Summary of KAGALI PE code

Lee Hyung Won, Inje University

07 December, 2019(ver. 1)

Coll. with Jeongcho Kim and Chunglee Kim

Contents

Introduction

- Main object
  - Development for PE pipe line for KAGALI
- Code development period
  - 25th December, 2017 ~ 07th February 2018
- Result
  - Finished for the very simple PE pipe line
  - Still exist some bugs

Purpose of the code

- Generate sufficient number of independent samples distributed as posterior $\pi(\vec{\theta})$
- Posterior is

$$\pi(\vec{\theta}) = \mathcal{L}(\vec{\theta})P(\vec{\theta})$$

Posterior

Likelihood

Physical Prior

Metropolis-Hasting algorithm

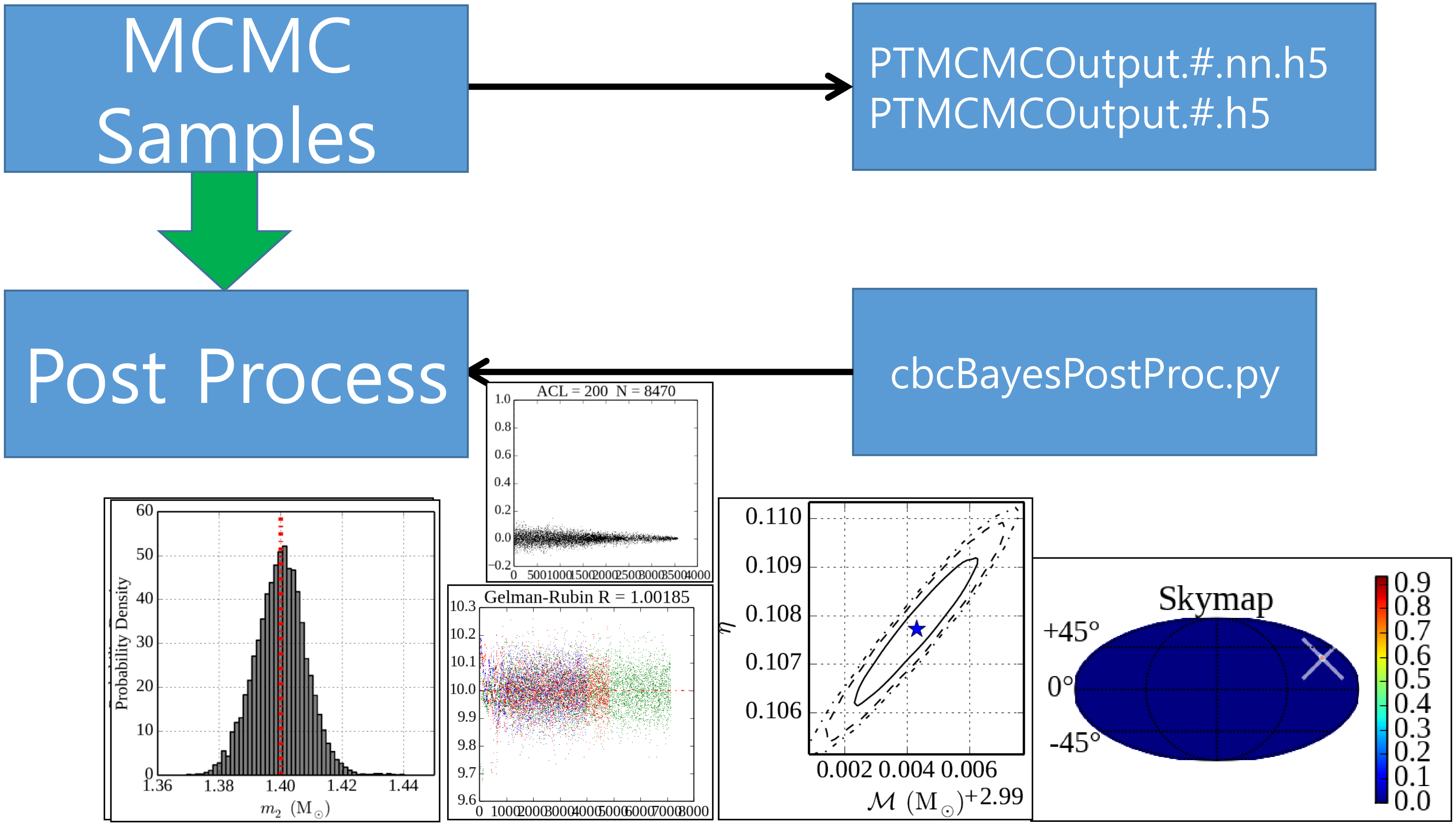
- Generate proposed parameter : $\vec{\theta}_* \sim q\left(\vec{\theta}_i \to \vec{\theta}_*\right)$
- Sample from uniform distribution : $u \sim U(0,1)$
- Calculate Hasting ratio : $H = \frac{\pi\left(\vec{\theta}_*\right) q\left(\vec{\theta}_i \to \vec{\theta}_*\right)}{\pi\left(\vec{\theta}_i\right) q\left(\vec{\theta}_* \to \vec{\theta}_i\right)}$
- if $u < \min(1, H)$, then $\vec{\theta}_{i+1} = \vec{\theta}_*$
- else $\vec{\theta}_{i+1} = \vec{\theta}_i$
- For Gaussian proposal : $H = \frac{\pi\left(\vec{\theta}_*\right)}{\pi\left(\vec{\theta}_i\right)}$

Parallel Tempering

- Posterior at Temperature T
- $\pi(\vec{\theta}) = \mathcal{L}(\vec{\theta})^{1/T} P(\vec{\theta})$

- Swapping ratio
- $SwapR = \left(\frac{\mathcal{L}(\vec{\theta}_2)_{T_2}}{\mathcal{L}(\vec{\theta}_1)_{T_1}} \right)^{\left(\frac{1}{T_1} - \frac{1}{T_2}\right)}$, $T_1 < T_2$
- Swap when $u < \min(1, SwapR)$

Parameter Estimation with Bayesian inference(lalsuite)



Parameter Estimation with Bayesian inference(kagali)

MCMC Samples

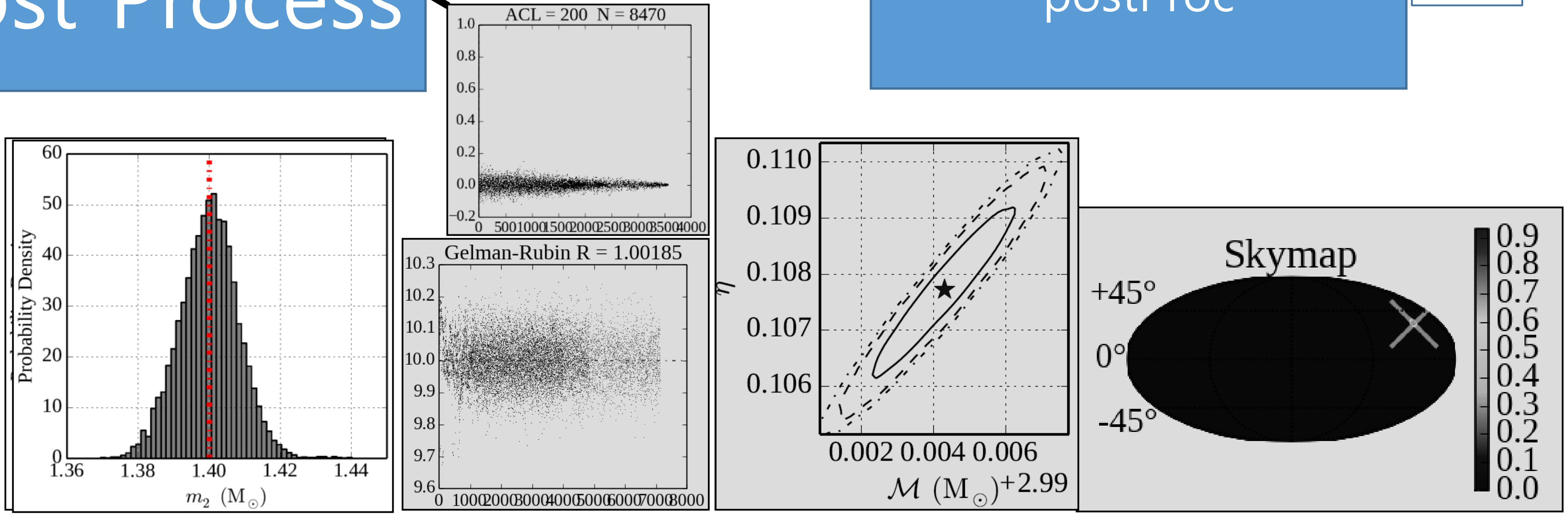
PTMCMC.output.#.nn.out
PTMCMC.output.#.out

inferenceMCMC

Post Process

postProc

*.ini



Likelihood

- $\mathcal{L}(\vec{\theta}) = e^{-\frac{1}{2}\langle d - h(\vec{\theta}) | d - h(\vec{\theta}) \rangle}$
- $\langle h_1 | h_2 \rangle = 4\Re \int_{f_{low}}^{f_{high}} \frac{h_1^* h_2}{S_h(f)} df$
- $logl = -\frac{1}{2}\langle d - h(\vec{\theta}) | d - h(\vec{\theta}) \rangle$
- $nullLogl = -\frac{1}{2}\langle d | d \rangle$
- $deltalogl = logl\ - nullLogl = \frac{1}{2}\langle d | h(\vec{\theta}) \rangle + \frac{1}{2}\langle h(\vec{\theta}) | d \rangle - \frac{1}{2}\langle h(\vec{\theta}) | h(\vec{\theta}) \rangle$
- $optimal\_snr = \sqrt{\langle h(\vec{\theta}) | h(\vec{\theta}) \rangle}$
- $\mathrm{m}atched\_filter\_snr = 2\langle d | h(\vec{\theta}) \rangle / \sqrt{\langle h(\vec{\theta}) | h(\vec{\theta}) \rangle}$

Prior

- All physical priors are assumed to be 1 (a priori we do not know)
- Distance prior comes from volume integration factor $\int d^3\vec{r} = \int r^2 dr \sin\theta d\theta d\phi = \int r^2\, dr d\mu d\phi, \mu = \cos\theta$
- Mass prior comes from parameter space change $\int dm_1\, dm_2 = \int \frac{m_1 m_1}{m_c} dm_c\, dq = \int \frac{(m_1+m_2)^2}{(m_1-m_2)\eta^{-3/5}} dm_c d\eta$
- $\int r^2 dr d\mu d\phi = \int r^3 d \ln r\, d\mu d\phi$

$$\int \frac{m_1 m_1}{m_c} dm_c\, dq = \int \frac{(m_1+m_2)^2}{(m_1-m_2)\eta^{-3/5}} dm_c d\eta$$

distance prior

mass prior for mc and q

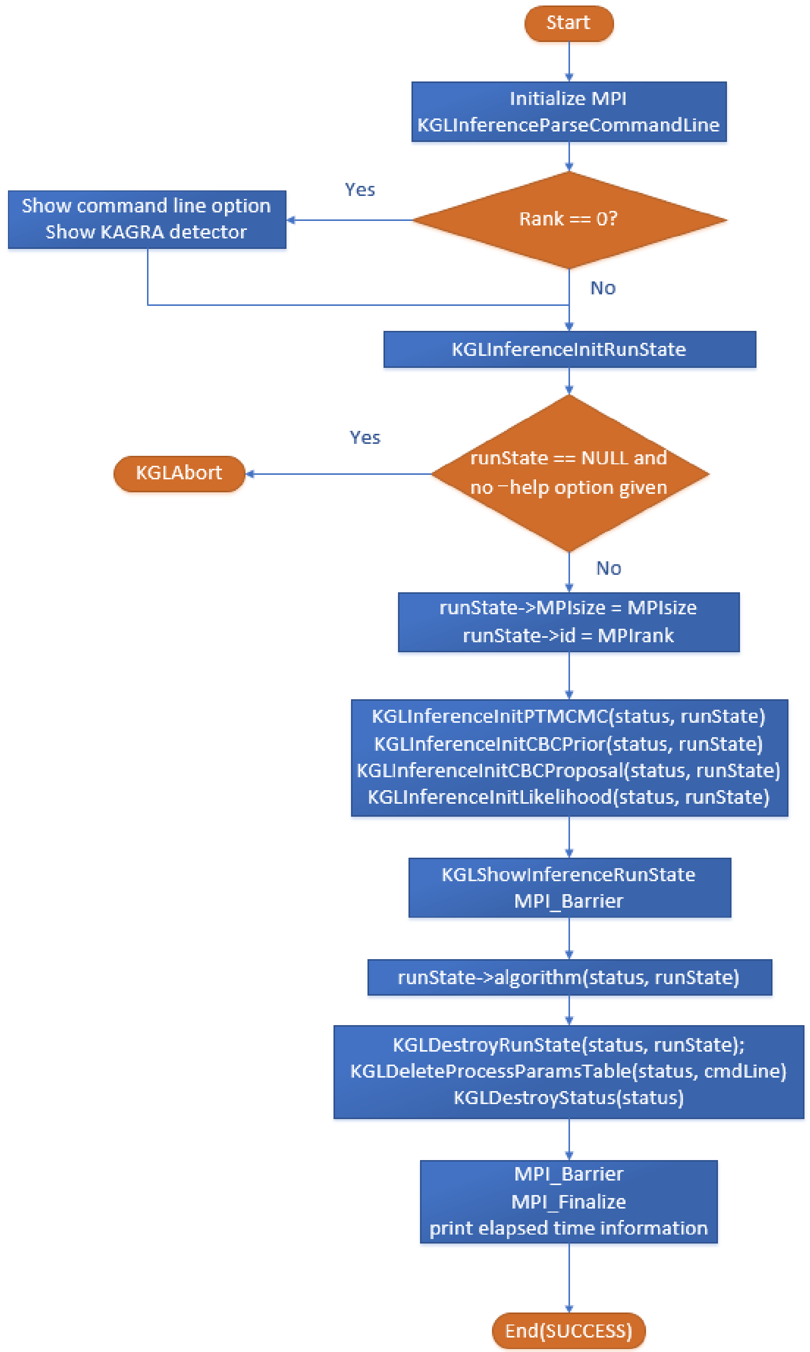
mass prior for mc and eta

inferenceMCMC.c

- main
  - Initialize MPI environment
  - Initialize RunState
  - Initialize PTMCMC
  - Initialize CBCPrior
  - Initialize CBCProposal
  - Initialize Likelihood
  - Call PTMCMC function(runState->algorithm)
  - Finalize MPI environment

inferenceMCMC.c

/kagali-v0r4a/kagaliapps/cbc/mpisrc/inferenceMCMC.c main()



KGLCommandUtil.c

- KGLParseCommandLine
  - For all command line options add option pair to ProcessParamTable

KGLInferenceRunState.c

- Allocate RunState variable
- ReadData for IFOs
- Allocate memories for parameters
- Set PT algorithm function pointer for runState->algorithm
  KGLParellelTemperingAlgorithm
- Set PT one step function pointer for runState->onestepFunction
  KGLInferenceMCMCOneStep
- Initialize random number generator
- Set random seed

KGLInferenceMCMC.c

- Initialize PT MCMC
  - Set various option values for PT algorithm
  - Set temperature ladder
  - Set function pointer for PTAlgorithm and OneStep
  - Randomize for each MPI process
- Initialize temperature ladder
  - Set $T(n) = T_{Min}(\Delta T)^n$
- Initialize Likelihood
  - Set likelihood function pointer for runState->likelihood

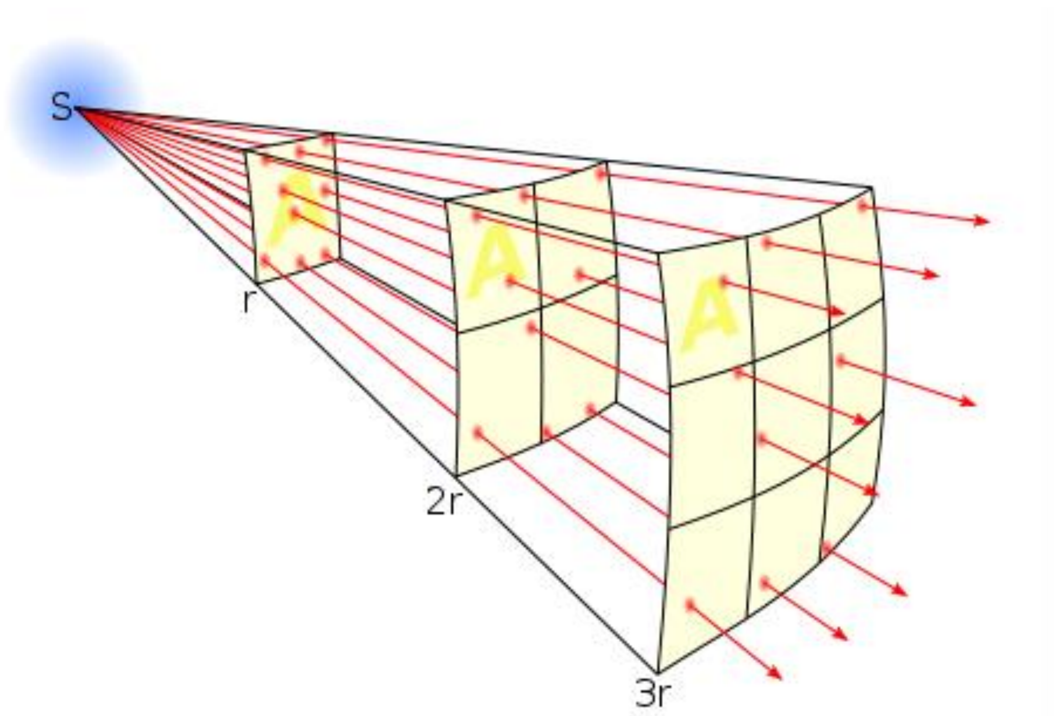
KGLInferencePrior.c

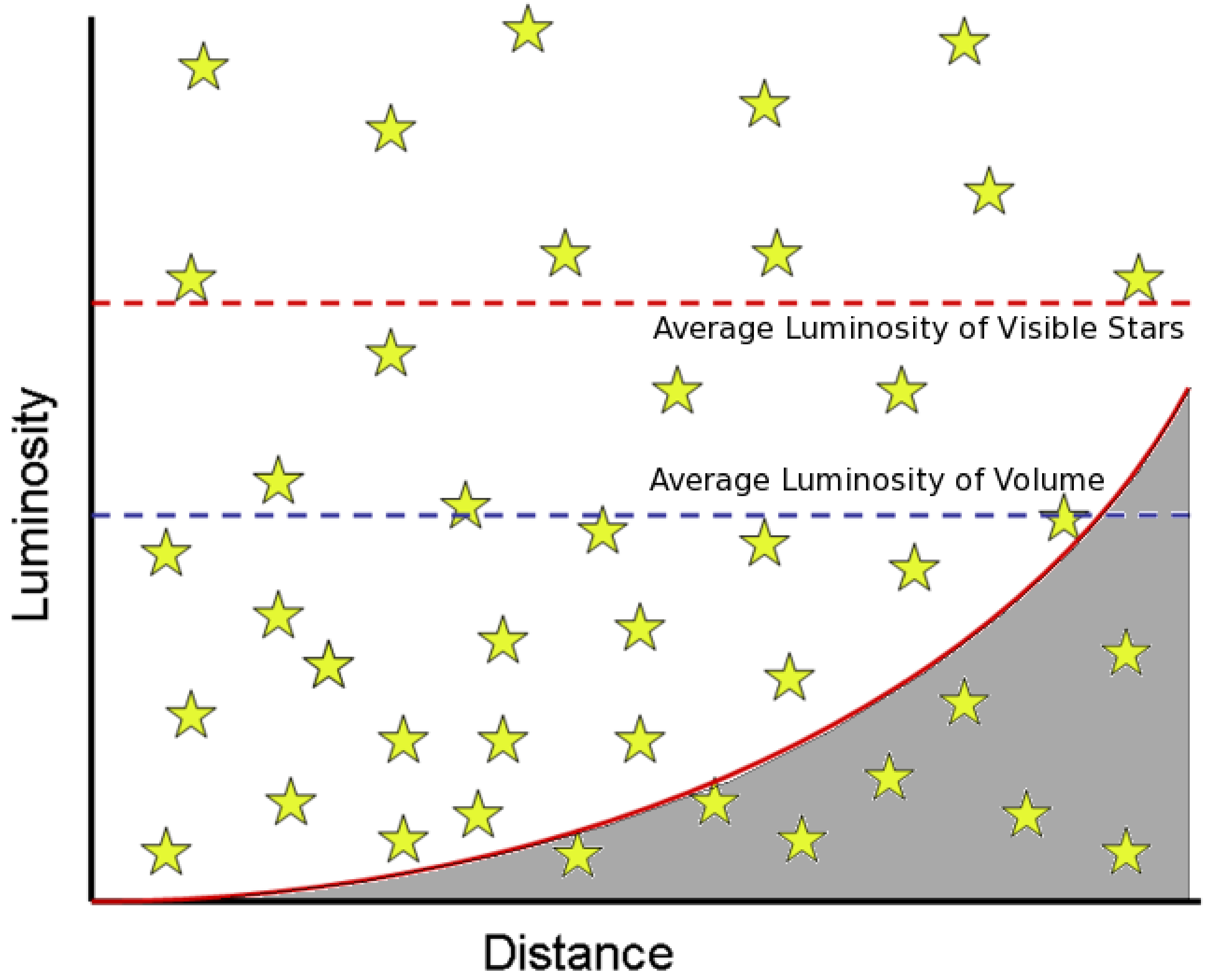
- KGLInferenceNullPrior(KGLStatus *, KGLInferenceVariable *, KGLInferenceParams *)
  - Return zero value for prior
- KGLInferenceInspiralPrior(KGLStatus *, KGLInferenceVariable *, KGLInferenceParams *)
  - Calculate log prior for distance, declination, masses, Malmquist correction, spin priors
- KGLInferenceInspiralCubeToPrior(KGLStatus *, KGLInferenceVariable *, KGLInferenceParams *, double *)
  - Convert unit random number to real parameter value

Malmquist bias

- Oxford reference
  - A statistical selection effect that arises in astronomical surveys that are complete to some apparent magnitude limit. At large distances from the observer, only objects that are intrinsically luminous can be seen. Nearer the observer, objects with average or below-average luminosity can also be seen. The statistical properties of the sample therefore depend on distance from the observer in a complicated way. This form of bias, first described in 1924 by the Swedish astronomer Karl Gunnar Malmquist (1893–1982), can be avoided by forming a more restricted *volume-limited sample*.
    Nowadays, the term Malmquist bias is often used to describe the systematic bias on a measured quantity due to random observational errors. For example, random errors in magnitude measurements will lead to an overestimate of the number of galaxies to a given magnitude limit, because there are more galaxies fainter than the limit which are scattered into the sample by measurement errors than there are galaxies brighter than the limit which are scattered out. A random measurement error thus leads to a systematic bias.
- https://en.wikipedia.org/wiki/Malmquist_bias

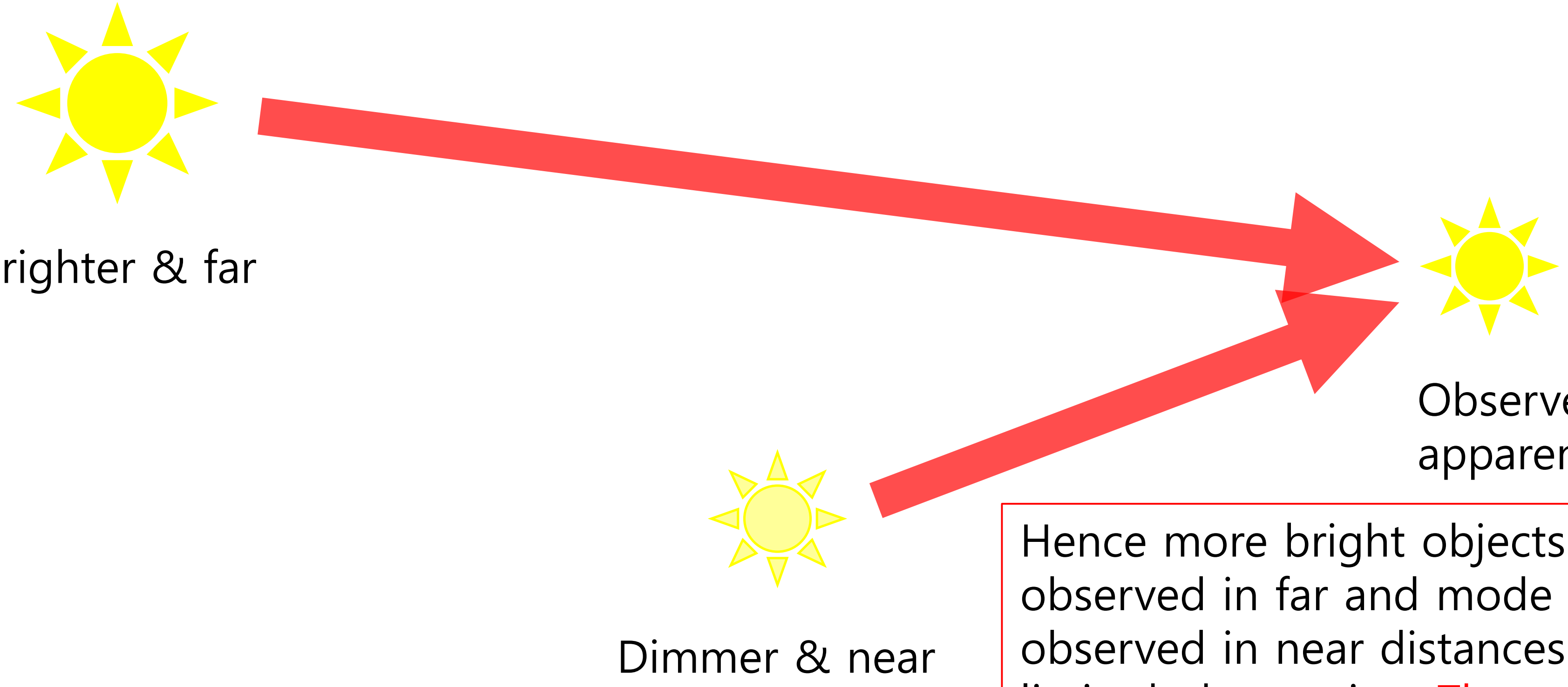
Malmquist bias





We need to include selection effect with SNRs, since GW observation is SNR limited observation.

Malmquist bias



Hence more bright objects than actual will be observed in far and mode dim objects than actual observed in near distances in luminosity limited observation. Thus number statistics will be biased.
GW observation is SNR limited observation.

Malmquist bias

- Typically, when looking at an area of sky filled with stars, only stars that are brighter than a limiting apparent magnitude can be seen. As discussed above, the very luminous stars that are farther away will be seen, as well as luminous and faint stars that are closer. There will appear to be more luminous objects within a certain distance from Earth than faint objects. However, there are many more faint stars, they simply cannot be seen because they are so dim. The bias towards luminous stars when observing a patch of sky affects calculations of the average absolute magnitude and average distance to a group of stars. Because of the luminous stars that are at a further distance, it will appear as if our sample of stars is farther away than it actually is, and that each star is intrinsically brighter than it actually is. This effect is known as the Malmquist bias.
- When studying a sample of luminous objects, whether they be stars or galaxies, it is important to correct for the bias towards the more luminous objects. There are many different methods that can be used to correct for the Malmquist bias as discussed below.
- The Malmquist bias is not limited to luminosities. It affects any observational quantity whose detectability diminishes with distance.

Malmquist bias in GW

- New Journal of Physics 15(2013), 053027, Messenger & Veitech

Avoiding selection bias in gravitational wave astronomy

C Messenger[1] **and J Veitch**[2]

[1] School of Physics and Astronomy, Cardiff University, Queen's Buildings, The Parade, Cardif, Wales CF24 3AA, UK
[2] Nikhef, Science Park 105, Amsterdam 1098-XG, Netherlands
E-mail: chris.messenger@astro.cf.ac.uk and johnv@nikhef.nl

New Journal of Physics **15** (2013) 053027 (12pp)
Received 9 January 2013
Published 16 May 2013
Online at http://www.njp.org/
doi:10.1088/1367-2630/15/5/053027

Malmquist correction

- Impose selection effects on the prior(within_malmquist)
- Loudest-snr
- Second-loudest-snr
- Network-snr

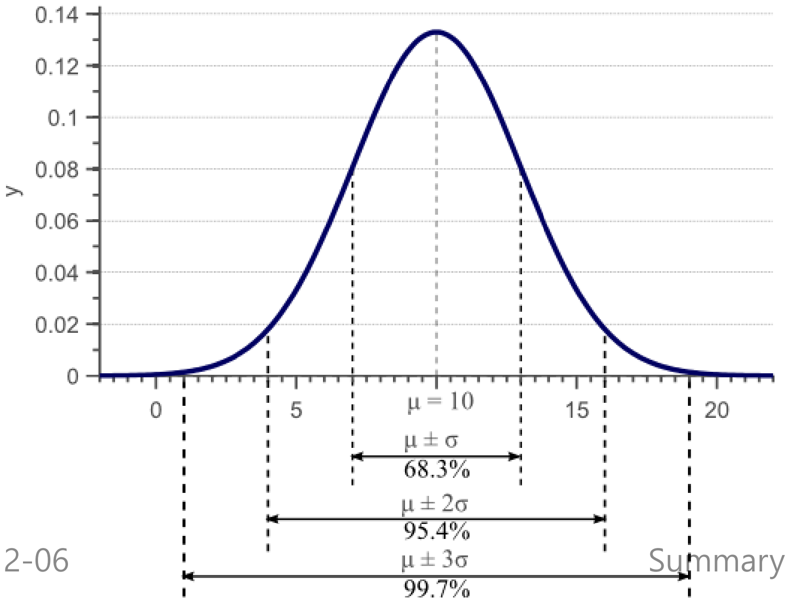
- if (loudest < Loudest-snr || second-loudest < Second-loudest-snr || model-snr < Network-snr) -> logPrior = -INFINITY and CubeToPrior returns 0

Prior function

- Distance prior
  - Logarithmic
  - Uniform
  - $\int r^2 \, dr d\mu d\phi = \int r^3 d \ln r \, d\mu d\phi$
- Mass prior
  - mc, q
  - mc, eta
  - $\int dm_1 dm_2 = \int \frac{m_1 m_1}{m_c} dm_c \, dq = \int \frac{(m_1+m_2)^2}{(m_1-m_2)\eta^{-3/5}} dm_c d\eta$

KGLInferenceProposal.c

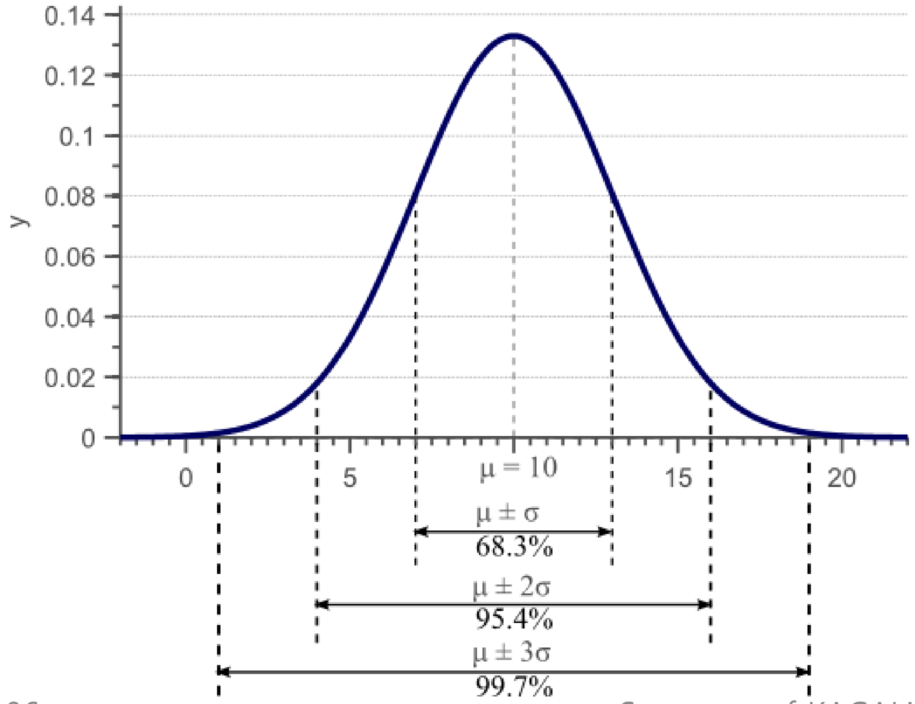
- KGLInferenceSingleProposal(KGLStatus *, KGLInferenceRunState *, KGLInferenceParams * current, KGLInferenceParams * proposed)
  - Generate a Gaussian proposal value for a single parameter
  - Return value is ratio of proposal probability, it is zero



Proposal function

- Symmetric Gaussian jump proposal

```
param->value.double_value += gsl_ran_ugaussian(GSLrandom)*sigma;
```



KGLInferenceSampler.c

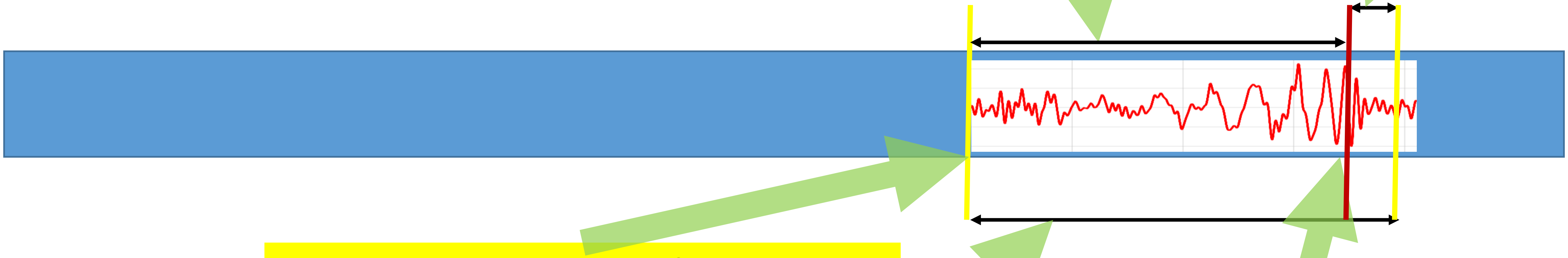
- KGLParellelTemperingAlgorithm
- KGLInferenceMCMCOneStep
- KGLInferencePTSwap

Likelihood function

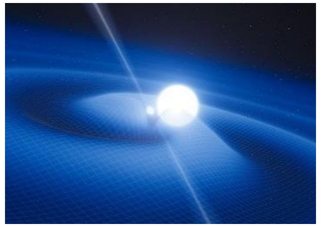
- KGLInferenceZeroLogLikelihood
  - Returns zero for test purpose
- KGLInferenceNullLogLikelihood
  - Calculate overlap using only strain data
  - $nullLogl = -\frac{1}{2}\langle d|d\rangle$
- KGLInferenceFusedFreqDomainLogLikelihood
  - Calculate overlap strain data subtracted projected template strain
  - $logl = -\frac{1}{2}\left\langle d - h(\vec{\theta})\middle|d - h(\vec{\theta})\right\rangle$

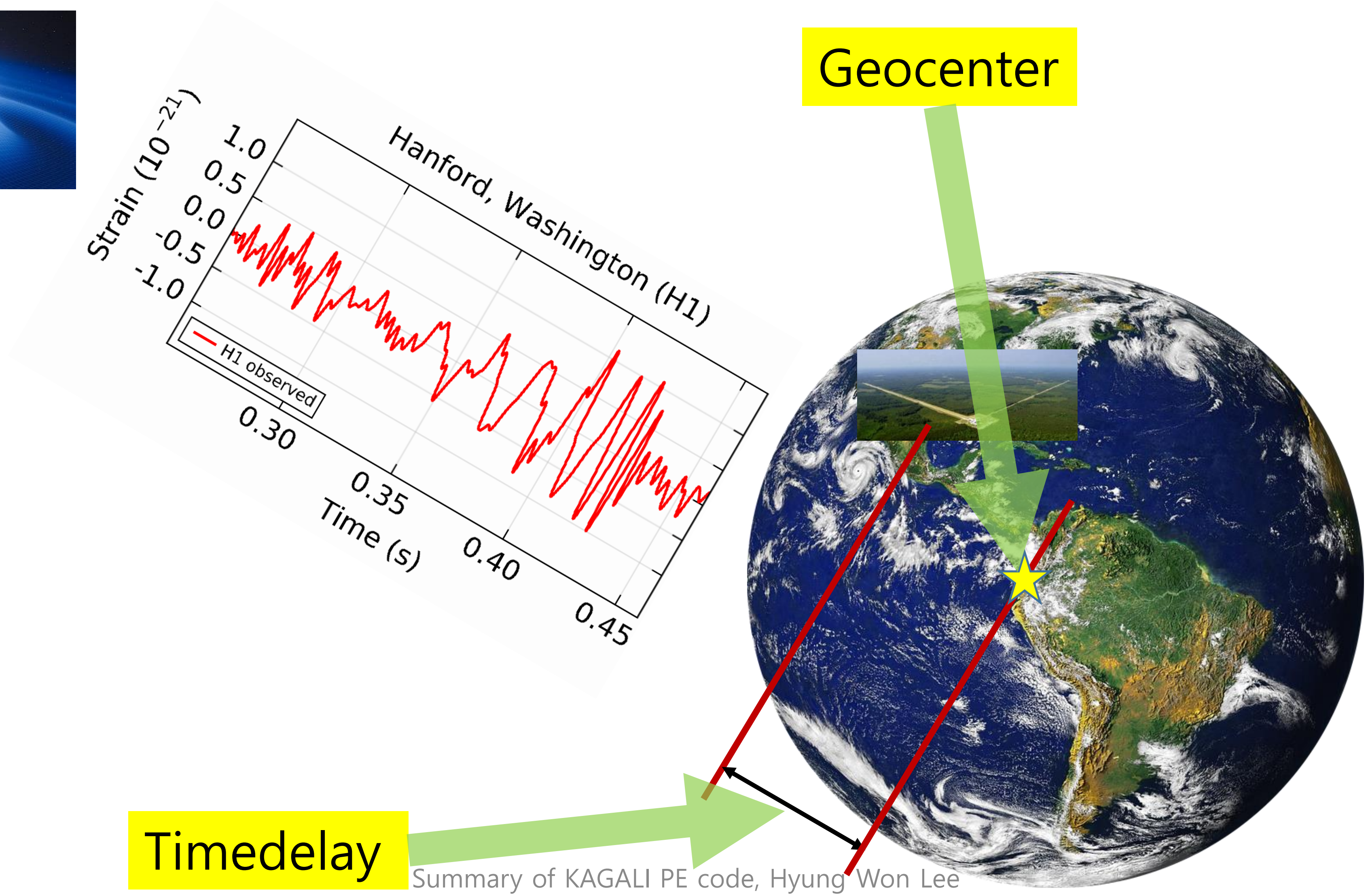
Time relations

- --trigtime : This option specifies the trigger time at geocenter, this time is same as injection time in injection table
- ifo->epoch : this time is segment start time



Timedelay



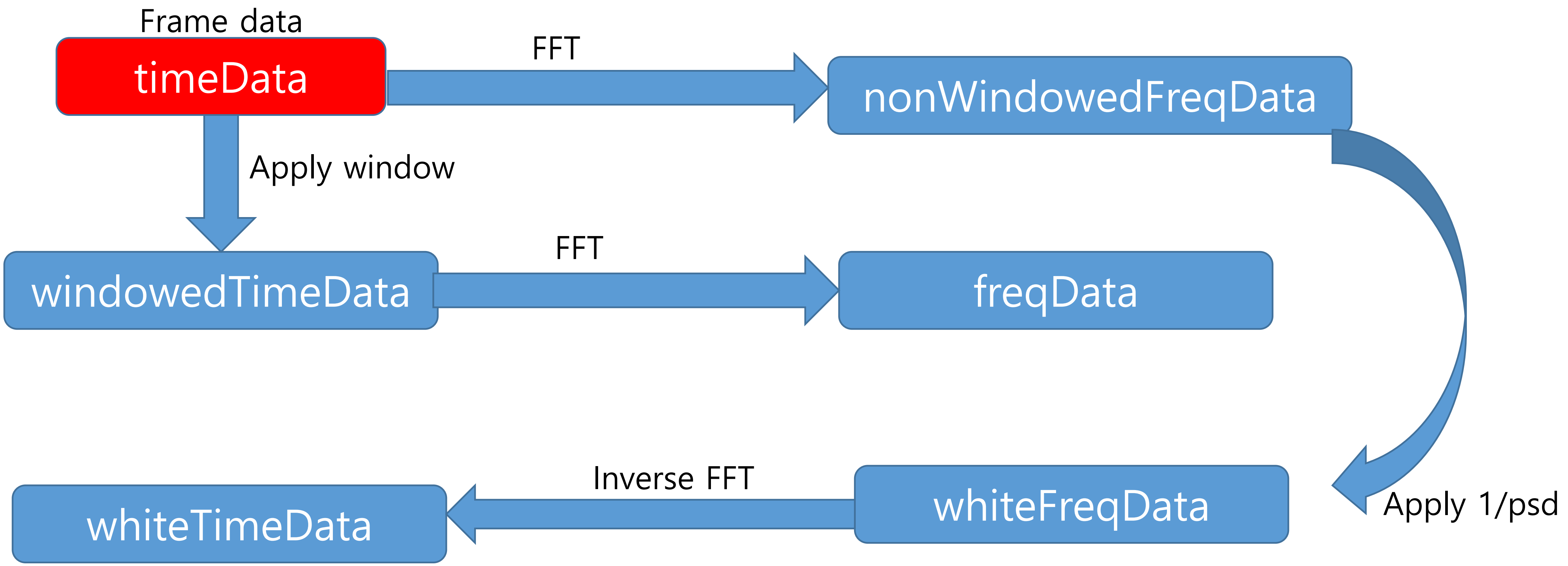


Hanford, Washington (H1)
Strain (10^{-21})
1.0
0.5
0.0
-0.5
-1.0
H1 observed
0.30
0.35
0.40
0.45
Time (s)
Geocenter
Timedelay

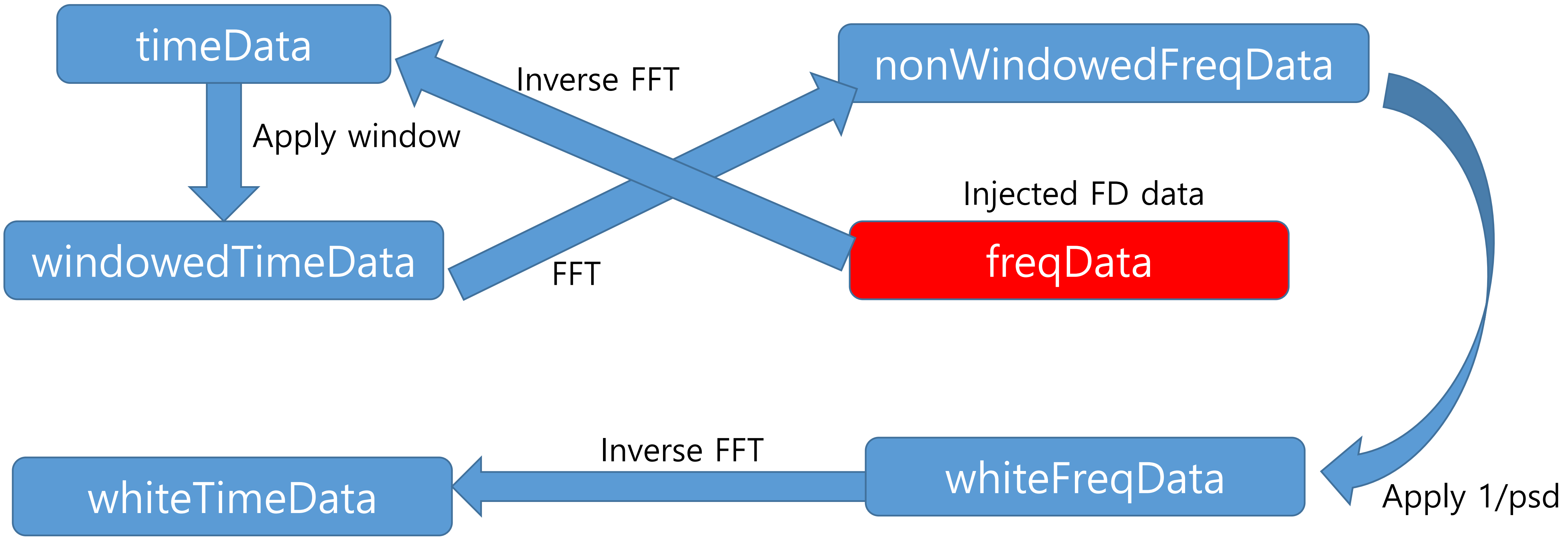
KGLInferenceReadData.c

- Read frame data file
  - Resample data file for time series
  - Evaluate PSD from frame data using Welch method(MedianMean)
- Inject a template wave form
  - Injection file read
  - Noise realization

KGLInferenceIFOData structure



KGLInferenceIFOData structure



Options implemented

- $./inferenceMCMC --help shows all options

```
#define USAGE  "\
 ==== Options for KGLInferenceReadData related to MCMC ========\n\
   --srate SRATE : sampling rate, we assume all detector has the same value [4096, Hz] \n\
   --seglen SEGLEN : segment length, we assume all detector has the same value [16, s] \n\
   --ifo H1,K1 : comma separated list of IFOs \n\
   --flow f1,f2 : comma separated list of low frequencies[defaut:10Hz], integration range in FD is flow~fhigh\n\
   --fhigh f1,f2 : comma separated list of high frequencies[default:srate/2 - 1/seglen]\n\
   --strain file1,file2 : comma separated list of strain frame files names\n\
                          file1.gwf, file2.gwf for real strain frame files, file should exist in the current directory\n\
                          approx1, approx2 for injection waveforms, usually these are the same\n\
   --channel channel1,channel2 : comma separated list of channel names when given real strain frame data files\n\
   --psd psd1,psd2 : comma separated list of psd options[default: use strain frame files]\n\
                     sim1, sim2 for simulated PSD data\n\
                           possible values are SimIniLIGO          : initial LIGO analytic function\n\
                                               SimAdvLIGO          : advanced LIGO analytic function\n\
                                               SimAdvVIRGO         : advanced VIRGO analytic function\n\
                                               SimGEO              : GEO analytic function\n\
                                               SimEGO              : EGO analytic function\n\
                                               SimTAMA             : TAMA analytic function\n\
                                               SimETB              : Einstein B analytic function\n\
                                               DesigniKAGRA        : Design sensitivity PSD file for iKAGRA(unknwon)\n\
                                               DesignbKAGRA-vrsed  : Design sensitivity PSD file for bKAGRA(BW2009_VRSED.dat)\n\
                                               DesignbKAGRA-vrseb  : Design sensitivity PSD file for bKAGRA(BW2009_VRSEB.dat)\n\
                                               DesignaLIGO-bhbh    : Design sensitivity PSD file for adv. LIGO(AdvLIGO_BHBH_20deg.dat)\n\
                                               DesignaLIGO-hf      : Design sensitivity PSD file for adv. LIGO(AdvLIGO_High_freq.dat)\n\
                                               DesignaLIGO-no-srm  : Design sensitivity PSD file for adv. LIGO(AdvLIGO_NO_SRM.dat)\n\
                                               DesignaLIGO-nsns    : Design sensitivity PSD file for adv. LIGO(AdvLIGO_NSNS_Opt.dat)\n\
                                               DesignaLIGO-zdhp    : Design sensitivity PSD file for adv. LIGO(AdvLIGO_ZERO_DET_high_P.dat)\n\
                                               DesignaLIGO-zdlp    : Design sensitivity PSD file for adv. LIGO(AdvLIGO_ZERO_DET_low_P.dat)\n\
                     file1.dat, file2.dat for known or private PSD data files\n\
                     file1.gwf, file2.gwf for real strain frame files from which calculate PSD(ex. average, etc.)\n\
   --psdchannel channel1,channel2 : comma separated list of channel names when given real strain frame data files for psd[default:use channel option]\n\
   --psdstart gps1,gps2 : comma separated list of gps times for psd start[default:start of frame file]\n\
   --psdlength len1,len2 : comma separated list of length in seconds for psd[default:GPSTrig - seglen + TRIG_MARGIN - psdStart]\n\
   --psdresample : use resampled frame data for PSD calculation[default:no]\n\
   --window hann,rectangular : comma separated list of window function for each IFOs[default:hann]\n\
   --trigtime trigger : time in 113847.5747 style in geocenter \n\
   --inj inj_file : injection data file name, this is not xml file it is plain text file similar to *.ini file.\n\
                    injection file data override --strain option for approximants\n\
   --noise n1,n2 : comma separated noise realization for IFOs, -1:random seed, 0:no-noise, #:given seed[defaut:0]\n\
   --inj_id n : injection id in the given injection file.\n\
   --dumptime : dump time series data[default:no]\n\
   --dumpfreq : dump frequency series data[default:no]\n\
   --dumppsd : dump psd and asd data[default:no]\n\
 ===============================================================\n\
"
```

--ifo option

- --ifo H1,K1
- comma separated list of IFOs

- Example
- --ifo H1,L1,V1,K1

--strain option

- --strain file1,file2
- comma separated list of strain frame files names
- file1.gwf, file2.gwf for real strain frame files, file should exist in the current directory or
- approx1, approx2 for injection waveforms, usually these are the same

- Example
- --strain H-H1_HOFT_C00_T1700401_v1-1187005000-4096.gwf,L-L1_HOFT_C00_T1700401_v1-1187005000-4096.gwf,V-V1O2Repro1A-1187005000-5000.gwf,TaylorF2

--channel option

- --channel channel1,channel2
- comma separated list of channel names when given real strain frame data files

- Example
- --channel H1:GDS-CALIB_STRAIN_T1700401_v1,L1:GDS-CALIB_STRAIN_T1700401_v1,V1:Hrec_hoft_V1O2Repro1A_16384Hz,V1:Hrec_hoft_V1O2Repro1A_16384Hz

--psd option

- --psd psd1,psd2
- comma separated list of psd options[default: use strain frame files]

Other options

- Other option can be understandable or figure out from help message

ReadData()

- if --help is given then print help and stop
- Get number of IFOs(--ifo)
- Set trigger time (--trigtime) it is mendatory
- Read injection table if option(--inj) given and show
- For each IFO do
  - Set IFO epoch as trigger time
  - Set IFO Data
  - Update network SNR
- Print network SNR
- Destroy Injection table

SetIFOData()

- Check if the prefix exist
- Set IFO frequency range (--flow, --fhigh)
- Set window function for IFO (--window)
- Set strain data (--strain)
- Set PSD data (--psd)
- Apply time shift for IFO
- Set white data for IFO only for real frame data
- Set noise realization for IFO only for injection data
- Dump data

SetIFODataFrequency()

- If --flow is given then set ifo->fLow as that value
- else set ifo->fLow as DEFAULT_FLOW(10.0Hz)

- If --fhigh is given then set ifo->fHigh as that value
- else set ifo->fHigh as srate – 1.0/seglen

- fLow and fHigh is actual integration range for snr and likelihood calculation

SetIFODataWindowFunction()

- If --window option is given set ifo->window as that value
- else set DEFAULT_WINDOW(Hann)

- This window function will be used for PSD calculation and windowedTimeSeries data

SetIFOStrainData()

- If --strain option is not given abort the program
- If approximant name is given
  - Check exist injection table
  - Set ifo->srate as --srate option value or default value(4096Hz)
  - Set ifo->seglen as --seglen option value or default value(16s)
  - Set appropriate frequency range
  - Call template function
  - Set response and time shift
  - Set type as Injected data
- else *.gwf is given
  - Read frame data(next slide)
- else abort the prorgam

Read frame data

- Check channel name is given (--channel)
- Check trigger time is within the frame data
- Set srate and seglen
- Resample frame data
- Set time data
- Set frequency data
- Set windowed time data
- Set windowed frequency data
- Set type as frame data

SetIFOPsdData()

- If --psd option is given
  - Set psd data
- else if --strain option is given
  - If frame file is given
    - Estimate psd using frame data
  - else if approximant is not gven
    - Set psd using “aLIGO-anl”
  - else
    - Abort the program
- else
  - Abort the program

SetIFOWhiteData()

- If type is inject data, then return
- Calculate white frequency data from frequency data
- Calculate reverse FFT to get white time data

ApplyIFOTimeShift()

- Apply time shift in frequency domain
- Calculate SNR for this IFO

SetIFONoiseData()

- If type is frame data, then return
- If --noise option is not given or -1 then
  - Set noise seed as random
- else if option is zero
  - No noise case return
- else
  - Set noise seed as the given value
- Add Gaussian noise in frequency data

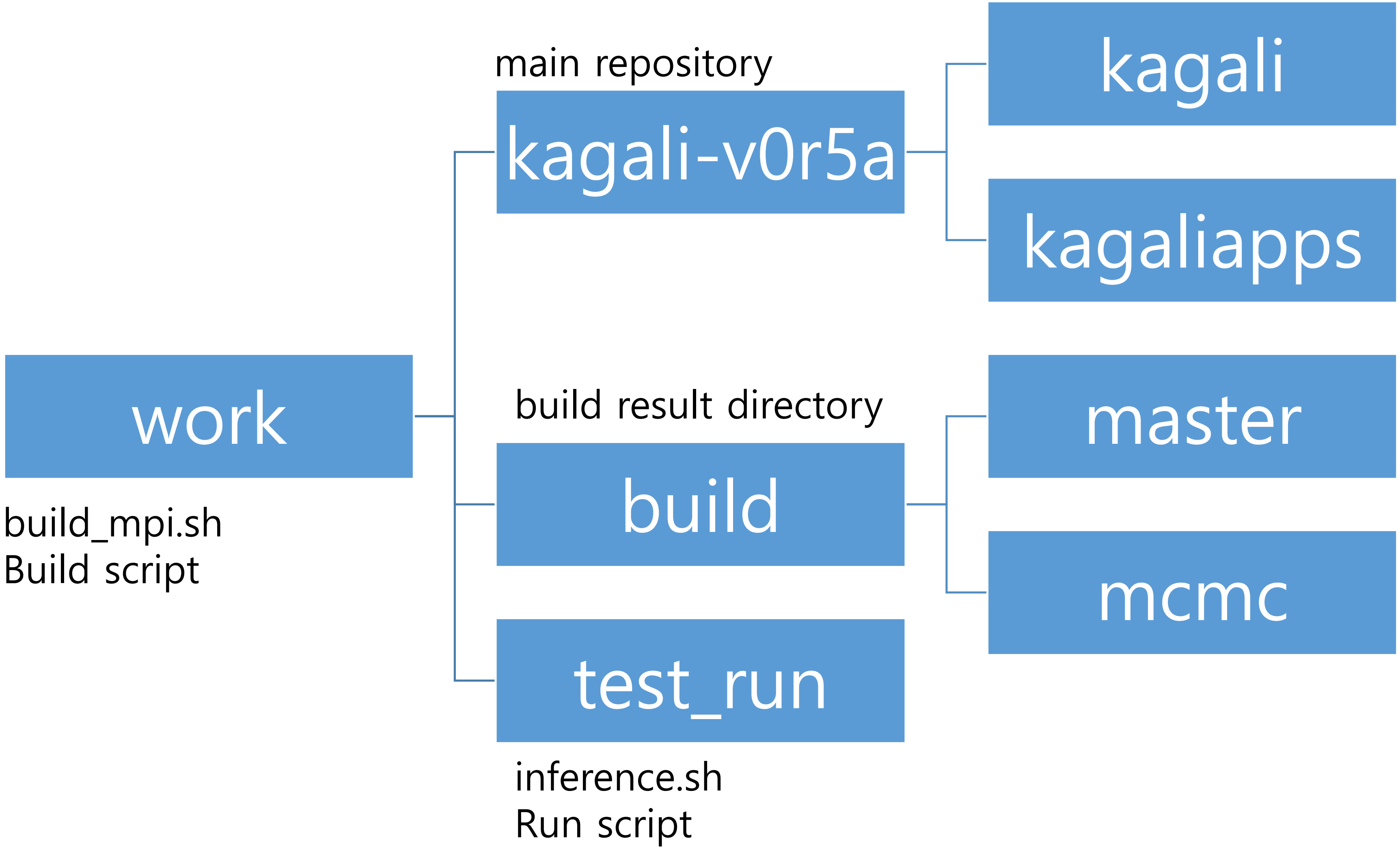
Dump data

- If --dumptime option is given
  - Dump time data (file : IFO_time_series_check.dat, IFO is H1, K1)
- If --dumpfreq option is given
  - Dump frequency data (file : IFO_freq_series_check.dat, IFO is H1, K1)
- If --dumppsd option is given
  - Dump power spectral data (file : IFO_psdasd_series_check.dat, IFO is H1, K1)

Post Processing(/kagali/inference/postproc/src)

- KGLParseCommandLineFromFile()
  - Read command line option from ini style file
- KGLReadInferencePostProcTable()
  - Read MCMC output file
- KGLMakeBinPostProc()
  - Make 1D bins for posterior samples
- KGLSaveBinAllPostProc()
  - Save 1D bins for all listed variable
- KGLSaveBinPostProc()
  - Save 1D bin for a specific variable

Directory structure



Preparing source

- Make a working directory, like work
- Clone kagali
- $cd work
- $git clone https://vt001.resceu.s.u-tokyo.ac.jp/git/kagali-v0r5a

- Check out mcmc branch
- $git fetch origin
- $git checkout –b mcmc origin/mcmc

Build application

- $cd work
- $nohup ./build_mpi.sh &
- $tail –f nohup.out
- Sample build result

```
rm -f config.h stamp-h1
rm -f libtool config.lt
rm -f TAGS ID GTAGS GRTAGS GSYMS GPATH tags
rm -f cscope.out cscope.in.out cscope.po.out cscope.files
This command is intended for maintainers to use
it deletes files that may require special tools to rebuild.
make[1]: Leaving directory '/home/hwlee/projects/KGL/kagali/kagali-v0r4a/kagaliapps'
rm -f config.status config.cache config.log configure.lineno config.status.lineno
rm -rf ./autom4te.cache
rm -f Makefile



==========**********************
 built for branch mcmc
 Do command source /home/hwlee/projects/KGL/kagali/build/mcmc/etc/kagali-user-env.[c]sh to use this software
2018. 01. 15. (월) 11:41:29 KST
==========**********************
```

Run application

- $cd test_run
- $./inference.sh
- Sample result

```
 xArm Altitude    :            3.141400e-03[rad]
 xArm Azimuth     :             1.054113e+00[rad]
 yArm Altitude    :           -3.627000e-03[rad]
 yArm Azimuth     :            -5.166798e-01[rad]
 xArm Midpoint    :             1.513254e+03[m]
 yArm Midpoint    :             1.511611e+03[m]
==============================================================
ifoData = 0x14ade20
cmdLine = 0x14aa610
RunState = 0x14a6fb0
RunState = 0x14a6fb0
RunState = 0x14a6fb0
 ==== This is thread 0 of 1 ====
 ======= run MCMC sampler ======= for thread 0 of 1
KGLInferenceOneStep called
KGLInferenceOneStep called
KGLInferenceOneStep called
KGLInferenceOneStep called
KGLInferenceOneStep called
KGLInferenceOneStep called
KGLInferenceOneStep called
KGLInferenceOneStep called
KGLInferenceOneStep called
KGLInferenceOneStep called
KGLInferenceOneStep called
Thread 0 has 11 effective samples. Stopping...
========== main(): finished. ==========
```

Run script

```
#!/bin/bash
masterdir=/home/hwlee/projects/KGL/kagali/build/mcmc
#masterdir=/home/hwlee/projects/KGL/kagali/build/mcmc_tmp
source $masterdir/etc/kagali-user-env.sh
export KGL_NOISEDIR=/home/hwlee/projects/KGL/kagali/test_run/noisedata
export KGL_INJECTIONDIR=/home/hwlee/projects/KGL/kagali/test_run/injection
export KGL_CACHEDIR=/home/hwlee/projects/KGL/kagali/frames
#export MALLOC_CHECK_=1
mpirun -np 1 $masterdir/bin/inferenceMCMC --zeroLogLike --proposal uniform --approximant Taylor
F2 --f_start 25.0 --ifo H1,L1 --trigtime 1126259462 --seglen 16.0 --strain H-H1_LOSC_16_V2-1126
257414-4096.gwf,L-L1_LOSC_16_V2-1126257414-4096.gwf --channel H1:LOSC-STRAIN,L1:LOSC-STRAIN --d
umptime --dumpfreq --dumppsd --window hann,hann --psdstart 1126257414,1126257414 --psdresample
--psd H-H1_LOSC_16_V2-1126257414-4096.gwf,L-L1_LOSC_16_V2-1126257414-4096.gwf --inj sample_inje
ction.inj --inj_id 1 --noise -1,0 --nsteps 5000 --nskip 100 --randomseed 20180209 --burnin 1000
00
#mpirun -np 1 $masterdir/bin/inferenceMCMC --approximant TaylorF2 --f_start 10.0 --ifo H1,L1,K1
 --trigtime 1126259462 --seglen 16.0 --strain H-H1_LOSC_16_V2-1126257414-4096.gwf,L-L1_LOSC_16_
V2-1126257414-4096.gwf,TaylorF2 --channel H1:LOSC-STRAIN,L1:LOSC-STRAIN,L1:LOSC-STRAIN --dumpti
me --dumpfreq --dumppsd --window hann,hann,hann --psdstart 1126257414,1126257414,1126257414 --p
sdresample --psd H-H1_LOSC_16_V2-1126257414-4096.gwf,L-L1_LOSC_16_V2-1126257414-4096.gwf,Design
bKAGRA-vrsed --inj sample_injection.inj --inj_id 1 --noise -1,0,3546579 --nsteps 5000 --nskip 1
00 --randomseed -1 --burnin 100000
```

Run Post process

- $cd test_run
- $./postprocess.sh

```
#!/bin/bash
masterdir=/home/hwlee/projects/KGL/kagali/build/mcmc
#masterdir=/home/hwlee/projects/KGL/kagali/build/mcmc_tmp
source $masterdir/etc/kagali-user-env.sh
export KGL_NOISEDIR=/home/hwlee/projects/KGL/kagali/test_run/noisedata
export KGL_INJECTIONDIR=/home/hwlee/projects/KGL/kagali/test_run/injection
export KGL_CACHEDIR=/home/hwlee/projects/KGL/kagali/frames
#export MALLOC_CHECK_=1
$masterdir/bin/postProc option.ini
```

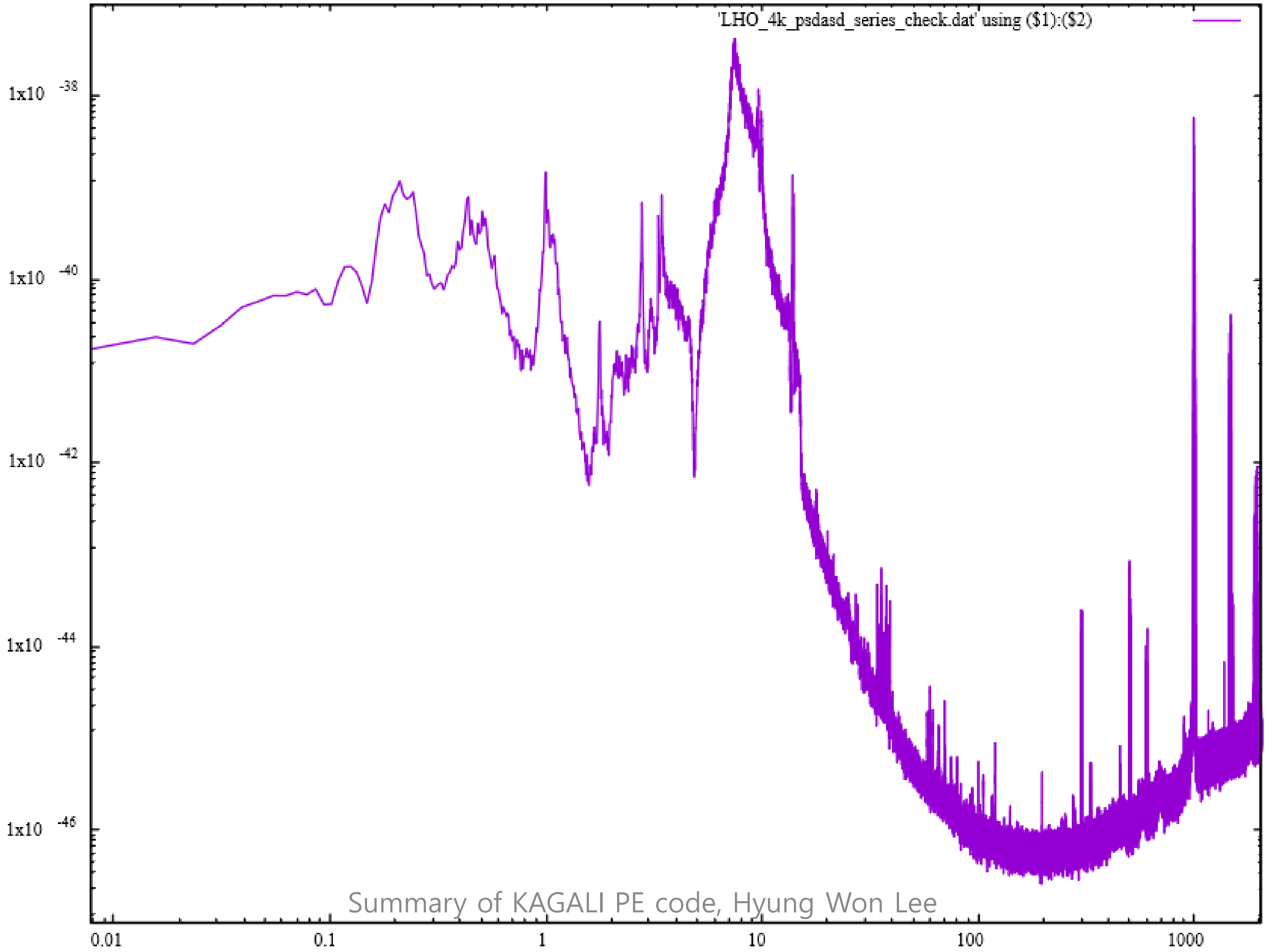
Option for post process

```
# This is a sampe option input file for KAGALI, created by hwlee
# sharp started line is a comment line
# after sharp in a line is comment
# blank line ignored
# [Section] line will be ignored
# table_name = TABLE_NAME should occur once in the file, if exist more accept only the first one
# each injection data start from injection_id = nnn til next injection start line
# injection file should be under directory KGL_INJECTIONDIR or current directory
[MCMC output file name]
outputFile = PTMCMC.output.20180209.out
[1D output]
1Doutput = all # save all variables
#1Doutput = m1,m2,mc,eta,distance,ra,dec,deltalogl

[Physical parameters]
m1 = 36  # in solar mass unit
m2 = 29  # in solar mass unit
spin1x = 0  # dimensionless spin value
spin1y = 0  # dimensionless spin value
spin1z = 0  # dimensionless spin value
spin2x = 0  # dimensionless spin value
spin2y = 0  # dimensionless spin value
spin2z = 0  # dimensionless spin value
```

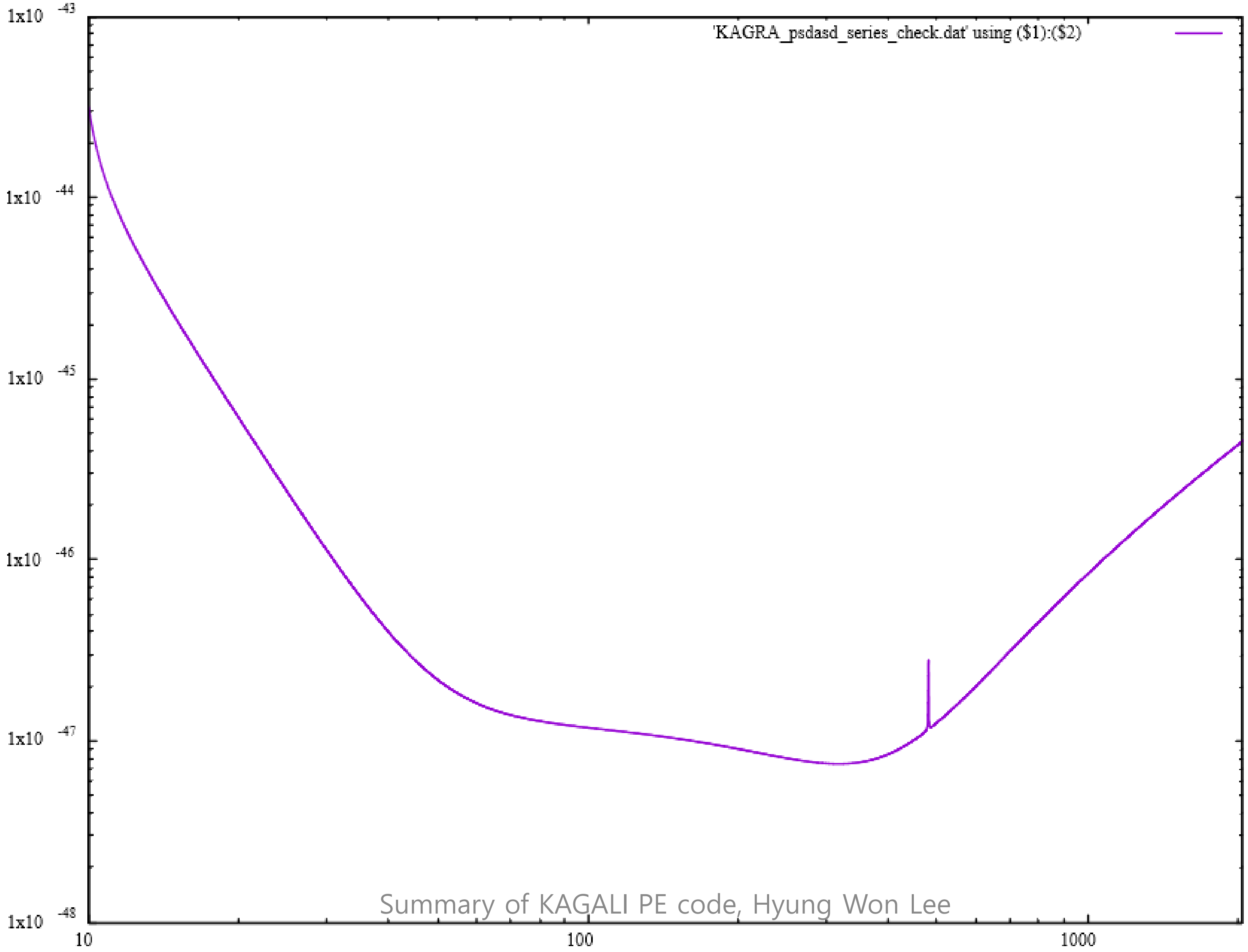
Some graphs(using gnuplot)

- PSDs



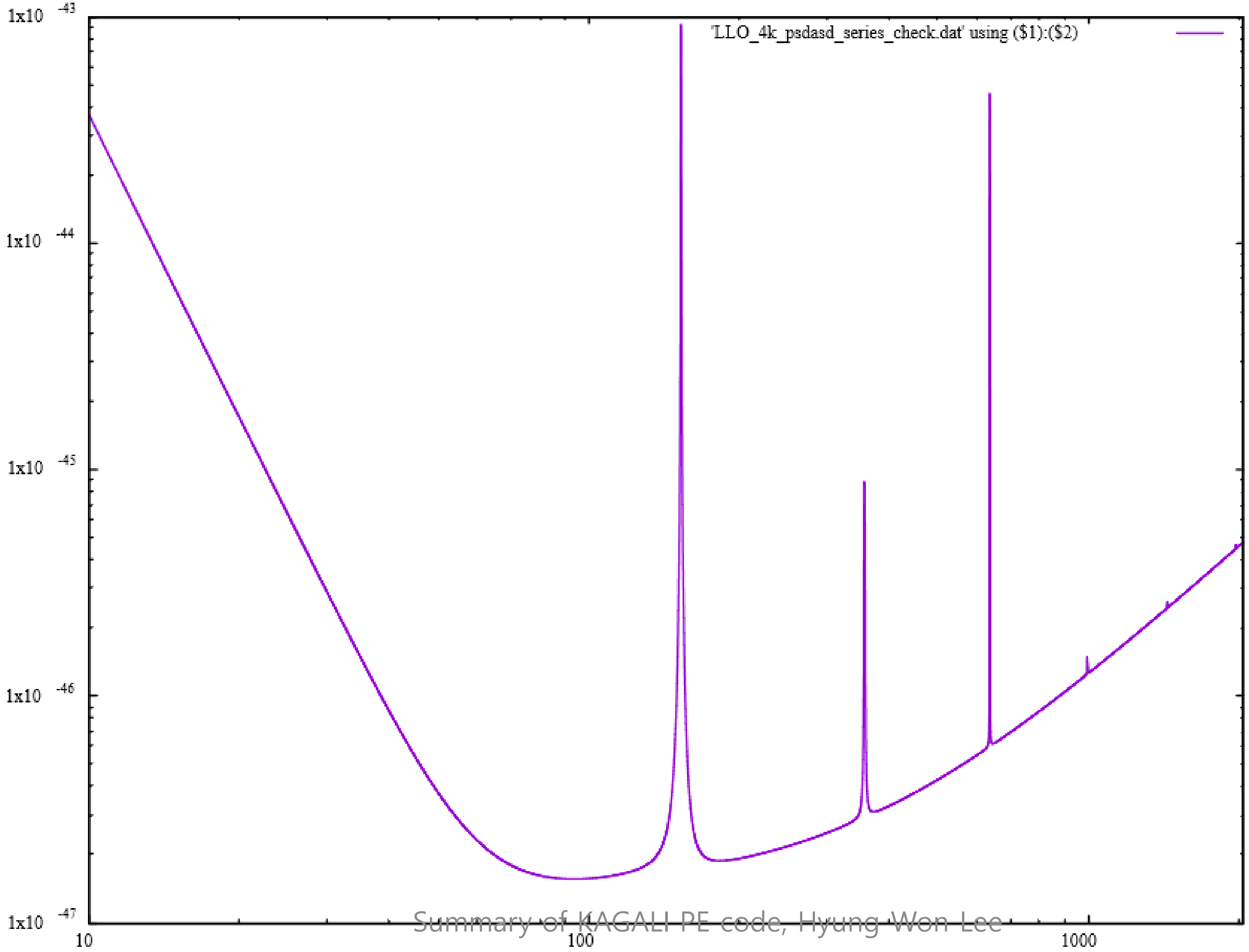
Some graphs(using gnuplot)

- PSDs



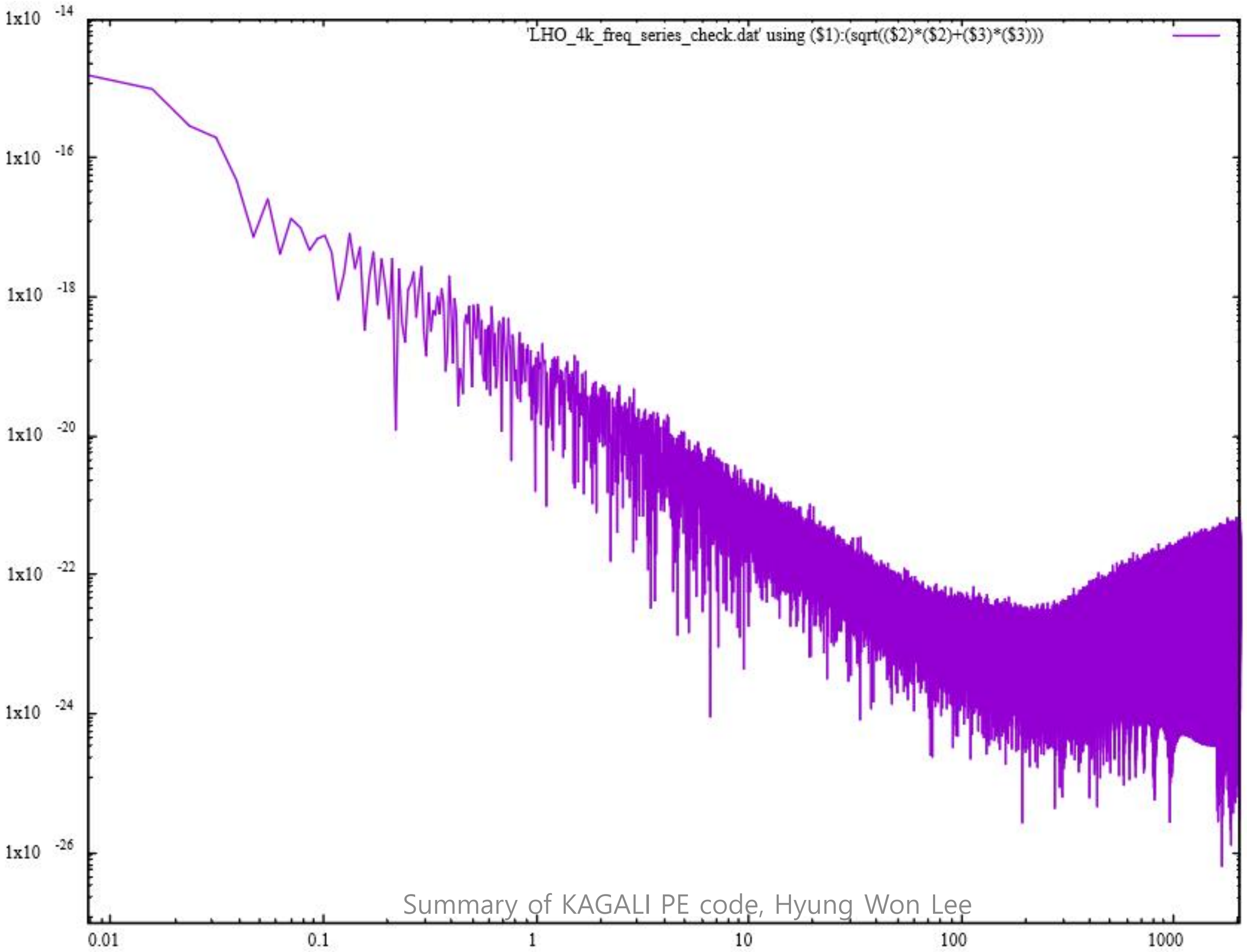
Some graphs(using gnuplot)

- PSDs



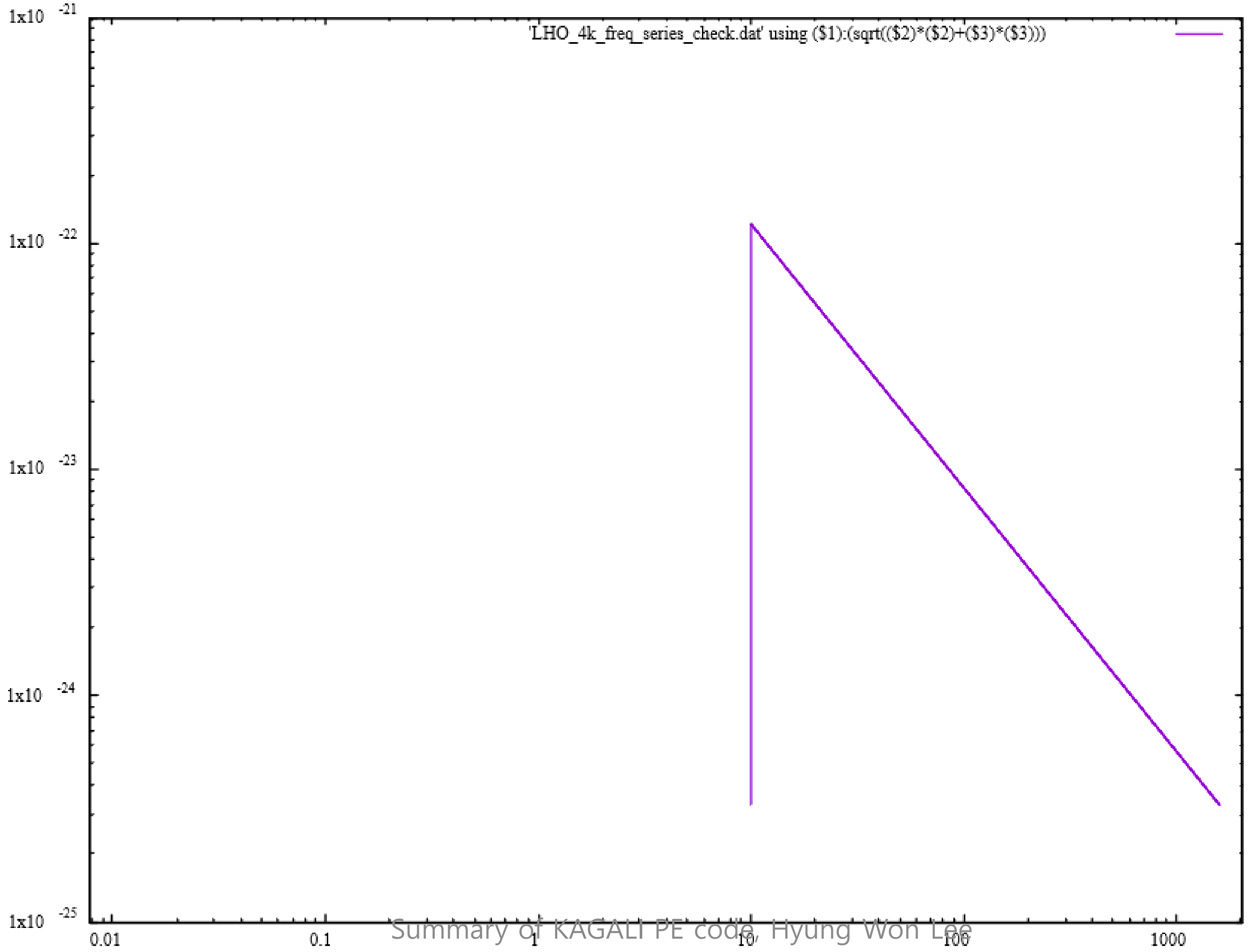
Some graphs(using gnuplot)

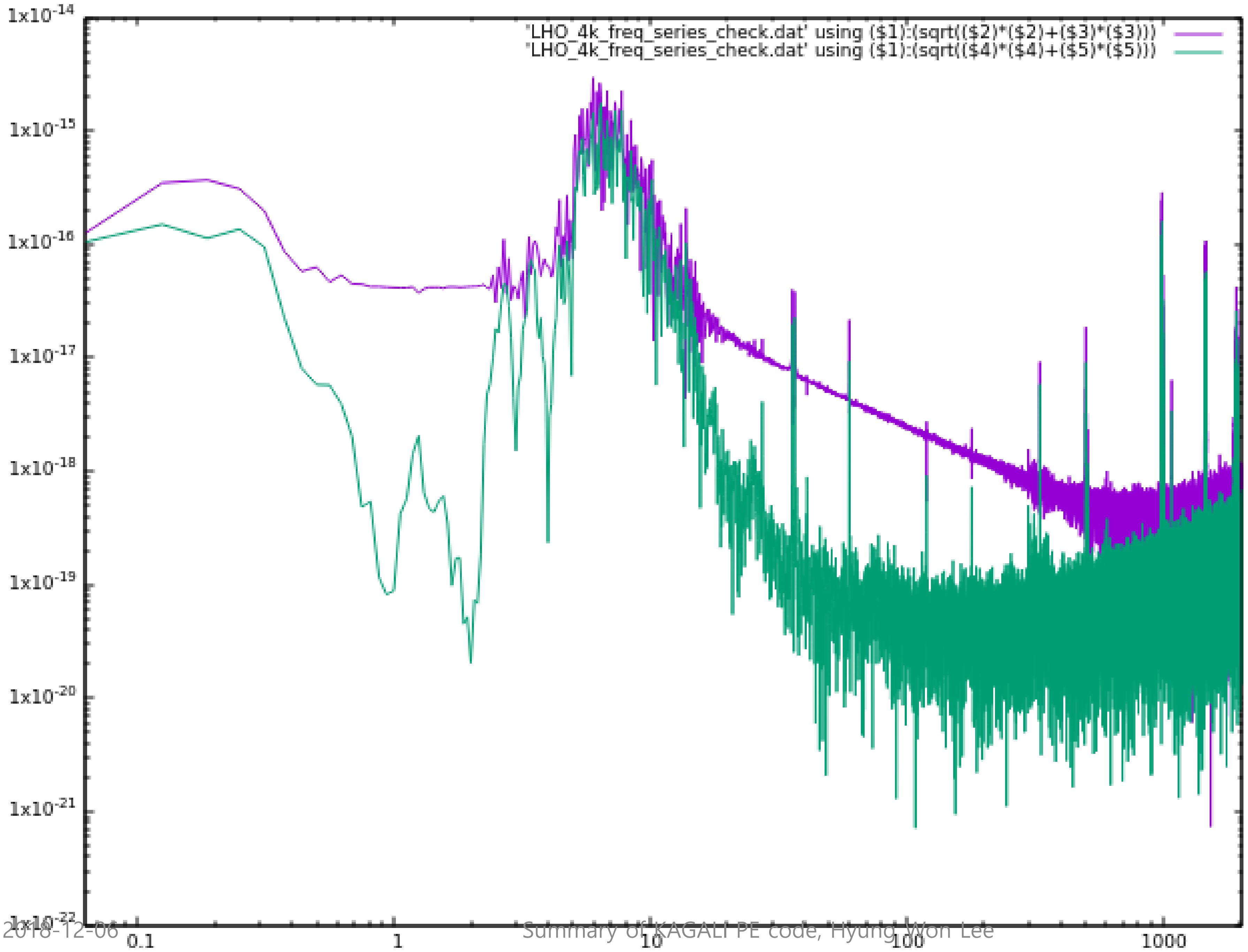
- PSDs

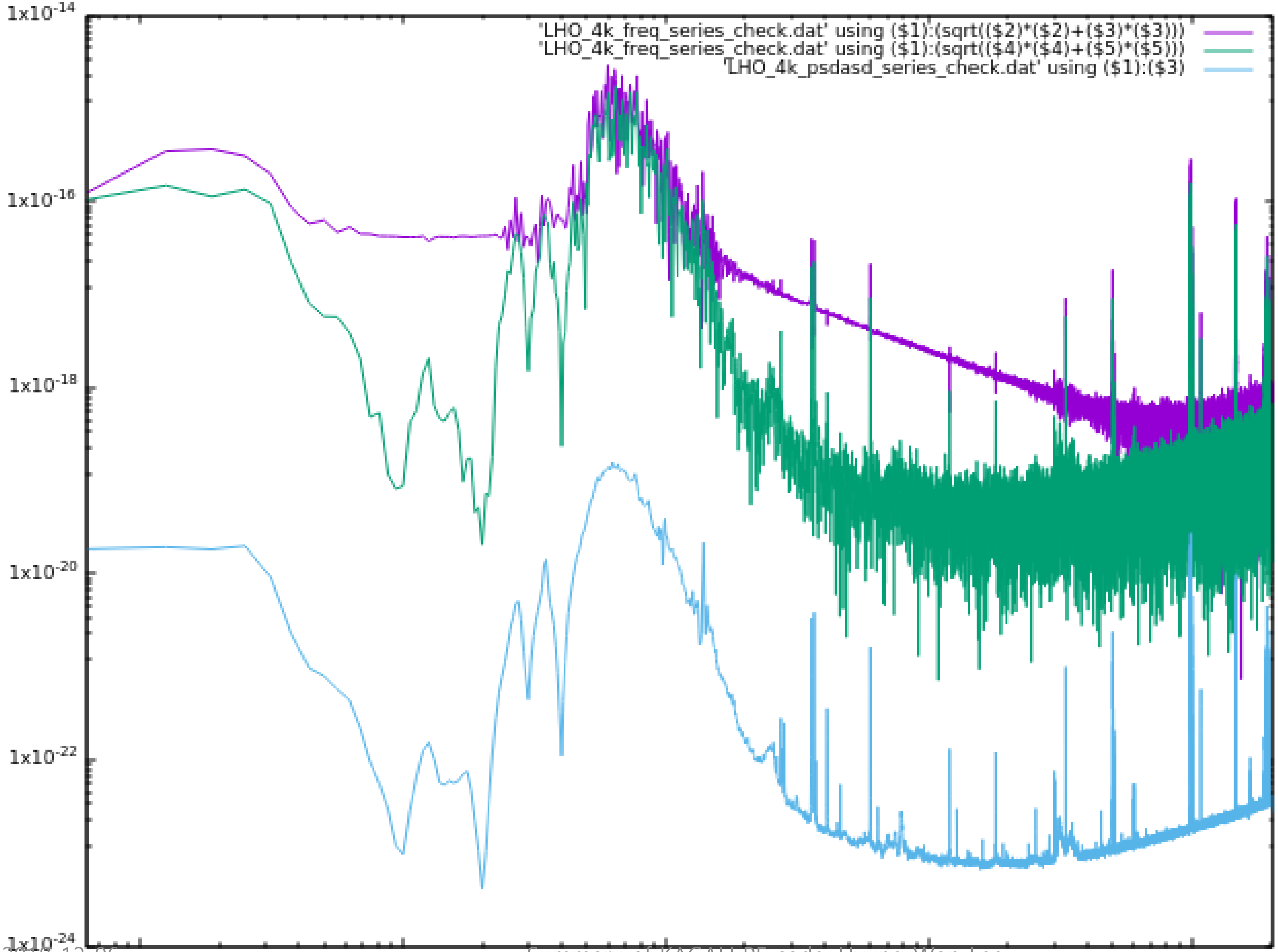


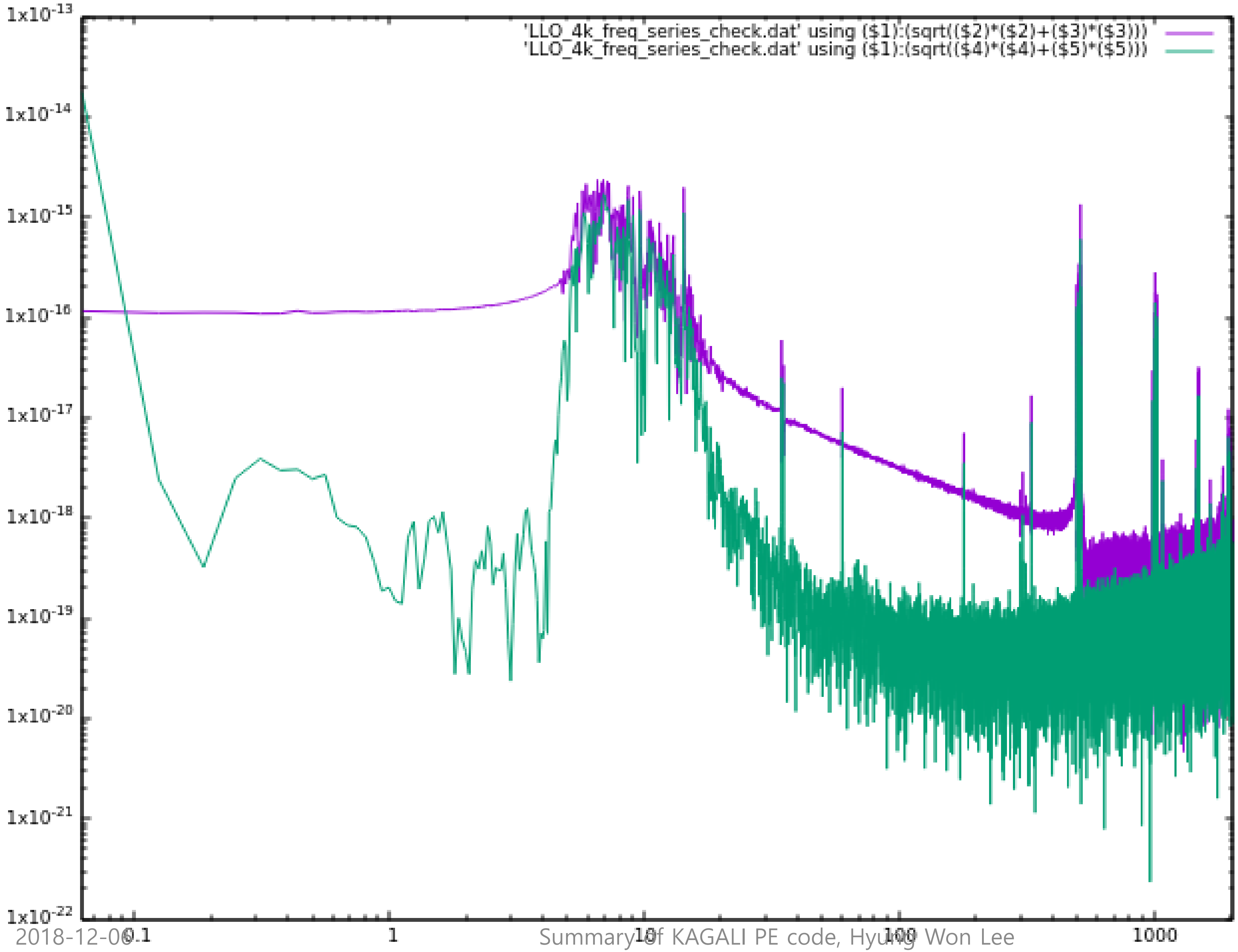
Some graphs(using gnuplot)

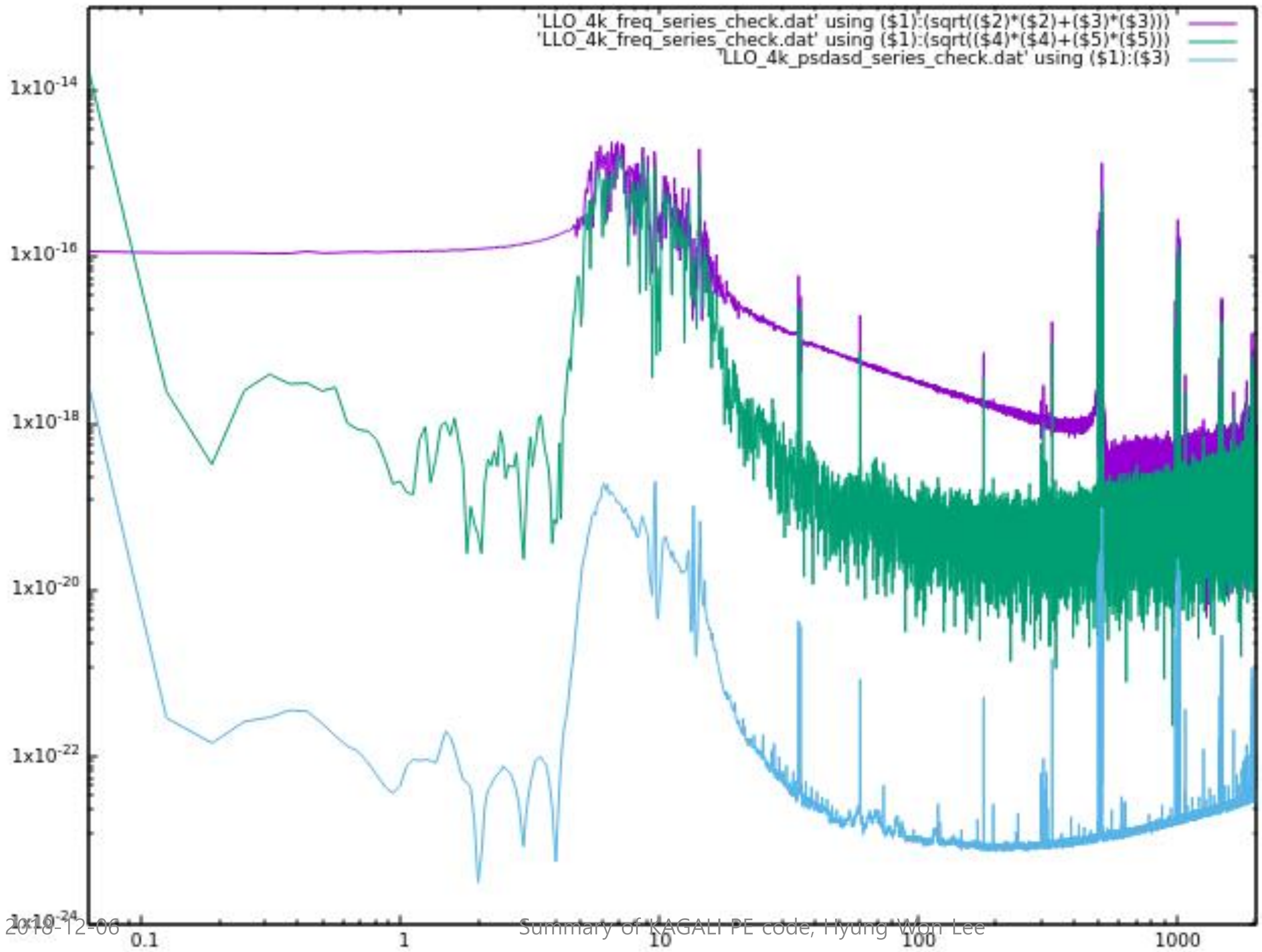
- PSDs




'LHO_4k_freq_series_check.dat' using ($1):(sqrt(($2)*($2)+($3)*($3)))
'LHO_4k_freq_series_check.dat' using ($1):(sqrt(($4)*($4)+($5)*($5)))
1x10-14
1x10-15
1x10-16
1x10-17
1x10-18
1x10-19
1x10-20
1x10-21
1x10-22
0.1
1
10
100
1000

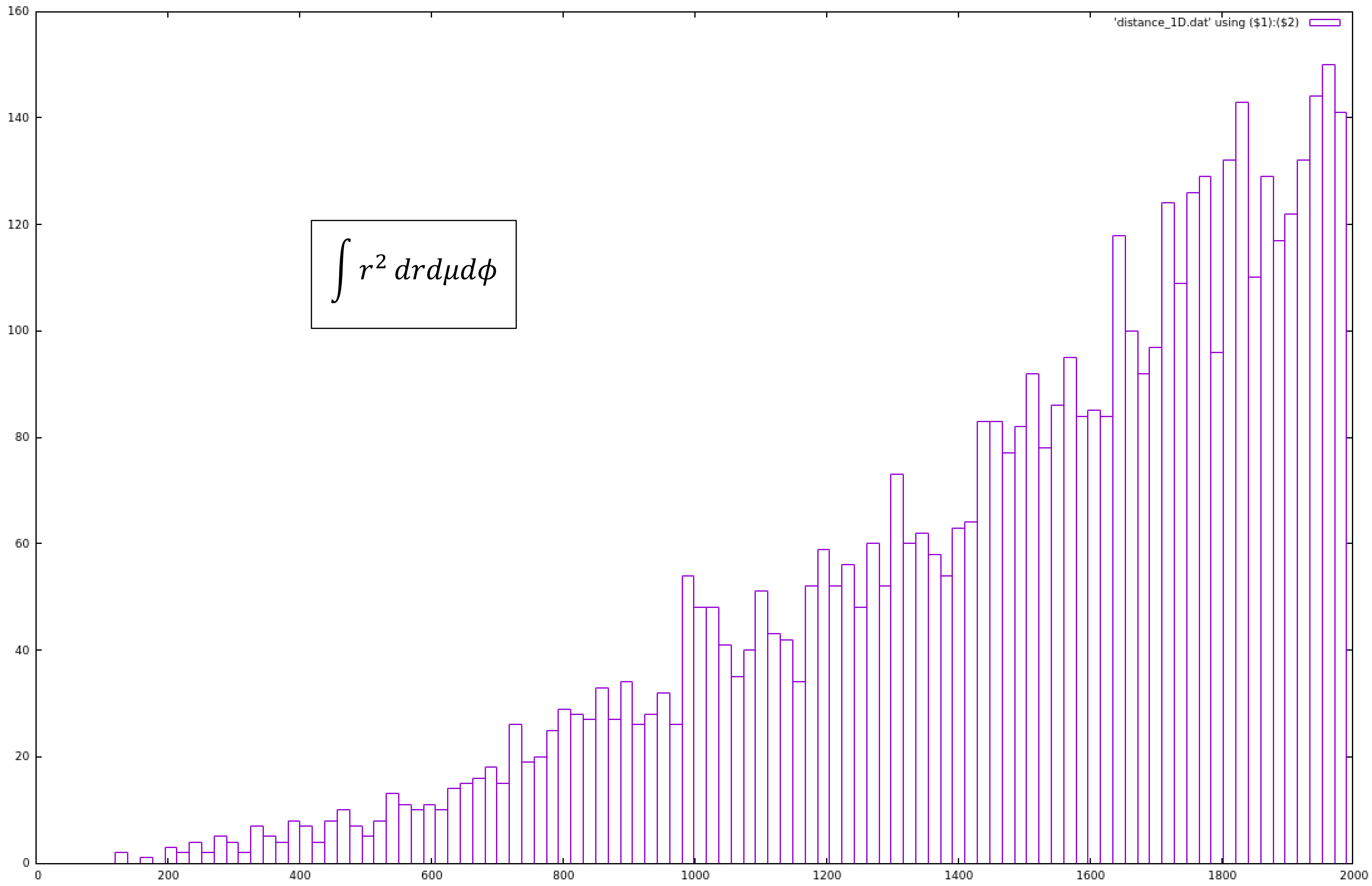

'LHO_4k_freq_series_check.dat' using ($1):(sqrt(($2)*($2)+($3)*($3)))
'LHO_4k_freq_series_check.dat' using ($1):(sqrt(($4)*($4)+($5)*($5)))
'LHO_4k_psdasd_series_check.dat' using ($1):($3)
1x10-14
1x10-16
1x10-18
1x10-20
1x10-22
1x10-24
0.1
1
10
100
1000


'LLO_4k_freq_series_check.dat' using ($1):(sqrt(($2)*($2)+($3)*($3)))
'LLO_4k_freq_series_check.dat' using ($1):(sqrt(($4)*($4)+($5)*($5)))
1x10-13
1x10-14
1x10-15
1x10-16
1x10-17
1x10-18
1x10-19
1x10-20
1x10-21
1x10-22
0.1
1
10
100
1000

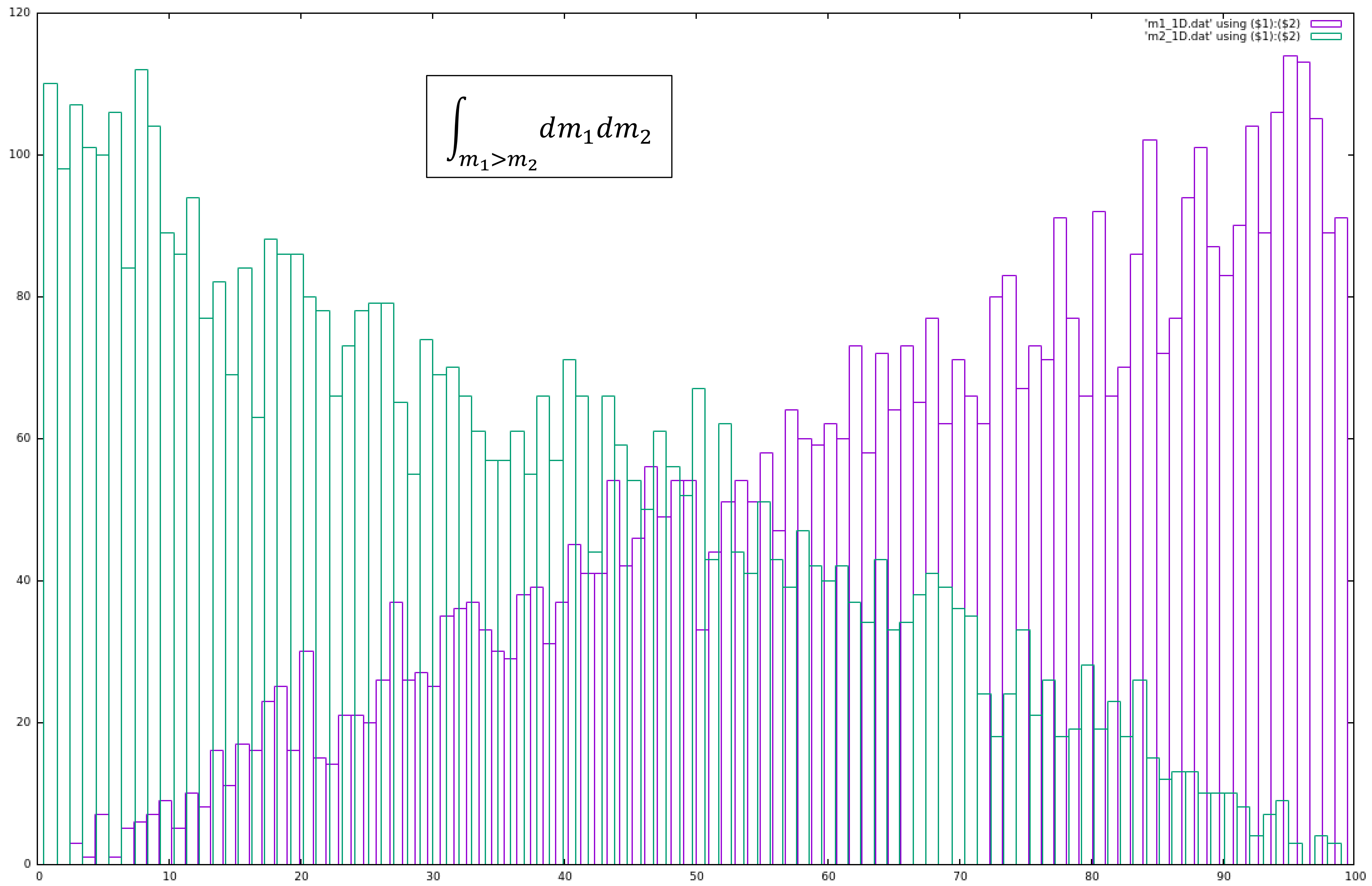

'LLO_4k_freq_series_check.dat' using ($1):(sqrt(($2)*($2)+($3)*($3)))
'LLO_4k_freq_series_check.dat' using ($1):(sqrt(($4)*($4)+($5)*($5)))
'LLO_4k_psdasd_series_check.dat' using ($1):($3)
1x10-14
1x10-16
1x10-18
1x10-20
1x10-22
1x10-24
0.1
1
10
100
1000

Outputs with zeroLike

- Likelihood function is analytical function returning zero.
- $\mathcal{L}(\vec{\theta}) = e^{-\frac{1}{2}\langle d - h(\vec{\theta}) | d - h(\vec{\theta}) \rangle} = 1$
- $d - h(\vec{\theta}) = 0$

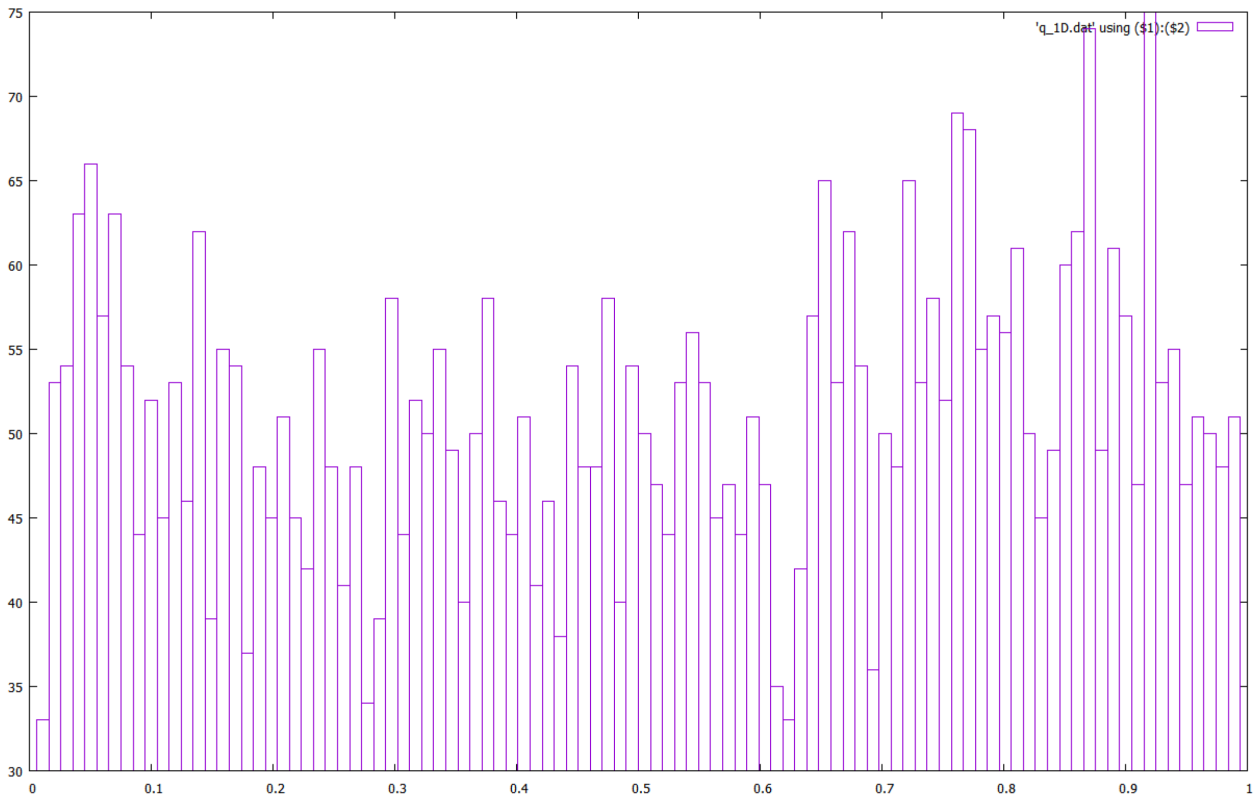


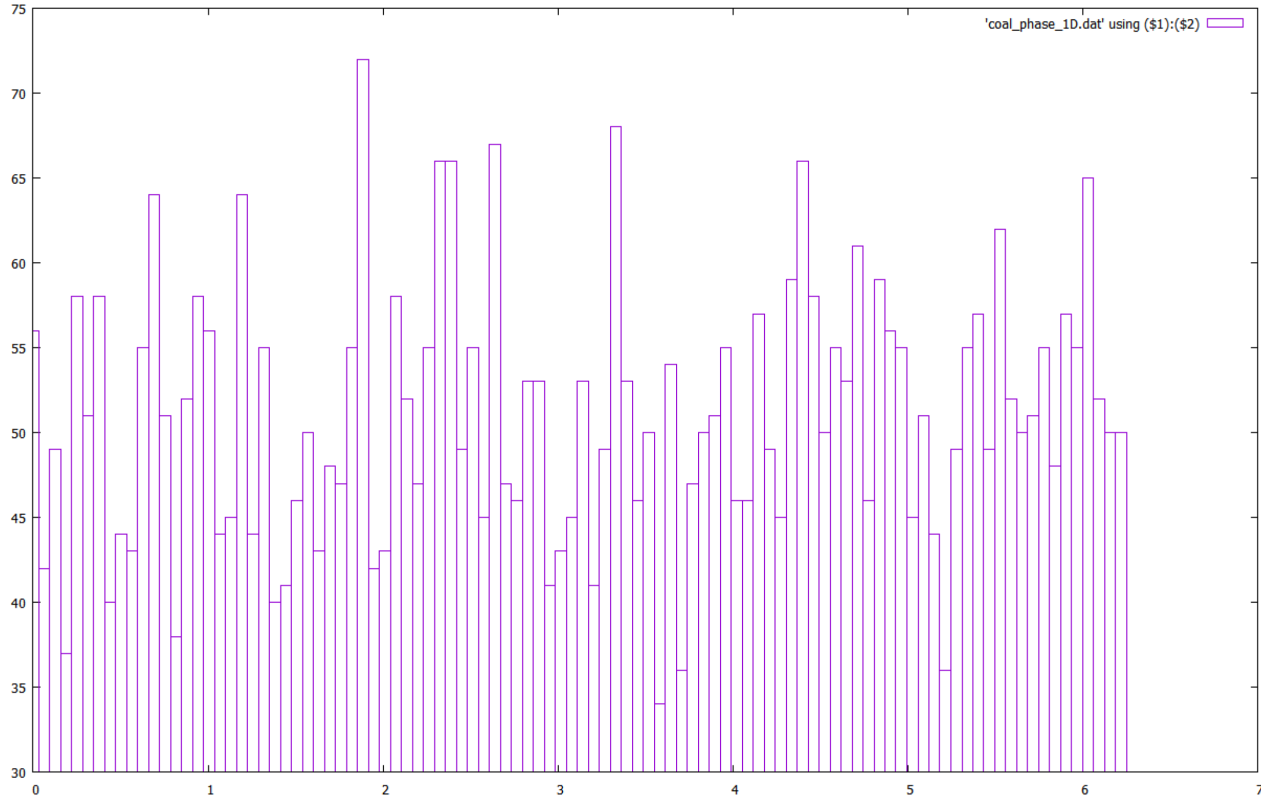
'distance_1D.dat' using ($1):($2)
$\int r^2\, dr d\mu d\phi$
0
20
40
60
80
100
120
140
160
0
200
400
600
800
1000
1200
1400
1600
1800
2000



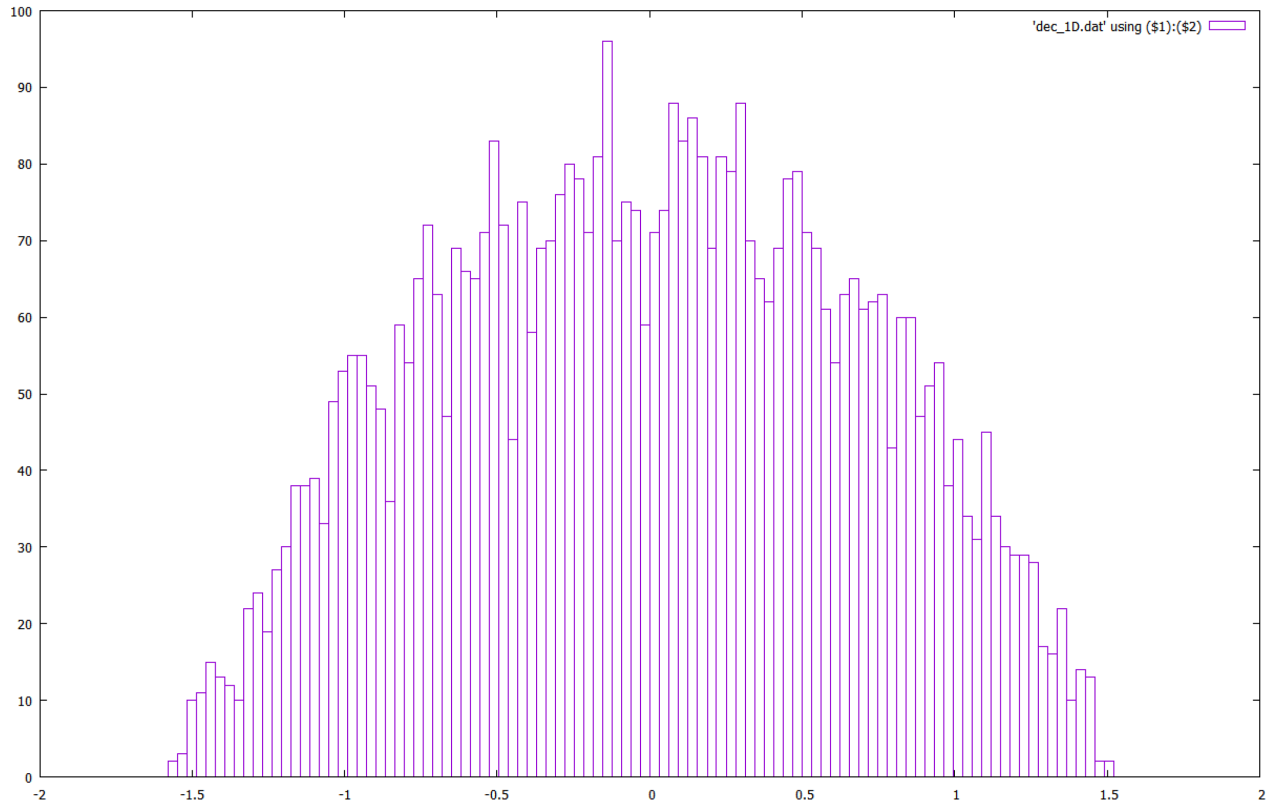
'm1_1D.dat' using ($1):($2)
'm2_1D.dat' using ($1):($2)
$$\int_{m_1>m_2} dm_1 dm_2$$
0
20
40
60
80
100
120
0
10
20
30
40
50
60
70
80
90
100

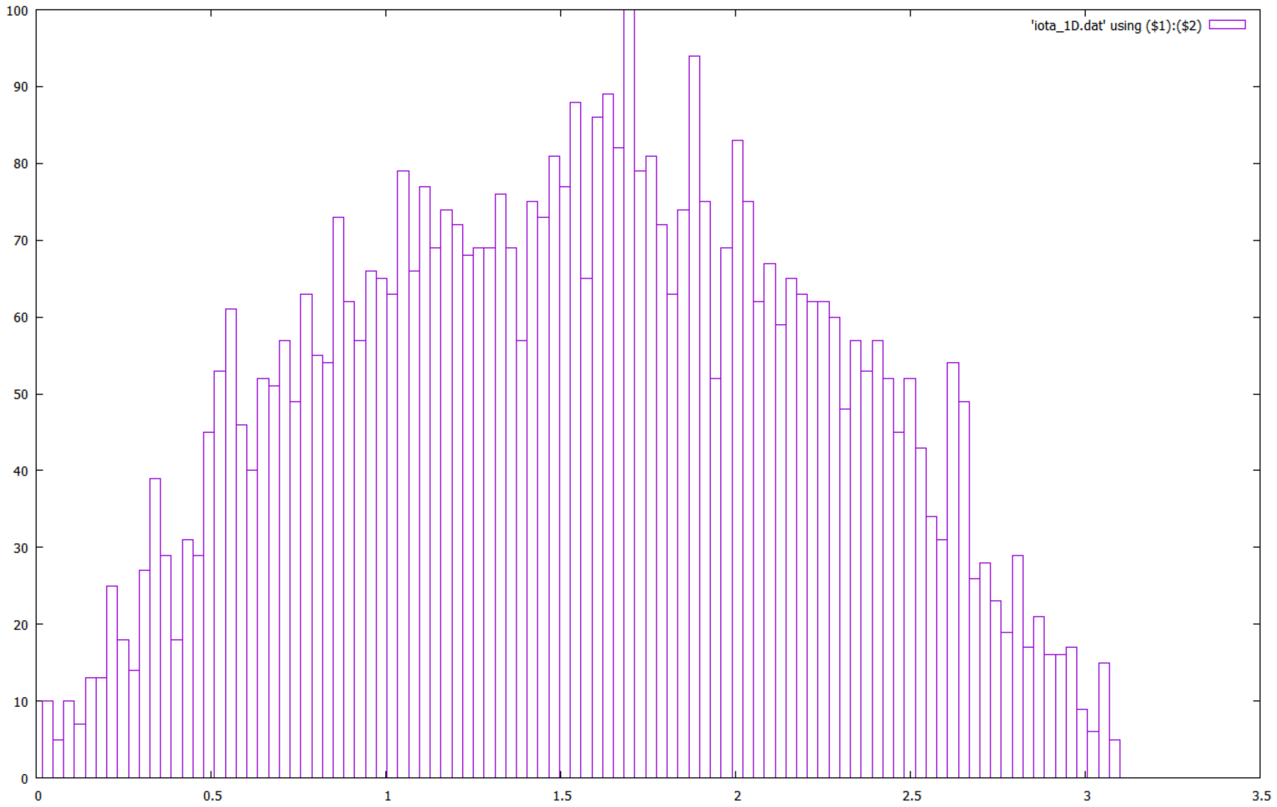

'mc_1D.dat' using ($1):($2)


'q_1D.dat' using ($1):($2)


'coal_phase_1D.dat' using ($1):($2)


'costheta_jn_1D.dat' using ($1):($2)


'dec_1D.dat' using ($1):($2)
100
90
80
70
60
50
40
30
20
10
0
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


'iota_1D.dat' using ($1):($2)
100
90
80
70
60
50
40
30
20
10
0
0
0.5
1
1.5
2
2.5
3
3.5

Issues

- Dead lock for parallel tempering -> resolved
- zeroLogLikelihood seems to work correctly
- Real likelihood seems not to work correctly

- Debugging is necessary

Future works

- Debugging to get correct result
- Implement features
  - Malmquist correction (on going)
  - ACL calculation and check convergence (on going)
  - Check swap operation (implemented)
- Plan
  1. Check time evolution
  2. Injection study test
  3. Comparison to lalsuite result