

# 電気 303 / 電情 303 数值解析 (10)

## 数值積分

## 今回の参考文献

- 杉原, 室田, 数値計算法の数理, 岩波書店, 1994
- 斎藤, 数値解析入門, 東京大学出版会, 2012

## 全般的注意

- 教科書を購入している者は、各自白で教科書のプログラムを試してみること (著者のホームページよりダウンロードできる).

# 数値積分の必要性

- 微分と積分は微分積分学の両輪.
- 微分積分学は工学の至る所で利用される.
- 関数が既知の関数の合成関数で表現されているとき, その関数の微分も既知の関数 (とその微分) の合成関数で表現される. この意味で, 微分は代数的な演算によって処理できる.

- 一方、関数が既知の関数の合成関数で表現されていても、その不定積分が既知の関数 (とその積分など) の合成関数で表現されることは稀である.
- したがって、工学などの応用で積分を使うときには、積分の数値的な評価 (数値積分) が本質的に重要になる.

- 応用上よく用いられる積分には, 実軸上で定義された実関数の積分, 多次元空間における積分, 複素積分, 線積分, 面積分などがある.
- この講義では実軸上で定義された実関数の積分のみについて議論する.

## Scilab における数値積分の概要

- 数値積分には、いろいろな方法があるが、Scilab はそれほど多くの数値積分法をサポートしているわけではない。
- Scilab がサポートしている数値積分法を以下に示す。各方法がどのようなものかは後述。

## § 実軸上で定義された実関数

<code>intg</code>	21 点 Gauss-Kronrod 則による数値積分
<code>inttrap</code>	複合台形則による数値積分
<code>intsplin</code>	スプライン補間による数値積分
<code>integrate</code>	複数の終端について積分を評価, 内部で <code>intg</code> を呼んでいる



## § 複素積分

`intl`                    複素積分 (積分路は円弧)

`intc`                    複素積分 (積分路は直線)

いずれも、内部では複素積分を実パラメータに関する積分に変換して `intg` を呼んでいる。数値解析の観点からは実積分と同じなので、この講義では取り扱わない。

## § 多次元空間における積分

`int2d`                    2次元空間における積分

`int3d`                    3次元空間における積分

この講義では関数名を紹介するだけに留める.

## 実軸上の数値積分

- 以下では, 区間  $[a, b]$  (ただし  $a < b$ ) における実関数  $f$  の積分, すなわち

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

を数値的に求めることを考える.  $f$  は Riemann 積分可能であると仮定する.

- 積分のもっとも素朴な計算法は、区間  $[a, b]$  に有限個の点  $a = x_0 < x_1 < \cdots < x_N = b$  (これを分点とよぶ) と、区間  $[x_k, x_{k+1}]$  ( $0 \leq k \leq N$ ) の点  $\xi_k$  (これを代表点という) を取り、

$$\int_a^b f(x) dx \simeq \sum_{k=0}^{N-1} f(\xi_k)(x_{k+1} - x_k)$$

によって近似値を求める、というものである。

- 先に述べた方法は Riemann 和の定義と同じ (杉浦, 解析入門, 東京大学出版会, 1980 などを参照).  $\xi_k$  は区間  $[x_k, x_{k+1}]$  のどの点でもよい.

- $\xi_k$  を区間  $[x_k, x_{k+1}]$  の中点に取る方法を **中点則**あるいは**中点公式**という

- **公式**という言葉は**計算の方法**という意味で用いられているが、ここで言う計算は近似計算である。したがって、公式を適用することによって厳密に正しい値が得られるとは限らない。
- 以下、公式という言葉が頻出するが、数学の公式とはニュアンスが異なるので混乱しないよう注意すること。

- $f$  が Riemann 積分可能であれば, 分割を細かくすると上記の数値積分値は真値に近づく.



- とはいっても、コンピュータでは無限回の計算はできないので、有限個の分点で良い近似値を得ることが望ましい。この観点から言うと、上記より良い方法がある (後述)。

- 数値積分を評価する際, 分点が計算中に
  - ▷ 調整できない場合
  - ▷ 調整できる場合

がある. 前者に対応するのは, 関数値を別の測定実験で求めており, 追加実験ができないような場合.

- 分点が動かさせないときには、標本点をスプライン補間してそれを積分する、という手法を用いることが一般的である。
- このようにするのは、精度が中点公式より良いことを期待してのことであるが、そうなることが理論的に保証されるわけではない。

- スプライン補間は代数的に積分できる。したがって、多項式の値と端点がわかっているならば、そこから代数演算によって数値積分を求めることができる。

- 分点が動かせるときには、数値積分の精度が十分でない場合には分点を取り直すことにすれば、より高精度の数値積分が得られる。

- この場合には, 分点で被積分関数と値が一致する Lagrange 補間多項式を作りそれを積分するという手法が取られる. この種の方法を補間型公式という.

- 補間型公式において, 補間や分点の取り方には様々なバリエーションがある.

- 区間を  $m$  等分して Lagrange 補間する方法を,  
**Newton-Cotes 公式** という.
- 区間の分点として直交多項式の零点を取る方法  
を, **Gauss 型 (積分) 公式** という.
- Newton-Cotes 公式および Gauss 型公式にも,  
色々なバリエーションがある.



- 前回までの講義で学んだ補間は, 数値積分において本質的に重要な役割を果たす.

# Newton-Cotes 公式

- 分点 (標本点) を等間隔に取った Lagrange 補間には, 区間の端の近くで補間関数ともとの関数の値が大きく異なる場合がある, という問題があった (Runge の現象).

- したがって、数値積分の精度を上げたいからといって、むやみに分点を増やすわけにはいかない。

- 実用上は、 $\int_a^b f(x)dx$  を計算する際、全区間  $[a, b]$  に後述の積分公式を適用するかわりに、 $[a, b]$  を  $a = x_0 < x_1 < \dots < x_N = b$  のように分割しておき、各区間  $[x_k, x_{k+1}]$  ( $0 \leq k \leq N - 1$ ) に積分公式を適用することが普通である。これを複合公式という。

- 以下では、複合公式を前提とし、細分後のある区間  $[x_k, x_{k+1}]$  における  $f$  における積分公式を考える。この場合、区間  $[x_k, x_{k+1}]$  の中に分点が取られることになるので、混乱しないよう注意すること。

- すでに区間  $[x_k, x_{k+1}]$  が十分細かく取られていると仮定すれば, Lagrange 補間で関数近似をする際に, 分点を多く取らなくても, 良い精度の近似が得られることが期待される.

- 実用上よく出てくるのは,  $m = 1$  (分割しない) と,  $m = 2$  (この区間を 2 等分) だけである.

- Newton-Cotes 公式で  $m = 1$ (分割なし) の場合には, Lagrange 補間多項式は,

$$(x_k, f(x_k)), \quad (x_{k+1}, f(x_{k+1}))$$

を通る直線となる. これを積分する (この直線と  $x$  軸が作る台形の面積を求める) 手法を **台形則** と呼ぶ.



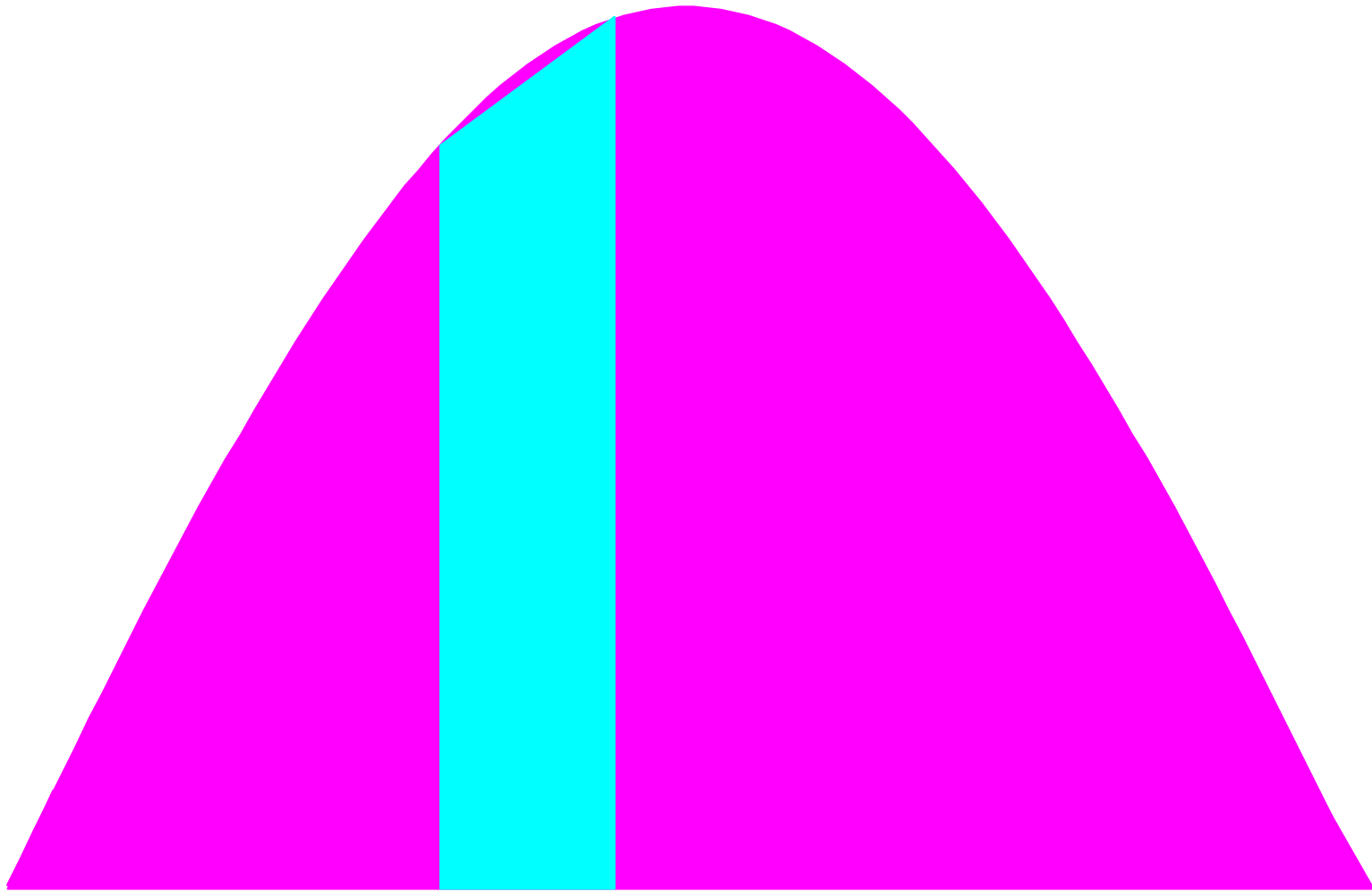
$(x_k, f(x_k)), (x_{k+1}, f(x_{k+1}))$  を通る直線は

$$p(x) = f(x_k) + \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k} (x - x_k)$$

だから、台形則は以下のように書き直される。  
これは確かに台形の面積である。

$$\int_{x_k}^{x_{k+1}} p(x) dx = \frac{f(x_{k+1}) + f(x_k)}{2} (x_{k+1} - x_k)$$

- 次ページは,  $\sin x$  の数値積分のために区間  $[1, 1.4]$  に台形則を適用した例である. 台形則によって近似された領域が水色で示されている.



- Newton-Cotes 公式で  $m = 2$  の場合には,  $z_k = (x_k + x_{k+1})/2$  とすると, Lagrange 補間多項式は,

$$(x_k, f(x_k)), (z_k, f(z_k)), (x_{k+1}, f(x_{k+1}))$$

を通る放物線となる. これを積分する手法を **Simpson 則** と呼ぶ.

2 次の Lagrange 補間多項式を積分することで (代数的にできる), Simpson 則の具体的な形が求められる.

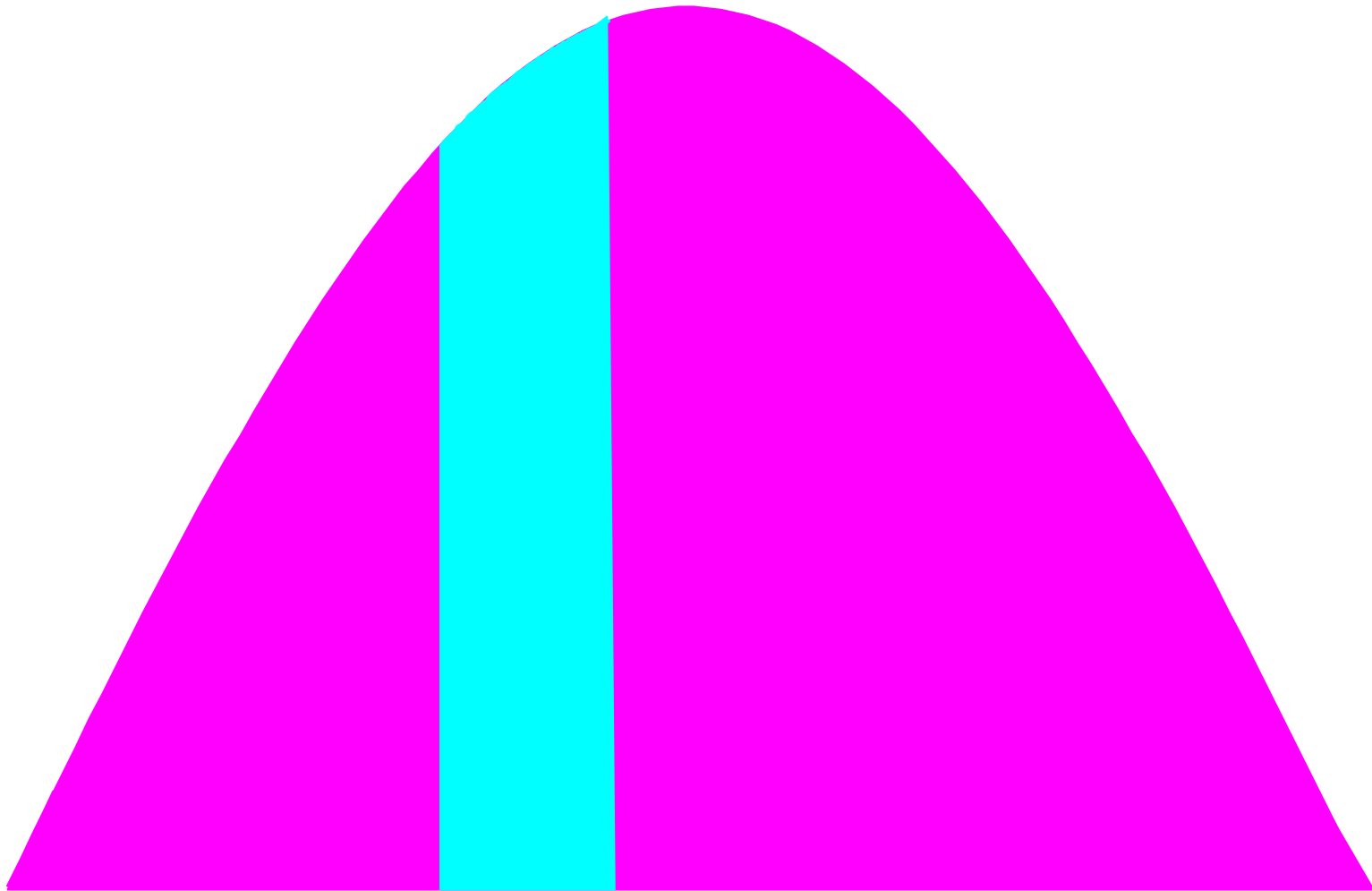
結果のみ書くと,

$$z_k = \frac{x_k + x_{k+1}}{2}$$

として, 以下のようにになる.

$$\begin{aligned} & \int_{x_k}^{x_{k+1}} f(x) dx \\ & \simeq \frac{x_{k+1} - x_k}{6} (f(x_k) + 4f(z_k) + f(x_{k+1})). \end{aligned}$$

- 次ページは、 $\sin x$  の数値積分のために区間  $[1, 1.4]$  に Simpson 則を適用した例である。Simpson 則によって近似された領域が水色で示されているが、正弦関数とほぼ重なっており、違いを目視で判別することは難しい。





- 積分公式が  $1, x, \dots, x^m$  まで正しい積分を与え、 $x^{m+1}$  に対して正しい値を与えないとき、その積分公式を  $m$  次の積分公式という。

- 台形則は 1 次, Simpson 則は 2 次の Lagrange 補間に基づいているから, 台形則は 1 次以上, Simpson 則は 2 次以上である.

- $x^2$  に対し、直接計算することにより、台形則が

$$\int_{x_k}^{x_{k+1}} x^2 dx$$

を正しく評価できないことが確認できるので、台形則は1次の積分公式である。

- 計算は略すが, Simpson 則は,  $x^3$  については正しい値を与え,  $x^4$  については正しい値を与えない. よって, Simpson 則は 3 次の積分公式である.

- 次に数値積分の誤差の上界を, 中点則と比較する. 中点則と台形則は  $f$  の 2 階微分, Simpson 則は  $f$  の 4 階微分に基づいて評価される (したがって, その階数の微分可能性を仮定しなければならない).

- 計算を略し (詳細は [斎藤], pp. 188–190), 次ページに結果のみ示す.

公式	誤差の上界
中点則	$\frac{1}{24}(x_{k+1} - x_k)^3 \ f''\ _\infty$
台形則	$\frac{1}{12}(x_{k+1} - x_k)^3 \ f''\ _\infty$
Simpson 則	$\frac{1}{2880}(x_{k+1} - x_k)^5 \ f^{(4)}\ _\infty$

ただし,  $\|g\|_\infty = \sup_{x \in [x_k, x_{k+1}]} |g(x)|$

- 各区間  $[x_k, x_{k+1}]$  ( $0 \leq k \leq N-1$ ) における  $f$  の積分の近似値を台形則によって求め、それを足し合わせることで  $\int_a^b f(x)dx$  の近似値を求める手法を、**複合台形則**という。



- 上記で、台形則のかわりに Simpson 則を用いる手法を、**複合 Simpson 則**という。
- 複合台形則、複合 Simpson 則を単に台形則、Simpson 則と呼ぶこともある。

# Gauss 型公式

- 続いて, Gauss 型公式について述べる.

- Gauss 型積分公式で取り扱われる問題は, 正の重み関数  $w(x)$  に対し,

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

を近似する問題である. ただし,

$$\int_a^b w(x)dx < \infty$$

と仮定する ( $w \equiv 1$  なら通常の積分).

- Gauss 型公式では, 必ずしも複合公式が使われるとは限らない. したがって, 以下では, 区間  $[a, b]$  に直接 Gauss 型公式を適用するという問題設定で議論を進める.

- 内積

$$(f, g)_w = \int_a^b f(x)g(x)w(x)dx$$

に関し,

$$\{\phi_k(x)\}_{k=0,1,2,\dots}$$

が直交多項式系をなし,  $\phi_k(x)$  は  $k$  次の多項式であるものとする.

- 直交多項式  $\phi_k$  は开区間  $(a, b)$  に  $k$  個の相異なる零点を持つことが示せる ([斎藤], pp. 156–157).

- ある  $n$  を固定し,  $(a, b)$  における  $\phi_{n+1}$  の零点を小さい方から並べたものを

$$x_0, \dots, x_n$$

とする. すなわち,

$$a < x_0 < \dots < x_n < b$$

である.

- $(n + 1)$  個の点  $x_0, \dots, x_n$  に関する  $f$  の  $n$  次の Lagrange 補間多項式を  $q_n$  とする.



- Gauss 型公式は,  $n$  次の Lagrange 補間多項式  $q_n(x)$  を使い, 積分の近似値を

$$\int_a^b q_n(x)w(x)dx$$

によって与える公式である.

- Lagrange 補間多項式は, 多項式

$$p_k(x) = \frac{\prod_{j \neq k} (x - x_j)}{\prod_{j \neq k} (x_k - x_j)}$$

の線形結合であり  $(0 \leq k \leq n)$ ,

$$q_n(x) = \sum_{k=0}^n f(x_k) p_k(x)$$

であった.

- 各  $k$  に対し,

$$W_K = \int_a^b p_k(x)w(x)dx$$

を事前に計算しておく.

- 以上の記法のもとで次式が成り立つ.

$$\int_a^b q_n(x)w(x)dx = \sum_{k=0}^n f(x_k)W_k$$

- この積分公式は,  $f$  が  $2n + 1$  次の多項式まで, 正しい積分の値を与えることが証明できる ([斎藤] pp. 205–206).

- Gauss 型公式には, 重み関数と直交多項式の取り方に応じて, 色々な種類があるが, 詳細は略す ([斎藤], [杉原, 室田] 参照).
- Scilab で数値積分の基本となっている `intg` でも, Gauss 型公式の一種が採用されている.

# Scilab における数値積分

- 以下では, Scilab の数値積分関連の関数について説明する.

# 台形則

- 台形則による積分には関数 `inttrap` を使う.
- `inttrap` の第一引数には  $x$  軸上の分点をならべたベクトルを, 第二引数には対応する関数値をならべたベクトルを与える.
- `inttrap` を使うときに必要なのは関数の値であって, 関数そのものではないことに注意.



- 第一引数を省略できるが, 省略すると横軸は1刻みであると解釈される. 予期しない結果になることがあるので, 第一引数を省略しない方がよい.
- 横軸を0から $\pi$ まで0.1刻みで動かし, `inttrap`で  $\int_0^\pi \sin x dx$  を計算する例を次に示す. 横軸の値が格納されたベクトルを `x`, 縦軸の値が格納されたベクトルを `y` とする.

## 実行例

```
x=0:0.1:%pi;  
y=sin(x);  
v=inttrap(x,y);
```

# 実行結果

```
-->v
```

```
v =
```

```
1.9974689
```

- $\int_0^{\pi} \sin x dx = 2$  となる筈であるが、数値積分の値には 0.1%程度の相対誤差が含まれている。

# スプライン補間による数値積分

---

- スプライン補間による積分をおこなう関数は `intsplin`. 使い方は `inttrap` と同じ.
- 先と同様に, 横軸を 0 から  $\pi$  まで 0.1 刻みで動かして, `inttrap` で  $\int_0^\pi \sin x dx$  を計算する例を次に示す.

## 実行例

```
x=0:0.1:%pi;  
y=sin(x);  
v=intsplin(x,y);
```

# 結果

-->v

v =

1.9991349

- $\int_0^{\pi} \sin x dx = 2$  となる筈であるが, 数値積分の値には 0.04%程度の相対誤差が含まれている.



# Gauss 型公式

- Scilab で Gauss 型公式 (21 点 Gauss-Kronrod 則) によって積分を計算する関数は `intg` であるが, この使い方はあまり直感的でない.

- 関数 `integrate` は, `intg` に使いやすいユーザインターフェースを提供しているので (内部では各区間に対して `intg` を使っている), こちらを使うとよい.

- `integrate` は, `inttrap` や `intsplin` と異なり, 積分したい関数の値ではなく被積分関数そのものを第一引数として要求する. 第二引数は被積分関数の変数である. 第三引数は積分の下限であるが, 第四引数には積分の上限をならべたベクトルである.

- 仮に, 被積分関数を  $f$ , 第三引数を  $s$  とする.
- 第四引数にスカラー (仮に  $t$  とする) を与えると, `integrate` は  $\int_s^t f(x)dx$  を計算する.
- 第四引数にベクトル (仮に  $t = (t_1, t_2, \dots, t_n)$  とする) を与えると, `integrate` は  $\int_s^{t_i} f(x)dx$  が格納されたベクトルを返す.

- 第一引数の被積分関数と第二引数は数値ではなく文字列であり, 引用符''で括って文字列であることを明示しなければならない.

- 以下に, `integrate` で  $\int_0^\pi \sin x dx$  を計算する  
例と,  $\int_0^{t_k} \sin(x) dx$  (ただし  $t_k = k\pi/2, 1 \leq k \leq 4$ ) を計算する例を示す.

## 実行例

```
v=integrate('sin(x)', 'x', 0, %pi);
```

# 結果

-->v

v =

2.



## 実行例

```
t=%pi/2*(1:4);  
v=integrate('sin(x)', 'x', 0, t);
```

# 結果

-->v

v =

1.        2.        1.    - 2.220D-16

# 微分

- 関数を数値的に微分するもっとも簡単な方法は、前進差分近似、すなわち  $h > 0$  として、

$$f'(x) \simeq \frac{f(x+h) - f(x)}{h}$$

とすること。これを**前進差分近似**という。

- 上記のかわりに,  $h > 0$  として,

$$f'(x) \simeq \frac{f(x) - f(x - h)}{h}$$

とすることもあり, これを後退差分近似という.

- 前進差分近似と中心差分近似の「偏り」をなくすために,  $h > 0$  として,

$$f'(x) \simeq \frac{f(x+h) - f(x-h)}{2h}$$

とする方法もある. これを**中心差分近似**という.

- 微分の近似値を  $Df$  とする.  $Df(x) = f'(x) + O(h^p)$  となっているとき,  $Df$  は  $p$  次の精度を持つという.
- ただし,  $g(x)$  が  $O(h^p)$  であるとは,

$$\lim_{h \rightarrow 0} \frac{|g(h)|}{|h|^p} < \infty$$

であることをいう.

- 前進差分近似と後退差分近似は 1 次, 中心差分近似は 2 次の精度である. より高い次数の公式を作ることもできる.

- 関数の評価値に誤差が含まれる場合 (測定実験などで関数値を得ていて値に雑音が含まれる場合など) には, 差分近似は誤差の影響を大きく受けるので, 注意が必要である.



- 差分近似に伴う誤差を回避するため、データ点に対してスプライン補間をおこない、スプライン補間多項式の微分を微分の推定値とする方法もある。

- 平滑化フィルタによって雑音 (と思われるもの) を除去してから数値的な微分をおこなうこともある。

- 合成関数の微分の手続きを利用して微分を計算する高速微分という方法もある (詳細は略, 興味がある者は [杉原, 室田] を参照).

# Scilab における微分

- 数値微分を計算するには関数 `numderivative` を使う (有限差分近似).

- 第一引数は関数名, 第二引数は微分を評価する点, あるいはそれをまとめたベクトルである.
- 第三引数 (オプション) で刻み幅を指定できる.
- 第四引数 (オプション) には精度に関する次数を指定する. デフォルトは 2 で, これ以外に 1 と 4 が選択可能である.

- 以下に例を示す.

## 実行例

```
v=numderivative(sin,%pi);
```

## 結果

```
-->v  
v    =  
- 1.
```