influx_si Documentation

Release 4.4.2

Serguei SOKOL

July 04, 2017

CONTENTS

CHAPTER

ONE

INTRODUCTION

`influx_s` and `influx_i` are programs designed for flux and metabolite concentration estimation based on labeling data using 13C isotopes. The former works with stationary data while the latter is able to simulate instationary labeling. Both work in metabolically stationary context. The whole project is referred as `influx_si`. Note also that the term `influx_si` is used in contexts where `influx_s` and `influx_i` are interchangeable.

influx_si

Flux and metabolite concentration values are obtained as a result of a fitting between simulated labeling data and the data measured by MS or NMR techniques. In this documentation the terms *fitting* and *optimization* are used as synonyms.

influx_s

For the theory behind flux calculations in stationary labeling context see the following papers:

Wiechert, W., Möllney, M., Isermann, N., Wurzel, M., and de Graaf, A. A. (1999). Bidirectional reaction steps in metabolic networks: III. Explicit solution and analysis of isotopomer labeling systems. *Biotechnol Bioeng,* 66(2), 69-85.

Antoniewicz, M. R., Kelleher, J. K., and Stephanopoulos, G. (2007). Elementary metabolite units (EMU): a novel framework for modeling isotopic distributions. *Metab Eng*, 9(1), 68-86.

Sokol, S., Millard, P., and Portais, J-C. (2012). influx_s: increasing numerical stability and precision for metabolic flux analysis in isotope labeling experiment. *Bioinformatics*, 2012, 28, 687-693

The main additional value to flux calculation of `influx_si` compared to other publicly available software (13CFlux, OpenFlux, INCA, ...) is the usage of NLSIC algorithm for fitting purposes. This algorithm provides:

- more reliable convergence which results in better numerical precision, i.e. even started from random initial points, it converges to the same solution if no local minima are present. So the spread of final solutions is close to zero.
- better accuracy, i.e. the found numerical solution lies closer to the theoretical solution than solutions provided by concurrent minimization algorithms. Thus, `influx_s` provides better numerical accuracy.

For more details, see the paper on `influx_s` cited above.

Moreover, `influx_s` provides:

- both cumomer and EMU frameworks for describing label distribution in the metabolites;
- parallel experiment treatment;
- metabolite concentration estimation in particular stationary contexts (since v2.0. A methodology behind metabolite concentration evaluation is not yet published at the moment of this writing.);
- a possibility to deal with metabolite pool confusion appearing either in compartmentation or in coelution;

- taking into account non carbon carrying fluxes like the balances of ADP/ATP, H2O, energy, electrons and so on;
- an optional automatic choice of free fluxes;
- optional equality and inequality constraint on fluxes and metabolite concentrations;
- short time execution and design for many core computers. So it facilitates high throughput flux calculations in parallel way;
- a 'least norm' option that, in presence of structurally non identifiable fluxes, still allows to estimate some of fluxes (those remained identifiable);
- a khi2 statistical test 'goodness of fit'
- an optional automatic elimination of outliers;
- a command line interface letting an easy integration in automatic processing chains as well as many others features and options;
- a possible scripting of post-treatment or graphic generating tasks;
- multi-platform support. It runs everywhere R and Python run, i.e. on Linux, Windows, MacOS and other Unix variants.

`influx_i`

Instationary labeling (hence the final 'i' in the name) is the domain of `influx_i`. The theory of instationary labeling was developed for example in

Katharina Nöh, Wolfgang Wiechert (2006) Experimental Design Principles for Isotopically Instationary 13C Labeling Experiments *Biotechnology and Bioengineering*, 94(2), 234-251

Sokol S, Portais J-C (2015) Theoretical Basis for Dynamic Label Propagation in Stationary Metabolic Networks under Step and Periodic Inputs. *PLoS ONE* 10(12): e0144652. doi:10.1371/journal.pone.0144652

As `influx_i` capitalizes on `influx_s` development and shares a big part of code, `influx_i` presents the same advantages as listed in the previous section. It uses the same FTBL file format for network and measurements definitions and includes all options available for `influx_s`. Instationary labeling data can be supplied by an additional tab formatted ASCII file making a shift from stationary to instationary calculations as simple as possible. Some of the advantages of `influx_i` over the concurrent software coping with instationary labeling data are:

- fast calculations (e.g. on our Intel Xeon 2.50GHz workstation, `e_coli_i` case runs in less then 25s and the most important optimization part takes as low as 17s);
- parallel experiment treatment;
- available choice between first and second order time schemes for ODE (ordinary differential equations) resolution;
- unconditional stability during ODE solving.

Documentation organization

Changes brought to every new version and bug fixes are resumed at the beginning of the next chapter *Change Log* which is also distributed as a stand alone PDF file.

The rest of the documentation is organized as follows. *Installation* chapter provides brief instructions for software installation. *Quick start* chapter gives an opportunity to a user to quickly start and evaluate the software and to see if it corresponds to what he is looking for. A more detailed but still short *User's manual* precedes a *Programmer's documentation*. The latter chapter can be safely skipped by a user not interested in developing new features or fixing some problems in `influx_si`. A small collection of *How to...* and *Troubleshooting* notice conclude the documentation.

Licensing

The original version of `influx_si` software was developed in the MetaSys team in the LISBP, Toulouse, FRANCE.

The software is licensed under the Educational Community License, Version 2.0 (the "License"); you may not use this software and documentation except in compliance with the License.

If you publish results obtained with `influx_s` you have to cite the original paper in Bioinformatics 2012 (cf. above). A paper describing `influx_i` is yet to publish.

If you re-distribute `influx_si` alone or included in other software packages, you have to ensure that the end user abide to the terms of this license.

You may obtain a copy of the License *here* or at

http://www.opensource.org/licenses/ECL-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Software and documentation author:

Serguei SOKOL, INRA, France <sokol [at] insa-toulouse.fr>

CHAPTER

TWO

CHANGE LOG FOR INFLUX_SI

2017-07-04 version 4.4.3

New features:

- added 95% quantile in monte-carlo/cost/ci field in _res.kvh file. It makes possible to do a mono-tail chi2 test for goodness of fit
- added possibility for FTBL files to be encoded in UTF16 and UTF32 (based on case reported by Lucille Stuani, INSERM, Toulouse, France)
- added an explicit error message if no label information could be found in parallel experiment FTBL (reported idem)

Bug fix:

- fixed a warning in Monte-Carlo iteration about multiple values in if() close (reported idem)

2017-06-15 version 4.4.2

New features:

- added a new field "constrained net-xch01 fluxes" to the result kvh file
- ff2ftbl.py: instead of only free fluxes, all fluxes are read in kvh file. Thus in a modified FTBL, a partition on free/dependent/constrained fluxes can differ from those used in the kvh file.
- ff2ftbl.py: if kvh and ftbl files have the same prefix, only this prefix can be given as a unique command line argument

Bug fix:

- ff2ftbl.py: fixed "end of line" bug on Windows platform
- plot_smeas.R: fixed metabolite names retrieving in parallel experiments
- plot_smeas.R: fixed disgraceful exit if simulation in influx_s failed

2017-05-24 version 4.4.1

New features:

- in plot_smeas.R and plot_imass.R few cosmetic improvements in plot titles and legends

Bug fix:

- fixed non varying free pools in influx_s
- fixed cases where some cumomer (or EMU) weights can have no cumomers (EMU)

- fixed libs.R by including some files (impacted preamble.R)

2017-05-22 version 4.4

New features:

- added plot_smeas.R file to be included in posttreat_R field in FTBL/OPTIONS. It plots all stationary measurements vs their simulated counterparts in a pdf file.
- added preamble.R, an example of starting session when working with mynetwork.RData issued from save_all.R or save_minenv.R
- R can be again of version 3.3+ (not necessarily 3.4+)
- minor speedup in instationary simulations

Bug fix:

- fixed names in dev vector of pool measurements

2017-04-28 version 4.3

New features:

- speed up of about 30-40% is achieved for instationary simulations with 2nd order time scheme (need upgrade R at least to 3.4.0).
- in plot_imass.R, each measured mass fragment is presented in a separate plot instead of regrouping all fragments for a given metabolite on the same plot.
- in plot_imass.R, non measured metabolites are plotted too now

Bug fix:

- fixed Monte-Carlo iterations with –np=1
- added a mention of python-libsbml in installation procedure

2017-03-30 version 4.2

New features:

- added a script ftbl2metxml.py converting an ftbl to an xml file suitable for visualization on http://metexplore.toulouse.inra.fr . Additionally, it reads flux values from corresponding ..._res.kvh file (if available) and put them in files ..._net.txt, ..._fwd.txt and ..._rev.txt for later copy/pasting on the MetExplore site (suggested by Tony Palama, INSA, Toulouse, France)
- added comment tags '###' to txt network format (recognized by txt2ftbl.py and respectively '//##' tag in FTBL format recognized by all programs using FTBL format) to mark a new pathway. It allows ftbl2metxml.py to assign reactions to pathways and thus make a network graph more readable.

Bug fix:

- a duly error message is added to signal a network without any label entry in a reduced cumomer network.

2017-03-03 version 4.1

New features:

- added explicit error message when label transitions are missing for any reaction in NETWORK section
- improved speed of labeling simulation in influx_i
- added parameter –tblimit[=0] for trace back limit in errors generated by python (for developers only).

Bug fix:

- fixed error appearing in influx_i during parallel experiments in situation where time intervals are different in different experiments (reported by Maria Fatarova, INSA, Toulouse, France)
- fixed file creation in plot_imass.R (it created pdf in the current directory instead of the working one)

2016-12-20 version 4.0.1

Bug fix:

- file txt2ftbl.py was lacking in the previous version (reported by Tony Palama, INSA, Toulouse, France)

2016-12-19 version 4.0

New features:

- parallel experiments (i.e. same metabolic state but different label entries) can now be processed both in stationary (influx_s) and instationary (influx_i) labeling
- reaction having more than 2 metabolites on ever side can now be entered in FTBL as a series of reactions with the same name
- metabolites with no carbon transitions (like ATP, NADP etc. when they are just co-factors) can now be entered in NETWORK section. They can have stoechiometric coefficients different from 1
- the same metabolite can now appear on both sides of a reaction. It can be helpful for some special carbon shuffles
- reactions without carbon transitions can now be entered in a new FTBL section NO-TRACER_NETWORK. It is a good place to enter for example biomass reaction. Stoechiometric coefficients different from one are allowed at this place
- new utility txt2ftbl.py translates an easier readable/writable format for chemical reactions to an FTBL file
- added –addnoise option to facilitate creation of realistic simulated measurements

Bug fixes:

- fixed a bug preventing Monte-Carlo simulations with influx_i (reported by Maria Fatarova, INSA, Toulouse, France)

2016-07-29 version 3.2

New features:

- added controls for coherence of label transitions
- added detection of incoherent fragments in MASS_MEASUREMENTS (e.g. longer one than a whole molecule)

- in LABEL_INPUT section, if incomplete labeled forms don't sum up to 1, and several labeled forms are absent, the lacking label fraction is assigned to the fully unlabeled form
- R package 'snow' is no more needed on windows platform to run Monte-Carlo simulations in parallel mode
- on all platforms, Monte-Carlo simulations are now run on a PSOCK cluster and no more on a FORK cluster (Linux) or SOCK (Windows)
- inequalities involving only constrained fluxes or depending solely on such fluxes are now simply ignored with a warning
- fixed Jacobian calculation when no free flux exists

Bug fixes:

- fixed building a library mult_bxxc.dll on Windows platform (reported by Tony Palama, INSA/MetaToul Toulouse, France)
- fixed building mult_bxxc.so in parallel context (multiple ftbls)
- fixed formulas in equalities and inequalities with flux names having parenthesis, brackets, spaces and alike
- cluster workers are parsimoniously created in case of multiple starting points

2016-06-13 version 3.1

New features:

- added controls and explicit error messages for DEVIATION=0 in FTBL file
- added column "p-value" to residual values in _res.kvh file (may help for outlier choice)
- added check-points for infinite values that can appear in residual and Jacobian

Bug fixes:

- fixed EMU mode in instationary case
- fixed renewal of mult_bxxc.so library in case of source update
- fixed cost value calculation in case of outlier exclusion

2016-04-18 version 3.0.1

Bug fixes:

- fixed including mult_bxxc.so file in zip archive which prevented from proper compiling of this dynamic library
- fixed an absence of C++11 flag on platforms where it is not a standard by default.

2016-04-15 version 3.0

New features:

- influx_i.py is introduced for instationary label modeling
- some critical calculations are written in C++ so some compilation is needed at first execution.
- new optional package limSolve is used and need to be installed (as well as its dependencies) if `--lim` option is used

- more TIMEIT points introduced for finer time control

Bug fixes:

- fixed performance issue in slam package for '-' and '+' operations
- fixed sparse matrix preparation when there is only one non zero entry

2016-02-18 version 2.15

New features:

- calculation speed was improved due to the use of packages slam and rmumps instead of Matrix;
- added "cpu" field when timing is requested

Bug fixes:

- fixed a bug preventing a use of a flux added in EQALITIES in measured fluxes (reported by Edern Cahoreau, INSA, Toulouse, France)
- fixed minor problems in MC iterations (parameter distribution was not significantly affected)

2015-01-19 version 2.14

New features:

- `commandArgs` field in FTBL file can have comments in it and occur more than once somewhere in the `OPTIONS` section
- `--DEBUG` option is removed as obsolete
- R package `bitops` is not required anymore to be installed (valid for R-3.0.0 or higher)

Bug fixes:

- fixed a bug in delivering an error message when `commandArgs` had a comment in it
- fixed the precedence of command line options over `commandArgs` given in FTBL
- fixed a bug in parsing FTBL file having a BOM (invisible utf8 encoding mark) in it (reported by Yanfen Fu, University of Washington, USA)
- fixed representation of growth fluxes by ftbl2xgmml.py utility

2014-09-17 version 2.13

New features:

- `posttreat_R` field can have several file names separated by `'; '`
- added explicit error message if a valid float value is missing for free or constrained flux
- added explicit error message if no dependent flux is included in the balance on any metabolite (suggested by a case submitted by Marc O. Warmoes, Cornell University, USA)
- in the documentation, added a paragraph about consulting offer
- result .kvh file is greatly shortened, keeping only essential information. Custom additional information can be stored in some file via `posttreat_R` option
- now, `influx_s` returns a non zero code to shell if an error occurred during execution;
- added a parameter `monotone` to the control list of NLSIC.

Bug fixes:

- fixed a bug in generating EMU systems (manifested in some special cases)
- fixed an error preventing from producing a message suggesting a new partition among dependent, free and constrained fluxes (reported by Stéphane Mottelet, University of Compiegne, France)
- fixed metabolite pooling weights (manifested in some special cases)
- Windows platform: fixed passing command line options to R code
- Windows platform: precompiled `nnls` R package (version 32 bits) can produce wrong results. Recompile it by hand or use 64 bits version.

2014-07-02 version 2.12

New features:

- parsing badly formatted ftbl files is made more robust

Bug fixes:

- fixed a bug in `--emu` option (was introduced in v.2.11)

2014-06-12 version 2.11.1

Bug fix:

- an option `--noopt` broken in 2.11 is repaired (reported by Pierre Millard, Manchester Institute of Biotechnology, UK)

2014-06-11 version 2.11

New features:

- a joint use of the options `--fseries` and `--irand` gives a possibility to mix fixed and random values in starting points
- post treatment option `posttreat_R` is introduced in FTBL file. A user script written in R can be used to chain flux estimation and customized data treatment, e.g. graph plotting in a pdf file or simply saving of all the environment for later use and exploring in an R interactive session
- added optional `INEQUALITIES/METAB` section in FTBL file. It can be helpful to limit variations of estimated metabolite concentrations (suggested by Marc Carnicer, INSA of Toulouse, France)
- added optional `EQUALITIES/METAB` section in FTBL file. It can be helpful to fix a ratio between varying metabolite concentrations (suggested by Marc Carnicer, idem)
- the default value of `btdesc` parameter in NLSIC algorithm is lowered from 0.75 to 0.1. In some cases, it can accelerate the optimization convergence.

Bug fixes:

- fixed EMU list of participants in measurements
- fixed measurement matrix when only one measurement is available
- fixed a fatal error when no free flux is available but at least one metabolite quantity must be estimated
- fixed a bug in pooled measurements. This bug was harmful only if the metabolite pooling was used in more than one type of measurements, e.g. mass *and* labeling. If only one type of measurements used pooling (e.g. mass), the bug was without effect

- where appropriate, a word "labeling" was replaced by "label" in the field names of the `_res.kvh` file
- fixed superfluous backtracking iterations present for some particular residual functions
- if a flux or a metabolite is present more than once in formulas of (IN)EQUALITIES sections, its coefficients are summed up instead of taking only the last one
- fixed a fatal error in generating inequality matrix for net fluxes

2014-04-08 version 2.10

New features:

- added an option `--tikhreg` which is an alternative for `--ln` option. In case of rank deficient Jacobian, it calculates an increment step of the smallest norm in *approximative* way. It is done by Tikhonov regularization
- added an option `--ffguess` which makes to ignore the partition between free and dependent fluxes defined in FTBL file(s) and automatically guess a new free/dependent flux partition (suggested by Roland Nilsson, Karolinska Institutet, Sweden)
- added utility `ftbl2kvh.py` which is useful for debugging purposes only
- utilities `ftbl2xgmml.py`, `ftbl2cumoAb.py`, `ftbl2netan.py` and `ftbl2kvh.py` are rewritten in such a way that if no output redirection (with operands '>' or '|') occurs on the command line, the name of the output file is automatically derived from the input one. The suffix `.ftbl` is simply replaced with `.xgmml`, `.sys`, `.netan` or `.kvh` respectively. Thus a plain drag-and-drop can work with these utilities
- option `--TIMEIT` reports times with subsecond precision. The actual precision depends on the platform but typically a 0.01 s precision should be available. On Windows, the precision is usually 1/60 of a second

Bug fixes:

- fixed `include_growth_flux` option for `ftbl2cumoAb.py` utility (reported by Marc Carnicer, INSA of Toulouse, France)
- fixed a bug preventing from checking for a linear dependence between rows of stoechiometric matrix if no constrained net flux is defined in the FTBL file (reported by Roland Nilsson, idem)

2014-02-05 version 2.9

New features:

- utility `ftbl2xgmml.py` replaces `ftbl2rsif.py`. Now, a standalone XGMML file describes both a network and its graphical properties instead of a collection of files where this information was spread. New graphical conventions are now used.
- an obsolete utility `ftbl2cytoscape.bat` is removed form the distribution.
- added utility `res2ftbl_meas.py` generating measurement section from a result file _res.kvh
- added utility `expa2ftbl.R` transforming stoechiometric information in EXPA format (http://gcrg.ucsd.edu/Downloads/ExtremePathwayAnalysis) to various sections of FTBL file, namely to `EQUALITY` section where non carbon carrying fluxes can appear
- files generated by `influx_s` and collecting values for graphical representation (like `edge.netflux.mynetwork` and others) are renamed by adding a suffix `.attrs` to make them compatible with Cytoscape v3.0

- utilities `ffres2ftbl.sh` and `ff2ftbl.py` distributed for a long time ago, are now mentioned in the documentation

Bug fixes:

- fixed `--fullsys` option broken in the previous release.

2014-01-27 version 2.8

New features:

- `EQUALITY` section in FTBL file may include fluxes absent in `NETWORK` section, e.g. fluxes involved in non carbon carrying reactions (suggested by Roland Nilsson, Karolinska Institutet, Sweden)
- when a meaningful partition between free and dependent fluxes cannot be made, a proposition is made as to stoechiometric equations to be eliminated by hand (suggested by Roland Nilsson, idem)
- when `--clownr` option is used, reduced size of cumomer system is more efficient than without this option (replace a fix in 2.6 version)

Bug fixes:

- fixed useless memory consumption during ftbl parsing when `--emu` option is used and very long molecules (say >20 carbons) are present (reported by Roland Nilsson, idem)
- some error messages are made more explicit during FTBL parsing
- fixed Jacobian calculation for condensing input reaction
- fixed matrix constructions when no free flux is defined
- fixed b term for full cumomer system
- fixed inequality enforcement when adaptive backtracking is used in NLSIC
- fixed inequality precedence, now specific inequalities from FTBL file prevail on `--cupn=CUPN` option

2013-10-22 version 2.7

New features:

- Monte-Carlo simulations are done in parallel on Windows platform too (needs R package `snow`)
- if the option `--seed=SEED` is used, Monte-Carlo simulations are now reproducible even if run in parallel on multiple cores
- for rank deficient Jacobian, the inequalities are now better enforced
- starting value for `maxstep` parameter is set to $10||p||$ instead of $||p||$ where p is a vector of starting values for free parameters to fit.

Bug fixes:

- fixed a bug preventing to report partial Monte-Carlo results if some simulations failed and some not
- fixed a bug making to use all available cores instead of only one when `NP` was set to 1
- fixed a fatal error when inequality enforcement fails
- error and log messages during zero cross passes are made more explicit
- fixed sending some error messages on standard output instead of `.err` file
- when cumomer matrix is singular, fixed an error message about zero fluxes

2013-10-02 version 2.6

New features:

- added option `--sln` (solution least norm) which applies 'least norm' to the whole solution vector of free parameters, not just to the increment vector (like `--ln` does)
- a parallel calculation of multiple FTBLs is moved from python to R code. In such a way, some economies of repeated R starting up and library loading are made
- when zero crossing is used (`--zc=ZC`) a third pass is added without any zc constraint.
- added an option `maxstep` to control list of nlsic(). In some situations, it can make the convergence more stable at early iterations.

Bug fixes:

- fixed a fatal error preventing from using BFGS optimization method
- fixed an error in calculating reduced size of cumomer or EMU systems. It did not impact the results (at least for well defined network) but made calculations a little bit longer (reported by Stephane Mottelet, University of Compiegne, France).
- a more explicit error message is generated when a given choice of free fluxes leads to a square but singular flux (stoechiometric) matrix.
- some error messages were printed on standard output instead of `.err` file.

2013-06-28 version 2.5

New features:

- an argument of the option `--np=NP` (number of processes) can be fractional, between 0 and 1 in which case the number of requested cores is calculated as NP*number_of_available_cores
- in documentation, added a section describing some problematic cases and measures which could be undertaken to solve or to work around them. Few more field names in the output file are described (based on discussions with Yanfen Fu, University of Washington, USA)
- missing values in measurements (NA as Non Available) are allowed in FTBL files.

Bug fixes:

- fixed a fatal error if the rights of generated R file cannot be changed
- fixed a bug for `--ln` (least norm) option when without inequalities, increments were not of least norm (reported by Stephane Massou, INSA of Toulouse, France)
- fixed an algorithm used in `--ln`. Now for all inequality systems, both least residual norm and least solution norm are achieved (before, for some systems it was not the case). **Due to this fix, we highly recommend to update to this version if you use `--ln` option**
- fixed a bug in "zero crossing" inequalities. Now, inequalities involving only constrained fluxes are canceled.

2013-04-11 version 2.4

New features:

- number of parallel processes (in case of multiple FTBL files) is limited to a number of cores or to an argument of the `--np` option
- some consistency controls were added on flux names in various FTBL sections.

Bug fixes:

- fixed a bug in formatting some error messages during FTBL parsing;
- fixed an accidental removing of kvh.py file from the previous release;
- fixed non signaling to check `.err` file while some parsing errors did produce;

2013-03-28 version 2.3

New features:

- external `multicore` R package is replaced by native `parallel` package;
- convergence information of Monte-Carlo simulations is reported in the result file;
- relative SD (rsd) in Monte-Carlo statistics is calculated as SD/abs(mean) and no more as SD/abs(estimated parameter);
- if the number of really calculated samples in Monte-Carlo is less than 2, statistics are not calculated;
- R code is self sufficient to be executed via source() function, even in parallel way;
- with a new option `--nocalc`, R code is generated but not executed.

Bug fixes:

- fixed concurrent access to a global variable in Monte-Carlo parallel execution;
- fixed scope issue in Monte-Carlo simulations preventing from update of the current solution;
- fixed some redundant warning messages;
- fixed placement of .err and .log files if FTBL(s) is (are) given with subdirectories in their names.

2013-03-15 version 2.2.1

Bug fixes:

- fixed a fatal error in Jacobian matrix construction when no measured fluxes are provided in FTBL file (reported by Yanfen Fu, University of Washington, USA);
- in the User's manual, added a naming convention for variable growth fluxes.

2013-03-13 version 2.2

New features:

- if more than one FTBL file is given in argument to `influx_s`, all files are proceeded simultaneously in independent processes;
- outliers in measurements can be automatically detected and excluded from parameter fitting.

Bug fixes:

- fixed an error preventing Monte-Carlo results to be written if `multicore` package is not installed;
- fixed a documentation error about $\ln(M)$ in `mynetwork.pres.csv` file;
- fixed warning resuming if there are many of them;
- fixed some error message generation on FTBL parsing.

2013-02-15 version 2.1

New features:

- in nlsic() a new field 'retres' is added to the list of returned values. It transfers "as is" the list returned by a last call to residual calculation function;
- added a writing of generalized inverse of Jacobian to the result file;

Bug fix:

- fixed a typo preventing Monte-Carlo statistics on forward-reverse fluxes to be written in the result file.

2013-02-05 version 2.0

New features:

- metabolite pooling is modeled. Such pooling can appear due to compartmentation phenomenon or due to isomer coelution in chromatography. Starting from this version, metabolite concentrations can be part of fitted parameters;
- adaptive backtracking algorithm is introduced to NLSIC algorithm;
- history of convergence during minimization can be retrieved;
- symbolic equations for dependent fluxes expressed as functions of free and constrained fluxes are generated by ftbl2cumoAb.py script;
- METAB_MEASUREMENTS section is added to FTBL format;
- added χ^2 test for evaluating the goodness of fit;
- removed `metab_scale` field from OPTIONS section in FTBL format;
- "dead end" internal metabolites are allowed in a network without being an output metabolite. As consequence, input-output fluxes must be explicitly declared as non reversible in the FTBL;
- added optional EMU framework (`--emu`);
- added optional series of starting points, fixed or randomly generated (`--fseries`, `--iseries`);
- matrix construction is reworked and fortran code is removed. Now, no more `Rtool` installation is required for running `influx_s`;
- some error messages are made more explicit and more precise;
- outdated R package `fUtilities` is no more required;

Bug fixes:

- fixed stoechiometric matrix construction when for a given metabolite; all fluxes are free or constrained;
- fixed candidate propositions for free fluxes;
- fixed standard deviation value for a `DD/T` field in `PEAK_MEASUREMENTS` section.

2011-10-11 version 1.0

Initial release. Main features:

- NLSIC algorithm;
- FTBL input format from 13CFlux project;
- reduced cumomer set for cumomer balance equations;
- sparse matrices;

- usage of `multicore` R package for Monte-Carlo simulations on Unix platform;
- usable on platforms having Python+numpy and R+some modules;
- command line interface;
- brief user's and programmer's documentation;
- OpenSource (ECL) license.

CHAPTER

THREE

INSTALLATION

To use the software `influx_si`, you'll need some dependencies listed bellow. The software was developed on Linux but can be used both on Linux (or other UNIX, MacOS included) and Windows platforms. If you are not used to install system wide environments like R or Python, ask your local computer support for help. We don't provide support for installation.

Note: The code examples here after are given for Unix shell environment. On windows, in DOS environment the syntax is often similar and in cygwin or Ubuntu environment (Unix tools on Windows) the syntax is identical to the Unix's one.

Dependencies

- R-3.3.0 (cf http://www.r-project.org/ or your system packaging solution) + the following packages.
  - nnls
  - rmumps (5.0.1.12 or higher)
  - arrApply
  - slam
  - limSolve (optional, needed only for `--lim` option)

  To install R modules, as administrator do in R

  ```
  > install.packages(c("nnls", "rmumps", "arrApply", "slam", "limSolve"), dep=TRUE)
  ```

  (you can adapt the package list according to your needs by removing optional packages)

  If you are not an administrator of your R installation, you can execute the command above in your own session and install necessary packages in your own disk space. Other users will have to do the same in their respective sessions if they want to use `influx_si`.
- python 2.6 (or higher but not 3.0 or higher) and modules
  - numpy
  - libsbml (optional, needed for ftbl2metxml.py)
- cytoscape is optional (http://www.cytoscape.org). It can be used to visualize your networks by intermediate of `ftbl2xgmml.py` utility. You can also map flux values returned by `influx_si` on some graphical parameter like edge width for visualizing purposes.

Python and R are advised to be in your PATH variable, in other words, they should be executable from any directory.

Warning: As of this writing (September 17, 2014), an R package `nnls` distributed in precompiled form on Windows platform, can produce wrong results if a 32 bits version is used on Windows 64 bits. To avoid this, use 64 bit version of R on Windows 64 bits or recompile it by hand. To be sure to use 64 bits version of R, check that the `Path` system variable has the R path ending by `\bin\x64` and not just by `\bin`.

Note: On some Python distributions (e.g. Anaconda) on Windows platform, association between `.py` files and Python interpreter is made in incomplete way: the file is executed but command line arguments are not passed to Python. To correct this, a user with administrator privileges has to edit register base with `regedit`. The key `HKEY_CLASSES_ROOT\py_auto_file\shell\open\command` must be changed from

```
"<path_to_your_python_dir>\python.exe" "%1"
```

to

```
"<path_to_your_python_dir>\python.exe" "%1" %*
```

It may happen (depending on your Windows version) that some other keys (related to Python too) have to be modified in similar way.

Compilation dependencies

Starting from version 3.0, some critical parts of code are written in C++ which will require a corresponding compiler installed on your system. It is strongly advised to use the same compiler that was used to compile your R software. You can find which one it was by checking the output of the following shell command

```
$ R CMD config CXX
```

It is likely to be `g++`. A compilation for a given version of `influx_si` will be done automatically only once at the very first execution of `influx_s.py` or `influx_i.py`.

On Linux, all tools necessary for compilation are often available by default. If not, install Linux package (as well as its dependencies) containing `g++` compiler (or what ever was used to compile R).

If you are on Windows platform, you have to install RTOOLS software collection available from https://cran.r-project.org/bin/windows/Rtools/ Be sure to pick up a frozen version that corresponds to your R version. This package will contain the necessary C++ compiler.

If you are on MacOS, your have to install Xcode from AppStore. Furthermore, if some of required R packages are not available in binary form for installation, they will be compiled from sources and this can require additional installation of gfortran-4.8 (or higher).

`influx_si` installation

Unpack the content of `influx_si-vX.Y.zip` (where X.Y is the version number) somewhere on your disk. If you want to make `influx_si` available system wide and install it in a protected directory, you need administrative privileges. Otherwise, `influx_si` will be available only in your personal session.

Add this new directory to your (or system wide) PATH variable (if you don't know what does it mean or how to do it, ask for help from your local computer service). This step is optional but if you don't do it, you need to type all the path to `influx_si` and their utilities every time you run it. It can be as cumbersome as

```
$ /home/joe/soft/bio/flux/influx_s-v2.9/influx_s.py mynetwork.ftbl
```

instead of simple

```
$ influx_s.py mynetwork.ftbl
```

If you want to make `influx_si` available system wide without modifying the PATH variable, add a symbolic link in a directory which is already in PATH. For example, as root you can do

```
$ cd /usr/local/bin
$ ln -s /path/to/dir/of/influx_s/{influx_s.py,influx_i.py,res2ftbl_meas.py,ftbl2cumoAb.py,ftbl2kvh
```

assuming that `/usr/local/bin` is already in the PATH.

First compilation

To accomplish the installation, you have to run `influx_s.py` or `influx_i.py` for the first time as a user having write permissions to the installation directory. I.e. if you have installed `influx_si` as system administrator you have to make a first run also as a system administrator. This first run will compile a shared library `mult_bxxc.so` (a suffix `.so` can be different on your platform) needed for further `influx_si` executions. An example of a command to run is given in the next session "Test of installation".

If in the future, for any reason (upgrading R version, changing the compiler, ...) you have to recompile the shared library, just remove the file `mult_bxxc.so` (or its equivalent if you are not on a Linux platform) and rerun `influx_si` on any FTBL file being a user with write permission on installation directory.

Test of installation

Open a shell window and set your current directory to the `<influx_si_install_dir>/test`. To run `influx_s` you can type

```
$ influx_s.py e_coli.ftbl
```

or

```
$ ../influx_s.py e_coli.ftbl
```

if it is not in the PATH

or drag-and-drop the icon of `e_coli.ftbl` to the icon of `influx_s.py`.

If everything was correctly installed, you should see in your shell window an output looking like:

```
"../influx_s.py" "e_coli.ftbl"
code gen: 2016-07-29 12:06:04
calcul  : 2016-07-29 12:06:04
end     : 2016-07-29 12:06:08
```

The meaning of this output is quit simple. First, an R code is generated from FTBL file then it is executed till it ends. Time moments at which these three events occur are reported.

At the very first execution, a compilation of auxiliary file `mult_bxxc.cpp` will occur which will modify the output in the following manner

```
"../influx_s.py" "e_coli"
code gen: 2016-04-12 10:45:31
calcul  : 2016-04-12 10:45:31
g++ -I/usr/local/src/R-3.2.4/include -DNDEBUG  -I/usr/local/include  -I"/home/local/src/R-3.2.4/li
g++ -shared -L/usr/local/src/R-3.2.4/lib -L/usr/local/lib64 -o sourceCpp_1.so mult_bxxc.o -L/usr/l
end     : 2016-04-12 10:45:44
```

On your system, the compilation commands and paths can differ from this example. That's normal.

The calculation result will be written in `e_coli_res.kvh`. It should be almost identical to the same file in `ok/` subdirectory. On Unix you can do

```
$ diff e_coli_res.kvh ok/e_coli_res.kvh
```

to see if there is any difference. Some small differences in numerical values can be ok. They might come from variations in versions of R and underlying numerical libraries (BLAS, LAPACK and so on).

If something went wrong, check the error messages in `e_coli.err`, interpret them, try to figure out why the errors occurred and correct them.

In high throughput context, you can find useful to run `influx_si` in parallel on many FTBL files. It can be done just by providing more than one FTBL file in argument. For example, with two of FTBLs provided with the package you can run:

```
$ ../influx_s.py e_coli.ftbl e_coli_growth.ftbl
```

In this case, the output looks sightly different than in one by one run:

```
"../influx_s.py" "e_coli.ftbl" "e_coli_growth.ftbl"
e_coli: code gen: 2016-07-29 12:13:32
e_coli_growth: code gen: 2016-07-29 12:13:32
//calcul: 2016-07-29 12:13:32
//end   : 2016-07-29 12:13:36
```

The time moments for code generation is preceded by a short version of FTBL file names. The symbol `//` means parallel proceeding. Parallel calculations are launched after all files are proceeded for the code generation.

It is the operating system that dispatches and equilibrates the charge among available CPUs and cores, not `influx_si` who simply launches these processes.

For a quick test of `influx_i`, you can run in the same directory

```
$ ../influx_i.py e_coli_i
```

Normal output looks like

```
"../influx_i.py" "e_coli_i"
code gen: 2016-04-12 10:43:10
calcul  : 2016-04-12 10:43:10
end     : 2016-04-12 10:43:35
```

Calculation results are written in `e_coli_i_res.kvh` and they can be compared with the same file in the `ok/` sub-directory. You can also visually check a generated graphic file `e_coli_i.pdf` to see if all simulated label kinetics based on estimated fluxes and metabolite concentrations are close to experimental data.

For a quick start guide, launch

```
$ influx_s.py --help
```

or

```
$ influx_i.py --help
```

depending on what context you want to treat: stationary or instationary labeling.

These commands show all available options with a brief description. For more detailed documentation read *User's manual*.

CHAPTER

FOUR

QUICK START

A basic work-flow with `influx_si` is composed of the following steps:

1. Create a FTBL file describing your metabolic reactions, carbon transitions, experimental data and some options. Let call an example file `mynetwork.ftbl`. The FTBL file must follow syntax rules elaborated for 13CFlux software. The FTBL file is a plain text file. The syntax rules will be more or less obvious for someone working on metabolism biochemistry. So, to go quickly, you can inspire from an example file `test/e_coli.ftbl` distributed with the `influx_si` software.

   Note: Staring from the version 2.5, `NA` values (as "Non Available") are admitted as measurements values where appropriate. The difference with FTBL where they are simply omitted is that NA measurements are simulated and are present in the vectors `simulated unscaled labeling measurements` and `simulated scaled labeling measurements` in the result kvh file.

   Note: In case of `influx_i`, label kinetics can be provided in a separate plain text file with values separated by tabulations. First column in this file gives measurement names, and all other columns correspond to a particular time point each. Time points are given on the first line of the file. In this file, there can be more time points than were used in a real experiment for sample harvesting. In this case, the labeling is simulated and reported for these fictitious time points but the least squares fitting is obviously done only at points where real data are reported.

   Empty cells in this file are equivalent to `NA`. Note also that is _not_ necessary to introduce empty columns at regular intervals just to increase the time resolution precision. There is a parameter `nsubdiv_dt` that is designed for this purpose. If it is greater than 1, each time interval defined in the text file is divided in `nsubdiv_dt` sub-intervals.

2. Set your current directory to the directory of `mynetwork.ftbl` and run

   ```
   $ influx_s.py mynetwork
   ```

or:

```
$ influx_i.py mynetwork
```

Depending on stationary or instationary labeling context. Note that the suffix `.ftbl` is optional and `influx_si` installation directory is supposed to be on the PATH.

The `influx_si` run will produce the following files in the same directory that `mynetwok.ftbl`

`mynetwork.log` containing the run-time output from various scripts, in particular, it contains a report on convergence history during the fitting process. It can be helpful for identifying potential problems but if everything is going well, the user does not have to examine the content of this file;

`mynetwork.err` containing the warning and error messages. Normally, this file should be empty (0 byte size);

`mynetwork_res.kvh` containing all of the results. KVH format is a lightweight plain text format for hierarchically structured data. It can be seen in a text editor or in a spreadsheet software as its fields are tab separated. It can also be processed by user's custom software for post-processing,

graphics output and alike. If `influx_si` is run on a series of starting points there will be generated a common result file `mynetwork_res.kvh` containing common information to all starting points but also a series of kvh files, one by starting point, e.g. `mynetwork_res.V1.kvh`, `mynetwork_res.V2.kvh` and so on;

`mynetwork.pres.txt` containing a matrix of fitted parameters and final cost values. Each column corresponds to a particular starting point if run with `--fseries` and /or `--iseries` options. If `influx_si` was run without these options, the file will contain only one column corresponding to the starting point defined in the `mynetwork.ftbl` file.

`edge.netflux.mynetwok`, `edge.xchflux.mynetwok`, `node.log2pool.mynetwork` as the middle name of this files suggest, they can be used to map the corresponding values on the network graph in the cytoscape software.

Note: All these files are silently overwritten if already exist. So take care to copy your results elsewhere if you want to protect them from overwriting.

---

custom files (e.g. `mynetwork.pdf`) These files can be produced by user supplied scripts that are executed at the end of `influx_si` simulations. For example, we provide a script `plot_imass.R` which can be used to plot label kinetics obtained by `influx_i`. One or many of such custom scripts can be given in FTBL file, section OPTIONS, field `posttreat_R` (cf. e_coli_i.ftbl for example)

Note: It can be helpful to do some "dry runs" by executing

```
$ influx_s.py --noopt mynetwork
```

before collecting actual measurement data to see if intended measurements will be sufficient to well define all fluxes or at least the fluxes of interest. It is possible to do because the measurement values in the FTBL file does not matter for flux SD calculation when `--noopt` option is used. So it can be used any values even NA at this moment. In the contrary, `dev` values set in measurement sections of the FTBL file, must be realistic. It is generally not a problem as they express measurements errors and are more or less known for a given measurement chain.

It is worthwhile to stress that a "dry run" is done for some presumed free fluxe values and if they reveal to be very different from actual flux values, it can happen that a network considered as well defined at moment of "dry run" turned into a badly defined network with actual measurement data and corresponding estimated fluxes. So it is important to do his best to guess the most realistic free fluxes for "dry runs".

---

3. See warning and error messages in `mynetwork.err` if any. Correct what has to be corrected and retry p. 2
4. Extract and use the numerical results from the `mynetwork_res.kvh` file.
5. Optionally, visualize net fluxes (or exchange fluxes or logarithm of metabolite concentrations $\log_2(M)$) in cytoscape using `edge.netflux.mynetwok.attrs`, `edge.xchflux.mynetwok.attrs` or `node.log2pool.mynetwork.attrs`.

CHAPTER

FIVE

USER'S MANUAL

Before diving in `influx_si` features let us present FTBL format evolution that was necessary to support `influx_si` innovations.

FTBL format evolution

FTBL format was conceived by authors of `13CFlux` software in late 1990's (cf. https://www.13cflux.net/). At the beginning of 2000's, `13CFlux` became well spread in scientific community working on metabolism and isotope labeling. When we published the first version of `influx_s` in 2011, we adopted FTBL format to avoid cumbersome rewriting of networks and data already in use by the community. Second version of 13CFlux, published in 2012, abandoned FTBL format which was replaced by FluxML (XML) and was accompanied by a tool for automatic conversion of FTBL to FluxML.

On our side, we decided to continue to use FTBL by extending and evolving some of its features. These extensions and evolution are presented hereafter. Version number in titles indicates when presented feature was first introduced to `influx_s(i)`.

`METABOLITE_POOLS` and `METAB_MEASUREMENTS` (v2.0)

Sections `METABOLITE_POOLS` and `METAB_MEASUREMENTS` concerning metabolite pools were added. These sections can be useful for stationary labeling when growth fluxes are modeled with μM terms (cf. Growth flux option) or when some metabolites are confounded in measurements due to cell compartmentation of co-elution during HPLC step or whatever reason. These sections become mandatory for `influx_i` usage for instationary labeling as not only fluxes but also metabolite concentrations impact label propagation dynamics.

`METABOLITE_POOLS` is structured in two columns named `META_NAME` and `META_SIZE` and as ussual for FTBL indented and separated by tabulations, e.g.

```
METABOLITE_POOLS
        META_NAME       META_SIZE
        AKG     -0.5
        ...
```

Note: The value `-0.5` is not aligned with its column name `META_SIZE` because by default, tab characters are expanded to 8 spaces. As `META_NAME` occupies 9 spaces, `META_SIZE` is just shifted to the next tab position. User has to use only one tab character to separate columns even if they don't look aligned on his screen.

For `influx_i`, every internal metabolite (i.e. metabolites present in `NETWORK` section and not being input or output metabolites) and participating in carbon exchange must be referenced in this section. The value given in the column `META_SIZE` is a metabolite concentration. The unit used for these values must be in accordance with the units used for fluxes. For example, if metabolite concentrations are measured in mM/g then fluxes are supposed to be measured in mM/(g*[time_unit]). If the value is positive then corresponding metabolite is considered as having constant concentration which does not vary during fitting iterations. If the value is negative, then this metabolite concentration will be part of fitted variables and its absolute value is used as a starting value for these iterations. A final fitted value will be expressed as a positive number.

For `influx_s`, this section is optional and only few (not all) internal metabolites can be present in this section.

`METAB_MEASUREMENTS` section regroups measurements of internal metabolite concentrations. Input and output metabolites may have concentrations varying during an experiment as they are consumed or produced. So they cannot appear in this section. `METAB_MEASUREMENTS` section has 3 columns: `META_NAME`, `VALUE` and `DEVIATION`, e.g.

```
METAB_MEASUREMENTS
        META_NAME       VALUE   DEVIATION
        Fru6P   0.43    0.01
        ...
```

Column names are self explanatory.

In case of confounded measurements, confounded metabolites can be given as a sum, e.g.

```
METAB_MEASUREMENTS
        META_NAME       VALUE   DEVIATION
        R5P_c+R5P_m     0.32    0.01
        ...
```

In this case, the value `0.32` will be fitted by a sum of simulated metabolite concentrations.

Long reactions (v4.0)

Initially, FTBL admitted no more than 2 metabolites on each side of reactions put in `NETWORK` section. We had to overcome this limit to facilitate FTBL creation for studies including reactions much longer than that. Now, chemical reaction having more than two metabolites on any side can be split in several sub-reactions, each of which has no more then 2 metabolites on every side. It is important that all sub-reactions be put together one after another and that they have the same name. Based on this name, `influx_si` will assemble all parts in one reaction. E.g. a reaction named `Val_syn`

```
Val_syn: Pyr (abc) + Pyr (def) + Glu (ghijk) + NADPH -> Val (abcef) + CO2 (d) + AKG (ghijk)
```

can be translated into FTBL format as

```
NETWORK
       FLUX_NAME       EDUCT_1 EDUCT_2 PRODUCT_1       PRODUCT_2
       Val_syn Pyr     Pyr     Val     CO2
               #abc    #def    #abcef  #d
       Val_syn Glu     NADPH   AKG
               #ghijk  #       #ghijk
```

If some reactions have the same name but not placed sequentially one after another, it will be signaled as an error.

Cofactors (v4.0)

Here, we call cofactors metabolites that does not participate in carbon transfer from one or several molecules to another. The main interest of entering cofactors in carbon transferring reactions is additional balance equations that we can put in stoechiometric system. Thus the number of free fluxes is diminished and fluxes are constrained to more realistic values, not violating cofactor balances.

To indicate that a metabolite is a cofactor, user can simply put an empty carbon string in the corresponding carbon transferring line. For example, a reaction

```
v8: PEP (abc) -> Pyr (abc) + ATP
```

can be translated into FTBL as

```
NETWORK
       FLUX_NAME       EDUCT_1 EDUCT_2 PRODUCT_1       PRODUCT_2
       v8      PEP             Pyr     ATP
               #abc            #abc    #
```

Note an empty carbon string # at the place corresponding to ATP. An important difference between cofactors and other metabolites that the former are allowed to have stoechiometric coefficients different from 1. These coefficients must be separated from cofactors by * sign, e.g. a reaction

```
v41: Asp (abcd) + 2 ATP + NH3 -> Asn (abcd)
```

can be translated into FTBL as

```
NETWORK
       FLUX_NAME       EDUCT_1 EDUCT_2 PRODUCT_1       PRODUCT_2
       v41     Asp     2*ATP   Asn
               #abcd   #       #abcd
       v41     NH3
               #
```

Note the presence of 2*ATP term.

Same metabolite on both sides of reaction (v4.0)

In some particular cases, it can be necessary to have a same metabolite on both sides of reaction. Let us illustrate this situation with the following example:

```
v71: CO2.unlabeled (a) + CO2 (b) -> CO2 (a) + CO2.out (b)
```

Metabolite CO2 is present on both sides of reaction but its carbon atom is not the same. This is the main reason for introducing this feature, to allow tracer rearrangement. In FTBL, it gives

```
NETWORK
       FLUX_NAME       EDUCT_1 EDUCT_2 PRODUCT_1       PRODUCT_2
       v71     CO2.unlabeled   CO2     CO2     CO2.out
               #a      #b      #a      #b
```

Section NOTRACER_NETWORK (v4.0)

In addition to reactions with carbon rearrangements, it can be useful to add reactions with no carbon transfer. The most known reaction of such type is biomass composition but it can there be others, e.g. involving exclusively cofactors:

```
v61: NADH + 0.5 O2 -> 2 ATP
```

This optional section is structured in 2 columns: FLUX_NAME and EQUATION:

```
NOTRACER_NETWORK
       FLUX_NAME       EQUATION
       v61     NADH+0.5*O2 = 2*ATP
```

You can see that the reaction is written in a manner very different form NETWORK section. Its sides are separated by = sign, metabolites are separated by + and they can have stoechiometric coefficients separated by * symbol. It is not visible in this example, but there can be as many metabolites as desired on each side of reaction. The limit "no more than 2 metabolites by side" proper to NETWORK section does not apply here.

Sub-sections EQUALITY/METAB and INEQUALITY/METAB (v2.11)

In the same manner as for fluxes, user can have to constrain variable metabolite concentrations. Constraints can be by equalities and inequalities. These subsections are organized in the same way as for fluxes. In EQUALITY/METAB there are 2 columns VALUE and FORMULA while in INEQUALITY/METAB there are 3 of them: VALUE, COMP and FORMULA. For example,

```
EQUALITIES
       METAB
                VALUE   FORMULA
                0       R5P - 1.5*X5P
                ...
INEQUALITIES
       METAB
                VALUE   COMP    FORMULA
                0.001   <=      PEP
                10      >=      PEP
                ...
```

NA in measurements (v2.5)

Missing values marked as NA are admitted in measurement sections, in columns designated to values. In contrast, they are not admitted in columns designated to standard deviations. The main difference between a measurement just omitted and those marked as NA is that the latter will be simulated and reported in corresponding simulation sections of the result file. This feature can be useful for preliminary simulations when there is no yet data available but user want to know e.g. if fluxes of interest will be well determined or not based on a supposed set of measurements. In this case, all presumed data can be set to NA (but not their SD).

Convention evolution

Not only FTBL format evolved but also some conventions between its parts and content did so. Here is a complete list of the

- user must explicitly declare input-output fluxes as non reversible (set them as C with a value 0 in the section FLUX/XCH) to make a distinction between input-output metabolites and "dead-end" metabolites (the latter are allowed since the version 2.0 and have net flux equal to 0 while exchange flux non zero).
- starting from the version 2.8, new fluxes (i.e. absent in the NETWORK section) may appear in EQUALITY section. They can come, for example, from stoechiometry on cofactors involving non carbon carrying fluxes. These new fluxes have still to be declared in FLUX/{NET,XCH} sections (even if this feature is maintained in v4.0 its interest has diminished since cofactors can now be directly introduced in NETWORK and NOTRACER_NETWORK sections);
- **in LABEL_INPUT section following conventions apply since v3.2:**
  - *"the rest is unlabeled"*: if many labeling forms are lacking in the file (including fully unlabeled metabolite) and the present forms does not sum up to 1, then the fully unlabeled form is considered as completing the set to 1;
  - *"guess the lacking one"*: if only one form is lacking in the file (no matter which one), then its fractions is considered as completing the present set to 1.
- starting from v4.2, a particular comment tag //## is used to introduce a pathway name. The information on pathways can be useful for visualization on a partner web site MetExplore (cf. ftbl2metxml in Additional tools section).

Basic influx_si usage

influx_si can be run without any option on most common cases. So its usage can be as simple as

```
$ influx_s.py mynetwork
```

or

```
$ influx_i.py mynetwork
```

we suppose here that a valid FTBL file `mynetwork.ftbl` was created. Moreover, we supposed `influx_s.py` and `influx_i.py` is in the PATH variable.

In the rest of this manual, we'll use just `influx_s.py` as example if the example is valid for both stationary and instationary contexts. If some usage is valid exclusively for `influx_i.py`, it will be duly signaled.

In a high throughput context, it can be useful to proceed many FTBL files in parallel. This can be done by giving all the FTBL names in a command line, e.g.

```
$ influx_s.py mynetwork1 mynetwork2
```

and so on. All files are then proceeded in separate independent processes launched almost simultaneously by a bunch of size equal to the number of available or requested cores (if an option `--np=NP` is used). It is an operating system who is in charge to make a distribution of all these processes among all available CPUs and cores.

Sometimes, particular cases need usage of special options of `influx_si`. The list of available options can be seen by running:

```
$ influx_s.py --help
```

If used with options, `influx_si` can be run like

```
$ influx_s.py [options] mynetwork
```

where `[options]` is an option list separated by a white character. Each option starts with a double dash `--` and can be followed by its argument if applicable. For example, to use BFGS optimization method instead of the default NLSIC algorithm, a user can run:

```
$ influx_s.py --meth BFGS mynetwork
```

or

```
$ influx_s.py --meth=BFGS mynetwork
```

The option names can be shortened till a non ambiguous interpretation is possible, e.g in the previous example, the option could be shortened as `--m BFGS` or `--m=BFGS` because there is no other option name starting by a letter `m`. But an option `--no` could not be distinguished between `--noopt` and `--noscale`. So at least `--nos` (for `--noscale`) or `--noo` (for `--noopt`) should be provided. There is only one option that does not admit a usage of an equal sign to provide an argument, it is `--excl_outliers`. Use only a space character to provide an argument to this option when required.

Here after the available options with their full names are enumerated and detailed.

`influx_si` command line options

--version show program's version number and exit

-h, --help show the help message and exit

--noopt no optimization, just use free fluxes as is (after a projection on feasibility domain), to calculate dependent fluxes, cumomers, stats and so on

--noscale no scaling factors to optimize => all scaling factors are assumed to be 1

This option can be useful if your measurements are already scaled to sum up to 1 which is often the case of MS data. Then, user saves some free parameters corresponding to scaling factors. This option can become mandatory if user wants to prevent scaling factors to be adjusted by optimization process.

--meth=METH method for optimization, one of nlsic|BFGS|Nelder-Mead. Default: nlsic

--fullsys calculate all cumomer set (not just the reduced one necessary to simulate measurements)

This option influences only post-optimization treatment. The fitting itself is still done with the reduced cumomer set or EMU variables if requested so. See the original paper on `influx_s` for more information on the reduced cumomer set.

--emu simulate labeling in EMU approach

This option should not produce a different result in parameter fitting. It is implemented and provided in a hope that on some network the results can be obtained in a shorter time

--irand ignore initial approximation for free parameters (free fluxes and metabolite concentrations) from the FTBL file or from a dedicated file (cf –fseries and –iseries option) and use random values drawn uniformly from [0,1]

It is recommended to use this option in conjunction with "–zc 0" option.

--sens=SENS sensitivity method: SENS can be 'mc[=N]', mc stands for Monte-Carlo. N is the number of Monte-Carlo simulations. Default for N: 10

The sensitivity information (i.e. the influence of the noise in the data on the estimated parameter variation) based on linearized statistics is always provided. So the user has to use this option only if he wants to compare this linearized information to the Monte-Carlo simulations. Note that the default value 10 for the number of simulations is far from to be sufficient to get reliable statistical estimations. This default option allows only to quickly check that this option is working as expected.

--cupx=CUPX upper limit for reverse fluxes. Must be in interval [0, 1]. Default: 0.999

--cupn=CUPN upper limit for net fluxes. Default: 1.e3

--cupp=CUPP upper limit for metabolite pool. Default: 1.e5

--clownr=CLOWNR lower limit for not reversible free and dependent fluxes. Zero value (default) means no lower limit

A byproduct of this option is that it can drastically reduce cumomer system sizes. As it ensures that non reversible fluxes cannot change the sign, revers fluxes can be eliminated from pathways leading to observable cumomers.

--cinout=CINOUT lower limit for input/output free and dependent fluxes. Must be non negative. Default: 0

--clowp=CLOWP lower limit for free metabolite pools. Must be positive. Default 1.e-8

--np=NP When integer >= 1, it is a number of parallel threads (on Unix) or subprocesses (on Windows) used in Monte-Carlo (M-C) simulations or for multiple FTBL inputs. When NP is a float number between 0 and 1, it gives a fraction of available cores (rounded to closest integer) to be used. Without this option or for NP=0, all available cores in a given node are used for M-C simulations.

--ln Least norm solution is used for increments during the non-linear iterations when Jacobian is rank deficient

Jacobian can become rank deficient if provided data are not sufficient to resolve all free fluxes. It can be useful to determine fluxes that can still be resolved by the available measurements. If the Jacobian does not become rank deficient, this option has no influence on the found solution neither on the optimization process. But if the Jacobian does become rank deficient, a warning message is printed in the error file even if the optimization process could go to the end.

Note: Use this option with caution, in particular, when used in conjunction with Monte-Carlo simulations. As undetermined fluxes will be given some particular value, this value can be more or less stable from one Monte-Carlo simulation to another. This can create an illusion that a flux is well determined. See the linearized statistics in the result file to decide which fluxes are badly resolved.

A correct way to deal with badly defined metabolic network is to provide additional data that can help to resolve all the fluxes and/or to optimize input label, not just put `--ln` option and cross the fingers.

> **Warning:** In this option, the notion of "least norm" is applied to *increments* during the optimization, not to the final solution. So undetermined fluxes could vary from one run to another if the optimization process is started from different points while well determined fluxes should keep stable values.

--sln Least norm of the solution of linearized problem (and not just of increments) is used when Jacobian is rank deficient

--tikhreg Approximate least norm solution is used for increments during the non-linear iterations when Jacobian is rank deficient

To obtain an approximate solution a Tikhonov regularization is used when solving an LSI problem. Only one of the options `--ln` and `--tikhreg` can be activated in a given run.

--lim The same as –ln but with a function limSolve::lsei()

--zc=ZC Apply zero crossing strategy with non negative threshold for net fluxes

This option can accelerate convergence in situations when a net flux has to change its sign during the optimization iterations. Once such flux is identified, it is better to write the corresponding reaction in an opposite sens in the FTBL file or to give a starting value with a correct sign to avoid such zero crossing situation.

--ffguess Don't use free/dependent flux definitions from FTBL file(s). Make an automatic guess.

The fact that free fluxes are chosen automatically does not allow to specify a starting point for optimization iterations so a random starting point is used (drawn uniformly in [0; 1] interval). An option `--seed` can be useful to make the results reproducible.

--fseries=FSERIES File name with free parameter values for multiple starting points. Default: '' (empty, i.e. only one starting point from the FTBL file is used)

The file must be formatted as plain text file with tab separator. There must be as many columns as starting points and at least as many rows as free parameters assigned in this file. A subset of free parameters can be used in this file. In this case, the rest of parameters take their unique starting values from the FTBL file. The first column must contain the names of free parameters used in this file. If there are extra rows whose names are not in the set of free parameter names, they are simply ignored. The first row must contain the names of starting points. These names can be just numbers from 1 to the number of starting points.

--iseries=ISERIES Indexes of starting points to use. Format: '1:10' – use only first ten starting points; '1,3' – use the first and third starting points; '1:10,15,91:100' – a mix of both formats is allowed. Default '' (empty, i.e. all provided starting points are used)

When used with conjunction with `--fseries`, this option indicates the starting points to use from FSERIES file. But this option can also be used in conjunction with `--irand` to generate a required number of random starting points, e.g. `influx_s.py --irand --iseries 1:10 mynetwork` will generate and use 10 random starting points.

For both `--fseries` and `--iseries`, one result file is generated per starting point, e.g. `mynetwork_res.V1.kvh`, `mynetwork_res.V2.kvh` and so on. If starting points comes from a `--fseries` then the suffixes `V1`, `V2`, ... are replaced by the column names from this file. In addition, a file `mynetwork.pres.csv` resuming all estimated parameters and final cost values is written.

--seed=SEED Integer (preferably a prime integer) used for reproducible random number generating. It makes reproducible random starting points (–irand) but also Monte-Carlo simulations for sensitivity analysis. Default: none, i.e. current system value is used, so random drawing will be varying at each run.

--excl_outliers This option takes an optional argument, a p-value between 0 and 1 which is used to filter out measurement outliers. The filtering is based on Z statistics calculated on reduced residual distribution. Default: 0.01.

Excluded outliers (if any) and their residual values are reported in the `mytework.log` file. Non available (`NA`) measurements are considered as outliers for any p-value. An optional p-value used here does not give a proportion of residuals that will be excluded from optimization process but rather a degree of beeing a valuable measurements. So, closer to zero is the p-value, the less data is filtered out. If in contary, you want to filter out more outliers than with the default p-value, use a value grater than the default value of 0.01, e.g.:

```
influx_s.py --excl_outliers 0.02 mynetwork.ftbl
```

Note: Don't use an equal sign "=" to give a p-value to this option. Here, only a white space can be used as a separator (see the example above).

--nocalc	generate an R code but not execute it. This option can be useful for parallel execution of the generated R files via `source()` function in cluster environment
--addnoise	Add centered gaussian noise to simulated measurements written to _res.kvh file. SD of this noise is taken from FTBL file This option can be helpful for generating synthetic FTBL files with realistic simulated measurements (cf. *How to make FTBL file with synthetic data?*).
--TIMEIT	developer option Some portions of code are timed and the results is printed in the log-file. A curious user can use this option without any harm.
--prof	developer option This option provides much more detailed profiling of the execution than `--TIMEIT` option. Only developers can be interested in using such information.

All command line options can be also provided in the FTBL file. A user can put them in the field `commandArgs` in the `OPTIONS` section. The corresponding portion of the FTBL file could look like

```
OPTIONS
       OPT_NAME        OPT_VALUE
       commandArgs     --meth BFGS --sens mc=100 --np 1
```

In such a way, a user can just drag-and-drop an FTBL file icon on the icon of the `influx_s.py` and the calculations will be done with the necessary options, assuming that the system was configured in appropriate way during the installation process.

If an option is provided both on the command line and in the FTBL file, it is the command line that has the priority. In such a way, a user is given an opportunity to overwrite any option at the run time. Nevertheless, there is no way to cancel a flag option (an option without argument) on a command line if it is already set in the FTBL file. For example, if `--fullsys` flag is set in the FTBL file, the full system information will be produced whatever command line options are.

Parallel experiments

Staring from v4.0, `influx_si` offers possibility to treat labeling data from parallel experiments. Parallel experiments for stationary labeling were described in the literature (e.g. cf. "Parallel labeling experiments and metabolic flux analysis: Past, present and future methodologies.", Crown SB, Antoniewicz MR., *Metab Eng.* 2013 Mar;16:21-32. doi: 10.1016/j.ymben.2012.11.010). But for instationary labeling, at the best of our knowledge, `influx_si` is the first software offering parallel experiments treatment.

The main interest of parallel experiments is increased precision of flux estimations. This comes at price of additional work for experiments and data gathering but the result is often worth the effort. As usual, before doing a real "wet" experiment, it can be useful to run few "dry" simulations to see if planned experiments will deliver desired precision.

To deal with parallel experiments, a user have to prepare a series of FTBL files, one per experiment. One of them will be referred to as a main file. It has to provide the following sections common to all experiments: `NETWORK`, `FLUXES`, `EQUALITIES` (if any), `INEQUALITIES` (if any), `FLUX_MEASUREMENTS` (if any), `METABOLITE_POOLS` (if any), `METAB_MEASUREMENTS` (if any) and some entries in `OPTIONS`.

The secondary FTBL files as well as the main one are to provide experimental labeling data corresponding to each experiment. These data have to be presented in the following sections: `LABEL_INPUT`, `LABEL_MEASUREMENTS` (if any), `PEAK_MEASUREMENTS` (if any), `MASS_SPECTROMETRY` (if any). In instationary context, text files with labeling kinetics have to be provided, one per experiment. Their names have to be placed in the field `OPTION/file_labcin` of a corresponding FTBL. Finally, the names of secondary FTBL

files have to be put in the field `OPTIONS/prl_exp` of the main file as plain list separated by semicolon `;` and optionally by one or more spaces.

This file architecture ensures that a network topology, flux and metabolite values are common to all experiments while entry label and measurements on labeled metabolites are proper to each experiment.

Secondary FTBL files can also contain `NETWORK` and other sections found in the main file but are simply ignored at processing step. When FTBL files are ready, you can run `influx_si` on them by providing the name of main FTBL on the command line (and only it, don't list secondary files), e.g. in installation directory run:

```
$ ./influx_s.py test/prl_exp/e_coli_glc1-6n
```

You can find an example of parallel experiment data in the directory `test/prl_exp` in files e_coli_glc1-6n.ftbl (main file), e_coli_glc2n.ftbl, e_coli_glc3n.ftbl, e_coli_glc4n.ftbl, e_coli_glc5n.ftbl, e_coli_glc6n.ftbl. These files correspond to stationary labeling experiments described in "Complete-MFA: Complementary parallel labeling experiments technique for metabolic flux analysis", Robert W. Leighty, Maciek R. Antoniewicz, *Metabolic Engineering* 20 (2013) 49–55 (with only difference that we use simulated and noised data instead of measured ones).

We also provide an example of simulated instationary parallel experiments in the files `e_coli_GX_prl.ftbl` (main file) and `e_coli_GX_X.ftbl` (secondary file) corresponding to simultaneous consumption of glucose and xylose. The network for this simulations was borrowed from "13C metabolic flux analysis of microbial and mammalian systems is enhanced with GC-MS measurements of glycogen and RNA labeling", Christopher P. Long, Jennifer Au, Jacqueline E. Gonzalez, Maciek R. Antoniewicz, Metabolic Engineering 38 (2016) 65–72. The experiment consisted in dynamic labeling by uniformly labeled glucose (main experiment) and by uniformly labeled xylose (secondary one). Labeling kinetics MS data are given in `e_coli_GX_MS.txt` and `e_coli_GX_X_MS.txt` files respectively. To play with this example, you can run (still in installation directory):

```
$ ./influx_i.py test/prl_exp/e_coli_GX_prl
```

The secondary files in all examples contain also the full information about the network, fluxes and so on, so they can be used as classical mono-experimental files to see how much the precision of flux estimation increased due to parallel experiment methodology.

Note that set of measured metabolite fragments as well as sampling time points for instationary labeling are not necessary the same for all parallel experiments. They do can differ.

Optimization options

These options can help to tune the convergence process of the NLSIC (or any other chosen algorithm). They can be given only in an FTBL file, in the section OPTIONS. These options are prefixed with `optctrl_` which is followed by a particular option name. For example, `optctrl_errx` corresponds to the stopping criterion hereafter and the corresponding FTBL portion could look like

```
OPTIONS
       OPT_NAME         OPT_VALUE
       optctrl_errx     1.e-3
```

All possible options and their default values for NLSIC algorithm follow:

errx=1.e-5 stopping criterion. When the L2 norm of the increment vector of free parameters is below this value, the iterations are stopped.

maxit=50 maximal number for non-linear iterations.

btstart=1. backtracking starting coefficient

btfrac=0.25 backtracking fraction parameter. It corresponds to the alpha parameter in the paper on `influx_s`

btdesc=0.1 backtracking descending parameter. It corresponds to the beta parameter in the paper on `influx_s`

btmaxit=15 maximal number of backtracking iterations

trace=1 report (=1) or not (=0) minimal convergence information

rcond=1.e10 condition number over which a matrix is considered as rank deficient

ci=list(p=0.95, report=F) confidence interval reporting. This option is own to `nlsic()` function. It has no impact on the reporting of linear stats information in the result kvh file after the post-optimization treatment. This latter is always done.

history=FALSE return or not (default) the matrices with optimization steps and residual vectors during optimization. These matrices can then be found as part of `optimization process information/history` field in `mynetwork_res.kvh` file. Use it with caution, big size matrices can be generated requiring much of memory and disk space.

adaptbt=TRUE use (default) or not an adaptive backtracking algorithm.

monotone=FALSE should or not the cost decrease be monotone. If TRUE, then at first non decrease of the cost, the iterations are stopped with a warning message.

Names and default values for BFGS and Nelder-Mead algorithms can be found in the R help on `optim()` function.

Growth flux option

If present, this option makes `influx_si` take into account growth fluxes $-\mu M$ in the flux balance, where μ is a growth rate and M is a concentration of an internal metabolite M by a unit of biomass. Only metabolites for which this concentration is provided in an FTBL section `METABOLITE_POOLS`, contribute to flux balance with a flux $-\mu M$. This flux can be varying or constant during optimization process depending on whether the metabolite M is part of free parameters to fit or not. Usually, taking into account of this kind of flux does not influence very much on the estimated flux values. So, this option is provided to allow a user to be sure that it is true in his own case.

The option is activated by a field `include_growth_flux` in the `OPTIONS` section:

```
OPTIONS
       OPT_NAME        OPT_VALUE
       include_growth_flux     1
```

Value 0 cancels the contribution of the growth fluxes to the general flux balance.

Another necessary option is `mu` giving the value of μ:

```
OPTIONS
       OPT_NAME        OPT_VALUE
       mu      0.12
```

Finally, the metabolite concentrations by a unit of biomass are reported in a section `METABOLITE_POOLS` as:

```
METABOLITE_POOLS
       META_NAME       META_SIZE
       Fum     2.47158569399681
       Suc     -15.8893144279264
       Mal     -6.47828321758155
       ...     ...
```

Metabolite names used in this section must be identical to those used in the `NETWORK` section and others. Negative value is used as indicator of a variable metabolite pool. Such varying metabolites are part of fitted parameters. Absolute values from this section are used as their starting values in the optimization process.

One of valuable originality of `influx_s`, it is a possibility to couple fluxomics and metabolomics in stationary experiments. It can be done because metabolite pools can influence labeling in two ways:

- through metabolite pooling (due to compartmentation and/or coelution during chromatography)

- through growth fluxes.

This last influence is often of low intensity compared to metabolite transformation fluxes. In literature, it is often neglected.

Note: METABOLITE_POOLS section was not present in the original FTBL format. It is added *ad hoc* and it is possible that its presence makes fail other software using such FTBL.

Another section that was added "ad hoc" to FTBL file is METAB_MEASUREMENTS:

```
METAB_MEASUREMENTS
       META_NAME       VALUE   DEVIATION
       Suc     15.8893144279264*1.e-3/10.7     1.e-2
       Mal     6.47828321758155*1.e-3/10.7     1.e-2
       Rub5P+Rib5P+Xul5P       1.66034545348219*1.e-3/10.7     1.e-2
```

Like for other measurements, user has to provide a name, a value and a standard deviation for each entry in this section. Metabolites listed in this section must be defined in the NETWORK section and must have a negative value in the METABOLITE_POOLS section. Numerical values can be simple arithmetic expressions (as in the example above) which are evaluated during file parsing.

When a metabolite name is given as a sum of metabolites (e.g. Rub5P+Rib5P+Xul5P) it is interpreted as a list of metabolites to be pooled. It is done proportionally to their concentrations. No numerical factor can appear in this sum. At least one of the metabolites from the list must have negative value in the METABOLITE_POOLS section. Otherwise, all metabolites from the list would be considered as having a fixed concentration and providing a measurement for such metabolites would be meaningless.

Note: There is no a specific option activating simulation of metabolite concentrations and taking them into account to the fitting process. Their simple presence in the METABOLITE_POOLS and METAB_MEASUREMENTS sections make concerned metabolites fittable parameters.

An example of an FTBL file having metabolite sections and involving growth fluxes can be found in test/e_coli_growth.ftbl.

Post treatment option

User can specify a name of one or several R scripts that will be automatically executed after non aborted influx_si run. This option can be useful, for example, for plain saving of calculation environment in a file for later exploring in an interactive R session or for plotting results in a pdf file and so on. A very basic example of such script is provided in the file test/save_all.R and its use can be found in the options of test/e_coli.ftbl file.

To activate this option, the script names must be provided in the OPTIONS section, in the field posttreat_R and separated by '; ', e.g.

```
OPTIONS
       OPT_NAME        OPT_VALUE
       posttreat_R     save_all.R; plot_something.pdf
```

The script name is interpreted as a relative path to the directory where the original FTBL file is located. After execution of save_all.R, a file e_coli.RData is created. This particular example can be used to restore a calculation R environment by launching R and executing:

```
> load("e_coli.RData")
```

After that, all variables defined in influx_si at the end of the calculations will be available in the current interactive session. To be able to launch custom calculations on these variables, user has to do some preliminary actions. An example of such actions can be found in a file preamble.R which can be adapted for users's case.

To write his own scripts for post treatments or explore the calculated values in an interactive session, a user have to know some basics about existent variables where all the calculation results and auxiliary information are stored. Here are few of them:

dirw is a working directory (where the original FTBL file is)

dirx is an executable directory (where influx_s.py is)

baseshort is a short name of the input FTBL file (without the suffix `.ftbl` neither the directory part of the path)

param is the vector of the estimated parameters composed of free fluxes, scaling parameters (if any) and metabolite concentrations (if any)

jx_f is a environment regrouping calculated quantities. Here are some of its fields:

fallnx a vector of all net and exchange fluxes (here, exchange fluxes are mapped on [0; 1[interval)

fwrv a vector of forward and reverse fluxes (reverse fluxes are "as is", i.e. not mapped)

x is an internal state label vector

simlab, simfmn and simpool are vectors of simulated measurements for label, net flux and metabolite pools respectively (fitting at the best of influx_s' capacity the provided measurements in the FTBL file)

res is the reduced residual vector, i.e. (simulated-measured)/SD

ures is the unreduced residual vector, i.e. (simulated-measured)

jacobian as its names indicates, is the Jacobian matrix (d res/d param)

udr_dp is the jacobian matrix for the unreduced residual vector (d ures/d param)

measurements is a list regrouping various measurements and their SD

nb_f is a list of various counts, like number of fluxes, parameters to fit, system sizes and so on

nm_list is a list of names for various vectors like fluxes, metabolites, label vectors, measurements, inequalities and so on

ui, ci are inequality matrix and right hand side respectively

A full list of all available variable and functions can be obtained in an R session by executing:

```
> ls()
```

This list of more than 400 items is too long to be fully described here. We hope that few items succinctly described in this section will be sufficient for basic custom treatments.

An inspirations for your own custom treatments and/or plotting can be found in files `plot_imass.R` and `plot_smeas.R` that plot instationary and stationary data respectively in pdf files.

Exclusive `influx_i` options

There is only one exclusive option that can be given on a command line:

--time_order=TIME_ORDER Time order for ODE solving (1 (default), 2 or 1,2). Order 2 is more precise but more time consuming. The value '1,2' makes to start solving the ODE with the first order scheme then continues with the order 2.

The scheme order can be important for the precision of flux and concentration estimations. The impact is not direct but can be very important. Please note that it can happen that order 1 fits the data with lower cost value function but it does not mean that the fluxes/concentrations are better estimated.

Other options occur as fields in the section `OPTIONS` of the FTBL file.

file_labcin gives the name of the text file with label kinetics. If the file name starts with a "/", it is considered as

The values must be organized in a matrix where each row corresponds to a measured isotopomer/cumomer/mass-isotopologue while each column corresponds to a given time point. First column gives the names of labeled measured species and the first row contains time points.

Matrix must be written one row per line and its entries (cells) must be separated by tabulations. Missing data can be signaled as `NA` or just an empty cell. Comments are allowed and must start with `#` sign. The rest of the line after `#` is simply ignored. Empty lines are ignored. In such a way, comments can help to annotate the data and empty lines can help to format the file for better human readability. All lines (a part from blank lines and comments) must have the same number of cells.

The specie names must fit the names used in corresponding measurement sections of FTBL file. For example, a name `m:Rib5P:1,2,3,4,5:0:693` is composed of several fields separated by a column `:`

m indicates that data are of `MASS_SPECTROMETRY` type. Other possible values are `l` for `LABEL_MEASUREMENTS` and `p` for `PEAK_MEASUREMENTS`

Rib5P metabolite name

1,2,3,4,5 carbon numbers present in the measured fragment

0 mass shift relative to fully unlabeled mass isotopologue: `0` corresponds to a fraction of unlabeled fragment, `1` to a fraction of fragments with only one labeled carbon atom and so on

693 line number in FTBL file corresponding to this measurement. If previous fields are sufficient to unambiguously identify the measurement, this field can be omitted.

Cf. `test/e_coli_msne.txt` (and corresponding `test/e_coli_i.ftbl`) for more examples.

The measurement precision (SD) is considered as constant during time and its values (one per measured specie) is given in the FTBL file, in the corresponding measurement section.

All time points must be positive and put in increasing order. The time point 0 must be absent and is considered as labeling start. At that point all species are supposed to be fully unlabeled. This means also that all label measurements must be provided with a correction for natural 13C labeling. To prepare MS data with such correction, a software IsoCor can help.

There can be fictitious time points without any data in them. This feature can be used to increase the time resolution at some time intervals. The simulation of label propagation will be done and reported at these fictitious time points but the fitting will be obviously done only at time points having real data in them. For a regular time interval sub-division, it is more practical to use a parameter `nsubdiv_dt` (cf. hereafter) instead of fictitious time point in this file.

If this field is empty or absent in the FTBL file then no fit can be done and a simple label simulation is calculated as if `--noopt` option were activated. Such simulation can be done only if a time grid is defined with the help of two other parameters: `dt` and `tmax` (cf. hereafter).

nsubdiv_dt integer number of sub-intervals by which every time interval is divided to increase the precision of time resolution.

It can happen that the value 1 (default) is sufficient for a satisfactory flux/concentration estimation. User can gradually increase this value (2, 3, ...) in successive `influx_i` runs to be sure that better time resolution does not impact parameter estimation. This property is called *grid convergence*. A grid convergence is necessary to overcome the result dependency on the choice of a numerical discretization scheme. A grid convergence can be considered as achieved when changes in estimated parameters provoked by a grid refinement are significantly lower than estimated confidence intervals for these parameters.

dt a real positive number, defines a time step in a regular grid in absence of a file in `file_labcin` field. If a file with label kinetics is well present then this parameter has no effect.

A regular time grid for label simulations can be useful on preliminary stage when user only elaborates FTBL file and wants to see if label simulation are plausible. It can also help to produce simulated measurements (which can be extracted from the `_res.kvh` file) for further numerical experiments like studying convergence speed, parameter identifiability, noise impact and so on.

`tmax` a real positive number, defines the end of a regular time grid if the field `file_labcin` is empty or absent. Parameters `dt` and `tmax` must be defined in such a way that there will be at least 2 time points greater then 0 in the time grid.

If a file with label kinetics is well present then this parameter can be used to limit time grid on which simulations are done. If the value in `tmax` is greater then the maximal time value defined in the kinetics file then this parameter has no effect.

Note: It is very important that the values for time, flux and metabolite concentrations be expressed in concordant units. It would be meaningless to give time in minutes, fluxes in mM/h/g and concentrations in mM. This will lead to wrong results.

For example, if the time is expressed in seconds and concentrations in mM/g then fluxes must be expressed in mM/s/g.

Note: Option `--noscale` must be always activated for instationary calculations. So that for example, MS measurements must be always composed of fully measured fragments (i.e. with all isotopologues present) and normalized to sum up to 1.

Result file fields

Generally speaking, the names of the fields in the result KVH file are chosen to be self explanatory. So there is no so much to say about them. Here, we provide only some key fields and name conventions used in the result file.

At the beginning of the `mynetwork_res.kvh` file, some system information is provided. Here "system" should be taken in two sens: informatics and biological. The information is reported in the fields `influx` and `system sizes`. These fields are followed by `starting point` information regrouping `starting free parameters`, `starting cost value`, `flux system (Afl)` and `flux system (bfl)`. Name conventions used in these and other fields are following:

net and exchange fluxes are prefixed by `n.` or `x.` respectively

free, dependent, constrained and variable growth fluxes are prefixed by `f.`, `d.`, `c.` and `g.` respectively. So, a complete flux name could look like `f.n.zwf` which means *free net ZWF flux*. Growth fluxes which depend on constant metabolite concentrations can be found in constrained fluxes. Constant or variable growth fluxes are postfixed with `_gr` (as *growth*) string. For example, a flux `g.n.Cit_gr` corresponds to a net growth flux of Citrate metabolite. The growth fluxes are all set as non reversible, so all exchange fluxes like `g.x.M_gr` or `c.x.M_gr` are set to 0.

scaling factors names are formed according to a pattern similar to `label;Ala;1` which corresponds to the first group of measurements on Alanine molecule in labeling experiments. Other possible types of experiments are `peak` and `mass`.

MID vector names are looking like `METAB+N` where `METAB` is metabolite name and `N` goes from 0 to the number of carbon atoms in the considered molecule.

cumomer names follow classical convention `METAB#pattern_of_x_and_1`, e.g. `Ala#x1x`

forward and reverse fluxes are prefixed by `fwd.` and `rev.` respectively, e.g. `fwd.zwf` or `rev.zwf`

measurement names have several fields separated by a colon `:`. For example, `l:Asp:#xx1x:694` deciphers like:

- `l` stands for *labeling* experiment (others possibilities are `p` for *peak*, `m` for *mass* and `pm` for *metabolite pool*)
- `Asp` is a metabolite name
- `#xx1x` is a measurement identification
- `694` is a line number in the FTBL file corresponding to this measurement.

The field `optimization process information` is the key field presenting the results of an optimization process. The fitted parameters are in the subfield `par`. Other subfields provide some additional information.

The final cost value is in the field `final cost`.

The values of vectors derived from free fluxes like dependent fluxes, cumomers, MID and so on are in the corresponding fields whose names can be easily recognized.

Linear stats and Monte-Carlo statistics are presented in their respective fields. The latter field is present only if explicitly requested by user with `--sens mc=MC` option. In this kvh section, a term `rsd` means "relative standard deviation" (in literature, it is often encountered a synonym CV as Coefficient of Variation), it is calculated as SD/Mean and if expressed in percentage then the formula becomes 100%*SD/Mean.

The field `jacobian dr_dp (without 1/sd_exp)` report a Jacobian matrix which is defined as a matrix of partial derivatives $\partial r/\partial p$ where r is residual vector (Simulated–Measured) and p is a free parameter vector including free fluxes, scaling factors (if any) and free metabolite pools (if any). Note that in this definition the residual vector is not yet scaled by standard deviation of measurements. Sometimes, Jacobian is called *sensitivity matrix* in which case a special care should be brought to the sens of derivation. Often, by sensitivity matrix, we intend a matrix expressing how estimated fluxes are sensible to variations in the measurement data. Such definition corresponds to generalized inverse of Jacobian and it is reported in the field `generalized inverse of jacobian dr_dp (without 1/sd_exp)`

Network values for Cytoscape

Several network values formatted for cytoscape are written by `influx_si` to their respective files. It can facilitate their visualizing and presentation in graphical mode. All these values can be mapped on various graphical attributes like edge width, node size or color scale of any of them. All these files are written at the end of calculations so if an error has interrupted this process, no such file will be produced. Take care to don't use an outdated copy of these files.

A file named `edge.netflux.mynetwork.attrs` can help to map net flux values on edges of a studied network. A file `edge.xchflux.mynetwork.attrs` do the same with exchange fluxes. And finally, `node.log2pool.mynetwork.attrs` provides logarithm (base 2) of pool concentrations. They can be mapped on some graphical attribute of network nodes.

See Additional tools section, ftbl2xgmml: cytoscape view paragraph to know how to produce files importable in Cytoscape from a given FTBL file. User's manual of Cytoscape has necessary information about using visual mapper for teaching how some values like net flux values can be mapped on graphical elements like edge width and so on.

Warning and error messages

The warning and error messages are logged in the `.err` suffixed file. For example, after running:

```
$ influx_s mynetwok
```

the warnings and errors will be written in the `mynetwork.err` file. This kind of messages are important for user not only to be aware that during calculations something went wrong but also to understand what exactly went wrong and to have an insight on how to fix it.

Problems can appear in all stages of a software run:

- parsing FTBL files

- R code writing
- R code execution
  - vector-matrix initialization
  - optimization
  - post-optimization treatment

Most of the error messages are automatically generated by underlying languages Python and R. These messages can appear somewhat cryptic for a user unfamiliar with these languages. But the most important error messages are edited to be as explicit as possible. For example, a message telling that free fluxes are badly chosen could look like:

```
Error : Flux matrix is not square or singular: (56eq x 57unk)
You have to change your choice of free fluxes in the 'mynetwork.ftbl' file.
Candidate(s) for free flux(es):
d.n.Xylupt_U
```

a message about badly structurally defined network could be similar to

```
Error : Provided measurements (isotopomers and fluxes) are not
        sufficient to resolve all free fluxes.
Unsolvable fluxes may be:
        f.x.tk2, f.n.Xylupt_1, f.x.maldh, f.x.pfk, f.x.ta, f.x.tk1
Jacobian dr_dff is dumped in dbg_dr_dff_singular.txt
```

a message about singular cumomer balance matrix could resemble to

```
lab_sim: Cumomer matrix is singular. Try '--clownr N' or/and '--zc N' options with small N, say 1
cit_c:16
ac_c:2
...
Zero fluxes are:
fwd.ACITL
...
```

Note: In this error message, we report cumomers whose balance gave a zero row in the cumomer matrix (here `cit_c:<N>` cumomers, where <N> is an integer, its binary mask indicates the "1"s in the cumomer definition) as well as a list of fluxes having 0 value. This information could help a user to get insight about a flux whose zero value led to a singular matrix. A workaround for such situation could be setting in the FTBL file an inequality constraining a faulty flux to keep a small non zero value. A more radical workaround could be restricting some flux classes (input-output fluxes with the option `--cinout=CINOUT` or even all non reversible ones with the option `--clownr=CLOWNR`) to stay out of 0, e.g.:

```
$ influx_s.py --clownr 0.0001 mynetwork
```

Adding such inequalities does not guaranty that cumomer matrix will become invertible but often it does help. It's up to user to check that an addition of such inequalities does not contradict biological sens of his network.

---

a message about badly statistically defined network could appear like

```
Inverse of covariance matrix is numerically singular.
Statistically undefined parameter(s) seems to be:
f.x.pyk
For more complete list, see sd columns in '/linear stats'
in the result file.
```

and so on.

A user should examine carefully any warning/error message and start to fix the problems by the first one in the list (if there are many) and not by the easiest or the most obvious to resolve. After fixing the first problem, rerun `influx_si` to see if other problems are still here. Sometimes, a problem can induce several others. So, fixing the first problem could eliminate some others. Repeat this process, till all the troubles are eliminated.

Problematic cases

Obviously, everyone would like be able just run a flux estimation software and simply get results but unfortunately it does not work in this way every time. In this section we review some problematic cases which can be encountered in practice.

Structurally non identifiable fluxes

It can happen that collected data are not sufficient to resolve some fluxes in your network. Due to non linear nature of the problem, this situation can appear for some set of free flux values and disappear for others or be persistent for any free flux values. An error is reported to signal such situation, e.g.

```
lsi: Rank deficient matrix in least squares
1 unsolvable variable(s):
f.n.PPDK        7
```

and execution is stopped.

Several options are then available for a user facing such situation.

1. Collect more data to resolve lacking fluxes. As a rule of thumb, data must be collected on metabolites which are node of convergence of badly defined fluxes or on metabolites situated downhill of convergence point and preserving labeling pattern. Nature of collected data can be also important. Examples can be constructed where mass data are not sufficient to determine a flux but RMN data can do the job.

   Before using real data collection, you can make a "dry run" with `--noopt` option and with fictitious values for intended metabolite in the FTBL file to see if with these new data, the network becomes well resolved. If the error message disappear and SD values in the section `linear stats` are not very high then chances are that additionally collected data can help to resolve the fluxes.

2. Optimize input label. It can happen that you do collect data on a metabolite situated in convergence point for undefined fluxes but incoming fluxes are bringing the same labeling pattern which prevents flux(es) to be resolved. May be changing substrate label can help in this situation. For label optimization you can use a software called IsoDesign, distributed under OpenSource licence and available here http:://metatoul.insa-toulouse.fr/metasys/software/isodes/ (may be you have received `influx_si` as part of IsoDesign package, in which case you have it already).

   Naturally, this label optimization should be done before doing actual experiments. See IsoDesing tutorial for more details on how to prepare and make such optimization.

   If you don't want or don't have a possibility to use a software for label optimization or you think to have an insight on what should be changed in substrate labeling to better define the fluxes, you can still make a try with `influx_s.py --noopt new_labeling.ftbl` option to see if a new labeling will do the job (here `new_labeling.ftbl` is an example name for a FTBL file that you will prepare with a new `LABEL_INPUT` section.)

3. Use `--ln` option. It wont make you fluxes well defined, it will just continue calculation trying to resolve what can be solved and assigning some particular values (issued from so called *least norm* solution for rank deficient matrices) to undefined fluxes. You will still have a warning similar to

   ```
   lsi_ln: Rank deficient matrix in least squares
   1 free variable(s):
   f.n.PPDK        7
   Least L2-norm solution is provided.
   ```

   informing you that some flux(es) in the network is(are) still undefined. This option can be helpful if undefined fluxes are without particular interest for biological question in hand and their actual values can be safely ignored.

4. You can give an arbitrary fixed value to an undefined flux by declaring it as constrained in the FTBL file (letter `C` in the column `FCD` in the `FLUXES` section).

Badly defined fluxes

Also known as *statistically undefined fluxes*, these fluxes have big or even huge SD values. The difference between these fluxes and structurally undefined fluxes is that the badly defined fluxes can become well defined if the noise is reduced or hypothetically eliminated while the latter will still be undetermined even in the absence of the noise. Despite this difference, all options presented in the previous section are applicable here (all but `--ln` which would be without effect here).

An additional measure can be taken which consist in experimental noise reduction. Generally, it can be done by using better protocols, better instruments or simply by increasing the measurement repetition number.

Once again, a use of `--noopt` with new hoped DEV values in the FTBL file can help to see if these new measurements with better noise characteristics will resolve or not the problem.

Slow convergence

Slow optimization convergence can manifest by following warnings:

```
nlsic: Maximal non linear iteration number is achieved
```

or/and

```
nlsic: Maximal backtrack iteration number is achieved
```

Theoretically, user can increase the limit for those two numbers (`optctrl_maxit` and `optctrl_btmaxit` respectively in the `OPTIONS` section of FTBL file) but generally it is not a good idea. It can help only in very specific situations that we cannot analyze here as we estimate them low probable. In all cases, a slow convergence is due to high non linearity of the solved problem. What can vary from one situation to another, it is the nature of this non linearity. Depending on this nature, several steps can be undertaken to accelerate optimization:

1. If a non linearity causing the slow convergence is due to the use of function absolute value $|x|$ in the calculation of forward and revers fluxes from net and exchange fluxes, then an option `--zc=ZC` (zero crossing) can be very efficient. This non linearity can become harmful when during optimization a net flux has to change its sign, in other words it has to cross zero.

   This option splits the convergence process in two parts. First, a minimum is searched for fluxes under additional constraints to keep the same sign during this step. Second, for fluxes that reached zero after the first step, a sign change is imposed and a second optimization is made with these new constraints. If `--zc` option is used with an argument 0 (`--zc=0` or `--zc 0`), it can happen that fluxes reaching zero produce a singular (non invertible) cumomer balance matrix. In this case, an execution is aborted with an error starting like

   ```
   Cumomer matrix is singular. Try '--clownr N' or/and '--zc N' options with small N, say 1.
   ...
   ```

   To avoid such situation, an argument to `--zc` must be a small positive number, say `--zc 0.001`. In this case, positive net fluxes are kept over 0.001 and negative fluxes are kept under -0.001 value. In this manner, an exact zero is avoided.

   Another way to avoid problem induced by using module function $|x|$ is to add inequality(-ies) imposing sens of reaction in `INEQUALITIES/NET` section, e.g.

   ```
   0.0001 <=      mae
   ```

   Naturally, in this example, you have to be sure that the reaction catalyzed by malic enzyme (here `mae`) must go in the sens written in your FTBL file.

   You can find potential candidates to impose sens of reaction by examining the flux values in `mynetwork_res.kvh` after a slow convergence and looking fluxes who's sign (positive or negative) looks suspicious to you. In our practice, we could observe a dramatic increase in convergence speed and stability just after imposing a sens of reaction to a "key" reaction. Obviously, such constraint must be in accordance with biological sens of a studied network and its biological condition.

2. A high non linearity can appear for some particular set of fluxes, especially when they take extreme values, e.g. when exchange fluxes are close to 1 or net fluxes take very high values of order 10^2 or even 10^3 (supposing that the main entry flux is normalized to 1). In such a case, user can low this limits (options `--cupx=CUPX` and `--cupn=CUPN` respectively) or try to exclude outliers (`--excl_outliers P-VALUE`) as outliers can attract the solution in weird zone of fluxes. In this latter case, the first convergence will continue to be slow and will generate corresponding warnings but the second one (after a possible automatic elimination of outliers) can be much quicker.

Convergence aborted

This situation is signaled by an error message:

```
nlsic: LSI returned not descending direction
```

This problem can occur for badly defined network which are very sensitive to truncation errors. The effect of such errors can become comparable to the effect of the increment step during optimization. It means that we cannot decrease the norm of residual vector under the values resulting from rounding errors. If it happens for relatively small increments then the results of convergence are still exploitable. If not, there is no so many actions that user could undertake except to make his system better defined as described in previous sections.

Note: By default, we use a very small value for increment norm as stopping criterion (10^{-5}). It can be considered as very drastic criterion and can be relaxed to 10^{-3} or 10^{-2} depending on required precision for a problem in hand (to do that, use an option `optctrl_errx` in the section `OPTIONS` of FTBL file).

Additional tools

Tools described in this section are not strictly necessary for running `influx_si` and calculating the fluxes. But in some cases, they can facilitate the task of tracking and solving potential problems in FTBL preparation and usage.

Most of the utilities produce an output written on standard output or in a file who's name is derived from the input file name. This latter situation is signaled with a phrase "The output redirection is optional" and in the usage examples the output redirection is taken in square brackets `[> output.txt]` which obviously should be omitted if an actual redirection is required. Such behavior is particularly useful for drag-and-drop usage.

txt2ftbl: conversion of txt format to FTBL format

An easily readable/writable text format can be used to create *de novo* an FTBL file. Reactions in this text format can look like:

```
v48: Asp (abcd) + Pyr (efg) + Glu (hijkl) + SucCoA (mnop) + ATP + 2 NADPH ->
 LL-DAP (0.5 abcdgfe + 0.5 efgdcba) + AKG (hijkl) + Suc (0.5 mnop + 0.5 ponm)
```

This long reaction illustrates several format features:

- `v48` is the reaction name. It is optional. If reaction names (and theirs separators `:` signs) are omitted, reactions will be just numbered. The numbering restarts after each comment block (a comment starts with a `#` sign). This is done to give an opportunity to organize reactions in pathways. In such a case, a comment is considered as stating a new pathway which is also numbered. Thus an automatic reaction name can look like `r2.3` where `2` is pathway number and `3` is a reaction number in this pathway.

  Note that in this example, the reaction is split in two lines only for convenience of presentation. In a text file, a reaction must be written on only one line. No line breaks are admitted in reactions and no more than one reaction can be written on a given line.

- `Asp (abcd)` is a metabolite name `Asp` followed by its optional carbon id string between parentheses `(abcd)`. All carbon id must be a unique letter on each side of the reaction and if present on one side of reaction, must also be present on the other one. Thus carbon atom balance is preserved. In case of symmetric molecule, the carbon scrambling can be indicated as `(0.5 mnop + 0.5 ponm)` as e.g. for succinate `Suc` in the example above. Numeric coefficients of carbon forms (here `0.5`) can be omitted as all forms are considered as equally probable and automatically normalized to sum up to 1. So a completely equivalent form could be `(mnop + ponm)`.
- `+` sign separates metabolites on each side of reaction
- `->` separates two sides of reaction and indicates that this reaction is irreversible, i.e. its exchange flux is zero. It does not precludes about the sens of reaction. Here we consider that a reaction can be irreversible and have a negative net flux. If in addition, you wish to indicate that a reaction must operate only from left to right, i.e. to have a positive net flux, then use `->>` sign. To indicate a reversible reaction use `<->` and a reversible reaction with imposed positive net flux use `<->>`.
- `ATP` is an example of a cofactor, it does not have a carbon id string. It participates in mass balance but not in carbon balance equations.
- `2 NADPH` is an example of a cofactor with a stoechiometric coefficient different from 1. Coefficients different from 1 are not allowed for metabolites participating in carbon exchanges in a given reaction. But if a reaction has no carbon exchanges, then all metabolites are allowed to have a coefficient different from 1 like for example in biomass reactions.
- `#` starts a comment that will be put in FTBL as is, except the first `#` hash sign that will be replaced by `//` (FTBL comment tag)
- `###` triple hash sign is used to introduce a pathway name. Respectively, `//##` will do the same in FTBL. Pathway name can be useful for `ftbl2metxml.py` script which prepare xml and txt files for visualization on a partner site MetExplore.

An example of a full featured metabolite network can be found in `test/prl_exp/e_coli_anto.txt`.

To convert it to FTBL file, you can run (in installation directory):

```
$ ./txt2ftbl.py test/prl_exp/e_coli_anto.txt [> test/prl_exp/e_coli_anto.ftbl]
```

Note that output redirection `> ...` is optional. In absence of such redirection, the output file name is guessed from input file by replacing `.txt` with `.ftbl` extension. Thus obtained FTBL file must be completed with several kinds of information like label input, label measurements and so on to be fully functional and suitable for `influx_si`.

ftbl2xgmml: cytoscape view

Once a valid FTBL file is generated, a user can visualize a graph representing his metabolic network in Cytoscape program. To produce necessary graph files, user can run:

```
$ ftbl2xgmml.py mynetwork[.ftbl] [> mynetwotk.xgmml]
```

or drag and drop `mynetwork.ftbl` icon on `ftbl2xgmml.py` icon.

The output redirection is optional.

This will produce a file in the XGMML format `mynetwork.xgmml` in the directory of `mynetwork.ftbl`:

Once a generated file `mynetwork.ftbl` is imported in cytoscape, a user can use one of automatic cytoscape layouts or edit node's disposition in the graph by hand. For those who use CySBML plugin, a saving of a particular layout in a file can be practical for later applying it to a new network.

Graphical conventions used in the generated XGMML are the following:

- metabolite are presented as rounded square nodes;

- simple (one to one) reaction are represented by simple edges;
- condensing and/or splitting reactions are represented by edges converging and/or diverging from additional almost invisible node having a label with the reaction name;
- all nodes and edges have tool tips, i.e. when a pointer is put over, their name (metabolite or reaction) appears in a tiny pop-up window;
- non reversible reaction are represented by a single solid line, have an arrow on the target end (i.e. produced metabolite) and nothing on the source end (i.e. consumed metabolite);
- reversible reactions are represented by a double parallel line and have a solid circle on the source end;
- color code for arrows:
  - green for free net flux;
  - blue for dependent net flux;
  - black for constrained net flux;
- color code for solid circles:
  - green for free exchange flux;
  - blue for dependent exchange flux;
  - black for constrained exchange flux.

ftbl2netan: FTBL parsing

To see how an FTBL file is parsed and what the parsing module "understands" in the network, a following command can be run:

```
$ ftbl2netan.py mynetwork[.ftbl] [> mynetwork.netan]
```

The output redirection is optional.

A user can examine `mynetwork.netan` in a plain text editor (not like Word) or in spreadsheet software. It has an hierarchical structure, the fields are separated by tabulations and the field values are Python objects converted to strings.

ftbl2cumoAb: human readable equations

Sometimes, it can be helpful to examine visually the equations used by `influx_si`. These equations can be produced in human readable form by running:

```
$ ftbl2cumoAb.py -r mynetwork[.ftbl] [> mynetwork.sys]
```

or:

```
$ ftbl2cumoAb.py --emu mynetwork[.ftbl] [> mynetwork.sys]
```

The output redirection is optional.

The result file `mynetwork.sys` will contain systems of stoichiometric and cumomer balance equations as well as a symbolic inversion of stoichiometric matrix, i.e. dependent fluxes are represented as linear combination of free and constrained fluxes and an optional constant value. In the examples above, the option `-r` stands for "reduced cumomer set" and `--emu` stands for "generate EMU framework equations". In this latter case, only isotopologues of mass+0 in each EMU are reported in `mynetwork.sys` file. For other mass weights, equations does not change and the right hand side term could get longer for condensation reactions but involves the same EMUs as in mass+0 weight.

If a full cumomer set has to be examined, just omit all options. Keep in mind that on real-world networks this can produce more than thousand equations by cumomer weight which could hardly be qualified as *human* readable form. So use it with caution.

For the sake of brevity, cumomer names are encoded in decimal integer form. For example, a cumomer `Metab#xx1x` will be referred as `Metab:2` because a binary number `0010` corresponds to a decimal number `2`. The binary mask `0010` is obtained from the cumomer mask `xx1x` by a plain replacement of every `x` by `0`.

For a given cumomer weight, the equations are sorted alphabetically.

expa2ftbl: non carbon carrying fluxes

Some reactions of carbon metabolism require cofactor usage like ATP/ADP and some others. A mass balance on cofactors can produce additional useful constraints on the stoechiometric system. Since the version 2.8, such mass balance equation on non carbon carrying metabolites can be put in `EQUATION` section of FTBL file. A utility `expa2ftbl.R` can be helpful for this purpose if a user has already a full set of reactions in expa format. To extract additional equation from an expa file, `expa2ftbl.R` can be used as:

```
$ R --vanilla --slave --args file.expa < expa2ftbl.R > file.ftbl_eq
```

Then an information for the generated `file.ftbl_eq` has to be manually copy/pasted to a corresponding FTBL file.

Note that `expa2ftbl.R` uses a Unix command `grep` and another utility described here above `ftbl2netan.py`.

res2ftbl_meas: simulated data

During preparation of a study, one of questions that biologist can ask is "Will the intended collected data be sufficient for flux resolution in a given network?" Some clue can be obtained by making "dry runs" of `influx_si` with `--noopt` (i.e. no optimization) option. User can prepare an FTBL file with a given network and supposed data to be collected. At first, the measurement values can be replaced by NAs while the SD values for measurements must be given in realistic manner. After running:

```
$ influx_s.py --noopt mynetwork
```

a utility `res2ftbl_meas.py` can be practical for preparing FTBL files with obtained simulated measurements:

```
$ res2ftbl_meas.py res2ftbl_meas.py mynetwork_res[.kvh] > mynetwork.ftbl_meas
```

(here `.kvh` suffix is optional). The information from the generated file `mynetwork.ftbl_meas` has to be manually copy/pasted into corresponding FTBL file. Getting an ftbl file with real values instead of NAs in measurement sections gives an opportunity to explore optimization behavior near a simulated point like convergence speed and/or convergence stability to cite few of them.

ffres2ftbl: import free fluxes

This utility imports free flux values and metabolite concentrations (if any) from a result file _res.kvh and inject them into an FTBL file. Usage:

```
$ ffres2ftbl.sh mynetwork_res.kvh [base.ftbl] > new.ftbl
```

If an optional argument `base.ftbl` is omitted, then the free flux values are injected into an FTBL file corresponding to the _res.kvh file (here `mynetwork.ftbl`). This script can be used on a Unix (e.g. Linux, MacOS) or on a cygwin (unix tools on Windows) platform. It makes use of another utility written in python `ff2ftbl.py`

ftbl2kvh: check ftbl parsing

This utility simply parses a ftbl file and write what was "understood" in a kvh file. No network analysis occurs here unlike in `ftbl2netan` utility. Usage:

```
$ ftbl2kvh.py mynetwork[.ftbl] [> mynetwork.kvh]
```

The output redirection is optional.

ftbl2metxml: prepare MetExplore visualization

Convert an FTBL file to an xml file suitable for visualization on MetExplore site. If a result kvh file `mynetwork_res.kvh` is present, it will be parsed to extract flux values corresponding to the last `influx_si` run and put them in `mynetwok_net.txt`, `mynetwork_fwd.txt` and `mynetwork_rev.txt`. As their names indicate, they will contain net, forward and revers flux values respectively.

IsoDesign: optimizing input label

One of means to increase a flux resolution can be an optimization of input label composition. A utility `IsoDesing` solving this problem was developed by Pierre Millard. It is not part of `influx_si` distribution and can be downloaded at http://metatoul.insa-toulouse.fr/metasys/software/isodes/. In a nutshell, it works by scanning all possible input label compositions with a defined step, running `influx_si` on each of them then collecting the SD information on all fluxes for all label compositions and finally selecting an input label optimal in some sens (according to a criterion chosen by a user).

CHAPTER

SIX

PROGRAMMER'S DOCUMENTATION FOR INFLUX_S

In this chapter, Application Programming Interface (API) docs are collected. It can be helpful for programmers desiring to extend some features of `influx_s` or to fix some bugs. This chapter can be safely skipped by users aiming at simple usage of `influx_s` for biological research.

C13_ftbl

- Parse .ftbl
- Analyse ftbl

Restrictions:

- metabolite name cannot have

  ":" - it's a separator in measure id "+" - in measurements it can be metab1+metab2+...

C13_ftbl.**aglom**(*na*, *ta*, *loop*)
: new matrix A (na), transpose A (ta) are used to aglomerate neigbour mutually influencing nodes in a supernode. Aglomerated noeds are put in the loop dictionnary. Return False if no nodes were aglomerated.

C13_ftbl.**aglom_loop1**(*A*)
: Agglomerate nodes of A if they are mutually influence each other i.e.they are in a loop of length 1. Return a new dictionary of influence where entries are those of A aglomerated and glued "by" tab symbol

C13_ftbl.**allprods**(*srcs*, *prods*, *isos*, *metab*, *isostr*)
: Return a set of tuples (cmetab, cisostr, vmetab, visostr) where cmetab and cisostr describe a contex metabolite which combined with metab+isostr produced vmetab+visostr. if metab is alone on its reaction part cmetab and cisostr are set to an empty string (""). The set covers all combination of metab+isostr and its co-substrates which produce isotopes having at least one labeled carbon from metab+isostr. Co-substrate isotops are in a dictionary isos[cmetab]=list(cisotopes).

C13_ftbl.**bcumo_decomp**(*bcumo*)
: bcumo is a string of the form #[01x]+. It has to be decomposed in the linear combination of cumomers #[1x]+. The coefficients of this linear combination are 1 or -1. So it can be represented as sum(cumos_positive)-sum(cumos_negative). The result of this function is a dictionary {"+": list of icumos, "-": list of icumos}. icumo is an integer whose binary form indicates 1's positions in a cumomer.

C13_ftbl.**conv_mid**(*x*, *y*) → z
: convolute two mid vectors (numpy arrays) and return the result as numpy array.

C13_ftbl.**cumo_infl**(*netan*, *cumo)->list(tuple(in_cumo*, *fl*, *imetab*, *iin_metab)*)
: return the list of tuples (in_cumo, fl, imetab, iin_metab): input cumomer, flux (fwd.fl or rev.fl), index of metab and index of in_metab generating cumo. cumo is in format "metab:icumo". Condenstation reaction will give the same flux and icumo but various iin_metab. Convergent point will give multiple fluxes.

C13_ftbl.**cumo_iw**(*w*, *nlen*)
: iterator for a given cumomer weight w in the carbon length nlen

C13_ftbl.**cumo_path**(*starts*, *A*, *visited=set([])*)
Enumerate cumomers along reaction pathways. Algo: start from an input, follow chemical pathways till no more neighbours or till only visited metabolite rest in network. Return a list of cumomer pathways. Each pathways is an ordered list.

C13_ftbl.**dom_cmp**(*A*, *i*, *j*)
Compares influances of i-th and j-th lements of A. Returns 0 if i and j are mutually influenced, 1 if i in A[j] (i influences j) , -1 if otherwise

C13_ftbl.**enum_path**(*starts*, *netw*, *outs*, *visited=set([])*)
Enumerate metabilites along to reaction pathways. Algo: start from an input, follow chemical pathways till an output or already visited metabolite. Returns a list of metabolite pathways. Each pathways is an ordered list.

C13_ftbl.**formula2dict**(*f*, *pterm=<_sre.SRE_Pattern object at 0x7f935aa017b0>*, *pflux=<_sre.SRE_Pattern object at 0x1bfc490>*)
parse a linear combination sum([+|-][a_i][*]f_i) where a_i is a positive number and f_i is a string starting by non-digit and not white character (# is allowed). Output is a dict f_i:[+-]a_i

C13_ftbl.**frag_prod**(*metab*, *frag*, *s*, *cmetab*, *cfrag*, *cs*, *prods*)
Get fragments from labeled substrates

C13_ftbl.**ftbl_netan**(*ftbl*, *netan*, *emu_framework=False*, *fullsys=False*, *case_i=False*)
analyse ftbl dictionary to find

- •network inputs (input)
- •network outputs (output)
- •substrates (subs)
- •products (prods)
- •metabolites (metabs)
- •reactions (reacs)
- •not reversible reactions (subset of reacs) (notrev) all above items are in named sets
- •stocheometric matrix (sto_r_m)
- •stocheometric matrix (sto_m_r)
- •fwd-rev flux matrix (flux_m_r)
- •cumomer balances (cumo_m_r_m)
- •carbon length (Clen)
- •reaction formula (formula)
- •metabolite network (metab_netw)
- •carbon transitions (carbotrans)
- •free fluxes (flux_free)
- •constrained fluxes (flux_constr)
- •measured fluxes (flux_measured)
- •variable growth fluxes (flux_vgrowth)
- •input isotopomers (iso_input)
- •input cumomers (cumo_input)
- •input reduced cumomers (rcumo_input)
- •flux inequalities (flux_ineqal)
- •flux equalities (flux_eqal)

- label measurements, H1 (label_meas)
- peak measurements, C13 (peak_meas)
- mass measurements (mass_meas)
- cumomer ordered lists (vcumo)
- unknown fluxes ordered lists (vflux)
- linear problem on fluxes (Afl, bfl)
- free fluxes ordered lists (vflux_free)
- fw-rv fluxes ordered lists (vflux_fwrv)
- row names ordered lists for Afl (vrowAfl)
- in-out fluxes (flux_in, flux_out)
- measured concentrations (metab_measured)

C13_ftbl.**ftbl_parse**(*f*) → dict
read and parse .ftbl file. The only input parameter f is a stream pointer with read permission or a file name. This function parses the input and returns a dictionnary with items corresponding to sections in .ftbl. One section is added. "TRANS" correponds to carbon transitions.

C13_ftbl.**infl**(*metab*, *netan)->set(fluxes*)
List incoming fluxes for this metabolite (fwd.reac, rev.reac, ...)

C13_ftbl.**iso2cumo**(*netan*, *strin*, *in_cumo*, *icumo*, *in_metab*)
calculate cumomer fraction from isotopomer ones

C13_ftbl.**iso2emu**(*netan*, *inmetab*, *mask*, *mpi*, *e*)
calculate emu fraction from isotopomer dict iso_input. The fraction corresponds to a fragment defined by a mask and the mass component mpi. Return a real number in [0; 1] interval.

C13_ftbl.**label_meas2matrix_vec_dev**(*netan*)
use netan["label_meas"] list to construct a corresponding list of measure matrix matx_lab such that scale_diag*metab_pool_diag*matx_lab*(cumos_vector,1) corresponds to label_measurements_vector. matx_lab is defined as list of dict{"scale":scale_name, "coefs":dict{icumo:coef}, "metab": metabolite, "poolid": metabolite pool id if pooled} where coef is a contribution of cumo in linear combination for given measure. scale_name is of the form "metabs;group". Group number is to group measurements of the same measurement set. poolid is the index of pool list in pooled where each list regroups 0-based indexes rows in returned matrix for what has to be pooled together. vec is a list of measurements (values in .ftbl) dev is a list of deviations. Elements in matx_lab, vec and dev are ordered in the same way. The returned result is a dict (mat,vec,dev)

C13_ftbl.**labprods**(*prods*, *metab*, *isostr*, *strs*)
Return a set of tuples (vmetab,visostr) which receive at least one labeled carbon from (metab, isostr)

C13_ftbl.**lowtri**(*A*)
Try low triangular ordering of matrix A entries

C13_ftbl.**mass_meas2matrix_vec_dev**(*netan*)
use netan["mass_meas"] list to construct a corresponding list of measure matrix matx_mass such that scale_diag*matx_mass*cumos_vector corresponds to mass_measures_vector. matx_mass is defined as matx_lab in label_meas2matrix_vec_dev() Elements in matx_mass, vec and dev are ordered in the same way. scale name is defined as "metab;fragment_mask" The returned result is a dict (mat,vec,dev)

C13_ftbl.**mat2graph**(*A*, *fp*)
write digraph file on file pointer fp representing links in matrix A given as bi-level dictionnary. A key of first level (row index) is influenced by keys of second level (column indicies).

C13_ftbl.**mat2pbm**(*A*, *v*, *fp*)
Write an image map of non-zero entries of matrix A to file pointer fp. Matrix A is a dictionnary, v is a list ordering keys of A.

C13_ftbl.**mecoparse**(*terms*, *pmeco=<_sre.SRE_Pattern object at 0x1c3c9a0>*)
Parse a string term from a list (or a sing string) of chemical equation entries. The general form of each term is 'coef*metab'. coef (if present) must be separated from metab by '*' and be convertible to float. metab can start with a number (e.g. '6PG') so the presence of '*' is mandatory to separate coef from metab.If coef is absent, it is considered to be 1. Return a list of (or a single for str) tuples (metab (str), coef (real)).

C13_ftbl.**ms_frag_gath**(*netan*)
gather metabolite fragments necessary to obtain a given set of data observed in MS measurements. The fragment mask is encoded in the same way as cumomers, Met:7 <=> Met#(0)111

C13_ftbl.**ntimes**(*n*)
Return charcater string 'once' for n=1, 'twice' for n=2 and 'n times' for other n

C13_ftbl.**peak_meas2matrix_vec_dev**(*netan*, *dmask={'S': 2*, *'D-': 6*, *'D+': 3*, *'DD': 7*, *'T': 7}*)
use netan["peak_meas"] list to construct a corresponding list of measure matrix matx_peak such that scale_diag*matx_peak*cumos_vector corresponds to peak_measures_vector. dmask is a dictionary with 3 carbon labeling pattern mask for various peak types. The middle bit corresponds to the targeted carbon, lower bit corresponds to the next neighbour (D+) and higher bit corresponds to previous carbon (D-). matx_peak is defined as matx_lab in label_meas2matrix_vec_dev() Elements in matx_peak, vec and dev are ordered in the same way. scale name is defined as "metab;c_no;irow" The returned result is a dict (mat,vec,dev)

C13_ftbl.**proc_kinopt**(*ftbl*, *netan*)
Proceed label kinetics options from OPTIONS section: file_labcin, dt, tmax, nsubdiv_dt

C13_ftbl.**proc_label_input**(*ftbl*, *netan*)
Proceed LABEL_INPUT section in ftbl and add result to the list netan["iso_input"] List item is a dict {}metab;{isotop_int_index:fraction}}

C13_ftbl.**proc_label_meas**(*ftbl*, *netan*)
Proceed LABEL_MEASUREMENT section of ftbl file, add the result to a list of dicts

C13_ftbl.**proc_mass_meas**(*ftbl*, *netan*)
Proceed PEAK_MEASUREMENT section of ftbl file, add the result to a list of dicts

C13_ftbl.**proc_peak_meas**(*ftbl*, *netan*)
Proceed PEAK_MEASUREMENT section of ftbl file, add the result to a list of dicts

C13_ftbl.**prod**(*metab*, *iso*, *s*, *cmetab*, *ciso*, *cs*, *prods)->set()*
get isotops from labeled substrates

C13_ftbl.**rcumo_sys**(*netan*, *emu=False*)
Calculate reduced cumomers or EMU systems A*x=b we start with observed cumomers (emus) of max weight and we include only needed involved cumomers (emus) A list of cumomer (emu) lists (by weight) is stored in netan["vrcumo"] (netan["vemu"])

C13_ftbl.**src_ind**(*substrate*, *product*, *iprod*)
For a given substrate and product carbon strings (e.g. "abc", "ab") calculate substrate index corresponding to product index. Return None if no source found. Return 0 if iprod==0 and intersection of product and substrate strings is not empty

C13_ftbl.**t_iso2cumo**(*n*)
t_iso2cumo(n) return transition matrix from isotopomers fractions to cumomer vector n - carbon number return numpy array of size (2**n,2**n)

C13_ftbl.**t_iso2m**(*n*)
t_iso2m(n) return transition matrix from isotopomers fractions to MID vector n - carbon number return numpy array of size (n+1,2**n)

C13_ftbl.**t_iso2pos**(*n*)
t_iso2pos(n) return transition matrix from isotopomers fractions to positional labelling vector (cumomers of weight 1) n - carbon number return numpy array of size (n,2**n)

C13_ftbl.**topo_order**(*A*, *tA*)
Try to sort keys of A in topological order. tA is just a transpose of A

C13_ftbl.**transpose**(*A*)
Transpose a matrix defined as a dict.

C13_ftbl.**werr**()
write(str) -> None. Write string str to file.

Note that due to buffering, flush() or close() may be needed before the file on disk reflects the data written.

C13_ftbl.**wout**()
write(str) -> None. Write string str to file.

Note that due to buffering, flush() or close() may be needed before the file on disk reflects the data written.

ftbl2code

Module for translation of .ftbl file to R code

ftbl2code.**netan2Abcumo_spr**(*varname*, *Al*, *bl*, *vcumol*, *minput*, *f*, *fwrv2i*, *incu2i_b1*)
Transform cumomer linear sytems collection (from ftbl file) to a R code calculating sparse matrix A and vector b in A*x+b=0 for a given weight of fragment iw (index in resulting list) Flux vector fl of all fwd. and rev. fluxes are known at R runtime.

Resulting code is a list sprAb indexed by cumomer weight (cf. generated R comments for details on sprAb) cumomer vector incu=c(1, xi, xl), xi - input cumomers, xl - lighter cumomers.

incu2i_b1 gives i in incu from cumomer name. i=1 corresponds to the constant 1.

ftbl2code.**netan2R_cumo**(*netan*, *org*, *f*) → dict
generate data structures for full cumomer matrices

ftbl2code.**netan2R_fl**(*netan*, *org*, *f*)
generate R code for flux and pool part for more details cf. netan2Rinit()

ftbl2code.**netan2R_ineq**(*netan*, *org*, *f*)
generate inequality code

ftbl2code.**netan2R_meas**(*netan*, *org*, *f*)
generate code for measure treatment

ftbl2code.**netan2Rinit**(*netan*, *org*, *f*, *fullsys*, *emu=False*, *ropts=[]*)
Write R code for initialization of all variables before cumomer system resolution by khi2 minimization. Args:

netan: a collection of parsed ftbl information f: R code output pointer fullsys (logical): write a code for the full or only reduced cumomer system emu (logical): write equations in EMU framework or cumomer (default) ropts: list of items "param=value" to be written as is in R file.

Return:

a dictionnary with some python variables:

- "measures": measures,
- "o_mcumos": o_mcumos,
- "cumo2i": cumo2i,
- ...

ftbl2netan

Parse ftbl file from stdin or from first parameter and write netan in kvh format on stdout usage: ftbl2netan.py network[.ftbl] [> network.netan]

ftbl2optR

Transform an ftbl to R code which will solve an optimization of flux analysis problem $\arg\min_\Theta S$, where $S = ||\text{Predicted} - \text{Observed}||^2_\Sigma$ and Θ is a vector of parameters to fit: free fluxes (net+xch), scaling parameters and metabolite concentrations pools. Two variants of R code can be generated: "s" and "i" for stationary and isotopically nonstationary labeling. Predicted vector is obtained from cumomer or emu vector x (calculated from free fluxes and divided in chunks according to the cumo weight) by multiplying it by the measurement matrices, weighted by metabolite pools (in case of pooling) and scale factor (for stationary case only), boths coming from ftbl file. Observed values vector xo is extracted from ftbl file for "s" case and from special text file for "i" case. It is composed of flux, label measurements and metabolite pools. Σ^2, covariance diagonal matrices sigma[flux|mass|label|peak|metab.pool] is orginated from the ftbl file.

usage: ./ftbl2optR.py [opts] organism where organism is the ftbl informative part of file name (before .ftbl), e.g. organism.ftbl after execution a file organism.R will be created. If it already exists, it will be silently overwritten. The system Afl*flnx=bfl is created from the ftbl file.

Important python variables:

- case_i - if True, the case is "i" otherwise it is the "s" case

Collections:

- netan - (dict) ftbl structured content
- tfallnx - (3-tuple[reac,["d"|"f"|"c"], ["net"|"xch"]] list)- total flux collection
- measures - (dict) exp data
- rAb - (list) reduced linear systems A*x_cumo=b (a system by weight)
- scale - unique scale names
- nrow - counts scale names
- o_sc - ordered scale names
- o_meas - ordered measurement types

File names (str):

- n_ftbl (descriptor f_ftbl)
- n_R (R code) (f)
- n_fort (fortran code) (ff)

Counts:

- nb_fln, nb_flx, nb_fl (dependent fluxes: net, xch, total), nb_ffn, nb_ffx (free fluxes)

Index translators:

- fwrv2i - flux names to index in R:fwrv
- cumo2i - cumomer names to index in R:x
- ir2isc - mapping measurement rows indexes on scale index isc[meas]=ir2isc[meas][ir]

Vector names:

- cumos (list) - names of R:x
- o_mcumos - cumomers involved in measurements

Important R variables:

Scalars:

- nb_w, nb_cumos, nb_fln, nb_flx, nb_fl (dependent or unknown fluxes),

- nb_ffn, nb_ffx, nb_ff (free fluxes),
- nb_fcn, nb_fcx, nb_fc (constrained fluxes),
- nb_ineq, nb_param, nb_fmn

Name vectors:

- nm_cumo, nm_fwrv, nm_fallnx, nm_fln, nm_flx, nm_fl, nm_par,
- nm_ffn, nm_ffx,
- nm_fcn, nm_fcx,
- nm_mcumo, nm_fmn

Numeric vectors:

- fwrv - all fluxes (fwd+rev)
- x - all cumomers (weight1+weight2+...)
- param - free flux net, free flux xch, scale label, scale mass, scale peak, metabolite concentrations
- fcn, fcx, fc - constrained fluxes
- bp - helps to construct the rhs of flux system
- xi -cumomer input vector
- fallnx - complete flux vector (constr+net+xch)
- bc - helps to construct fallnx
- li - inequality vector (mi%*%fallnx>=li)
- ir2isc - measure row to scale vector replicator
- ci - inequalities for param use (ui%*%param-ci>=0)
- measvec - measurement vector
- fmn - measured net fluxes

Matrices:

- Afl, qrAfl, invAfl,
- p2bfl - helps to construct the rhs of flux system
- mf, md - help to construct fallnx
- mi - inequality matrix (ftbl content)
- ui - inequality matrix (ready for param use)
- measmat - for measmat*x+memaone=vec of simulated not-yet-scaled measurements

Functions:

- lab_sim - translate param to flux and cumomer vector (initial approximation)
- cumo_cost - cost function (khi2)
- cumo_gradj - implicit derivative gradient

ftbl2xgmml

read a .ftbl file from a parameter and translate to .xgmml file. The generated xgmml file can be then imported into Cytoscape (www.cytoscape.org). Reactions involving two substrates or two products are represented by an additional almost invisible node while one-to-one reactions are just edges. Node and edge attributes are written in respective xml attributes. Compatibility: cytoscape v2.8.3 and v3.0

usage: ftbl2xgmml.py [-h|–help] mynetwork.ftbl [> mynetwork.xgmml]

OPTIONS -h, –help print this message and exit

param mynetwork the base of an ftbl file (mynetwork.ftbl)

returns mynetwork.xgmml – file of the network definition suitable for cytoscape

kvh

kvh.**dict2kvh**(*d*, *fp=sys.stdout*, *indent=0*)
Write a nested dictionary on the stream fp (stdout by default).

kvh.**escape**(*s*, *spch="|&;<>()$\"' tn*?[#~=%"*, *ech="\"*)
escape special characters in s. The special characters are listed in spch. Escaping is done by putting an ech string before them. Default spch and ech corresponds to quoting Shell arguments in accordance with http://www.opengroup.org/onlinepubs/009695399/utilities/xcu_chap02.html Example: os.system("ls %s" % escape(file_name_with_all_meta_chars_but_newline)); .. note:

1.Escaped <newline> is removed by a shell if not put in a single-quotted string (' ')

2.A single-quote character even escaped cannot appear in a single-quotted string

kvh.**kvh2dict**(*fp*)
Read a kvh file from fp pointer then translate its tlist structure to a returned hierarchical dictionnary. Repeated keys at the same level of a dictionnary are silently overwritten

kvh.**kvh2obj**(*fp*)
Read a kvh file from fp pointer then translate its tlist structure to a returned object hierarchy. Repeated fields at the same level of an object are silently overwritten

kvh.**kvh2tlist**(*fp, lev=[0], indent=[0]*)
Read a kvh file from fp stream descriptor and organize its content in list of tuples [(k1,v1), (k2,[(k2.1, v2.1)])] If fp is a string, it is used in open() operator

kvh.**kvh_get_matrix**(*fp*, *keys*)
Get matrix or vector whose key suite is in a list keys from a kvh file given in fp (file pointer of file name). For big kvh files, this function can be much faster than kvh2tlist()+kvh_getv_by_k() Return a matrix which is a list of lists (rows). The first item in each row is the row name. In case of matrix (i.e. "row_col" is present in kvh file), the very first row contain column names.

kvh.**kvh_getv_by_k**(*kvt*, *kl*) → None|String|kvh tlist
get value from kvt (kvh tlist) according to the key hierarchy defined in the list of keys kl. Return None if no key is found

kvh.**kvh_read_key**(*fp*)

Read a string from the current position till the first unescaped ,

or the end of stream fp. Return tuple (key, sep). sep=None at the end of the stream

kvh.**kvh_read_val**(*fp*)

Read a string from current position till the first unescaped

or the end of file. Return the read string.

kvh.**kvh_tlist2dict**(*tlist*)
Translate a tlist structure read from a kvh file to a hierarchical dictionnary. Repeated keys at the same level of a dictionnary are silently overwritten

kvh.**kvh_tlist2obj**(*tlist*)
Translate a tlist structure read from a kvh file to a hierarchical dictionnary. Repeated keys at the same level of a dictionnary are silently overwritten

kvh.**tlist2kvh**(*d*, *fp=sys.stdout*, *indent=0*)
Write a (hierarchichal) list of 2-tuples on the stream fp (stdout by default).

tools_ssg

tools_ssg.**aff**(*name*, *obj*, *ident=0*, *f=<open file '<stdout>', mode 'w' at 0x7f9367ae6150>*)

print formatted object: name=obj

tools_ssg.**arr2pbm**(*A*, *fp*)
Write an image map of non-zero entries of matrix A to file pointer fp. Matrix A is an array

tools_ssg.**asort**(*d*)
sorts a dictionnary by value preserving key=value association the result is a list of tuples (key,value)

tools_ssg.**cumsum**(*l*, *tot=0*)

Returns an iterable of the length len(l)+1 with cumulated sum of items in l. First element in cumsum is equal to initial value of tot. Result depends on the meaning of "+" operator for l items and of tot type.

```
>>> list(cumsum("abc",tot=""))
['', 'a', 'ab', 'abc']

>>> list(cumsum(xrange(1,5)))
[0, 1, 3, 6, 10]
```

tools_ssg.**expandbit**(*i*, *pos*)
copy bits set to 1 in i to the result position given in the list pos. length of pos must be greater or equal to bitlength of i

tools_ssg.**icumo2iiso**(*icumo*, *size*)

Returns iterator on isotopomers composing a given icumo. size is carbon number

tools_ssg.**isstr**(*s*)

Returns True if the argument is a string

tools_ssg.**iterbit**(*i*, *size=0*)
iterator on bits in integer starting from 0-position. The iterator stops at highest non-zero bit

tools_ssg.**iternumbit**(*i*, *size=0*)
iterator on bits and its number in integer starting from 0-position. The iterator yields tuples (n,bit). If optional size is zero then it stops at highest non-zero bit. If not, it will stop at bit number size-1.

tools_ssg.**join**(*c*, *l*, *p=''*, *s=''*, *a=''*)
join the items of the list (or iterator) l separated by c. Each item is prefixed with p and suffixed with s. If the join result is empty for any reason, an alternative a is returned. p, s and a are optional

tools_ssg.**joint**(*c*, *l*, *p=''*, *s=''*, *a=''*)
join "true" items of the list (or iterator) l separated by c. Each item is prefixed with p and suffixed with s. If the join result is empty for any reason, an alternative a is returned. p, s and a are optional

tools_ssg.**list2count**(*l*, *incr=1*)
count values in a (short) list l incrementing the counter by optional incr.

Returns a dictionary {item:count}

tools_ssg.**read_table**(*f*) → dict(mat, col_names) read a plain text file f in a numpy mat. If some columns are not numerical, they are replaced by np.nan. If header=True, number of column names in the first row after skip must be the same as the number of values in each following row.

tools_ssg.**reverse**(*it*)
reverse order of an iterable

tools_ssg.**rstrbit**(*i*, *size=0*)

Returns the integer as reversed string binary representation. The lowest bit is on the left side

tools_ssg.**setbit32**(*i*, *nb*)
set a bit number nb (0 based) in an integer i

tools_ssg.**setcharbit**(*s*, *ch*, *i*)
set character ch in a string s everywhere a corresponding bit of i is set

tools_ssg.**ssign**(*i*, *sp='+'*, *sm='-'*)

Returns a string of i sign: sp (i>=0) or sm (i<0).

tools_ssg.**strbit**(*i*, *size=0*)

Returns the lowest part of integer as string binary representation

tools_ssg.**strbit2int**(*s*)
translate a string of 0's and 1's interpreted as bits to an integer all characters different from 0,1 are silently ignored

tools_ssg.**strbit32**(*i*)

Returns a string of 0-1s (in chunk of 4) in an 32 bit integer

tools_ssg.**sumbit**(*i*)

Returns sum of bits in an integer

tools_ssg.**trd**(*l*, *d*, *p=''*, *s=''*, *a=''*)
translate items in an iterable l by a dictionary d, prefixing translated items by optional p and suffixing them by optional s. If an item is not found in the dictionnary alternative string a is used. If a==None, the item is left unchanged. No prefix or suffix are applied in both case.

Returns iterator

tools_ssg.**ulong**(*i*) → workarounded ulong

tools_ssg.**valval**(*o*, *keepNone=True*)

Returns an iterator over values of values, i.e. collapsing values of fisrt two nested lists in one list, for example.

tools_ssg.**wxlay2py**(*kvt, parent=[None]*)

Returns a string with python code generating wxWindow widget layout described in kvh tlist sturcture

CHAPTER

SEVEN

HOW TO ...

... choose free fluxes?

You can define in FTBL all not constrained fluxes as dependent (put a letter `D` in the column `FCD` of the FTBL sections `FLUXES/NET` and `FLUXES/XCH`), run `influx_si` and see an error message that will suggest some candidates for free fluxes. For these fluxes, put a letter `F` in the column `FCD` and some numeric value in the next column `VALUE(F/C)` to provide a starting value for the fitting. Don't use 0 as starting value as it might lead to singular matrices in cumomer balances.

If you want to create an FTBL *de novo*, consider using application `txt2ftbl.py` included in `influx_si` package. Not only it translates an easily readable/writable text format into FTBL one, but it also automatically assigns some fluxes to be free.

... get statistical information for a given set of free fluxes without fitting measurements?

Put these values in the corresponding FTBL file as starting values for free fluxes and use `influx_si` with `--noopt` option.

... accelerate calculations?

You can relax stopping criterion and pass from 1.e-5 (by default) to, for example, 1.e-2 if this precision is sufficient for you. Use `optctrl_errx` option in FTBL file (section `OPTIONS`) for this.

If you mean to accelerate Monte-Carlo simulations in Unix environment, you can use a hardware with many cores. In this case, the wall clock time can be reduced significantly. Note that distant nodes, even inside of the same cluster, are not used in the such kind of Monte-Carlo simulations.

Check that your system is not using swap (disk) memory. If it is the case, stop other applications running in parallel with `influx_si`. If possible extend the RAM on your hardware.

... extend upper limit for non linear iterations?

By default, this value is 50 which should be largely sufficient for most cases. If not, you can set another value via `optctrl_maxit` option in the FTBL file (section `OPTIONS`). But most probably, you would like to check your network definition or to add some data or to change a substrate labeling, anyway to do something to get a well defined network instead of trying to make converge the fitting on some biologically almost meaningless situation.

... make FTBL file with synthetic data?

Follow for example steps outlined hereafter:

- edit FTBL file(s) with `NA` in measurements and realistic SD, name it e.g. `new_NA.ftbl`
- simulate data:

```
$ influx_s.py --noopt --addnoise new_NA
```

- prepare FTBL sections with simulated data:

```
$ res2ftbl_meas.py new_NA_res.kvh
```

It will create file (or files if there are parallel experiments) with synthetic data formatted for inclusion in FTBL file: `new_NA_sim1.ftbl`, `new_NA_sim2.ftbl`, etc.)

- copy/paste simulated data to a new file `new.ftbl` with numeric data instead of `NA`.
- use FTBL with synthetic data:

```
$ influx_s.py new.ftbl
```

... do custom post-treatment of Monte-Carlo iterations?

Let suppose that you want to filter some of Monte-Carlo (MC) iterations based on their cost values. In `OPTIONS/psottreat_R` of your FTBL file add `save_all.R`. The file `save_all.R` can be found in `test` directory of `influx_si` distribution and must be copied to the directory where your FTBL file resides. Execution of `save_all.R` at the end of calculations will simply save all session variables in `mynetwork.RData` file (supposing that your FTBL file is names `mynetwork.ftbl`). In particular, you need `free_mc` matrix which contains free parameters (each column results from a given MC iteration). After that you can open an interactive R session in your working directory and run something similar to:

```
# preparations
load("mynetwork.RData")
source(file.path(dirx, "libs.R"))
source(file.path(dirx, "opt_cumo_tools.R"))
#source(file.path(dirx, "opt_icumo_tools.R")) # uncoment for influx_i use
tmp=sparse2spa(spa)

# doing something useful
# here, we calculate a vector of cost values, one per MC iteration
cost_mc=apply(free_mc, 2, function(p) cumo_cost(p, labargs))
# do something else ...
```

If, instead of cost values, you need for example a full set of net-xch fluxes then do

```
allflux_mc=apply(free_mc, 2, function(p) param2fl(p, labargs)$fallnx)
```

for residuals, do:

```
resid_mc=apply(free_mc, 2, function(p) lab_resid(p, FALSE, labargs)$res)
```

After that, you can filter or do whatever needed with obtained vectors and matrices.

CHAPTER

EIGHT

TROUBLESHOOTING

The software is provided "AS IS" so for the troubleshooting you are mostly on your own. We don't provide any support of any kind for the software itself. Nevertheless, if you need help for your label experiment design and/or realization, you can contact our platform MetaToul (cf. *Consulting and more*)

Anyway, you can try to solve some current problems by yourself or with a local help.

If you have a problem during installation, you can ask for help from your local computer desk.

If you have a problem with FTBL editing, you can read the documentation from 13CFlux and/or interpret error messages generated during FTBL parsing.

If you have some difficulties in choosing free fluxes, define all not constrained fluxes as dependent (put a letter `D` in the column `FCD` of the FTBL sections `FLUXES/NET` and `FLUXES/XCH`) and see an error message that will suggest candidates for free fluxes.

If your resulting fluxes are badly defined (statistically or structurally), i.e. they have big confidence intervals or the Jacobian is rank deficient, you can try to play with input labeling (cf. IsoDesign software at http://metatoul.insa-toulouse.fr/metasys/software/isodes/) or try to collect some additional data on metabolites not yet measured. To have some insights on what part of the network is already well defined and which one still needs additional measurements, you can try to run `influx_si` with an option `--ln` (as *least norm*) (in addition to `--noopt` option) and examine standard deviation of the fluxes/concentrations in the resulting KVH file.

If you think to discover a bug in `influx_si` you can report it to the author by email `sokol [at] insa-toulouse [dot] fr`. At this moment, please be sure to use the latest available release as the bug may be already corrected or not be actual any more. Note also that we can't guarantee that any particular bug can be fixed in any particular release or can be fixed at all. It is possible, that we ask you to send us an ftbl file on which an error occur. It will be done only for purposes of bug reproducing and its identification. The received ftbl file will not be transmitted to any third party.

Once again, if you could not resolve your problem by your own, see the next section *Consulting and more*.

CHAPTER

NINE

CONSULTING AND MORE

If you need help in design, conducting and interpretation of label experiments, you can expose your problem in a brief email to our platform MetaToul (metatoul [at] insa-toulouse [dot] fr) located in Toulouse, France. A dedicated person will take contact with your to detail what can be done to help you and to draw up a quote.

You don't have to ask for a consulting for a simple bug submission. A bug submission can be directly made to sokol [at] insa-toulouse [dot] fr.

For more details about the platform MetaToul, you can visit our web site http://www.metatoul.fr (english version is available).

CHAPTER

TEN

LICENSE FOR INFLUX_S SOFTWARE

LICENCE for influx_s software

Educational Community License, Version 2.0 [OSI Approved License]

Educational Community License Version 2.0, April 2007

The Educational Community License version 2.0 ("ECL") consists of the Apache 2.0 license, modified to change the scope of the patent grant in section 3 to be specific to the needs of the education communities using this license. The original Apache 2.0 license can be found at:
http://www.apache.org/licenses/LICENSE-2.0

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION
1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version
of the Work and any modifications or additions to that Work or Derivative Works
thereof, that is intentionally submitted to Licensor for inclusion in the Work
by the copyright owner or by an individual or Legal Entity authorized to submit
on behalf of the copyright owner. For the purposes of this definition,
"submitted" means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems, and
issue tracking systems that are managed by, or on behalf of, the Licensor for
the purpose of discussing and improving the Work, but excluding communication
that is conspicuously marked or otherwise designated in writing by the copyright
owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf
of whom a Contribution has been received by Licensor and subsequently
incorporated within the Work.

2. Grant of Copyright License.

Subject to the terms and conditions of this
License, each Contributor hereby grants to You a perpetual, worldwide,
non-exclusive, no-charge, royalty-free, irrevocable copyright license to
reproduce, prepare Derivative Works of, publicly display, publicly perform,
sublicense, and distribute the Work and such Derivative Works in Source or
Object form.

3. Grant of Patent License.

Subject to the terms and conditions of this License,
each Contributor hereby grants to You a perpetual, worldwide, non-exclusive,
no-charge, royalty-free, irrevocable (except as stated in this section) patent
license to make, have made, use, offer to sell, sell, import, and otherwise
transfer the Work, where such license applies only to those patent claims
licensable by such Contributor that are necessarily infringed by their
Contribution(s) alone or by combination of their Contribution(s) with the Work
to which such Contribution(s) was submitted. If You institute patent litigation
against any entity (including a cross-claim or counterclaim in a lawsuit)
alleging that the Work or a Contribution incorporated within the Work
constitutes direct or contributory patent infringement, then any patent licenses
granted to You under this License for that Work shall terminate as of the date
such litigation is filed. Any patent license granted hereby with respect to
contributions by an individual employed by an institution or organization is
limited to patent claims where the individual that is the author of the Work is
also the inventor of the patent claims licensed, and where the organization or
institution has the right to grant such license under applicable grant and
research funding agreements. No other express or implied licenses are granted.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof
in any medium, with or without modifications, and in Source or Object form,
provided that You meet the following conditions:

You must give any other recipients of the Work or Derivative Works a copy of
this License; and

You must cause any modified files to carry prominent notices stating that You
changed the files; and

You must retain, in the Source form of any Derivative Works that You distribute,
all copyright, patent, trademark, and attribution notices from the Source form
of the Work, excluding those notices that do not pertain to any part of the

Derivative Works; and

If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Educational Community License to your work

To apply the Educational Community License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner] Licensed under the Educational Community License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.osedu.org/licenses/ECL-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

influx_sim (hereafter "the program") is a scientific software designed to model various situation in metabolic networks.

All its parts are freely distributed according to GPL2 terms http://www.gnu.org/licenses/gpl-2.0.txt (permission to execute, to copy, to modify and to redistribute; obligation to preserve the original copyright notice and to keep your modifications OpenSource only if you redistribute them alone or as part of influx_sim)

WARRANTY

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

COPYRIGHT 2011-2014 INRA

CHAPTER

ELEVEN

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

c

f

k

t

R

S

T

U

V

W