IV. 数値計算による偏微分方程式の解

9．差分法による偏微分方程式の解法

偏微分方程式に出てくる微分係数の計算を有限差分商で置き換えて解く方法を**有限差分法** (Finite difference method, FDM)という。フーリエ級数を使う方法と比べると，変数分離できない場合や，境界条件が周期的でなく複雑な場合にも利用することができる。

9.1 熱伝導の有限差分方程式の導出

ここでは，1次元の熱伝導方程式

$$\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2} + \frac{H}{C_p} \tag{9.1}$$

と同じ形の拡散方程式

$$\frac{\partial \phi}{\partial t} = \kappa \frac{\partial^2 \phi}{\partial x^2} + h \tag{9.2}$$

を考える。連続した x 座標を Δx 毎に離れた点の集合と考える。連続した座標を点などの集合で置き換えることを**離散化** (discretization)という。テイラー展開を用いると，Δx だけ離れた２つの点における ϕ の値は

$$\phi(x+\Delta x,t) = \phi(x,t) + \frac{\partial \phi}{\partial x}\Delta x + \frac{1}{2}\frac{\partial^2 \phi}{\partial x^2}(\Delta x)^2 + \cdots \tag{9.3}$$

$$\phi(x-\Delta x,t) = \phi(x,t) - \frac{\partial \phi}{\partial x}\Delta x + \frac{1}{2}\frac{\partial^2 \phi}{\partial x^2}(\Delta x)^2 - \cdots \tag{9.4}$$

と表される。ここで，次のような差を作る。

(9.3) − $\phi(x,y)$:

$$\phi(x+\Delta x,t) - \phi(x,t) = \frac{\partial \phi}{\partial x}\Delta x + \frac{1}{2}\frac{\partial^2 \phi}{\partial x^2}(\Delta x)^2 + \cdots \tag{9.5}$$

$\phi(x,y)$ − (9.4):

$$\phi(x,t) - \phi(x-\Delta x,t) = \frac{\partial \phi}{\partial x}\Delta x - \frac{1}{2}\frac{\partial^2 \phi}{\partial x^2}(\Delta x)^2 + \cdots \tag{9.6}$$

(9.3) − (9.4):

$$\phi(x+\Delta x,t) - \phi(x-\Delta x,t) = 2\frac{\partial \phi}{\partial x}\Delta x + \frac{1}{3}\frac{\partial^3 \phi}{\partial x^3}(\Delta x)^3 + \cdots \tag{9.7}$$

それぞれの式から一回微分を求めると

$$\frac{\partial \phi}{\partial x} = \frac{\phi(x+\Delta x,t)-\phi(x,t)}{\Delta x} + O(\Delta x) \tag{9.8}$$

$$\frac{\partial \phi}{\partial x} = \frac{\phi(x,t)-\phi(x-\Delta x,t)}{\Delta x} + O(\Delta x) \tag{9.9}$$

$$\frac{\partial \phi}{\partial x} = \frac{\phi(x+\Delta x,t)-\phi(x-\Delta x,t)}{2\Delta x} + O\left((\Delta x)^2\right) \tag{9.10}$$

のように偏微分係数を差分商で表すことができる。$O(\Delta x)$の項は偏微分係数と差分商の差を表し，その大きさがΔx 1 乗の大きさであることを表す。この大きさは微分を差分で近似したときの誤差を表し，打ち切り誤差とよばれる。ここで，(9.8)を**前進差分**，(9.9)を**後退差分**，(9.10)を**中心差分**とよぶ。前進差分と後退差分は $O(\Delta x)$の打ち切り誤差，中心差分は $O((\Delta x)^2)$の打ち切り誤差を持つ。

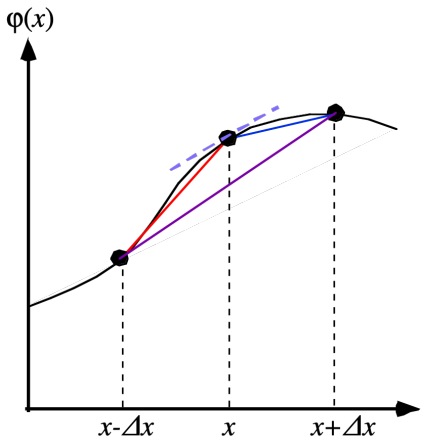

図 9.1 微分係数と差分商の意味

２階微分は(9.8) + (9.9)により得られる。

$$\phi(x+\Delta x,t)+\phi(x-\Delta x,t) = 2\phi(x,t)+\frac{\partial^2\phi}{\partial x^2}(\Delta x)^2+\frac{1}{12}\frac{\partial^4\phi}{\partial x^4}(\Delta x)^4+\cdots \tag{9.11}$$

より，

$$\frac{\partial^2 \phi}{\partial x^2} = \frac{\phi(x+\Delta x,t)+\phi(x-\Delta x,t)-2\phi(x,t)}{(\Delta x)^2} + O\left((\Delta x)^2\right) \tag{9.12}$$

となる。これらを組み合わせると拡散方程式の差分方程式が得られる。時間座標もΔt毎の点の集合と考え，時間微分に前進差分，空間微分に(9.12)を用いると，

$$\frac{\phi(x,t+\Delta t)-\phi(x,t)}{\Delta t}=\kappa\frac{\phi(x+\Delta x,t)+\phi(x-\Delta x,t)-2\phi(x,t)}{\Delta x^2}+h(x,t) \tag{9.14}$$

となる。この式を変形すると，

$$\phi(x,t+\Delta t)=\phi(x,t)+\kappa\Delta t\frac{\phi(x+\Delta x,t)+\phi(x-\Delta x,t)-2\phi(x,t)}{\Delta x^2}+\Delta t h(x,t) \tag{9.15}$$

を得る。この式はΔtだけ未来の時間の値が現在の温度と発熱量から計算できることを表している。このように未来の時間の値を直接得ることができる方法を**陽解法** (explicit method)とよぶ。この式は時間に対して$O(\Delta t)$，空間に対して$O((\Delta x)^2)$の打ち切り誤差を持つ。時間に対して1次の誤差を持つ陽解法を**オイラー法** (Euler method)とよぶ。

差分法の式を表すのに「(引数)」を用いて表すのは煩雑である。このため，xをi番目の点，tをn番目の時間と考え，

$$\phi(x,t)=\phi_i^n \tag{9.16}$$

のように上下の添え字で表す。このとき，(9.15)は

$$\phi_i^{n+1}=\phi_i^n+\kappa\Delta t\frac{\phi_{i+1}^n+\phi_{i-1}^n-2\phi_i^n}{\Delta x^2}+\Delta t h_i^n \tag{9.17}$$

と表すことができる。

9.2 境界条件

(9.17)によると，最初の値を与えれば，その後の変化がすべて計算できそうである。計算するには，これに加えてたとえば，xの2つの端の値が必要なことが分かる。これは**境界条件** (boundary conditions)とよばれる。熱伝導方程式は時間に1階，空間に2階の偏微分方程式である。従って解を求めるために，時間に対して1つの境界条件，空間に対して2つの境界条件が必要である。時間の境界条件を時間$t=0$に対して与える場合，これを**初期条件** (initial conditions)，あるいは**初期値** (initial values)と呼ばれ，初期値から計算を時間進行で行う問題を**初期値問題**という。

元の微分方程式では，初期値は空間全ての点について与える。すなわち1次元では，

$$\phi(x,t=0)=\phi^{ini}(x) \tag{9.21}$$

である。境界条件はすべての時間に対して与える。このとき，次のような組み合わせが考えられる。

・2つの境界の両方に値を与える

・2つの境界の1つに値，1つに勾配を与える

・1つの境界に値と勾配を与える

値そのものを与えるタイプの境界条件を**ディリクレ型** (Dirichlet type)，勾配を与えるものを**ノイマン型** (Neumann type)境界条件とよぶことがある。

$x=0$ の境界に温度を与えるには，

$$\phi(x=0,t)=\phi_0(t) \tag{9.22}$$

とする。一方，勾配を与える場合は

$$\frac{d\phi}{dx}=\frac{q}{k} \tag{9.23}$$

を用いる。

9.3 境界条件の差分法における扱い

差分法では温度の境界を与えるのは簡単である。境界に点を置き，その点の温度に境界の値を入れてやればよい。添え字付きの式で表すと，

$$\phi_0^n=\phi_0 \tag{9.24}$$

と書くことができる。一方，勾配を与える場合は次のように考える。境界には外側はないことに注意する。$x=L$ の点では，境界の点とその1つ内側の点と2つの点で差分を作る。すなわち，後退差分である。つまり，

$$\frac{\phi(L,t)-\phi(L-\Delta x,t)}{\Delta x}=\frac{q}{k} \tag{9.25}$$

となる(ここで q は右から入ってくる方をプラスにしている)。境界条件を表すこの差分が，$O(\Delta x)$の打ち切り誤差となっていることに注意が必要である。このとき，境界の値は

$$\phi(L,t)=\phi(L-\Delta x,t)+\Delta x\frac{q}{k} \tag{9.26}$$

と書くことができる。このとき，境界条件を表すこの差分が，$O(\Delta x)$の打ち切り誤差を持っていることに注意が必要である。境界を m 番目の点と考え，添え字を使うと

$$\phi_m^n=\phi_{m-1}^n+\Delta x\frac{q}{k} \tag{9.26}$$

と表される。

9.4 数値的安定性

理由は後で説明するが，(9.17)は無条件に計算が可能なわけではない。振動や発散が起きて物理的に正しい答えにならなくなったり，計算が不能になったりする。数値計算の際に起こるこのような現象を**数値的不安定** (numerical instability)とよぶ。数値的不安定を起こさないためには計算に誤差が発生したときに徐々に小さくなる(収束する)ような条件が必要である。この条件を**数値的安定性の条件** (numerical stability conditions)という。１次元の拡散方程式を(9.17)のようにオイラー法で解く場合は，

$$\Delta t \le \frac{\Delta x^2}{\kappa} \tag{9.27}$$

が数値的安定性の条件となる。拡散係数κが大きいほど，また，計算点の間隔Δxが小さいほど厳しい条件となる。

１０．差分法の応用：プレート冷却のシミュレーション

熱伝導方程式の解を差分法で解くことにより，プレートの冷却をシミュレートする。境界条件の与え方によって，半無限体冷却モデルでもプレート冷却モデルのどちらでも計算することが可能である。

10.1 基礎方程式と境界条件

基礎方程式は，熱伝導方程式である。１億年程度では放射性元素の影響による内部加熱項は無視できるので，この項を０とした式を用いる。今一度書いておくと，

$$\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial z^2} \tag{10.1}$$

である。境界条件は，表面では温度T_sで一定である。つまり，

$$T_0^n = T_s \tag{10.2}$$

である。底面の境界条件は，プレートモデルの場合，温度一定である。つまり，

$$T_m^n = T_M \tag{10.3}$$

となる。半無限体冷却モデルの場合は，境界は無限遠にある。しかし，無限遠は表現できないので，十分遠いところにあるとする。この場合には，温度一定の(10.3)で与えることも可能であるし，温度勾配を０とした境界で与えることも可能である。後者の場合は，

$$\frac{T_m^n - T_{m-1}^n}{\Delta x} = 0 \tag{10.4}$$

より,

$$T_m^n = T_{m-1}^n \tag{10.5}$$

となる。初期条件は表面を除くすべての点に,

$$T_i^0 = T_M \tag{10.6}$$

と与える。

10.2 プレート冷却シミュレーションのアルゴリズム

問題 IV-1 に実際のプログラムを示す。プログラミングをする際のアルゴリズムは下記のようになる。

(1) 初期条件はタイムステップ0のみ

For i=1 to m

$$T_i^0 = T_M \tag{10.11}$$

(2) 温度の値をファイルに書き出す

(3) 時間進行：オイラー法

For i = 1 to m - 1

$$T_i^{n+1} = T_i^n + \kappa \Delta t \frac{T_{i+1}^n + T_{i-1}^n - 2T_i^n}{\Delta x^2} \tag{10.12}$$

(4) 境界条件の設定

$$T_0^n = T_s \tag{10.13}$$

$$T_m^n = T_{m-1}^n \tag{10.14}$$

(5) 新旧変数の入れ替え

$$T_i^n = T_i^{n+1} \tag{10.15}$$

この等号は代入文を意味することに注意せよ。

(6) 必要に応じて温度の値をファイルに書き出す

(7) (3)へ戻る

問題 IV-1

(1) メインルーチンとサブルーチン２つの計３つのプログラムを入力して，コンパイルせよ。

(2) プログラムを実行し，１千万年，２千万年，４千万年，８千万年，１億６千万年のプレートの温度分布を計算せよ。結果をグラフに表せ。

(3) 地殻熱流量がプレートの厚さと同様な時間依存性を持つことを確認せよ。

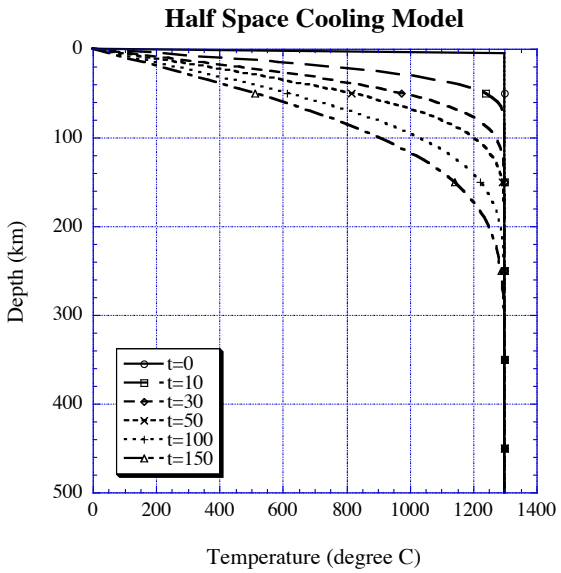

図 10.1 半無限体冷却モデルによるプレートの温度

１１．他の離散化の方法：有限体積法

差分法は座標を離散的な点の集合で表し，テイラー展開で近似することにより，微分を差分で置き換える方法である。したがって，数学的に離散化した式を導いたということができる。ここでは，物理的な保存則に注目した離散化の方法を考える。その方法は**有限体積法**(finite volume method)とよばれる。

11.1 ガウスの定理と有限体積法

熱伝導方程式は，熱が熱伝導のみによって輸送される場合の熱エネルギーの保存則を表している。つまり，ある体積 V 内の物理量の増加はその体積を囲む閉曲面 S を通して入る物理量と等しいということである。式で表すと，

$$\frac{\partial}{\partial t}\int_V \phi\, dV = -\oint_S \left(-\kappa\nabla\phi\right)\cdot d\boldsymbol{S} + \int_V h\, dV \tag{11.1}$$

である(出るほうが正に面ベクトルは定義されるので，マイナスがついている)。ここで，$d\boldsymbol{S}$ は微小な面に垂直な大きさ dS のベクトルである。右辺の面積分はガウスの定理を用いると，

$$\oint_S \left(\kappa\nabla\phi\right)\cdot d\boldsymbol{S} = \int_V \nabla\cdot\left(\kappa\nabla\phi\right) dV + \int_V h\, dV \tag{11.2}$$

のように体積分に置き換えることができる。積分する体積が変化しない場合は，左辺の時間微分を積分の中に入れることができるので，

$$\int_V \frac{\partial\phi}{\partial t} dV = \int_V \left[\nabla\cdot\left(\kappa\nabla\phi\right) + h\right] dV \tag{11.3}$$

となる。この式が，任意の体積に対して成り立つためには，左辺と右辺の積分の内部が等しくなければならないので，拡散方程式(9.2)が導かれる。

有限体積法では，計算する領域を図のように小さな体積に分割し，分割された体積に(11.1)を適用することを考える。分割された小さな体積を**有限体積** (finite volume)あるいは**コントロールボリューム** (control volume)という。さらに，体積が十分小さく，物理量は体積内で一様に分布，すなわち一定であると考える。1次元の場合を考えると，(11.1)は

$$\left(\frac{\partial\phi}{\partial t}\right)_i \Delta V_i = \left[\left(\kappa\frac{\partial\phi}{\partial z}\right)_{i+1/2} - \left(\kappa\frac{\partial\phi}{\partial z}\right)_{i-1/2}\right]\Delta S + h_i \Delta V \tag{11.4}$$

ここで，添字に中途半端な 1/2 という数がついているのは，面が i 番目と $i+1$ 番目の有限体積間の境界であることを表している。図 11.1 の Δz は $\Delta z_{3/2}$ である。ここで，流束に出てくる微分は有限差分で近似することにする。すなわち，

$$\left(\kappa\frac{\partial\phi}{\partial z}\right)_{i+1/2} = \kappa_{i+1/2}\frac{\phi_{i+1}-\phi_i}{\Delta z_{i+1/2}} \tag{11.5}$$

である。この式は κ が空間的に変化する場合でもそのまま適用できる。

境界条件は次のように考える。計算する物体を有限体積に分けたのだから，境界は有限体積の境界に一致すると考える。図の場合は 0 番と 1 番，m 番と $m+1$ 番の有限体積

の間，すなわち 1/2 および $m + 1/2$ の面にある。境界条件をディリクレ型，すなわち，ϕ の値であたえる場合は，ϕ_0 と ϕ_1 の平均が境界値であると考える。よって，

$$\frac{\phi_0 + \phi_1}{2} = \phi_B \tag{11.6}$$

より，

$$\phi_0 = 2\phi_B - \phi_1 \tag{11.7}$$

とする。ノイマン型の境界条件，すなわち流束や勾配で与えられる場合は，

$$\kappa_{1/2} \frac{\phi_1 - \phi_0}{\Delta z_{1/2}} = q \tag{11.8}$$

より，

$$\phi_0 = \phi_1 - \frac{q\Delta z_{1/2}}{\kappa_{1/2}}$$

と与えることができる。

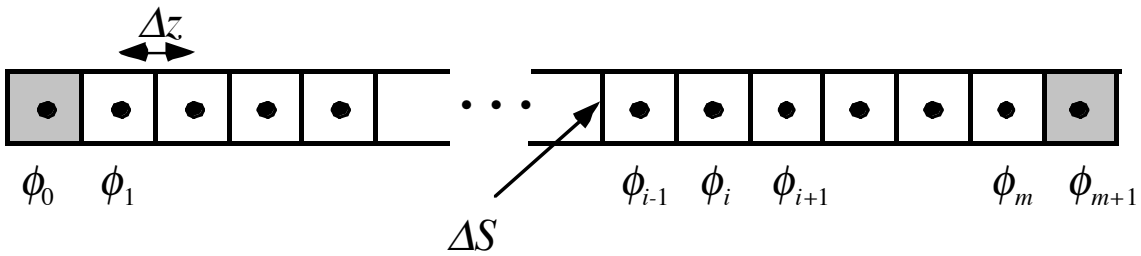

図 11.1 1 次元有限体積法

問題 IV-1　プログラム

プログラムのほか，パラメータを入力するためのデータファイルが必要である。それぞれのファイルを同じディレクトリに置いて，コンパイル，実行すること。

メインプログラムファイル　pl_main_FDM.f90

```
! ************************************************************************
! *                                                                      *
! *     1-D Half-space cooling model for plate thermal evolution         *
! *                solved by finite difference method                    *
! *                                                                      *
! ************************************************************************

      program plate_hscm

      implicit none

      real(8), parameter::   SecY = 365.2422d0*24.0d0*3600.0d0
      real(8), allocatable:: Tmp(:),Tnew(:)
      real(8):: rho,Cp,ak,D
      real(8):: age,T0,TM
      real(8):: z,zk,zmax,zkmax,dz,dzk,dz2
      real(8):: t,tmax,tevo,dt,dtmy,dtlim
      integer:: j,nz
      integer:: it,ntmax
      integer:: nf
      integer:: itskp1,itskp2
      character(40):: apara

!  *****  read parameters  *****

!  Open file
```

```
      open ( 10, file = 'pl_para_2.dat' )

!  read parameter file

      read (10,*) apara
      read (10,*) rho
      write (6,*) apara,' = ',rho

      read (10,*) apara
      read (10,*) Cp
      write (6,*) apara,' = ',Cp

      read (10,*) apara
      read (10,*) ak
      write (6,*) apara,' = ',ak

      read (10,*) apara
      read (10,*) age
      write (6,*) apara,' = ',age

      read (10,*) apara
      read (10,*) T0
      write (6,*) apara,' = ',T0

      read (10,*) apara
      read (10,*) TM
      write (6,*) apara,' = ',TM

      read (10,*) apara
      read (10,*) nz
      write (6,*) apara,' = ',nz
```

```
      read (10,*) apara
      read (10,*) zkmax
      write (6,*) apara,' = ',zkmax

      read (10,*) apara
      read (10,*) dtmy
      write (6,*) apara,' = ',dtmy

      read (10,*) apara
      read (10,*) itskp1
      write (6,*) apara,' = ',itskp1

      read (10,*) apara
      read (10,*) itskp2
      write (6,*) apara,' = ',itskp2

!  convert unit km -> m, Ma -> sec

      zmax = zkmax  * 1.0d3
      tmax = age  * 1.0d6 * SecY
      dt   = dtmy * 1.0d6 * SecY

!  dz: grid size  ****

      dz   = zmax / dfloat( nz )
      dz2  = dz * dz

!  maximum time step number

      ntmax = int( tmax/dt + 0.5d0 )

!  thermal diffusivity
```

```
      D = ak / ( rho*Cp )

!  *****  check dt for numerical stability  ****

      dtlim = 0.5d0 * dz**2 / D
      if ( dt.gt.dtlim ) then
        dt = dtlim * 0.5d0
        write (6,*) '******  caution!!  time step is changed  ******'
      end if

!  allocate dimension variables

      allocate ( Tmp(0:nz), Tnew(nz) )

!  ****  Set initial condition  ****

      it = 0
      t  = 0.0d0

      Tmp(0)  = T0
      do j = 1,nz
        Tmp(j) = TM
      end do

!  *** write initial condition to file ***

      nf = 0

      call      writefile( Tmp, dz, nz, nf )

!  ***  Open file for haet flux  ****

      open ( 20,file = 'hf2.dat',status='new' )
```

```
      call      heatflux( Tmp(0), Tmp(1), t, ak, dz )

!  ==========  Loop of time marching  ================================

      do it = 1,ntmax

!  ***  Time ****

        t  = dt * dfloat( it )

! ***  Time marching by Euler method  ***

        do j = 1,nz-1
          Tnew(j) = Tmp(j) + dt * D / dz2                          &
                          *( Tmp(j-1)+Tmp(j+1)-2.0d0*Tmp(j) )
        end do

!  ***  Change old and new temperatures

        do j = 1,nz-1
          Tmp(j) = Tnew(j)
        end do

!  ***  Boundary condition: dT/dz=0 at bottom  ***

        Tmp(nz) = Tmp(nz-1)

!  ***  Write results to file in each itskp step  ***

        if ( mod(it,itskp1).eq.0 .or. itskp1.eq.1 ) then
          tevo = t / SecY /1.0d6
          write (6,*) it,tevo
          nf = nf + 1
```

```
          call  writefile( Tmp, dz, nz, nf )
        end if

!  ***  calculate heat flux and write to file in each itskp2 step  ***

        if ( mod(it,itskp2).eq.0 .or. itskp2.eq.1 )                 &
          call  heatflux( Tmp(0), Tmp(1), t, ak, dz )

      end do

!  =================  End of the loop  ==============================

      close (20)
      stop
      end program plate_hscm
```

サブルーチンプログラムファイル-1　pl_heatflux.f

```
! ***********************************************************************
! *  heat flux subroutine for 1-D heat conduction                       *
! ***********************************************************************

      subroutine heatflux( Tmp0,   Tmp1,   t, ak, dz )

      implicit  none
      real(8), parameter:: SecY = 365.2422d0*24.0d0*3600.0d0
      real(8):: Tmp0,Tmp1,Qsur
      real(8):: t,ak,dz
      real(4):: ts,Qsurs
      integer:: nz

!  ***  evolution time  ***
```

```
      ts = sngl( t / SecY / 1.0d6 )

!  ***  heat flux at the surface  ***

      Qsur = ak * ( Tmp1 - Tmp0 ) / dz

      Qsurs = sngl( Qsur )
      write (20,*) ts, Qsurs

      return
      end subroutine heatflux
```

サブルーチンプログラムファイル-2　pl_writefile.f

```
! ********************************************************************
! *  file write subroutine for 1-D heat conduction                   *
! ********************************************************************

      subroutine writefile( Tmp, dz, nz, nf )

      implicit none
      real(8):: Tmp(0:nz)
      real(8):: dz
      real(4):: Tmps,zs
      integer:: j,nz
      integer:: nf
      character(1):: cnf1
      character(2):: cnf2
      character(3):: cnf
      character(4):: fname1

!  ***  set number to file name  ***
```

```
      if ( nf.lt.10 ) then
        write (cnf1,'(i1)') nf
        cnf = '00'//cnf1
       else if ( nf.lt.100 ) then
        write (cnf2,'(i2)') nf
        cnf = '0'//cnf2
       else
        write (cnf,'(i3)') nf
      end if

      fname1 = 't'//cnf

!  ***  open new file and write  ***

      open ( 10, file = fname1, status='new' )

      do j = 0,nz
        zs = sngl( dz *dfloat(j) / 1.0d3 )
        Tmps = sngl( Tmp(j) )
        write (10,*) zs,Tmps
      end do

      close (10)

      return
      end subroutine writefile
```

パラメータデータファイル　pl_para_2.dat

```
'density of the mantle (kg/m**3)'
3300.0d0
'specific heat (J/kg)'
1200.0d0
```

```
'thermal conductivity (W/m)'
4.0d0
'age of plate (Ma)'
100.0d0
'surface temperature (deg. C)'
0.0d0
'mantle temperature (deg. C)'
1300.0d0
'number of grid points'
150
'maximum depth (km)'
300.0d0
'time step interval (My)'
0.05d0
'interval for writing temperature (itskp1)'
200
'interval for writing heat flux (itskp2)'
10
```

差分法で1次元熱伝導方程式の数値解を求めるプログラムである。入力されたインターバルで深さを変えながら温度を計算する。

プログラムは3つの部分に分けられており，メインルーチンとサブルーチン2つである。1つは，地殻熱流量の計算とその値のファイルへの書き込みをするサブルーチンである。もう1つは，温度のファイルの書き込みを行っている。

メインルーチン　pl_main_FDM.f

`allocatable Tmp(:),Tnew(:)`：allocatable 文。`Tmp` と `Tnew` を動的割り付け配列変数に指定する。

`dtlim = 0.5d0 * dz**2 / D`：代入文。ここでは，数値安定性の限界となる時間間隔値を計算している。これ以下の if 文で，ユーザーが与えた時間間隔値がこの限界を超えていないかチェックしている。超えた場合は強制的に限界値の半分に変更される。

`allocate ( Tmp(0:nz), Tnew(nz) )` ：allocate 文。動的割り付け配列の大きさを決定する。`Tmp` は `0` から `nz` まで，`Tnew` は `1` から `nz` までとする（`Tnew` は `nz-1` までででも良いが)。

Tmp(0) = T0：代入文。表面温度を境界条件として与えている。表面温度が配列変数の 0 番目の値に代入される。

do j = 1,nz：do 文。j は空間座標を表す添え字で，1 からなので，表面以外の場所である。

Tmp(j) = TM：代入文。表面以外にマントルの温度を初期温度として与える。

open (20,file = 'hf2.dat',status='new') ：open 文。熱流量の値を出力するファイルをオープン(ここでは新規作成)する。

call writefile(Tmp, dz, nz, nf)：call 文。初期温度を保存するため，ファイルに出力するためのサブルーチンを呼び出す。nf はファイルの番号である。

call heatflux(Tmp(0), Tmp(1), t, ak, dz)：call 文。時間ゼロでの熱流量を計算しファイルに出力する。

do it = 1,ntmax：do 文。時間を進行させるための繰り返し。it はタイムステップ値。

t = dt * dfloat(it) ：代入文。時間を計算している。dt は時間間隔値，it は時間ステップ数。

do j = 1,nz-1：do 文。表面と底を除く点で次の時間ステップでの温度が計算される。

Tnew(j) = Tmp(j) + dt * D / dz2 *(Tmp(j-1)+…：代入文。オイラー法による差分方程式である。新しい時間の温度値が計算され，左辺の Tnew に代入される。

Tmp(j) = Tnew(j) ：代入文。温度の値を新しい時間ステップの値に更新する。

Tmp(nz) = Tmp(nz-1) ：代入文。底の境界条件で，温度勾配ゼロ(熱流量ゼロ)を与える。

if (mod(it,itskp1).eq.0 .or. itskp1.eq.1) then：if 文。タイムステップ値 it が itskp1 毎(itskp1 が 1 以外の時は割って余り 0)に温度をファイルに出力する。

nf = nf + 1：代入文。ファイルの番号を 1 つ増やす。

call writefile(Tmp, dz, nz, nf) ：call 文。温度をファイルに出力するためのサブルーチンを呼び出す。

if (mod(it,itskp2).eq.0 .or. itskp2.eq.1)…：if 文。it が itskp2 毎に熱流量を計算するサブルーチンを呼び出す。

サブルーチン　pl_heatflux.f

subroutine heatflux(Tmp0,　Tmp1,　t, ak, dz)：subroutine 文。熱流量の計算をしてファイルに出力する。ファイル名はメインルーチン内で指定している。装置番号がメインルーチンで指定されたものが，そのまま使われることに注意。メインルーチンの Tmp(0)と Tmp(1)はサブルーチンの(ローカル)変数 Tmp1 と Tmp2 に渡される。

Qsur = ak * (Tmp1 - Tmp0) / dz：代入文。熱流量を計算する。熱伝導率と温度勾配を掛けている。温度勾配は前進差分で計算している。

サブルーチン　pl_heatflux.f

`subroutine writefile( Tmp, dz, nz, nf )`：subroutine 文。温度の値を保存するため，温度値をファイルへ出力するサブルーチンである。

`dimension Tmp(0:nz)`：dimension 文。ここで，`nz` はメインルーチンから渡された変数であることに注意。サブルーチンでは，dimension 文の指定に変数を使って，その大きさをメインルーチンと合わせることが出来る。このような配列を整合配列(ajastable dimension)と呼ぶ。メインルーチンでは数かそれと同等なものであるパラメータしか使えないので注意。後から指定するには，動的割り付け配列を使う。

`if ( nf.lt.10 ) then` から `end if` まで: if 文。この if 文では，ファイル名に番号をつけるため，順番を表す番号 `nf` を文字変数 `cnf` に変換している。変換する関数もあるが，変換は 1 文字ずつしか出来ない。そこで，装置番号に変数を置くという変なやり方で変換している。

`cnf = '00'//cnf1`: 代入文。右辺は文字の連結式である。`nf` が 1 桁の場合，`cnf` は 1 文字に変換されるので，前に `00` をくっけて，`003` のような文字列を作っている。これが文字変数 `cnf` に代入される。

`fname1 = 't'//cnf`: 代入文。右辺は文字の連結式である。`fname1` は出力ファイル名である。nf が 3 のとき，`t003` というファイル名になる。

`do j = 0,nz`：do 文。温度の値を添え字 0 から `nz` まで出力するのを繰り返すためである。

`zs = sngl( dz *dfloat(j) / 1.0d3 )`：代入文。深さを km で計算し，単精度変数に代入する。

`write (10,*) zs,Tmps`：do 文。深さと温度を装置番号 `10` のファイルに出力する。`Tmps` は温度を単精度に変換した変数である。

パラメータファイル　pl_para_2.dat

上から，マントルの密度(kg m^{-3})，定圧比熱(J kg^{-1})，熱伝導率(W m^{-1})，プレートの年代(Ma)，地表温度(°C)，マントルの温度(°C)，計算する深さの間隔(km)，計算する最大の深さ(km)である。