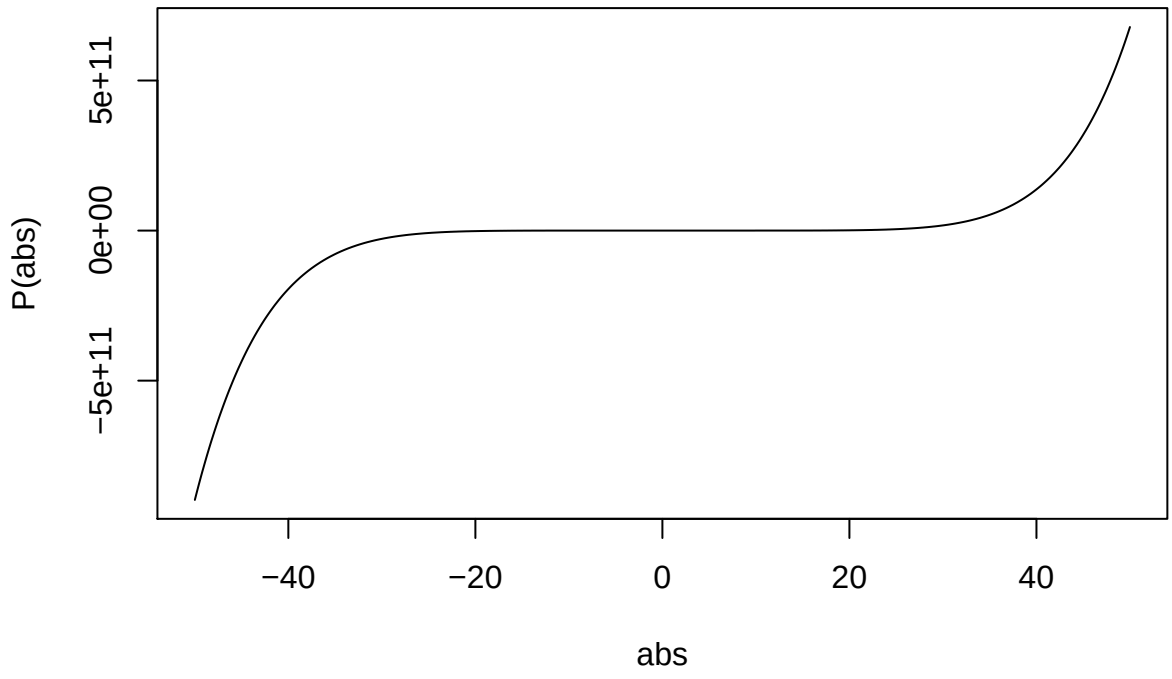
TP3 Correction

Clément LAROCHE

11 janvier 2019

Un drôle de polynôme

```
P <- function(x)
{
  x^7-7*x^6+21*x^5-35*x^4+35*x^3-21*x^2+7*x-1
}
abs <- seq(from = -50,to = 50,length.out = 1000)
plot(x = abs,y = P(abs),type = "l")
```




```
# A partir de cette étude numérique on peut en déduire que
# P admet une racine et que cette racine se trouve près de
# 0 (dans les alentours disons)
```

On peut donc réécrire P comme suit, avec x_0 la racine de ce polynome et $Q(x)$ un polynôme de degré inférieur :

$$P(x) = (x - x_0)Q(x)$$

On suppose que Q est de la forme : $ax^6 + bx^5 + cx^4 + dx^3 + ex^2 + fx + g$.
On va développer ce polynome et identifier terme à terme les coefficients. Cela donne :

$$\begin{aligned}(x - x_0)Q(x) &= (x - x_0)(ax^6 + bx^5 + cx^4 + dx^3 + ex^2 + fx + g) \\ &= ax^7 + bx^6 + cx^5 + dx^4 + ex^3 + fx^2 + gx - \\ &ax_0x^6 - bx_0x^5 - cx_0x^4 - dx_0x^3 - ex_0x^2 - fx_0x - gx_0 \\ &= ax^7 + (b - ax_0)x^6 + (c - bx_0)x^5 + (d - cx_0)x^4 + \\ &(e - dx_0)x^3 + (f - ex_0)x^2 + (g - fx_0)x - gx_0\end{aligned}$$

On obtient donc le système d'équations suivant

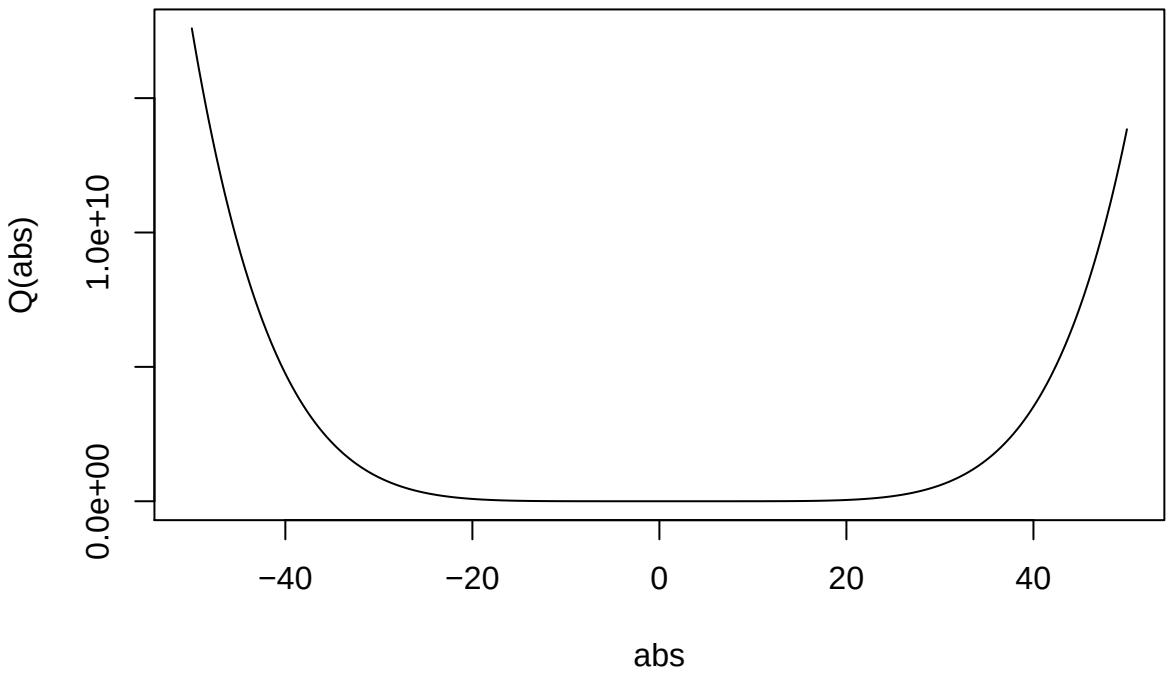
$$\begin{cases} a & = 1 \\ b - x_0 & = -7 \\ c - bx_0 & = 21 \\ d - cx_0 & = -35 \\ e - dx_0 & = 35 \\ f - ex_0 & = -21 \\ g - fx_0 & = 7 \\ gx_0 & = -1 \end{cases}$$

Maintenant que le système d'équations est posé, remarquons que le polynôme P admet une racine évidente qui vaut 1. Le système d'équations devient alors :

$$\begin{cases} a & = 1 \\ b & = -6 \\ c & = 15 \\ d & = -20 \\ e & = 15 \\ f & = -6 \\ g & = -1 \end{cases}$$

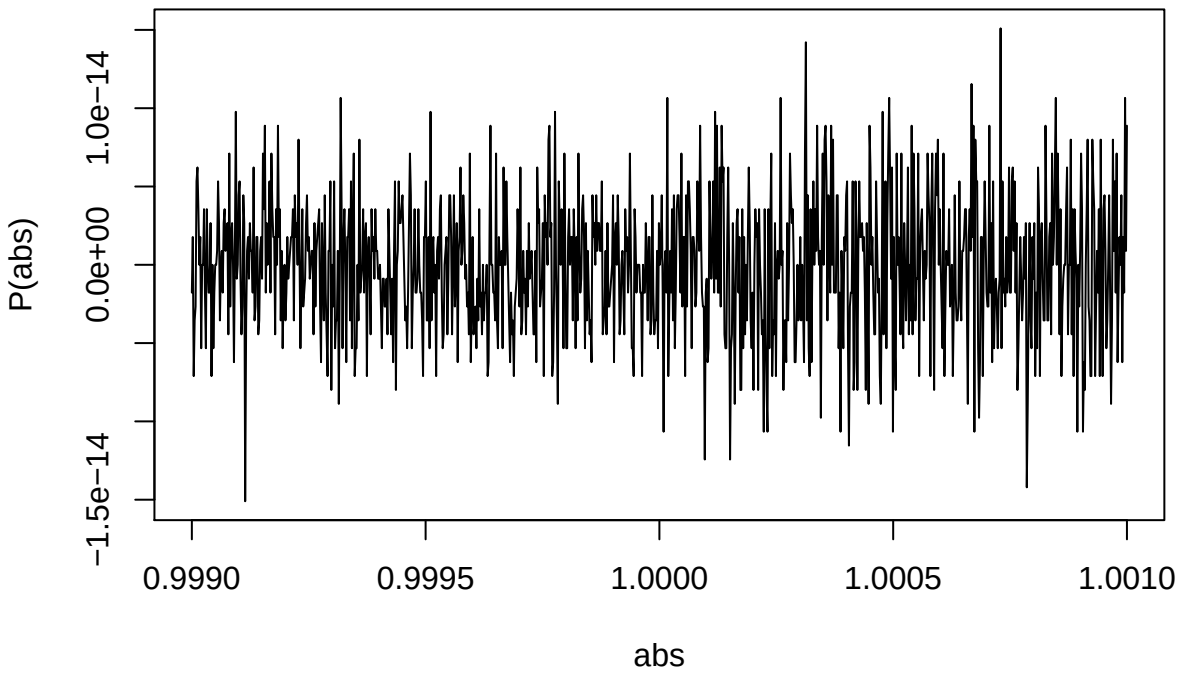
Traçons ce polynôme pour la curiosité.

```
Q <- function(x)
{
  x^6-6*x^5+15*x^4-20*x^3+15*x^2-6*x-1
}
abs <- seq(from = -50,to = 50,length.out = 1000)
plot(x = abs,y = Q(abs),type = "l")
```



Ce polynôme n'admet pas de racine (on peut le vérifier graphiquement). Nous avons donc factorisé P autant qu'il était possible.

```
abs <- seq(from = 0.999,to = 1.001,length.out = 1000)
plot(abs,P(abs),type = "l")
```



Nous voyons que P oscille très vite entre des valeurs positives et négatives. Cela impliquerait que P aurait de nombreuses racines alors que nous venons de montrer qu'il n'en avait qu'une. En fait, P n'admet qu'une seule racine, ce qui est représenté ici est la précision numérique de R.

Une intégrale qui se voudrait exacte

On veut montrer la relation suivante $I_{n+1} = 10I_n - \frac{1}{n}$ pour $I_n = \int_0^1 \frac{t^{n-1}}{10-t}\mathrm{d}t$ et ceci pour tout $n \geq 1$. On a,$\forall n \in \mathbb{N}$:

$$\begin{aligned}
10I_n - \frac{1}{n} &= 10\int_0^1 \frac{t^{n-1}}{10-t}\mathrm{d}t - \frac{1}{n} \\
&= \int_0^1 \frac{10t^{n-1}}{10-t}\mathrm{d}t - \frac{1}{n} \\
&= \int_0^1 \frac{10t^{n-1} - t^n + t^n}{10-t}\mathrm{d}t - \frac{1}{n} \\
&= \int_0^1 \frac{t^n}{10-t}\mathrm{d}t + \int_0^1 t^{n-1}\mathrm{d}t - \frac{1}{n} \\
&= I_{n+1} + \int_0^1 t^{n-1}\mathrm{d}t - \frac{1}{n} \\
&= I_{n+1} \quad \text{ce calcul est élémentaire.}
\end{aligned}$$

On a donc montré que pour tout $n \in \mathbb{N}$ la relation demandée.
Calcul de I_1 :

$$
\begin{aligned}
I_1 &= \int_0^1 \frac{1}{10-t}\mathrm{d}t \\
&= [-\ln(10-t)]_0^1 \\
&= \ln(10) - \ln(9) \\
&= \ln(\frac{10}{9})
\end{aligned}
$$

La dernière inégalité à trouver s'obtient en minorant et majorant le dénominateur de la fraction intégrée sur l'intervalle $[0, 1]$. Cela s'écrit de la manière suivante :

$$
\begin{gathered}
\int_0^1 \frac{t^{n-1}}{10}\mathrm{d}t \leq \int_0^1 \frac{t^{n-1}}{10-t}\mathrm{d}t \leq \int_0^1 \frac{t^{n-1}}{9}\mathrm{d}t \\
\frac{1}{10n} \leq \int_0^1 \frac{t^{n-1}}{10-t}\mathrm{d}t \leq \frac{1}{9n}
\end{gathered}
$$

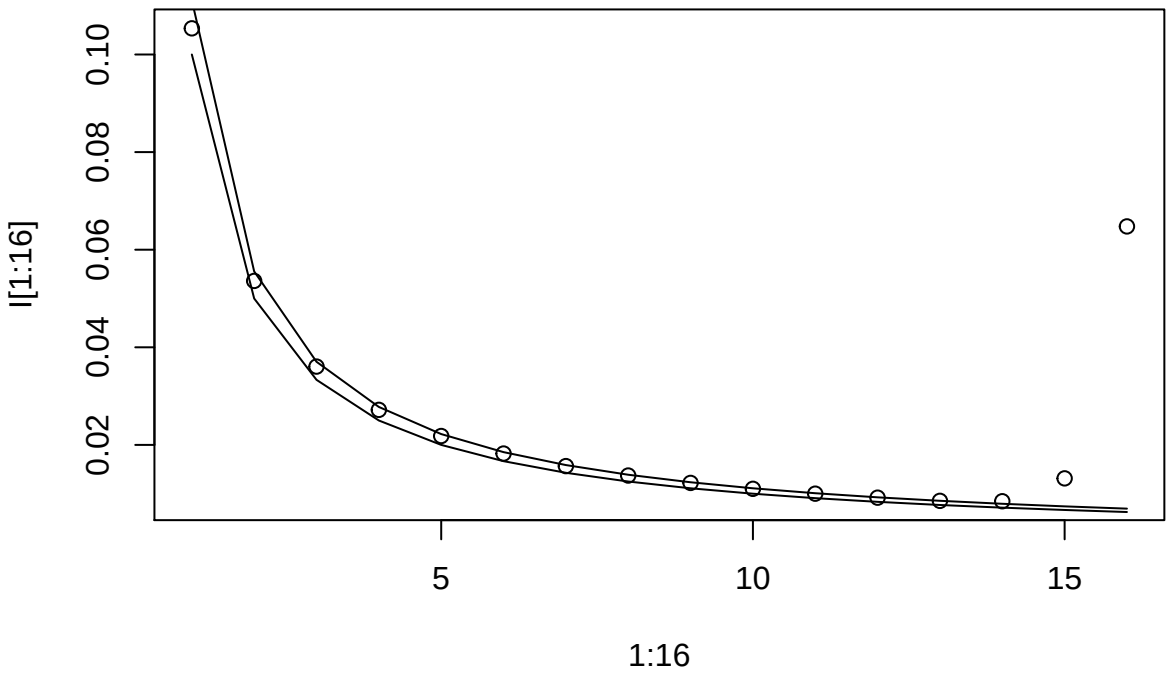
```
# Initialisation normale

I = 0
I[1] = log(10/9)
for(n in 1:20)
{
  I[n+1] <- 10*I[n]-1/n
}
I
```

```
##  [1] 1.053605e-01 5.360516e-02 3.605157e-02 2.718232e-02 2.182324e-02
##  [6] 1.823245e-02 1.565783e-02 1.372112e-02 1.221121e-02 1.100096e-02
## [11] 1.000962e-02 9.187137e-03 8.538032e-03 8.457245e-03 1.314388e-02
## [16] 6.477215e-02 5.852215e-01 5.793391e+00 5.787836e+01 5.787309e+02
## [21] 5.787259e+03
```

```
I[-1]-I[-length(I)]
```

```
##  [1] -5.175536e-02 -1.755359e-02 -8.869241e-03 -5.359080e-03 -3.590796e-03
##  [6] -2.574623e-03 -1.936706e-03 -1.509913e-03 -1.210245e-03 -9.913395e-04
## [11] -8.224862e-04 -6.491043e-04 -8.078694e-05  4.686636e-03  5.162827e-02
## [16]  5.204493e-01  5.208170e+00  5.208497e+01  5.208526e+02  5.208528e+03
```

```
{
  plot(1:16,I[1:16])
  lines(1:16,1/(9*1:16))
  lines(1:16,1/(10*1:16))
}
```



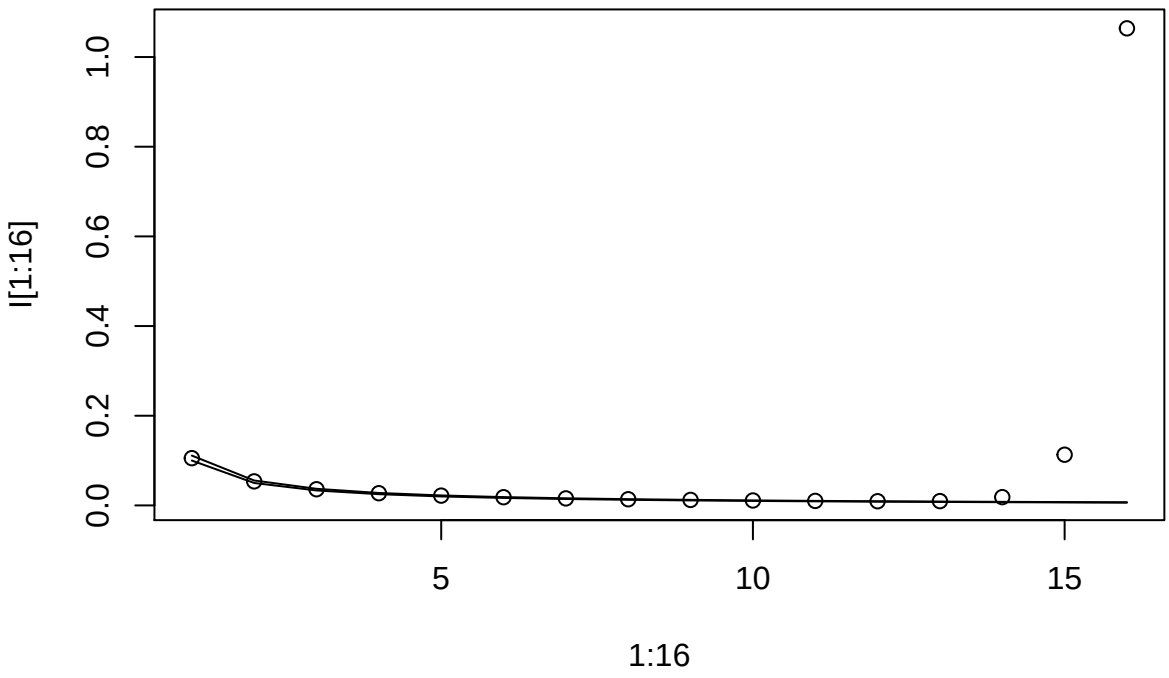

```
# Initialisation + 1e-15

I = 0
I[1] = log(10/9)+1e-15
for(n in 1:20)
{
  I[n+1] <- 10*I[n]-1/n
}
I
```

```
##  [1] 1.053605e-01 5.360516e-02 3.605157e-02 2.718232e-02 2.182324e-02
##  [6] 1.823245e-02 1.565783e-02 1.372113e-02 1.221131e-02 1.100196e-02
## [11] 1.001961e-02 9.287057e-03 9.537233e-03 1.844925e-02 1.130640e-01
## [16] 1.063973e+00 1.057723e+01 1.057135e+02 1.057079e+03 1.057074e+04
## [21] 1.057073e+05
```

```
I[-1]-I[-length(I)]
```

```
##  [1] -5.175536e-02 -1.755359e-02 -8.869241e-03 -5.359080e-03 -3.590796e-03
##  [6] -2.574622e-03 -1.936697e-03 -1.509823e-03 -1.209346e-03 -9.823467e-04
## [11] -7.325581e-04  2.501763e-04  8.912020e-03  9.461470e-02  9.509089e-01
## [16]  9.513256e+00  9.513623e+01  9.513656e+02  9.513659e+03  9.513659e+04
```

```
{
  plot(1:16,I[1:16])
  lines(1:16,1/(9*1:16))
  lines(1:16,1/(10*1:16))
}
```



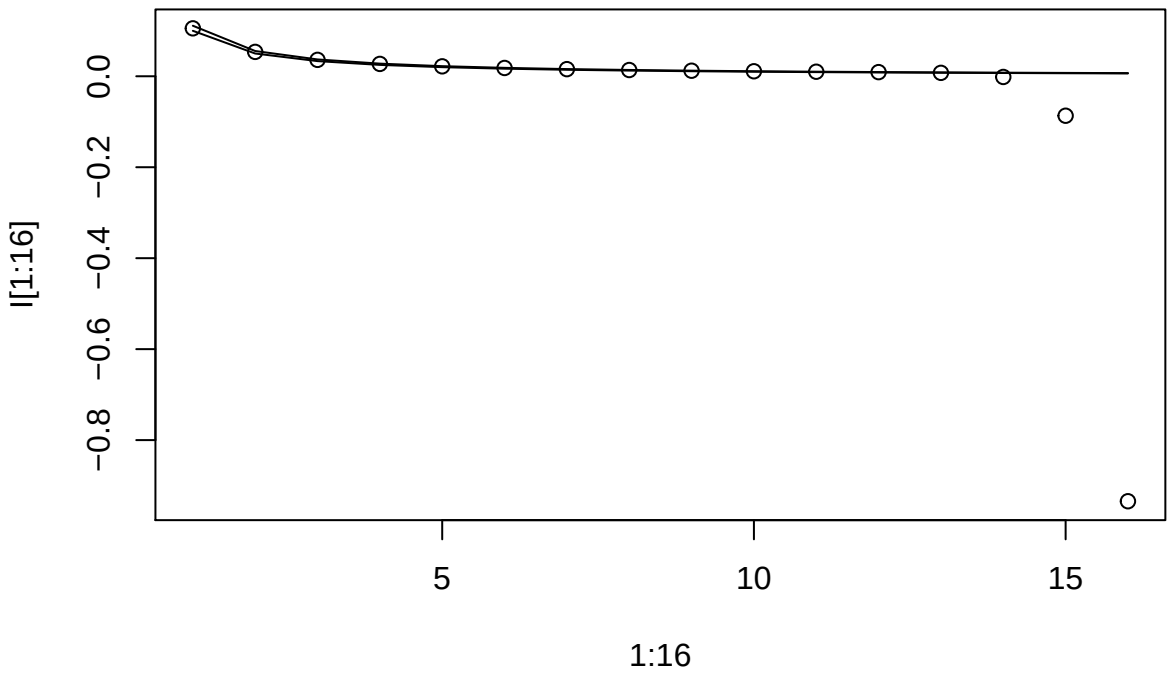

```
# Initialisation - 1e-15

I = 0
I[1] = log(10/9)-1e-15
for(n in 1:20)
{
  I[n+1] <- 10*I[n]-1/n
}
I
```

```
##  [1]  1.053605e-01  5.360516e-02  3.605157e-02  2.718232e-02  2.182324e-02
##  [6]  1.823245e-02  1.565783e-02  1.372111e-02  1.221111e-02  1.099996e-02
## [11]  9.999631e-03  9.087216e-03  7.538831e-03 -1.534762e-03 -8.677619e-02
## [16] -9.344286e-01 -9.406786e+00 -9.412668e+01 -9.413224e+02 -9.413276e+03
## [21] -9.413281e+04
```

```
I[-1]-I[-length(I)]
```

```
##  [1] -5.175536e-02 -1.755359e-02 -8.869241e-03 -5.359080e-03 -3.590796e-03
##  [6] -2.574624e-03 -1.936715e-03 -1.510003e-03 -1.211144e-03 -1.000332e-03
## [11] -9.124143e-04 -1.548385e-03 -9.073593e-03 -8.524143e-02 -8.476524e-01
## [16] -8.472357e+00 -8.471990e+01 -8.471957e+02 -8.471954e+03 -8.471954e+04
```

```
{
  plot(1:16,I[1:16])
  lines(1:16,1/(9*1:16))
  lines(1:16,1/(10*1:16))
}
```



On voit que le calcul numérique diverge au bout d'un certain nombre d'itérations. Les points des graphiques (résultats des calculs numériques) ne sont plus compris entre les lignes (résultats des calculs théoriques).

Des inversions linéaires très approximatives de systèmes linéaires pourtant bien simples

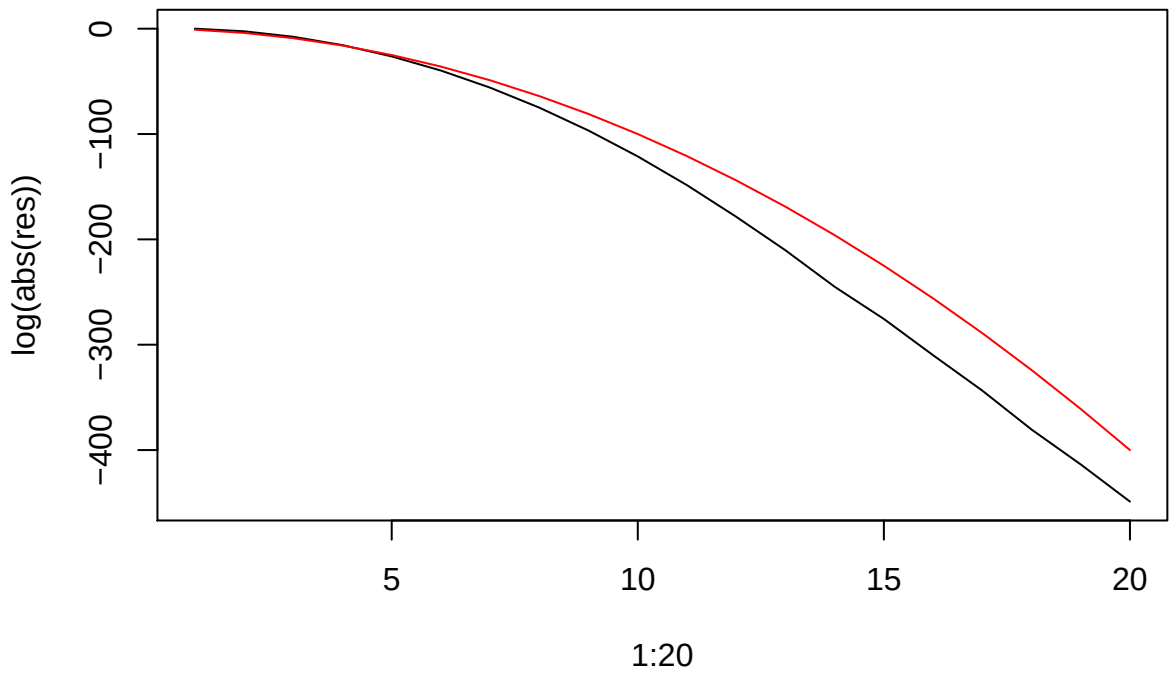
Il est assez simple de vérifier que $\forall i, j \in [1, n]$, $h_{ij}^{(n)} = h_{ji}^{(n)}$. Il suffit d'écrire les formule des deux coefficients pour s'en rendre compte. La matrice est donc symétrique.
On écrit le code pour coder la matrice H :

```
n <- 10 # Rappel : on peut faire choisir n par l'utililsateur avec la fonction readline de R, on prendr
matl <- matrix(data = rep(x = 1:n,n),nrow = n,ncol = n,byrow = FALSE) # création matrice indice ligne
matc <- matrix(data = rep(x = 1:n,n),nrow = n,ncol = n,byrow = TRUE) # création matrice indice colonne
mat1 <- matrix(data = 1,nrow = n,ncol = n) # création matrice de 1
matden <- matl+matc-mat1 # création de la matrice contenant tous les dénominateurs de H
H <- mat1/matden # création de H
rm(matc,matl,mat1,matden)
```

On code maintenant la représentation graphique du déterminant demandée :

```
res <- c()
for(i in 1:20)
{
  matl <- matrix(data = rep(x = 1:i,i),nrow = i,ncol = i,byrow = FALSE)
  matc <- matrix(data = rep(x = 1:i,i),nrow = i,ncol = i,byrow = TRUE)
  mat1 <- matrix(data = 1,nrow = i,ncol = i)
```

```
  matden <- matl+matc-mat1
  H <- mat1/matden
  res <- c(res,det(H))
}

{
  plot(1:20,log(abs(res)),type = "l")
  lines(1:20,-(1:20)^2,col="red")
}
```



La vitesse de décroissance du déterminant de H semble dépendre de n. Notamment, on voit que la décroissance est supérieure à e^{-n^2}. On va se servir de la formule théorique pour se faire une meilleure idée. On utilise la formule du déterminant des matrices de Cauchy. On regarde comment le numérateur de la formule se comporte dans notre cas, on a :

$$\begin{aligned}\Pi_{1\leq i<j\leq n}(j-i)(j-i) &= \Pi_{1\leq j\leq n}(j-1)(j-2)...(j-(j-1))(j-1)(j-2)...(j-(j-1)) \\ &= \Pi_{1\leq j\leq n}((j-1)!)^2\end{aligned}$$

Pour le dénominateur, on obtient :

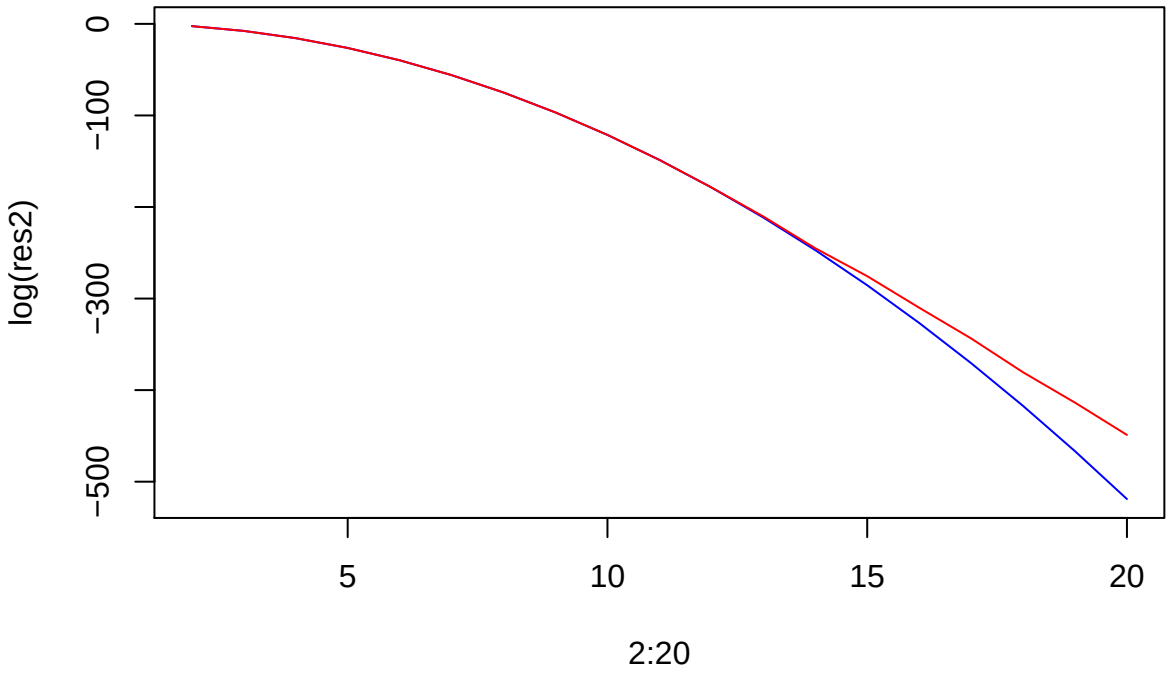
$$\begin{aligned}\Pi_{1\leq i,j\leq n}(j+i-1) &= \Pi_{1\leq j\leq n} j(j+1)(j+2)...(j+n-1) \\ &= \Pi_{1\leq j\leq n}\frac{(j+n-1)!}{(j-1)!}\end{aligned}$$

On obtient donc comme formule finale :

$$det(H^{(n)}) = \Pi_{1\leq j\leq n}\frac{((j-1)!)^3}{(j+n-1)!}$$

Codons le calcul de cette formule :

```
res2 <- c()
n <- 2
while(n <= 20)
{
  ele <- c()
  for(j in 1:n)
  {
    ele <- c(ele,factorial(j-1)^3/(factorial(j+n-1)))
  }
  ele <- prod(ele)
  res2 <- c(res2,ele)
  n <- n+1
}
{
  plot(2:20,log(res2),type="l",col="blue")
  lines(2:20,log(abs(res[2:20])),col="red")
}
```



On voit que le calcul théorique contrairement au calcul approché n'est pas soumis au probléme de précision de la machine. On voit l'illustration de la précision numérique dans le fait qu'au bout d'un certain n, la ligne bleue (calcul théorique) et la ligne rouge (calcul numérique) diverge.

Construction de la matrice $H^{(n)}$ pour $n = 10$:

```
n <- 10
matl <- matrix(data = rep(x = 1:n,n),nrow = n,ncol = n,byrow = FALSE)
```

```
matc <- matrix(data = rep(x = 1:n,n),nrow = n,ncol = n,byrow = TRUE)
mat1 <- matrix(data = 1,nrow = n,ncol = n)
matden <- matl+matc-mat1
H <- mat1/matden
rm(matc,matl,mat1,matden)
```

Resolution numérique du système linéaire :

```
B <- rep(0,n)
X <- solve(a = H,b = B)
X
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

Théoriquement, on raisonne de la manière suivante. H a un déterminant petit certes mais non nul. La formule explicite du déterminant de H calculée plus haut nous le montre. Donc H est inversible. On a donc que :

$$H^{(n)}X = B_n \implies X = (H^{(n)})^{-1}B_n$$

Comme nous avons fixé B_n étant égal au vecteur nul, cela implique X est égal au vecteur nul.

```
n = 10
B = runif(n,-1,1)*1e-12
# vecteur de valeurs numériquement proches de 0 (négatives ou positives)
XX <- solve(a = H,b = B)
norm(as.matrix(X-XX),type = "O") # différence en norme 1
```

```
## [1] 16.03168
```

```
norm(as.matrix(X-XX),type = "I") # différence en norme infinie
```

```
## [1] 4.624179
```

```
norm(as.matrix(X-XX),type = "2") # différence en norme deux
```

```
## [1] 7.489609
```

```
norm(as.matrix(X-XX),type = "F") # différence en norme de Frobenius
```

```
## [1] 7.489609
```

On se rend compte que les différences en norme sont élevées !!!

Effectuons le même travail pour le vecteur unitaire (vecteur de 1).

```
B <- rep(1,n)
X <- solve(a = H,b = B)
B = 1+runif(n,-1,1)*1e-12
XX <- solve(a = H,b = B)
norm(as.matrix(X-XX),type = "O") # différence en norme 1
```

```
## [1] 4.246399
```

```
norm(as.matrix(X-XX),type = "I") # différence en norme infinie
```

```
## [1] 1.222478
```

```
norm(as.matrix(X-XX),type = "2") # différence en norme deux
```

```
## [1] 1.98675
```

```r
norm(as.matrix(X-XX),type = "F") # différence en norme de Frobenius
```

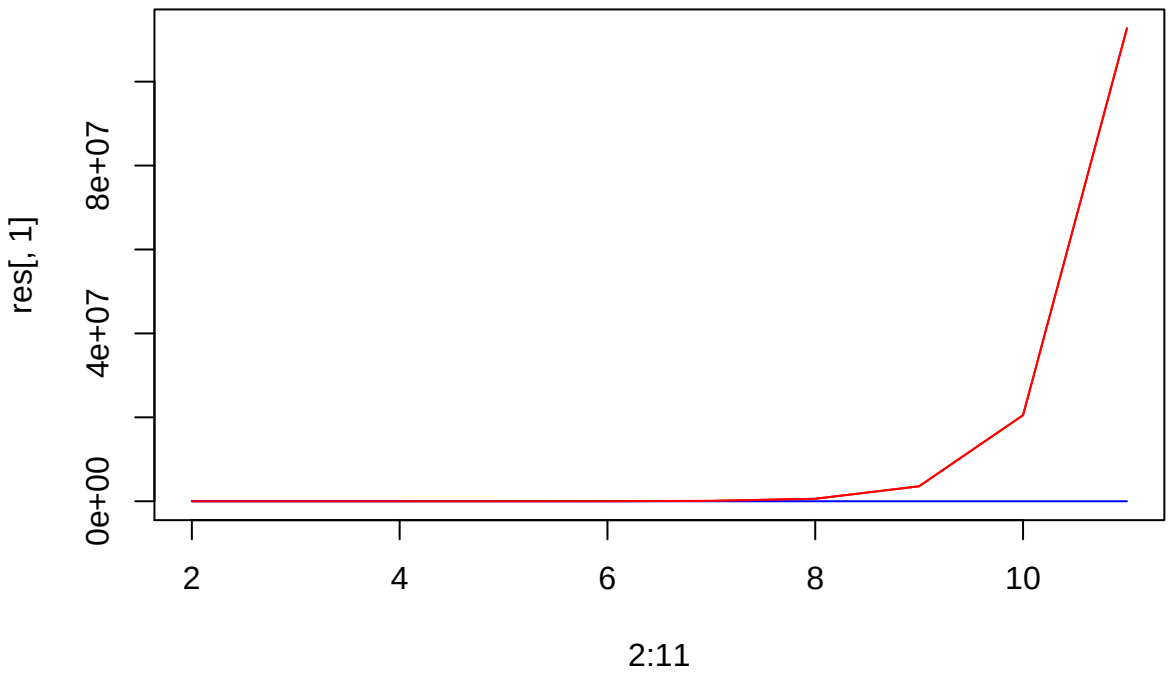
```
## [1] 1.98675
```

On voit que les différences en normes sont exceptionnellement grandes !!!

On va calculer les conditionnements pour les différentes matrices $H^{(n)}$.

Attenzione !!! Il y a une coquille dans l'énoncé, $||A||_2 = \sup_{X\in\mathbb{R}^n} \sqrt{\frac{{}^tX\,{}^tAAX}{{}^tXX}}$.
On peut le coder de la manière suivante pour c_1 et c_∞ :

```r
res <- matrix(0,nrow = length(2:11),ncol = 3)
for(n in 2:11)
{
  matl <- matrix(data = rep(x = 1:n,n),nrow = n,ncol = n,byrow = FALSE) # création matrice indice ligne
  matc <- matrix(data = rep(x = 1:n,n),nrow = n,ncol = n,byrow = TRUE) # création matrice indice colonn
  mat1 <- matrix(data = 1,nrow = n,ncol = n) # création matrice de 1
  matden <- matl+matc-mat1 # création de la matrice contenant tous les dénominateurs de H
  H <- mat1/matden # création de H
  rm(mat1,matc,matl,matden)
  c1 <- max(apply(X = H,MARGIN = 2,FUN = "sum"))*max(apply(X = solve(H),MARGIN = 2,FUN = "sum"))
  res[n-1,1] <- c1
  c2 <- max(sqrt(abs(eigen(t(H)%*%H)$values)))
  res[n-1,2] <- c2
  c3 <- max(apply(X = H,MARGIN = 1,FUN = "sum"))*max(apply(X = solve(H),MARGIN = 2,FUN = "sum"))
  res[n-1,3] <- c3
}
{
  plot(2:11,res[,1],type = "l",col = "red")
  lines(2:11,res[,2],col= "blue")
  lines(2:11,res[,3], col ="red")
}
```




```
kappa(z = H,norm = "O")
```

```
## [1] 7.962404e+14
```

```
kappa(z = H,norm = "I")
```

```
## [1] 1.670556e+13
```

```
rcond(x = H,norm = "I")
```

```
## [1] 8.127637e-16
```

```
rcond(x = H,norm = "O")
```

```
## [1] 8.127637e-16
```

On voit que les résultats de`kappa` et `rcond` sont plus précis.

Exercice 1

La matrice A pour $n = 5$ de l'enoncé s'écrit donc ainsi :

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

C'est matrice dont la diagonale et la diagonale supérieure vaut 1. Le reste des coefficient vaut 0.
Regardons pour $n = 3$ puis pour $n = 4$ l'allure de l'inverse de A_n. Cela nous donnera l'intuition pour une forme générale de l'inverse de A_n. On commence pour $n = 3$:

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & -1 & 1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$

Pour $n = 3$, A^{-1} s'écrit donc :

$$\begin{pmatrix} 1 & -1 & 1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$

En procédant de la même manière pour $n = 4$, on obtiendra que A^{-1} s'écrit :

$$\begin{pmatrix} 1 & -1 & 1 & -1 \\ 0 & 1 & -1 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Que peut on en déduire ?
On peut en déduire que la formule générale de A^{-1} pour un n quelconque est une matrice dont les coefficients diagonaux valent 1. Ceux de la diagonale supérieure valent -1. Ceux de la diagonale à une distance de 2 de la diagonale valent 1. Etc... En fait on peut résumer comme suit, les coefficients des diagonales supérieures qui sont à une distance paire de la diagonale principale de A^{-1} valent 1. Les coefficients des diagonales supérieures qui sont à une distance impaire de la diagonale principale de A^{-1} valent -1.

Maintenant que nous avons la forme générale de A^{-1}, calculons c_1 et c_∞ de A.
Remarquons immédiatement que $||A||_1$ et $||A||_\infty$ valent 2. En effet, ces normes reviennent à calculer respectivement la plus grande somme des coefficients (en valeur absolue) d'une colonne de A et la plus grande somme des coefficients (en valeur absolue) d'une ligne de A. Telle que la matrice A est construite, le maximum vaut 2 pour la somme des coefficients d'une ligne ou d'une colonne.
Remarquons maintenant que pour A^{-1}, il assez évident de déduire que la valeur des normes 1 et infinie vaut n. Cela se déduit des définitions de ces normes et de l'allure générale A^{-1} que nous avons mise en évidence plus haut.
On obtient finalement que :

$$c_1(A) = c_\infty(A) = 2 \times n$$

Attenzione !!! : coquille dans l'énoncé ?

Heureusement pour nous, A et A^{-1} sont des matrices triangulaires. Le calcul de leur déterminant est donc le produit de leurs coefficients diagonaux. Cela signifie que $det(A) = 1$ pour tout n. Bien que ce ne soit pas demandé, notez qu'il en est de même pour A^{-1}.

```
# fonction qui construit les matrices A_n
Anne <- function(n)
{
```

```
  A <- matrix(0,ncol = n,nrow = n)
  diag(A) <- 1
  diag(A[1:(n-1),2:n]) <- 1
  return(A)
}

# fonction qui calcule le conditionnement pour une matrice donnée
cond <- function(A,norme)
{
  res <- norm(x = A,type = norme)*norm(x = solve(A),type = norme)
  return(res)
}

# Création du vecteur des n
n <- 3:100
# calcul pour chaque n du conditionnement pour la norme 1
for(i in 1:length(n))
{
  A <- Anne(n[i])
  n[i] <- cond(A = A,norme = "O")
}
# on affiche le résultat
n
```

```
##  [1]   6   8  10  12  14  16  18  20  22  24  26  28  30  32  34  36  38  40  42
## [20]  44  46  48  50  52  54  56  58  60  62  64  66  68  70  72  74  76  78  80
## [39]  82  84  86  88  90  92  94  96  98 100 102 104 106 108 110 112 114 116 118
## [58] 120 122 124 126 128 130 132 134 136 138 140 142 144 146 148 150 152 154 156
## [77] 158 160 162 164 166 168 170 172 174 176 178 180 182 184 186 188 190 192 194
## [96] 196 198 200
```

```
# Création du vecteur des n
n <- 3:100
# on refait de même pour la norme infinie
for(i in 1:length(n))
{
  A <- Anne(n[i])
  n[i] <- cond(A = A,norme = "I")
}
# on affiche le résultat
n
```

```
##  [1]   6   8  10  12  14  16  18  20  22  24  26  28  30  32  34  36  38  40  42
## [20]  44  46  48  50  52  54  56  58  60  62  64  66  68  70  72  74  76  78  80
## [39]  82  84  86  88  90  92  94  96  98 100 102 104 106 108 110 112 114 116 118
## [58] 120 122 124 126 128 130 132 134 136 138 140 142 144 146 148 150 152 154 156
## [77] 158 160 162 164 166 168 170 172 174 176 178 180 182 184 186 188 190 192 194
## [96] 196 198 200
```

Notez qu'avec la manière dont j'ai codé les matrices A_n, je ne peux pas créer des matrices 2×2. En effet la fonction `Anne` renvoie une erreur. C'est parce que la diagonale supérieure d'une matrice 2×2 est un scalaire et R ne supporte pas ce format pour la fonction `diag`.

On résoud théoriquement le système :

$$AX = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

On note $X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$. Résoudre cette équation matricielle revient à résoudre le système suivant :

$$\begin{aligned} x_1 + x_2 &= 1 \\ x_2 + x_3 &= 0 \\ &\vdots \\ x_{n-1} + x_n &= 0 \\ x_n &= 0 \end{aligned}$$

Une solution évidente de ce système est le vecteur $X = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$. Ceci est valable pour tout n. Passons maintenant à l'application numérique.

```
# on met les trois n demandés dans un vecteur
n <- c(10,50,100)
# on résoud le système pour chaque n
for(i in 1:length(n))
{
  A <- Anne(n[i])
  b <- rep(0,n[i])
  b[1] <- 1
  print(solve(a = A,b = b))
}
```

```
##  [1] 1 0 0 0 0 0 0 0 0 0
##  [1] 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [39] 0 0 0 0 0 0 0 0 0 0 0 0
##   [1] 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

La méthode numérique renvoie donc le même résultat !