

Méthodes numériques et langage C

Intégration numérique

R. Flamary

7 décembre 2015

Définitions

$$I = \int_a^b f(x) dx$$

Hypothèses

- ▶ a et b deux réels avec $a < b$.
- ▶ On suppose f intégrable sur $[a, b]$.
- ▶ f ne dispose pas de singularités sur $[a, b]$.

Mise en oeuvre

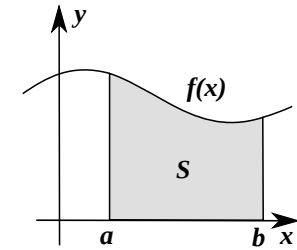
Approche classique :

- ▶ Décomposition du domaine en morceaux (un intervalle en sous-intervalles contigus).
- ▶ Intégration approchée de la fonction sur chaque morceau.
- ▶ Sommation des résultats numériques ainsi obtenus.

Utilisation de polynômes pour approcher la fonction sur chaque morceau.

Intégration numérique

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



Objectif

- ▶ Calcul numérique de la surface s à partir d'un nombre fini d'appels à la fonction.
- ▶ Cas particulier lorsque l'on a accès à un échantillonnage régulier.
- ▶ On appelle une formule de quadrature une expression linéaire fournissant une intégration approchée sur un intervalle.

Raisons

- ▶ f n'est connue qu'en certains points.
- ▶ Primitive F connue mais pas une fonction élémentaire. (intégrale de $\exp(x^2)$).
- ▶ Primitive trop difficile à calculer numériquement.

2 / 16

Intégration par quadrature simple

$$I = \int_a^b f(x) dx \quad \text{avec} \quad h = (b - a)$$

Formule du rectangle

$$I = hf(a) \text{ (ou } hf(b)) + O\left(\frac{h^2}{2} f'(\mu)\right) \quad \mu \in [a, b]$$

Méthode d'ordre 0, exacte pour les fonctions constantes.

Formule du point milieu

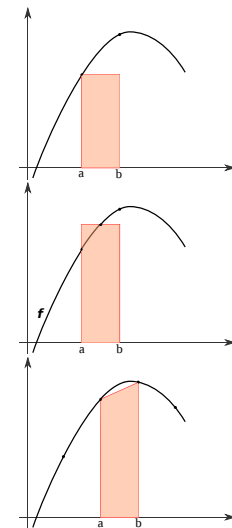
$$I = hf\left(\frac{a+b}{2}\right) + O\left(\frac{h^3}{24} f''(\mu)\right) \quad \mu \in [a, b]$$

Méthode d'ordre 1, exacte pour des fonctions linéaires.

Formule du trapèze

$$I = h \frac{f(a) + f(b)}{2} + O\left(\frac{h^3}{12} f''(\mu)\right) \quad \mu \in [a, b]$$

Méthode d'ordre 1, exacte pour des fonctions linéaires.



Intégration par quadrature simple (2)

Formule de Simpson

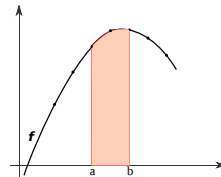
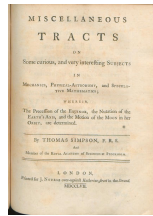
- Popularisée pas Simpson mais utilisée par Kepler 100 ans plus tôt.
- Interpolation de f par un polynôme d'ordre 2.
- On calcule la valeur de la fonction en a, b et $m = \frac{a+b}{2}$.
- Interpolation polynomiale de Lagrange :

$$P(x) = f(a) \frac{(x-m)(x-b)}{(a-m)(a-b)} + f(m) \frac{(x-a)(x-b)}{(m-a)(m-b)} + f(b) \frac{(x-a)(x-m)}{(b-a)(b-m)}.$$

- Intégrale approchée obtenue en intégrant le polynôme :

$$I = h \frac{f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)}{6} + O\left(\frac{h^5}{90 * 2^5} f^{(4)}(\mu)\right)$$

- Méthode d'ordre 2, exacte pour des fonctions quadratiques et cubiques.



5 / 16

Formule de Newton-Cotes (2)

$$I = \int_a^b f(x) dx \approx \sum_{i=0}^n w_i f(x_i)$$

- Permet de retrouver les formules de quadrature simple :

Degré	Nom	Formule	Erreur
1	Trapèze	$\frac{h}{2}(f_0 + f_1)$	$-\frac{h^3}{12} f^{(2)}(\mu)$
2	Simpson	$\frac{h}{6}(f_0 + 4f_1 + f_2)$	$-\frac{h^5}{2880} f^{(4)}(\mu)$
4	Boole	$\frac{h}{90}(7f_0 + 32f_1 + 12f_2 + 32f_3 + 7f_4)$	$-\frac{h^7}{1935360} f^{(6)}(\mu)$

pour $\mu \in [a, b]$ et $h = (b - a)$.

- Formule définie pour n'importe quel degré n .
- Problème d'instabilité numérique pour de grand n (Phénomène de Runge).
- En pratique, on préfère découper la fonction en petits intervalles et utiliser des quadratures de degré faible sur ces intervalles.

7 / 16

Formules de Newton-Cotes

- Intégration numérique sur $[a, b]$ sur $n + 1$ points régulièrement échantillonnés.
- Soit $x_i = a + i\Delta$ avec $\Delta = \frac{(b-a)}{n} = \frac{h}{n}$ et $f_i = f(x_i), \forall i$.
- La formule de degré n est définie par

$$I = \int_a^b f(x) dx \approx \sum_{i=0}^n w_i f(x_i)$$

où les w_i sont appelés coefficients de quadrature.

- On déduit les poids w_i d'une interpolation de Lagrange de la fonction :

$$f(x) \approx L(x) = \sum_{i=0}^n f(x_i) l_i(x), \quad \text{avec} \quad l_i(x) = \prod_{j=0, j \neq i}^n \frac{x - x_j}{x_i - x_j}$$

- L'intégrale devient donc

$$I \approx \int_a^b L(x) dx = \int_a^b \sum_{i=0}^n f(x_i) l_i(x) dx = \sum_{i=0}^n f(x_i) \underbrace{\int_a^b l_i(x) dx}_{w_i}. \quad (1)$$

6 / 16

Quadrature de Gauss

- On approche encore une fois l'intégrale par

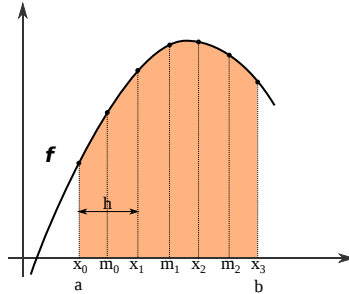
$$I = \int_a^b f(x) dx \approx \sum_{i=1}^n w_i f(x_i)$$

- Les w_i sont les coefficients de quadrature et les x_i sont choisis comme les racines de polynômes orthogonaux.
- On n'utilise pas d'échantillonnage régulier, possibilité d'avoir une meilleure approximation de f .
- Intégration exacte pour des polynômes de degré $2n - 1$.
- Exemple pour $[a, b] = [-1, 1]$ avec les polynômes de Legendre :

Nb de points	Poids w_i	Points x_i	Poly. de Legendre
1	2	0	x
2	1, 1	$-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}$	$(3x^2 - 1)/2$
3	$\frac{5}{9}, \frac{8}{9}, \frac{5}{9}$	$-\frac{\sqrt{3}}{5}, \frac{\sqrt{3}}{5}$	$(5x^3 - 3x)/2$

8 / 16

Méthodes composites



- ▶ Les formules de Newton Cotes ont toutes une erreur sous la forme d'une puissance de $(b - a)$.
- ▶ En pratique on découpe $[a, b]$ en n sous intervalles et on utilise les formules de Newton-Cotes sur les petits intervalles.
- ▶ La longueur de l'intervalle d'intégration devient donc $h = \frac{b-a}{n}$.
- ▶ Nous définissons les points d'échantillonnage régulier suivants :

$$x_k = a + kh, \quad m_k = a + (k + 1/2)h, \quad \forall k \in 0, \dots, n$$

9 / 16

Méthodes composites (2)

Méthode du point milieu

$$I = h \sum_{k=0}^{n-1} f(m_k)$$

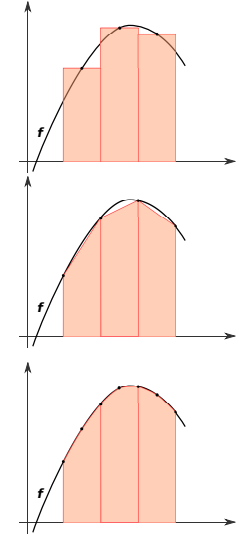
Méthode du trapèze

$$I = h \left(\frac{f(a) + f(b)}{2} + \sum_{k=1}^{n-1} f(x_k) \right)$$

Méthode de Simpson

$$I = \frac{h}{6} \left(f(a) + f(b) + 2 \sum_{k=1}^{n-1} f(x_k) + 4 \sum_{k=0}^{n-1} f(m_k) \right)$$

avec $h = \frac{b-a}{n}$.



10 / 16

Pointeur de fonction en C

- ▶ Comment passer une fonction en paramètre pour l'intégrer ?
- ▶ On utilise des pointeurs de fonction.
- ▶ Un pointeur de fonction est déclaré par :
type_ret (*nomfunc)(params);

Code source

```
1 double sum_f_a_b(double (*f)(double)
2     ,double a,double b)
3 {
4     return f(a)+f(b);
5 }
6 int main()
7 {
8     printf("sqrt(1)+sqrt(2)=%f\n",
9         sum_f_a_b(sqrt,1,2));
10    printf("cos(0)+cos(1)=%f\n",
11        sum_f_a_b(cos,0,1));
12    printf("exp(0)+exp(1)=%f\n",
13        sum_f_a_b(exp,0,1));
14 }
```

Sortie

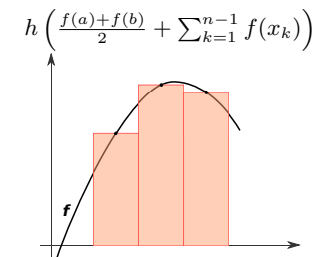
```
1 ./ex_pfunc
2 sqrt(1)+sqrt(2)=2.414214
3 cos(0)+cos(1)=1.540302
4 exp(0)+exp(1)=3.718282
```

11 / 16

Implémentation des méthodes composites en C

Méthode du point milieu

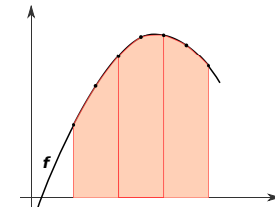
```
1 double int_middle(double (*f)(double)
2     ,double a,double b,int n)
3 {
4     double res=0;
5     double h=(b-a)/n;
6     for (int i=0;i<n;i++)
7         res+=f(a+(i+.5)*h);
8     return res*h;
9 }
```



Méthode de Simpson

```
1 double int_simpson(double (*f)(
2     double),double a,double b,int n)
3 {
4     double h=(b-a)/n;
5     double res=f(a)+f(b)+4*f(a+h/2);
6     for (int i=1;i<n;i++)
7         res+=2*f(a+i*h)+4*f(a+(i+.5)*h);
8     return res*h/6;
9 }
```

$$\frac{h}{6} \left(f(a) + f(b) + 2 \sum_{k=1}^{n-1} f(x_k) + 4 \sum_{k=0}^{n-1} f(m_k) \right)$$



12 / 16

Méthodes adaptatives

Principe

- ▶ Échantillonnage fin pas nécessaire sur tout l'intervalle $[a, b]$.
- ▶ On adapte l'échantillonnage le long de l'intervalle de manière récursive.
- ▶ `integrate(f, a, b, tau)` :
 1. Calcul de $I \approx \int_a^b f(x)dx$
 2. Estimation de l'erreur $\epsilon \approx |I - \int_a^b f(x)dx|$
 3. Si $\epsilon > \tau$,
 $I = \text{integrate}(f, a, (a+b)/2, \tau) + \text{integrate}(f, (a+b)/2, b, \tau)$
 4. Retourner I
- ▶ Approches classiques :
 - ▶ Méthode de Romberg (Trapèze+ extrapolation de Richardson).
 - ▶ Méthode de Simpson adaptative (Simpson+ extrapolation de Richardson).

Méthode de Simpson adaptative

Sur l'intervalle $[a, b]$ avec $m = \frac{a+b}{2}$, la méthode de Simpson s'appelle avec $S(a, b)$.

1. Calculer $S(a, b)$, $S(a, m)$ et $S(m, b)$.
2. Si l'erreur $|S(a, b) - S(a, m) - S(m, b)|/15 > \tau$ on divise l'intervalle $[a, b]$.
3. Sinon, on retourne $S(a, m) + S(m, b) + (S(a, m) + S(m, b) - S(a, b))/15$.

13 / 16

Méthode de Simpson adaptative en C

Implémentation simplifiée

```

1 double simpson(double (*f)(double),
2     double a, double b)
3 {
4     return (b-a)/6*(f(a)+4*f((a+b)/2)+f(b));
5 }
6 double int_adaptsimpson(double (*f)(double), double a, double b, double tau)
7 {
8     double m=(a+b)/2;
9     double Sab=simpson(f, a, b), Sam=simpson(f, a, m), Smb=simpson(f, m, b);
10    if (fabs(Sab-Sam-Smb)/15<tau)
11        return Sam+Smb+(Sam+Smb-Sab)/15;
12    else
13        return int_adaptsimpson(f, a, m, tau)
14            +int_adaptsimpson(f, m, b, tau);
15 }

```

Discussion

- ▶ Fonction récursive (complexité dépend de f et τ).
- ▶ Paramètre de précision τ directement lié à l'erreur acceptable.
- ▶ Implémentation non efficace, comment faire mieux ?

14 / 16

Méthodes de Monte Carlo

$$I = \int_a^b f(x)dx \approx \frac{b-a}{n} \sum_{i=1}^n f(x_i), \quad x_i \sim U(a, b) \quad \forall i$$

- ▶ Intégration numérique utilisant des réalisations de variables aléatoires.
- ▶ La méthode converge vers la bonne valeur lorsque $n \rightarrow \infty$
- ▶ L'erreur en espérance $\epsilon = O\left(\frac{1}{\sqrt{n}}\right)$ décroît très lentement.
- ▶ Très utilisée pour les intégrales multiples car erreur indépendante de la dimension.
- ▶ Version plus efficaces basées sur l'échantillonnage préférentiel (VEGAS, MISER).

Implémentation

```

1 double int_montecarlo(double (*f)(double), double a, double b, int n)
2 {
3     double res=0;
4     for (int i=0; i<n; i++)
5         res+=f(a+rand()*1.0/RAND_MAX*(b-a));
6     return res/n;
7 }

```

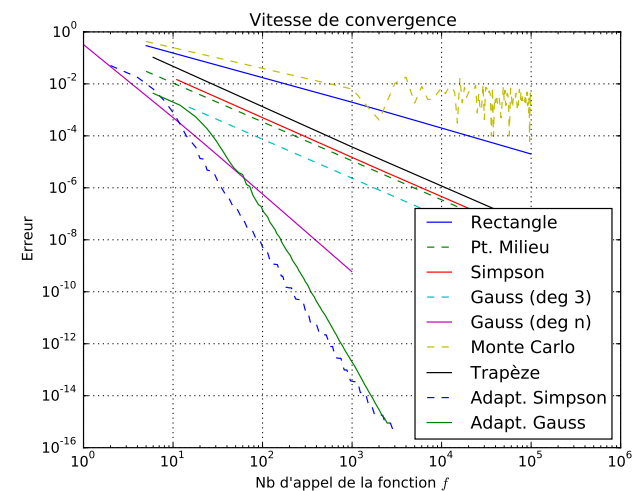
Discussion

- ▶ n nombre de réalisations.
- ▶ `rand()/RAND_MAX` réalisation d'une variable aléatoire uniforme sur $[0, 1]$.
- ▶ `a+rand()*1.0/RAND_MAX*(b-a)` réalisation d'une variable aléatoire uniforme sur $[a, b]$.

15 / 16

Comparaison numérique

$$I = \int_0^1 4\sqrt{1-x^2}dx = \pi$$



16 / 16