

Global Optimization and Broadband Analysis Software for Interstellar Chemistry (GOBASIC) Tutorial Version 4.4 Revised

Original Code: Mary L. Radhuber
Current Developer: Luyao Zou

June 29, 2015

This tutorial assumes that the user is familiar with basic Matlab language and Unix systems.

Table of Contents

1 GOBASIC Desktop Version

- Overview
- Observational Data Section
- Catalog Section
- Fit Parameter Section
- Output & Previous Fits Section
- Manipulate Fits with Python Scripts
- Simulate Spectra with GOBASIC

2 GOBASIC for Scientific Computing Clusters

- System & Software Requirements
- File Modification
- Submit Jobs

Overview

GOBASIC is written in Matlab language. The basic program contains two folders:

- `./GOBASIC_mainbody/`
stores the source codes
- `./CatalogFile/`
stores tab-delimited JPL/CDMS catalog files used for the program

It is recommended to put observational data and fitting data in separate folders, such as `./Observation/` and `./Fit/`

Fitting Routine

1. Launch Matlab, change working directory to `GOBASIC_mainbody` folder
2. If one needs to generate a tab-delimited catalog file, copy the `*.cat` file into this folder, and type in `perl('delimitcatalogfiles.pl')` in Matlab terminal.
(Alternatively, one may also type in `perl delimitcatalogfiles.pl <filename>` in his OS terminal.
Then move the tab-delimited `*.txt` file to the catalog folder.
3. Edit `definevars.m`
4. Type in `Run` in Matlab terminal
5. Examine results

Path and Beam Info

definevars.m

```
comp_num = 2;

obspath =
    '../Observation/obs.csv';

apeff = 0.57; % for CS0

sourcesize = [0; 10; %0; ... ];
    % if each component has its
    % own source size
    % OR
    zeros(comp_num,1);
    % if all 0
% dish diameter of the telescope
dish = 10.4;
```

- **comp_num** specifies the number of molecular components to be fitted
- **obspath** defines the file path of observational data
- **apeff** defines the aperture efficiency of the telescope, which corrects the main beam temperature.
- **sourcesize** vector defines the source size (Gaussian FWHM diameter) of each component, in arc seconds. Must match **comp_num**.
- **dish** defines the size of a telescope dish in meters. The beam size will be calculated on the fly using equation $\theta_b = \frac{1.39c}{\pi\nu D} \times 180 \times 3600$ (arcsec)

On Beam-Filling Correction

The beam-filling factor is calculate by $\eta_b = \theta_s^2 / (\theta_s^2 + \theta_b^2)$, and the main beam temperature is corrected by $T_{MB} = T_b \eta_b$. This should be used only when the source size information is known. Value 0 will set η_b to be unity.

At optical thin limit, the beam-filling factor simply acts like a scalar on column density. However, when the optical thick limit is reached, applying beam-filling factor may or may not represent the reality of that particular source. The user determines the interpration of column density.

Flagging and Uncertainty of Antenna Temperature

definevars.m

```
flagfreq = [%230400,230900;
            %min,max;
            ];

ta_err_from_fit = true;

ta_err_deconv = .03;
```

- **flagfreq** defines the flag frequency windows. Leave a blank matrix `[]`; if no flagging is needed
- **ta_err_from_fit** determines the way σ_{T_a} is estimated. It is a logic value. When **true**, it estimates σ_{T_a} from the goodness of the fit, i.e.

$$\sigma_{T_a} = \sqrt{\sum_i (y_i - \text{fit}_i)^2 / (N - 4j)}.$$
 When **false**, it estimates σ_{T_a} from the noise level of the raw data.
- **ta_err_deconv** assigns σ_{T_a} from the noise level of the raw data before deconvolution manually.

Import Catalog and Partition Function Files

definevars.m

```
molpath =
{'../CatalogFile/1.txt';
 '../CatalogFile/2.txt';
 '%../CatalogFile/more.txt';
};

parpath =
{'../CatalogFile/1par.par';
 '../CatalogFile/2par.par';
 '%../CatalogFile/morepar.par';
};

label =
{'MoleculeName1';
 'MoleculeName2';
 % 'MoreNames';
};

freq_uc_cutoff = 0;
int_cutoff = 0;
elow_cutoff = 0;
```

- **molpath** defines the file path of the catalog file for each fitting component.
- **parpath** defines the file path of partition function files for each fitting component.
- **label** defines the labels of each component in the legend and output files. Can be left blank as **label = {};**, in which case default names **fit1**, **fit2**, **etc.** are used.
- **freq_uc_cutoff**, **int_cutoff** and **elow_cutoff** are used to omit molecular lines with large frequency uncertainties, low intensities or high lower state energies.

Tips On Catalog Files

It is highly recommended to use [relative path](#). It helps to keep the codes easy to understand, and it works across platforms. One may notice that Windows system uses backslash \ while Unix systems use slash / as directory separators. But in GOBASIC please use /. It works for all platforms.

The partition function files [*.par](#) are typed manually, with the first column as $Q(T)$ and second column as temperature (K). These numbers can be retrieved from JPL catalog documents. GOBASIC fits these numbers to a function. T and Q numbers can be delimited by [A SINGLE TAB](#) or comma. *DO NOT type more tabs even though the two columns are not aligned by appearance.*

The user may want to set some cut-off of the intensity and frequency uncertainty based on the physical conditions of the source, so that super-weak or super-uncertain lines are omitted. Less molecular lines means faster fit. So it is a good way to do a rough test, for example, of the existence of the target molecule in a particular source.

Automatic catalog name generator

A Python3 script is provided in v4.3 to help the user typeset the long list of catalog files. The script follows the current file naming rule, which adds an extension of `.txt` after the name of the molecule for catalog file, `par.par` for partition function file, and the molecule's name itself as the label.

The user needs to create a text file containing all target molecular components. Molecule names should match the catalog and partition function file names stored. Duplicated names are necessary if multiple components of the same molecule is to be fitted.

In the OS terminal, type in

```
python3 comp_label_gen.py compname.list
```

A file called `component_label.m` will be generated, which contains the full sorted `molpath`, `parpath` and `label` list in Matlab format. Copy the contents into the corresponding section in `definevars.m`

Set Patternsearch Options

definevars.m

```
options =  
    psoptimset('Display','iter','CompletePoll','On','UseParallel','always',...  
        'MaxFunEval',100000*comp_num,'MaxIter',100000,'ScaleMesh','on',...  
        'TolMesh',1e-10,'TolX',1e-10,'TolFun',1e-10,'InitialMeshSize',1);
```

Most times there is no need to change the options. In experience, **1e-10** tolerance is enough for decent fit. If one only needs a quick and rough fit, the tolerance can be changed to **1e-5**. In case of any other requirement, please read <http://www.mathworks.cn/cn/help/gads/psoptimset.html> for more information.

Set Boundaries and Initial Guess

definevars.m

```
% log10(NT), FWHM,
% 0.01*Temp, Shift
low = [10,1,2,-10;
        %10,1,2,-10; ...
    ];

% or
% repmat([10,1,2,-10],...
        comp_num,1);

up = [20,20,10,10;
        %20,20,10,10; ...
    ];

init = [15,10,6,0;
        %15,10,6,0; ...
    ];
```

4 parameters are fitted in GOBASIC:
column density (cm^{-2}) in 10-base logarithm,
FWHM (km/s), 0.01 times temperature (K),
and Doppler shift (km/s).

Each row in the matrices represents one molecular component. Columns stand for the four parameters by the order shown on the left. The **low** and **up** matrices set low and up boundary of the parameter space. The **init** matrix defines initial guess. Number of rows must match number of components. If all rows are identical, use **repmat** to simplify input.

ATTENTION!

Input column density is in its 10 based logarithm. Input temperature is in its natural logarithm.

How do I continue a fit when it halts after reaching max iteration steps?

Set the initial guess as the fitting results, and '**InitialMeshSize**' as the last mesh size shown in the Matlab output terminal. Change output file name, leave everything else unmodified, and run again. But, it is unlikely to happen.

Set Output Path & Include Previous Fittings

definevars.m

```
outputpath = '../Fit/';
outputname = 'name';

pfitPath =
    {'../fitdir/fitname.csv';
    %***
    }
```

- **outputpath** defines the folder to store output files. DO NOT FORGET the last /
 - **outputname** defines the output file names. Two files will be generated.
 - **testFit.csv** stores the fitted spectra
 - **testResults.csv** saves the values of fitting parameters
- The suffixes '**Fit.csv**' and '**Results.csv**' are automatically added to the end of **outputname**.
- One can include previous fitted spectra by adding string entries in **pfitpath**. GOBASIC will automatically load previous fits.

Display Results on Matlab GUI

The results are displayed on Matlab terminal using the following format:

```

----- Name 1 -----
NT    =   *** +/- * (cm^-2)
FWHM  =   *** +/- * (km/s)
Temp  =   *** +/- * (K)
shift =   *** +/- * (km/s)
----- Name 2 -----
etc.

```

Notice that if `label` is left blank (see Slide 9), `component#` will be used as default names.

Plot is shown on Matlab GUI. Legends are set based on `label`. The observational data is always black, the totalfit always cyan, and each component a random color. If colors are too similar to be distinguished, simply close the plot window, and type in `displayresult` again in Matlab terminal, as a new series of random colors will be applied.

Output Format

Three files are generated for each successful fitting in the latest GOBASIC version. `trialnameFit.csv` stores the spectra, while `trialnameResults.csv` stores the fitted parameters in high accuracy, and `trialnameTable.txt` stores the fitted parameters in a readable tab-delimited file.

Each column in `trialnameFit.csv` stands for one component. Headers are set based on `label` (see Slide 9). The format is designed to be easily imported into `Igor`. Simply choose 'load general text' in `Igor` and waves will be automatically imported. The format is also compatible with `gnuplot` for Linux users. Output files has headers including GOBASIC version information.

Search for Molecules and Pick Components in a Fit File

GOBASIC also carries a couple of Python3 scripts to help the user manipulate fitted data. Requires [Python3+s](#) and [Numpy1.8+](#)

[PickGOBASIC.py](#) reads a fitted csv file, searches through its header for input molecule names, picks out corresponding components, and writes these data into a new file. One types

```
python3 PickGOBASIC.py InputFileName.csv
```

in the OS terminal. The script reads the file, and asks the user to input the name of the molecules for search. Type in, for example, [CH3OH](#) [CH3CN](#) *etc*, separated by blank space. The script tries to find matches in the header of the input fitted file. If matches found, the script picks out the data columns

of these molecular components, and outputs a new csv file formatted as [obsfreq](#), [obsint](#), [previousfit](#), [matched_columns](#), [sum_matched_columns](#)

The output file name is typed in by the user in the terminal.

Corresponding header information will be carried into the new file.

Concatenate Multiple Fit Files

`SumGOBASIC.py` reads a series of fitted csv files, and concatenate all data columns together to a new csv file.

One types

```
python3 SumGOBASIC.py InputFileName1.csv InputFileName2.csv etc
```

in the OS terminal. The script reads all specified files, concatenates all molecular components in a sorted order (by label names), and outputs a new csv file formatted as `obsfreq`, `obsint`, `previousfit`, `all-components`, `sum-all-comps`. The output file name is typed in by the user in the terminal. The column `previousfit` is only used for plotting. It does not go into the summation of intensity.

Simulate Spectra with GOBASIC

GOBASIC can also be used to simulate molecular spectra. To do this

1. type in the following codes in Matlab terminal:

```
freq = (FREQ_MIN:STEP:FREQ_MAX)'; % set frequency window
dlmwrite('your-spectra-path+name.csv',...
        horzcat(freq,zeros(length(freq),1)));
```

2. run **definevars** with the desired number of components, catalog and partition function files, and initial guess settled.
3. edit **definevars.m** again: assign **obspath** to the filename typed in above
4. run **definevars** again
5. type in the following codes in Matlab terminal:

```
analysisisscript; anonfun4; spectra = spec(init);
dlmwrite(obspath,horzcat(obsfreq,spectra),'precision',10);
```

Table of Contents

- ① GOBASIC Desktop Version
 - Overview
 - Observational Data Section
 - Catalog Section
 - Fit Parameter Section
 - Output & Previous Fits Section
 - Manipulate Fits with Python Scripts
 - Simulate Spectra with GOBASIC
- ② GOBASIC for Scientific Computing Clusters
 - System & Software Requirements
 - File Modification
 - Submit Jobs

System & Software Requirements

One needs

- A cluster server which has Matlab and Matlab compiler installed
- An accessible account on some scientific computing cluster server: `user@server`
- A client software to communicate with the server. For Linux and Mac users,

```
ssh user@server # or  
ssh -Y user@server # enables remote matlab GUI
```

For Windows users, use SSH client such as [Putty](#)

- A ftp software to upload and download files

Modification of Code Files

- If not using remote GUI, the spectra plotting functionality in GOBASIC is recommended to be switched off. It's in the very bottom of `optest.m`. Comment out `displayresult` by Matlab comment symbol `%`.
- Configure Matlab parpool profile. Add these codes in `Run.m`:

```
...  
% choose correct parpool profile,  
% and specify numbers of clusters available  
fitjob = parpool('local',12);  
optest;  
% close parpool before exit  
delete(fitjob)  
...
```

Compile Codes

Type in `mcc -mv Run.m` in the server terminal.

3 files are generated during the compilation. `Run` is the binary machine readable file. `run_Run.sh` is the shell script to launch `Run`. `mccExcludedFiles.log` is the log file. DO NOT edit these files!

After this job is finished, `Run.out` saves all information that is expected to appear in Matlab terminal.

Submit Jobs Through Load Leveler

The machine-readable files compiled from the previous slide can be submitted for calculation. Please consult your server manager for technical support.

Here we show an example using Load Leveler system.

1. Create Load Leveler job file `llqjobname.cmd`
2. Submit this job by typing `llsubmit llqjobname.cmd`
3. View jobs: `llq -u username`

A typical Load Leveler job file looks like this:

```
#!/bin/ksh
# @ initialdir = /user/home/folder/GOBASIC/dir/
# @ requirements = (Arch == "ArchNumber") && (OpSys == "OSNumber")
# @ class = cluster_name
# @ notify_user = user@email
# @ error = error_file
# @ queue
cd $PBS_O_WORKDIR
./run_Run.sh /server/MATLAB/DIR/R2014a > Run.out
```

Thanks For Your Attention!