Luminosity_Distance_GW-sources

August 22, 2020

1 $d_L(z)$ Simulation for Einstein Telescope with Fake SFR Distribution

First of all let us load all necessaries libraries to run our code. Here we are going to use the Flat ΛCDM model as fiducial model with cosmological parameters $H_0 = 73\ km/s/Mpc$ and $\Omega_m = 0.3$.

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import quad, odeint
from pynverse import inversefunc
from astropy.cosmology import FlatLambdaCDM
from tqdm import tqdm

cosmo = FlatLambdaCDM(H0=73,Om0=0.3)
km_Mpc = 1.e3/3.086e22
```

The next step is define the estimated uncertainty of d_L (toy model) using the Einstein Telescope. This toy model consistis in supose that the uncertainty $\Delta d_L(z)$ behaves inversely proportional to the Signal to Noise ratio (SNR):

$$\frac{\Delta d_L}{d_L} \approx \frac{2}{SNR} \qquad where \qquad SNR = 2\left[\int_0^\infty \frac{df}{S_n(f)}|\tilde{h}(f)|^2\right]^{1/2} \tag{1}$$

Also we are going to define a function that returns the comoving volume per unity of redshift as:

$$\frac{dV_c}{dz} = \frac{4\pi d_L^2(z)}{(1+z)^2 H(z)} \tag{2}$$

[2]:
```
# Computing dL uncertainty:
def Err(z,dL): return ( 0.1618*z - 0.0289*z**2 + 0.002*z**3 )*dL

# Computing Comoving Volume per Redshift:
def dVc_dz(z):
        dL = cosmo.luminosity_distance(z).value
        H = cosmo.H(z).value*km_Mpc
        return 4*np.pi*dL**2/((1.+z)**2*H)
```

In the following code block we are going to start modeling our fake events distribution modifying the equation of star formation rate given in (arXiv:1403.0007):

$$\psi(z) \equiv \frac{dN}{dtdV_c} = \phi_0 \frac{(1+z)^\alpha}{1+\left[\frac{1+z}{C}\right]^\beta} \tag{3}$$

where the coefficients of this expression are (see arXiv:1403.0007) :

$$\phi_0 = 0.0015 \qquad \alpha = 0.015 \qquad \beta = 5.6 \qquad C = 2.9 \tag{4}$$

Our modification will be modifying the coefficient C in order to pull the peack of the distribution to be approximatelly in $z = 1$.

[9]:
```
z_star = 1 # Redshift of the Maximum Distribution
phi0, alpha, beta = 0.015, 2.7, 5.6
C = (z_star+1)*pow(beta/alpha-1, 1/beta) # C = 2.9 in (arXiv:1403.0007)
def SFR(z): return phi0*(1+z)**alpha/(1+((1+z)/C)**beta) # Star Formation Rate
def Auxiliar(z): return dVc_dz(z)*SFR(z)/(1+z)
```

The function called Auxiliar(z) will define our source distribution that will be normalized aftermore.

$$\frac{dN}{dt_{obs}dz} = \frac{\psi(z)}{n(1+z)} \frac{dV_c}{dz} \tag{5}$$

where n is choosen such that:

$$\int_0^{z_{max}} \frac{dN}{dt_{obs}dz} dz = N_{tot} \tag{6}$$

Here we would like to deal with a total number of sources $N_{tot} = 100$ distributed in the redshift range between 0 and 2

[4]:
```
# Computing the Normalization Constant
Ntot = 100 ; z_max = 2         # Total number of sources and maximum redshift
n, _ = quad(Auxiliar, 0, z_max)
n /= Ntot
```

Computed the normalization constant n we can define the resulting cumulative distribution function $\mathcal{N}(z)$:

$$\mathcal{N}(z) \equiv \int_0^\infty \frac{dN}{dt_{obs}dz} dz \tag{7}$$

[5]:
```
# Defining Our Data Distribution Function
def Pz(z): return Auxiliar(z)/n

def Nz(z): # N(z)
```

```
        x,_ = quad(Pz, 0, z)
        return x
```

With the function $\mathcal{N}(z)$ in hand we would like to invert this function to be $z(\mathcal{N})$ in order to set the redshifts were our sources will be placed. For example, the first source will be placed at $z(\mathcal{N}=1)$, the second one in $z(\mathcal{N}=2)$ and so on until the last source be placed at $z(\mathcal{N}=N_{tot})$.

With the redshift source positions and the values of $\Delta d_L(z)$, we are going to compute the luminosity distance $d_L(z)$ using our fiducial model and scattering these values from a Gaussian distribution:

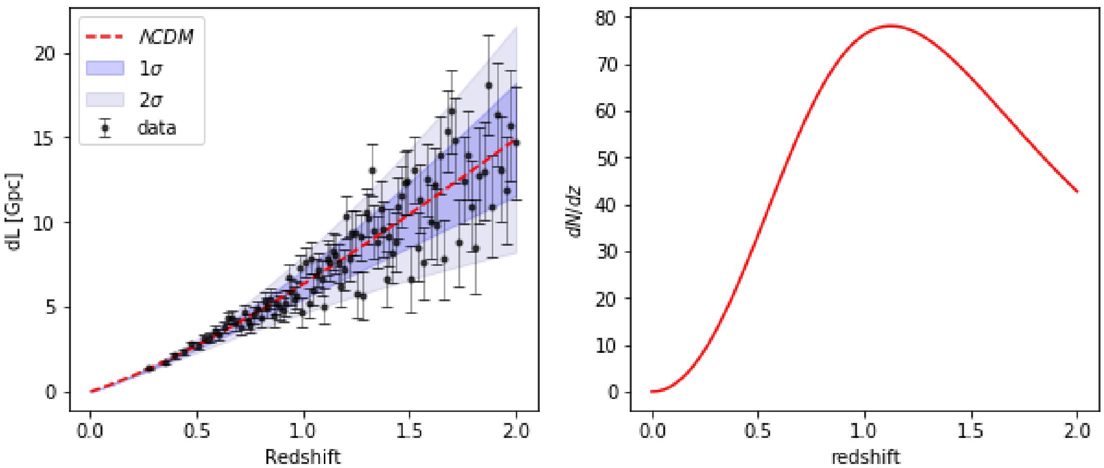
$$\mathbf{data}(d_L) \sim N(\mu = d_L, \sigma = \Delta dL) \tag{8}$$

[12]:
```
N = np.linspace(1,Ntot,Ntot)
z = np.zeros(Ntot)
for i in tqdm(range(Ntot)): z[i] = inversefunc(Nz , N[i], domain=[1.e-10,3])

dL = cosmo.luminosity_distance(z).value
Error = Err(z,dL)
data = np.random.normal( dL, Error )
```

```
100%|      | 100/100 [01:29<00:00,  1.12it/s]
```

Finaly we can plot our results.

[7]:
```
data/=1.e3 ; Error/=1.e3 ; dL/=1.e3
fig = plt.figure(figsize=(10,4))
plt.subplot(1,2,1)
plt.errorbar( z, data, Error, fmt='k.', capsize=3, elinewidth=0.5, alpha=0.
 ↪7,label='data')
z = np.linspace(0,2,100); dL = cosmo.luminosity_distance(z).value/1.e3; Error =␣
 ↪Err(z,dL)
plt.fill_between(z,dL+Error,dL-Error,color='blue',alpha=0.2,label='$1\sigma$')
plt.fill_between(z,dL+2*Error,dL-2*Error,color='darkblue',alpha=0.
 ↪1,label='$2\sigma$')
plt.plot(z,dL,'r--',label='$\Lambda CDM$'); plt.legend(loc='best')
plt.xlabel('Redshift'); plt.ylabel('dL [Gpc]')

plt.subplot(1,2,2)
plt.plot(z,Pz(z),'r-')
plt.xlabel('redshift'); plt.ylabel('$dN/dz$')
fig.savefig('simple_p.png')
plt.show()
```



ΛCDM
1σ
2σ
data
dL [Gpc]
Redshift
dN/dz
redshift