Planche 1 :

```
import matplotlib.pyplot as plt
import numpy as np
from math import log

# Q1  :

def P(n,x)  :
    puiss = 1
    somme = 1
    for k in range(2*n)  :
        puiss *= x
        somme += puiss
    return somme

def Q1()  :
    plt.clf()
    N=100
    X=[-2+4*k/N for k in range(N+1)]
    for n in range(1,10)  :
        Y = [P(n,x) for x in X]
        plt.plot(X,Y,label="n="+str(n))
    plt.legend()
    plt.axis([-2, 2, 0, 5])
    plt.savefig("Q1.png")

def Q1bis(a,b)  :
    plt.clf()
    N=100
    X=[a+(b-a)*k/N for k in range(N+1)]
    for n in range(1,10)  :
        Y = [P(n,x) for x in X]
        plt.plot(X,Y,label="n="+str(n))
    plt.legend()
    plt.savefig("Q1bis.png")

# prendre a = -0.9 et b= -0.35 pour bien voir les minima. Un seul par fonction
# sur [-2,2].

# Q2  : si x  != 1, P_n(x)=(1-x**(2n+1))/(1-x), donc
# P_n'(x)=(2nx**(2n+1)-(2n+1)x**(2n)+1)/(1-x)**2.


# Q3  : u_n'(x) = 2n(2n+1)x**(2n-1) * (x-1) et u_n(0)=1, u_n(1)=0
# donc u_n est négative donc P_n décroissante sur ]-inf,a_n] et
# positive donc P_n croissante sur [a_n,+inf[.
# pour un certain a_n < 0.

# Q4  : u_n(-1)=-4n <0 donc a_n\in]-1,0[. On fait une dichotomie sur cet
# intervalle.
```

```
def u(n,x)  :
    puiss = 1
    y = x**2
    for i in range(n)  :
        puiss *= y
    return 2*n*x*puiss-(2*n+1)*puiss+1

def dicho (n)  :
    a = -1
    b = 0
    e = 10**(-4)
    while abs(b-a)>e  :
        m = (a+b)/2
        if u(n,m) > 0  : b = m
        else  : a = m
    return (a+b)/2

# Q5  :

def Q5()  :
    X = [n for n in range(1,501)]
    plt.clf()
    Y = [dicho(n) for n in X]
    plt.plot(X,Y)
    plt.savefig("Q5.png")

# On conjecture que cette suite tend vers -1.

# Q6  :

# équivalent simple  : ln(n).
# on écrit u_n(a_n)=0, on passe au log, on a 2n ln|a_n| = -ln(2n+1-2na_n)
# donc ln|a_n| equivalent à -ln(n) / 2n, qui tend donc vers 0, donc |a_n| -> 1
# or a_n<0 donc a_n -> -1.


# Q7  :

# -h_n ~ ln|a_n| ~ -(ln n)/2n donc h_n ~ (ln n)/(2n).

# Q8  :

def w(n)  :
    return dicho(n)+1-log(n)/(2*n)-log(2)/n

def Q8()  :
    X = [n for n in range(1,501)]
    plt.clf()
    Y = []
    somme = 0
```

```
    for i in range(1,501)  :
        somme += w(i)
        Y.append(somme)
    plt.plot(X,Y)
    plt.savefig("Q8.png")
    return Y

# Ca semble converger.

# Q9  : on reprend ln|a_n| = -ln(2n+1-2na_n), on développe, on trouve
# h_n=(ln n)/2n + (ln 2)/n + o(h_n**2) or h_n**2 ~ (ln n)**2 / 4n**2, dont la
# série conv (série de Bertrand classique).
```

Planche 2 :

RMS2023 1059. PYTHON .

```
import math

import matplotlib.pyplot as plt

import numpy
import numpy.polynomial as pol

def A(n) :
    U=pol.Polynomial([1])
    if n==0 :
        return(U)
    else :
        B=A(n-1).integ()
        C=B.integ(1,0)
        D=B-C(1)
    return(D)

def u(x) :
    return x /(math.exp(x)-1)

def w(x) :
    S=A(0)(0)
    for k in range (1,11)  :
        S+=A(k)(0)*x**k
    return S

X=numpy.linspace(-2,2,10)

Y=[u(x) for x in X]

Z=[w(x) for x in X]

plt.plot(X, Y)
```

```
plt.plot(X, Z)

plt.show()
```

1) On a directement $A_1 = X - \frac{1}{2}$ puis, par intégration et annulation de l'intégrale, $A_2 = \frac{X^2}{2} - \frac{X}{2} + \frac{1}{12}$. On trouve aussi $A_3 = \frac{X^3}{6} - \frac{X^2}{4} + \frac{X}{12}$.

2) Voici un programme qui renvoie A_n en tant qu'objet Python de type Polynomial.

```
import numpy.polynomial as pol
def A(n):
    U=pol.Polynomial([1])
    if n==0:
        return(U)
    else
    B=A(n-1).integ()
    C=B.integ(1,0)
    D=B-C(1)
    return(D)
```

3) On compare $A_n(0)$ et $A_n(1)$ pour les 9 premières valeurs via le programme suivant :

```
for n in range (1,10) :
  print(n, A(n)(0), A(n)(1))
```

Le résultat obtenu suggère la conjecture suivante :

Conjecture 1. On a $A_n(0) = A_n(1)$ pour tout $n \geqslant 2$.

On peut également composer les polynômes (le code de programmation de composition des polynômes est intuitif) :

```
for n in range (1,10) :
    S=pol.Polynomial([1,-1])
    print(n)
    print(A(n))
    print(A(n)(S))
```

Au signe près, on trouve les mêmes valeurs numériques (non facilement identifiables) pour les coefficients de $A_n(1-X)$ et A_n. La conjecture qui se profile est :

Conjecture 2. On a $A_n(1-X) = (-1)^n A_n$ pour tout $n \in \mathbf{N}$.

4) Passons à la représentation graphique demandée.

Conjecture 3. On a

$$\forall x \in \mathbf{R}\backslash\{0\}, \quad \frac{x}{e^x - 1} = \sum_{k=0}^{+\infty} A_k(0) x^k$$

5) Preuve de la conjecture 1 . Soit $n \in \mathbf{N}$ tel que $n \geqslant 2$. On a tout simplement :

$$A_n(1) - A_n(0) = \int_0^1 A_n'(t)\mathrm{d}t = \int_0^1 A_{n-1}(t)\mathrm{d}t = 0$$

Preuve de la conjecture 2 par récurrence. La formule est triviale pour $n = 0$. On suppose que l'on a $A_n(1-X) = (-1)^n A_n$ et l'on souhaite montrer la formule $A_{n+1}(1-X) = (-1)^{n+1} A_{n+1}$. Or le polynôme différence $A_{n+1}(1-X) - (-1)^{n+1} A_{n+1}$ est clairement d'intégrale nulle de 0 à 1 (quitte à faire un changement de variable linéaire immédiat sur le terme en $1-X$) et son polynôme dérivé est

$$-A'_{n+1}(1-X)-(-1)^{n+1}A'_{n+1}=-A_n(1-X)-(-1)^{n+1}A_n=0$$

Ainsi, $A_{n+1}(1-X)-(-1)^{n+1}A_{n+1}$ est forcément le polynôme nul.

Preuve de la conjecture 3. ll y a deux difficultés dans cette conjecture. D'abord, il faut justifier que la série du second membre converge et préciser pour quelles valeurs de x. En l'occurrence, on a peut-être été trop généreux en s'autorisant à choisir x dans $\mathbf{R}$ (privé de $\{0\}$). Il serait déjà satisfaisant d'avoir une information pour x voisin de l'origine.

Justifions que la suite $(A_k(0))$ est bornée. Cela prouvera, d'après la théorie des séries entières, que la série $\sum A_k(0)x^k$ converge pour tout $x\in]-1,1[$ (le rayon de la série entière est au moins égal à 1). D'après la définition des polynômes A_k, l'inégalité $\sup_{-1\leqslant x\leqslant 1}|A_k(x)|\leqslant 1$ s'obtient immédiatement par récurrence grâce au lemme suivant :

Lemme 1. Soit $f\in\mathscr{C}^1([0,1],R)$ vérifiant $\int_0^1 f(x)\mathrm{d}x=0$ et $\|f'\|_\infty\leqslant 1$. Alors $\|f\|_\infty\leqslant 1$.

Preuve. Pour tout $x\in[-1,1]$, on écrit

$$f(x)=(f(x)-f(1/2))+f(1/2)$$

L'inégalité des accroissements finis donne

$$|f(x)-f(1/2)|\leqslant|x-1/2|\times\|f'\|_\infty\leqslant\frac{1}{2}$$

En intégrant () sur $[0,1]$ et en exploitant (), on obtient

$$|f(1/2)|=\left|\int_0^1 f(x)-f(1/2)\mathrm{d}x\right|\leqslant\int_0^1|f(x)-f(1/2)|\mathrm{d}x\leqslant\frac{1}{2}$$

Grâce à () et à ( ) , on conclut que $|f(x)|\leqslant 1$. $\square$.

Ensuite, il faut trouver un moyen d'identifier les coefficients de deux côtés ! Il est plus simple de tenter de démontrer la formule suivante :

$$\forall x\in]-1,1[,\quad x=(e^x-1)\sum_{k=0}^{+\infty}A_k(0)x^k$$

Remarque. L'étude de l'intervalle maximal sur lequel la série est convergente (c'est-à-dire le calcul exact du rayon de convergence) dépasse largement le cadre du programme des classes préparatoires (il faudrait invoquer un cours d'analyse complexe et l'appliquer à la fonction holomorphe $z\mapsto\frac{z}{e^z-1}$ sur le disque complexe ouvert de centre 0 et de rayon 2π). D'ailleurs, si on reprend les représentations graphiques précédentes sur un intervalle strictement plus grand que $[-2\pi,2\pi]$, on constate effectivement une explosion à l'extérieur de cet intervalle :

Revenons à la preuve. Par convergence absolue des séries envisagées, le membre droit se reformule via un produit de Cauchy :

$$\forall x\in]-1,1[,\left(\sum_{k=1}^{+\infty}\frac{x^k}{k!}\right)\times\left(\sum_{k=0}^{+\infty}A_k(0)x^k\right)=\sum_{n=1}^{+\infty}\left(\sum_{k=1}^{n}\frac{A_{n-k}(0)}{k!}\right)x^n.$$

Le coefficient d'indice $n=1$ vaut $A_0(0)=1$ (d'après la question a)). La formule () sera donc conséquence des identités suivantes :

$$\forall n\geqslant 2,\quad\sum_{k=1}^{n}\frac{A_{n-k}(0)}{k!}=0$$

Cette formule est a priori non évidente mais devient très abordable si l'on invoque le lemme suivant :

Lemme 2. Pour tout polynôme $P \in \mathbb{R}_n[X]$ on a la formule $P(1) = \sum_{k=0}^{n} \frac{P^{(k)}(0)}{k!}$.

Preuve. Il suffit d'appliquer la formule de Taylor de 0 à 1 .

On applique le lemme précédent au polynôme A_n (qui est de degré n et vérifie $A_n^{(k)} = A_{n-k}$ par récurrence immédiate) :

$$A_n(1) = A_n(0) + \sum_{k=1}^{n} \frac{A_{n-k}(0)}{k!}$$

La conjecture 1 (prouvée ci-dessus) affirme que $A_n(1) = A_n(0)$.

Planche 3 :

```
import matplotlib.pyplot as plt
import numpy as np
from math import pi, cos, sin, sqrt

# Q1  :

def S(N,x)  :
    assert type(x)  != int, "x ne doit pas être un entier"
    return 1/x + sum([(2*x)/(x**2-n**2) for n in range(1,N+1)])
# Q2  :

def Q2 (N,a,b)  :
    plt.clf()
    d = 1000
    h = (b-a)/d
    X = [a+k*h for k in range(d+1)]
    Y = [S(N,x) for x in X]
    plt.plot(X,Y)
    plt.axis([a,b,-40,20])
    plt.savefig("Q2.png")


# Q3  : simple comparaison à une série de Riemann.

# Q4  : il y a convergence normale.

# Q5  : continuité et imparité résultent de la convergence uniforme.
# S_N(x+1) = somme(1/(x+n), n = -N+1 .. N+1)=S_N(x) - 1/(x+N+1) + 1/(x-N) et
# on passe à la limite quand N -> +inf.

# Q6  : S_N(x/2)+S_N((x+1)/2)=2 somme 1/(x+2n) + 2 somme 1/(x+2n+1)
#           = 2S_{2N}(x) + 2/(x+2N+1) et on passe à la limite.

# Q7  : cotan(pi*x) vérifie cela par formule de trigo,
# donc la fonction demandée aussi.

# Q8  : on fait un DL en 0  : pi*cotan(pi*x) = 1/x +o(1), et pour S_N, pour -1<x<1
# et x !=0, on  sort le terme 1/x, et tous les autres tendent vers série
```

```
#(2x)/(x**2-n**2) pour n de 1 à +infty, ce qui vaut 0 en 0.
# Donc la différence est prolongeable par continuité par 0 en 0. Idem pour les autres
# entiers par 1-périodicité.
# Sur [0,1] f atteint en max M en un x0. Or 2M >= f(x0/2)+f((x0+1)/2)=2f(x0)=2M
# donc f(x0/2)=f((x0+1)/2)=M, donc par récurrence le max est atteint en tous les
# f(x0/2**n), qui par continuité tend vers f(0)=0. Idem pour le min. Donc f= cte = 0.


# Donc S(x) = pi*cotan(pi*x). On peut le vérifier graphiquement  :

cotan = lambda x  : cos(x)/sin(x)


def Q8 (N,a,b)  :
    plt.clf()
    d = 1000
    h = (b-a)/d
    X = [a+k*h for k in range(d+1)]
    Y = [S(N,x) for x in X]
    plt.plot(X,Y,label="S_"+str(N))
    Y = [pi*cotan(pi*x) for x in X]
    plt.plot(X,Y,label="pi*cotan(pi*x)")
    plt.axis([a,b,-40,20])
    plt.legend()
    plt.savefig("Q8.png")

eps = sqrt(2)/1000000
a = -1+eps
b = 1-eps
```

Planche 4 :

1) On donne une première fonction de test, qui utilise la question **2)**, et une seconde pour générer une des 2^{n^2} matrices à tester à partir de l'entier correspondant (ordre lexicographique inverse), et la fonction principale...

```
import numpy as n p

def test(M, n) :
M = n p.transpose(M).dot(M) - n*n p.eye(n)
for i in range(n) :
    for j in range(n) :
        if M[i, j] ! = 0 :
            return False
return True

def matrice(k, n) :
l = n p.array([1 for i in range(n**2)])
for i in range(n**2) :
    if k%2 = = 1 :
        l[i] = -1
```

```
    k = k//2
return l.reshape(n, n)

def denombre(n) :
compteur = 0
for k in range(2**(n**2)) :
    M = matrice(k, n)
    if test(M, n) :
        compteur + = 1
return compteur
```

2) Soit $A \in \mathscr{M}_n(\mathbb{R})$ dont les coefficients appartiennent tous à $\{-1, 1\}$. Si A vérifie $\mathscr{H}_n$, ses colonnes $C_1, \dots, C_n$ sont de norme $\sqrt{n}$, orthogonales deux à deux, donc la famille $(\frac{1}{\sqrt{n}}C_1, \dots, \frac{1}{\sqrt{n}}C_n)$ est orthonormée, si bien que $\frac{1}{\sqrt{n}}A$ est orthogonale.

Réciproquement si $\frac{1}{\sqrt{n}}A$ est orthogonale, alors la famille $(\frac{1}{\sqrt{n}}C_1, \dots, \frac{1}{\sqrt{n}}C_n)$ est orthonormée donc $(C_1, \dots, C_n)$ est orthogonale, c'est-à-dire que A vérifie $\mathscr{H}_n$.

3) Les 8 matrices vérifiant $\mathscr{H}_2$ sont exactement les 4 matrices dont trois coefficients valent 1 et le dernier -1, et les 4 matrices dont trois coefficients valent -1 et le dernier 1. En divisant par $\sqrt{2}$, on obtient 8 matrices orthogonales dont 4 sont symétriques et correspondent à des matrices de symétries orthogonales, et les 4 autre correspondent à des matrices de rotation d'angles respectifs $\pm\frac{\pi}{4}$ et $\pm\frac{3\pi}{4}$. Les endomorphismes associés à nos 8 matrices sont donc les composées commutatives de ces symétries ou rotations avec l'homothétie de rapport $\sqrt{2}$, soit des similitudes (directes ou indirectes) du plan.

4) Il suffit de noter que les coefficients de $A^\top$ restent dans $\{-1, 1\}$ et que $\frac{1}{\sqrt{n}}A$ est orthogonale si et seulement si $\frac{1}{\sqrt{n}}A^\top$ l'est.

5) Changer les signes sur la colonne j revient à changer C_j en $-C_j$, et préserve bien sûr le caractère orthogonal de la famille des colonnes.

Et changer les signes sur la ligne i correspond à changer les signes sur la colonne i de $A^\top$, ce qui préserve la propriété $\mathscr{H}_n$ pour $A^\top$ donc pour A.

6) Il suffit de tester en plus la première ligne et la première colonne… On obtient 6 matrices, pas si compliquées à trouver à la main…

```
def testbis(M, n) :
N = n p.transpose(M).dot(M) - n*n p.eye(n)
for k in range(n) :
    if M[k, 0] ! = 1 or M[0, k] ! = 1 :
        return False
for i in range(n) :
    for j in range(n) :
        if N[i, j] ! = 0 :
            return False
return True

def denombrebis() :
n = 4
compteur = 0
for k in range(2**(n**2)) :
    M = matrice(k, n)
    if testbis(M, n) :
```

```
            compteur + = 1
    return compteur
```