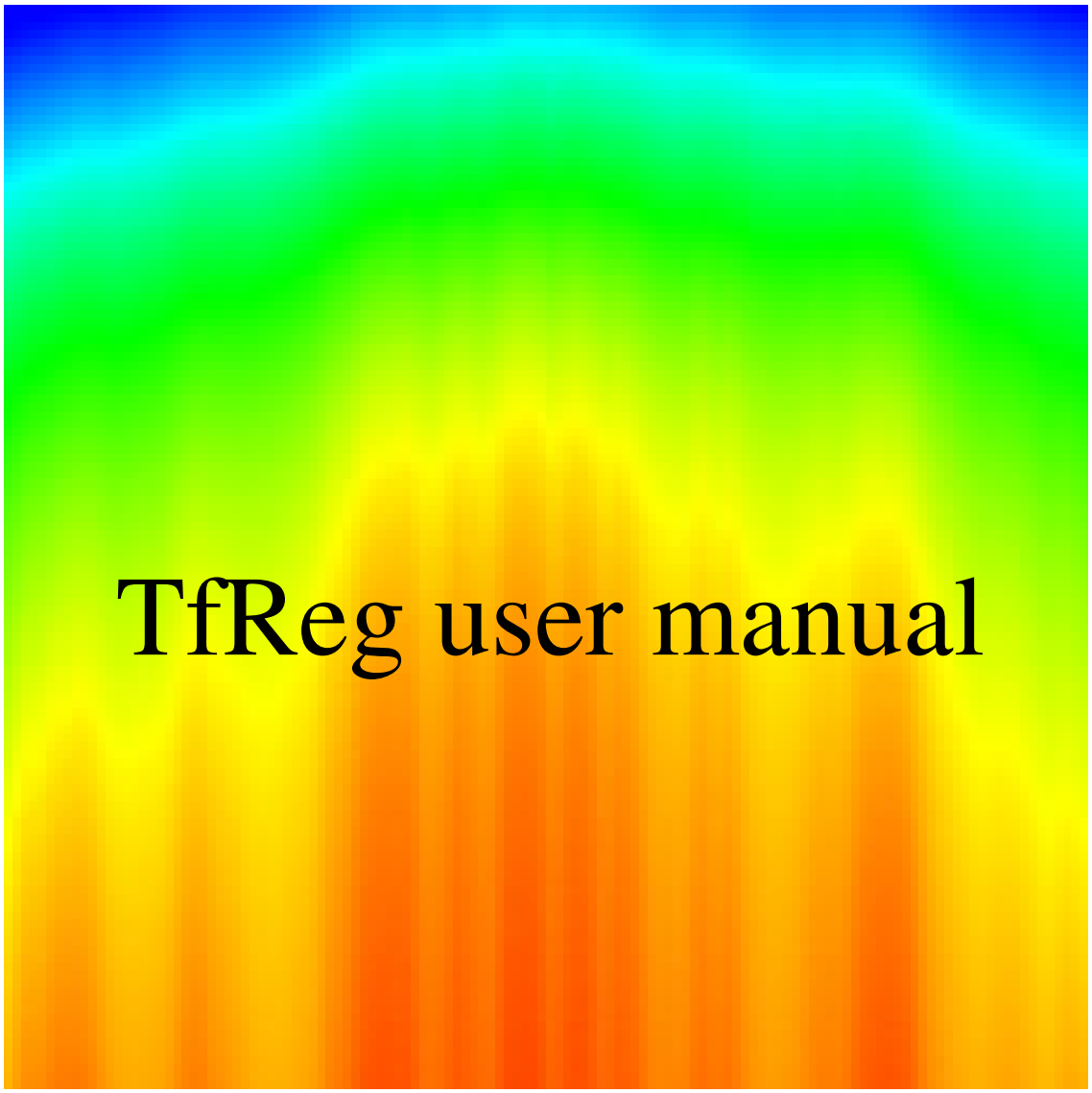

TfReg user manual

TfReg Version 3.0 user manual

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

January 22, 2016

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 oligunucleotide.

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, January 22, 2016

2 Changes

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` 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 [3].
- 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 5.6.
- 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`.

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`, see section 5.5.
- Added new option `-res=nncheck`, see section 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 [4], see section 8.5.
- `.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` from reg files also generated with `-pm=-1`.
- Removed generation of TEX files.
- Adding new option `-dict` 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 8.3.

Version 1.0, November 2012

First release of TfReg.

3 Installation

The easiest way to install is to visit the OpenSUSE Build service (`http://build.opensuse.org`) and search TfReg for your Linux distribution, this should take care of obtaining the correct libraries which are needed for TfReg to run.

3.1 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 and example scripts are to be found in

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

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

`/usr/share/doc/packages/TfReg or /usr/share/doc/packages/tfreg`

3.2 Specific instructions

3.2.1 OpenSUSE

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

`http://download.opensuse.org/repositories/home:/drgweber/openSUSE_13.2/`

then search for the package TfReg and select install. If you have an older OpenSUSE then change the last numbers 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

`http://download.opensuse.org/repositories/home:/drgweber/openSUSE_13.2/`

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

`http://download.opensuse.org/repositories/home:/drgweber/openSUSE_13.2/x86_64/TfReg-3.0-1.21.1.x86_64.rpm`

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

`zypper install TfReg-3.0-1.21.1.x86_64.rpm`

3.3 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 the OpenSUSE build service at

`https://build.opensuse.org/package/show?package=TfReg&project=home%3Adrgweber`

or from my personal webpage

`https://sites.google.com/site/geraldweberufmg/tfreg`

Typically the package is called something like `tfreg-3.0.tar.bz2`After unpacking the `tar` package

```
tar -xvjf tfreg-3.0.tar.bz2
```

change into the unpacked folder

```
cd TfReg-3.0
```

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).

3.3.1 Libraries needed for compiling the source code

This software was developed and tested under OpenSUSE Linux 13.2 and depends on some libraries to function properly:

1. `libboost_filesystem libboost_system libboost_regex` `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.

3.4 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".

3.4.1 If you don't have a Linux system

You may copy an OpenSUSE Live system on a CD or onto a USB memory stick and reboot your computer with this system. If you do, proceed as you were using a normal OpenSUSE 13.1 system, following the instructions above. See installation instructions for Live OpenSUSE at `http://software.opensuse.org/131/en`

3.4.2 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.

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 Arguments

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.

4.1.3 -par=<filename>

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>

specifies the input file containing nucleotide sequences and melting temperatures. 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 50000 sequences which will require about 1GB of memory to run.

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 column is the sequence (from $5' \to 3'$) and the second column is the secondary strand (from $3' \to 5'$). 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.

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.

`-res=prediction` predict temperatures for a given set of parameters and a given regression file.

`-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. Note that for short sequences the temperature needs to be unrealistically low, see example 8.6.

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 [5], see section 5.2

`-model=hms` PB model with added solvent potential [6], see section 5.3

`-model=jb` finite enthalpy model [7–11], see section 5.4

`-model=mes` Morse-exact stacking model [12], see section 5.5

`-model=test` Reserved for *your* test models, please see section 5.6 for instructions. Please do not use without implementing the changs

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`.

(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. [13].

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`

4.1.11 -cutoff=<integer number>

This controls the truncation P in Eq. (22) of Ref. 14, 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. 14 and the size M of the matrices.

(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 the `-data` option) 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`.

4.1.17 -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, within a group of length N, to be considered in calculating regression parameters.

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

(example) `-mar=2` accepts at least 2 sequences for any group

4.1.19 -pm=<prediction method number>

Selects the prediction/regression method.

(default) `-pm=2` this considers two equations for regression

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

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

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

where N is the size of the sequence and τ is the 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`).

`-pm=3` 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 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`. Do not use

`-pm=-1` this 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. [13].

4.1.20 -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`

4.1.21 -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 $3' \to 5'$

4.1.22 -salt=<salt concentration>

In the case where a `-reg` 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 8.3.*

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 url/usr/share/tfreg/data) with file extension `.par`. Note that the parameters are specified per nucleotide type and follow a very specific notation described in section 6.

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 and

$$V_{\text{Morse}}(y_i) = D\left(e^{-y_i/\lambda} - 1\right)^2, \tag{5.1}$$

and the nearest-neighbour stacking interaction as a harmonic oscillator

$$w_{\text{harm.}}(y_i, y_{i-1}) = \frac{k}{2}\left(y_i - y_{i-1}\right)^2. \tag{5.2}$$

because of the divergence of the partition function [15] we modified this to

$$w_{\text{harm.}}(y_i, y_{i-1}) = \frac{k}{2}\left(y_i^2 - 2y_i y_{i-1}\cos\theta + y_{i-1}^2\right), \tag{5.3}$$

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
$\frac{k}{2}\left(y_i^2 - 2y_i y_{i-1}\cos\theta + y_{i-1}^2\right)$	k	`harmonic.k`	eV/Å^2	NN
	θ	`harmonic.theta`	rad	NN

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

In 1993 Dauxois, Peyrard and Bishop introduced an anharmonicity term to account for sharp transitions in the original PB model [5],

$$w_{\text{an.}}(y_i, y_{i-1}) = \left[1 + \rho e^{-\alpha(y_i + y_{i-1})}\right] w_{\text{harm.}}(y_i, y_{i-1}), \tag{5.4}$$

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
$\times \frac{k}{2}\left(y_i^2 - 2y_i y_{i-1}\cos\theta + y_{i-1}^2\right)$	θ	`harmonic.theta`	rad	NN
	ρ	`anharmonic.rho`	adimensional	NN
	α	`anharmonic.alpha`	adimensional	NN

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,

$$V(y_i) = V_{\text{Morse}}(y_i) - f_s D \left[\tanh(y_i/\lambda_s) + 1\right], \tag{5.5}$$

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
$-f_s D\left[\tanh[(y_i+y_e)/\lambda_s]+s\right]$	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

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
```

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

The model by Joyeux and Buyukdagli [7–10] introduces a finite stacking enthalphy

$$w_{\text{fin.}}(y_i, y_{i-1}) = \frac{\Delta H}{C}\left[1 - e^{-b(y_i - y_{i-1})^2}\right] + \frac{K_b}{2}\left(y_i - y_{i-1}\right)^2. \tag{5.6}$$

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
```

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
$\frac{k}{2}\left(y_i-y_{i-1}\right)^2 s$	k	`harmonic.k`	eV/Å^2	NN
$\frac{\Delta H}{C}\left[1-e^{-b(y_i-y_{i-1})^2}\right]$	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

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

This Hamiltonian [12] rewrites the torsional 3D Hamiltonian [16] in the notation of the original PB model [1] by setting the angles to zero ($\phi_i = 0$ and $\theta_0 = 0$), we obtain

$$U_{i,i-1} = D(e^{-\sqrt{2}y_i/\lambda} - 1)^2 + k\left(\sqrt{h^2 + \frac{1}{2}(y_i - y_{i-1})^2} - h\right)^2 \tag{5.7}$$

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

For the computational implementation the original PB Morse potential is implemented unchanged and the additional factor $\sqrt{2}$ should be embedded into the `Morse.lambda` parameter. Note that the `harmonic.theta` parameter used for the PB model is unnecessary here.

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

5.6 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 3.3.

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 5.6.2.

5.6.1 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 3.3. To run use `-model=test`.

5.6.2 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 3.3. To run use `-model=test`.

6 Parameter specification

This chapter explains how to write a parameter file.

Let's start with the most simple possible parameter file

data/generic-bp.par

```
initial
*:harmonic.theta   0.01
*:harmonic.k   0.025
*:morse.D 0.0324083
*:morse.lambda 0.350149
```

The first line holds a simple identifier, this identified can be letters, numbers and some other characters but should contain no spaces. This identifier is used when you specify the `-matrix` option.

The second line starts with an asterisk which means that this parameter should apply to any base pair. In case of a DNA sequence this will apply to AT and CG base-pairs. TA base pairs are considered equivalent to TA, and CG are equivalent to GC, from the perspective of the models considered in this manual.

6.1 Generic * parameters

The generic base * specification can be used when a given parameter should be applied to any base pair (see previous example). 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
```

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 [3]. 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` [3].

6.3 NN parameters

Nearest neighbour (NN) parameters follow the same conventions as usually found in linear regression models [17, 18]. 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.

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
```

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
```

6.4 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.*

7 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`.

7.1 .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

$$T_p(N = 10, [\mathrm{Na}^+] = 69) = -39.4819 + 30.0373\tau, \tag{7.1}$$

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),

$$a_0([\mathrm{Na}^+] = 69) = -86.7785 + 48.7698N^{1/2}, \tag{7.2}$$
$$a_1([\mathrm{Na}^+] = 69) = 15.3185 - 6.02284N^{1/2}. \tag{7.3}$$

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

7.2 .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 8.2.2).

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
3 ATACTTACTGATTAG/TATGAATGACTAATC 4 69 2 0 0 35.8269 0 0 0 0 0 0 2 6.318 2.51356
4 GTACACTGTCTTATA/CATGTGACAGAATAT 4 69 2 0 0 39.712 0 0 0 0 0 0 2 7.106 2.66571
```

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.

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

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 8.6. The first column is the base pair position and the second column is the average opening $\langle y \rangle$, results are given in Ångstrom.

7.4 .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

```
0.811472 0.771594 115.354 1.11975 prediction method=2
```

where the first number average difference in melting temperature prediction

$$\langle \Delta T \rangle = \frac{1}{N} \sum_{i=1}^{N} |\Delta T_i| \,, \tag{7.4}$$

the second is the standard deviation of ΔT_i,

$$\delta(\Delta T) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left[\Delta T_i - \langle \Delta T \rangle\right]^2} \tag{7.5}$$

the third is

$$\chi^2 = \sum_i \left[T_i - T_i'(P_k)\right] , \tag{7.6}$$

and the last is

$$\Delta T_{RMS} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} [\Delta T_i]^2} = \sqrt{\frac{\chi^2}{N}}. \tag{7.7}$$

7.5 Matrix files if used with -matrix=

If you specify a matrix directory say, `-matrix=open-1`, 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. 15] or `C` [Eq. (56) of Ref. 15]. 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. 15.

8 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.

8.1 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`)

8.1.1 Results of Ref. 19

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

```

8.1.2 Results of Ref. 11

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
```

8.2 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)

8.2.1 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
```

8.2.2 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/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
ATACTTACTGATTAG/TATGAATGACTAATC 4 69 2 0 0 35.8269 0 0 0 0 0 0 2 6.318 2.51356
GTACACTGTCTTATA/CATGTGACAGAATAT 4 69 2 0 0 39.712 0 0 0 0 0 0 2 7.106 2.66571
```

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

8.3 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=3 -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 19).

8.4 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`.

What you will need:

1. A set of parameters for RNA (for example `data/rna_pb.par` from Ref. [13])
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. [13])

In the following example we calculate the melting temperatures of all sequences from Ref. 20.

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
```

8.5 Task: prediction of melting temperatures DNA containing inosine mismatches

Optimized parameters were calculated in Ref. 4, 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 [21] 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.

8.6 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. 15). Please note that the temperatures used here are unrelated to the predicted melting temperatures of section 8.2, 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`)

8.6.1 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`.

8.6.2 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).

8.6.3 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.

Bibliography

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

[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] T. D. Amarante, G. Weber, Evaluating hydrogen bonds and base stackings of single, tandem and terminal GU in RNA mismatches with a mesoscopic model, Journal of Chemical Information and Modelingdoi:10.1021/acs.jcim.5b00571.

[4] 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.

[5] T. Dauxois, M. Peyrard, A. R. Bishop, Entropy-driven DNA denaturation, Phys. Rev. E 47 (1) (1993) R44–R47.

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

[7] M. Joyeux, S. Buyukdagli, Dynamical model based on finite stacking enthalpies for homogeneous and inhomogeneous DNA thermal denaturation, Phys. Rev. E 72 (2005) 051902.

[8] 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.

[9] S. Buyukdagli, M. Joyeux, Theoretical investigation of finite size effects at DNA melting, Phys. Rev. E 76 (2) (2007) 021917.

[10] S. Buyukdagli, M. Joyeux, Statistical physics of the melting of inhomogeneous DNA, Physical Review E 77 (3) (2008) 031903.

[11] 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

[12] 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

[13] 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

[14] 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.

[15] 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.

[16] M. Barbi, S. Lepri, M. Peyrard, N. Theodorakopoulos, Thermal denaturation of a helicoidal DNA model, Phys. Rev. E 68 (2003) 061909.

[17] 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.

[18] 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.
URL http://www.pnas.org/cgi/content/abstract/95/4/1460

[19] 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.

[20] 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.

[21] 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