Statistical Physics IV: Non-equilibrium statistical physics
ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE (EPFL)

Exercise No.6

Solution: Distribution of maximum step-sizes in Lévy flights

Question 1 The probability of observing an increment with size $y \geq \bar{y}$ is

$$Q(\bar{y}) = \int_{\bar{y}}^{\infty} P(y)dy \,.$$

The probability of observing a single event n_0 among n with a step-size $\bar{y} \geq y_n$, $n \neq n_0$ is given by

$$Q(\bar{y}, n) = n \cdot Q(\bar{y}) \cdot (1 - Q(\bar{y})) \,.$$

The step size with the maximum probability is given by the condition $\frac{dQ(\bar{y},n)}{d\bar{y}} = 0$. Taking the derivative and rearranging we obtain the condition:

$$1 - n \cdot Q(\bar{y}) = 0 \Leftrightarrow \int_{\bar{y}}^{\infty} P(y)dy = \frac{1}{n} \,.$$

Question 2 The limit of large n is equivalent to the limit in which $\bar{y} \to \infty$. We use the Taylor approximation of y^2 in the argument of the exponential to approximate the integral.

$$\begin{aligned}
\frac{1}{n} &= \int_{\bar{y}_n}^{\infty} \frac{1}{\sqrt{4\pi D\tau}} \exp\left(-\frac{y^2}{4D\tau}\right) dy \\
&\approx \int_{\bar{y}_n}^{\infty} \frac{1}{\sqrt{4\pi D\tau}} \exp\left(-\frac{\bar{y}_n^2 + 2\bar{y}_n(y - \bar{y}_n)}{4D\tau}\right) dy \\
&= \frac{1}{\sqrt{4\pi D\tau}} \exp\left(-\frac{\bar{y}_n^2}{4D\tau}\right) \int_0^{\infty} \exp\left(-\frac{2\bar{y}_n y}{4D\tau}\right) dy \\
&= \frac{1}{\sqrt{4\pi D\tau}} \exp\left(-\frac{\bar{y}_n^2}{4D\tau}\right) \frac{4D\tau}{2\bar{y}_n} \\
-\log(n) &= -\frac{\bar{y}_n^2}{4D\tau} - \log(\bar{y}_n) + \log\left(\frac{\sqrt{4D\tau}}{2\sqrt{\pi}}\right)
\end{aligned}$$

As $n \to \infty$ we only keep the highest order term in $\bar{y}_n$ on the right-hand side of the above equation. Thus we obtain:

$$\bar{y}_n \approx \sqrt{4D\tau \log n} = \sigma\sqrt{2\log n} \,,$$

so that

$$\lim_{n\to\infty} \frac{\bar{y}_n}{\sqrt{\langle x(n)^2 \rangle}} = \lim_{n\to\infty} \frac{\sigma\sqrt{2\log n}}{\sigma n} = 0 \,.$$

This suggests that the maximum most likely step-size does not contribute significantly to the second order moment of the random walk.

Question 3 For a Cauchy process the calculation is identical but significantly less technical.

$$\begin{aligned} n &= \left(\int_{\bar{y}_n}^{\infty} \frac{b}{y^2} dy \right)^{-1} \\ \Leftrightarrow \bar{y}_n &= nb \end{aligned}$$

In this case rare events do contribute to the average position after n steps:

$$\lim_{n\to\infty} \frac{\bar{y}_n}{\langle x(n) \rangle} = b \, .$$

Question 4 To prove the divergence of the second and first order moments we consider that the probability distribution only resembles $\frac{b}{y^{1+\mu}}$ for $y \geq y_0$. This is necessary to guarantee the normalisability of $P(y)$.

$$\begin{aligned} \langle y^\alpha \rangle &= \underbrace{\int_0^{y_0} y^\alpha P(y) dy}_{=A_\alpha < \infty} + \int_{y_0}^{\infty} y^\alpha P(y) dy \\ &= A_\alpha + b \int_{y_0}^{\infty} y^{\alpha-\mu-1} dy \end{aligned}$$

$$\langle y^\alpha \rangle = A_\alpha + \begin{cases} \infty, & \alpha = 2 \, , \; 0 < \mu < 2 \\ \infty, & \alpha = 1 \, , \; 0 < \mu < 1 \end{cases}$$

Solution: Arrhenius cascade

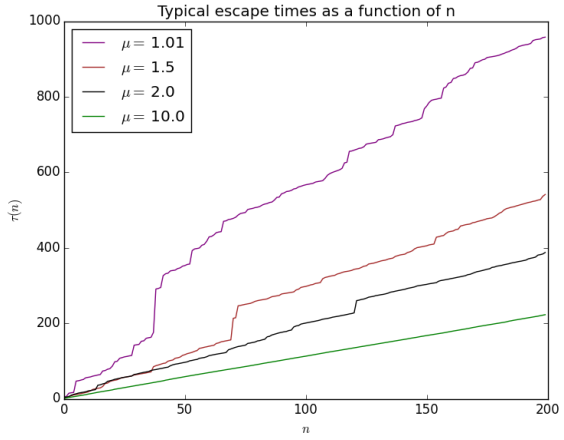
Question 1 In the case of constant well heights

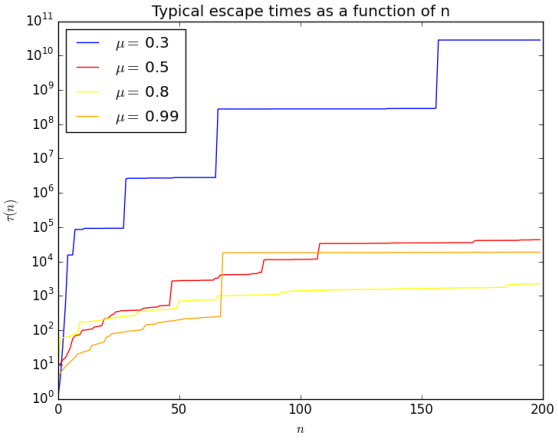
$$\tau(n) = \sum_{i=1}^{n} \tau_0 \exp(\beta V_0) = n\tau_0 \exp(\beta V_0)\,,$$

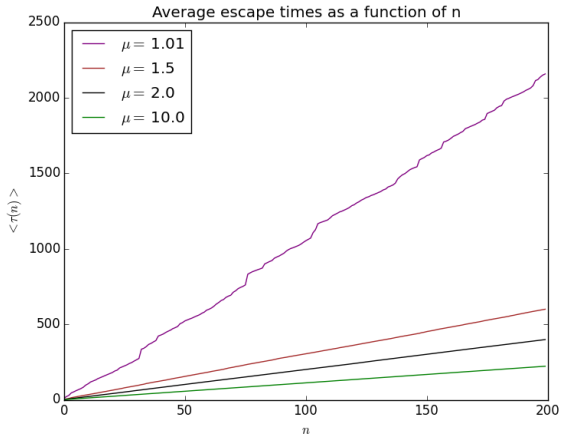
where $\beta = \frac{1}{k_B T}$ and V_0 is the constant energy height of the wells.
In the case of exponentially distributed well heights

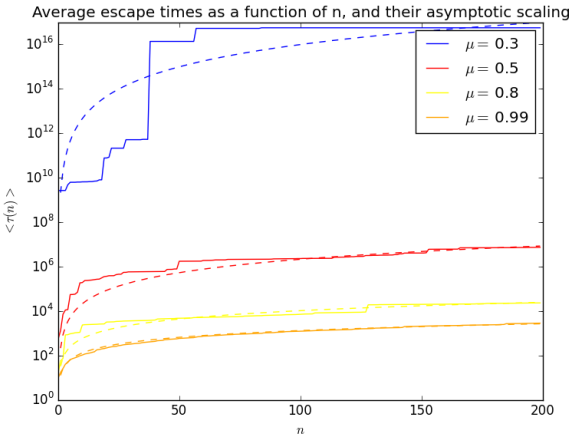
$$\begin{aligned}
\langle \tau(n) \rangle &= \sum_{i=1}^{n} \int_0^\infty \tau_0 \exp(\beta V_i) P(V_i) dV_i \\
&= \tau_0 n \int_0^\infty \frac{1}{E_0} \exp\left(\beta V - \frac{V}{E_0}\right) dV \\
&\overset{\frac{1}{\beta E_0} > 1}{=} \frac{n\tau_0}{1 - \beta E_0}
\end{aligned}$$

Question 2 See the supplementary scripts. The following plots show the simulations asked for.



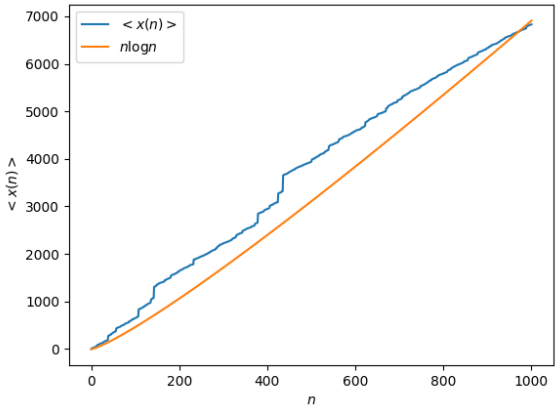




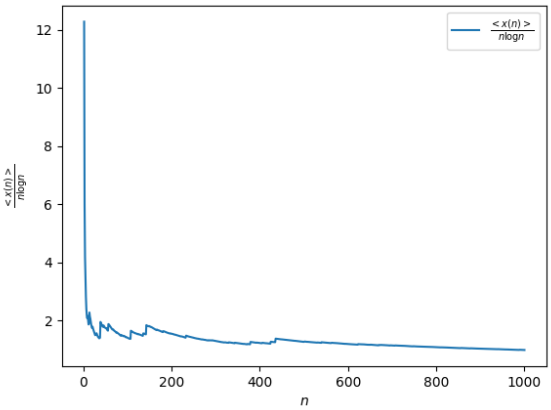


Solution: Simulation of Lévy flights

See the supplementary Python scripts



Question 1



7. Supplementary Codes for Arrhenius-cascade

7.1 .ipynb

Arrhenius cascade

April 2019

In this notebook we simulate escape time of a particle from a washboard potential using the Kramer's escape rate.

In [369]:
```
import numpy as np
%matplotlib notebook
import matplotlib.pyplot as plt
```

Washboard potential

A particle is moving in a washboard potential like figure below characterized by a series of potential wells of depth V_i. The potential barriers V_i are randomly distributed, according to an exponential distribution,

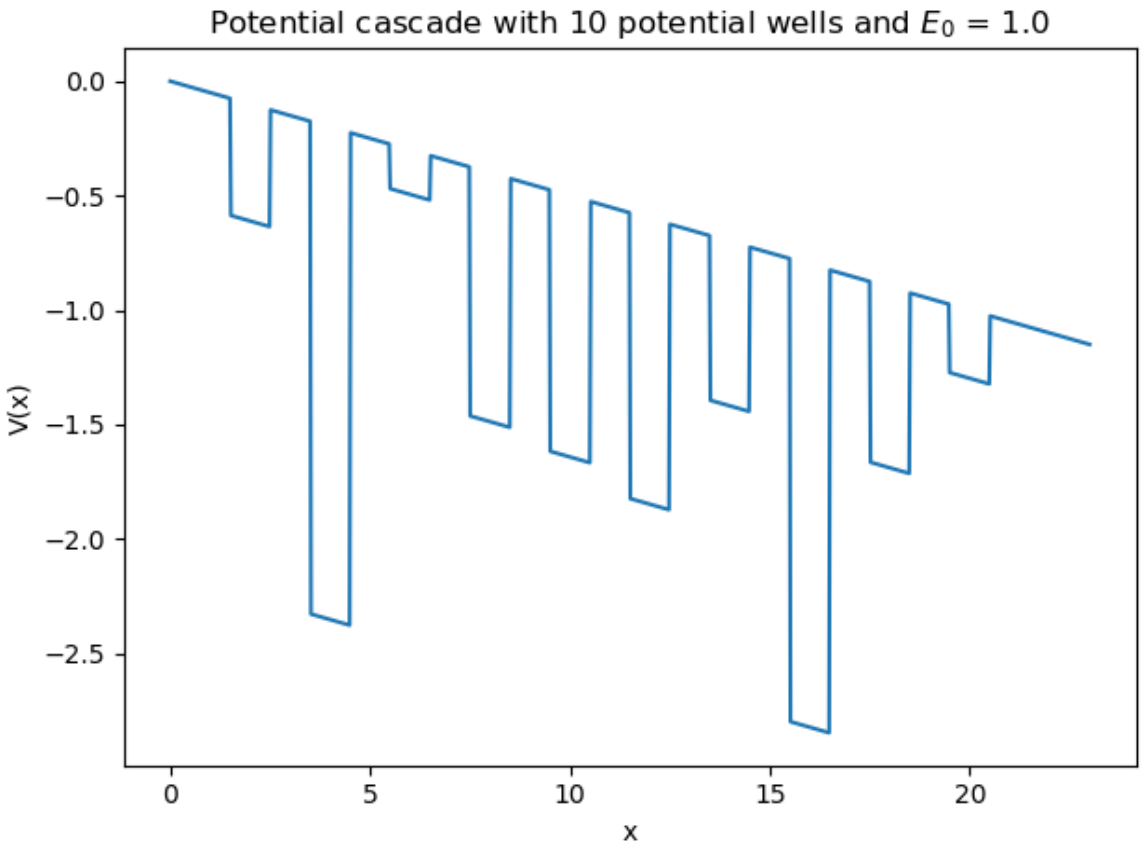
$$P(V) = \frac{1}{E_0} e^{-\frac{V}{E_0}}$$

where E_0 is the average depth.

In [370]:
```
N = 500# Maximum number of wells
N_traj = 100     # Number of trajectories

E0 = 1      # Average well depth

# Generating N_traj * N matrix of exponential random variables with
mean E0
V = np.random.exponential(E0, (N_traj, N))
```

Visualising the potential

In [371]:
```
# This code cell is just for visualising the potential and is not g
oing to be used in the simulation.
x = np.linspace(0,23,1000)
def p(x):
    return 1.0*(x>-0.5)-1.0*(x>0.5)
v = -0.05*x + np.sum([-V[0,i]*p(x-2*(i+1)) for i in range(10)],axis
=0)
plt.figure('Washboard potential')
plt.plot(x,v)
plt.title(r'Potential cascade with 10 potential wells and $E_0$ = %
.1f'%E0)
plt.xlabel('x')
plt.ylabel('V(x)')
plt.tight_layout();
```



Escape times

Dew to Kramer's escape rate formula the rate at which particles escape the i_{th} barrier with height V_i is given by

$$r_i = 1/\tau_i = 1/\tau_0 e^{-V_i/k_B T}.$$

Therefor, the time at which the particle reaches the n_{th}, $\tau(n)$ well is given by the cummulative sum of τ_is

$$\tau(n) = \sum_{i=1}^{n} \tau_i$$

We define the μ parameter as

$$\mu = \frac{k_B T}{E_0}$$

We investigate $\mu > 1$ and $\mu < 1$ cases separatly.

Case 1

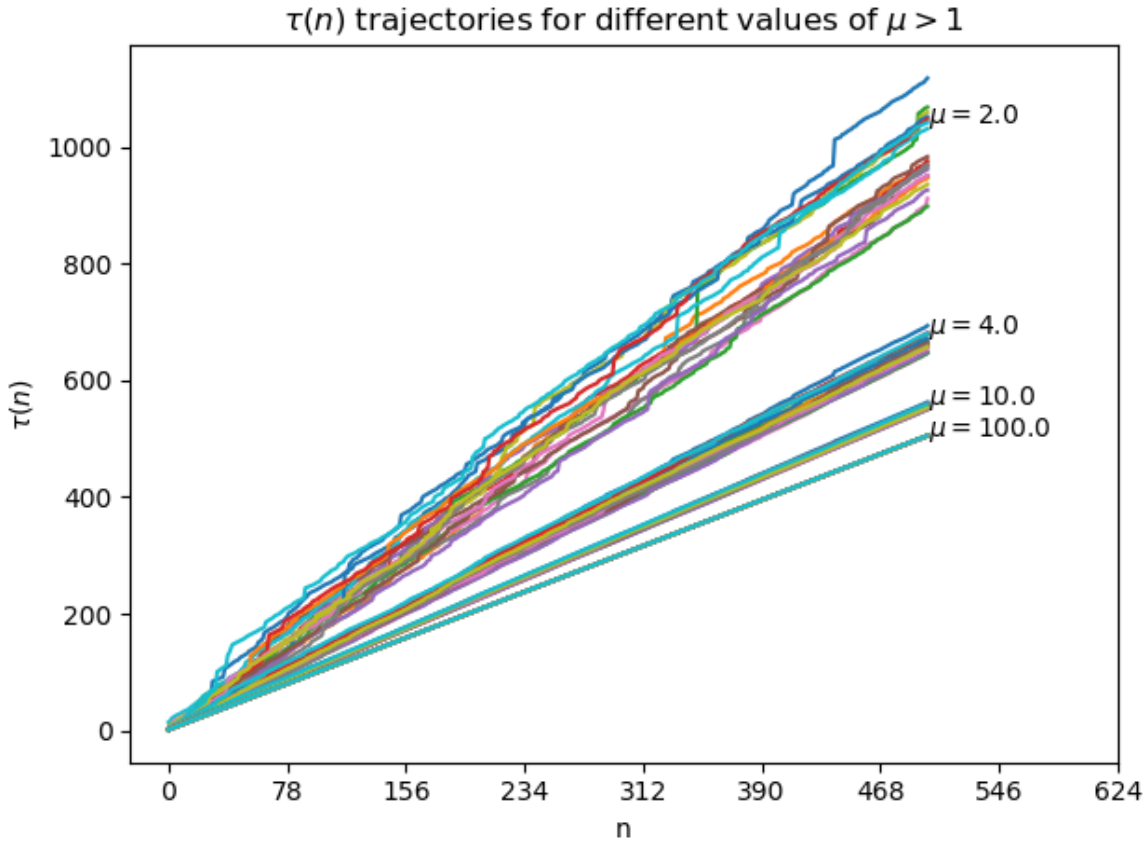
$$\mu > 1$$

in this case the process reaches a stationary state and the average $\langle \tau(n) \rangle$ is given by

$$\langle \tau(n) \rangle = \frac{\mu}{\mu - 1} n \tau_0$$

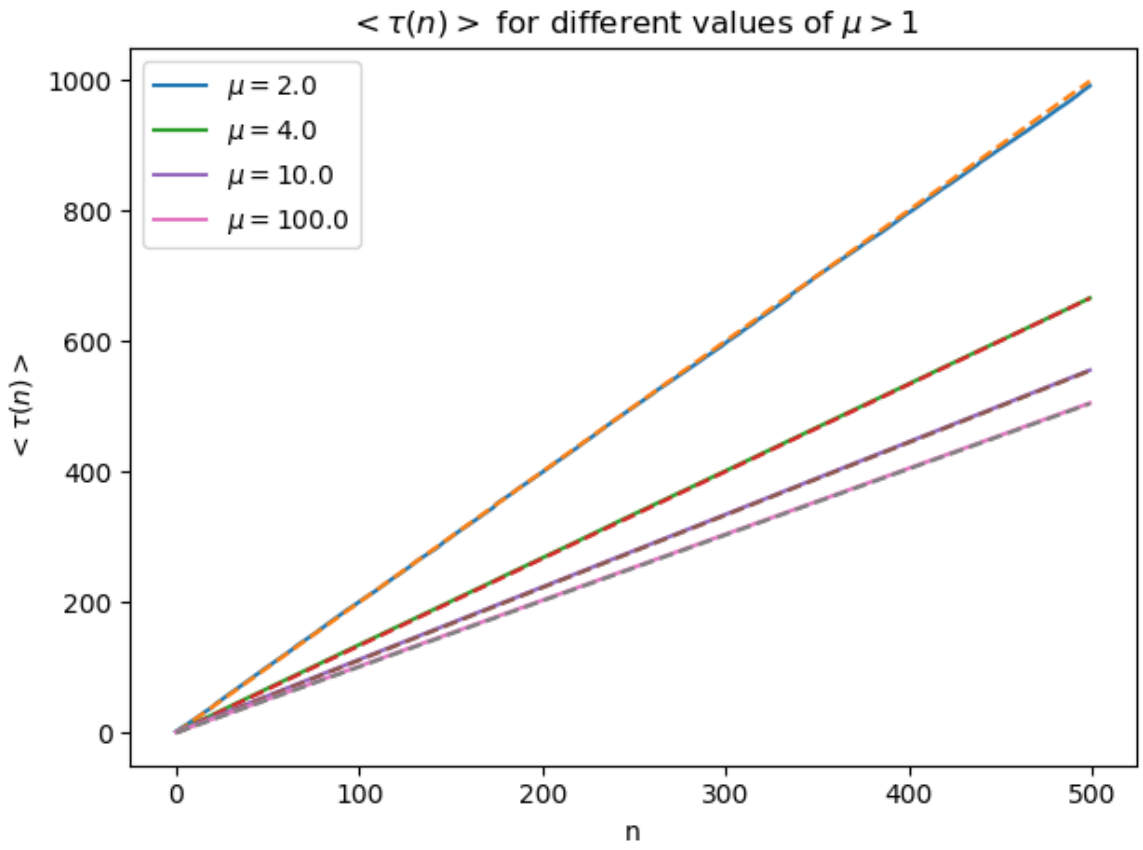
Trajectories

In [372]:
```
tau0 = 1
mu = np.array([2, 4, 10, 100])
kT = E0 * mu*1.0

tau = tau0 *np.array([np.exp(V/kT[i]) for i in range(len(mu))],dtyp
e='float')
tau_n = np.cumsum(tau, axis = 2)

plt.figure('Tau_n')
n = np.arange(N)
for i in range(len(mu)):
    for j in range(20):
        plt.plot(n, tau_n[i,j,:]);
    plt.text(N,tau_n[i,j,N-1],r'$\mu=%.1f$'%mu[i])
plt.xticks(np.arange(0,int(N*1.25),int(N*1.25/8)))
plt.xlabel('n')
plt.ylabel(r'$\tau(n)$')
plt.title(r'$\tau(n)$ trajectories for different values of $\mu>1$'
)
plt.tight_layout();
```



Average escape time

In [373]:
```
Meantau_n = np.mean(tau_n, axis=1)    # The average escape time aft
er n steps over all trajectories

n = np.arange(N)
plt.figure('<Tau_n>')
for i in range(len(mu)):
    plt.plot(n, Meantau_n[i,:],label = r'$\mu=%.1f$'%mu[i])
    plt.plot(n,n*tau0*mu[i]/(mu[i]-1),'--')   # The theory curve wi
th the linear relation
plt.legend()
plt.xlabel('n')
plt.ylabel(r'$<\tau(n)>$')
plt.title(r'$<\tau(n)>$ for different values of $\mu>1$')
plt.tight_layout();
```



As we can see above, the average values fit with the theory curves (in dashed lines) $\langle \tau(n) \rangle = \frac{\mu}{\mu-1} n\tau_0$.

Case 2

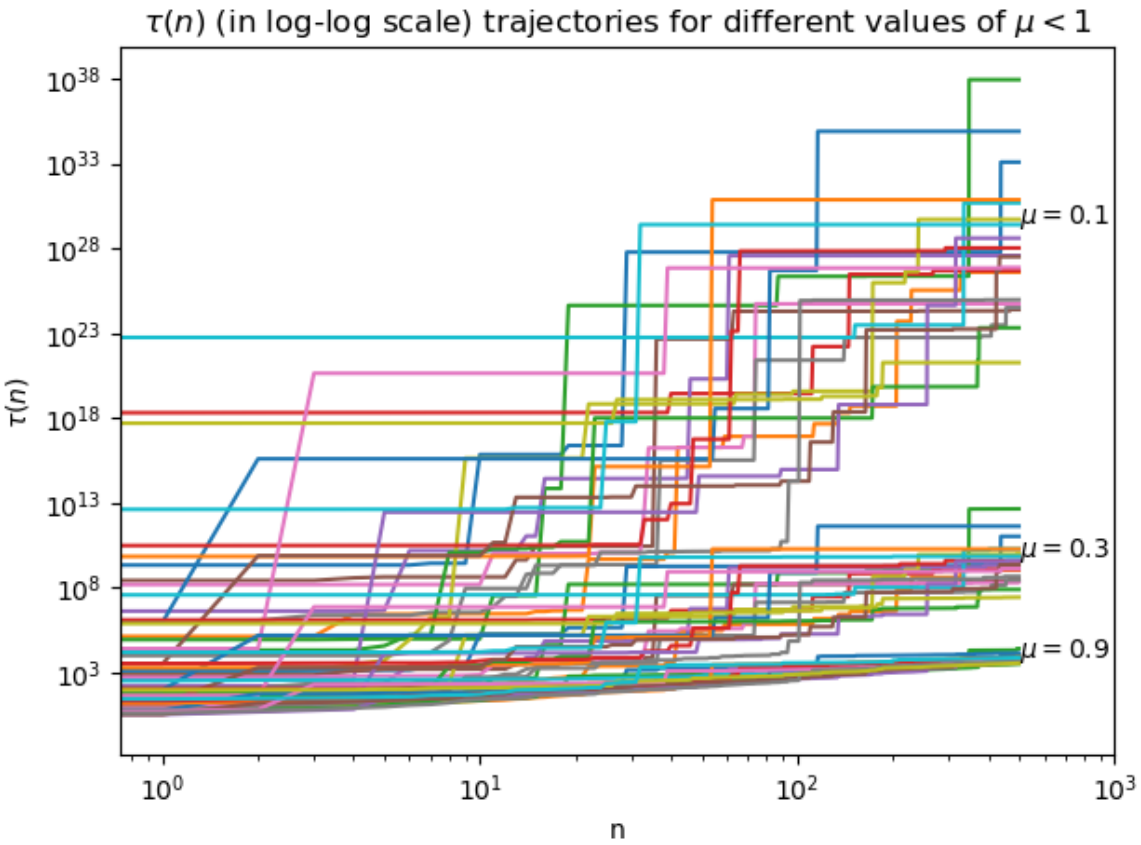
$$\mu < 1$$

in this case the process becomes a Levy flight and the average $\langle \tau(n) \rangle$ does grows with the relation

$$\langle \tau(n) \rangle \sim n^{1/\mu}$$

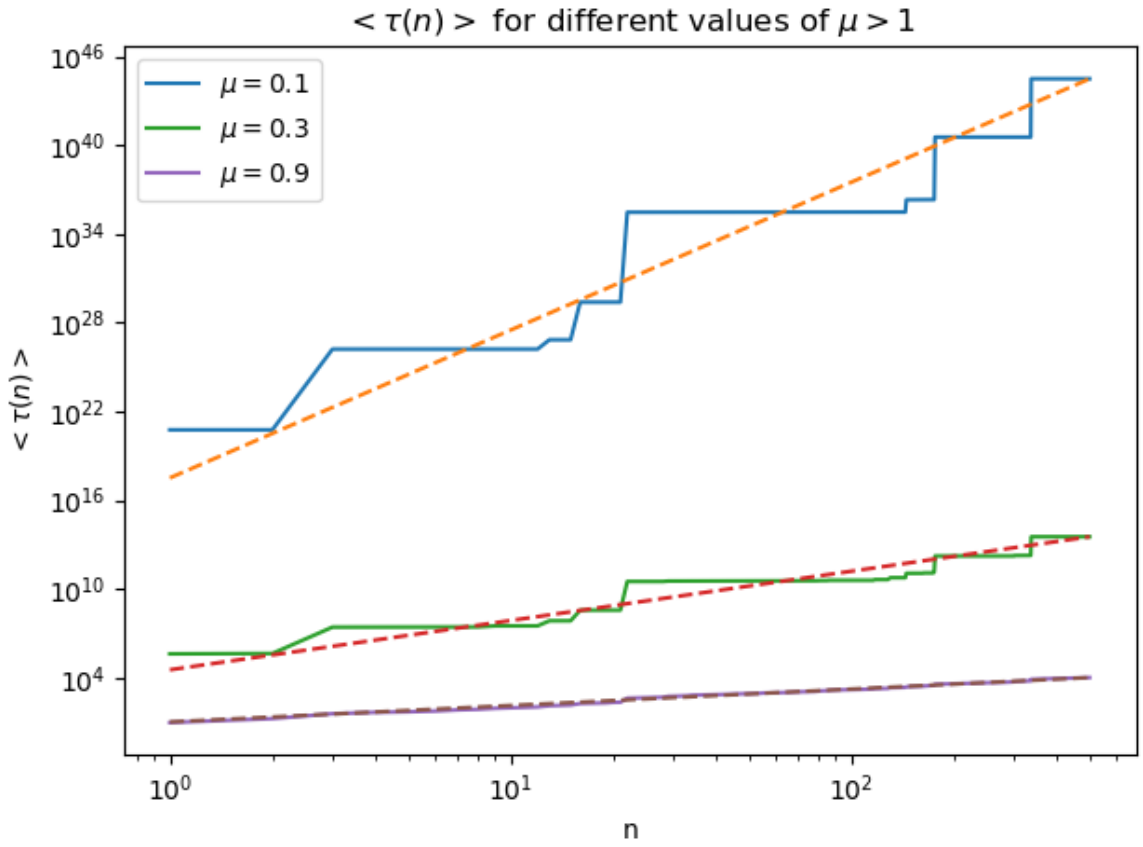
Trajectories

In [374]:
```
tau0 = 1
mu = np.array([ .1,.3,0.9])
kT = E0 * mu*1.0

tau = tau0 *np.array([np.exp(V/kT[i]) for i in range(len(mu))],dtyp
e='float')
tau_n = np.cumsum(tau, axis = 2)

plt.figure('Tau_n_Levy')
n = np.arange(N)
for i in range(len(mu)):
    for j in range(20):
        plt.loglog(n, tau_n[i,j,:]);
    plt.text(N,tau_n[i,j,N-1],r'$\mu=%.1f$'%mu[i])
plt.xticks(np.logspace(0,np.log10(N*2),4))
plt.xlabel('n')
plt.ylabel(r'$\tau(n)$')
plt.title(r'$\tau(n)$ (in log-log scale) trajectories for different
values of $\mu<1$')
plt.tight_layout();
```



Average escape time

In [377]:
```
Meantau_n = np.mean(tau_n, axis=1)    # The average escape time aft
er n steps over all trajectories

n = np.arange(1,N+1)
plt.figure('<Tau_n_Levy>')
for i in range(len(mu)):
    plt.loglog(n, Meantau_n[i,:],label = r'$\mu=%.1f$'%mu[i])
    # The theory curve with the power law relation
    # We normalize to maximum value of the numeric curve to be able
to compare the slope in loglog scale
    plt.loglog(n,Meantau_n[i,-1]*n**(1.0/mu[i])/(N**(1.0/mu[i])),'-
-')
plt.legend()
plt.xlabel('n')
plt.ylabel(r'$<\tau(n)>$')
plt.title(r'$<\tau(n)>$ for different values of $\mu>1$')
plt.tight_layout();
```



As we can see above, the average values fit with the theory curves (in dashed lines) $\langle \tau(n) \rangle \sim n^{1/\mu}$.

7.2 Python Code

```
    # Python code example
    import numpy as np
import matplotlib.pylab as plt
import scipy.optimize as opt

E_0 = 1
tau_0 = 1
mu = np.array([0.3,0.5,0.8,0.99,1.01,1.5,2,10])
colours = np.array(['blue','red','yellow','orange','purple','brown','
    black','green'])

length = 200
sim_num = 1000


def potential(n):
    return np.random.exponential(E_0,n)

def escape_time(x,MU):
    return np.exp(x/(E_0*MU))*tau_0

tau = np.zeros((sim_num,length,len(mu)))
tau_avrg = np.zeros((length,len(mu)))
for j in range(len(mu)):
    MU = mu[j]
    for i in range(sim_num):
        tau[i,:,j]  = np.cumsum(escape_time(potential(length),MU))
for l in range(length):
    tau_avrg[l,:] = np.mean(tau[:,l,:],axis=0)

plt.figure()
for i in range(len(mu)):
    MU = mu[i]
    if MU > 1:
        plt.plot(tau[0,:,i],label=r'$\mu = $'+str(MU),color = colours[i
            ])
plt.legend(loc='best')
plt.xlabel(r'$n$')
plt.ylabel(r'$\tau(n)$')
plt.title('Typical escape times as a function of n')
plt.show()
plt.close()

plt.figure()
for i in range(len(mu)):
    MU = mu[i]
    if MU < 1:
        plt.semilogy(tau[0,:,i],label=r'$\mu = $'+str(MU),color =
            colours[i])
plt.legend(loc='best')
plt.xlabel(r'$n$')
plt.ylabel(r'$\tau(n)$')
plt.title('Typical escape times as a function of n')
plt.show()
plt.close()

plt.figure()
```

```
for i in range(len(mu)):
    MU = mu[i]
    if MU > 1:
        plt.plot(tau_avrg[:,i],label=r'$\mu = $ '+str(MU),color =
            colours[i])
plt.legend(loc='best')
plt.xlabel(r'$n$')
plt.ylabel(r'$<\tau(n)>$')
plt.title('Average escape times as a function of n')
plt.show()
plt.close()

plt.figure()
for i in range(len(mu)):
    MU = mu[i]
    if MU < 1:
        plt.semilogy(tau_avrg[:,i],label=r'$\mu = $ '+str(MU),color =
            colours[i])
        func = lambda x,a: a*x**(1/MU)
        n = np.arange(0,length)
        c = opt.curve_fit(func,n,tau_avrg[:,i])[0]
        plt.semilogy(n,c*n**(1/MU),'--',color = colours[i])
plt.legend(loc='best')
plt.xlabel(r'$n$')
plt.ylabel(r'$<\tau(n)>$')
plt.title('Average escape times as a function of n, and their
    asymptotic scaling')
plt.show()
plt.close()
```