Calcul numérique d'intégrale [nr03] - Exercice

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Université Haute-Alsace

Version 22 mai 2018

Table des matières

C++ - Calcul numérique d'intégrale (TP)

Mots-Clés Numérique ■
Requis Structures de base, Algorithmes paramétrés, Structures répétitives, Schéma itératif ■
Difficulté • ○ ○ (2 h 30) ■

Objectif
Cet exercice étudie quelques méthodes d'approximation classiques d'intégrales (méthode des rectangles, méthode des trapèzes, méthode de Simpson) et compare la vitesse de convergence de la méthode des trapèzes et de Simpson.

1 Calcul numérique d'intégrale / pgintegr

1.1 Intégration numérique

L'intégrale $\int_a^b f$ d'une fonction f peut être vue comme l'aire (algébrique) de la partie du plan entre a et b comprise entre l'axe des abscisses et la courbe de f.

On approxime le calcul de cette surface en divisant l'intervalle $[a, b]$ en n sous-intervalles $[x_j, x_{j+1}]$ de même longueur $h = (b - a)/n$ (donc $x_j = a + j\,h$) et en approximant l'intégrale $\int_{x_j}^{x_{j+1}} f$ par une surface $\delta(j)$.

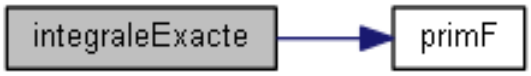
L'intégrale de la fonction est alors :

$$\int_a^b f = \sum_{j=0}^{n-1} \delta(j)$$

Soit le type `Fonction` comme étant (un pointeur d')une fonction ayant un paramètre de réel et retournant un réel.

Écrivez une fonction `integraleExacte(F,a,b)` qui calcule et renvoie l'intégrale exacte en utilisant la `Fonction` primitive F de f :

$$\int_a^b f = F(b) - F(a)$$



Écrivez une fonction `f(x)` qui servira de test (calculez et retournez le cosinus de `x` par exemple) ainsi qu'une fonction `primF(x)` qui calcule et renvoie une primitive exacte F de f (la primitive du cosinus est le sinus).

Outil C++
Les fonctions cosinus `cos(x)` et sinus `sin(x)` sont définies dans la bibliothèque `<cmath>`.

Écrivez une procédure `saisirLimites(a,b)` qui saisit des limites d'intégration dans `a` (réel) et dans `b` (réel) en s'assurant que `a < b`. Affichez l'invite :

```
Limites d'intégration?
```

Écrivez une fonction `saisirIntervalles()` qui renvoie un entier saisi par l'utilisateur, entier qui représente un nombre d'intervalles. La fonction doit s'assurer que cet entier est ≥ 1. Affichez l'invite :

```
Nombre d'intervalles?
```

Écrivez une procédure `test_exacte` qui saisit les limites d'intégration puis calcule et affiche l'intégrale exacte.

Testez. Exemple d'exécution :

```
Limites d'intégration? -0.1 0.7
==> Integrale exacte : 0.7440511038
```

1.2 Méthodes des rectangles et trapèzes

Les méthodes des rectangles et des trapèzes approxime l'aire $\delta(i)$ de la fonction dans l'intervalle $[x_j, x_{j+1}]$ par :

- le rectangle gauche s'appuyant sur f en x_j, soit $\delta(j) = f(x_j) \times h$
- le rectangle droit s'appuyant sur f en x_{j+1}, soit $\delta(j) = f(x_{j+1}) \times h$
- le trapèze s'appuyant sur f en x_j et x_{j+1}, soit $\delta(j) = \frac{1}{2}(f(x_j + f(x_{j+1})) \times h$

Les sommes correspondantes donnent le calcul approché de l'intégrale.

Écrivez une fonction `integraleRectangleG(f,a,b,n)` qui calcule et renvoie l'intégrale approchée de la `Fonction f` en subdivisant un intervalle de réels `[a,b]` en `n` sous-intervalles et en approximant l'aire par les rectangles gauches.

$$\int_a^b f = \sum_{j=0}^{n-1} \delta(j) = h \sum_{j=0}^{n-1} f(x_j) = h \sum_{j=0}^{n-1} f(a + j\,h)$$

Écrivez une fonction `integraleRectangleD(f,a,b,n)` qui calcule et renvoie l'intégrale approchée de la `Fonction f` en subdivisant un intervalle de réels `[a,b]` en `n` sous-intervalles et en approximant l'aire par les rectangles droits.

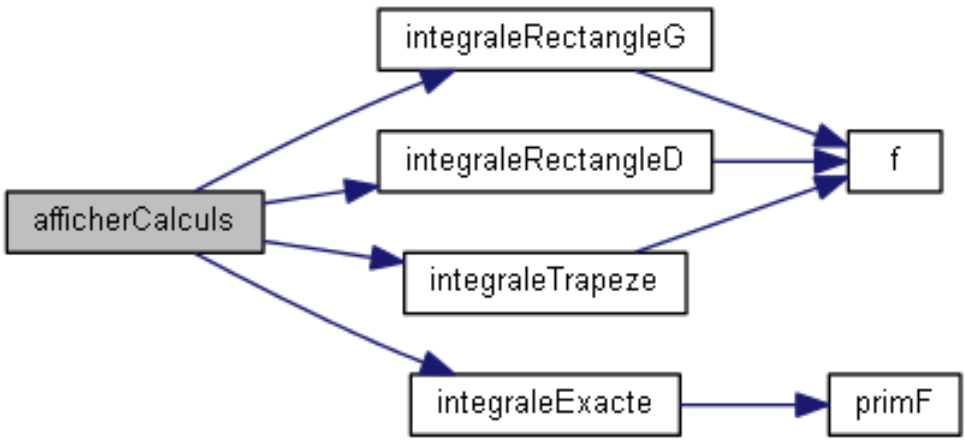
$$\int_a^b f = \sum_{j=0}^{n-1} \delta(j) = h \sum_{j=0}^{n-1} f(x_{j+1}) = h \sum_{j=1}^{n} f(a + j\,h)$$

Écrivez une fonction `integraleTrapeze(f,a,b,n)` qui calcule et renvoie l'intégrale approchée de la `Fonction f` en subdivisant un intervalle de réels `[a,b]` en `n` sous-intervalles et en approximant l'aire par les trapèzes.

$$\begin{aligned}\int_a^b f &= \sum_{j=0}^{n-1} \delta(j) = h \sum_{j=0}^{n-1} \frac{1}{2}(f(x_j + f(x_{j+1})) \\ &= h\left(\frac{1}{2}f(x_0) + \sum_{j=1}^{n-1} f(x_j) + \frac{1}{2}f(x_n)\right) \\ &= h\left(\frac{1}{2}f(a) + \sum_{j=1}^{n-1} f(a+j\,h) + \frac{1}{2}f(b)\right)\end{aligned}$$

Écrivez une procédure `afficherCalculs(a,b,n)` qui affiche les résultats des trois méthodes en subdivisant un intervalle de réels `[a,b]` en `j` sous-intervalles variant de 2 à `n`. Affichez aussi la valeur exacte de l'intégrale comme dans l'extrait d'exécution suivant :

```
Limites d'intégration? -0.1 0.7
Nombre d'intervalles? 20
==> 2 sous-intervalles
Rectangle Gauche : 0.7801362618
Rectangle Droit  : 0.6880714706
Trapeze          : 0.7341038662
==> 3 sous-intervalles
Rectangle Gauche : 0.7703249413
Rectangle Droit  : 0.7089484139
Trapeze          : 0.7396366776
...
==> 20 sous-intervalles
Rectangle Gauche : 0.748555134
Rectangle Droit  : 0.7393486549
Trapeze          : 0.7439518944
==> Calcul exact
Integrale exacte : 0.7440511038
```



Écrivez une procédure `test_recTrapezes` saisit les limites d'intégration et le nombre d'intervalles puis affiche les résultats des calculs correspondants.

Outil C++
Incluez la bibliothèque `<iomanip>` et ajoutez en première ligne du programme l'instruction l'instruction suivante (afin d'obtenir suffisamment de décimales, ici 15) :

```
cout<<precision(15);
```

Testez avec l'extrait d'exécution ci-avant, l'intégralité de l'exemple d'exécution étant fourni en téléchargement ici.
@[rsintegrat2.txt]

Quelle semble être la meilleure des trois méthodes ?
Pourquoi ?

1.3 Méthode de Simpson

La méthode de Simpson approxime la fonction f dans un sous-intervalle par une parabole. La formule est la suivante :

$$\int_a^b f = \frac{h}{6}\left(f(a)+f(b)+2\sum_{j=1}^{n-1} f(x_j)+4\sum_{j=0}^{n-1} f(x_j+h/2)\right)$$

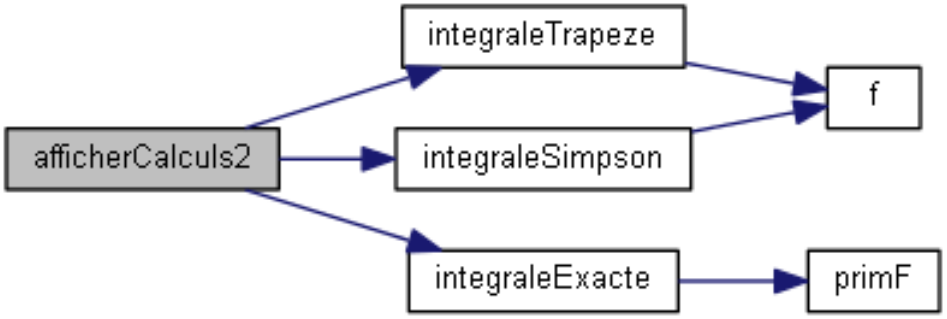
Notez que f est aussi calculée aux points milieux des intervalles dans cette méthode.

Écrivez une fonction `integraleSimpson(f,a,b,n)` qui calcule et renvoie l'intégrale approchée de la `Fonction f` en subdivisant un intervalle de réels `[a,b]` en `n` sous-intervalles et en approximant l'aire par la méthode de Simpson.

Écrivez une procédure `afficherCalculs2(a,b,n)` qui affiche les résultats de la méthode des trapèzes et de la méthode de Simpson en subdivisant un intervalle de réels `[a,b]` en `j` sous-intervalles variant de 2 à `n`, et en passant en paramètre à la méthode des trapèzes le double du nombre d'intervalles pour que la comparaison soit juste. Affichez aussi la valeur exacte de l'intégrale comme dans l'extrait d'exécution suivant :

```
Limites d'intégration? -0.2 0.5
Nombre d'intervalles? 20
==> 2*2 sous-intervalles
Trapeze : 0.6763634308
Simpson : 0.6780984155
==> 3*2 sous-intervalles
Trapeze : 0.6773255595
Simpson : 0.6780955684
...
```

```
==> 20*2 sous-intervalles
Trapeze : 0.6780775638
Simpson : 0.6780948698
Exacte  : 0.6780948694
```



Écrivez une procédure `test_simpson` qui saisit les limites d'intégration et le nombre d'intervalles puis affiche les résultats des calculs correspondants.

Testez avec l'extrait d'exécution ci-avant, l'intégralité de l'exemple d'exécution étant fourni en téléchargement ici.
@[rsintegrat3.txt]

1.4 Convergence des méthodes

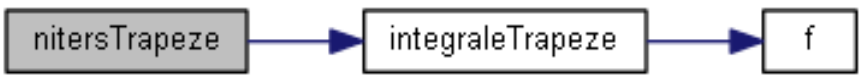
Ce problème compare la vitesse de convergence de la méthode des trapèzes et de Simpson.

Pour chaque méthode, on veut calculer la valeur à ε près et connaître le nombre d'itérations correspondant, c.-à-d. on veut calculer n et $int(a,b,n)$ tel que

$$\Delta = |int(a,b,n) - int(a,b,n-1)| < \varepsilon$$

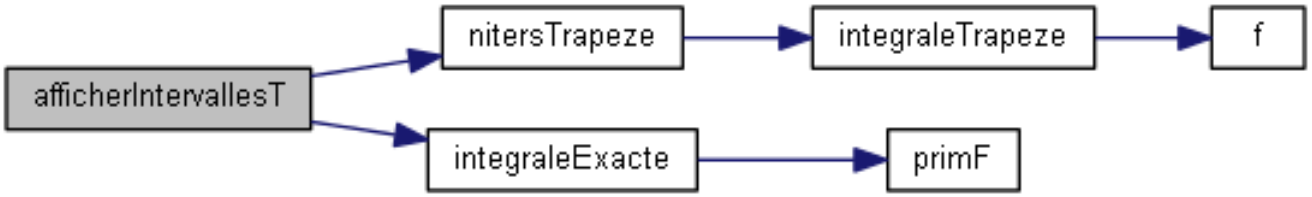
où *int* désigne l'intégrale par la méthode des trapèzes ou la méthode de Simpson.

Écrivez une fonction `nitersTrapeze(a,b,epsilon,integr)` qui calcule et restitue dans `integr` (réel) la valeur de l'intégrale approchée avec la méthode des trapèzes en subdivisant un intervalle de réels `[a,b]` en `n` sous-intervalles de sorte que $\Delta < epsilon$. La fonction renvoie le nombre d'intervalles `n` calculé.



Écrivez une procédure `afficherIntervallesT(a,b,n)` qui affiche les valeurs des intégrales et du nombre d'intervalles correspondants pour des précisions $\varepsilon = \varepsilon_0 \left(\frac{1}{2}\right)^n$ avec $\varepsilon_0 = 0.1$ pour la méthode des trapèzes comme dans l'extrait d'exécution suivant :

```
Limites d'intégration? -0.3 0.7
Nombre d'itérations? 20
==> Trapeze : 0.920077958 en 2 intervalles (0.1)
==> Trapeze : 0.9310204609 en 3 intervalles (0.05)
==> Trapeze : 0.9310204609 en 3 intervalles (0.025)
...
==> Trapeze : 0.9397292167 en 95 intervalles (0.0000001907)
==> Exacte  : 0.9397378939
```



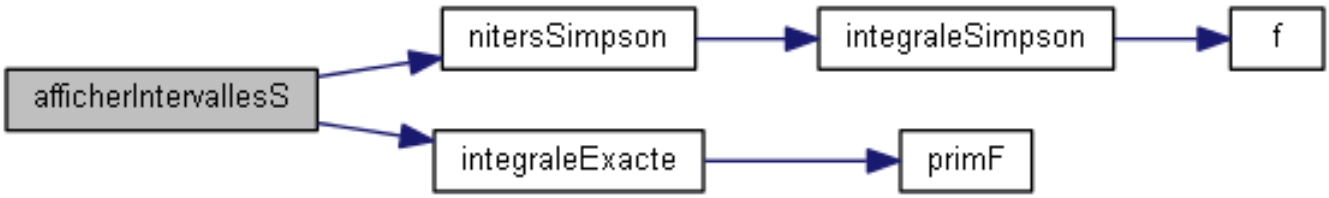
Écrivez une procédure `test_cvtrapeze` saisit les limites d'intégration et le nombre d'itérations pour les précisions de ε puis affiche les résultats des calculs correspondants.

Testez avec l'extrait ci-avant, l'intégralité de l'exemple d'exécution étant fourni en téléchargement ici.
@[rsintegrat4a.txt]

Copiez/Collez la fonction `nitersTrapeze` en `nitersSimpson(a,b,epsilon,integr)`, fonction qui calcule `integr` (réel) et renvoie le nombre d'intervalles `n` tel que `integr` est la valeur de l'intégrale approchée avec la méthode de SIMPSON en subdivisant un intervalle `[a,b]` en `n` sous-intervalles de sorte que $\Delta < epsilon$. **Attention,** renvoyez deux fois le nombre d'intervalles car cette méthode calcule aussi les points milieu.

Copiez/Collez la procédure `afficherIntervallesT` en `afficherIntervallesS(a,b,n)`, procédure qui affiche les valeurs des intégrales et du nombre d'intervalles correspondants pour des précisions $\varepsilon = \varepsilon_0 \left(\frac{1}{2}\right)^n$ avec $\varepsilon_0 = 0.1$ pour la méthode de SIMPSON comme dans l'extrait d'exécution suivant :

```
Limites d'intégration? -0.3 0.7
Nombre d'itérations? 20
==> Simpson : 0.9397584403 en 4 intervalles (0.1)
==> Simpson : 0.9397584403 en 4 intervalles (0.05)
...
==> Simpson : 0.9397380299 en 14 intervalles (0.0000001907)
==> Exacte  : 0.9397378939
```

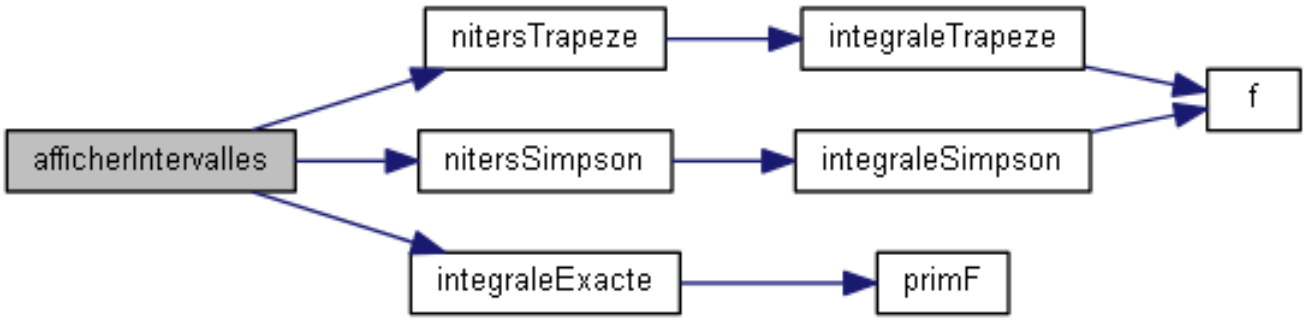


Écrivez une procédure `test_cvsimpson` saisit les limites d'intégration et le nombre d'itérations pour les précisions de ε puis affiche les résultats des calculs correspondants.

Testez avec l'extrait d'exécution ci-avant, l'intégralité de l'exemple d'exécution étant fourni en téléchargement ici.
@[rsintegrat4b.txt]

Copiez/Collez la procédure `afficherIntervallesT` en `afficherIntervalles(a,b,n)`. Modifiez la procédure de sorte qu'elle affiche les valeurs des intégrales et du nombre d'intervalles correspondants pour des précisions $\varepsilon = \varepsilon_0 \left(\frac{1}{2}\right)^n$ avec $\varepsilon_0 = 0.1$ pour les méthodes des trapèzes et de Simpson comme dans l'extrait d'exécution suivant :

```
Limites d'intégration? -0.3 0.7
Nombre d'itérations? 20
==> precision : 0.1
Trapeze : 0.920077958 en 2 intervalles
Simpson : 0.9397584403 en 4 intervalles
==> precision : 0.05
Trapeze : 0.9310204609 en 3 intervalles
Simpson : 0.9397584403 en 4 intervalles
...
==> precision : 0.0000001907
Trapeze : 0.9397292167 en 95 intervalles
Simpson : 0.9397380299 en 14 intervalles
Exacte : 0.9397378939
```



Écrivez une procédure `test_convergence` qui saisit les limites d'intégration et le nombre d'itérations pour les précisions de ε puis affiche les résultats des calculs correspondants.

Testez avec l'extrait d'exécution ci-avant, l'intégralité de l'exemple d'exécution étant fourni en téléchargement ici.
@[rsintegrat4c.txt]