starfit

Conrad Chan, Alexander Heger, Thomas Nordlander, David Liptai

Jun 16, 2023

CONTENTS:

Python package for matching stellar abundance measurements against a database of model stellar explosions. Based on the old IDL code by Alexander Heger.

StarFit can match combined abundances of multiple models. For single stars and combinations of multiple stars, a complete search can be found. For three or more stars, the problem is extremely expensive, so a Genetic Algorithm has been implemented by Conrad Chan to efficiently find an approximate solution.

An online interface (with a subset of functionality) is available at starfit.org.

CHAPTER

ONE

INSTALLATION

Tested with Python 3.10, 3.11

Optional: A working LaTeX installation and dvipng is required to create plots with LaTeX labels (ideal for publication). Otherwise, Matplotlib's default MathText is used, which may not render all symbols correctly.

1.1 From PyPI (recommended)

```
pip install starfit
```

The PyPI package includes the necessary data files.

1.2 Developer instructions

The data files are not included into the Git repo, and must first be downloaded from the web-server (starfit.org/data) before installing from the Git repo.

```
git clone git@github.com:conradtchan/starfit.git
cd starfit

# Download data files
./download-data.sh

# "-e" creates an editable install, "[testing]" installs additional dependencies for␣
↪testing
pip install -e ".[testing]" --no-build-isolation

# Run all tests
python -m pytest
```

CHAPTER

TWO

USAGE

2.1 Single star matches

starfit.Single fits an abundance pattern to a single model from the database. Example usage:

```
import starfit

s = starfit.Single(
    filename = 'HE1327-2326.dat',
    db = 'znuc2012.S4.star.el.y.stardb.gz',
    combine = [[6, 7, 8]],
    z_max = 30,
    z_exclude = [3, 24, 30],
    z_lolim = [21, 29],
    upper_lim = True,
    cdf = True,
    constraints = 'energy <= 5',
    )

s.print()
```

The *StarFit.print()* method allows to specify the number of lines to be printed (`n`), the offset for the first entry to print (`n0`, default is `0`) and the maximum number of columns to use as a "wide" table (`wide`, default `12`). A `format` and be specified as `"html"` or `"unicode"`, plain text otherwise. Default is `unicode`.

```
s.print(n0=3, n=1, wide=8, format=None)
```

The *StarFit.info()* method allows to print information about individual table entries, starting with index `0` (default). For example, to print the third model info use

```
s.info(2)
```

The **database indices** of the best fitting models (sorted from best to worst) are given by:

```
s.sorted_stars['index']
```

The corresponding **reduced** ^{2} values are:

```
s.sorted_fitness
```

The physical properties of the models corresponding to these indices can be accessed using the database:

```
i_bestfit = s.sorted_stars['index'][0]
s.db.fielddata[i_bestfit]
```

The chemical yield of the models (and the respective element names) are:

```
list(zip(s.list_db, s.full_abudata[:, i_bestfit]))
```

2.1.1 Plots

StarFit.plot() can be used to make the same plots as the web version:

```
s.plot()
```

If you want to plot a solution other than the best one, use the parameter `num` (default: `0`) To plot the 5th best solution, skipping the first `4`, use

```
s.plot(num=4)
```

The legend as well as the star name and copyright string can be moved (dragged).

2.2 Full multi-star search

`starfit.Multi` fits an abundance pattern to a combination of models from the database(s). This can take a long time as there can be many combinations.

Additional arguments:

- `fixed_offsets`: Use dilution factors based on the ejecta mass, rather than solving for the optimal dilution ratio of each explosion independently (decreases solve time)
- `threads`: Number of threads to use. Default is to use the CPU count (including hyper-threading)
- `nice`: Nice level of background threads. Default is 19 (lowest priority on Unix systems).
- `group`: by default, all data are merged in one big list and all possible combinations (excluding duplicates) are explored. If `group` is specified, only combinations form different databases are considered. This can significantly reduce the cost and often may be more what is intended. In this case, the number of data base partitions needs to match the number of stars (`sol_size`). `group` can be a vector with number of data bases to group into each group. The number of groups needs match the `sol_size` vector.

Changed arguments:

- sol_size can now be a vector with one entry for each partition. The number of entries need to match the number of groups. A scalar value is equivalent to vector with that many 1s. All combinations in each group (without repetitions) are tested.

```
s = starfit.Multi(
    filename = 'SMSS2003-1142.dat',
    db = (
        'he2sn.HW02.star.el.y.stardb.gz',
        'rproc.just15.star.el.y.stardb.xz',
        'rproc.wu.star.el.y.stardb.xz',
        ),
    z_max = 999,
```

(continues on next page)

(continued from previous page)

```
    z_exclude = [3, 24, 30],
    z_lolim = [21, 29],
    upper_lim = True,
    cdf = True,
    fixed_offsets = False,
    sol_size = [2,1],
    group = [1,2],
    )
```

2.3 Genetic algorithm

`starfit.Ga` fits an abundance pattern to a combination of two or more models from the database. The solution is approximate, but approaches the best solution with increased run time.

Additional arguments:

- `gen`: maximum number of generations (iterations) to search; no limit if `0` or `None` (default: `1000`).
- `time_limit`: maximum amount of time (in seconds) to search for solution. Infinite if `None` (default: `20 s`).
- `sol_size`: number of nucleosynthesis models to combine for the solution (default: `2`).
- `pop_size`: GA parameter - number of solutions in the population (default: `200`).
- `tour_size`: GA parameter - number of solutions per tournament selection (default: `2`).
- `frac_mating_pool`: GA parameter - fraction of solutions in the mating pool (default: `1`).
- `frac_elite`: GA parameter - top fraction of elite solutions (default: `0.5`).
- `mut_rate_index`: GA parameter - mutation rate of the star index in the databases (default: `0.2`).
- `mut_rate_offset`: GA parameter - mutation rate of the dilution factor (default: `0.1`).
- `mut_offset_magnitude`: GA parameter - size of the mutation of the dilution factor (default `1`).
- `local_search`: GA parameter - solve for the best dilution factors rather than relying on the GA (default: `True`).
- `spread`: GA parameter - ensure no database sources are skipped unless there are fewer stars than data bases. This can be useful if there is a large disparity in the number of models between the different data bases and if you have a prior that all data bases should be used. Eventually, the genetic algorithm should find all combinations that match best anyway, however.
- `group`: grouping of data bases, for use with `spread`: try to cover each group but not each database within it separately. Provide a vector of group length or of tuples with database indices (`0`-based), no duplications allowed. Same rules as above apply: if group is specified, you need to a provide grouping that covers each database listed by index.
- `pin`: number or list of groups to require to be included. Repetitions are allowed to enforce multiple selections from that group.

The default GA parameters should be used unless you really know what you are doing.

```
s = starfit.Ga(
    filename = 'HE1327-2326.dat',
    db = (
        'rproc.just15.star.el.y.stardb.xz',
        'znuc2012.S4.star.el.y.stardb.gz',
```

(continues on next page)

(continued from previous page)

```
        'rproc.wu.star.el.y.stardb.xz',
        ),
    combine = [[6, 7, 8]],
    z_max = 30,
    z_exclude = [3, 24, 30],
    z_lolim = [21, 29],
    upper_lim = True,
    cdf = True,
    time_limit = 20,
    sol_size = 2,
    spread = True,
    group=[[0,2],[1]]
    )
```

The execution can be terminated pressing the <Enter> key.

2.3.1 Evolutionary History

The history of fitness evolution can be plotted using the `plot_fitness` method.

Additional arguments:

- `gen`: when set to `True`, plot as a function of generation number. Otherwise, plot as a function of computational time (default).

```
s.plot_fitness(gen=True)
```

2.4 Matching specific star combinations

`starfit.Direct` allows to find the best fit to pre-selected group, or groups of stars.

Additional arguments:

- `stars`: Nested list of lists of models. For each model, specify a list of database and index. Both, the database index and the star index, are `0`-based.
- `offsets`: Nested list of offsets . For each model, specify a list of offsets. If not provided, a default (starting) value (`1e-4` total) will be assumed.
- `optimize`: Whether to find best matching offset or use offsets as is (default: `True`).

The following selects two groups of models: the first selects model index `0` from the first database with index `0`, and the second model (index `1`) from the second database (index `1`); the second selects the third model (index `2`) from the first database (index `0`) and the fourth model (index `3`) from the second database (index `1`):

```
s = starfit.Direct(
    filename = 'HE1327-2326.dat',
    db = (
        'he2sn.HW02.star.el.y.stardb.gz',
        'rproc.just15.star.el.y.stardb.xz',
        ),
    stars = [
        [[0,0], [1,1]],
```

(continues on next page)

(continued from previous page)

```
        [[0,2], [1,3]]],
    )
```

The results are sorted by fitness and stored in the returned object as usual, allowing to print and plot the results.

2.5 Multiple databases info

By default, data bases are numbered in the order provided. The database numbers are only listed when there is more than one database provided. Full database information can be printed using the `print_comments` method of the solution object:

```
s.print_comments()
```

or if the `full` parameter is specified to the `print` method

```
s.print(full=True)
```

2.6 Error matrix plots

The `StarFit` object provide three functions analyse error and plot error contributions.

2.6.1 Error matrix of data

`plot_star_matrix` plots the error as computed/used by the fitter.

Arguments are

- `zoom`: How much to zoom in around zero. Use `False` to disable. (default: `1000`)
- `nlab`: How many labels to draw on the colorbar (default: `9`).
- `compress`: Whether to skip elements for which there are no measurements (default: True).

2.6.2 Inverse of error matrix of data

`plot_star_inverse` plots the inverted error matrix as computed/used by the fitter.

Arguments are

- `zoom`: How much to zoom in around zero. Use `False` to disable. (default: `0.1`)
- `nlab`: How many labels to draw on the colorbar (default: `9`).
- `compress`: Whether to skip elements for which there are no measurements (default: True).

2.6.3 Error contributions as computed

`plot_error_matrix` plots the error contributions as computed by the fitter for a given star.

Arguments are

- `num`: Number of solution to plot, counted from the best (default: `0`).
- `zoom`: How much to zoom in around zero. Use `False` to disable. (default: `1000`)
- `nlab`: How many labels to draw on the colorbar (default: `9`).

CHAPTER

THREE

CUSTOM DATA DIRECTORY

Custom stellar data and model database files can always be used by providing a full path in the argument. However, users may optionally specify their own data directory using the environment variable `STARFIT_DATA` for convenience:

```
export STARFIT_DATA='/your/custom/data'
```

Files found in the custom data directory will take precedence over the default data directory. Your custom data directory must have the same structure as `src/starfit/data`, i.e. it should contain the `db`, `ref`, and `stars` directories:

```
 ls
db
ref
stars
```

CHAPTER

FOUR

CONTRIBUTING TO STARFIT

Contributions to the StarFit code are welcome. The `main` branch is protected and cannot be committed to directly. Instead, please create a Pull Request with your proposed contributions. To make a new branch and set to track `origin`

```
git checkout -b <new_branch>
git push --set-upstream origin <new_branch>
```

1. If you changed the Fortran code and want to test locally, remember to re-compile or install as editable packages:

```
# "-e" creates an editable install, "[testing]" installs additional dependencies for␣
↪testing
pip install -e ".[testing]" --no-build-isolation
```

To make this step more convenient we provide a `Makefile` in the root directory that does all three steps:

```
make
```

If there are issues with the `fitness` sub-module, there is a `Makefile` in its source directory that can be used to compile a test program outside of the python package build process.

Two automated checks (on Github Actions) must be passed (Items 2 and 3):

2. Code formatting using pre-commit. To ensure your changes are compliant with this project's linters, we recommend installing pre-commit prior to making any commits locally.

```
pip install pre-commit
pre-commit install
```

If you have already made non-compliant commits prior to installing pre-commit, then the pre-commit check on `GitHub` will fail. To make the code compliant again, run

```
pre-commit run --all
```

3. Code tests using `pytest`. New tests can be added to the `tests/` directory.

Run these tests as a check

```
python -m pytest
```

and include any necessary changes in the commit.

4.1 Development branch

Development branches are generated and uploaded to Test PyPI if the version number ends in `.dev*` where `*` can be blank or a optional number. For example, '`0.3.11.dev22`. They may also be flagged as pre-releases.

To install packages from Test PyPI use

```
pip install --index-url https://test.pypi.org/simple/ --extra-index-url https://pypi.org/
↪simple/ starfit
```

You may include the `-pre` flag or specify a specific version.

4.2 Adding new database files

Database files specified in the `.hashlist` files in `src/starfit/data/db,ref,stars` are downloaded from the web server. To add new data files:

1. Add the new files to the web server hosting the data files at `/var/www/html/data`
2. Generate the hash using `shasum -a 256` (or `sha256sum`)
3. Add an entry into the hash list

When adding new databases into `data/db`, add corresponding labels into the file `data/db/labels` and a description of the data base into the file `data/db/databases` on the web server.

4.3 Creating database files

New database files can be made using the `StarDB` class in `autils/stardb.py`. A demonstration may be found at `src/starfit/example/lc12_stardb.py`. This file serves as a demonstration only and will not work as is.

CHAPTER

FIVE

PUBLISHING TO PYPI

Github releases will automatically be published to pypi.org/project/starfit/

CHAPTER

SIX

API

6.1 starfit package

6.1.1 Subpackages

starfit.fitness package

Submodules

starfit.fitness.solver module

starfit.fitness.solver.**check_offsets**(*offsets*, *flags*, *tag*)

starfit.fitness.solver.**fitness**(*observed*, *error*, *detection*, *covariance*, *abu*, *offsets*, *cdf=None*, *dst=None*, *local_search=True*, *limit_solution=None*, *limit_solver=None*, *return_matrix=False*)

starfit.fitness.solver.**get_complete_inverse**(*star*, *cdf*)

starfit.fitness.solver.**get_complete_matrix**(*star*, *cdf*)

Module contents

starfit.solgen package

Module contents

6.1.2 Submodules

6.1.3 starfit.direct module

find best fot for a selection of stars

class starfit.direct.**Direct**(**args*, *stars=None*, *offsets=None*, *fixed_offsets=False*, *optimize=True*, ***kwargs*)

Bases: *StarFit*

Find the best fit for stars you specify

when multiple DBs are specified, provide tuples of (DB, index) where DB is 1-based

6.1.4 starfit.fit module

Fitting stars

starfit.fit.**gen_map**(*gen_start*, *gen_end*, *num*, *size=None*, *com=None*)

starfit.fit.**get_fitness**(*trimmed_db*, *eval_data*, *z_exclude_index*, *sol*, *ejecta=[]*, *fixed_offsets=False*, *local_search=True*, ***kwargs*)

Evaluate the fitness of a set of solutions. If abundance is directly given in the case of smart GA, use that.

starfit.fit.**get_solution**(*gen_start*, *gen_end*, *exclude_index*, *trimmed_db*, *fixed_offsets=False*, *ejecta=None*, *return_size=None*, *sol_size=None*, *num=None*, *size=None*, *com=None*, *index=None*, ***kwargs*)

Local search solve for a set of stars. Used for splitting into different processes. This needs to be a separate function that can be pickled.

6.1.5 starfit.ga module

Genetic algorithm

class starfit.ga.**Ga**(**args*, *gen=1000*, *time_limit=20*, *pop_size=200*, *sol_size=None*, *tour_size=2*, *frac_mating_pool=1*, *frac_elite=0.5*, *mut_rate_index=0.2*, *mut_rate_offset=0.1*, *mut_offset_magnitude=1*, *local_search=True*, *fixed_offsets=False*, *seed=None*, *max_pop=8192*, *spread=None*, *group=None*, *pin=None*, *n_top=20*, *interactive=True*, ***kwargs*)

Bases: *StarFit*

Genetic Algorithm.

plot_fitness(*gen=False*, *fig=None*, *ax=None*, *show_copyright=True*, *figsize=(10, 6)*, *dpi=102*)

plot fitness as a function of time

print_empty_table()

print_header()

print_update()

6.1.6 starfit.multi module

class starfit.multi.**Multi**(**args*, *n_top=10*, *fixed_offsets=False*, *threads=None*, *nice=19*, *save=False*, *webfile=None*, *path=None*, *block_size=131072*, *sol_size=None*, *group=None*, ***kwargs*)

Bases: *StarFit*

Find the best fit for multiple stars (complete search). The search results are printed live.

group:
if number of data bases matches sol_size, take one from each DB

sol_size:

None:
if group is True, set sol_size to number of DBs

TODO - allow multiple contributions form one group
(sol_size becomes vector with length of number of groups)

init_futures(*threads=None*, *nice=19*)

print_empty_table()

print_header()

print_update()

save_file(*file_name*)

working_arrays()

6.1.7 starfit.single module

class starfit.single.**Single**(**args*, ***kwargs*)

Bases: *StarFit*

Find the best fit for single stars (complete search)

Example:

```
import starfit

s = starfit.Single(
    filename = 'HE1327-2326.dat',
    db = 'znuc2012.S4.star.el.y.stardb.gz',
    combine = [[6, 7, 8]],
    z_max = 30,
    z_exclude = [3, 24, 30],
    z_lolim = [21, 29],
    upper_lim = True,
    cdf = True,
    constraints = 'energy <= 5',
    )
```

sol_size = 1

Single has `sol_size` hardcoded to 1. All other arguments are inherited from *StarFit*

6.1.8 starfit.star module

Reads elemental abundance file into an array File format:

Version 100200

Add tetection threshold. Otherwise identical to and backward compatible with Format 10100

10200 <— VERSION NUMBER SDSS12345-6789 <– STAR NAME ApJ 222, 333 (1999) <– SOURCE/REFERENCE data automatically generated by Anna Frebel <– SOME COMMENT 1 1 <— DATA FORMAT, MODE (DETECTION THRESHOLD PRESENT = 1)

<— SYMBOL OF ELEMENT OR VALUE TO WHICH DATA IS NORMALIZED, IF ANY

17 <— NUMBER OF ELEMENTS C -1.23 +0.07 -1.07 +0.05 -0.05 N 1.23 +0.14 -2.14 -0.10 +0.10 O -1.23 +0.21 -3.21 -0.15 -0.15 <— MEASUREMNT, Fe 1.23 +0.28 -4.28 +0.20 +0.20 UNCORRELATED ERROR, Mg -1.23 +0.35 -5.35 +0.25 -0.25 DETECTION THRESHOLD, Si 1.23 +0.42 -6.42 +0.30 +0.30 COVARIANCE VECTORS Sc 1.23 +0.57 -7.57 -0.35 -0.35 Ca -1.23 +0.49 -8.49 Ti -1.23 +0.64 -9.64 V +1.23 +0.64 -8.64 <— MEASUREMNT, Cr 1.23 +0.10 -6.64 UNCORRELATED ERROR, Mn -1.23 +0.20 -6.64 DETECTION THRESHOLD Co -1.23 -0.30 Ni -1.23 -0.40 Cu 1.23 -0.50 Sr -1.23 -0.60 <— UPPER LIMIT, Zr 1.23 -0.70 (UNCORRELATED) UNCERTAINTIES Ba -1.23 -0.80 Lo09 <— SOLAR REFERENCE; USE "-" IF NOT PROVIDED

DATA FORMAT 0 Y (mol/g), error is absolute 1 log epsilon (PREFERRED DUE TO DIFFERENT SOLAR ABUNDANCES, give H mol/g as norm!) 2 [ ] 3 [X/Y] Y= norm element, provide [Y/H] in column for Y 4 log X/Si + 6 (by number), norm is assumeed log(Si) + 6 (mol/g) 5 log mole fraction (REALLY PREFERRED) 6 [X/H] - provide [H] as norm, otherwise BBN value is assumed 7 X/Si * 1e6 (by number), norm is Si/Si_sun, error is absolute

The first number is the data, the second is the uncorrelated (random)error in dex. Use negative random error values to indicate upper limits.

The 3rd column is the detection threshold, if present. May be "-" if missing.

The following columns give first correlated errors.

The covariances can have positive or negative signs, the uncorrelated (random) error always is inherently non-negative.

The square of the total error is the sum of the squares of all errors.

If no covariances are provide, the entire error is random.

As of the time of this writing, StarFit does not use correlated upper limits, but that may change in the future.

Version 10100

Adds covariant errors, format compatible with Version 10002

10100 <— VERSION NUMBER SDSS12345-6789 <– STAR NAME ApJ 222, 333 (1999) <– SOURCE/REFERENCE data automatically generated by Anna Frebel <– SOME COMMENT 1 <— DATA FORMAT (RECOMEND USE LOG EPSILON OR LOG MOL FRACTION)

<— NAME OF ELEMENT OR VALUE TO WHICH DATA IS NORMALIZED, IF ANY

17 <— NUMBER OF ELEMENTS C -1.23 +0.07 +0.05 -0.05 N 1.23 +0.14 -0.10 +0.10 <— MEASUREMNT, O -1.23 +0.21 -0.15 -0.15 UNCORRELATED ERROR, Fe 1.23 +0.28 +0.20 +0.20 COVARIANCE VECTORS Mg -1.23 +0.35 +0.25 -0.25 Si 1.23 +0.42 +0.30 +0.30 Ca -1.23 +0.49 -0.35 -0.35 Sc 1.23 +0.57 Ti -1.23 +0.64 Cr 1.23 +0.10 <— MEASUREMENT, Mn -1.23 +0.20 UNCORRELATED ERRORS Ni -1.23 -0.40 Cu 1.23 -0.50 Sr -1.23 -0.60 <— UPPERL LIMIT, Zr 1.23 -0.70 (UNCORRELATED) UNCERTAINTIES Ba -1.23 -0.80 Lo09 <— SOLAR REFERENCE; USE "-" IF NOT PROVIDED

DATA FORMAT 1 log epsilon (PREFERRED DUE TO DIFFERENT SOLAR ABUNDANCES, give H mol/g as norm!) 2 [ ] 3 [X/Y] Y= norm element, provide [Y/H] in column for Y 4 log X/Si + 6 (by number), norm is assumeed log(Si) + 6 (mol/g) 5 log mole fraction (REALLY PREFERRED) 6 [X/H] - provide [H] as norm, otherwise BBN value is assumed 7 X/Si * 1e6 (by number), norm is Si/Si_sun, error is absolute

The first number is the data, the second is the uncorrelated error in dex. Use negative random error values to indicate upper limits.

The following columns give first covariant errors.

The covariant errors can have positive or negative signs, the uncorrelated (random) error always is inherently non-negative.

The square of the total error is the sum of the squares of all errors.

If no covariances are provide, the entire error is random.

As of the time of this writing, StarFit does not use correlated upper limits, but that may change in the future.

Version 100002

10002 <— VERSION NUMBER (FILE FORMAT) CS30336-049 <– STAR NAME ApJ 222, 333 (1999) <– SOURCE/REFERENCE data automatically generated by Anna Frebel <– SOME COMMENT 1 <— DATA FORMAT (USE LOG EPSILON OR LOG MOLE FRACTION)

<— NORM, IF ANY (ELEMENT OR ABUNDANCE OFFSET)

17 <— NUMBER OF ELEMENTS C -1.23 0.1 N 1.23 0.2 O -1.23 0.3 Fe 1.23 0.4 <— MEASUREMENT, Mg -1.23 0.5 UNCORRELATED ERRORS Si 1.23 0.6 Ca -1.23 0.7 Sc 1.23 0.8 Ti -1.23 0.9 Cr 1.23 -0.1 Mn -1.23 -0.2 Co 1.23 -0.3 Ni -1.23 -0.4 <— UPPERL LIMIT, Sr -1.23 -0.6 (UNCORRELATED) UNCERTAINTIES Zr 1.23 -0.7 Ba -1.23 -0.8 Lo09 <— SOLAR REFERENCE; USE "-" IF NOT PROVIDED

DATA FORMAT

1 log epsilon (PREFERRED DUE TO DIFFERENT SOLAR ABUNDANCES, give H mol/g as norm!) 2 [ ] 3 [X/Y] Y = norm element, provide [Y/H] in column for Y 4 log X/Si + 6 (by number), norm is assumeed log(Si) + 6 (mol/g) 5 log mole fraction (REALLY PREFERRED) 6 [X/H] - provide [H] as norm, otherwise BBN value is assumed 7 X/Si * 1e6 (by number), norm is Si/Si_sun, error is absolute

The first number is the data, the second is the error in dex. Use negative values to indicate upper limits; all need to be non-zero.

Version 100001

10001 <— VERSION NUMBER (FILE FORMAT) CS30336-049 <– STAR NAME ApJ 222, 333 (1999) <– SOURCE/REFERENCE data automatically generated by Anna Frebel <– SOME COMMENT 1 <— DATA FORMAT (USE LOG EPSILON OR LOG MOLE FRACTION)

<— NORM, IF ANY (ELEMENT OR ABUNDANCE OFFSET)

17 <— NUMBER OF ELEMENTS C -1.23 0.1 N 1.23 0.2 O -1.23 0.3 Fe 1.23 0.4 <— MEASUREMENT, Mg -1.23 0.5 UNCORRELATED ERRORS Si 1.23 0.6 Ca -1.23 0.7 Sc 1.23 0.8 Ti -1.23 0.9 Cr 1.23 -0.1 Mn -1.23 -0.2 Co 1.23 -0.3 Ni -1.23 -0.4 <— UPPERL LIMIT, Sr -1.23 -0.6 (UNCORRELATED) UNCERTAINTIES Zr 1.23 -0.7 Ba -1.23 -0.8

DATA FORMAT

1 log epsilon (PREFERRED DUE TO DIFFERENT SOLAR ABUNDANCES, give H mol/g as norm!) 2 [ ] 3 [X/Y] Y = norm element, provide [Y/H] in column for Y 4 log X/Si + 6 (by number), norm is assumeed log(Si) + 6 (mol/g) 5 log mole fraction (REALLY PREFERRED) 6 [X/H] - provide [H] as norm, otherwise BBN value is assumed 7 X/Si * 1e6 (by number), norm is Si/Si_sun, error is absolute

The first number is the data, the second is the error in dex. Use negative values to indicate upper limits; all need to be non-zero.

Version 100000

10001 <— VERSION NUMBER (FILE FORMAT) CS30336-049, data from a friend <– STAR NAME AND COMMENT 1 <— DATA FORMAT (USE LOG EPSILON OR LOG MOLE FRACTION)

<— NORM, IF ANY (ELEMENT OR ABUNDANCE OFFSET)

17 <— NUMBER OF ELEMENTS C -1.23 0.1 N 1.23 0.2 O -1.23 0.3 Fe 1.23 0.4 <— MEASUREMENT, Mg -1.23 0.5 UNCORRELATED ERRORS Si 1.23 0.6 Ca -1.23 0.7 Sc 1.23 0.8 Ti -1.23 0.9 Cr 1.23 -0.1 Mn -1.23 -0.2 Co 1.23 -0.3 Ni -1.23 -0.4 <— UPPERL LIMIT, Sr -1.23 -0.6 (UNCORRELATED) UNCERTAINTIES Zr 1.23 -0.7 Ba -1.23 -0.8

DATA FORMAT

1 log epsilon (PREFERRED DUE TO DIFFERENT SOLAR ABUNDANCES, give H mol/g as norm!) 2 [ ] 3 [X/Y] Y = norm element, provide [Y/H] in column for Y 4 log X/Si + 6 (by number), norm is assumeed log(Si) + 6 (mol/g) 5 log mole fraction (REALLY PREFERRED) 6 [X/H] - provide [H] as norm, otherwise BBN value is assumed 7 X/Si * 1e6 (by number), norm is Si/Si_sun, error is absolute

The first number is the data, the second is the error in dex. Use negative values to indicate upper limits; all need to be non-zero.

class starfit.star.**Star**(*filename*, *silent=False*)

Bases: Logged

All the data read from each file

get_covariances()

get_detection_thresholds()

get_elements()

return all elements in dataset

get_input_data_format()

get_measured()

get_n_covariances()

get_norm()

get_upper_limits()

list_to_array(*data_list*)

exception starfit.star.**StarFileError**

Bases: Exception

6.1.9 starfit.starfit module

Results objects from the various algorithms

exception starfit.starfit.**ConstraintsParseError**

Bases: Exception

class starfit.starfit.**Convert_from_5**(*starfit*, *yscale=2*, *ynorm=None*, *ylabel=None*)

Bases: object

err(*err*)

property ylabel

class starfit.starfit.**StarFit**(**args*, *silent=False*, ***kwargs*)

Bases: Logged

Base class for running the various fitting algorithms.

Parameters

- **filename** (*str*) – Filename of star. Can be absolute or relative path. The files will also be searched for in the distribution files and in the search path specified by environment variable STARFIT_DATA in subdirectory stars.

- **db** (str or `pathlib.Path`, optional) – database file or tuple of data base files. String or `Path` object. Can be absolute or relative path. Files will also be searched in the distribution files and in the search path specified by environment variable `STARFIT_DATA` in subdirectory `db`. You may also use the wildcard (`"*"`) in the data base name. The code will then try to resolve all matching data bases in the first source directory that contains any matching file. The plain `*` argument will include all data bases in the first source that contains any data base; the matching is done against the pattern `*.stardb.*`. The `Ellipis` (`...` Python object, not in quotation marks) will do the same as the plain `*` argument, but will continue searching through all data souces. This allows for an easy way to search across all model data bases available. combine (list, optional): A list of lists of element charge numbers to treat as combined abundances (e.g. combine the CNO elements using `[[6,7,8]]`).
- **z_min** (*int, optional*) – Lowest element charge number to fit.
- **z_max** (*int, optional*) – Highest element charge number to fit.
- **z_exclude** (*list, optional*) – Element charge numbers to exclude from fit.
- **lim_exclude** (*bool, optional*) – Treat `z_min` and `z_max` limits as exclusions (default: `True`). Otherwise databases are "trimmed" to save memory and data cannot be plotted in interactive mode.
- **z_lolim** (*list, optional*) – Elements that are *model* lower limits (effectively the same as *observational* upper limits).
- **upper_lim** (*bool, optional*) – Include observational upper limits in data fitting.
- **cdf** (*bool, optional*) – Use the uncertainty of upper limits to calculate a cumulative distribution function when calculating error contribution (otherwise treat the upper limit as a simple one-sided χ^2 error).
- **det** (*bool, optional*) – Use the detection limits when calculating error contribution (experimental).
- **cov** (*bool, optional*) – Use the error covariances when calculating error contribution (experimental).
- **dst** (*bool, optional*) – Use statistical error only for detection treshold (default: True; experimental).
- **limit_solver** (*bool, optional*) – Solver/search will only allow solutions for each star that contribute no more than 100%.
- **limit_solution** (*bool, optional*) – Solver/search will only allow solutions where the total adds up to no more than 100% contributions from all stars. Results from the search are renormalised accordingly.
- **y_floor** (*bool, optional*) – Floor value for abundances to assume in models (default: `1e.0e-99`). This is useful for elements not produced in a model, otherwise χ^2; or $-\infty$; may result.
- **db_label** (*list, optional*) – A list of labels for the data bases to be used in plots and tables. Will only be shown if there is more than one database specified. If present, needs to match the number of databases specified. If not present, databases will be numbered starting with `0`, unless the `StarDB` has a `label` field that will be used instead. The maximum label length currently allowed is `8`.
- **show** (*bool, optional*) – Show list of loaded databases with label/number
- **name** (*and*) –
- **quit.** (*then*) –

- **constraints** (*str, optional*) – String with list of conditions separated by comma (acts as "and"). Conditions for specific databases can be prefixed with a number (zero-based) if the index of the database in the list, followed by a colon(`:`). Entries for different databases are separated by semicolon (`;`). The fieldname has to be given first, then the operator, and finally the comparison value. Allowed operators are <, <=, ==, >=, >, and `!=`.
- **constraints_error** (*str, optional*) – one of `warn` (default), `raise`, `ignore`. How StarDB deal with errors in `constraints`.

W()

format(**args*, ***kwargs*)

format_comments(*npad=72*, *dbx=None*)

format_db(*ind=0*, *pad=''*, ***kwargs*)

info(*i=0*, ***kwargs*)

plot(*num=0*, *fig=None*, *ax=None*, *fontsize=None*, *annosize=None*, *figsize=(10, 6)*, *dpi=102*, *show_copyright=True*, *dist=None*, *xlim=None*, *ylim=None*, *data_size=3*, *save=None*, *save_format=None*, *return_plot_data=False*, *yscale=2*, *ynorm='Fe'*, *range_det=False*, *range_lim=True*, *range_zmin=3*, *pad_abu=0.1*, *pad_det=0.05*, *multi=0*, *xlabel=None*, *ylabel=None*)

Create a plot of the solution.

Note: The legend as well as the star name and copyright string can be moved (dragged).

Parameters

- **num** (*int*) – Number of solution, from the top (default: `0`).
- **yscale** (*int*) – select the y-scale of the plot. Numerical value identical to those used for the star data formats.
- **ynorm** (*str*) – elements to use a norm for `[X/Y]` plots (`yscale=3`).
- **multi** (*int*) – plot this many best solutions as grey lines (default: `0`). For `multi=-1` lines will be shaded according to relative data point probability based on χ^2; and assuming multi-dimensional Gaussian error.
- **save** (*str*) – filename to save plot.
- **range_det** (*bool*) – adjust range to include detection thresholds (default: `False`)
- **range_lim** – (bool) adjust range to include detection limits (default: `True`)
- **range_zmin** (*int*) – minimum Z to consider for determining y range (default: `3`)
- **pad_abu** (*float*) – fraction of plot range to use at bottom/top boundary (default: `0.1`).
- **pad_det** (*float*) – fraction of plot range to pad detection thresholds (default: `0.05`).
- **figsize** (*tuple*) – dimensions of figure in inches (default: `(10, 6)`).
- **dpi** (*int*) – resolution of image (default: `102`).
- **xlim** (*tuple*) – overwrite x range (low, high).
- **ylim** (*tuple*) – overwrite y range (low, high).
- **data_size** (*int*) – Size of data lines and symbols (default: `3`).

- **fontsize** (*int*) – size used for axis labels (default: 12).
- **annosize** (*str*) – size used for element symbols (default: small).
- **dist** (*float*) – distance of labels from data points.
- **fig** (:class:matplotlib.figure.Figure) – figure object to use as canvas, otherwise as new figure is created.
- **ax** (:class:matplotlib.axis.Axis) – axis objects to use for drawing, otherwise axis and parent figure are created as needed.
- **xlabel** (*str*) – overwrite label for x-axis.
- **ylabel** (*str*) – overwrite label for y-axis.

plot_error_matrix(*num=0*, *zoom=False*, *nlab=9*)

plot_star_inverse(*zoom=False*, *nlab=9*, *compress=True*)

plot_star_matrix(*zoom=False*, *nlab=9*, *compress=True*)

print(**args*, ***kwargs*)

print_comments(**args*, ***kwargs*)

print_db(**args*, ***kwargs*)

run(*stars*, *offsets=None*, *optimize=True*, ***kwargs*)

Solve for the specified stars

text_db(*dbx=None*, *filename=False*, *fields=False*)

text_result(*n=20*, ***, *n0=0*, *format='unicode'*, *best=True*, *wide=12*, *show_index=True*, *_return_dbx=False*)

Print data of best fit.

static textpad(*s*, *n*)

6.1.10 starfit.starplot module

Various plotting routines

class starfit.starplot.**IntFormatter**(**args*, ***kwargs*)

Bases: FuncFormatter

starfit.starplot.**leg_copyright**(*ax*)

starfit.starplot.**leg_info**(*ax*, *lines*)

starfit.starplot.**leg_starname**(*ax*, *name*)

6.1.11 starfit.utils module

starfit.utils.**find_all**(*subdir*, *pattern*, *complete=False*)

Search through all the data directories for the filename. Return the absolute path of all files - in the first directory one is found if complete is False - in all subdirectories if complete is True Search precedence is: 1) absolute path if provided 2) STARFIT_DATA env var (set in __init__.py) 3) Installation DATA_DIR (set in __init__.py)

starfit.utils.**find_data**(*subdir*, *filename*)

Search through all the data directories for the filename. Return the absolute path if found. Precedence is: 1) absolute path provided 2) STARFIT_DATA env var (set in __init__.py) 3) Installation DATA_DIR (set in __init__.py)

starfit.utils.**getch**()

starfit.utils.**set_priority**(*value: int*)

6.1.12 Module contents

class starfit.**Direct**(**args*, *stars=None*, *offsets=None*, *fixed_offsets=False*, *optimize=True*, ***kwargs*)

Bases: *StarFit*

Find the best fit for stars you specify

when multiple DBs are specified, provide tuples of (DB, index) where DB is 1-based

class starfit.**Ga**(**args*, *gen=1000*, *time_limit=20*, *pop_size=200*, *sol_size=None*, *tour_size=2*, *frac_mating_pool=1*, *frac_elite=0.5*, *mut_rate_index=0.2*, *mut_rate_offset=0.1*, *mut_offset_magnitude=1*, *local_search=True*, *fixed_offsets=False*, *seed=None*, *max_pop=8192*, *spread=None*, *group=None*, *pin=None*, *n_top=20*, *interactive=True*, ***kwargs*)

Bases: *StarFit*

Genetic Algorithm.

plot_fitness(*gen=False*, *fig=None*, *ax=None*, *show_copyright=True*, *figsize=(10, 6)*, *dpi=102*)

plot fitness as a function of time

print_empty_table()

print_header()

print_update()

class starfit.**Multi**(**args*, *n_top=10*, *fixed_offsets=False*, *threads=None*, *nice=19*, *save=False*, *webfile=None*, *path=None*, *block_size=131072*, *sol_size=None*, *group=None*, ***kwargs*)

Bases: *StarFit*

Find the best fit for multiple stars (complete search). The search results are printed live.

group:
if number of data bases matches sol_size, take one from each DB

sol_size:

None:
if group is True, set sol_size to number of DBs

TODO - allow multiple contributions form one group
(sol_size becomes vector with length of number of groups)

init_futures(*threads=None*, *nice=19*)

print_empty_table()

print_header()

print_update()

save_file(*file_name*)

working_arrays()

class starfit.**Single**(**args*, ***kwargs*)

Bases: *StarFit*

Find the best fit for single stars (complete search)

Example:

```
import starfit

s = starfit.Single(
    filename = 'HE1327-2326.dat',
    db = 'znuc2012.S4.star.el.y.stardb.gz',
    combine = [[6, 7, 8]],
    z_max = 30,
    z_exclude = [3, 24, 30],
    z_lolim = [21, 29],
    upper_lim = True,
    cdf = True,
    constraints = 'energy <= 5',
    )
```

sol_size = 1

Single has `sol_size` hardcoded to 1. All other arguments are inherited from *StarFit*

class starfit.**Star**(*filename*, *silent=False*)

Bases: `Logged`

All the data read from each file

get_covariances()

get_detection_thresholds()

get_elements()

return all elements in dataset

get_input_data_format()

get_measured()

get_n_covariances()

get_norm()

get_upper_limits()

list_to_array(*data_list*)

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

P

R

S

T

W

Y