NUR Assignment I: solutions

Tina Neumann

March 6, 2023

Abstract

This document displays my solution for the first assignment in the course numerical recipes for astrophysics, summer term 2023.

1 Exercise 1: Poisson distribution

In this exercise the poisson probability distribution for integer k should be calculated with the given values.

The poisson probability distribution is given as:

$$P_\lambda(k) = \frac{\lambda^k exp(-)}{k!} \tag{1}$$

While calculations a straight-forward derivation with 32-bit values would run into overflow-problems for the k-factorial, the given equation was converted into log-space which gives the following equation:

The result is calculated the following way:

$$log(P_\lambda(k)) = \lambda * k - -\sum(k) \tag{2}$$

The $P_\lambda(k)$ for the given values are calculated for 32-bit and 64-bit values with into log-space converted equation and as comparison the same is derived using *scipy* via the given equation.

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

# # Numerical recipies
# Assignment (09.03.23)
#
# Tina Neumann

# In[28]:


import numpy as np
import timeit


# In[40]:


### Exercise 1: Poisson distribution
# define given distribution
def poiss_32(lam, k):
    '''Function determines the poisson distribution P(k) of
    input: a positive mean (lam) and an integer (k)
    output: P(k)'''
    lam = np.float32(lam) #redefine as 32-bit integers
    k = np.int32(k)
    # to decrease the calculation time:
    # multiply by inverse instead of dividing
    # the values are considered to be in log-scale: log(products)--> sums
    # define k!
```

```
    f_frac = 0.
    for f in range(k+1): #range leaves out last element
        f_frac += f

    log_p = np.int32(k)*np.float32(lam) - np.float32(lam) - np.int32(f_frac) # logarithmic P(k)
    p = np.exp(np.float32(log_p)) #revert log-scale
    print('k! = ', f_frac)
    print('log(P(k)) = ', log_p)
    return p


# In[41]:


def poiss_64(lam, k):
    '''Function determines the poisson distribution P(k) of
    input: a positive mean (lam) and an integer (k)
    output: P(k)'''
    lam = np.float64(lam) #redefine as 64-bit integers
    k = np.int64(k)
    # define k!
    f_frac = 0.
    for f in range(k+1): #range leaves out last element
        f_frac += f

    log_p = k*lam - lam - f_frac #logarithmic P(k)
    p = np.exp(log_p) #revert log-scale
    print('k! = ', f_frac)
    print('log(P(k)) = ', log_p)
    return p


# In[42]:


from scipy.special import factorial
def poiss(lam, k):
    '''Function determines the poisson distribution P(k) of
    input: a positive mean (lam) and an integer (k)
    output: P(k)'''
    lam = np.float64(lam)
    k = np.int64(k)
    #with numpy functions
    # define k!
    f_frac = factorial(k)
    p =  lam**k*np.exp(-lam)*f_frac**(-1)

    return p


# In[43]:


# read-in given values
with open('input_1a.txt') as f:
    lines = f.readlines()[2:]
    for line in lines:
            #define lambda, k
            mean_lam, k_int = line.split('\t')
            print(mean_lam,k_int)
            print('The given values are lam & k: '+ mean_lam, ' & '+ k_int)
            print('For 32 bits this results in a poisson distribution of P(k) = ', np.round(poiss_32(mean_lam,k_int),6)) #print 6 relevant digits
            print('For 64 bits this results in a poisson distribution of P(k) = ', np.round(poiss_64(mean_lam,k_int),6)) #print 6 relevant digits
            print('For numpy-functions this results in a poisson distribution of P(k) = ', np.round(poiss(mean_lam,k_int),6)) #print 6 relevant digits
```

1_PoissonDistribution.py

The result of $P_\lambda(k)$ are given in: