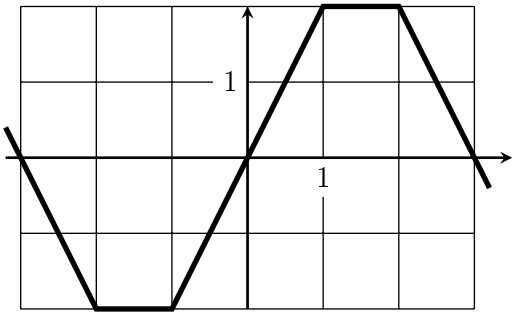
Dans ce TP, on présente l'utilisation du module de calcul formel avec **Python** : **Sympy**. Nous utiliserons **EduPython** qui est un environnement Python (permet de programmer et exécuter confortablement du Python) sur lequel le module Sympy est installé.

1 Exemple



On considère la fonction impaire, périodique et de période $T = 6$ définie selon la courbe ci-contre.

Comme la fonction est **impaire**, on sait que $a_0 = a_n = 0$ et que :

$$b_n = \frac{4}{T}\int_0^{\frac{T}{2}} f(t)\cdot\sin(n\,\omega\,t)\,\mathrm{d}t$$

$$\text{Puissance } P = \frac{2}{T}\int_0^{\frac{T}{2}} f(t)^2\,\mathrm{d}t = \frac{1}{2}\sum_{k\geqslant 1} b(n)^2$$

(a) On commence par importer les modules utiles

```
Py
from sympy import *  # calcul formel
import matplotlib.pyplot as plt # graphique
```

(b) On commence par définir T et ω.

```
Py
T = 6
omega = 2*pi/T # pi est défini dans Sympy
```

(c) On aura besoin de symboles mathématiques. Ces symboles ont une valeur inconnue mais on peut préciser leur domaine de valeurs.

```
Py
n = symbols('n', nonnegative=True, integer=True)
t = symbols('t', real=True)
```

(d) Définir $f(t)$. **Important :** On peut bien sûr se contenter de définir f sur $[0;3]$ et exploiter la parité. Mais puisque l'on utilise du calcul formel, on n'a pas vraiment à économiser les calculs.

```
Py
f = Piecewise((-2*t-6,t<-2), (-2,t<-1), (2*t,t<1), (2,t<2), (-2*t+6,
    True))
```

(e) Calcul dex `a0`, `a(n)` et `b(n)`

```
Py
a0 = 1/T*integrate(f,(t,-T/2,T/2))
an = 2/T*integrate(f*cos(n*omega*t),(t,-T/2,T/2))
bn = 2/T*integrate(f*sin(n*omega*t),(t,-T/2,T/2))
# affichage
print("a0 =",a0,"; a(n) =",an,"; b(n) =",bn)
```

Le résultat de `b(n)` mériterait d'être simplifié. Modifiez en écrivant `bn.simplify()` dans la ligne du `print`.

Remarque : En Python, `**` signifie puissance.

Vous pouvez aussi essayer `pprint(bn.simplify())` pour un meilleur affichage.

(f) On voudrait afficher une liste des valeurs du spectre. Pour le calcul de A_n on doit envisager deux cas : $n = 0$ ou $n > 0$.

Voyez ci-dessous avec un exemple de calcul puis l'affichage de toute une séquence :

```python
An = Piecewise((abs(a0), n==0), (sqrt(an**2+bn**2), True))
print("A(6) = ",An.subs(n,6)) # calcul en remplaçant n par 6
print("A(6) = ",An.subs(n,6).evalf()) # approx

for i in range(10):
    print("A(",i,") =",An.subs(n,i).evalf)
```

On peut afficher le spectre :

```python
plt.figure()
x_list = range(30)
y_list = [An.subs(n,x).evalf() for x in x_list]
plt.bar(x_list, y_list)
plt.show()
```

(g) Calcul de la puissance.

```python
P = 1/T*integrate(f**2,(t,-T/2,T/2))
print("P =",P.evalf())
```

(h) Calcul de la puissance de la série partielle.

```python
N = symbols('N', nonnegative=True, integer=True)
S = a0**2 + 1/2*Sum(an**2+bn**2, (n,1,N))
for i in range(10):
  print("S(",i,") =",S.subs(N,i).evalf() )
```

(i) Dernière chose utile : chercher pour quel N le rapport $\frac{S_N}{P}$ dépasse par exemple 99,9 %.

```python
i = 1
while S.subs(N,i).evalf() / P.evalf() < 0.999:
    i = i + 1
print("Il faut aller jusque N=",i," pour atteindre 99,9 % de P")
```

(j) On peut tracer la courbe. *Le calcul a pris du temps chez moi...*

```python
f2 = a0 + Sum(an*cos(n*omega*t) + bn*sin(n*omega*t), (n,1,5))
plt.figure()
x_list = [x/100 for x in range(600)]
y_list = [f2.subs(t,x).evalf() for x in x_list]
plt.plot(x_list, y_list)
plt.show()
```

2 Autre exemple

Faites le même travail avec la fonction f, de période 6 et définie par : $f(t) = \begin{cases} t \text{ si } 0 \leqslant t < 2 \\ 3 - \frac{t}{2} \text{ si } 2 \leqslant t < 6 \end{cases}$