ASTR5550 Homework 2

Girish M. Duvvuri

Due 2 February, 2018

Problem 1

a) Let $\mu = rAt$:

$$\implies P(k|t) = \frac{\mu^k e^{-\mu}}{k!} = \frac{\mu^3}{6} e^{-\mu} \tag{1}$$

b) Let $\frac{P(t)}{P(k)} = C$:

$$\implies P(t|k) = C \times \frac{\mu^3}{6} e^{-mu} \tag{2}$$

$$\implies 1 = C \int_0^\infty \frac{(rAt)^k}{k!} e^{-rAt} dt \tag{3}$$

$$\implies 1 = \frac{C}{6} \times \frac{\Gamma(k+1)}{rA} \tag{4}$$

$$\implies C = \frac{6rA}{\Gamma(4)} = rA \tag{5}$$

$$\implies P(t|k) = \frac{(rA)^4}{6} t^3 e^{-(rAt)} \tag{6}$$

I also verified that the numerical integral of this expression was close to 1. Using a $t-$array from 0 to 10^{10} years with 10^5 log-spaced points, the trapezoidal-rule integrated probability was $1 + 8.4 \times 10^{-9}$.

c) 68% of the likelihood $P(t|k)$ is enclosed within the interval $15.59 < t \leq 51.38$ Myr. The maximum likelihood is at $t = 30$ Myr.

d) I predict that the range will decrease since we have a greater sample size to work with. This was correct: with the new measurements we get the maximum likelihood $t = 37$ Myr with a 68% confidence interval: $31.24 < t < 43.43$ Myr (See Figure 1).

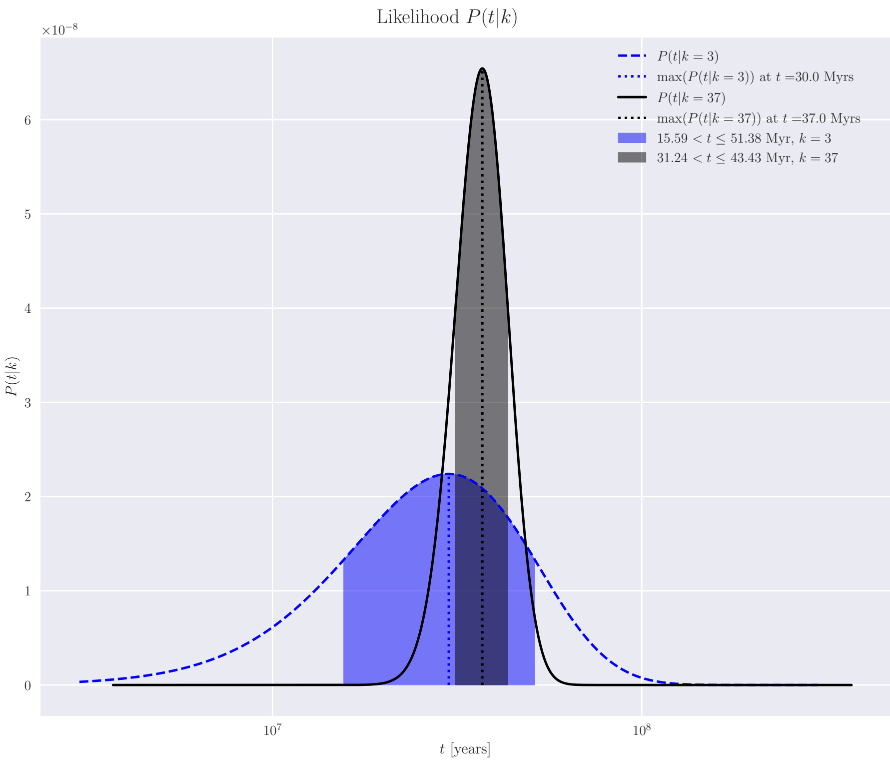

Figure 1: Increasing the measurement area and getting the new count value $k = 37$ narrows the probability distribution to a new value within the 68% confidence interval we had when $k = 3$, but offset from the original peak value of 30 Myr. The newer measurement is 37 Myr with a 68% confidence interval $31.24 < t < 43.43$ Myr.

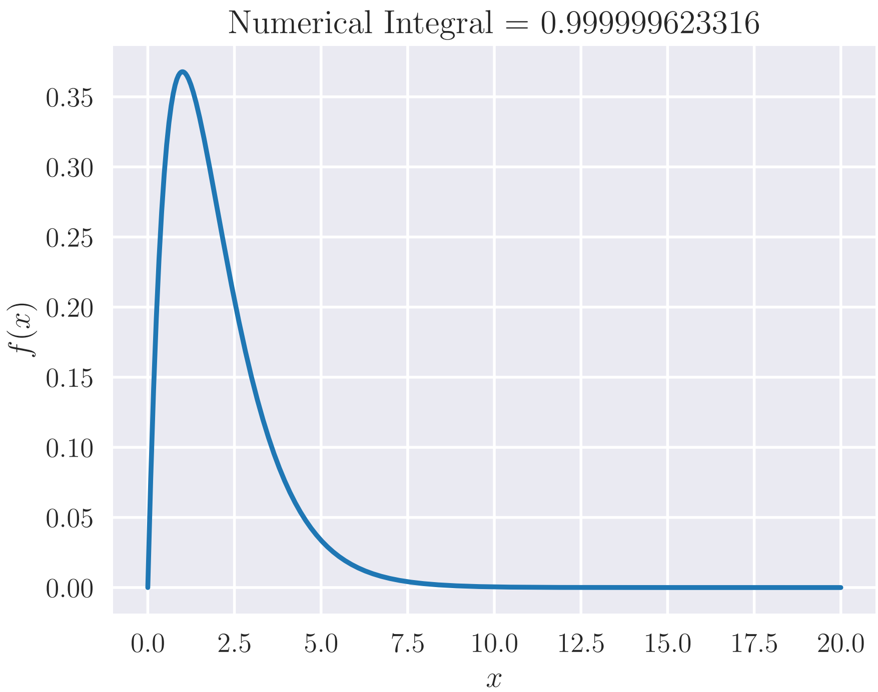

Figure 2: With only a range between 0 and 20 spaced with 10^4 points the numerical integral is nearly 1.

Problem 2

a) See Figure 2 for the plot. The numerical integral is shown in the plot, but analytically:

$$\begin{aligned}
\int_{-\infty}^{\infty} f(x)dx &= \int_{-\infty}^{0} 0\ dx + \int_{0}^{\infty} xe^{-x}dx \\
&= (-xe^{-x})|_0^\infty - \int_0^\infty -e^{-x}dx \\
&= (0-0) - \left(e^{-x}\right)|_0^\infty \\
&= 0 - (0-1) = 1 \\
\implies & \boxed{\int_{-\infty}^{\infty} f(x)dx = 1}
\end{aligned}$$

b)

$$
\begin{aligned}
F(x) &= \int_{-\infty}^{x} f(x')dx' \\
&= \int_{0}^{x} x'e^{-x'}dx' \\
&= (-x'e^{-x'})|_0^x - \int_0^\infty -e^{-x'}dx' \\
&= \left(-xe^{-x} - 0\right) - \left(e^{-x'}\right)|_0^x \\
&= -xe^{-x} - \left(e^{-x} - 1\right) = 1 - e^{-x}\left(1+x\right) \\
\implies & \boxed{F(x) = 1 - e^{-x}\left(1+x\right)}
\end{aligned}
$$

c)

$$
\begin{aligned}
E[x] &= \int_{-\infty}^{\infty} xf(x)dx \\
&= \int_0^\infty x^2e^{-x}dx \\
&= (-x^2e^{-x})|_0^\infty - \int_0^\infty -2xe^{-x}dx \\
&= (0-0) + 2\int_0^\infty xe^{-x}dx \\
&= 0 + 2(1) \quad \text{From part } \mathbf{a} \\
\implies & \boxed{E[x] = 2}
\end{aligned}
$$

d)

$$
\begin{aligned}
V[x] &= E[k^2] - (E[k])^2 \\
&= \int_{-\infty}^{\infty} x^2 f(x)dx - 4 \\
&= \int_0^\infty x^3e^{-x}dx - 4 \\
&= (-x^3e^{-x})|_0^\infty - \int_0^\infty -3x^2e^{-x}dx - 4 \\
&= (0-0) + 3\int_0^\infty x^2e^{-x}dx - 4 \\
&= 0 + 3(2) - 4 \quad \text{From part } \mathbf{c} \\
\implies & \boxed{V[x] = 2}
\end{aligned}
$$

The total probability within the range $E[k] \pm \sqrt{V[x]}$ is:

$$\begin{aligned}\int_{2-\sqrt{2}}^{2+\sqrt{2}} xe^{-x}dx &= (2-\sqrt{2}+1)e^{-(2-\sqrt{2})} - (2+\sqrt{2}+1)e^{-(2+\sqrt{2})} \\ &\approx 0.73\end{aligned}$$

Problem 3

a)

$$f(k) = \frac{\lambda^k e^{-\lambda}}{k!} \tag{7}$$

b)

$$E[k(k-1)] = E[k^2 - k] = E[k^2] - E[k] = V[k] + \lambda^2 - \lambda = V[k]$$

$$\begin{aligned}E[k(k-1)] &= \sum_{k=0}^{\infty} k(k-1)\frac{\lambda^k e^{-\lambda}}{k!} \\ &= \sum_{k=2}^{\infty} k(k-1)\frac{\lambda^k e^{-\lambda}}{k!} \\ &= \sum_{k=2}^{\infty} \lambda^2 \frac{\lambda^{k-2} e^{-\lambda}}{(k-2)!} \\ &= \lambda^2 \sum_{k=2}^{\infty} \frac{\lambda^{k-2} e^{-\lambda}}{(k-2)!} \\ &= \lambda^2 \times 1 \quad \text{Sum is identical to summing over original Poisson distribution}\end{aligned}$$

$$\boxed{\implies \lambda^2 = V[k]} \tag{8}$$

c, d, e)

$$\sigma_\gamma = \lambda \tag{9}$$

$$\begin{aligned}\mu_\gamma &= \lambda = \sigma_\gamma \\ \mu_{\mathrm{DN}} = \sigma_{\mathrm{DN}} &= \frac{\lambda}{G}\end{aligned}$$

f) Yes, j also follows a Poisson distribution indicating that the uncertainty of a CCD pixel measurement will follow a Poisson distribution as well.

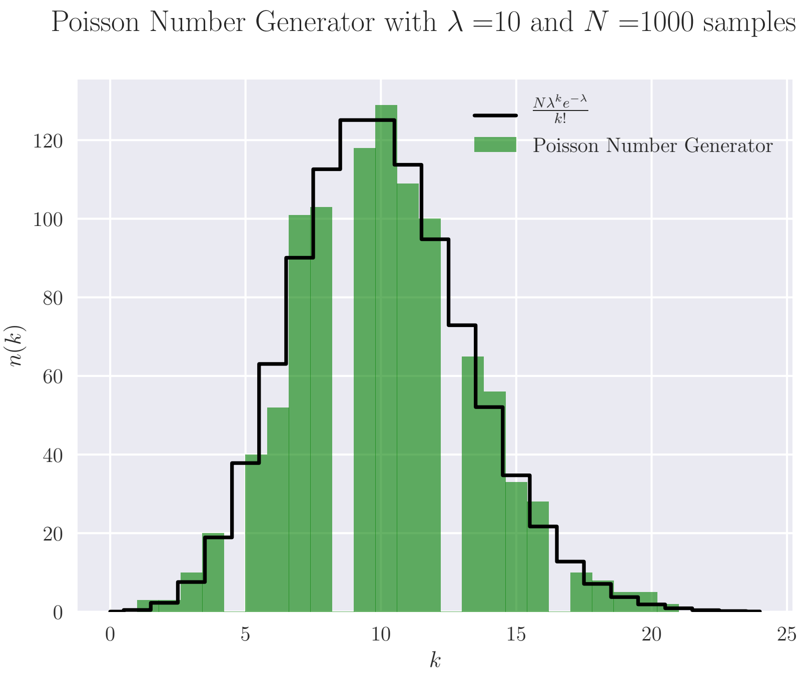

Figure 3: The histogram looks a little weird because of the automatic binsize choice, but it follows the PDF quite well.

Problem 4

a) The algorithm in [Press et. al (2007)] had sections for the case of small $\lambda < 5$ and large $\lambda > 13.5$ which were not relevant in our case for $\lambda = 10$. Excluding these sections, I adapted the code to Python and ran the sampler for $N = 1000$ cases. The results are shown in Figure 3.

Code for all Problems:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from astropy import units as u
sns.set_style('darkgrid')
sns.set_context('talk')
plt.rc('text', usetex=True)
plt.rc('font', family='serif')


def make_pdf(t_array=np.logspace(5.0, 10.0, 10**4)*(u.yr), k=3.0,
             A=10.0*((u.km)**2.0), r=0.01/((u.km**2)*(10.0**6.0*u.yr))):
    mu = (r*A*t_array).decompose()
    norm_constant = r*A
    return (r*A)*(mu**k)*np.exp(-mu)/(np.math.factorial(int(k))), t_array


def calc_pdf_things(t_array=np.logspace(5.0, 10.0, 10**4)*(u.yr),
                    k=3.0,
                    A=10.0*((u.km)**2.0),
                    r=0.01/((u.km**2)*(10.0**6.0*u.yr))):
    pdf, t_array = make_pdf(t_array, k, A, r)
    max_loc = np.argmax(pdf.value)
    t_max = t_array[max_loc]
    pdf_max_array = np.linspace(0.0*pdf[0], pdf[max_loc], 1000)
    t_max_array = np.ones_like(pdf_max_array)*t_max
    equal_points = np.array([np.where(pdf[max_loc:] <= pdf[i])[0][0]
                             for i in range(0, max_loc)])
    integral_value = np.array([np.trapz(pdf[i:max_loc + equal_points[i]],
                                        t_array[i:max_loc + equal_points[i
    ]])
                               for i in range(0, max_loc)])
    test_integral = np.array([np.abs(integral_value[i] - 0.68)
                              for i in range(0, max_loc)])
    loc_low = np.argmin(test_integral)
    loc_high = max_loc + equal_points[np.argmin(test_integral)]
    t_low = t_array[loc_low]
    t_high = t_array[loc_high]

    max_label = r'max$(P(t\vert k=' + str(int(k)) + r'))$ at $t=$'
    max_label += "{:.1f}".format(t_max.value/(10**6)) + r' Myrs'
```

```
    fill_label = "{:.2f}".format(t_low.value/(10**6)) + r' '
    fill_label += r'\/$< t \leq $\/' + r' '
    fill_label += "{:.2f}".format(t_high.value/(10**6))
    fill_label += r'\textrm{ Myr}, $k=' + str(int(k)) + r'$'
    return [[pdf, t_array],
            [pdf_max_array, t_max_array],
            [loc_low, loc_high, max_loc, t_low, t_high],
            max_label, fill_label]


def plot_pdf_things(t_array=np.logspace(5.0, 10.0, 10**4)*(u.yr),
                    k=3.0,
                    A=10.0*((u.km)**2.0),
                    r=0.01/((u.km**2)*(10.0**6.0*u.yr)),
                    diff_plot=2000,
                    save_name='meteor_1.pdf',
                    title_label=r'Likelihood $P(t \vert k)$',
                    figure_size=(12, 10),
                    bbox=(0.98, 1.0),
                    suptitle_y=0.998):
    [[pdf, t_array], [pdf_max_array, t_max_array],
     [loc_low, loc_high, max_loc, t_low, t_high],
     max_label, fill_label] = calc_pdf_things()
    mask = np.arange(max_loc - diff_plot, max_loc + diff_plot)
    [[pdf_2, t_array_2], [pdf_max_array_2, t_max_array_2],
     [loc_low_2, loc_high_2, max_loc_2, t_low_2, t_high_2],
     max_label_2, fill_label_2] = calc_pdf_things(k=37.0, A=100.0*(u.km
    **2.0))
    mask_2 = np.arange(max_loc_2 - diff_plot, max_loc_2 + diff_plot)
    plt.figure(figsize=figure_size)
    plt.semilogx(t_array[mask], pdf[mask], '--b', label=r'$P(t\vert k=3)$')
    plt.plot(t_max_array, pdf_max_array, ':b', label=max_label)
    plt.fill_between(t_array[loc_low:loc_high],
                     pdf[loc_low:loc_high], 0.0,
                     color='b', alpha=0.5,
                     label=fill_label)
    plt.semilogx(t_array_2[mask_2],
                 pdf_2[mask_2], '-k', label=r'$P(t\vert k=37)$')
    plt.plot(t_max_array_2, pdf_max_array_2, ':k', label=max_label_2)
    t_fill_array = np.array(t_array_2[loc_low_2:loc_high_2], dtype=float)
    pdf_fill_array = np.array(pdf_2[loc_low_2:loc_high_2], dtype=float)
    plt.fill_between(t_fill_array,
                     pdf_fill_array, 0.0,
```

```
                     color='k', alpha=0.5,
                     label=fill_label_2)
    plt.ylabel(r'$P(t\vert k)$')
    plt.xlabel(r'$t$ [years]')
    plt.legend(bbox_to_anchor=bbox)
    plt.suptitle(title_label,
                 y=suptitle_y)
    plt.tight_layout()
    plt.savefig(save_name)
    plt.show()


def do_single_press_Poisson(mean_rate=10):
    old_mean = -1.0
    mean_now = mean_rate
    logfact = np.ones((1024), dtype=np.longdouble)*(-1.0)
    if mean_now != old_mean:
        sqlam = np.sqrt(mean_now)
        loglam = np.math.log(mean_now)
    blah = True
    while blah:
        u = 0.64*np.random.uniform()
        v = -0.68 + 1.28*np.random.uniform()
        k = int(np.floor(sqlam*(v/u)+mean_now+0.5))
        if k < 0:
            continue
        u2 = u*u
        if k < 1024:
            if logfact[k] < 0:
                logfact[k] = np.math.log(np.math.factorial(k))
            lfac = logfact[k]
        else:
            lfac = np.math.log(np.math.factorial(k))
        p = sqlam*np.exp(-mean_now + k*loglam - lfac)
        if u2 < p:
            break
    mean_old = mean_now
    return k


def do_N_Poisson_samples(mean_rate, N):
    return np.array([do_single_press_Poisson(mean_rate)
                     for i in range(0, N)])
```

```python


def problem_4(mean_rate=10, N=1000):
    sample_results = do_N_Poisson_samples(mean_rate, N)
    plt.figure(figsize=(8, 6))
    plt.hist(sample_results,
             label='Poisson Number Generator', alpha=0.6, bins='auto',
             color='green')
    k_array = np.arange(0, 25)
    factorial_array = np.array([np.math.factorial(k) for k in k_array],
                               dtype=float)
    pdf_array = np.array([(1.0/np.math.factorial(k))*(mean_rate**k)*np.exp(-mean_rate)
                          for k in range(0, 25)])
    plt.plot(k_array, N*pdf_array,
             label=r'$\frac{N \lambda^k e^{-\lambda}}{k!}$', color='k', drawstyle='steps-mid')
    plt.xlabel(r'$k$')
    plt.ylabel(r'$n(k)$')
    title_label = r'Poisson Number Generator with $\lambda=$' + str(mean_rate)
    title_label += r' and $N=$' + str(N) + r' samples'
    plt.suptitle(title_label)
    plt.legend()
    plt.savefig('poisson.pdf')
    plt.show()


if __name__ == '__main__':
    plot_pdf_things()
    problem_2()
    problem_4()
```

hw_2.py

References

[Press et. al (2007)] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery. *Numerical Recipes*, Cambridge University Press, Third edition, 2007.