MACS 205 : interpolation et quadrature
Étude de Cas

DOROZHKO Anton

'22/02/2018'

Todo list

donnez un ordre de grandeur la précision de votre estimation. 18

1 Introduction

L'étude de cas consiste à utiliser les résultats théoriques et les méthodes numériques vus en cours pour

1. l'estimation d'une densité de probabilité, étant donné un histogramme formé à partir d'un échantillon de variables indépendantes et identiquement distribuées selon cette densité,
2. l'estimation de la probabilité pour la variable aléatoire associée d'appartenir à un certain intervalle, étant donnée sa densité de probabilité.

On s'intéressera en particulier à la loi Beta de paramètres $\alpha > 0$, $\beta > 0$, de densité

$$\phi_{\alpha,\beta}(x) = x^{\alpha-1}(1-x)^{\beta-1}/B_{\alpha,\beta} \tag{1}$$

où $B_{\alpha,\beta}$ est la constante de normalisation

$$B_{\alpha,\beta} = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} \tag{2}$$

Dans toute la suite, on prendra

$$\alpha = 1.7, \beta = 5.1$$

La première partie du sujet vous amènera à utiliser les méthodes d'interpolation vues en cours sur la densité $\phi_{1.7,5.1}$, ainsi que sur sa "version bruitée", c'est-à-dire l'histogramme d'un échantillon i.i.d $(X_i, i = 1, ..., n)$ suivant la loi Beta de paramètre $(1.7, 5.1)$. La deuxième partie consiste à exploiter les méthodes d'intégration numériques pour approcher la quantité

$$p_{0.1} = \mathbb{P}(0.1 \leq X_1 \leq 0.9) = \int_{0.1}^{0.9} \phi_{1.7,5.1}(t)dt \tag{3}$$

Consignes

L'ensemble de votre travail devra être rassemblé dans un seul dossier portant votre nom, et archivé au format .zip ou tar.gz. Ce dossier doit contenir

— Un Notebook R au format pdf ou html, contenant contenant les réponses rédigées au questions, les figures et résultats numériques obtenus.

— La source du notebook (au format **.R** ou **.Rmd**, exécutable par le correcteur depuis le répertoire de travail sans modification. *Attention aux chemins d'accès ! ceux-ci doivent être relatifs au répertoire courant*

— Toutes les fonctions utilisées dans le notebook (y compris les fonctions codées en cours) rassemblées dans un fichier **functions_ETC2018.R**

— L'histogramme des échantillons générés, sauvegardé dans un fichier **hist1.rda**.

Le non respect des consignes ci-dessus sera pénalisé d'un malus variant de 1 à 5 points en fonction du temps perdu par le correcteur pour débugger.

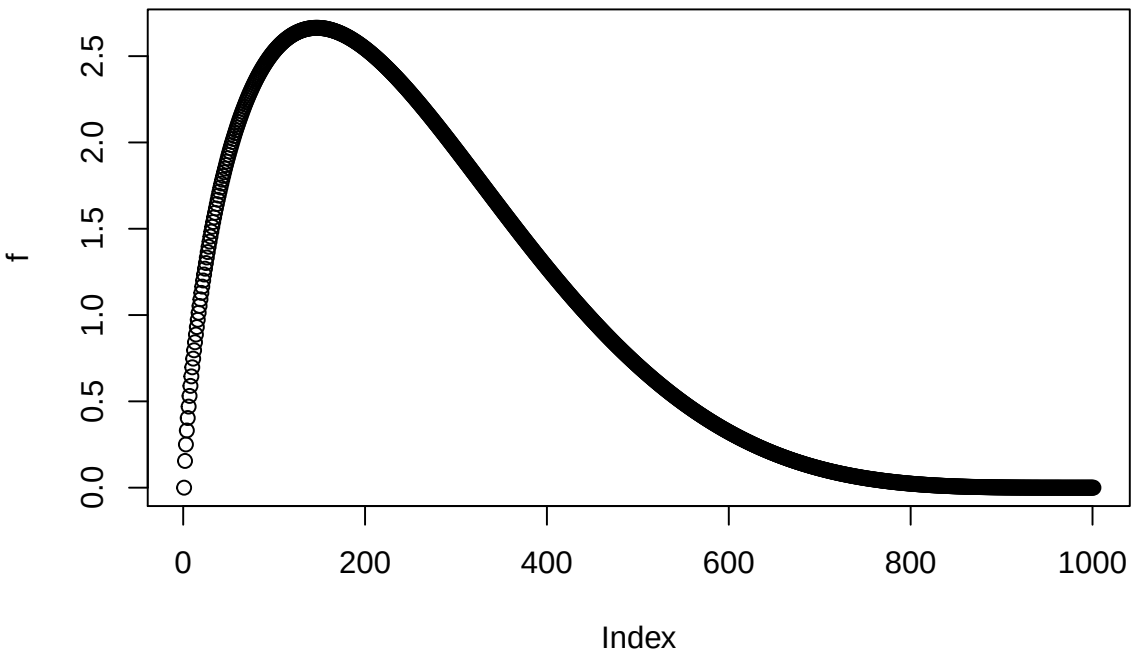
Les dossiers doivent être déposés sur l'espace de partage sur Eole avant le dimanche 4 mars 2016, 23h59. Chaque heure de retard vous coûtera deux points (sur vingt).

En cas de problème d'ordre technique, vous pouvez envoyer un mail à l'adresse : anne.sabourin@telecom-paristech.fr, avec en objet l'intitulé "EDC MACS205".

2 Génération des données et fonctions préliminaires

1. Implémentez une fonction densite qui à partir d'un vecteur contenant $n \geq 1$ réels $(x_1, ..., x_n)$, appartenant à $[0, 1]$, retourne le vecteur $(\phi_{1.7,5.1}(x_1), ..., \phi_{1.7,5.1}(x_n))$. On utilisera pour cela l'expression (1) et la fonction **dbeta** de **R**.

```
# TODO: to functions_ETC2018.R
densite = function (x) {
 phi = dbeta(x, shape1 = 1.7, shape2 = 5.1)
 return (phi)
}

f = densite(seq(0,1,0.001))
plot(f)
```



On considérera aussi une version approchée de $\phi_{1.7,5.1}$, basée sur un histogramme construit à partir de N_w échantillons i.i.d w_i de loi $Beta(1.7, 5.1)$. On définit pour tout $x \in [0, 1[$,

$$\hat{\phi}^{N_w}_{1.7,5.1} = (2^8/N_w) \sum_{i=1}^{N_w} \mathbf{1}_{w_i \in [k(x)2^{-8},(k(x)+1)2^{-8}[}$$

où $k(x)$ est la partie entière de $2^8 x$.

2. Soit $x \in [0,1[$. Quelle est la limite presque sûr de $\hat{\phi}^{N_w}_{1.7,5.1}(x)$ lorsque N_w tend vers l'infini ? On notera $\hat{\phi}^{\infty}_{1.7,5.1}(x)$ cette limite.

Reponse

$\hat{\phi}^{\infty}_{1.7,5.1}(x) = \lim_{N_w \to \infty} (2^8/N_w) \sum_{i=1}^{N_w} \mathbf{1}_{w_i \in [k(x)2^{-8},(k(x)+1)2^{-8}[} = 2^8 \int_{k(x)2^{-8}}^{(k(x)+1)2^{-8}} \phi_{1.7,5.1}(t)dt$

3. Donner une majoration de $\hat{\phi}^{\infty}_{1.7,5.1}(x) - \phi_{1.7,5.1}(x)$ en fonction de $x \in [0,1[$.

Reponse

$\hat{\phi}^{\infty}_{1.7,5.1}(x) - \phi_{1.7,5.1}(x) = max(\phi_{1.7,5.1}(k(x)2^{-8}), \phi_{1.7,5.1}((k(x)+1)2^{-8})) - \phi_{1.7,5.1}(x)$

Pour générer l'histogramme et l'estimateur associé, il vous est fourni un script **ini.R**, et deux fonctions **hist__value**, **generation__X** placées dans un fichier **ini__functions.R**. Récupérer ces fichiers (répertoire **script__et__fonctions.tar.gz** sur le site pédagogique) et placez-les dans votre répertoire de travail. Exécutez le script **ini.R** pour générer 10^7 échantillons i.i.d. selon un loi $Beta(1.7, 5.1)$ et l'histogramme associé. L'histogramme doit s'afficher, et un fichier **hist1.rda** sera créé dans votre répertoire de travail. **n'oubliez pas de joindre ce dernier à votre rendu !**

4. On sera amené dans la suite à utiliser les fonctions implémentées en cours, en leur passant en argument la fonction **FUN = hist__value**. Que fait la fonction **hist__value** ?

Reponse Elle retourne la valeur de l'histogramme associé au x

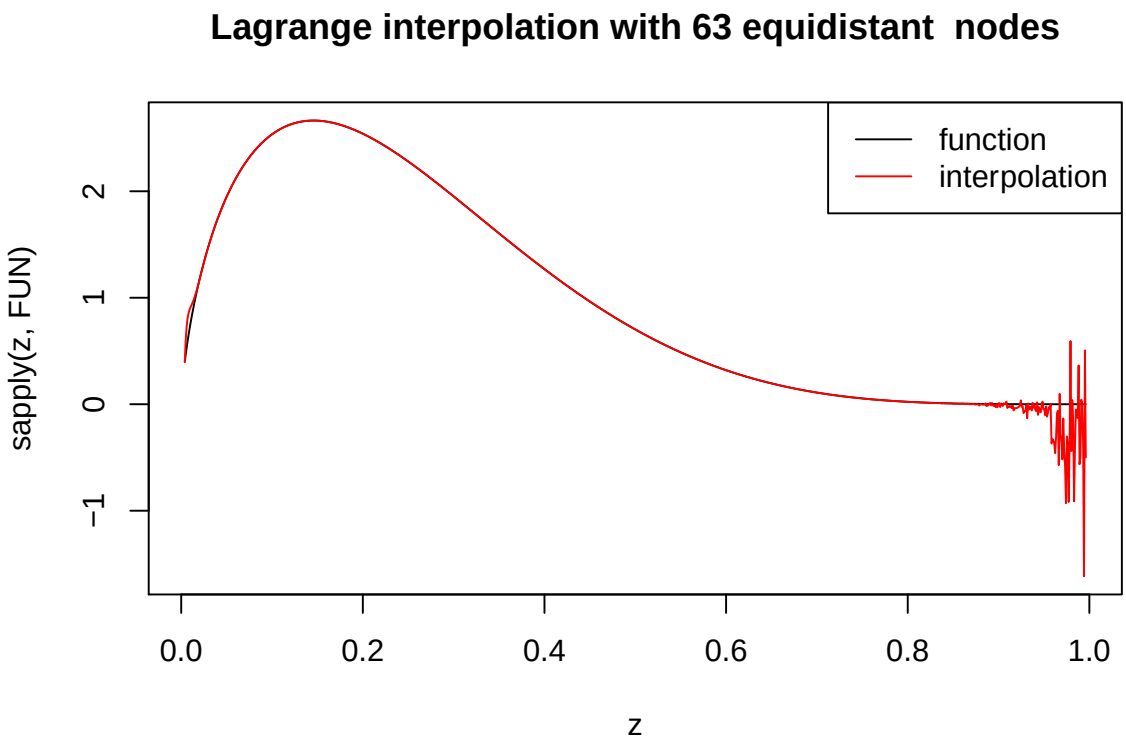
3 Interpolation polynomiale

3.1 Interpolation à partir de l'expression exacte de $\phi_{1.7,5.1}(x)$

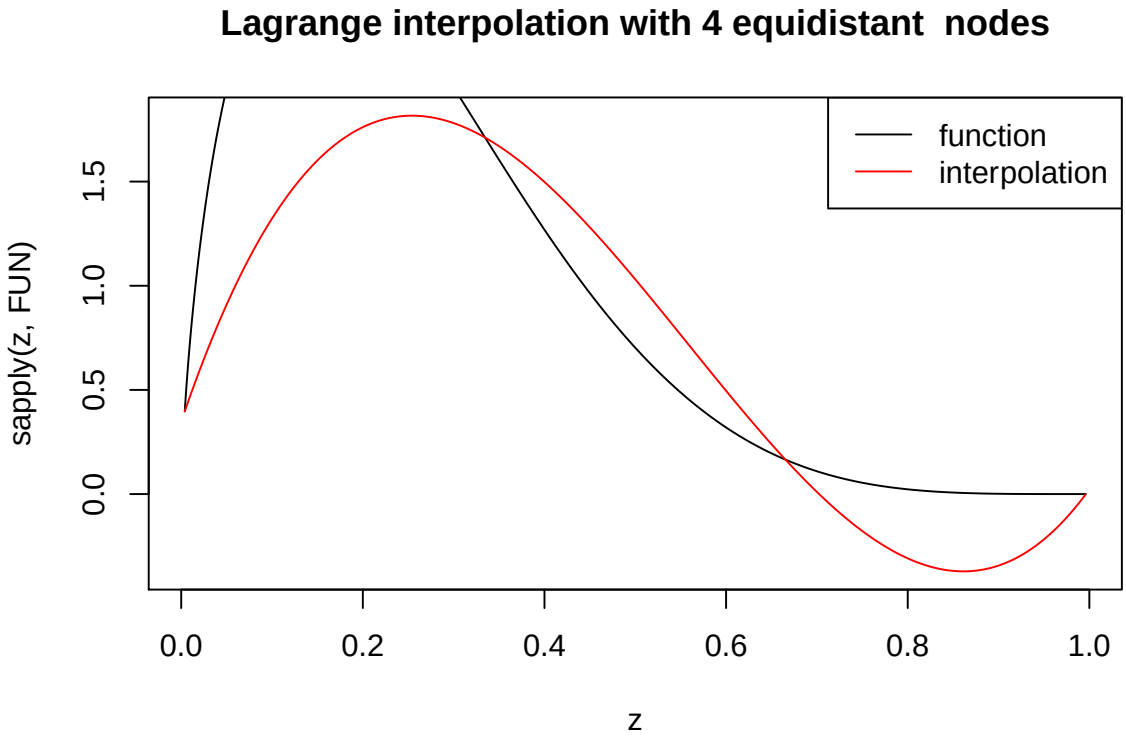
1. Construisez le polynôme d'interpolation de Lagrange de la fonction $\phi_{1.7,5.1}(x)$ avec des noeuds d'interpolation équi-distants, évalué en 1000 points de l'intervalle $[a = 2^{-8}, b = 2^{-8}]$. Faites varier le degré du polynôme d'interpolation entre 2 et 50.
   (a) A partir de quel degré voyez- vous apparaître le phénomène de Runge ?
   (b) Affichez trois exemples mettant en évidence les défauts de l'interpolation avec un degré trop bas ou trop haut. Vous tracerez sur chaque graphique le résultat de l'interpolation et le graphe de la fonction de référence.

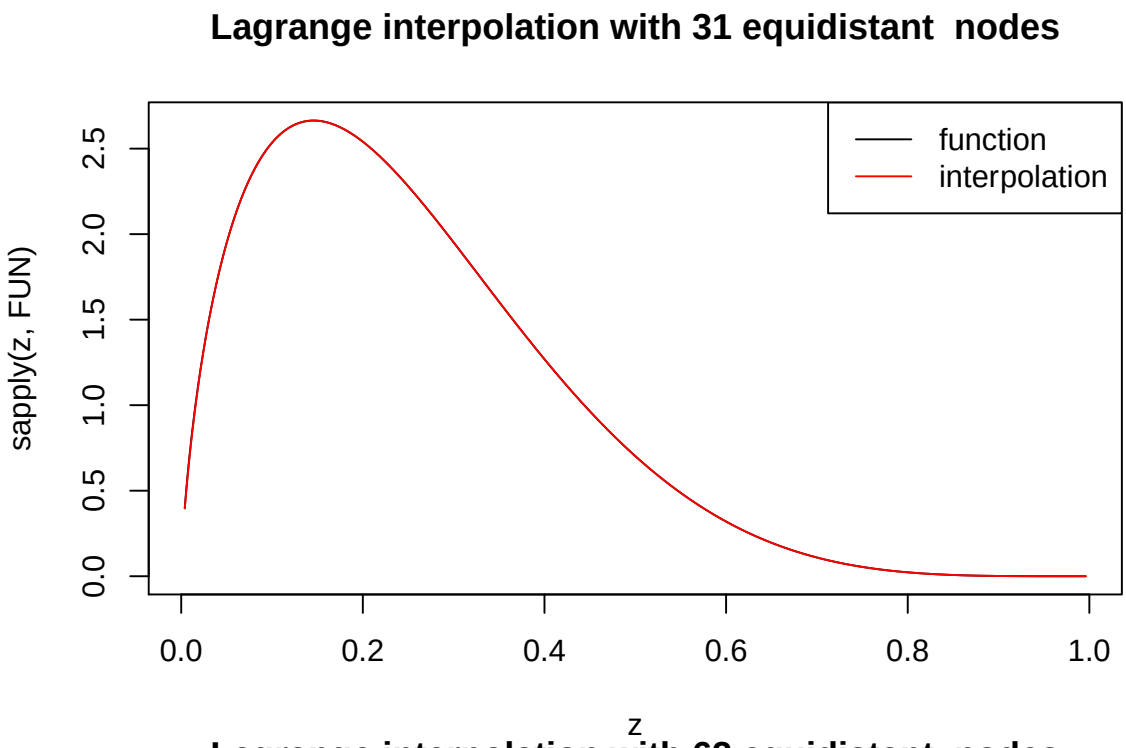
Réponse

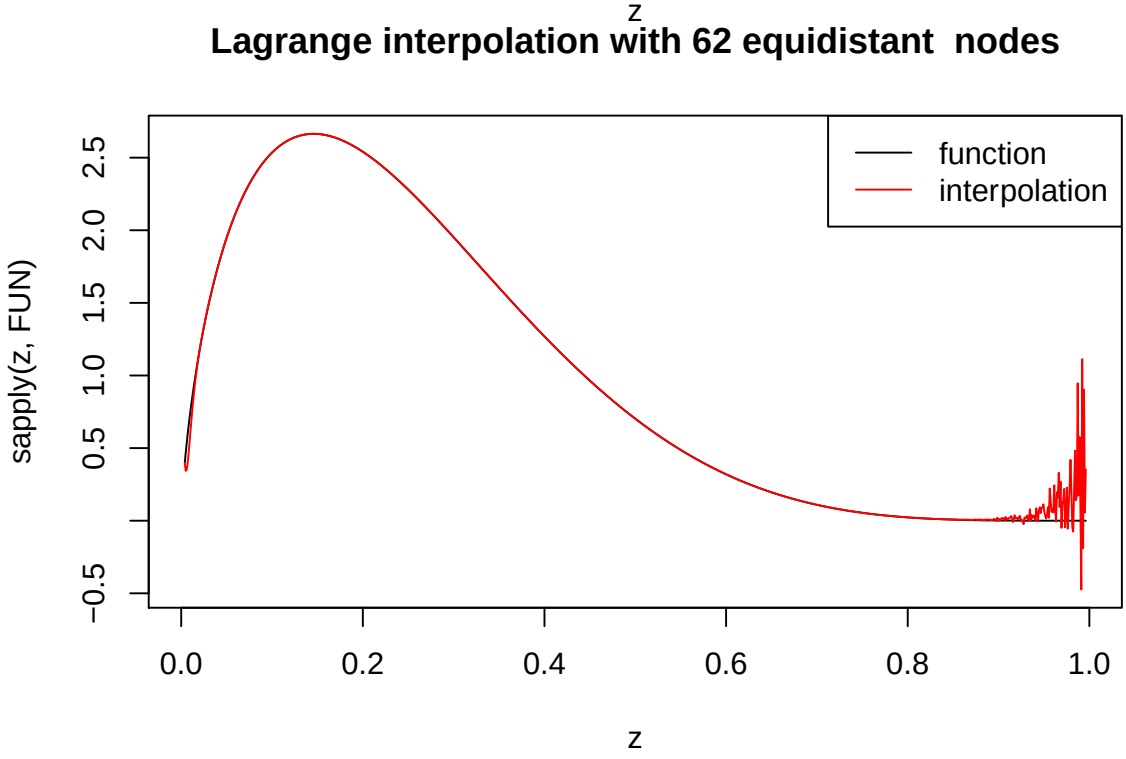
a) Le phénomène de Runge commence à apparaître a partir de degré 60


Lagrange interpolation with 63 equidistant nodes
function
interpolation
sapply(z, FUN)
−1
0
1
2
0.0
0.2
0.4
0.6
0.8
1.0
z

b)


Lagrange interpolation with 4 equidistant nodes
function
interpolation
sapply(z, FUN)
0.0
0.5
1.0
1.5
0.0
0.2
0.4
0.6
0.8
1.0
z

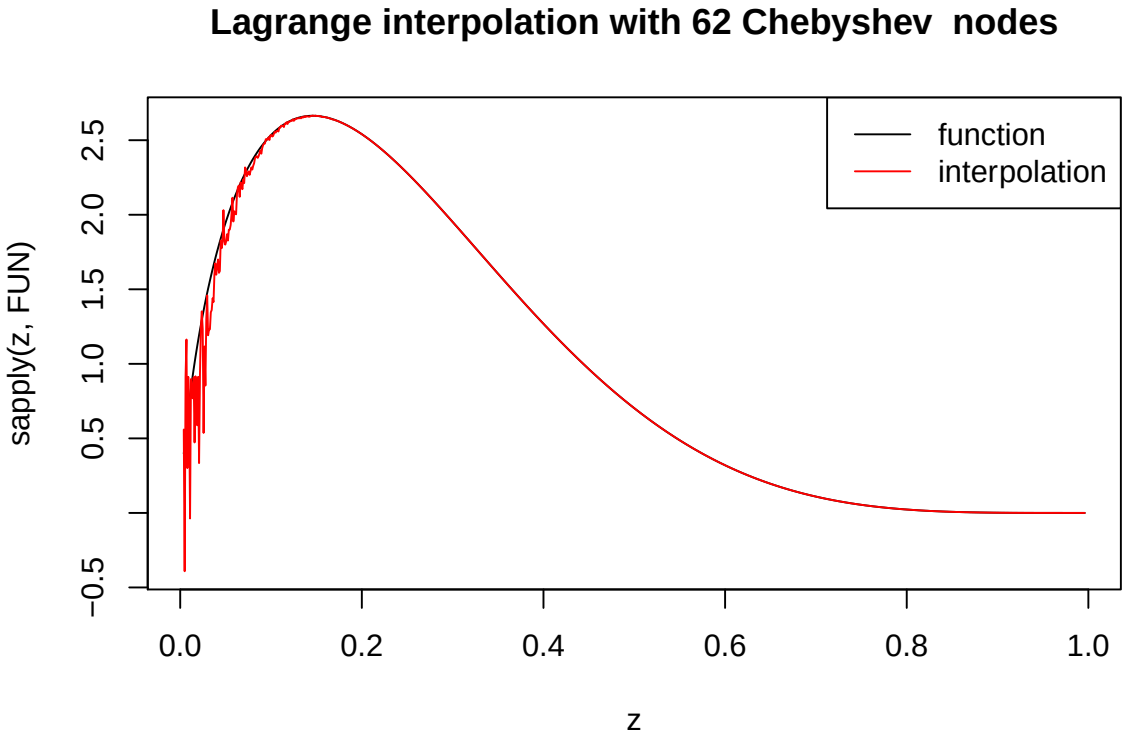




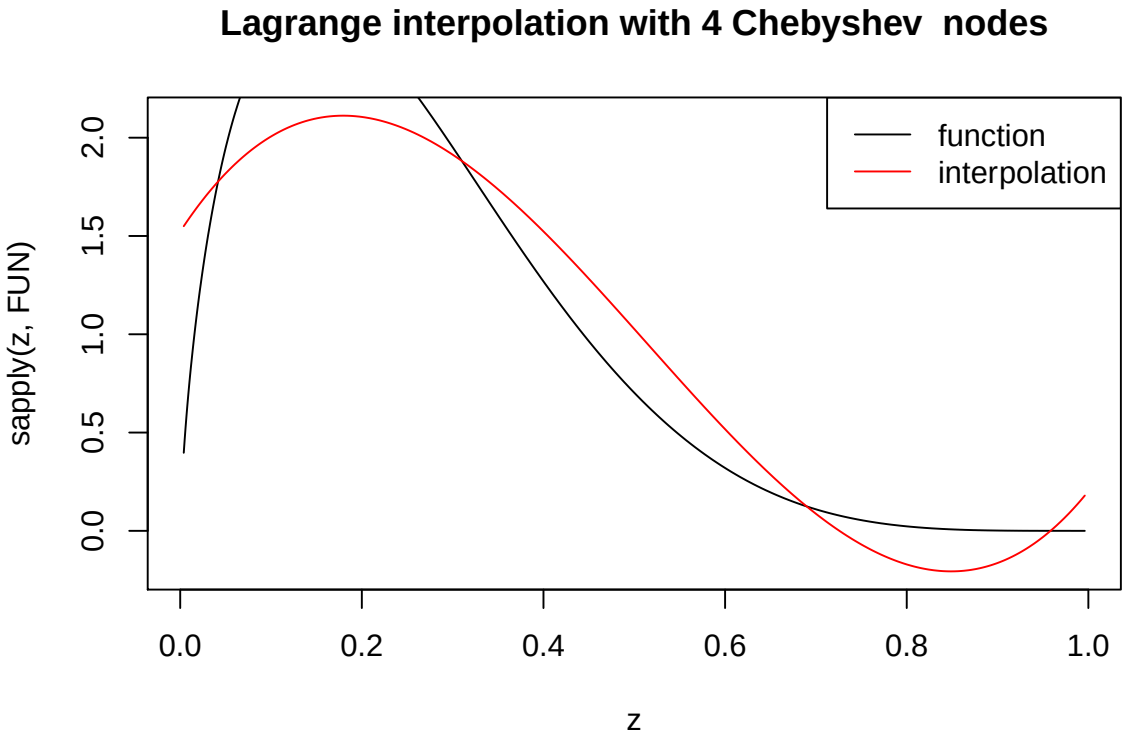
3. Reprenez les étapes de la question 1, en utilisant cette fois les noeuds de Tchebychev. commentez.

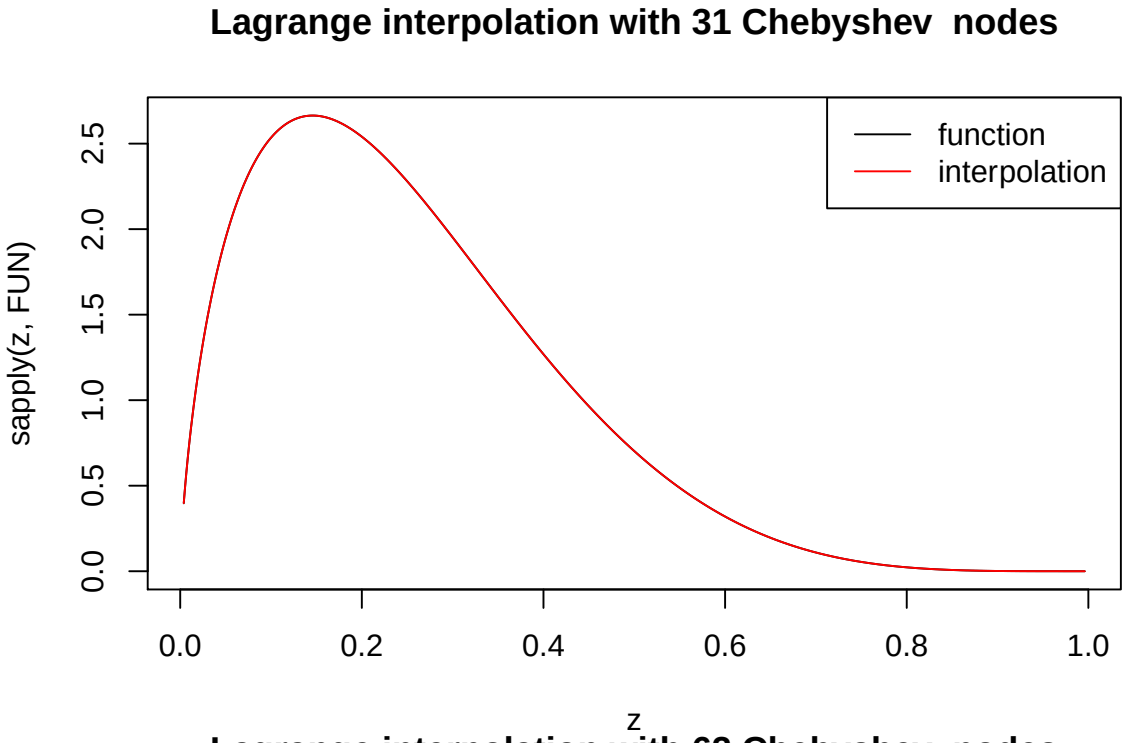
Reponse

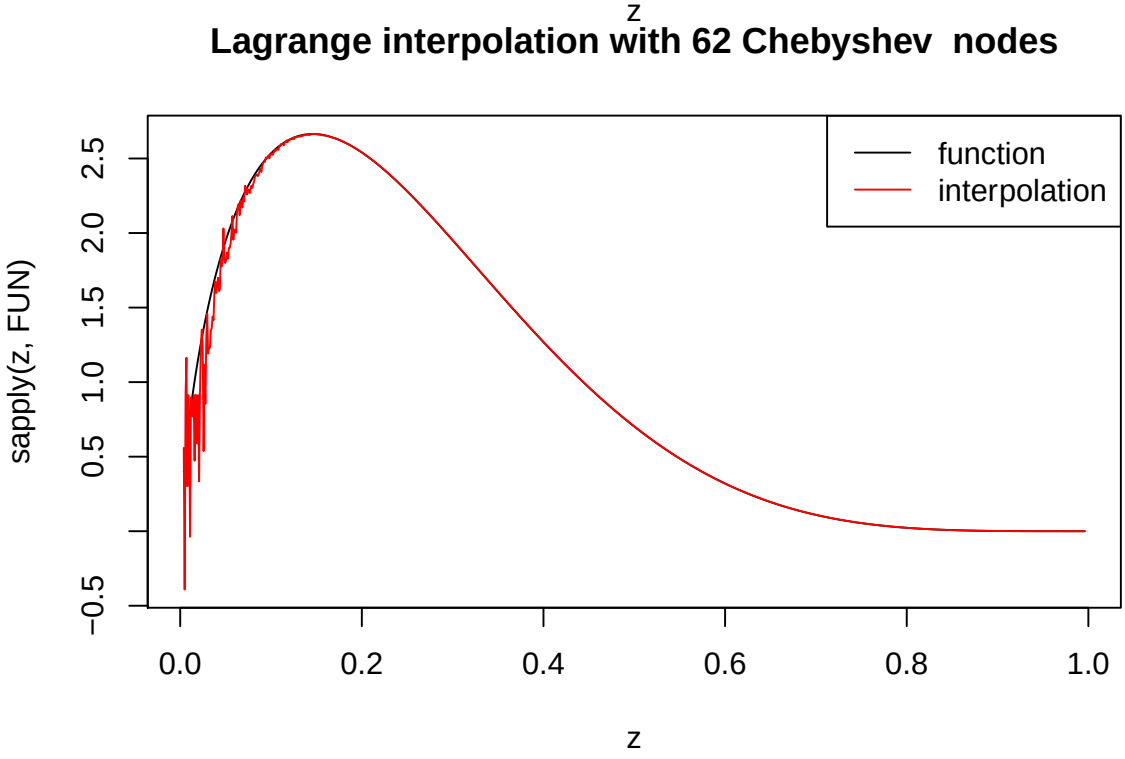
a) Le phénomène de Runge commence à apparaître a partir de degré 60


Lagrange interpolation with 62 Chebyshev nodes
function
interpolation
sapply(z, FUN)
−0.5
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0
z

b)


Lagrange interpolation with 4 Chebyshev nodes
function
interpolation
sapply(z, FUN)
0.0
0.5
1.0
1.5
2.0
0.0
0.2
0.4
0.6
0.8
1.0
z

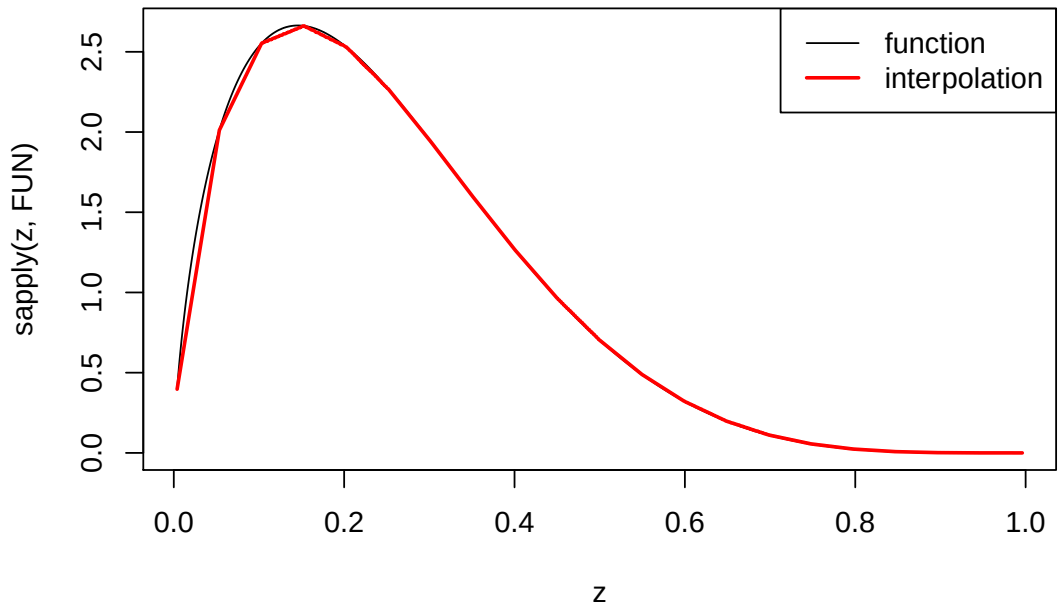


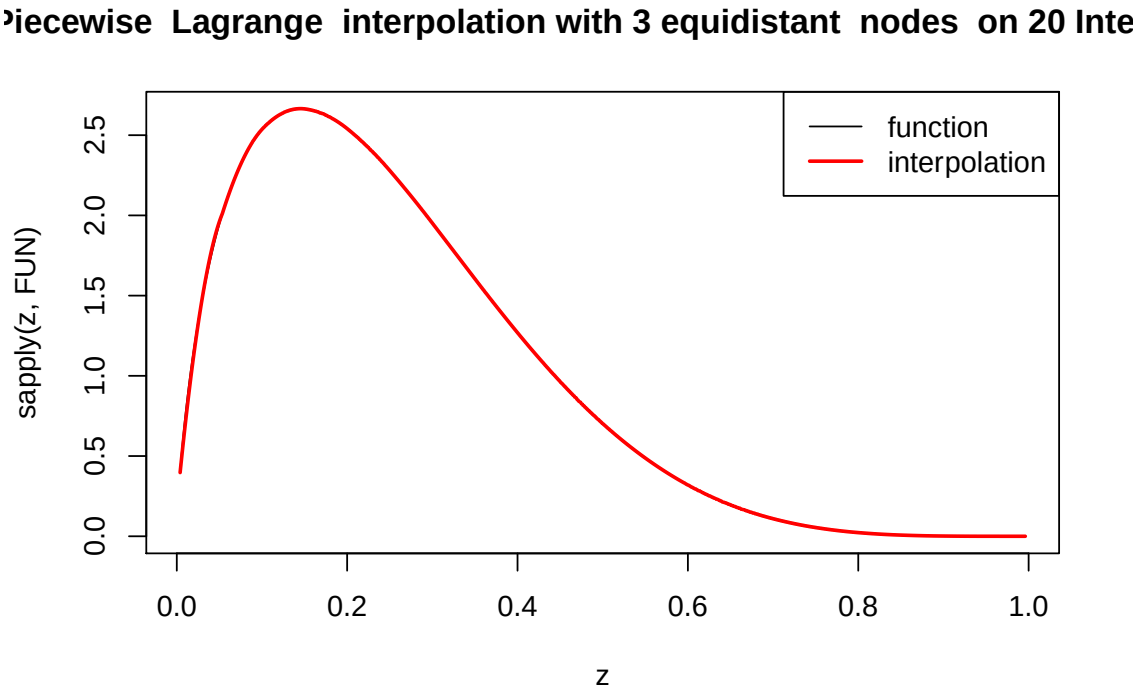


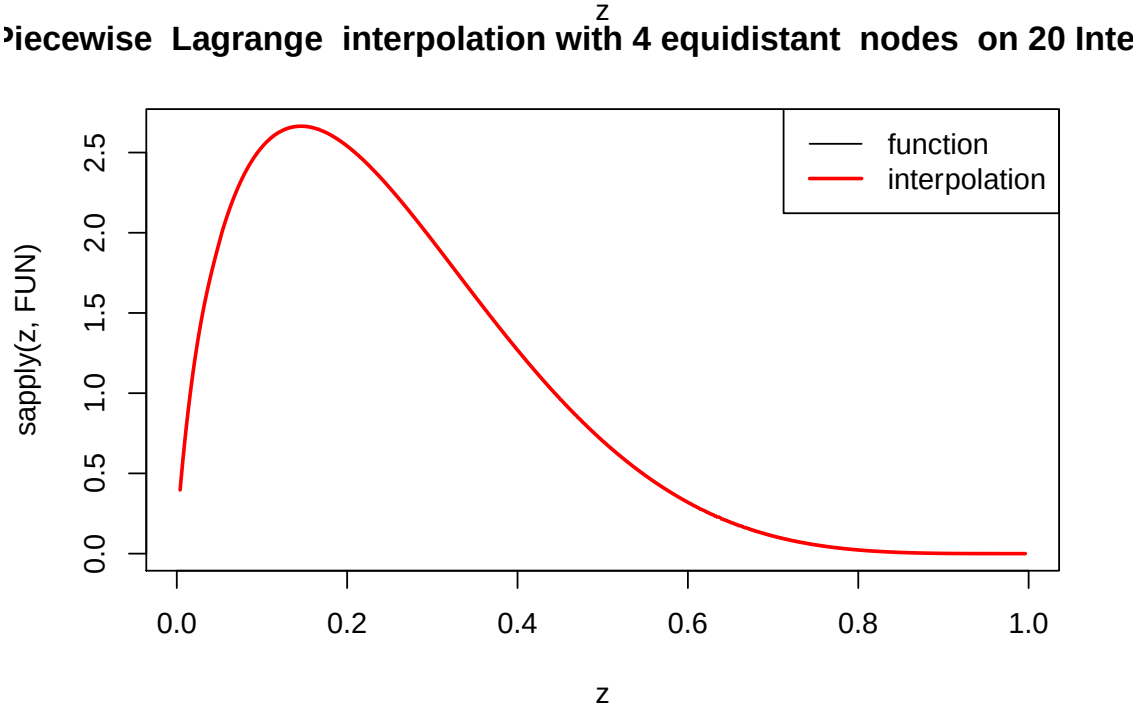
3. On considère maintenant une interpolation par morceaux, de degré $n = 1, 2$ ou $n = 3$, avec noeuds équi-distants sur chaque sous-intervalle, de pas $h = (b - a)/M$. Commencez par modifier la fonction hornerNewton codée en cours pour assurer son bon fonctionnement pour un nombre de noeuds d'interpolation inférieur à 2.
   Affichez le résultat de l'interpolation en même temps que la fonction de référence, pour $M = 20$ et $n = 1, 2, 3$.

```r
source('functions_ETC2018.R')
a = space_int
b = 1 - space_int
neval = 1000

M = 20
ns = c(1, 2, 3)

for (n in c(1, 2, 3)) {
  piecewiseInterpol(n=n,nInt=M,a=a,b=b,neval=1000, nodes = "equi", FUN=densite, Plot=TRUE)
}
```

Piecewise Lagrange interpolation with 2 equidistant nodes on 20 Inte







4. On veut comparer l'évolution de l'erreur pour l'interpolation par morceau de degré $n = 3$ et M sous-intervalles comme ci-dessus avec l'erreur de l'interpolation de Lagrange 'simple' de degré $n' = 3 * M + 1$, avec des noeuds d'interpolation equirépartis

   (a) Expliquez le choix de n 0 pour effectuer cette comparaison.

   On approche la norme infinie de l'erreur par le maximum des valeurs absolues des erreurs évaluées sur la grille d'interpolation (définie par le parametre **neval**). Pour chaque valeur de M on prendra **neval = floor((2^10 -1)/M) +1** comme argument de la fonction **piecewiseInterpol** et **neval = 2^10** comme argument de la fonction **interpolLagrange**.

   (b) justifiez le choix de **neval** dans les deux cas.

(c) Faites varier M de 1 à 20, et pour chaque valeur de M , calculez une approximation de la norme infinie de l'erreur comme expliqué ci-dessus. Tracez sur le même graphique l'évolution de cette approximation en fonction de M , pour les deux méthodes considérées. Commentez.

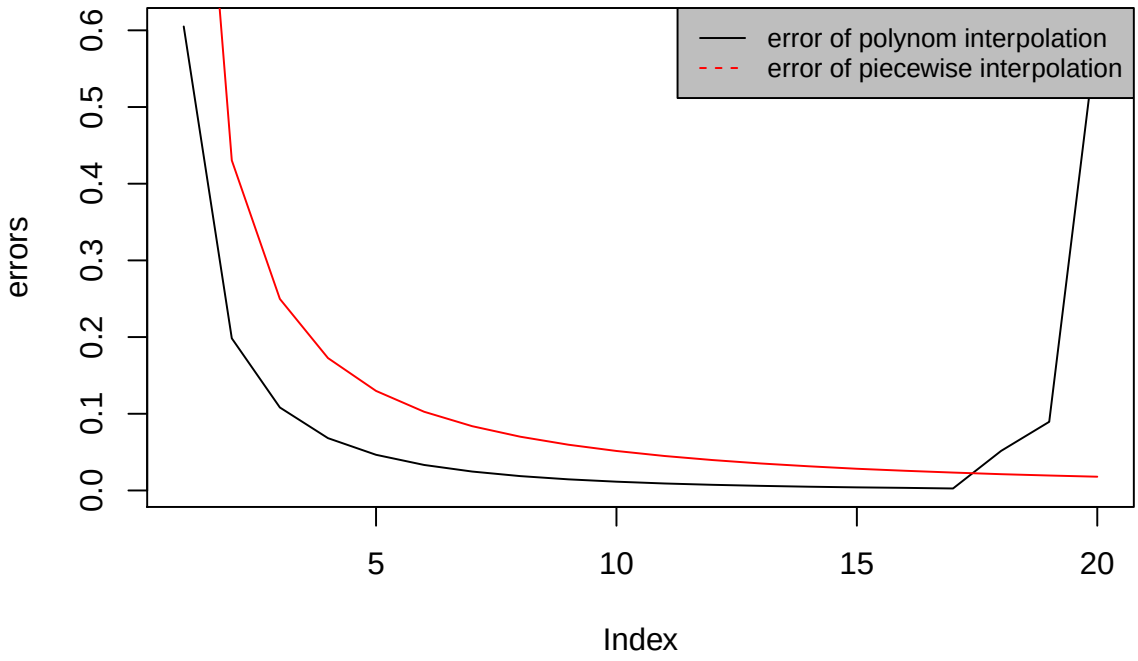
Réponse

a) On a choisi le n' pour avoir presque la même quantité des noeuds.

b) On a choisi neval pour que le nombre des noeuds de l'évaluation sera distribué de la même façon

c) Pour petites valeurs de M l'interpolation de Lagrange 'simple' donne mieux résultats, mais assez vite en augmentent M on va découvrir qu'erreur va exploser à cause du phénomène de Runge.

```
source('functions_ETC2018.R')
a = space_int
b = 1 - space_int

errors = c()
errors_piecewise = c()

for (M in seq(1, 20, 1)) {
  n = 3
  neval = floor( (2^10-1)/M) + 1

  res_piecewise = piecewiseInterpol(n=n,nInt=M,a=a,b=b,
                                    neval=neval, nodes = "equi",
                                    FUN=densite, Plot=FALSE)
  errors_piecewise = c(errors_piecewise, max(abs(res_piecewise[1,] - densite(res_piecewise[2,] ))))

  n_simple = 3*M + 1
  neval_simple = 2^10

  grid = seq(a, b, length.out=neval_simple)

  res = interpolLagrange(n=n_simple, a, b,
                         neval=neval_simple,nodes='equi',
                         FUN=densite, Plot=FALSE)
  errors = c(errors, max(abs(res - densite(grid))))
}

plot(errors, type='l')
lines(errors_piecewise, col='red')
legend("topright", legend=c("error of polynom interpolation",
                            "error of piecewise interpolation"),
       col=c('black', 'red'), lty=1:2, cex=0.8, bg="gray")
```

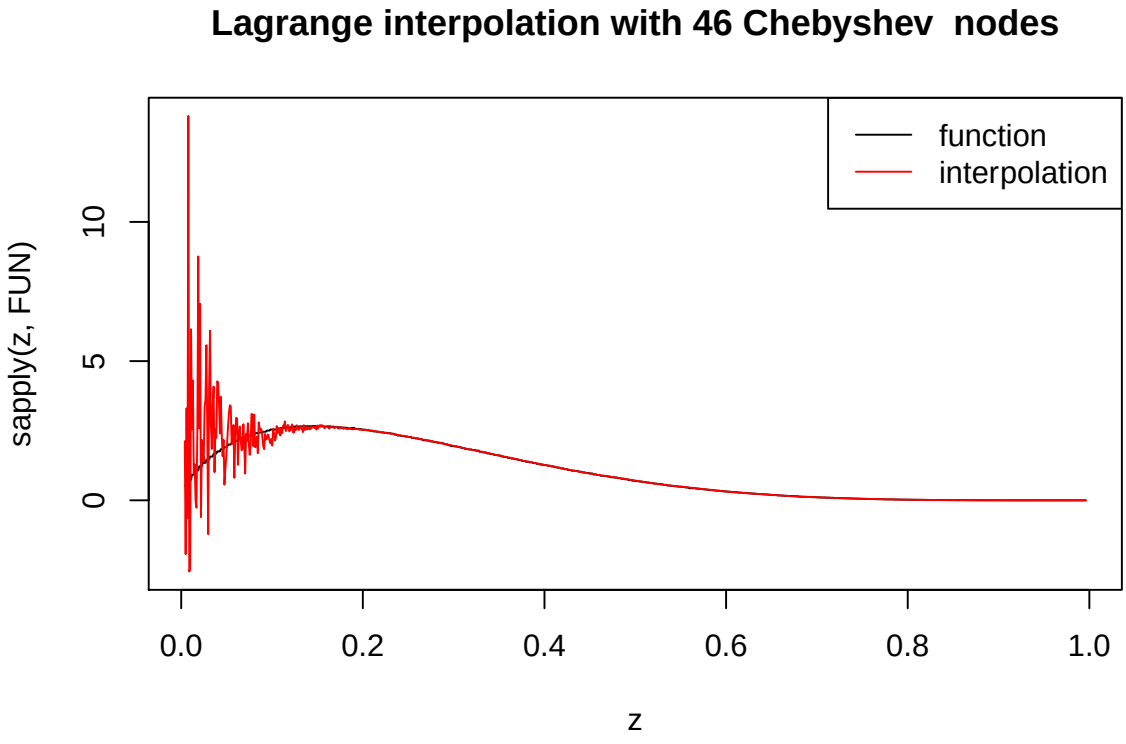


3.2 Interpolation à partir de l'histogramme

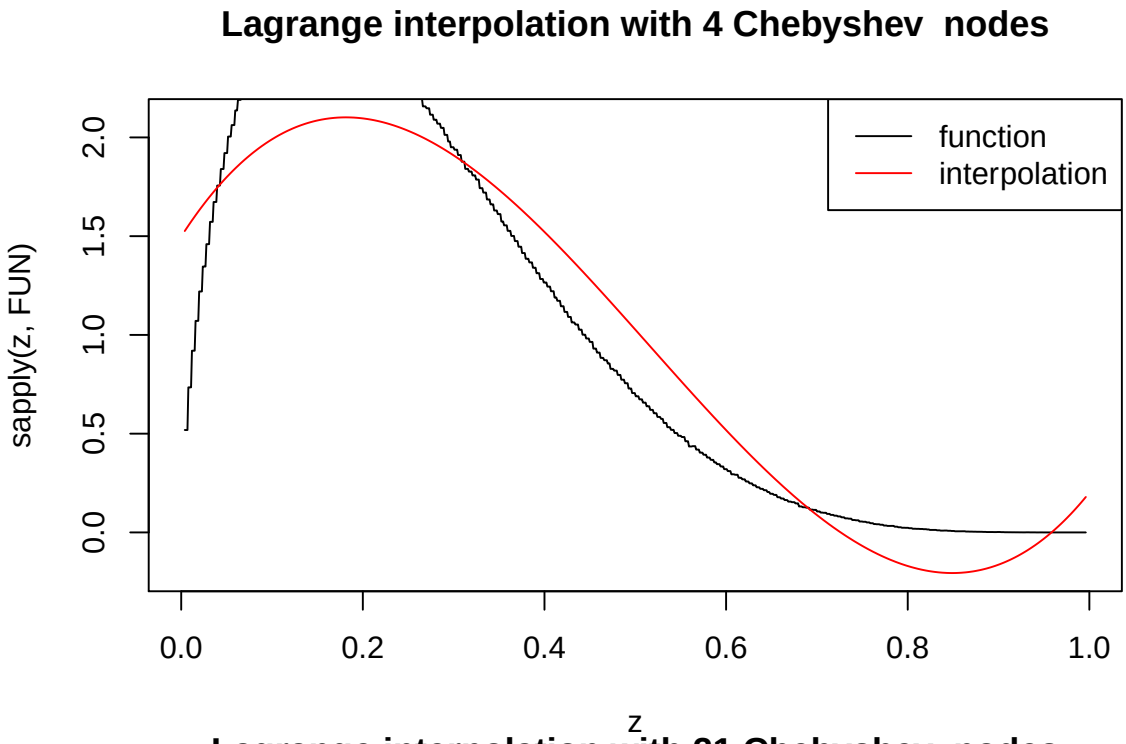
4. Reprendre les questions 1-2-3 en utilisant l'histogramme généré à la partie 2 (en pratique, vous utiliserez la fonction **hist_value**). Que remarquez-vous à propos des deux premières méthodes (questions 1 et 2) ? Quelle méthode d'interpolation vous parait la plus judicieuse ?

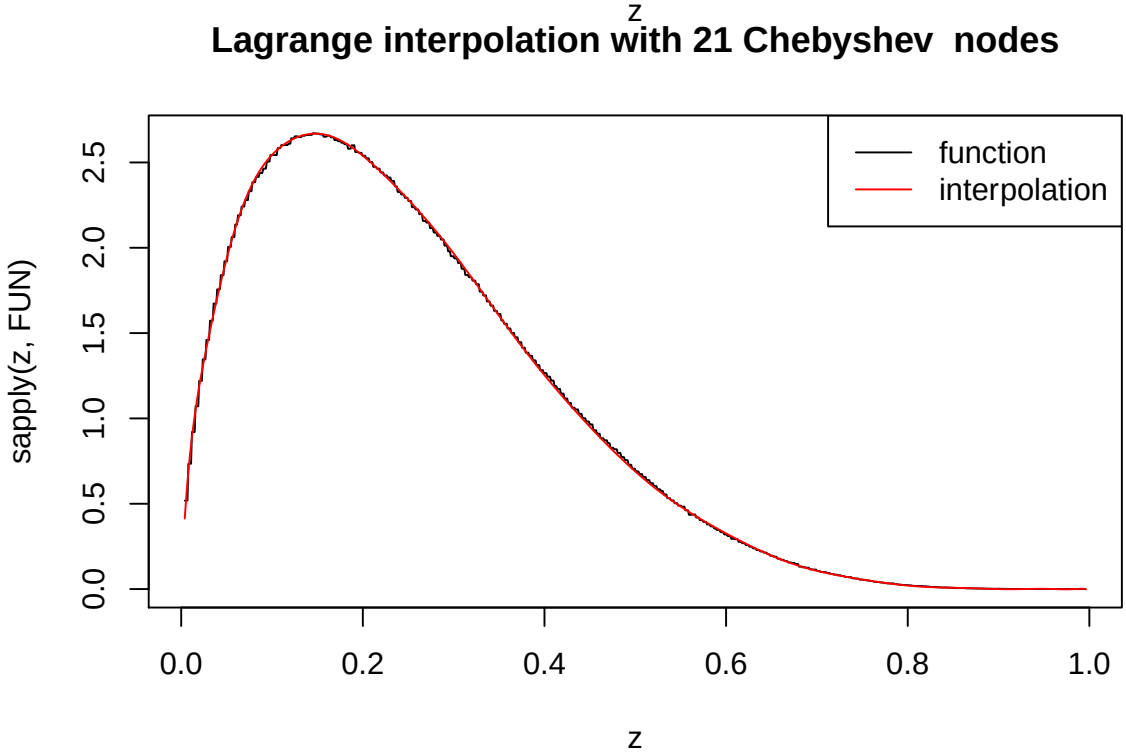
Réponse

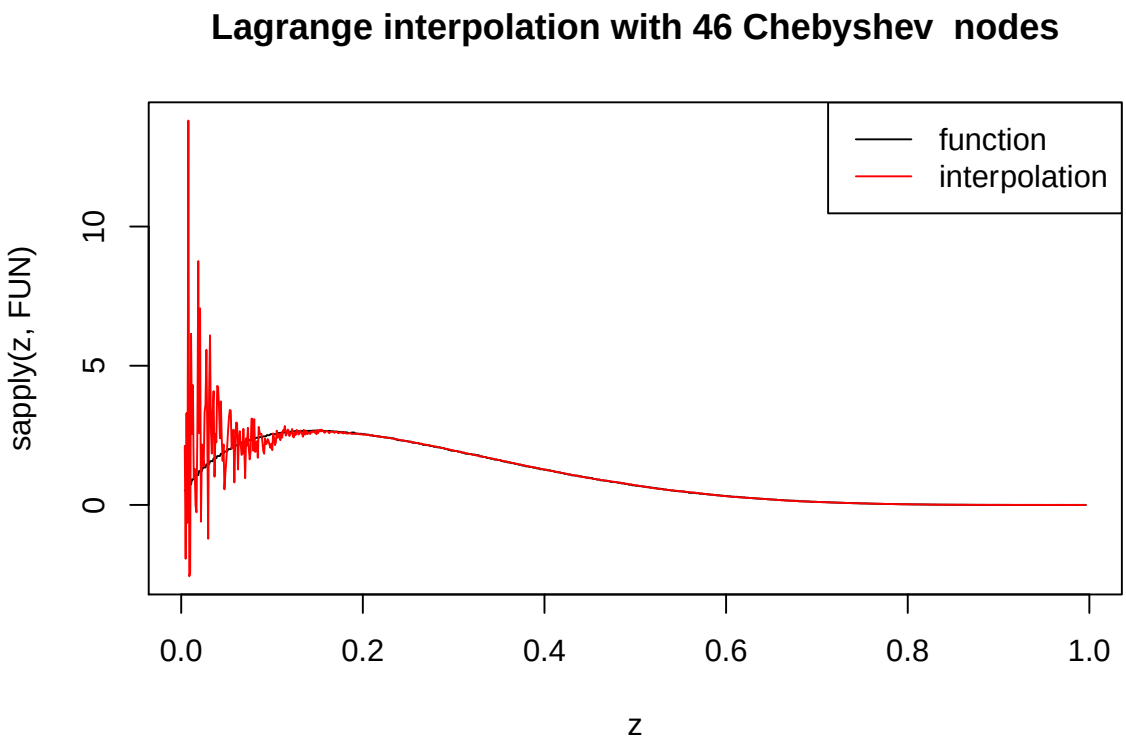
3.1.1 a) Le phénomène de Runge commence à apparaître a partir de degré 45



b)

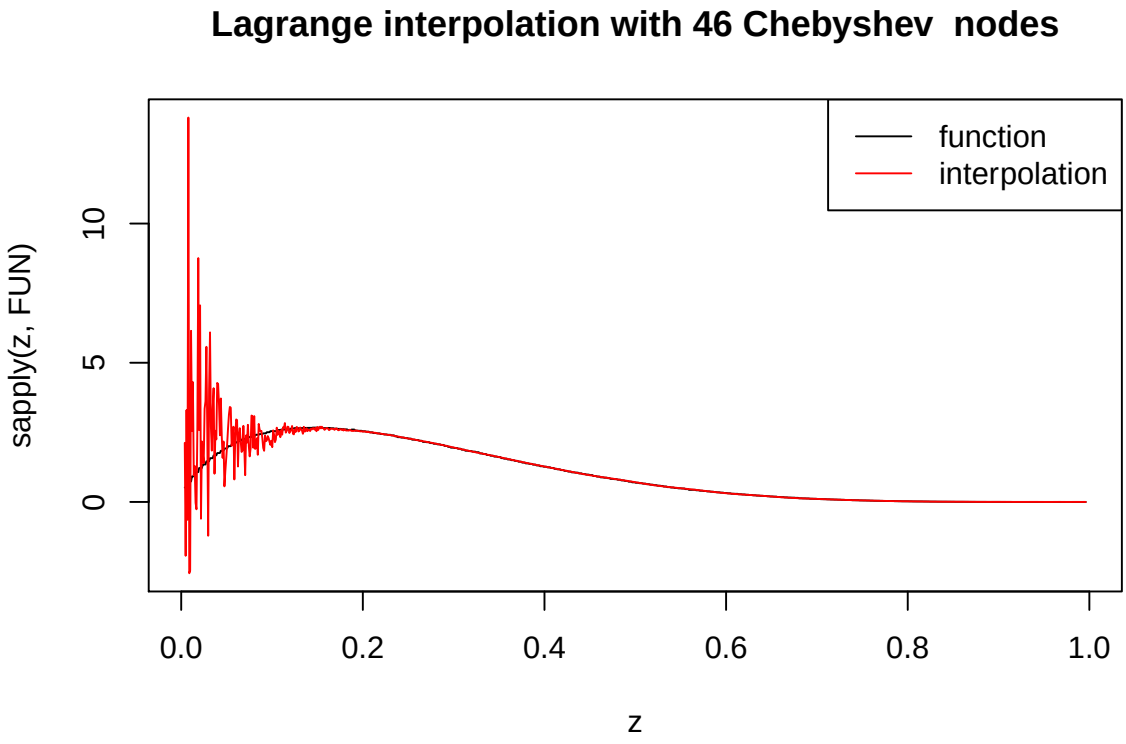

Lagrange interpolation with 4 Chebyshev nodes
function
interpolation
sapply(z, FUN)
0.0
0.5
1.0
1.5
2.0
0.0
0.2
0.4
0.6
0.8
1.0
z


Lagrange interpolation with 21 Chebyshev nodes
function
interpolation
sapply(z, FUN)
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0
z

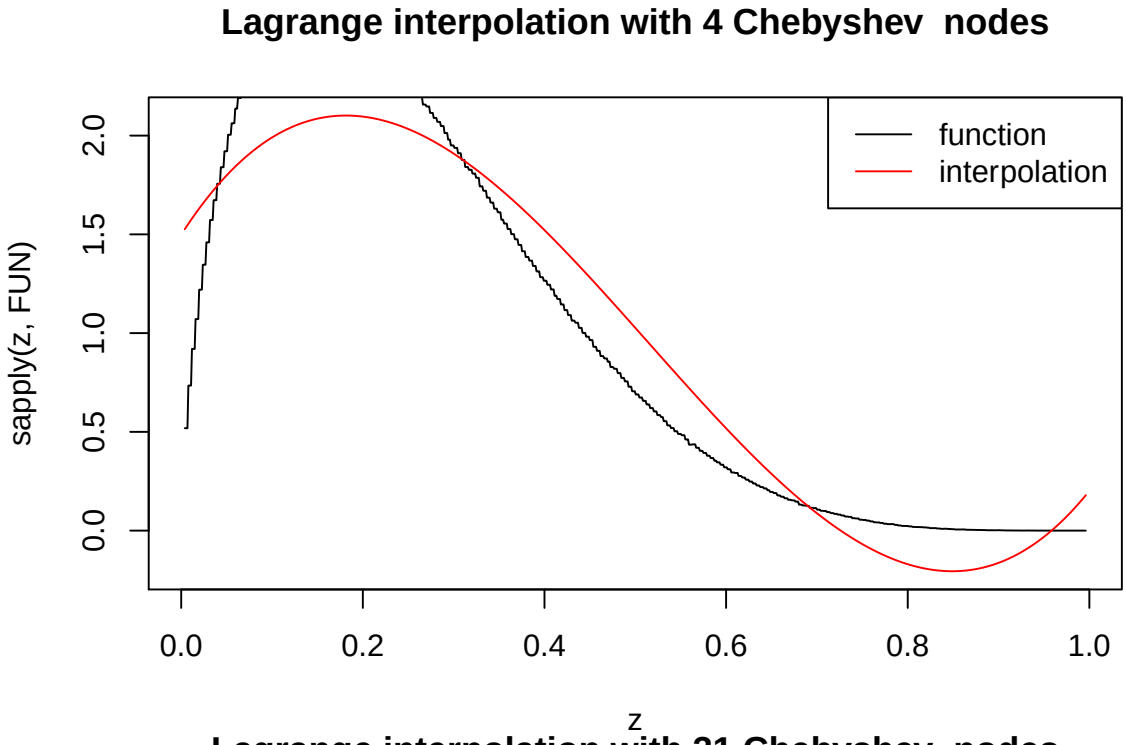


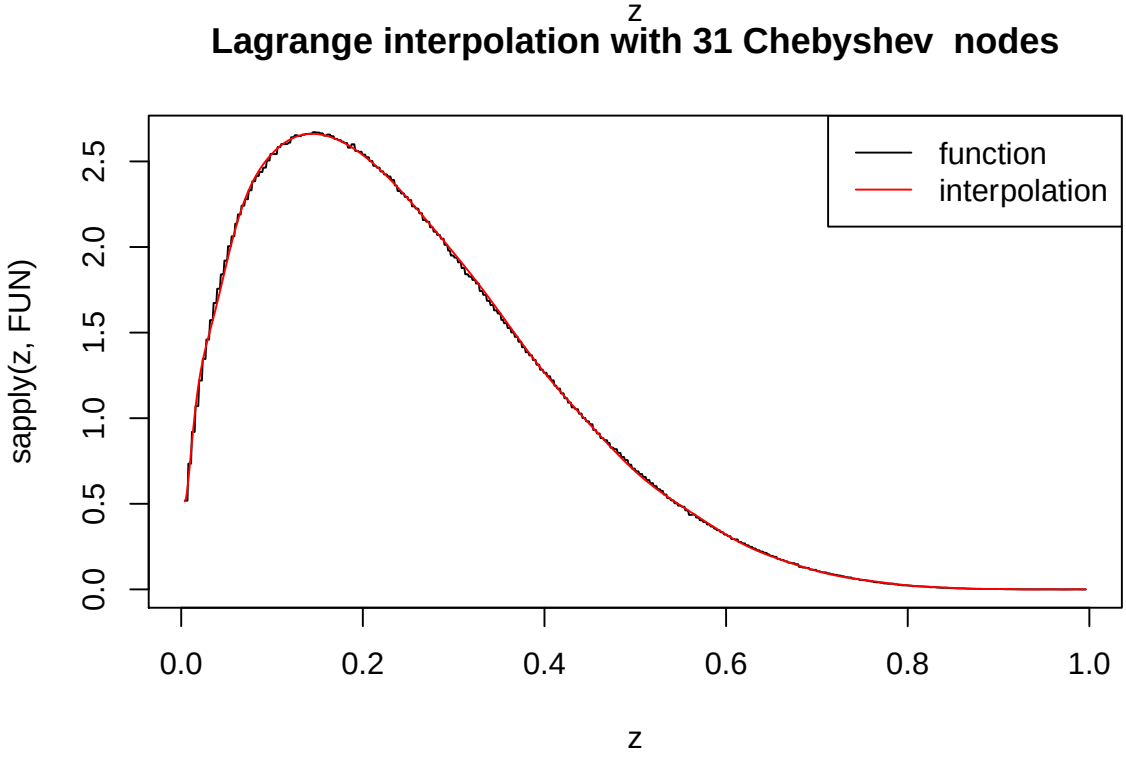
3.1.2

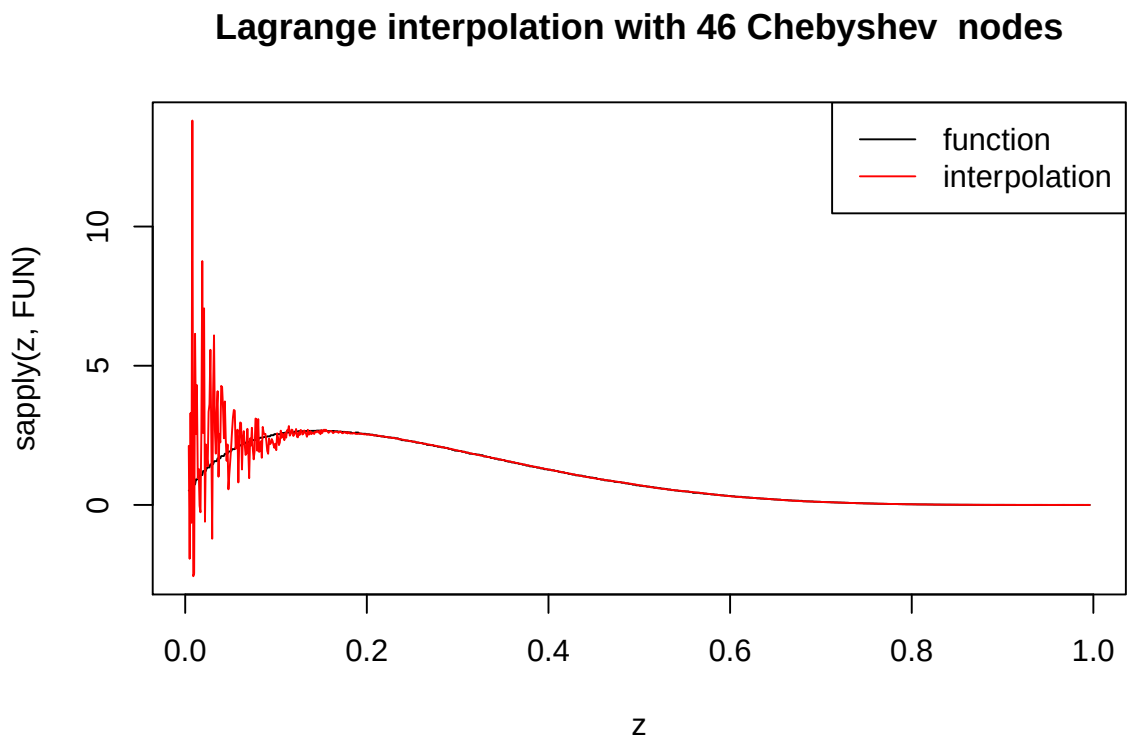
a) Le phénomène de Runge commence à apparaître a partir de degré 45



b)

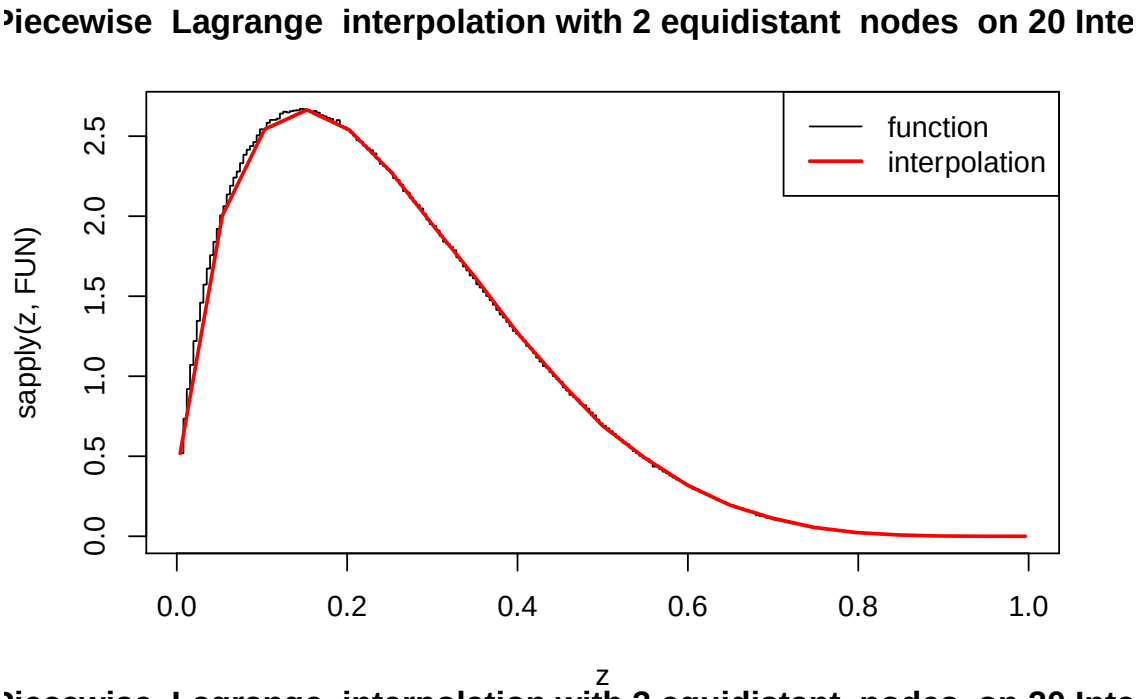

Lagrange interpolation with 4 Chebyshev nodes
function
interpolation
sapply(z, FUN)
0.0
0.5
1.0
1.5
2.0
0.0
0.2
0.4
0.6
0.8
1.0
z

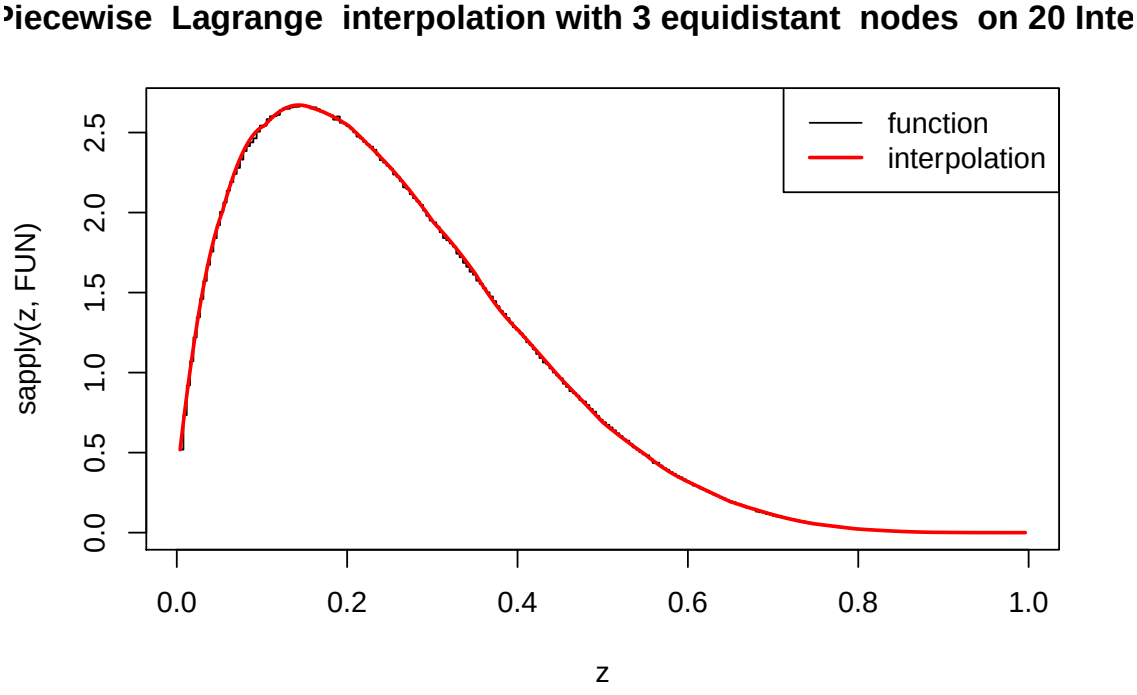

Lagrange interpolation with 31 Chebyshev nodes
function
interpolation
sapply(z, FUN)
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0
z

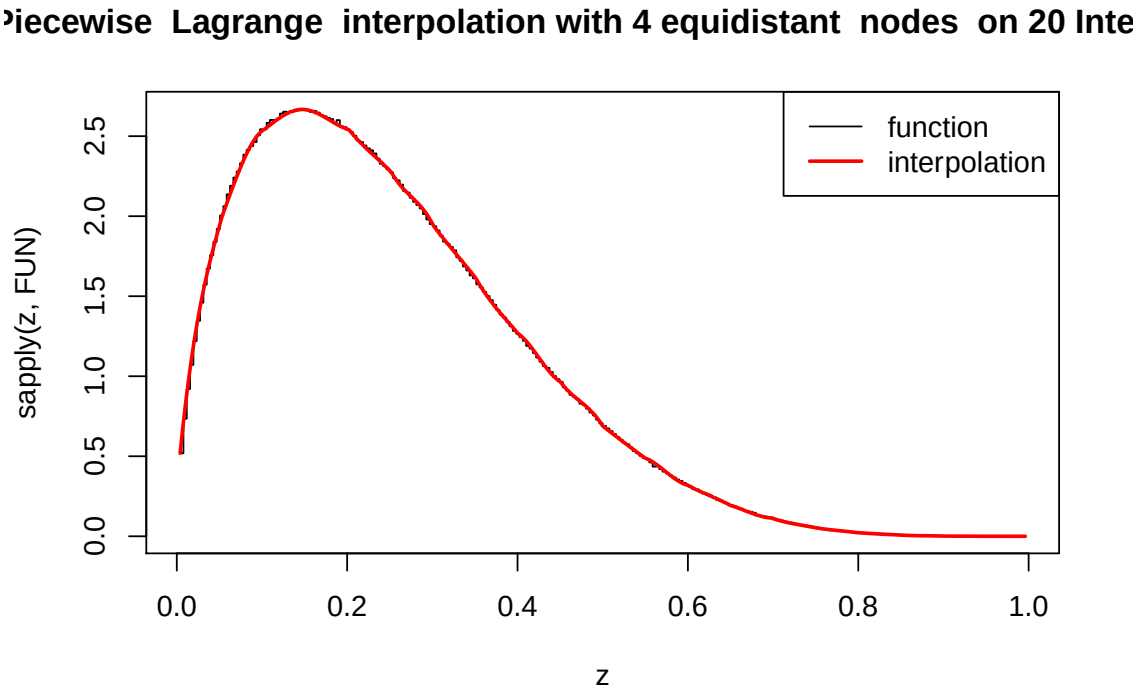


3.1.3

```
source('functions_ETC2018.R')
a = space_int
b = 1 - space_int
neval = 1000

M = 20

for (n in c(1, 2, 3)) {
  piecewiseInterpol(n=n,nInt=M,a=a,b=b,neval=1000, nodes = "equi", FUN=f_hist_value, Plot=TRUE)
}
```


Piecewise Lagrange interpolation with 2 equidistant nodes on 20 Inte
function
interpolation
sapply(z, FUN)
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0
z


Piecewise Lagrange interpolation with 3 equidistant nodes on 20 Inte
function
interpolation
sapply(z, FUN)
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0
z



Dans question 1 et 2 on peut remarquer que le nombre des noeuds à partir lequel on a le phénomène de Runge est moins que pour la fonction **densite**.

La méthode d'interpolation par morceaux me parait la plus judicieuse, parce qu'il n'y a pas de phénomène de Runge avec le nombre des noeuds plus grands que dans les méthodes précédentes.

4 Méthodes de quadrature

4.1 Estimation de $p_{0.1}$ à partir de la densité $\phi_{1.7,5.1}$

On se place dorénavant dans le cas où la densité est connue, mais où son intégrale ne l'est pas.

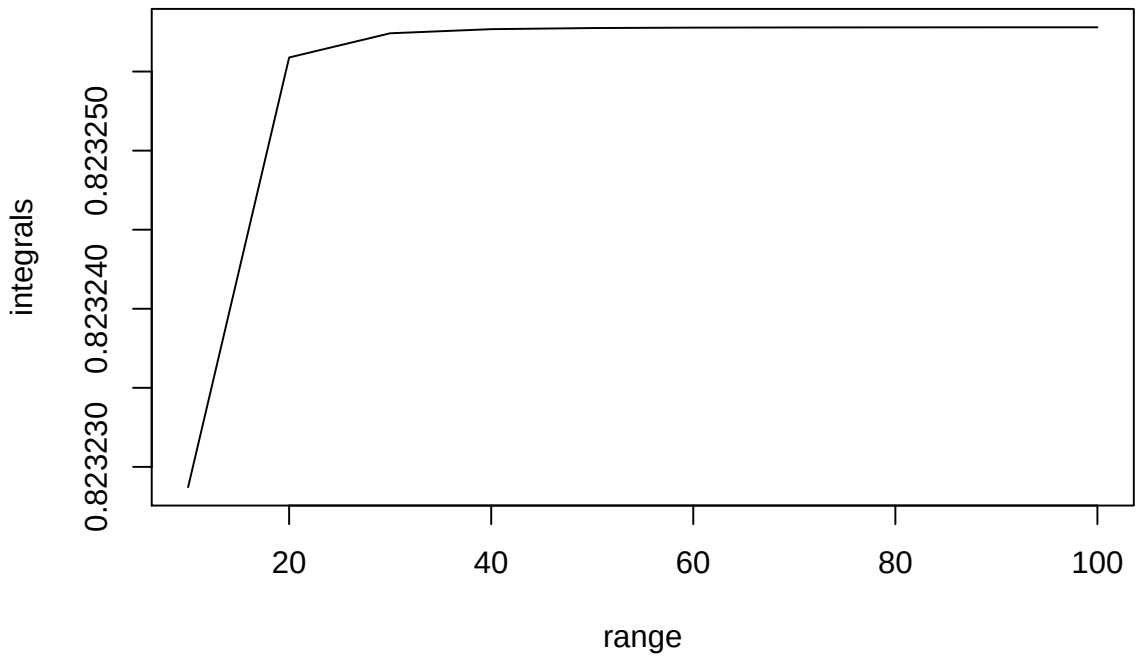
Pour estimer la valeur de $p_{0.1}$ définie par l'équation (3), on utilise la méthode de quadrature composite de Cavalieri-Simpson, où la fonction à intégrer est la densité $\phi_{1.7,5.1}$, connue, les bornes d'intégration sont $a = 0.1, b = 0.9$. On appelle $\hat{I}_M^{simp}$ le résultat obtenu par une méthode de simpson composite avec M intervalles de même taille.

1. Affichez le graphe de $\hat{I}_M^{simp}$ en fonction de M , pour M variant sur une plage appropriée. En déduire une première estimation de $p_0.1$. Au vu des résultats numériques, donnez un ordre de grandeur la précision de votre estimation.

Réponse

```
source('functions_ETC2018.R')
a = 0.1
b = 0.9

integrals = c()

range = seq(10, 100, 10)
```

```r
for (M in range) {
  I = simpsonInt(FUN=densite, a=a, b=b, M=M)
  integrals = c(integrals, I)
}

plot(range, integrals, type='l')
```



Première estimation de $p_{0.1} = 0.8232578$

donnez un ordre de grandeur la précision de votre estimation.

2. On s'intéresse à l'erreur commise en fonction du nombre M de sous intervalles d'intégration.
   (a) Quel est l'ordre de grandeur théorique de l'erreur en fonction de $h = \frac{b-a}{M}$?
   (b) La fonction **pbeta** de **R** donne une très bonne approximation de la fonction de répartition de la loi Beta. Calculez une valeur 'de référence' $p_{0.1,table}$ en utilisant la fonction **pbeta**. Tracez le graphe du logarithme de la valeur absolue de l'erreur, $log\|p_{0.1,table} - I_M^{simp}\|$ en fonction de $log(M)$. Que pouvez-vous en déduire et cela est-il en accord avec le cours ?
   (c) Évaluez a posteriori la valeur absolue de l'erreur, (sans utiliser la fonction **pbeta**) pour M variant entre 14 et 80.
      i Tracez le graphe "log-log" de l'erreur absolue contre le nombre de pas M .
      ii Estimez la pente dans une portion linéaire
      iii Comparez vos résultats à ceux de la question (b)

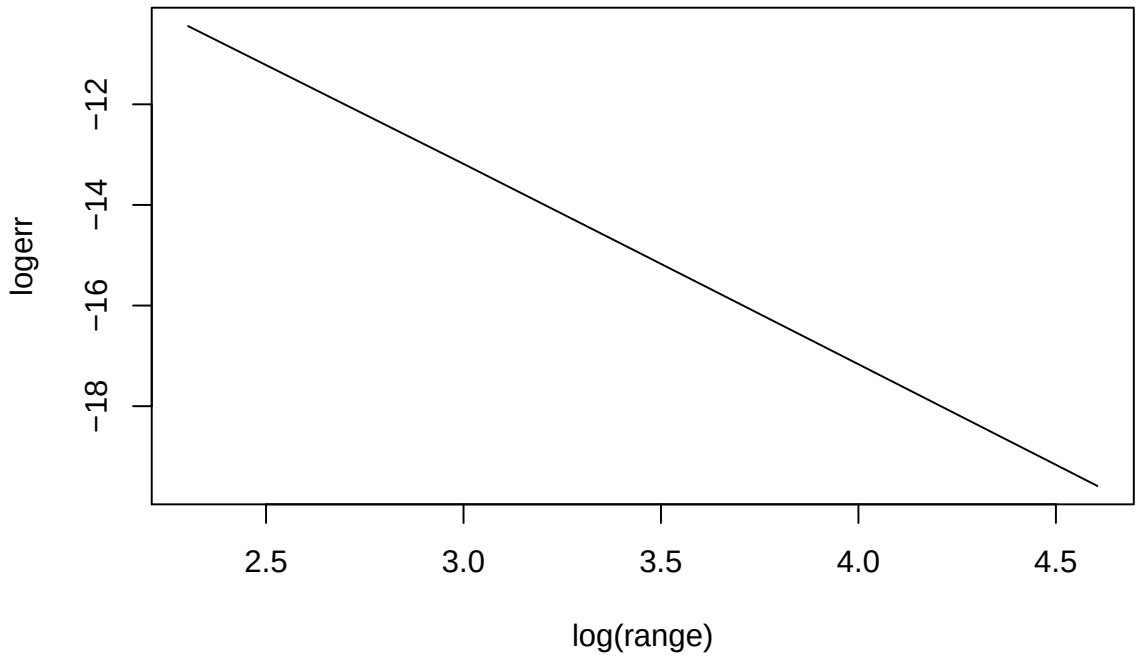
Réponse

a) L'ordre de grandeur théorique de l'erreur en fonction de h : $\mathcal{O}(h^4)$

b)

```r
p_table = pbeta(0.9, 1.7, 5.1) - pbeta(0.1, 1.7, 5.1)

integral_errors = abs(integrals - p_table)
```

```r
logerr = log(integral_errors)

plot(log(range), logerr, type='l')
```

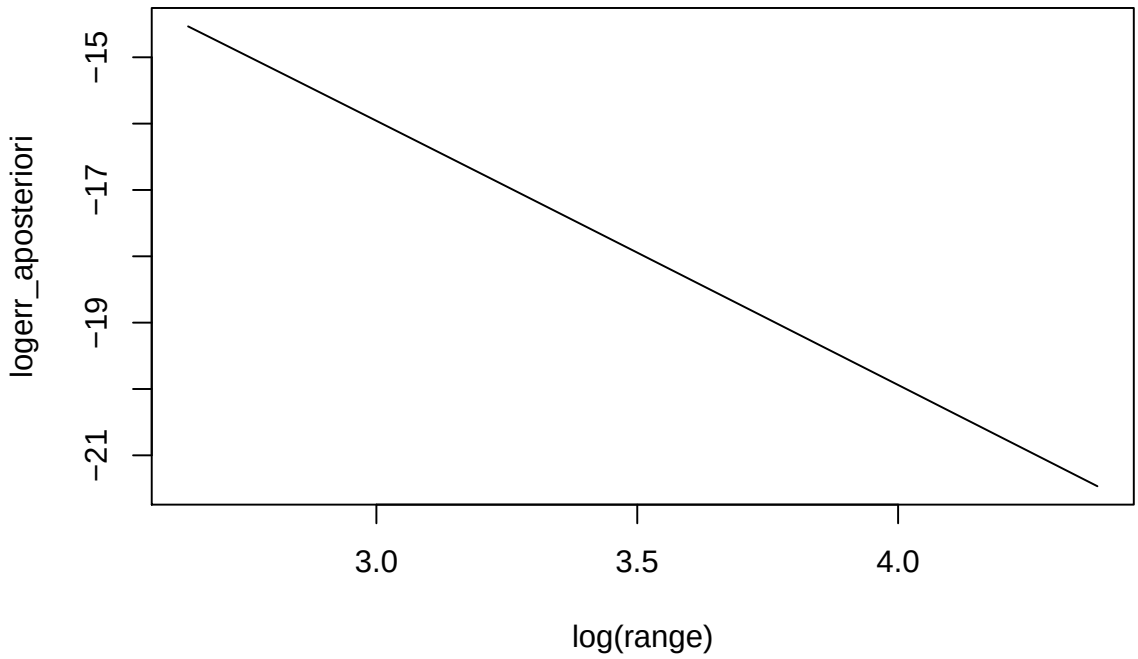



```r
N = length(logerr)
pente = (logerr[N] - logerr[1])/(log(range[N]) - log(range[1]))
sprintf("Pente de courbe est: %f", pente)
```

```
## [1] "Pente de courbe est: -3.969410"
```

c)

```r
source('functions_ETC2018.R')

range=seq(14,80,1)

integral_errs_aposteriori = c()

for (M in range) {
  res = evalErrSimpson(FUN=densite, a=0.1, b=0.9, M)
  integral_errs_aposteriori = c(integral_errs_aposteriori, res[1])
}

logerr_aposteriori =  log(abs(integral_errs_aposteriori))

plot(log(range),logerr_aposteriori, type='l')
```




```
N = length(logerr_aposteriori)
pente = (logerr_aposteriori[N] - logerr_aposteriori[1])/(log(range[N]) - log(range[1]))
sprintf("Pente de courbe est: %f", pente)
```

```
## [1] "Pente de courbe est: -3.976888"
```

Le pente des courbes sont pres de -4, parce que on a utilisé le méthode d'ordre 3.

4. On voudrait une méthode automatique pour choisir M en fonction de l'estimation de l'erreur a posteriori, pour avoir une estimation $\hat{I}$ à 10^{-6} près de $p_{0.1}$. Écrire un algorithme utilisant la fonction evalErrSimpson.

```
source('functions_ETC2018.R')

res = chooseM(FUN=densite, a=0.1, b=0.9, tolerance=10^(-8))

sprintf("Estimation a 10^(-8) pres : %f", res[2, length(res[2,]) - 1])
```

```
## [1] "Estimation a 10^(-8) pres : 0.823258"
```

Donnez maintenant une estimation de $p_{0.1}$ à 10^{-8} près.

$p_{0.1} = 0.823258$

4.2 Estimation de $p_{0.1}$ à partir $\hat{\phi}_{1.7,5.1}$

On se demande dans cette partie comment les erreurs en entrée (sur les données, i.e. sur l'intégrande $\phi_{1.7,5.1}$) impactent la sortie (la valeure de l'intégrale approchée)

1. Ecrire une fonction **densite_bruitee** prenant en argument un vecteur x (dont les éléments sont dans l'intervalle $[0, 1]$) et renvoyant un vecteur y de même taille dont chaque élément est la valeur de l'histogramme $\hat{\phi}_{1.7,5.1}$ évalué en l'élément de x associé. On utilisera la fonction **hist_value** fournie.

On considère

— l'approximation $\hat{p}^{bruit}_{0.1,M}$ de $p_{0.1}$ obtenue par la la méthode de composite de Simpson appliquée à **densite_bruitee**, avec M sous-intervalles.

— l'approximation $\hat{p}_{0.1,M}$ obtenue par la même méthode, appliquée à la vraie densité **densite**.

On s'intéresse à la différence absolue

$$\Delta_M = |\hat{p}^{bruit}_{0.1,M} - \hat{p}_{0.1,M}|.$$

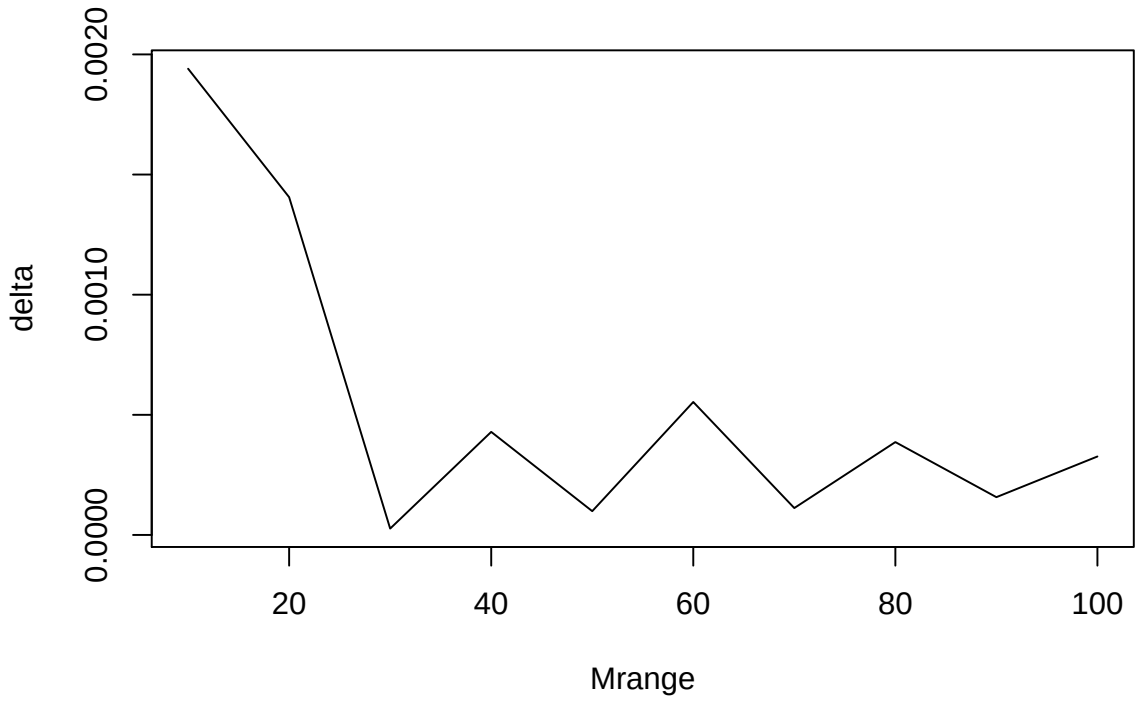
2. Que vous dit la théorie sur l'allure de la courbe de Δ_M en fonction de M ?

Réponse

L'erreur va diminuer au certain niveau et après à cause de bruit l'approximation va osciller.

3. Affichez cette courbe et commentez le résultat vis-à-vis de la question précédente.

```
source('functions_ETC2018.R')

Mrange = seq(10, 100, 10)

delta = c()

for (M in Mrange) {
  res = simpsonInt(FUN=densite, a=0.1, b=0.9, M)
  res_bruitee = simpsonInt(FUN=densite_bruitee, a=0.1, b=0.9, M)

  delta = c(delta, abs(res_bruitee - res))
}

plot(Mrange, delta, type='l')
```



5 Extrapolation de Richardson et méthode de Romberg

Dans cette partie, on cherche à évaluer à nouveau $p_{0.1}$, en utilisant la "vraie" densité $\phi_{1.7,5.1}$, par la méthode des trapèzes, avec une étape d'extrapolation de Romberg.

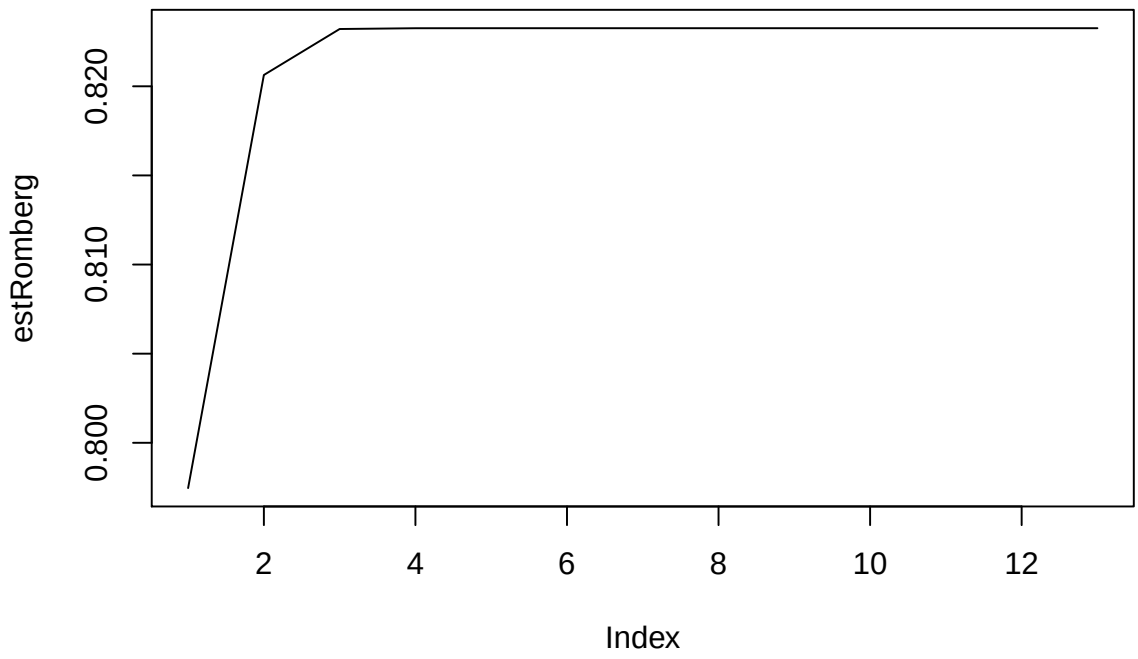
Estimation de $p_{0.1}$

Estimez la valeur de $p_{0.1}$ avec la méthode de Romberg : Passez en argument de la fonction romberg un nombre initial d'intervalles $M = 3$ et un nombre de raffinements successifs $n = 12$.

1. Nommez **estRomberg** le vecteur renvoyé par la fonction **Romberg** avec les paramètres indiqués, et affichez-le.

Reponse

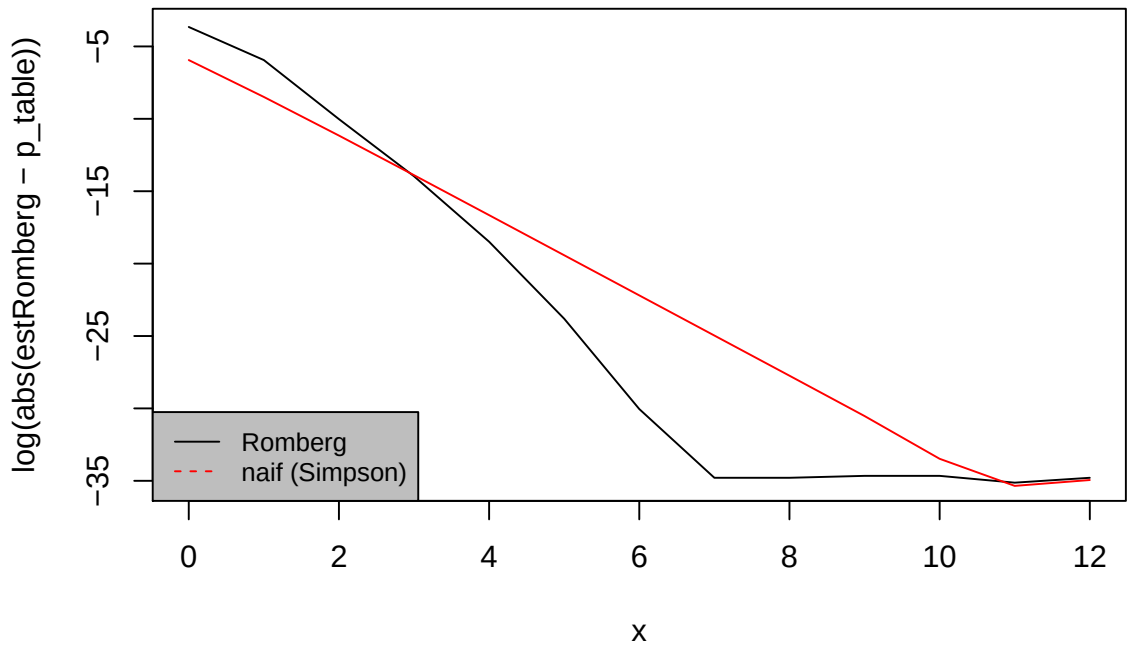
```
source('functions_ETC2018.R')
M = 3
n = 12

estRomberg = romberg(FUN = densite, n=12, a=0.1, b=0.9, M=3)

plot(estRomberg, type='l')
```



2. On prendra comme "vraie" valeur de l'intégrale la quantité $p_{0.1,table}$ calculée à la Section 4. Tracez sur le même graphique : le vecteur $0 : n$ en abscisses ; et en ordonnées : le logarithme de l'erreur $log|estRomberg - p_{0.1,table}|$ et le logarithme de l'erreur des estimateurs "naïfs" correspondants, c'est à dire les $log|\hat{I}_m - p_{0.1,table}|$, pour $m = M, 2M, ..., 2^n M$. Que constatez-vous ? Comment interprétez-vous le palier ? l'autre portion de la courbe ?

```
source('functions_ETC2018.R')
M = 3
n = 12
```

```r
x = 0:n

estRomberg = romberg(FUN = densite, n=12, a=0.1, b=0.9, M=3)

naif = c()
for (p in x) {
  m = M*2^p
  res = simpsonInt(FUN=densite, a=0.1, b=0.9, m)
  naif = c(naif, res)
}



plot(x, log(abs(estRomberg-p_table)), type='l')
lines(x, log(abs(naif - p_table)), col='red')

legend("bottomleft", legend=c("Romberg",
                              "naif (Simpson)"),
       col=c("black", 'red'), lty=1:2, cex=0.8, bg="gray")
```



Méthode de Romberg converge plus vite. L'erreur sur palier est comparable avec .Machine$double.eps.