计算物理第七次作业

近代物理系 张浩然 SA21004048

Question 1 ▷ Gumbel Distribution

For random variables $x_1, x_2, \cdots, x_n$ $(x_1 < x_2 < \cdots < x_n)$, find the distribution of

$$\Delta = \max\{x_{i+1} - x_i\}.$$

Solution

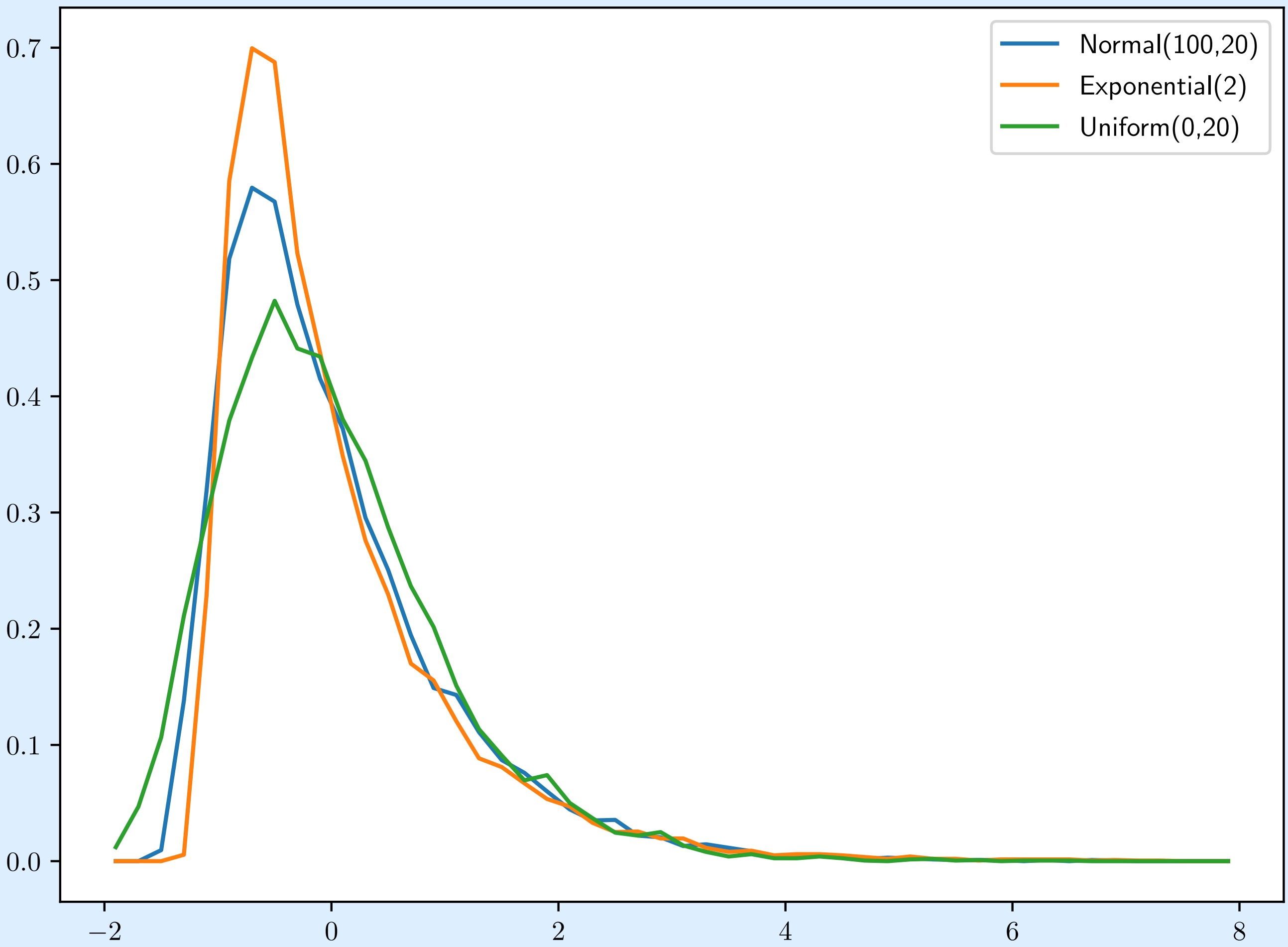
使用Python分别计算$\{x_i\}$分别服从正态分布$N(100, 20)$、指数分布$Exp(2)$和均匀分布$U(0, 20)$的Δ分布, 并按照

$$\Delta' = \frac{\Delta - \langle\Delta\rangle}{\sigma_\Delta} \tag{1}$$

进行标准化, 画出标准化后的Δ'分布:

```python
import numpy as np
from numpy import random
import matplotlib.pyplot as plt

def normalize(x):
    return ( np.array(x) - np.mean(x) ) / np.std(x)

Nx = int(1e4)
Ns = int(1e5)

spacing_expo = [np.max(np.diff(np.sort(random.exponential(2, Nx)))) for i in range(Ns)]
spacing_norm = [np.max(np.diff(np.sort(random.normal(100, 20, Nx)))) for i in range(Ns)]
spacing_unif = [np.max(np.diff(np.sort(20*random.rand(Nx)))) for i in range(Ns)]

nbins=50
hist1, binedges = np.histogram(normalize(spacing_norm),nbins,range=(-2,8),density=True)
hist2, binedges = np.histogram(normalize(spacing_expo),nbins,range=(-2,8),density=True)
hist3, binedges = np.histogram(normalize(spacing_unif),nbins,range=(-2,8),density=True)
bins_mean = [0.5 * (binedges[i] + binedges[i+1]) for i in range(nbins)]

plt.plot(bins_mean,hist1,label='Normal(100,20)');
plt.plot(bins_mean,hist2,label='Exponential(2)');
plt.plot(bins_mean,hist3,label='Uniform(0,20)');
```

得到的数据如下图所示:



Gumble分布的表达式为

$$f(x;\mu,\beta)=\frac{1}{\beta}\mathrm{e}^{-(z)+\mathrm{e}^{-z}},\quad \text{where}\quad z=\frac{x-\mu}{\beta}. \tag{2}$$

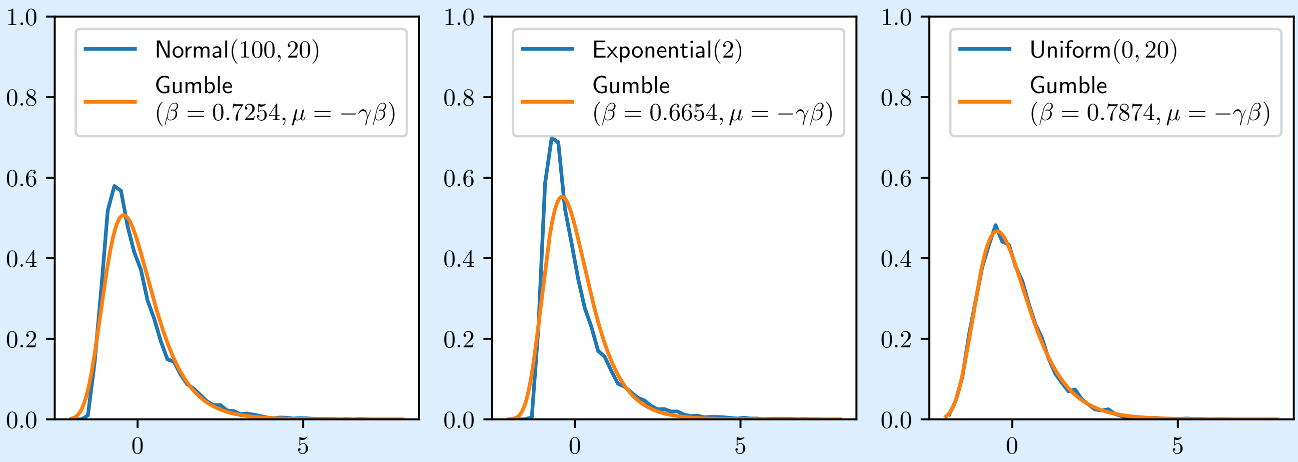
其中μ是众数, 与均值$\langle x\rangle$的关系为

$$\langle x\rangle=\mu+\gamma\beta,\quad \text{where}\quad \gamma\simeq 0.5772\ \text{(Euler–Mascheroni constant)}. \tag{3}$$

由于上面的过程中我们已经进行了标准化使得Δ'的均值为0, 故要使用均值为0的Gumble分布进行拟合. 均值为0的Gumble分布有$\mu=-\gamma\beta$. 使用python进行拟合, 并作出拟合后的Gumble分布与Δ'的分布的对比图.

```
def gumble(x,beta):
    mu = -.5772 * beta # mean = mu + 0.5772 beta = 0
    z = (x - mu) / beta
    return np.exp( -(z + np.exp(-z) ) ) / beta
from scipy import optimize
beta1 = optimize.curve_fit(gumble, bins_mean[:nbins//2], hist1[:nbins//2])[0][0]
beta2 = optimize.curve_fit(gumble, bins_mean[:nbins//2], hist2[:nbins//2])[0][0]
beta3 = optimize.curve_fit(gumble, bins_mean[:nbins//2], hist3[:nbins//2])[0][0]
xlst = np.linspace(-2,8,1000)
fig,axes=plt.subplots(1,3,figsize=(9,3),dpi=200)
for i in range(len(axes)):
    axes[i].set_ylim(0,1)
ax = axes[0].
```

```python
ax.plot(bins_mean,hist1,label=r'Normal$(100,20)$');
ax.plot(xlst,gumble(xlst,beta1),label='Gumble\n'+r'$(\beta=%.4f,\mu=-\gamma\beta)$'%beta1)
ax.legend(loc='upper right')
ax = axes[1]
ax.plot(bins_mean,hist2,label='Exponential$(2)$');
ax.plot(xlst,gumble(xlst,beta2),label='Gumble\n'+r'$(\beta=%.4f,\mu=-\gamma\beta)$'%beta2)
ax.legend(loc='upper right')
ax = axes[2]
ax.plot(bins_mean,hist3,label='Uniform$(0,20)$');
ax.plot(xlst,gumble(xlst,beta3),label='Gumble\n'+r'$(\beta=%.4f,\mu=-\gamma\beta)$'%beta3)
ax.legend(loc='upper right')
plt.savefig('07q1-f2.pdf',dpi=200,bbox_inches='tight')
```



Question 2 ▷ Langevin Equation

For $U(x) = ax^2 + bx^4\,(a < 0),\; m = 1$, solve

$$m\ddot{x} = -\alpha\dot{x} - \nabla U + \xi(t),$$

where

$$\langle \xi(t) \rangle = 0, \quad \langle \xi(t)\xi(t') \rangle = D\delta(t - t').$$

Solution

运动方程为

$$\begin{aligned} \dot{x} &= v, \\ \dot{v} &= -\alpha - 2ax - 4bx^3 - \xi(t). \end{aligned} \tag{4}$$

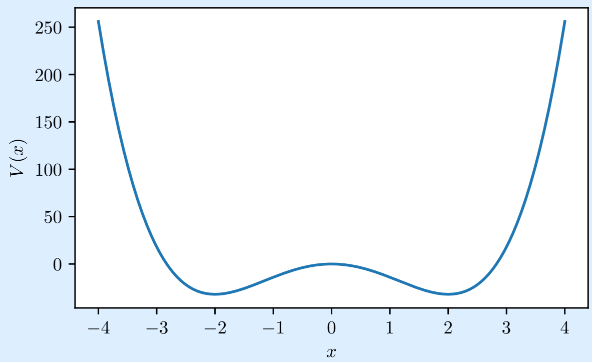
离散化后为

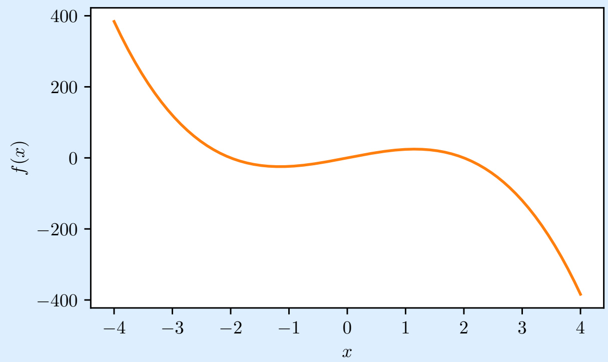
$$\begin{aligned} x_{n+1} &= x_n + v_n \mathrm{d}t, \\ v_{n+1} &= v_n + \left[-\alpha v_n + f(x_n) + \sqrt{\frac{D}{\mathrm{d}t}}\xi_n \right] \mathrm{d}t, \end{aligned} \tag{5}$$

with

$$f(x) = -2ax^2 - 4bx^4 \qquad \text{and} \qquad \xi_n \sim N(0, 1). \tag{6}$$

取 $a = -16b = 2$, $V(x), f(x)$ 的图像如下图所示, 两个平衡位置为 $\pm x_b$ $(x_b = 2)$



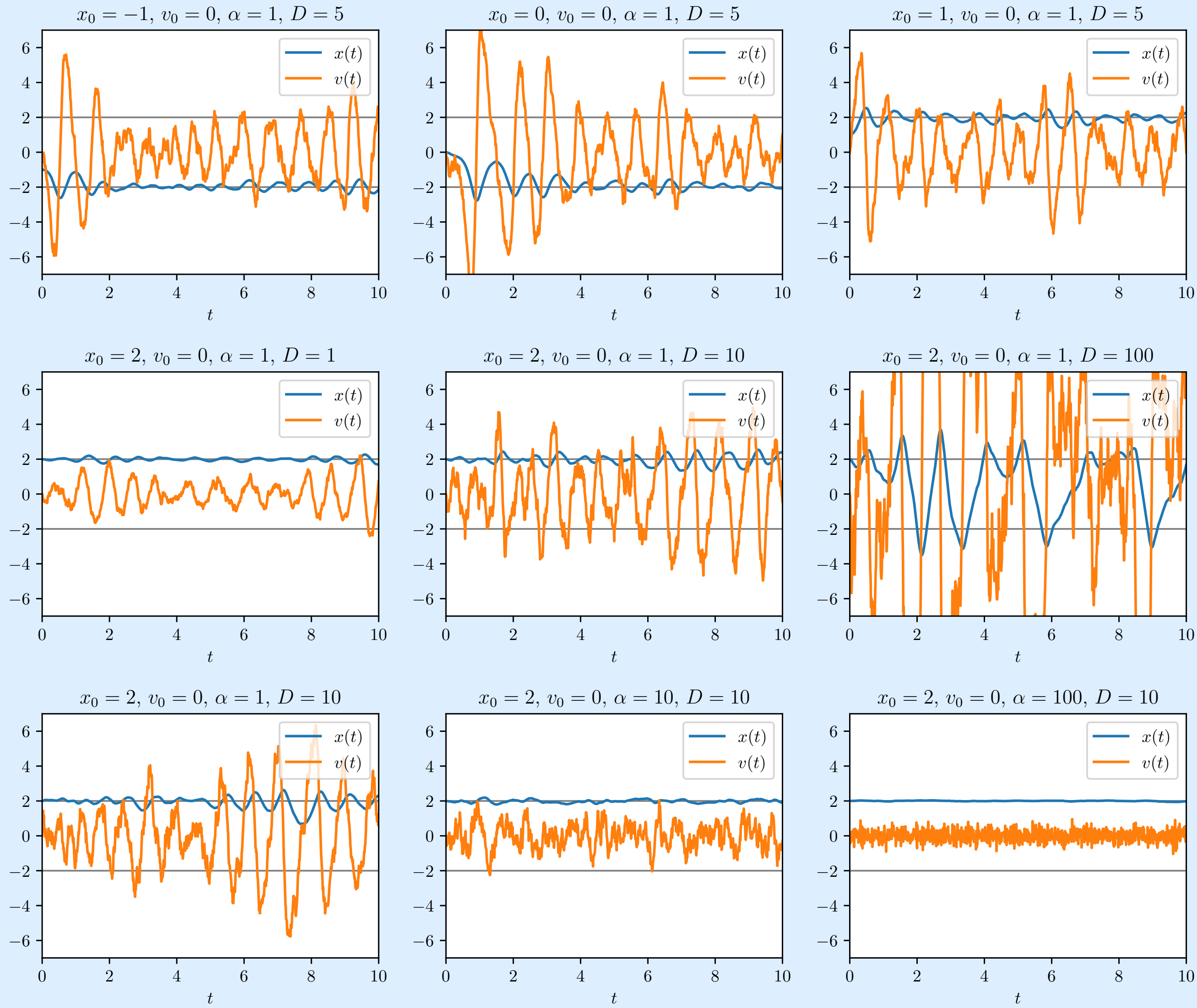


使用python进行计算, 代码如下:

```python
import numpy as np
from numpy import random
import matplotlib.pyplot as plt

a = -16; b = 2;

def f(x):
    return - 2*a*x - 4*b*(x**3)

def evolve(x0, v0, alpha, D):
    xi = random.normal(size = N)
    x = x0; v = v0;
    xs = [x]; vs = [v];
    for i in range(N):
```

```python
        x += v * dt
        v += ( - alpha*v + f(x) + (D/dt)**.5 * xi[i] ) * dt
        xs.append(x)
        vs.append(v)
    return xs, vs

fig,axes=plt.subplots(3,3,figsize=(12,10),dpi=200)

v0 = 0; alpha = 1; D = 5;
for i in range(3):
    ax = axes[0,i]
    x0 = 1 * (i-1)
    xs, vs = evolve(x0, v0, alpha, D)
    ax.plot(ts,xs,label='$x(t)$');
    ax.plot(ts,vs,label='$v(t)$');

x0 = 2; v0 = 0; alpha = 1;
for i in range(3):
    ax = axes[1,i]
    D = 10**i
    xs, vs = evolve(x0, v0, alpha, D)
    ax.plot(ts,xs,label='$x(t)$');
    ax.plot(ts,vs,label='$v(t)$');

x0 = 2; v0 = 0; D = 10;
for i in range(3):
    ax = axes[2,i]
    alpha = 10**i
    xs, vs = evolve(x0, v0, alpha, D)
    ax.plot(ts,xs,label='$x(t)$');
    ax.plot(ts,vs,label='$v(t)$');
```

计算结果如下图所示:



首先固定其他参数而改变初始位置 x_0. 可以看到, 当 D 不太大时, 在两个势阱的其中一个中释放粒子, 粒子将在这个阱中运动. 而当在 $x = 0$ 处静止释放粒子时, 粒子则会随机地进入其中一个势阱.

然后固定其他参数而改变随机力大小 D. 当 D 不太大时, 粒子将始终在初始时刻所在的势阱中运动, 且震荡幅度随 D 增大而增大. 而当 D 足够大时, 粒子将有可能在随机力的作用下在两个势阱之间跃迁.

最后固定其他参数而改变阻尼大小 α. 可见粒子的震荡幅度随 α 增大而减小.