Mathématiques 2, Centrale PSI

Exercice A

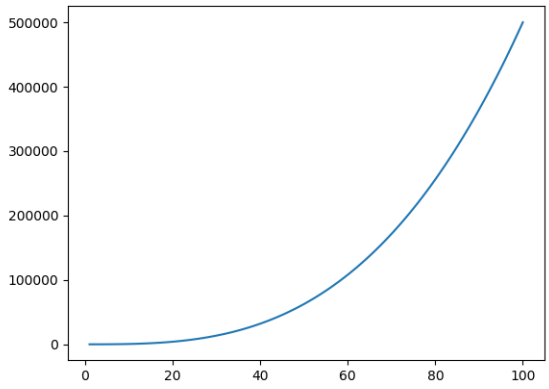
On commencera à numéroter les coefficients des matrices à 0 (et non pas à 1)

```
1)
import numpy as np
import matplotlib.pyplot as plt
def f(n,i,j):
    a=i*n+1
    if i%2==0:
        a=a+j
    else:
        a=a+n-1-j
    return a

2)
def M(n):
    m=np.zeros((n,n))
    for i in range(n):
        for j in range(n):
            m[i,j]=f(n,i,j)
    return m

for i in range(2,6): # tests
    print(M(i))

4)
for i in range(1,11):
    print(np.rank(M(i)))
```

On conjecture donc que le rang de $M(n)$ vaut 2 (pour $n \geq 2$, le rang valant 1 pour $n = 1$)
Démontrons ce résultat.

On remarque que pour $k \in [\![2, n]\!]$, $C_k - C_{k-1} = \begin{pmatrix} 1 \\ -1 \\ 1 \\ \vdots \\ (-1)^{n+1} \end{pmatrix}$ on a donc $Im(M(n)) = Vect(C_1, C_2 - C_1)$ et, comme C_1 et C_2 ne sont pas colinéaires alors $rg(M(n)) = 2$ et on a le résultat voulu.

```
5)
def trace(n):
    LX,LY=[],[]
    for k in range(1,n+1):
        LX.append(k)
        LY.append(np.trace(M(k)))
    plt.figure()
    plt.plot(LX,LY)
    plt.show()
    plt.pause(5)
    plt.close()
trace(100)
```




```
6)
for n in range(1,101):
    print(np.trace(M(n))/n**3)
```

7) La question 6) permet de conjectuer que $tr(M(n)) \sim \frac{n^3}{2}$

8) Avec 1) on a, pour $i \in [\![0; n-1]\!]$, $(M(n))_{i,i} = \begin{cases} in+1+i \text{ si } i \text{ pair} \\ (in+1)+n-1-i = (i+1)n-i \text{ si } i \text{ impair} \end{cases}$

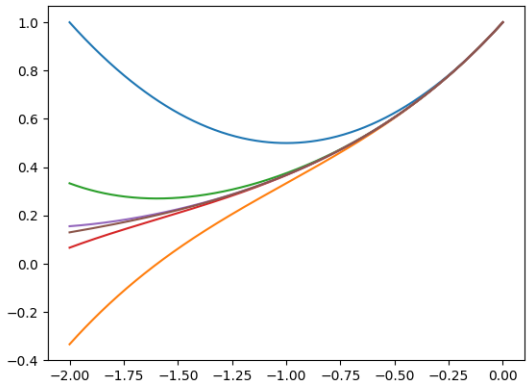
Donc si n pair, on écrit $n = 2p$ et

$$\begin{aligned} & tr(M(n)) \\ &= \sum_{k=0}^{p-1} [(M(n))_{2k,2k} + (M(n))_{2k+1,2k+1}] \\ &= \sum_{k=0}^{p-1} [(2kn+1+2k) + ((2k+2)n - (2k+1))] \\ &= \sum_{k=0}^{p-1} [(2n+2+2n-2)k + 2n] \\ &= 4n\frac{(p-1)p}{2} + p2n \\ &= 4n\frac{(\frac{n}{2}-1)n}{4} + p2n \\ &= n\frac{n-2}{2}n + n^2 \\ &= \frac{n^3}{2} \end{aligned}$$

Pour n impair on obtient de même : $tr(M(n)) = \frac{n^3+n}{2}$ et donc $tr(M(n)) \sim \frac{n^3}{2}$.

Exercice B

```
1a)
import numpy as np
import matplotlib.pyplot as plt

def P(n,t):
    rp=1
    f,p=1,1 # f=factorielle, p=puissance de t
    for i in range(1,n+1):
        f,p=f*i,p*t
        rp+=p/f
    return rp
X=np.linspace(-2,0,500)
plt.figure()
for n in range(2,8):
    Y=P(n,X)
    plt.plot(X,Y)
plt.show()
```



On conjecture que P_n à une racine réelle (négative) si n est impaire et n'a pas de racines réelles si n est pair.

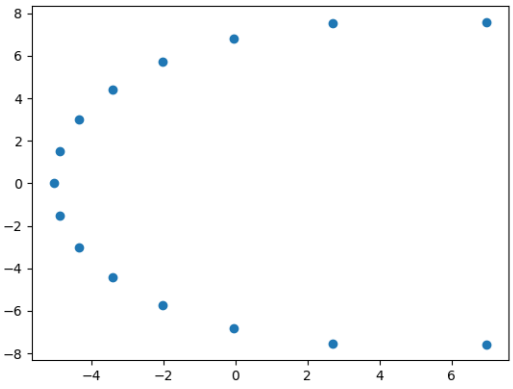
1b) Pour utiliser la méthode Newton, il faut calculer P'_n. On remarque que $P'_n = P_{n-1}$

```
def Newton(n): # on part de 0 car Pn(0)=1
    eps=0.001  # critère d'arrêt
    a=1
    N=0
    while abs(P(n,a))>eps and N<200:
        N=N+1
        if N==200 or abs(P(n-1,a)<eps):
            return 'échec'
        a=a-P(n,a)/P(n-1,a)
    return a
for n in range(2,8):
    print('n=',n,' ->',Newton(n))
```

L'éxécution des lignes ci-dessus donne :

```
n= 2  -> échec
n= 3  -> -1.5960734851112373
n= 4  -> échec
n= 5  -> -2.180964133512295
n= 6  -> échec
n= 7  -> -2.7590193982889297
```

```
1c)
from numpy.polynomial import Polynomial

def traceRacinePn(n):
    Pn=[1]
    f=1
    for i in range(1,n+1):
        f=f*i
        Pn=Pn+[1/f]
    Pn=Polynomial(Pn)
    LR=Pn.roots() # liste des racines de Pn
    X,Y=[],[]
    for i in range(n):
        X.append(LR[i].real)
        Y.append(LR[i].imag)
    plt.figure()
    plt.plot(X,Y,marker='o',linestyle='')
    plt.show()
    plt.pause(5)
    plt.close()
traceRacinePn(15) donne
```



2a) On remarque que $P_n(a) = P_n'(a) + \frac{a^n}{n!}$ donc si a est racine d'ordre au moins 2, alors $P_n(a) = P_n'(a) = 0$ et donc $\frac{a^n}{n!} = 0$. Ce qui donne $a = 0$ mais $P_n(a) = 1$ Absurde Les racines de P_n sont simples.

2b) Si $t \geq 0$ alors $\frac{t^n}{n!} \geq 0$ et donc $P_n(t) \geq 1$ et donc P_n n'a pas de racine dans $\mathbb{R}_+$

2c) Par récurrence sur k. Initialisation :
$P_1 = 1 + t$ donc une seule racine.
$P_2 = 1 + t + t^2/2$ $\Delta = 1 - 2 = -1 < 0$ pas de racines réelles.
Hérédité :
$P_{2k}' = P_{2k-1}$ admet une seule racine, P_{2k} admet son minimum en cette racine r_{k-1} et ce minimum vaut
$P_{2k}(r_{k-1}) = P_{2k-1}(r_{k-1}) + \frac{r_{k-1}^{2k}}{(2k)!} = 0 + + \frac{r_{k-1}^{2k}}{(2k)!} > 0$ car $r_{k-1} \neq 0$
Donc $P_{2k}(t) > 0$ pour tout t et P_{2k} n'a pas de racine.
$P_{2k+1}' = P_{2k}$ donc P_{2k+1}' est strictement croissante, et vu les limites aux bornes, P_{2k+1} admet une unique racine.

On a la conclusion voulue.

2d) $P_{2k+1}(-1)$ est la somme partielle d'une série alternée convergente de premier 1 et de second terme 0 , on en déduit donc $P_{2k+1}(-1) > 0$
$P_{2k+1}(-(2k+1))$
$= \sum_{i=0}^{2k+1} \frac{-(2k+1)^i}{i!}$
$= \sum_{p=0}^{k} [\frac{(2k+1)^{2p}}{(2p)!} - \frac{(2k+1)^{2p+1}}{(2p+1)!}] - \frac{(2k+1)^{2k+1}}{(2k+1)!}$
$= \sum_{p=0}^{k} \frac{(2k+1)^{2p}}{(2p+1)!} \underbrace{[\frac{1}{2p+1} - (2k+1)]}_{\leq 0} \underbrace{- \frac{(2k+1)^{2k+1}}{(2k+1)!}}_{\leq 0}$
Donc $P_{2k+1}(-(2k+1)) \leq 0$ et $P_{2k+1}(-1) > 0$, donc par le TVI : $-(2k+1) =\leq r_k \leq -1$

2e) $P_{2k+3}(r_k) = P_{2k+1}(r_k) + \frac{r_k^{2k+2}}{(2k+2)!} + \frac{r_k^{2k+3}}{(2k+3)!}$ mais $P_{2k+1}(r_k) = 0$ donc
$P_{2k+3}(r_k) = \frac{r_k^{2k+2}}{(2k+2)!} + \frac{r_k^{2k+3}}{(2k+3)!} = \frac{r_k^{2k+2}}{(2k+2)!}(1 + \frac{r_k}{(2k+3)}) = \frac{r_k^{2k+2}}{(2k+2)!}(\frac{2k+3+r_k}{(2k+3)!})$
mais $-(2k+1) =\leq r_k \Rightarrow 2 \leq r_k + 2k + 3$ et donc $P_{2k+3}(r_k) \geq 0$,
vu les variation de P_{2k+3} on a donc $r_{k+1} \leq r_k$ et la suite (r_k) est décroissante.

Exercice C

1a) Les lignes proposées affichent 3 valeurs approchées des solutions de $\begin{cases} 2x^2+3y-11=0 \\ 3x-2xy-2=0 \end{cases}$

1b) Comme un sauvage!! On écrit $U=\begin{pmatrix} x_0 & x_2 \\ x_1 & x_3 \end{pmatrix}$ avec $\begin{cases} x_0^2+x_1^2=x_2^2+x_3^2=1 \\ x_0x_2+x_1+x_3=0 \end{cases}$

On pose $S^{-1}=\begin{pmatrix} x_4 & x_5 \\ x_5 & x_6 \end{pmatrix}$ et on écrit $AS^{-1}=U$ qui donne $\begin{cases} x_4+x_5=x_0 \\ x_5+x_6=x_2 \\ x_5=x_1 \\ x_6=x_3 \end{cases} \Leftrightarrow \begin{cases} x_4+x_1=x_0 \\ x_1+x_3=x_2 \\ x_5=x_1 \\ x_6=x_3 \end{cases}$

```
def g(x):
    return [x[0]**2+x[1]**2-1,x[2]**2+x[3]**2-1,x[0]*x[2]+x[1]*x[3],x[4]+x[1]-x[0],x[1]+x[3]-x[2],0]
s=fsolve(g,[1,1,1,1,1,1])
U=np.array([[s[0],s[2]],[s[1],s[3]]])
invS=np.array([[s[4],s[1]],[s[1],s[3]]])
S=alg.inv(invS)
print('sqrt(5)U=',np.dot(U,np.sqrt(5)))
print('sqrt(5)S=',np.dot(S,np.sqrt(5)))
```

donne alors $U=\frac{1}{\sqrt{5}}\begin{pmatrix} 2 & 1 \\ -1 & 2 \end{pmatrix}$ et $S=\frac{1}{\sqrt{5}}\begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix}$
(on vérifie que les valeurs propres de S sont strictement positives.

2a) On a A^TA qui est symétrique réelle donc diagonalisable dans une base orthonormée (théorème spectral) et on a $\exists P \in O_n(\mathbb{R})$, $A^TA=PDP^t$ avec $D=diag((\lambda_i)_{i\in[\![1;n]\!]})$
Soit X un vecteur propre de A^TA associée à λ_i.
Alors $||AX||^2=(AX)^T(AX)=X^T(A^TA)X=X^T\lambda_iX=\lambda_i||X||^2$ et donc $\lambda_i \geq 0$ et même $\lambda_i>0$ car A^TA est inversible (puisque A l'est - déterminant par exemple).
Posons $\Delta=diag((\sqrt{\lambda_i})_{i\in[\![1;n]\!]})$, $S=P\Delta P^T$ et $U=AS^{-1}$ (S est bien inversible car Δ est bien inversible).
$S^T=(P\Delta P^T)^T=P(\Delta)^TP^T=P\Delta P^T=S$ (cas Δ diagonale et donc S est symétrique réelle, à valeurs propres strictement positives car ce sont les $\sqrt{\lambda_i}>0$)
$U^TU=(AS^{-1})^T(AS^{-1})=S^{-1}A^TAS^{-1}$ mais $A^TA=S^2$ donc $U^TU=S^{-1}S^2S^{-1}=I_n$ et donc U est orthogonale.

On a bien $A=US$ avec $U\in O_n(\mathbb{R})$ et S symétrique réelle à valeurs propres strictement positives.

2b) Avec les notations ci-dessus :
$A=US \Rightarrow A=UP\Delta P^T \Rightarrow (UP)^TAP=\Delta$
En posant $D=\Delta$ (diagonale), $V=(UP)^T$ (orthogonale) et $W=P$ (orthogonale) on a le résultat voulu.

```
2c)
A=np.array([[1,1],[0,1]])
AA=np.dot(np.transpose(A),A)
reduc=alg.eig(AA)
P=reduc[1]
D=reduc[0]
D=np.array([[np.sqrt(D[0]),0],[0,np.sqrt(D[1])]])
P=np.dot(P,1/(np.sqrt(P[0,0]**2+P[1,0]**2)))
print('W=P=',P)
V=np.transpose(np.dot(U,P))
print('V=',V)
print('D=',D)
```

permet d'obtenir les matrices V_1, W_1 et D_1

Exercice D

1a) $A \in M_{n+1}(\mathbb{R})$ est triangulaire donc ses valeurs propres sont sur la diagonale, il y en a $n+1$ distinctes donc A est diagonalisable.

1b) A est sembable à $D = diag(1, \frac{1}{2}, \ldots, \frac{1}{n+1}) = diag((\frac{1}{i})_{1 \leq i \leq n+1})$
Donc $\exists Q \in GL_n(\mathbb{R})$, $A = QDQ^{-1}$
On a $A^k = QD^kQ^{-1}$ et $D^k = diag(((\frac{1}{i})^k)_{1 \leq i \leq n+1}) \underset{k \to +\infty}{\longrightarrow} diag(1, 0, \ldots, 0)$
Donc (A^k) converge vers $P = Qdiag(1, 0, \ldots, 0)Q^{-1}$
Comme $diag(1, 0, \ldots, 0)^2 = diag(1, 0, \ldots, 0)$, P est la matrice d'un projecteur (sur e_1)

```
1c)
import numpy as np
def matriceA(n):
    A=np.zeros((n+1,n+1))
    for j in range(n+1):
        for i in range(j+1):
            A[i,j]=1/(j+1)
    return A
```

```
1d) Le code :
import numpy.linalg as alg
print(alg.eig(A(3)))
```

permettent de voir que $(1, 0, \ldots, 0)^T$ est un vecteur propre de $A(n)$ associé à la valeur propre 1 (simple). (évident !)

```
2)
from numpy.random import randint
def tirage(k,n):
    L=[n]
    for i in range(k):
        L.append(randint(L[-1]+1))
    return L
```

3a) $(X_k = i)_{i \in [\![0;n]\!]}$ est un système complet d'événements, donc par la formule des proabilités totales :
$P(X_{k+1} = j) = \sum_{i=0}^{n} P(X_{k+1} = j | X_k = i) P(X_k = i)$
Clairement $P(X_{k+1} = j | X_k = i) = 0$ si $j > i$ donc $P(X_{k+1} = j) = \sum_{i=j}^{n} P(X_{k+1} = j | X_k = i) P(X_k = i)$
Avec la loi uniforme on a (si $i \leq j$) : $P(X_{k+1} = j | X_k = i) = \frac{1}{j+1}$ donc $P(X_{k+1} = j) = \sum_{i=j}^{n} \frac{1}{j+1} P(X_k = i)$

3b) On déduit de 3a) que $W_{k+1} = AW_k$ et par analogie avec les suites géométriques : $W_k = A^k W_0$

```
3c)
def Wk(k,n):
    W0=np.zeros((n+1,1))
    W0[n,0]=1
    A=matriceA(n)
    for i in range(k):
        W0=np.dot(A,W0)
    return W0
```

4a) $B = \begin{pmatrix} 0 & 1 & 2 & \ldots & n-1 & n \end{pmatrix}$ car alors $BW_k = \sum_{i=0}^{n} iP(X_k = i) = E(X_k)$

4b) $BA = \begin{pmatrix} c_0 & c_1 & \ldots & c_n \end{pmatrix}$ avec $c_j = \sum_{i=0}^{j-1} i\frac{1}{j} = \frac{1}{j}\frac{j(j-1)}{2} = \frac{j}{2}$ Donc $BA = \frac{1}{2}B$

4c) $E(X_k + 1) = BW_{k+1} = B(AW_k) = (BA)W_k = \frac{1}{2}BW_k$ et donc $E(X_{k+1}) = \frac{1}{2}E(X_k)$
4d) On en déduit $E(X_k) = \frac{1}{2^k}E(X_0)$ donc $E(X_k) = \frac{n}{2^k}$ qui tend vers 0.
Cohérent avec les essais éffectués ...

Exercice E

1) Les nombre qui s'écrivent avec k chiffres vont de 10^{k-1} à $10^k - 1$
Il y en a : $(10^k - 1) - 10^{k-1} + 1 = 10^k - 10^{k-1} = 10^{k-1}(10 - 1) = 9.10^{k-1}$

2a) On a : $10^{c(n)-1} \leq n < 10^{c(n)} \Rightarrow (c(n) - 1)ln(10) \leq ln(n) < c(n)ln(10) \Rightarrow \frac{ln(n)}{ln(10)} < c(n) \leq \frac{ln(n)}{ln(10)} + 1$
Donc $c(n) = \lfloor \frac{ln(n)}{ln(10)} \rfloor + 1$ et on écrit

```
def c(n):
    for i in range(1,25):
        if n==10**i: # problème d'arrondi
            return i+1
    return np.ceil(np.log(n)/np.log(10))
a=1 # test sur 100!
for i in range(2,101):
    a=a*i
print(c(a*1.)) # donne 158 (le point pour passer au flottant et éviter une erreur avec int)
```

```
2b)
def Couples(k):
    compteur=0
    for a in range(10**(k-1),10**k):
        for b in range(10**(k-1),10**k):
            if c(a*b)==2*k:
                compteur=compteur+1
    return compteur

for k in range(1,4):
    print(Couples(k)/(81*10**(2*k-2)))
```

donne 0.7160493827160493, 0.8160493827160494 et 0.8257320987654321

3a) Avec 1) $p_n = \frac{Couples(n)}{(9.10^{n-1})^2} = \frac{Couples(n)}{81 \times 10^{2n-2}}$

3b) Soit k et p deux nombres à n chiffres, alors $\begin{cases} 10^{n-1} \leq k \leq 10^n - 1 \\ 10^{n-1} \leq p \leq 10^n - 1 \end{cases}$
Donc $\underbrace{10^{2n-2}}_{2n-1 \text{ chiffres}} \leq pk < \underbrace{10^{2n}}_{2n+1 \text{ chiffres}}$ et pk est composée de $2n-1$ ou de $2n$ chiffres.
Cherhons, à k fixé, le plus grand p tel que $c(pk) = 2n - 1$
On a : $pk \leq 10^{2n-1} - 1 \Rightarrow p \leq \frac{10^{2n-1}-1}{k} \Rightarrow p \leq \lfloor \frac{10^{2n-1}}{k} \rfloor$
A k fixé il y a donc $\lfloor \frac{10^{2n-1}-1}{k} \rfloor - 10^{n-1} + 1$ nombre k à n chiffres tels que kn est $2n-1$ chiffres.

Bilan : $1 - p_n = \frac{1}{81 \times 10^{2n-2}} \sum_{k=10^{n-1}}^{10^n-1} \left(\lfloor \frac{10^{2n-1}-1}{k} \rfloor - 10^{n-1} + 1 \right)$

4a) A_n
$= \sum_{k=10^{n-1}}^{10^n-1} (10^{n-1} - 1)$
$= (10^n - 1 - 10^{n-1} + 1)(10^{n-1} - 1)$
$= (10^n - 10^{n-1})(10^{n-1} - 1) = 10^{2n-1} - 10^n - 10^{2n-2} + 10^{n-1}$ donc $A_n \sim 10^{2n-1}$

4b) $\frac{10^{2n-1}}{k} - 1 \leq \lfloor \frac{10^{2n-1}-1}{k} \rfloor \leq \frac{10^{2n-1}}{k}$
donc $\sum_{k=10^{n-1}}^{10^n-1} \frac{10^{2n-1}-1}{k} - 1 \leq B_n \leq \sum_{k=10^{n-1}}^{10^n-1} \frac{10^{2n-1}}{k}$

$C_n = \sum_{k=10^{n-1}}^{10^n-1} \frac{10^{2n-1}}{k}$
$= 10^{2n-1} \sum_{k=10^{n-1}}^{10^n-1} \frac{1}{k}$ calculs à vérifiés

Mais $\int_k^{k+1} \frac{1}{t}dt \leq \frac{1}{k} \leq \int_{k-1}^{k} \frac{1}{t}dt$ donc par relation de Chasles :

$10^{2n-1} \int_{10^{n-1}}^{10^n} \frac{1}{t}dt \leq C_n \leq 10^{2n-1} \int_{10^{n-1}-1}^{10^n-1} \frac{1}{t}dt$

donc $10^{2n-1} ln(\frac{10^n}{10^{n-1}}) \leq C_n \leq 10^{2n-1} ln(\frac{10^n-1}{10^{n-1}-1})$

donc $ln(10) \leq \frac{C_n}{10^{2n-1}} \leq ln(\frac{10^n-1}{10^{n-1}-1}) \underset{n\to+\infty}{\longrightarrow} ln(10)$

On en déduit $C_n \sim ln(10)10^{2n-1}$

Finalement : $B_n \sim C_n \sim ln(10)10^{2n-1}$

4c) On a donc $1 - p_n \underset{n\to+\infty}{\longrightarrow} \frac{1}{81}(ln(10) - \frac{1}{10})$ et donc p_n a une limite

Exercice F

```
1)
b=np.zeros((13,13))
def binome(i,j):
    if j>i:
        return 0
    res=1
    for k in range(j):
        res=res*(i-k)/(k+1)
    return res
```

2a) Pour $k > n+2$: $u_{n,k} = \frac{k^n}{k!} = \frac{\prod_{i=1}^{n} k}{\prod_{i=1}^{k} i} = \underbrace{[\prod_{i=1}^{n} \frac{k}{i}]}_{\text{cst}} \underbrace{[\prod_{i=n+1}^{k-2} \frac{1}{i}]}_{\leq 1} \frac{1}{k-1}\frac{1}{k} \leq \frac{cst}{(k-1)^2}$ et donc A_n est convergente par comparaison à une série de Riemann.

2b) $A_0 = \sum_{k=0}^{+\infty} \frac{1}{k!} = e$ et $A_1 = \sum_{k=0}^{+\infty} \frac{k}{k!} = \sum_{k=1}^{+\infty} \frac{1}{(k-1)!} = A_0 = e$

2)c) $A_{n+1} = \sum_{k=0}^{+\infty} \frac{k^{n+1}}{k!} = \sum_{k=1}^{+\infty} \frac{k^n}{(k-1)!} = \sum_{k=0}^{+\infty} \frac{(k+1)^n}{k!} = \sum_{k=0}^{+\infty} \frac{\sum_{i=0}^{n} \binom{n}{i} k^i}{k!} = \sum_{i=0}^{n} \binom{n}{i} \sum_{k=0}^{+\infty} \frac{k^i}{k!}$

Donc $A_{n+1} = \sum_{i=0}^{n} \binom{n}{i} A_i$

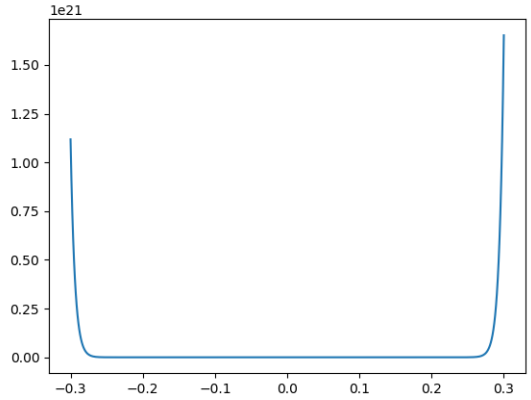
```
2d)
A=[1,1] # en fait on calcul A/e ...
for n in range (1,12):
    a=0
    for j in range(n+1):
        a=a+binome(n,j)*A[j]
    A.append(a)
    print('A(',n+1,')=',A[n+1],'e')
```

donne

```
A( 2 )= 2.0 e
A( 3 )= 5.0 e
A( 4 )= 15.0 e
A( 5 )= 52.0 e
A( 6 )= 203.0 e
A( 7 )= 877.0 e
A( 8 )= 4140.0 e
A( 9 )= 21147.0 e
A( 10 )= 115975.0 e
A( 11 )= 678570.0 e
A( 12 )= 4213597.0 e
```

3a) Par récurrence on montrer que $A_n \leq en!$
Initialisation directe car $A_0 = A_1 = e$
Hérédité : on suppose le résultat vrai de 0 à n
$A_{n+1} = \sum_{i=0}^{n} \binom{n}{i} A_i \leq \sum_{i=0}^{n} \frac{n!}{i!(n-i)!} i! = n! \sum_{i=0}^{n} \frac{1}{(n-i)!} \leq n! \sum_{i=0}^{n} 1 = n!(n+1) = (n+1)!$
On a donc le résultat au rang $n+1$
Conclusion : $\forall n \in \mathbb{N}\ ,\ A_n \leq n!$
On a donc $0 \leq \frac{A_n}{n!} \leq 1$ et comme $RC(\sum 1.x^n) = 1$, alors par règle de comparaison pour les séries entières $R \geq 1$
3b) On calcul les A_i comme ci-dessus et on fait

```
def f(x):
    s=0
    px=1
    for i in range(51):
        s=s+A[i]*px
        px=px*x
    return s
plt.figure()
X=np.linspace(-0.3,0.3,500)
Y=f(X)
plt.plot(X,Y)
plt.show()
```



3c) En tant que SE, f est dérivable terme à terme sur $]-R, R[$ et on a :
$\forall x \in]-R, R[\ ,\ f'(x) = \sum_{n=1}^{+\infty} \frac{A_n}{n!} n x^{n-1} = \sum_{n=0}^{+\infty} \frac{A_{n+1}}{n!} x^n$
Mais : $A_{n+1} = \sum_{i=0}^{n} \binom{n}{i} A_i = \sum_{i=0}^{n} \frac{n!}{i!(n-i)!} A_i$
$\Rightarrow \frac{A_{n+1}}{n!} = \sum_{i=0}^{n} \frac{A_i}{i!} \frac{1}{(n-i)!}$
Donc $f'(x) = \sum_{n=0}^{+\infty} [\sum_{i=0}^{n} \frac{A_i}{i!} x^i \frac{x^{n-i}}{(n-i)!}]$
On reconnaît le produit de Cauchy de deux séries entières de rayons de convergence R et $+\infty$. Donc $\forall x \in]-R, R[$: $f'(x) = f(x)exp(x)$

3d) On résout l'EDL_1 ci-dessus et on a : $f(x) = cexp(exp(x))$ avec $c \in \mathbb{R}$, mais $f(0) = e \Rightarrow c = e$ donc $f(x) = exp(exp(x) + 1) = eexp(exp(x))$