TP5 Correction

Clement LAROCHE

5 avril 2019

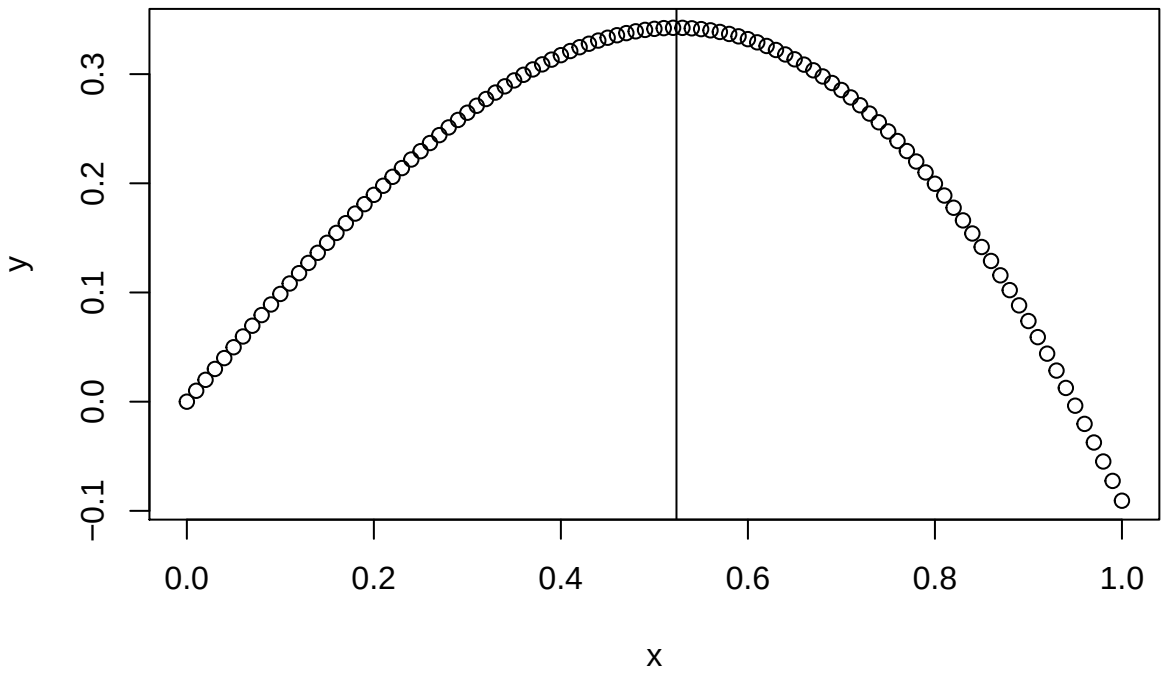
Un problème de résolution d'équation

On calcule la dérivée de f_1.

$$f_1'(x) = 2\cos(2x) - 1$$

Sur l'intervalle $[0, 1]$, on voit que la dérivée sera positive entre $x = 0$ et $x = \frac{\pi}{6}$. Elle devient ensuite négative jusqu'à $x = 1$. On en déduit donc les variations de f_1.
Une solution triviale de cette équation est $x_0 = 0$. On peut la retrouver de manière théorique, c'est bien le x_0 demandé dans l'énoncé. Comme f_1 est strictement décroissante sur $[\frac{\pi}{6}, 1]$ et que $f_1(\frac{\pi}{6}) = \frac{\sqrt{3}}{2} - \frac{\pi}{6} \geq 0$ et $f_1(1) = \sin(2) - 1 \leq 0$, cela veut dire qu'il existe une solution moins triviale x_1 dans cet intervalle.

```
# axe des abscisses
x <-seq(from = 0,to = 1,by = 0.01)
# application de la fonction à mon axe des abscisses
y <- sin(2*x)-x
# tracé de la fonction et de l'abscisse où la dérivée s'annule
{
  plot(x,y)
  abline(v = pi/6)
}
```



```r
# on cherche quelles coordonnées du vecteur des abscisses sont telles que f(vecteur à cette coordonnée)
which(abs(y[-1]) < 10^-1)
```

```
##  [1]   1   2   3   4   5   6   7   8   9  10  89  90  91  92  93  94  95
## [18]  96  97  98  99 100
```

```r
# on affiche les valeurs du vecteur des absicces concernées
x[which(abs(y[-1]) < 10^-1)]
```

```
##  [1] 0.00 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.88 0.89 0.90 0.91
## [15] 0.92 0.93 0.94 0.95 0.96 0.97 0.98 0.99
```

A 0.1 près on peut donc dire que 0.9 est une coordonnée proche de x_1.

Méthode d'approximation par dichotomie

```r
a = 0.9
b = 1
n <- 1
crit = TRUE
pt <- (a+b)/2
f1 <- function(x)
{
  sin(2*x)-x
}
while (crit == TRUE) {
  if(f1(a)*f1(pt) < 0)
  {b <- pt}
  else if(f1(b)*f1(pt) < 0)
  {a <- pt}
  pt <- (a+b)/2
  n <- n+1
  if(abs(b-a)<10^-10)
  {crit <- FALSE}
}
n
```

```
## [1] 31
```

```r
10*log(10)/log(2)
```

```
## [1] 33.21928
```

Notez que le nombre d'itérations nécessaires pour atteindre une certaine précision dépend de la largeur de l'intervalle initial choisi.

```r
a = 0.5
b = 1.5
n <- 1
crit = TRUE
pt <- (a+b)/2
f1 <- function(x)
{
  sin(2*x)-x
}
while (crit == TRUE) {
```

```
  if(f1(a)*f1(pt) < 0)
  {b <- pt}
  else if(f1(b)*f1(pt) < 0)
  {a <- pt}
  pt <- (a+b)/2
  n <- n+1
  if(abs(b-a)<10^-10)
  {crit <- FALSE}
}
n
```

```
## [1] 35
```

Méthode de la sécante

L'équation de cette droite s'obtient assez simplement si on effectue le tracé de cette fameuse sécante. Cette équation s'écrit (dans notre cas particulier pour f_1) :

$$y = \frac{f(u_1) - f(u_0)}{u_1 - u_0} x + f(u_1) - u_1 \frac{f(u_1) - f(u_0)}{u_1 - u_0}$$

On sait que u_2 est la valeur où $y = 0$, donc on a que :

$$\frac{f_1(u_1) - f_1(u_0)}{u_1 - u_0} u_2 + f_1(u_1) - u_1 \frac{f_1(u_1) - f_1(u_0)}{u_1 - u_0} = 0 \iff \frac{f_1(u_1) - f_1(u_0)}{u_1 - u_0} u_2 = u_1 \frac{f_1(u_1) - f_1(u_0)}{u_1 - u_0} - f(u_1) \tag{1}$$

$$\iff u_2 = u_1 - f_1(u_1) \frac{u_1 - u_0}{f_1(u_1) - f_1(u_0)} \tag{2}$$

```
a = 0.9
b = 1
n <- 1
crit = TRUE
f1 <- function(x)
{
  sin(2*x)-x
}
pt <- b - f1(b)*(b-a)/(f1(b)-f1(a))
while (crit == TRUE) {
  if(f1(a)*f1(pt) < 0)
  {
    stop <- abs(b-pt)
    b <- pt
  }
  else if(f1(b)*f1(pt) < 0)
  {
    stop <- abs(a-pt)
    a <- pt
  }
  if(stop<10^-10)
  {crit <- FALSE}
  pt <- b - f1(b)*(b-a)/(f1(b)-f1(a))
```

```
  n <- n+1
}
n
```

```
## [1] 9
```

```
a <- 0.9
b <- 1
n <- 1
crit = TRUE
f1 <- function(x)
{
  sin(2*x)-x
}
pt <- b - f1(b)*(b-a)/(f1(b)-f1(a))
while (crit == TRUE) {
  if(f1(a)*f1(pt[length(pt)]) < 0)
  {
    stop <- abs(b-pt[length(pt)])
    b <- pt[length(pt)]
  }
  else if(f1(b)*f1(pt[length(pt)]) < 0)
  {
    stop <- abs(a-pt[length(pt)])
    a <- pt[length(pt)]
  }
  if(stop<10^-10)
  {crit <- FALSE}
  pt <- c(pt,b - f1(b)*(b-a)/(f1(b)-f1(a)))
  n <- n+1
}
n
```

```
## [1] 9
```

```
psi <- (1+sqrt(5))/2
m1 <- min(abs(2*cos(2*seq(from = 0.9,to = 1,by = 0.001))-1))
m2 <- max(abs(-4*sin(2*seq(from = 0.9,to = 1,by = 0.001))))
C <- 0.5*m2/m1
K <- log(C*abs(0.9-pt[length(pt)]))/psi-log(C*abs(1-pt[length(pt)]))
log((10*log(10)-log(C))/K)/log(psi)
```

```
## [1] 6.576147
```

Méthode de Newton Raphson

```
f1 <- function(x)
{
  sin(2*x)-x
}
fprime1 <- function(x)
{
  2*cos(2*x)-1
}
u <- 0.9
```

```r
crit <- TRUE
n <- 1
while(crit == TRUE)
{
  u[length(u)+1] <- u[length(u)] - f1(u[length(u)])/fprime1(u[length(u)])
  if(abs(u[length(u)-1]-u[length(u)]) < 10^-10)
  {crit <- FALSE}
  n <- n+1
}
n
```

```
## [1] 6
```

```r
m1 <- min(abs(2*cos(2*seq(from = 0.9,to = 1,by = 0.001))-1))
m2 <- max(abs(-4*sin(2*seq(from = 0.9,to = 1,by = 0.001))))
C <- 0.5*m2/m1
log((log(C)-10*log(10))/log(C*abs(u[1]-u[length(u)])))/log(2)
```

```
## [1] 3.047448
```

Remarque :* Notez que l'on considère, pour les trois méthodes exposée ici, différents critères d'arrêt dans les algorithmes. Plus précisément ;

- pour la méthode par dichotomie : on a arrêté notre algorithme quand l'intervalle obtenu après moult découpages avait une largeur en dessous de 10^{-p}
- pour la méthode de la sécante et la méthode de Newton-Raphson : on a arrêté notre algorithme quand l'actualiation du point d'approximation faisait gagner moins de 10^{-p} en précision