情報数学C

Mathematics for Informatics C

第7回 数値積分
(区分求積法,台形公式,シンプソン公式)

情報メディア創成学類
藤澤誠

今回の講義内容

- **今日の問題**
- 区分求積法と台形公式
- シンプソン公式
- ロンバーグ法
- ガウス型積分公式
- 重積分, モンテカルロ積分

今回の講義で解く問題

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

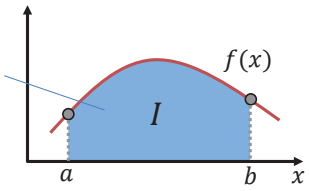
今回の講義で解く問題

xについての方程式$f(x)$を区間$[a,b]$で積分した時の値を求める問題(求積法:quadrature)

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

数値計算による積分を**数値積分**(numerical integration)と呼ぶ

1次元関数の積分は**面積**,
2重積分は**体積**を表す



今回の講義で扱う問題

数値積分は**どんなところで使われる?**

基本的には**面積や体積**を求めるために使われるが蓄積されたデータの**平滑化, フィルタリング**などにも応用される

信号処理・画像処理 信号・画像の平滑化フィルタ, 画像の面積計算など

制御工学 フィードバック制御の一種であるPID制御では積分を使う(PID:Proportional-**Integral**-Differential)

物理シミュレーション 熱や流体, 弾塑性体などの物理的な物体の変形や動きの計算, 電磁波や音の伝播の計算など幅広く使われる有限要素法(FEM)は積分に基づく方法

今回の講義で解く問題

積分問題の例

例) $f(x) = 2x^2 - 9x + 14 - \frac{9}{x} + \frac{2}{x^2} \quad (x > 0)$

について曲線$y = f(x)$とx軸とで囲まれた部分の面積を求めよ.

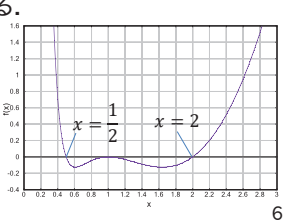
(2017年度筑波大学前期日程 数学問題より)

[数学での解き方] $f(x) = 0$の根を求めた結果と極値を調べた結果から, $\frac{1}{2} \le x \le 2$で積分する.

$$S = -\int_{\frac{1}{2}}^{2} \left(2x^2 - 9x + 14 - \frac{9}{x} + \frac{2}{x^2}\right) dx$$

$$= -\left[\frac{2}{3}x^3 - \frac{9}{2}x^2 + 14x - 9\log_e|x| - \frac{2}{x}\right]_{\frac{1}{2}}^{2}$$

$$= -\frac{99}{8} + 18\log_e 2 \cong 0.1016492501$$



今回の講義で解く問題

今回の講義で扱う「積分」についての前提条件

- データ点(サンプリング点)x_iとそこでの関数値(観測値)f_iは与えられている
- 不定積分$\int f(x)dx$ではなく**定積分**$\int_a^b f(x)dx$を計算する(積分範囲(境界)が決まっている)：$x_i \in [a,b]$
- 積分の問題は次回説明予定の常微分方程式の形で表すことも可能($\frac{df}{dx} = g(x,y)$)だけど，今回は**積分をそのまま数値的に計算**する方法を扱う

今回の講義内容

- 今日の問題
- **区分求積法と台形公式**
- シンプソン公式
- ロンバーグ法
- ガウス型積分公式
- 重積分，モンテカルロ積分

ニュートン・コーツ型積分公式

データ点(x_i, f_i)が得られたときにそれを用いて**数値積分することは可能か？**

⇒ **ラグランジュ補間**で$f(x)$を近似した$L_n(x)$を使えばどうか？

ラグランジュ補間公式

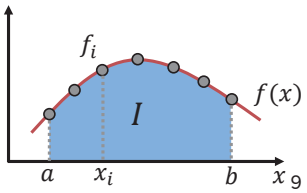
$$L_n(x) = \sum_{i=0}^{n} f_i\, l_i(x)$$

$$l_i(x) = \prod_{j=0\sim n, j\neq i} \frac{x - x_j}{x_i - x_j}$$

$$I = \int_a^b f(x)dx \approx \int_a^b \sum_{i=0}^{n} f_i\, l_i(x)\, dx$$

⇩ f_iは観測値なので積分$\int$の外に出せる

$$I \approx \sum_{i=0}^{n} f_i \int_a^b l_i(x)dx$$



ニュートン・コーツ型積分公式

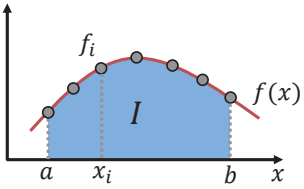
$\alpha_i = \int_a^b l_i(x)dx$とすると：

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

⇒ 観測点f_iに重み係数α_iを掛けて足し合わせれば積分が計算できる！

ニュートン・コーツ公式

この形の数値積分公式を**ニュートン・コーツ型積分公式**と呼ぶ．



区分求積法

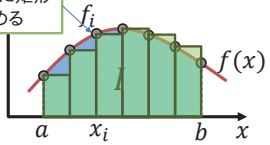
区分求積法：ニュートン・コーツ型積分公式で最も簡単(その代わり低精度)な方法

⇒ 積分を**矩形(四角形)の集合**で近似

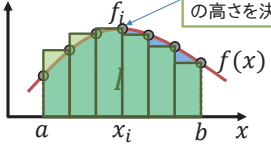
積分区間$[a,b]$をn等分し，各区分の境界をx_i $(i = 0,1,\ldots,n)$とすると，各区分の幅(**刻み幅**)は$h = x_i - x_{i-1} = (b-a)/n$となる．

$$I \approx \sum_{i=0}^{n-1} hf_i \quad \text{もしくは} \quad I \approx \sum_{i=1}^{n} hf_i$$

左端を基準に矩形の高さを決める



右端を基準に矩形の高さを決める



*$f_i = f(x_i)$でx_iは$x_i = a + ih$で計算できる

区分求積法

参考) 関数$f(x+\Delta x)$のx周りのテイラー展開：
$f(x+\Delta x) = f(x) + \Delta x f'(x) + \frac{\Delta x^2}{2!} f''(x) + \cdots$

区分求積法の精度(誤差)

右端を使ったときの1区間の面積(**近似値**)をテイラー展開：

$$\widehat{I_i} = f(x_i + h)h = h\left(f(x_i) + hf'(x_i) + \frac{h^2}{2!}f''(x_i) + \cdots\right)$$

一方で$f(x)$の不定積分を$F(x)$とする($F'(x) = f(x)$)と**真値**は：

$$I_i = \int_{x_i}^{x_i+h} f(x)dx = [F(x)]_{x_i}^{x_i+h} = F(x_i + h) - F(x_i)$$

この項をテイラー展開

$$= F(x_i) + hF'(x_i) + \frac{h^2}{2!}F''(x_i) + \frac{h^3}{3!}F'''(x_i) + \cdots - F(x_i)$$

$$= h\left(f(x_i) + \frac{h}{2}f'(x_i) + \frac{h^2}{6}f''(x_i) + \cdots\right)$$

$[x_i, x_{i+1}]$の誤差：$E_i = I_i - \widehat{I_i} = \frac{h^2}{2}f'(x) + O(h^3)$

1区間の誤差

区分求積法

区分求積法の精度(誤差)

1区間あたりの誤差が出たのでこれを全体区間$[a,b]$分足し合わせると:

$$|E| \le \frac{h^2}{2} f'(x) \times n = \frac{h^2}{2} f'(x) \frac{b-a}{h} = \frac{b-a}{2} h f'(x)$$

誤差のオーダは$O(h)$(もしくは$O(1/n)$)でほぼ**刻み幅に比例する精度**となる.

許容誤差εの精度を得るために必要な刻み幅 : $\frac{2\varepsilon}{(b-a)\,f'}$

例) $f'(x)$がほぼ1の関数を$[0,1]$で積分するときの計算量:

許容誤差$\varepsilon = 10^{-3}$での反復数(分割数) : 500回
許容誤差$\varepsilon = 10^{-6}$での反復数(分割数) : 50万回

⇒ できれば$O(h^2)$とかにしたい($O(h^2)$ならそれぞれ23回,708回で済む)

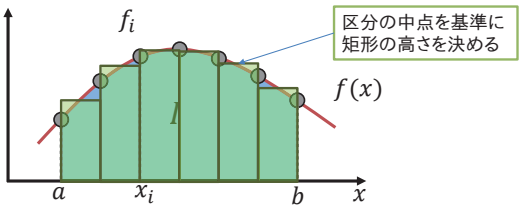
区分求積法

各区分の端の観測値を使うより**区分の真ん中(中点)**の方がより精度が高いのでは?

区間$[x_{i-1}, x_i]$の中点は $(x_{i-1}+x_i)/2$なので

$$I \approx \sum_{i=1}^{n} h\, f\left(\frac{x_{i-1}+x_i}{2}\right)$$

中点法



区分求積法

中点法による数値積分の誤差(先ほどと同じくテイラー展開を使う)

$$\hat{I}_i = hf\left(x_i + \frac{h}{2}\right) = h\left(f(x_i) + \frac{h}{2}f'(x_i) + \frac{h^2}{4\cdot 2!}f''(x_i) + \cdots\right)$$

区分の両端を使う場合と比べると真値との差でf'の項まで消えるので, n区間だと $|E| \le \frac{b-a}{12}h^2 f''(x)$

$$\Rightarrow \text{誤差は} O(h^2)\left(or\ O\left(\frac{1}{n^2}\right)\right)$$

- より精度の高い方法もあるが, 区分求積法は**分割数を∞にする(刻み幅を0にする)と真値と等しくなる**ため, 他の方法の**精度検証**に使われたりもする

台形公式

中点を使う方法だと$a = x_0, b = x_n$としてしまうと全体区間の両端部分で工夫が必要

⇒ 各区間の**両端値(x_iとx_{i+1})を使う場合でも精度**を上げられないか?

全区間$[a,b]$を1区間(要は$n=1$)としたときのニュートン・コーツ公式を具体的に求めてみよう!

$$\alpha_0 = \int_a^b l_0(x)dx = \int_a^b \frac{x-x_1}{x_0-x_1}dx = -\frac{1}{h}\left[\frac{1}{2}x^2 - bx\right]_a^b = \frac{h}{2}$$

$$\alpha_1 = \int_a^b l_1(x)dx = \int_a^b \frac{x-x_0}{x_1-x_0}dx = \frac{1}{h}\left[\frac{1}{2}x^2 - ax\right]_a^b = \frac{h}{2}$$

$n=1$なので$a=x_0, b=x_n=x_1$
更に, $h = x_1 - x_0$

$l_i(x) = \prod_{j=0\sim n, j\neq i} \frac{x-x_j}{x_i-x_j}$

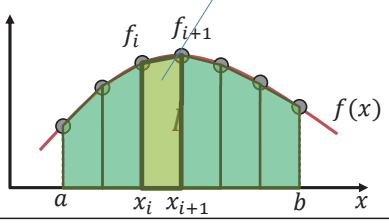
台形公式

α_0, α_1から積分公式は: $I \approx \sum_{i=0}^{1} \alpha_i f_i = \alpha_0 f_0 + \alpha_1 f_1 = \frac{h}{2}(f_0 + f_1)$

1分割ではなくn分割とすると:

$$I \approx \sum_{i=0}^{n-1} \alpha_i f_i = \sum_{i=0}^{n-1} \frac{h}{2}(f_i + f_{i+1}) = \frac{h}{2}(f_0 + 2\sum_{i=1}^{n-1} f_i + f_n)$$

台形公式



台形公式

台形公式の精度(誤差)

参考) 関数$f(x+\Delta x)$のx周りのテイラー展開 : $f(x+\Delta x) = f(x) + \Delta x f'(x) + \frac{\Delta x^2}{2!}f''(x) + \cdots$

台形公式はテイラー展開からも求められる.

$f_{i+1} = f(x_i + h)$のテイラー展開(次の計算のための準備):

$$f_{i+1} = f_i + hf_i' + \frac{h^2}{2}f_i'' + \frac{h^3}{6}f_i''' + \cdots$$

f_i'について解く ⇒

$$f_i' = \frac{f_{i+1}-f_i}{h} - \frac{h}{2}f_i'' - \frac{h^2}{6}f_i''' + \cdots$$

積分$I = \int_a^b f(x)dx$の範囲を$[x_i, x_{i+1}]$ ($x = x_i + z$とすると$z \in [0,h]$)にしてテイラー展開:

$$I_i = \int_a^b f(x)dx = \int_0^h f(x_i+z)dz$$

x_i周りでテイラー展開後, 定積分を計算

$$= hf_i + \frac{h^2}{2}f_i' + \frac{h^3}{6}f_i'' + \frac{h^4}{24}f_i''' + \cdots$$

f_i'の式を代入

台形公式

台形公式の精度(誤差)

台形公式はテイラー展開からも求められる.

$$I_i = h\frac{f_{i+1}+f_i}{2} - \frac{h^3}{12}f_i'' - \frac{h^5}{120}f_i'''' - \cdots$$

台形公式　　誤差($O(h^3)$)

1区間あたりの誤差が約$\frac{h^3}{12}f_i''$なので,n区間ならば,

$$E \approx \frac{nh^3}{12}f_i'' = \frac{b-a}{12}h^2 f_i''$$

台形公式の**誤差は**$O(h^2)(or\ O\left(\frac{1}{n^2}\right))$

(中点を使う場合と同じ)

区分求積法と台形公式

n分割の区分求積法(右端)と台形公式のコード例

(式をそのまま計算するだけなので計算手順はなしでコード例のみ)

区分求積法(右端)

```
double h = (b-a)/n; // 分割区間の横幅
S = 0;
for(int i = 0; i < n; ++i){
    double f = func(a+(i+1)*h);
    S += f*h;
}
```

区間iの右側境界$x_{i+1} = a + (i+1)h$での関数値を使って積分を計算

台形公式

```
double h = (b-a)/n; // 分割区間の横幅
double f1, f2;// 分割区間の縦幅(台形の長辺と短辺)
f2 = func(a);
S = 0;
for(int i = 0; i < n; ++i){
    f1 = f2;
    f2 = func(a+(i+1)*h);
    S += (f1+f2)*h/2;
}
```

区間iの両端x_iとx_{i+1}での関数値を計算するが,$f(x_i)$については**前の反復で計算済みの値**が使えることに注意

*このコード例では任意位置の関数値から計算しているが,離散データからも直接計算できる

区分求積法と台形公式

区分求積法(右端)と台形公式の実行結果例

誤差評価のために,$f(x) = e^x$と設定して,区間$[a,b] = [0,1]$で数値積分(真値は$e-1$)

$n = 10$の場合

```
segment int f(x) = 1.805627583,  error = 0.08734575435   区分求積法
trapezoidal int f(x) = 1.719713491,  error = 0.00143166293 台形公式
ground truth = 1.718281828
```

$n = 100$の場合

```
segment int f(x) = 1.726887557,  error = 0.008605728134   区分求積法
trapezoidal int f(x) = 1.718296147,  error = 1.431899137e-05台形公式
ground truth = 1.718281828
```

台形公式の方が**同じ分割数でもより精度が高い**(誤差が小さい)

分割数を更に大きくすれば誤差の差はなくなっていくが,刻み幅を小さくしすぎると今度は**丸め誤差の影響**が出るので注意が必要

今回の講義内容

- 今日の問題
- 区分求積法と台形公式
- **シンプソン公式**
- ロンバーグ法
- ガウス型積分公式
- 重積分,モンテカルロ積分

シンプソン公式

ここまでの方法はn個に分割した各区間で**1次補間多項式**で近似していた.

⇒ **2次の補間多項式**を使えばもっと精度が良くなる?

シンプソンの公式(Simpson rule)

2次多項式には**係数が3つ**あるので,2点(x_i, x_{i+1})だけでは計算できない

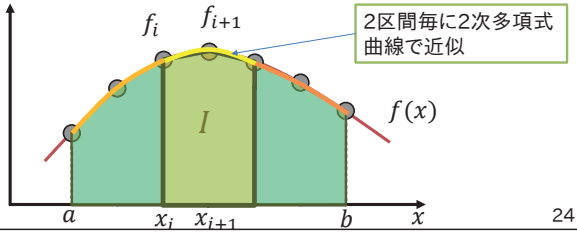
- 1区間だけでなく**2区間3点**(x_{i-1}, x_i, x_{i+1})での観測値$f(x)$を使う
- 2区間を1つとするのでn個の補間多項式のために**分割数を**$2n$**とする**

シンプソン公式

ここまでの方法はn個に分割した各区間で**1次補間多項式**で近似していた.

⇒ **2次の補間多項式**を使えばもっと精度が良くなる?

シンプソンの公式(Simpson rule)



シンプソン公式

台形公式と同じく，ニュートン･コーツ公式からシンプソン公式を導出してみよう!

2次のラグランジュ補間多項式を使うとすると：

$$\alpha_0 = \int_a^b \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}dx = \frac{1}{2h^2}\int_a^b (x-x_1)(x-x_2)dx$$

ここで$x_0 = a, x_1 = a+h, x_2 = a+2h = b$となり，$x = a+th$で変数変換

$$\alpha_0 = \frac{1}{2h^2}\int_0^2 (t-1)(t-2)h^3 dt = \frac{h}{2}\left[\frac{t^3}{3} - \frac{3}{2}t^2 + 2t\right]_0^2 = \frac{h}{3}$$

同様にして計算すると $\alpha_1 = \frac{4}{3}h, \alpha_2 = \frac{h}{3}$

ニュートン･コーツ公式から：$I_0 \approx \frac{h}{3}(f_0 + 4f_1 + f_2)$

ニュートン･コーツ公式

$$I \approx \sum_{i=0}^{n} \alpha_i f_i, \alpha_i = \int_a^b l_i(x)dx \quad l_i(x) = \prod_{j=0\sim n, j\neq i} \frac{x-x_j}{x_i-x_j}$$

シンプソン公式

前ページ青枠の式を$2n$区間$(x_0, x_1, \ldots, x_{2n})$に広げると

$$I = \int_a^b f(x)dx \approx \sum_{i=0}^{n-1} \frac{h}{3}(f_{2i} + 4f_{2i+1} + f_{2i+2})$$

（f_{2i}：偶数項，$4f_{2i+1}$：奇数項，f_{2i+2}：偶数項）

奇数項には係数4が掛かる，偶数項はiと$i+1$で同じ項が2つずつ

$$= \frac{h}{3}\{f_0 + 4(f_1 + f_3 + \cdots + f_{2n-1}) + 2(f_2 + f_4 + \cdots + f_{2n-2}) + f_{2n}\}$$

$$I \approx \frac{h}{3}\left\{f_0 + 4\sum_{i=1}^{n} f_{2i-1} + 2\sum_{i=1}^{n-1} f_{2i} + f_{2n}\right\}$$

シンプソンの1/3公式

同様にして3区間で3次補間多項式で近似すると：

$$I \approx \frac{h}{8}\left\{f_0 + 3\sum_{i=1}^{n} f_{3i-2} + 3\sum_{i=1}^{n} f_{3i-1} + 2\sum_{i=1}^{n-1} f_{3i} + f_{3n}\right\}$$

シンプソンの3/8公式

シンプソン公式

シンプソン公式の精度(誤差)

シンプソン公式もテイラー展開で求められる．

$f_{i+1} = f(x_i + h)$と$f_{i-1} = f(x_i - h)$のテイラー展開(次の計算のための準備)：

$$f_{i+1} = f_i + hf_i' + \frac{h^2}{2}f_i'' + \frac{h^3}{6}f_i''' + \frac{h^4}{24}f_i^{(4)} + \frac{h^5}{120}f_i^{(5)} + \cdots$$

$$f_{i-1} = f_i - hf_i' + \frac{h^2}{2}f_i'' - \frac{h^3}{6}f_i''' + \frac{h^4}{24}f_i^{(4)} - \frac{h^5}{120}f_i^{(5)} + \cdots$$

積分$I = \int_a^b f(x)dx$の範囲を$[x_{i-1}, x_{i+1}]$にしてテイラー展開：

$$I_i = \int_a^b f(x)dx = \int_{-h}^{h} f(x_i + z)dz$$

x_i周りでテイラー展開後，定積分を計算

$$= 2hf_i + \frac{h^3}{3}f_i'' + \frac{h^5}{60}f_i^{(4)} + \frac{h^7}{2520}f_i^{(6)} + \cdots$$

$f_{i-1} + f_{i-1}$からf_i''の式を求めて代入

シンプソン公式

シンプソン公式の精度(誤差)

f_i''の式を代入して整理すると：

$$I_i = \underbrace{\frac{h}{3}(f_{i-1} + 4f_i + f_{i+1})}_{\text{シンプソン公式}} \underbrace{- \frac{h^5}{90}f_i^{(4)} - \frac{h^7}{1890}f_i^{(6)} - \cdots}_{\text{誤差}(O(h^5))}$$

1区間あたりの誤差が約$\frac{h^5}{90}f_i^{(4)}$なので，n区間ならば$(n = \frac{b-a}{2h})$，

$$E \approx \frac{nh^5}{90}f_i^{(4)} = \frac{b-a}{180}h^4 f_i^{(4)}$$

シンプソン公式の**誤差は**$O(h^4)\left(or\ O\left(\frac{1}{n^4}\right)\right)$

シンプソン公式

ニュートン･コーツ公式の精度(誤差)

シンプソンの3/8公式での誤差も同様に求めて，ここまでの誤差を整理

- 中点法(0次多項式)：$E \approx \frac{b-a}{12}h^2 f_i''$
- 台形公式(1次多項式)：$E \approx \frac{b-a}{12}h^2 f_i''$
- シンプソン1/3公式(2次多項式)：$E \approx \frac{b-a}{180}h^4 f_i^{(4)}$
- シンプソン3/8公式(3次多項式)：$E \approx \frac{b-a}{80}h^4 f_i^{(4)}$

偶数次と奇数次で同じオーダの誤差で，**偶数の方が次数が低くても誤差が小さい**(f_iの微分次第なので観測値次第ではそうでないこともあるが)

シンプソン公式

シンプソン公式のコード例

(式をそのまま計算するだけなので計算手順はなしでコード例のみ)
このコードでのnはこれまでのnと違い分割数/2になっていることに注意

```
double h = (b-a)/(2*n); // 分割区間の横幅
double f;// 分割区間の縦幅
S = func(a)+func(b);

// 奇数項
for(int i = 1; i <= n; ++i){
    f = func(a+(2*i-1)*h);
    S += 4*f;
}
// 偶数項
for(int i = 1; i <= n-1; ++i){
    f = func(a+(2*i)*h);
    S += 2*f;
}
S *= h/3;
```

分割幅は$2n$なので観測点は$2n+1$必要

f_0とf_{2n}は先に計算してSに足し合わせておく

奇数項と偶数項に分けて計算している．分けないで1つの反復内でif文を使っても良い．

シンプソン公式

シンプソン公式の実行結果例

誤差評価のために，$f(x) = e^x$と設定して，区間$[a, b] = [0,1]$で数値積分(真値は$e - 1$)

$2n(or\ 3n) = 12$の場合

```
trapezoidal int f(x) = 1.719276089,  error = 0.0009942609873    台形公式
simpson int f(x) = 1.718282288,  error = 4.599789756e-07        シンプソン1/3
simpson(3/8) int f(x) = 1.718282863,  error = 1.034098449e-06   シンプソン3/8
ground truth = 1.718281828
```

$2n(or\ 3n) = 120$の場合

```
trapezoidal int f(x) = 1.718291772,  error = 9.943749073e-06    台形公式
simpson int f(x) = 1.718281829,  error = 4.603517567e-11        シンプソン1/3
simpson(3/8) int f(x) = 1.718281829,  error = 1.035782571e-10   シンプソン3/8
ground truth = 1.718281828
```

2次多項式と3次多項式を使った場合で前者の方が精度が高いことが分かる．ただし，どちらも**台形公式に比べると精度は十分高い**

今回の講義内容

- 今日の問題
- 区分求積法と台形公式
- シンプソン公式
- **ロンバーグ法**
- ガウス型積分公式
- 重積分，モンテカルロ積分

ロンバーグ法

より高次(で偶数次)の多項式を使えば精度はより高くなりそうだが，ラグランジュ補間多項式を使うため，**ルンゲ現象**により振動して**誤差が指数関数的に大きく**なる

⇒ **1次多項式(台形公式)**のままで**精度を上げられる?**

⇩

刻み幅を1/2したときの**台形公式で計算した結果**を**リチャードソンの補外で修正**することで精度を上げる

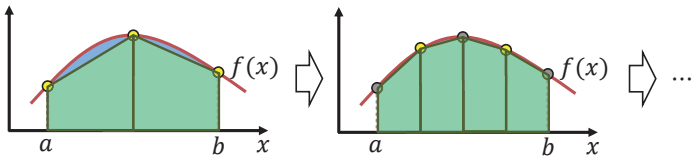
ロンバーグ法

ロンバーグ法

最初からn分割するのではなく，徐々に分割数を増やしていくことを考えてみよう．

分割数$n = 2^k$として，kを0から1ずつ増やしていく
⇒ $n = 1, 2, 4, 8, 16, \ldots$と分割数を倍々にしていく

このときの刻み幅は $h_k = \frac{b-a}{2^k}$ となる



2分法の時と同じく最初からn分割した場合と$f(x_i)$の計算回数は変わらないことに注意

ロンバーグ法

分割数を2^kで増やしたときの**台形公式による数値積分値I_kの変化**を検証してみよう

$k = 0$のとき $(n = 1)$

$$h_0 = b - a, \qquad I_0 = \frac{h_0}{2}\{f(a) + f(b)\}$$

$k = 1$のとき $(n = 2)$

$$h_1 = \frac{b-a}{2} = \frac{h_0}{2}, \quad I_1 = \frac{h_1}{2}\{f(a) + 2f(a + h_1) + f(b)\}$$
$$= \frac{h_0}{4}\{f(a) + f(b)\} + h_1 f(a + h_1)$$
$$= \frac{I_0}{2} + h_1 f(a + h_1)$$

ロンバーグ法

$k = 2$のとき $(n = 4)$

$$h_2 = \frac{b-a}{4} = \frac{h_1}{2} = \frac{h_0}{4}, \quad I_2 = \frac{h_2}{2}\left\{f(a) + 2\sum_{j=1}^{3} f(a + jh_2) + f(b)\right\}$$
$$= \frac{I_1}{2} + h_2\{f(a + h_2) + f(a + 3h_2)\}$$

- 台形公式では$n \to \infty$で真値Iになることを考えると，$I_0 \to I_1 \to I_2 \to \cdots$と**徐々に真値に近づいている**はず
- I_1はI_0を含み，I_2は**I_0とI_1を含んだ形**になっている

⇩

I_kとI_{k+1}から**より真値に近い解を計算できる**のでは?

ロンバーグ法

I_kとI_{k+1}の**誤差を比べてみよう!** (真値をIとする)

I_kの誤差： $E_k = |I_k - I| \approx \frac{b-a}{12} h_k^2 f_i''$

I_{k+1}の誤差： $E_{k+1} = |I_{k+1} - I| \approx \frac{b-a}{12} h_{k+1}^2 f_i'' = \frac{b-a}{12} \frac{h_k^2}{4} f_i''$

$I_k - I > 0$かつ$I_{k+1} - I > 0$のとき，上の2式から$\frac{b-a}{12} h_k^2 f_i''$を消すと

$$I_k - I \approx 4(I_{k+1} - I)$$

⇩ 真値Iについて解く

$$I \approx \frac{4I_{k+1} - I_k}{3}$$

⇨

- **I_kとI_{k+1}から真値により近い修正された積分値を求めることができる**
  (誤差の時点で近似値($O(h^4)$以降の項を無視している)ので真値になるわけではない)
- $O(h^2)$の誤差項を消去することになるので，**精度は$O(h^4)$まで向上**している

ロンバーグ法

前ページの青枠の式は実際**何を計算しているのか**?

刻み幅の2乗h^2と積分値Iをそれぞれ軸とした2次元座標系(h^2, I)上の2点$(h_{k+1}^2, I_{k+1}), (h_k^2, I_k)$を考える

$h^2 \to 0$ $(n \to \infty)$ としたときに真値Iが得られる

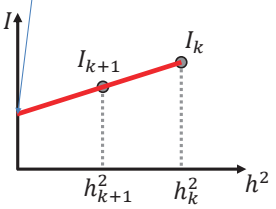
⇒ 2点を結ぶ直線と縦軸との交点が真値?

$$g(h^2) = \frac{I_k - I_{k+1}}{h_k^2 - h_{k+1}^2}(h^2 - h_{k+1}^2) + I_{k+1}$$

(前回説明した線形補間式)

⇩ 整理すると

$$g(h^2) = \frac{I_k(h^2 - h_{k+1}^2) - I_{k+1}(h^2 - h_k^2)}{h_k^2 - h_{k+1}^2}$$



ロンバーグ法

$h^2 \to 0$で真値になるとすると

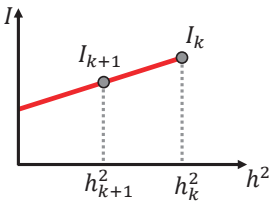
$$I \approx g(0) = \frac{I_k h_{k+1}^2 - I_{k+1} h_k^2}{h_k^2 - h_{k+1}^2} = \frac{4I_{k+1} - I_k}{3}$$

⇨ 前々ページの青枠の式が出てきた!

$(h_{k+1})^2 = \left(\frac{h_k}{2}\right)^2$を代入して整理

I_kとI_{k+1}による修正式は2点$(h_{k+1}^2, I_{k+1}), (h_k^2, I_k)$の**補外**をしている
⇒ **リチャードソン(Richardson)の補外***

*ロンバーグ法はリチャードソンの補外でロンバーグ数列
$1, 2, 4, 8, \ldots, 2^k, \ldots$を用いたものになる



ロンバーグ法

補外を用いたロンバーグ法の漸化式(ぜんかしき)と誤差

更新した積分値を見分けるためにI_kを$I_{k,m}$として，**台形公式で求めた$I_{k,0}$から$I_{k,1} \to I_{k,2} \to \cdots$と改良**していくとする

先ほどの式を書き換えると： $I_{k+1,1} = \frac{4I_{k+1,0} - I_{k,0}}{3}$

$m-1 \to m$の式にすると： $I_{k+1,m} = \frac{4^m I_{k+1,m-1} - I_{k,m-1}}{4^m - 1}$

(2点$(h_{k+m}^2, I_{k+1,m-1}), (h_k^2, I_{k,m-1})$の補外として求められる)

ロンバーグ法の漸化式

1回の改良(補外)で精度が$O(h^2) \to O(h^4)$と向上する
⇒ m回の改良(補外)で**精度は$O(h^2) \to O(h^{2m+2})$と向上**

ロンバーグ法

ロンバーグ法の具体的な手順

1. 台形公式で$I_{0,0}$を計算
2. 台形公式で$I_{1,0}$を計算
3. $I_{0,0}$と$I_{1,0}$から漸化式で$I_{1,1}$を計算
4. 台形公式で$I_{2,0}$を計算
5. $I_{1,0}$と$I_{2,0}$から漸化式で$I_{2,1}$を計算
6. $I_{1,1}$と$I_{2,1}$から漸化式で$I_{2,2}$を計算

⋮

台形公式で$I_{k,0}$を計算

⇩

漸化式: $I_{k+1,m} = \frac{4^m I_{k+1,m-1} - I_{k,m-1}}{4^m - 1}$
で$I_{k,1} \sim I_{k,k}$を計算$(m = 1 \sim k)$

をk = 0から$I_{k,m}$が許容誤差以内になるまで繰り返し

k (h_k)	$m=0$	$m=1$	$m=2$	$m=3$	$m=4$	…
0 (h_0)	$I_{0,0}$					
1 ($h_0/2$)	$I_{1,0}$	$I_{1,1}$				
2 ($h_0/4$)	$I_{2,0}$	$I_{2,1}$	$I_{2,2}$			
3 ($h_0/8$)	$I_{3,0}$	$I_{3,1}$	$I_{3,2}$	$I_{3,3}$		
4 ($h_0/16$)	$I_{4,0}$	$I_{4,1}$	$I_{4,2}$	$I_{4,3}$	$I_{4,4}$	
⋮	⋮	⋮	⋮	⋮	⋮	⋱

ロンバーグ法

ロンバーグ法の計算手順

1. 初期範囲$h = b - a$，積分値$I_{0,0}$を計算
2. 以下の手順を収束するまで繰り返す$(k = 1, 2, \ldots)$
   a. 分割幅を1/2にする：$h \leftarrow h/2$
   b. 台形公式で$I_{k,0}$を計算
   c. $|I_{k,0} - I_{k-1,0}| < \varepsilon$なら処理終了
   d. 以下の手順を$m = 1, 2, \ldots, k$で繰り返す(漸化式の計算)
      ① 改良された積分値$I_{k,m}$を計算:$I_{k,m} = \frac{4^m I_{k-1,m-1} - I_{k,m-1}}{4^m - 1}$
      ② $|I_{k,m} - I_{k,m-1}| < \varepsilon$なら処理終了

⇒ 処理終了時点の$I_{k,m}$が解となる

台形公式での$I_{k,0}$の計算では新しい分割点での$f(x_i)$を計算するだけなので，計算量は2^k分割時の台形公式と大きくは変わらない

ロンバーグ法

ロンバーグ法のコード例

ここまでのコードと異なり分割数nを与えるのではなく，
最大反復数k_{max}と許容誤差εを指定

```
double h = b-a; // 分割幅(初期分割幅はn=1のもの)
vector<double> I(k_max+1, 0.0);
I[0] = (h/2)*(func(a)+func(b)); // I_0,0の計算
int k, n = 1, l;
for(k = 1; k <= k_max; ++k){
    h = h/2; n *= 2; // 分割幅を1/2にしていく
    // 台形公式でI_k,0を計算
    I[k] = trapezoidal_integration(func, a, b, n);
    // 収束判定
    if((e = fabs(I[k]-I[k-1])) < eps) return I[k];

    // 漸化式の計算
    int m, m4 = 4; // 4^m
    for(m = 1; m <= k; ++m){
        int i = k-m; // I_k,mを格納する配列上の位置
        I[i] = (m4*I[i+1]-I[i])/(m4-1); // I_k,mを計算
        m4 *= 4;
        // 収束判定2
        if(i >= 1 && (e = fabs(I[i]-I[i-1])) < eps) return I[i];
    }
}
```

初期刻み幅h_0と$I_{0,0}$の計算

台形公式による$I_{k,0}$の計算: 計算済みのf_iを再利用するようにすべきだけど，ここではコードの簡易化のために毎回すべて計算している

漸化式による補外計算: 計算した結果はmが大きい方から配列Iに$I[0], I[1], \ldots$の順番に格納している(これにより$k_{max}+1$の大きさの配列があれば良いことになる)

ロンバーグ表で一つ上の値と比較して収束判定

ロンバーグ法

ロンバーグ法の実行結果例

誤差評価のために，$f(x) = e^x$と設定して，区間$[a, b] = [0,1]$で数値積分

ロンバーグ法で$k_{max} = 5$ $(n_{max} = 2^5 = 32)$，許容誤差$\varepsilon = 10^{-6}$と設定

```
1.859140914
1.753931092   1.718861152
1.727221905   1.718318842   1.718282688
1.720518592   1.718284155   1.718281842
romberg int f(x) = 1.718281842,  error = 1.375939518e-08
n = 8, eps = 8.457063168e-07
ground truth = 1.718281828
```

実際には$k = 3$ $(n = 8)$で処理終了

台形公式，シンプソン公式で$n = 8$とした場合の結果

```
trapezoidal int f(x) = 1.720518592,  error = 0.002236763705
simpson int f(x) = 1.718284155,  error = 2.326240852e-06
```

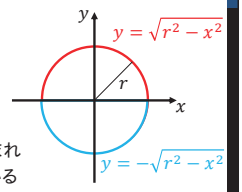
ロンバーグ法を使えば，同等の計算時間で**精度を大幅に改善**できる!

ロンバーグ法

円の面積

数値積分で円の面積*を計算してみる

$$S = \int_{-r}^{r} \sqrt{r^2 - x^2}\, dx - \int_{-r}^{r} -\sqrt{r^2 - x^2}\, dx$$



$S = 2\int_{-r}^{r} \sqrt{r^2 - x^2}\, dx$でも良いがここでは2関数に挟まれた領域の面積を求める例とするために上記の式を用いる

台形公式，シンプソン公式で$n = 32$とした場合の結果$(r = 1)$

```
trapezoidal int f(x) =  3.123253038,  error = 0.01833961576
simpson int f(x) = 3.139052218,  error = 0.002540435696
ground truth = 3.141592654
```

ロンバーグ法で$k_{max} = 5$ $(n_{max} = 2^5 = 32)$，許容誤差$\varepsilon = 10^{-6}$と設定$(r = 1)$

```
romberg int f(x) = 3.135517095,  error = 0.006075558516
ground truth = 3.141592654
```

$k = k_{max}$でも許容誤差内に収まらなかった

関数の形によってはロンバーグ法でも精度が上がらないこともある

*単に円周率の計算をしたいなら平方根計算がない$\int_{-1}^{1} \frac{1}{1+x^2} dx = \frac{\pi}{2}$などを使った方が良い

今回の講義内容

- 今日の問題
- 区分求積法と台形公式
- シンプソン公式
- ロンバーグ法
- **ガウス型積分公式**
- 重積分，モンテカルロ積分

ガウス型積分公式

ニュートン・コーツ型の積分公式は**関数$f(x)$の形に精度が大きく依存**してしまう．

例えば，台形公式の誤差$\frac{b-a}{12} h^2 f_i''$はf''を含み，**関数の変化が大きいところでは誤差が大きく**なると予想できる

⇒

関数の変化が大きいところだけ**刻み幅hを小さく**したら**精度が上がるのでは?**

ガウス型積分公式

ガウス型積分公式

ガウス型積分公式を求めるための準備

積分範囲を$[a, b]$から$[-1, 1]$に変換する

$$I = \int_a^b g(z) dz \quad \Longrightarrow \quad I = \frac{b-a}{2} \int_{-1}^{1} f(x) dx$$

$z = \frac{b-a}{2} x + \frac{a+b}{2}$として変数変換

以降，この部分の積分について考えていく

$$I = \int_{-1}^{1} f(x) dx$$

最終的に求めた積分公式から以下のように計算することで範囲$[a, b]$の積分値を得る

$$I = \frac{b-a}{2} \int_{-1}^{1} f\left(\frac{b-a}{2} x + \frac{a+b}{2}\right) dx$$

ガウス型積分公式

$[-1,1]$の範囲内にn個の計算点(分点)x_iを配置したとする
(x_iは等間隔でなくても良いことに注意)

$f(x_i)$に重みw_iを掛けて足し合わせたもので**積分を近似**してみよう!

$$I=\int_{-1}^{1} f(x)dx \approx \sum_{i=0}^{n-1} w_i\, f(x_i)$$ ⇨ 誤差: $E=\sum_{i=0}^{n-1} w_i\, f(x_i)-\int_{-1}^{1} f(x)dx$

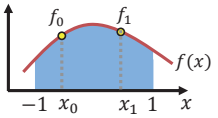
⇒ 分点x_iと重みw_iをどうやって決めよう?

[条件と仮定]
- $f(x)$を**多項式と仮定**しよう(多項式で近似する)
  例) $f(x)=x^2$や$f(x)=x^3$など
- 少なくとも分点x_iでは**誤差Eが0になる**ようにしよう

ガウス型積分公式

分点の数が2の場合で考えてみる($n=2$)

$$I \approx \sum_{i=0}^{1} w_i f(x_i) = w_0 f_0 + w_1 f_1$$



未知変数はx_0, x_1, w_0, w_1の4つ. **$f(x)$の形を具体的に考えながら**これらを求めていく

$f(x)=x^0=1$の場合 (0次多項式) ⇨ $f_0=f_1=1$

x_0, x_1で$E=0$とすると: $w_0+w_1=\int_{-1}^{1} 1\,dx=2$

$f(x)=x^1=x$の場合 (1次多項式) ⇨ $f_0=x_0, f_1=x_1$

x_0, x_1で$E=0$とすると: $w_0x_0+w_1x_1=\int_{-1}^{1} x\,dx=0$

ガウス型積分公式

$f(x)=x^2$の場合 (2次多項式) ⇨ $f_0=x_0^2, f_1=x_1^2$

x_0, x_1で$E=0$とすると: $w_0x_0^2+w_1x_1^2=\int_{-1}^{1} x^2\,dx=\frac{2}{3}$

$f(x)=x^3$の場合 (3次多項式) ⇨ $f_0=x_0^3, f_1=x_1^3$

x_0, x_1で$E=0$とすると: $w_0x_0^3+w_1x_1^3=\int_{-1}^{1} x^3\,dx=0$

青枠の式を**連立方程式として解く**とx_0, x_1, w_0, w_1が得られる($x_0<x_1$)

$$\begin{cases} w_0+w_1=2 \\ w_0x_0+w_1x_1=0 \\ w_0x_0^2+w_1x_1^2=2/3 \\ w_0x_0^3+w_1x_1^3=0 \end{cases}$$ ⇨

$$x_0=-\sqrt{\frac{1}{3}},\ x_1=\sqrt{\frac{1}{3}}$$
$$w_0=w_1=1$$

という解が得られる

ガウス型積分公式

分点n個に対して**$2n$個の式があれば, x_i, w_i $(i=0,\dots,n-1)$を計算できる!** (多項式の最高次数は$2n-1$次)

nを増やしたときの分点と重み

$n=2$: $x_i=\left(-\sqrt{\frac{1}{3}}, \sqrt{\frac{1}{3}}\right)$, $w_i=(1,1)$

$n=3$: $x_i=\left(-\sqrt{\frac{3}{5}},\ 0,\ \sqrt{\frac{3}{5}}\right)$, $w_i=(\frac{5}{9},\ \frac{8}{9},\ \frac{5}{9})$

$n=4$: $x_i=\left(-\sqrt{\frac{3+2\sqrt{\frac{6}{5}}}{7}}, -\sqrt{\frac{3-2\sqrt{\frac{6}{5}}}{7}}, \sqrt{\frac{3-2\sqrt{\frac{6}{5}}}{7}}, \sqrt{\frac{3+2\sqrt{\frac{6}{5}}}{7}}\right)$

$w_i=(\frac{18-\sqrt{30}}{36},\ \frac{18+\sqrt{30}}{36},\ \frac{18+\sqrt{30}}{36},\ \frac{18-\sqrt{30}}{36})$

ガウス型積分公式

求めたx_iは**ルジャンドル多項式の根**になっている

⇩

ガウス・ルジャンドル公式

重みは $w_i=\frac{2}{nP_{n-1}(x_i)P_n'(x_i)}$ で計算できる

x_iがその他の多項式の根になる例:

$\int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}}dx \approx \sum_{i=0}^{n-1} w_i f(x_i)$ と近似 ⇒ **チェビシェフ・ガウス公式**

$\int_{-1}^{1} e^{-x^2} f(x)dx \approx \sum_{i=0}^{n-1} w_i f(x_i)$ と近似 ⇒ **ガウス・エルミート公式**

$\int_{-1}^{1} e^{-x} f(x)dx \approx \sum_{i=0}^{n-1} w_i f(x_i)$ と近似 ⇒ **ガウス・ラゲール公式**

ガウス型積分公式

ガウス・ルジャンドル公式($n=3$)のコード例

(式をそのまま計算するだけなので手順説明はなし)

```
// 分点と重みの計算(n=3)
double x[3], w[3];
x[0] = -sqrt(3.0/5.0);
x[1] = 0;
x[2] = -x[0];
w[0] = w[2] = 5.0/9.0;
w[1] = 8.0/9.0;

// Σ wi f(xi)の計算([a,b]と[-1,1]の変換付き)
S = 0.0;
for(int i = 0; i < 3; ++i){
    S += w[i]*func((b-a)*x[i]/2+(a+b)/2);
}
S *= (b-a)/2;
```

分点x_iと重みw_iの計算:
$n=3$なのでそれぞれ3つ

$\sum_{i=0}^{n-1} w_i f(x_i)$の計算.
範囲$[-1,1]$と$[a,b]$の
変換も含まれているので
注意

ガウス型積分公式

ガウス・ルジャンドル公式の実行結果例

誤差評価のために, $f(x) = e^x$と設定して*, 区間$[a, b] = [0,1]$で数値積分

ガウス・ルジャンドル公式による積分値

```
gauss2 int f(x) = 2.306612746,  error = 0.5883309177
gauss3 int f(x) = 1.718281004,  error = 8.240865232e-07
gauss4 int f(x) = 1.718281828,  error = 9.32967037e-10
ground truth = 1.718281828
```

上から$n = 2, 3, 4$の結果

ロンバーグ法で$k_{max} = 5$ ($n_{max} = 2^5 = 32$), 許容誤差$\varepsilon = 10^{-6}$と設定

```
romberg int f(x) = 1.718281842,  error = 1.375939518e-08
n = 8, eps = 8.457063168e-07
```

台形公式,シンプソン公式で$n = 8$とした場合の結果

```
trapezoidal int f(x) = 1.720518592,  error = 0.002236763705
simpson int f(x) = 1.718284155,  error = 2.326240852e-06
```

$n = 3$の3分割でも他の手法の8分割と**同等かそれ以上の精度**が得られている

*数値計算上は関数を多項式で近似して求めていることになる

今回の講義内容

- 今日の問題
- 区分求積法と台形公式
- シンプソン公式
- ロンバーグ法
- ガウス型積分公式
- **重積分, モンテカルロ積分**

重積分

体積を求める時には**重積分**が必要となる

$$I = \int_a^b \int_{y_1(x)}^{y_2(x)} f(x,y)dy\,dx$$

重積分の場合でも, $F(x) = \int_{y_1(x)}^{y_2(x)} f(x,y)dy$ とすれば

$$I = \int_a^b F(x)\,dx$$

となり, 1変数の積分としてこれまでに説明した方法(台形公式やシンプソン公式など)が使える

重積分

台形公式の重積分版

$$I \approx \sum_{i=0}^{n-1} \frac{h_1}{2}(F_i + F_{i+1}) = \frac{h_1}{2}(F_0 + 2\sum_{i=1}^{n-1} F_i + F_n)$$

$F(x) = \int_{y_1(x)}^{y_2(x)} f(x,y)dy$なので同様に台形公式から:

$$F_i = F(x_i) = \frac{h_2}{2}\left(f(x_i, y_0) + 2\sum_{j=1}^{m-1} f(x_i, y_j) + f(x_i, y_n)\right)$$

ここで, x方向の分割数をn, 刻み幅$h_1 = \frac{b-a}{n}$

y方向の分割数をm, 刻み幅$h_2 = \frac{y_2(x_i)-y_1(x_i)}{n}$ としている.

シンプソン公式を用いる場合も同じ. また, 3重以上の積分でも同様にすれば近似解が得られる

重積分

台形公式を用いた重積分のコード例

(式をそのまま計算するだけなので手順説明はなし)

```
double h1 = (b-a)/n; // x方向刻み幅
double S = 0.0;
for(int i = 0; i <= n; ++i){
    double xi = a+i*h1;
    double h2 = (y2f(xi)-y1f(xi))/m; // y方向刻み幅
    double y1 = y1f(xi);

    // 台形公式によるy方向積分(Fiの計算)
    double f1, f2;
    f2 = func(xi, y1);
    double Fi = 0.0;
    for(int j = 0; j < n; ++j){
        f1 = f2;
        f2 = func(xi, y1+(j+1)*h2);
        Fi += (f1+f2)*h2/2;
    }

    if(i == 0 || i == n) S += Fi;
    else S += 2*Fi;
}
S *= h1/2;
```

x方向の積分のためのループ

y方向の積分F_iの計算. 台形公式による通常の積分計算とほぼ同じ($f(x)$が$f(x,y)$になっていることだけが違い)

重積分

台形公式を用いた重積分の実行結果例

半径rの球の体積を重積分で表すと:

$$V = 2\int_{-r}^{r} \int_{-\sqrt{r^2-x^2}}^{\sqrt{r^2-x^2}} \sqrt{r^2 - x^2 - y^2}\,dydx$$

$f(x,y) = \sqrt{r^2 - x^2 - y^2}$, $y_1(x) = -\sqrt{r^2 - x^2}$, $y_2(x) = \sqrt{r^2 - x^2}$
として台形公式で重積分を計算

$r = 1, n = 20$とした場合の結果

```
trapezoidal int2 f(x) = 4.129009375,  error = 0.05978082995
ground truth = 4.188790205
```

$r = 1, n = 100$とした場合の結果

```
trapezoidal int2 f(x) = 4.183939579,  error = 0.004850625572
ground truth = 4.188790205
```

真値($V = \frac{4\pi r^3}{3}$)に近い値がnを大きくすれば求められている

モンテカルロ積分

積分範囲の形状が複雑になると数値積分としての**計算手順も複雑**になる.

⇒ 範囲形状の**複雑さに依存しない方法**はないのか?

積分範囲V内で一様分布する**確率密度関数**$p(\boldsymbol{x})$を考えてみよう

$$p(\boldsymbol{x}) = \begin{cases} \frac{1}{V} & \boldsymbol{x} \in V \\ 0 & otherwise \end{cases}$$

Vは1次元なら$[a,b]$のような範囲となる. 積分の式を$p(\boldsymbol{x})$を含む形に変形する.

$$\int_V f(\boldsymbol{x})d\boldsymbol{x} = \int_V \frac{f(\boldsymbol{x})}{p(\boldsymbol{x})} p(\boldsymbol{x})d\boldsymbol{x} = V \underbrace{\int_V f(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}}_{\text{期待値}}$$

モンテカルロ積分

期待値はサンプル点(ランダムな点)の数nが十分大きくなると, **平均値に近づいていく**ので:

$$\int_V f(\boldsymbol{x})d\boldsymbol{x} \approx \frac{V}{n}\sum_{i=0}^{n-1} f(\boldsymbol{x}_i)$$

モンテカルロ積分

このときの誤差は $V\underbrace{\sqrt{\frac{\langle f^2\rangle - \langle f\rangle^2}{n}}}_{\text{標準偏差}}$ となる ($\langle f^2\rangle$はf^2の平均値, $\langle f\rangle$はfの平均値)

例) 1次元で積分範囲が$[a,b]$なら$V = b-a$となり,

$$\int_a^b f(x)dx \approx \frac{b-a}{n}\sum_{i=0}^{n-1} f(x_i)$$

モンテカルロ積分

モンテカルロ積分の手順

1. サンプル点数nを設定し, $f(x_i)$の合計値を格納する変数Sとカウンタcを初期化:$S = 0, c = 0$
2. 以下の処理をn回繰り返し($i = 0,1,\dots n-1$)
   a. ランダムな点$\boldsymbol{x}_i$を乱数で求める
   b. $\boldsymbol{x}_i \in V$ならば, $S = S + f(\boldsymbol{x}_i), c = c + 1$
3. 積分値を計算 : $I = V\frac{S}{n}$

ここでのVは, 1次元なら$[a,b]$のような範囲, 2次元なら各辺が軸と平行な矩形, 3次元なら直方体を用いるのが一般的

モンテカルロ積分

モンテカルロ積分の例 : 円の面積を求める

円の面積を求める問題を2重積分で表すと:

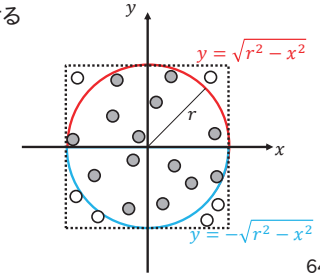
$$S = \int_{-r}^{r}\int_{-\sqrt{r^2-x^2}}^{\sqrt{r^2-x^2}} 1\, dy\, dx$$

となり, $f(\boldsymbol{x}) = 1$なので単純に範囲内の点を数え, それを総サンプリング数nで割り, $V = 4r^2$を掛ける

$$S \approx 4r^2\frac{n_{in}}{n}$$

ここでn_{in}は以下の条件を満たしたサンプリング点の数

$$x^2 + y^2 \le r^2$$



モンテカルロ積分

モンテカルロ積分の例 : 円の面積を求める

$r = 1$としてサンプリング数nを変えて実行した結果

```
n=        10 : area of circle = 3.6,       error = 0.4584073464
n=       100 : area of circle = 3.28,      error = 0.1384073464
n=      1000 : area of circle = 3.168,     error = 0.02640734641
n=     10000 : area of circle = 3.1348,    error = 0.00679265359
n=    100000 : area of circle = 3.1414,    error = 0.0001926535898
n=   1000000 : area of circle = 3.140884,  error = 0.0007086535898
n=  10000000 : area of circle = 3.1415684, error = 2.425358979e-05
ground truth = 3.141592654
```

nが大きくなれば誤差は小さくなっているが, **乱数**を使っているのである程度のぶれはある

10回ずつ実行して誤差の平均を取った結果

```
n=        10 : avg. error = 0.5476613408
n=       100 : avg. error = 0.1956508459
n=      1000 : avg. error = 0.07714285714
n=     10000 : avg. error = 0.01799580205
n=    100000 : avg. error = 0.007057142857
n=   1000000 : avg. error = 0.002798845934
n=  10000000 : avg. error = 0.000332883663
```

モンテカルロ積分は**多次元**で**複雑な領域の積分**を行いたいときに特に有効

講義内容のまとめ

- 今日の問題 : $\int_a^b f(x)dx$
- 等間隔区分で積分を近似する方法
  - 区分求積法, 台形公式, シンプソン公式
- 区分の取り方を工夫して精度を上げる方法
  - ロンバーグ法, ガウス型積分公式
- 重積分の計算方法
  - 台形公式の拡張, 乱数を用いたモンテカルロ積分

Appendix

(以降のページは補足資料です)

ガウス型積分公式の分点・重み計算

$$\begin{cases} w_0 + w_1 = 2 & \cdots① \\ w_0 x_0 + w_1 x_1 = 0 & \cdots② \\ w_0 x_0^2 + w_1 x_1^2 = 2/3 & \cdots③ \\ w_0 x_0^3 + w_1 x_1^3 = 0 & \cdots④ \end{cases}$$

①式から

$$w_1 = 2 - w_0$$

②式に代入

$$w_0 x_0 + (2 - w_0) x_1 = 0$$

$$w_0 = \frac{2x_1}{x_1 - x_0} \quad w_1 = -\frac{2x_0}{x_1 - x_0} \quad \cdots⑤$$

④式に⑤式を代入して，x_0, x_1だけの式にする

$$\frac{2x_1}{x_1 - x_0} x_0^3 - \frac{2x_0}{x_1 - x_0} x_1^3 = 0 \Rightarrow x_0^2 - x_1^2 = 0 \Rightarrow x_0 = \pm x_1$$

$x_0 < x_1$なので$x_0 = -x_1$

③式に⑤式を代入

$$\frac{2x_1 x_0^2}{x_1 - x_0} - \frac{2x_0 x_1^2}{x_1 - x_0} = \frac{2}{3} \Rightarrow$$

$x_0 = -x_1$を代入して整理

$x_0 < x_1$より

$$x_0 = -\sqrt{\frac{1}{3}},\ x_1 = \sqrt{\frac{1}{3}}$$

⑤式に代入して

$$w_0 = w_1 = 1$$

多項式色々

- **ルジャンドル多項式**$(x \in [-1,1])$

$$\{P_k\} = \left\{1, x, \frac{1}{2}(3x^2 - 1), \frac{1}{2}(5x^3 - 3x), \ldots, \frac{1}{2^n n!}\frac{d^n}{dx^n}(x^2 - 1)^n\right\} \quad (n = 0,1,\ldots)$$

- **第一種チェビシェフ多項式**$(x \in [-1,1])$

$$\{P_k\} = \{1, x, 2x^2 - 1, \ldots, \cos(nt)\} \quad (x = \cos(t), n = 0,1,\ldots)$$

- **エルミート多項式**$(x \in [-\infty, \infty])$

$$\{P_k\} = \left\{1, 2x, 4x^2 - 2, 8x^3 - 12x, \ldots, (-1)^n e^{x^2}\frac{d^n}{dx^n}e^{-x^2}\right\} \quad (n = 0,1,\ldots)$$

- **ラゲール多項式**$(x \in [0, \infty])$

$$\{P_k\} = \left\{1, -x + 1, x^2 - 4x + 2, \ldots, \frac{e^x}{n!}\frac{d^n}{dx^n}(x^n e^{-x})\right\} \quad (n = 0,1,\ldots)$$

疑似乱数いろいろ

- **線形合同法**(LCG)：$R_{n+1} = (A \times R_n + B) \bmod M$で計算される乱数．簡易的にできるので**C言語のデフォルト乱数生成法**として有名．下位ビットのランダム性が低く，**規則性が出やすい**などの欠点がある．周期:2^{32}(ただしパラメータ設定次第)
- **Xorshift**：2003年に提案された疑似乱数生成法．排他的論理和(XOR)とビットシフト演算のみなので**非常に高速**で，乱数生成法のテストである**Diehardテストもパス**している．周期:$2^{32} - 1$
- **メルセンヌ・ツイスタ** (MT:Mersenne twister)：1996年に発表されたメルセンヌ素数を用いた疑似乱数生成法．1周期で623次元空間に均等分布することが証明されている．生成速度も線形合同法よりも高速．C++11以上やPythonなど多くの言語で採用されている．周期:$2^{19937} - 1$