Approximation numérique

1 Polynômes de Lagrange

Question 1 Calculer (en fonction de n) les L_k pour des a_k équirépartis : $a_k = -1 + \frac{2k}{n}$

On peut utiliser la fonction `mul`, mais il faut alors scinder le produit en deux (pour les indices $j < i$ et $j > i$), et il faut que l'expression reste valide pour $i = 0$, donc un bête calcul avec des boucles fait aussi bien l'affaire :

```
poly := proc(n)
    local L, i, j, P, a;
    L := []:
    for i from 0 to n do
        P := 1:
        for j from 0 to n do
            a := -1 + (2*j)/n:
            if i <> j then
                P := P * (X - a) / (-1 + (2*i)/n - a):
            fi:
        od:
        L := [op(L), P]:
    od:
    L
end:
```

Question 2 Calculer l'unique polynôme P de degré $\leq n$ tel que pour tout $0 \leq i \leq n, P(a_i) = f(a_i)$

Les polynômes de Lagrange forment une base de $\mathbb{R}_n[X]$, donc P se calcule comme ceci :

$$P = \sum_{i=0}^{n} f(a_i) L_i$$

En Maple, cela donne :

```
f := x -> 1/(1 + 25*x^2):

approximer := proc(n)
    local L;
    L := poly(n):
    add(f(-1 + (2*k)/n) * L[k+1], k = 0..n):
end
```

Question 3 Faire varier n. On pourra utiliser la commande `plot[animate]` pour faire une jolie animation de la « convergence » vers f

Il est plus simple d'évaluer d'abord la liste des polynômes à afficher puis d'appler la commande `animate`.

```
A := [seq(approximer(k), k=1..12)]:
animate(plot, [[f(X), A[k]], X=-1..1], k = [seq(1..12)]);
```

❃ **Question 4** Calculer $\|f - P\|_\infty$ pour différents n

On pour de petits degrés, la commande suivante renvoie une valeur numérique :

```
evalf(maximize(abs(f(X) - A[degre]), X = -1..1));
```

mais pour de plus grands degrés elle renvoie une expressioèn formelle. Trouver une approximation du maximum d'une fonction n'est pas toujours facile. La partie suivante propose différents algorithmes de calcul d'extrema.

2 Minimisation et recherche de zéros

2.1 Recherche dichotomique

❃ **Question 5** Montrer que $a_n \to x_0$ et $b_n \to x_0$

Il suffit de vérifier que (a_n) et (b_n) sont des suites adjacentes.

❃ **Question 6** Écrire une procédure qui approxime x_0 par recherche dichotomique

```
dicho := proc(a, b, epsilon)
    local c:
    if b - a < epsilon then
        [a,b]
    else
        c := (a+b)/2:
        if evalf(f(c)) > 0 then
            dicho(a,c,epsilon)
        else
            dicho(c,b,epsilon)
        fi:
     fi:
end:
```

❃ **Question 7** Comment utiliser cet algorithme pour trouver une valeur approchée de la solution strictement positive de l'équation $x^2 = Arctan(x)$ (notée α dans la suite) ?

On applique l'algorithme à $x \mapsto x^2 - Arctan(x)$ qui est bien croissante.

2.2 Descente de gradient

Supposons qu'on cherche le minimum d'une fonction $f : \mathbb{R}^d \to \mathbb{R}$ de classe $\mathcal{C}^1$ (s'il existe).

La méthode de descente de gradient consiste à descendre le long de la courbe dans le sens opposé au gradient : $u_{k+1} = u_k - \eta \operatorname{grad} f(u_k)$. Dans la suite, on se restreint au fonctions réelles à une variable réelle ($d = 1$).

❃ **Question 8** Écrire une procédure qui, avec les arguments f, u_0, η et N calcule u_N

```
gradesc := proc(f,u0, eta, N)
    local u, g, i:
    u := u0:
    g := D(f):
    for i from 1 to N do
        u := u - eta * evalf(g(u)):
    od:
    u
end:
```

❃ **Question 9** Avec cette méthode, rechercher le minimum de $x^2 - Arctan(x)$ sur $\mathbb{R}$. Essayer différentes valeurs de u_0, η et de N

```
f := x -> (x^2 - Arctan(x))^2:
gradesc(f, 1, 0.01, 100):
```

❃ **Question 10** Adapter cette méthode pour estimer α

Il nous faut une fonction qui atteint un minimum local en α. On peut intégrer la fonction $x \mapsto x^2 - Arctan(x)$. Prendre $x \mapsto |x^2 - Arctan(x)|$ n'est pas judicieux car elle n'est pas dérivable en α. Par contre, $x \mapsto (x^2 - Arctan(x))^2$ convient.

```
f := x -> (x^2 - Arctan(x))^2:
gradesc(f, 2, 0.01, 100):
```

La commande `fsolve` est plus rapide et plus précise.

2.3 Méthode de Newton-Raphson

❃ **Question 11** Exprimer u_{k+1} en fonction de u_k pour la méthode de Newton

$$u_{k+1} = u_k - \frac{f(u_k)}{f'(u_k)}$$

❃ **Question 12** Écrire une procédure qui, avec les arguments adéquats, calcule v_N

```
newton := proc(f, a0, N)
    local a, i, g:
    a := a0 :
    g := D(f):
    for i from 1 to N do
        a := evalf(a - f(a)/g(a)):
    od:
    a
end:
```

❃ **Question 13** Combien d'itérations faut-il exécuter pour obtenir une valeur aussi précise que celle de `fsolve` ? Comparer avec les autres méthodes.

Pour la descente de gradient, il faut environ 1000 itérations pour arriver à la précision de `fsolve`. Avec la dichotomie ou la méthode de Newton, il en faut environ 30.

2.4 Vitesse de convergence

❃ **Question 14** Montrer que pour tout $u \in [a; b]$ il existe $c \in [a; b]$ tel que

$$\phi(u) - x = \frac{f''(c)}{f'(u)} \frac{(u-x)^2}{2}$$

On applique Taylor-Lagrange à f en x : il existe $c \in [a; b]$ tel que

$$f(u) = f(x) + (u-x)f'(x) + \frac{(u-x)^2}{2} f''(c)$$

$f(x) = 0$ et on divise par $f'(u) \neq 0$:

$$\frac{f(u)}{f'(u)} = (u - x) + \frac{(u-x)^2}{2}\frac{f''(c)}{f'(u)}$$

d'où $\phi(u) - x = \frac{(u-x)^2}{2}\frac{f''(c)}{f'(u)}$

Question 15 En déduire une majoration de $|\phi^n(u) - x|$. Comparer la vitesse de convergence de la méthode de Newton avec la dichotomie

Par récurrence, montrons que $|\phi^n - u| \leq (\frac{M}{2m})^{2^n-1}|u - x|^{2^n}$. C'est vrai pour $n = 0$, et

$$\begin{aligned}
|\phi^{n+1}(u) - x| &\leq \frac{M}{2m}(\phi^n(u) - x)^2 \\
&\leq \frac{M}{2m}((\frac{M}{2m})^{2^n-1}|u - x|^{2^n})^2 \\
&= (\frac{M}{2m})^{2^{n+1}-1}|u - x|^{2^{n+1}}
\end{aligned}$$

Si $\frac{M}{2m}|u - x| < 1$, la convergence est donc très rapide. Celle de la dichotomie est de l'ordre de 2^{-n}.