SPECIAL BLEND manual

nuLC collaboration
Akira Harada, Yudai Suwa, Masayuki Harada, Yusuke Koshio, Masamitsu Mori,
Fumi Nakanishi, Ken'ichiro Nakazato, Kohsuke Sumiyoshi, and Roger A. Wendell

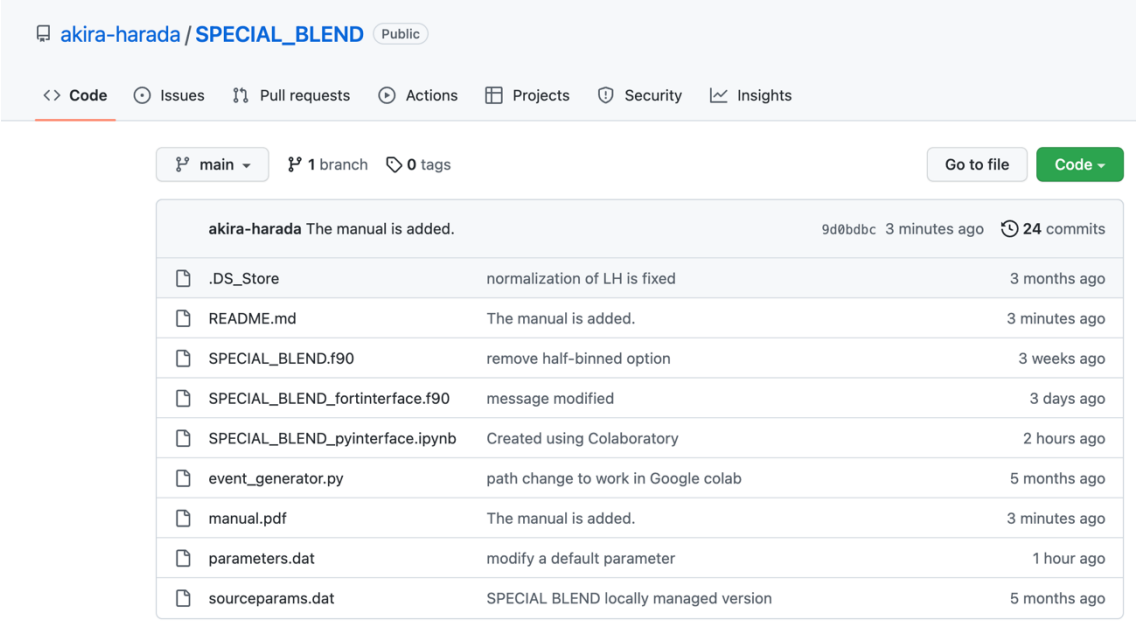
1. Introduction

Supernova Parameter Estimation Code based on Insight on Analytic Late-time Burst Light curve at Earth Neutrino Detector (SPECIAL BLEND) is the code that estimates the mass and radius of the proto-neutron star at the center of a supernova and the emitted neutrino energy from observational data of supernova neutrinos with a water Cerenkov detector such as Super Kamiokande. It is based on the analytical model of the supernova neutrino light curve. The design and performance evaluation of the code is presented in Harada et al. (2023). This document explains how to run SPECIAL BLEND on Google Colaboratory and your personal computer (PC). Google Colaboratory is just an example of a ready-to-use service and can be used in any similar environment where Jupyter notebook and f2py are available: a GPU is not required. The Fortran version of SPECIAL BLEND is also offered to run on your PC. It, of course, requires a Fortran environment. In addition, the event generator associated with SPECIAL BLEND requires a python environment where numpy can be used.

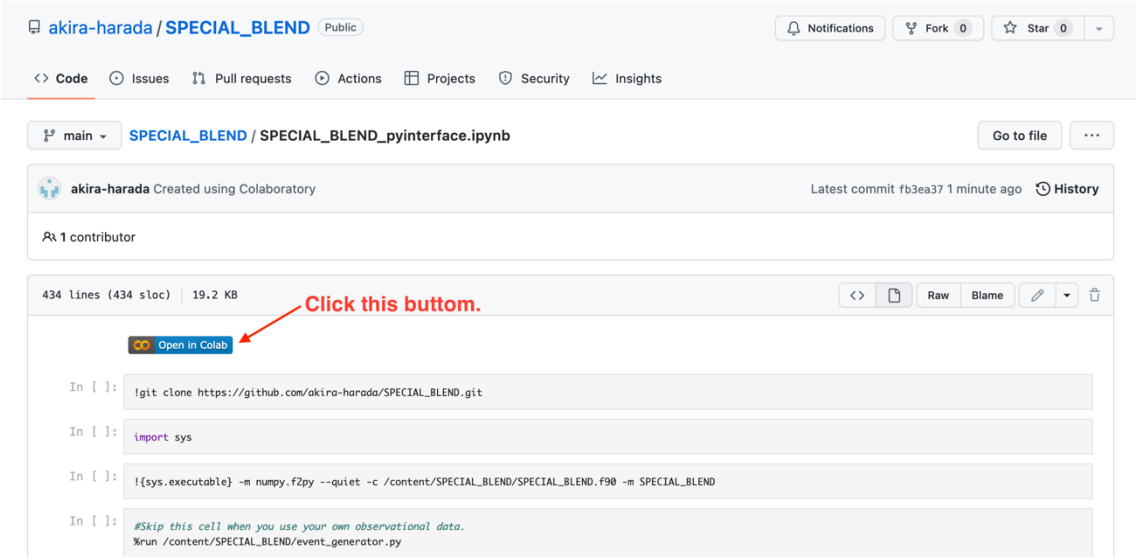
This manual first presents the quick start guide of SPECIAL BLEND on Google Colaboratory in section 2. Then, the detailed manual of the python interface is explained in section 3. Afterward, the manual for the Fortran interface is presented in section 4.

2. Getting started

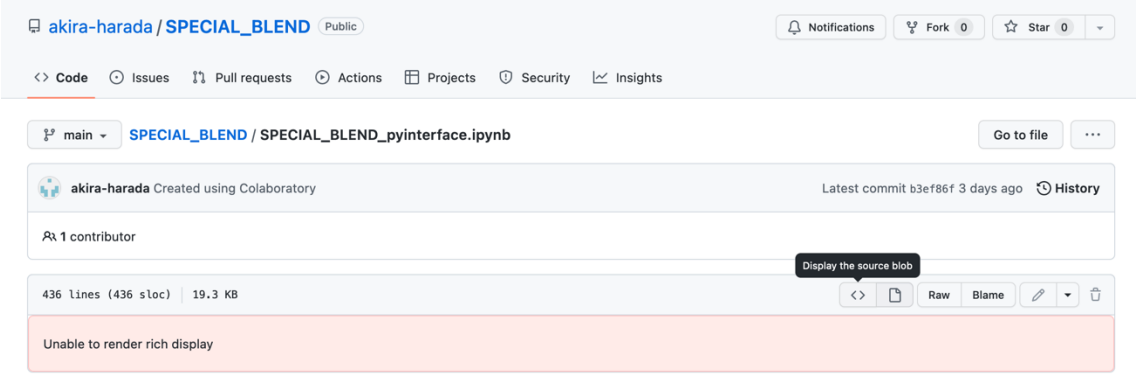
When you use SPECIAL BLEND on Google Colaboratory, the file SPECIAL_BLEND_pyinterface.ipynb is the interface. The screenshot of the repository is as follows.



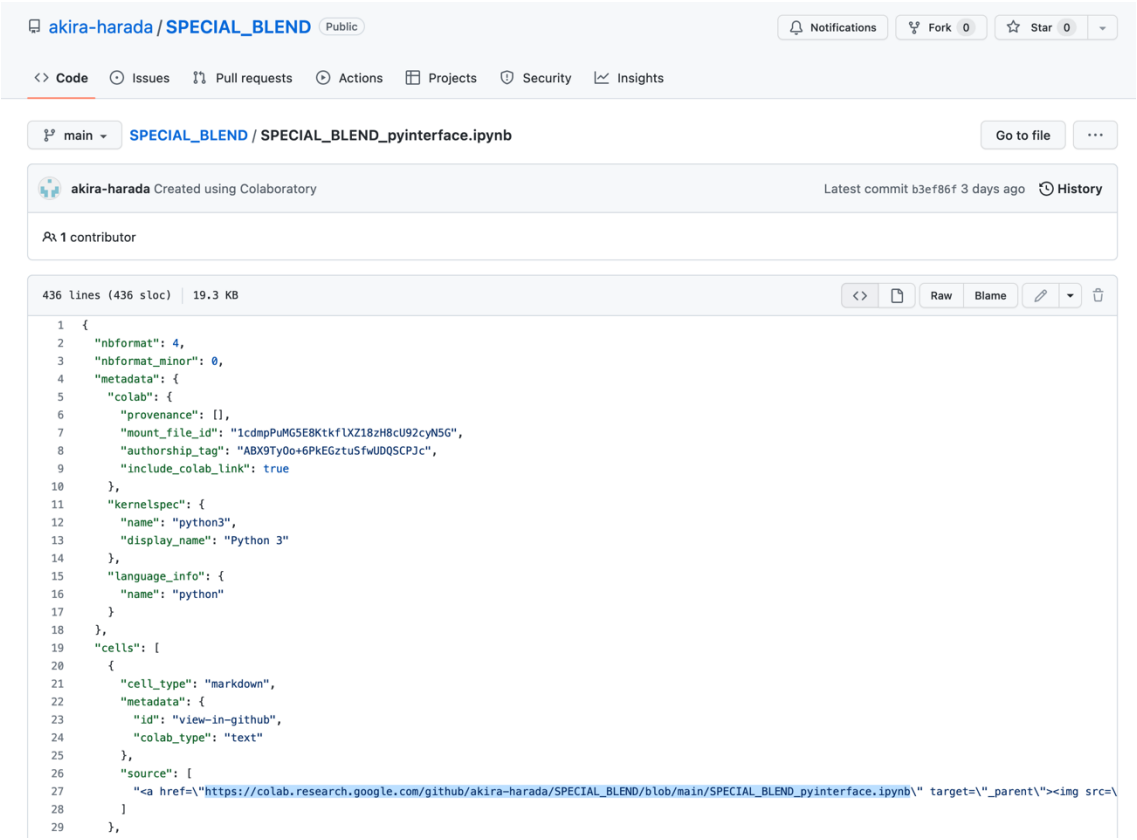
Click “SPECIAL_BLEND_pyinterface.ipynb” leads to the following screenshot.



Then, click “Open in Colab” buttom to open SPECIAL BLEND in Google Colaboratory. Sometimes the notebook style display fails.

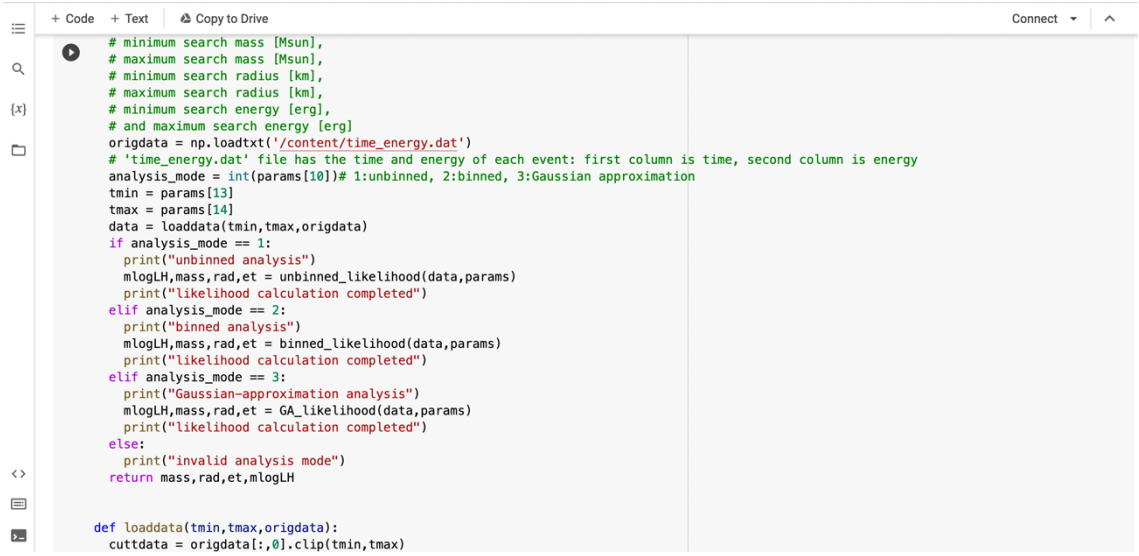


In this case, click "< >" button (the message "Display the source blob" will appear when hover the pointer), then the source code of the notebook directly appears.



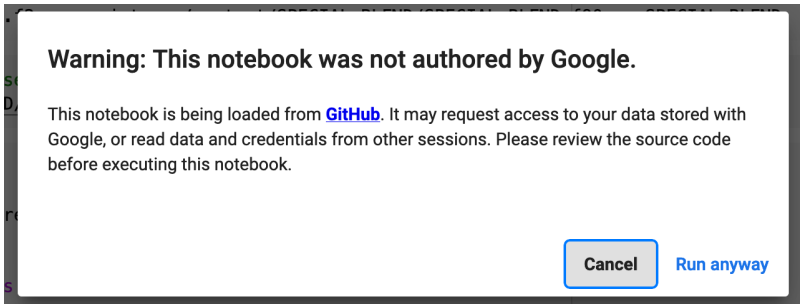
The URL selected in the screenshot is to open SPECIAL BLEND in Google Colaboratory. Note that you need to log in with a Google account when you use Google Colaboratory.

When you open SPECIAL_BLEND_pyinterface.ipynb in Google Colaboratory, the window becomes like the following screenshot.



First, let us consider the tutorial case: generating mock observational data based on the analytic model and then estimating the parameter from the data by SPECIAL BLEND. For this purpose, run each cell by shift+enter from top to bottom[1]. It results in the window like the following screenshots, and you will see the graphs of normalized likelihoods (as defined in Harada et al. (2023), this is the Bayesian posterior probability density function).

[1] When you start running a cell, the following caution appears.



Please click “Run anyway” to proceed.

+ Code + Text Copy to Drive RAM Disk

[1] !git clone https://github.com/akira-harada/SPECIAL_BLEND.git

```
Cloning into 'SPECIAL_BLEND'...
remote: Enumerating objects: 90, done.
remote: Counting objects: 100% (90/90), done.
remote: Compressing objects: 100% (63/63), done.
remote: Total 90 (delta 48), reused 54 (delta 24), pack-reused 0
Unpacking objects: 100% (90/90), 534.47 KiB | 2.03 MiB/s, done.
```

[2]
```
import sys
```

[3]
```
!{sys.executable} -m numpy.f2py --quiet -c /content/SPECIAL_BLEND/SPECIAL_BLEND.f90 -m SPECIAL_BLEND
```

[4]
```
#Skip this cell when you use your own observational data.
%run /content/SPECIAL_BLEND/event_generator.py
```

[5]
```
#@title
# define functions
%config InlineBackend.figure_format = 'retina'
import numpy as np
import csv
import matplotlib.pyplot as plt
import SPECIAL_BLEND as SB

def main():
  params = np.loadtxt('/content/SPECIAL_BLEND/parameters.dat')
  # 'parameters.dat' file has the following contents:
```

+ Code + Text Copy to Drive RAM Disk

[6]
```
# calculate likelihood (mlogLH = minus log likelihood)
%time mass,rad,et,mlogLH=main()
```

```
used event number 2963 /total event number 2974
11 events are outside [tmin,tmax] and neglected
binned analysis
likelihood calculation completed
CPU times: user 41.4 s, sys: 192 ms, total: 41.6 s
Wall time: 41.7 s
```

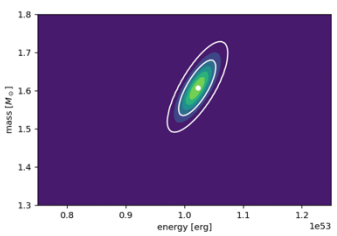
[7]
```
# marginalize likelihood
LH_MR,MRvals,LH_RE,REvals,LH_EM,EMvals,LH_M,LH_R,LH_E = marginalizeLH(mass,rad,et,mlogLH)
```

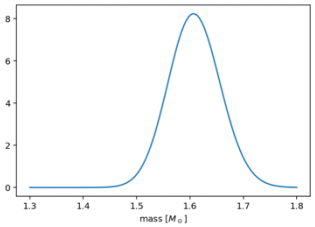
```
1D marginalized result
mass = 1.606643e+00 +5.05e-02/-4.91e-02 (68%) +9.80e-02/-9.28e-02 (95%)
radius = 1.224094e+01 +2.67e-01/-2.62e-01 (68%) +5.43e-01/-5.23e-01 (95%)
energy = 1.021479e+53 +2.14e+51/-2.11e+51 (68%) +4.16e+51/-4.06e+51 (95%)

2D marginalized result
M-R: the best fit is (M,R)=(1.60e+00,1.22e+01), and the levels of CI is 6.73e+00 (68%) and 1.06e+00 (95%)
R-E: the best fit is (R,E)=(1.23e+01,1.02e+53), and the levels of CI is 1.32e-52 (68%) and 2.07e-53 (95%)
E-M: the best fit is (E,M)=(1.02e+53,1.61e+00), and the levels of CI is 7.77e-52 (68%) and 1.19e-52 (95%)
```

[8]
```
# visualize likelihood
visualize(mass,rad,et,mlogLH,LH_MR,MRvals,LH_RE,REvals,LH_EM,EMvals,LH_M,LH_R,LH_E)
```

+ Code + Text Copy to Drive RAM Disk

[8]
```
# visualize likelihood
visualize(mass,rad,et,mlogLH,LH_MR,MRvals,LH_RE,REvals,LH_EM,EMvals,LH_M,LH_R,LH_E)
```





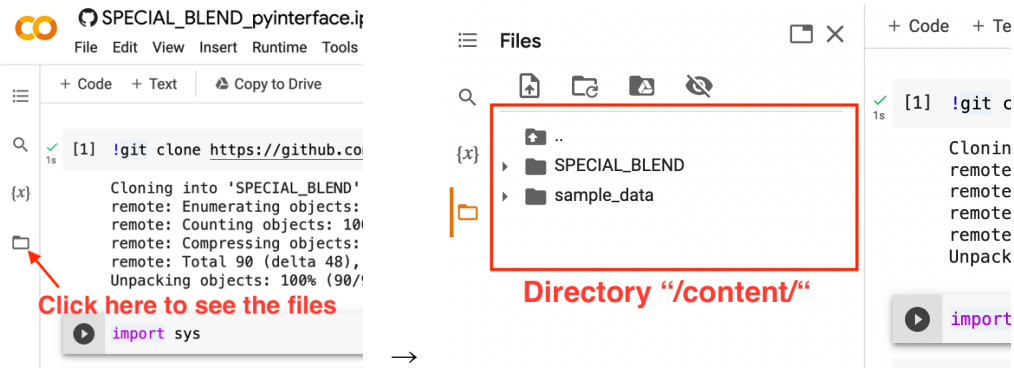
3. Detailed usages of python interface

In the following, the function of each cell will be explained. How to apply your own data will also be explained.

The first cell downloads the files in github to Google Colaboratory.

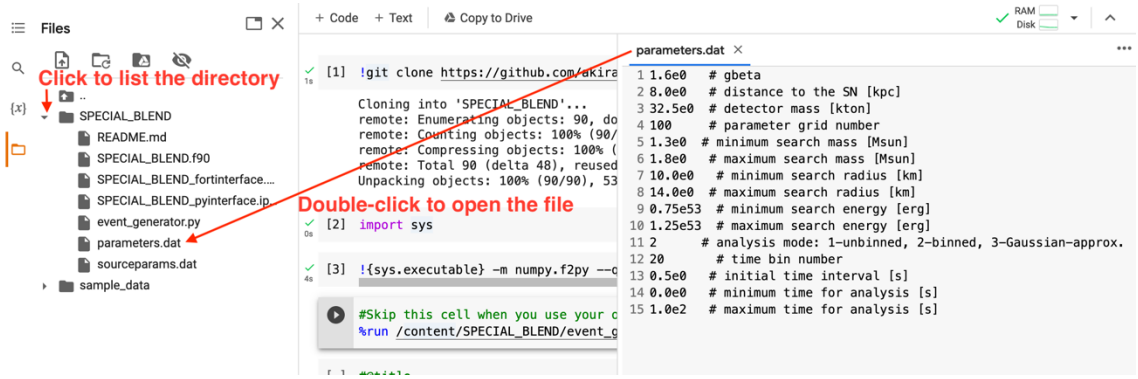
```
[1]  !git clone https://github.com/akira-harada/SPECIAL_BLEND.git
```

By running this cell, the directory /content/SPECIAL_BLEND/ is generated on Google Colaboratory. To see this, click the directory symbol to open the file browser, then you will see the file browser like these screenshots.



The "home" directory when using Google Colaboratory is "/content/". Especially the files parameters.dat and SPECIAL_BLEND.f90 in /content/SPECIAL_BLEND/ are the most important here. If you want to generate mock data files on Google Colaboratory, the files sourceparams.dat and event_generator.py is also required.

The file parameters.dat contains the parameters to run the code. If needed, edit this file by double-clicking to open on Google Colaboratory.



The modified file (named "*parameters.dat") by the editor appearing at the right side will be automatically saved in your runtime session after a while when the file name

returns to “parameters.dat.” The content of parameters.dat is summarized in the footnote[2]. sourceparams.dat file contains the supernova parameters required to generate the mock observational data by event_generator.py. The file format is also indicated in the footnote[3].

The second cell prepares for f2py.

```
[2]  import sys
```

The third cell generates the module SPECIAL_BLEND which contains the subroutines written in SPECIAL_BLEND.f90. For this purpose, the f2py command, which converts the Fortran file to python module, is called from the notebook. Note that f2py is associated with numpy.

```
[3]  !{sys.executable} -m numpy.f2py --quiet -c /content/SPECIAL_BLEND/SPECIAL_BLEND.f90 -m SPECIAL_BLEND
```

The fourth cell generates the mock observational data by running event_generator.py.

```
[4]  #Skip this cell when you use your own observational data.
     %run /content/SPECIAL_BLEND/event_generator.py
```

---

[2] From top to bottom with the format of variable name (default value): explanation,

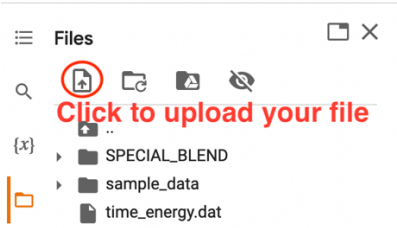
$g\beta$ (1.6): the product of the surface corrections
D (8): the distance to the supernova [kpc]
M_{det} (32.5): the detector mass [kton]
N_p (100): the grid number of the supernova parameters to search
M_{min} (1.3): the minimum mass of the search range [$M_\odot$]
M_{max} (1.8): the maximum mass of the search range [$M_\odot$]
R_{min} (10): the minimum radius of the search range [km]
R_{max} (14): the maximum radius of the search range [km]
E_{min} (0.75×10^{53}): the minimum energy of the search range [erg]
E_{max} (1.25×10^{53}): the maximum energy of the search range [erg]
mode (1): analysis mode: 1 for UBLH, 2 for BLH, and 3 for GALH
N_{time} (20): the number of time bin
dt_1 (0.5): the interval of the first time bin [s]
t_{min} (0): the minimum time for analysis [s]
t_{max} (100): the maximum time for analysis [s]

The events whose times are out of the range $[t_{min}, t_{max}]$ are excluded from analysis. For the parameter estimation, the parameter space $[M_{min}, M_{max}] \times [R_{min}, R_{max}] \times [E_{min}, E_{max}] \subset \mathbb{R}^3$ is divided into $N_p \times N_p \times N_p$ grids, then the likelihood $\mathcal{L}(\theta)$ is calculated on each grid point.

[3] From top to bottom with the format of variable name (default value): explanation,

M (1.52): PNS mass [$M_\odot$]
R (11.8): PNS radius [km]
E (10^{52}): total energy of emitted neutrinos [erg]
$g\beta$ (1.6): the product of the surface corrections

If you use your own data, upload your file to /content/ directly instead of running this fourth cell.



The file format is as follows:

$$t_1\ \epsilon_1$$
$$t_2\ \epsilon_2$$
$$\vdots$$
$$t_{N_{\text{events}}}\ \epsilon_{N_{\text{events}}}$$

For example,

time_energy.dat

1	1.640609761777284928e-04	1.756581081081080953e+01
2	2.826339490470040645e-03	9.958963963963963550e+00
3	8.147331711620171488e-03	2.156941441441441398e+01
4	9.104111987625434360e-03	1.626463963963963977e+01
5	1.250184146530250624e-02	1.726554054054053822e+01
6	1.299265795752708487e-02	1.576418918918918699e+01
7	1.949281441954288027e-02	1.926734234234234222e+01
8	2.257025097799239891e-02	3.237914414414414210e+01
9	2.528190089442447150e-02	4.954459459459458870e+00
10	2.576182742320438923e-02	7.056351351351350942e+00

Note that your original data file should be saved in your computers/disks/clouds. Google Colaboratory offers a temporal runtime, and hence the data contained in /content/ will be removed after the session.

When you run event_generator.py in the fourth cell, a mock observational data file time_energy.dat is generated in /content/ based on the parameters written in parameters.dat and sourceparams.dat. This event generator overwrites the existing time_energy.dat file. Thus, if you make the mistake of running the fourth cell after uploading your own time_energy.dat, re-upload the file before running the sixth cell, where the likelihood is calculated (explained later).

The fifth cell defines the interface functions to use SPECIAL BLEND from the Jupyter notebook.

```
[6] #@title
    # define functions
    %config InlineBackend.figure_format = 'retina'
    import numpy as np
    import csv
    import matplotlib.pyplot as plt
    import SPECIAL_BLEND as SB

    def main():
      params = np.loadtxt('/content/SPECIAL_BLEND/parameters.dat')
      # 'parameters.dat' file has the following contents:
```

The sixth cell calculates the likelihood (exactly speaking, negative logarithmic likelihood $-\log \mathcal{L}$).

```
[7] # calculate likelihood (mlogLH = minus log likelihood)
    %time mass,rad,et,mlogLH=main()
```

```
used event number 3025 /total event number 3036
11 events are outside [tmin,tmax] and neglected
binned analysis
likelihood calculation completed
CPU times: user 42.3 s, sys: 132 ms, total: 42.4 s
Wall time: 42.5 s
```

When you run this cell, the number of events inside and outside of $[t_{\mathrm{min}}, t_{\mathrm{max}}]$ and the analysis mode (UBLH/BLH/GALH) are indicated. If the likelihood is successfully calculated, a message “likelihood calculation completed” and the elapse time are displayed. If it fails, error messages are shown. In that case, please re-run the sixth cell with different parameters, for example, the number of time bins $\mathrm{N}_{\mathrm{time}}$.

```
[10] # calculate likelihood (mlogLH = minus log likelihood)
     %time mass,rad,et,mlogLH=main()
```

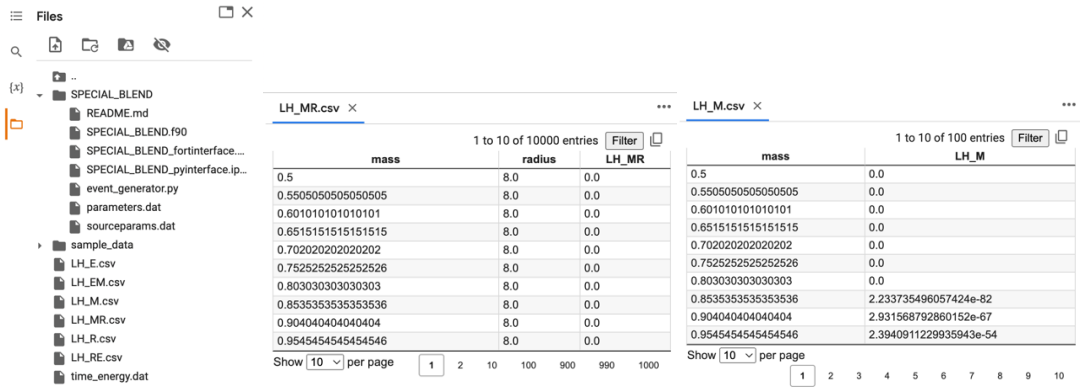
```
used event number 20 /total event number 20
0 events are outside [tmin,tmax] and neglected
Gaussian-approximation analysis
time binning error, try another bin number
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
```

```
[9]  # calculate likelihood (mlogLH = minus log likelihood)
     %time mass,rad,et,mlogLH=main()

     used event number 20 /total event number 20
     0 events are outside [tmin,tmax] and neglected
     Gaussian-approximation analysis
     1-th time bin has no events
     try another time bin number
     ---------------------------------------------------------------------------
     NameError                                 Traceback (most recent call last)
```

The seventh cell marginalizes the 3D likelihood obtained in the sixth cell to give the 1D/2D likelihoods.

```
[13] # marginalize likelihood
     LH_MR,MRvals,LH_RE,REvals,LH_EM,EMvals,LH_M,LH_R,LH_E = marginalizeLH(mass,rad,et,mlogLH)

     1D marginalized result
     mass = 1.618462e+00 +4.72e-02/-4.60e-02 (68%) +9.67e-02/-9.16e-02 (95%)
     radius = 1.238578e+01 +2.80e-01/-2.75e-01 (68%) +5.46e-01/-5.25e-01 (95%)
     energy = 1.056691e+53 +2.14e+51/-2.11e+51 (68%) +4.16e+51/-4.06e+51 (95%)

     2D marginalized result
     M-R: the best fit is (M,R)=(1.62e+00,1.24e+01), and the levels of CI is 6.88e+00 (68%) and 1.07e+00 (95%)
     R-E: the best fit is (R,E)=(1.24e+01,1.06e+53), and the levels of CI is 1.33e-52 (68%) and 2.07e-53 (95%)
     E-M: the best fit is (E,M)=(1.06e+53,1.62e+00), and the levels of CI is 7.76e-52 (68%) and 1.21e-52 (95%)
```

At the same time, this cell outputs the best-fit values and their 68/95% CIs (1D) or the levels of the likelihoods to give the CIs (2D). For the 2D case, the contours with the levels depict the CIs. The cell also outputs the CSV files of the marginalized likelihoods. After running this cell, the 2D likelihood files of LH_MR.csv, LH_RE.csv, and LH_EM.csv, and 1D likelihood files of LH_M.csv, LH_R.csv, and LH_E.csv are generated in /content/ directory. Each CSV file indicates the parameters (M, R, and E or combinations of them) and the corresponding values of the likelihood.



If something wrong happens in marginalization, error messages are shown. For

example, a frequent error is that the hierarchy among the best fits and their uncertainties, i.e.,

lower side of 95% CI < lower side of 68% CI < best fit

< upper side of 68% CI < upper side of 95% CI

for the 1D case and

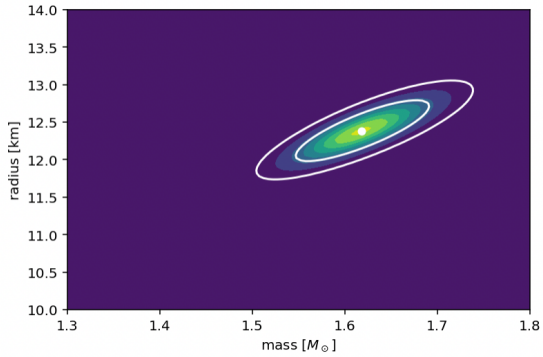
the level of the likelihood for 95% CI < the level of the likelihood for 68% CI

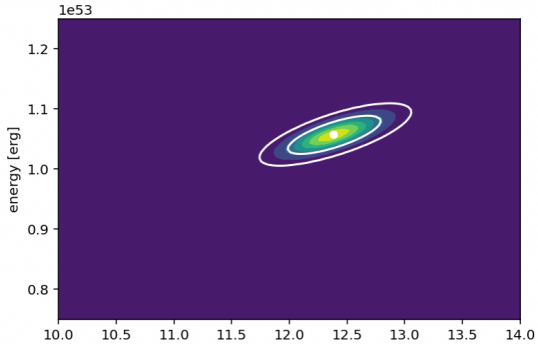
for the 2D case, does not hold. In that case, retrying the sixth cell with another parameter search range sometimes solves the problem. Note that "CIBFM/CIBFR/CIBFE" appearing in the error message are the arrays to contain the values of Credible Intervals and Best Fits for the parameter $M/R/E$.

```
[17] # marginalize likelihood
     LH_MR,MRvals,LH_RE,REvals,LH_EM,EMvals,LH_M,LH_R,LH_E = marginalizeLH(mass,rad,et,mlogLH)

     1D marginalized result
     mass = 6.981564e-01 +-6.98e-01/--6.95e-01 (68%) +-6.98e-01/--6.99e-01 (95%)
     1D CI estimation error in function marginalizeLH()
     array CIBFM containes [lower edge of 95%-CI, lower edge of 68%-CI, best fit, upper edge of 68%-CI, upper edge of 95%-CI]
     but ascending nature of CIBFM does not hold
     retry with another parameter search range
     or data quality is too low to analyze
     radius = 6.525725e+00 +8.44e+00/-6.53e+00 (68%) +8.44e+00/-6.53e+00 (95%)
     energy = 1.021900e+53 +1.43e+51/-1.42e+51 (68%) +2.53e+51/-2.49e+51 (95%)

     2D marginalized result
     M-R: the best fit is (M,R)=(1.40e+00,1.30e+01), and the levels of CI is 0.00e+00 (68%) and 8.98e+02 (95%)
     2D CI estimation error in function marginalizeLH()
     95%-CI contour may be inside 68%-CI contour
     retry with another parameter search range
     or data quality is too low to analyze
     R-E: the best fit is (R,E)=(1.30e+01,1.02e+53), and the levels of CI is 3.25e-51 (68%) and 7.33e-52 (95%)
     E-M: the best fit is (E,M)=(1.02e+53,1.40e+00), and the levels of CI is 3.89e-50 (68%) and 3.41e-51 (95%)
```

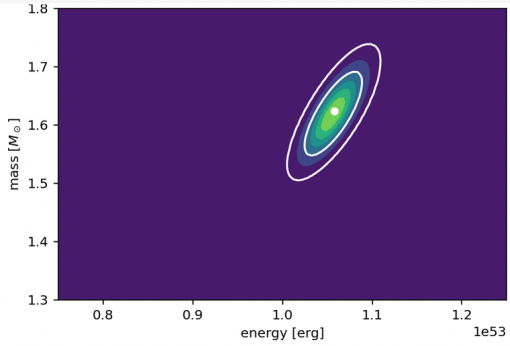
The eighth cell visualizes the marginalized likelihoods with colormaps (2D) and graphs (1D).

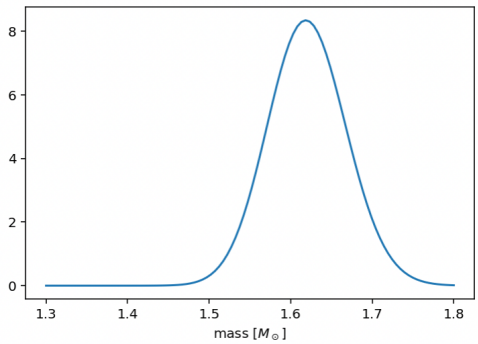
```
[20] # visualize likelihood
     visualize(mass,rad,et,mlogLH,LH_MR,MRvals,LH_RE,REvals,LH_EM,EMvals,LH_M,LH_R,LH_E)
```



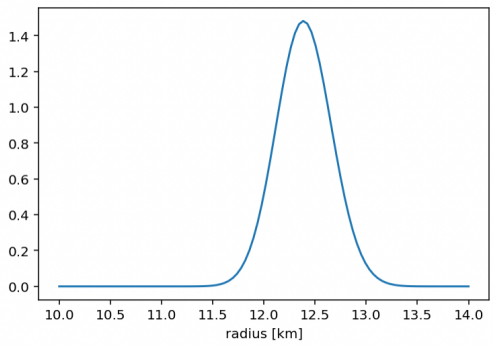


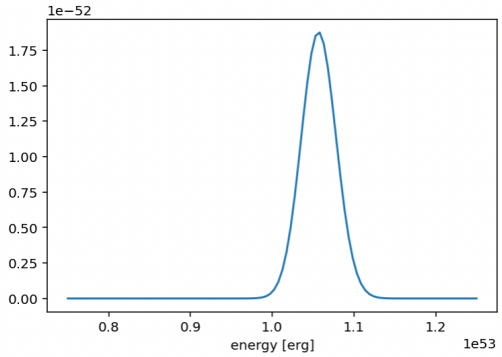

```
[20] # visualize likelihood
     visualize(mass,rad,et,mlogLH,LH_MR,MRvals,LH_RE,REvals,LH_EM,EMvals,LH_M,LH_R,LH_E)
```



[20]





It helps you to see the estimated parameters. If you want to visualize by yourself, download the CSV files created with the seventh cell and use your own plotting software.

4. SPECIAL BLEND with the Fortran interface

The Fortran interface of SPECIAL BLEND can run on your PC if a Fortran compiler is installed. When you use the Fortran interface, SPECIAL_BLEND.f90 and SPECIAL_BLEND_fortinterface.f90 are required. If you also generate the mock observational data, a python environment with numpy is required to run event_generator.py. In the following, the terminal commands are written like

```
$ cd ./SPECIAL_BLEND
```

and the contents of the file are written like

```
program SPECIAL_BLEND
implicit none
```

To run SPECIAL BLEND on your PC, clone github by the following command:

```
$ git clone https://github.com/akira-harada/SPECIAL_BLEND.git
```

And go to the generated directory “SPECIAL_BLEND”

```
$ cd ./SPECIAL_BLEND
$ ls
README.md
SPECIAL_BLEND.f90
SPECIAL_BLEND_fortinterface.f90
SPECIAL_BLEND_pyinterface.ipynb
event_generator.py
manual.pdf
parameters.dat
sourceparams.dat
```

Then, compile SPECIAL BLEND. Because only two files are involved, makefile is not used. Instead, compile as

```
$ gfortran -O3 SPECIAL_BLEND.f90 SPECIAL_BLEND_fortinterface.f90
```

if you use gfortran compiler. Without -o option, the executable file a.out will be generated.

Next, tune parameters by editing parameters.dat similarly to the fourth cell in section 3, and put your observational data in the current directory where the compiled executable file exists. If you want to generate a mock data file with event_generator.py, edit sourceparams.dat. Note that the directory structure assumed in event_generator.py is adjusted to Google Colaboratory. In order to run it on your PC, edit LL. 15-16 of event_generator.py from

```
pnsparams = np.loadtxt('/content/SPECIAL_BLEND/sourceparams.dat')
obsparams = np.loadtxt('/content/SPECIAL_BLEND/parameters.dat')
```

to

```
pnsparams = np.loadtxt('./sourceparams.dat')
obsparams = np.loadtxt('./parameters.dat')
```

Then,

```
$ python3 event_generator.py
```

will produce the time_energy.dat file in the current directory.

After completing the above preparation, execute a.out to run the subroutines corresponding to the sixth and seventh cells in section 3. Similarly to the python interface, the number of events inside and outside of $[t_{\min}, t_{\max}]$ and the analysis mode

will be displayed. In the case of GALH mode, the program will stop if there are empty time bins. In that case, change the number of time bins and retry. Finally, the elapse time will be presented.

The Fortran version of SPECIAL BLEND generates the files of marginalized likelihoods. For visualization, use these files with your plotting software, such as gnuplot. The file format is as follows:
In the 2D likelihood (LH_MR.dat, LH_RE.dat, and LH_EM.dat) files, the comments

```
# (levels of the likelihood for 95/68% CIs)
# (best fit values)
# (column description)
```

come first, and then the list of the parameter values and the value of the likelihood at that parameter follows. The list has blank lines after each scan of the parameter in the second column. This format will help to display the 2D colormap with the gnuplot option of “with image.” In order to draw the 68/95% Cis, open the file to read the levels and draw the contour of the levels.
The 1D likelihood (LH_M.dat, LH_R.dat, and LH_E.dat) files contain the comments

```
# (best fit +/- 68/95% CIs)
# (column description)
```

and the list of the parameter and likelihood.