**プログラミング演習 (2023)**

2023.04.28 田中

**0.0. この演習の目的**

Matlabを使って波動方程式を差分法で解き，結果を可視化する.

注：本資料はmatlab\_2022bを使っている人が書いています．また，素人が書いているので誤りが豊富に含まれている可能性があります．計算コードも最適なものではなく，あくまでも一例です．

**0.1. Matlabの基本的な使い方**

Matlabでプログラミングする方法は主に下記の二つです.

・コマンドウィンドウに直接コードを書く.

・スプリクトファイルにコードを書き込む.

将来的には後者を使うことが多いと思います.

基本的な操作は[このサイト](https://toarusw.com/matlab/matlab-basic)を見ればほぼ問題ないと思います．

他の操作もインターネットで「matlab ＊＊」と検索すればたいてい出てきます．

**1.0. 1階常微分方程式**

まずは，常微分方程式

を，条件

で

の区間で解きます．

これをコンピューターで(Matlabを使って)解くためには，微分方程式を”離散化”する必要があります．離散化について次の節で説明します．

**1.1. 微分方程式の離散化 (差分法)**

コンピューターは離散的なデータしか記憶できません．微分方程式を数値的に解くとき，記憶させるのは，飛び飛びの座標 と，その座標における変数の値 です．間の座標における値が必要な場合は適当な関数で補間します．

微分方程式を離散化する方法はいくつか(有限要素法，有限体積法など)ありますが，ここでは差分法を紹介します．(他の手法でも本質的にはほぼ変わりません．)

差分法(１次精度)では１階微分を

のように近似します．

そのために，まず，区間 を 等分して

としてみます．このとき差分法で近似した方程式は

となります．例えば左の式を使うならば，

というように の値を使ってお隣の の値を求めることができます．

実際に計算しましょう．Matlabでコードを書いてみると以下のような感じになります．

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

% ←これを書いた後ろ(緑字部分)は実行されない

% プログラムが開始したら画面に「始まった!!」と出す

disp('始まった!!');

%//////////////////////////////////////////////////////////

% 前世の記憶を削除(これまでの計算が影響しないようにワークスペースを初期化)

%//////////////////////////////////////////////////////////

clear;

%//////////////////////////////////////////////////////////

% 配列(記憶してほしい値の入れ物)を用意する(用意しなくてもできる)

%//////////////////////////////////////////////////////////

N=100; % 区間の分割数

dx=1/N; % 区間の分割幅

% 101×1の配列(中身はすべて0)を定義している

x=zeros(101,1); % xの配列

y=zeros(101,1); % yの配列

% 配列の要素には１から順番に番号がついている

%//////////////////////////////////////////////////////////

% xとy(x=0)の値を配列に格納する

%//////////////////////////////////////////////////////////

% for文----------------------------------------------------

% i=1~N+1までforとendの間に書いた処理を繰り返す

%----------------------------------------------------------

for i=1:N+1

x(i)=(i-1)\*dx; %さっき用意した配列xのi番目に(i-1)\*dxが格納される

end

y(1)=1;

%//////////////////////////////////////////////////////////

% 差分法で微分方程式を解く

%//////////////////////////////////////////////////////////

for i=2:N+1

y(i)=(1-dx)\*y(i-1);

end

%//////////////////////////////////////////////////////////

% 結果をグラフにする

%//////////////////////////////////////////////////////////

% 差分法で計算した結果をプロット

plot(x(:),y(:),'-','Color','r','LineWidth',1);

% plotコマンド-------------------------------------------------

% x(:)で配列xすべてをx軸に指定

% x(1:50)などとすれば一部だけ指定することも可能

% '-'で線の形を実線に指定，'Color','r'で色を赤に指定，

% 'LineWidth',1で線の太さを1に指定

% 他にもいろいろオプションがある

%--------------------------------------------------------------

% 解析解もプロットしてみる

for i=1:N+1

z(i)=exp(-x(i));

% z配列を先に宣言しなかったのでこの処理をするたびにzのサイズが変わる

end

hold on; % hold on で同じグラフにプロット

plot(x(:),z(:),'-.','Color','b','LineWidth',1);

legend('est','exa'); %凡例をplotした順につける

% x=1での計算誤差を画面に表示させてみる

error=y(101)-z(101);

disp('x=1での誤差は');

disp(error);

% プログラムが終了したら画面に「終わった!!」と出す

disp('終わった!!');

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

これを実行するとコマンドウィンドウには以下のように出力されます．

背景パターン

低い精度で自動的に生成された説明

また，Figure1というウィンドウに以下のようなグラフが表示されます．

グラフィカル ユーザー インターフェイス, グラフ

自動的に生成された説明

**2.0. 2階常微分方程式(その1)**

を，条件

で

の区間で解きます．

この場合は，連立微分方程式

を解けばよいので，[1.2．](#微分方程式離散化（差分法）)と同じく1階微分を差分法で近似して解くことができます．

コードを書いてみると以下のような感じになります．(分割数を3パターン用意して計算しています．)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

% プログラムが開始したら画面にメッセージを出す

disp('2階常微分方程式(その1)が始まった!!');

clear;

% 配列はforループの前に宣言．ループ内で毎回サイズが変わると計算時間が増える．

x=zeros(41,3);

y=zeros(41,3);

z=zeros(41,3);

for j=1:3

N=10\*2^(j-1); % 分割数を10,20,40と変えてみる

dx=2\*pi/N; % 区間の分割幅

% xとy(x=0),z(x=0)の値を配列に格納する

for i=1:N+1

x(i,j)=(i-1)\*dx;

end

y(1,j)=1;

z(1,j)=0;

% 差分法で微分方程式を解く

for i=1:N

y(i+1,j)=y(i,j)+dx\*z(i,j);

z(i+1,j)=z(i,j)-dx\*y(i+1,j);

end

% 計算結果をプロット

if j==1

　　　　　　clf('reset'); % figureをリセット

plot(x(1:N+1,j),y(1:N+1,j),'-','Color','r','LineWidth',1.5);

hold on;

elseif j==2

plot(x(1:N+1,j),y(1:N+1,j),'-','Color','g','LineWidth',1.5);

else

plot(x(1:N+1,j),y(1:N+1,j),'-','Color','b','LineWidth',1.5);

end

end

% 解析解も

x\_exa=zeros(101,1);

y\_exa=zeros(101,1);

for i=1:101

x\_exa(i)=(i-1)\*2\*pi/100;

y\_exa(i)=cos(x\_exa(i));

end

plot(x\_exa(:),y\_exa(:),'-.','Color','k','LineWidth',1.5);

% グラフの表示設定

legend('N=10','N=20','N=40','exa','Location','best');

xlim([0 2\*pi]); % x軸の表示範囲

ylim([-1.1 1.1]); % y軸の表示範囲

% プログラムが終了したら画面にメッセージと出す

disp('終わった!!');

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

グラフは以下のようなものが出力されます．

グラフィカル ユーザー インターフェイス, ヒストグラム

自動的に生成された説明

当然分割数 を大きくするほど計算量は増えますが，計算結果が解析解に近づきます．

**2.1. 2階常微分方程式(その2)**

を，条件

で

の区間で解きます．

この場合は差分法で2階微分を

のように近似します．

これを で連立すると，

となります．この連立方程式を自力で解いても良いですが，今回はMatlabに解いてもらうことにします．

コードを書いてみると以下のような感じになります．

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

% プログラムが開始したら画面にメッセージを出す

% （プログラム要所でメッセージを出せばバグがあったときに分かりやすい）

disp('2階常微分方程式(その2)が始まった!!');

clear;

% 配列はforループの前に宣言．ループ内で毎回サイズが変わると計算時間が増える．

x=zeros(41,3);

y=zeros(41,3);

for j=1:3

N=5\*2^(j-1); % 分割数を5,10,20と変えてみる

dx=pi/2/N; % 区間の分割幅

A=zeros(N-1,N-1); % 係数行列を宣言

b=zeros(N-1,1); % 右辺ベクトルを宣言

% xとy(x=0),y(x=1)の値を配列に格納する

for i=1:N+1

x(i,j)=(i-1)\*dx;

end

y(1,j)=1;

y(N+1,j)=1;

% 係数行列，右辺ベクトルの値を格納する

A(1,1)=2-dx^2; A(1,2)=-1; b(1)=y(1,j);

for i=2:N-2

A(i,i-1)=-1; A(i,i)=2-dx^2; A(i,i+1)=-1;

end

A(N-1,N-2)=-1; A(N-1,N-1)=2-dx^2; b(N-1)=y(N+1,j);

% Matlabに連立方程式を解かせる(部分ピボットを使ったLU分解で解いている)

y(2:N,j)=linsolve(A,b);

% LU分解については[こちら](https://cattech-lab.com/science-tools/lecture-mini-lu-decomposition/)を参照

% 部分ピボットを使ったLU分解については[こちら](https://cattech-lab.com/science-tools/lecture-mini-pivotting-lu/)を参照

% 計算結果をプロット

if j==1

clf('reset'); %figureをリセット

plot(x(1:N+1,j),y(1:N+1,j),'-','Color','r','LineWidth',1.5);

hold on;

elseif j==2

plot(x(1:N+1,j),y(1:N+1,j),'-','Color','g','LineWidth',1.5);

else

plot(x(1:N+1,j),y(1:N+1,j),'-','Color','b','LineWidth',1.5);

end

end

% 解析解も

x\_exa=zeros(101,1);

y\_exa=zeros(101,1);

for i=1:101

x\_exa(i)=(i-1)\*pi/2/100;

y\_exa(i)=sin(x\_exa(i))+cos(x\_exa(i));

end

plot(x\_exa(:),y\_exa(:),'-.','Color','k','LineWidth',1.5);

% % グラフの表示設定

legend('N=5','N=10','N=20','exa','Location','best');

xlim([0 pi/2]); % x軸の表示範囲

ylim([1 1.45]); % y軸の表示範囲

xticks([0 pi/8 pi/4 pi\*3/8 pi/2]); % x軸の目盛

xticklabels({'0','\pi/8','\pi/4','3\pi/8','\pi/2'}); % x軸の目盛

% プログラムが終了したら画面にメッセージと出す

disp('終わった!!');

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

グラフは以下のようなものが出力されます．

グラフ

自動的に生成された説明

**3.0. 線形移流方程式(陽解法)**

1次元線形移流方程式

を初期条件

と境界条件

で区間

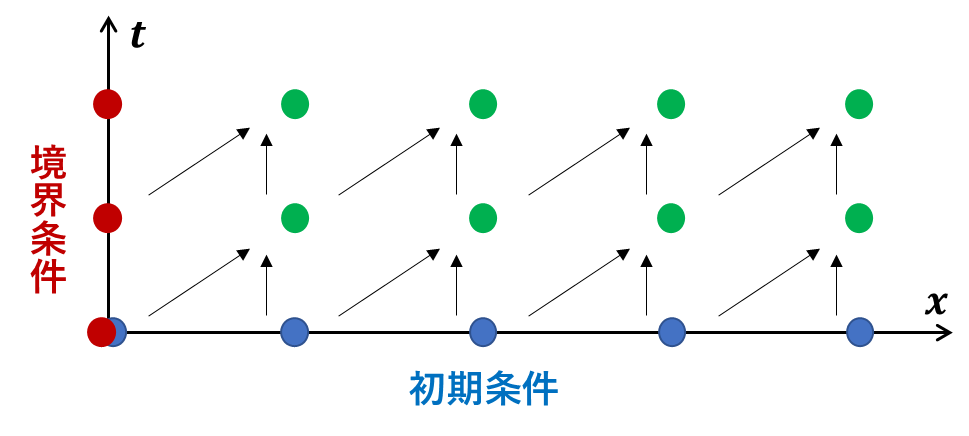
で解きます．

まずは時間と空間を離散化します．

そして，差分近似を以下のようにとってみます．

差分近似した方程式は

となります．時刻の値を時刻の値だけで計算することができます(←陽解法と言う)．

これは図で考えてみると以下のようになります．

初期値 (と境界条件 )を使って を計算し， と境界条件 を使って を計算できます．

差分近似の取り方によっては計算ができないので注意が必要です．

コードを書いてみると以下のような感じになります．

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

% プログラムが開始したら画面にメッセージを出す

disp('線形移流方程式(陽解法)が始まった!!');

clear;

% 配列の準備．(以下は3つの異なるdtで計算する場合)

x=zeros(11,1);

u\_old=zeros(11,3); %古い時間ステップでの値

u\_new=zeros(11,3); %新しい時間ステップでの値

% パラメーター設定

c=1;

N=10; % 空間分割数

step\_tot=5; % 時間ステップ総数

dx=1/N;

% x座標

for i=1:11

x(i,1)=(i-1)\*dx;

end

% 初期値

u\_old(1,:)=1; % 境界条件

u\_old(2:11,:)=0;

for j=1:3

dt=dx/c/2\*j; % dt=0.5\*dx/c,dx/c,1.5\*dx/cの3パターン

for t=1:step\_tot

u\_new(1,j)=1; % 境界条件

for i=2:11

% 差分方程式からu\_newを計算

u\_new(i,j)=(1-c\*dt/dx)\*u\_old(i,j)+c\*dt/dx\*u\_old(i-1,j);

end

if t==1

clf('reset');

fig=figure; % Figureを生成

hold off;

plot(x(:,1),u\_old(:,j),'-','LineWidth',1.5); % 初期値をプロット

hold on;

plot(x(:,1),u\_new(:,j),'-','LineWidth',1.5); % 直近の結果をプロット

else

plot(x(:,1),u\_new(:,j),'-','LineWidth',1.5); % 直近の結果をプロット

end

u\_old(:,j)=u\_new(:,j); % 次の時間ステップのためにu\_oldを更新

end

legend('t=0','t=dt','t=2dt','t=3dt','t=4dt','t=5dt','Location','best');

% FigureをJPEGで保存

　　　% 保存したいフォルダー

dname='保存したいフォルダーのパス(C:\Users\tanaka\Documents\とか)';

　　　% 絶対パス込みのファイル名

fname=append(dname,'好きな名前',sprintf('%i',j),'.jpg');

　　　% JPEGで保存

print(fig,'-djpeg',fname,'-r600');

end

% プログラムが終了したら画面にメッセージと出す

disp('終わった!!');

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

グラフは以下のようなものが出力されます．

グラフ

自動的に生成された説明

グラフ

自動的に生成された説明

グラフ, 折れ線グラフ

自動的に生成された説明

**3.1. 線形移流方程式(陰解法)**

1次元線形移流方程式

を初期条件

と境界条件

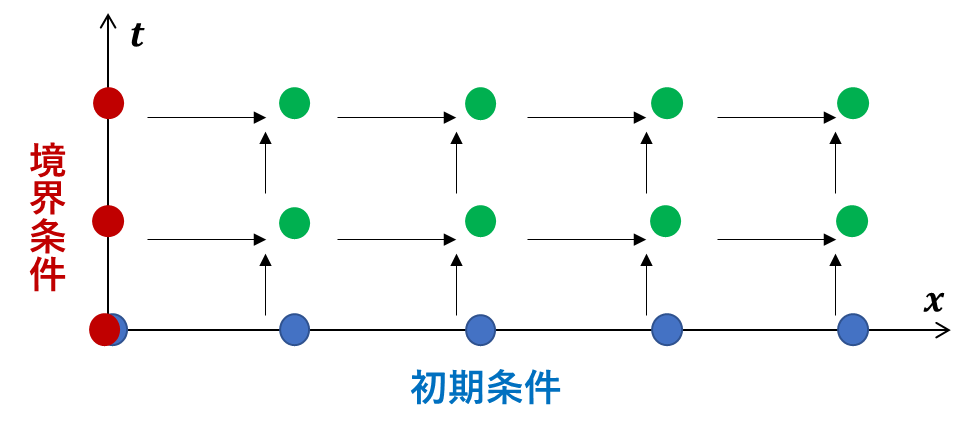
で区間

で陰解法で解きます．

差分近似を以下のようにとってみます．

差分近似した方程式は

となります．時刻の値を時刻 での値も使って計算します(←陰解法と言う)．

これは図で考えてみると以下のようになります．

初期値 と境界条件 を使って を計算，初期値 と 件を使って を，・・・・・・，境界条件 と を使って を計算，・・・・・・，としていきます．陽解法の場合と異なり，が計算できるまでは は計算できません．

コードを書いてみると以下のような感じになります．