```python
# E(z) en andere kosmische parameters berekenen
#
# een paar bibliotheken klaarzetten
#
import numpy as np
import matplotlib.pyplot as plt
import math
#
#
const=977.8 #(1 hubble is bijna 1000 miljard jaar)
hubble=67.3 # LCMD
#
age_const=const/hubble #(1/hubble) = Hubble time
#
dist_const=3e5/hubble # in Mpc
dist_const2=dist_const*1000 # in kpc

#
dz = 0.001 # stap redshift
z_start = 0 # begin van de redshift
z_end = 20 # eind van de redshift
n_steps = int ( round (( z_end - z_start )/dz)) # number of timesteps
print(n_steps)
#
z_arr = np. zeros ( n_steps ) # array of zeros
#
Ez_arr = np. zeros ( n_steps) # array of zeros
Comoving_arr = np. zeros ( n_steps ) # array of zeros
App_diam_arr = np. zeros ( n_steps ) # array of zeros
diam_arr = np. zeros ( n_steps )     # array of zeros
Lum_dist_arr = np. zeros ( n_steps ) # array of zeros
Dist_mod_arr = np. zeros ( n_steps ) # array of zeros
Lookback_arr = np. zeros ( n_steps ) # array of zeros
#
Ez_arr2 = np. zeros ( n_steps) # array of zeros
Comoving_arr2 = np. zeros ( n_steps ) # array of zeros
App_diam_arr2 = np. zeros ( n_steps ) # array of zeros
diam_arr2 = np. zeros ( n_steps )     # array of zeros
Lum_dist_arr2 = np. zeros ( n_steps ) # array of zeros
Dist_mod_arr2 = np. zeros ( n_steps ) # array of zeros
Lookback_arr2 = np. zeros ( n_steps ) # array of zeros


# eerste berekening
Omega_m=0.3
Omega_l=0.7
Omega_r=0.0 # is eigenlijk 0.0001

str_m1=str(Omega_m)
str_l1=str(Omega_l)
label1='lambda='+str_l1+',matter='+str_m1

Omega_k=1-Omega_m-Omega_l-Omega_r
print(Omega_k,Omega_m,dist_const)

#
# calculalting Ez
for i in range (0, n_steps):
    z = z_start + i * dz
    z_arr [i] = z
    Ez  = Omega_r * ((z+1)**4) + Omega_m * ((z+1)**3) + (Omega_k * (z+1)**2) + Omega_l
    Ez_arr [i] = np.sqrt(Ez)

#
integral=0
integral2=0
for i in range (0, n_steps):
    z = z_arr [i]
    integral = integral + dz/(Ez_arr [i])
    Comoving = integral * dist_const2 #in kpc
    App_diam_arr [i] = Comoving/(z+1)
    Comoving_arr [i] = integral * dist_const  #in Mpc
```

```
    Lum_dist_arr [i] = Comoving_arr [i]*(z+1)
    Dist_mod_arr [i] = 5 * math.log(Lum_dist_arr [i], 10) + 25
    diam_arr [i] = 1/App_diam_arr [i] # in radialen
    diam_arr [i] = diam_arr [i]*57.3*3600 # in arcsec/kpc
    integral2 = integral2 + dz/((z+1)* Ez_arr [i])
    Lookback_arr [i] = integral2 * age_const


# tweede berekening
Omega_m=1.0
Omega_l=0.0
Omega_r=0.0

str_m2=str(Omega_m)
str_l2=str(Omega_l)
label2='lambda='+str_l2+',matter='+str_m2

Omega_k=1-Omega_m-Omega_l-Omega_r
print(Omega_k,Omega_m,dist_const)

#
# calculating Ez2
for i in range (0, n_steps):
    z = z_arr [i]
    Ez = Omega_r * ((z+1)**4) + Omega_m * ((z+1)**3) + (Omega_k * (z+1)**2) + Omega_l
    Ez_arr2 [i] = np.sqrt(Ez)

#
integral=0
integral2=0
for i in range (0, n_steps):
    z = z_arr [i]
    integral = integral + dz/(Ez_arr2 [i])
    Comoving = integral * dist_const2 #in kpc
    App_diam_arr2 [i] = Comoving/(z+1)
    Comoving_arr2 [i] = integral * dist_const #in Mpc
    Lum_dist_arr2 [i] = Comoving_arr2 [i]*(z+1)
    Dist_mod_arr2 [i] = 5 * math.log(Lum_dist_arr2 [i], 10) + 25
    diam_arr2 [i] = 1/App_diam_arr2 [i] # in radialen
    diam_arr2 [i] = diam_arr2 [i]*57.3*3600 # in arcsec/kpc
    integral2 = integral2 + dz/((z+1)* Ez_arr2 [i])
    Lookback_arr2 [i] = integral2 * age_const

#
fig = plt. figure (1)
plt . plot (z_arr , App_diam_arr , linewidth = 2, color ='blue', label = label1)
plt . plot (z_arr , App_diam_arr2 , linewidth = 2, color ='red', label = label2)
plt.title ("App_diam distance", fontsize = 12)
plt.grid ( True )
plt.xlabel ('redshift')
plt.ylabel ('value (kpc)')
plt.legend  ()
#plt.axis ([0, 20 , 0, 14])

#
fig = plt. figure (2)
plt . plot (z_arr , Comoving_arr , linewidth = 2, color ='blue', label = label1)
plt . plot (z_arr , Comoving_arr2 , linewidth = 2, color ='red', label = label2)
plt.title ("Comoving distance", fontsize = 12)
plt.grid ( True )
plt.xlabel ('redshift')
plt.ylabel ('value (Mpc)')
plt.legend  ()
#plt.axis ([0, 20 , 0, 0.1])

#
fig = plt. figure (3)
plt . plot (z_arr , diam_arr , linewidth = 2, color ='blue', label = label1)
plt . plot (z_arr , diam_arr2 , linewidth = 2, color ='red', label = label2)
plt.title ("apparent diameter", fontsize = 12)
plt.grid ( True )
plt.xlabel ('redshift')
```

```
plt.ylabel ('arcsec/kpc')
plt.legend  ()
plt.axis ([0, 20 , 0, 2])

#
fig = plt. figure (4)
plt . plot (z_arr , Lum_dist_arr , linewidth = 2, color ='blue', label = label1)
plt . plot (z_arr , Lum_dist_arr2 , linewidth = 2, color ='red', label = label2)
plt.title ("luminosity distance", fontsize = 12)
plt.grid ( True )
plt.xlabel ('redshift')
plt.ylabel ('value (Mpc)')
plt.legend  ()
# plt.axis ([0, 20 , 0, 2])

#
fig = plt. figure (5)
plt . plot (z_arr , Dist_mod_arr , linewidth = 2, color ='blue', label = label1)
plt . plot (z_arr , Dist_mod_arr2 , linewidth = 2, color ='red', label = label2)
plt.title ("distance modulus", fontsize = 12)
plt.grid ( True )
plt.xlabel ('redshift')
plt.ylabel ('value (mag)')
plt.legend  ()
plt.axis ([0, 20 , 40, 55])

#
fig = plt. figure (6)
plt . plot (z_arr , Lookback_arr  , linewidth = 2, color ='blue', label = label1)
plt . plot (z_arr , Lookback_arr2 , linewidth = 2, color ='red', label = label2)
plt.title ("Lookback time", fontsize = 12)
plt.grid ( True )
plt.xlabel ('redshift')
plt.ylabel ('value (Gjaar)')
plt.legend  ()

#
plt . show ()

# end
```