AM783: Applied Markov processes | Statistical or Monte Carlo estimation

Hugo Touchette

13 August 2020

Last updated: 11 August 2022

Python 3

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [6]:
```
# Magic command for vectorised figures
%config InlineBackend.figure_format = 'svg'
```

Point estimator

We want to estimate the mean $E[R]$ of a sample of values known to be distributed from the Rayleigh distribution

$$p(r) = re^{-r^2/2}.$$

For this, we generate a sample of L values and calculate the estimator

$$\hat{\mu}_L = \frac{1}{L}\sum_{i=1}^{L} X_i.$$

By increasing L, the estimator should get closer to the theoretical expectation, which is known to be

$$E[R] = \int_0^\infty r\,p(r)\,dr = \sqrt{\frac{\pi}{2}}.$$

In [24]:
```
L = 10**6
sample = np.random.rayleigh(1, L)  # Using the built-in Rayleigh generato
mean_est = np.mean(sample)
print('Estimator value = ', mean_est)
print('Theoretical value = ', np.sqrt(np.pi/2))
```

```
Estimator value =  1.2525155110475363
Theoretical value =  1.2533141373155001
```

Repeating the calculation, we indeed see that estimated values get close to the theoretical expectation.

Gaussian error bars

From the sample used to obtain the mean estimate, we can also calculate the Gaussian error, as shown in the lecture notes.

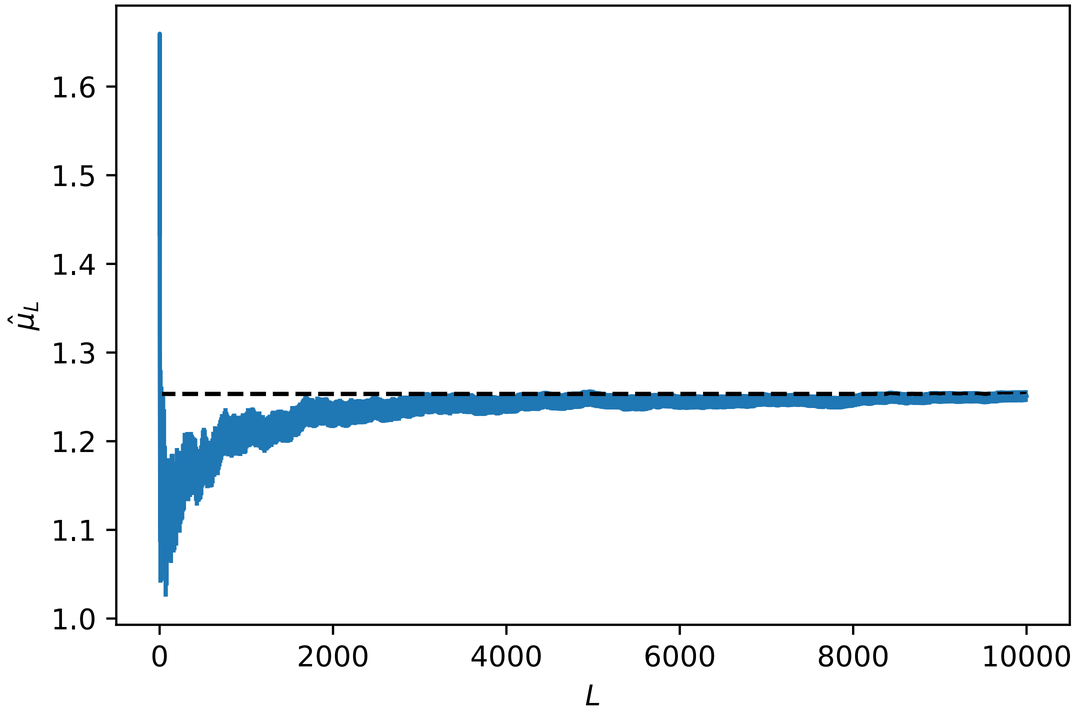
In [3]:
```
L = 10**6
sample = np.random.rayleigh(1, L)  # Using the built-in Rayleigh generato
mean_est = np.mean(sample)
mean_est_err = np.std(sample)/np.sqrt(L)
print('Estimator value = ', mean_est, "±", mean_est_err)
print('Absolute error = ', np.abs(mean_est-np.sqrt(np.pi/2)))
```

```
Estimator value =  1.252569432127885 ± 0.0006548888077747249
Absolute error =  0.0007447051876150468
```

Convergence analysis

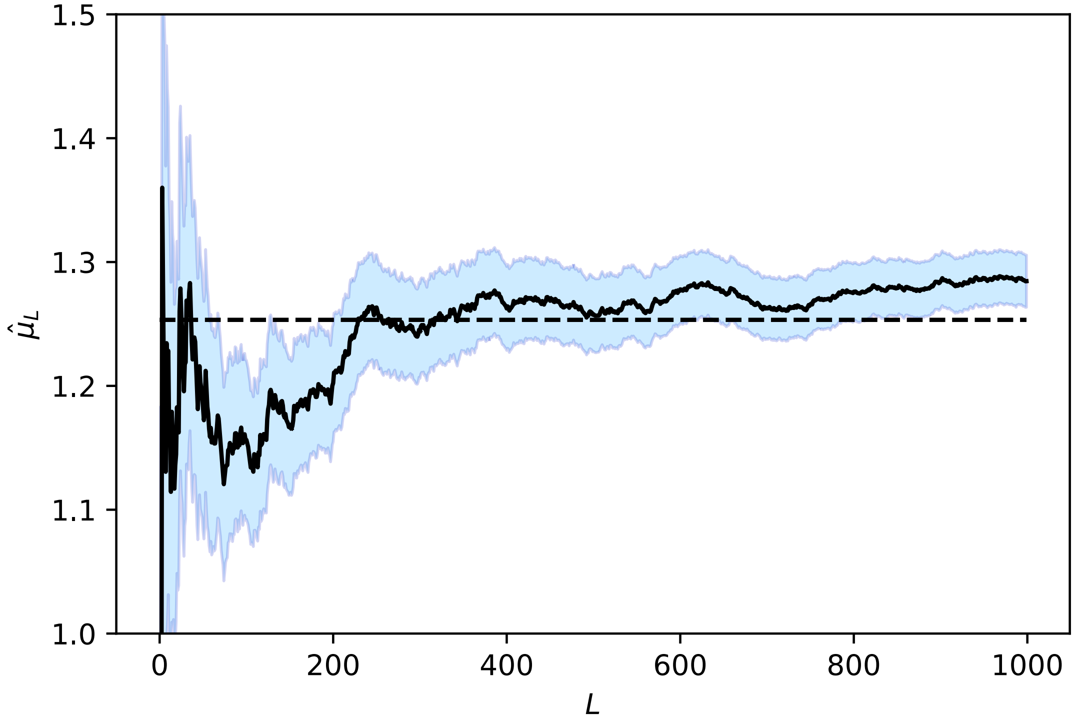
An estimation should be repeated for increasing values of L to show that it converges. The code next does this efficiently without recalculating the estimator and its error for each L.

In [29]:
```
Lmax = 10**4

# Empty lists to receive results
mean_est_list = np.zeros(Lmax)
mean_est_err_list = np.zeros(Lmax)

# Initial values of sums
s = 0.0
v = 0.0

for i in range(Lmax):
    r = np.random.rayleigh(1)
    s += r      # Accumulate the RVs in sum to get mean estimator
    v += r**2   # Accumulate the RVs^2 to get the error bar

    mean_est = s/(i+1.0)      # Mean estimator
    sec_mom_est = v/(i+1.0)   # Second moment estimator
    mean_est_err = np.sqrt(sec_mom_est-mean_est**2) / np.sqrt(i+1.0)   # E

    # Put results in lists
    mean_est_list[i] = mean_est
    mean_est_err_list[i] = mean_est_err

# Plot estimator with error bar
plt.errorbar(range(Lmax), mean_est_list, mean_est_err_list, errorevery=10

# Show theoretical value
plt.plot(range(Lmax), np.zeros(Lmax)+np.sqrt(np.pi/2), 'k--')
plt.xlabel(r'$L$')
plt.ylabel(r'$\hat\mu_L$')
plt.show()
```



Alternatively, we can show the error as a filled plot:

In [26]:
```
Lmax = 10**3
mean_est_list = np.zeros(Lmax)
mean_est_err_list = np.zeros(Lmax)

s = 0.0
v = 0.0

for i in range(Lmax):
    r = np.random.rayleigh(1)
    s += r
    v += r**2

    mean_est = s/(i+1.0)
    sec_mom_est = v/(i+1.0)
    mean_est_err = np.sqrt(sec_mom_est-mean_est**2)/np.sqrt(i+1.0)

    mean_est_list[i] = mean_est
    mean_est_err_list[i] = mean_est_err

plt.plot(range(Lmax), mean_est_list, 'k-')
plt.fill_between(range(Lmax),
                 mean_est_list-mean_est_err_list,
                 mean_est_list + mean_est_err_list,
                 alpha=0.2,
                 edgecolor='#1B2ACC',
                 facecolor='#089FFF')
plt.plot(range(Lmax), np.zeros(Lmax)+np.sqrt(np.pi/2), 'k--')
plt.ylim([1.0, 1.5])
plt.xlabel(r'$L$')
plt.ylabel(r'$\hat\mu_L$')
plt.show()
```



In [ ]: