TD2 Scilab - Intégrales

Eric Ringeisen

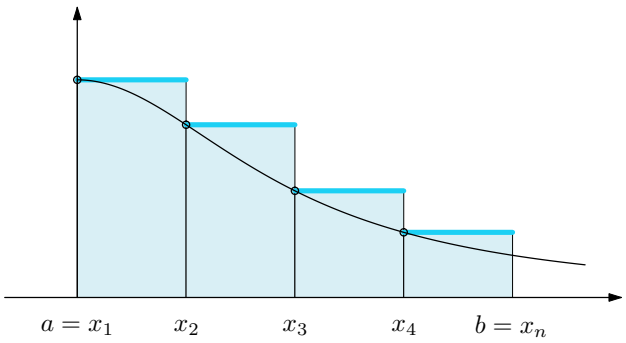
Novembre 2016

Méthode des rectangles

La méthode des rectangles pour le calcul de l'intégrale

$$I(f,a,b) = \int_a^b f(t)\,dt$$

est décrite par le graphique suivant



La valeur de l'intégrale est approchée par la somme des aires des rectangles qui sont de hauteur $f(x_i)$ et de largeur $x_{i+1} - x_i$ pour $i \in [1, n-1]$. On obtient donc l'intégrale approchée

$$R_g(f,a,b,n) = \sum_{i=1}^{n-1} f(x_i)(x_{i+1} - x_i)$$

Les points x_i sont les bornes d'un découpage de l'intervalle $[a,b]$ en $n-1$ intervalles qu'on prend généralement de même longueur, c'est à dire la longueur

$$h = \frac{b-a}{n-1} = x_{i+1} - x_i \qquad \text{on a} \qquad x_i = a + (i-1)h, \quad \forall i \in [1,n]$$

En Scilab, on peut obtenir le vecteur des abscisses x_i par `x = linspace(a, b, n)`. L'intégrale approchée peut se programmer comme suit (noter la ligne de calcul des ordonnées $y_i = f(x_i)$, et aussi les point-virgules en fin de ligne qui inhibent l'affichage des valeurs à la console)

```
function r = rectangle_gauche(f, a, b, n)
    h = (b - a) / (n - 1);
    x = linspace(a, b, n);
    y = f(x(1:n-1));
    r = h * sum(y);
endfunction
```

Q1 Définir en scilab la fonction réelle $\text{truc}(t) = \dfrac{4}{1+t^2}$ (utiliser les opérateurs `./` et `.^` pour le quotient et la puissance). Evaluer son intégrale sur l'intervalle $[0,1]$ par

```
format(20)
resu = rectangle_gauche(truc, 0, 1, 20)
```

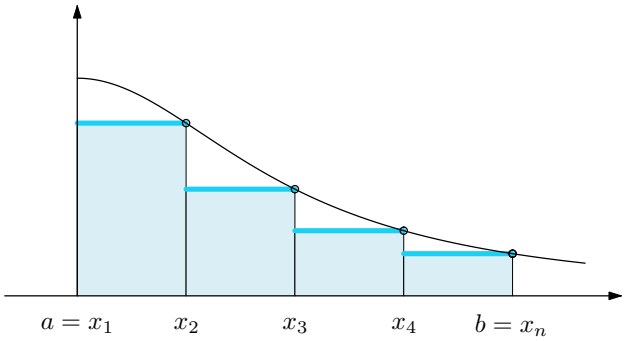
L'instruction `format` permet de définir le nombre de décimales que scilab doit afficher pour les nombres réels.

Q2 Dire pourquoi l'intégrale $I = \int_0^1 \text{truc}(t)\, dt$ vaut normalement π.

Pour contrôler la qualité du résultat approché, puisqu'on connait le résultat théorique, on peut calculer et afficher l'erreur

```
err = abs(resu - %pi)
```

La méthode des rectangles à droite est une variation sur la même idée. Le dessin suivant montre son fonctionnement



La formule d'approximation est cette fois

$$R_d(f, a, b, n) = \sum_{i=2}^{n} f(x_i)(x_{i+1} - x_i)$$

Q3 Ecrire en scilab une fonction `rectangle_droite(f, a, b, n)` qui implémente cette formule d'approximation de l'intégrale. Calculer la valeur de l'intégrale I et de l'erreur pour différentes valeurs de n, par exemple 20, 500, 10000.

Un théorème donne une majoration de l'erreur pour les deux méthodes de rectangles :

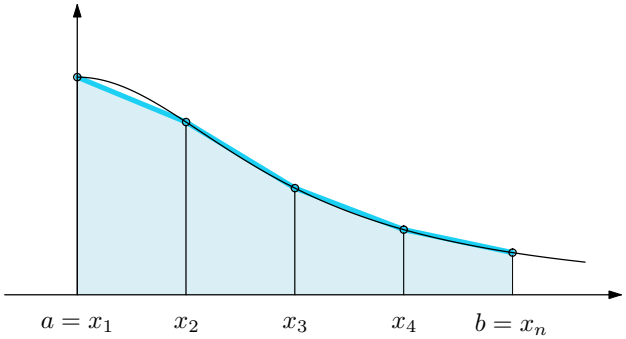
$$|I(f, a, b) - R(f, a, b, n)| \leqslant \frac{(b-a)^2}{2(n-1)} M$$

lorsque la dérivée de f est continue et que M est la plus grande valeur de $|f'(t)|$ pour $t \in [a, b]$.

Q4 Définir en scilab une fonction `trucD(t)` qui donne la dérivée de `truc` (il faudra d'abord calculer l'expression de cette dérivée à la main). Utiliser cette fonction pour donner un ordre de grandeur de la constante M en utilisant découpant l'intervalle $[0, 1]$ avec un grand nombre de points x_i et en calculant $\max(f'(x_i))$. En déduire une majoration numérique de l'erreur commise pour plusieurs valeurs de n.

Méthode des trapèzes

Cette méthode est décrite par le tracé suivant



L'intégrale est approchée par la somme des aires de *trapèzes* obtenus en reliant les points $[x_i, f(x_i)]$ situés sur la courbe, ce qui donne la formule d'approximation de l'intégrale

$$T(f, a, b, n) = \sum_{i=1}^{n-1} \frac{f(x_i) + f(x_{i+1})}{2} (x_{i+1} - x_i)$$

Q5 En scilab, étant donné un vecteur `y` de longueur n contenant les quantités $f(x_i)$, quelle expression donne un vecteur de longueur $n-1$ contenant les quantités $\dfrac{f(x_i)+f(x_{i+1})}{2}$?

Q6 Ecrire en scilab une fonction `trapeze(f, a, b, n)` qui permet de calculer l'intégrale approchée d'une fonction f par cette méthode. Calculer l'erreur commise pour plusieurs valeurs de n dans le cas de la fonction `truc` vue précédement.

Q7 On a en réalité la relation

$$T(f,a,b,n) = \frac{1}{2}\left(R_g(f,a,b,n) + R_d(f,a,b,n)\right)$$

En déduire une deuxième façon de programmer la fonction `trapeze` (on pourra appeler `trapeze2` cette deuxième implémentation). Vérifier qu'elle donne le même résultat que la première, aux erreurs d'arrondi près. Un théorème donne une majoration de l'erreur pour la méthode des trapèzes

$$|I(f,a,b) - T(f,a,b,n)| \leqslant \frac{(b-a)^3}{12(n-1)^2} M$$

lorsque la dérivée seconde f'' est continue sur $[a,b]$ et M est cette fois-ci la plus grande valeur de $|f''(t)|$ lorsque $t \in [a,b]$.

Q8 Ecrire en scilab une fonction `trucDD(t)` qui calcule la dérivée seconde de `truc` (après avoir trouvé à la main l'expression de cette dérivée seconde). L'utiliser pour donner un ordre de grandeur de la constante M et en déduire des majorations numériques de l'erreur commise pour plusieurs valeurs de n.

Méthode de Simpson

La formule d'approximation de l'intégrale est ici

$$S(f,a,b,n) = h\left(\frac{1}{6}f(a) + \frac{1}{3}\sum_{i=2}^{n-1} f(x_i) + \frac{2}{3}\sum_{i=1}^{n-1} f(m_i) + \frac{1}{6}f(b)\right)$$

avec

$$h = \frac{b-a}{n-1} \qquad \text{et} \qquad m_i = \frac{x_i + x_{i+1}}{2}$$

Q9 Ecrire en scilab une fonction `simpson(f, a, b, n)` qui calcule cette approximation.

Q10 Lorsque f est un polynôme de degré 3, la formule d'approximation de Simpson donne un résultat exact (aux erreurs d'arrondi de la machine près, qui sont typiquement de l'ordre de 10^{-16} pour des nombres flottants usuels). Vérifiez-le par exemple sur le calcul de $\displaystyle\int_1^2 \left(4t^3 - 3t^2 + 1\right) dt$, avec $n = 7$.

Q11 Calculez l'intégrale de `truc` sur l'intervalle $[0,1]$ avec la méthode de Simpson pour différentes valeurs de n et constatez que l'erreur est beaucoup plus faible qu'avec les méthodes vues ci-dessus.

Fonction prédéfinie

Scilab possède déjà une fonction prédéfinie qui calcule les intégrales, la fonction `intg(a, b, f)`. L'ordre des arguments est différent, et il n'y a pas de paramètre n. Scilab choisit seul la discrétisation en fonction de l'erreur maximale qu'on veut sur le résultat. Consultez `help(intg)` pour en savoir plus

Q12 Utilisez `intg` pour calculer l'intégrale de la fonction `truc` sur $[0,1]$. Affichez la valeur numérique de l'erreur.

Primitivation

Le calcul d'une primitive est différent de celui d'une intégrale définie puisque le résultat est une fonction au lieu d'une valeur numérique. Concrètement, si F est une primitive de la fonction f, on cherchera à calculer les valeurs $y_i = F(x_i)$ de cette primitive pour un vecteur d'abscisses `x` donné. Un shéma possible est d'approximer une dérivée par une différence finie

$$\frac{y_{i+1} - y_i}{x_{i+1} - x_i} = \frac{F(x_{i+1}) - F(x_i)}{x_{i+1} - x_i} \approx F'(x_i) = f(x_i)$$

On en déduit une formule de récurrence pour le calcul des y_i

$$y_{i+1} = y_i + (x_{i+1} - x_i) f(x_i)$$

Pour l'implémenter en scilab, il faut apprendre à remplir un vecteur à l'aide d'une *boucle*

Digression : remplissage d'un vecteur à l'aide d'une boucle

Considérons une suite de valeurs a_i obéissant à une certaine loi de formation, par exemple

$$a_{i+1} = \frac{3}{4} a_i + i - 2 \qquad \text{avec} \qquad a_1 = 5$$

On peut alors utiliser une instruction de boucle en scilab pour remplir un vecteur de taille n avec les valeurs de la suite

```
n = 10;
a = zeros(n);
a(1) = 5
for i =1:n-1
    a(i+1) = (3/4)*a(i) + i -2;
end;
```

Le vecteur a est d'abord initialisé comme un vecteur de taille n contenant des valeurs nulles, la première valeur est initialisée à 5 et le bloc `for ... end` exécute une instruction pour chaque valeur de `i` dans le vecteur `1:n-1`, c'est à dire le vecteur $(1, 2, \cdots, n-1)$. A la sortie de la boucle, le vecteur a est calculé.

Primitivation numérique

Q13 Ecrivez une fonction `primi(f, a, b, n)` qui calculera un vecteur des n valeurs y_i d'une primitive de f sur l'intervalle $[a, b]$ en utilisant la formule de récurrence indiquée plus haut pour les y_i

Q14 Calculez `primi(truc, 0, 1, 20)`. La valeur de la primitive au point 1 est elle très éloignée de π ? Comparez avec la fonction prédéfinie `integrate` de Scilab. On effectuera

```
z = integrate("truc(t)", "t", 0, linspace(0,1,20))
```

pour obtenir de scilab un vecteur de valeurs.

Q15 S'il vous reste du temps, utilisez vos connaissances du TD1 pour tracer sur un même graphique les primitives calculées avec votre fonction `primi` et avec la fonction prédéfinie `integrate` de scilab.