6th Numerical Lattice Field Theory Training Camp

Lattice QCD Data Analysis

Lattice Parton Collabration

Outline

Outline

- Theory on two-point function

Outline

- Theory on two-point function
- Data analysis
    - Programing language/packages
    - `` `iog` `` data format
    - Resampling
    - least χ^2 fit

Outline

- Theory on two-point function
- Data analysis
    - Programing language/packages
    - `` `iog` `` data format
    - Resampling
    - least χ^2 fit
- Linux Basics

Outline

- Theory on two-point function
- Data analysis
  - Programing language/packages
  - `` `iog` `` data format
  - Resampling
  - least χ^2 fit
- Linux Basics
- *Introduction to HDF5*

I. Theory on two-point function

I. Theory on two-point function

Two-point correlation function (two-point function, 2pt)

I. Theory on two-point function

Two-point correlation function (two-point function, 2pt)

$$\langle \mathcal{O}(t_{\text{snk}})\mathcal{O}(t_{\text{src}})^\dagger \rangle = \sum_{n=0}^{\infty} \langle 0|\mathcal{O}|n\rangle\langle n|\mathcal{O}^\dagger\rangle e^{-E_n(t_{\text{snk}}-t_{\text{src}})} = c_0 e^{-E_0 t} + c_1 e^{-E_1 t} + \cdots$$

I. Theory on two-point function

Two-point correlation function (two-point function, 2pt)

$$\langle \mathcal{O}(t_{\text{snk}})\mathcal{O}(t_{\text{src}})^\dagger \rangle = \sum_{n=0}^{\infty} \langle 0|\mathcal{O}|n\rangle\langle n|\mathcal{O}^\dagger\rangle e^{-E_n(t_{\text{snk}}-t_{\text{src}})} = c_0 e^{-E_0 t} + c_1 e^{-E_1 t} + \cdots$$

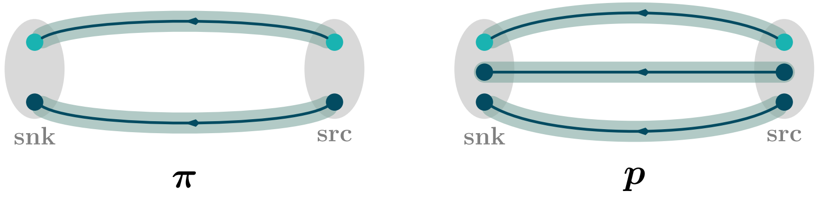
> 《格点量子色动力学导论》刘川，§20

I. Theory on two-point function

Two-point correlation function (two-point function, 2pt)

$$\langle \mathcal{O}(t_{\text{snk}})\mathcal{O}(t_{\text{src}})^\dagger \rangle = \sum_{n=0}^{\infty} \langle 0|\mathcal{O}|n\rangle\langle n|\mathcal{O}^\dagger\rangle e^{-E_n(t_{\text{snk}}-t_{\text{src}})} = c_0 e^{-E_0 t} + c_1 e^{-E_1 t} + \cdots$$

《格点量子色动力学导论》刘川，§20

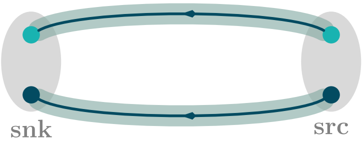


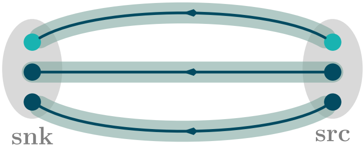
I. Theory on two-point function

Two-point correlation function (two-point function, 2pt)

$$\langle \mathcal{O}(t_{\mathrm{snk}})\mathcal{O}(t_{\mathrm{src}})^\dagger \rangle = \sum_{n=0}^{\infty} \langle 0|\mathcal{O}|n\rangle\langle n|\mathcal{O}^\dagger\rangle e^{-E_n(t_{\mathrm{snk}}-t_{\mathrm{src}})} = c_0 e^{-E_0 t} + c_1 e^{-E_1 t} + \cdots$$

> 《格点量子色动力学导论》刘川，§20





- **For large enough $\tau_f - \tau_i$,** higher excited states decay faster

  $$\langle \mathcal{O}(t_{\mathrm{snk}})\mathcal{O}(t_{\mathrm{src}})^\dagger \rangle \approx$$

  - $c_0 e^{-E_0 t}$ ⇨ one state analysis
  - $c_0 e^{-E_0 t} + c_1 e^{-E_1 t}$ ⇨ two state analysis

- Determine the hadron mass spectrum from lagrangian (first-principle or ab initio calculation)

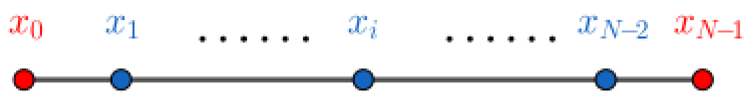
Boundary conditions

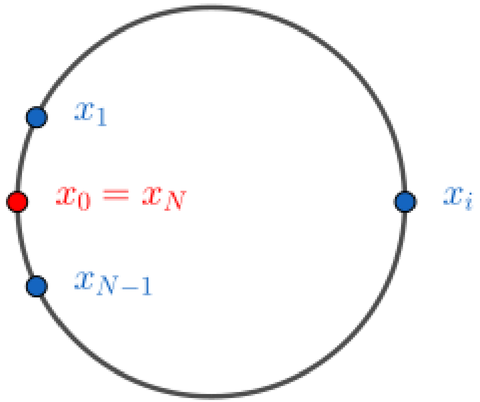
Boundary conditions

- Open bondary condition
    - $\phi_i = 0$ if $i \geq N$ or $i < N_0$

Boundary conditions

- Open bondary condition
  - $\phi_i = 0$ if $i \geq N$ or $i < N_0$
- Periodic/AntiPeriodic boundary condition
  - $\phi_N = \pm\phi_0, \phi_{N+i} = \pm\phi_i \Rightarrow \phi_{i+N} = \pm\phi_{(i+N) \bmod N}$





- Considering periodic boundary condition
  - propagate in both $\pm t$ direction

$$C_2(t) \approx C_0 e^{-\frac{T}{2}E_0} \cosh\left[E_0\left(\frac{T}{2} - t\right)\right]$$

> 《格点量子色动力学导论》刘川，§20

- Considering periodic boundary condition
  - propagate in both $\pm t$ direction

    $$C_2(t) \approx C_0 e^{-\frac{T}{2}E_0} \cosh\left[E_0\left(\frac{T}{2} - t\right)\right]$$

> 《格点量子色动力学导论》刘川，§20

cosh

log

- Determine hadron effective energy E_0 (m_{eff})
  - A glance
    - For $0 < t \ll T/2$ ⇨ $e^{-E_0 t} \gg e^{-E_0(T/2-t)}$ ⇨ $\ln \dfrac{C_2(t)}{C_2(t+a)} \approx aE_0$
    - $\dfrac{C_2(t)}{C_2(t+a)} = \dfrac{\cosh[E_0(T/2-t)]}{\cosh[E_0(T/2-(t+a))]}$ or $\dfrac{C_2(t+a)+C_2(t-a)}{2C_2(t)} = \cosh aE_0$
  - More systematically way ⇨ Fit (see sec. **Least χ^2 fit** for detail)

II. 2pt Data Analysis

Programing Language and Packages

II. 2pt Data Analysis

Programing Language and Packages

- Demo provided in `` `python3` ``

II. 2pt Data Analysis

Programing Language and Packages

- Demo provided in `` `python3` ``
- Python package
  - importing data : `` `ctypes` ``, `` `pandas` ``
  - Data Processing: `` `numpy` ``
  - Fitting: `` `gvar` ``, `` `lsqfit` ``
  - Visualization: `` `matplotlib` ``

II. 2pt Data Analysis

Programing Language and Packages

- Demo provided in `python3`
- Python package
  - importing data : `ctypes`, `pandas`
  - Data Processing: `numpy`
  - Fitting: `gvar`, `lsqfit`
  - Visualization: `matplotlib`
- Useful but not mandatory
  - Regular expression (python package `re`)

II. 2pt Data Analysis

Programing Language and Packages

- Demo provided in `python3`
- Python package
  - importing data : `ctypes`, `pandas`
  - Data Processing: `numpy`
  - Fitting: `gvar`, `lsqfit`
  - Visualization: `matplotlib`
- Useful but not mandatory
  - Regular expression (python package `re`)
- Advanced
  - HDF5, `h5py`
  - MPI

- Documentation

Library/Package	Documentation
pandas	**https://pandas.pydata.org/docs/**
numpy	**https://numpy.org/doc/stable/**
gvar	**https://gvar.readthedocs.io/en/latest/overview.html**
lsqfit	**https://lsqfit.readthedocs.io/en/latest/lsqfit.html**
Scipy	https://scipy.org/
HDF5	https://portal.hdfgroup.org/display/support/documentation
h5py	https://docs.h5py.org/en/stable/

- Installation
  - `pip3`: package installer for python
  - `pip3 install package_name --user`
  - `--user` install under current user enviorment, does not need root permission
- Python devel package (required by `iog.so`)
  - header `/usr/include/python3.6m/`

Data Format

Data Format

- `` `iog` ``: io general, used in LPC
  - Needs to write your own parser or dump as text file
  - Interface to `` `pandas` `` dataframe and to `` `HDF5` ``, `` `h5py` `` are provided

Data Format

- `iog`: io general, used in LPC
  - Needs to write your own parser or dump as text file
  - Interface to `pandas` dataframe and to `HDF5`, `h5py` are provided

```python
#!/usr/bin/env python3
# encoding: utf-8
from iog_reader import iog_read

iog_file = "./2pt_5350.dat.iog"
intrptr = ["cnfg","hdrn","t"]

iog = iog_read(iog_file,intrptr)
print(iog,'\n--------\n')

iog_pi=iog.loc[(iog['hdrn']=='0')]
print(iog_pi['Re'].to_numpy(),'\n--------\n')
print(iog_pi['Re'].to_list(),'\n--------\n')
```

Data Format

- `iog`: io general, used in LPC
  - Needs to write your own parser or dump as text file
  - Interface to `pandas` dataframe and to `HDF5`, `h5py` are provided

```python
#!/usr/bin/env python3
# encoding: utf-8
from iog_reader import iog_read

iog_file = "./2pt_5350.dat.iog"
intrptr = ["cnfg","hdrn","t"]

iog = iog_read(iog_file,intrptr)
print(iog,'\n--------\n')

iog_pi=iog.loc[(iog['hdrn']=='0')]
print(iog_pi['Re'].to_numpy(),'\n--------\n')
print(iog_pi['Re'].to_list(),'\n--------\n')
```

Data Format

- `iog`: io general, used in LPC
  - Needs to write your own parser or dump as text file
  - Interface to `pandas` dataframe and to `HDF5`, `h5py` are provided

```python
#!/usr/bin/env python3
# encoding: utf-8
from iog_reader import iog_read

iog_file = "./2pt_5350.dat.iog"
intrptr = ["cnfg","hdrn","t"]

iog = iog_read(iog_file,intrptr)
print(iog,'\n--------\n')

iog_pi=iog.loc[(iog['hdrn']=='0')]
print(iog_pi['Re'].to_numpy(),'\n--------\n')
print(iog_pi['Re'].to_list(),'\n--------\n')
```

Data Format

- `iog`: io general, used in LPC
  - Needs to write your own parser or dump as text file
  - Interface to `pandas` dataframe and to `HDF5`, `h5py` are provided

```python
#!/usr/bin/env python3
# encoding: utf-8
from iog_reader import iog_read

iog_file = "./2pt_5350.dat.iog"
intrptr = ["cnfg","hdrn","t"]

iog = iog_read(iog_file,intrptr)
print(iog,'\n--------\n')

iog_pi=iog.loc[(iog['hdrn']=='0')]
print(iog_pi['Re'].to_numpy(),'\n--------\n')
print(iog_pi['Re'].to_list(),'\n--------\n')
```

Data Format

- `iog`: io general, used in LPC
  - Needs to write your own parser or dump as text file
  - Interface to `pandas` dataframe and to `HDF5`, `h5py` are provided

```python
#!/usr/bin/env python3
# encoding: utf-8
from iog_reader import iog_read

iog_file = "./2pt_5350.dat.iog"
intrptr = ["cnfg","hdrn","t"]

iog = iog_read(iog_file,intrptr)
print(iog,'\n--------\n')

iog_pi=iog.loc[(iog['hdrn']=='0')]
print(iog_pi['Re'].to_numpy(),'\n--------\n')
print(iog_pi['Re'].to_list(),'\n--------\n')
```

Data Format

- `iog`: io general, used in LPC
  - Needs to write your own parser or dump as text file
  - Interface to `pandas` dataframe and to `HDF5`, `h5py` are provided

```python
#!/usr/bin/env python3
# encoding: utf-8
from iog_reader import iog_read

iog_file = "./2pt_5350.dat.iog"
intrptr = ["cnfg","hdrn","t"]

iog = iog_read(iog_file,intrptr)
print(iog,'\n--------\n')

iog_pi=iog.loc[(iog['hdrn']=='0')]
print(iog_pi['Re'].to_numpy(),'\n--------\n')
print(iog_pi['Re'].to_list(),'\n--------\n')
```

Data Format

- `iog`: io general, used in LPC
  - Needs to write your own parser or dump as text file
  - Interface to `pandas` dataframe and to `HDF5`, `h5py` are provided

```python
#!/usr/bin/env python3
# encoding: utf-8
from iog_reader import iog_read

iog_file = "./2pt_5350.dat.iog"
intrptr = ["cnfg","hdrn","t"]

iog = iog_read(iog_file,intrptr)
print(iog,'\n--------\n')

iog_pi=iog.loc[(iog['hdrn']=='0')]
print(iog_pi['Re'].to_numpy(),'\n--------\n')
print(iog_pi['Re'].to_list(),'\n--------\n')
```

- Detailed usage of `pandas.DataFrame.loc`: see documentation

- `iog_reader.py` : `iog` ⇨ pandas.dataframe

```python
#!/usr/bin/env python3
# encoding: utf-8

from ctypes import *
import os
import numpy as np
import pandas as pd

lib = cdll.LoadLibrary(os.getcwd()+ '/iog.so')

def iog_read(iog_fl, intrprtrs):
    iog_file = c_char_p(iog_fl.encode('utf-8'))
    size = lib.getsize(iog_file)

    class iog_type(Structure):
        _fields_ = [('info_shape', POINTER(c_int)), ('info', POINTER(c_int)),
                    ('Re', POINTER(c_double)),  ('Im', POINTER(c_double))]

    lib.getdat.restype = POINTER(iog_type)
    iog = lib.getdat(iog_file)

    info_shape = np.ctypeslib.as_array(iog.contents.info_shape, shape=[2])
    info = np.ctypeslib.as_array(iog.contents.info, shape=[np.prod(info_shape)])
    re = np.ctypeslib.as_array(iog.contents.Re, shape=[size])
```

- `iog_reader.py`: `iog` ➩ pandas.dataframe

```python
#!/usr/bin/env python3
# encoding: utf-8

from ctypes import *
import os
import numpy as np
import pandas as pd

lib = cdll.LoadLibrary(os.getcwd()+ '/iog.so')

def iog_read(iog_fl, intrprtrs):
    iog_file = c_char_p(iog_fl.encode('utf-8'))
    size = lib.getsize(iog_file)

    class iog_type(Structure):
        _fields_ = [('info_shape', POINTER(c_int)), ('info', POINTER(c_int)),
                    ('Re', POINTER(c_double)),  ('Im', POINTER(c_double))]

    lib.getdat.restype = POINTER(iog_type)
    iog = lib.getdat(iog_file)

    info_shape = np.ctypeslib.as_array(iog.contents.info_shape, shape=[2])
    info = np.ctypeslib.as_array(iog.contents.info, shape=[np.prod(info_shape)])
    re = np.ctypeslib.as_array(iog.contents.Re, shape=[size])
```

- `iog_reader.py`: `iog` ⇨ pandas.dataframe

```python
#!/usr/bin/env python3
# encoding: utf-8

from ctypes import *
import os
import numpy as np
import pandas as pd

lib = cdll.LoadLibrary(os.getcwd()+ '/iog.so')

def iog_read(iog_fl, intrprtrs):
    iog_file = c_char_p(iog_fl.encode('utf-8'))
    size = lib.getsize(iog_file)

    class iog_type(Structure):
        _fields_ = [('info_shape', POINTER(c_int)), ('info', POINTER(c_int)),
                    ('Re', POINTER(c_double)),  ('Im', POINTER(c_double))]

    lib.getdat.restype = POINTER(iog_type)
    iog = lib.getdat(iog_file)

    info_shape = np.ctypeslib.as_array(iog.contents.info_shape, shape=[2])
    info = np.ctypeslib.as_array(iog.contents.info, shape=[np.prod(info_shape)])
    re = np.ctypeslib.as_array(iog.contents.Re, shape=[size])
```

- In case you need to compile ``iog.so`` by yourself

- In case you need to compile `` `iog.so` `` by yourself
- Source located under `` `demo/iog` ``

- In case you need to compile `` `iog.so` `` by yourself
- Source located under `` `demo/iog` ``
- Set the value of `` `INCL` `` pointing to the path containing python develop tools header in `` `Makefile` ``

  `` `INCL = -I/dssg/opt/icelake/linux-centos8-icelake/gcc-8.3.1/python-3.9.12-omx7ttx5qbayml44yyhwifxuooiebvh7/include` ``

- In case you need to compile `` `iog.so` `` by yourself
- Source located under `` `demo/iog` ``
- Set the value of `` `INCL` `` pointing to the path containing python develop tools header in `` `Makefile` ``

  `` `INCL = -I/dssg/opt/icelake/linux-centos8-icelake/gcc-8.3.1/python-3.9.12-omx7ttx5qbayml44yyhwifxuooiebvh7/include` ``
- Execute `` `make iog.so` ``

- Output of `` `iog.py` ``

- Output of `iog.py`

```
        cnfg hdrn   t              Re             Im
0       5350    0   0   591279.959489       0.000005
1       5350    0   1   522915.505562      -0.000003
2       5350    0   2   482360.005234      -0.000012
3       5350    0   3   440684.923986      -0.000002
4       5350    0   4   387859.756810       0.000004
...      ...  ...  ..             ...            ...
1147    5350   15  67    26763.208835    7141.283387
1148    5350   15  68    42259.980047   12240.936307
1149    5350   15  69    79407.008466   17624.534133
1150    5350   15  70   141667.387280   25814.545022
1151    5350   15  71   245504.136076   36646.858214

[1152 rows x 5 columns]
--------

[591279.9594888687, 522915.5055618286,...
--------

[591279.95948887 522915.50556183 482360.00523376 ...
```

- Output of `iog.py`

```
        cnfg hdrn   t             Re            Im
0       5350    0   0  591279.959489      0.000005
1       5350    0   1  522915.505562     -0.000003
2       5350    0   2  482360.005234     -0.000012
3       5350    0   3  440684.923986     -0.000002
4       5350    0   4  387859.756810      0.000004
...      ...  ...  ..            ...           ...
1147    5350   15  67   26763.208835   7141.283387
1148    5350   15  68   42259.980047  12240.936307
1149    5350   15  69   79407.008466  17624.534133
1150    5350   15  70  141667.387280  25814.545022
1151    5350   15  71  245504.136076  36646.858214

[1152 rows x 5 columns]
--------

[591279.9594888687, 522915.5055618286,...
--------

[591279.95948887 522915.50556183 482360.00523376 ...
```

- Output of `iog.py`

```
        cnfg hdrn   t             Re            Im
0       5350    0   0  591279.959489      0.000005
1       5350    0   1  522915.505562     -0.000003
2       5350    0   2  482360.005234     -0.000012
3       5350    0   3  440684.923986     -0.000002
4       5350    0   4  387859.756810      0.000004
...      ...  ...  ..            ...           ...
1147    5350   15  67   26763.208835   7141.283387
1148    5350   15  68   42259.980047  12240.936307
1149    5350   15  69   79407.008466  17624.534133
1150    5350   15  70  141667.387280  25814.545022
1151    5350   15  71  245504.136076  36646.858214

[1152 rows x 5 columns]
--------

[591279.9594888687, 522915.5055618286,...
--------

[591279.95948887 522915.50556183 482360.00523376 ...
```

- Output of `iog.py`

```
        cnfg  hdrn   t              Re             Im
0       5350     0   0   591279.959489       0.000005
1       5350     0   1   522915.505562      -0.000003
2       5350     0   2   482360.005234      -0.000012
3       5350     0   3   440684.923986      -0.000002
4       5350     0   4   387859.756810       0.000004
...      ...   ...  ..             ...            ...
1147    5350    15  67    26763.208835    7141.283387
1148    5350    15  68    42259.980047   12240.936307
1149    5350    15  69    79407.008466   17624.534133
1150    5350    15  70   141667.387280   25814.545022
1151    5350    15  71   245504.136076   36646.858214

[1152 rows x 5 columns]
--------

[591279.9594888687, 522915.5055618286,...
--------

[591279.95948887 522915.50556183 482360.00523376 ...
```

- Output of `iog.py`

```
        cnfg hdrn   t             Re            Im
0       5350    0   0  591279.959489      0.000005
1       5350    0   1  522915.505562     -0.000003
2       5350    0   2  482360.005234     -0.000012
3       5350    0   3  440684.923986     -0.000002
4       5350    0   4  387859.756810      0.000004
...      ...  ...  ..            ...           ...
1147    5350   15  67   26763.208835   7141.283387
1148    5350   15  68   42259.980047  12240.936307
1149    5350   15  69   79407.008466  17624.534133
1150    5350   15  70  141667.387280  25814.545022
1151    5350   15  71  245504.136076  36646.858214

[1152 rows x 5 columns]
--------

[591279.9594888687, 522915.5055618286,...
--------

[591279.95948887 522915.50556183 482360.00523376 ...
```

- Output of `iog.py`

```
       cnfg hdrn   t             Re            Im
0      5350    0   0  591279.959489      0.000005
1      5350    0   1  522915.505562     -0.000003
2      5350    0   2  482360.005234     -0.000012
3      5350    0   3  440684.923986     -0.000002
4      5350    0   4  387859.756810      0.000004
...     ...  ...  ..            ...           ...
1147   5350   15  67   26763.208835   7141.283387
1148   5350   15  68   42259.980047  12240.936307
1149   5350   15  69   79407.008466  17624.534133
1150   5350   15  70  141667.387280  25814.545022
1151   5350   15  71  245504.136076  36646.858214

[1152 rows x 5 columns]
--------

[591279.9594888687, 522915.5055618286,...
--------

[591279.95948887 522915.50556183 482360.00523376 ...
```

string

- Output of `iog.py`

```
       cnfg hdrn   t             Re            Im
0      5350    0   0  591279.959489      0.000005
1      5350    0   1  522915.505562     -0.000003
2      5350    0   2  482360.005234     -0.000012
3      5350    0   3  440684.923986     -0.000002
4      5350    0   4  387859.756810      0.000004
...     ...  ...  ..            ...           ...
1147   5350   15  67   26763.208835   7141.283387
1148   5350   15  68   42259.980047  12240.936307
1149   5350   15  69   79407.008466  17624.534133
1150   5350   15  70  141667.387280  25814.545022
1151   5350   15  71  245504.136076  36646.858214

[1152 rows x 5 columns]
--------


[591279.9594888687, 522915.5055618286,...
--------


[591279.95948887 522915.50556183 482360.00523376 ...
```

string

- Filter by value: `iog.loc[18<=iog['t'].astype(int)]`
- Use logical operator: `iog_pi=iog.loc[(iog['hdrn']=='0') & (iog['t'].astype(int)<18)]`

What is intrptr ? - iog convention

```
       cnfg hdrn   t               Re           Im
0      5350    0   0   591279.959489     0.000005
1      5350    0   1   522915.505562    -0.000003
2      5350    0   2   482360.005234    -0.000012
3      5350    0   3   440684.923986    -0.000002
4      5350    0   4   387859.756810     0.000004
...     ...  ...  ..             ...          ...
```

What is intrptr ? - iog convention

```
         cnfg hdrn   t               Re          Im
0        5350    0   0  591279.959489    0.000005
1        5350    0   1  522915.505562   -0.000003
2        5350    0   2  482360.005234   -0.000012
3        5350    0   3  440684.923986   -0.000002
4        5350    0   4  387859.756810    0.000004
...       ...  ...  ..            ...         ...
```

What is intrptr ? - iog convention

```
          cnfg hdrn   t               Re          Im
   0      5350    0   0    591279.959489    0.000005
   1      5350    0   1    522915.505562   -0.000003
   2      5350    0   2    482360.005234   -0.000012
   3      5350    0   3    440684.923986   -0.000002
   4      5350    0   4    387859.756810    0.000004
   ...     ...  ...  ..              ...         ...
```

- `cnfg`: configuration number

What is intrptr ? - iog convention

```
          cnfg hdrn    t             Re          Im
0         5350    0    0  591279.959489    0.000005
1         5350    0    1  522915.505562   -0.000003
2         5350    0    2  482360.005234   -0.000012
3         5350    0    3  440684.923986   -0.000002
4         5350    0    4  387859.756810    0.000004
...        ...  ...   ..            ...         ...
```

- `cnfg` : configuration number
- `hdrn` (hadron) : ["PION", "KAON", "PROTON", "RHO", "D", "ETAS", "ETAC", "JPSI", "LAMBDA", "LAMBDAC", "SIGMA", "XI", "XIC", "OMEGA", "DELTA"]

What is intrptr ? - iog convention

```
          cnfg hdrn   t             Re           Im
0         5350    0   0  591279.959489     0.000005
1         5350    0   1  522915.505562    -0.000003
2         5350    0   2  482360.005234    -0.000012
3         5350    0   3  440684.923986    -0.000002
4         5350    0   4  387859.756810     0.000004
...        ...  ...  ..            ...          ...
```

- `cnfg`: configuration number
- `hdrn` (hadron) : ["PION", "KAON", "PROTON", "RHO", "D", "ETAS", "ETAC", "JPSI", "LAMBDA", "LAMBDAC", "SIGMA", "XI", "XIC", "OMEGA", "DELTA"]
- `t`: time slice

What is intrptr ? - iog convention

```
          cnfg hdrn   t             Re          Im
0         5350    0   0  591279.959489    0.000005
1         5350    0   1  522915.505562   -0.000003
2         5350    0   2  482360.005234   -0.000012
3         5350    0   3  440684.923986   -0.000002
4         5350    0   4  387859.756810    0.000004
...        ...  ...  ..            ...         ...
```

- `cnfg`: configuration number
- `hdrn` (hadron) : ["PION", "KAON", "PROTON", "RHO", "D", "ETAS", "ETAC", "JPSI", "LAMBDA", "LAMBDAC", "SIGMA", "XI", "XIC", "OMEGA", "DELTA"]
- `t`: time slice
- `Re`, `Im`: real and imaginary part of two-point function

> theoretically, 2pt should be real valued, but due to numberical errors, it may have a neglectable smallimaginary part

more complicated iog convention (not in this traning camp)

```
        cnfg tsrc lnk snksmr      hdrn    mntm   t   Re   Im
0       2125   18   0      0    200505  505050   0  0.0  0.0
1       2125   18   0      0    200505  505050   1  0.0  0.0
...      ...  ...  ..    ...       ...     ... ...  ...  ...
4606    2125   18   0      1 100001515  505045 126  0.0  0.0
4607    2125   18   0      1 100001515  505045 127  0.0  0.0
```

more complicated iog convention (not in this traning camp)

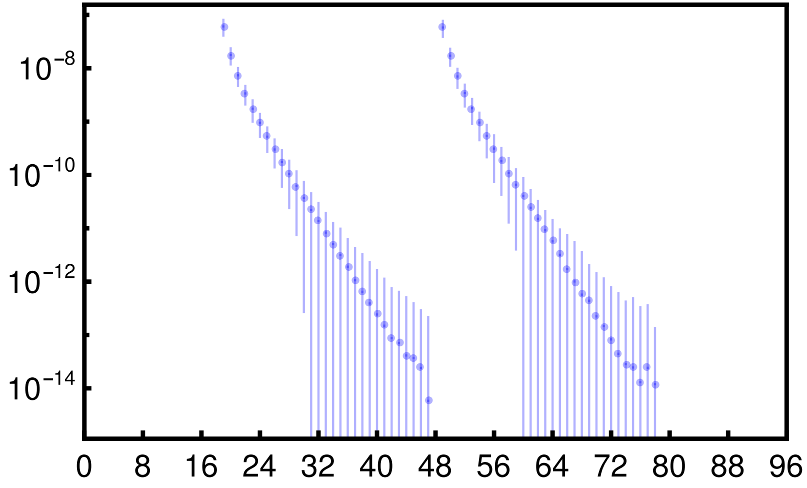
```
        cnfg tsrc lnk snksmr       hdrn    mntm    t   Re   Im
0       2125   18   0      0     200505  505050    0  0.0  0.0
1       2125   18   0      0     200505  505050    1  0.0  0.0
...      ...  ...  ..    ...        ...     ...  ...  ...  ...
4606    2125   18   0      1  100001515  505045  126  0.0  0.0
4607    2125   18   0      1  100001515  505045  127  0.0  0.0
```

- `cnfg`: configuration number

more complicated iog convention (not in this traning camp)

```
        cnfg tsrc lnk snksmr       hdrn    mntm    t   Re   Im
0       2125   18   0      0     200505  505050    0  0.0  0.0
1       2125   18   0      0     200505  505050    1  0.0  0.0
...      ...  ...  ..    ...        ...     ...  ...  ...  ...
4606    2125   18   0      1  100001515  505045  126  0.0  0.0
4607    2125   18   0      1  100001515  505045  127  0.0  0.0
```

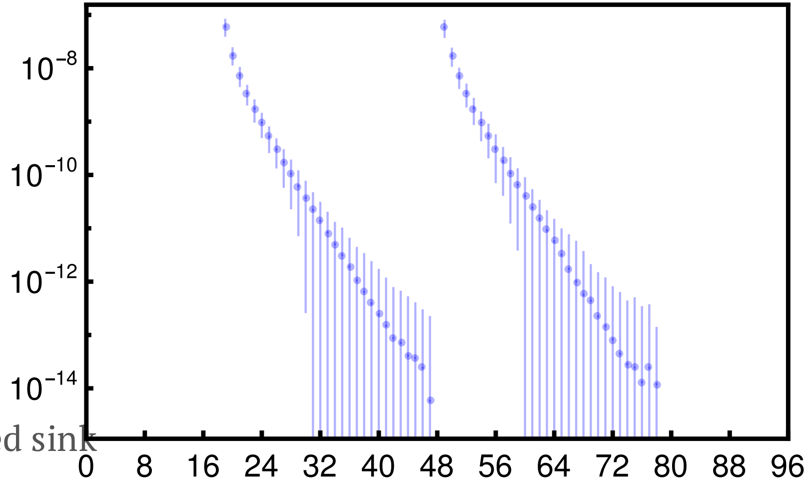
- `cnfg`: configuration number
- `tsrc`: position of 1st source in time direction



more complicated iog convention (not in this traning camp)

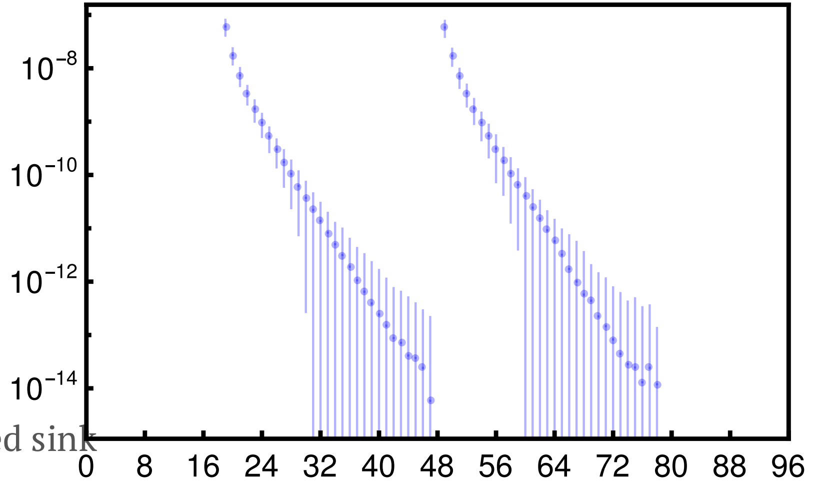
```
        cnfg tsrc lnk snksmr      hdrn    mntm    t   Re   Im
0       2125   18   0      0    200505  505050    0  0.0  0.0
1       2125   18   0      0    200505  505050    1  0.0  0.0
...      ...  ...  ..    ...       ...     ...  ...  ...  ...
4606    2125   18   0      1 100001515  505045  126  0.0  0.0
4607    2125   18   0      1 100001515  505045  127  0.0  0.0
```

- `cnfg`: configuration number
- `tsrc`: position of 1st source in time direction
- `snksmr` (sink smear): `0` ⇨ unsmeared sink, `1` ⇨ smeared sink



more complicated iog convention (not in this traning camp)

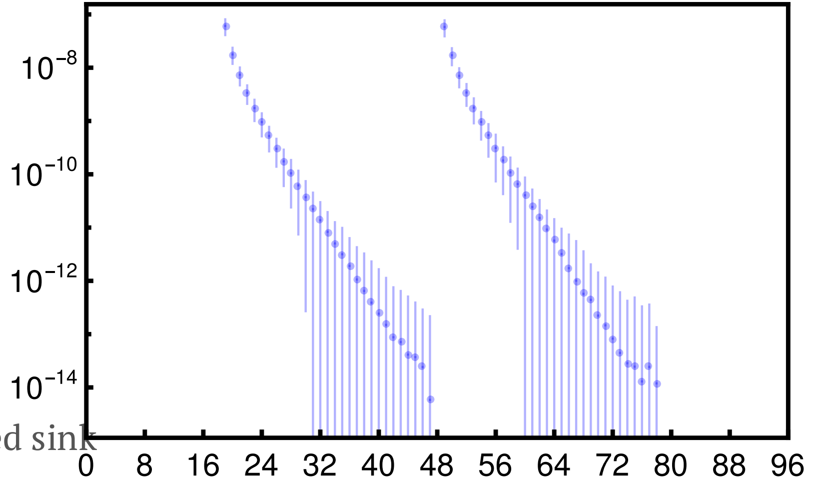
```
        cnfg tsrc lnk snksmr       hdrn    mntm    t   Re   Im
0       2125   18   0      0     200505  505050    0  0.0  0.0
1       2125   18   0      0     200505  505050    1  0.0  0.0
...      ...  ...  ..    ...        ...     ...  ...  ...  ...
4606    2125   18   0      1  100001515  505045  126  0.0  0.0
4607    2125   18   0      1  100001515  505045  127  0.0  0.0
```



- `cnfg`: configuration number
- `tsrc`: position of 1st source in time direction
- `snksmr` (sink smear): `0` ⇨ unsmeared sink, `1` ⇨ smeared sink
- `lnk`: dummy label in two-point data file

more complicated iog convention (not in this traning camp)

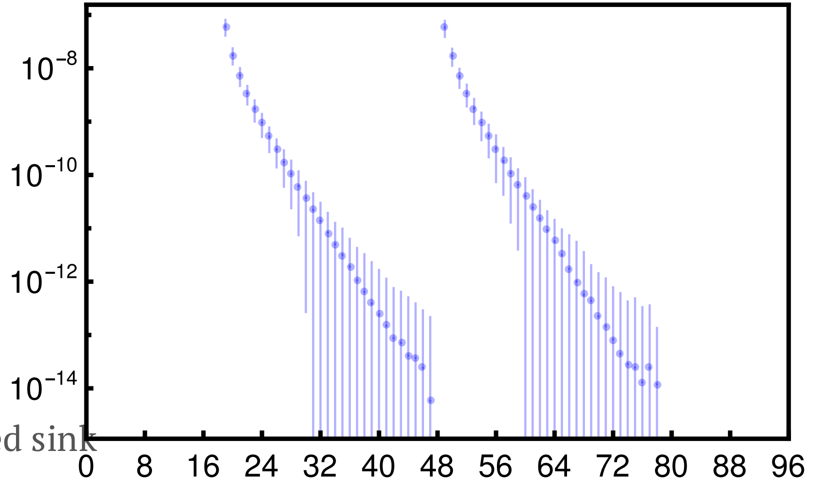
```
        cnfg tsrc lnk snksmr      hdrn   mntm   t  Re  Im
0       2125   18   0      0    200505 505050   0 0.0 0.0
1       2125   18   0      0    200505 505050   1 0.0 0.0
...      ...  ...  ..    ...       ...    ... ... ... ...
4606    2125   18   0      1 100001515 505045 126 0.0 0.0
4607    2125   18   0      1 100001515 505045 127 0.0 0.0
```



- `cnfg`: configuration number
- `tsrc`: position of 1st source in time direction
- `snksmr` (sink smear): `0` ⇨ unsmeared sink, `1` ⇨ smeared sink
- `lnk`: dummy label in two-point data file
- `hdrn` (hadron) : `200505` ⇨ nucleon, `1001515` ⇨ π

more complicated iog convention (not in this traning camp)

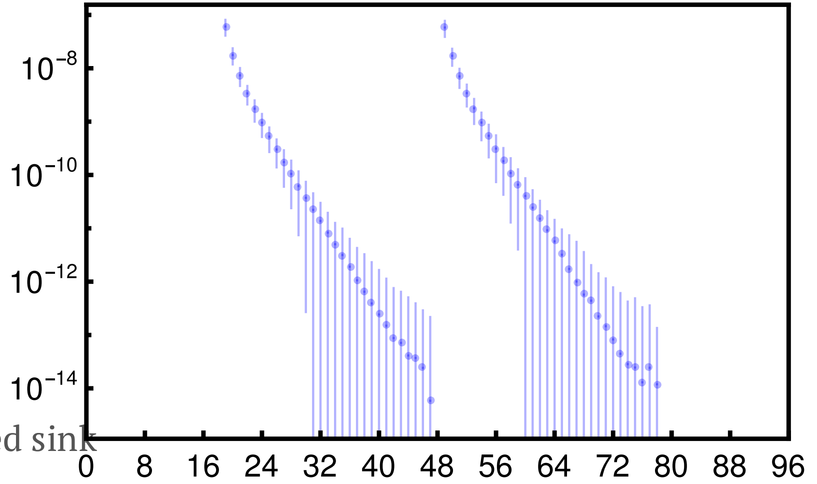
```
        cnfg  tsrc  lnk  snksmr       hdrn    mntm    t   Re   Im
0       2125    18    0       0     200505  505050    0  0.0  0.0
1       2125    18    0       0     200505  505050    1  0.0  0.0
...      ...   ...   ..     ...        ...     ...  ...  ...  ...
4606    2125    18    0       1  100001515  505045  126  0.0  0.0
4607    2125    18    0       1  100001515  505045  127  0.0  0.0
```



- `cnfg`: configuration number
- `tsrc`: position of 1st source in time direction
- `snksmr` (sink smear): `0` ⇨ unsmeared sink, `1` ⇨ smeared sink
- `lnk`: dummy label in two-point data file
- `hdrn` (hadron) : `200505` ⇨ nucleon, `1001515` ⇨ π
- `mntm (momentum)`: *next slide*

more complicated iog convention (not in this traning camp)

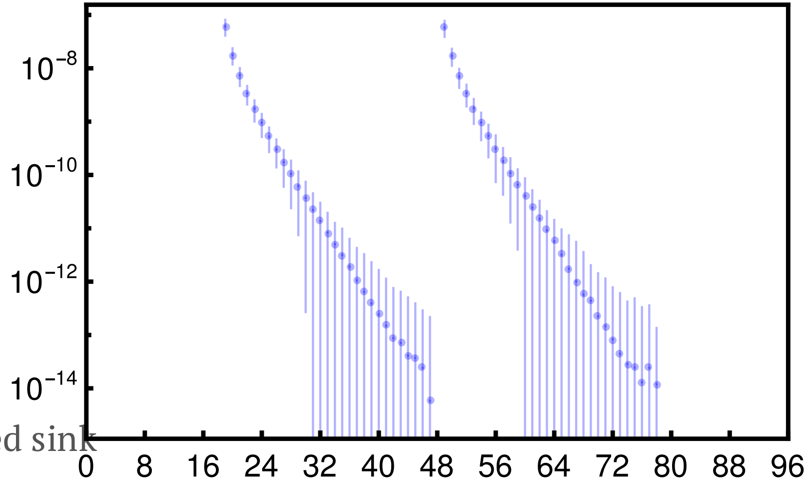
```
        cnfg tsrc lnk snksmr      hdrn    mntm    t   Re   Im
0       2125   18   0      0    200505  505050    0  0.0  0.0
1       2125   18   0      0    200505  505050    1  0.0  0.0
...      ...  ...  ..    ...       ...     ...  ...  ...  ...
4606    2125   18   0      1 100001515  505045  126  0.0  0.0
4607    2125   18   0      1 100001515  505045  127  0.0  0.0
```



- `cnfg`: configuration number
- `tsrc`: position of 1st source in time direction
- `snksmr` (sink smear): `0` ⇨ unsmeared sink, `1` ⇨ smeared sink
- `lnk`: dummy label in two-point data file
- `hdrn` (hadron) : `200505` ⇨ nucleon, `1001515` ⇨ π
- `mntm (momentum)`: *next slide*
- `t`: time slices

more complicated iog convention (not in this traning camp)

```
        cnfg tsrc lnk snksmr      hdrn   mntm   t  Re  Im
0       2125   18   0      0    200505 505050   0 0.0 0.0
1       2125   18   0      0    200505 505050   1 0.0 0.0
...      ...  ...  ..    ...       ...    ... ... ... ...
4606    2125   18   0      1 100001515 505045 126 0.0 0.0
4607    2125   18   0      1 100001515 505045 127 0.0 0.0
```



- `cnfg`: configuration number
- `tsrc`: position of 1st source in time direction
- `snksmr` (sink smear): `0` ⇨ unsmeared sink, `1` ⇨ smeared sink
- `lnk`: dummy label in two-point data file
- `hdrn` (hadron) : `200505` ⇨ nucleon, `1001515` ⇨ π
- `mntm (momentum)`: *next slide*
- `t`: time slices
- `Re`, `Im`: real and imaginary part of two-point function

- Momentum label (not used in this traning camp)

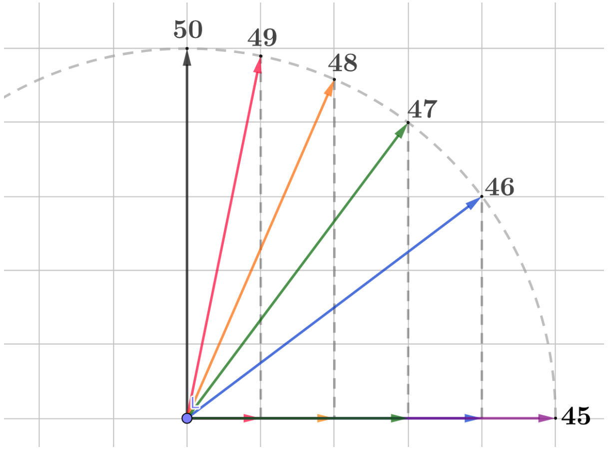
- Momentum label (not used in this traning camp)
- momentum : `` `505050` `` ⇨ 50 50 50 ⇨ $0, 0, 0$ ⇨ $\vec{P} = (0, 0, 0)$
  - lattice momentum unit: $\dfrac{2\pi}{L}, L = na$

    > Discertized Fourier transformation

- Momentum label (not used in this traning camp)
- momentum : `505050` ⇨ 50 50 50 ⇨ 0, 0, 0 ⇨ $\vec{P} = (0,0,0)$
  - lattice momentum unit: $\dfrac{2\pi}{L}, L = na$

  > Discertized Fourier transformation

- $\tilde{p}_x, \tilde{p}_y, \tilde{p}_z$, eg. 50 50 48
  - $\vec{P} = [(50,50,50) - (\tilde{p}_x, \tilde{p}_y, \tilde{p}_z)] \times \dfrac{2\pi}{L} = (0,0,2) \times \dfrac{2\pi}{L}$ ⇨ 2 unit momentum in z-direction



Resampling

- Conventional resampling method
  - Jack-knife
  - Bootstrap

Why Resampling

Why Resampling

- Suppose C is directly calculate on the lattice, $\mathcal{O} = F(\langle C\rangle)$ is what we are intrested in

> F usually is a nonlinear function

- Usually, $\mathcal{O} = F(\langle C\rangle) \neq \langle F(C)\rangle$
- Error estimation of $\mathcal{O}$
  - usually, the target $\mathcal{O}$ has a complicate dependence (usually nonlinear) on lattice calculable $\mathcal{C}$
  - Error propagation $\sigma_{\mathcal{O}} \approx F'(\langle C\rangle)\sigma_{\mathcal{C}}$?
    - 😭 F is nonlinear
    - 😭 $\sigma_{\mathcal{C}}$ is not small enough to neglect $\mathcal{O}(\sigma_{\mathcal{C}}^2)$ term

Why Resampling

- Suppose C is directly calculate on the lattice, $\mathcal{O} = F(\langle C \rangle)$ is what we are intrested in

> F usually is a nonlinear function

- Usually, $\mathcal{O} = F(\langle C \rangle) \neq \langle F(C) \rangle$
- Error estimation of $\mathcal{O}$
  - usually, the target $\mathcal{O}$ has a complicate dependence (usually nonlinear) on lattice calculable $\mathcal{C}$
  - Error propagation $\sigma_{\mathcal{O}} \approx F'(\langle C \rangle)\sigma_{\mathcal{C}}$?
    - 😭 F is nonlinear
    - 😭 $\sigma_{\mathcal{C}}$ is not small enough to neglect $\mathcal{O}(\sigma_{\mathcal{C}}^2)$ term

Solution: Resampling

1. Jack-knife Resampling

1. Jack-knife Resampling

- Original Datasets:
  - cnfg 1 $\Rightarrow \vec{C}^{(1)} = (C_0^{(1)}, \cdots, C_t^{(1)})$
  - $\cdots\cdots\cdots$
  - cnfg $N \Rightarrow \vec{C}^{(N)} = (C_0^{(N)}, \cdots, C_t^{(N)})$

  > time slice: $0, 1, \cdots, t$

1. Jack-knife Resampling

- Original Datasets:
  - cnfg 1 $\Rightarrow \vec{C}^{(1)} = (C_0^{(1)}, \cdots, C_t^{(1)})$
  - $\cdots\cdots\cdots$
  - cnfg $N \Rightarrow \vec{C}^{(N)} = (C_0^{(N)}, \cdots, C_t^{(N)})$

  > time slice: $0, 1, \cdots, t$

- ith Jack-knife sample
  - $\tilde{C}^{(i)} = \frac{N}{N-1}\bar{C} - \frac{1}{N-1}\vec{C}^{(i)} \Rightarrow$ drop cnfg i, calculate mean value of the rest

1. Jack-knife Resampling

- Original Datasets:
  - cnfg 1 ⇨ $\vec{C}^{(1)} = (C_0^{(1)}, \cdots, C_t^{(1)})$
  - $\cdots\cdots\cdots$
  - cnfg N ⇨ $\vec{C}^{(N)} = (C_0^{(N)}, \cdots, C_t^{(N)})$

  > time slice: $0, 1, \cdots, t$

- ith Jack-knife sample
  - $\tilde{C}^{(i)} = \frac{N}{N-1}\bar{C} - \frac{1}{N-1}\vec{C}^{(i)}$ ⇨ drop cnfg i, calculate mean value of the rest
- Estimator
  - $\bar{C} = \frac{1}{N}\sum_i \vec{C}^{(i)} = \frac{1}{N}\sum_i \tilde{C}^{(i)}$
  - $\mathrm{cov} = (N-1) \times \tilde{\mathrm{cov}}, \tilde{\mathrm{cov}}$: covaiance matrix of Jack-knife samples
  - $\sigma = \sqrt{N-1} \times \tilde{\sigma}$

- Jacknife samples are highly correlated
  - Mean of almost identical data
  - $\left(\tilde{C}^{(i)} - \bar{C}\right)^2 = \left(\frac{N}{N-1}\bar{C} - \frac{1}{N-1}\vec{C}^{(i)} - \bar{C}\right)^2 = \frac{1}{(N-1)^2}\left(\vec{C}^{(i)} - \bar{C}\right)^2$

Coding tips

Coding tips

- Vectroize your code instead of using loop

Coding tips

- Vectroize your code instead of using loop

```python
#!/usr/bin/env python3
# encoding: utf-8

import numpy as np
from os import listdir
from iog_reader import iog_read
from re import match
import cProfile
from functools import reduce

iog_path = "./Data/"
T = 72

files = listdir(iog_path)
iogs = [file for file in files if match(r".*2pt.*.dat.iog",file)!=None]
Ncnfg = len(iogs)

c2pt = np.zeros((Ncnfg,T))
for indx in range(0,Ncnfg):
    dat=iog_read(iog_path+iogs[indx],["cnfg","hdrn","t"])
    c2pt[indx] = dat.loc[(dat["hdrn"]=="0")]["Re"].to_numpy()

c2pt_jcknf_1 = np.zeros((Ncnfg,T))
c2pt_jcknf_2 = np.zeros((Ncnfg,T))
```

Coding tips

- Vectroize your code instead of using loop

```
#!/usr/bin/env python3
# encoding: utf-8

import numpy as np
from os import listdir
from iog_reader import iog_read
from re import match
import cProfile
from functools import reduce

iog_path = "./Data/"
T = 72

files = listdir(iog_path)
iogs = [file for file in files if match(r".*2pt.*.dat.iog",file)!=None]
Ncnfg = len(iogs)

c2pt = np.zeros((Ncnfg,T))
for indx in range(0,Ncnfg):
    dat=iog_read(iog_path+iogs[indx],["cnfg","hdrn","t"])
    c2pt[indx] = dat.loc[(dat["hdrn"]=="0")]["Re"].to_numpy()

c2pt_jcknf_1 = np.zeros((Ncnfg,T))
c2pt_jcknf_2 = np.zeros((Ncnfg,T))
```

Coding tips

- Vectroize your code instead of using loop

```python
#!/usr/bin/env python3
# encoding: utf-8

import numpy as np
from os import listdir
from iog_reader import iog_read
from re import match
import cProfile
from functools import reduce

iog_path = "./Data/"
T = 72

files = listdir(iog_path)
iogs = [file for file in files if match(r".*2pt.*.dat.iog",file)!=None]
Ncnfg = len(iogs)

c2pt = np.zeros((Ncnfg,T))
for indx in range(0,Ncnfg):
    dat=iog_read(iog_path+iogs[indx],["cnfg","hdrn","t"])
    c2pt[indx] = dat.loc[(dat["hdrn"]=="0")]["Re"].to_numpy()

c2pt_jcknf_1 = np.zeros((Ncnfg,T))
c2pt_jcknf_2 = np.zeros((Ncnfg,T))
```

Coding tips

- Vectroize your code instead of using loop

```python
#!/usr/bin/env python3
# encoding: utf-8

import numpy as np
from os import listdir
from iog_reader import iog_read
from re import match
import cProfile
from functools import reduce

iog_path = "./Data/"
T = 72

files = listdir(iog_path)
iogs = [file for file in files if match(r".*2pt.*.dat.iog",file)!=None]
Ncnfg = len(iogs)

c2pt = np.zeros((Ncnfg,T))
for indx in range(0,Ncnfg):
    dat=iog_read(iog_path+iogs[indx],["cnfg","hdrn","t"])
    c2pt[indx] = dat.loc[(dat["hdrn"]=="0")]["Re"].to_numpy()

c2pt_jcknf_1 = np.zeros((Ncnfg,T))
c2pt_jcknf_2 = np.zeros((Ncnfg,T))
```

Coding tips

- Vectroize your code instead of using loop

```python
#!/usr/bin/env python3
# encoding: utf-8

import numpy as np
from os import listdir
from iog_reader import iog_read
from re import match
import cProfile
from functools import reduce

iog_path = "./Data/"
T = 72

files = listdir(iog_path)
iogs = [file for file in files if match(r".*2pt.*.dat.iog",file)!=None]
Ncnfg = len(iogs)

c2pt = np.zeros((Ncnfg,T))
for indx in range(0,Ncnfg):
    dat=iog_read(iog_path+iogs[indx],["cnfg","hdrn","t"])
    c2pt[indx] = dat.loc[(dat["hdrn"]=="0")]["Re"].to_numpy()

c2pt_jcknf_1 = np.zeros((Ncnfg,T))
c2pt_jcknf_2 = np.zeros((Ncnfg,T))
```

Coding tips

- Vectroize your code instead of using loop

```python
#!/usr/bin/env python3
# encoding: utf-8

import numpy as np
from os import listdir
from iog_reader import iog_read
from re import match
import cProfile
from functools import reduce

iog_path = "./Data/"
T = 72

files = listdir(iog_path)
iogs = [file for file in files if match(r".*2pt.*.dat.iog",file)!=None]
Ncnfg = len(iogs)

c2pt = np.zeros((Ncnfg,T))
for indx in range(0,Ncnfg):
    dat=iog_read(iog_path+iogs[indx],["cnfg","hdrn","t"])
    c2pt[indx] = dat.loc[(dat["hdrn"]=="0")]["Re"].to_numpy()

c2pt_jcknf_1 = np.zeros((Ncnfg,T))
c2pt_jcknf_2 = np.zeros((Ncnfg,T))
```

Coding tips

- Vectroize your code instead of using loop

```
         115780 function calls in 0.082 seconds
   Ordered by: standard name
   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
     4824    0.003    0.000    0.039    0.000 <__array_function__ internals>:177(delete)
     4824    0.002    0.000    0.036    0.000 <__array_function__ internals>:177(mean)
     ......
 ------
         1612 function calls in 0.001 seconds
   Ordered by: standard name
   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
       67    0.000    0.000    0.001    0.000 <__array_function__ internals>:177(delete)
       67    0.000    0.000    0.001    0.000 <__array_function__ internals>:177(mean)
    ......
------
         13 function calls in 0.000 seconds
   Ordered by: standard name
   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.000    0.000 <__array_function__ internals>:177(sum)
        1    0.000    0.000    0.000    0.000 <string>:1(<module>)
    ......
------
   [ True  True  True  True  True  True  True  True  True  True  True  True ......
```

```
c2pt_jcknf_2 = np.zeros((Ncnfg,T))
```

2. Bootstrap Resampling

2. Bootstrap Resampling

- Original Datasets:
  - cnfg 1 $\Rightarrow \vec{C}^{(1)} = (C_0^{(1)}, \cdots, C_t^{(1)})$
  - $\cdots\cdots\cdots$
  - cnfg $N \Rightarrow \vec{C}^{(N)} = (C_0^{(N)}, \cdots, C_t^{(N)})$

  > time slice: $0, 1, \cdots, t$

2. Bootstrap Resampling

- Original Datasets:
  - cnfg 1 ⇨ $\vec{C}^{(1)} = (C_0^{(1)}, \cdots, C_t^{(1)})$
  - $\cdots\cdots\cdots$
  - cnfg N ⇨ $\vec{C}^{(N)} = (C_0^{(N)}, \cdots, C_t^{(N)})$

  > time slice: $0, 1, \cdots, t$

- ith Bootstrap sample
  - Randomly (uniformly) sample N cnfgs from original datasets, **allow repetition**,
  - $\tilde{C}^{(i)}$ ⇨ mean of previsouly sampled data

2. Bootstrap Resampling

- Original Datasets:
  - cnfg 1 ⇨ $\vec{C}^{(1)} = (C_0^{(1)}, \cdots, C_t^{(1)})$
  - $\cdots\cdots\cdots$
  - cnfg N ⇨ $\vec{C}^{(N)} = (C_0^{(N)}, \cdots, C_t^{(N)})$

    time slice: $0, 1, \cdots, t$

- ith Bootstrap sample
  - Randomly (uniformly) sample N cnfgs from original datasets, **allow repetition**,
  - $\tilde{C}^{(i)}$ ⇨ mean of previsouly sampled data

- Repeat resampling M times, $M > N$, usually $M \sim \mathcal{O}(10^3)$
  - resampled dataset ⇨ $\left(\tilde{C}^{(1)}, \cdots, \tilde{C}^{(M)}\right)$

- Estimator
  - $\bar{C} = \frac{1}{N} \sum_i \vec{C}^{(i)} = \frac{1}{N} \sum_i \tilde{C}^{(i)}$
  - $\mathrm{cov} = \tilde{\mathrm{cov}}, \tilde{\mathrm{cov}}$: covaiance matrix of bootstrap samples
  - $\sigma = \tilde{\sigma}$

Estimator for $\mathcal{O} = F(\langle C \rangle)$

Estimator for $\mathcal{O} = F(\langle C \rangle)$

- $\tilde{C} = \tilde{C}^{(1)}, \tilde{C}^{(2)}, \cdots \tilde{C}^{(N)}$ denote the resampled data

Estimator for $\mathcal{O} = F(\langle C \rangle)$

- $\tilde{C} = \tilde{C}^{(1)}, \tilde{C}^{(2)}, \cdots \tilde{C}^{(N)}$ denote the resampled data
- Calculate $\tilde{\mathcal{O}} = \tilde{\mathcal{O}}^{(1)}, \tilde{\mathcal{O}}^{(2)}, \cdots \tilde{\mathcal{O}}^{(N)} = F(\tilde{C}^{(1)}), F(\tilde{C}^{(2)}), \cdots F(\tilde{C}^{(N)})$

Estimator for $\mathcal{O} = F(\langle C \rangle)$

- $\tilde{C} = \tilde{C}^{(1)}, \tilde{C}^{(2)}, \cdots \tilde{C}^{(N)}$ denote the resampled data
- Calculate $\tilde{\mathcal{O}} = \tilde{\mathcal{O}}^{(1)}, \tilde{\mathcal{O}}^{(2)}, \cdots \tilde{\mathcal{O}}^{(N)} = F(\tilde{C}^{(1)}), F(\tilde{C}^{(2)}), \cdots F(\tilde{C}^{(N)})$
- Centeral value $\bar{\mathcal{O}} = \bar{\tilde{\mathcal{O}}}$

Estimator for $\mathcal{O} = F(\langle C \rangle)$

- $\tilde{C} = \tilde{C}^{(1)}, \tilde{C}^{(2)}, \cdots \tilde{C}^{(N)}$ denote the resampled data
- Calculate $\tilde{\mathcal{O}} = \tilde{\mathcal{O}}^{(1)}, \tilde{\mathcal{O}}^{(2)}, \cdots \tilde{\mathcal{O}}^{(N)} = F(\tilde{C}^{(1)}), F(\tilde{C}^{(2)}), \cdots F(\tilde{C}^{(N)})$
- Centeral value $\bar{\mathcal{O}} = \bar{\tilde{\mathcal{O}}}$
- Error estimation
  - Jack-knife resample ⇨ $\sigma_{\mathcal{O}} = \sqrt{N-1} \times \sigma_{\tilde{\mathcal{O}}}$
  - Bootstrap resample ⇨ $\sigma_{\mathcal{O}} = \sigma_{\tilde{\mathcal{O}}}$

$$\sigma^2_{\tilde{\mathcal{O}}} = \frac{1}{N-1} \sum_i \left(\mathcal{O}^{(i)} - \bar{\mathcal{O}} \right)^2$$

Estimator for $\mathcal{O} = F(\langle C \rangle)$

- $\tilde{C} = \tilde{C}^{(1)}, \tilde{C}^{(2)}, \cdots \tilde{C}^{(N)}$ denote the resampled data
- Calculate $\tilde{\mathcal{O}} = \tilde{\mathcal{O}}^{(1)}, \tilde{\mathcal{O}}^{(2)}, \cdots \tilde{\mathcal{O}}^{(N)} = F(\tilde{C}^{(1)}), F(\tilde{C}^{(2)}), \cdots F(\tilde{C}^{(N)})$
- Centeral value $\bar{\mathcal{O}} = \bar{\tilde{\mathcal{O}}}$
- Error estimation
  - Jack-knife resample ➩ $\sigma_{\mathcal{O}} = \sqrt{N-1} \times \sigma_{\tilde{\mathcal{O}}}$
  - Bootstrap resample ➩ $\sigma_{\mathcal{O}} = \sigma_{\tilde{\mathcal{O}}}$

  $$\sigma^2_{\tilde{\mathcal{O}}} = \frac{1}{N-1} \sum_i \left(\mathcal{O}^{(i)} - \bar{\mathcal{O}} \right)^2$$
- Parallel your code to do $F(\tilde{C}^{(i)})$
  - e.g. MPI: each process calculate one $F(\tilde{C}^{(i)})$

Least fit

Least fit

- Data:
  - cnfg 1 $\Rightarrow \vec{C}^{(1)} = (C_0^{(1)}, \cdots, C_T^{(1)})$
  - $\cdots\cdots\cdots$
  - cnfg $N \Rightarrow \vec{C}^{(N)} = (C_0^{(N)}, \cdots, C_T^{(N)})$

  > $0 \cdots T$: time slice indices $0 \cdots N$: cnfg indices

Least fit

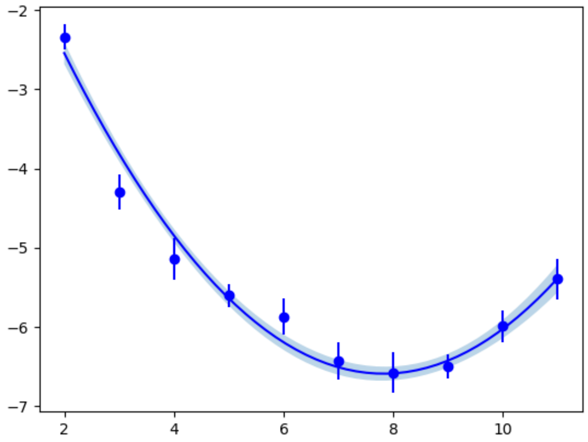
- Data:
  - cnfg 1 ⇨ $\vec{C}^{(1)} = (C_0^{(1)}, \cdots, C_T^{(1)})$
  - $\cdots\cdots\cdots\cdots$
  - cnfg N ⇨ $\vec{C}^{(N)} = (C_0^{(N)}, \cdots, C_T^{(N)})$

  > $0 \cdots T$: time slice indices $0 \cdots N$: cnfg indices

- Model: $f(t, \{p_i\})$
  - t: time slices (w.r.t source)
  - $\{p_i\}$: fit parameters ⇨ $c_0, c_1, E_0, \Delta E$
  - e.g. For two-point function (two-state fit)

$$f(t, \{p_i\}) \implies c_0 e^{-E_0 t}(1 + c_1 e^{-\Delta E t})$$

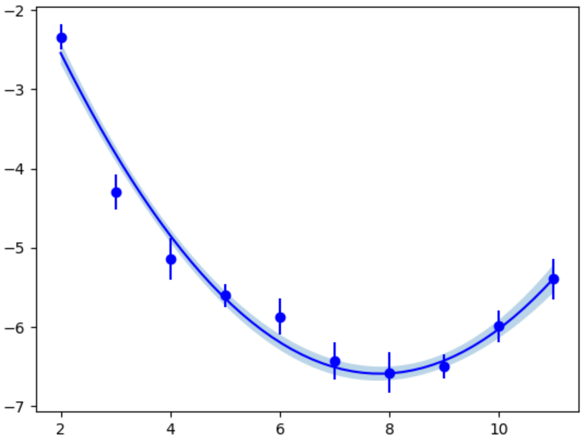
χ^2 definition



χ^2 definition

- Uncorrelated case

$$\chi^2 = \sum_t \frac{\left[\bar{C}_t - f(t, \{p_i\})\right]^2}{\sigma_t^2}$$

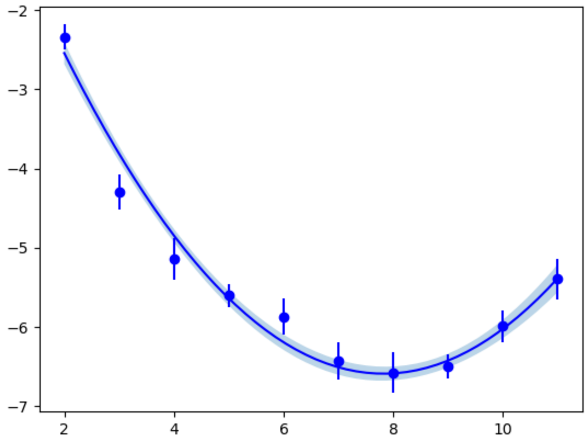


> - $\bar{C}_t$ ⇨ mean value of the t-th timeslices
> - σ_t ⇨ standard deviation of the t-th timeslices

- No interference between different t ⇦⇨ uncorrelated

χ^2 definition

- Uncorrelated case



$$\chi^2 = \sum_t \frac{\left[\bar{C}_t - f(t, \{p_i\})\right]^2}{\sigma_t^2}$$

> - $\bar{C}_t$ ⇨ mean value of the t-th timeslices
> - σ_t ⇨ standard deviation of the t-th timeslices

  - No interference between different t ⇦⇨ uncorrelated

- Correlated case

$$\chi^2 = \sum_{t,t'} \left[\bar{C}_t - f(t, \{p_i\})\right] \left(\mathrm{cov}^{-1}\right)_{t,t'} \left[\bar{C}_{t'} - f(t', \{p_i\})\right]$$

  - cov : covariance matrix

$$(\mathrm{cov})_{t,t'} = \frac{1}{N-1} \sum_i (C_t^{(i)} - \bar{C}_t)(C_{t'}^{(i)} - \bar{C}_{t'})$$

Coding Tips

Coding Tips

- `` `python` `` array index starts from 0

Coding Tips

- `` `python` `` array index starts from 0
- `` `numpy` `` package has build-in `` `mean` ``, `` `std` ``, `` `cov` `` functions
  - e.g. a[2:6] ⇨ a_2, a_3, a_4, a_5

Coding Tips

- `` `python` `` array index starts from 0
- `` `numpy` `` package has build-in `` `mean` ``, `` `std` ``, `` `cov` `` functions
  - e.g. a[2:6] ⇨ a_2, a_3, a_4, a_5
- Vectorize (and broadcast) your code instead of using loops

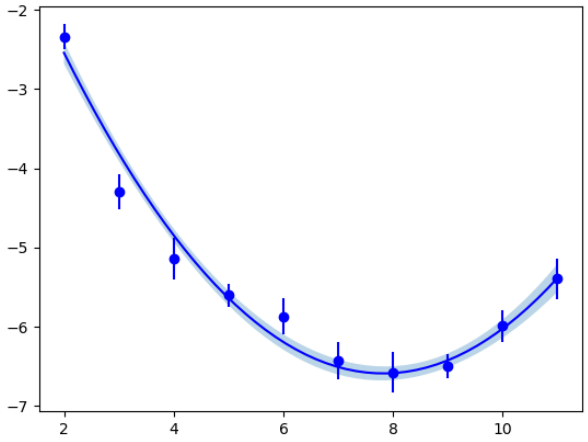
Coding Tips

- `python` array index starts from 0
- `numpy` package has build-in `mean`, `std`, `cov` functions
  - e.g. a[2:6] ⇨ a_2, a_3, a_4, a_5
- Vectorize (and broadcast) your code instead of using loops
- `numpy`: `numpy.mean`, `numpy.std`, `numpy.cov` support high rank tensors
  - Set `axis` parameter in numpy function
  - e.g. $\mathrm{R} = R_{i,j,k,l}, i = 1, \ldots, n_0, j = 1, \ldots, n_1, k = 1, \ldots, n_2, l = 1, \ldots, n_3$
    - Assuming `R, R'` are `numpy` arrays
    - `S=np.mean(R,axis=1)` ⇦⇨ $\mathrm{S}_{i,k,l} = \frac{1}{n_1} \sum_j R_{i,j,k,l}$
    - `R*R'` ⇨ $(\mathrm{R} * \mathrm{R}')_{i,j,k,l} = \mathrm{R}_{i,j,k,l} \times \mathrm{R}'_{i,j,k,l}$

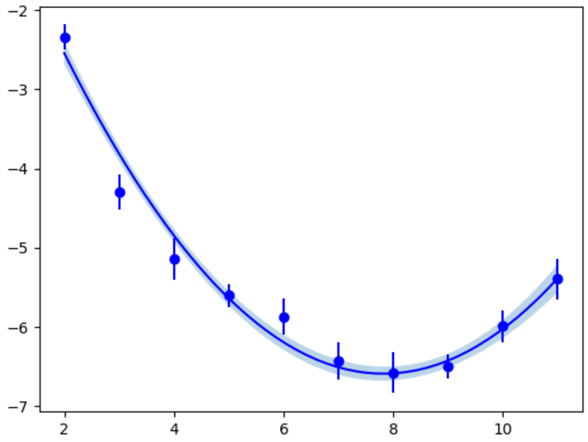
Coding Tips

- `` `python` `` array index starts from 0
- `` `numpy` `` package has build-in `` `mean` ``, `` `std` ``, `` `cov` `` functions
  - e.g. a[2:6] ⇨ a_2, a_3, a_4, a_5
- Vectorize (and broadcast) your code instead of using loops
- `` `numpy` ``: `` `numpy.mean` ``, `` `numpy.std` ``, `` `numpy.cov` `` support high rank tensors
  - Set `` `axis` `` parameter in numpy function
  - e.g. $\mathrm{R} = R_{i,j,k,l}, i = 1, ..., n_0, j = 1, ..., n_1, k = 1, ..., n_2, l = 1, ..., n_3$
    - Assuming `` `R, R'` `` are `` `numpy` `` arrays
    - `` `S=np.mean(R,axis=1)` `` ⇦⇨ $\mathrm{S}_{i,k,l} = \frac{1}{n_1}\sum_j R_{i,j,k,l}$
    - `` `R*R'` `` ⇨ $(\mathrm{R} * \mathrm{R}')_{i,j,k,l} = \mathrm{R}_{i,j,k,l} \times \mathrm{R}'_{i,j,k,l}$
- Not sure about which axis? Print out the shape of R, S: `` `R.shape` ``, `` `S.shape` ``

- $\chi^2 \Rightarrow$ measure the deviation between data and model



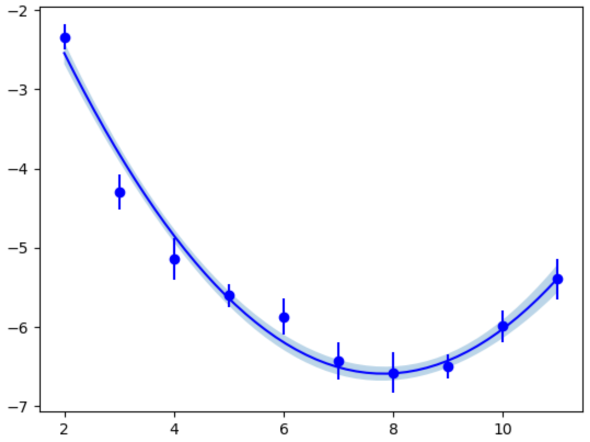
- χ^2 ⇨ measure the deviation between data and model
- least χ^2 fit ⇨ optimize $\{p_i\}$ for minimal **BUT resonable** χ^2



- χ^2 ⇨ measure the deviation between data and model
- least χ^2 fit ⇨ optimize $\{p_i\}$ for minimal **BUT resonable** χ^2
- `lsqfit` ⇨ a python package perform ***l**east* χ-***sq**uare **fit***

```
fit = lsqfit.nonlinear_fit(data=(var_val, dat_val),
                           fcn=model, prior=ini_val, debug=False)
```

  - `var_val` : value of variables ⇨ x-axis
  - `dat_val` : value of datas (with erros) ⇨ y-axis (with err/co
  - `model` : fit model function
  - `ini_val` : initial values of (also constrains on) fit parameters in `model`



`lsqfit` - an example

```python
#!/usr/bin/env python3
# encoding: utf-8

import gvar as gv
import lsqfit
import numpy as np
import matplotlib.pyplot as plt

c2pt_raw = np.load("./data_2pt.npy")
c2pt = c2pt_raw[:, 0, :, -2]
Ncnfg = c2pt.shape[0]
T = c2pt.shape[1]
T_hlf = T//2

#? jack-knife resample
#! only used in plot , compare original data with fit
c2pt_sum = np.sum(c2pt,axis=0)
c2pt_jcknf = ((c2pt_sum) - c2pt)/(Ncnfg-1) # jack-knife resample
c2pt_cntrl = np.mean(c2pt_jcknf,axis=0) # jack-knife mean
c2pt_err = np.sqrt(Ncnfg-1)*np.std(c2pt_jcknf,axis=0) # jack-knife std

#? drop the first data point, normalized to c2[1] ~ 1
t_ary = np.array(range(1,T))
c2pt_jcknf = c2pt_jcknf/c2pt_cntrl[1]
```

`lsqfit` - an example

```
#!/usr/bin/env python3
# encoding: utf-8

import gvar as gv
import lsqfit
import numpy as np
import matplotlib.pyplot as plt

c2pt_raw = np.load("./data_2pt.npy")
c2pt = c2pt_raw[:, 0, :, -2]
Ncnfg = c2pt.shape[0]
T = c2pt.shape[1]
T_hlf = T//2

#? jack-knife resample
#! only used in plot , compare original data with fit
c2pt_sum = np.sum(c2pt,axis=0)
c2pt_jcknf = ((c2pt_sum) - c2pt)/(Ncnfg-1) # jack-knife resample
c2pt_cntrl = np.mean(c2pt_jcknf,axis=0) # jack-knife mean
c2pt_err = np.sqrt(Ncnfg-1)*np.std(c2pt_jcknf,axis=0) # jack-knife std

#? drop the first data point, normalized to c2[1] ~ 1
t_ary = np.array(range(1,T))
c2pt_jcknf = c2pt_jcknf/c2pt_cntrl[1]
```

`lsqfit` - an example

```python
#!/usr/bin/env python3
# encoding: utf-8

import gvar as gv
import lsqfit
import numpy as np
import matplotlib.pyplot as plt

c2pt_raw = np.load("./data_2pt.npy")
c2pt = c2pt_raw[:, 0, :, -2]
Ncnfg = c2pt.shape[0]
T = c2pt.shape[1]
T_hlf = T//2

#? jack-knife resample
#! only used in plot , compare original data with fit
c2pt_sum = np.sum(c2pt,axis=0)
c2pt_jcknf = ((c2pt_sum) - c2pt)/(Ncnfg-1) # jack-knife resample
c2pt_cntrl = np.mean(c2pt_jcknf,axis=0) # jack-knife mean
c2pt_err = np.sqrt(Ncnfg-1)*np.std(c2pt_jcknf,axis=0) # jack-knife std

#? drop the first data point, normalized to c2[1] ~ 1
t_ary = np.array(range(1,T))
c2pt_jcknf = c2pt_jcknf/c2pt_cntrl[1]
```

`lsqfit` - an example

```python
#!/usr/bin/env python3
# encoding: utf-8

import gvar as gv
import lsqfit
import numpy as np
import matplotlib.pyplot as plt

c2pt_raw = np.load("./data_2pt.npy")
c2pt = c2pt_raw[:, 0, :, -2]
Ncnfg = c2pt.shape[0]
T = c2pt.shape[1]
T_hlf = T//2

#? jack-knife resample
#! only used in plot , compare original data with fit
c2pt_sum = np.sum(c2pt,axis=0)
c2pt_jcknf = ((c2pt_sum) - c2pt)/(Ncnfg-1) # jack-knife resample
c2pt_cntrl = np.mean(c2pt_jcknf,axis=0) # jack-knife mean
c2pt_err = np.sqrt(Ncnfg-1)*np.std(c2pt_jcknf,axis=0) # jack-knife std

#? drop the first data point, normalized to c2[1] ~ 1
t_ary = np.array(range(1,T))
c2pt_jcknf = c2pt_jcknf/c2pt_cntrl[1]
```

`lsqfit` - an example

```python
#!/usr/bin/env python3
# encoding: utf-8

import gvar as gv
import lsqfit
import numpy as np
import matplotlib.pyplot as plt

c2pt_raw = np.load("./data_2pt.npy")
c2pt = c2pt_raw[:, 0, :, -2]
Ncnfg = c2pt.shape[0]
T = c2pt.shape[1]
T_hlf = T//2

#? jack-knife resample
#! only used in plot , compare original data with fit
c2pt_sum = np.sum(c2pt,axis=0)
c2pt_jcknf = ((c2pt_sum) - c2pt)/(Ncnfg-1) # jack-knife resample
c2pt_cntrl = np.mean(c2pt_jcknf,axis=0) # jack-knife mean
c2pt_err = np.sqrt(Ncnfg-1)*np.std(c2pt_jcknf,axis=0) # jack-knife std

#? drop the first data point, normalized to c2[1] ~ 1
t_ary = np.array(range(1,T))
c2pt_jcknf = c2pt_jcknf/c2pt_cntrl[1]
```

`lsqfit` - an example

```python
#!/usr/bin/env python3
# encoding: utf-8

import gvar as gv
import lsqfit
import numpy as np
import matplotlib.pyplot as plt

c2pt_raw = np.load("./data_2pt.npy")
c2pt = c2pt_raw[:, 0, :, -2]
Ncnfg = c2pt.shape[0]
T = c2pt.shape[1]
T_hlf = T//2

#? jack-knife resample
#! only used in plot , compare original data with fit
c2pt_sum = np.sum(c2pt,axis=0)
c2pt_jcknf = ((c2pt_sum) - c2pt)/(Ncnfg-1) # jack-knife resample
c2pt_cntrl = np.mean(c2pt_jcknf,axis=0) # jack-knife mean
c2pt_err = np.sqrt(Ncnfg-1)*np.std(c2pt_jcknf,axis=0) # jack-knife std

#? drop the first data point, normalized to c2[1] ~ 1
t_ary = np.array(range(1,T))
c2pt_jcknf = c2pt_jcknf/c2pt_cntrl[1]
```

`lsqfit` - an example

```python
#!/usr/bin/env python3
# encoding: utf-8

import gvar as gv
import lsqfit
import numpy as np
import matplotlib.pyplot as plt

c2pt_raw = np.load("./data_2pt.npy")
c2pt = c2pt_raw[:, 0, :, -2]
Ncnfg = c2pt.shape[0]
T = c2pt.shape[1]
T_hlf = T//2

#? jack-knife resample
#! only used in plot , compare original data with fit
c2pt_sum = np.sum(c2pt,axis=0)
c2pt_jcknf = ((c2pt_sum) - c2pt)/(Ncnfg-1) # jack-knife resample
c2pt_cntrl = np.mean(c2pt_jcknf,axis=0) # jack-knife mean
c2pt_err = np.sqrt(Ncnfg-1)*np.std(c2pt_jcknf,axis=0) # jack-knife std

#? drop the first data point, normalized to c2[1] ~ 1
t_ary = np.array(range(1,T))
c2pt_jcknf = c2pt_jcknf/c2pt_cntrl[1]
```

`lsqfit` - an example

```python
#!/usr/bin/env python3
# encoding: utf-8

import gvar as gv
import lsqfit
import numpy as np
import matplotlib.pyplot as plt

c2pt_raw = np.load("./data_2pt.npy")
c2pt = c2pt_raw[:, 0, :, -2]
Ncnfg = c2pt.shape[0]
T = c2pt.shape[1]
T_hlf = T//2

#? jack-knife resample
#! only used in plot , compare original data with fit
c2pt_sum = np.sum(c2pt,axis=0)
c2pt_jcknf = ((c2pt_sum) - c2pt)/(Ncnfg-1) # jack-knife resample
c2pt_cntrl = np.mean(c2pt_jcknf,axis=0) # jack-knife mean
c2pt_err = np.sqrt(Ncnfg-1)*np.std(c2pt_jcknf,axis=0) # jack-knife std

#? drop the first data point, normalized to c2[1] ~ 1
t_ary = np.array(range(1,T))
c2pt_jcknf = c2pt_jcknf/c2pt_cntrl[1]
```

`lsqfit` - an example

```
#!/usr/bin/env python3
# encoding: utf-8

import gvar as gv
import lsqfit
import numpy as np
import matplotlib.pyplot as plt

c2pt_raw = np.load("./data_2pt.npy")
c2pt = c2pt_raw[:, 0, :, -2]
Ncnfg = c2pt.shape[0]
T = c2pt.shape[1]
T_hlf = T//2

#? jack-knife resample
#! only used in plot , compare original data with fit
c2pt_sum = np.sum(c2pt,axis=0)
c2pt_jcknf = ((c2pt_sum) - c2pt)/(Ncnfg-1) # jack-knife resample
c2pt_cntrl = np.mean(c2pt_jcknf,axis=0) # jack-knife mean
c2pt_err = np.sqrt(Ncnfg-1)*np.std(c2pt_jcknf,axis=0) # jack-knife std

#? drop the first data point, normalized to c2[1] ~ 1
t_ary = np.array(range(1,T))
c2pt_jcknf = c2pt_jcknf/c2pt_cntrl[1]
```

`lsqfit` - an example

```python
#!/usr/bin/env python3
# encoding: utf-8

import gvar as gv
import lsqfit
import numpy as np
import matplotlib.pyplot as plt

c2pt_raw = np.load("./data_2pt.npy")
c2pt = c2pt_raw[:, 0, :, -2]
Ncnfg = c2pt.shape[0]
T = c2pt.shape[1]
T_hlf = T//2

#? jack-knife resample
#! only used in plot , compare original data with fit
c2pt_sum = np.sum(c2pt,axis=0)
c2pt_jcknf = ((c2pt_sum) - c2pt)/(Ncnfg-1) # jack-knife resample
c2pt_cntrl = np.mean(c2pt_jcknf,axis=0) # jack-knife mean
c2pt_err = np.sqrt(Ncnfg-1)*np.std(c2pt_jcknf,axis=0) # jack-knife std

#? drop the first data point, normalized to c2[1] ~ 1
t_ary = np.array(range(1,T))
c2pt_jcknf = c2pt_jcknf/c2pt_cntrl[1]
```

`lsqfit` - an example

```python
#!/usr/bin/env python3
# encoding: utf-8

import gvar as gv
import lsqfit
import numpy as np
import matplotlib.pyplot as plt

c2pt_raw = np.load("./data_2pt.npy")
c2pt = c2pt_raw[:, 0, :, -2]
Ncnfg = c2pt.shape[0]
T = c2pt.shape[1]
T_hlf = T//2

#? jack-knife resample
#! only used in plot , compare original data with fit
c2pt_sum = np.sum(c2pt,axis=0)
c2pt_jcknf = ((c2pt_sum) - c2pt)/(Ncnfg-1) # jack-knife resample
c2pt_cntrl = np.mean(c2pt_jcknf,axis=0) # jack-knife mean
c2pt_err = np.sqrt(Ncnfg-1)*np.std(c2pt_jcknf,axis=0) # jack-knife std

#? drop the first data point, normalized to c2[1] ~ 1
t_ary = np.array(range(1,T))
c2pt_jcknf = c2pt_jcknf/c2pt_cntrl[1]
```

`lsqfit` - an example

```
#!/usr/bin/env python3
# encoding: utf-8

import gvar as gv
import lsqfit
import numpy as np
import matplotlib.pyplot as plt

c2pt_raw = np.load("./data_2pt.npy")
c2pt = c2pt_raw[:, 0, :, -2]
Ncnfg = c2pt.shape[0]
T = c2pt.shape[1]
T_hlf = T//2

#? jack-knife resample
#! only used in plot , compare original data with fit
c2pt_sum = np.sum(c2pt,axis=0)
c2pt_jcknf = ((c2pt_sum) - c2pt)/(Ncnfg-1) # jack-knife resample
c2pt_cntrl = np.mean(c2pt_jcknf,axis=0) # jack-knife mean
c2pt_err = np.sqrt(Ncnfg-1)*np.std(c2pt_jcknf,axis=0) # jack-knife std

#? drop the first data point, normalized to c2[1] ~ 1
t_ary = np.array(range(1,T))
c2pt_jcknf = c2pt_jcknf/c2pt_cntrl[1]
```

- `` `var_val` ``, `` `dat_val` ``, `` `model` ``, `` `ini_val` ``: dictionary

- `` `var_val` ``, `` `dat_val` ``, `` `model` ``, `` `ini_val` ``: dictionary

- `var_val`, `dat_val`, `model`, `ini_val`: dictionary
- dictionary `{index_1: value_1, index_2: value_2, ... }`

- `var_val`, `dat_val`, `model`, `ini_val`: dictionary

key

- dictionary `{index_1: value_1, index_2: value_2, ... }`

- `var_val`, `dat_val`, `model`, `ini_val`: dictionary

key

- dictionary `{index_1: value_1, index_2: value_2, ... }`

value

- `var_val`, `dat_val`, `model`, `ini_val`: dictionary

key

- dictionary `{index_1: value_1, index_2: value_2, ... }`

value

- Example

```python
dct = {"a":1.0, "b":2.0}
dct["a"]
```

- Why wrap data in dictionaries?
  - Consider the case of
    - fit 2 datasets with 2 models
    - 2 models may have shared parameters

Combined fit

- Combined fit: e.g. 2 data sets, 2 models (with shared parameters)

label	label1	label2
data	$(\vec{x}^{(1)}, \vec{y}^{(1)})$	$(\vec{x}^{(2)}, \vec{y}^{(2)})$
model	$f(x, \{\vec{p}, \vec{q}\})$	$g(x, \{\vec{p}, \vec{r}\})$
ini_val	$\vec{p}_0, \vec{q}_0$	$\vec{p}_0, \vec{r}_0$

- $\vec{p}$ denotes the shared parameters

- Wraping data for combined fit

```
var_val = {'label1': [x1_1,x1_2,...], 'label2': [x2_1,x2_2,...

dat_val = {'label1': [y1_1,y1_2,...], 'label2': [y2_1,y2_2,...

prior = {'p_1': p_1,'p_2': p_2, ..., 'q_1': q_1, 'q_2': q_2, ... ,'r_1': r_1, 'r_2': r_2}
```

  - `y_1,2`: gvar class ⇨ `y_1 = gvar(y_1_cntrl, y_1_cov)` ...
  - `p_1,2`: gvar class ⇨ `p_1 = gvar(p_1_cntrl, p_1_err)` ...
  - Feed the `var_val`, `dat_val` and `prior` to `lsqfit.nonlinear_fit`

Wrapping model for combined fit

```
def fit_mdls(t_dctnry, p):
  mdls = {}
  mdls['label1'] = ...
  mdls['label2'] = ...
  return mdls
```

Multiple/Single source, Open/Periodic boundary condition

Multiple/Single source, Open/Periodic boundary condition

- Put multiple source in time direction to increase the statistics

Multiple/Single source, Open/Periodic boundary condition

- Put multiple source in time direction to increase the statistics

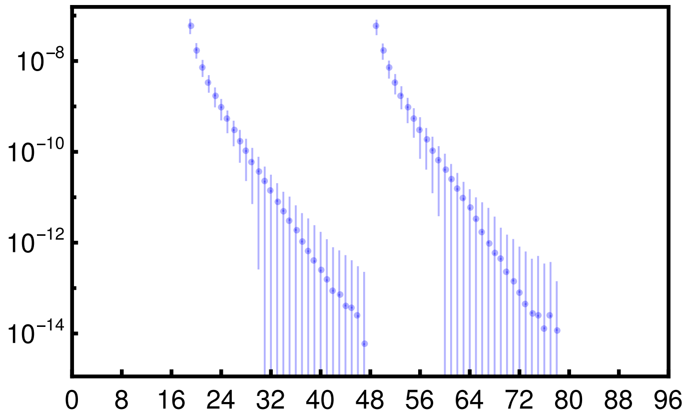
- Average over sources within each config (iog file)

Multiple/Single source, Open/Periodic boundary condition

- Put multiple source in time direction to increase the statistics

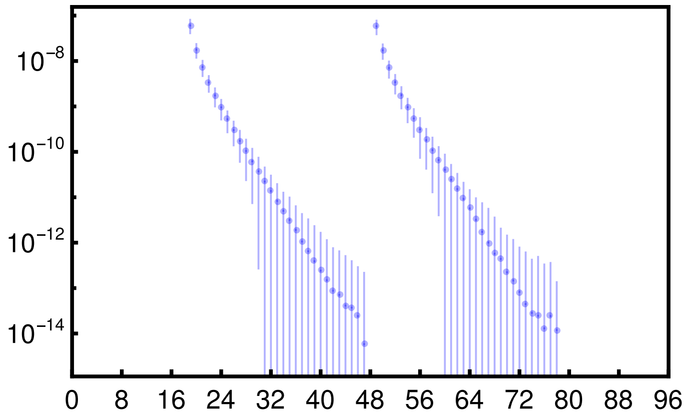
- Average over sources within each config (iog file)

- Example of open boundary condition, two sources



Multiple/Single source, Open/Periodic boundary condition

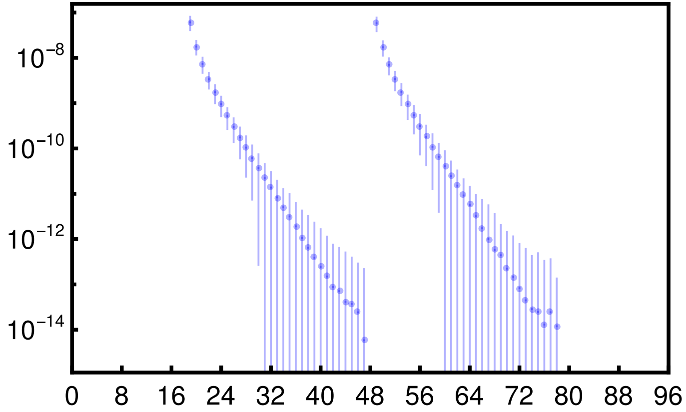
- Put multiple source in time direction to increase the statistics
- Average over sources within each config (iog file)
- Example of open boundary condition, two sources



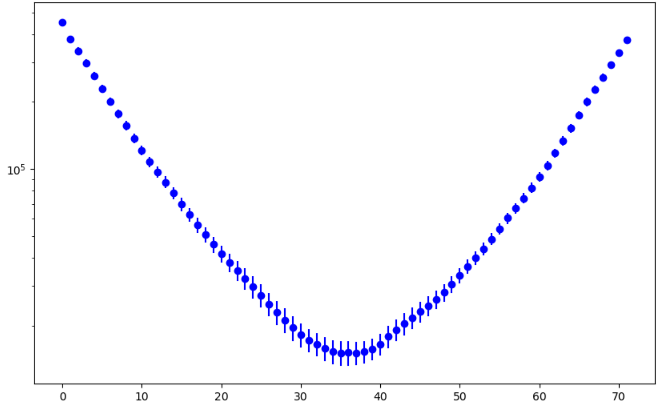
- Single source, periodic boundary condition
- Example
  - $C_(T+a) = C(0)$
  - $C(t) \approx C_0 e^{-E_0 \frac{T}{2}} \cosh\left[E_0\left(\frac{T}{2} - t\right)\right]$

Multiple/Single source, Open/Periodic boundary condition

- Put multiple source in time direction to increase the statistics
- Average over sources within each config (iog file)
- Example of open boundary condition, two sources



- Single source, periodic boundary condition
- Example
  - $C_(T+a) = C(0)$
  - $C(t) \approx C_0 e^{-E_0 \frac{T}{2}} \cosh\left[E_0\left(\frac{T}{2} - t\right)\right]$



Fit Results - A glance

```
Least Square Fit:
 chi2/dof [dof] = 0.96 [20]    Q = 0.5    logGBF = 116.43

Parameters:
            c0     0.252 (10)     [  0.0 (5.0) ]
            E0    0.1090 (27)     [  0.20 (50) ]
```

- Fences: two-point function data
- Bands: fitted model

> Error has been magnified with factor of 5 in figure

- Fited effective mass: $E_0 =$
  - The fitted E_0 is in lattice unit.
  - $E_0 t$ in $e^{-E_0 t}$ must be dimensionless ⇨ $E_0 = \dfrac{E_{0,\text{latt.}} \times 0.1973}{a}$ [GeV]

Output log

```
Least Square Fit:
  chi2/dof [dof] = 0.96 [20]    Q = 0.5    logGBF = 116.43

Parameters:
             c0     0.252 (10)     [  0.0 (5.0) ]
             E0    0.1090 (27)     [  0.20 (50) ]

Fit:
       key          y[key]       f(p)[key]
----------------------------------------
    c2pt 0       0.277 (12)     0.2712 (79)
         1       0.248 (11)     0.2435 (74)
         2       0.223 (11)     0.2186 (69)
         3       0.199 (10)     0.1964 (66)
         4      0.1790 (94)     0.1765 (62)
         5      0.1610 (87)     0.1586 (59)
         6      0.1446 (79)     0.1427 (56)
         7      0.1304 (75)     0.1284 (54)
         8      0.1180 (71)     0.1157 (51)
         9      0.1073 (68)     0.1044 (49)
        10      0.0981 (68)     0.0943 (47)
        11      0.0900 (69)     0.0853 (44)
        12      0.0825 (68)     0.0773 (42)
        13      0.0758 (66)     0.0703 (41)
```

Print this detailed log by setting `print(fit.format(True))`

Output log

```
Least Square Fit:
  chi2/dof [dof] = 0.96 [20]    Q = 0.5    logGBF = 116.43

Parameters:
             c0     0.252 (10)     [  0.0 (5.0) ]
             E0    0.1090 (27)     [  0.20 (50) ]

Fit:
      key          y[key]       f(p)[key]
----------------------------------------
    c2pt 0      0.277 (12)     0.2712 (79)
         1      0.248 (11)     0.2435 (74)
         2      0.223 (11)     0.2186 (69)
         3      0.199 (10)     0.1964 (66)
         4     0.1790 (94)     0.1765 (62)
         5     0.1610 (87)     0.1586 (59)
         6     0.1446 (79)     0.1427 (56)
         7     0.1304 (75)     0.1284 (54)
         8     0.1180 (71)     0.1157 (51)
         9     0.1073 (68)     0.1044 (49)
        10     0.0981 (68)     0.0943 (47)
        11     0.0900 (69)     0.0853 (44)
        12     0.0825 (68)     0.0773 (42)
        13     0.0758 (66)     0.0703 (41)
```

Print this detailed log by setting `print(fit.format(True))`

Output log

```
Least Square Fit:
  chi2/dof [dof] = 0.96 [20]    Q = 0.5    logGBF = 116.43

Parameters:
             c0    0.252 (10)     [  0.0 (5.0) ]
             E0   0.1090 (27)     [  0.20 (50) ]

Fit:
      key          y[key]      f(p)[key]
----------------------------------------
    c2pt 0      0.277 (12)    0.2712 (79)
         1      0.248 (11)    0.2435 (74)
         2      0.223 (11)    0.2186 (69)
         3      0.199 (10)    0.1964 (66)
         4     0.1790 (94)    0.1765 (62)
         5     0.1610 (87)    0.1586 (59)
         6     0.1446 (79)    0.1427 (56)
         7     0.1304 (75)    0.1284 (54)
         8     0.1180 (71)    0.1157 (51)
         9     0.1073 (68)    0.1044 (49)
        10     0.0981 (68)    0.0943 (47)
        11     0.0900 (69)    0.0853 (44)
        12     0.0825 (68)    0.0773 (42)
        13     0.0758 (66)    0.0703 (41)
```

Print this detailed log by setting `print(fit.format(True))`

Output log

```
Least Square Fit:
  chi2/dof [dof] = 0.96 [20]    Q = 0.5    logGBF = 116.43

Parameters:
             c0     0.252 (10)     [  0.0 (5.0) ]
             E0    0.1090 (27)     [  0.20 (50) ]

Fit:
       key          y[key]      f(p)[key]
----------------------------------------
    c2pt 0      0.277 (12)     0.2712 (79)
         1      0.248 (11)     0.2435 (74)
         2      0.223 (11)     0.2186 (69)
         3      0.199 (10)     0.1964 (66)
         4     0.1790 (94)     0.1765 (62)
         5     0.1610 (87)     0.1586 (59)
         6     0.1446 (79)     0.1427 (56)
         7     0.1304 (75)     0.1284 (54)
         8     0.1180 (71)     0.1157 (51)
         9     0.1073 (68)     0.1044 (49)
        10     0.0981 (68)     0.0943 (47)
        11     0.0900 (69)     0.0853 (44)
        12     0.0825 (68)     0.0773 (42)
        13     0.0758 (66)     0.0703 (41)
```

Print this detailed log by setting `print(fit.format(True))`

Output log

```
Least Square Fit:
  chi2/dof [dof] = 0.96 [20]    Q = 0.5    logGBF = 116.43

Parameters:
             c0     0.252 (10)     [  0.0 (5.0) ]
             E0    0.1090 (27)     [  0.20 (50) ]

Fit:
       key           y[key]      f(p)[key]
----------------------------------------
    c2pt 0      0.277 (12)     0.2712 (79)
         1      0.248 (11)     0.2435 (74)
         2      0.223 (11)     0.2186 (69)
         3      0.199 (10)     0.1964 (66)
         4     0.1790 (94)     0.1765 (62)
         5     0.1610 (87)     0.1586 (59)
         6     0.1446 (79)     0.1427 (56)
         7     0.1304 (75)     0.1284 (54)
         8     0.1180 (71)     0.1157 (51)
         9     0.1073 (68)     0.1044 (49)
        10     0.0981 (68)     0.0943 (47)
        11     0.0900 (69)     0.0853 (44)
        12     0.0825 (68)     0.0773 (42)
        13     0.0758 (66)     0.0703 (41)
```

Print this detailed log by setting `print(fit.format(True))`

Output log

```
Least Square Fit:
  chi2/dof [dof] = 0.96 [20]    Q = 0.5    logGBF = 116.43

Parameters:
             c0     0.252 (10)     [  0.0 (5.0) ]
             E0    0.1090 (27)     [  0.20 (50) ]

Fit:
       key          y[key]      f(p)[key]
---------------------------------------
    c2pt 0      0.277 (12)    0.2712 (79)
         1      0.248 (11)    0.2435 (74)
         2      0.223 (11)    0.2186 (69)
         3      0.199 (10)    0.1964 (66)
         4     0.1790 (94)    0.1765 (62)
         5     0.1610 (87)    0.1586 (59)
         6     0.1446 (79)    0.1427 (56)
         7     0.1304 (75)    0.1284 (54)
         8     0.1180 (71)    0.1157 (51)
         9     0.1073 (68)    0.1044 (49)
        10     0.0981 (68)    0.0943 (47)
        11     0.0900 (69)    0.0853 (44)
        12     0.0825 (68)    0.0773 (42)
        13     0.0758 (66)    0.0703 (41)
```

Print this detailed log by setting `print(fit.format(True))`

Access the fitted model function

```
fit = lsqfit.nonlinear_fit(data=(tsep_dctnry, c2pt_dctnry),
fcn=ft_mdls, prior=ini_prr, debug=True)

c2pt_fit_gvar = fit.fcn({'c2pt':t_lst}, fit.p)['c2pt']
```

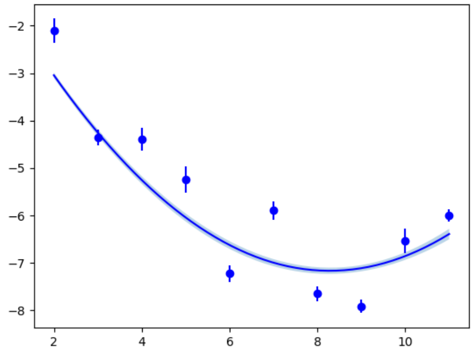
- `` `tsep_dctnry = {'c2pt':np.array(range(T_strt,T_end))}` ``
- `` `fit.p` `` ⇨ fitting parameter, dictionry
  - `` `fit.p['E0']` ``

What is a resonable fit?

Case 1

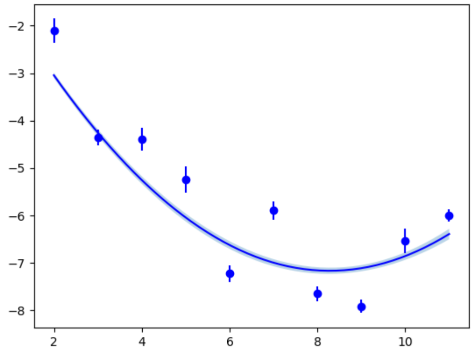
What is a resonable fit?

Case 1



What is a resonable fit?

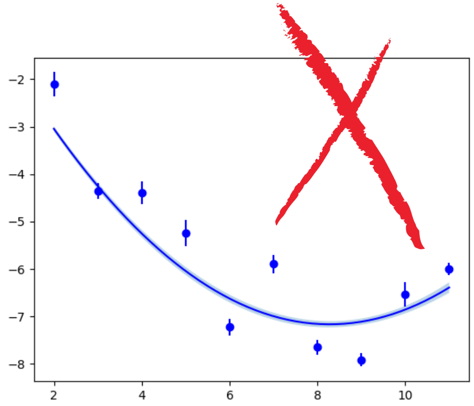
Case 1




```
Least Square Fit:
  chi2/dof [dof] = 13 [10]
  Q = 5.2e-24
  logGBF = -74.281
```

What is a resonable fit?

Case 1

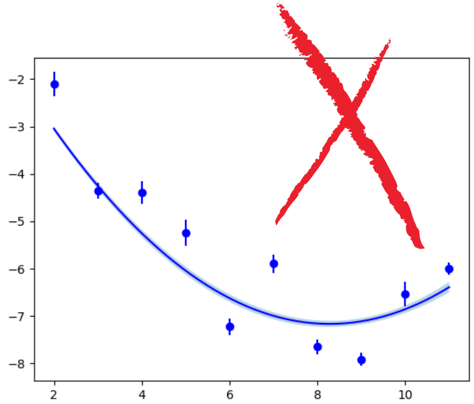



```
Least Square Fit:
  chi2/dof [dof] = 13 [10]
  Q = 5.2e-24
  logGBF = -74.281
```

- Underfit
  - Model failed to discribe data

What is a resonable fit?

Case 1



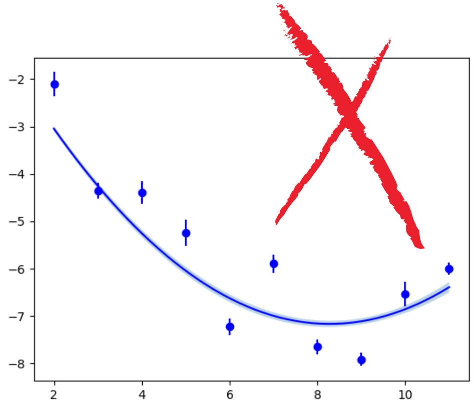

```
Least Square Fit:
  chi2/dof [dof] = 13 [10]
  Q = 5.2e-24
  logGBF = -74.281
```

- Underfit
  - Model failed to discribe data

Case 2

What is a resonable fit?

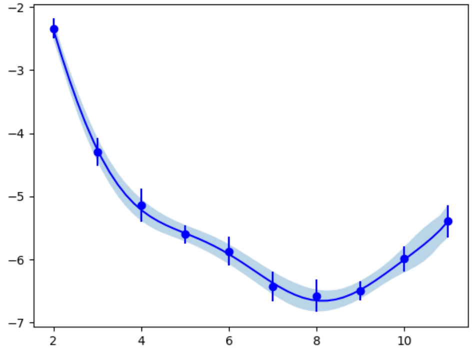
Case 1




```
Least Square Fit:
  chi2/dof [dof] = 13 [10]
  Q = 5.2e-24
  logGBF = -74.281
```

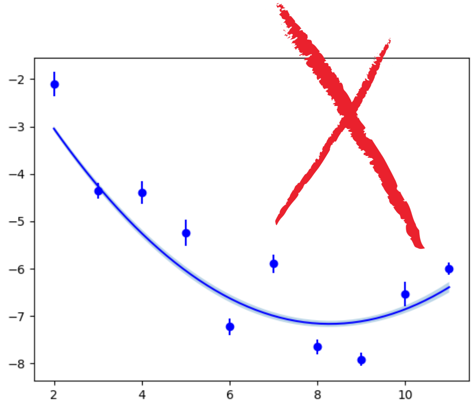
- Underfit
  - Model failed to discribe data

Case 2



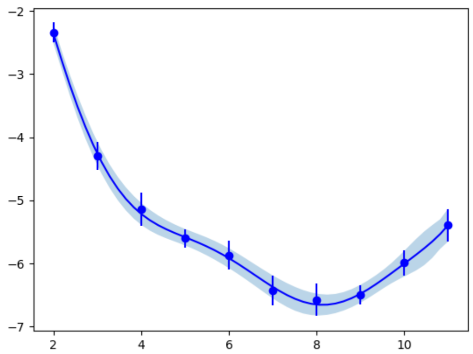
What is a resonable fit?

Case 1



```
Least Square Fit:
  chi2/dof [dof] = 13 [10]
  Q = 5.2e-24
  logGBF = -74.281
```

Case 2

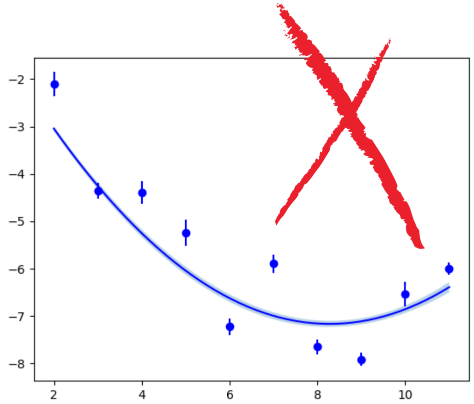


```
Least Square Fit:
  chi2/dof [dof] = 0.07 [10]
  Q = 1
  logGBF = -46.654
```

- Underfit
  - Model failed to discribe data

What is a resonable fit?

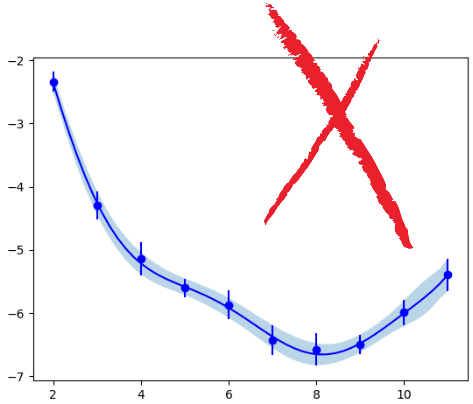
Case 1




```
Least Square Fit:
  chi2/dof [dof] = 13 [10]
  Q = 5.2e-24
  logGBF = -74.281
```

- Underfit
  - Model failed to discribe data

Case 2

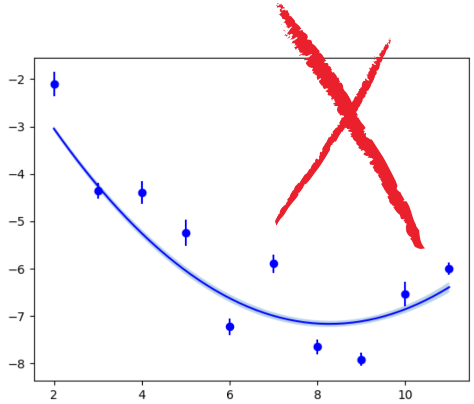



```
Least Square Fit:
  chi2/dof [dof] = 0.07 [10]
  Q = 1
  logGBF = -46.654
```

- Over fit
  - Try to extract too much info from data

What is a resonable fit?

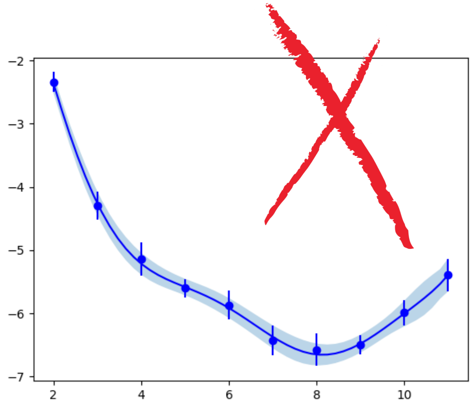
Case 1




```
Least Square Fit:
  chi2/dof [dof] = 13 [10]
  Q = 5.2e-24
  logGBF = -74.281
```

- Underfit
  - Model failed to discribe data

Case 2



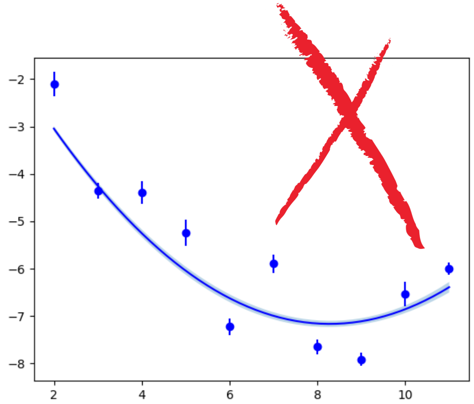

```
Least Square Fit:
  chi2/dof [dof] = 0.07 [10]
  Q = 1
  logGBF = -46.654
```

- Over fit
  - Try to extract too much info from data

Case 3

What is a resonable fit?

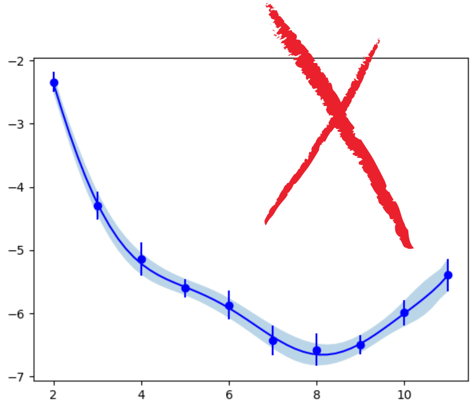
Case 1




```
Least Square Fit:
  chi2/dof [dof] = 13 [10]
  Q = 5.2e-24
  logGBF = -74.281
```

- Underfit
  - Model failed to discribe data

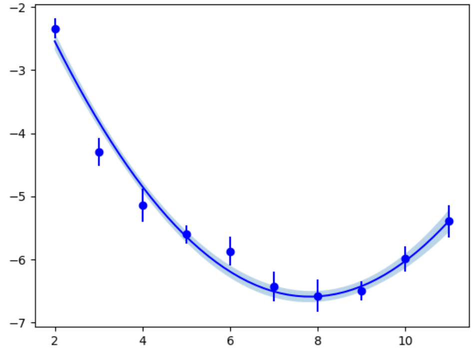
Case 2




```
Least Square Fit:
  chi2/dof [dof] = 0.07 [10]
  Q = 1
  logGBF = -46.654
```

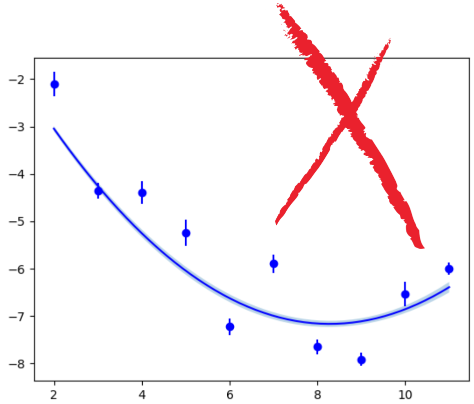
- Over fit
  - Try to extract too much info from data

Case 3



What is a resonable fit?

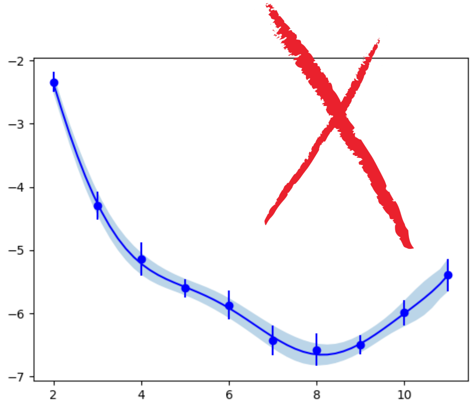
Case 1




```
Least Square Fit:
  chi2/dof [dof] = 13 [10]
  Q = 5.2e-24
  logGBF = -74.281
```

- Underfit
  - Model failed to discribe data

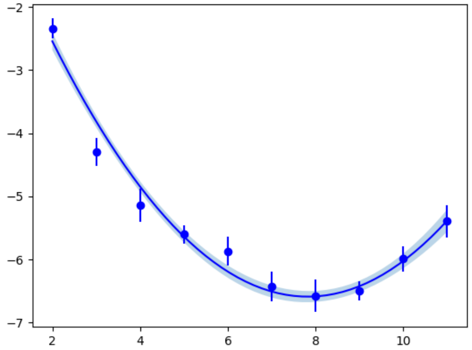
Case 2




```
Least Square Fit:
  chi2/dof [dof] = 0.07 [10]
  Q = 1
  logGBF = -46.654
```

- Over fit
  - Try to extract too much info from data

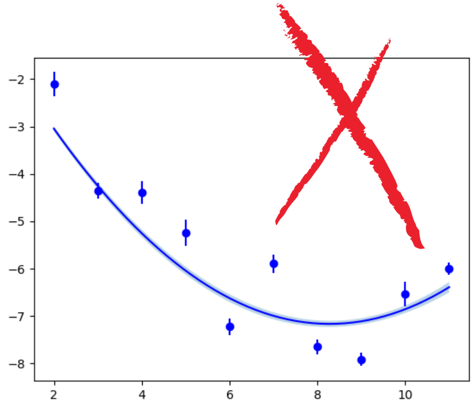
Case 3




```
Least Square Fit:
  chi2/dof [dof] = 1 [10]
  Q = 0.41
  logGBF = -15.05
```

What is a resonable fit?

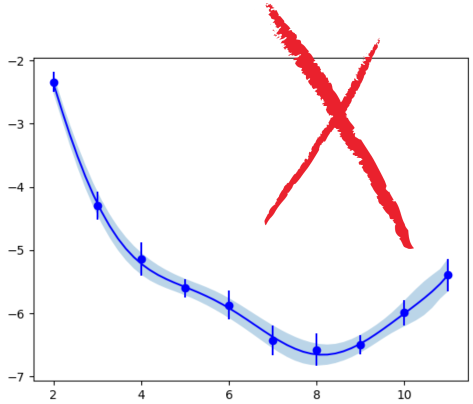
Case 1




```
Least Square Fit:
  chi2/dof [dof] = 13 [10]
  Q = 5.2e-24
  logGBF = -74.281
```

- Underfit
  - Model failed to discribe data

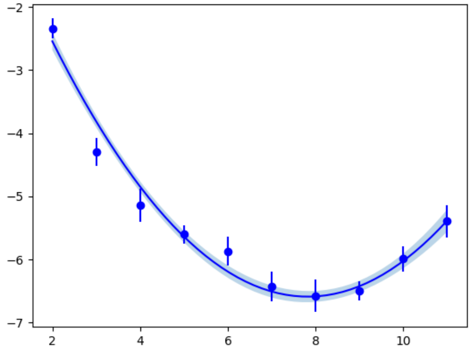
Case 2




```
Least Square Fit:
  chi2/dof [dof] = 0.07 [10]
  Q = 1
  logGBF = -46.654
```

- Over fit
  - Try to extract too much info from data

Case 3




```
Least Square Fit:
  chi2/dof [dof] = 1 [10]
  Q = 0.41
  logGBF = -15.05
```

- ***Resonable fit***

How good is a fit?

Interpretation of `lsqfit` log

```
Least Square Fit:
  chi2/dof [dof] = 1 [10]    Q = 0.41    logGBF = -15.05
```

How good is a fit?

Interpretation of `lsqfit` log

```
Least Square Fit:
  chi2/dof [dof] = 1 [10]    Q = 0.41    logGBF = -15.05
```

- `chi2/dof`: better around 1

How good is a fit?

Interpretation of `lsqfit` log

```
Least Square Fit:
  chi2/dof [dof] = 1 [10]    Q = 0.41    logGBF = -15.05
```

- `chi2/dof`: better around 1
- `Q`: The probability that the χ^2 from the fit could have been larger, by chance, assuming the best-fit model is correct.
  - Good fits have Q values larger than 0.1 or so

How good is a fit?

Interpretation of `lsqfit` log

```
Least Square Fit:
  chi2/dof [dof] = 1 [10]    Q = 0.41    logGBF = -15.05
```

- `chi2/dof`: better around 1
- `Q`: The probability that the χ^2 from the fit could have been larger, by chance, assuming the best-fit model is correct.
  - Good fits have Q values larger than 0.1 or so
- `logGBF`: $\ln P(\text{data}|\text{fit})$ ⇨ Probability of obtaining the data by randomly sampling the fitted model
  - Useful for comparing fits of the same data to different models, with different priors and/or fit functions.
  - The model with the largest value of `logGBF` is the one preferred by the data.
  - Differences in fit.logGBF smaller than 1 are not very significant.

Coding Tips

Coding Tips

- Vectorize your code, instead of looping over

Coding Tips

- Vectorize your code, instead of looping over
- Modulize
  - e.g. in 2-point analysis demo
    - module 1: extract required data (hadron id, momentum id, smearing...) from iog/h5, export to data file
    - module 2: load data from 1, Jack-knife / bootstrap analysis, export results in data file
    - module 3: load data from 2, perform lsqfit, outupt

Coding Tips

- Vectorize your code, instead of looping over
- Modulize
  - e.g. in 2-point analysis demo
    - module 1: extract required data (hadron id, momentum id, smearing...) from iog/h5, export to data file
    - module 2: load data from 1, Jack-knife / bootstrap analysis, export results in data file
    - module 3: load data from 2, perform lsqfit, outupt
- Naming convention and comments
  - e.g. `c2pt_cntrl`, `c2pt_cov`, `t_ary` v.s. `a`, `b`, `t`

Coding Tips

- Vectorize your code, instead of looping over
- Modulize
  - e.g. in 2-point analysis demo
    - module 1: extract required data (hadron id, momentum id, smearing...) from iog/h5, export to data file
    - module 2: load data from 1, Jack-knife / bootstrap analysis, export results in data file
    - module 3: load data from 2, perform lsqfit, outupt
- Naming convention and comments
  - e.g. `c2pt_cntrl`, `c2pt_cov`, `t_ary` v.s. `a`, `b`, `t`
- Version control (git, SVN)

Exercises

Exercises

1. **Read the pion two-point function data**

Exercises

1. Read the pion two-point function data

- Create a directory named by your name under the `home` directory
- copy `/dssg/home/acct-phyww/phyww/qazhang/training_camp_2023/class1_xiong/Ex1_Latt6th.tar.gz/` to the created directory and unzip it

Exercises

1. Read the pion two-point function data

- Create a directory named by your name under the `home` directory
- copy `/dssg/home/acct-phyww/phyww/qazhang/training_camp_2023/class1_xiong/Ex1_Latt6th.tar.gz/` to the created directory and unzip it

2. Jack-knife analysis of the two-point function

Exercises

1. Read the pion two-point function data

- Create a directory named by your name under the `home` directory
- copy `/dssg/home/acct-phyww/phyww/qazhang/training_camp_2023/class1_xiong/Ex1_Latt6th.tar.gz/` to the created directory and unzip it

2. Jack-knife analysis of the two-point function

3. Using `lsqfit` to fit the pion's 2pt data and determine the gournd state energy

Exercises

1. Read the pion two-point function data

- Create a directory named by your name under the `home` directory
- copy `/dssg/home/acct-phyww/phyww/qazhang/training_camp_2023/class1_xiong/Ex1_Latt6th.tar.gz/` to the created directory and unzip it

2. Jack-knife analysis of the two-point function

3. Using `lsqfit` to fit the pion's 2pt data and determine the gournd state energy

4. Comparing your fit and data

Exercises

1. Read the pion two-point function data

- Create a directory named by your name under the `home` directory
- copy `/dssg/home/acct-phyww/phyww/qazhang/training_camp_2023/class1_xiong/Ex1_Latt6th.tar.gz/` to the created directory and unzip it

2. Jack-knife analysis of the two-point function

3. Using `lsqfit` to fit the pion's 2pt data and determine the gournd state energy

4. Comparing your fit and data

> For this exercise: lattice spacing: 0.108 fm, $m_\pi \approx 200\mathrm{MeV}$

Hint:

Hint:

- `iog_reader`, `pandas.DataFrame.loc`, `pandas.DataFrame.to_numpy`, `numpy.mean`, `numpy.cov`, `numpy.std`, `lsqfit.nonlinear_fit`
- `Data` containing the 50 two-point function iog data files, from 50 configuration
- 72 lattice spacings in time direction
- `iog_reader` is provided in `iog_read_demo.py`
- `x=gv.gvar('2(0.5)')` : `x.mean` ⇨ `2.00`, `x.sdev` ⇨ `0.50`
- Construct gvar from tuple `x=gv.gvar((2.0.5))`

Notes

- `` `Python` `` indices starts from 0, *Mathmatica*, Julia starts from 1
- `` `numpy.ndarray` `` slice, range doe not include the last index e.g.
  - `` `a[2:5]` `` ⇨ [a_2,a_3,a_4]
  - `` `range(2,5)` `` ⇨ [2,3,4]
- `` `iog_read.py` `` returns `` `pandas.Dataframe` ``, columns in this data frame are strings (not integers)

III. Linux Basics

- If windows
  - try WSL

    > https://docs.microsoft.com/zh-cn/windows/wsl/install

  - Powershell

Search for help and tips

III. Linux Basics

- If windows
  - try WSL
    > https://docs.microsoft.com/zh-cn/windows/wsl/install
  - Powershell

Search for help and tips

- Usually `` `command --help` `` and `` `man command` ``
  - `` `man` `` for "mannual"
  - e.g. `` `man --help` ``, `` `man man` ``

III. Linux Basics

- If windows
  - try WSL

    > https://docs.microsoft.com/zh-cn/windows/wsl/install

  - Powershell

Search for help and tips

- Usually `` `command --help` `` and `` `man command` ``
  - `` `man` `` for "mannual"
  - e.g. `` `man --help` ``, `` `man man` ``
- ***Read the error message carefully !***

III. Linux Basics

- If windows
    - try WSL

      > https://docs.microsoft.com/zh-cn/windows/wsl/install

    - Powershell

Search for help and tips

- Usually `command --help` and `man command`
    - `man` for "mannual"
    - e.g. `man --help`, `man man`
- ***Read the error message carefully !***
- Online forum
    - https://unix.stackexchange.com/
    - https://stackoverflow.com/
    - Search online ...

Path

Path

- Absolute path: starts with `/`
    - print by `pwd` (*print work directory*)
    - e.g. `/home/chimaoshu/Nutstore_Files`

Path

- Absolute path: starts with `` `/` ``
  - print by `` `pwd` `` (*print work directory*)
  - e.g. `` `/home/chimaoshu/Nutstore_Files` ``
- Relative path: relative to `` `pwd` ``

  e.g. `` `./demo/c2pt_fit.png` ``, `` `../LPC_trnsvrsty` ``

Path

- Absolute path: starts with `/`
    - print by `pwd` (*print work directory*)
    - e.g. `/home/chimaoshu/Nutstore_Files`
- Relative path: relative to `pwd`

  e.g. `./demo/c2pt_fit.png`, `../LPC_trnsvrsty`
- Abbreviation of some paths
    - home directory `~` ⇨ `/home/USR`
    - current directory `.`
    - Parent directory `..`, parent of parent `../..`

Path

- Absolute path: starts with `/`
  - print by `pwd` (*print work directory*)
  - e.g. `/home/chimaoshu/Nutstore_Files`
- Relative path: relative to `pwd`

  e.g. `./demo/c2pt_fit.png`, `../LPC_trnsvrsty`
- Abbreviation of some paths
  - home directory `~` ⇨ `/home/USR`
  - current directory `.`
  - Parent directory `..`, parent of parent `../..`
- Change directory `cd path`

Path

- Absolute path: starts with `/`
  - print by `pwd` (*print work directory*)
  - e.g. `/home/chimaoshu/Nutstore_Files`
- Relative path: relative to `pwd`

  e.g. `./demo/c2pt_fit.png`, `../LPC_trnsvrsty`
- Abbreviation of some paths
  - home directory `~` ⇨ `/home/USR`
  - current directory `.`
  - Parent directory `..`, parent of parent `../..`
- Change directory `cd path`
- List files and directories under a path `ls path`

Path

- Absolute path: starts with `/`
    - print by `pwd` (*print work directory*)
    - e.g. `/home/chimaoshu/Nutstore_Files`
- Relative path: relative to `pwd`

  e.g. `./demo/c2pt_fit.png`, `../LPC_trnsvrsty`
- Abbreviation of some paths
    - home directory `~` ⇨ `/home/USR`
    - current directory `.`
    - Parent directory `..`, parent of parent `../..`
- Change directory `cd path`
- List files and directories under a path `ls path`
- Autocomplete/suggestions
    - Press `Tab`

Create/Remove/Move/Copy file, directory

Create/Remove/Move/Copy file, directory

- create a file
    - `touch file_name` ⇨ create under PWD
    - `touch path/file_name` ⇨ create under `path`

Create/Remove/Move/Copy file, directory

- create a file
  - `touch file_name` ⇨ create under PWD
  - `touch path/file_name` ⇨ create under `path`
- create a directory
  - `mkdir dir_name` ⇨ create under PWD
  - `mkdir path/dir_name` ⇨ create under `path`

Create/Remove/Move/Copy file, directory

- create a file
  - `touch file_name` ⇨ create under PWD
  - `touch path/file_name` ⇨ create under `path`
- create a directory
  - `mkdir dir_name` ⇨ create under PWD
  - `mkdir path/dir_name` ⇨ create under `path`
- remove (delete) a file
  - `rm file_name`, `rm path/file_name`

Create/Remove/Move/Copy file, directory

- create a file
  - `` `touch file_name` `` ⇨ create under PWD
  - `` `touch path/file_name` `` ⇨ create under `` `path` ``
- create a directory
  - `` `mkdir dir_name` `` ⇨ create under PWD
  - `` `mkdir path/dir_name` `` ⇨ create under `` `path` ``
- remove (delete) a file
  - `` `rm file_name` ``, `` `rm path/file_name` ``
- remove a directory (and everything inside)
  - `` `rm -r dir_name` `` `` `rm -r path/dir_name` ``
  - **CAUTION when using** `` `rm -rf` ``

Create/Remove/Move/Copy file, directory

- create a file
  - `touch file_name` ⇨ create under PWD
  - `touch path/file_name` ⇨ create under `path`
- create a directory
  - `mkdir dir_name` ⇨ create under PWD
  - `mkdir path/dir_name` ⇨ create under `path`
- remove (delete) a file
  - `rm file_name`, `rm path/file_name`
- remove a directory (and everything inside)
  - `rm -r dir_name` `rm -r path/dir_name`
  - **CAUTION when using** `rm -rf`
- Similar for copy `cp` and move(rename) `mv`

cat, grep, >

Compress and uncompress

cat, grep, >

- `` `cat` ``: show contents of a file
    - e.g. `` `cat ./demo/lsqfit_2pt_iog.py` ``

Compress and uncompress

cat, grep, >

- `cat` : show contents of a file
  - e.g. `cat ./demo/lsqfit_2pt_iog.py`
- `grep` : catch specified contents of a file
  - e.g. `cat ./demo/lsqfit_2pt_iog.py | grep gvar`

> '|' : pipe

Compress and uncompress

cat, grep, >

- `` `cat` ``: show contents of a file
    - e.g. `` `cat ./demo/lsqfit_2pt_iog.py` ``
- `` `grep` ``: catch specified contents of a file
    - e.g. `` `cat ./demo/lsqfit_2pt_iog.py | grep gvar` ``

> '|' : pipe

- `` `>` ``: redirect output stream
    - e.g. `` `./lsqfit_2pt.py > lsq.log 2>&1` ``

> `` `2>&1` `` append error message

Compress and uncompress

cat, grep, >

- `cat`: show contents of a file
  - e.g. `cat ./demo/lsqfit_2pt_iog.py`
- `grep`: catch specified contents of a file
  - e.g. `cat ./demo/lsqfit_2pt_iog.py | grep gvar`

> '|' : pipe

- `>`: redirect output stream
  - e.g. `./lsqfit_2pt.py > lsq.log 2>&1`

> `2>&1` append error message

Compress and uncompress

- compress `tar`
  - try `tar --help`

SSH login

SSH login

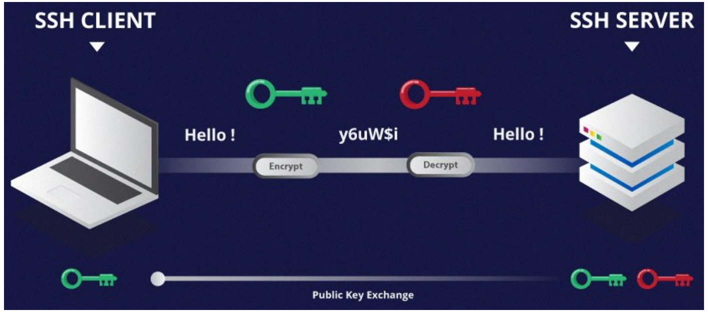
- On Windows you can use powerhsell’s built-in SSH client

SSH login

- On Windows you can use powerhsell's built-in SSH client
- `` `ssh usr@host` ``
    - e.g. `` `ssh stu1947@sylogin.hpc.sjtu.edu.cn` ``
    - `` `usr` ``: username, `` `host` ``: remote machine

SSH login

- On Windows you can use powerhsell's built-in SSH client
- `ssh usr@host`
  - e.g. `ssh stu1947@sylogin.hpc.sjtu.edu.cn`
  - `usr`: username, `host`: remote machine
- Use `key` instead of password
  - `ssh-keygen`: generate RSA key (public and private)
  - distribute pubkey `id_rsa.pub` to others
  - keep your private key `id_rsa` private
  - `ssh-copyid usr@host`: install your pubkey to host



SSH login

- SSH config file
  - `~/.ssh/config`

```
Host sjtu
 HostName sylogin.hpc.sjtu.edu.cn
 User stu1947
 IdentityFile /home/chimaoshu/.ssh/id_rsa
 AddKeysToAgent yes
```

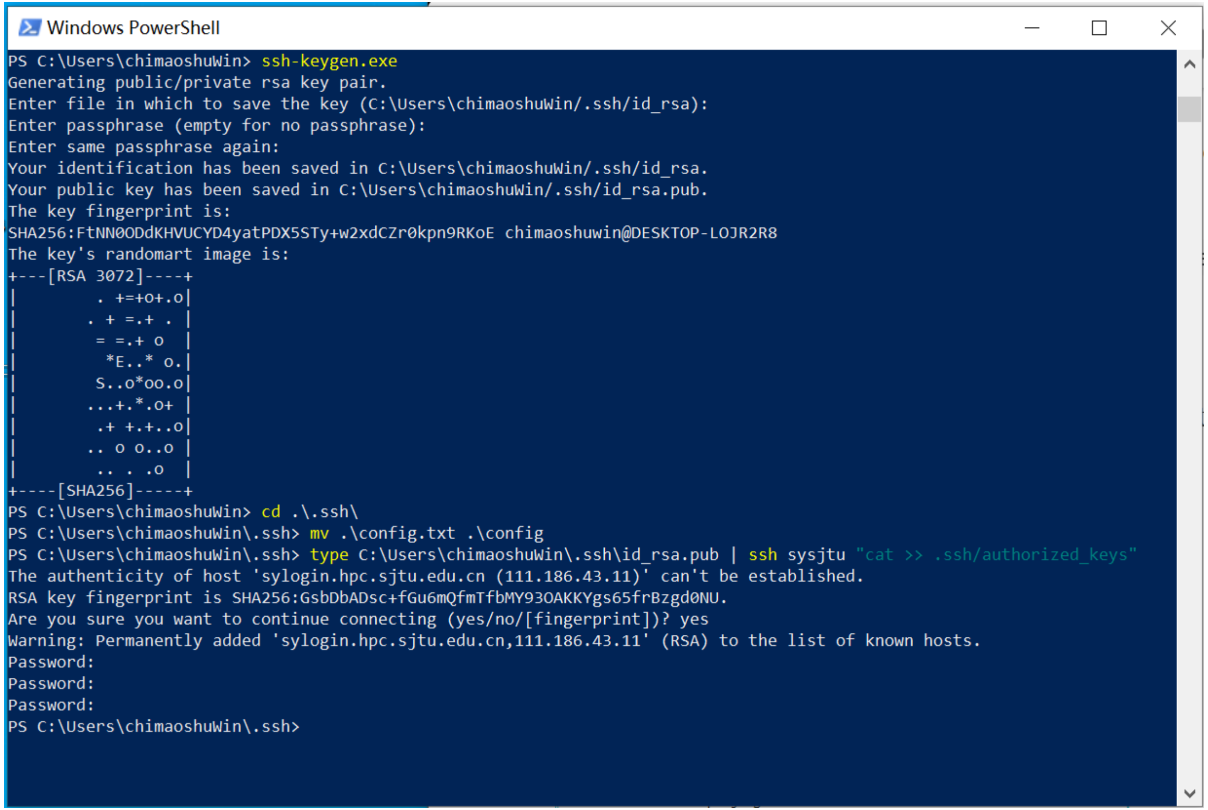
- After setting ssh config file ⇨ `ssh sjtu`

SSH via Windows powershell (But recommend using vscode)

- `` `powershell` `` has built-in `` `ssh.exe` ``, `` `ssh-keygen.exe` ``, `` `scp.exe` ``

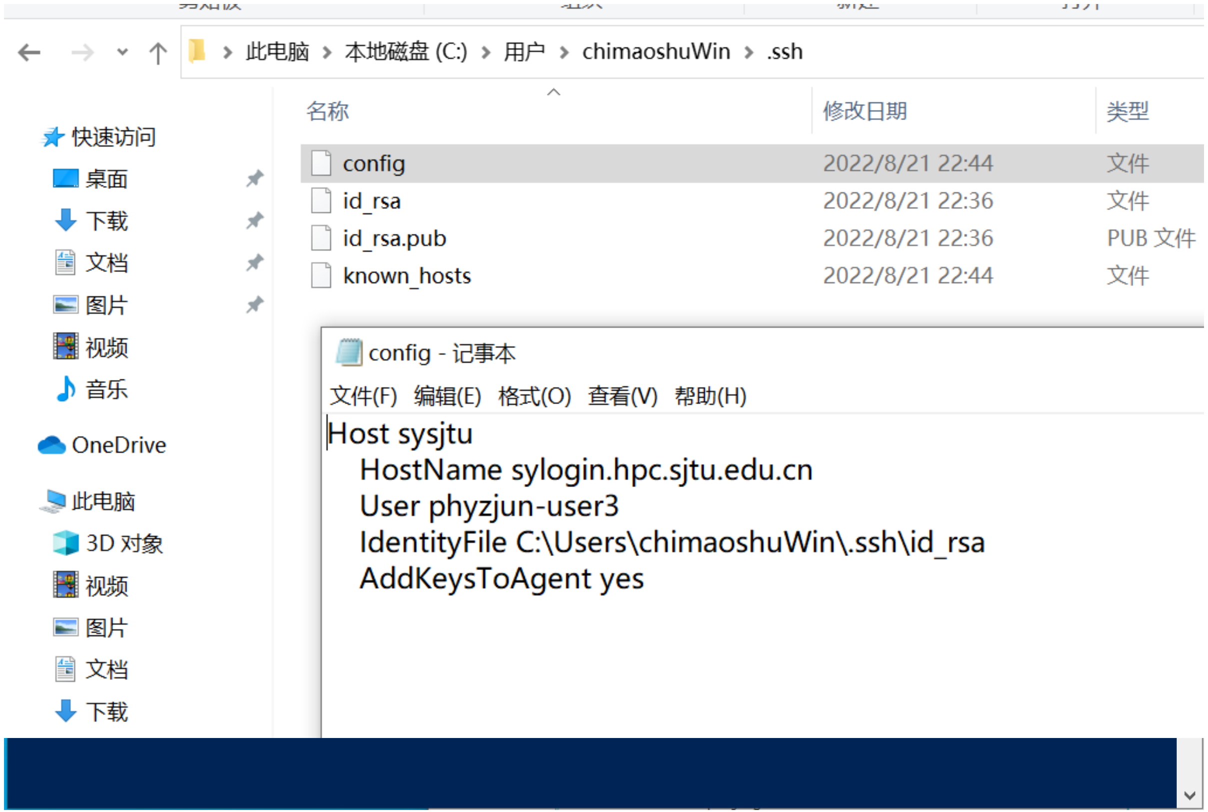
SSH via Windows powershell (But recommend using vscode)

- `powershell` has built-in `ssh.exe`, `ssh-keygen.exe`, `scp.exe`



SSH via Windows powershell (But recommend using vscode)

- `powershell` has built-in `ssh.exe`, `ssh-keygen.exe`, `scp.exe`



SSH via Windows powershell (But recommend using vscode)

- `powershell` has built-in `ssh.exe`, `ssh-keygen.exe`, `scp.exe`

SSH via Windows powershell (But recommend using vscode)

- `powershell` has built-in `ssh.exe`, `ssh-keygen.exe`, `scp.exe`

OpenSSH SSH client

```
PS C:\Users\chimaoshuWin> ssh sysjtu
Password:
Last login: Sun Aug 21 23:12:52 2022 from 42.48.113.177
slurm常用指令：
    sinfo 查看队列状态和信息
    sacct 显示用户作业历史
    squeue 显示当前作业状态
    sbatch 提交作业
    scancel 取消指定作业

集群设置以下队列，使用限制与说明如下：
            允许单作业CPU核数为1~60000，每核配比8G内存；单节点配置为64核，512G内存
         允许单作业GPU卡数为1~92，每卡配比CPU上限为16，每核配比8G内存；单节点配置为64核，512G内存，4块40G显存的A100 GPU
卡
                 仅用于短时间测试，请勿批量投递作业进行完整计算。作业最多申请2节点，运行60分钟，每核配比8G内存；单节点配
置为64核，512G内存
              仅用于短时间测试，请勿批量投递作业进行完整计算。作业最多申请1节点，运行20分钟；单节点配置64核，512G内存，2
8块5G显存的虚拟GPU卡

用户帮助文档：https://docs.hpc.sjtu.edu.cn/

登陆节点禁止运行作业和并行编译，如需交互操作，请申请计算资源： $ srun -p 64c512g -n 4 --pty /bin/bash

登录节点不适合进行大批量数据传输，请通过传输节点data.hpc.sjtu.edu.cn进行数据拷贝，参考 https://docs.hpc.sjtu.edu.cn/tran
sport

邮件支持：hpc@sjtu.edu.cn
acct-phyzjun账户存储使用量为：2.5T
phyzjun-user3用户存储使用量为：2.0G

[phyzjun-user3@sylogin1 ~]$
```

SSH via Windows powershell (But recommend using vscode)

- `powershell` has built-in `ssh.exe`, `ssh-keygen.exe`, `scp.exe`

- `ssh-copy-id` equivlent: type path_to_id_rsa.pub | ssh sjtu "cat >> ~/.ssh/authorized_keys"

Copy between local and remote machines

- `scp source target`, (`scp` = `s`(ssh) + `cp`) e.g.
  - `scp ./demo/lsqfit_2pt.py sysjtu:~/demo`
  - `scp -r sjtu:~/demo /home/chimaoshu`

> `sjtu` could be replaced by `usr@host` if `.ssh/config` is not set

Modify files on remote machine

Modify files on remote machine

- vi, vim

  🤣 Q: How do you generate a random string?
  A: Put a Windows user in front of vi, and tell them to exit

Modify files on remote machine

- vi, vim
  🤣 Q: How do you generate a random string?
  A: Put a Windows user in front of vi, and tell them to exit

- Download to local ⇨ edit locally ⇨ upload to remote ⇨ compile/excute ⇨ download to local, …

Modify files on remote machine

- vi, vim
  🤣 Q: How do you generate a random string?
  A: Put a Windows user in front of vi, and tell them to exit
  

- Download to local ⇨ edit locally ⇨ upload to remote ⇨ compile/excute ⇨ download to local, ...

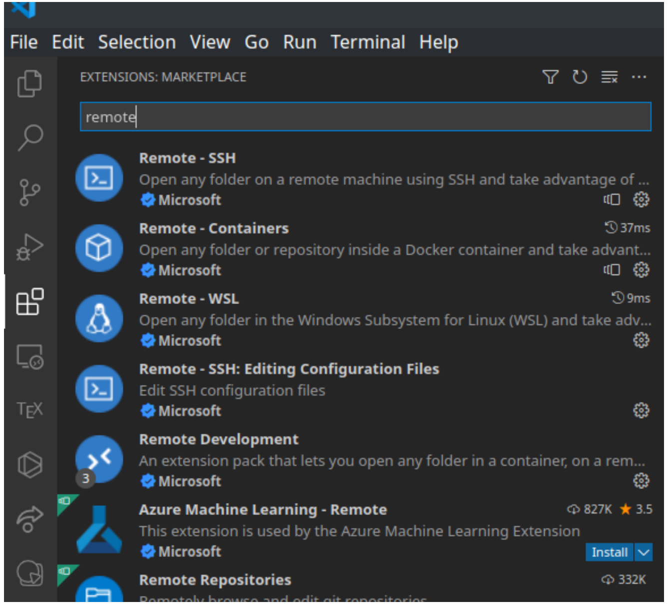
- Recommanded: vscode + remote development extension

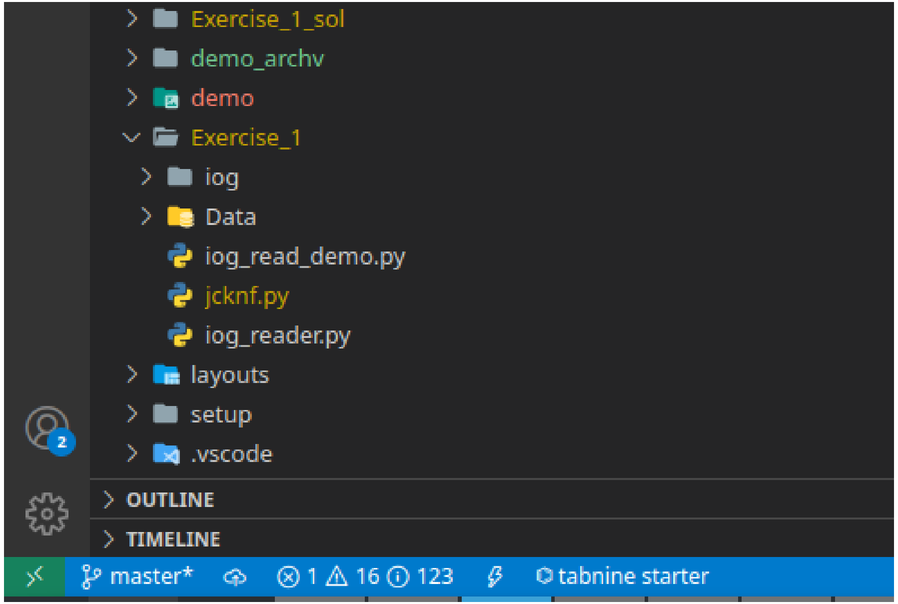
> https://code.visualstudio.com/docs/remote/ssh-tutorial

> better set `.ssh/config`

vscode

- https://code.visualstudio.com/Download
- Extensions
  - Remote Development





vscode

- https://code.visualstudio.com/Download
- Extensions
  - Python by Microsoft
  - Pylint by Microsoft
  - C/C++ by Microsoft
  - Kite
  - …

Enviormental Modules

- \``module avail`\`: list avaliable modules
- \``module load mod`\`: load mod
- \``module list`\`: list currenly loaded modules
- \``module purge`\`: unload all modules
- …

Setup Python Enviroment

Setup Python Enviroment

- Use `export` to set the relavent enviroment variables (`PATH` and `PYTHONPATH`)
  - `export PATH=$PATH:/dssg/home/acct-phyww/phyww/qazhang/packages/anaconda3/bin`
  - `export PYTHONPATH=/dssg/home/acct-phyww/phyww/.local/lib/python3.7/site-packages`
  - Use `python`, do not use `python3`, `python3.x`

Setup Python Enviroment

- Use `` `export` `` to set the relavent enviroment variables (`` `PATH` `` and `` `PYTHONPATH` ``)
    - `` `export PATH=$PATH:/dssg/home/acct-phyww/phyww/qazhang/packages/anaconda3/bin` ``
    - `` `export PYTHONPATH=/dssg/home/acct-phyww/phyww/.local/lib/python3.7/site-packages` ``
    - Use `` `python` ``, do not use `` `python3` ``, `` `python3.x` ``
- Appdend them to `` `~/.bashrc` `` only if you know what you are doing

Setup Python Enviroment

- Use `export` to set the relavent enviroment variables (`PATH` and `PYTHONPATH`)
  - `export PATH=$PATH:/dssg/home/acct-phyww/phyww/qazhang/packages/anaconda3/bin`
  - `export PYTHONPATH=/dssg/home/acct-phyww/phyww/.local/lib/python3.7/site-packages`
  - Use `python`, do not use `python3`, `python3.x`
- Appdend them to `~/.bashrc` only if you know what you are doing
- Or excute them everytime after login/reconnect...

A glance on HDF5

A glance on HDF5

- ``h5glance``: installed via ``pip``: ``pip3 install h5glance``

A glance on HDF5

- `h5glance`: installed via `pip`: `pip3 install h5glance`
- Basic usage:
    - `h5glance h5_file [path]`
    - `path`: unix like path to data (string)

A glance on HDF5

- `` `h5glance` ``: installed via `` `pip` ``: `` `pip3 install h5glance` ``
- Basic usage:
  - `` `h5glance h5_file [path]` ``
  - `` `path` ``: unix like path to data (string)
- Advanced usage:
  - show attributes : `` `h5glance --attrs h5_file [path]` ``
  - peek data: `` `h5glance --attrs h5_file path` ``
  - slice data: `` `h5glance -s slice --attrs h5_file path` ``
    - `` `slice` ``: `` `numpy` `` like slice, e.g: `` `2,10:15` ``

IV. Introduction to HDF5

IV. Introduction to HDF5

- HDF5: Hierarchical Data Format Version 5

> **High-performance data management and storage suite** Utilize the HDF5 high performance data software library and file format to manage, process, and store your heterogeneous data. HDF5 is built for fast I/O processing and storage.

  - Sell sheet : https://www.hdfgroup.org/wp-content/uploads/2017/12/HDF512-17.pdf

IV. Introduction to HDF5

- HDF5: Hierarchical Data Format Version 5

> **High-performance data management and storage suite** Utilize the HDF5 high performance data software library and file format to manage, process, and store your heterogeneous data. HDF5 is built for fast I/O processing and storage.

  - Sell sheet : https://www.hdfgroup.org/wp-content/uploads/2017/12/HDF512-17.pdf

- Open source library

IV. Introduction to HDF5

- HDF5: Hierarchical Data Format Version 5

> **High-performance data management and storage suite** Utilize the HDF5 high performance data software library and file format to manage, process, and store your heterogeneous data. HDF5 is built for fast I/O processing and storage.

  - Sell sheet : https://www.hdfgroup.org/wp-content/uploads/2017/12/HDF512-17.pdf

- Open source library

- Supported by C/C++, Julia, Mathmetaica, Python ...

  - HDF5 does not provide native complex type, different software may have different interpretation of complex type

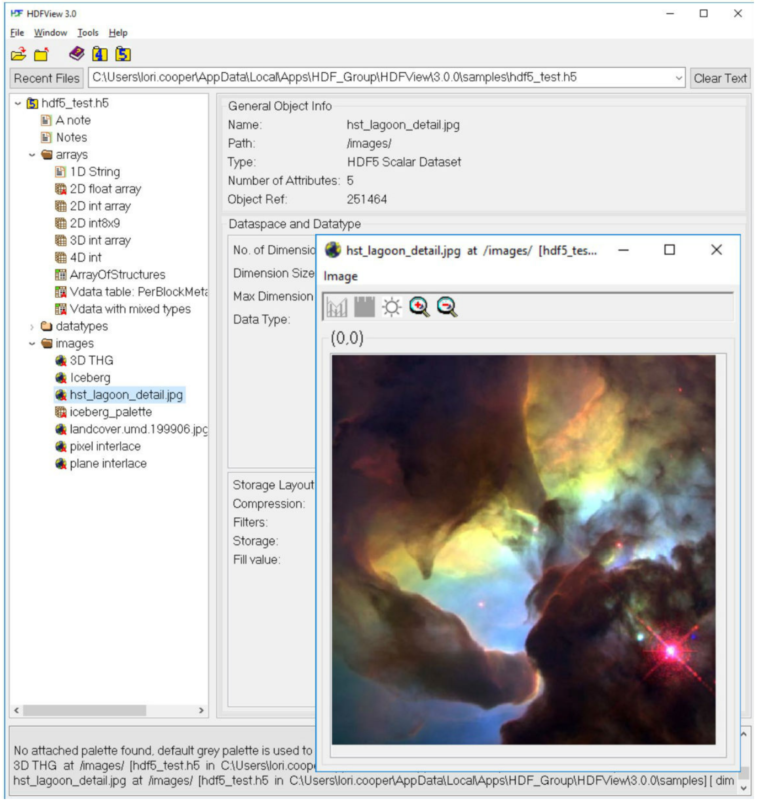
    - e.g. Interpreted as association (dictironary) in mathmatica

```
 In[1]:= c = 1 + 0.2*I;
 In[2]:= Export["cmplx.h5", "/cmplx" -> c];
 In[3]:= Import["cmplx.h5", {"Datasets", "/cmplx"}]
Out[4] = <|"Re" -> 1., "Im" -> 0.2|>
```

Other Features

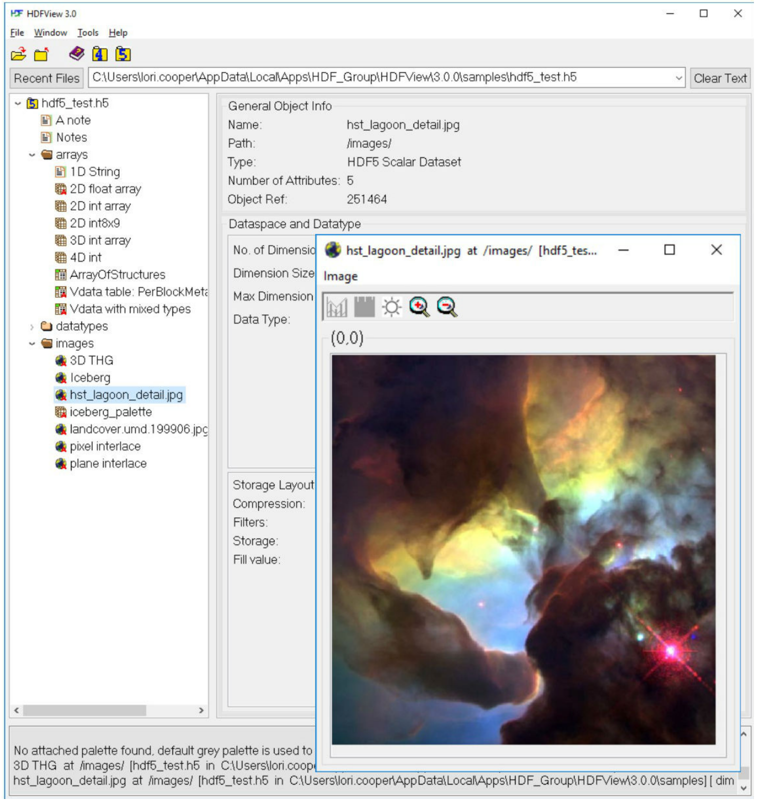
Other Features

- Float, integer, string, array, image … all in a single h5 file



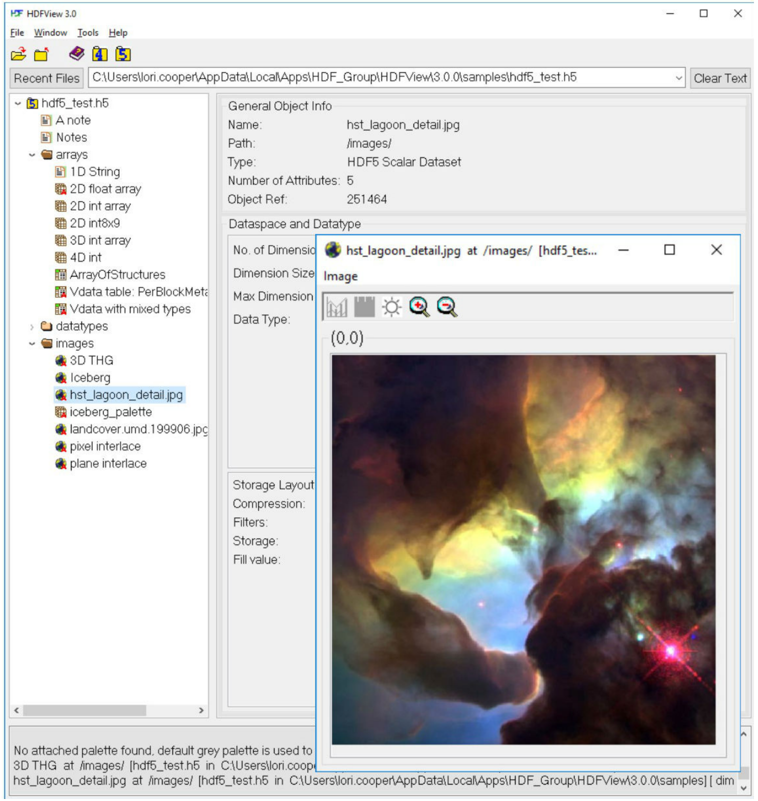
Other Features

- Float, integer, string, array, image … all in a single h5 file
- Compressed storage e.g. `gzip`
  - in `h5py` by setting `filter`



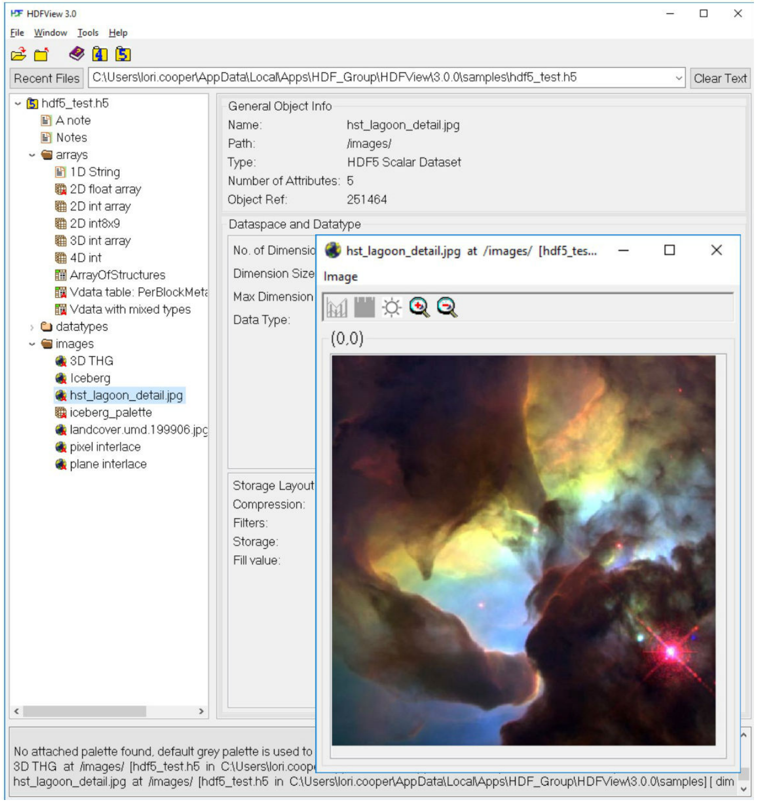
Other Features

- Float, integer, string, array, image ... all in a single h5 file
- Compressed storage e.g. `gzip`
  - in `h5py` by setting `filter`
- Suport meta data (to describe the data)
  - e.g. attributes of datagroup and datasets



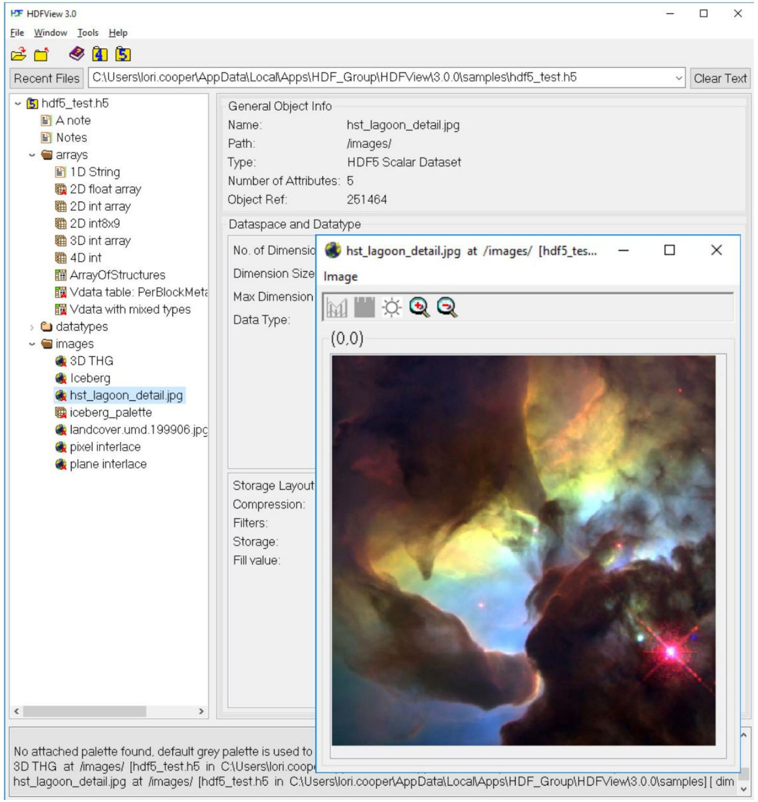
Other Features

- Float, integer, string, array, image … all in a single h5 file
- Compressed storage e.g. `gzip`
  - in `h5py` by setting `filter`
- Suport meta data (to describe the data)
  - e.g. attributes of datagroup and datasets
- Support parallel io (mpi)
  - tricky if file system does not have file lock



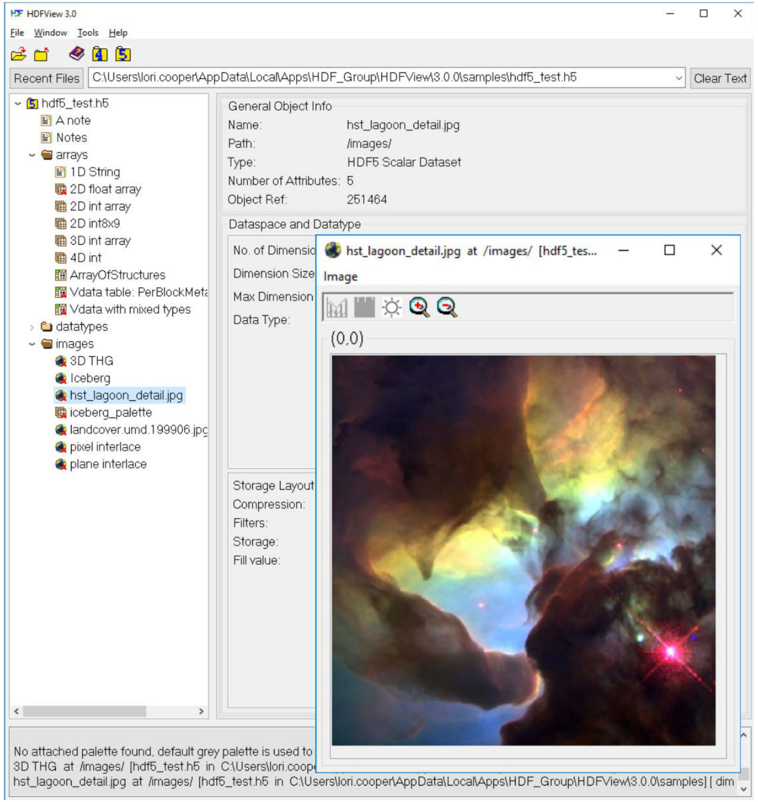
Other Features

- Float, integer, string, array, image … all in a single h5 file
- Compressed storage e.g. `gzip`
  - in `h5py` by setting `filter`
- Suport meta data (to describe the data)
  - e.g. attributes of datagroup and datasets
- Support parallel io (mpi)
  - tricky if file system does not have file lock
- Virtual dataset
  - "symbolic link" to data in other HDF5 files
  - not supported by *Mathematica*, Julia



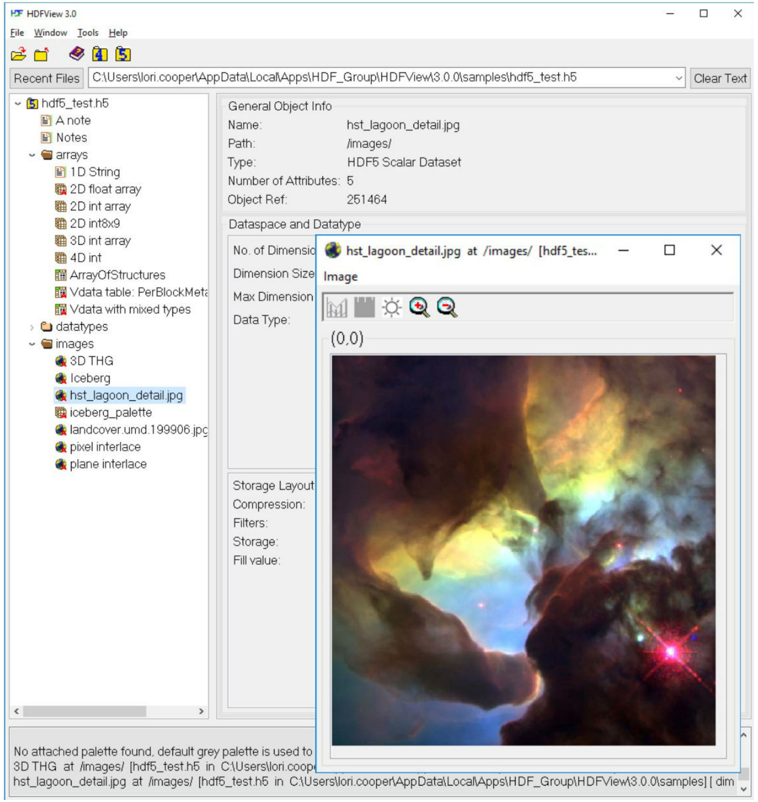
Other Features

- Float, integer, string, array, image … all in a single h5 file
- Compressed storage e.g. `gzip`
    - in `h5py` by setting `filter`
- Suport meta data (to describe the data)
    - e.g. attributes of datagroup and datasets
- Support parallel io (mpi)
    - tricky if file system does not have file lock
- Virtual dataset
    - "symbolic link" to data in other HDF5 files
    - not supported by *Mathematica*, Julia
- HDF WebService
    - alow access to data via HTTP, `h5serv`



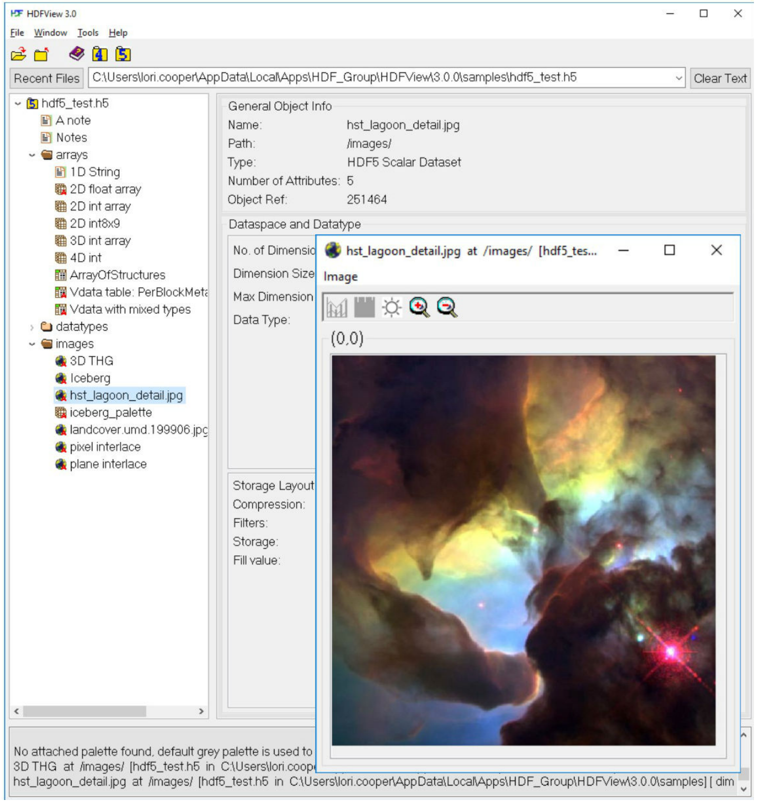
Other Features

- Float, integer, string, array, image ... all in a single h5 file
- Compressed storage e.g. `gzip`
  - in `h5py` by setting `filter`
- Suport meta data (to describe the data)
  - e.g. attributes of datagroup and datasets
- Support parallel io (mpi)
  - tricky if file system does not have file lock
- Virtual dataset
  - "symbolic link" to data in other HDF5 files
  - not supported by *Mathematica*, Julia
- HDF WebService
  - alow access to data via HTTP, `h5serv`
- ...



Other Features

- Float, integer, string, array, image … all in a single h5 file
- Compressed storage e.g. `gzip`
  - in `h5py` by setting `filter`
- Suport meta data (to describe the data)
  - e.g. attributes of datagroup and datasets
- Support parallel io (mpi)
  - tricky if file system does not have file lock
- Virtual dataset
  - "symbolic link" to data in other HDF5 files
  - not supported by *Mathematica*, Julia
- HDF WebService
  - alow access to data via HTTP, `h5serv`
- …
- > *Python and HDF5*, Andrew Collette, ISBN: 9781449367831 (2013)



iog to h5

- Based on `` `iog.so` ``, `` `iog_reader.py` ``, requiring `` `h5py` `` python package
  - iog2h5_2pt.py: convert 2pt iog files to h5
  - iog2h5_3pt.py: convert 3pt iog files to h5

> provided on demand