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

Exercise 1

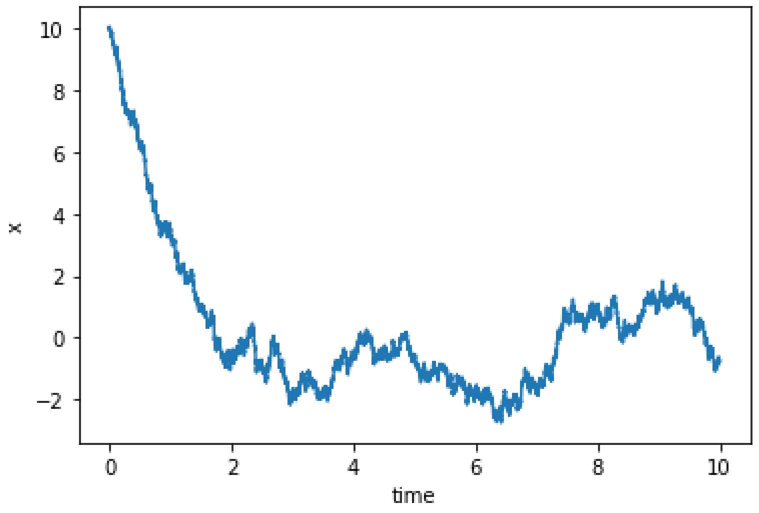
Write a program to integrate the following stochastic differential equation:

dx = −x dt + $\sqrt{2}$dW

Use Euler integrator (i.e. just replace dt with Δt and dW with $\sqrt{\Delta t}$ R where R is a Gaussian number with zero average and unitary variance;choose a short timestep Δt = 0.001).

In [ ]:
```python
# Here we define the integrating function:

def run(x0=10,dt=0.001,seed=None,nsteps=10000):
    if seed is not None:
        np.random.seed(seed)
    x=x0
    traj=[]
    for istep in range(nsteps):
        x+=-x*dt+np.sqrt(2*dt)*np.random.normal()
        traj.append((istep*dt,x))
    return np.array(traj)
```

In [ ]:
```python
# Here we integrate once and plot the result:

traj=run()
plt.plot(traj[:,0],traj[:,1])
plt.xlabel("time")
plt.ylabel("x")
plt.show()
```



Exercise 2

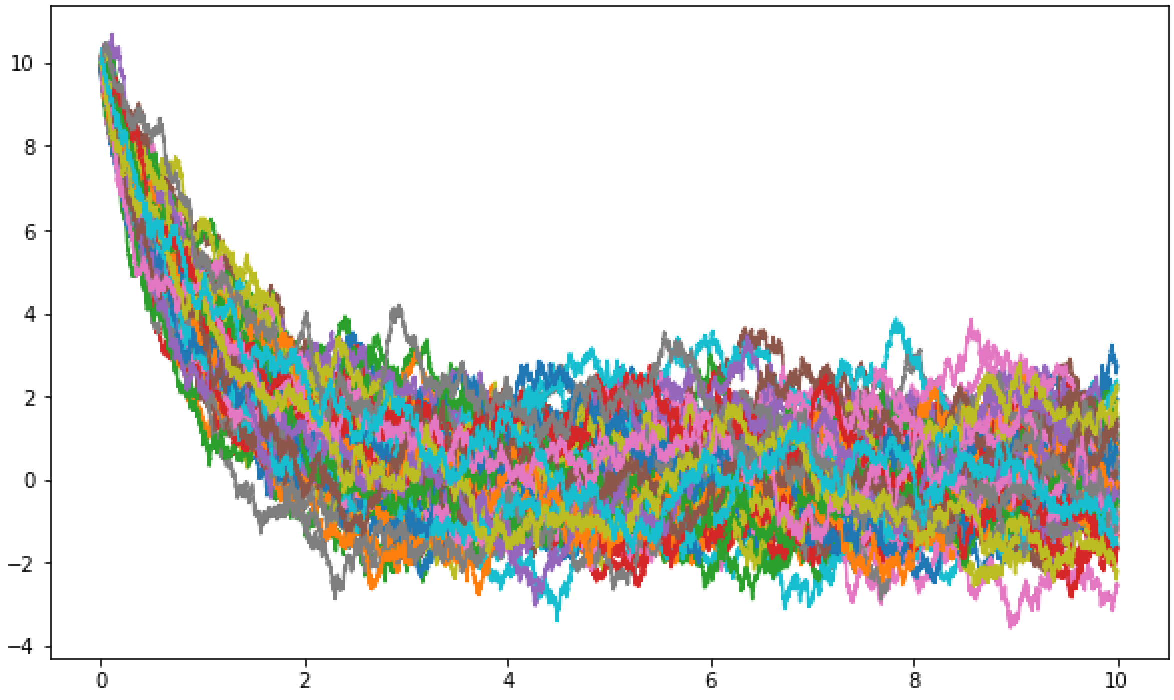
Using this program, compute ten trajectories with same initial condition (x= 10) and different seeds for the random number generator. At fixed value of time t, compute the average over the trajectories of the value of x and its standard deviation. How do these two quantities depend on t?

```
In [ ]: # Run 100 simulations and store them in traj list:

        traj=[]
        for itraj in range(100):
            traj.append(run(x0=0))
        traj=np.array(traj)
```

```
In [ ]: traj.shape # 100 runs composed of (10000 values, 10000 times)
```

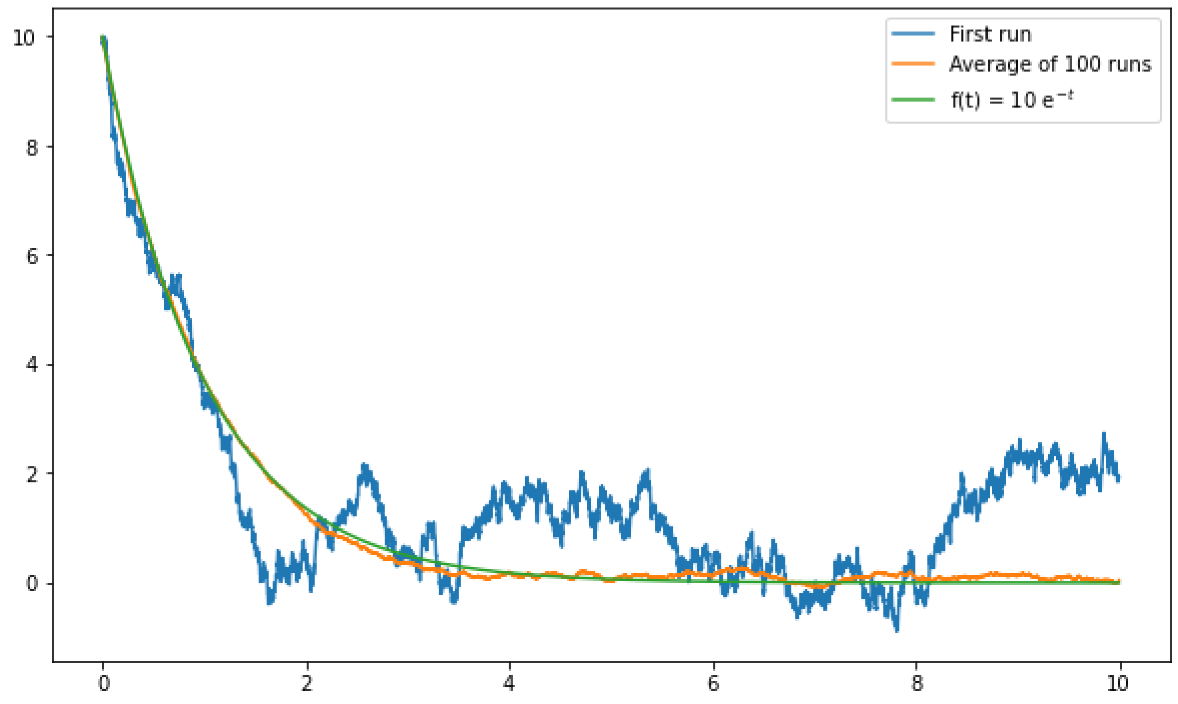
```
Out[ ]: (100, 10000, 2)
```

```
In [ ]: # Plot the 100 trajectories:

        f = plt.figure(figsize= (10,6))
        for itraj in range(len(traj)):
            plt.plot(traj[itraj,:,0],traj[itraj,:,1])
```



```
In [ ]: # Here we confront 3 different plots:
        # - 1 normal integration
        # - The average of 100 integrations
        # - The function f(t) = 10 e^{-t}

        f = plt.figure(figsize= (10,6))
        plt.plot(traj[0,:,0],traj[0,:,1], label = 'First run')
        plt.plot(traj[0,:,0],np.average(traj[:,:,1],axis=0), label = 'Average of 100
        plt.plot(traj[0,:,0],10*np.exp(-traj[0,:,0]), label = 'f(t) = 10 e$^{-t}$')
        plt.legend()
```

```
Out[ ]: <matplotlib.legend.Legend at 0x7fb9b476ff98>
```



Randomness of the average:

The SDE we are dealing with is of the form

$$dx = A(x)dt + BdW, \tag{1}$$

where in our case $A = -x, B = \sqrt{2}$.

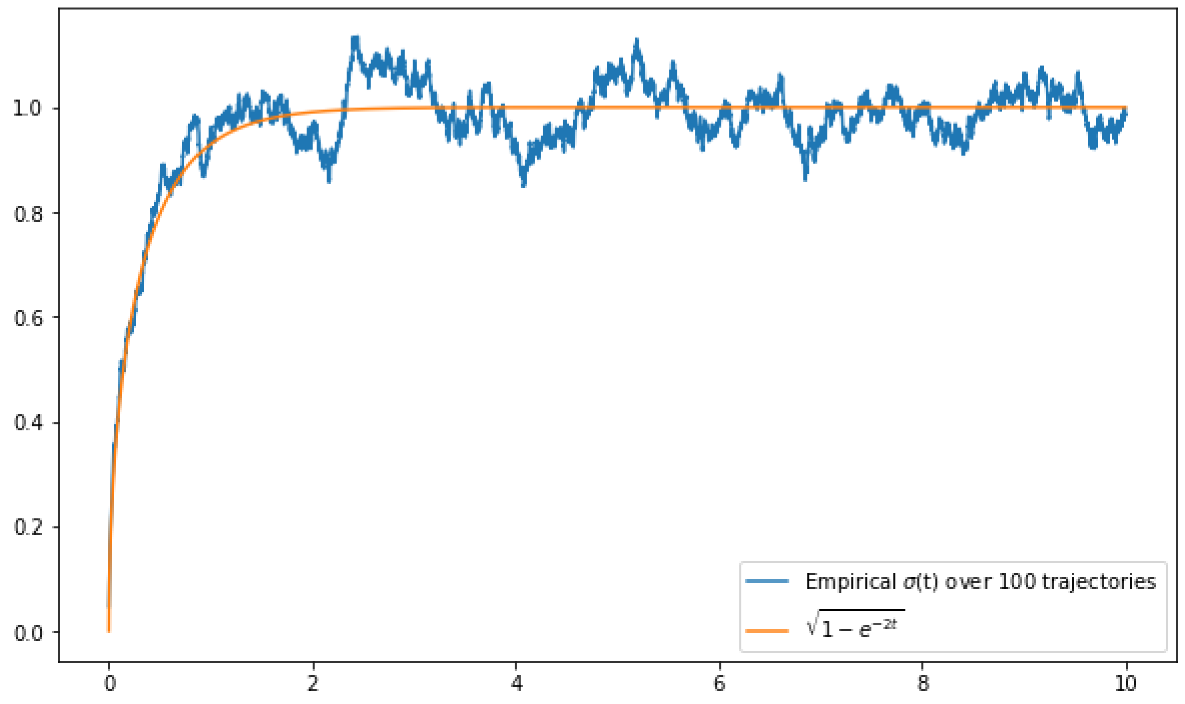
When we take the average over N repetitions of the random part we have

$$\frac{1}{N}\sum_i dW_i = \frac{1}{N}\sqrt{dt}\sum_i R_i = \frac{\sqrt{N}}{N}\sqrt{dt} = \frac{dW}{\sqrt{N}}. \tag{2}$$

That is, the standard deviation computed over time of the orange line is $\frac{1}{\sqrt{N}}$ the standard deviation of the blue line.

In [ ]:
```
# Here we compute the standard deviation at each time-step over the 100 traje

f = plt.figure(figsize= (10,6))
plt.plot(traj[0,:,0],np.std(traj[:,:,1],axis=0), label = 'Empirical $\sigma$(
plt.plot(traj[0,:,0],np.sqrt(1-np.exp(-2*traj[0,:,0])), label = '$\sqrt{1 - e
plt.legend()
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x7fb9b2e6e4a8>
```



Standard deviation:

Now we want to compute the standard deviation.

The average is given by

$$a = \frac{1}{N}\sum_i x_i, \tag{3}$$

its increment is therefore given by

$$da = -\frac{1}{N}\sum_i x_i dt + \frac{1}{N}\sum_i \sqrt{2}dW_i = -adt + \frac{1}{\sqrt{N}}\sqrt{2}dW \overset{N\to\infty}{\sim} -adt. \tag{4}$$

Thus

$$a(t) = a_0 e^{-t}. \tag{5}$$

Let $v = \frac{1}{N}\sum_i x_i^2$. Applying the Ito chain rule we obtain

$$dv = \frac{1}{N}2\sum_i x_i(-x_i dt + \sqrt{2}dW_i) + \frac{1}{N}\sum_i 2\frac{1}{2}2dt = -2\frac{\sum_i x_i^2}{N}dt + 2dt = -2(v -$$

Thus

$$v(t) = ke^{-2t} + 1, \quad v(0) = x_0^2 \implies v(t) = (x_0^2 - 1)e^{-2t} + 1. \tag{7}$$

In particular we see that it correctly tend to 1 as $t \to \infty$. We can now compute the standard deviation

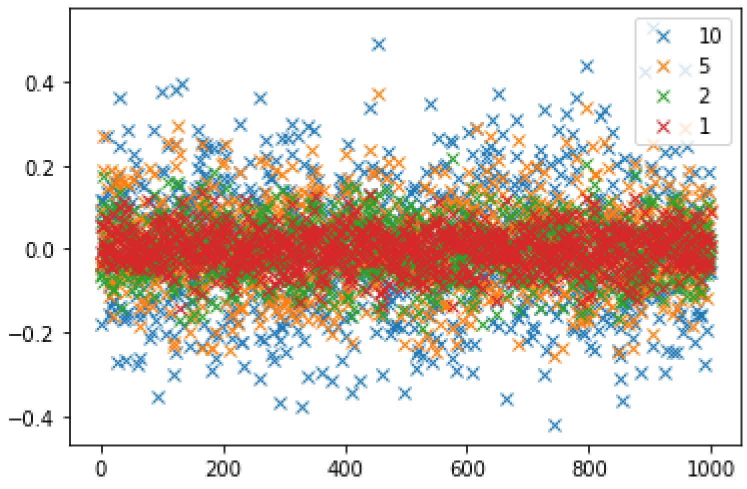
$$std(t) = \sqrt{\text{Var(t)}} = \sqrt{v(t) - a^2(t)} = \sqrt{1 - e^{-2t}}. \tag{8}$$

Exercise 3

Analyze the trajectories and compute their time increment, defined as g(t) =x(t + τ) − x(t). How does the increment depend on τ? How does it depend on x?

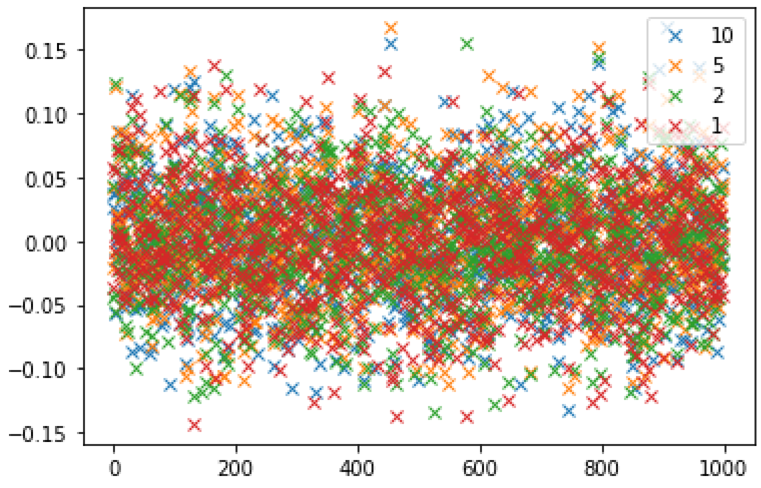
In order to graph the function $g(t)$ for different τ we subtract from the trajectory staring from $t+\tau$ itself, discarding the last τ points.
We can see that as τ increases the increment gets bigger and bigger, while for small τ we get values closer to 0.
In the code we use `n`, that is τ/dt.

In [ ]:
```python
for n in (10,5,2,1):
    plt.plot((traj[0,n:,1]-traj[0,:-n,1])[::10],"x",label=str(n))
plt.legend()
plt.show()
```



We actually know how these values depend on `n`, indeed the dependece of the increment is with $\sqrt{n}$. This is shown in the plot below.

In [ ]:
```python
for n in (10,5,2,1):
    plt.plot(((traj[0,n:,1]-traj[0,:-n,1])/np.sqrt(n))[::10],"x",label=str(n)
plt.legend()
plt.show()
```



Even better we can show this dependece using the std of $g(t)$, where we also notice that for big `n` this dependece does not hold. This is because only for small `n` the stochastic term is relevant.

(Although this works only if we have `x0 = 0`, Bussi does not really know why)

In [ ]:
```
for n in (1,2,5,10,20,50,100,200,500,1000,2000,5000):
    print(n,np.std((traj[0,n:,1]-traj[0,:-n,1]))/np.sqrt(n))
```

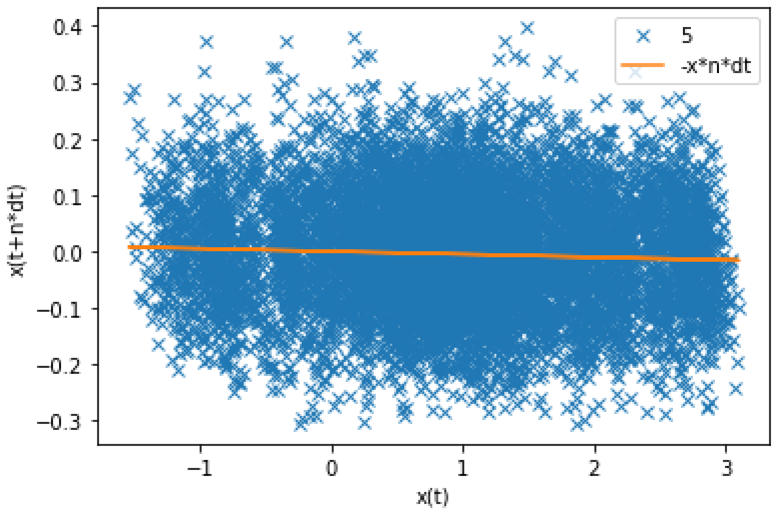
```
1 0.04479852173414821
2 0.04460289910511773
5 0.04484141334732319
10 0.04483377443602207
20 0.045649795485277964
50 0.046026183916232435
100 0.047094341594395246
200 0.046470085119936624
500 0.045813095108177916
1000 0.04490434473136618
2000 0.03851118376800101
5000 0.009996313536132019
```

Now how $g(t)$ depends on x?

The orange line is what we expect from the deterministic part of the equation for x, but the stochastic part is way bigger and cover this trend. On the big scale though, we saw that the deterministic part will bring x down.

In [ ]:
```
# tau = n * dt
n=5
plt.plot(traj[0,:-n,1],(traj[0,n:,1]-traj[0,:-n,1]),"x",label=str(n))
plt.xlabel("x(t)")
plt.ylabel("x(t+n*dt)")
plt.plot(traj[0,:-n,1],-traj[0,:-n,1]*n*0.001,label="-x*n*dt")
plt.legend()
```

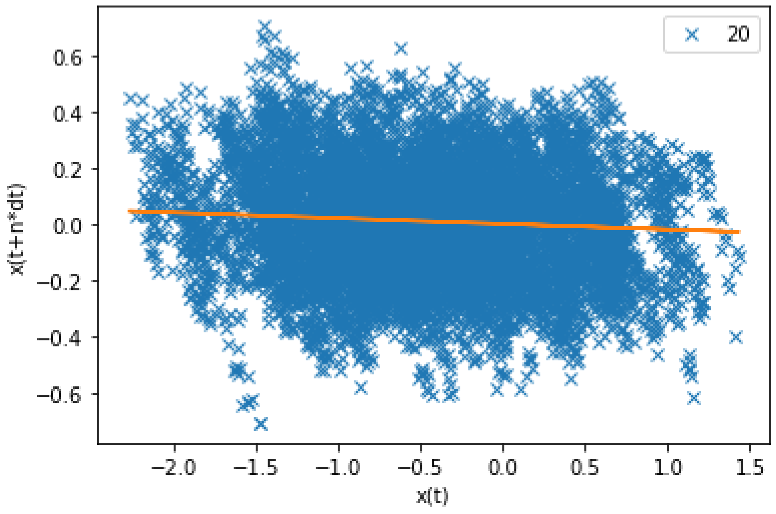
Out[ ]:
```
<matplotlib.legend.Legend at 0x7fb9b28b5f28>
```



Here with bigger n we can see the deterministic trend more easily.

In [ ]:
```
n=20
plt.plot(traj[0,:-n,1],(traj[0,n:,1]-traj[0,:-n,1]),"x",label=str(n))
plt.xlabel("x(t)")
plt.ylabel("x(t+n*dt)")
plt.plot(traj[0,:-n,1],-traj[0,:-n,1]*n*0.001)
plt.legend()
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x7fb9b4dd5e80>
```



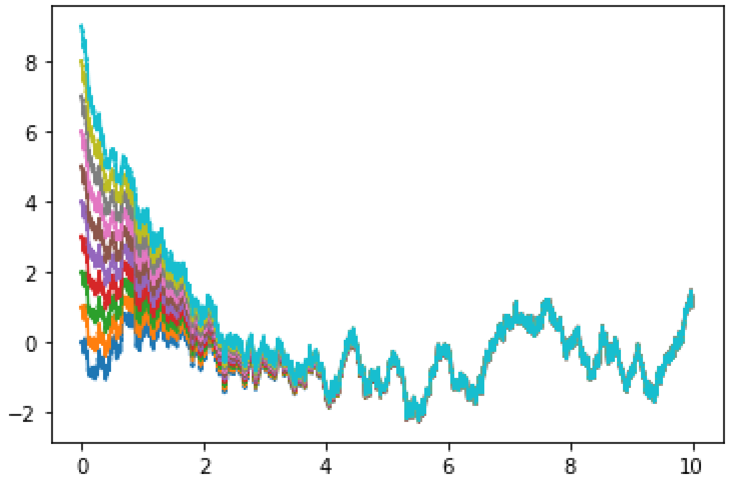
The take home message is that on the short scale the stochastic term dominates, but on the long run, the deterministic part has an important role too.

Bonus

If we make trajectories with different starting points, but with the same seed for the random number generator, we end up with all the trajectories that reunite and become the same thing. So always be careful with random number generators.

```
In [ ]: traj=[]
        for itraj in range(10):
            traj.append(run(x0=itraj,seed=1977))
        traj=np.array(traj)
```

```
In [ ]: for itraj in range(len(traj)):
            plt.plot(traj[itraj,:,0],traj[itraj,:,1])
```



```
In [ ]:
```