(3a) 差分法の考え方 (1/5)

微分・積分： 関数の連続性が前提

微分方程式： 微分形式で表された方程式→積分によって解を求める

例. $\frac{dy}{dx} = x^2$ → $y = \frac{1}{3}x^3 + C$

一般に, $\frac{dy}{dx} = f(x)$ で $f(x)$ という関数が既知

→ 数学的に（解析的に）$y = F(x)$ が求まる

２階以上の微分でも同様

- $f(x)$ がきれいな関数で表されないとき
- 実験データのように時々刻々と変化するデータであるとき

→ 解析的に解を求めることが困難（不可能）

微分を差分で近似 → 近似的に解を求める

連続的な微分方程式を，離散的な差分方程式で近似し，
解析的に解を求める代わりに，数値的に解を求める（数値解析）

(3a) 差分法の考え方 (2/5)

1階微分の場合

微分の定義

$$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{y(x+\Delta x) - y(x)}{\Delta x}$$

※正確には $\Delta x \to \pm 0$
時間微分や境界条件では注意

単純に,

$$\frac{dy}{dx} \cong \frac{\Delta y}{\Delta x}$$

ただし, Δx は十分に小さくとること

$\frac{dy}{dx} = f(x)$ とすると, $\Delta y = y(x+\Delta x) - y(x)$ であるから,

$$y(x+\Delta x) = y(x) + f(x)\Delta x$$

となる.

関数 $f(x)$ が既知で, $x = x_0$ での y の値 y_0 (初期値) がわかれば,
あらゆる x に対する値 y が求まる

$$y(x_1) = y(x_0) + f(x_0)\Delta x$$
$$y(x_2) = y(x_1) + f(x_1)\Delta x$$
$$y(x_3) = y(x_2) + f(x_2)\Delta x$$

(3a) 差分法の考え方 (3/5)

2階微分の場合

テーラー展開

$$y(x+\Delta x) = y(x) + \frac{dy}{dx}\Delta x + \frac{1}{2}\frac{d^2y}{dx^2}\Delta x^2 + \ldots$$

$$y(x-\Delta x) = y(x) - \frac{dy}{dx}\Delta x + \frac{1}{2}\frac{d^2y}{dx^2}\Delta x^2 - \ldots$$

両辺の和をとると，

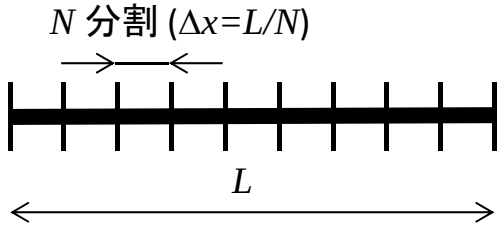
$$y(x+\Delta x) + y(x-\Delta x) = 2y(x) + \frac{d^2y}{dx^2}\Delta x^2 + \ldots$$

ここで，右辺の3次以上の項は，Δx が小さいときには無視できる

$$\frac{d^2y}{dx^2} = \frac{y(x+\Delta x) + y(x-\Delta x) - 2y(x)}{\Delta x^2}$$

(3a) 差分法の考え方 (4/5)

1次元熱伝導方程式 $\dfrac{dT}{dt}=\kappa\dfrac{d^2T}{dx^2}$　　ただし，$0 \leqq x \leqq L$



$$\frac{T(t+\Delta t)-T(t)}{\Delta t}=\kappa\frac{T(x+\Delta x)+T(x-\Delta x)-2T(x)}{\Delta x^2}$$

正確には，$T = T(x, t)$ なので，左辺の T は座標 x の値，右辺の T は時刻 t の値とすると，

$$\frac{T(x,t+\Delta t)-T(x,t)}{\Delta t}=\kappa\frac{T(x+\Delta x,t)+T(x-\Delta x,t)-2T(x,t)}{\Delta x^2}$$

すなわち，

$$T(x,t+\Delta t)=T(x,t)+\kappa\frac{T(x+\Delta x,t)+T(x-\Delta x,t)-2T(x,t)}{\Delta x^2}\Delta t$$

見やすく書くと，

$$T(t+\Delta t)=T(t)+\left\{T(x+\Delta x)+T(x-\Delta x)-2T(x)\right\}\frac{\kappa\Delta t}{\Delta x^2}$$

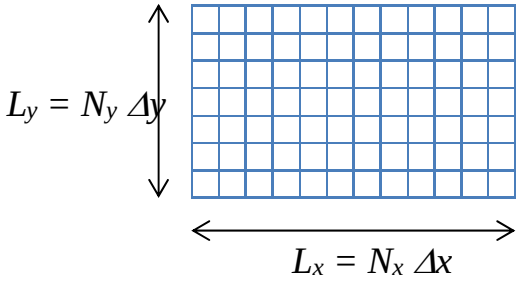
右辺はすべて時刻 t の値
→ 時刻 t における温度分布が既知なら，時刻 $t + \Delta t$ での温度分布が計算できる
→ 時刻 $t = 0$ での温度分布を初期条件として設定

ただし，x=0 における x-Δx と x=L における x+Δx は規定できない
→境界条件として設定

(3a) 差分法の考え方 (5/5)

2次元熱伝導方程式

$$\frac{dT}{dt} = \kappa\left(\frac{d^2T}{dx^2} + \frac{d^2T}{dy^2}\right)$$



右辺の差分化において，$\Delta x = \Delta y$ ととると，

$$T(t+\Delta t) = T(t) + \Delta T(x,y)\frac{\kappa\Delta t}{\Delta x^2}$$

$$\Delta T(x,y) = T(x+\Delta x, y) + T(x-\Delta x, y) + T(x, y+\Delta y) + T(x, y-\Delta y) - 4T(x,y)$$

→ x=0, L_x, y=0, L_y での値（境界条件）と，$t = 0$ での温度分布（初期条件）を与えると，
その後の全領域での温度変化が計算できる．

境界条件 （例は，1次元熱伝導方程式の $x = 0$ での境界条件）

温度一定条件： 設定温度を与えればよい　$T(x=0) = T_0$

対称条件: 境界の外側に，内側と同じ分布が存在

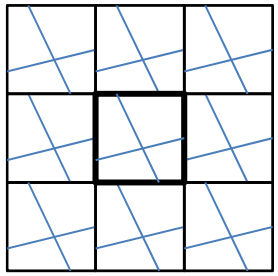
仮想的に x=-1 という格子点に x=1 と同じ温度を設定

$$\Delta T(0) = T(0+\Delta x) + T(0-\Delta x) - 2T(x) = 2T(\Delta x) - 2T(0)$$

周期境界条件: 計算領域と同じ温度分布が無限に繰り返す

仮想的に x =1 という格子点に x=L と同じ温度を設定

$$\Delta T(0) = T(0+\Delta x) + T(0-\Delta x) - 2T(x) = T(\Delta x) + T(L) - 2T(0)$$



周期境界条件：
実際には連続的
に周期性が満た
される分布となる

断熱条件： 熱の流れが0→温度勾配が0

境界の値は1つ内側と同じ値　$T(0) = T(\Delta x)$

熱伝達条件： 熱の流れが一定値

$$T(0) = T(\Delta x) + Q\Delta x$$

Q は一定値 or Q = h (T(0)-Te)

(3b) 熱伝導方程式（差分法）のプログラム (1/3)

```
public void paint(Graphics g){
int     i, j, cx, cy, x0, y0, px, py, w, h;
float   colc;
double  dx, dy;
    （略）
    dx=(double)BW/(double)NX;
    dy=(double)BH/(double)NY;

if(kk==0) {initset(); kk++;}

for(int ii=0;ii<kgo;ii++){
    sabun(ii);
    drawstep(g,kk);
    kk++;
}//next ii
kgo=0;

for(i=0;i<NX;i++){
for(j=0;j<NY;j++){
px = (int)(i*dx);
py = (int)(j*dy);
colc = 0.7f*(1.0f-(float)Tn[i][j]);
g.setColor(Color.getHSBColor(colc,0.9f,0.9f));
g.fillRect(px+x0, py+y0, (int)dx, (int)dy);
}
}
    （略）
}// end paint()
```

メインプログラム←paint()

double [][] Tn, Tp;　倍精度の二次元配列

Tn=new double[NX][NY];　次の時刻の温度

Tp=new double[NX][NY];　現時刻の温度

初期設定 initset();

```
public void initset(){
for(i=0;i<NX;i++){
for(j=0;j<NY;j++){
if(i<NX/2) Tn[i][j] = 0.0;
else       Tn[i][j] = 1.0;
Tp[i][j] = Tn[i][j];
}}}
```

差分を指定ステップ数ずつ，ひたすら繰り返す

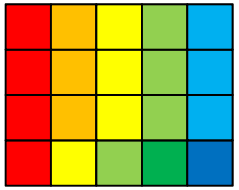
画面上にstep数を表示

（kgo は画面上のボタンを押す度に設定される）

描画

差分点を温度で色分けして正方形表示

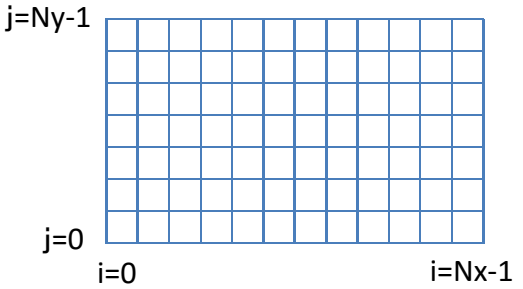
HSBカラーでTn=0で0.7（青），Tn=1で0（赤）



(3b) 熱伝導方程式（差分法）のプログラム (2/3)

```
public void sabun(int step){
int     i, j, i1, i2, j1, j2;
double  dtx, dty, delT;

for(i=0;i<NX;i++){
    for(j=0;j<NY;j++) {
        Tp[i][j]=Tn[i][j];
    }
}

for(i=0;i<NX;i++){
    if(i==0)    i1=1;   else i1=i-1;
    if(i==NX-1) i2=i-1; else i2=i+1;
    for(j=0;j<NY;j++){
        if(j==0)    j1=1;    else j1=j-1;
        if(j==NY-1) j2=j-1; else j2=j+1;

        dtx = Tp[i1][j]+Tp[i2][j]-2.0*Tp[i][j];
        dty = Tp[i][j1]+Tp[i][j2]-2.0*Tp[i][j];

        delT = Kap*(dtx + dty)*DeT/DeL/DeL;

        Tn[i][j] = Tp[i][j] + delT;
    }// end j
}// end i

}// end sabun
```

前進差分のサブルーチン sabun()



前の時刻の温度を繰り上げ

x方向に左端から右端まで

対称（断熱）境界条件の設定

y方向に下端から上端まで

対称（断熱）境界条件の設定

右辺の計算（現時刻の温度を使う）

$$\{T(x+\Delta x,y)+T(x-\Delta x,y)+T(x,y+\Delta y)+T(x,y-\Delta y)-4T(x,y)\}\frac{\kappa\Delta t}{\Delta x^2}$$

DeL $=\Delta x=\Delta y$, DeT $=\Delta t$, Kap $=\kappa$

次の時刻の温度を計算

$$T(t+\Delta t)=T(t)+\Delta T(x,y)$$

(3b) 熱伝導方程式（差分法）のプログラム (3/3)

HeatCond.java

```
////////////////////////////
private void drawstep(Graphics g, int tt){
g.setColor(Color.white);
g.fillRect(0,50,300,25);
g.setColor(Color.black);
g.drawString("kk="+tt+" (t="+tt*DeT+")",10,70);
}
////////////////////////////////// Paint (main)
public void paint(Graphics g){
int                 i, j, cx, cy, x0, y0, px, py, w, h;
float               colc;
double              dx, dy;
super.paint(g);

w=getSize().width;  h=getSize().height;
cx=w/2; cy=(h-120)/2;
x0=cx-BW/2; y0=cy+BH/2+60;
dx=(double)BW/(double)NX;
dy=(double)BH/(double)NY;

if(kk==0) {initset(); kk++;}

for(int ii=0;ii<kgo;ii++){
sabun(ii);
drawstep(g,kk);   kk++;
}//next ii
kgo=0;

for(i=0;i<NX;i++){
for(j=0;j<NY;j++){
px = (int)(i*dx);   py = (int)(j*dy);
colc = 0.7f*(1.0f-(float)Tn[i][j]);
g.setColor(Color.getHSBColor(colc,0.9f,0.9f));
g.fillRect(x0+px, y0-py, (int)dx, (int)dy);
}}
g.setColor(Color.black);
g.drawRect(x0-1,y0-BH+(int)dy-1,BW+1,BH+1);
drawstep(g,kk-1);
}// end paint()
```

```
////////////////////////////////////////
////////////////////////////////////////
public void initset(){
for(int i=0;i<NX;i++){
for(int j=0;j<NY;j++){
if(i<NX/2)          Tn[i][j] = 0.0;
else                Tn[i][j] = 1.0;
Tp[i][j] = Tn[i][j];
}}}
////////////////////////////////////////
////////////////////////////////////////
public void sabun(int step){
int                 i, j, i1, i2, j1, j2;
double              dtx, dty, delT;

for(i=0;i<NX;i++){
for(j=0;j<NY;j++) {Tp[i][j]=Tn[i][j]; }
}
for(i=0;i<NX;i++){
if(i==0) i1=1; else i1=i-1;
if(i==NX-1) i2=i-1; else i2=i+1;
for(j=0;j<NY;j++){
if(j==0) j1=1; else j1=j-1;
if(j==NY-1) j2=j-1; else j2=j+1;
dtx = Tp[i1][j]+Tp[i2][j]-2.0*Tp[i][j];
dty = Tp[i][j1]+Tp[i][j2]-2.0*Tp[i][j];
delT = Kap*(dtx + dty)*DeT/DeL/DeL;
Tn[i][j] = Tp[i][j] + delT;
}// end j
}// end i
}// end sabun
```

```
//////////////////////////// Main
public static void main(String[] args)
throws Exception{

HeatCond hcrun = new HeatCond();

hcrun.setSize(600, 600);
hcrun.setBackground(Color.white);
hcrun.addWindowListener(new WindowAdapter(){
public void windowClosing(WindowEvent e){
 System.exit(0);}});
hcrun.show();
}
}
```