DM physique

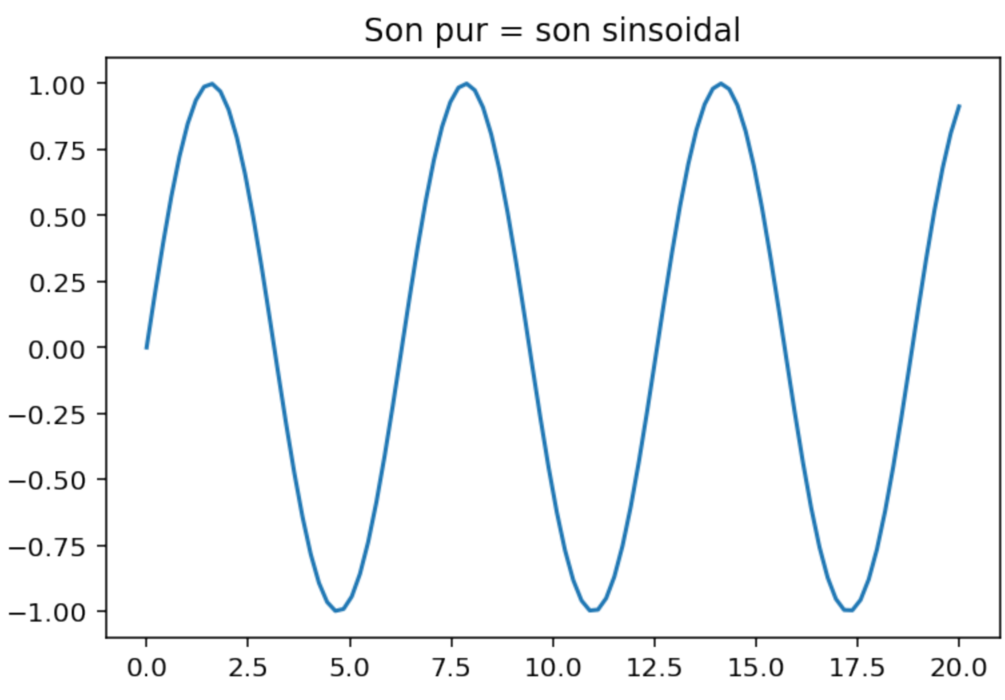
April 29, 2020

Q1. Pour un son de fréquence fondamentale f_0, l'octave correspond au son de fréquence double $2f_0$
.

Q2.a. Un son pur est constitué d'une unique composante sinusoidale.

[2]:
```
import matplotlib.pyplot as plt
from numpy import sin, linspace

plt.title("Son pur = son sinsoidal")
t=linspace(0,20,100)
plt.plot(t,sin(t))
plt.show()
```



Les sons fig1 et fig2 sont donc des sons composés.

Q2.b. Calcul de la fréquence f_1

[11]:

```
T_1=11.8e-3/3
f_1=round(1/T_1)
print("fréquence 1 = ",f_1,"Hz")
```

```
fréquence 1 =  254 Hz
```

Calcul de la fréquence f_2

[13]:

```
T_2=12.8e-3/5
f_2=round(1/T_2)
print("fréquence 2 = ",f_2,"Hz")
```

```
fréquence 2 =  391 Hz
```

Conclusion

- note 1: Do3
- note 2: Sol3

Q3. L'inégalité $1 < f < \frac{4}{3}$ multipliée par $\frac{3}{2}$ vaut $\frac{3}{2} < \frac{3}{2}f < \frac{3}{2} \times \frac{4}{3} = 2$

De même

$\frac{4}{3} < f < 2$ multipliée par $\frac{3}{2}$ vaut $2 < \frac{3}{2}f < 3$...

Q4.

[44]:

```
f=1.5
n=1
while f>1.02:
    f=1.5*f
    n=n+1
    if f>=2:
        f=f/2
    print(n," -> ",round(f,2))
```

```
2  ->  1.12
3  ->  1.69
4  ->  1.27
5  ->  1.9
6  ->  1.42
7  ->  1.07
8  ->  1.6
9  ->  1.2
10  ->  1.8
11  ->  1.35
12  ->  1.01
```

Q5. Pour 12 itérations, on retrouve $f \approx 1$ avec $f > 1$.

Q6.a. $\frac{3^m}{2^n} = 1$ donc $3^m = 2^n$, ce qui est impossible car 2^n est pair alors que 3^m est impair.

Q6.b. L'inégalité nétant jamais vérifiée, l'algorithme ne s'arrête jamais.

Q7. cf Q5.

Q8. Pour une fréquence fondamentale de $262Hz$ on obtient la série suivante

[55]:
```
f_0=262
f=1.5*f_0
n=1
gamme=[f_0,f]
while f>1.05*262:
    f=1.5*f
    n=n+1
    if f>=2*f_0:
        f=f/2
    gamme.append(round(f))
gamme.sort()
print(gamme)
```

```
[262, 266, 280, 295, 315, 332, 354, 373, 393.0, 420, 442, 472, 497]
```

[ ]:
```
Q8.a. Elles sont légèrement différentes.

Q8.b.
```

[57]:
```
round(266/262,3),round(280/266,3)
```

[57]:
```
(1.015, 1.053)
```

Les proportions d'une note à la suivante ne sont pas les mêmes.

Q9.a.

[71]:
```
round(278/262,3),round(294/278,3)
```

[71]:
```
(1.061, 1.058)
```

Q9.b. Sur le piano il s'agit de la gamme *tempérée* pour laquelle les rapports entre deux demi-tons successifs sont toujours les mêmes.

[75]:
```
from numpy import exp,log

gamme_tempérée=[]
gamme_tempérée_exacte=[]
for i in range(13):
    gamme_tempérée.append(round(f_0*2**(i/12)))
    gamme_tempérée_exacte.append(f_0*2**(i/12))
gamme_tempérée
```

[75]:
```
[262, 278, 294, 312, 330, 350, 371, 393, 416, 441, 467, 495, 524]
```

[80]:
```
for n in range(12):
    print("Rapport entre note ",n+1," et note ",n," =␣
 ↪",gamme_tempérée_exacte[n+1]/gamme_tempérée_exacte[n])
```

```
Rapport entre note  1  et note  0  =  1.0594630943592953
Rapport entre note  2  et note  1  =  1.0594630943592953
Rapport entre note  3  et note  2  =  1.0594630943592953
Rapport entre note  4  et note  3  =  1.0594630943592953
Rapport entre note  5  et note  4  =  1.0594630943592953
Rapport entre note  6  et note  5  =  1.0594630943592953
Rapport entre note  7  et note  6  =  1.059463094359295
Rapport entre note  8  et note  7  =  1.059463094359295
Rapport entre note  9  et note  8  =  1.0594630943592953
Rapport entre note  10  et note  9  =  1.0594630943592953
Rapport entre note  11  et note  10  =  1.0594630943592953
Rapport entre note  12  et note  11  =  1.0594630943592953
```