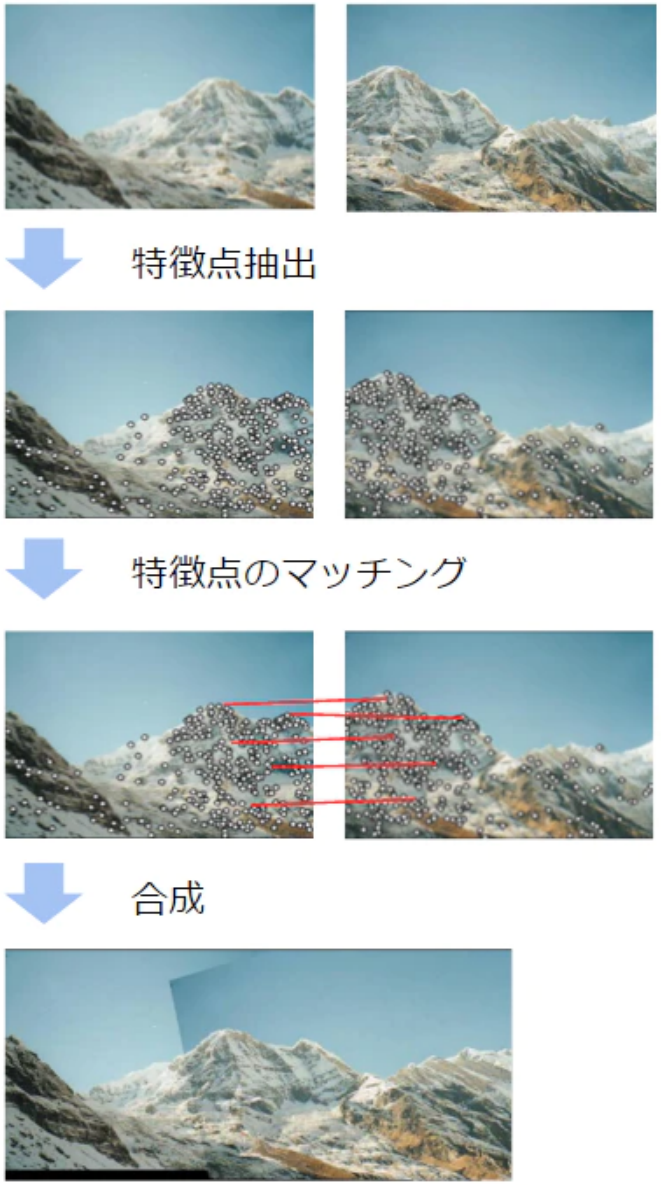
2020年10月25日

本スライドは、画像処理の数式を見て石になった時のための、金の針に加筆したものになる。

画像の特徴点とは

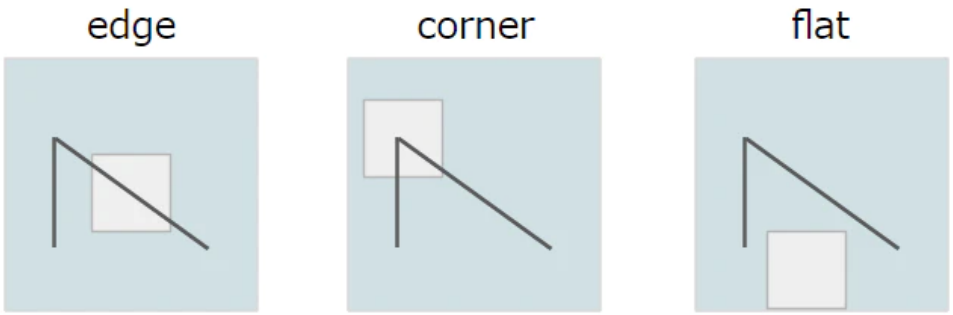
人間がジグソーパズルを組み立てられるのは、私たちが各ピースの特徴(形状、描かれている模様など)を把握して、それと似た、連続する特徴を見つけ出してつなぎ合わせることができるから、といえる。

同様に、複数の写真をつなげてパノラマ写真にすることができるのは、写真間で共通する特徴を見つけ出してつなぎ合わせているから。



特徴点の種類

「特徴点」をシンプルに考えていくと、最終的に以下の 3 つに集約することができる。



ここで、

- **edge**: 差異が認識できる境界がある
- **corner**: edge が集中する点
- **flat**: edge でも corner でもない、特徴が何も認識できない点

2020年10月25日

特徴点を見つけるには

「特徴点」は、以下のルールに則って検出される。

- 再現性: ある特徴点は常に特徴点として認識される。
- 識別性: ある特徴点は、その他の特徴点と明確に異なると識別できる。

再現性

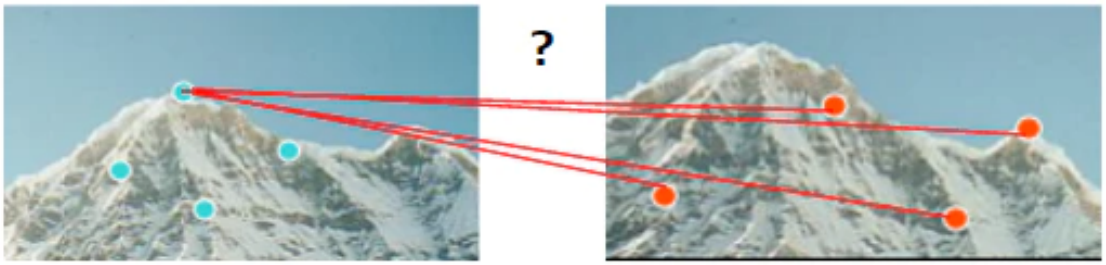
パノラマ写真を作成する場合、2 枚の写真の撮影角度が少し異なる程度で特徴点が大きく変化するとなると、特徴点どうしをつなぎ合わせる処理そのものが破綻してしまう。

認識する点が異なり、対応させられない

そのため、特徴点は**画像の角度や拡大率などに対して頑健性を持つ**ものが良いということになる。

識別性

また、認識した特徴点が固有の特徴を持たない場合、対応付ける特徴点を決定することができない。



どの特徴点と一致させるべきかわからない

そのため、**各特徴点が一意に識別できる表現方法**が重要になる。

まとめ

これらは、**特徴点の検出** (Feature Detection) と **特徴点の表現方法** (Feature Description) **の獲得** それぞれに求められていることと一致する。まとめると、「画像の角度や拡大率などで変化しない、頑健な特徴点」を検出し、それをなるべく「一意に識別できる表現方法」を獲得することが画像における特徴点の認識の目標になる。

2020年10月25日

特徴点の検出(Feature Detection)

特徴点の検出は、概ね次のような流れで進められる。
「edge 検出」→ edge が集中する「corner 検出」

edge 検出

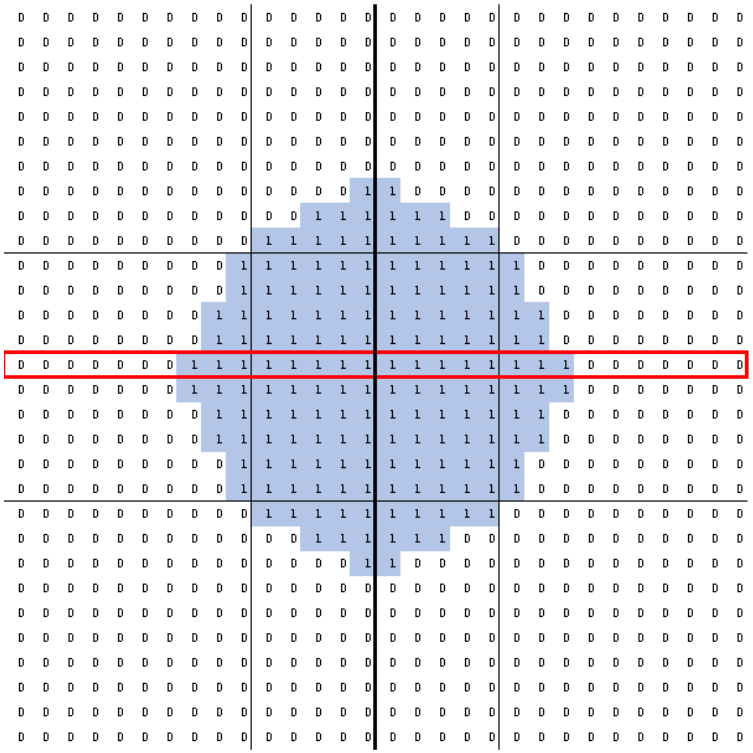
edge とは、具体的には**輝度が大きく変化している点**になる。

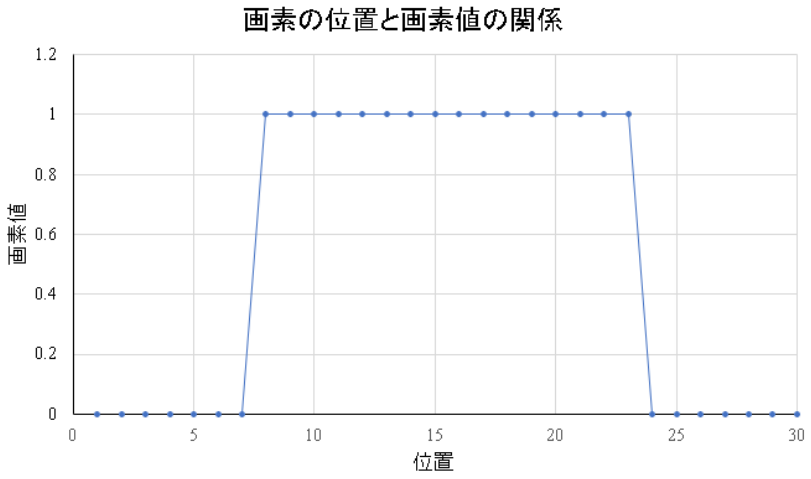
例えば、黒いタイルの真ん中に白い円が描かれた画像を考える。その画像に対して、下図のように引いた赤い直線上の輝度を座標に沿ってプロットすると、右のようなグラフが得られる。

ここで、私たちが検出したいのは **edge**。
つまりグラフ上で「暗い -> 明るい」、「明るい -> 暗い」と輝度が大きく変化しているポイントになる。

そのためには、グラフの傾きが大きく変化している点を探せばよい。
すなわち、画像全体の輝度を関数として画素ごとにその微分を求め、微分した値がその前後の画素値に対して大きく変化している画素が edge となる。

2020年10月25日

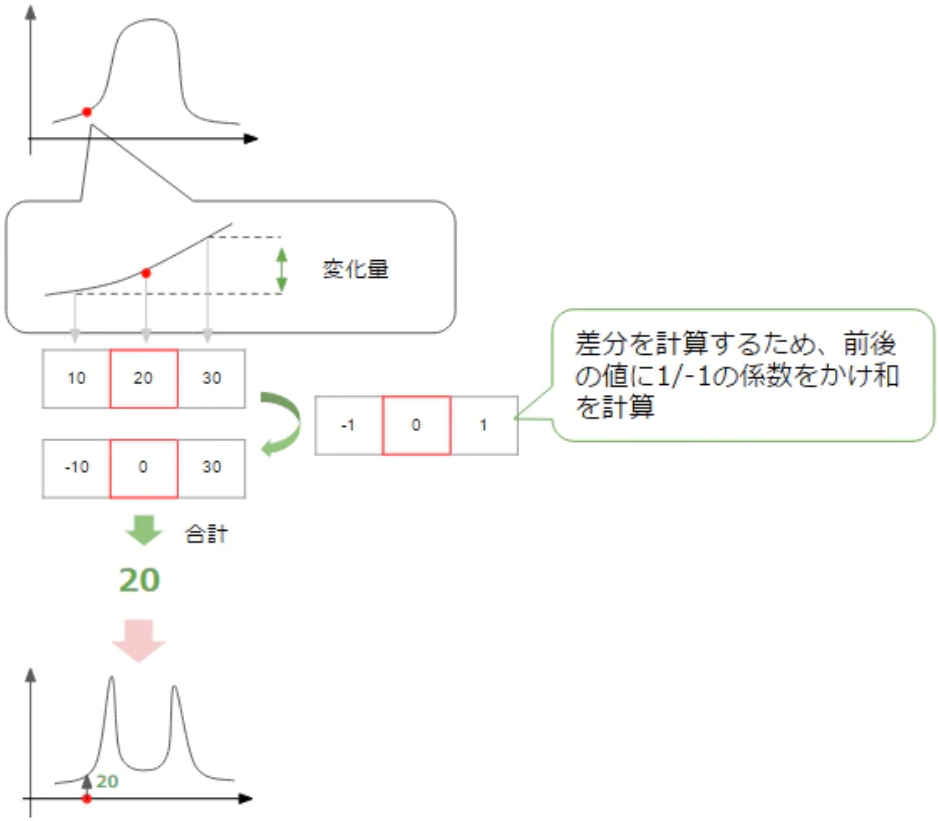




2020年10月25日

微分は、無限と連続の世界であるが、コンピュータ上での計算は有限かつ 0 と 1 の世界である。そのため、コンピュータ上で微分を計算する際には、連続場を離散化して考えなければならない。

このような条件下で、関数の微分値を近似する方法が差分法である。以下では、その中でも 2 次の中心差分を用いて計算を行う。



すると、上図のように変化が大きな点、つまり edge に該当する箇所で変化率が上昇するグラフを得ることができる (変化量には ± があるがここでは絶対値でプロットしているものとする)。また、上図は水平方向での変化量を表しているが、垂直方向の場合も同様に考えることができる。

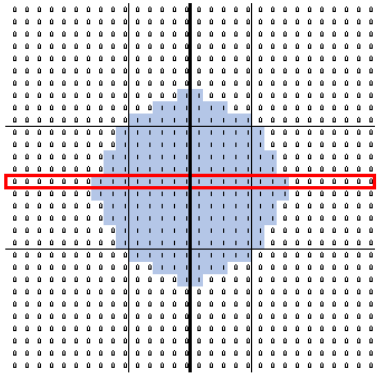
最終的に、水平方向、垂直方向双方の変化量うぃ合計し(通常は二乗平方根)、その値がある閾値より大きい点を採集する。

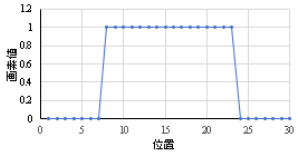
これが edge 検出の基本的な考え方である。

先ほどの白い円の画像に対して、輝度値の差分を用いた edge 検出を行った結果を次に示す(左図：二値画像，右図：[-1, 0, 1]のフィルタリングを行った画像(太線は元のオブジェクトの輪郭、オレンジ色は edge として検出された領域))。

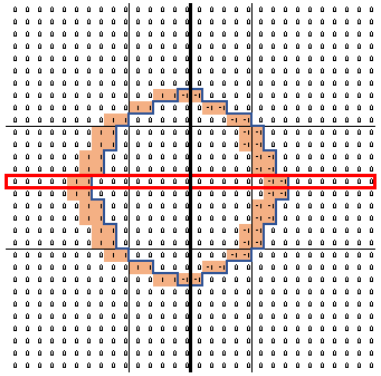
2020年10月25日

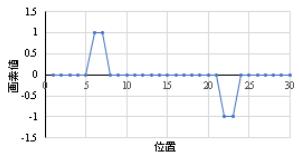
元画像



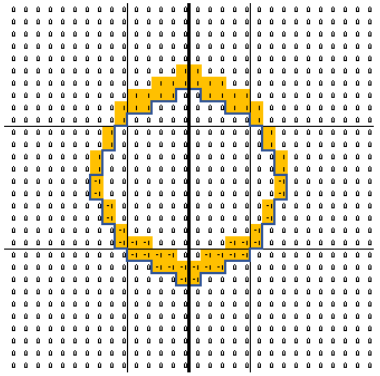


水平方向にフィルタリング

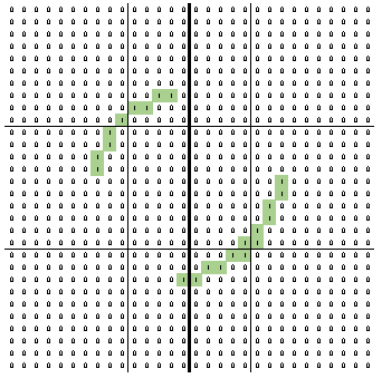




垂直方向にフィルタリング



$$I(x,y) = \begin{cases} 1 & if\ \sqrt{画像2の画素値^2 + 画像3の画素値^2} > 1 \\ 0 & otherwise \end{cases}$$

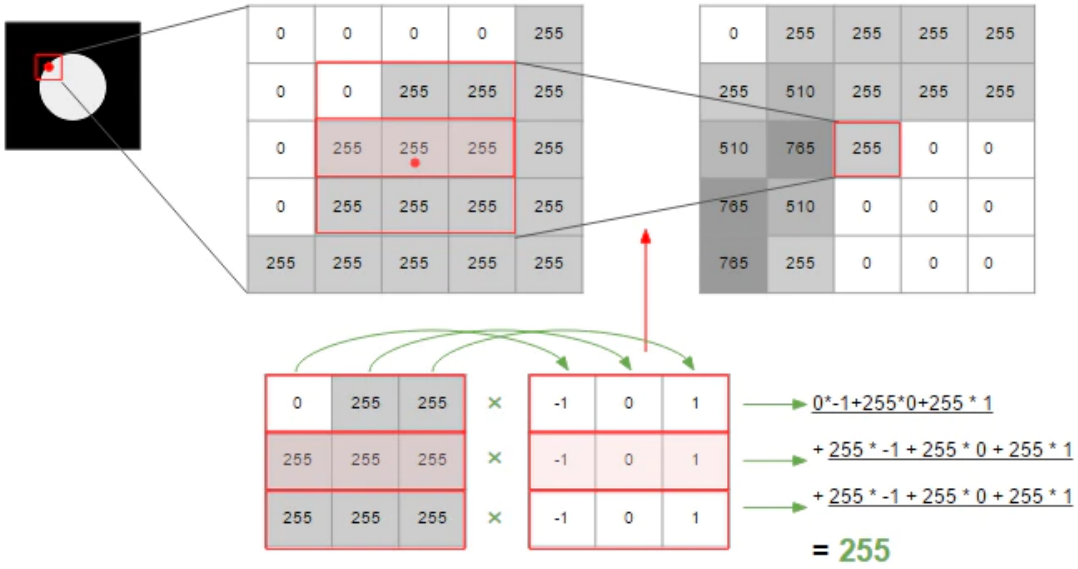


2020年10月25日

上図からわかるように、この edge 検出は精度が低い。そこで、より精度を上げるための方法を次に説明する。

スムージング

輝度値の変化量を計算する際に、周辺部分との差も考慮する方法。
ざっくり言うと、平均を取るようなイメージ。平均を取るよ変化をならすことができ(スムージング)、よって滑らかにつながる edge を検出することができる。



上図では、変化量を計算するために 3×3 の、$-1/0/1$ の値が入った行列を用いて計算している。こうした変化量を計算するための行列を**フィルタ**と呼ぶ。

このフィルタには様々な種類があり、上図で使っていたのはこのうちの Prewitt フィルタになる。Sobel フィルタは Prewitt フィルタより中心に隣接するものを重視したものになる。
Gaussian フィルタは正規分布の関数を利用し、中心を頂点として端に向かってなだらかに係数をかけることができる。edge の検出でよく利用される Canny 法はこの Gaussian フィルタを使用している。

Prewitt

水平方向

-1	0	1
-1	0	1
-1	0	1

垂直方向

-1	-1	-1
0	0	0
1	1	1

Sobel

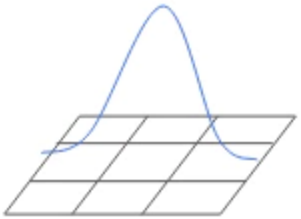
水平方向

-1	0	1
-2	0	2
-1	0	1

垂直方向

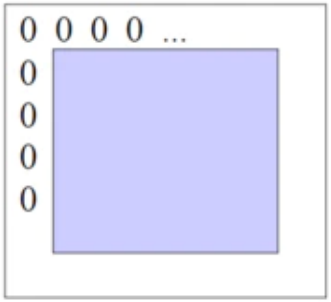
-1	-2	-1
0	0	0
1	2	1

Gaussian



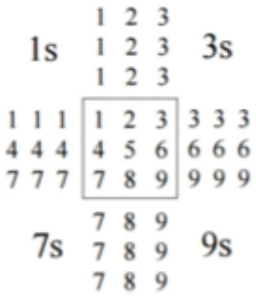
なお、画像の端の部分については、隣接する点がないため補間を行う必要がある。この補間方法には以下のような方法がある。

zero padding



ゼロで埋める

Replication

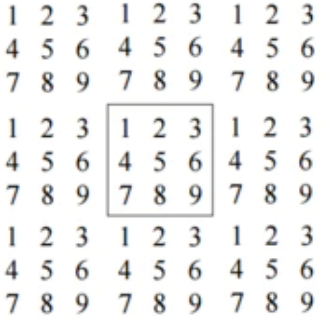


端の値を複製する

Reflection

端を中心に対象にする

Wrap-Around



囲む

CSE/EE486 Computer Vision I, Lecture3, p26~

Edge 検出まとめ

- edge を検出するには、画像における**輝度の変化量**を手掛かりにする。
- 変化量を計算する際は、一般的に**スムージング**を行う。これは、周辺の変化量を加味することで検出精度を上げるためのものである。
- この「周辺」の対象範囲と、それに対する重みづけを定義するのが**フィルタ**であり、様々な種類がある。
- 変化量は水平方向と垂直方向の 2 方向でとることができる。この値の二乗和平方根を変化量とし、これを **Magnitude** と呼ぶ。
- Magnitude が一定の閾値 θ を超えた場合に edge と判定することで、edge の検出が可能になる。すなわち

$$edge(x,y) = \begin{cases} 1 & if \ \ magnitude > \theta \\ 0 & otheres \end{cases}$$

微分と差分法

宇宙磁気流体・プラズマシミュレーションサマースクール 2016から転用。

はじめに

- 微分方程式
  - 未知関数とその導関数を含む方程式
  - 自然現象などを記述する基礎方程式

$$m\frac{d^2r}{dt^2}=F(r,t),\ V(t)=R(I)I+L(I)\frac{dI}{dt},\ \frac{dX}{dt}=\mu(X,t)+\sigma(X,t)\frac{dB}{dt},$$

$$\Delta\phi=\frac{\rho}{\varepsilon},\ i\hbar\frac{\partial\Psi}{\partial t}=-\frac{\hbar}{2m}\Delta\Psi+V(r)\Psi,\ \begin{cases}\frac{\partial \boldsymbol{u}}{\partial t}+\boldsymbol{u}\cdot\nabla\boldsymbol{u}=-\nabla p+\nu\Delta\boldsymbol{u}\\ \nabla\cdot\boldsymbol{u}=0\end{cases},\ \cdots$$

はじめに

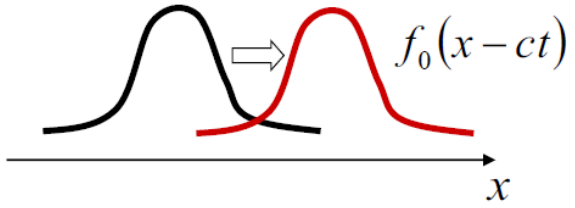
- 双曲型方程式
  - 線形移流方程式
  - 非粘性Burgers方程式
  - Maxwell方程式
  - Euler方程式
  - 理想MHD方程式

微分方程式を計算機で解きたい！

はじめに

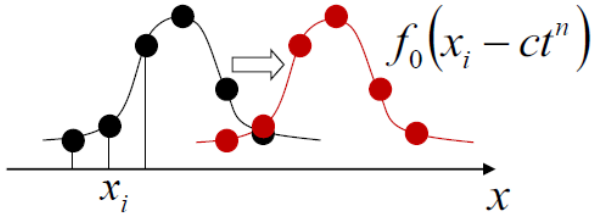
- 微分方程式の世界

無限と連続の世界



- 計算機の世界

有限の0と1の世界



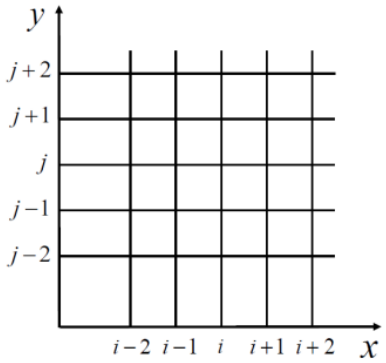
- 連続場の離散化
  - 空間　：$x_0, x_1, \cdots$
  - 時間　：$t^0, t^1, \cdots$
  - 実数値：$0.1, 0.2, \cdots$

```
program main
   implicit none
   real(8) :: a
   a = 0.1    ; write(*,*) a
   a = 0.1d0; write(*,*) a
end program main
```

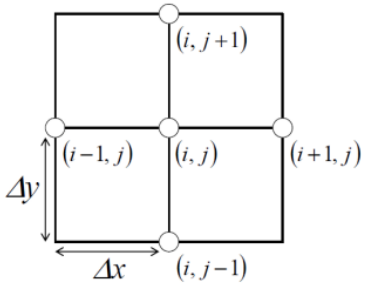
```
$ ./a.out
0.100000001490116
0.100000000000000
```

はじめに

座標および変数の離散表記法

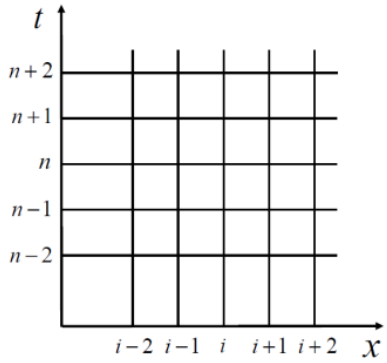


$$x_i, y_j, u_{i,j} = u(x_i, y_j)$$

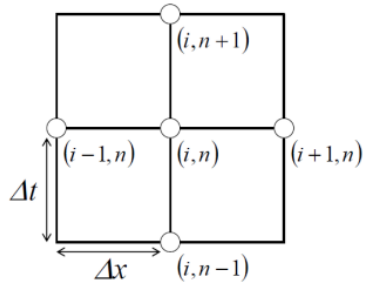


はじめに

時間・空間座標および変数の離散表記法



$$x_i, t^n, u_i^n = u(x_i, t^n)$$



2020年10月25日

微分法

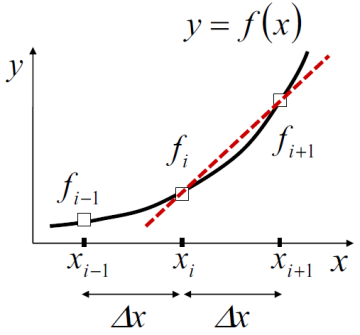
$$\frac{\partial f}{\partial x}=\lim_{\Delta x\to 0}\frac{f(x+\Delta x)-f(x)}{\Delta x}$$

差分法

$$\left(\frac{\partial f}{\partial x}\right)_i=\frac{f(x_i+\Delta x)-f(x_i)}{\Delta x}$$

$$=\frac{f_{i+1}-f_i}{\Delta x}$$

ただし、$x_{i+1}\equiv x_i+\Delta x$



前進差分 という 以上

前進差分の誤差

$$\left(\frac{\partial f}{\partial x}\right)_i=\frac{f(x_i+\Delta x)-f(x_i)}{\Delta x}$$

Taylor展開

$$=\frac{1}{\Delta x}\left(f(x_i)+\Delta x\frac{\partial f(x_i)}{\partial x}+\frac{\Delta x^2}{2!}\frac{\partial^2 f(x_i)}{\partial x^2}+\cdots-f(x_i)\right)$$

$$=\frac{\partial f(x_i)}{\partial x}+\frac{\Delta x}{2!}\frac{\partial^2 f(x_i)}{\partial x^2}+O(\Delta x^2)$$

$$\Rightarrow\left(\frac{\partial f}{\partial x}\right)_i-\frac{\partial f(x_i)}{\partial x}=O(\Delta x)$$

誤差がΔxの1次に比例

中心差分

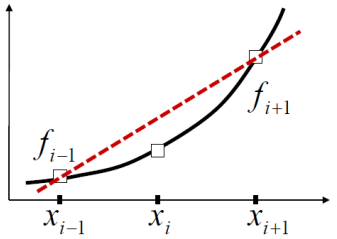
$$f_{i+1}=f_i+\Delta x\frac{\partial f(x_i)}{\partial x}+\frac{\Delta x^2}{2!}\frac{\partial^2 f(x_i)}{\partial x^2}+\frac{\Delta x^3}{3!}\frac{\partial^3 f(x_i)}{\partial x^3}+\cdots$$

$$f_{i-1}=f_i-\Delta x\frac{\partial f(x_i)}{\partial x}+\frac{\Delta x^2}{2!}\frac{\partial^2 f(x_i)}{\partial x^2}-\frac{\Delta x^3}{3!}\frac{\partial^3 f(x_i)}{\partial x^3}+\cdots$$

$$\Rightarrow\frac{\partial f(x_i)}{\partial x}=\frac{f_{i+1}-f_{i-1}}{2\Delta x}-\frac{\Delta x^2}{3!}\frac{\partial^3 f(x_i)}{\partial x^3}+O(\Delta x^4)$$

$$\Rightarrow\left(\frac{\partial f}{\partial x}\right)_i=\frac{f_{i+1}-f_{i-1}}{2\Delta x}$$

誤差がΔxの2次に比例



1階差分法のまとめ

$$\left(\frac{\partial f}{\partial x}\right)_i=\frac{f_i-f_{i-1}}{\Delta x}$$ （1次後退差分）

$$\left(\frac{\partial f}{\partial x}\right)_i=\frac{f_{i+1}-f_i}{\Delta x}$$ （1次前進差分）

$$\left(\frac{\partial f}{\partial x}\right)_i=\frac{f_{i+1}-f_{i-1}}{2\Delta x}$$ （2次中心差分）

$$\left(\frac{\partial f}{\partial x}\right)_i=\frac{3f_i-4f_{i-1}+f_{i-2}}{2\Delta x}$$ （2次後退差分）

$$\left(\frac{\partial f}{\partial x}\right)_i=\frac{-f_{i+2}+8f_{i+1}-8f_{i-1}+f_{i-2}}{12\Delta x}$$ （4次中心差分）

差分法

二階中心差分

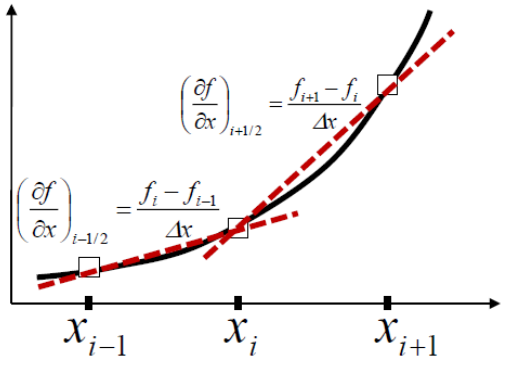
$$f_{i+1} = f_i + \Delta x \frac{\partial f(x_i)}{\partial x} + \frac{\Delta x^2}{2!}\frac{\partial^2 f(x_i)}{\partial x^2} + \frac{\Delta x^3}{3!}\frac{\partial^3 f(x_i)}{\partial x^3} + \cdots$$

$$f_{i-1} = f_i - \Delta x \frac{\partial f(x_i)}{\partial x} + \frac{\Delta x^2}{2!}\frac{\partial^2 f(x_i)}{\partial x^2} - \frac{\Delta x^3}{3!}\frac{\partial^3 f(x_i)}{\partial x^3} + \cdots$$

$$\Rightarrow \frac{\partial^2 f(x_i)}{\partial x^2} = \frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta x^2} + O(\Delta x^4)$$

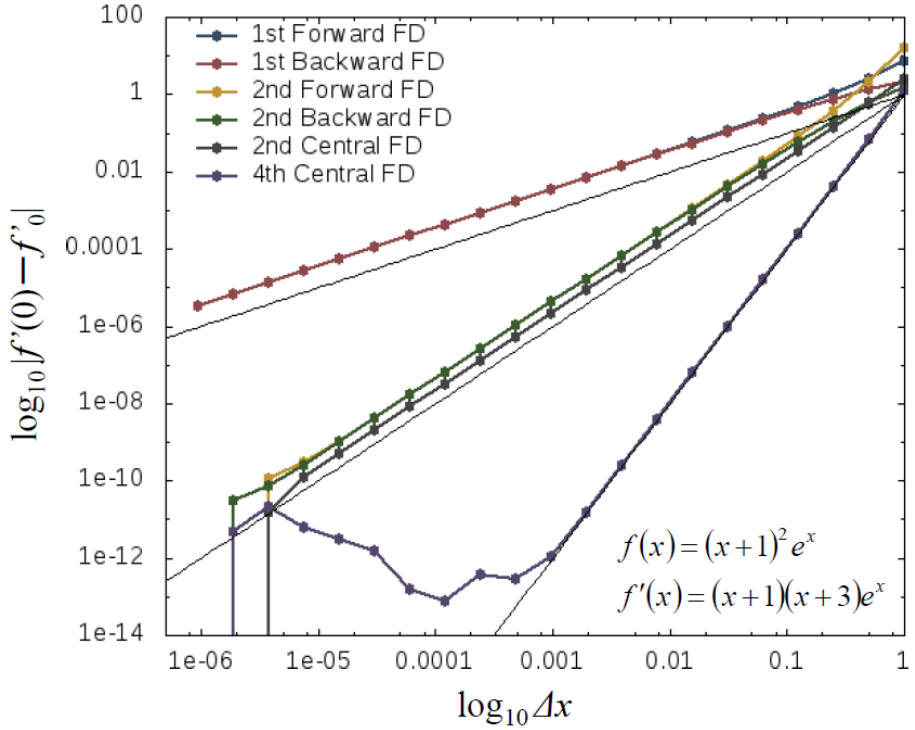
$$\Rightarrow \left(\frac{\partial^2 f}{\partial x^2}\right)_i = \frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta x^2}$$

誤差がΔxの2次に比例



差分法

誤差の比較


1st Forward FD
1st Backward FD
2nd Forward FD
2nd Backward FD
2nd Central FD
4th Central FD
100
1
0.01
0.0001
1e-06
1e-08
1e-10
1e-12
1e-14
1e-06
1e-05
0.0001
0.001
0.01
0.1
1
$\log_{10}|f'(0)-f'_0|$
$\log_{10}\Delta x$
$f(x)=(x+1)^2e^x$
$f'(x)=(x+1)(x+3)e^x$