truncated 2D exponential

1) $f_{X,Y}(x,y) = \begin{cases} c\exp(-\lambda xy) & , (x,y) \in \underbrace{[0,A]}_{\Omega_x} \times \underbrace{[0,B]}_{\Omega_y} \\ 0 & , \text{otherwise} \end{cases}$

state space for MC but image of rvs

a) $f(x) = \int_{y \in \Omega_Y} f(x,y)\,dy = \int_0^B c\, e^{-\lambda xy} dy = c\left(-\frac{1}{\lambda x}\right) e^{-\lambda xy}\Big|_{y=0}^{B} = -c\frac{1}{\lambda x}\left(e^{-\lambda xB} - 1\right)$

$(x \in [0,A]]$

$$\Rightarrow \boxed{f(x) = c\,\frac{1-e^{-\lambda Bx}}{\lambda x} \;,\; x \in [0,A]}$$

By symmetry, $\boxed{f(y) = c\,\frac{1-e^{\lambda Ay}}{\lambda y} \;,\; y \in [0,B]}$

b) $f(x|y_0) = \frac{f(x,y_0)}{f(y_0)} = \frac{c\,e^{-\lambda xy_0}}{c\,\frac{1-e^{\lambda Ay_0}}{\lambda y_0}} = \boxed{\frac{\lambda y_0\, e^{-\lambda xy_0}}{1-e^{\lambda Ay_0}} \;,\; (x,y_0) \in [0,A]\times[0,B]}$

else conditional is meaningless

$f(y|x_0) = \frac{f(x_0,y)}{f(x_0)} = \frac{c\,e^{-\lambda x_0 y}}{c\,\frac{1-e^{\lambda Bx_0}}{\lambda x_0}} = \boxed{\frac{\lambda x_0\, e^{-\lambda x_0 y}}{1-e^{\lambda Bx_0}} \;,\; (x_0,y) \in [0,A]\times[0,B]}$

c) $F(x|y_0) = \mathbb{P}(X \le x \mid Y = y_0) = \int_0^x f(t|y_0)\,dt = \frac{\lambda y_0}{1-e^{\lambda Ay_0}} \int_0^x e^{-\lambda y_0 t} = \frac{\lambda y_0}{1-e^{\lambda Ay_0}} \cdot \frac{1}{-\lambda y_0} e^{-\lambda y_0 t}$

$= \frac{e^{-\lambda y_0 t}}{e^{\lambda Ay_0} - 1}\Big|_{t=0}^{x}$

$$\Rightarrow F(x|y_0) = \begin{cases} 0 & , x < 0 \\ \frac{e^{-\lambda y_0 x} - 1}{e^{\lambda Ay_0} - 1} & , 0 \le x \le A \\ 1 & , x > A \end{cases} \;,\; 0 \le y_0 \le B$$

By symmetry, $F(y|x_0) = \begin{cases} 0 & , y < 0 \\ \frac{e^{-\lambda x_0 y} - 1}{e^{\lambda Bx_0} - 1} & , 0 \le y \le B \\ 1 & , y > B \end{cases} \;,\; 0 \le x \le A$

d) Inverse transform thm:

$X \sim f(x)$ with CDF $F(x)$

$$\Rightarrow F^{-1}(U) = Y \sim f(x) \qquad (F(Y) = U)$$

where $U \sim Unif(0,1)$

$$F(x \mid y_0) = \frac{1-e^{-\lambda x y_0}}{1-e^{-\lambda y_0 A}} \Rightarrow F_{X|y_0}(W) = \frac{1-e^{-\lambda W y_0}}{1-e^{-\lambda y_0 A}} = U$$

$$\Rightarrow 1-e^{-\lambda W y_0} = U(1-e^{-\lambda y_0 A}) \Rightarrow e^{-\lambda W y_0} = 1-U(1-e^{-\lambda y_0 A})$$

$$\Rightarrow -\lambda W y_0 = \ln\left[1-U(1-e^{-\lambda y_0 A})\right]$$

$$\Rightarrow F^{-1}_{X|y_0}(U) = \frac{-\ln\left[1-U(1-e^{-\lambda y_0 A})\right]}{\lambda y_0} = W \sim f(x \mid y_0)$$

By symmetry, $F^{-1}_{Y|x_0}(U) = \frac{-\ln\left[1-U(1-e^{-\lambda x_0 B})\right]}{\lambda x_0} = V \sim f(y \mid x_0)$

Algo:

1) Sample $U_1, U_2 \overset{iid}{\sim} Unif(0,1)$
2) Calculate $W \sim f(x|y_0) = F^{-1}_{X|y_0}(U_1)$ and $V \sim f(y|x_0) = F^{-1}_{y|x_0}(U_2)$
3) Return (w, v)

e)

Algo:

1) Generate $X_0 \sim Unif(0, A)$ and $Y_0 \sim Unif(0, B)$
2) For $i = 0, \ldots, N-1$:
   a) Generate $X_{i+1} \sim f(x \mid y_i)$
   b) Generate $Y_{i+1} \sim f(y \mid x_{i+1})$

   sampled values
3) Return $\{(X_i, y_i)\}_{i=0}^{N}$

Gibbs_HW

March 11, 2025

[24]:
```
import numpy as np
import matplotlib.pyplot as plt
```

0.1 Sampling from a 2D exponential distribution using Gibbs sampling

In this assignment, you'll sample from the joint pdf

$$f_{X,Y}(x,y) = \begin{cases} c\exp(-\lambda xy), & \text{for } (x,y) \in [0, \text{xlim}] \times [0, \text{ylim}], \\ 0, & \text{otherwise.} \end{cases}$$

You'll need to use the equations you've derived in the written assignment to construct the Gibbs sampler. Start by defining methods in the `GibbsSampler` class to sample from the conditional distributions. Then, implement the `rvs` method to run the Gibbs sampler. The only source of randomness should be `rng.random()`. You should implement your own *inverse transform sampling method* to sample from the truncated exponential distribution.

[25]:
```
class GibbsSampler:
    def __init__(self, lambda_, xlim, ylim, seed=0):
        self.lambda_ = lambda_
        self.xlim = xlim
        self.ylim = ylim
        # It's a good practice to use a separate random number generator for␣
↪each class
        # use self.rng.random() instead of np.random.uniform() to generate␣
↪random numbers
        self.rng = np.random.default_rng(seed)

    def X_given_Y(self, x, y0):
        """ Returns a single sample from the conditional distribution of X␣
↪given Y=y0 """
        u = self.rng.random()
        return -np.log(1 - u * (1 - np.exp(-self.lambda_ * y0 * self.xlim))) /␣
↪(self.lambda_ * y0)

    def Y_given_X(self, y, x0):
        """ Returns a single sample from the conditional distribution of Y␣
↪given X=x0 """
        u = self.rng.random()
```

```
        return -np.log(1 - u * (1 - np.exp(-self.lambda_ * x0 * self.ylim))) /␣
↪(self.lambda_ * x0)

    def rvs(self, size=1):
        """ Returns samples of size `size` from the joint distribution of X and␣
↪Y """
        x, y = np.zeros(size), np.zeros(size)
        x[0], y[0] = self.rng.random() * self.xlim, self.rng.random() * self.
↪ylim
        for i in range(0, size - 1):
            x[i+1] = self.X_given_Y(None, y[i]) # doesn't depend on current x
            y[i+1] = self.Y_given_X(None, x[i+1]) # doesn't depend on current y
        return x, y
```

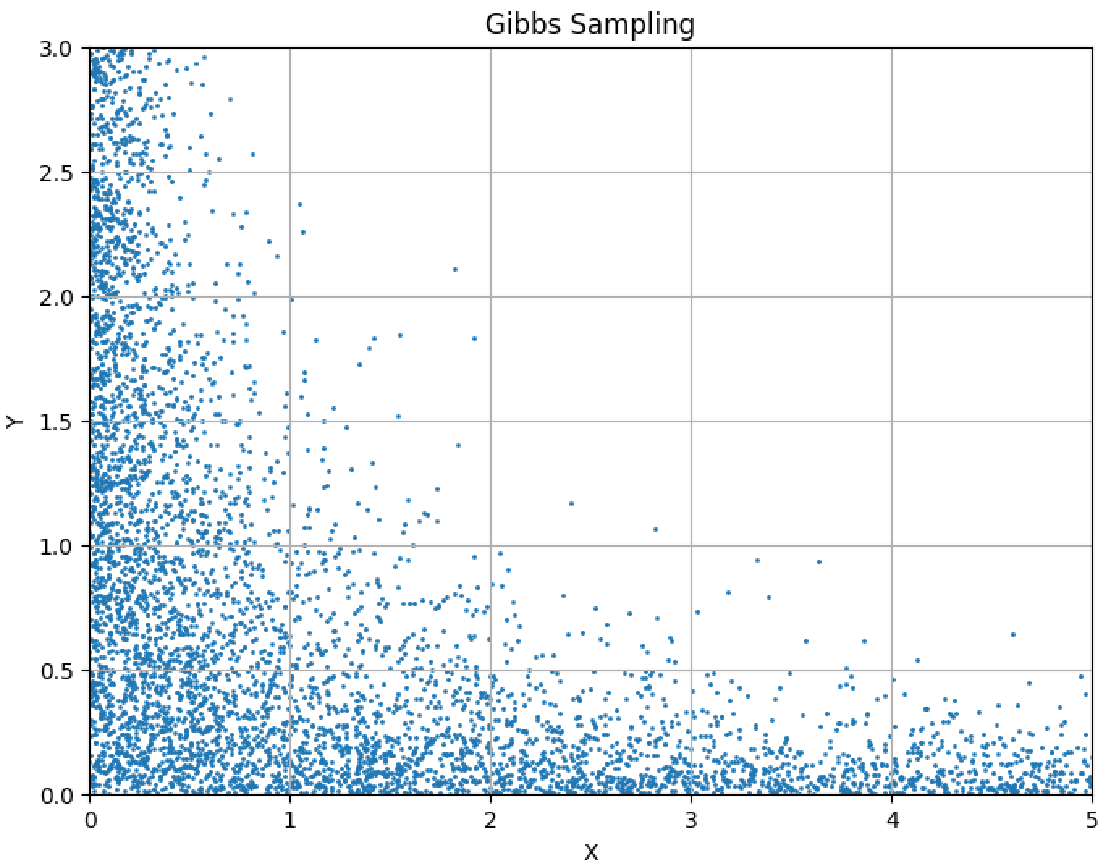
0.2 Trace Plots

Generate 5000 samples with the parameters in the cell below and make a scatter plot of the samples.

[26]:
```
N = 5000
# Parameters
lambda_, xlim, ylim = 2, 5, 3
sampler = GibbsSampler(lambda_, xlim, ylim)

x, y = sampler.rvs(N)

# scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(x, y, s=1)
plt.xlim(0, xlim)
plt.ylim(0, ylim)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Gibbs Sampling')
plt.grid()
plt.show()
```



0.3 Autocorrelation

Compute the autocorrelation of the samples for both X and Y with lags between 1 and 20. Plot the autocorrelation functions. Use `np.corrcoef` to compute the correlation matrix and extract the correlation between X and Y. Note that `np.correlate` is not the same as the correlation coefficient. This should (hopefully) show that there is no need to thin the samples or burn any of the initial samples in this case. See also: http://users.stat.umn.edu/~geyer/mcmc/burn.html

[27]:
```
lags = np.arange(1, 20, 1)

corrX = np.zeros(len(lags))
corrY = np.zeros(len(lags))

for i, lag in enumerate(lags):
    corrmatX = np.corrcoef(x[:N-lag], x[lag:N])
    corrmatY = np.corrcoef(y[:N-lag], y[lag:N])
    corrX[i] = corrmatX[0, 1]
    corrY[i] = corrmatY[0, 1]

plt.figure(figsize=(8, 6))
```

```
plt.plot(lags, corrX, label='X')
plt.plot(lags, corrY, label='Y')
plt.legend()
plt.xlabel('Lag')
plt.ylabel('Correlation')
plt.title('Autocorrelation of X and Y with Lag')
plt.grid()
plt.show()
```