EWF User Manual

Contents

1 Installing Dependencies

EWF requires the following components to be run

1. g++ compiler
2. R
3. libconfig library
4. boost library

The following instructions are meant to illustrate how to install the above, and are functional as of the 28th of December 2022. Please note that new releases of the above software might require different installation procedures to the ones below, and thus there is no guarantee that these instructions are up to date nor correct for your specific platform!

1.1 Installing the g++ compiler

1.1.1 On Windows

If g++ is not present on your distribution (you can check this by typing g++ -v in Windows PowerShell, which will return all the information regarding the installed g++ compiler together with its location if g++ is present, otherwise an error will be returned):

1. Download the latest MinGW (mingw-get-setup.exe) from https://osdn.net/projects/mingw/releases/
2. Follow the installation prompts, note down where MinGW is installed (typically this would be C:\MinGW) and in the package selection menu choose the option mingw32-gcc-g++-bin. From the "Installation" drop down menu click on "Apply Changes"
3. Once installation is complete, go to System Properties and under the Advanced tab click on "Environment Variables". Click on the Path field and edit it to include the location where the MinGW bin file was installed to (under a typical installation this would be C:\MinGW\bin)

To ensure that g++ was installed as necessary, re-run g++ -v within Windows PowerShell which should now print out the location and further information regarding the compiler you installed.

1.1.2 On Mac

If g++ is not present on your platform (you can check this by running g++ -dumpversion | cut -f1 -d . within Terminal), download the latest version of Xcode from the Mac App Store (this might take a while!) and install following the prompts. To check that g++ was installed as necessary, re-run g++ -dumpversion | cut -f1 -d . within Terminal.

1.1.3 On Linux

If g++ is absent from your distribution (within Terminal type g++ --version), then from Terminal run sudo apt update followed by sudo apt install build-essential. To check that installation was successful, re-run g++ --version in Terminal. This should work on most Linux distributions, but if it does not please search online for an installation procedure for your specific distribution!

1.2 Installing R

To install R, please visit https://www.r-project.org/, find the correct installer based on your OS and follow the instructions provided.

1.3 Installing the libconfig library

From http://hyperrealm.github.io/libconfig/ download the latest libconfig tarball (found at the bottom of the webpage), and follow instructions 1 through to 5 under "Basic Installation" on https://github.com/hyperrealm/libconfig/blob/master/INSTALL to install the library onto your system.

1.4 Installing the boost library

1.4.1 On Windows and Linux

From https://www.boost.org/users/download/ download the latest version of boost (please note that EWF was written using boost version 1.78.0 and thus we cannot ensure that certain features are not superseded or deprecated in more up to date version of boost - if the latest version produces error on compiling, please download and install version 1.78.0). Once downloaded, extract the files to your desired location on your platform.

1.4.2 On MacOS

Download "HomeBrew" from `brew.sh` (installation instruction provided on webpage), and in Terminal run `brew install boost` for the latest version of `boost`.

2 Compiler Instructions

To configure the compiler instruction, open the file `Makefile` with a text editor. In the LDFLAGS field enter the following within the same line and separated by a single space:

1. Copy and paste the path to which the `boost` download was extracted together with the prefix `-I`
2. Copy and paste the path to where the EWF source code resides together with the prefix `-I`
3. Copy the paste the path to the file `libconfig++.a` (found wherever you have downloaded and installed your `libconfig` library) without any prefix

The above would look something like this:

On Linux:

```
LDFLAGS := -I/home/ubuntu/Desktop/boost_1_78_0
-I/home/ubuntu/Desktop/EWF /usr/local/lib/libconfig++.a
```

On Windows:

```
LDFLAGS := -I"C:\boost_1_78_0" -I"C:\Desktop\EWF"
"C:\Desktop\libconfig++.a"
```

On Mac:

```
LDFLAGS := -I/Users/username/Desktop/boost_1_78_0
-I/Users/username/Desktop/EWF /usr/local/bin/libconfig++.a
```

All other options and flags within `Makefile` should remain untouched.

3 Program Execution

Once the above flags and paths have been included and the `Makefile` saved, the program can be compiled and executed by running `run run.sh` in terminal. This invoke the file `run.sh` which in turn first compiles the C++ files through the use of `make`, and subsequently invokes the compiled program.

Note that the `run.sh` file as distributed is configured to run the horse coat coloration demo example found in Figure 1 of the article and uses the parameters found within `configHorseCoat.cfg` to simulate the requested paths. To alter the parameters of this demo through the configuration file, please see the relevant subsection within Section 4.

To run the program as a diffusion or diffusion bridge simulator, within `run.sh` replace `./main horses` with `./main`. The program will then ask the user whether they wish to simulate draws from the law of a diffusion or diffusion bridge, whether they wish to condition on non-absorption, and further offers the option of computing a truncation to the transition density.

4 Parameter configuration

Recall that EWF returns paths distributed according to the law of a diffusion or diffusion bridge satisfying the following stochastic differential equation

$$dX_t = \frac{1}{2}\left[\sigma X_t(1-X_t)\eta(X_t) - \theta_2 X_t + \theta_1(1-X_t)\right]dt + \sqrt{X_t(1-X_t)}dW_t \tag{1}$$

for $t \geq 0$ with $X_0 \in [0,1]$, where $\eta(x)$ is a finite degree polynomial in x, i.e. $\eta(x) = \sum_{i=0}^{n} a_i x^i$ for $n \in \mathbb{N}$. Such a formulation allows for a wide class of non-neutral regimes including the case of:

1. Genic selection - here $\eta(x)$ is set to be the constant function 1, such that the contribution from selection to the drift component in (1) becomes $\sigma X_t(1-X_t)$ and σ is the only free parameter.
2. Diploid selection - here $\eta(x)$ is typically formulated as $\eta(x) = h + x(1-2h)$ with the free parameter $h \in \mathbb{R}$ determining the relative fitness of the heterozygotes and commonly termed the dominance parameter or degree of dominance. Thus in this case selection contributes a factor of $\sigma X_t(1-X_t)(h + X_t(1-2h))$ to the drift in (1), where we now have 2 free parameters σ and h.

In the general case where $\eta(x)$ is a polynomial with degree $n \in \mathbb{N}$, the selection contribution becomes $\sigma X_t(1-X_t)\sum_{i=0}^{n} a_i {X_t}^i$ and we now have $n+2$ free parameters: σ and $\{a_i\}_{i=0}^{n}$.

The presence or absence of the mutation parameter $\boldsymbol{\theta} = (\theta_1, \theta_2)$ dictates whether the boundary points $\{0,1\}$ are absorbing or entrance/reflecting. Furthermore, whenever $\theta_1, \theta_2 > 0$, the corresponding boundary is reflecting and attainable if $\theta_i < 1$ or entrance (and therefore unattainable) if $\theta_i \geq 1$.

5 Configuring the configuration files

To run EWF, the user needs to input the parameters with which they wish to simulate into the appropriate configuration file, which then supplies this information to the program. To do this, we make use of four separate configuration files: `configHorseCoat.cfg`, `config.cfg`, `configDiffusion.cfg` and `configBridge.cfg`. Below we outline what each configuration file does and contains, and what each field contained within the configuration file expects as an entry. The program runs some preliminary checks at the start, with any detected problems in the provided input being flagged up and printed to terminal. **NB :** Please note that any floating point numbers must be specified with a first decimal place (e.g. 1.0 as opposed to 1), otherwise silent errors will be produced!

5.1 configHorseCoat.cfg

The file `configHorseCoat.cfg` is used to configure the program to produce the horse coat coloration trajectories. It contains the following entries:

- `g_entry` - this specifies the generation time in years.
- `mu_entry` - a 2 dimensional vector that specifies the *pre-limiting* mutation entries, such that $\theta_i := 2 * \texttt{Ne\_entry} * \texttt{mu\_entry}_i$.
- `nonneutral_entry` - a boolean entry that specifies whether selection is present (`true`) or not (`false`).
- `selSetup_entry` - an integer which specifies whether the program should assume: genic selection (in which case the user should set this entry to 0), diploid selection (in which case this entry should be set to 1), or a more general polynomial selection (in which case this quantity should be set to 2). Note that if any other value is supplied, the program will return an error!

- `s_entry` - the pre-limiting selection coefficient, such that $\sigma := 2 * \texttt{Ne\_entry} * \texttt{s\_entry}$
- `Ne_entry` - the effective population size
- `dominance_entry` - this entry is only required when diploid selection is desired and `selSetup_entry` is set to 1. In this case it determines the relative fitness of the heterozygotes.
- `polyDeg_entry` - this entry is only required when general polynomial selection (i.e. not genic nor diploid selection) is desired and `selSetup_entry` is set to 2. In this case it determines the number of polynomial coefficients for the polynomial $\eta(x) = \sum_{i=0}^{n} a_i x^i$ (and thus indirectly the degree of the polynomial).
- `polyCoeffs_entries` - this entry is only required when general polynomial selection (i.e. not genic nor diploid selection) is desired and `selSetup_entry` is set to 2. In this case it determines the polynomial coefficients $\{a_i\}_{i=0}^{n}$ in *increasing order of power*.
- `Absorption_entry` - a boolean entry that specifies whether the diffusion bridge is absorbed upon hitting the boundary (`true`) or whether one should condition on non-absorption (`false`).
- `nEndpoints` - integer specifying the number of observations to be supplied by the user
- `observationSamples_entry` - a vector containing the total number of observed samples at each observation time (note that the size of this vector should match the value provided for `nEndpoints`).
- `observationCount_entry` - a vector containing the number of observed samples having the tracked allele at each observation time (note that the size of this vector should match the value provided for `nEndpoints`).
- `observationTimes_entry` - a vector containing the observation time (note that the size of this vector should match the value provided for `nEndpoints`).
- `nSim_entry` - integer specifying the number of desired trajectories to be simulated.
- `nInterTimes` - number of intermediate times at which draws from the corresponding diffusion bridge should be sampled (this specifies the *total* number of such sampling times over the whole length of the bridge).

NB : if running the horse coat coloration demo, no other configuration files need to be altered as this is the only file necessary for the program to run the example!

5.2 config.cfg

The file `config.cfg` is used to configure the parameters of the Wright–Fisher law one desires to sample from. It contains the following entries:

- `theta_entries` - a 2 dimensional vector specifying the mutation vector, $\theta_i = i^{\text{th}}$ entry of `theta_entries`.
- `nonneutral_entry` - a boolean entry that specifies whether selection is present (set to `true`) or not (set to `false`).
- `sigma_entry` - the selection coefficient σ given in (1).
- `dominance_entry` - this entry is only required when diploid selection is desired and `selSetup_entry` is set to 1. In this case it determines the relative fitness of the heterozygotes.
- `polyDeg_entry` - this entry is only required when general polynomial selection (i.e. not genic nor diploid selection) is desired and `selSetup_entry` is set to 2. In this case it determines the number of polynomial coefficients of the polynomial $\eta(x) = \sum_{i=0}^{n} a_i x^i$ (and thus indirectly the degree of the polynomial).

- `polyCoeffs_entries` - this entry is only required when general polynomial selection (i.e. not genic nor diploid selection) is desired and `selSetup_entry` is set to 2. In this case it determines the polynomial coefficients $\{a_i\}_{i=0}^n$ in *increasing order of power*.

- `selSetup_entry` - an integer which specifies whether the program should assume: genic selection (in which case the user should set this entry to 0), diploid selection (1), or a more general polynomial selection (2). Note that if any value other than 0, 1 or 2 is supplied, the program will return an error!

NB : Whenever EWF is called outside of the horse coat coloration demo (i.e. when it is used as a diffusion or diffusion bridge sampler), `config.cfg` is where all the relevant diffusion parameters present in (1) are specified!

5.3 configDiffusion.cfg

The file `configDiffusion.cfg` is used to configure the program when *diffusion* trajectories are desired. It contains the following entries:

- `Absorption_entry` - a boolean entry that specifies whether the desired diffusion is absorbed on hitting the boundary (`true`) or whether one should condition on non-absorption (`false`).

- `startDiff_entry` - a vector specifying the starting points for the diffusion, where each entry must be within $[0, 1]$.

- `startDiffTime_entry` - a vector specifying the start times for the diffusion.

- `sampleDiffTime_entry` - a vector specifying the sampling times for the diffusion, each entry must be strictly greater than the corresponding `startDiffTime_entry` entry.

- `nSim_entry` - a vector specifying the number of simulations desired for each diffusion.

- `meshSize_entry` - this entry is only necessary when the user desires to further compute an approximation to the transition density. In this case, this specifies a vector of mesh-sizes to be used when calculating the corresponding truncated transition density, with each entry an integer.

NB : If multiple diffusion simulations are desired (say N and sharing the same parameters specified within `config.cfg`), the relevant quantities specified above should be inputted as an N-dimensional vector. Thus all entries `startDiff_entry`, `startDiffTime_entry`, `sampleDiffTime_entry`, `nSim_entry`, and (if necessary) `meshSize_entry` *must* have the same size N. See Section 6.2.1 for an example.

5.4 configBridge.cfg

The file `configBridge.cfg` is used to configure the program when *diffusion bridge* trajectories are desired. It contains the following entries:

- `Absorption_entry` - a boolean entry that specifies whether the desired diffusion bridge is absorbed on hitting the boundary (`true`) or whether one should condition on non-absorption (`false`).

- `nEndpoints` - a vector specifying the number of endpoints contained within simulation, where each entry must be an integer.

- `bridgePoints_entry` - a vector specifying the start and end points for each the diffusion. These should be concatenated into a single vector: $(x_0^{b_1}, x_1^{b1}, \ldots, x_{n_1}^{b1}, x_0^{b2}, \ldots, x_{n_2}^{b2}, \ldots, x_0^{b_k}, \ldots, x_{n_k}^{b_k})$. See Section 6.2.2 for an illustrative example.

- `bridgeTimes_entry` - a vector specifying the time stamps for the corresponding diffusion bridge, such that the first entry corresponds to the time stamp when the diffusion bridge takes the value x_0^{b1}, the second corresponds to the time stamp when the diffusion bridge takes the value x_1^{b1}, etc.

- `nSampleTimes_entry` - a vector specifying the number of sampling times within each subsequent pair of time stamps appearing in `bridgeTimes_entry`. If no samples are requested between a pair of observations, this should still be recorded as a 0 in this vector.
- `sampleTimes_entry` - a vector specifying the sampling times desired listed in increasing order for each bridge
- `nSim_entry` - a vector specifying the number of simulations desired for each diffusion bridge.
- `meshSize_entry` - this entry is only necessary when the user desires to further compute an approximation to the transition density. In this case, this specifies a vector of mesh-sizes to be used when calculating the corresponding truncated transition density, with each entry an integer.

The following relations must be satisfied by the provided input:

1. The number of entries in `bridgePoints_entry`, `bridgeTimes_entry`, `sampleTimes_entry`, `nSim_entry` and (if necessary) `meshSize_entry` should equal the sum of all entries of `nEndpoints`
2. The number of entries in `nSampleTimes_entry` should equal the sum of all entries of `nEndpoints` minus the number of different bridges being simulated

NB : The input format illustrated above allows for multiple bridge simulations - an example explaining this clearly through two examples is provided in Section 6.2.2

6 Examples

6.1 Horse Coat Coloration demo

The provided `configHorseCoat.cfg` is set up to have the following parameters:

- N_e = `Ne_entry` $= 10,000$, g = `g_entry` $= 5$ years, $\boldsymbol{\theta} = (\theta_1, \theta_2) = (0, 0)$ together with `Absorption_entry` set to `false` which therefore imposes conditioning on non-absorption.
- `selSetup_entry` $= 0$, which therefore allows for genic selection where further we have $\sigma = 2 * N_e *$ `s_entry` $= 14$ (and all other fields `dominance_entry`, `polyDeg_entry`, `polyCoeffs_entries` are redundant because we are only interested in genic selection).
- `nEndpoints` $= 6$ with

$$\begin{aligned} \texttt{startDiffTime\_entry} &= (20000, 13100, 3700, 2800, 1100, 500) \\ \text{Total number of observations} &= (10, 22, 20, 20, 36, 38) \\ \text{Count of samples with tracked allele} &= (0, 1, 15, 12, 15, 18) \\ \texttt{bridgePoints\_entry} &= (0, \frac{1}{22}, \frac{3}{4}, \frac{3}{5}, \frac{5}{12}, \frac{9}{19}) \end{aligned}$$

- `nSim_entry` $= 30$ and we construct a vector of sampling times of size `nIntertimes` $= 1000$ consisting of equally spaced time stamps over the interval $(2000, 500)$ years before present.

If one desired to change from *genic* to *diploid* selection:

- `selSetup_entry` should be set to 1
- `dominance_entry` should be set accordingly - note that in the provided file it is currently set to $h = \frac{1}{2}$ which is a special case that reverts to genic selection!

whilst the quantities `polyDeg_entry` and `polyCoeffs_entries` are redundant.
If on the other hand some general polynomial selection is desired, the following modifications are required:

- `selSetup_entry` should be set to 2
- `polyDeg_entry` should be set to the size of the corresponding vector of polynomial coefficients - in the provided file we chose a polynomial of degree 5 (any integer is allowed however we would not recommend running EWF with polynomials of degree higher than 25!)
- `polyCoeffs_entries` should be set to the desired vector of polynomial coefficients *in increasing order of power* - in the provided file the vector is set to (0.5, 0.25, 0.5, 0.25, 0.5, 0.25) which corresponds to the following polynomial $\eta(x)$

$$\eta(x) = 0.5 + 0.25x + 0.5x^2 + 0.25x^3 + 0.5x^4 + 0.25x^5,$$

and in this case `dominance_entry` can be ignored.

6.2 Diffusion and diffusion bridge demos

For both the diffusion and diffusion bridge demos we use the following parameter setups:

- $\boldsymbol{\theta} = \mathbf{0}$
- No selection, so $\sigma = 0$ and thus all input specifying $\eta(x)$ and h can be ignored

Nonetheless if one wants to include selection, the same instructions as those present in the horse coat coloration demo above can be followed verbatim to allow for the various non-neutral setups EWF accommodates for.

6.2.1 Diffusion demo

In the provided `configDiffusion.cfg` we replicate the setups present in Section 7.1 of the Supplementary Information, namely we generate 10,000 samples for the following cases:

Diffusion start time stamp	0	0	0	0	0	0	0	0	0
Diffusion start time value	0	0	0	0.5	0.5	0.5	1	1	1
Sampling time	0.01	0.05	0.5	0.01	0.05	0.5	0.01	0.05	0.5

with `Absorption_entry = false`. Thus we have the following input:

$$\texttt{startDiff\_entry} = (0.0, 0.0, 0.0, 0.5, 0.5, 0.5, 1.0, 1.0, 1.0)$$
$$\texttt{startDiffTime\_entry} = (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0)$$
$$\texttt{sampleDiffTime\_entry} = (0.01, 0.05, 0.5, 0.01, 0.05, 0.5, 0.01, 0.05, 0.5).$$

If one wanted to further include the simulation of a diffusion started at x at time t_0 and sampled at time t, the provided `configDiffusion.cfg` would be modified by adding an additional entry of:

- x at the end of the vector `startDiff_entry`
- t_0 at the end of the vector `startDiffTime_entry`
- t at the end of the vector `sampleDiffTime_entry`

The unconditioned diffusion simulation scenarios found in Section 7.2 of the Supplementary Information can be replicated by setting

- `Absorption_entry = true`
- `startDiff_entry`$= (0.25, 0.25, 0.25, 0.5, 0.5, 0.5, 0.75, 0.75, 0.75)$
- `startDiffTime_entry`$= (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0)$
- `sampleDiffTime_entry`$= (0.05, 0.25, 0.5, 0.05, 0.25, 0.5, 0.05, 0.25, 0.5)$

6.2.2 Diffusion bridge demo

In the provided `configBridge.cfg` we replicate the setups present in Section 7.3 of the Supplementary Information, namely we generate 10,000 samples for the following bridges:

	(t_0, x_0)	(t_1, x_1)	(t_2, x_2)	(t_3, x_3)
Bridge 1	(0,0)	(0.05,0.1)	(0.1,0.25)	
Bridge 2	(0.2,0.1)	(0.3,0.3)	(0.4,0.4)	(0.5,0.5)
Bridge 3	(0,1)	(0.5,0.95)		

In the first bridge we have 3 observations, in the second bridge 4 and in the third bridge only 2, and thus `nEdnpoints` $= (3, 4, 2)$. We also concatenate the above provided start times and values into

$$\texttt{bridgePoints\_entry} = (0.0, 0.1, 0.25, 0.1, 0.3, 0.4, 0.5, 1.0, 0.95)$$
$$\texttt{bridgeTimes\_entry} = (0.0, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.0, 0.5)$$

For each bridge, we have the following sampling times:

	s_1	s_2	s_3
Bridge 1	0.025	0.065	0.085
Bridge 2	0.25	0.35	0.45
Bridge 3	0.1	0.25	0.3

from which we deduce that for the first bridge we need to sample 1 point from within the interval $(0, 0.05)$ and 2 points from within the interval $(0.05, 0.1)$; for the second bridge we sample 1 point from each of the intervals $(0.2, 0.3), (0.3, 0.4)$ and $(0.4, 0.5)$; whilst for the third bridge we sample 3 points from the interval $(0, 0.5)$. Thus we have the following entries for

$$\texttt{nSampleTimes\_entry} = (1, 2, 1, 1, 1, 3)$$
$$\texttt{sampleTimes\_entry} = (0.025, 0.065, 0.085, 0.25, 0.35, 0.45, 0.1, 0.25, 0.3)$$

The unconditioned diffusion bridge simulations in Section 7.3 of the Supplementary Information can be replicated by setting `Absorption_entry = true` as well as

$$\texttt{bridgePoints\_entry} = (0.25, 1.0, 0.5, 0.0)$$
$$\texttt{bridgeTimes\_entry} = (0.0, 0.3, 0.0, 0.5)$$
$$\texttt{nSampleTimes\_entry} = (3, 3)$$
$$\texttt{sampleTimes\_entry} = (0.05, 0.15, 0.25, 0.05, 0.25, 0.45)$$