13CFLUX 2 Reference Guide

Forschungszentrum Juelich, IBG 1: Biotechnology,
Model and Simulation Group

October 1, 2015

13CFLUX 2 Reference Guide
by Forschungszentrum Juelich, IBG 1: Biotechnology, Model and Simulation Group

Contents

1 Overview **1**

2 13CFLUX 2 Reference Pages **3**

2.1 Simulation and Parameter Fitting . . . 3

2.1.1 fwdsim . . . 4

2.1.2 fitfluxes . . . 8

2.1.3 multifit . . . 11

2.1.4 multifitfluxes . . . 13

2.1.5 multifwdsim . . . 16

2.1.6 mcbootstrap . . . 18

2.2 Sampling and Stoichiometry Initialization . . . 19

2.2.1 sscanner . . . 19

2.2.2 ssampler . . . 21

2.2.3 perturb . . . 23

2.2.4 multiperturb . . . 25

2.3 Experimental Design . . . 27

2.3.1 edscanner . . . 28

2.3.2 edopt . . . 30

2.4 Metabolic Network Conversion . . . 34

2.4.1 ftbl2fml . . . 35

2.4.2 sbml2fml . . . 36

2.4.3 fml2sbml . . . 37

2.5 Data Transfer and Report Generation . . . 38

2.5.1 setfluxes . . . 39

2.5.2 setmeasurements . . . 41

2.5.3 fwdsimflt . . . 42

2.5.4 simreport . . . 44

2.5.5 collectfitdata . . . 45

2.5.6 fwdsim2csv . . . 46

2.5.7 hdf5tocsv . . . 47

2.6 Validation and Benchmark . . . 48

2.6.1 fmllint . . . 49

2.6.2 fmlsign . . . 50

2.6.3 benchmark . . . 51

3 Document Formats **53**

3.1 fluxml . . . 53

3.2 fwdsim . . . 67

4 Bibliography **75**

List of Tables

2 13CFLUX 2 Reference Pages

2.1 Format of CSV labeling and flux measurements 41

Chapter 1

Overview

Metabolic Flux Analysis (MFA) based on isotopic labeling experiments is a widely established tool for the determination of reaction rates (fluxes) in metabolic pathways. MFA allows to determine quantitatively all fluxes in the (central) metabolism of micro organisms or higher cells. The metabolic flux maps resulting from this analysis serve to compare different strains of a micro organism in order to diagnose the effect of a genetic manipulation or may even give hints how to further improve the production capabilities of biotechnologically used organisms.

MFA is based on an Isotope Labeling Experiment where isotope-labeled substrates are fed to the cells. Due to the metabolic activity, the isotopes are then distributed all over the metabolic network. Finally, the enrichment of the isotopes is in the intra-cellular metabolite pools tends to an isotopically stationary state, which simply means that constant fractions of labeled and unlabeled atoms are encountered in all metabolic pools. In this state the isotopic enrichment is measured by Mass Spectrometry (MS) or Nuclear Magnetic Resonance (NMR) instruments. From the obtained measurement data the metabolic fluxes are estimated based on a mathematical model of the labeling dynamics in the system. The biological knowledge required for this procedure is the structure of the metabolic network, and the atom transitions in each enzyme-driven reaction step.

The central concept for the modeling of isotope labeling systems is that of an isotopomer of a metabolite. If a certain metabolite has N labeling positions (e.g. carbon atoms) then there are 2ˆN differently labeled species of this metabolite, depending on whether each single labeling position is labeled (13C) or unlabeled (12C). These differently labeled species are commonly called the isotopomers (or isotopologues) of a metabolite. Isotopomers are quantified by isotopomer fractions which must necessarily sum up to one, that is 100%, for each metabolite pool.

During MFA, the isotope labeling experiment is simulated, requiring the determination of at least a subset of the metabolic network's isotopomer fractions (or linear combinations of them). Because the total number of different isotopomers can be quite high, this simulation requires the solution of high dimensional, nonlinear balance equation systems. Basically, the 13CFLUX 2 software represents a highly efficient generator and solver for these equation systems. Finally the metabolic fluxes are determined by solving the inverse problem of MFA, i.e. the flux distribution is determined by an optimization algorithm comparing the simulation results with the measured intra-cellular labeling enrichment and measured extra-cellular flux rates.

This document is intended to give a brief overview over the most important programs in 13CFLUX 2. The list of these applications is complete, however only the most important command line arguments are discussed. A more complete documentation can be found in the corresponding manual pages.

Chapter 2

13CFLUX 2 Reference Pages

2.1 Simulation and Parameter Fitting

fwdsim The main task of the **fwdsim**(1) tool is to simulate an isotope labeling network specification provided in form of a FluxML document (see **fluxml**(5)). **fwdsim**(1) allows to perform a (linearized) statistical analysis in order to judge about the relyability and sensitivity of the current flux distribution. The simulation results and the generated synthetic measurement values are written to a **fwdsim**(5) XML document. Apart from the XML output, fwdsim allows to export various data and symbolic equations to HDF5, MathML, and text documents.

Important command line switches:

`-s` evaluate statistics

`-A` full arbitrary precision solution

`-H` HDF5 export of various data

fitfluxes The progam **fitfluxes**(1) intelligently adjusts the flux distribution of a metabolic network in order to reproducec a set of isotope labeling measurement values. For this purpose **fitfluxes**(1) reads the metabolic network an measurement configuration from a FluxML document (see **fluxml**(5)), analyzes the network's stoichiometry and uses an optimization algorithm for adjusting certain 'free' variables in the flux distribution. Given the resulting new flux distribution the network and the measurements are simulated and the resulting synthetic measurement values are compared to the real ones. This process continues until synthetic and real measurement values agree or a certain termination condition is fulfilled. The resulting flux distribution and the corresponding synthetic measurement values are written to a **fwdsim**(5) XML document.

Important command line switches:

`-O` choice of optimization algorithm

`-P` optimization algorithm properties

`-g` choice of method for gradient computation

multifit The program **multifit**(1) runs a multi-processor, multi-start optimization using **fitfluxes**(1) based on random flux distributions created by the **ssampler**(1) tool. **multifit**(1) performs a simple type of load balancing in order to use the available computational resource as efficient as possible.

Important command line switches:

`-n` number of samples to generate

`-t` number or range of parallel tasks to run

multifitfluxes The program **multifitfluxes**(1) is a parallel implementation of **fitfluxes**(1) (and newer variant of multifit). The output is either a directory containing the result files (i.e., FWDSIM files containing parameter estimation results), or a unified HDF5 file comprising all best fit flux results. In contrast to **multifit**(1) an output folder (or output HDF5) has to be specified and flux distributions (generated by **ssampler**(1)) have to be provided.

Important command line switches:

`-H` Input file containing flux samples (HDF5)

`-f` Output file containing flux samples (HDF5)

`-N` Number of maximum parallel processes

multifwdsim The program **multifwdsim**(1) is a parallel implementation of **fwdsim**(1) for variation studies and sensitivity analyses. As for **fwdsim**(1), FluxML documents are taken as input along with several parameters. The output is a directory containing the FWDSIM files containing simulation results.

Important command line switches:

`-s` evaluate statistics

`-H` Input file containing flux samples (HDF5)

`-N` Number of maximum parallel processs

mcbootstrap The program **mcbootstrap** implements a Monte Carlo bootstrap algorithm for nonlinear statistical assessment of flux confidence regions

Important command line switches:

`-n` Number of samples - random flux distributions

`-m` Number of executed perturbtions

2.1.1 fwdsim

fwdsim — a tool for performing a forward simulation of an isotope labeling network.

Synopsis

`fwdsim` [*`options`*]

Description

Fwdsim is a tool for simulating a FluxML specification of an isotope labeling network including all measurements.

For this purpose **fwdsim** reads the metabolic network and measurement configuration from a FluxML file, analyzes the network's stoichiometry and simulates the isotopic labeling distribution in order to provide synthetic measurement values. **fwdsim** allows to perform a (linearized) statistical analysis in order to assess the reliability and sensitivity of the current flux distribution.

The simulation results and the generated synthetic measurement values are written to a **fwdsim**(5) XML document. Apart from the XML output, **fwdsim** allows to export various data and symbolic equations to HDF5, MathML, and text documents.

Common Options

The following options are common to **fwdsim** and **fitfluxes**:

`-h, --help` Show a brief help for all command line options.

`-i, --in <FILE> [default:stdin]` The name of the FluxML (XML) input file. If omitted, the FluxML document is expected on standard input.

`-o, --out <FILE> [default:stdout]` The name of the FWDSIM (XML) output file. If omitted, the generated FWDSIM document is written to standard output.

`-L, --list` Specifying this option results in a list of allowed configuration names for the specified FluxML document. The program exits immediately after emitting the list.

`-c, --configure <CFG> [default:'default']` Because FluxML documents may contain several `<configuration/>` elements this option allows to specify the configuration that should be used for the simulation. If this option is omitted it is assumed that the FluxML document contains a configuration with the name "default".

`-l, --log DEST` Specify the destination for the internal logging. In the most simple case `DEST` is a file name of a log file. In case the file exists new log messages are appended. Apart from log files it is possible to publish log messages to file descriptors, UNIX domain sockets, UDP and SCTP ports, and a small graphical user interface.

A file descriptor is specified by `fd:[num]`, where `[num]` is the number of the file descriptor.

A unix domain socket in the local file system is specified by `unix:[name]`, where `[name]` is the name of the socket file

A UDP or (connectionless) SCTP port is specified by `[proto]:[host]:[port]`, where `[proto]` is either "udp" or "sctp" and `[host]` is the name of the destination host and `[port]` is a UDP or SCTP port number on the destination host. Please note that the length of log messages is bounded by the minimum safe UDP packet size - log messages containing more than 548 characters will be truncated.

Finally, log messages can also be sent to a small GUI by specifying the destination `@gui@`. The GUI requires a working Perl/Tk installation and a running X server.

In order to capture all log messages concerning the command line processing this option should be specified in front of all other options.

`-v, --verbose 0..10 [default:5]` Specify the verbosity 0, 1, ..., 10 of generated / emitted log messages. The meaning of the different log levels is as follows:

- `0 (QUIET)` do not emit log messages at all.
- `1 (ERROR)` only emit severe error messages.
- `2 (WARNING)` only report severe errors and warnings.
- `3 (NOTICE)` report all errors and warnings including important informal messages.
- `4 (INFO)` report all errors, warnings and all informal messages.
- `5 (THROW)` in case of an exception, try to give a diagnosis of the error; sometimes even gives a backtrace of the current function stack.
- `6 (DEBUG0)` emit the more important debugging messages.
- `7 (DEBUG1)` emit the less important debugging messages
- `8 (DEBUG2)` emit the superfluous debugging messages
- `9 (DEBUG3)` emit annoying debugging messages.
- `10 (DEBUG4)` don't dare to use it!

`-t, --tolerance <val> [default:1e-9]` Specifies a constraint violation tolerance value. This parameter can be used to tolerate a certain constraint violation. Use with care.

Solver Options

There are three options affecting the behavior of the solver:

`-a, --apsolve` Solve the individual linear equation systems of the cumomer or EMU cascade using arbitrary precision arithmetic. Note that the results are converted to double precision immediately after solution. This option eliminates any round-off for the solution of the individual network levels.

`-A, --exact` In addition to option `-a` absolutely all computations are carried out using arbitrary precision arithmetic. If this option is used in conjunction with an analytical gradient (`-g analytic`) this results in exact derivatives. The results are converted to double precision just before writing them to the output file. This option is intended for debugging purposes. For large networks models the use of arbitrary precision arithmetic is probably to expensive.

`-d, --dbgsolve` When this option is used the network model is simulated in numerical debugging mode: for every equation system condition numbers are computed and the residual of the solution is checked.

`-n, --dry-run` Performs a dry-run, i.e. aborts the application after preprocessing of the input file and stoichiometry.

Statistics Options

fwdsim allows to evaluation linearized statistics.

`-s, --statistics` Compute linearized statistics for the current flux setting. This option is most beneficial if the flux setting is already optimized, i.e. results from an invocation of **fitfluxes**. Indeterminable fluxes will be reported. For all determinable fluxes standard deviations will be written to the output file (`-o`). If specified together with option `-H` (HDF5 export) the Jacobian of the measurements as well as the covariance matrix of the fluxes are exported into the HDF5 file.

`-g, --gradient <arg> [default:fd4]` The desired method for computaton of partial derivatives in case statistics are to be computed (option `-s`). Possible values are `'fd1'`, `'fd2'`, `'fd3'`, `'fd4'` for finite difference approximations, and `'analytic'` for exact derivatives. The default value is `'fd4'`, i.e. the gradient is approximated using an O(h^4) finite difference formula (which is slightly faster than using the exact derivatives).

Data Export Options

fwdsim allows to export the preprocessed stoichiometry, the generated system matrices, the symbolic system equations, the Jacobian of measurements, and covariance matrices into HDF5 and text files. A fully-fledged simulator-code for Octave/MATLAB can be exported into a `.m` file.

`-H, --hdf5 <FILE>` Export the stoichiometry and the left and right hand sides of the Cumomer / EMU equation systems into a HDF5 file. If, in addition, the computation of linearized statistics is requested (by option `-s`) the Jacobian of the measurements and the covariance matrix of the fluxes are exported, as well.

The exported data in the HDF5 files is intended to be used in MATLAB, but may also be used in other scientific software, such as Octave. The data in HDF5 files is organized in a directory-like structure. For example, in order to load the stoichiometric matrix out of the generated HDF5 file in MATLAB, the following command may be used:

```
S =hdf5read('file.h5','/stoichiometry/matrix');
```

Currently the following data is exported:

- `'/stoichiometry/matrix'`: The stoichiometric matrix
- `'/stoichiometry/r'`: The row labels of the stoichiometric matrix, i.e. string values. Assuming the row labels are imported into a MATLAB array `'rows'` the proper way to access the `k`'th label is `'rows(k).Data'`.
- `'/stoichiometry/c'`: The column labels of the stoichiometric matrix.
- `'/stoichiometry/fluxes'`: The flux distribution in net / exchange coordinates including the flux types. Possible flux types are:
  - `FREE` (free): the flux value may be chosen freely. In **fitfluxes**, these fluxes are adjusted by the optimization algorithm.
  - `CONS` (constraint): the flux value has a constant value. In **fitfluxes**, these fluxes must not be adjusted by the optimization algorithm.
  - `DEPD` (dependent): the flux value is determined by a number of constraint and free fluxes.
  - `QCON` (quasi-constraint): the flux value is completely determined by constraint fluxes.

  The first two columns in `'/stoichiometry/fluxes'` contain the values of the net and exchange fluxes. The remaining columns are boolean flags indicating the type of the type of the net and exchange flux.
- `'/stoichiometry/r_fluxes'`: The row labels of the matrix `'/stoichiometry/fluxes'`, i.e. the flux names.
- `'/stoichiometry/c_fluxes'`: The column labels of the matrix `'/stoichiometry/fluxes'`. If the column label has the suffix `'.n'`, it refers the net flux (e.g. `FREE.n`). The suffix `'.x'` flags the exchange flux (e.g. `QCON.x`).

- `'/stoichiometry/kernel/matrix_net'` and `'/stoichiometry/kernel/matrix_xch'`: The kernel matrices for net and exchange fluxes. Given the values of the free net and exchange fluxes the kernel matrices are used for recomputing the values of the dependent fluxes. For a kernel matrix `K`, this is achieved by evaluating `v =K*[1;v_f];`, where v_f contains the values of the free fluxes. Because 13CFLUX 2 determines kernel matrices using arbitrary precision arithmetic, the kernel matrices are the exact solutions of the stoichiometric system.
- `'/stoichiometry/kernel/r_net'` and `'/stoichiometry/kernel/r_xch'`: The row labels of the kernel matrices for net and exchange fluxes, i.e. all flux values.
- `'/stoichiometry/kernel/c_net'` and `'/stoichiometry/kernel/c_xch'`: The column labels of the kernel matrices for net and exchange fluxes, i.e. the names of the free net and free exchange fluxes. The label of the first column always contains the label `'1'`, flagging that the first column of the kernel matrices contain the particular solution.

In case option `-s` is passed additional statistical data is written to the HDF5 files:

- `'/cov_free/matrix'`: The covariance matrix of the free fluxes.
- `'/cov_free/r'` and `'/cov_free/c'`: The row and column labels of the covariance matrix, i.e. the names of the free fluxes. Net and exchange fluxes are distinguished by the suffix `'.n'` or `'.x'`.
- `'/cov/matrix'`: The full covariance matrix of all fluxes.
- `'/cov/r'` and `'/cov/c'`: The row and column labels of the full covariance matrix.
- `'/jacobian/matrix'`: The Jacobian of the measurements, i.e. partial derivatives of the measurement values with respect to the free fluxes.
- `'/jacobian/r'`: The row labels of the Jacobian, i.e. names of the measurement values. The names of the measurement values are given in short notation (see bibliography).
- `'/jacobian/c'`: The column labels of the Jacobian, i.e. the names of the free fluxes. Net and exchange fluxes are distinguished by the suffix `'.n'` or `'.x'`.

-M, --mathml <FILE> Export the cascade's symbolic systems into a MathML file. The resulting Content-MathML file may be imported in computer algebra systems like Maple and Mathematica.

-T, --text <FILE> Exports the cumomer / EMU cascade's symbolic equation system into the specified text file.

-m, --matlab <FILE> Generate an Octave/MATLAB simulator function for the current network in the specified text file. The function name will match the name of the text file. The external function `cumulate` is contained in the 13CFLUX 2 installation.

Examples

Simulate an isotope labeling network and safe the XML output to file network.fwd (most simple case):

```
fwdsim -i network.fml -o network.fwd
```

Convert an old FTBL to FluxML, simulate and filter out the simulated measurement values (**fwdsim** reads from stdin and writes to stdout):

```
ftbl2fml -i network.ftbl | fwdsim | fwdsimflt -m
```

Simulate an isotope labeling network and generate all possible output; send the XML output to /dev/null:

```
fwdsim -i net.fml -o /dev/null -M m.mml -T t.txt -m f.m -H net.h5
```

Simulate an isotope labeling network and and compute linearized statistics. In addition to the simulation results, write estimated standard deviations to the output file. Save the stoichiometry, the cascaded equation systems, the Jacobian of measurements, and the covariance matrices to a HDF5 file:

```
fwdsim -s -i network.fml -o network.fwd -H network.h5
```

See Also

fitfluxes(1), fwdsimflt(1), ftbl2fml(1), simreport(1)

Author

Michael Weitzel
This manpage was written by Michael Weitzel <info@13cflux.net>.

2.1.2 fitfluxes

fitfluxes — fit a flux distribution of a network in order to reproduce a set of isotope labeling measurement values.

Synopsis

`fitfluxes` [*`options`*]

Description

Fitfluxes is a program for intelligently adjusting a flux distribution of a metabolic network in order to reproduce a set of isotope labeling measurement values.

For this purpose **fitfluxes** reads the metabolic network and measurement configuration from a FluxML file, analyzes the network's stoichiometry and uses an optimization algorithm for adjusting certain 'free' variables in the flux distribution. Given the resulting new flux distribution the network and the measurements are simulated and the resulting synthetic measurement values are compared to the real ones. This process is repeated until synthetic and real measurement values agree or a certain termination condition is fulfilled (e.g. the maximum number of iterations is exceeded).

The resulting flux distribution and the corresponding synthetic measurement values are written to a **fwdsim**(5) XML document, i.e. a document in the same format also generated by the **fwdsim**(1) command.

Common Options

The following options are common to **fwdsim** and **fitfluxes**:

`-h, --help` Show a brief help for all command line options.

`-i, --in <FILE> [default:stdin]` The name of the FluxML (XML) input file. If omitted, the FluxML document is expected on standard input.

`-o, --out <FILE> [default:stdout]` The name of the FWDSIM (XML) output file. If omitted, the generated FWDSIM document is written to standard output.

`-L, --list` Specifying this option results in a list of allowed configuration names for the specified FluxML document. The program exits immediately after emitting the list.

`-c, --configure <CFG> [default:'default']` Because FluxML documents may contain several `<configuration/>` elements this option allows to specify the configuration that should be used for the simulation. If this option is omitted it is assumed that the FluxML document contains a configuration with the name "default".

`-l, --log DEST` Specify the destination for the internal logging. In the most simple case `DEST` is a file name of a log file. In case the file exists new log messages are appended. Apart from log files it is possible to publish log messages to file descriptors, UNIX domain sockets, UDP and SCTP ports, and a small graphical user interface.

A file descriptor is specified by `fd:[num]`, where `[num]` is the number of the file descriptor.

A unix domain socket in the local file system is specified by `unix:[name]`, where `[name]` is the name of the socket file

A UDP or (connectionless) SCTP port is specified by `[proto]:[host]:[port]`, where `[proto]` is either "udp" or "sctp" and `[host]` is the name of the destination host and `[port]` is a UDP or SCTP port number on the destination host. Please note that the length of log messages is bounded by the minimum safe UDP packet size - log messages containing more than 548 characters will be truncated.

Finally, log messages can also be sent to a small GUI by specifying the destination `@gui@`. The GUI requires a working Perl/Tk installation and a running X server.

In order to capture all log messages concerning the command line processing this option should be specified in front of all other options.

`-v, --verbose 0..10 [default:5]` Specify the verbosity 0, 1, ..., 10 of generated / emitted log messages. The meaning of the different log levels is as follows:

- `0 (QUIET)` do not emit log messages at all.
- `1 (ERROR)` only emit severe error messages.
- `2 (WARNING)` only report severe errors and warnings.
- `3 (NOTICE)` report all errors and warnings including important informal messages.
- `4 (INFO)` report all errors, warnings and all informal messages.
- `5 (THROW)` in case of an exception, try to give a diagnosis of the error; sometimes even gives a backtrace of the current function stack.
- `6 (DEBUG0)` emit the more important debugging messages.
- `7 (DEBUG1)` emit the less important debugging messages
- `8 (DEBUG2)` emit the superfluous debugging messages
- `9 (DEBUG3)` emit annoying debugging messages.
- `10 (DEBUG4)` don't dare to use it!

`-t, --tolerance <val> [default:1e-9]` Specifies a constraint violation tolerance value. This parameter can be used to tolerate a certain constraint violation. Use with care.

Solver Options

There are three options affecting the behavior of the solver. The same options are supported by the **fwdsim** command:

`-a, --apsolve` Solve the individual linear equation systems of the cumomer or EMU cascade using arbitrary precision arithmetic. Note that the results are converted to double precision immediately after solution. This option eliminates any round-off for the solution of the individual network levels.

`-A, --exact` In addition to option `-a` absolutely all computations are carried out using arbitrary precision arithmetic. If this option is used in conjunction with an analytical gradient (`-g analytic`) this results in exact derivatives. The results are converted to double precision just before writing them to the output file. This option is intended for debugging purposes. For large networks models the use of arbitrary precision arithmetic is probably to expensive.

`-d, --dbgsolve` When this option is used the network model is simulated in numerical debugging mode: for every equation system condition numbers are computed and the residual of the solution is checked.

Optimization Options

`-O, --optimizer <arg> [default:IPOPT]` The desired optimization algorithm. Currently available are `Ipopt`, `CFSQP`, and `NAGNLP` (the latter two are distributed under a commercial license and may not be available at your site).

`-P, --properties <arg>` This option allows fine-grained control over the optimization algorithm's behavior and settings. The argument to option `-P` is a comma-separated list of key-value pairs. The general syntax is: `[optimizer].[property]=[type]([value])`.

Valid `[optimizer]`'s are '`ipopt`' and '`cfsqp`'.

Valid `[type]`'s are:

- `integer` - in this case `[value]` has to be an integer.

- `string` - the contents of `[value]` are allowed to be any string.
- `real` - for real (double precision) values.
- `boolean` - restricts the contents of `[value]` to the values `true` and `false`.

Using this interface all of the numerous settings of Ipopt can be accessed. A full list of valid options can be obtained by setting the boolean property `ipopt.print_options_documentstion` to `true`. See the examples below. Other interesting properties include:

- `ipopt.max_cpu_time` - allows to abort Ipopt after a specified number of CPU seconds is exceeded.
- `ipopt.print_level` - the verbosity of Ipopt. Valid settings are integers from 0 to 12. The default setting is 5.
- `ipopt.max_iter` - the maximum number of iterations before termination. The default value is 3000.
- `ipopt.linear_solver` - allows to choose the linear solver used by Ipopt. The default setting is '`mumps`'. Using another solver may improve convergence speed and optimization results.

CFSQP supports significantly fewer properties (a more detailed description can be found in the CFSQP documentation):

- `cfsqp.mode` - the type and mode of the used solver. The default setting is 200.
- `cfsqp.iprint` - the verbosity of CFSQP. The default setting is 1. Other settings are 0 (quiet), 2, or 3 (more verbose output).
- `cfsqp.miter` - the maximum number of iterations before termination. The default value is 1000.
- `cfsqp.bigbnd` - allows setting the value which plays the role of infinity. The default value is 1e20.
- `cfsqp.eps` - final norm of the newton gradient. Must be bigger than the machine epsilon. The default value is 1e-10.

NAGNLP refers to the "nag_opt_nlp" ("e04ucc") optimizer found in the commercial NAGC library. Most important settings are accessible via the following properties (cf. NAGC documentation):

- `nag_opt_nlp.max_iter` - the maximum number of iterations before termination. The default value is 250.
- `nag_opt_nlp.list` - print settings. The default value is "false".
- `nag_opt_nlp.print_level` - the major print level. The default setting is "Nag_Soln_Iter". See the NAGC documentation for other available settings.
- `nag_opt_nlp.minor_print_level` - the minor print level. The default setting is "Nag_NoPrint". See the NAGC documentation for other available settings.
- `nag_opt_nlp.verify_grad` - allows to enable the gradient checker. The default setting is "Nag_NoCheck". Set to "Nag_SimpleCheck" to turn on gradient checking.

`-S, --shrink <eps> [default:5e-7]` This option may be used to tighten the complete set of constraints exposed to the optimization packages. If you think of the constraints as a convex polyhedron this option causes the polyhedron to be shrunk symmetrically about the specified argument `eps`. Specify this option if you notice warning messages reporting that the optimizer is violating constraints. Because this seems to be a common problem in the used optimization packages (probably due to numerical inaccuracies) `fitfluxes` uses a default shrinking of 5-e7. Specify `-S 0` to disable this feature.

`-g, --gradient <arg> [default:fd4]` The desired method for computaton of partial derivatives in case statistics are to be computed (option `-s`). Possible values are '`fd1`','`fd2`','`fd3`','`fd4`' for finite difference approximations, and '`analytic`' for exact derivatives. The default value is '`fd4`', i.e. the gradient is approximated using an O(h^4) finite difference formula (which is slightly faster than using the exact derivatives).

Examples

Find a new flux distribution for a metabolic network specified in a FluxML file in order to reproduce a set of istope labeling measurement values. Safe the XML output to file network.fwd (most simple case):

```
fitfluxes -i network.fml -o network.fwd
```

Use the optimization package CFSQP instead of Ipopt and compute the required gradient analytically (instead of fourth order finite differences):

```
fitfluxes -i network.fml -o network.fwd -O CFSQP -g analytic
```

Convert an old FTBL to FluxML, find a new flux distribution using Ipopt, and filter out the flux distribution in net / echange01 coordinates (the flux coordinate system of 13CFLUX 1). Send all log messages of fitfluxes to a GUI (requires a running X server):

```
ftbl2fml -i network.ftbl | fitfluxes -l @gui@| fwdsimflt -s -X
```

Request the documentation of all settings from Ipopt. Also set option `max_iter` to 0 in order to abort as soon as possible. Redirect the output to text file ipopt_settings.txt:

```
fitfluxes -i network.fml -o /dev/null -P "ipopt.print_options_documentation=
boolean(true), ipopt.max_iter=integer(0)" > ipopt_settings.txt
```

Use the commercial NAG NLP SQP optimizer (nag_opt_nlp, e04ucc) and restrict the maximum number of iterations to 50 (default: 1000):

```
fitfluxes -i network.fml -o network.fwd -O nagnlp -P "nag_opt_nlp.max_iter=
integer(50)"
```

Request a list of available options along with their default settings from the NAG NLP optimizer (the options and their meaning is discussed in the NAG manual of e04ucc):

```
fitfluxes -i network.fml -o /dev/null -O nagnlp -P "nag_opt_nlp.print_avai
lable_options=boolean(true)"
```

See Also

fwdsim(1), fwdsim(5), fwdsimflt(1), ftbl2fml(1), simreport(1), setfluxes(1), multifit(1)

Author

Michael Weitzel

This manpage was written by Michael Weitzel <info@13cflux.net>.

2.1.3 multifit

multifit — a multi-start, multi-processor frontend for fitfluxes.

Synopsis

`multifit` [*`options`*] [*`--[additional parameters for fitfluxes]`*]

Description

The program **multifit** runs a multi-processor, multi-start optimization using **fitfluxes**(1) based on random flux distributions created by the **ssampler**(1) tool. **multifit** performs a simple type of load balancing in order to use the available computational resource as efficient as possible.

Via command line option `-g, --generate` it is possible to generate a configuration file which can be used to customize the command lines for preprocessing (sampling), processing (parameter fitting), and postprocessing (report generation). As soon as the preprocessing is finished it is safe to interrupt **multifit** by pressing Ctrl-C. In this case **multifit** will wait for the pending jobs to finish. After all jobs have finished, the index of the last finished job will be reported. This number may be used to restart **multifit** later (option `-r`).

All extra options specified after `--` are directly passed to the **fitfluxes** invocations.

Options

`-h, --help` Show a brief help for all command line options.

`-i, --in <FILE> [default:stdin]` The name of the FluxML (XML) input file. If omitted, the FluxML document is expected on standard input.

`-o, --outdir <DIR> [default:./]` Name of output directory. If none is specified the current directory is used. If the specified output directory does not exist it is created first.

`-p, --prefix <PKG> [default:IPOPT]` A prefix for the output file names written to the output directory. If no prefix is given, the prefix defaults to the name of the FluxML file.

`-c, --configure <CFG> [default:'default']` Because FluxML documents may contain several `<configuration/>` elements this option allows to specify the configuration that should be used for the simulation. If this option is omitted it is assumed that the FluxML document contains a configuration with the name "default".

`-n, --samples <NUMBER>` The number of random flux distribution to generate. This equals to the number of fitfluxes jobs. This option is required if option `-f` is omitted or the file specified by option `-f` does not exist

`-f, --fluxdist <FILE>` The name of a HDF5 file containing random flux distributions. If the specified file does not exist it will be created. If this option is omitted a temporary file is created and erased on exit.

`-t, --tasks <RANGE> [default:<automatically determined>]` This option specifies a range for the number of parallel paths. If this option is not specified the number of parallel tasks is set to the number of currently idle processors. Possible values for <RANGE> are:

- `N` - where N is a fixed number of parallel tasks. This number is not changed during the processing.
- `N:` - where N is a lower bound for the number of parallel tasks. This number may be increased depending on the number of installed processors and current load of the machine.
- `:N` - where N is an upper bound of the number of parallel tasks. This number may be decreased depending on the number of installed processors and current load of the machine.
- `M:N` - where M and N are lower and upper bounds for the number of parallel tasks. Within these bounds, the number of parallel tasks may be adjusted depeneding on the number of installed processors and current load of the machine.

`-r, --restart <INDEX>` This option causes **multifit** to restart at a given dataset/job index and is useful if **multifit** was previously interrupted by pressing Ctrl-C.

`-v, --verbose` Echo task-id and command line of any executed command. This option is intended for debugging purposes.

`-g, --generate` When passing this option a template configuration file '$HOME/.x3cflux/.multifitrc' is generated. The configuration file may be used to adjust the command lines for sampling and parameter fitting.

`-m, --manpage` Displays an old, no longer maintained version of this manual page.

Examples

Run 100 multi-start optimizations on parallal on all free CPU cores. Save results to directory 'foobar' into files having prefix 'network'. Pass options '-O IPOPT -P ipopt.print_level=integer\(0\)' directly to **fitfluxes**(1):

```
   multifit -i network.fml -n 100 -o foobar -p network ---O IPOPT -P ipopt.pri
nt_level=integer\(0\)
```

See Also

ssampler(1), fitfluxes(1), setfluxes(1), collectfitdata(1)

Author

Michael Weitzel
This manpage was written by Michael Weitzel <info@13cflux.net>.

2.1.4 multifitfluxes

multifitfluxes — parallel 13CFLUX2 parameter estimation tool

Synopsis

`multifitfluxes` [*`options`*]

Description

Multifitfluxes is a parallel implementation of fitfluxes(1). Like the original **fitfluxes(1)** application, FluxML documents are taken as input along with several parameters. Unlike **fitfluxes(1)** , the output is either a directory eventually containing the result files (i.e., FWDSIM files containing parameter estimation results), or an unified HDF5 comprising flux values.

Common Options

`-h, --help` Show a brief help for all command line options.

`-i, --in <FILE> [default:stdin]` The name of the FluxML (XML) input file. If omitted, the FluxML document is expected on standard input.

`-o, --out <PATH> [default:stdout]` Output directory for FWDSIM (XML) files.

`-c, --configure <CFG> [default:'default']` Because FluxML documents may contain several `<configuration/>` elements this option allows to specify the configuration that should be used for the simulation. If this option is omitted it is assumed that the FluxML document contains a configuration with the name "default".

`-H, --hdf5-in <File>` Input file containing flux samples (HDF5)

`-f, --hdf5-out <File>` Output file containing flux samples (HDF5)

`-N, --max-procs <NUM> [default:1]` Number of maximum parallel processes

`-B, --block-size <NUM> [default:auto]` Number of samples per task. This option controls the parallelism granularity

`-l, --log DEST` Specify the destination for the internal logging. In the most simple case `DEST` is a file name of a log file. In case the file exists new log messages are appended. Apart from log files it is possible to publish log messages to file descriptors, UNIX domain sockets, UDP and SCTP ports, and a small graphical user interface.

A file descriptor is specified by `fd:[num]`, where `[num]` is the number of the file descriptor.

A unix domain socket in the local file system is specified by `unix:[name]`, where `[name]` is the name of the socket file

A UDP or (connectionless) SCTP port is specified by `[proto]:[host]:[port]`, where `[proto]` is either "udp" or "sctp" and `[host]` is the name of the destination host and `[port]` is a UDP or SCTP port number on the destination host. Please note that the length of log messages is bounded by the minimum safe UDP packet size - log messages containing more than 548 characters will be truncated.

Finally, log messages can also be sent to a small GUI by specifying the destination `@gui@`. The GUI requires a working Perl/Tk installation and a running X server.

In order to capture all log messages concerning the command line processing this option should be specified in front of all other options.

`-v, --verbose 0..10 [default:5]` Specify the verbosity 0, 1, ..., 10 of generated / emitted log messages. The meaning of the different log levels is as follows:

- `0 (QUIET)` do not emit log messages at all.
- `1 (ERROR)` only emit severe error messages.
- `2 (WARNING)` only report severe errors and warnings.
- `3 (NOTICE)` report all errors and warnings including important informal messages.
- `4 (INFO)` report all errors, warnings and all informal messages.
- `5 (THROW)` in case of an exception, try to give a diagnosis of the error; sometimes even gives a backtrace of the current function stack.
- `6 (DEBUG0)` emit the more important debugging messages.
- `7 (DEBUG1)` emit the less important debugging messages
- `8 (DEBUG2)` emit the superfluous debugging messages
- `9 (DEBUG3)` emit annoying debugging messages.
- `10 (DEBUG4)` don't dare to use it!

`-t, --tolerance <val> [default:1e-9]` Specifies a constraint violation tolerance value. This parameter can be used to tolerate a certain constraint violation. Use with care.

Solver Options

`-a, --apsolve` Solve the individual linear equation systems of the cumomer or EMU cascade using arbitrary precision arithmetic. Note that the results are converted to double precision immediately after solution. This option eliminates any round-off for the solution of the individual network levels.

`-A, --exact` In addition to option `-a` absolutely all computations are carried out using arbitrary precision arithmetic. If this option is used in conjunction with an analytical gradient (`-g analytic`) this results in exact derivatives. The results are converted to double precision just before writing them to the output file. This option is intended for debugging purposes. For large networks models the use of arbitrary precision arithmetic is probably to expensive.

`-d, --dbgsolve` When this option is used the network model is simulated in numerical debugging mode: for every equation system condition numbers are computed and the residual of the solution is checked.

Optimization Options

`-O, --optimizer <arg> [default:IPOPT]` The desired optimization algorithm. Currently available are `Ipopt`, `CFSQP`, and `NAGNLP` (the latter two are distributed under a commercial license and may not be available at your site).

`-P, --properties <arg>` This option allows fine-grained control over the optimization algorithm's behavior and settings. The argument to option `-P` is a comma-separated list of key-value pairs. The general syntax is: `[optimizer].[property]=[type]([value])`.

Valid `[optimizer]`'s are '`ipopt`' and '`cfsqp`'.

Valid `[type]`'s are:

- `integer` - in this case `[value]` has to be an integer.
- `string` - the contents of `[value]` are allowed to be any string.
- `real` - for real (double precision) values.
- `boolean` - restricts the contents of `[value]` to the values `true` and `false`.

Using this interface all of the numerous settings of Ipopt can be accessed. A full list of valid options can be obtained by setting the boolean property `ipopt.print_options_documentstion` to `true`. See the examples below. Other interesting properties include:

- `ipopt.max_cpu_time` - allows to abort Ipopt after a specified number of CPU seconds is exceeded.
- `ipopt.print_level` - the verbosity of Ipopt. Valid settings are integers from 0 to 12. The default setting is 5.
- `ipopt.max_iter` - the maximum number of iterations before termination. The default value is 3000.
- `ipopt.linear_solver` - allows to choose the linear solver used by Ipopt. The default setting is '`mumps`'. Using another solver may improve convergence speed and optimization results.

CFSQP supports significantly fewer properties (a more detailed description can be found in the CFSQP documentation):

- `cfsqp.mode` - the type and mode of the used solver. The default setting is 200.
- `cfsqp.iprint` - the verbosity of CFSQP. The default setting is 1. Other settings are 0 (quiet), 2, or 3 (more verbose output).
- `cfsqp.miter` - the maximum number of iterations before termination. The default value is 1000.
- `cfsqp.bigbnd` - allows setting the value which plays the role of infinity. The default value is 1e20.
- `cfsqp.eps` - final norm of the newton gradient. Must be bigger than the machine epsilon. The default value is 1e-10.

NAGNLP refers to the "nag_opt_nlp" ("e04ucc") optimizer found in the commercial NAGC library. Most important settings are accessible via the following properties (cf. NAGC documentation):

- `nag_opt_nlp.max_iter` - the maximum number of iterations before termination. The default value is 250.
- `nag_opt_nlp.list` - print settings. The default value is "false".
- `nag_opt_nlp.print_level` - the major print level. The default setting is "Nag_Soln_Iter". See the NAGC documentation for other available settings.
- `nag_opt_nlp.minor_print_level` - the minor print level. The default setting is "Nag_NoPrint". See the NAGC documentation for other available settings.
- `nag_opt_nlp.verify_grad` - allows to enable the gradient checker. The default setting is "Nag_NoCheck". Set to "Nag_SimpleCheck" to turn on gradient checking.

`-S, --shrink <eps> [default:5e-7]` This option may be used to tighten the complete set of constraints exposed to the optimization packages. If you think of the constraints as a convex polyhedron this option causes the polyhedron to be shrunk symmetrically about the specified argument `eps`. Specify this option if you notice warning messages reporting that the optimizer is violating constraints. Because this seems to be a common problem in the used optimization packages (probably due to numerical inaccuracies) `fitfluxes` uses a default shrinking of 5-e7. Specify `-S 0` to disable this feature.

`-g, --gradient <arg> [default:fd4]` The desired method for computaton of partial derivatives in case statistics are to be computed (option `-s`). Possible values are '`fd1`','`fd2`','`fd3`','`fd4`' for finite difference approximations, and '`analytic`' for exact derivatives. The default value is '`fd4`', i.e. the gradient is approximated using an O(h^4) finite difference formula (which is slightly faster than using the exact derivatives).

Examples

[source,shell]:
```
multifitfluxes -i model.fml -H samples.hdf5 -f output.hdf5
```

See Also

fluxml(5), fitfluxes(5), fwdsim(1),

Author

Multifitfluxes extends fitfluxes(1), which was originally written by Michael Weitzel.
This manpage was written by Tolga Dalman and Michael Weitzel <info@13cflux.net>.

2.1.5 multifwdsim

multifwdsim — parallel 13CFLUX2 forward simulation tool

Synopsis

`multifwdsim` [*options*]

Description

Multifwdsim is a parallel implementation of **fwdsim(1)** . Like the original **fwdsim(1)** application, FluxML documents are taken as input along with several parameters. Unlike **fwdsim(1)**, the output is a directory containing the result files (i.e., FWDSIM files containing simulation results)

Common Options

`-h, --help` Show a brief help for all command line options.

`-i, --in <FILE> [default:stdin]` The name of the FluxML (XML) input file. If omitted, the FluxML document is expected on standard input.

`-o, --out <PATH> [default:stdout]` Output directory for FWDSIM (XML) files.

`-c, --configure <CFG> [default:'default']` Because FluxML documents may contain several `<configuration/>` elements this option allows to specify the configuration that should be used for the simulation. If this option is omitted it is assumed that the FluxML document contains a configuration with the name "default".

`-H, --hdf5-in <File>` Input file containing flux samples (HDF5)

`-N, --max-procs <NUM> [default:1]` Number of maximum parallel processes

`-B, --block-size <NUM> [default:auto]` Number of samples per task. This option controls the parallelism granularity

`-l, --log DEST` Specify the destination for the internal logging. In the most simple case `DEST` is a file name of a log file. In case the file exists new log messages are appended. Apart from log files it is possible to publish log messages to file descriptors, UNIX domain sockets, UDP and SCTP ports, and a small graphical user interface.

A file descriptor is specified by `fd:[num]`, where `[num]` is the number of the file descriptor.

A unix domain socket in the local file system is specified by `unix:[name]`, where `[name]` is the name of the socket file

A UDP or (connectionless) SCTP port is specified by `[proto]:[host]:[port]`, where `[proto]` is either "udp" or "sctp" and `[host]` is the name of the destination host and `[port]` is a UDP or SCTP port number on the destination host. Please note that the length of log messages is bounded by the minimum safe UDP packet size - log messages containing more than 548 characters will be truncated.

Finally, log messages can also be sent to a small GUI by specifying the destination `@gui@`. The GUI requires a working Perl/Tk installation and a running X server.

In order to capture all log messages concerning the command line processing this option should be specified in front of all other options.

`-v, --verbose 0..10 [default:5]` Specify the verbosity 0, 1, ..., 10 of generated / emitted log messages. The meaning of the different log levels is as follows:

- `0 (QUIET)` do not emit log messages at all.

- `1 (ERROR)` only emit severe error messages.
- `2 (WARNING)` only report severe errors and warnings.
- `3 (NOTICE)` report all errors and warnings including important informal messages.
- `4 (INFO)` report all errors, warnings and all informal messages.
- `5 (THROW)` in case of an exception, try to give a diagnosis of the error; sometimes even gives a backtrace of the current function stack.
- `6 (DEBUG0)` emit the more important debugging messages.
- `7 (DEBUG1)` emit the less important debugging messages
- `8 (DEBUG2)` emit the superfluous debugging messages
- `9 (DEBUG3)` emit annoying debugging messages.
- `10 (DEBUG4)` don't dare to use it!

`-t, --tolerance <val> [default:1e-9]` Specifies a constraint violation tolerance value. This parameter can be used to tolerate a certain constraint violation. Use with care.

Solver Options

`-a, --apsolve` Solve the individual linear equation systems of the cumomer or EMU cascade using arbitrary precision arithmetic. Note that the results are converted to double precision immediately after solution. This option eliminates any round-off for the solution of the individual network levels.

`-A, --exact` In addition to option `-a` absolutely all computations are carried out using arbitrary precision arithmetic. If this option is used in conjunction with an analytical gradient (`-g analytic`) this results in exact derivatives. The results are converted to double precision just before writing them to the output file. This option is intended for debugging purposes. For large networks models the use of arbitrary precision arithmetic is probably to expensive.

`-d, --dbgsolve` When this option is used the network model is simulated in numerical debugging mode: for every equation system condition numbers are computed and the residual of the solution is checked.

Statistics Options

`-s, --statistics` Compute linearized statistics for the current flux setting. This option is most beneficial if the flux setting is already optimized, i.e. results from an invocation of **fitfluxes**. Indeterminable fluxes will be reported. For all determinable fluxes standard deviations will be written to the output file (`-o`). If specified together with option `-H` (HDF5 export) the Jacobian of the measurements as well as the covariance matrix of the fluxes are exported into the HDF5 file.

`-g, --gradient <arg> [default:fd4]` The desired method for computaton of partial derivatives in case statistics are to be computed (option `-s`). Possible values are '`fd1`', '`fd2`', '`fd3`', '`fd4`' for finite difference approximations, and '`analytic`' for exact derivatives. The default value is '`fd4`', i.e. the gradient is approximated using an O(h^4) finite difference formula (which is slightly faster than using the exact derivatives).

Examples

[source,shell]:

```
multifwdsim -i model.fml -H samples.hdf5
```

See Also

fluxml(5), fitfluxes(5), fwdsim(1),

Author

Multifwdsim extends fwdsim(1), which was originally written by Michael Weitzel.
This manpage was written by Tolga Dalman and Michael Weitzel <info@13cflux.net> .

2.1.6 mcbootstrap

mcbootstrap — nonlinear statistical assessment of flux maps by Monte Carlo bootstrap

Synopsis

`mcbootstrap` [*`options`*]

Description

mcbootstrap is a parallel implementation of the Monte Carlo bootstrap algorithm employing the 13CFLUX2 tools **ssampler(1)**, **perturb(1)**, **multifit(1)** and **collectfitdata(1)** as described in Dalman et al. FGCS, 2011, 10.1016/j.future.2011.10.007.

Common Options

`-h, --help` Show a brief help for all command line options.

`-i, --in <FILE> [default:stdin]` The name of the FluxML (XML) input file. If omitted, the FluxML document is expected on standard input.

`-o, --out <PATH> [default:stdout]` [default: .] Output directory for FWDSIM (XML) files. If omitted, the simulated FWDSIM files are placed in the current directory.

`-c, --configure <CFG> [default:'default']` Because FluxML documents may contain several `<configuration/>` elements this option allows to specify the configuration that should be used for the simulation. If this option is omitted it is assumed that the FluxML document contains a configuration with the name "default".

`-n, --samples <NUM> [default:1]` Number of random initial flux vectors generated with **ssampler(1)**

`-m, --samples <NUM> [default:1]` Number of perturbtions generated with **perturb(1)**

`-v, --verbose 0..10 [default:5]` Specify the verbosity 0, 1, ..., 10 of generated / emitted log messages. The meaning of the different log levels is as follows:

- `0 (QUIET)` do not emit log messages at all.
- `1 (ERROR)` only emit severe error messages.
- `2 (WARNING)` only report severe errors and warnings.
- `3 (NOTICE)` report all errors and warnings including important informal messages.
- `4 (INFO)` report all errors, warnings and all informal messages.
- `5 (THROW)` in case of an exception, try to give a diagnosis of the error; sometimes even gives a backtrace of the current function stack.
- `6 (DEBUG0)` emit the more important debugging messages.
- `7 (DEBUG1)` emit the less important debugging messages
- `8 (DEBUG2)` emit the superfluous debugging messages
- `9 (DEBUG3)` emit annoying debugging messages.
- `10 (DEBUG4)` don't dare to use it!

Examples

[source,shell]:
```
  mcbootstrap -i model.fml -o ouput_folder -n 2 -m 3
```

See Also

fluxml(5), fwdsim(5), fwdsim(1),ssampler(1),perturb(1), multifit(1), collectfitdata(1)

Author

This manpage was written by Tolga Dalman and Michael Weitzel <info@13cflux.net>.

2.2 Sampling and Stoichiometry Initialization

sscanner The program sscanner is used to generate an initial flux distribution for a metabolic network. In order to perform this task sscanner reads a FluxML document (see also **fluxml**(1)), chooses some free fluxes and reinitializes the flux values with the analytical center of the specified constraints.

Important command line switches:

`-s` choice of sampling targets

`-b` bound for magnitude of unbounded net fluxes

`-p` bound for unbounded exchange fluxes

ssampler The program **ssampler**(1) is used for sampling uniformly distributed, random flux distributions from the stoichiometry of a metabolic reaction network. The specification of the reaction network is assumed to be contained in a FluxML (see **fluxml**(5)) document. The generated flux distributions are saved to a HDF5 file. The main purpose of **ssampler**(1) is the generation of initial feasible flux distributions and random samples for distributed multi-start optimization.

Important command line switches:

`-S` choice of the sampling algorithm

`-b, -p` bound for unbounded net and exchange fluxes

`-f` faster sampling

perturb The program **perturb** is used for perturbing flux and labeling measurement values given in a FluxML document based on normally distributed random numbers. The mean of a resulting (perturbed) measurement value is given by the original measurement value. The standard deviation used for perturbation is the (scaled) standard deviation specified along with the original measurement value. On exit, **perturb** writes a modified FluxML document containing the perturbed measurement values.

Important command line switches:

`-m` perturbation of measurements

`-f` perturbation of fluxes

`-t` choice of measurement types for perturbation

multiperturb The program **multiperturb** is a parallel implemenetation of **perturb(1)**. Like the **perturb(1)** application, FluxML documents are taken as input along with several parameters.

Important command line switches:

`-H` output HDF5 file containing flux samples

`-n` number of total perturbations

`-m` perturbation of measurements

`-f` perturbation of fluxes

`-t` choice of measurement types for perturbation

2.2.1 sscanner

sscanner — generate an initial flux distribution for a metabolic network

Synopsis

`sscanner` [*`options`*]

Description

The program **sscanner** is used to generate an initial flux distribution for a metabolic network. In order to perform this task **sscanner** reads a FluxML document (see also **fluxml**(1)), chooses some free fluxes and reinitializes the flux values with the analytical center of the specified constraints.

Common Options

`-h, --help` Show a brief help for all command line options.

`-i, --in <FILE> [default:stdin]` The name of the FluxML (XML) input file. If omitted, the FluxML document is expected on standard input.

`-o, --out <FILE> [default:stdout]` The name of the output FluxML file. If this option is omitted the resulting FluxML document is written to standard output.

`-c, --configure <CFG> [default:'default']` Because FluxML documents may contain several `<configuration/>` elements this option allows to specify the configuration that should be used for the simulation. If this option is omitted it is assumed that the FluxML document contains a configuration with the name "default".

`-L, --list` Specifying this option results in a list of allowed configuration names for the specified FluxML document. The program exits immediately after emitting the list.

`-l, --log DEST` Specify the destination for the internal logging. In the most simple case `DEST` is a file name of a log file. In case the file exists new log messages are appended. Apart from log files it is possible to publish log messages to file descriptors, UNIX domain sockets, UDP and SCTP ports, and a small graphical user interface.

A file descriptor is specified by `fd:[num]`, where `[num]` is the number of the file descriptor.

A unix domain socket in the local file system is specified by `unix:[name]`, where `[name]` is the name of the socket file

A UDP or (connectionless) SCTP port is specified by `[proto]:[host]:[port]`, where `[proto]` is either "udp" or "sctp" and `[host]` is the name of the destination host and `[port]` is a UDP or SCTP port number on the destination host. Please note that the length of log messages is bounded by the minimum safe UDP packet size - log messages containing more than 548 characters will be truncated.

Finally, log messages can also be sent to a small GUI by specifying the destination `@gui@`. The GUI requires a working Perl/Tk installation and a running X server.

In order to capture all log messages concerning the command line processing this option should be specified in front of all other options.

`-v, --verbose 0..10 [default:5]` Specify the verbosity 0, 1, ..., 10 of generated / emitted log messages. The meaning of the different log levels is as follows:

- `0 (QUIET)` do not emit log messages at all.
- `1 (ERROR)` only emit severe error messages.
- `2 (WARNING)` only report severe errors and warnings.
- `3 (NOTICE)` report all errors and warnings including important informal messages.
- `4 (INFO)` report all errors, warnings and all informal messages.
- `5 (THROW)` in case of an exception, try to give a diagnosis of the error; sometimes even gives a backtrace of the current function stack.
- `6 (DEBUG0)` emit the more important debugging messages.
- `7 (DEBUG1)` emit the less important debugging messages
- `8 (DEBUG2)` emit the superfluous debugging messages
- `9 (DEBUG3)` emit annoying debugging messages.
- `10 (DEBUG4)` don't dare to use it!

Special Options

-B, --bounds Determine the lower and upper bounds for the settings of the free fluxes by using an LP solver. The program exits immediately after emitting the list.

-C, --constraints This option causes a list of user defined and automatically inferred equality and inequality constraints to be generated. The program exits immediately after emitting the list.

-m, --montecarlo <NUM> Determine the setting of the free fluxes from the centroid of point cloud generated using Gibbs sampling, a Markov Chain Monte Carlo technique. This parameter requires the number of Monte Carlo samples to be specified.

-r, --resample <n,x,nx> [default:nx] This option is used to specify what to resample. Possible values are 'n', 'x', and 'nx' - for sampling only net fluxes, only exchange fluxes, or both net and exchange fluxes. The default is to resample both net and exchange fluxes.

-b, --bound-net <VALUE> [default:100] The bound for the magnitude of unbounded net fluxes. Defaults to 100.

-p, --bound-xch <VALUE> [default:100] The bound for unbounded exchange fluxes. Defaults to 100.

Examples

Generate a new flux distribution for the network in FluxML file `network.fml` representing the analytical center of the specified stoichiometric constraints. In case the stoichiometry of `network.fml` is unbounded a default bound of 100 is used for all unbounded net and exchange fluxes. The resulting flux distribution is written to another FluxML file.

```
sscanner -i network.fml -o new_network.fml
```

The same as above, however, the free fluxes are initialized to represent the centroid of a cloud of 100 uniformly distributed points. If 1 instead of 100 points are used this allows a random initialization of the stoichiometry (see also ssampler(1)):

```
sscanner -m 100 -i network.fml -o new_network.fml
```

Compute the lower and upper bounds of the free net and exchange fluxes. The result is written to standard output:

```
sscanner -i network.fml -B
```

See Also

fwdsim(1), fitfluxes(1), ssampler(1)

Author

Michael Weitzel

This manpage was written by Michael Weitzel <info@13cflux.net>.

2.2.2 ssampler

ssampler — sample random flux distributions from the stoichiometry of a metabolic network.

Synopsis

`ssampler` [*options*] [-o <FILE>]

Description

Ssampler (stoichiometry sampler) is a tool for sampling uniformly distributed, random flux distributions from the stoichiometry of a metabolic reaction network. The specification of the reaction network is assumed to be contained in a FluxML document. The generated flux distributions are saved to a HDF5 file. **Ssampler** is used to generate initial feasible flux distributions and random samples for distributed multi-start optimization.

Common Options

`-h, --help` Show a brief help for all command line options.

`-i, --in <FILE> [default:stdin]` The name of the FluxML (XML) input file. If omitted, the FluxML document is expected on standard input.

`-o, --out <FILE>` The name of the HDF5 output file. This option is required (HDF5 files are never written to standard output).

`-c, --configure <CFG> [default:'default']` Because FluxML documents may contain several `<configuration/>` elements this option allows to specify the configuration that should be used for the simulation. If this option is omitted it is assumed that the FluxML document contains a configuration with the name "default".

`-L, --list` Specifying this option results in a list of allowed configuration names for the specified FluxML document. The program exits immediately after emitting the list.

`-l, --log DEST` Specify the destination for the internal logging. In the most simple case `DEST` is a file name of a log file. In case the file exists new log messages are appended. Apart from log files it is possible to publish log messages to file descriptors, UNIX domain sockets, UDP and SCTP ports, and a small graphical user interface.

A file descriptor is specified by `fd:[num]`, where `[num]` is the number of the file descriptor.

A unix domain socket in the local file system is specified by `unix:[name]`, where `[name]` is the name of the socket file

A UDP or (connectionless) SCTP port is specified by `[proto]:[host]:[port]`, where `[proto]` is either "udp" or "sctp" and `[host]` is the name of the destination host and `[port]` is a UDP or SCTP port number on the destination host. Please note that the length of log messages is bounded by the minimum safe UDP packet size - log messages containing more than 548 characters will be truncated.

Finally, log messages can also be sent to a small GUI by specifying the destination `@gui@`. The GUI requires a working Perl/Tk installation and a running X server.

In order to capture all log messages concerning the command line processing this option should be specified in front of all other options.

`-v, --verbose 0..10 [default:5]` Specify the verbosity 0, 1, ..., 10 of generated / emitted log messages. The meaning of the different log levels is as follows:

- `0 (QUIET)` do not emit log messages at all.
- `1 (ERROR)` only emit severe error messages.
- `2 (WARNING)` only report severe errors and warnings.
- `3 (NOTICE)` report all errors and warnings including important informal messages.
- `4 (INFO)` report all errors, warnings and all informal messages.
- `5 (THROW)` in case of an exception, try to give a diagnosis of the error; sometimes even gives a backtrace of the current function stack.
- `6 (DEBUG0)` emit the more important debugging messages.
- `7 (DEBUG1)` emit the less important debugging messages
- `8 (DEBUG2)` emit the superfluous debugging messages
- `9 (DEBUG3)` emit annoying debugging messages.
- `10 (DEBUG4)` don't dare to use it!

Sampling Options

`-n, --nsamples <NUM> [default:1000]` The number of samples to be generated. Defaults to 1000 if omitted.

`-s, --sample <n,x,nx> [default:n]` This option is used to specify what to sample. Possible values are 'n', 'x', and 'nx' - for sampling only net fluxes, only exchange fluxes, or both net and exchange fluxes. The default is to sample only the distribution of net fluxes.

`-S, --sampler <g,h> [default:g]` Use this parameter to specify the sampling method. The default sampling method is Gibbs sampling ('g') which usually gives the best results. Alternatively, Hit-And-Run sampling ('h') may be used (slightly faster).

`-b, --bound-net <VALUE> [default:100]` The bound for the magnitude of unbounded net fluxes. Defaults to 100.

`-p, --bound-xch <VALUE> [default:100]` The bound for unbounded exchange fluxes. Defaults to 100.

`-f, --faster` Especially if the metabolic network is very large, the sampling process may be very slow. In this case the user has to option to skip the warmup phase performed after each step of the sampler. This results in much faster sampling. However, the generated samples may be of lower quality, i.e. less uniformly distributed.

`-r, --randomize <NUM> [default:0]` This option allows to specify a randomization interval. After the specified number of steps the sampler is re-initialized with a randomized start point. Specify a value greater than 0 if you have the feeling that the generated random samples are not uniformly distributed, which is sometimes the case for a degenerate flux space. This option defaults to 0, i.e. randomization is disabled.

Examples

Sample a random net flux distribution consisting of 1000 random samples using the Gibbs sampling method (most simple case):

```
ssampler -i network.fml -o samples.h5
```

Sample net and exchange fluxes and generate 10 flux distributions. Bound the net fluxes to +/-15.5 and use the Hit-And-Run sampling method:

```
ssampler -i network.fml -o samples.h5 -n 10 -p 15.5 -S h -o network_new.fml
```

See Also

fwdsim(1), fitfluxes(1), sscanner(1)

Author

Michael Weitzel

This manpage was written by Michael Weitzel <info@13cflux.net>.

2.2.3 perturb

perturb — flux and measurement value perturbation

Synopsis

`perturb` [*`options`*]

Description

The program **perturb** is used for perturbing flux and labeling measurement values given in a FluxML document based on normally distributed random numbers. The mean of a resulting (perturbed) measurement value is given by the original measurement value. The standard deviation used for perturbation is the (scaled) standard deviation specified along with the original measurement value. On exit, **perturb** writes a modified FluxML document containing the perturbed measurement values.

This program is useful for statistical applications, like the computation of Monte Carlo bootstrap statistics.

Common Options

`-h, --help` Show a brief help for all command line options.

`-i, --in <FILE> [default:stdin]` The name of the FluxML (XML) input file. If omitted, the FluxML document is expected on standard input.

`-o, --out <FILE> [default:stdout]` The name of the output FluxML file. If this option is omitted the resulting FluxML document is written to standard output.

`-c, --configure <CFG> [default:'default']` Because FluxML documents may contain several `<configuration/>` elements this option allows to specify the configuration that should be used for the simulation. If this option is omitted it is assumed that the FluxML document contains a configuration with the name "default".

`-m, --measurements <FACTOR>` This option specifies the scaling factor for the standard deviation of the measurement values. A scaling factor of one indicated that the original standard deviation should be used (i.e. no scaling).

`-f, --fluxes <PERCENTAGE>` Perturb flux values randomly by <NUM> percent according to a uniform distribution.

`-t, --types <TYPE LIST>` This option can be used to restrict the types of measurement groups which get perturbed. TYPE LIST is a comma-separated list of measurement group types. Valid group types are: MS, MSMS, 1HNMR, 13CNMR, GENERIC, FLUX, and POOL. The default behavior is to perturb all types of measurement groups.

`-l, --log DEST` Specify the destination for the internal logging. In the most simple case `DEST` is a file name of a log file. In case the file exists new log messages are appended. Apart from log files it is possible to publish log messages to file descriptors, UNIX domain sockets, UDP and SCTP ports, and a small graphical user interface.

A file descriptor is specified by `fd:[num]`, where `[num]` is the number of the file descriptor.

A unix domain socket in the local file system is specified by `unix:[name]`, where `[name]` is the name of the socket file

A UDP or (connectionless) SCTP port is specified by `[proto]:[host]:[port]`, where `[proto]` is either "udp" or "sctp" and `[host]` is the name of the destination host and `[port]` is a UDP or SCTP port number on the destination host. Please note that the length of log messages is bounded by the minimum safe UDP packet size - log messages containing more than 548 characters will be truncated.

Finally, log messages can also be sent to a small GUI by specifying the destination `@gui@`. The GUI requires a working Perl/Tk installation and a running X server.

In order to capture all log messages concerning the command line processing this option should be specified in front of all other options.

`-v, --verbose 0..10 [default:5]` Specify the verbosity 0, 1, ..., 10 of generated / emitted log messages. The meaning of the different log levels is as follows:

- `0 (QUIET)` do not emit log messages at all.
- `1 (ERROR)` only emit severe error messages.

- `2 (WARNING)` only report severe errors and warnings.
- `3 (NOTICE)` report all errors and warnings including important informal messages.
- `4 (INFO)` report all errors, warnings and all informal messages.
- `5 (THROW)` in case of an exception, try to give a diagnosis of the error; sometimes even gives a backtrace of the current function stack.
- `6 (DEBUG0)` emit the more important debugging messages.
- `7 (DEBUG1)` emit the less important debugging messages
- `8 (DEBUG2)` emit the superfluous debugging messages
- `9 (DEBUG3)` emit annoying debugging messages.
- `10 (DEBUG4)` don't dare to use it!

Examples

There are two principal ways of using **perturb**.

Flux values are perturbed by supplying the -f flag with a percentage value. The following example will set the fluxes perturbed by 100 percent:

```
perturb -i network.fml -o network_noise.fml -f 1
```

Measurements are perturbed with respect to the defined measurement specification in the FluxML document, that is the standard deviations of the measurement device.

```
perturb -i network.fml -o network_noise.fml -m 2
```

This command will perturb all measurements by 200 percent of the standard deviation.

With parameter -t, only defined measurement groups are perturbed. The following example only perturbs MS and MSMS measurements by 100 percent of the device's standard deviation:

```
perturb -i network.fml -o network_noise.fml -m 1 -t MS,MSMS
```

Perturb also allows defining -m and -f simultaneously, causing both measurements and flux values to be perturbed:

```
perturb -i network.fml -o network_noise.fml -m 1 -f 1
```

See Also

sscanner(1), ssampler(1), setfluxes(1)

Author

Tolga Dalman and Michael Weitzel

This manpage was written by Tolga Dalman <t.dalman@fz-juelich.de>.

2.2.4 multiperturb

multiperturb — parallel 13CFLUX2 measurement and flux value perturbator

Synopsis

`multipertub` [*`options`*]

Description

Multiperturb is a parallel implemenetation of **perturb(1)**. Like the **perturb(1)** application, FluxML documents are taken as input along with several parameters. Unlike perturb(1), the output is either a directory eventually containing the result files (i.e., FluxML files containing perturbed measurements or flux values). Alternatively, **multiperturb** emits an HDF5 file containing perturbed measurements resp. flux values.

This program is especially useful for Monte Carlo Bootstrap in combination with **multifitfluxes(1)** (see EXAMPLES section below).

Common Options

-h, --help Show a brief help for all command line options.

-i, --in <FILE> [default:stdin] The name of the FluxML (XML) input file. If omitted, the FluxML document is expected on standard input.

-o, --out <PATH> [default:stdout] Output directory for FWDSIM (XML) files.

-c, --configure <CFG> [default:'default'] Because FluxML documents may contain several `<configuration/>` elements this option allows to specify the configuration that should be used for the simulation. If this option is omitted it is assumed that the FluxML document contains a configuration with the name "default".

-N, --max-procs <NUM> [default:1] Number of maximum parallel processes

-B, --block-size <NUM> [default:auto] Number of samples per task. This option controls the parallelism granularity

-l, --log DEST Specify the destination for the internal logging. In the most simple case `DEST` is a file name of a log file. In case the file exists new log messages are appended. Apart from log files it is possible to publish log messages to file descriptors, UNIX domain sockets, UDP and SCTP ports, and a small graphical user interface.

A file descriptor is specified by `fd:[num]`, where `[num]` is the number of the file descriptor.

A unix domain socket in the local file system is specified by `unix:[name]`, where `[name]` is the name of the socket file

A UDP or (connectionless) SCTP port is specified by `[proto]:[host]:[port]`, where `[proto]` is either "udp" or "sctp" and `[host]` is the name of the destination host and `[port]` is a UDP or SCTP port number on the destination host. Please note that the length of log messages is bounded by the minimum safe UDP packet size - log messages containing more than 548 characters will be truncated.

Finally, log messages can also be sent to a small GUI by specifying the destination `@gui@`. The GUI requires a working Perl/Tk installation and a running X server.

In order to capture all log messages concerning the command line processing this option should be specified in front of all other options.

-v, --verbose 0..10 [default:5] Specify the verbosity 0, 1, ..., 10 of generated / emitted log messages. The meaning of the different log levels is as follows:

- `0 (QUIET)` do not emit log messages at all.
- `1 (ERROR)` only emit severe error messages.
- `2 (WARNING)` only report severe errors and warnings.
- `3 (NOTICE)` report all errors and warnings including important informal messages.
- `4 (INFO)` report all errors, warnings and all informal messages.
- `5 (THROW)` in case of an exception, try to give a diagnosis of the error; sometimes even gives a backtrace of the current function stack.
- `6 (DEBUG0)` emit the more important debugging messages.
- `7 (DEBUG1)` emit the less important debugging messages
- `8 (DEBUG2)` emit the superfluous debugging messages
- `9 (DEBUG3)` emit annoying debugging messages.
- `10 (DEBUG4)` don't dare to use it!

Perturb Options

`-H, --hdf5-out <FILE>` Output HDF5 file containing flux samples.

`-n, --perturbations <NUM>` Number of total perturbations (either measurements of flux samples).

`-m, --measurements <NUM>` Perturb measurements using standard model error scaled by factor <NUM> . This is useful for tuning larger or smaller perturbation ranges of the input FluxML model.

`-f, --fluxes <NUM>` Perturb flux values with a randomly by <NUM> percent according to a uniform The flux distribution

`-t, --types <TYPE LIST>` When perturbing measurements, a list of measurement types can be specified instead of all types. 'LIST' is a comma-separated list of measurement types. Valid values are: MS, MSMS, 1HNMR, 13CNMR, GENERIC, FLUX, and POOL. See the EXAMPLES section for an elaborated example.

Examples

[source,shell]:

```
multiperturb -i model.fml -H output.hdf5
```

See Also

fluxml(5), perturb(1),mcbootstrap(1)

Author

This manpage was written by Tolga Dalman and Michael Weitzel <info@13cflux.net>.

2.3 Experimental Design

edscanner Given a FluxML file and a specification of a mixture of different available labeling substrates the program **edscanner** samples the possibly high dimensional mixing simplex at evenly spaced points. At each sampling point a linearized statistical analysis is performed and the volume of the ellipsoid corresponding to the fluxes' covariance matrix is computed. The coordinates of the sampling points including the volume information are written to a HDF5 file. A further analysis of this data may include the visualization of mixing triangles or the description of the resulting statistical data using high-dimension density functions.

Important command line switches:

- **`-m`** input file containing mixture specification
- **`-n`** number of samples
- **`-o`** name of HDF5 output file

edopt Given a FluxML file with an optimized flux distribution the program **edopt** can be used to generate an optimal design for a subsequent isotope labeling experiment. For this purpose, **edopt** performs a linearized statistical analysis and determines the mixture of substrates which minimizes the volume of the flux distribution's covariance ellipsoid, i.e. allows the most accurate flux determination having the smallest standard deviations.

Important command line switches:

- **`-m`** input file containing mixture specification
- **`-O`** choice of optimization algorithm
- **`-r`** number of randomly initialized multi-start runs

2.3.1 edscanner

edscanner — experimental design scanner (for statistical mixture quality).

Synopsis

`edscanner` [*`options`*]

Description

Given a FluxML file and a specification of a mixture of different available labeling substrates the program **edscanner** samples the possibly high dimensional mixing simplex at evenly spaced points. At each sampling point a linearized statistical analysis is performed and the volume of the ellipsoid corresponding to the fluxes' covariance matrix is computed.

The coordinates of the sampling points including the volume information are written to a HDF5 file. A further analysis of this data may include the visualization of mixing triangles (using the provided MATLAB script drawMixingTriangle.m) or the description of the resulting statistical data using high-dimension density functions.

Be aware that successful experimental design may require to specify an error model the extrapolation of standard errors of simulated measurements. Details can be found in **fluxml**(5) (element "errormodel").

Options

`-h, --help` Show a brief help for all command line options.

`-i, --in <FILE> [default:stdin]` The name of the FluxML (XML) input file. If omitted, the FluxML document is expected on standard input.

`-m, --mixture <FILE>` The file containing the substrate specification. See below for the format of this file.

`-o, --out <FILE>` Name of an output HDF5 file. This option is required. The generated HDF5 file contains a single matrix with the path '`/mix/matrix`'. For `n` different mixing partners each row first contains the `n-1` cartesian coordinates `x_1, ..., x_{n-1}`, followed by the `n` barycentric coordinates `t_1, ..., t_n`, and finally the information index (optimality criterion) of the covariance matrix (cf. option `-C`):

`((x_1, ..., x_n-1, t_1, ..., t_n,)+ inf.index, cost)`

Dataset '`/mix/c`' contains the column labels for matrix '`/mix/matrix`'. The value '`/mix/central_value`' contains the information index of the covariance matrix corresponding to the center of the mixing simplex, i.e. same proportions of all mixing partners.

`-n, --nsamples <NUM> [default:10]` The number of samples to be generated. Defaults to 10.

`-C, --criterion [ADEM|expr] [default :D]` This option allows to choose between different statistical optimality criteria / information indices computed from the covariance matrix of free fluxes. The four basic choices are (Atkinson and Donev: Optimum experimental designs. Oxford University Press, 1992):

- `A` (A-optimality): the value trace(C)/n, where trace(C) is the sum of the free fluxes' variances and n=dim(C) is the number of degrees of freedom (i.e. the number of free fluxes).
- `D` (D-optimality): the value det(C)^(1/n), where det(C) is the determinant of the free fluxes' covariance matrix and n=dim(C) is the number of degrees of freedom (i.e. the number of free fluxes).
- `E` (E-optimality): the maximum eigenvalue of the covariance matrix (proportional to the length of the longest principal axis of the covariance ellipsoid).
- `M` (M-optimality): the maximum variance, i.e. the maximum diagonal element of the covariance matrix (the square of the largest confidence interval).

In addition to the four basic choices, this command line option allows to specify an arbitrary formula including also the following variables:

- `t`, the trace of the free fluxes' covariance matrix.
- `Dim`, the dimension of the full covariance matrix (including all fluxes).
- `dim`, the dimension of the covariance matrix. This value is identical to the number of the free fluxes and the degrees of freedom.
- `d`, the determinant of the free fluxes' covariance matrix.
- `e`, the minimum eigenvalue of the covariance matrix.
- `c`, the minimum confidence interval / standard deviation, i.e. the root of the minimum diagonal value.
- `S`, the maximum standard deviation, i.e. the root of the maximum diagonal value.
- `s`, the minimum standard deviation, i.e. the root of the minimum diagonal value.

Allowed operators include: +, -, *, /, ^, abs(x), sqr(x), sqrt(x,y), exp(x), log(x), log2(x), log10(x), min(x,y), max(x,y), =, !=, <, > <=, >=

`-L, --list` Specifying this option results in a list of allowed configuration names for the specified FluxML document. The program exits immediately after emitting the list.

`-c, --configure <CFG> [default:'default']` Because FluxML documents may contain several `<configuration/>` elements this option allows to specify the configuration that should be used for the simulation. If this option is omitted it is assumed that the FluxML document contains a configuration with the name "default".

`-t, --tolerance <val> [default:1e-9]` Specifies a constraint violation tolerance value. This parameter can be used to tolerate a certain constraint violation. Use with care.

`-l, --log DEST` Specify the destination for the internal logging. In the most simple case `DEST` is a file name of a log file. In case the file exists new log messages are appended. Apart from log files it is possible to publish log messages to file descriptors, UNIX domain sockets, UDP and SCTP ports, and a small graphical user interface.

A file descriptor is specified by `fd:[num]`, where `[num]` is the number of the file descriptor.

A unix domain socket in the local file system is specified by `unix:[name]`, where `[name]` is the name of the socket file

A UDP or (connectionless) SCTP port is specified by `[proto]:[host]:[port]`, where `[proto]` is either "udp" or "sctp" and `[host]` is the name of the destination host and `[port]` is a UDP or SCTP port number on the destination host. Please note that the length of log messages is bounded by the minimum safe UDP packet size - log messages containing more than 548 characters will be truncated.

Finally, log messages can also be sent to a small GUI by specifying the destination `@gui@`. The GUI requires a working Perl/Tk installation and a running X server.

In order to capture all log messages concerning the command line processing this option should be specified in front of all other options.

`-v, --verbose 0..10 [default:5]` Specify the verbosity 0, 1, ..., 10 of generated / emitted log messages. The meaning of the different log levels is as follows:

- `0 (QUIET)` do not emit log messages at all.
- `1 (ERROR)` only emit severe error messages.
- `2 (WARNING)` only report severe errors and warnings.
- `3 (NOTICE)` report all errors and warnings including important informal messages.
- `4 (INFO)` report all errors, warnings and all informal messages.
- `5 (THROW)` in case of an exception, try to give a diagnosis of the error; sometimes even gives a backtrace of the current function stack.
- `6 (DEBUG0)` emit the more important debugging messages.
- `7 (DEBUG1)` emit the less important debugging messages
- `8 (DEBUG2)` emit the superfluous debugging messages
- `9 (DEBUG3)` emit annoying debugging messages.
- `10 (DEBUG4)` don't dare to use it!

Solver Options

`-a, --apsolve` Solve the individual linear equation systems of the cumomer or EMU cascade using arbitrary precision arithmetic. Note that the results are converted to double precision immediately after solution. This option eliminates any round-off for the solution of the individual network levels.

`-A, --exact` In addition to option `-a` absolutely all computations are carried out using arbitrary precision arithmetic. If this option is used in conjunction with an analytical gradient (`-g analytic`) this results in exact derivatives. The results are converted to double precision just before writing them to the output file. This option is intended for debugging purposes. For large networks models the use of arbitrary precision arithmetic is probably to expensive.

`-d, --dbgsolve` When this option is used the network model is simulated in numerical debugging mode: for every equation system condition numbers are computed and the residual of the solution is checked.

Mixture Specification

The XML document for mixture specification is part of the FluxML document format and collects an arbitrary number of <input/> elements for possible different substrate pools under one <mixture/> element. In order to allow a further description of the resulting mixture each of the <input/> is required to have an `id` attribute with unique value. Example (including corrections for natually occuring isotopes):

```
<?xml version="1.0"?>
<mixture xmlns="http://www.13cflux.net/fluxml">
  <input id="first_mix" pool="A" type="isotopomer">
    <label cfg="110" purity="0.98">1</label>
  </input>
  <input id="second_mix" pool="A" type="isotopomer">
    <label cfg="011" purity="0.96">1</label>
  </input>
  <input id="third_mix" pool="A" type="isotopomer">
    <label cfg="101" purity="0.99">1</label>
  </input>
</mixture>
```

Examples

Raster a possibly high dimensional mixing simplex with 500 evenly spaced samples. Carry-out all computations using arbitrary precision arithmetic:

```
edscanner -i network.fml -m mixture.mix -o samples.h5 -n 500 -A
```

In case there are only three mixing partners the result can be visualized in form of a triangle plot using the supplied MATLAB script `drawMixingTriangle.m`:

```
>> drawMixingTriangle('samples.h5');
```

See `help drawMixingTriangle` for more info.

See Also

edopt(1), fwdsim(1)

Author

Michael Weitzel

This manpage was written by Michael Weitzel <info@13cflux.net>.

2.3.2 edopt

edopt — substrate mixture optimization (experimental design).

Synopsis

`edopt` [*options*]

Description

Given a FluxML file with an optimized flux distribution the program **edopt** can be used to generate an optimal design for a subsequent isotope labeling experiment.

For this purpose, **edopt** performs a linearized statistical analysis and determines the mixture of substrates which minimizes the volume of the flux distribution's covariance ellipsoid, i.e. allows the most accurate flux determination having the smallest standard deviations.

Be aware that successful experimental design may require to specify an error model the extrapolation of standard errors of simulated measurements. Details can be found in **fluxml**(5) (element "errormodel").

Options

-h, --help Show a brief help for all command line options.

-i, --in <FILE> [default:stdin] The name of the FluxML (XML) input file. If omitted, the FluxML document is expected on standard input.

-m, --mixture <FILE> The file containing the substrate specification. See below for the format of this file.

-o, --out <FILE> [default:stdout] Name of an XML document (FluxML fragment) containing the optimized mixture. If this parameter is omitted the optimized substrate mixture is written to standard output.

-O, --optimizer <PKG> [default:IPOPT] The optimization algorithm to be used. The default is to use Ipopt. Other possible choices are CFSQP, NAGNLP, and SUBPLEX (CFSQP and NAGNLP are commercial and may not be accessible for your installation).

-P, --properties <arg> This option allows fine-grained control over the optimization algorithm's behavior and settings. The argument to option `-P` is a comma-separated list of key-value pairs. The general syntax is: `[optimizer].[property]=[type]([value])`.

Valid `[optimizer]`'s are '`ipopt`' and '`cfsqp`'.

Valid `[type]`'s are:

- `integer` - in this case `[value]` has to be an integer.
- `string` - the contents of `[value]` are allowed to be any string.
- `real` - for real (double precision) values.
- `boolean` - restricts the contents of `[value]` to the values `true` and `false`.

Using this interface all of the numerous settings of Ipopt can be accessed. A full list of valid options can be obtained by setting the boolean property `ipopt.print_options_documentstion` to `true`. See the examples below. Other interesting properties include:

- `ipopt.max_cpu_time` - allows to abort Ipopt after a specified number of CPU seconds is exceeded.
- `ipopt.print_level` - the verbosity of Ipopt. Valid settings are integers from 0 to 12. The default setting is 5.
- `ipopt.max_iter` - the maximum number of iterations before termination. The default value is 3000.
- `ipopt.linear_solver` - allows to choose the linear solver used by Ipopt. The default setting is '`mumps`'. Using another solver may improve convergence speed and optimization results.

CFSQP supports significantly fewer properties (a more detailed description can be found in the CFSQP documentation):

- `cfsqp.mode` - the type and mode of the used solver. The default setting is 200.
- `cfsqp.iprint` - the verbosity of CFSQP. The default setting is 1. Other settings are 0 (quiet), 2, or 3 (more verbose output).
- `cfsqp.miter` - the maximum number of iterations before termination. The default value is 1000.
- `cfsqp.bigbnd` - allows setting the value which plays the role of infinity. The default value is 1e20.
- `cfsqp.eps` - final norm of the newton gradient. Must be bigger than the machine epsilon. The default value is 1e-10.

NAGNLP refers to the "nag_opt_nlp" ("e04ucc") optimizer found in the commercial NAGC library. Most important settings are accessible via the following properties (cf. NAGC documentation):

- `nag_opt_nlp.max_iter` - the maximum number of iterations before termination. The default value is 250.
- `nag_opt_nlp.list` - print settings. The default value is "false".
- `nag_opt_nlp.print_level` - the major print level. The default setting is "Nag_Soln_Iter". See the NAGC documentation for other available settings.
- `nag_opt_nlp.minor_print_level` - the minor print level. The default setting is "Nag_NoPrint". See the NAGC documentation for other available settings.
- `nag_opt_nlp.verify_grad` - allows to enable the gradient checker. The default setting is "Nag_NoCheck". Set to "Nag_SimpleCheck" to turn on gradient checking.

`-r, --random <NUM> [default:1]` The number of randomly initialized multi-start optimization runs. This option defaults to 1, i.e. single optimization run.

`-C, --criterion [ADEM|expr] [default :D]` This option allows to choose between different statistical optimality criteria / information indices computed from the covariance matrix of free fluxes. The four basic choices are (Atkinson and Donev: Optimum experimental designs. Oxford University Press, 1992):

- `A` (A-optimality): the value trace(C)/n, where trace(C) is the sum of the free fluxes' variances and n=dim(C) is the number of degrees of freedom (i.e. the number of free fluxes).
- `D` (D-optimality): the value det(C)^(1/n), where det(C) is the determinant of the free fluxes' covariance matrix and n=dim(C) is the number of degrees of freedom (i.e. the number of free fluxes).
- `E` (E-optimality): the maximum eigenvalue of the covariance matrix (proportional to the length of the longest principal axis of the covariance ellipsoid).
- `M` (M-optimality): the maximum variance, i.e. the maximum diagonal element of the covariance matrix (the square of the largest confidence interval).

In addition to the four basic choices, this command line option allows to specify an arbitrary formula including also the following variables:

- `t`, the trace of the free fluxes' covariance matrix.
- `Dim`, the dimension of the full covariance matrix (including all fluxes).
- `dim`, the dimension of the covariance matrix. This value is identical to the number of the free fluxes and the degrees of freedom.
- `d`, the determinant of the free fluxes' covariance matrix.
- `e`, the minimum eigenvalue of the covariance matrix.
- `c`, the minimum confidence interval / standard deviation, i.e. the root of the minimum diagonal value.
- `S`, the maximum standard deviation, i.e. the root of the maximum diagonal value.
- `s`, the minimum standard deviation, i.e. the root of the minimum diagonal value.

Allowed operators include: +, -, *, /, ^, abs(x), sqr(x), sqrt(x,y), exp(x), log(x), log2(x), log10(x), min(x,y), max(x,y), =, !=, <, > <=, >=

`-L, --list` Specifying this option results in a list of allowed configuration names for the specified FluxML document. The program exits immediately after emitting the list.

`-c, --configure <CFG> [default:'default']` Because FluxML documents may contain several `<configuration/>` elements this option allows to specify the configuration that should be used for the simulation. If this option is omitted it is assumed that the FluxML document contains a configuration with the name "default".

`-l, --log DEST` Specify the destination for the internal logging. In the most simple case `DEST` is a file name of a log file. In case the file exists new log messages are appended. Apart from log files it is possible to publish log messages to file descriptors, UNIX domain sockets, UDP and SCTP ports, and a small graphical user interface.

A file descriptor is specified by `fd:[num]`, where `[num]` is the number of the file descriptor.

A unix domain socket in the local file system is specified by `unix:[name]`, where `[name]` is the name of the socket file

A UDP or (connectionless) SCTP port is specified by `[proto]:[host]:[port]`, where `[proto]` is either "udp" or "sctp" and `[host]` is the name of the destination host and `[port]` is a UDP or SCTP port number on the destination host. Please note that the length of log messages is bounded by the minimum safe UDP packet size - log messages containing more than 548 characters will be truncated.

Finally, log messages can also be sent to a small GUI by specifying the destination `@gui@`. The GUI requires a working Perl/Tk installation and a running X server.

In order to capture all log messages concerning the command line processing this option should be specified in front of all other options.

`-v, --verbose 0..10 [default:5]` Specify the verbosity 0, 1, ..., 10 of generated / emitted log messages. The meaning of the different log levels is as follows:

- `0 (QUIET)` do not emit log messages at all.
- `1 (ERROR)` only emit severe error messages.
- `2 (WARNING)` only report severe errors and warnings.
- `3 (NOTICE)` report all errors and warnings including important informal messages.
- `4 (INFO)` report all errors, warnings and all informal messages.
- `5 (THROW)` in case of an exception, try to give a diagnosis of the error; sometimes even gives a backtrace of the current function stack.
- `6 (DEBUG0)` emit the more important debugging messages.
- `7 (DEBUG1)` emit the less important debugging messages
- `8 (DEBUG2)` emit the superfluous debugging messages
- `9 (DEBUG3)` emit annoying debugging messages.
- `10 (DEBUG4)` don't dare to use it!

Solver Options

`-a, --apsolve` Solve the individual linear equation systems of the cumomer or EMU cascade using arbitrary precision arithmetic. Note that the results are converted to double precision immediately after solution. This option eliminates any round-off for the solution of the individual network levels.

`-A, --exact` In addition to option `-a` absolutely all computations are carried out using arbitrary precision arithmetic. If this option is used in conjunction with an analytical gradient (`-g analytic`) this results in exact derivatives. The results are converted to double precision just before writing them to the output file. This option is intended for debugging purposes. For large networks models the use of arbitrary precision arithmetic is probably to expensive.

`-d, --dbgsolve` When this option is used the network model is simulated in numerical debugging mode: for every equation system condition numbers are computed and the residual of the solution is checked.

Mixture Specification

The XML document for mixture specification is part of the FluxML document format and collects an arbitrary number of <input/> elements for possible different substrate pools under one <mixture/> element. In order to allow a further description of the resulting mixture each of the <input/> is required to have an `id` attribute with unique value. Example (including corrections for natually occuring isotopes):

```
<?xml version="1.0"?>
<mixture xmlns="http://www.13cflux.net/fluxml">
  <input id="first_mix" pool="A" type="isotopomer">
    <label cfg="110" purity="0.98">1</label>
  </input>
  <input id="second_mix" pool="A" type="isotopomer">
    <label cfg="011" purity="0.96">1</label>
  </input>
  <input id="third_mix" pool="A" type="isotopomer">
    <label cfg="101" purity="0.99">1</label>
  </input>
</mixture>
```

Examples

Optimize a subtrate mixture 50 times starting from a randomized initialization:

```
edopt -r 50 -i network.fml -m mixture_in.mix -o mixture_out.mix
```

See Also

fitfluxes(1), edscanner(1)

Author

Michael Weitzel
This manpage was written by Michael Weitzel <info@13cflux.net>.

2.4 Metabolic Network Conversion

ftbl2fml The program **ftbl2fml** converts a metabolic network model in FTBL file format (old 13CFLUX) into the new FluxML (XML) document format (see **fluxml**(5)). This conversion includes the parametrization of the stoichiometry, constraint (in)equalities, measurement specifications and measurement data. Optionally, **ftbl2fml** will try to preserve the comments contained in the FTBL file.

Important command line switches:

`-c, -d` preserve original FTBL comments

`-l` list names of all (removed) sink pools in a XML comment

sbml2fml The small script sbml2fml converts the metabolic network model contained in a SBML file into a FluxML document. Because SBML does not allow to specify labeling positions and atom transitions the resulting network model is purely stoichiometric.

Important command line switches:

`-p` create a pretty-printed FluxML document

`-v` create a FluxML document which can be validated (xmllint)

fml2sbml The small script sbml2fml converts the metabolic network model contained in a FluxML document into a SBML document. N.b.: SBML does not allow to specify labeling positions and atom transitions.

2.4.1 ftbl2fml

ftbl2fml — create a FluxML file from an old FTBL file.

Synopsis

`ftbl2fml` [*`options`*]

Description

Ftbl2fml converts a metabolic network model in FTBL file format (old 13CFLUX) into the new FluxML (XML) document format. This conversion includes the parametrization of the stoichiometry, constraint (in)equalities, measurement specifications and measurement data. Optionally, **ftbl2fml** will try to preserve the comments contained in the FTBL file.

Although **ftbl2fml** will, in general, do a good job the user of this program should be aware that there is a small semantic difference between FTBL and FluxML: 13CFLUX 2 and FluxML do not support 'output pools'. From the perspective of 13CFLUX 2 a metabolic network specification consists of

- substrate pools (aka 'input pools') carrying isotopic labeling information
- inner pools (aka 'intracellular pools'), from which a subset is required to be simulated in order to evaluate the measurement specifications.
- metabolic reactions (aka 'fluxes') connecting the inner pools.
- effluxes leaving the system and necessary to formulate the mass balance.

In addition, a network specification for the old 13CFLUX contains 'output pools' which collect all material transported by the effluxes. Because these output pools, on the one hand, violate mass conservation and, on the other hand, artificially increase the size of the network, output pools are no longer supported by 13CFLUX 2.

In **ftbl2fml**, output pools are automatically removed during conversion. Although this should be fine for most network specifications it results in an error if an isotopomer distribution of a removed output pool is required for the evaluation of a measurement specification. In case this problem occurs, it is necessary to reintroduce the output pool and an additional efflux, manually.

Common Options

`-h, --help` Show a brief help for all command line options.

`-i, --in <FILE> [default:stdin]` The name of the FTBL input file.

`-o, --out <FILE> [default:stdout]` The name of the FluxML (XML) output file.

`-c, --comments` Tries to preserve FTBL comments. FTBL comments are woven into the FluxML file in form of XML comments. As far as possible, this is done at the corresponding positions.

`-d, --unmapped` Preserves the comments of the unmapped sections of the FTBL file, i.e. the sections which are not converted into the FluxML file. This option requires option `-c`.

`-e, --effluxes` Preserves all output pools of the FTBL file by appending extra effluxes to them. The use of this option enlarges the network model, so that the simulation problem solved by 13CFLUX 2 is actually larger than the problem exposed to the old 13CFLUX. The default name of the newly introduced effluxes is 'efflux_' (see option `-e`) followed by the name of the corresponding output pool. Use this option with care.

`-f, --fprefix <PREFIX> [default:efflux_]` Specifies the prefix of the new effluxes introduced if option `-e` is given. The default prefix is 'efflux_'.

-F, --freeconst Transform the values of the free fluxes in the input FTBL file into constraints in the generated FluxML document.

-k, --noautoms By default, and if applicable, the sums of labeling patterns in section LABEL_MEASUREMENTS are translated into compact MS specifications. The resulting MS specifications are easier to read and can be simulated with greater efficiency (EMU method). Although this translation should always be correct there might be an interest to retain the original sum formulas. In this case automatic translation can be turned of by specifying this parameter.

-l, --listsinks Create an XML comment containing the names of all removed output pools.

-m, --mathml Translate all constraint equalities and inequalities into the machine-readable Content-MathML. Since 13CFLUX 2 also allows a human-readable textual notation it should not be necessary to specify this parameter.

-n, --nonpretty Normally, **ftbl2fml** generates human-readable, pretty-printed FluxML. If no human intends to read / edit the resulting FluxML document this option may be specified.

-r, --remsinks <REGEX> Using this parameter it is possible to remove only selected output pools whose name matches a given regular expression. See regex(7) for the syntax of regular expressions. This option is intended for expert use only. Do not use it.

-t, --tsfromfile If this option is specified the timestamp in the FluxML info header is generated from the timestamp of the FTBL input file. If this option is omitted **ftbl2fml** tries to parse the timestamp in the FTBL file. This is often problematic since this timestamp does not follow a fixed syntax.

-v, --valid Create a FluxML document that can be validated. If this option is specified the generated FluxML document can be validated by external XML tools, such as xmllint(1). Since the FluxML parser will validate the generated FluxML document it is not explicitly necessary to specify this option.

-y, --pypretty This option enforces to use Python/minidom for pretty printing rather than the external tool xmllint(1), which is the default and results in prettier FluxML. In case xmllint(1) is not available Python/minidom is used automatically.

-z, --gzip Compress the generated FluxML document using the gzip(1) compression algorithm. Be aware that you have to specify an output file (option `-o`) whose name should end with '.gz' - **ftbl2fml** will not write compressed data to its stdout.

Examples

Convert an FTBL file into a FluxML file using the default settings (most simple case):

```
ftbl2fml -i network.ftbl -o network.fml
```

Convert an old FTBL to FluxML, simulate and filter out the simulated measurement values (**ftbl2fml** reads from stdin and writes to stdout):

```
ftbl2fml < network.ftbl | fwdsim | fwdsimflt -m
```

Convert an FTBL file into a compressed FluxML, preserve comments, list removed output pools, and encode all formulas using Content-MathML:

```
ftbl2fml -clmz -i network.ftbl -o network.fml.gz
```

See Also

xmllint(1), fwdsim(1), fmllint(1)

Author

Michael Weitzel
This manpage was written by Michael Weitzel <info@13cflux.net>.

2.4.2 sbml2fml

sbml2fml — import a SBML metabolic network model into FluxML

Synopsis

`sbml2fml` [*`options`*]

Description

The small script **sbml2fml** converts the metabolic network model contained in a SBML file into a FluxML document. Because SBML does not allow to specify labeling positions and atom transitions the resulting network model is purely stoichiometric.

Common Options

`-h, --help` Show a brief help for all command line options.

`-i, --in <FILE> [default:stdin]` The name of the SBML (XML) input file.

`-o, --out <FILE> [default:stdout]` The name of the generated FluxML (XML) output file.

`-p, --pretty` Specifying this option results in pretty-printed (i.e. human readable) FluxML. The external tool **xmllint**(1) is used for pretty-printing.

`-y, --pypretty` This option results, as well, in pretty-printed FluxML. The pretty-printing, however, is done using Python/minidom module.

`-z, --gzip` If this option is specified, the resulting FluxML document is compressed using a **gzip**(1) compatible compression.

Examples

Convert a SBML file into a FluxML file using the default settings (most simple case):

```
sbml2fml -i network.sbml -o network.fml
```

Convert a SBML file into a pretty-printed FluxML file:

```
sbml2fml -p -o network.fml < network.sbml
```

See Also

ftbl2fml(1), fml2sbml(1)

Author

Michael Weitzel
This manpage was written by Michael Weitzel <info@13cflux.net>.

2.4.3 fml2sbml

fml2sbml — export a FluxML reaction network model into a SBML document

Synopsis

`fml2sbml` [*`options`*]

Description

The small script **sbml2fml** converts the metabolic network model contained in a FluxML document into a SBML document. N.b.: SBML does not allow to specify labeling positions and atom transitions.

Common Options

`-h, --help` Show a brief help for all command line options.

`-i, --in <FILE> [default:stdin]` The name of the FluxML (XML) input file. If omitted, the FluxML document is expected on standard input.

`-o, --out <FILE> [default:stdout]` The name of the generated SBML (XML) output file.

Examples

Convert a FluxML file into a SBML file:

```
fml2sbml -i network.fml -o network.sbml
```

See Also

ftbl2fml(1), sbml2fml(1)

Author

Michael Weitzel
This manpage was written by Michael Weitzel <info@13cflux.net>.

2.5 Data Transfer and Report Generation

setfluxes Setfluxes is a tool for transferring settings of free fluxes from FWDSIM (XML) (see **fwdsim**(5)), HDF5, or CSV documents to FluxML.

Important command line switches:

`-C,-F,-H` the name of a CSV, FWDSIM(XML), or HDF5 file containing flux values

`-l` a line (row) number referring to a specific flux distribution (CSV, HDF5)

`-f` force to replace the free fluxes

fwdsimflt The tool **fwdsimflt** (**fwdsim**(5) filter) can be used to filter the FWDSIM (XML) documents generated by the commands **fwdsim**(1) and **fitfluxes**(1) to a human-readable CSV format.

Important command line switches:

`-m,-s,-t` output the simulated measurement values, flux distribution, or standard deviations

`-U` output all computed unknowns and translate them into isotopomer fractions

`-f` filter the output using a regular expression

simreport Given a FluxML file and a corresponding simulation results in form of a FWDSIM (XML) file simreport generates a CSV file comparing the given and simulated measurement values. Norm values, measuring the discrepancy between measurement and simulation are provided for all measurement values.

Important command line switches:

`-i` the name of the FluxML file

`-f` the name of the FWDSIM (XML) file belonging to the specified FluxML file

`-o` the name of the generated CSV file containing the report

collectfitdata The program **collectfitdata** is used for collecting together the results (i.e. net fluxes, exchange fluxes, and measurement values) from multiple FWDSIM (XML) files (see **fwdsim**(5)). The collected data is saved to a HDF5 file.

Important command line switches:

`-f` the path to the directory containing the FWDSIM (XML) files

- **`-a`** run in "append mode"
- **`-m`** include measurment values into the output

fwdsim2csv The script **fwdsim2csv** is used to transfer the flux settings from one or multiple FWDSIM (XML) document, possibly generated by the commands **fitfluxes**(1) or **multifit**(1), into human-readable CSV data. The CSV data is always written to standard output.

Important command line switches:

- **`-a`** include all fluxes in the output, not just the free fluxes.
- **`-n, -x`** output only net (`-n`) or only xch (`-x`) fluxes
- **`-f`** include the file names of the FWDSIM (XML) files in a separate column

hdf5tocsv The script **hdf5tocsv** converts an XML file to a CSV formatted output. It is a helper function for post-processing of simulation results.

Important command line switches:

- **`-D`** HDF5 path to the data matrix in the HDF5 file
- **`-C`** HDF5 path to the header column texts in the HDF5 file
- **`-d`** select a different seperator character in the output CSV file
- **`-H`** the output of the column texts is omitted

2.5.1 setfluxes

setfluxes — transfer (free) flux settings from FWDSIM, HDF5, or CSV to FluxML.

Synopsis

`setfluxes` [*`options`*]

Description

Setfluxes is a tool for transferring settings of free fluxes from FWDSIM (XML), HDF5, or CSV documents to FluxML. Most common use cases are:

1. using **fitfluxes**(1), you generated optimized flux settings which are now available in a FWDSIM (XML) file. In order to do linearized statistics with **fwdsim**(1) you use **setfluxes** for transferring the flux settings to a FluxML document.
2. using **ssampler**(1), you generated thousands of random flux distributions in order to do a multi-start optimization with **fitfluxes**(1). You can now use **setfluxes**(1) to write the flux distributions, one after the other, to a FluxML document.
3. using the spreadsheet application of your choice, you created a CSV file containing a flux distribution for your network. Instead of manually editing the FluxML fail you can use **setfluxes** to automate this task.

Common Options

`-h, --help` Show a brief help for all command line options.

`-i, --in <FILE> [default:stdin]` The name of the FluxML (XML) input file. If omitted, the FluxML document is expected on standard input.

`-o, --out <FILE> [default:stdout]` The name of the FluxML output file. If omitted, the modified FluxML document is written to standard output.

`-c, --configure <CFG> [default:'default']` Because FluxML documents may contain several `<configuration/>` elements this option allows to specify the configuration that should be used for the simulation. If this option is omitted it is assumed that the FluxML document contains a configuration with the name "default".

Data Source Options

`-F, --fwd <FILE>` Specify a FWDSIM (XML) file as data source, usually generated by either **fwdsim**(1) or **fitfluxes**. The layout of these files is documented in manual page **fwdsim**(5).

`-H, --hdf <FILE>` Use a HDF5 file as data source. The HDF5 file is expected to have the following layout:

- `'/flux/names'` should be a vector of strings containing the flux names with suffixes '.n' or '.x' for net and exchange fluxes
- `'/flux/data'` should be a matrix of double precision floating point numbers. The columns of the matrix are associated with the flux names in the elements of `'/flux/names'`

In case the line number (option `-l`) is omitted it is assumed that the first dataset in line 1 should be transferred.

`-C, --csv <FILE>` Use this parameter to specify a CSV file containing the flux distribution. The CSV file should have the following layout:

- a column heading (the first row) should list the flux names with suffixes '.n' or '.x' for net and exchange fluxes
- starting with the second row each row of the CSV file contains one complete flux setting

In case the line number (option `-l`) is omitted it is assumed that the first dataset in line 1 (row 2) should be transferred.

`-l, --line <LINENUM> [default:1]` When using a HDF5 or CSV file as data source, specify the line number (row number) of the flux distribution which should be written to the generated FluxML document. If this parameter is omitted the first first flux distribution is selected.

Special Options

`-f, --force` Write all flux settings to the generated FluxML document without checking whether they existed in the input FluxML document. Use this option if your input FluxML document contained no settings for the free fluxes, or the choice of free fluxes in the input FluxML file does not match the choice in the data source. If this parameter is not specified **setfluxes** tries to detect non-matching combinations of data source and FluxML input document.

`-D, --data <PATH> [default:'/flux/data']` For a HDF5 data source, specify the path to the flux values. Use with care.

`-N, --names <PATH> [default:'/flux/names']` For a HDF5 data source, specify the path to the flux names. Use with care.

Examples

Transfer the flux distribution found in a FWDSIM document to a corresponding FluxML document network.fml (most simple case):

```
    setfluxes -i network.fml -F network.fwd -o network_new.fml
```

Transfer the 123'rd flux distribution found in a HDF5 file samples.h5 to a corresponding FluxML document network.fml:

```
    setfluxes -i network.fml -H samples.h5 -l 123 -o network_new.fml
```

Convert an old FTBL network specification to FluxML and replace the flux settings by the values contained in line 5 of a CSV file. Simulate the result using **fwdsim**(1):

```
    ftbl2fml -i network.ftbl | setfluxes -C samples.csv -l 5 | fwdsim -o resu
lts.fwd
```

See Also

fwdsim(1), fitfluxes(1), ftbl2fml(1)

Author

Michael Weitzel and Tolga Dalman
This manpage was written by Michael Weitzel <info@13cflux.net>.

2.5.2 setmeasurements

setmeasurements — Transferring labeling and flux measurements from FWDSIM, FluxML or CSV to FluxML.

Synopsis

`setmeasurements` [*`options`*]

Description

setmeasurements is a tool for transferring labeling or flux measurements from FWDSIM, FluxML or CSV documents to FluxML. Most common use cases are:

1. using **fwdsim** with simulation type full to generate a set of labeling and flux measurements in a FWDSIM file.
2. building up defined flux and labeling measurements in a CSV file to import them into a FluxML file.
3. copying a set of measurement data from an existing FluxML file to a new FluxML file without data.

Common Options

`-h, --help` Show a brief help for all command line options.

`-i, --in <FILE> [default:stdin]` The name of the FluxML (XML) input file. If omitted, the FluxML document is expected on standard input.

`-o, --output <FILE> [default:stdout]` The name of the FluxML output file. If omitted, the modified FluxML document is written to standard output.

`-c, --configure <CFG> [default:'default']` Because FluxML documents may contain several `<configuration/>` elements this option allows to specify the configuration that should be used for the simulation. If this option is omitted it is assumed that the FluxML document contains a configuration with the name "default".

Data Source Options

`-F, --fwd <FILE>` Specify a FWDSIM (XML) file as data source, usually generated by either **fwdsim**(1) or **fitfluxes**. The layout of these files is documented in manual page **fwdsim**(5).

`-X, --fml <FILE>` Use a FluxML file as data source.

`-C, --csv <FILE>` Use this parameter to specify a CSV file containing the flux and labeling measurements. The CSV file should have the following layout:

Table 2.1 Format of CSV labeling and flux measurements

#specifications	**measured values**	**measured stddevs**
net:co2_exp	4.147718258	0.538999605
"PGA#M0,1,2,3"	0.54637224	0.007434449
	0.129020493	0.001872506
	0.070576643	0.001663439
	0.254030624	0.004767783

The first column contains the specifications of the used flux or labeling measurement. Fluxes has to be marked with "net:" for automated detection. In the second column the measured value of the flux or the labeling fraction is stored with the corresponding standard deviation in the third column. The specifications of the used measurement groups is denoted in the .xsd scheme at www.13cflux.net/fluxml. The CSV file should use comma as separator. All values without a specification are added to the previous analyte.

`-f, --filter_flux <REGEX>` Only output the data sets which labeling measurement group name is matched by the specified regular expression. If omitted all measurement groups are imported. Only in combination with -F. See regex(7) for the syntax of regular expressions.

`-l, --filter_label <REGEX>` Only output the data sets which flux measurement name is matched by the specified regular expression. If omitted, none of the flux measurements is imported. Only in combination with -F. See regex(7) for the syntax of regular expressions.

Special Options

`-R, --num <NUMBER> [default:0.05]` Relative standard deviation for measurement data from FWDSIM files. Only in combination with -F.

`-A, --num <NUMBER> [default:0.01']` Absolute standard deviation for measurement data from FWDSIM files. Only in combination with -F.

Examples

Transfer all labeling measurements found in a FWDSIM document to a corresponding FluxML document network.fml (most simple case):

```
setmeasurements -i network.fml -F network.fwd -o network_new.fml
```

As above with specific absolute and relative deviations:

```
setmeasurements -i network.fml -F network.fwd -o network_new.fml -R 0.1 -
A 0.02
```

Import labeling and flux measurements from an CSV file to a FluxML document network.fml:

```
setmeasurements -i network.fml -C measurement_data.csv -o network_new.fml
```

Transfer all labeling measurements found in a FluxML document to a corresponding FluxML document network.fml:

```
setmeasurements -i network.fml -X network2.fml -o network_new.fml
```

See Also

setfluxes(1), fwdsim(1), fitfluxes(1), ftbl2fml(1)

Author

Stephan Miebach
This manpage was written by Stephan Miebach and Michael Weitzel <info@13cflux.net>.

2.5.3 fwdsimflt

fwdsimflt — filter the output of the FWDSIM (XML) documents generated by **fwdsim**(1) and **fitfluxes**(1)

Synopsis

`fwdsimflt [`*`options`*`]`

Description

The tool **fwdsimflt** (fwdsim filter) can be used to filter the FWDSIM (XML) documents generated by the commands **fwdsim**(1) and **fitfluxes**(1) to a human-readable CSV format. In general, all information filtered by **fwdsimflt** is sent to stdout.

Common Options

`-h, --help` Show a brief help for all command line options.

`-i, --input <FILE> [default:stdin]` The name of the input FWDSIM (XML) file. If omitted, the FWDSIM document is expected on standard input.

Filter Options

`-m, --measurements` Filter out simulated measurement values. The first column contains the index of the measurement value. The second column contains the simulated measurement value itself. The rows with the special index 0 contain a heading:

- column two contains the ID of the measurement group.
- column three contains the norm for the measurement group.
- the fourth column contains the short notation of the measurement specification.

`-s, --scompact` Specifying this option generates a list of flux values. The first column contains the flux name, the second and third column contain the value of the net and exchange flux.

`-S, --sfull` Output the full information about the stoichiometry. In particular, this includes:

1. the flux name.
2. the type of the net flux ('f'=free, 'd'=dependent, 'c'=constraint, 'q'=quasi-constraint).
3. the value of the net flux.
4. the symbolic equation of the net flux.
5. the type of the exchange flux.
6. the value of the exchange flux.
7. the symbolic equation of the net flux.

`-t, --stddevs` Output the standard deviations (if this information is contained in the input document).

`-u, --unknowns` Output all computed unknowns in either Cumomer or EMU coordinates. The output depends on the simulation method and type of network reduction specified in the original FluxML document.

`-O, --optimflags` Output used optimization flags and sensitivities.

Filter-Filter Options

The output generated for the above options can be further filtered by a number of filter-filter options:

`-f, --filter <REGEX>` Only output the data sets which flux name, pool name, or measurement group name is matched by the specified regular expression. See **regex**(7) for the syntax of regular expressions.

`-F, --fwdbwd` Transform the flux values (`-s, -S`) into forward / backward coordinates.

`-X, --netxch01` Transform the flux values (`-s, -S`) into net / exchange-[0,1] coordinates. This flux coordinate system is used by the old 13CFLUX software.

`-U, --isotopomers` Transform the unknowns from Cumomer or EMU coordinates (`-u`) into isotopomer fractions. Note that this is not always possible if a reduced network is simulated, i.e. not all Cumomers or EMUs were computed. If you need a full listing of all isotopomer fractions modify the underlying FluxML document in order to perform a 'full' simulation.

Examples

Read from file network.fwd and output all computed measurement values (most simple case):

```
fwdsimflt -i network.fwd -m
```

Read from stdin, filter output for measurement groups A, B, C, and G:

```
fwdsim -i network.fml | fwdsimflt -m -f "[A-C]" -f G
```

Read from file, output simulated unknowns of pools A and F:

```
fwdsimflt -u -f A -f F -i network.fwd
```

Read from file, output full stoichiometry:

```
fwdsimflt -S -i network.fwd
```

Read from stdin, output flux names, net- and exchange-[0,1] values:

```
fwdsimflt -s -X < network.fwd
```

See Also

fwdsim(1), fitfluxes(1)

Author

Michael Weitzel
This manpage was written by Michael Weitzel <info@13cflux.net>.

2.5.4 simreport

simreport — simulation report generator.

Synopsis

`simreport` [*options*]

Description

Given a FluxML file and a corresponding simulation results in form of a FWDSIM (XML) file **simreport** generates a CSV file comparing the given and simulated measurement values. Norm values, measuring the discrepancy between measurement and simulation are provided for all measurement values.

Options

`-h, --help` Show a brief help for all command line options.

`-i, --fwdsim <FILE> [default:stdin]` The name of the FWDSIM (XML) input document. Default is to read from standard input.

`-f, --fluxml <FILE>` The name of the FluML input file (optional). In case the name of the FluxML file is omitted simreport generates just a listing of the simulated measurement values.

`-o, --out <FILE> [default:stdout]` Name of the generated CSV file. If this option is not specified the generated CSV file is written to standard output.

`-c, --configure <CFG> [default:'default']` Because FluxML documents may contain several `<configuration/>` elements this option allows to specify the configuration that should be used for the simulation. If this option is omitted it is assumed that the FluxML document contains a configuration with the name "default".

Examples

Generate a CSV simulation report file from a FluxML document and a FWDSIM (XML) document (most simple case):

```
simreport -i network.fwd -f network.fml -o report.csv
```

Combine **simreport** with **fwdsim**, i.e. read the FWDSIM (XML) document from standard input:

```
fwdsim -i network.fml | simreport -f network.fml -o report.csv
```

Produce a nice representation of the simulated measurement values. The generated CSV file will not contain the real measurement values, norms, or group scaling factors:

```
fwdsim -i network.fml | simreport -o report.csv
```

See Also

fwdsim(1), fitfluxes(1), fwdsimflt(1)

Author

Michael Weitzel
This manpage was written by Michael Weitzel <info@13cflux.net>.

2.5.5 collectfitdata

collectfitdata — collect together simulation results into a HDF5 file.

Synopsis

`collectfitdata` [*options*] [-o <FILE>]

Description

Collectfitdata is a tool for collecting together the results (i.e. net fluxes, exchange fluxes, and measurement values) from multiple FWDSIM (XML) files (see **fwdsim**(5)). The collected data is saved to a HDF5 file.

Common Options

`-h, --help` Show a brief help for all command line options.

`-f, --filelist <PATH>` Path to a directory containing FWDSIM files. It is assumed, that these FWDSIM files are created by fitfluxes or fwdsim from the same FluxML file! If omitted, collectfitdata accepts a single FWDSIM file from stdin.

`-o, --out <FILE>` The name of the HDF5 output file. This option must be supplied (HDF5 files are never written to standard output).

`-a, --append` When specified, this option causes the HDF5 file to be appended, if the file existed before. Otherwise a new HDF5 is created.

`-l, --log DEST` Specify the destination for the internal logging. In the most simple case `DEST` is a file name of a log file. In case the file exists new log messages are appended. Apart from log files it is possible to publish log messages to file descriptors, UNIX domain sockets, UDP and SCTP ports, and a small graphical user interface.

A file descriptor is specified by `fd:[num]`, where `[num]` is the number of the file descriptor.

A unix domain socket in the local file system is specified by `unix:[name]`, where `[name]` is the name of the socket file

A UDP or (connectionless) SCTP port is specified by `[proto]:[host]:[port]`, where `[proto]` is either "udp" or "sctp" and `[host]` is the name of the destination host and `[port]` is a UDP or SCTP port number on the destination host. Please note that the length of log messages is bounded by the minimum safe UDP packet size - log messages containing more than 548 characters will be truncated.

Finally, log messages can also be sent to a small GUI by specifying the destination `@gui@`. The GUI requires a working Perl/Tk installation and a running X server.

In order to capture all log messages concerning the command line processing this option should be specified in front of all other options.

`-v, --verbose 0..10 [default:5]` Specify the verbosity 0, 1, ..., 10 of generated / emitted log messages. The meaning of the different log levels is as follows:

- `0 (QUIET)` do not emit log messages at all.
- `1 (ERROR)` only emit severe error messages.
- `2 (WARNING)` only report severe errors and warnings.
- `3 (NOTICE)` report all errors and warnings including important informal messages.
- `4 (INFO)` report all errors, warnings and all informal messages.
- `5 (THROW)` in case of an exception, try to give a diagnosis of the error; sometimes even gives a backtrace of the current function stack.
- `6 (DEBUG0)` emit the more important debugging messages.
- `7 (DEBUG1)` emit the less important debugging messages
- `8 (DEBUG2)` emit the superfluous debugging messages
- `9 (DEBUG3)` emit annoying debugging messages.
- `10 (DEBUG4)` don't dare to use it!

Content Control Options

`-m, --measurements` With this option, measurement entries are also collected into the HDF5 file. Specifically, measurement group values from the FWDSIM file under the node <measurements> are written into the HDF5 file into the field /measurements.

`-F, --free-only` Only collect free variables into the HDF5 file. Constrained and dependent fluxes are omitted.

`-N, --omit-net-fluxes` With this option defined, net fluxes are not collected into the HDF5 file.

`-X, --omit-xch-fluxes` With this option defined, xch fluxes are not collected into the HDF5 file.

Examples

Collecting together the results from a directory containing a number of FWDSIM (XML) files can be performed like this:

```
collectfitdata -f fwdsim_dir -o results.hdf5
```

Single FWDSIM (XML) files can be piped into collectfitdata:

```
cat data.fwd | collectfitdata -o results.hdf5
```

Collecting only free net fluxes is performed as follows:

```
collectfitdata -f fwdsim_dir -o results.hdf5 -X -F
```

See Also

fwdsim(1), fitfluxes(1), multifit(1)

Author

Tolga Dalman

This manpage was written by Tolga Dalman and Michael Weitzel <info@13cflux.net>.

2.5.6 fwdsim2csv

fwdsim2csv — transfer of flux settings from multiple FWDSIM (XML) documents into one CSV file

Synopsis

`fwdsim2csv` [*options*] [*FWDSIM file names*]

Description

The script **fwdsim2csv** is used to transfer the flux settings from one or multiple FWDSIM (XML) document, possibly generated by the commands **fitfluxes**(1) and **multifit**(1) into human-readable CSV data. The CSV data is always written to standard output.

Common Options

`-h, --help` Show a brief help for all command line options.

Output Options

`-s, --separator <str> [default:","]` The CSV field separator to use.

`-a, --all` Include all fluxes in the output, not just the free fluxes.

`-n, --net-only` Output only net fluxes and omit the exchange fluxes.

`-x, --xch-only` Output only exchange fluxes and omit the net fluxes.

`-H, --no-headers` Omit all column headers.

`-R, --no-residual` Omit the column containing the obtained residuals.

`-p, --opt-flags` Include additional columns containing the optimization flags.

`-f, --filenames` Include a column containing the file names of the FWDSIM files.

Examples

Read multiple FWDSIM (XML) files and output the values of all (free and dependent) net fluxes, and include the file names:

```
fwdsim2csv -anf *.fwd > fitdata.csv
```

Read one FWDSIM (XML) file from standard input and output the values of the free net and exchange fluxes, excluding the residuals:

```
fwdsim2csv -R < fit.fwd > fitdata.csv
```

See Also

fwdsim(1), fitfluxes(1), multifit(1)

Author

Michael Weitzel

This manpage was written by Michael Weitzel <info@13cflux.net>.

2.5.7 hdf5tocsv

hdf5tocsv — convert an XML file to a CSV formatted output.

Synopsis

`hdf5tocsv` [*options*]

Description

Hdf5tocsv converts double matrices or vectors in HDF5 files from 13cflux(5) programs such as **ssampler(1)**, **collectfitdata(1)**, **multifwdsim(1)**, **multifitfluxes(1)**, or **multiperturb(1)** into a human-readable comma-separated text format (CSV). For instance, this is useful in situations where third-party tools have no option for processing HDF5 files.

Common Options

`-h, --help` Show a brief help for all command line options.

`-i, --in <FILE>` Path to HDF5 input file containing matrix data (--data) and column texts(--columns).

`-o, --out <FILE> [default:stdout]` Output CSV file. Unless specified with --no-header, the first line contains the header column texts.

`-D, --data <H5FIELD> [default:/flux/data]` HDF5 path to the data matrix in the HDF5 file.

`-C, --columns <H5FIELD> [default:/flux/names]` HDF5 path to the header column texts in the HDF5 file.

Special Options

`-d, --csv-delimiter <CHAR> [default:',' ]` Select a different separator character in the output CSV file.

`-H, --no-header` If set, the output of the column texts is omitted. This is useful for concatenating multiple HDF5 files into a single CSV output.

Examples

[source,shell]:

```
hdf5tocsv -i samples.hdf5 -o samples.csv
```

See Also

ssampler(1), collectfitdata(1), multifwdsim(1), multifwdsim(1), multiperturb(1), h5dump(1)

Author

This manpage was written by Tolga Dalman <info@13cflux.net> and Michael Weitzel <info@13cflux.net>.

2.6 Validation and Benchmark

fmllint The program **fmllint** is a simple FluxML validation tool. In addition to the syntactic check provided by the underlying Xerces-C XML Schema validation **fmllint** performs a thorough semantic check of a FluxML document.

fmlsign The program **fmlsign** is used for signing FluxML files and downloading updated license files. A signed FluxML file is bound to a specific 13CFLUX2 license. Please note that, in order to be signed, a FluxML file needs to be syntactically and semantically correct **fmllint**(1) is run to ensure this.

benchmark The program **benchmark** can be used to generate a performance benchmark for different linear solvers. The tool gives useful information about the simulated cumomer / EMU network and performs an analysis of the computational graphs.

Important command line switches:

-r the number of benchmark runs

-s the names of the solvers to use for the benchmark

-L creates a list of the available solvers

2.6.1 fmllint

fmllint — a lint tool (syntax checker) for FluxML.

Synopsis

`fmllint` [*options*]

Description

The program **fmllint** is a simple FluxML validation tool. In addition to the syntactic check provided by the underlying Xerces-C XML Schema validation **fmllint** performs a thorough semantic check of a FluxML document. A name of a FluxML file or an URL may be passed on the command line. In case the command line is empty, the FluxML document is expected on standard input.

Common Options

-h, --help Show a brief help for all command line options.

-i, --in <FILE> [default:stdin] The name of the FluxML (XML) input file. If omitted, the FluxML document is expected on standard input.

-o, --out <FILE> [default:no output] The name of the output FluxML file. If this option is omitted no output is generated. Use "-" for generating output on standard output

-L, --list Specifying this option results in a list of allowed configuration names for the specified FluxML document. The program exits immediately after emitting the list.

-l, --log DEST Specify the destination for the internal logging. In the most simple case `DEST` is a file name of a log file. In case the file exists new log messages are appended. Apart from log files it is possible to publish log messages to file descriptors, UNIX domain sockets, UDP and SCTP ports, and a small graphical user interface.

A file descriptor is specified by `fd:[num]`, where `[num]` is the number of the file descriptor.

A unix domain socket in the local file system is specified by `unix:[name]`, where `[name]` is the name of the socket file

A UDP or (connectionless) SCTP port is specified by `[proto]:[host]:[port]`, where `[proto]` is either "udp" or "sctp" and `[host]` is the name of the destination host and `[port]` is a UDP or SCTP port number on the destination host. Please note that the length of log messages is bounded by the minimum safe UDP packet size - log messages containing more than 548 characters will be truncated.

Finally, log messages can also be sent to a small GUI by specifying the destination `@gui@`. The GUI requires a working Perl/Tk installation and a running X server.

In order to capture all log messages concerning the command line processing this option should be specified in front of all other options.

-v, --verbose 0..10 [default:5] Specify the verbosity 0, 1, ..., 10 of generated / emitted log messages. The meaning of the different log levels is as follows:

- `0 (QUIET)` do not emit log messages at all.
- `1 (ERROR)` only emit severe error messages.
- `2 (WARNING)` only report severe errors and warnings.
- `3 (NOTICE)` report all errors and warnings including important informal messages.
- `4 (INFO)` report all errors, warnings and all informal messages.

- `5 (THROW)` in case of an exception, try to give a diagnosis of the error; sometimes even gives a backtrace of the current function stack.
- `6 (DEBUG0)` emit the more important debugging messages.
- `7 (DEBUG1)` emit the less important debugging messages
- `8 (DEBUG2)` emit the superfluous debugging messages
- `9 (DEBUG3)` emit annoying debugging messages.
- `10 (DEBUG4)` don't dare to use it!

Special Options

`-C, --challenge` Specifying this option results in the computation of a network-specific authentication challenge. This option and its result are reserved for internal use.

Example

Check a local FluxML file for semantic and syntactic correctness:
```
fmllint -i network.fml
```
Check a remote file on a HTTP server for correctness:
```
fmllint -i http://www.13cflux.net/network.fml
```

See Also

xmllint(1)

Author

Michael Weitzel
This manpage was written by Michael Weitzel <info@13cflux.net>.

2.6.2 fmlsign

fmlsign — a tool for signing FluxML files.

Synopsis

`fmlsign` [*`options`*]

Description

The program **fmlsign** is used for signing FluxML files and downloading updated license files. A signed FluxML file is bound to a specific 13CFLUX2 license. Please note that, in order to be signed, a FluxML file needs to be syntactically and semantically correct **fmllint**(1) is run internally to ensure this.

Common Options

`-h, --help` Show a brief help for all command line options.

`-i, --in <FILE> [default:stdin]` The name of the FluxML (XML) input file. If omitted, the FluxML document is expected on standard input.

`-o, --out <FILE> [default:name of input file]` The name of the output file for options `-i` and `-l`. If this option is omitted the signed FluxML is written to the file name specified for `-i`. If option `-i` is omitted or the name of the output file is set to '-' the result is written to stdout.

`-l, --license [<ID>]` Specifying this option results in downloading the most recent license file for your version of 13CFLUX2. After successful download the license file is written to the filename specified by `-o` (or stdout, if `-o` is omitted). The license ID is optional - in case it is omitted your current license ID is used. If you do not have a license file yet a valid license ID must be specified.

Example

The following command line causes a FluxML file to be checked and signed:

```
fmlsign -i network.fml
```

A recent license file may be downloaded like this:

```
fmlsign -l -o ~/.x3cflux/license.dat
```

See Also

fmllint(1)

Author

Michael Weitzel

This manpage was written by Michael Weitzel <info@13cflux.net>.

2.6.3 benchmark

benchmark — performance benchmark for the 13CFLUX 2 software

Synopsis

`benchmark` [*`options`*]

Description

The program **benchmark** can be used to generate a performance benchmark for different linear solvers. The tool gives useful information about the simulated cumomer / EMU network and performs an analysis of the computational graphs. For details the reader is referred to Weitzel et al. BMC Bioinformatics 8:315, 2007.

Common Options

`-h, --help` Show a brief help for all command line options.

`-i, --in <FILE> [default:stdin]` The name of the FluxML (XML) input file. If omitted, the FluxML document is expected on standard input.

`-s, --solvers X,Y,...[default:all built-in solvers]` This option takes a comma-separated list of names of the linear solvers to be used for the performance benchmark.

`-L, --list-solvers` This option causes a list of valid solver names to be generated.

`-S, --statistics` This option causes additional statistics on the (reduced) network and its computational graphs to be generated.

`-c, --configure <CFG> [default:'default']` Because FluxML documents may contain several `<configuration/>` elements this option allows to specify the configuration that should be used for the simulation. If this option is omitted it is assumed that the FluxML document contains a configuration with the name "default".

`-l, --log DEST` Specify the destination for the internal logging. In the most simple case `DEST` is a file name of a log file. In case the file exists new log messages are appended. Apart from log files it is possible to publish log messages to file descriptors, UNIX domain sockets, UDP and SCTP ports, and a small graphical user interface.

A file descriptor is specified by `fd:[num]`, where `[num]` is the number of the file descriptor.

A unix domain socket in the local file system is specified by `unix:[name]`, where `[name]` is the name of the socket file

A UDP or (connectionless) SCTP port is specified by `[proto]:[host]:[port]`, where `[proto]` is either "udp" or "sctp" and `[host]` is the name of the destination host and `[port]` is a UDP or

SCTP port number on the destination host. Please note that the length of log messages is bounded by the minimum safe UDP packet size - log messages containing more than 548 characters will be truncated.

Finally, log messages can also be sent to a small GUI by specifying the destination `@gui@`. The GUI requires a working Perl/Tk installation and a running X server.

In order to capture all log messages concerning the command line processing this option should be specified in front of all other options.

`-v, --verbose 0..10 [default:5]` Specify the verbosity 0, 1, ..., 10 of generated / emitted log messages. The meaning of the different log levels is as follows:

- `0 (QUIET)` do not emit log messages at all.
- `1 (ERROR)` only emit severe error messages.
- `2 (WARNING)` only report severe errors and warnings.
- `3 (NOTICE)` report all errors and warnings including important informal messages.
- `4 (INFO)` report all errors, warnings and all informal messages.
- `5 (THROW)` in case of an exception, try to give a diagnosis of the error; sometimes even gives a backtrace of the current function stack.
- `6 (DEBUG0)` emit the more important debugging messages.
- `7 (DEBUG1)` emit the less important debugging messages
- `8 (DEBUG2)` emit the superfluous debugging messages
- `9 (DEBUG3)` emit annoying debugging messages.
- `10 (DEBUG4)` don't dare to use it!

Examples

Perform 123 benchmark runs using only solver "CSparseLU":

```
benchmark -r 123 -s CSparseLU -i network.fml
```

See Also

fwdsim(1), fitfluxes(1)

Author

Michael Weitzel
This manpage was written by Michael Weitzel <info@13cflux.net>.

Chapter 3

Document Formats

3.1 fluxml

fluxml — the 13CFLUX 2 XML document format for network definition

Description

The mathematical modeling of an Isotope Labeling Experiment is parametrized by the atom transitions of the isotope labeling network, the stoichiometric constraints, the specification of measurement configuration and the measurement data. In the old 13CFLUX this data is provided by a file in the FTBL file format. FTBL consists of tabulator-separated fields organized in blocks which are separated by headings. The files are intended to be edited in spreadsheet applications.

The new 13CFLUX 2 replaces the simplistic FTBL file format with the modern XML-based **FluxML** document format. A primary concern of **FluxML** was to adopt the content model of the old FTBL, so that old network specifications can be easily converted into the new format (see **ftbl2fml**(1)). Another design goal was that the format should be readable and editable by humans -- at least to some degree. This demand had a few consequences on the design of FluxML:

- The specification of the reaction's atom transitions basically follows the old FTBL syntax, i.e. the reactions' permutation property is encoded by a pair of character strings consisting of letters and digits. A more flexible, alternate format is supported, as well.
- Measurement specifications use expressive, human-readable short notations instead of length, hierarchical, XML-style specifications.
- Formulas used in the stoichiometric constraints and measurement specifications may be authored either in a human-editable textual notation, or in a machine-readable Content-MathML prefix notation, supported also by SBML and CellML

Because there is certain semantical intersection between FluxML and SBML the FluxML specification of a metabolic reaction network can be easily exported into a stoichiometric SBML model (see **fml2sbml**(1) and **sbml2fml**(1)).

The FluxML Elements

Being a XML format, the layout of FluxML is hierarchic. FluxML documents, as well as their elements, belong to the XML namespace denoted by the URI "http://www.13cflux.net/fluxml". FluxML's root element usually includes corresponding namespace attribute ("xmlns"). The preamble and root element of a FluxML document should look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<fluxml xmlns="http://www.13cflux.net/fluxml">
...
</fluxml>
```

The following briefly presents the elements of a FluxML document.

Element: /fluxml

A FluxML document consists of an optional info header, the specification of a reactionnetwork, optional stoichiometric constraints, and zero or more configurations (each including a substrate specification, possibly additional constraints, measurement specifications):

```
<fluxml xmlns="http://www.13cflux.net/fluxml">
   <info> ... </info> [0..1]
   <reactionnetwork> ... </reactionnetwork> [1]
   <constraints> ... </constraints> [0..1]
   <configuration> ... </configuration> [0..*]
</fluxml>
```

Element: /fluxml/info

All fields in a FluxML's info header are optional. The fields in the info header allow the specification of the network name, a document version, a timestamp (in "YYYY-MM-DD hh:mm:ss" format), and a comment.

```
<info>
  <name> xsd:string </name> [0..1]
  <version> xsd:string </version> [0..1]
  <date> \d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2} </date> [0..1]
  <comment> xsd:string </comment> [0..1]
</info>
```

Element: /fluxml/reactionnetwork

The specification of the reaction network includes a nonempty list of metabolites and a list of one or more reactions:

```
<reactionnetwork>
  <metabolitepools> ... </metabolitepools> [1]
  <reaction> ... </reaction> [1..*]
</reactionnetwork>
```

Element: /fluxml/reactionnetwork/metabolitepools

The list of metabolites consists of two or more "pool" elements:

```
<metabolitepools>
  <pool> ... </pool> [2..*]
</metabolitepools>
```

Element: /fluxml/reactionnetwork/metabolitepools/pool

Each pool element is required to have a unique ID and specifies the number of labling positions ("atoms"), e.g. its number of carbon atoms. If the "atoms" attribute is omitted the metabolite is assumed to have zero labeling positions. This allows to model purely stoichiometric networks in 13CFLUX 2.

```
<pool
 id="xsd:ID [1]"
 atoms="AtomType [0..1]">
```

```
  <annotation> ... </annotation> [0..*]
</pool>
```

Element: /fluxml/reactionnetwork/.../annotation

The specifications of metabolites and reactions may include additional annotations, which can be used to store additional information. The annotations are not evaluated by the FluxML parser and their content is purly informational:

```
<annotation name="xsd:string [1]"/>
```

Element: /fluxml/reactionnetwork/reaction

A reaction consists of an optional list of annotations, a non-empty list of "educts" (i.e. starting material) and a possibly empty list of products. In case a reaction does not specify any products it is considered to be an efflux from the network. The name of a reaction is given by a unique id. The attribute "bidirectional" allows to introduce simple directionality constraints: setting "bidirectional" to "false" restricts the reaction's exchange flux to zero and the net flux to a non-negative value.

For a "scrambling reaction" multiple variants of the same reaction may exist. In this case the "id" attribute has to specify multiple identifiers for the different variants of the reaction. See below for more information.

```
<reaction
 id="xsd:ID [1..*]"
 bidirectional="xsd:boolean [0..1]">
  <annotation> ... </annotation> [0..*]
  <reduct> ... </reduct> [1..*]
  <rproduct> ... </rproduct> [0..*]
</reaction>
```

Element: /fluxml/reactionnetwork/reaction/{reduct,rproduct}

The specification of an educt or a product references the unique ID of a pool. In case the referenced metabolite includes a non-zero number of labeling positions the "cfg" attribute has to specify a string associating letters (and possibly digits) with labeling positions of the educts and product side. See below for examples and help.

Support for scrambling reactions. A list of optional "variant" elements can be used to specify variants of the reaction differing in their atom transitions. Each "variant" element has to specify a "cfg" attribute. In case any "variant" elements are present the "reduct" or "rproduct" elements must not specify an own "cfg" attribute. The optional "ratio" attributes of the "variant" elements allow to specify the likelihoods of the individual reaction variants. In case no "ratio" attributes are specified all variants of the reaction will have the same likelihood. Internally, the likelihoods are implemented using automatically generated stoichiometric equality constraints.

```
<{reduct,rproduct}
 id="xsd:IDREF [1]"
 cfg="CfgType [0..1]">
  <variant cfg="CfgType [1]" ratio="xsd:double [1]"/>
  ...
</{reduct,rproduct}>
```

Element: /fluxml/[configuration/]constraints

The element "constraints" is optional and lists a number of linear stoichiometric constraints. There are separate sub-elements for constraints on net and exchange fluxes.

```
<constraints>
  <net> ... </net> [0..1]
  <xch> ... </xch> [0..1]
</constraints>
```

Element: /fluxml/[configuration/]constraints/{net,xch}

Elements "net" and "xch" list constraints consisting of linear equalities and inequalities on the values of net and exchange fluxes. Constraints are specified either in a textual notation, or in Content-MathML (see http://www.w3.org/Math/ for more information).

```
<{net,xch}>
  Start Choice [1]
  <textual> ... </textual> [1]
  <mml:math> ... </mml:math> [1]
  End Choice
</{net,xch}>
```

Element: /fluxml/[configuration/]constraints/{net,xch}/textual

Element textual may include a list of textual constraint equalities and inequalities on the flux values. Multiple (in)equalities have to be separated by semicolons. If a constraint follows a textual description and a colon, the constraint is associated with that description. Example:

`my constraint:v1 =v2 -1.2*v3;another constraint:v2=5;...`

Beware that, for inequality constraints in a XML document, you have to encode "<=" and ">=" as &lt;= and &gt;= .

```
<textual> xsd:string </textual>
```

Element: /fluxml/configuration

A FluxML configuration represents a parametrization of a static network structure with isotopic labeling for the substrates, additional constraints and measurement values. Sub-element simulation contains the settings for the simulator and lists the flux values. Each configuration has to specify a unique name, may include a textual comment.

```
<configuration
 name="xsd:ID [1]">
  <comment> ... </comment> [0..1]
  <input> ... </input> [1..*]
  <constraints> ... </constraints> [0..1]
  <measurement> ... </measurement> [0..1]
  <simulation> ... </simulation> [0..1]
</configuration>
```

Element: /fluxml/configuration/comment

A configuration's comment is optional and can be used to provide useful information to a human reader.

```
<comment> xsd:string </comment>
```

Element: /fluxml/configuration/input

Each configuration is required to provide a specification of the isotopic labeling sources. By using the attribute "pool", the corresponding "input" element references the unique name of a metabolite pool (which, of course, is required to be a substrate pool). Attribute "type" gives the type of the specification, which may be either "isotopomer" (the default) or "cumomer" (unusual).

If the "input" element does not contain any "label" elements the isotopomer distribution is initialized according to the natural abundance of 12C (98.945%) and 13C (1.055%) isotopes.

```
<input
 id="xsd:ID [0..1]"
 pool="xsd:IDREF [1]"
 type="xsd:string matching (cumomer|isotopomer) [0..1]">
  <label> ... </label> [0..*]
</input>
```

Element: /fluxml/configuration/input/label

Element "label" is used to specify the fractional abundance of a certain isotopomer (or cumomer) fraction in a labeling source. Attribute "cfg" defines the labeling composition of a certain isotopomer based on a string of "1"'s (labeled positions) and "0"s (unlabeled positions; "X" for cumomer fractions).

The optional attribute "purity" is used to specify the positional isotopic purity of labeled carbon atom positions. All unlabeled carbon atom positions are assumed to be naturally labeled (i.e. 98.945% 12C, 1.055% 13C). The value of "purity" is assumed to be a fractional value between 0 (0% purity) and 1 (100% purity). Attribute "purity" has no default value - if it is omitted no automatic isotope correction is performed and the substrate specfcation is used as provided.

```
<label
 cfg="xsd:string matching pattern [01xX]+ [1]"
 purity="xsd:double [0..1]">
xsd:double
</label>
```

Element: /fluxml/configuration/measurement

The element "measurement" contains an optional info header (element "mlabel") and specifies the measurement model and the associated measurement data.

```
<measurement>
  <mlabel> ... </mlabel> [0..1]
  <model> ... </model> [1]
  <data> ... </data> [1]
</measurement>
```

Element: /fluxml/configuration/measurement/mlabel

A measurement specification includes an optional info header which can be used to provide additional information about the labeling experiment.

```
<mlabel>
  <date> \d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2} </date> [0..1]
  <version> xsd:string </version> [0..1]
  <comment> xsd:string </comment> [0..1]
  <fluxunit> xsd:string </fluxunit> [0..1]
</mlabel>
```

Element: /fluxml/configuration/{measurement,simulation}/model

The measurement model of 13CFLUX 2 may be used to describe labeling enrichment measurements and flux measurements.

```
<model>
  <labelingmeasurement> ... </labelingmeasurement> [0..1]
  <fluxmeasurement> ... </fluxmeasurement> [0..1]
</model>
```

Element: /.../{measurement,simulation}/model/labelingmeasurement

This element consists of a list of measurement group specifications. Each mesurement group is a group of measurement values sharing a common, usually automatically determined scaling factor.

```
<labelingmeasurement>
  <group> ... </group> [0..*]
</labelingmeasurement>
```

Element: /fluxml/.../model/labelingmeasurement/group

A group of labeling measurements has to specify a unique ID. Attribute "scale" controls whether the measurement values are automatically scaled (value "auto") or not (value "one"). The measurement specification may can be formulated in Content-MathML or a textual notation and consists either of a single identifier specifying multiple measurement values (in case of a MS, MS/MS, 1H-NMR, or 13C-NMR measurement) or of a vector / semicolon-separated list of formulas. In the latter case each of these formulas has to refer to a single measurement value. For details, see MEASUREMENT SHORT NOTATIONS below.

```
<group
 id="xsd:ID [1]"
 scale="xsd:string matching (auto|one) [0..1]">
  <errormodel> ... </errormodel> [0..1]
  Start Choice [1]
  <textual> ... </textual> [1]
  <mml:math> ... </mml:math> [1]
  End Choice
</group>
```

Element: /fluxml/.../model/labelingmeasurement/group/errormodel

When doing experimental design (cf. **edscanner**(1), **edopt**(1)), an error model may be defined for extrapolating the standard deviations of the simulated measurement values. Such an error model is a formula composed of values, the predefined variables "meas_real" (provided measurement value), "std_real" (provided standard deviation), and "meas_sim" (simulated measurement value), standard mathematical operators and the built-in functions "abs(x)", "exp(x)", "max(x,y)", "min(x,y)", "sqrt(x)", "log(x)", "log2(x)", "log10(x)", and "sqr(x)".

In case the error model consists of a single formula, that formula is used for all measurement values in the measurement group. In case more than one formula is specified (separated by ";") it is assumed that the number of error model formulas equals the number of measurment values. That is, it is possible to specify different error models for individual measurement values.

In case the error model is completely omitted no extrapolation of the standard error is performed and the original provided standard error is used when computing the covariance matrix required for experimental design. The error model may be set to "1" for disabling the scaling of the covariance matrix completely.

```
<errormodel>
  Start Choice [1]
  <textual> ... </textual> [1]
  <mml:math> ... </mml:math> [1]
  End Choice
</errormodel>
```

Element: /.../{measurement,simulation}/model/fluxmeasurement

Flux measurements can be specified by using arbitrary formulas containing the unique name given in the "id" attribute of the "reaction" element. Because net and exchange fluxes are linearly independent and the corresponding flux measurement specifications may not be mixed, there are two types of subelements available.

```
<fluxmeasurement>
  Start Choice [0..*]
  <netflux> ... </netflux> [1]
  <xchflux> ... </xchflux> [1]
  End Choice
</fluxmeasurement>
```

Element: /fluxml/configuration/.../fluxmeasurement/{netflux,xchflux}

Elements "netflux" and "xchflux" may specify arbitrary equations in order to model the measurement of net and exchange fluxes. The equations are formulated in either Content-MathML or in a textual notation. All identifies in the equations must refer to the unique "id" attribute of the "reaction" elements.

```
<{netflux,xchflux}
 id="xsd:ID [1]">
Start Choice [1]
<textual> ... </textual> [1]
<mml:math> ... </mml:math> [1]
End Choice
</{netflux,xchflux}>
```

Element: /fluxml/configuration/measurement/data

Element "data" contains an optional info-header and a list of "datum" elements providing the measurement data.

```
<data>
  <dlabel> ... </dlabel> [0..1]
  <datum> ... </datum> [1..*]
</data>
```

Element: /fluxml/configuration/measurement/data/dlabel

The measurement data may be equipped with an info-header providing additional information about the labeling experiment. All sub-elements are optional. The timestamps specified in elements "start" and "finish" have to follow the pattern "YYYY-MM-DD hh:mm:ss".

```
<dlabel>
<start> \d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2} </start> [0..1]
<finish> \d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2} </finish> [0..1]
<people> xsd:string </people> [0..1]
<strain> xsd:string </strain> [0..1]
<comment> xsd:string </comment> [0..1]
</dlabel>
```

Element: /fluxml/configuration/measurement/data/datum

A single measurement datum has to reference the unique ID of a measurement group specification. Along with the measurement value a standard deviation has to be provided. The other attributes are used to associate a measurement datum with one of the measurement group's measurements:

- attribute "row" refers to a generic measurement group; in particular to a row containing a formula describing a single measurement value. "row" is expected to be an integer >0.
- attribute "weight" refers to a MS or MS/MS measurement. For MS measurements the value of "weight" is a simple integer. For MS/MS measurements "weight" takes two comma-separated integer values (the weight of the fragment and the weight of the fragment's fragment).
- attributes "pos" and "type" give the labeling position and multiplet-type for NMR measurements. Attribute "pos" is used by both 1H-NMR and 13C-NMR measurements. Attribute "type" identifies the multiplet type and is reserved for 13C-NMR measurements. Supported values for "type" are: "S" (signlet), "DL" (left doublet), "DR" (right doublet), "DD" (double doublet), and "T" (triplet).

```
<datum
 id="xsd:IDREF [1]"
 stddev="xsd:double [1]"
 row="xsd:integer (1 <= value <= 256) [0..1]"
 weight="xsd:string maching \d+(\s*,\s*\d+)? [0..1]"
 pos="AtomType [0..1]"
 type="xsd:string matching (S|DL|DR|DD|T) [0..1]"/>
```

Element: /fluxml/configuration/simulation

Element "simulation" collects simulation variables and settings. The attribute "type" controls the behavior of network reduction:

- setting "type" to the value "auto" (the default) selects automatic network reduction. This results in the simulation of a Cumomer or EMU network which is minimal for the specified measurement model. This setting is relyable and should be used to obtain the fastest possible simulation.
- setting "type" to "full" deactivates automatic network reduction. As a consequence a simulation of the full / unreduced Cumomer or EMU network is enforced - although this might not be necessary. Using this setting, simulations are usually much slower or even impossible for large networks comprising metabolites with many labeling positions. Use with care.
- setting "type" to "explicit" results in evaluation of the sub-element "model", explicitly listing the measurement specifications that are to be simulated. This option does neither require nor allow the specification of element "measurement" (including measurement values). In this mode, all measurement values are assumed to be zero and all standard deviations are assumed to have the value one.

The choice of the simulation method, Cumomer or EMU, may be influenced by attribute "method". The default value is "auto", which selects the optimal simulation method for the specified measurement model. Other posible choices are "cumomer" (for all NMR and MS/MS measurements) and "emu" (for MS measurements). A non-automatic selection of the simulation method makes sense combination with "type" set to "full".

Sub-element "variables" lists the choice of fluxes (including initial values) which are the free parameters of the stoichiometric system (i.e. "free fluxes").

```
<simulation
 type="xsd:string matching (auto|full|explicit) [0..1]"
 method="xsd:string matching (auto|cumomer|emu) [0..1]">
  <model> ... </model> [0..1]
  <variables> ... </variables> [0..1]
</simulation>
```

Element: /fluxml/configuration/simulation/variables

```
<variables>
  Start Choice [0..*]
  <fluxvalue> ... </fluxvalue> [1]
  <poolsize> ... </poolsize> [1]
  End Choice
</variables>
```

Element: /fluxml/configuration/simulation/variables/fluxvalue

The fluxes listed in this element are the free parameters of the stoichiometric system. The attribute "flux" references the unique "id" attribute of a single "reaction" element. Allowed values for attribute "type" are "net" (net flux value) or "xch" (exchange flux value). The value of this element is the initial flux value.

A third attribute, "edweight", is used only for experimental design (cf. **edscanner**(1), **edopt**(1)). The attribute takes a floating-point value between 0 and 1 (inclusive) and is used to adjust the influence of a free flux on the elements of the covariance matrix. Setting "edweight" to 0 elimates the influence of a free flux. Any value between 0 and 1 results in a scaling of the corresponding row and column of the covariance matrix. The default value of the attribute is 1.

```
<fluxvalue
 flux="xsd:IDREF [1]"
 type="xsd:string matching (net|xch) [1]"
 edweight="xsd:double from range [0,1] [0..1]"/>
```

Specification of Atom Transitions

The (original and new) 13CFLUX software models a molecule as a simple tuple of labeling positions. Although this representation is very limited it is possible to encode some of the structural information by enumerating the labeling positions in a "topological order" where the index of a pair of labeling positions adjacent in the molecule differs only by one. In molecules with branched atom backbones, however, the indexes of labeling positions describing a molecule fragment or a local neighborhood may not be consecutive. When writing down the atom transitions of a metabolic reaction network for 13CFLUX, it is completely up to the modeler to chose a unique number for all labeling positions.

In the old and new 13CFLUX atom transitions are specified by using simple letter strings. Example:
`Fru6P#abcdef ==> DHAP#cba + GA3P#def`

Each of the letters "a" to "f" represent a labeling position which is transported from the input side to the product side of a reaction. For instance, in the above reaction carbon atom #1 of metabolite Fru6P is transported to carbon atom position #3 of metabolite DHAP. In total, the carbon atom chain of Fru6P is split into two pieces consisting of three carbon atoms, where the one of DHAP consists of the first three atoms of Fru6P, but in reversed order. In FluxML notation the above reaction looks as follows:

```
<reaction id="some_id" bidirectional="false">
  <reduct id="FruBP" cfg="abcdef"/>
  <rproduct id="DHAP" cfg="cba"/>
  <rproduct id="GA3P" cfg="def"/>
</reaction>
```

By using lower case letters, upper case letters, and digits metabolites with up to 62 labeling positions can be used. However, there exists an alternate notation with improved readability which allows an unlimited number of labeling positions:

```
<reaction id="some_id" bidirectional="false">
  <reduct id="FruBP" cfg="C#1@1 C#2@1 C#3@1 C#4@1 C#5@1 C#6@1"/>
  <rproduct id="DHAP" cfg="C#3@1 C#2@1 C#1@1"/>
  <rproduct id="GA3P" cfg="C#4@1 C#5@1 C#6@1"/>
</reaction>
```

Here, each labeling position has to be specified using the scheme
`[chem.element]#[position index]@[educt metabolite index]`

Measurement Short Notations

In 13CFLUX 2 the measurement group specifications are encoded in short notations. There are different short notations for MS, MS/MS, 1H-NMR, and 13C-NMR. Usually, a single short notation encodes a vector of measurement values. In turn, a (smaller) short notation can be given for every element of such a vector. The differnt short notations share a set of common syntax elements:

digit "0"|"1"|...|"9"; letter "A"|...|"Z"|"a"|...|"z"|"_"; number ("1"|...|"9"), { }; numberlist , { ",", }; poolname , [(|), { ",", (|) }]; rangeitem , ["-",]; range , { ",", };

MS Measurements

A simple mass spectrometry (MS) measurement describes a single metabolite or fragment in different "mass-increments". A mass-increment of n indicates that exactly "n" of the labeling positions of the molecule or fragment are occupied by an isotopic labeling. The measurement value is simulated by summing up all isotopomer fractions which fit this pattern. In the real MS measurement, a specific mass-increment of a metabolite results in a peak in the spectrogram and the measurement value is obtained by an integration of the peak.

msrangespec "[", , "]"; MSspec [ ], "#M", ;

Semantics: "msrangespec" identifies the labeling positions contained in a molecule fragment. "numberlist" lists the different mass-increments (compared to the weight of the unlabeled fragment). If "msrangespec" is omitted the full molecule is described.

MS/MS Measurements

A MS/MS (Tandem-MS) measurement describes a MS measurement followed by a fragmentation of the molecule and a second MS measurement on the resulting fragments. Since the first MS measurement may also refer to a molecule fragment, a MS/MS measurement value describes a certain mass-increment observed in a fragment of a fragment.

pair "(", , ",", , ")"; msmsrspec "[", , ":", "]"; MSMSspec [ ], "#M", , { ",", };

Semantics: the two range specifications in msmsrspec denote the fragment and the fragment of the fragment. Consequently, the second range specification describes a subset of labeling positions appearing in the first range specification. The list of weight pairs following the "#M" denote the mass-increments of the fragement and the fragment's fragment.

1H-NMR Measurements

The proton and electron of a hydrogen atom have a non-zero spin with opposite signs. For an unpaired hydrogen atom these spins would normally calcel out since the electron's spin shields the spin of the proton. However, for a hydrogen in a chemical bond and depending on the electronegativity of the bonding partners, the electron is pulled away from the proton an the proton shows up as a peak in the NMR spectrum. This chemical shift results in different resonance frequencies, and thus different hydrogens in the molecules can be discriminated. More importantly it is possible to discriminate labeled from unlabeled atom positions in the neughborhood of hydrogens, and thus, this 1H-NMR allows the description of positional 13C labeling enrichment.

A 1H-NMR measurement group specification is essentially a list of 13C labeling positions. It is assumed that all listed carbon atom positions have at least one bond to a hydrogen atom.

1HNMRspec , "#P", ;

Semantics: each number in "numberlist" describes an index of a labeling position connected to a hydrogen atom. The number of measurement values described by a 1H-NMR measurement group specification corresponds to the number of entries in "numberlist". The "#P" ("positions") discriminates the 1H-NMR specification from the other short notations.

13C-NMR Measurements

The nuclear spin of the most frequently occuring carbon isotope 12C (98.89%) is zero. However, the 13C isotopes (1.11%) possess an extra neutron and thus an non-zero spin. Consequently, a labeling position occupied by a 13C atom can be recognized in a NMR peak spectrum.

In addition, because the resonance frequency of a 13C position changes with additional 13C isotopes in the neighborhood, certain labeling constellations can be discriminated. The NMR device provides the information on how many 13C neighbors exist for particular 13C or a group or 13Cs. In general the frequency response will be split into multiplets having n+1 peaks where n is the number of adjacent 13Cs. The 13CFLUX 2 software allows the description of a neighborhood of up to three carbon atoms, which results in four different multiplets:

- Singlet. If there are no 13C isotopes on the adjacent labeling positions then the frequency response will (ideally) show a single peak, a singlet.
- Left and Right Doublet. If there is one 13C isotope on an adjacent labeling position the frequency response will split into two peaks of equal size, a doublet. Furthermore, it is usually possible to identify the location of the 13C isotope, which characterizes the doublet as a left doublet or a right doublet.
- Triplet. In the doublet case, it may happen that both adjacent labeling positions are chemically equivalent (e.g. in a symmetric molecule). In this situation it is impossible to a left ffrom a right doublet and a triplet can be observed in the frequency spectrum. The resonance will be split into three peaks with height ration 1:2:1.
- Doublet of Doublets (Double Doublet). In case a 13C isotope is surrounded by two 13C isotopes a special doublet can be observed. This doublet consists of two peaks of equal height. In contrast to the original doublet each peak is again split into two peaks, which gives the doublet of doublets.

In contrast to the 1H-NMR measurement group specifications a 13C-NMR specification not only lists the positions for which measurement data is available, but contains also the information about the observed multiplet type:

multiplet ("S" | "DL" | "DR" | "DD" | "T"), ; 13CNMRspec , "#", , { ",", };

Semantics: each entry in "numberlist" corresponds to a labeled atom position (i.e. 13C). Each multiplet associates an NMR multiplet measurement with the carbon atom postions p given in the subsequent "numberlist". Each multiplet type {S,DL,DR,DD,T} describes the labeling state of at most two carbon atoms in the neighborhood of the carbon atom at positions p:

- A "singlet" (S) describes the measurement of a 13C atom position p with only 12C atoms in the neighborhood positions p-1 and p+1. Positions p-1 and p+1 do not need to exist.
- A "left doublet" (DL) describes the measurement of a 13C atom position p having a 13C atom in the left neighborhood p-1. The atom position in the right neighborhood p+1 may not exist - in case it exists it has to be unlabeled (i.e. 12C).
- The "right doublet" (DR) is similar to the left doublet but assumes a 13C atom in the right neighborhood p+1.
- The "triplet" (T) describes the measurement of a 13C atom position p having a 13C isotope in either the left (p-1) or right (p+1) neighborhood. The triplet corresponds to the doublet in a chemically equivalent neighborhood.
- The "doublet of doublets" (DD) describes the measurement of a 13C labeled atom position having 13C isotopes in the left and right (i.e. a neighborhood of three 13C isotopes). From the viewpoint of mathematical modeling the difference between triplet and doublet of doublets is purely informational.

The number of measurement values described by a 13C-NMR measurement group specification corresponds to the number of entries in all "numberlist"s.

Generic Specification of Measurements

After all, the short notations for NMR and MS measurements given above are nothing but macros allowing the convenient description of sums of subsets of the full set of a metabolite's isotopomer fractions.

For the rare cases where these macros do not provide enough flexibility to model a measurement 13CFLUX 2 provides a generic way for the description of a set of measurements. Such a generic measurement specification consists of arbitrary formulas which may include the short notations for MS and NMR measurments and an additional "generalized cumomer" short notation describing a single sum of isotopomer fractions:

patternlist ("0" | "1" | "x"), { ("0" | "1" | "x") }; CumoSpec , "#", ;

Semantics: each symbol "0", "1", "x" appearing at position i in patternlist denotes the labeling state of the i-th atom of the metabolite specifed by poolname:

- the "0" denotes the absence of an isotopic labeling,
- the "1" denotes the presence of an isotopic labeling,
- the "x" denotes either absence of presence of an isotopic labeling, i.e. an indeterminate labeling state

Each generalized cumomer notation containing at least one "x" describes a sum of isotopomer fractions because a single "x" describes the labeled as well as the unlabeled state. The set of isotopomer fractions contained in this sum can be obtained by varying all n "x"-positions with all 2ˆn combinations of "0" and "1".

Now, a generic measurement specification consists of a semicolon-separated list of formulas. The identifiers in those formulas are covered by the EBNF rules for non-terminals MSspec, MSMSspec, 1HNMRspec, 13CNMRspec, and CumoSpec. However, each of these identifiers has to describe exactly one measurement value. The formulas are specified in a standard formula grammar (e.g. known from programming languages; omitted here for brevity) and support various operators ("+", "-", "*", "/", "yˆx", "sqrt(x)") including some built-in functions (abs, min, max, exp, ln, log2, log10).

FluxML Validation

FluxML documents can be validated based on the XML Schema grammar in file "fluxml.xsd" (part of the 13CFLUX 2 distribution). In general, there are two types of validation:

1. syntactical validation using an validating XML parser, e.g. used by the program **xmllint**(1).
2. semantical validation using the FluxML parser library, e.g. by using the program **fmllint**(1).

Semantical validation is done automatically by all applications using the FluxML parser library, like **fwdsim**(1), **ssampler**, etc. In order to use syntactical validation the root element of the FluxML document needs to be modified. The modified root element should look like this:

```
<fluxml
 xmlns="http://www.13cflux.net/fluxml"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.13cflux.net/fluxml
   http://www.13cflux.net/xml-schema/fluxml.xsd">
...
</fluxml>
```

Be aware that a syntactical validation of FluxML requires the download of server fairly large MathML XML Schema definition. In order to prevent this download you may use **fmllint**(1) for syntactical validation, which uses a locally installed version of the MathML grammar.

Example

```
<?xml version="1.0"?>
<fluxml xmlns="http://www.13cflux.net/fluxml">
  <info>
    <name>Small Example Network</name>
  </info>
  <reactionnetwork>
    <metabolitepools>
      <pool atoms="3" id="A"/>
      <pool atoms="3" id="B"/>
      <pool atoms="2" id="C"/>
      <pool atoms="3" id="D"/>
      <pool atoms="1" id="E"/>
      <pool atoms="3" id="F"/>
    </metabolitepools>
    <reaction id="v1">
      <reduct cfg="abc" id="A"/>
      <rproduct cfg="abc" id="B"/>
    </reaction>
    <reaction id="v2">
      <reduct cfg="abc" id="B"/>
      <rproduct cfg="abc" id="D"/>
    </reaction>
    <reaction id="v3">
      <reduct cfg="abc" id="D"/>
      <rproduct cfg="abc" id="B"/>
    </reaction>
    <reaction id="v4">
      <reduct cfg="abc" id="B"/>
      <rproduct cfg="bc" id="C"/>
      <rproduct cfg="a" id="E"/>
    </reaction>
    <reaction bidirectional="false" id="v5">
      <reduct cfg="abc" id="B"/>
      <reduct cfg="de" id="C"/>
      <rproduct cfg="bcd" id="D"/>
      <rproduct cfg="a" id="E"/>
      <rproduct cfg="e" id="E"/>
    </reaction>
    <reaction bidirectional="false" id="v6">
      <reduct cfg="abc" id="D"/>
      <rproduct cfg="abc" id="F"/>
    </reaction>
    <reaction id="efflux_E">
      <reduct cfg="a" id="E"/>
    </reaction>
    <reaction id="efflux_F">
      <reduct cfg="abc" id="F"/>
    </reaction>
  </reactionnetwork>
  <configuration name="default">
    <comment>Eine Belegung für die Input-Pools</comment>
    <input pool="A" type="isotopomer">
      <label cfg="010">1</label>
    </input>
    <constraints>
      <net><textual>v2 &gt;= v3; v1=100;</textual></net>
      <xch><textual>v2=0; v3=0; v5=0; v6=0; v4=0</textual></xch>
    </constraints>
    <measurement>
      <model>
        <labelingmeasurement>
          <group id="m1" scale="one">
            <textual>F#M0,1,2,3</textual>
          </group>
        </labelingmeasurement>
      </model>
      <data>
        <datum id="m1" stddev="0.003" weight="0">0.0001</datum>
        <datum id="m1" stddev="0.003" weight="1">0.8008</datum>
        <datum id="m1" stddev="0.003" weight="2">0.1983</datum>
        <datum id="m1" stddev="0.003" weight="3">0.0009</datum>
      </data>
    </measurement>
    <simulation>
      <variables>
        <fluxvalue flux="v2" type="net">110</fluxvalue>
        <fluxvalue flux="v4" type="net">20</fluxvalue>
      </variables>
    </simulation>
  </configuration>
</fluxml>
```

See Also

edopt(1), edscanner(1), fitfluxes(1), fmllint(1), ftbl2fml(1), fwdsim(1), simreport(1), ssampler(1), sscanner(1), xmllint(1)

Author

Michael Weitzel
    This manpage was written by Michael Weitzel <info@13cflux.net>.

3.2 fwdsim

fwdsim — the 13CFLUX 2 XML document format for simulation data

Description

The FWDSIM XML document format is used to store the simulation data generated by the programs **fwdsim**(1) and **fitfluxes**(1). In particular, FWDSIM documents contain:

- the parametrization of the stoichiometry including the analytical solution of the stoichiometric system and all flux values (optionally with computed standard deviations),
- all or selected EMU vectors or cumomer fractions for the metabolite pools of the reaction network (necessary for evaluation of the measurement model),
- the simulated measurement model, including all simulated measurement values arranged in measurement groups and the corresponding residuals,
- for **fitfluxes**, the settings of the optimization algorithm and (sometimes) a vector of flux sensitivities.

In contrast to FluxML, the FWDSIM XML documents are never edited by humans. The layout of the FWDSIM document format is briefly discussed below.

The FWDSIM Elements

The elements of the FWDSIM document format belong to the XML namespace denoted by the URI "http://www.13cflux.net/fwdsim". FWDSIM's root element usually includes corresponding namespace attribute ("xmlns"). The preamble and root element of a FWDSIM document should look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<fwdsim xmlns="http://www.13cflux.net/fwdsim">
...
</fwdsim>
```

The following briefly presents the elements of a FWDSIM document.

Element: /fwdsim

The root element of FWDSIM contains four sub-elements which are all mandatory. These elements describe the parametrization of the stoichiometry, list the computed EMUs or Cumomer fractions, give simulated measurement values (and residuals), and provide information about the optimization run performed by **fitfluxes**(1).

```
<fwdsim xmlns="http://www.13cflux.net/fwdsim">
  <stoichiometry> ... </stoichiometry> [1]
  <simulation> ... </simulation> [1]
  <measurements> ... </measurements> [1]
  <optimization> ... </optimization> [1]
</fwdsim>
```

Element: /fwdsim/stoichiometry

The parametrization of the stoichiometry consists of at least two "flux" elements (one coming from a substrate and the other leaving the network).

```
<stoichiometry>
  <flux> ... </flux> [2..*]
</stoichiometry>
```

Element: /fwdsim/stoichiometry/flux

A "flux" references the unique ID of a reaction in underlying FluxML document. The flux value is separated into net and exchange flux.

```
<flux
 id="xsd:ID [1]">
  <net> ... </net> [1]
  <xch> ... </xch> [1]
</flux>
```

Element: /fwdsim/stoichiometry/flux/{net,xch}

The "net" and "xch" elements give the numerical flux values of a reaction's net and exchange flux. In case **fwdsim**(1) was instructed perform a statistical analysis (command line option `-s`) the attribute "stddev" contains the computed standard deviation value. In case "stddev" contains a "?" the flux was indeterminable. All net and exchange fluxes are classified as "free", "dependent", "constraint", or "quasi-constraint":

- "free" fluxes are the free parameters of the stoichiometric system. Their value may be chosed freely as long as it satisfies the inequality constraints.
- "constraint" fluxes have a fixed value which may not be adjusted by the optimization algorithm. Each constraint flux reduces the degrees of freedom of the stoichiometric system by one.
- "dependent" fluxes are determined by free fluxes or by a combination of free fluxes and constraint fluxes through the linear relation given in the value of the "net" and "xch" elements. These fluxes may not be modified by the optimization algorithm.
- "quasi-constraint" fluxes are dependent fluxes which are fully determined by constraints, i.e. they do not depend on any of the free fluxes. The dependency relation is given by the value of the "net" and "xch" elements.

```
<{net,xch}
 type="xsd:string matching ('free'|'dependent'|'constraint'|'quasi-constraint')  ↩
    [1]"
 value="xsd:double [1]"
 stddev="xsd:double [0..1]"/>
```

Element: /fwdsim/simulation

Element "simulation" lists the simulation results for selected or all EMUs or Cumomer fractions of the metabolite pools (what is being computed depends on the settings in the "simulation" element of the underlying FluxML document; see **fluxml**(5)). The attribute "type" is used to discriminate the simulation methods (EMU / Cumomer).

```
<simulation
 type="xsd:string (value comes from list: {'emu'|'cumomer'}) [1]">
  <pool> ... </pool> [2..*]
```

```
</simulation>
```

Element: /fwdsim/simulation/pool

A list of simulation values of the labeling enrichment of metabolite pools are given by element "pool". The attribute "id" refers to a metabolite pool in the corresponding FluxML document (see **fluxml**(5)). The simulated EMUs or Cumomer fractions of the metabolite pool are given by a number of "value" elements.

```
<pool
 id="xsd:ID [1]">
  <value> ... </value> [1..*]
</pool>
```

Element: /fwdsim/simulation/pool/value

By the use of the attribute "cfg" the numerical value(s) in the "value" element are associated with a specific labeling composition (Cumomer) or molecule fragment (EMU). For an EMU simulation the "value" element contains double precision numbers separated by "sep" elements (the vector of mass isotopomer fractions). For a Cumomer simulation, "value" contains a single double precision number (the Cumomer fraction)

```
<value
 cfg="xsd:string (pattern = [1x]+) [1]">
  <!-- Mixed content -->
  Start Sequence [0..*]
  <sep/> [1]
  End Sequence
</value>
```

Element: /fwdsim/measurements

Element "measurements" lists all simulated measurement groups of the measurement model. Attribut "residual" gives the value of the objective function that is used for the optimization in **fitfluxes**(1). The value of "residual" equals the sum of the residuals given in the subordinate "mgroup" elements.

```
<measurements
 residual="xsd:double [1]">
  <mgroup> ... </mgroup> [1..*]
</measurements>
```

Element: /fwdsim/measurements/mgroup

Element "mgroup" gives the specification of the measurement group (i.e. a short notation or a measurement model expression in element "spec") and a list of simulated measurement values. Attribute "id" refers to the original unique ID of the measurement group in the underlying FluxML file. Attribute "scale" gives the computed group scaling fractor. The "type" attribute characterizes the used measurement technology. The measurement values in element "value" are given as scalar double precision floating point numbers.

```
<mgroup
 id="xsd:ID [1]"
 norm="xsd:double [1]"
 scale="xsd:double [1]"
 type="xsd:string ('MS'|'MSMS'|'1HNMR'|'13CNMR'|'GENERIC'|'FLUX'|'POOL') [1]">
  <spec> xsd:string </spec> [1]
  <value> xsd:double </value> [1..*]
</mgroup>
```

Element: /fwdsim/optimization

Element "optimization" and sub-element "sensitivities" give a list of "param" elements containing the settings and return status of the optimization algorithm used by **fitfluxes**(1) and the objective function's sensitivities (gradient) computed by **fwdsim**(1) and **fitfluxes**(1).

```
<optimization>
  <param> ... </param> [0..*]
  <sensitivities> [0..1]
    <param> ... </param> [0..*]
  </sensitivities>
</optimization>
```

Element: /fwdsim/.../param

Element "param" represents a key-value pair with additional type information. The key is given by attribute "name". Attribute "type" gives the type information. The value of the "param" element contains the value of the key-value pair.

```
<param
 name="xsd:string [1]"
 type="xsd:string matching ('boolean'|'integer'|'real'|'string'|'net'|'xch')  ↩
    [1]"/>
```

FWDSIM Validation

FWDSIM documents can be validated based on the XML Schema grammar in file "fwdsim.xsd" (part of the 13CFLUX 2 distribution).

Semantical validation is done automatically by all applications using the FWDSIM parser library, like **collectfitdata**(1). In order to use syntactical validation the root element of the FWDSIM document needs to be modified. The modified root element should look like this:

```
<fwdsim
 xmlns="http://www.13cflux.net/fwdsim"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.13cflux.net/fwsim
   http://www.13cflux.net/xml-schema/fwdsim.xsd">
...
</fwdsim>
```

Example

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<fwdsim xmlns="http://www.13cflux.net/fwdsim">
```

```
<stoichiometry>
  <flux id="efflux_E">
    <net type="dependent" value="60">3*v4.n</net>
    <xch type="constraint" value="0">efflux_E.x</xch>
  </flux>
  <flux id="efflux_F">
    <net type="dependent" value="80">v1.n-v4.n</net>
    <xch type="constraint" value="0">efflux_F.x</xch>
  </flux>
  <flux id="v1">
    <net type="constraint" value="100">v1.n</net>
    <xch type="constraint" value="0">v1.x</xch>
  </flux>
  <flux id="v2">
    <net type="free" value="110">v2.n</net>
    <xch type="constraint" value="0">v2.x</xch>
  </flux>
  <flux id="v3">
    <net type="dependent" value="50">-v1.n+v2.n+2*v4.n</net>
    <xch type="constraint" value="0">v3.x</xch>
  </flux>
  <flux id="v4">
    <net type="free" value="20">v4.n</net>
    <xch type="constraint" value="0">v4.x</xch>
  </flux>
  <flux id="v5">
    <net type="dependent" value="20">v4.n</net>
    <xch type="constraint" value="0">v5.x</xch>
  </flux>
  <flux id="v6">
    <net type="dependent" value="80">v1.n-v4.n</net>
    <xch type="constraint" value="0">v6.x</xch>
  </flux>
</stoichiometry>

<simulation type="emu">
  <pool id="A">
    <value cfg="111">0
      <sep/>1
      <sep/>0
      <sep/>0
    </value>
    <value cfg="x11">0
      <sep/>1
      <sep/>0
    </value>
    <value cfg="1x1">1
      <sep/>0
      <sep/>0
    </value>
    <value cfg="11x">0
      <sep/>1
      <sep/>0
    </value>
    <value cfg="1xx">1
      <sep/>0
    </value>
    <value cfg="x1x">0
      <sep/>1
    </value>
    <value cfg="xx1">1
      <sep/>0
    </value>
  </pool>
```

```
    <pool id="B">
      <value cfg="111">2.1164021164021168e-05
        <sep/>.9335873015873015
        <sep/>.0660952380952381
        <sep/>.0002962962962962963
      </value>
      <value cfg="x11">.004444444444444445
        <sep/>.991111111111111
        <sep/>.0044444444444444444
      </value>
      <value cfg="1xx">.9333333333333333
        <sep/>.06666666666666667
      </value>
      <value cfg="xxx">1</value>
      <value cfg="xx1">.9333333333333333
        <sep/>.06666666666666667
      </value>
      <value cfg="x1x">.06666666666666667
        <sep/>.9333333333333333
      </value>
    </pool>
    <pool id="C">
      <value cfg="11">.0044444444444444444
        <sep/>.9911111111111112
        <sep/>.0044444444444444444
      </value>
      <value cfg="x1">.9333333333333333
        <sep/>.06666666666666667
      </value>
      <value cfg="xx">1</value>
      <value cfg="1x">.06666666666666668
        <sep/>.9333333333333333
      </value>
    </pool>
    <pool id="D">
      <value cfg="111">6.34920634920635e-05
        <sep/>.8007619047619047
        <sep/>.1982857142857143
        <sep/>.0008888888888888889
      </value>
      <value cfg="x11">.013333333333333334
        <sep/>.9733333333333332
        <sep/>.013333333333333332
      </value>
      <value cfg="1xx">.7999999999999998
        <sep/>.19999999999999998
      </value>
      <value cfg="xxx">1</value>
      <value cfg="xx1">.8
        <sep/>.2
      </value>
      <value cfg="x1x">.19999999999999998
        <sep/>.7999999999999998
      </value>
    </pool>
    <pool id="E">
      <value cfg="1">.9333333333333333
        <sep/>.06666666666666667
      </value>
      <value cfg="x">1</value>
    </pool>
    <pool id="F">
      <value cfg="111">6.34920634920635e-05
        <sep/>.8007619047619047
        <sep/>.1982857142857143
        <sep/>.0008888888888888889
      </value>
      <value cfg="xxx">1</value>
    </pool>
  </simulation>

  <measurements residual=".0001641421903442334">
    <mgroup
     id="m1"
     norm=".0001641421903442334"
     scale="1.0000490054637636"
     type="MS">
      <spec>F#M0,1,2,3</spec>
      <value>6.34920634920635e-05</value>
      <value>.8007619047619047</value>
      <value>.1982857142857143</value>
      <value>.0008888888888888889</value>
    </mgroup>
  </measurements>

  <optimization>
    <sensitivities>
      <param name="v2" type="net">0</param>
      <param name="v4" type="net">0</param>
    </sensitivities>
  </optimization>

</fwdsim>
```

See Also

collectfitdata(1), fitfluxes(1), fwdsim(1), fwdsimflt(1), simreport(1), xmllint(1)

Author

Michael Weitzel
This manpage was written by Michael Weitzel <info@13cflux.net>.

Chapter 4

Bibliography

[1] Michael Weitzel, *High Performance Algorithms for Metabolic Flux Analysis*, University of Siegen.