

La somme partielle $\sum_{n=0}^N \frac{t_n}{2^{n+1}}$ est une valeur approchée par défaut à $2^{-(N+1)}$ près de la limite de la série. Aussi pour une valeur approchée à 2^{-n} près il suffit de calculer la somme partielle jusqu'en $n - 1$.

```
from math import log

# Pré et post-traitement
def constantePTM(n):
    L = [0]
    C = constante_rec(L,n-1)    # Calcul récursif
    S = 0                       # Calcul de la somme partielle
    for k in range(n):
        S += C[k]*2**(-k-1)
    return S

# Partie récursive : calcul après N itérations
def constante_rec(L,N):
    if N == 0:
        return L
    newL = []
    for x in L:
        if x == 0:
            newL.extend([0,1])
        else:
            newL.extend([1,0])
    return constante_rec(newL,N-1)
```

Exemple : le nombre est situé entre 0,25 et 0,5. La meilleure précision que l'on puisse atteindre en flottant 64 bits est 2^{-54} . Avec 15 chiffres significatifs corrects :

```
In [2]: 2**-54
Out[2]: 5.551115123125783e-17

In [3]: constantePTM(53)
Out[3]: 0.4124540336401076
```

