計算科学特論

代用電荷法演習

linux環境 or Windows環境にログインして
授業情報ウェブサイトにアクセスしてください

URL: http://comp.cs.ehime-u.ac.jp/
~okano/computation/

今日と次回の予定

今日のテーマ:

Laplace方程式の境界値問題にスペクトル法の一種である代用電荷法を適用し、得られた近似解の特徴を知ります。

具体的には、Scilab (サイラボ)のプログラムで差分法演習・有限要素法演習と同様の問題を解きます。

今日の予定

サンプルプログラム解説

サンプルプログラムを変更して
　電荷点・拘束点の数や位置を変更することで近似解にどのような変化があるか比較します。

サンプルプログラム解説

授業資料とサンプルプログラム

授業情報ウェブサイトからダウンロードしてください。

http://comp.cs.ehime-u.ac.jp/~okano/computation/

授業資料の大半はサンプルプログラムリストです。

サンプルプログラムは2種類あります。
Linux用：　　csm-linux.sci
Windows用：　csm-windows.sci
※コメントと改行コードが違うだけです。

サンプルプログラム(csm-*.sci)

```
funcprot(0);
//stacksize('max');
```

ファイル実行時の警告を抑制する

関数定義、以降のコメント部分の例として考えてください。

```
function [x,y] = SetupSquare(z1,z2,xn,yn);
// SetupSquare 矩形領域の問題に必要な設定
// z1 z2 領域の対角頂点、xn yn x方向・y方向の辺の分割数
// x 電荷点(特異点)座標、y 拘束点(補間点)座標
// 拘束点(補間点)座標yのために、左下隅を起点に左回りに座標列を取る
```

宣言部

機能説明

入力要素説明

出力要素説明

～途中省略～

```
endfunction
```

サンプルプログラム(csm-*.sci)

```
funcprot(0);
//stacksize('max');
```

ファイル実行時の警告を抑制する
利用可能なスタックサイズを最大化する
(ver. 5.5以前のscilabで必要)

関数定義 以降のコメント部分の例として考えてください

```
fun
//
//
//
//
endfunction
```

説明

説明

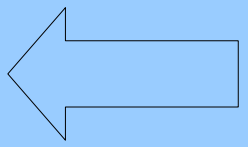
列を取る

Scilab ver. 6.0以前を利用する場合の注意事項:

第6版以前と以降でメモリ割り当ての方式が変更になっています。
ver. 表記が6未満(5.5がインストールされていることがあります。)の場合に
ではスタックサイズを指定しないと容量が足りなくなるかもしれません。
その際は stacksize 関数を使用してください。

※ 今日のサンプルプログラムであれば、右例から左例に変更してください。

```
funcprot(0);
stacksize('max');
```



```
funcprot(0);
//stacksize('max');
```

```
function [x,y] = SetupSquare(z1,z2,xn,yn)
xmin = min(real(z1),real(z2));
xmax = max(real(z1),real(z2));
ymin = min(imag(z1),imag(z2));
ymax = max(imag(z1),imag(z2));
xwid = xmax - xmin;
ywid = ymax - ymin;
y = [xmin+[0:xn-1]*xwid/xn+%i*ymin];
y = [y,xmax+%i*(ymin+[0:yn-1]*ywid/yn)];
y = [y,xmax-[0:xn-1]*xwid/xn+%i*ymax];
y = [y,xmin+%i*(ymax-[0:yn-1]*ywid/yn)];

// 境界拡大法と呼ばれる簡易な方法で電荷点を生成する。
// 電荷点(特異点)座標のために、拘束点(補間点)を中心点から1.5倍に拡大する
y0 = (xmax+xmin)/2 + %i*(ymax+ymin)/2;
x = (y - y0)*1.5 + y0;
endfunction
```

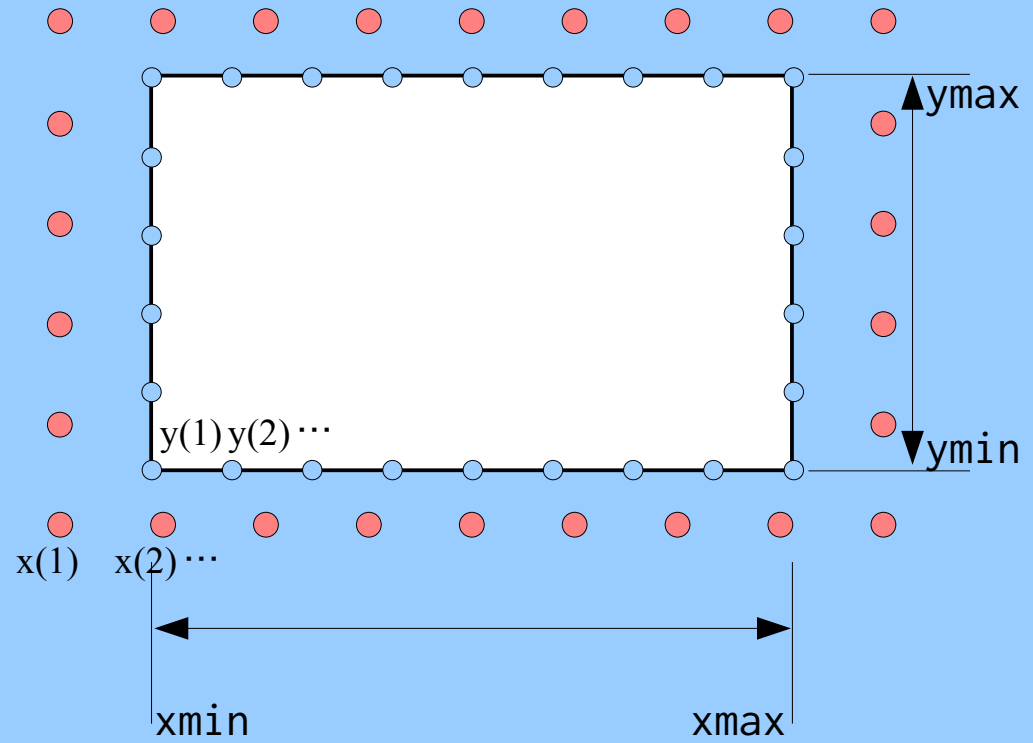
問題設定に応じた電荷点 x と拘束点 y を定義します。

矩形領域の対角頂点座標
領域の上下左右の座標と
縦横の幅を求めます。

左下スミをx(1),y(1)として
四辺を左回りに巡るように
それぞれの辺の分割点に
拘束点をとります。

コメントの通り、境界上の
拘束点を中心から相似拡
大して電荷点を作ります。

```
function [x,y] = SetupSquare(z1,z2,xn,yn)
xmin = min(real(z1),real(z2));
xmax = max(real(z1),real(z2));
y
y
x
y
y
y
y
y
x
y
x
end function
```



を生成する。

点)を中心点から1.5倍に拡大する

問題設定に応じた電荷点 x と拘束点 y を定義します。

矩形領域の対角頂点座標
領域の上下左右の座標と
縦横の幅を求めます。

左下スミをx(1),y(1)として
四辺を左回りに巡るように
それぞれの辺の分割点に
拘束点をとります。

コメントの通り、境界上の
拘束点を中心から相似拡
大して電荷点を作ります。

係数行列を計算する

```
function A = Amat(x,y)
//Amat 代用電荷法の係数行列を計算する
//x 電荷点(特異点)座標、y 拘束点(補間点)座標
//A 係数行列
A = [0,ones(x); ones(y)',log(abs(ones(y)'*x-y.'*ones(x)))];
endfunction
```

$f(z) \sim F(z) = Q_0 + Q_1 \log|z - x_1| + \cdots + Q_n \log|z - x_n|$ なので、

連立方程式は $Q_1 + Q_2 + \cdots + Q_n = 0$

$$
\begin{aligned}
F(y_1) &= Q_0 + Q_1 \log|y_1 - x_1| + \ldots + Q_n \log|y_1 - x_n| = f(y_1) \\
F(y_2) &= Q_0 + Q_1 \log|y_2 - x_1| + \ldots + Q_n \log|y_2 - x_n| = f(y_2) \\
&\vdots \\
F(y_n) &= Q_0 + Q_1 \log|y_n - x_1| + \ldots + Q_n \log|y_n - x_n| = f(y_n)
\end{aligned}
$$

すなわち、

$$
\begin{bmatrix}
0 & 1 & \cdots & 1 \\
1 & \log|y_1 - x_1| & \cdots & \log|y_1 - x_n| \\
\vdots & \vdots & & \vdots \\
1 & \log|y_n - x_1| & \cdots & \log|y_n - x_n|
\end{bmatrix}
\begin{bmatrix} Q_0 \\ Q_1 \\ \vdots \\ Q_n \end{bmatrix}
=
\begin{bmatrix} 0 \\ f(y_1) \\ \vdots \\ f(y_n) \end{bmatrix}
$$

係数行列を計算する

```
function A = Amat(x,y)
//Amat 代用電荷法の係数行列を計算する
//x 電荷点(特異点)座標、y 拘束点(補間点)座標
//A 係数行列
A = [0,ones(x); ones(y)',log(abs(ones(y)'*x-y.'*ones(x)))];
endfunction
```

```
A=[0              ones(x)
   ones(y)'   log(abs(ones(y)'*x-y.'*ones(x))))]
```

なので、

$$A=\begin{bmatrix} 0 & \mathbf{ones}(\boldsymbol{x}) \\ \mathbf{ones}(\boldsymbol{y}^{\mathrm{t}}) & \log|\mathbf{ones}(\boldsymbol{y}^{\mathrm{t}})\boldsymbol{x}-\boldsymbol{y}^{\mathrm{t}}\mathbf{ones}(\boldsymbol{x})| \end{bmatrix}$$

$$=\begin{bmatrix} 0 & [1,\ldots,1] \\ \begin{bmatrix}1\\ \vdots\\ 1\end{bmatrix} & \log\left|\begin{bmatrix}x_1 & \cdots & x_n\\ \vdots & \ddots & \vdots\\ x_1 & \cdots & x_n\end{bmatrix}-\begin{bmatrix}y_1 & \cdots & y_1\\ \vdots & \ddots & \vdots\\ y_n & \cdots & y_n\end{bmatrix}\right| \end{bmatrix}$$

$f(z) \sim F(z) =$

連立方程式は Q

$F(y_1) = Q_0 + Q$

$F(y_2) = Q_0 + Q$

$\vdots$

$F(y_n) = Q_0 + Q$

すなわち、

$$\begin{bmatrix} 0 & 1 & & \\ 1 & \log|y_1 - x_1| & \cdots & \log|y_1 - x_n| \\ \vdots & \vdots & & \vdots \\ 1 & \log|y_n - x_1| & \cdots & \log|y_n - x_n| \end{bmatrix} \begin{bmatrix} Q_1 \\ \vdots \\ Q_n \end{bmatrix} = \begin{bmatrix} f(y_1) \\ \vdots \\ f(y_n) \end{bmatrix}$$

```
function w = F(z,x,q)
//F 代用電荷法の近似関数
//z 引数、x 電荷点座標、q 電荷
// x = [x1,x2,...], q=[q0,q1,...] のように1×n行列で格納される。
//w 近似関数値
q0 = q(1); //定数関数の係数
qq = q(2:size(q,'*')); //定数関数以外の基底関数の係数
w = q0*ones(z); //返値が引数の行列と同サイズになるように代入
for i = 1:size(x,'*')
  w = w + qq(i)*log(abs(z-x(i)*ones(z)));
end
endfunction
```

求めたQ0,Q1～Qnを使って近似関数を計算する。

ones(z) は zと 同サイズで全成分が1の行列を表す。
z が複数の座標を含む行列であった場合に対応している。

$$f(z) \approx F(z) = Q_0 + Q_1 \log|z - x_1| + \cdots + Q_n \log|z - x_n|$$

```
function w = Bfunc(z)
//Bfunc 境界値関数
//z 引数(複素平面座標)
//w 境界値
w = zeros(z);
// zの実部が0の成分だけを再計算
w(real(z)==0)=imag(z(real(z)==0)).*(1-imag(z(real(z)==0)));
endfunction
```

実際の境界関数を与える関数。
←例では複素座標の虚部=0の場合のみ、x(1-x)を値として返しています。

行列を要素に持つ条件式は真偽値を成分とする同サイズの行列を返す。

a が [1,2,3,4,5] のとき
a>3 は [F,F,F,T,T] となる

真偽値行列を引数とする行列は、真値を持つ要素からなる部分行列を返す。

a(a>3) は [4,5] となり、
a(a>3)=[3,3] とすれば、
a ha [1,2,3,3,3] となる

つまり、次の代入はzの実部が0であるようなwの部分行列の値を再計算して代入することになる。

```
w(real(z)==0)=imag(z(real(z)==0)).*(1-imag(z(real(z)==0)));
```

```
//メインルーチン
[x,y] = SetupSquare(0,1+%i,16,16);
b = [0; Bfunc(y).'];
A = Amat(x,y);
q = A\b;
//縦横n等分した点での値を用いてグラフ表示する
   (grayplotの座標のとりかたに注意)
n=1000;
z =  ones(n+1,1)*linspace(0,1,n+1)
      +linspace(0,1,n+1).'*ones(1,n+1)*%i;
w = F(z,x,q);
grayplot(linspace(0,1,n+1),linspace(0,1,n+1),w');
```

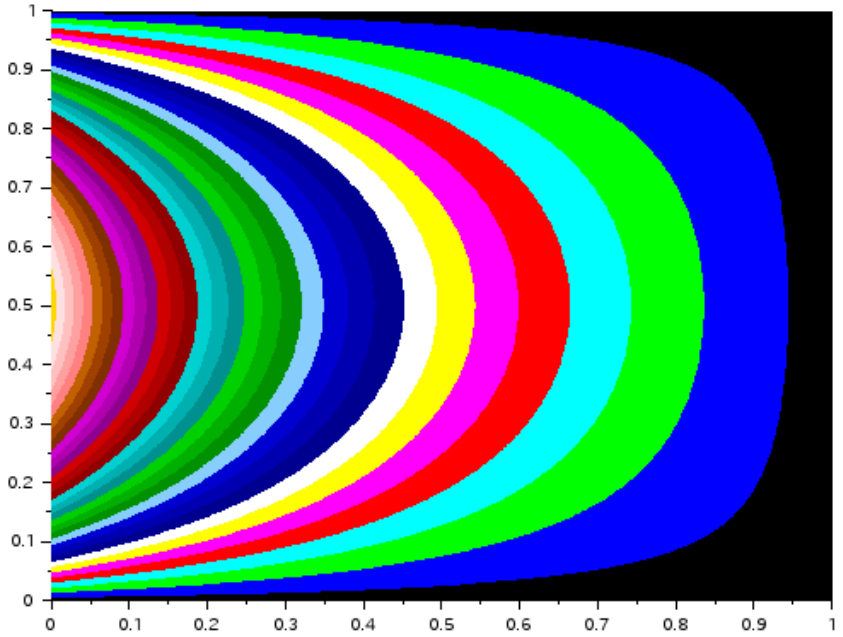
1辺の長さが1の正方形を縦横で16分割して拘束点を置き、境界拡大法で電荷点を置く。

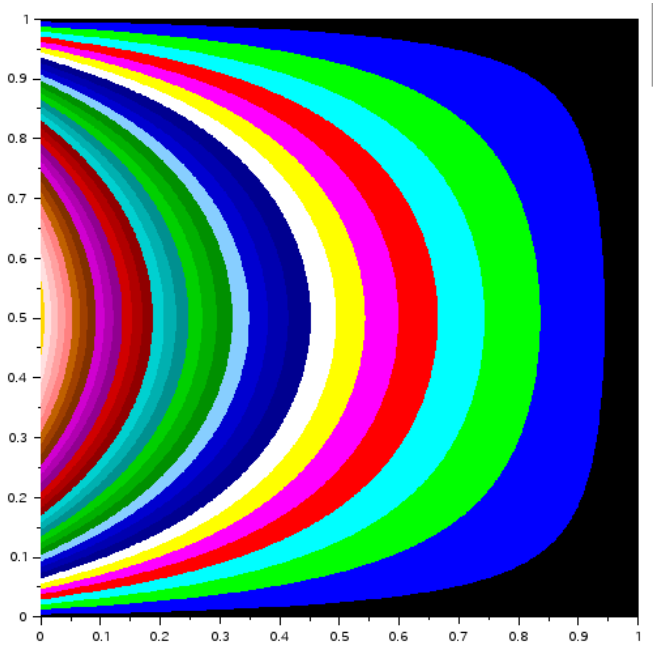
連立方程式 Aq=b の右辺 b と係数行列 A を代入し、qを求める。

1000×1000の点で近似関数値を求め、grayplotでグラフを表示する。

代用電荷法の近似関数は電荷点･拘束点の数や位置に関わらず領域内の任意の点での近似値を与える。

サンプルプログラムをそのまま実行すると、次のようなgraymapが現われます。
（右は縦横比を同じにしたもの、square(0,0,1,1)とした場合。）





差分法･有限要素法の演習で用いたものと異なる条件が使われています。
向きを揃えるために、代用電荷法のプログラムを以下のように変更します。

```
function w = Bfunc(z)
//Bfunc 境界値関数
// zの実部が0の成分だけを再計算
w(real(z)==0) = real(z(real(z)==0)).*(1-real(z(real(z)==0)));
endfunction
```

```
// zの虚部が0の成分だけを再計算
w(imag(z)==0) = real(z(imag(z)==0)).*(1-real(z(imag(z)==0)));
```

計算結果を図に示すだけでなく、誤差を調べて近似精度を評価しましょう。
一番簡単な方法は、あらかじめ厳密解を知っておき、これと比較することです。

```
function w = Efunc(z)
//Efunc 厳密解(調和関数)
//z 引数(複素平面座標)
//w 関数値
w = real(z)+imag(z);
//w = real(sin(z));
endfunction
```

厳密解を与える関数。
Laplace方程式の解になるものを
定義しておく。

$$\left[\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right](x+y)=0$$

適当な解析関数の実部等で良い。

境界条件は厳密解を利用して計算すればよい。

```
function w = Bfunc(z)
//Bfunc 境界値関数
//z 引数(複素平面座標)
//w 境界値
w = Efunc(z);
endfunction
```

zは境界点列座標の行列で与えられる。
Efunc()はzが行列であっても正しく動作
するように用意する必要がある。

グラフを表示するのと同様に厳密解Efunc()と近似解F()の差を調べましょう。

```
//メインルーチン
[x,y] = SetupSquare(0,1+%i,16,16);
b = [0; Bfunc(y).'];
A = Amat(x,y);
q = A\b;
//縦横n等分した点での値を用いてグラフ表示する
   (grayplotの座標のとりかたに注意)
n=1000;
z =  ones(n+1,1)*linspace(0,1,n+1)
      +linspace(0,1,n+1).'*ones(1,n+1)*%i;
w = F(z,x,q);
grayplot(linspace(0,1,n+1),linspace(0,1,n+1),w');
```

1辺の長さが1の正方形を縦横で16分割して拘束点を置き、境界拡大法で電荷点を置く。

連立方程式 Aq=b の右辺 b と係数行列 A を代入し、qを求める。

1000×1000の点で近似関数値を求め、grayplotでグラフを表示する。

```
//近似関数値の最大誤差を表示
write(%io(2),max(abs(F(z,x,q)-Efunc(z))));
```

レポート(13)

学籍番号・氏名を記し提出してください。

- Scilabによる代用電荷法演習を実施して、サンプルスクリプトの問題設定方法を理解してください。

  プログラム中で解く連立一次方程式のサイズが、前々回の差分法演習や前回の有限要素法演習のサンプルプログラムと同程度になるようなパラメタ設定をして、その結果を比較してください。

授業レポート用紙:氏名(　　)学籍番号(　　)

2020年2月3日(月)

できれば授業の感想も書いてください。