惑星学実習 B: 偏微分方程式の数値解法 *

岩山 隆寛, 平田直之, 大槻圭史

神戸大学 大学院理学研究科 惑星学専攻

2017 年 12 月 1, 8, 15, 22 日

1 はじめに

地球惑星科学の諸現象は, 偏微分方程式の形で書かれることが多い. 線形の偏微分方程式であれば多くの場合, 解析的に（手計算で）解くことができるが, 非線形の偏微分方程式や, 境界条件が複雑な場合には, 解析的に解くことは困難である. このようなときには計算機を用いて, その偏微分方程式を解く. ここでは, 偏微分方程式の数値解法の入門として, 解析的に解くことができるが, 拡散方程式や波動方程式などの簡単な偏微分方程式を計算機を用いて解いてみる.

本実習では, 簡単化のために空間 1 次元とし, 以下の偏微分方程式を取り扱う.

1. 拡散方程式:

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial x^2}, \tag{1}$$

   ここで, ν は正の定数である.

2. 線形移流方程式:

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0, \tag{2}$$

   ここで, c は定数である.

* 本実習の資料の作成に際して, 名古屋工業大学 大学院工学研究科 渡邊威 准教授と神戸大学 大学院理学研究科研究員 村上真也 博士 (現 宇宙科学研究所　研究員) の協力を頂きました.

3. 線形移流拡散方程式:

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, \tag{3}$$

ここで, c は定数, ν は正の定数である.

4. Burgers 方程式:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, \tag{4}$$

ここで, ν は正の定数である.

5. 波動方程式:

$$\frac{\partial^2 u}{\partial t^2} = c^2\frac{\partial^2 u}{\partial x^2}, \tag{5}$$

ここで, c は正の定数である.

(1)〜(4) の方程式は, 流体力学の基礎方程式である (1 次元の）Navier–Stokes 方程式,

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\frac{\partial^2 u}{\partial x^2}, \tag{6}$$

を想定している. ここで, ν は正の定数である.

2 gnuplot による png ファイルの作成と, アニメーションの作り方

偏微分方程式の解は, 物理量の空間分布が時間発展するので, 解をアニメーションで表示すると見やすいであろう. ここでは, 与えられた関数を gnuplot を使用して png(という形式の) ファイルを作り, そのファイルから（パラパラ漫画式の）アニメーションを作成する方法を解説する. ここで解説する方法は, 計算結果からアニメーションを作成する方法にも応用可能である.

2.1 スクリプトファイルを利用した png ファイルの作成方法

gnuplot を利用して多くの図を作成するときに, gnuplot のコマンドをいちいち打つことは効率が悪い. gnuplot のコマンドをファイル（スクリプトファイルと呼ばれる）に記

述しておき, それを gnuplot で読み込んで図を作成する方法を解説する.

1. 次の内容を, 適当なエディタを使用して記述し, test.plt という名前で保存する. コマンドの意味については, 本講義の過去の資料を参考にしてほしい.

```
set xrange[0:2*pi]
set yrange[-1:1]
set xlabel 'x'
set ylabel 'u(x,t)'
plot sin(x) title 'sin(x)'
set term png
set output 'sincurve.png'
replot
```

2. gnuplot を起動して, 以下のコマンドを打ち, $\sin(x)$ が画面に表示され, さらに sincurve.png ファイルができ, そのファイルに表示された図と同じ図 (図 1 参照) が書き込まれていることを確認する.

```
gnuplot> cd 'test.plt を保存したディレクト名'
gnuplot> load 'test.plt'
```

2.2 gif アニメーションの作成方法：その 1

1. 以下の内容のスクリプトファイル (test2.plt) を作成し, gunplot で実行しなさい.[*1]

```
set xrange[0:2*pi]
set yrange[-1:1]
set xlabel 'x'
set ylabel 'u(x,t)'
set term png
set size 1, 1

set output 'sincurve_000.png'
```

[*1] eps ファイルでアニメーションを作ると, 前の時刻の画像が透けて見えるため, png ファイルでアニメーションを作ることにした.

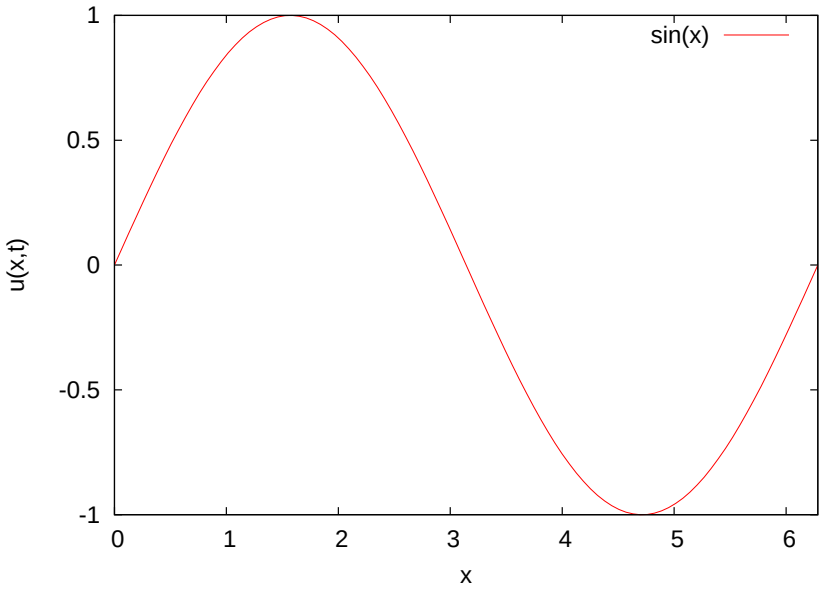

図 1　$\sin(x)$ のグラフ.

```
plot sin(x) title 't=0'
set output 'sincurve_001.png'
plot sin(x+pi/8.) title 't=1'
set output 'sincurve_002.png'
plot sin(x+2.*pi/8.) title 't=2'
       .
       .
       .
set output 'sincurve_015.png'
plot sin(x+15.*pi/8.) title 't=15'
```

2. コマンドプロンプトから ImageMagic の convert コマンドを使用して, png ファイルを結合して gif アニメーションを作成する.[*2]

```
> convert sincurve_???.png sincurve.gif
```

[*2] ?は任意の 1 つの文字列を表す.

3. sincurve.gif を適当なブラウザで開くと, ファイル名順に結合された gif アニメーションを見ることができる.

2.3 gif アニメーションの作成方法：その２

FORTRAN によって計算結果をファイルに書き出し, そのファイルに書かれているデータを gnuplot を用いて作図し, png ファイルとして保存する. さらに, convert コマンドを用いて, png ファイルを結合してアニメーションを作成する方法の例を示す.

1. 次の FORTRAN プログラム (sample.f90) を解読しなさい.[*3]

```
! サンプルプログラム
! sample.f90
! 作成者 渡邊威（名古屋工業大学 工学研究科 シミュレーション工学専攻）
! 改変: 2015.06.23, 2016.12.2, 2017.11.30 岩山隆寛
! 0 \le x \le Lx の領域を N 分割
! 0 \le t \le t_max の区間を dt 間隔で分割 j_max=t_max/dt ステップまで計算
! t_out 間隔でディスクに保存（j_out=t_out/dt ステップ毎に保存）
! c : 位相速度
! pi: 円周率
! u(x,t)=sin((x-x_0)+c(t-t_0)) のデータを書き出す
!-----------------------------------------------------------------------
      implicit none
      integer ::    i, j, k, j_max, j_out
      real(8) ::    t_0
      real(8) ::    dx, x_0
      real(8) ::    c
      integer, parameter :: N=256
      real(8), parameter ::pi=3.141592, Lx=2.D+0*pi
```

[*3] 下記のプログラムは, 区間 $0 \le x \le L_x$ を N 分割し, 時間間隔 $0 \le t \le t_{\max}$ を dt 間隔で離散化して $\sin((x-x_0)+c(t-t_0))$ を計算して $t_{\rm out}$ 毎にファイル (fort_????.dat) に書き出すプログラムである. $L_x = 2\pi$, $N = 256$, $c = 1$, $t_{\max} = 40$, $dt = 10^{-2}$, $t_{\rm out} = 0.25$, $x_0 = 0$, $t_0 = 0$ という条件を設定している. この条件だと, fort_0000.dat〜fort_0160.dat までの 161 個のファイルができる.

```
      real(8), parameter :: t_max=40.0d0, t_out=.25d+0, dt=1.D-2

      real(8) ::    x(0:N+1), t, u(0:N+1)

      dx=Lx/dble(n)

      c=1.D+0
      x_0=0.D+0
      t_0=0.D+0

      j_max=int(t_max/dt)
      j_out=int(t_out/dt)

! 初期条件
      k=0
      j=0
      t=t_0
      do i=0,n
        x(i)=dble(i)*dx
        u(i)=dsin((x(i)-x_0)+c*(t-t_0))
      enddo
! 初期条件の保存
      call save_data(k, x, t, u, n)

! 時間発展
      do j=1, j_max
        t=t+dt
        do i=0, n
          x(i)=dble(i)*dx
          u(i)=dsin((x(i)-x_0)+c*(t-t_0))
        enddo
! データの保存
        if (mod(j, j_out)==0) then
```

```
          k=k+1
          call save_data(k, x, t, u, n)
        endif

      end do

      stop

    end program
!-----------------------------------------------------------------
      subroutine counter(i, data_number)
      integer ::  i, o_4, o_3, o_2, o_1
      character(len=10) :: i_data='0123456789'
      character(4) :: data_number

       o_4=i/1000
       o_3=(i-o_4*1000)/100
       o_2=(i-o_4*1000-o_3*100)/10
       o_1= i-o_4*1000-o_3*100-o_2*10
       data_number=i_data(o_4+1:o_4+1)// &
                   i_data(o_3+1:o_3+1)// &
                   i_data(o_2+1:o_2+1)// &
                   i_data(o_1+1:o_1+1)

      return
    end subroutine counter
!-------------------------------------------------------------------------
    subroutine save_data(k, x, t, u, n)

      integer :: k, i, n
      real(8) :: x(0:n), t, u(0:n)
      character(4) :: data_number
```

```
      call counter(k, data_number)
      do i=0, n
         open(50,file='fort_'//data_number//'.dat', &
         status='unknown')
         write(50, *) x(i), t, u(i)
      enddo

    end subroutine save_data
```

2. sample.f90 を実行し, fort_0000.dat〜fort_0160.dat の 161 個のファイルができていることを確認しなさい.

```
> gfortran sample.f90
> ./a.out
```

3. gnuplot のスクリプトファイル (sample.plt)[*4]を実行して, fort_0000.dat〜fort_0160.dat のファイルの中身を作図しなさい. (mov_0000.png〜mov_0160.png ができる.)

```
gnuplot> load 'sample.plt'
```

sample.plt の中身は次のようになっている. 6 行目以降に記述されているように, gnuplot のコマンドでも FORTRAN のように DO 文を使用してある処理を自動的に繰り返し行える.

```
set xrange [0:2*pi]
set yrange [-1:1]
set term png
set size square
set xlabel 'x'
```

[*4] これは以下のディレクトリにあるので, 各自自分の作業しているディレクトリにコピーして使用してください.

```
/home3/iwayama/exp_17/pde/exp1/sample.plt
```

```
set ylabel 'u(x)'

do for [i = 0:160] {          <---ファイルの数に応じて数値を変える
  ii = sprintf("%04d", i)
  outfile = "mov_" . ii . ".png"
  infile = "fort_" . ii . ".dat"
  set output outfile
  plot infile u 1:3 w l t "t=".ii
}
```

4. convert コマンドを打って, mov_0000.png〜mov_0160.png を結合して一つのファイル move.gif を作成する.

```
> convert mov_????.png mov.gif
```

2.4 宿題

sample.f90 を diffusion.f90 にコピーし, さらにその中身を自分で改変し, 無限領域内の 1 次元拡散方程式の基本解,

$$\frac{1}{\sqrt{4\pi\nu(t+t_0)}}\exp\left\{-\frac{(x-x_0)^2}{4\nu(t+t_0)}\right\} \tag{7}$$

の発展をアニメーションで示しなさい. 拡散係数 ν や x_0, t_0, 時間間隔 $dt, t_{\rm out}$, 等は自分で適当な値を設定しなさい. sample.plt も diffusion.plt にコピーし, 縦軸や横軸の範囲を関数に最適な範囲に設定し，png ファイルの作成とアニメーション（gif ファイル）の作成を行うこと. この機会に拡散方程式, 拡散方程式の基本解の復習を行っておくとこの宿題における t_0 の適切な設定の仕方, および次週の実習の復習になります.[*5]

作図した関数に関する説明と gif ファイルを添付ファイルとして e-mail で iwayama@kobe-u.ac.jp 宛に送信してください.

[*5] $t_0 = 0$ と設定して $t = 0$ から (7) の図を書きだそうとするとエラーが吐き出されます. 何故か考えてみましょう.

3　拡散方程式の数値解法

1 次元拡散方程式,

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial x^2}, \tag{8}$$

を数値的に解く. ここで, ν は正の定数である.

3.1　解説

- 時間と空間の離散化:
  拡散方程式 (8) を解く領域は, $0 \leq x \leq L_x$ とする. この区間を N 等分する. そこで, 座標変数 x を正の整数 i を用いて,

$$x \to x_i = i\Delta x, \quad \Delta x = \frac{L_x}{N}, \tag{9}$$

  と表せる. また, 時間も数値計算では離散化して扱うので, j を正の整数として,

$$t \to t_j = j\Delta t, \tag{10}$$

  と表す. (8) の解は, x, t に依存する量なので, したがって, 数値計算では (8) の解は, i, j に依存することになる:

$$u(x,\, t) \to u(x_i,\, t_j) \tag{11}$$

- 空間微分の表現の仕方:
  (8) は時間と空間に関する微分を含むが, 数値計算では微分は近似的にしか取り扱えない. ここでは空間微分の近似の仕方について述べる.

  x_i における $\partial^2 u/\partial x^2$ を x_{i-1}, x_i, x_{i+1} の 3 点における u の値を使って表現してみる. Δx が微小のとき, $u(x_{i\pm1}, t_j)$ を Taylor 展開すると

$$u(x_{i\pm1}, t_j) = u(x_i, t_j) \pm \frac{\partial u}{\partial x}\Delta x + \frac{1}{2}\frac{\partial^2 u}{\partial x^2}(\Delta x)^2 + \mathcal{O}(\Delta x^3), \tag{12}$$

  となる. ここで, $\mathcal{O}(\Delta x^3)$ は $(\Delta x)^3$ の大きさの項を表す. したがって, 上式を $\partial^2 u/\partial x^2$ について解いて

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x_{i+1}, t_j) + u(x_{i-1}, t_j) - 2u(x_i, t_j)}{(\Delta x)^2} + \mathcal{O}(\Delta x^2) \tag{13}$$

を得る.[*6] 以上から, $(\Delta x)^2$ の精度で

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x_{i+1}, t_j) + u(x_{i-1}, t_j) - 2u(x_i, t_j)}{(\Delta x)^2} \tag{14}$$

と近似する.

- 時間微分の表現の仕方:
  時間微分に関しては, 常微分方程式の解法の際に採用した Euler 法を使用する. このとき, Δt の 1 次の精度で,

$$\frac{\partial u}{\partial t} = \frac{u(x_i, t_{j+1}) - u(x_i, t_j)}{\Delta t} \tag{15}$$

  と近似する.

- まとめ:
  以上の議論から, 拡散方程式 (8) を数値的に解く場合, 空間微分に関しては 2 次精度, 時間微分に関しては 1 次精度で

$$\frac{u(x_i, t_{j+1}) - u(x_i, t_j)}{\Delta t} = \nu \frac{u(x_{i+1}, t_j) + u(x_{i-1}, t_j) - 2u(x_i, t_j)}{(\Delta x)^2}, \tag{16}$$

  もしくは,

$$u(x_i, t_{j+1}) = u(x_i, t_j) + \nu \frac{\Delta t}{(\Delta x)^2} \{u(x_{i+1}, t_j) + u(x_{i-1}, t_j) - 2u(x_i, t_j)\} \tag{17}$$

  によって, 時刻 t_j における u の分布から, t_{j+1} における u の分布が得られる.

3.2 課題

(17) に従って, 拡散方程式を数値的に解きなさい. ここで, 境界条件と初期条件は以下のとおりとする.

境界条件： $0 \leq x \leq L_x$ の領域で拡散方程式を解く. このとき, 境界条件は周期境界条件,

$$u(0, t) = u(L_x, t), \tag{18}$$

とする.[*7]

[*6] $\mathcal{O}(\Delta x)$ 項は相殺されて, 誤差は $\mathcal{O}(\Delta x)^2$ 程度の大きさになることに注意.
[*7] 数値計算では $u(x_0, t_j) = u(x_N, t_j)$ と表現されている.

初期条件： 領域の中心を分布の中心とした Gauss 分布,

$$u(x,0) = \exp\left[-a\left(x - \frac{L_x}{2}\right)^2\right] \tag{19}$$

とする.

その他の条件： $\nu = 2 \times 10^{-3}$, $L_x = 2\pi$, $N = 256$, $\Delta t = 1.0 \times 10^{-2}$, $a = 10$ とする.

- 拡散方程式を数値シミュレーションするためのサンプルプログラム (diffusion_sample.f90) を完成させる. 時間発展の様子をアニメーションにより観察する. gnuplot のスクリプトファイルは, diffusion_sample.plt を使用しなさい.[*8]
- 結果が妥当であるかどうかを検証するために, シミュレーションで得られた結果と解析解を比較する. 解析解を計算するプログラムは, diffusion_theory.f90 に用意されている.[*9]

3.3 サンプルプログラム: diffusion_sample.f90

```
!1 次元拡散方程式の数値解法
! Euler 法
! 作成者: Takahiro IWAYAMA
! 2014.07.27
! 2015.07.07
! 2016.12.06--08
! 2017.12.2
!---------------------------------------------

      program diffusion

      implicit none
```

[*8] ダウンロードするファイルは/home3/iwayama/exp_17/pde/exp2/に置いてあるので適宜ダウンロードしてください.

[*9] 解析解の結果は theory_0000.dat〜のファイル名で保存される. gunplot で同じ瞬間の u の場を比べてみる. 例えば, fort_0150.dat の時刻の数値解に対応する解析解は theory_0150.dat である. この解析解は, 初期条件は (19) と同じであるが, 無限領域 $-\infty < x < \infty$ における拡散方程式の解析解を図示するものである. 初期の発展は, シミュレーションと解析解は一致するが長時間積分すると両者は特に領域の端からずれ始める.

```
      integer ::  i, j, k
      real(8) ::  t, dt, t_max, t_out
      real(8) ::  dx
      real(8) ::  pi, Lx, nu
      integer, parameter :: N=256
      real(8), parameter :: c=1.0d-1, a=10.d+0
      real(8) ::  x(0:N+1)  !x 座標
      real(8) ::  u(0:N+1)  !拡散方程式の解
      real(8) ::  v(0:N+1)  !拡散方程式の解
      real(8) ::  d2u(1:N)  !u の x による 2 階微分
! parameters for output
      integer ::    t_step, j_out


! 各種のパラメターの定義
      t_max=100.0d0      !t_max まで計算
      dt=1.0d-2          !時間間隔 t_k=k*dt
      t_out=1.0D+0       !t_out 間隔にデータ保存
      t_step=int(t_max/dt)
      j_out=int(t_out/dt)
      nu=2.0D-3          !拡散係数の値

      pi=acos(-1.0D+0)   !円周率
      Lx=2.0D+0*pi       !領域の広さ
      dx=Lx/dble(N)      !空間間隔

! 初期条件の設定
      k=0
      t=dble(0)
      do i=1, N
        x(i)=dble(i)*dx
        u(i)=dexp(-a*(x(i)-Lx/dble(2))**2)
      enddo
```

```
! 境界条件を課す
      call bound_cond(u,n)
! 初期値の保存
      call save_data(k, x, t, u, n)



! 時間発展の計算----------------------------------
      do j=1, t_step
        t=t+dt
! u の 2 階微分の計算
        call second_deriv(u, d2u, dx, N)
! 各格子点で拡散方程式の発展を計算
        do i=1, N
!    Euler 法による拡散方程式の公式
                                                                        !<--
この行に公式に従った計算式を書く
        enddo
! 境界条件を課す
        call bound_cond(u, N)
! データの保存
        if (mod(j, j_out)==0) then
          k=k+1
          call save_data(k, x, t, u, n)
        endif
      enddo

      stop
      end program diffusion
!-------------------------------------------------------------------------
      subroutine second_deriv(u, d2u, dx, N)
! u の x による 2 階微分
      integer ::    N, i
      real(8) ::    u(0:N+1)
```

```
      real(8) ::    d2u(1:N)
      real(8) ::    dx

      do i=1,n
        d2u(i)=(u(i+1)+u(i-1)-dble(2)*u(i))/dx**2
      end do

      return
    end subroutine second_deriv
!-----------------------------------------------------------------------
      subroutine bound_cond(u,N)
! 周期境界条件を課す
      integer ::  N
      real(8) ::  u(0:N+1)

      u(0)=u(N)
      u(N+1)=u(1)

      return
    end subroutine bound_cond
!-----------------------------------------------------------------
      subroutine counter(i, data_number)
      integer ::  i, o_4, o_3, o_2, o_1
      character(len=10) :: i_data='0123456789'
      character(4) :: data_number

       o_4=i/1000
       o_3=(i-o_4*1000)/100
       o_2=(i-o_4*1000-o_3*100)/10
       o_1= i-o_4*1000-o_3*100-o_2*10
       data_number=i_data(o_4+1:o_4+1)// &
                   i_data(o_3+1:o_3+1)// &
                   i_data(o_2+1:o_2+1)// &
```

```
                    i_data(o_1+1:o_1+1)

      return
    end subroutine counter
!---------------------------------------------------------------------------
    subroutine save_data(k, x, t, u, n)

      integer :: k, i, n
      real(8) :: x(0:n+1), t, u(0:n+1)
      character(4) :: data_number

      call counter(k, data_number)
      do i=1, n
         open(50,file='fort_'//data_number//'.dat', &
         status='unknown')
         write(50, *) x(i), t, u(i)
      enddo

    end subroutine save_data
```

3.4 宿題

拡散係数 ν を様々な値に変えて, 拡散方程式を数値的に解きなさい. 結果をアニメーションにより観察し, 結果の意味をよく吟味しなさい.

- 拡散方程式を数値的に解いた結果を図示した gif ファイルを提出しなさい.
  - 計算条件（初期条件, 境界条件, 領域の大きさ L_x, 解像度 N, Δt) の説明文も同封すること.
  - 考察も付け加えること. (ν の値をかえたらどうなったか. 次節の von Neumann の安定性条件を破る場合を計算し, 数値計算が破たんする様子を眺めてもよい.)

3.5 von Neumann の安定性解析

空間の差分の間隔 Δt, 時間の差分間隔 Δt が十分小さければ, 微分の差分による表現はより正確になっていくであろう. しかしながら, 拡散係数 ν と Δx, Δt との間に以下の不等式を満足する範囲内でしか, 拡散方程式は数値的に安定には解けない:

$$\Delta t \leq \frac{(\Delta x)^2}{2\nu}. \tag{20}$$

(20) は von Neumann の安定性条件と呼ばれる. (20) によれば, Δt, ν を固定したもとで Δx を小さくしていくと数値計算は安定に実行できないことが分かる. [*10]

[*10] $u(x_i, t_{j+1}) = \lambda u(x_i, t_j) e^{ikx_j}$ のように x 依存性は Fourier 級数で表現し, 以前の解析と同様に増幅因子 λ を導入すると, Euler 法による計算の増幅因子

$$\begin{aligned} \lambda_{\text{Euler}} &= 1 + 2\frac{\nu\Delta t}{(\Delta x)^2}(\cos(k\Delta x) - 1) \\ &= 1 - 4\frac{\nu\Delta t}{(\Delta x)^2}\sin^2\frac{k\Delta x}{2} \end{aligned} \tag{21}$$

を得る. von Neumann の安定性の条件

$$-1 \leq \lambda \leq 1 \tag{22}$$

を満たすためには, (21) の右辺第 2 項は $\nu > 0$ ならば負定値なので上限の条件は常に満たしている. さらに, 下限の条件から

$$\frac{\nu\Delta t}{(\Delta x)^2} \leq \frac{1}{2} \tag{23}$$

を満たしている必要がある. 一方, $u(x, t + \Delta t) = \lambda u(x, t) e^{ikx}$ とすると, 解析解の増幅因子は

$$\lambda_{\text{theor}} = e^{-k^2\nu\Delta t} \tag{24}$$

である. (21) と (24) が一致するためには, (21) において, $k\Delta x \ll 1$ のとき,

$$\lambda_{\text{Euler}} = 1 - \nu k^2 \Delta t + \mathcal{O}((k\Delta x)^2) \tag{25}$$

である. 一方 (24) において, $k^2\nu\Delta t \ll 1$ のとき,

$$\lambda_{\text{theor}} = 1 - \nu k^2 \Delta t + \mathcal{O}((k^2\nu\Delta t)^2) \tag{26}$$

となる. つまり主要項は両者で一致する.

4　線形移流方程式の数値解法

[*11] 1 次元線形移流方程式,

$$\frac{\partial u}{\partial t} = -c\frac{\partial u}{\partial x}, \tag{27}$$

を数値的に解く. ここで, c は定数である.

4.1　移流方程式の解の性質

移流方程式 (27) は $x - ct$ を独立変数とする任意の関数 $f(x - ct)$ が解になっているという性質がある:

$$u(x, t) = f(x - ct). \tag{28}$$

(28) は c が正のとき x の正の方向に速度 c で進行する解, c が負のとき x の負の方向に速度 c で進行する解を表す.

4.2　解説

- 時間と空間の離散化:
  移流方程式 (27) を解く領域は, $0 \leq x \leq L_x$ とする. この区間を N 等分する. そこで, 座標変数 x を正の整数 i を用いて,

$$x \to x_i = i\Delta x, \quad \Delta x = \frac{L_x}{N}, \tag{29}$$

  と表す. また, 時間も数値計算では離散化して扱うので, j を正の整数として,

$$t \to t_j = j\Delta t, \tag{30}$$

[*11] この回に必要なファイルは,

`/home3/iwayama/exp_16/pde/exp3/`

以下に用意されている.

```
> cp /home3/iwayama/exp_16/pde/exp3/a* .
```

で自分のディレクトリにコピーして作業する.

と表す. (27) の解は, x, t に依存する量なので, したがって, 数値計算では (27) の解は, i, j に依存することになる:

$$u(x,\, t) \to u(x_i,\, t_j) \tag{31}$$

- 空間微分の表現の仕方:
  (8) は時間と空間に関する微分を含むが, 数値計算では微分は近似的にしか取り扱えない. ここでは空間微分の近似の仕方について述べる.

  x_i における $\partial u/\partial x$ を $x_{i-1},\ x_i,\ x_{i+1}$ の 3 点における u の値を使って表現してみる. Δx が微小のとき, $u(x_{i\pm1}, t_j)$ を Taylor 展開すると

$$u(x_{i\pm1}, t_j) = u(x_i, t_j) \pm \frac{\partial u}{\partial x}\Delta x + \frac{1}{2}\,\frac{\partial^2 u}{\partial x^2}(\Delta x)^2 + \mathcal{O}(\Delta x^3), \tag{32}$$

  となる. ここで, $\mathcal{O}(\Delta x)^3$ は $(\Delta x)^3$ の大きさの項を表す. したがって, 上式を $\partial u/\partial x$ について解くと, 3 つの表現が得られる:

  1. 前進差分

$$\frac{\partial u}{\partial x} = \frac{u(x_{i+1}, t_j) - u(x_i, t_j)}{\Delta x} + \mathcal{O}(\Delta x). \tag{33}$$

  2. 後退差分

$$\frac{\partial u}{\partial x} = \frac{u(x_i, t_j) - u(x_{i-1}, t_j)}{\Delta x} + \mathcal{O}(\Delta x). \tag{34}$$

  3. 中央差分

$$\frac{\partial u}{\partial x} = \frac{u(x_{i+1}, t_j) - u(x_{i-1}, t_j)}{2\Delta x} + \mathcal{O}(\Delta x^2). \tag{35}$$

  上で見たように, 数値計算における微分の表現の仕方には, 素朴に 3 つの方法が考えられる. このうち, 前進差分, 後退差分は $\mathcal{O}(\Delta x)$ (Δx の 1 次) の精度であるが, 中央差分は $\mathcal{O}(\Delta x^2)$ (Δx の 2 次) の精度である.

- 時間微分の表現の仕方:
  時間微分に関しては, 常微分方程式の解法の際に採用した Euler 法を使用する. このとき, Δt の 1 次の精度で,

$$\frac{\partial u}{\partial t} = \frac{u(x_i, t_{j+1}) - u(x_i, t_j)}{\Delta t} \tag{36}$$

  と近似する.

- まとめ:

  以上の議論から, 移流方程式 (27) を数値的に解く場合, 空間微分に関しては 1 次精度, 時間微分に関しては 1 次精度で

  1. 前進差分

$$u(x_i, t_{j+1}) = u(x_i, t_j) - \frac{c\Delta t}{\Delta x}\left\{u(x_{i+1}, t_j) - u(x_i, t_j)\right\}. \tag{37}$$

  2. 後退差分

$$u(x_i, t_{j+1}) = u(x_i, t_j) - \frac{c\Delta t}{\Delta x}\left\{u(x_i, t_j) - u(x_{i-1}, t_j)\right\}. \tag{38}$$

  空間微分に関しては 2 次精度, 時間微分に関しては 1 次精度で

  3. 中央差分

$$u(x_i, t_{j+1}) = u(x_i, t_j) - \frac{c\Delta t}{2\Delta x}\left\{u(x_{i+1}, t_j) - u(x_{i-1}, t_j)\right\}. \tag{39}$$

  によって, 時刻 t_j における u の分布から, t_{j+1} における u の分布が得られる.

4.3 宿題

(37)〜(39) に従って, 移流方程式を数値的に解きなさい. ここで, 境界条件と初期条件は以下のとおりとする.

境界条件： 周期境界条件,

$$u(0, t) = u(L_x, t), \tag{40}$$

とする.[*12]

初期条件： 波長 $L_x/2$, 振幅 1 の正弦波,

$$u(x, 0) = \sin\left(\frac{4\pi x}{L_x}\right), \tag{41}$$

とする.

その他の条件： $c = 0.1$, $L_x = 2\pi$, $N = 256$, $\Delta t = 1.0 \times 10^{-2}$.

[*12] 数値計算では $u(x_0, t_j) = u(x_N, t_j)$ と表現されている.

- 移流方程式を数値シミュレーションするためのプログラム (advection.f) を前回拡散方程式を解くために使用したプログラムを参考に, サンプルプログラム (advection_sample.f) を書き換えて完成させる. 時間発展の様子をアニメーションにより観察する. gnuplot のスクリプトファイルは, advection.plt を適宜変更して[*13]使用しなさい.
- 結果が妥当であるかどうかを検証するために, シミュレーションで得られた結果と解析解を比較する. 解析解を計算するプログラムは, advection_theory.f に用意されている.
- 移流方程式を数値的に解いた結果を図示した gif ファイルを提出しなさい.
  - 計算条件 (初期条件, 境界条件, 領域の大きさ, 解像度, Δt) を明記する. (注意: プログラム上での表現でなく, 数学的な表現で書くこと.)
  - 考察も付け加えること. (次の CFL 条件, von Neumnann の安定性解析や (42) についても考慮して考察するとなおよい.)
- 後方差分を用いて移流方程式を解いたときには振幅が減衰する. 移流方程式を後方差分 (風上差分) で差分化した式は, 移流拡散方程式

$$\frac{\partial u}{\partial t} = -c\frac{\partial u}{\partial x} + \tilde{\nu}\frac{\partial^2 u}{\partial x^2} \tag{42}$$

を中央差分で差分化した式, 但し, $\tilde{\nu} = c\,\Delta x/2$, と同等な式であることが示せる. つまり方程式を差分近似するときに (42) の右辺第 2 項に相当する拡散項が入るような差分化の仕方を行っているために振幅が減少するのである.
- Courant-Friedrichs-Lewy(CFL) の条件について調べなさい.[*14]

[*13] 縦軸, 横軸, 凡例, 出力ファイル名などを現在の設定に適切なものに変更する.

[*14] CFL 条件だけを調べるのではなく, その条件がどのような議論によって導出されるのかを調べる. (依存領域, 影響領域といったキーワードを手掛かりに CFL 条件の導出について調べてみるてください.)

以下では, 移流方程式の各差分近似に対して von Neumann の安定性解析を行ってみる. $u(x_l, t_j) = \lambda^j \hat{u}_0 e^{ikx_l}, x_l = l\Delta x, t_j = j\Delta t$ を (37), (38), (39) に代入する. それぞれの差分近似の増幅因子は

$$\lambda_{\text{forward}} = 1 - \gamma\left(e^{ik\Delta x} - 1\right) = 1 + \gamma - \gamma e^{ik\Delta x}, \tag{43}$$

$$\lambda_{\text{backward}} = 1 - \gamma\left(1 - e^{-ik\Delta x}\right) = 1 - \gamma + \gamma e^{-ik\Delta x}, \tag{44}$$

$$\lambda_{\text{central}} = 1 - i\gamma \sin k\Delta x, \tag{45}$$

となる. ここで $\gamma \equiv c\Delta t/\Delta x$ は Courant 数と呼ばれる. それぞれの増幅因子の大きさの 2 乗は

$$|\lambda_{\text{forward}}|^2 = 1 + 4\gamma(1+\gamma)\sin^2 k\Delta x, \tag{46}$$

$$|\lambda_{\text{backward}}|^2 = 1 - 4\gamma(1-\gamma)\sin^2 k\Delta x \tag{47}$$

$$|\lambda_{\text{central}}|^2 = 1 + \gamma^2 \sin^2 k\Delta x, \tag{48}$$

5　波動方程式の数値解法

[*15]

1 次元波動方程式,

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}, \tag{50}$$

を数値的に解く. ここで, c は定数である[*16].

5.1　波動方程式の解の性質

波動方程式 (50) は $x - ct$ と $x + ct$ を独立変数とする任意の関数 $f(x - ct)$, $g(x + ct)$ の重ね合わせが解になっているという性質がある:

$$u(x, t) = f(x - ct) + g(x + ct). \tag{51}$$

(51) は d'Alembert 解と呼ばれ, c が正のとき $f(x - ct)$ は x の正の方向に速度 c で進行する解, $g(x + ct)$ は x の負の方向に速度 c で進行する解を表す. この解の関数形 f, g は初期条件, 境界条件によって決定される.

5.2　解説

(50) は (8) とよく似ているが, 時間に関する微分が (50) では 2 階なのに対し, (8) では 1 階である. 『常微分方程式の数値解法』で, 2 階の常微分方程式を 1 階の連立常微分方程

となる. (46), (48) は $\gamma > 0$ であれば常に $|\lambda| > 1$ である. 一方, (47) は

$$0 < \gamma \leq 1 \tag{49}$$

であれば von Neumann の安定性条件を満たす. (49) は CFL 条件と同じ条件である. 上記の解析によると, 中央差分を採用すると計算は不安定であるが, γ を小さくとっておくと, 前方差分を採用したときよりも振幅の増幅は小さく抑えられることが分かる.

[*15] この回に必要なファイルは,

```
/home3/iwayama/exp_16/pde/exp4/
```

以下に用意されている.

```
> cp /home3/iwayama/exp_16/pde/exp3/m* .
```

で自分のディレクトリにコピーして作業する.

[*16] 位相速度と呼ばれる.

式系として解くことを学んだ. ここでも, (50) を時間に関する 1 階の連立偏微分方程式系に書き直して, それを解く.

新しい変数 $v(x,t) \equiv \frac{\partial u}{\partial t}$ を導入すると, (50) は

$$\frac{\partial u}{\partial t} = v, \tag{52a}$$

$$\frac{\partial v}{\partial t} = c^2 \frac{\partial^2 u}{\partial x^2}, \tag{52b}$$

と書ける.

時間と空間の離散化の仕方は, 拡散方程式のときと同様に, 空間差分に関しては中央差分, 時間差分に関しては Euler 法を採用する. 空間微分に関しては 2 次精度, 時間微分に関しては 1 次精度で

$$\frac{u(x_i, t_{j+1}) - u(x_i, t_j)}{\Delta t} = v(x_i, t_j), \tag{53a}$$

$$\frac{v(x_i, t_{j+1}) - v(x_i, t_j)}{\Delta t} = c^2 \, \frac{u(x_{i+1}, t_j) + u(x_{i-1}, t_j) - 2u(x_i, t_j)}{(\Delta x)^2}, \tag{53b}$$

もしくは,

$$u(x_i, t_{j+1}) = u(x_i, t_j) + v(x_i, t_j)\Delta t \tag{54a}$$

$$v(x_i, t_{j+1}) = v(x_i, t_j) + c^2 \, \frac{\Delta t}{(\Delta x)^2} \left\{ u(x_{i+1}, t_j) + u(x_{i-1}, t_j) - 2u(x_i, t_j) \right\}, \tag{54b}$$

によって, 時刻 t_j における u, v の分布から, t_{j+1} における u, v の分布が得られる. このとき, 初期条件として u, v の値が必要であることに注意しなさい.[*17]

5.3 課題

(54) に従って, 波動方程式を数値的に解きなさい. ここで, 境界条件と初期条件は以下のとおりとする.

境界条件： $0 \leq x \leq L_x$ の領域で波動方程式を解く. 境界条件は開放端条件,

$$\frac{\partial u(x,t)}{\partial x} = 0, \ (x = 0, L_x) \tag{55}$$

とする.[*18]

[*17] 時間に関して 2 階の微分を含む方程式なので, 2 個初期条件が必要である.

[*18] 数値計算では $u(x_0, t_j) = u(x_1, t_j), u(x_{N+1}, t_j) = u(x_N, t_j)$ と表現されている.

初期条件： 領域の中心を分布の中心とした Gauss 分布で初速度をゼロ,

$$u(x,0) = \exp\left[-a\left(x - \frac{L_x}{2}\right)^2\right],\ v(x,0) = 0, \tag{56}$$

とする.

その他の条件： $c = 1 \times 10^{-1}$, $L_x = 2\pi$, $N = 256$, $\Delta t = 1.0 \times 10^{-2}$, $a = 10$, $0 \leq t \leq 150$ とする.

- 波動方程式を数値シミュレーションするためのサンプルプログラム (wave_sample.f) を完成させる. これは拡散方程式を解くプログラムである. これを波動方程式を解くプログラムに改変する. 時間発展の様子をアニメーションにより観察する. gnuplot のスクリプトファイルは, wave.plt を使用しなさい.[*19]

5.4 宿題

1. 課題と同じ条件で, ただし境界条件を固定端条件,

$$u(0,t) = 0,\ u(L_x,t) = 0 \tag{57}$$

として (50) のシミュレーションを行いなさい, 開放端条件のときとの違いについて考察しなさい.

2. Euler 法で数値計算を行うと, 長時間の発展ののち計算が破たんする. Adams-Bashforth 法を用いて波動方程式を解くことにより, 長時間安定に計算できることを (余力があれば) 確かめなさい.
3. (さらに余力があれば)Euler 法を採用した場合の, von Neumann の安定性解析を行いなさい.[*20]

5.5 サンプルプログラム

1 次元拡散方程式を解くプログラムを以下に掲載する. 波動方程式を解くプログラムに改変するために, 変更の必要な箇所にコメントを挿入しているので参考にしてください.

[*19] ダウンロードするファイルは/home3/iwayama/exp_16/pde/exp4/に置いてあるので適宜ダウンロードしてください.

[*20] ヒント：拡散方程式に対する von Neumann の安定性解析 3.5 節を参考にするとよい.

```
c1 次元拡散方程式の数値解法
c Euler 法
c 作成者: Takahiro IWAYAMA
c 2014.07.27
c 2015.07.07
c 2016.12.06--08
c-----------------------------------------------
c234567890123456789012345678901234567890123456789012345678901234567890123456789012

      program diffusion

      implicit none
      integer  i, j, k, N
      real*8   t, dt, t_max, t_out
      real*8   x, dx
      real*8   pi, nu, Lx, a                ! <- nu -> c
      parameter (nu=2.0d-3, a=10.d+0)       ! <- c の値
      parameter (N=256)
      real*8   u(0:N+1)  !拡散方程式の解    ! <- 新しい変数 v を宣言
      real*8   d2u(1:n)  !u の x による 2 階微分
c parameters for output
      integer   t_step, j_out


c 各種のパラメターの定義
      t_max=100.0d0      !t_max まで計算       ! <-時間の長さを変更
      dt=1.0d-2          !時間間隔 t_k=k*dt
      t_out=1.0D+0       !t_out 間隔にデータ保存
      t_step=int(t_max/dt)
      j_out=int(t_out/dt)


      pi=acos(-1.0D+0)  !円周率
```

```
      Lx=2.0D+0*pi       !領域の広さ
      dx=Lx/dble(N)      !空間間隔

c 初期条件の設定
      k=0
      t=dble(0)
      do i=1, N
        x=dble(i)*dx
        u(i)=dexp(-a*(x-Lx/dble(2))**2)     ! <- v の初期条件を設定
      enddo
c 境界条件を課す
      call bound_cond(u,n)
c 初期値の保存
      do i=0, N
          write(k+100,100) real(i)*dx, t, u(i)
      enddo

c 時間発展の計算----------------------------------
      do j=1, t_step
        t=t+dt
c u の 2 階微分の計算
        call second_deriv(u, d2u, dx, N)
c 各格子点で拡散方程式の発展を計算
        do i=1, N
c    Euler 法による拡散方程式の公式
          u(i)=u(i)+nu*d2u(i)*dt          ! <-u, v の発展方程式を記述
        enddo
c 境界条件を課す
        call bound_cond(u, N)
c データの保存
        if (mod(j, j_out)==0) then
          k=k+1
          do i=0, N
```

```
            write(k+100,100) real(i)*dx, t, u(i)
          enddo
            close(k+100)
        endif
      enddo


 100  format (3(1x,e12.5))
        stop
      end
c-----------------------------------------------------------------------
      subroutine second_deriv(u,d2u,dx,N)
c u の x による 2 階微分
      integer  N, i
      real*8   u(0:N+1)
      real*8   d2u(1:N)
      real*8   dx

      do i=1,n
        d2u(i)=(u(i+1)+u(i-1)-dble(2)*u(i))/dx**2
      end do

      return
      end
c-----------------------------------------------------------------------
      subroutine bound_cond(u,N)
c 周期境界条件を課す
      integer    N
      real*8     u(0:N+1)

      u(0)=u(N)                ! <-境界条件を変更
      u(N+1)=u(1)              ! <-境界条件を変更
```

```
      return
      end
c-----------------------------------------------------------------------
```

参考文献

- 大関 誠, 「スペクトルモデル入門」, 気象研究ノート 第 211 号, 日本気象学会, 2006 年.
- 川上 一郎, 「数値計算」, 第 6 章, 岩波書店, 1989 年.