[PS5-1]

i. The values seem to be asymmetric about the middle value. This is bad because it implies that mass is not conserved. The distance a diffusing particle travels in a time t grows in a time sqrt(t).

520	5.1800	0.6058
521	5.1900	0.6030
522	5.2000	0.6000
523	5.2100	0.5970
524	5.2200	0.5938

[PS5-2]

i. The total time that has elapsed is 9.99 seconds ~~ 10 seconds.
ii. It compares perfectly. Below I've attached an image of the analytical solution along with the value I get for the density a t=10 along with the numerical solution for the density at t=10.

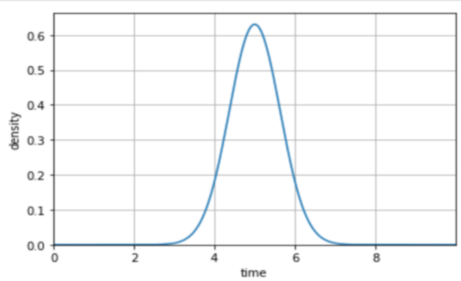
In [28]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [29]:
```
file_path1 = '/Users/Austin/Desktop/PHY 40/Homework/5/slughorn5_2.txt'

x, y = np.loadtxt(file_path1, unpack = True) #Method to extract/import informa

# print(min(x), max(x))
# print(min(y), max(y))


plt.plot(x,y)
plt.xlabel('time')
plt.ylabel('density')
#plt.legend()
ax=plt.gca() #Allows for axis limits
ax.set(xlim=(min(x), max(x)), ylim=(min(y), 1.05*max(y)))
plt.grid()
plt.show()
```



Hw [5-2]

Analytic Solution

$$\rho(x,t=0) = \delta(x-500)$$

$$C(k) = \frac{1}{2\pi}\int dx\, e^{-ikx}\cdot\delta(x-500) = \frac{1}{2\pi}e^{-ik(500)}$$

$$\therefore \rho(x,t) = \int dk \frac{1}{2\pi} e^{-ik500} e^{-Dk^2t} e^{ikx}$$

$$\rho(x,t) = \int dk \frac{1}{2\pi} e^{ik(x-500)} \cdot e^{Dk^2t}$$

$$\rho(x,t) = \frac{1}{\sqrt{4\pi Dt}} e^{-(x-500)/4Dt}$$

$x = x[500]\cdot dx$
$= (500)\cdot .01$
$= 5$
$t = 10$

$$\rho(0,10) = \frac{1}{\sqrt{4\pi(.02)(10)}} e^{-(10-5)/4(.02)(10)}$$

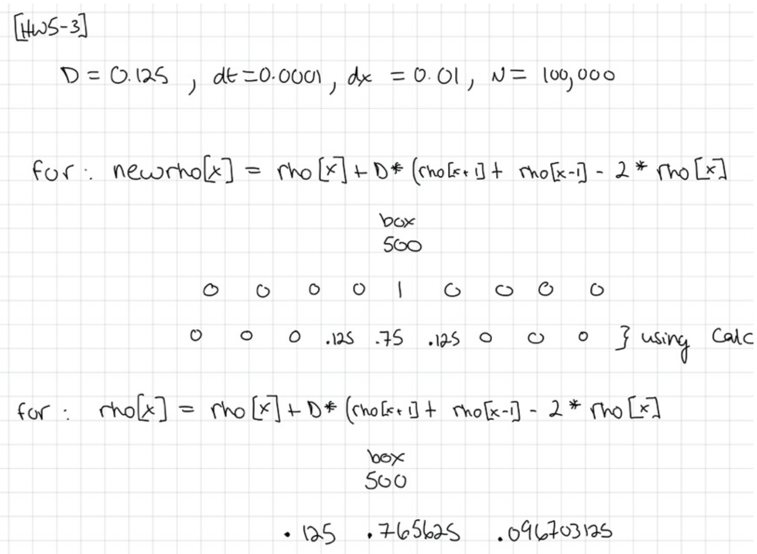
$$= .00000235 \sim 0$$

1000	9.9800	0.0000
1001	9.9900	0.0000

[HW5-3]

i. The value I get can be seen in the picture below.

ii. The original function of my code is more reasonable as it gives symmetric values about the center element whereas the older code gives asymmetric values about the center element – that is the center elements value is .75 the element to the right of that is .0957 and the value left of center is .125 to which the sum is *about* 1.0 as you can see below.

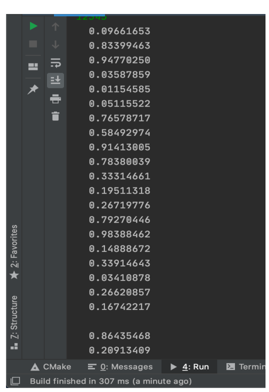


[HW5-4]

i. I see that the numbers are all between 0 and 1. I also see the value of each decimal value is a random value, however, they are not random in the sense that if you run the code again with the same N and seed you get the same decimal values printed in the same exact order.

[HW5-5]

i. I Notice that while there is now 6 moments, the first moment is the same set of values from the previous code [HW5-4] as you can see below.



[HW5-7]

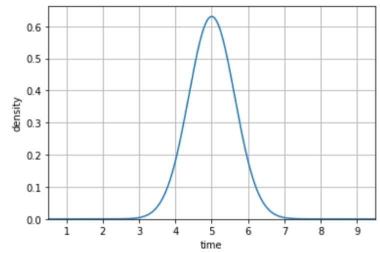
i. As can be seen from the picture below, it compares to that from 5-2.

```
In [22]: #PS5_7
file_path1 = '/Users/Austin/Desktop/PHY 40/Homework/5/madeyemoody5_7.txt'

x, y = np.loadtxt(file_path1, unpack = True) #Method to extract/import informa

# print(min(x), max(x))
# print(min(y), max(y))


plt.plot(x,y)
plt.xlabel('time')
plt.ylabel('density')
#plt.legend()
ax=plt.gca() #Allows for axis limits
ax.set(xlim=(min(x), max(x)), ylim=(min(y), 1.05*max(y)))
plt.grid()
plt.show()
```



[HW5-8]

i. No it doesn't matter. The density is just shifted by 100 elements or by time - either to the right or the left. For rho[400] the norm is t = 4 whereas for rho[600], the norm is at t=6 second. The probability density is still the same though. It can be compared from the two pictures below

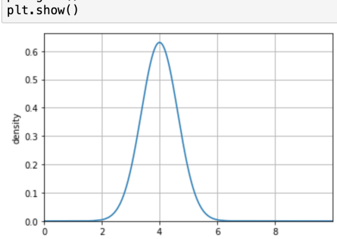
ii. For rho[400]

```
In [30]: #PS5_8
file_path1 = '/Users/Austin/Desktop/PHY 40/Homework/5/slughorn5_8.txt'

x, y = np.loadtxt(file_path1, unpack = True) #Method to extract/import info

# print(min(x), max(x))
# print(min(y), max(y))


plt.plot(x,y)
plt.xlabel('time')
plt.ylabel('density')
#plt.legend()
ax=plt.gca() #Allows for axis limits
ax.set(xlim=(min(x), max(x)), ylim=(min(y), 1.05*max(y)))
plt.grid()
plt.show()
```



iii. For rho[600]

In [31]:
```
#PS5_8
file_path1 = '/Users/Austin/Desktop/PHY 40/Homework/5/slughorn5_8.txt'

x, y = np.loadtxt(file_path1, unpack = True) #Method to extract/import information from txt file

# print(min(x), max(x))
# print(min(y), max(y))


plt.plot(x,y)
plt.xlabel('time')
plt.ylabel('density')
#plt.legend()
ax=plt.gca() #Allows for axis limits
ax.set(xlim=(min(x), max(x)), ylim=(min(y), 1.05*max(y)))
plt.grid()
plt.show()
```