol. 15 (6) (2018) 066001. doi:10.1088/1478-3975/aac61c.

[40] N. F. Ribeiro, E. Drigo, Filho, Thermodynamics of a Peyrard–Bishop one-dimensional lattice with on-site "hump" potential, Brazilian J. Phys. 41 (2011) 195–200.

[41] M. Peyrard, S. Cuesta-López, G. James, Modelling DNA at the mesoscale: a challenge for nonlinear science?, Nonlinearity 21 (6) (2008) T91.

[42] G. G. Slade, Modelos mecânicos para o estudo da dinâmica do DNA, PhD thesis (2013).
URL https://repositorio.unesp.br/handle/11449/110350

[43] K. J. Breslauer, R. Frank, H. Blocker, L. A. Marky, Predicting DNA duplex stability from the base sequence, Proc. Natl. Acad. Sci. USA 83 (11) (1986) 3746–3750. doi:10.1073/pnas.83.11.3746.

[44] J. SantaLucia, Jr., A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics, Proc. Natl. Acad. Sci. USA 95 (4) (1998) 1460–1465. arXiv:http://www.pnas.org/cgi/reprint/95/4/1460.pdf, doi:10.1073/pnas.95.4.1460.
URL http://www.pnas.org/cgi/content/abstract/95/4/1460

[45] D. G. Norman, R. J. Grainger, D. Uhrín, D. M. Lilley, Location of cyanine-3 on double-stranded DNA: importance for fluorescence resonance energy transfer studies, Biochem. 39 (21) (2000) 6317–6324. doi:10.1021/bi992944a.

[46] B. G. Moreira, Y. You, R. Owczarzy, Cy3 and Cy5 dyes attached to oligonucleotide terminus stabilize DNA duplexes: Predictive thermodynamic model, Biophys. Chem. 198 (2015) 36–44. doi:10.1016/j.bpc.2015.01.001.

[47] G. Weber, J. W. Essex, C. Neylon, Probing the microscopic flexibility of DNA from melting temperatures, Nat. Phys. 5 (2009) 769–773. doi:10.1038/nphys1371.

[48] T. Xia, J. SantaLucia, Jr., M. E. Burkard, R. Kierzek, S. J. Schroeder, X. Jiao, C. Cox, D. H. Turner, Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson-Crick base pairs, Biochem. 37 (1998) 14719–14735. doi:10.1021/bi9809425.

[49] J. Watkins, Norman E., J. SantaLucia, John, Nearest-neighbor thermodynamics of deoxyinosine pairs in DNA duplexes, Nucleic Acids Res. 33 (19) (2005) 6258–6267. arXiv:http://nar.oxfordjournals.org/cgi/reprint/33/19/6258.pdf, doi:10.1093/nar/gki918.
URL http://nar.oxfordjournals.org/cgi/content/abstract/33/19/6258

[50] J. L. Chen, A. L. Dishler, S. D. Kennedy, I. Yildirim, B. Liu, D. H. Turner, M. J. Serra, Testing the nearest neighbor model for canonical RNA base pairs: Revision of GU parameters, Biochem. 51 (16) (2012) 3508–3522. doi:10.1021/bi3002709.

[51] K.-U. Schöning, P. Scholz, S. Guntha, X. Wu, R. Krishnamurthy, A. Eschenmoser, Chemical etiology of nucleic acid structure: the α-threofuranosyl-$(3' \rightarrow 2')$ oligonucleotide system, Science 290 (5495) (2000) 1347–1351. doi:10.1126/science.290.5495.1347.

[52] H. H. Lackey, E. M. Peterson, Z. Chen, J. M. Harris, J. M. Heemstra, Thermostability trends of TNA:DNA duplexes reveal strong purine dependence, ACS Synth. Biol. 8 (5) (2019) 1144–1152. doi:10.1021/acssynbio.9b00028.

[53] R. Owczarzy, Y. You, B. G. Moreira, J. A. Manthey, L. Huang, M. A. Behlke, J. A. Walder, Effects of sodium ions on DNA duplex oligomers: Improved predictions of melting temperatures, Biochem. 43 (2004) 3537–3554. doi:10.1021/bi034621r.

[54] P. Wu, Y. Kawamoto, H. Hara, N. Sugimoto, Effect of divalent cations and cytosine protonation on thermodynamic properties of intermolecular DNA double and triple helices, J. Inorg. Biochem. 91 (2002) 277–285. doi:10.1016/S0162-0134(02)00444-0.

[55] I. Ferreira, E. A. Jolley, B. M. Znosko, G. Weber, Replacing salt correction factors with optimized RNA nearest-neighbour enthalpy and entropy parameters, Chem. Phys. 521 (2019) 69–76. doi:10.1016/j.chemphys.2019.01.016.
URL https://www.sciencedirect.com/science/article/abs/pii/S0301010418311200

[56] R. Owczarzy, B. Moreira, Y. You, M. Behlke, J. Walder, Method for estimating a melting temperature of a nucleic acid in buffers containing magnesium ions, US Patent 20,120,123,751 (2012).

[57] K. Fakhfakh, O. Marais, X. B. J. Cheng, J. R. Castañeda, C. B. Hughesman, C. Haynes, Molecular thermodynamics of LNA:LNA base pairs and the hyperstabilizing effect of 5′-proximal LNA:DNA base pairs, AIChE J. 61 (9) (2015) 2711–2731. doi:10.1002/aic.14916.

[58] Y. You, B. G. Moreira, M. A. Behlke, R. Owczarzy, Design of LNA probes that improve mismatch discrimination, Nucleic Acids Res. 34 (8) (2006) e60. doi:10.1093/nar/gkl175.

[59] P. M. McTigue, R. J. Peterson, J. D. Kahn, Sequence-dependent thermodynamic parameters for locked nucleic acid (LNA)-DNA duplex formation, Biochem. 43 (18) (2004) 5388–5405. doi:10.1021/jp073198j.

[60] N. Zimmermann, E. Meggers, P. G. Schultz, A novel silver (I)-mediated DNA base pair, J. Am. Chem. Soc. 124 (46) (2002) 13684–13685. doi:10.1021/ja0279951.

[61] A. Ono, S. Cao, H. Togashi, M. Tashiro, T. Fujimoto, T. Machinami, S. Oda, Y. Miyake, I. Okamoto, Y. Tanaka, Specific interactions between silver (I) ions and cytosine–cytosine pairs in DNA duplexes, Chem. Commun. (39) (2008) 4825–4827. doi:10.1039/B808686A.

[62] A. Ono, H. Torigoe, Y. Tanaka, I. Okamoto, Binding of metal ions by pyrimidine base pairs in DNA duplexes, Chem. Soc. Rev. 40 (12) (2011) 5855–5866. doi:10.1039/c1cs15149e.

[63] H. Torigoe, Y. Miyakawa, A. Ono, T. Kozasa, Thermodynamic properties of the specific binding between Ag+ ions and C: C mismatched base pairs in duplex DNA, Nucleos. Nucleot. Nucl. 30 (2) (2011) 149–167. doi:10.1080/15257770.2011.553210.

[64] T. Funai, M. Aotani, R. Kiriu, J. Nakamura, Y. Miyazaki, O. Nakagawa, S.-i. Wada, H. Torigoe, A. Ono, H. Urata, Silver (I)-ion-mediated cytosine-containing base pairs: Metal ion specificity for duplex stabilization and susceptibility toward DNA polymerases, ChemBioChem 21 (4) (2020) 517–522. doi:10.1002/cbic.201900450.

[65] T. Funai, N. Adachi, M. Aotani, S.-i. Wada, H. Urata, Effects of metal ions on thermal stabilities of DNA duplexes containing homo-and heterochiral mismatched base pairs: comparison of internal and terminal substitutions, Nucleos. Nucleot. Nucl. 39 (1-3) (2020) 310–321. doi:10.1080/15257770.2019.1658116.

[66] H. Urata, E. Yamaguchi, Y. Nakamura, S.-i. Wada, Pyrimidine–pyrimidine base pairs stabilized by silver (I) ions, Chem. Commun. 47 (3) (2011) 941–943. doi:10.1039/C0CC04091F.

[67] O. Nakagawa, A. Fujii, Y. Kishimoto, Y. Nakatsuji, N. Nozaki, S. Obika, 2′O, 4′C-methylene-bridged nucleic acids stabilize metal-mediated base pairing in a DNA duplex, ChemBioChem 19 (22) (2018) 2372–2379. doi:10.1002/cbic.201800448.

[68] A. Fujii, O. Nakagawa, Y. Kishimoto, T. Okuda, Y. Nakatsuji, N. Nozaki, Y. Kasahara, S. Obika, 1,3,9-triaza-2-oxophenoxazine: An artificial nucleobase forming highly stable self-base pairs with three Ag^{I} ions in a duplex, Chem. — Eur. J. 25 (31) (2019) 7443–7448. doi:10.1002/chem.201900373.

[69] I. Schönrath, V. B. Tsvetkov, T. S. Zatsepin, A. V. Aralov, J. Müller, Silver (I)-mediated base pairing in parallel-stranded dna involving the luminescent cytosine analog 1, 3-diaza-2-oxophenoxazine, JBIC, J. Biol. Inorg. Chem. 24 (5) (2019) 693–702. doi:10.1007/s00775-019-01682-1.

[70] H. Torigoe, A. Ono, T. Kozasa, Hg^{II} ion specifically binds with T:T mismatched base pair in duplex DNA, Chem. — Eur. J. 16 (44) (2010) 13218–13225. doi:10.1002/chem.201001171.

[71] O. P. Schmidt, A. S. Benz, G. Mata, N. W. Luedtke, Hg^{II} binds to C–T mismatches with high affinity, Nucleic Acids Res. 46 (13) (2018) 6470–6479. doi:10.1093/nar/gky499.

[72] Y. Miyake, H. Togashi, M. Tashiro, H. Yamaguchi, S. Oda, M. Kudo, Y. Tanaka, Y. Kondo, R. Sawa, T. Fujimoto, et al., $Mercury^{ii}$-mediated formation of thymine-Hg^{II}-thymine base pairs in DNA duplexes, J. Am. Chem. Soc. 128 (7) (2006) 2172–2173. doi:10.1021/ja056354d.

[73] H. Yamaguchi, J. Šebera, J. Kondo, S. Oda, T. Komuro, T. Kawamura, T. Dairaku, Y. Kondo, I. Okamoto, A. Ono, et al., The structure of metallo-dna with consecutive thymine–Hg^{II}–thymine base pairs explains positive entropy for the metallo base pair formation, Nucleic Acids Res. 42 (6) (2014) 4094–4099. doi:10.1093/nar/gkt1344.

[74] T. Funai, C. Tagawa, O. Nakagawa, S.-i. Wada, A. Ono, H. Urata, Enzymatic formation of consecutive thymine–Hg^{II}–thymine base pairs by DNA polymerases, Chem. Commun. 56 (2020) 12025. doi:10.1039/d0cc04423g.

[75] G. Mata, O. P. Schmidt, N. W. Luedtke, A fluorescent surrogate of thymidine in duplex DNA, Chem. Commun. 52 (25) (2016) 4718–4721. doi:10.1039/c5cc09552b.

[76] J. H. Han, S. Hirashima, S. Park, H. Sugiyama, Highly sensitive and selective mercury sensor based on mismatched base pairing with ^{diox}t, Chem. Commun. 55 (69) (2019) 10245–10248. doi:10.1039/c9cc05123f.