NOBLEFIT

A Matlab/GNU Octave toolbox to fit (environmental) data

Matthias S. Brennwald

matthias.brennwald@eawag.ch

29th August 2014

Contents

1 What is `NOBLEFIT`?

`NOBLEFIT` is a flexible Matlab/GNU Octave toolbox for quantitative interpretation of (environmental) tracers in terms of environmental processes and models (e.g., dissolved noble gases, other atmospheric gases, or just about anything else that deserves quantitative model-based interpretation).

In contrast to similar tools[1] where the possible tracers and models are hard-wired into the code, `NOBLEFIT` is designed to allow the user to define his own tracers and models.

2 Installation and Setup

This text assumes you are familiar with Matlab or GNU Octave, and that you have a working installation of either Matlab or GNU Octave on your computer.

2.1 Download `NOBLEFIT`

There are two ways to get `NOBLEFIT`:

- The easy method is to download `NOBLEFIT` as a ZIP archive, and expand the files from the archive. You can download the ZIP archive from `http://sourceforge.net/p/noblefit/code/HEAD/tarball`.

- Alternatively, if you have subversion (SVN) software installed on your computer, you can get `NOBLEFIT` as a SVN repository, which makes upgrading `NOBLEFIT` very easy. You can checkout the SVN repository from `http://sourceforge.net/p/noblefit/code/HEAD/tree`.

2.2 Install `NOBLEFIT`

Once you have a copy of `NOBLEFIT` downloaded to your computer, you should move the `NOBLEFIT` folder to a convenient location on your computer. I like to keep all your Matlab/GNU Octave code and packages in one directory, which contains several subdirectories for the different packages and projects. This greatly helps Matlab/GNU Octave to find your files. For instance, I keep all my m-files in `~/m-files/`, so `NOBLEFIT` goes to `~/m-files/noblefit`. Then I tell Matlab/GNU Octave where to look for the `NOBLEFIT` functions by including the command `addpath('~/m-files/noblefit')` in the `startup.m` file (Matlab) or the `.octaverc` file (GNU Octave).

3 Getting started with NOBLEFIT

The NOBLEFIT tools work the same as any other Matlab/GNU Octave tool. First you need to make sure Matlab or GNU Octave knows where to look for the NOBLEFIT files (Sec. 2.2). Then you just type the name of the functions to the command shell to execute them.

The core of NOBLEFIT is the noblefit.m function, which fits the data to the model using the χ^2 regression method [2]. It reports the best-fit values of the fitted model parameters, their errors (calculated by propagation of the data errors), and the some statistics about the goodness of the fit (χ^2, p value, and degrees of freedom of the regression).

Before calling the noblefit.m function, however, you need to load and format the data, define the model function, and specify the initial and fixed values of the model parameters. The function the noblefit.m facilitats loading and formatting the data, and there are a few predefined model functions in the the modelfunctions/ folder which you can use as examples.

Finally, there are a few complete examples of how to use NOBLEFIT in the the examples folder. While I am working to improve and expand this manual, I'd recommend you take a look at those worked examples to get started.

4 NOBLEFIT tools reference

This section lists all NOBLEFIT functions and describes their functionality.

4.1 nf_atmos_gas

```
[C_atm,v_atm,D,M_mol,H] = nf_atmos_gas (gas,T,S,p_atm,year,
   hemisphere)


Returns dissolved gas concentrations in air-saturated water,
    volumetric gas content in dry air and molecular
   diffusivity in water.
Concentrations are calculated as gas amount (ccSTP) per mass
    of water (g) at temperature T and salinity S

INPUT:
T:       temperature of water in deg. C
S:       salinity in per mille (g/kg)
p_atm: total atmospheric air pressure, including water
   vapour (hPa, which is the same as mbar)
gas:      'He', 'He-3', 'He-4' (after Weiss)
```

```
              'RHe' (3He/4He ratio)
              'Ne', 'Ne-20', 'Ne_20','Ne-22', 'Ne_22' (after
                 Weiss, isotope fractionation from Beyerle)
              'Ar', 'Ar-36', 'Ar_36', 'Ar-40', 'Ar_40' (after
                 Weiss, isotope fractionation from Beyerle)
              'Kr', 'Kr-78', 'Kr_78', 'Kr-80', 'Kr_80', 'Kr-82',
                 'Kr_82', 'Kr-84', 'Kr_84', 'Kr-86', 'Kr_86 (
                 after Weiss)
              'Xe', 'Xe-124', 'Xe_124', 'Xe-126', 'Xe_126', Xe
      -128', 'Xe_128', 'Xe-129', 'Xe_129', 'Xe-130', 'Xe_130', '
     Xe-131', 'Xe_131', 'Xe-132', 'Xe_132', 'Xe-134', 'Xe_134',
       'Xe-136', 'Xe_136' (after Clever)
              'SF6'
              'CFC11', 'CFC12', 'CFC113'
              'O2', 'O2-34', 'O2-35', 'O2-36'
              'N2', 'N2-28, 'N2-29', 'N2-30'
  year:    year of gas exchange with atmosphere (calendar
     year, with decimals; example: 1975.0 corresponds to 1.
     Jan of 1975.098 corresponds to 5. Feb. 1975, etc.). This
     is only relevant for those gases with time-variable
     partial pressures in the atmosphere (e.g. CFCs, SF6)
  hemisphere: string indicating hemisphere (one of 'north', '
     south', or 'global'). If the hemisphere argument is not
     specified, hemisphere = 'global' is used.


  OUTPUT:
  C_atm: concentration in air-saturated water (ccSTP/g)
  v_atm: volume fraction in dry air
  D:        molecular diffusivities (m^2/s)
  M_mol:   molar mass of the gas (g/mol), values taken from
     http://www.webqc.org/mmcalc.php
  H:        Henrys Law coefficient in (hPa/(ccSTP/g)), as in p
     * = H * C_atm, where p* is the partial pressure of the
     gas species in the gas phase

  EXAMPLES:
  1. To get the Kr ASW concentration (ccSTP/g) in fresh water
     (temperature = 7.5 deg.C) at atmospheric pressure of 991
     hPa:
  [C_atm,v_atm,D,M_mol,H] = nf_atmos_gas ('Kr',7.5,0,991);
     C_atm


  2. To get the SF6 ASW concentration (ccSTP/g) in mid-1983
     northern hemisphere in fresh water (temperatures of 0-10
     deg.C, salinity 6 g/kg) at atmospheric pressure of 983
     hPa:
  [C_atm,v_atm,D,M_mol,H] = nf_atmos_gas ('SF6'
     ,[0:10],6,983,1983.5,'north'); C_atm
```

4.2 nf_objfun

```
G = nf_objfun (PF,P0,PFmin,PFmax,mdl,X_val,X_err)
```

Objective function for mimization in parameter regression.
   Determines the modelled values using the model for the
   given parameter values, and calculates the chi^2 value
   from the difference to the data values. If called with
   fit parameter values that exceed the limits in PFmin and
   PFmax, the chi^2 value will be calculated such that the
   chi^2 minimizer will look elsewhere for a 'better'
   optimum (see commented code for details).

```
INPUT:
PF: vector of fitted model parameter values
P0: vector of constant model parameter values
PFmin: vector of minimum values allowed for the fitted
   parameters
PFmax: vector of maximum values allowed for the fitted
   parameters
mdl: string containing call to the model function using PF
   and P0
X_val: values of observed/measured data (rows correspond to
   samples, columns correspond to one tracers)
X_err: standard errors of X_val

OUTPUT:
G: chi^2 value
```

4.3 nf_read_datafile

```
[data,tracers] = nf_read_datafile (file,options);
```

Reads data from a formatted text file. The data needs to be
   organized in columns as follows:
– Columns are assumed to be separated by tabs. Other
   delimiters may be specified using 'options'.
– The first line must be a header line with names of the
   data in the columns.
– The first column must contain sample names (treated as
   string).
– The remaining columns must contain the data values (either
    numbers, NA, NaN, or empty).
– If column titles include units (or anything else) in
   parentheses, the parentheses part is removed from the
   name (it may be useful to have the units in the data file
   , but the unit will be in the way for data formatting)

6

- Data columns containing the data uncertainties (errors)
  are identified by adding 'err' somehere in the title.
  Example: if the Ne concentrations are given in column
  with title 'Ne', the column title of the corresponding
  errors could be 'Ne␣err', 'err.␣Ne', 'Ne_err', etc.

```
INPUT:
file: file name, may include path to file (string)
options (optional): struct to provide options. May be useful
    to provide details about the format of the data file,
    may be useful to specify special file formats (e.g.
    ouptut from 4D database or input files for Franks noble
    fitter. Known options:
- options.replace_zeros: if set, replace data values equal
    to zero by opt_replace_zeros (scalar)
- options.filter_InvIsotopeRatios: if set to non-zero value
    (scalar), try to make sure that isotope ratios are such
    that the low-mass isotope is divided by the high-mass
    isotope (e.g., replace 40Ar/36Ar by 36Ar/40Ar).

OUTPUT:
data: array of structs with data values and corresponding
    errors. Every struct corresponds to one line in the data
    file. Fieldnames correspond to the column titles in the
    header line. Sample names are stored in field 'name'.
```

4.4 noblefit

```
[par_val,par_err,chi2,DF,pVal,cov,res] = noblefit (model,
    tracer_data,tracers,par_usage,par0,par_norm,par_range,
    par_min,par_max);
```

```
Frontend to fit a given model to observed data using chi^2
    regression.
```

```
INPUT:
model: model name (string). More specifically, this is the
    name of the function that returnscan either be the name
    of one of the standard models provided with gasfit, or a
    custom model function provided by the user (function name
     must be on the search path).
tracer_data: either the name of an ASCII file containing the
     data (with column headers that correspond to the tracer
    names) or a struct variable containing the observed data
    (either a single struct corresponding to one sample, or a
     vector of structs for more than one sample). The tracer
    names must correspond to those in the model function.
```

tracers: a cell string containing the names of the tracers
that are to be used in the fit (names must correspond to
those used by the model function).
par_usage: vector indicating usage of the model parameter
values. par_usage is of the same format as the vector
used as the input argument of the model function. Values
are as follows:
 – par_usage(i) = 0: The i-th model parameter is used as a
   constant in the minimization problem, i.e., it is not
   optimized during fitting of the model to the data.
 – par_usage(i) = 1: the i-th parameter is optimized to
   obtain the best model fit for each individual sample.
 – other values may be implemented in a later version (e.g
   ., for ensemble fits of a model parameter to an
   ensemble of data from multiple samples)
par0: parameter values used for the regression (vector of
the same format as used for the input argument of the
model function). Depending on the parameter usage (see
par_usage), the values are used as fixed values or
initial values used in the minimization problem.
par_norm (optional): typical scale of variation of the
parameter values, used to normalise fitting parameters
during model fitting (vector or matrix of sime size as
par0, values used for fit parameters must not be zero).
Note that the scaling factors reflect the RANGE OF
VARIATION, not the absolute value. For instance, if
infiltration date is somewhere between 1950 and 2013, a
suitable scaling factor would be 10, not 1000.
par_min (optional): min allowed parameter values for fit (
vector or matrix of same size as par0, only values for
fitted parameters will be used). Values may be -Inf to
indicate no limits. Note: fit results may sightly exceed
the limits, if the best fit value is close to the limit.
This is due to the way the limits are treated in the
fitting routine. The effect should be small, but please
make sure the values are ok for you.
par_max (optional): max allowed parameter values for fit (
vector or matrix of same size as par0, only values for
fitted parameters will be used). Values may be Inf to
indicate no limits. Note: fit results may sightly exceed
the limits, if the best fit value is close to the limit.
This is due to the way the limits are treated in the
fitting routine. The effect should be small, but please
make sure the values are ok for you.

OUTPUT:
par_val: best-fit estimates of the parameter values (vector
of the same format as used for the input argument of the

```
    model function).
par_err: standard errors of par_val (vector).
chi2: chi2 of fit
DF: degrees of freedom of fit ( = number of data points -
    number of fitted model parameters)
pVal: p-value of chi2, as given from cumulative chi2-density
     function with DF degrees of freedom: pVal = 1 - chi2cdf(
    chi2,DF)
cov: covariance matrices of the fits (cell array of matrices
    )
res: error-weighted residuals of observed values (X) of
    sample k relative to modeled values (M), normalised by
    the standard error (E) of the observed values, i.e.: res(
    k,i) = (X_i - M_i) / E_i, where i is an index to the
    corresponding tracer (res is a matrix, each row
    corresponds to one sample, columns correspond to 'tracers
    ')

EXAMPLES:
see files in Examples folder.
```

References

[1] W. Aeschbach-Hertig, F. Peeters, U. Beyerle, and R. Kipfer. Interpretation of dissolved atmospheric noble gases in natural waters. *Water Resour. Res.*, 35(9):2779–2792, 1999.

[2] W. H. Press, P. F. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes*. Cambridge University Press, Cambridge, London, New York, 1986.