数値計算/シミュレーションハンドブック

Ver. 1.0.0

勉強会参加メンバー

2016 年 8 月 21 日

目次

第 1 章

ルール及び事前準備

1.1 ルール

短いコードなら pdf に書き込み可.

1.2 索引の書き方

索引は index で可能.

1.3 C/C++ から gnuplot を呼び出す

C/C++ でプログラムを組んでいる場合に，ソースコード内に gnuplot を呼び出すコマンドを記述しておけば，いちいち実行とは別に gnuplot を開いて実行データのプロットを行う手間が省ける.

1.3.1 Linux での方法

Linux で作成中の場合，`popen` 関数[*1]で gnuplot を呼び出すことができる.

ソースコード 1.1　C++ のコード内から gnuplot を呼び出し $\sin x$ を描く

```
#include <iostream>
#include <cstdio>
using namespace std;

int main()
{
	FILE *fp = popen("gnuplot", "w");
	if (fp == NULL){
		return 1;
	}
	fputs("set mouse\n", fp);
	fputs("plot sin(x)\n", fp);
	fflush(fp);
	cin.get();
	pclose(fp);
	return 0;
}
```

*1 外部コマンドをプログラム内で使用できる一つの関数．終わりには `pclose` 関数で閉じないといけない.

説明

1-3 行目: `iostream` は読み込みや書き出しを行う関数が入っているライブラリ．C 言語での `stdio.h` に対応している．また，`cstdio` は FILE ポインタや `fputs` 関数などが入っているライブラリである．3 行目は名前空間の宣言であるが無視して構わない．

7 行目: FILE 型で fp というファイルを用意する．そこに，`popen` 関数を用いて gnuplot と書き込む (`"w"`)．記号 `*` はポインタを表すが別の記事を参照せよ．

8-17 行目: もし fp が確保できないならそこでお終い．確保できるなら，`fputs` 関数で ( ) 内の文字列を fp に書き出す．一つ目はマウスで操作できるようにする宣言文で，二つ目は $\sin x$ をプロットする宣言文である．`fflush` 関数により FILE ポインタ fp のバッファに格納されているデータを吐き出させる．最後に `pclose` で fp を閉じれば良い．

1.3.2 Windows での場合

Windows (Visual C++) を使う場合は，マウスが自動的に有効になっているので宣言する必要はない．また，`popen` 関数と `pclos` 関数の代わりに `_popen` 関数と `_pclose` 関数を使う．

ソースコード 1.2　Visual C++ のコード内から gnuplot を呼び出し $\sin x$ を描く

```
1  #include <iostream>
2  #include <cstdio>
3  using namespace std;
4
5  int main()
6  {
7          FILE *fp = _popen("pgnuplot.exe", "w");
8          if (fp == NULL){
9                  return 1;
10         }else{
11                 fputs("plot sin(x)\n", fp);
12                 fflush(fp);
13                 cin.get();
14                 _pclose(fp);
15                 return 0;
16         }
17 }
```

第 2 章

常微分方程式の数値計算

偏微分方程式の数値計算において

2.1 問題設定

常微分方程式

$$\frac{du(t)}{dt} = f(t, u(t)) \quad (t > 0)$$

について考えよう．

2.1.1 問題設定

問題設定

第 3 章

偏微分方程式の差分法

3.1 移流方程式

線形移流方程式

$$\partial_t u + \nu \cdot \nabla u = f \quad \text{in } (0, \infty) \times \Omega \tag{3.1}$$

を有限差分法[*1]で計算することを考える．ここで，$\nu \in \mathbb{R}^d$ はゼロでない定数ベクトルである．

3.1.1 解の存在など

3.1.2 ソースコード

3.2 Poisson 方程式

3.3 熱方程式

以下の記号を用いる．

- $\delta *$ $(* = t, x, y)$：時間及び空間刻み幅
- $(t_n, x_i, y_j) := (n\delta t, i\delta x, j\delta y)$, $(n, i, j = 0, 1, 2, \cdots)$：時空間の格子点
- $u^n_{i,j} := u(t_n, x_i, y_j)$：時空間の格子点上での未知数の値

3.3.1 1 次元の問題

次の熱方程式に対する Dirichlet 境界値問題を数値計算することを考える．

$$\begin{cases} \partial_t u(t,x) - c\partial_{xx} u(t,x) = 0 & \text{for } (t,x) \in (0,\infty) \times (0,1), \\ u(0,x) = \phi(x) & \text{for } x \in [0,1], \\ u(t,0) = u(t,1) = 0 & \text{for } t \in [0,\infty). \end{cases} \tag{3.2}$$

*1 finite difference method; FDM

3.3.2 陽解法

内部の方程式は,

$$\frac{u_i^{n+1}-u_i^n}{\delta t} - c\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\delta x)^2} = 0$$

すなわち,

$$u_i^{n+1} = \lambda u_{i+1}^n + (1-2\lambda)u_i^n + \lambda u_{i-1}^n$$

と分解される．ここで，$\lambda := c\delta t/(\delta x)^2$.

3.3.3　2 次元の問題

3.4 波動方程式

以下の記号を用いる．

- $\delta *$ $(* = t, x, y)$：時間及び空間刻み幅
- $(t_n, x_i, y_j) := (n\delta t, i\delta x, j\delta y)$, $(n, i, j = 0, 1, 2, \cdots)$：時空間の格子点
- $u_{i,j}^n := u(t_n, x_i, y_j)$：時空間の格子点上での未知数の値
-

3.4.1　1 次元の問題

次の波動方程式に対する Dirichlet 境界値問題 (固定境界値問題) を数値計算することを考える．

$$\begin{cases} \partial_{tt}u(t,x) - c^2\partial_{xx}u(t,x) = 0 & \text{for } (t,x) \in (0,\infty)\times(0,1), \\ u(0,x) = \phi(x), \quad \partial_t u(0,x) = \psi(x) & \text{for } x \in [0,1], \\ u(t,0) = u(t,1) = 0 & \text{for } t \in [0,\infty). \end{cases} \tag{3.3}$$

陽解法

内部の方程式は,

$$\frac{u_i^{n+1} - 2u_i^n + u_i^{n-1}}{(\delta t)^2} - c^2\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\delta x)^2} = 0$$

すなわち,

$$u_i^{n+1} = \lambda u_{i+1}^n + 2(1-\lambda)u_i^n + \lambda u_{i-1}^n - u_i^{n-1}$$

と分解される．ここで，$\lambda := (c\delta t/\delta x)^2$.

初期条件は，$u_i^0 = \Phi_i$ 及び

$$u(\delta t, x) \simeq u(0,x) + \delta t\partial_t u(0,x) + \frac{(\delta t)^2}{2}\partial_{tt}u(0,x)$$

より

$$u_i^1 = \Phi_i + \Psi_i\delta t + \frac{\lambda}{2}(\Phi_{i+1} - 2\Phi_i + \Phi_{i-1})$$

で決まる．ここで，$\Phi_i := \phi(x_i)$ 及び $\Psi_i := \psi(x_i)$ である．なお，境界条件は $u_0^n = u_N^n = 0$ で決まる．

ソースコード 3.1　C++ のコード内から gnuplot を呼び出し $\sin x$ を描く

```
//
// main_wave.cpp
// 1d wave_equation: u_tt - c^2 u_xx=0 (c:const.)
//
// Created by 難波時永 on 5/28/16.
```

```
// Copyright (c) 2016 難波時永. All rights reserved.
//

# include <cstdlib>
# include <iostream>
# include <iomanip>
# include <fstream>
# include <ctime>
# include <cmath>
# include <cstring>

using namespace std;

# include "fd1d_wave.hpp"


void main()
{

}

//compute lambda
double fd1d_wave_alpha ( int x_num, double x1, double x2, int t_num, double t1, double t2, double c )
{
        double delta_t;
        double delta_x;
        double lambda;

        delta_t = ( t2 - t1 ) / ( double ) ( t_num - 1 );
        delta_x = ( x2 - x1 ) / ( double ) ( x_num - 1 );
        alpha = c * delta_t / delta_x;

        cout << "\n";
        cout << "Stability condition LAMBDA = C * DT / DX = " << lambda << "\n";

        if ( 1.0 < r8_abs ( alpha ) ){
        cerr << "\n";
        cerr << "FD1D_WAVE_ALPHA - Warning!\n";
        cerr << "The stability condition |ALPHA| <= 1 fails.\n";
        cerr << "Computed results are liable to be inaccurate.\n";
        }

        return alpha;
}

//first step: solve wave eq.
double *fd1d_wave_start ( int x_num, double x_vec[], double t, double t_delta, double alpha,
                                        double u_x1 ( double t ), double u_x2 ( double t ),
                                        double *ut_t1 ( int x_num, double x_vec[] ), double u1[] )
{
        int j;
        double *u2;
        double *ut;

        ut = ut_t1 ( x_num, x_vec );

        u2 = new double[x_num];

        u2[0] = u_x1 ( t );

        for ( j = 1; j < x_num - 1; j++ )
        {
        u2[j] = alpha * alpha * u1[j+1] / 2.0
                + ( 1.0 - alpha * alpha ) * u1[j]
```

```
                + alpha * alpha * u1[j-1] / 2.0
                + t_delta * ut[j];
        }

        u2[x_num-1] = u_x2 ( t );

        delete [] ut;

        return u2;
}

//computes steps
double *fd1d_wave_step ( int x_num, double t, double alpha,
  double u_x1 ( double t ), double u_x2 ( double t ), double u1[], double u2[] )
{
        int j;
        double *u3;

        u3 = new double[x_num];

        u3[0] = u_x1 ( t );

        for ( j = 1; j < x_num - 1; j++ ){
        u3[j] = alpha * alpha * u2[j+1]
                        + 2.0 * ( 1.0 - alpha * alpha ) * u2[j]
                + alpha * alpha * u2[j-1]
                - u1[j];
        }

        u3[x_num-1] = u_x2 ( t );

        return u3;
}

//evaluate a piecewise linear spline
double *piecewise_linear ( int nd, double xd[], double yd[], int nv, double xv[] )
{
        int id;
        int iv;
        double *yv;

        yv = new double[nv];

        for ( iv = 0; iv < nv; iv++ ){
        if ( xv[iv] < xd[0] ){
                yv[iv] = yd[0];
        }
        else if ( xd[nd-1] < xv[iv] ){
                        yv[iv] = yd[nd-1];
        }
        else{
                for ( id = 1; id < nd; id++ ){
                        if ( xv[iv] < xd[id] ){
                yv[iv] = ( ( xd[id] - xv[iv] ) * yd[id-1]
                           + ( xv[iv] - xd[id-1] ) * yd[id] )
                           / ( xd[id] - xd[id-1] );
                        break;
                        }
                }
        }
        }
        return yv;
}
```

```
//return the absolute vale of an R8
double r8_abs ( double x )
{
        double value;

        if ( 0.0 <= x ){
            value = + x;
        }
        else{
        value = - x;
        }
        return value;
}

//writes an R8MAT file
void r8mat_write ( string output_filename, int m, int n, double table[] )
{
        int i;
        int j;
        ofstream output;
//
// Open the file.
//
        output.open ( output_filename.c_str ( ) );

        if ( !output ){
        cerr << "\n";
        cerr << "R8MAT_WRITE - Fatal error!\n";
        cerr << "Could not open the output file.\n";
        exit ( 1 );
        }
//
// Write the data.
//
        for ( j = 0; j < n; j++ ){
                for ( i = 0; i < m; i++ ){
                output << "  " << setw(24) << setprecision(16) << table[i+j*m];
        }
        output << "\n";
        }
//
// Close the file.
//
        output.close ( );

        return;
}

//creates a vector of linearly spaced values
double *r8vec_linspace_new ( int n, double a_first, double a_last )
{
        double *a;
        int i;

        a = new double[n];

        if ( n == 1 ){
        a[0] = ( a_first + a_last ) / 2.0;
    }
        else{
        for ( i = 0; i < n; i++ ){
                    a[i] = ( ( double ) ( n - 1 - i ) * a_first
                     + ( double ) ( i ) * a_last )
                / ( double ) ( n - 1 );
```

```
                }
        }
        return a;
}

//prints the current YMDHMS date as a time stamp
void timestamp ( )
{
        # define TIME_SIZE 40

        static char time_buffer[TIME_SIZE];
        const struct std::tm *tm_ptr;
        size_t len;
        std::time_t now;

        now = std::time ( NULL );
        tm_ptr = std::localtime ( &now );

        len = std::strftime ( time_buffer, TIME_SIZE, "%d %B %Y %I:%M:%S %p", tm_ptr );

        std::cout << time_buffer << "\n";

        return;
        # undef TIME_SIZE
}
```

3.4.2 2 次元の問題

正方領域上の波動方程式に対する Dirichlet 境界値問題 (固定境界値問題) を数値計算することを考える．

$$\begin{cases} \partial_{tt}u(t,x,y) - c^2\Delta u(t,x,y) = 0 & \text{for } (t,x,y) \in (0,\infty)\times(0,1)\times(0,1), \\ u(0,x,y) = \phi(x,y), \quad \partial_t u(0,x,y) = \psi(x,y) & \text{for } (x,y) \in [0,1]\times[0,1], \\ u(t,x,0) = u(t,0,y) = 0 & \text{for } (t,x,y) \in [0,\infty)\times[0,1]\times[0,1]. \end{cases} \tag{3.4}$$

3.4.3 陽解法

内部の方程式は

$$\frac{u_{i,j}^{n+1} - 2u_{i,j}^n + u_{i,j}^{n-1}}{(h_t)^2} - c^2\left(\frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{(h_x)^2} + \frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{(h_y)^2}\right) = 0$$

すなわち，

$$f$$

3.5 いくつかの応用

3.5.1 熱移流方程式

3.5.2 分散波動方程式

第 4 章

偏微分方程式の有限要素法

第 5 章

その他の偏微分方程式

5.1 Hamilton-Jacobi 方程式

5.2 自由境界問題

第 6 章

モンテカルロシミュレーション

第 7 章

セルオートマトン