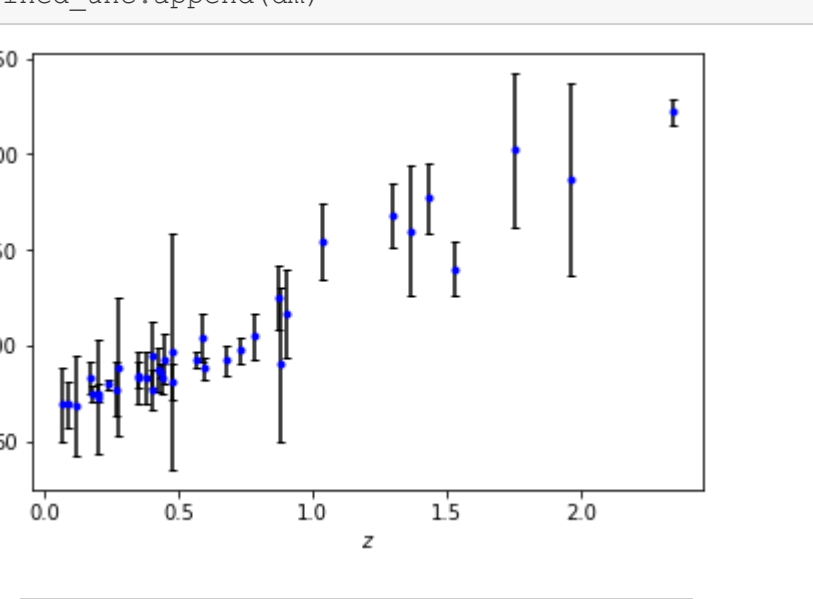
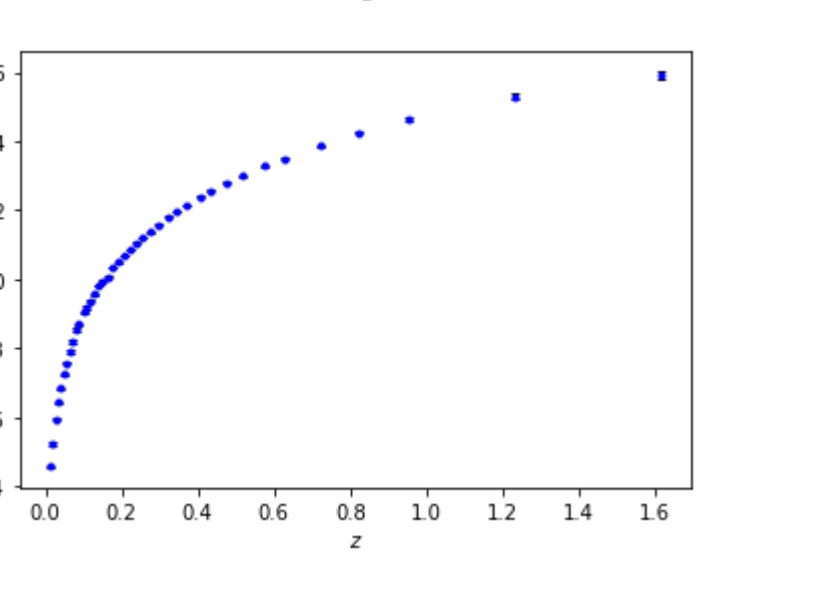
```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        from time import time
        import pandas as pd
        from scipy.constants import c
        from scipy.integrate import quad
        import scipy.stats

        #Nested sampling package
        import ultranest
        import corner
```

I. Import data

```
In [2]: # Import Hubble H(z) data

        hubble_data = pd.read_csv('hubble_data.csv', header=0)

        z_H = np.array(hubble_data['z'])
        H = np.array(hubble_data['H'])
        dH = np.array(hubble_data['dH'])

        plt.figure()
        plt.errorbar(z_H, H, yerr=dH, marker = '.', color='blue', ecolor='black', capsize=2, ls='none')
        plt.ylabel(r'$H(z)$')
        plt.xlabel(r'$z$')
        plt.show()

        # Import apparent magnitude m(z) data
        m_data = pd.read_csv('m_data.txt', sep = ' ', header = 0)
        m_sys_unc = pd.read_csv('m_sys_unc.txt', sep = ' ', header = 0)
        m_sys_unc = np.array(m_sys_unc['40']).reshape(40, 40)

        tot = m_sys_unc + np.diag(m_data['dmb']**2)

        z_m = np.array(m_data['zcmb'])
        m = np.array(m_data['mb'])
        dm = np.sqrt(np.diag(tot))

        plt.figure()
        plt.errorbar(z_m, m, yerr=dm, marker = '.', color='blue', ecolor='black', capsize=2, ls='none')
        plt.ylabel(r'$m(z)$')
        plt.xlabel(r'$z$')
        plt.show()

        # Combine redshifts for likelihood computation later
        combined_z = []
        combined_z.append(z_H)
        combined_z.append(z_m)

        # Combine data
        combined_data = []
        combined_data.append(H)
        combined_data.append(m)

        # Combine uncertainties
        combined_unc = []
        combined_unc.append(dH)
        combined_unc.append(dm)
```

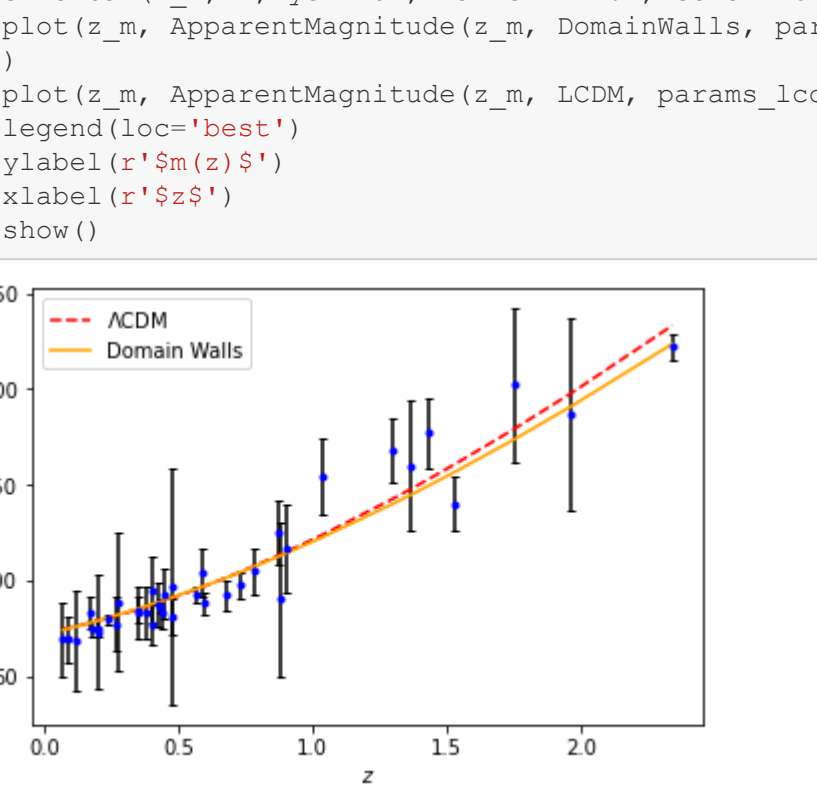


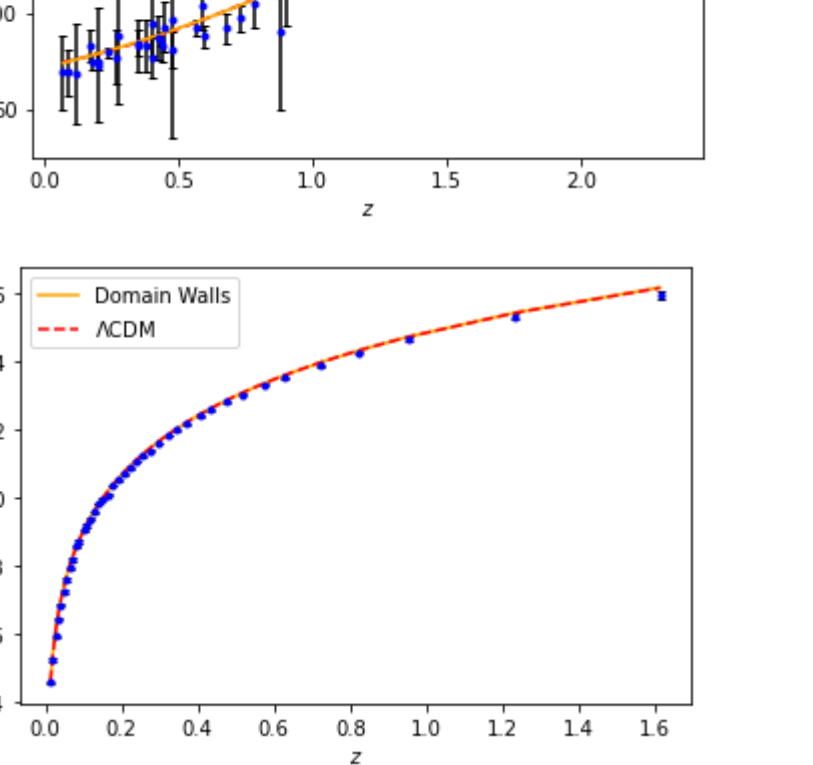


II. Define models

```
In [3]: 'Define LCDM Hubble model'
        def LCDM(z, params):
            H0 = params[0]
            OM = params[1]
            OL = params[2]

            return H0*np.sqrt(OM*(1+z)**3 + OL)

        'Define Domain Walls Model'
        def DomainWalls(z, params):
            H0 = params[0]
            OM = params[1]
            OD = params[2]

            return H0*np.sqrt(OM*(1+z)**3 + OD*(1+z)**(1/3))

        'Define apparent magnitude function'
        def ApparentMagnitude(z, Hubble, params):

            def integrand_dl(z, Hubble, params):
            #integrand of luminosity distance formula
                return params[0]/Hubble(z, params)

            def dl(z, Hubble, params):
            #dimensionless luminosity distance at redshift z (input array)
                rz_array = np.zeros(len(z))
                for i in np.arange(len(z)):
                    rz_each = quad( integrand_dl, 0, z[i],
                                   args = (Hubble, params) )[0]
                    rz_array[i] = rz_each

                return (1+z)*rz_array

            return 5*np.log10((c*100/params[0])*dl(z, Hubble, params)) - 19.25
```

```
In [4]: params_lcdm = [72.21, 0.26, 0.74]
        params_domainwalls = [73.37, 0.22, 0.73]

        plt.figure()
        plt.errorbar(z_H, H, yerr=dH, marker = '.', color='blue', ecolor='black', capsize=2, ls='none')
        plt.plot(z_H, LCDM(z_H, params_lcdm), color='red', ls='--', label=r'$\Lambda$CDM')
        plt.plot(z_H, DomainWalls(z_H, params_domainwalls), color='orange', label='Domain Walls')
        plt.legend(loc='best')
        plt.ylabel(r'$H(z)$')
        plt.xlabel(r'$z$')
        plt.show()

        plt.figure()
        plt.errorbar(z_m, m, yerr=dm, marker = '.', color='blue', ecolor='black', capsize=2, ls='none')
        plt.plot(z_m, ApparentMagnitude(z_m, DomainWalls, params_domainwalls), color='orange', label='Domain Wa
        lls')
        plt.plot(z_m, ApparentMagnitude(z_m, LCDM, params_lcdm), color='red', ls='--', label=r'$\Lambda$CDM')
        plt.legend(loc='best')
        plt.ylabel(r'$m(z)$')
        plt.xlabel(r'$z$')
        plt.show()
```





III. Nested Sampling: LCDM

```
In [30]: def Prior_LCDM(cube):

             #H0 Prior: [0,100]
             H0_min = 0
             H0_max = 100

             #OM Prior: [0,1]
             OM_min = 0
             OM_max = 1

             #OL Prior: [0,1]
             OL_min = 0
             OL_max = 1

             #M Prior: [0,1]
             #M_min = -21
             #M_max = -18

             #Extract values
             H0prime = cube[0]
             OMprime = cube[1]
             OLprime = cube[2]
             #Mprime = cube[3]

             H0 = H0prime*(H0_max-H0_min) + H0_min
             OM = OMprime*(OM_max-OM_min) + OM_min
             OL = OLprime*(OL_max-OL_min) + OL_min
             #M = Mprime*(M_max-M_min) + M_min

             return np.array([H0, OM, OL])

         def LogLikelihood_LCDM(params):

         #     if params[1] + params[2] > 1:
         #         return -10**3


             # calculate the model
             hubble_model = LCDM(z_H, params)
             apparent_magnitude_model = ApparentMagnitude(z_m, LCDM, params)

             #calculate the likelihood
             residual_H = H - hubble_model
             residual_m = m - apparent_magnitude_model
             sig_H = 1/dH
             sig_m = 1/dm

             lnL_H = -0.5*np.sum((residual_H*sig_H)**2)
             lnL_m = -0.5*np.sum((residual_m*sig_m)**2)

             return  lnL_H + lnL_m
```

```
In [53]: t_i = time()
         sampler_LCDM = ultranest.ReactiveNestedSampler(['H0', 'OM', 'OL'], LogLikelihood_LCDM, Prior_LCDM)
         result_LCDM = sampler_LCDM.run()
         sampler_LCDM.print_results()
         t_f = time()

         print('Sampling time: {} s'.format(t_f-t_i))
```

```
[ultranest] Sampling 400 live points from prior ...


[ultranest] Explored until L=-4e+01   [-37.4485..-37.4484]*| it/evals=5696/136419 eff=4.1877% N=400
       00 0
[ultranest] Likelihood function evaluations: 136419
[ultranest]   logZ = -47.09 +- 0.09403
[ultranest] Effective samples strategy satisfied (ESS = 1651.8, need >400)
[ultranest] Posterior uncertainty strategy is satisfied (KL: 0.47+-0.07 nat, need <0.50 nat)
[ultranest] Evidency uncertainty strategy is satisfied (dlogz=0.21, need <0.5)
[ultranest]   logZ error budget: single: 0.15 bs:0.09 tail:0.01 total:0.09 required:<0.50
[ultranest] done iterating.

logZ = -47.079 +- 0.206
  single instance: logZ = -47.079 +- 0.147
  bootstrapped   : logZ = -47.093 +- 0.206
  tail           : logZ = +- 0.010
insert order U test : converged: True correlation: inf iterations

    H0                  76.6 +- 9.8
    OM                  0.244 +- 0.058
    OL                  0.73 +- 0.17
Sampling time: 624.2965879440308 s
```

```
In [54]: points_LCDM = np.array(result_LCDM["weighted_samples"]["points"])
         weights_LCDM = np.array(result_LCDM["weighted_samples"]["weights"])
         scaledweights_LCDM = weights_LCDM / weights_LCDM.max()
         mask_LCDM = np.random.rand(len(scaledweights_LCDM)) < scaledweights_LCDM

         samples_LCDM = points_LCDM[mask_LCDM, :]
```

```
In [88]: # Open saved samples
         def open_samples(file):

             chains = pd.read_csv(file)
             nparams = len(chains.columns)

             if nparams == 2:
                 samples = np.vstack((np.array(chains.iloc[:, 0]), np.array(chains.iloc[:, 1]))).T
             if nparams == 3:
                 samples = np.vstack((np.array(chains.iloc[:, 0]), np.array(chains.iloc[:, 1]), np.array(chains.
         iloc[:, 2]))).T
             if nparams == 4:
                 samples = np.vstack((np.array(chains.iloc[:, 0]), np.array(chains.iloc[:, 1]), np.array(chains.
         iloc[:, 2]), np.array(chains.iloc[:, 3]))).T

             return samples

         # Extract and compile chains
         #lcdm_chain_1 = samples_LCDM[:,0] # extract chain of H0 values
         #lcdm_chain_2 = samples_LCDM[:,1] # extract chain if OM values
         #lcdm_chain_3 = samples_LCDM[:,2] # extract chain if OL values
         #lcdm_samples = np.vstack((lcdm_chain_1, lcdm_chain_2, lcdm_chain_3)).T

         # Save chains and evidence (do not forget)
         #np.savetxt('lcdm_chains_Aug11_713PM.csv', lcdm_samples, delimiter=",")
         #np.savetxt('lcdm_Z_Aug11_713PM.csv', [[result_LCDM['logz'] , result_LCDM['logzerr']]])

         # Open saved chains
         #samples_open =  open_samples('lcdm_chains_Aug11_713PM.csv')

         #fig = corner.corner(samples_open, labels=[r"$H_0$", r"$\Omega_m$", r"$\Omega_\Lambda$"],color='blue',
          label_kwargs={"fontsize": 20},quantiles=[0.16, 0.5, 0.84],
         #                        show_titles=True, title_kwargs={"fontsize": 20})
         #fig.savefig('lcdm_Aug11_713PM.pdf')
```

IV. Nested Sampling: Domain Walls

```
In [58]: def Prior_DW(cube):

             #H0 Prior: [0,100]
             H0_min = 0
             H0_max = 100

             #OM Prior: [0,1]
             OM_min = 0
             OM_max = 1

             #OL Prior: [0,1]
             OD_min = 0
             OD_max = 1

             #Extract values
             H0prime = cube[0]
             OMprime = cube[1]
             ODprime = cube[2]

             H0 = H0prime*(H0_max-H0_min) + H0_min
             OM = OMprime*(OM_max-OM_min) + OM_min
             OD = ODprime*(OD_max-OD_min) + OD_min

             return np.array([H0, OM, OD])

         def LogLikelihood_DW(params):


             # calculate the model
             hubble_model = DomainWalls(z_H, params)
             apparent_magnitude_model = ApparentMagnitude(z_m, DomainWalls, params)

             #calculate the likelihood
             residual_H = H - hubble_model
             residual_m = m - apparent_magnitude_model
             sig_H = 1/dH
             sig_m = 1/dm

             lnL_H = -0.5*np.sum((residual_H*sig_H)**2)
             lnL_m = -0.5*np.sum((residual_m*sig_m)**2)

             return  lnL_H + lnL_m
```

```
In [59]: t_i = time()
         sampler_DW = ultranest.ReactiveNestedSampler(['H0', 'OM', 'OL'], LogLikelihood_DW, Prior_DW)
         result_DW = sampler_DW.run()
         sampler_DW.print_results()
         t_f = time()

         print('Sampling time: {} s'.format(t_f-t_i))
```

```
[ultranest] Sampling 400 live points from prior ...


[ultranest] Explored until L=-4e+01   [-36.6163..-36.6162]*| it/evals=5758/123867 eff=4.6636% N=400
       00  0
[ultranest] Likelihood function evaluations: 123867
[ultranest]   logZ = -46.41 +- 0.1208
[ultranest] Effective samples strategy satisfied (ESS = 1587.1, need >400)
[ultranest] Posterior uncertainty strategy is satisfied (KL: 0.46+-0.04 nat, need <0.50 nat)
[ultranest] Evidency uncertainty strategy is satisfied (dlogz=0.26, need <0.5)
[ultranest]   logZ error budget: single: 0.15 bs:0.12 tail:0.01 total:0.12 required:<0.50
[ultranest] done iterating.

logZ = -46.402 +- 0.175
  single instance: logZ = -46.402 +- 0.148
  bootstrapped   : logZ = -46.409 +- 0.175
  tail           : logZ = +- 0.010
insert order U test : converged: True correlation: inf iterations

    H0                  77.1 +- 9.5
    OM                  0.209 +- 0.048
    OL                  0.73 +- 0.17
Sampling time: 633.0258281230927 s
```

```
In [60]: points_DW = np.array(result_DW["weighted_samples"]["points"])
         weights_DW = np.array(result_DW["weighted_samples"]["weights"])
         scaledweights_DW = weights_DW / weights_DW.max()
         mask_DW = np.random.rand(len(scaledweights_DW)) < scaledweights_DW

         samples_DW = points_DW[mask_DW, :]
```

```
In [90]: # Extract and compile chains
         #DomainWalls_chain_1 = samples_DW[:,0] # extract chain of H0 values
         #DomainWalls_chain_2 = samples_DW[:,1] # extract chain if OM values
         #DomainWalls_chain_3 = samples_DW[:,2] # extract chain if OL values
         #DomainWalls_samples = np.vstack((DomainWalls_chain_1, DomainWalls_chain_2, DomainWalls_chain_3)).T

         # Save chains and evidence (do not forget)
         #np.savetxt('domainwalls_chains_Aug11_728PM.csv', DomainWalls_samples, delimiter=",")
         #np.savetxt('domainwalls_Z_Aug11_728PM.csv', [[result_DW['logz'] , result_DW['logzerr']]])

         # Open saved chains
         samples_open =  open_samples('domainwalls_chains_Aug11_728PM.csv')


         fig = corner.corner(samples_open, labels=[r"$H_0$", r"$\Omega_m$", r"$\Omega_d$"],color='blue', label_k
         wargs={"fontsize": 20},quantiles=[0.16, 0.5, 0.84],
                                 show_titles=True, title_kwargs={"fontsize": 20})
         fig.savefig('domainwalls_Aug11_728PM.pdf')
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-90-717ca09a1271> in <module>
     10
     11 # Open saved chains
---> 12 samples_open =  open_samples('domainwalls_chains_Aug11_728PM.csv')
     13
     14

<ipython-input-88-dfbd5e61dffa> in open_samples(file)
      2 def open_samples(file):
      3
----> 4     chains = pd.read_csv(file)
      5     nparams = len(chains.columns)
      6

/opt/anaconda3/lib/python3.8/site-packages/pandas/io/parsers.py in read_csv(filepath_or_buffer, sep,
 delimiter, header, names, index_col, usecols, squeeze, prefix, mangle_dupe_cols, dtype, engine, conv
erters, true_values, false_values, skipinitialspace, skiprows, skipfooter, nrows, na_values, keep_def
ault_na, na_filter, verbose, skip_blank_lines, parse_dates, infer_datetime_format, keep_date_col, dat
e_parser, dayfirst, cache_dates, iterator, chunksize, compression, thousands, decimal, lineterminato
r, quotechar, quoting, doublequote, escapechar, comment, encoding, dialect, error_bad_lines, warn_bad
_lines, delim_whitespace, low_memory, memory_map, float_precision)
    684     )
    685
--> 686     return _read(filepath_or_buffer, kwds)
    687
    688

/opt/anaconda3/lib/python3.8/site-packages/pandas/io/parsers.py in _read(filepath_or_buffer, kwds)
    450
    451     # Create the parser.
--> 452     parser = TextFileReader(fp_or_buf, **kwds)
    453
    454     if chunksize or iterator:

/opt/anaconda3/lib/python3.8/site-packages/pandas/io/parsers.py in __init__(self, f, engine, **kwds)
    944             self.options["has_index_names"] = kwds["has_index_names"]
    945
--> 946         self._make_engine(self.engine)
    947
    948     def close(self):

/opt/anaconda3/lib/python3.8/site-packages/pandas/io/parsers.py in _make_engine(self, engine)
   1176     def _make_engine(self, engine="c"):
   1177         if engine == "c":
-> 1178             self._engine = CParserWrapper(self.f, **self.options)
   1179         else:
   1180             if engine == "python":

/opt/anaconda3/lib/python3.8/site-packages/pandas/io/parsers.py in __init__(self, src, **kwds)
   2006         kwds["usecols"] = self.usecols
   2007
-> 2008         self._reader = parsers.TextReader(src, **kwds)
   2009         self.unnamed_cols = self._reader.unnamed_cols
   2010

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader.__cinit__()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._setup_parser_source()

FileNotFoundError: [Errno 2] No such file or directory: 'domainwalls_chains_Aug11_728PM.csv'
```

V. Nested Sampling: LCDM with smaller unc

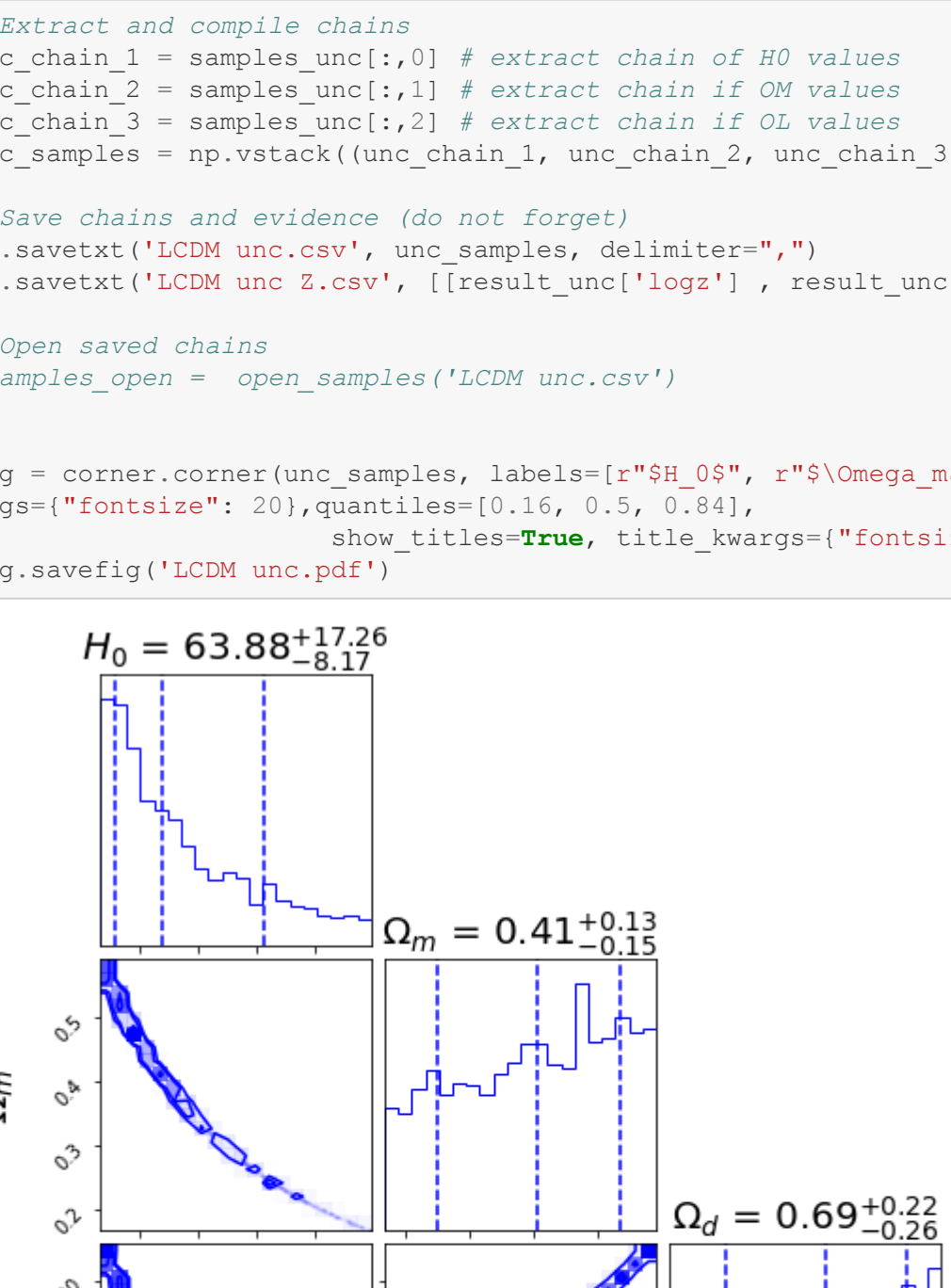
```
In [73]: dH_mock = np.array([np.random.normal(loc = 0, scale = 5) for i in range(len(H))])

         def LogLikelihood_LCDM_unc(params):


             # calculate the model
             hubble_model = LCDM(z_H, params)
             apparent_magnitude_model = ApparentMagnitude(z_m, LCDM, params)

             #calculate the likelihood
             residual_H = H - hubble_model
             residual_m = m - apparent_magnitude_model
             sig_H = 1/dH_mock
             sig_m = 1/dm

             lnL_H = -0.5*np.sum((residual_H*sig_H)**2)
             lnL_m = -0.5*np.sum((residual_m*sig_m)**2)

             return  lnL_H + lnL_m
```

```
In [74]: t_i = time()
         sampler_unc = ultranest.ReactiveNestedSampler(['H0', 'OM', 'OL'], LogLikelihood_LCDM_unc, Prior_LCDM)
         result_unc = sampler_unc.run()
         sampler_unc.print_results()
         t_f = time()

         print('Sampling time: {} s'.format(t_f-t_i))
```

```
[ultranest] Sampling 400 live points from prior ...


Z=-4329.6(96.84%) | Like=-4317.93..-4317.90 [-4317.9259..-4317.9259]*| it/evals=6038/419124 eff=1.442
0% N=400    0

/opt/anaconda3/lib/python3.8/site-packages/ultranest/integrator.py:1601: UserWarning: Sampling from r
egion seems inefficient (0/40 accepted in iteration 2500). To improve efficiency, modify the transfor
mation so that the current live points are ellipsoidal, or use a stepsampler, or set frac_remain to a
lower number (e.g., 0.5) to terminate earlier.
  warnings.warn(warning_message)

[ultranest] Explored until L=-4e+03  317.90 [-4317.9062..-4317.9062]*| it/evals=6501/713964 eff=0.911
1% N=400
[ultranest] Likelihood function evaluations: 713964
[ultranest]   logZ = -4330 +- 0.1087
[ultranest] Effective samples strategy satisfied (ESS = 1600.2, need >400)
[ultranest] Posterior uncertainty strategy is satisfied (KL: 0.45+-0.06 nat, need <0.50 nat)
[ultranest] Evidency uncertainty strategy is satisfied (dlogz=0.29, need <0.5)
[ultranest]   logZ error budget: single: 0.16 bs:0.11 tail:0.01 total:0.11 required:<0.50
[ultranest] done iterating.

logZ = -4329.551 +- 0.287
  single instance: logZ = -4329.551 +- 0.163
  bootstrapped   : logZ = -4329.567 +- 0.287
  tail           : logZ = +- 0.010
insert order U test : converged: True correlation: inf iterations

    H0                  67 +- 12
    OM                  0.40 +- 0.12
    OL                  0.68 +- 0.20
Sampling time: 4959.338680028915 s
```

```
In [76]: points_unc = np.array(result_unc["weighted_samples"]["points"])
         weights_unc = np.array(result_unc["weighted_samples"]["weights"])
         scaledweights_unc = weights_unc / weights_unc.max()
         mask_unc = np.random.rand(len(scaledweights_unc)) < scaledweights_unc

         samples_unc = points_unc[mask_unc, :]
```

```
In [93]: # Extract and compile chains
         unc_chain_1 = samples_unc[:,0] # extract chain of H0 values
         unc_chain_2 = samples_unc[:,1] # extract chain if OM values
         unc_chain_3 = samples_unc[:,2] # extract chain if OL values
         unc_samples = np.vstack((unc_chain_1, unc_chain_2, unc_chain_3)).T

         # Save chains and evidence (do not forget)
         np.savetxt('LCDM unc.csv', unc_samples, delimiter=",")
         np.savetxt('LCDM unc Z.csv', [[result_unc['logz'] , result_unc['logzerr']]])

         # Open saved chains
         #samples_open =  open_samples('LCDM unc.csv')


         fig = corner.corner(unc_samples, labels=[r"$H_0$", r"$\Omega_m$", r"$\Omega_d$"],color='blue', label_kw
         args={"fontsize": 20},quantiles=[0.16, 0.5, 0.84],
                                 show_titles=True, title_kwargs={"fontsize": 20})
         fig.savefig('LCDM unc.pdf')
```



VI. Nested Sampling: LCDM with Gaussian prior

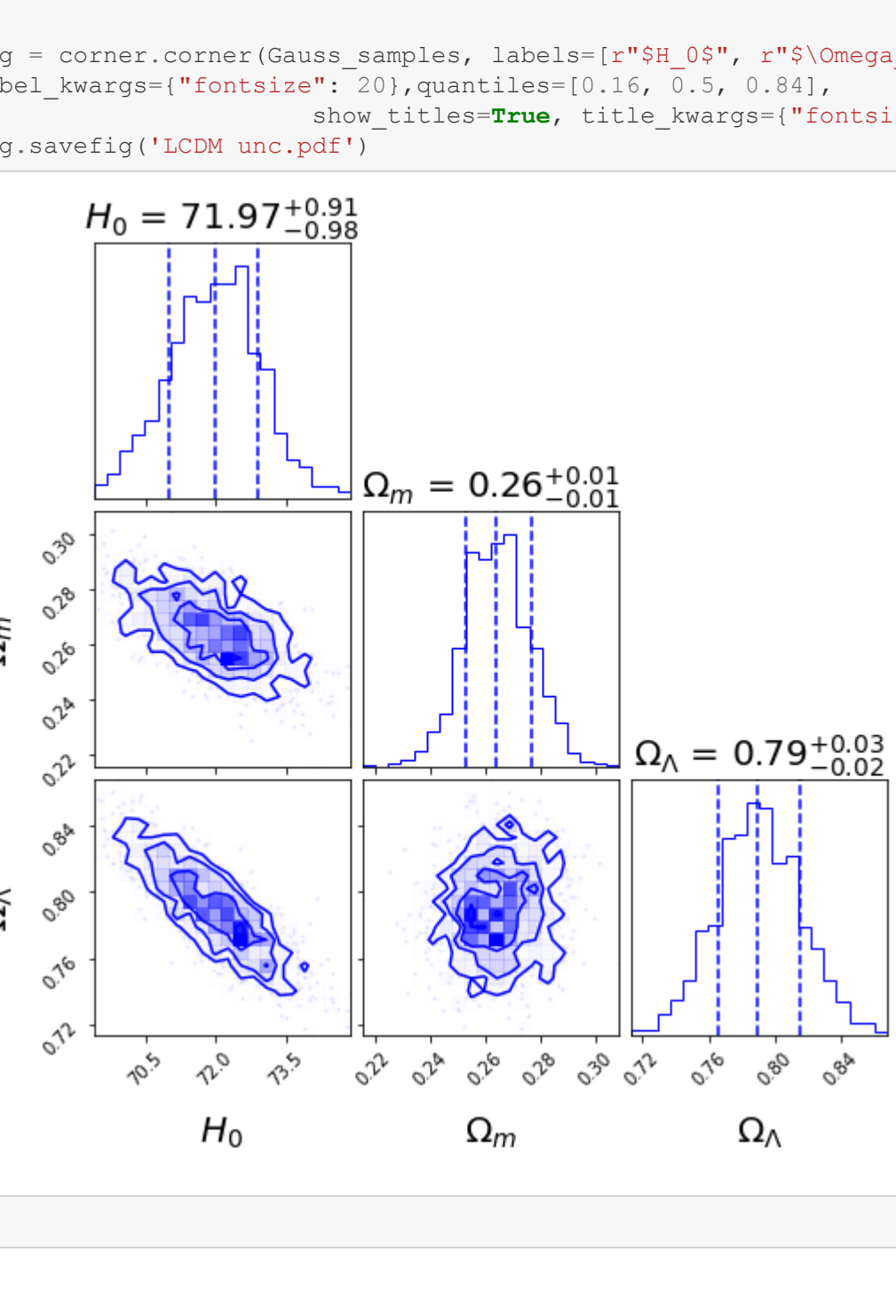
```
In [44]: def Prior_Gaussian(cube):

             H0 = scipy.stats.norm.ppf(cube[0], 72, 1)

             OM_min = 0
             OM_max = 1

             OD_min = 0
             OD_max = 1

             OMprime = cube[1]
             ODprime = cube[2]


             OM = OMprime*(OM_max-OM_min) + OM_min
             OD = ODprime*(OD_max-OD_min) + OD_min

             #OM = scipy.stats.norm.ppf(cube[1], 0.3111, 0.0056)
             #OD = scipy.stats.norm.ppf(cube[2], 0.6889, 0.0056)

             return np.array([H0, OM, OD])
```

```
In [45]: t_i = time()
         sampler_Gauss = ultranest.ReactiveNestedSampler(['H0', 'OM', 'OL'], LogLikelihood_LCDM, Prior_Gaussian)
         result_Gauss = sampler_Gauss.run()
         sampler_Gauss.print_results()
         t_f = time()

         print('Sampling time: {} s'.format(t_f-t_i))
```

```
[ultranest] Sampling 400 live points from prior ...


[ultranest] Explored until L=-4e+01   [-37.4506..-37.4505]*| it/evals=5036/23827 eff=21.4966% N=400

[ultranest] Likelihood function evaluations: 23842
[ultranest]   logZ = -45.39 +- 0.1094
[ultranest] Effective samples strategy satisfied (ESS = 1559.7, need >400)
[ultranest] Posterior uncertainty strategy is satisfied (KL: 0.46+-0.06 nat, need <0.50 nat)
[ultranest] Evidency uncertainty strategy is satisfied (dlogz=0.23, need <0.5)
[ultranest]   logZ error budget: single: 0.13 bs:0.11 tail:0.01 total:0.11 required:<0.50
[ultranest] done iterating.

logZ = -45.433 +- 0.228
  single instance: logZ = -45.433 +- 0.133
  bootstrapped   : logZ = -45.394 +- 0.228
  tail           : logZ = +- 0.010
insert order U test : converged: True correlation: inf iterations

    H0                  71.96 +- 0.96
    OM                  0.264 +- 0.012
    OL                  0.790 +- 0.026
Sampling time: 138.66106009483337 s
```

```
In [46]: points_Gauss = np.array(result_Gauss["weighted_samples"]["points"])
         weights_Gauss = np.array(result_Gauss["weighted_samples"]["weights"])
         scaledweights_Gauss = weights_Gauss / weights_Gauss.max()
         mask_Gauss = np.random.rand(len(scaledweights_Gauss)) < scaledweights_Gauss

         samples_unc = points_Gauss[mask_Gauss, :]
```

```
In [47]: # Extract and compile chains
         Gauss_chain_1 = samples_unc[:,0] # extract chain of H0 values
         Gauss_chain_2 = samples_unc[:,1] # extract chain if OM values
         Gauss_chain_3 = samples_unc[:,2] # extract chain if OL values
         Gauss_samples = np.vstack((Gauss_chain_1, Gauss_chain_2, Gauss_chain_3)).T

         # Save chains and evidence (do not forget)
         np.savetxt('LCDM Gaussian.csv', Gauss_samples, delimiter=",")
         np.savetxt('LCDM Gaussian Z.csv', [[result_Gauss['logz'] , result_Gauss['logzerr']]])

         # Open saved chains
         #samples_open =  open_samples('LCDM unc.csv')


         fig = corner.corner(Gauss_samples, labels=[r"$H_0$", r"$\Omega_m$", r"$\Omega_\Lambda$"],color='blue',
         label_kwargs={"fontsize": 20},quantiles=[0.16, 0.5, 0.84],
                                 show_titles=True, title_kwargs={"fontsize": 20})
         fig.savefig('LCDM unc.pdf')
```




```
In [ ]:
```