数値計算演習

演習の概要

1. 演習の目的
   - 数値計算に必要な計算機の利用の仕方を修得する。
   - 数値計算の基礎的な概念を理解する。
   - 実際の計算機上でプログラムを組み、数値計算による問題解決を体験する。
2. 演習の内容
   (a) 数値計算の必要性
   (b) クライアントコンピュータ（Windows パソコン）の使い方
   (c) サーバコンピュータ（Unix ワークステーション）の使い方
   (d) 数値計算の方法
      - 数値積分法
      - 常微分方程式の数値解法
      - 一般方程式の数値解法
      - モンテカルロ法

この資料は 1996-2003 年度に作成した数値計算演習のページ (根本幸児、大西 明、及び TA 作成) を自習用に少し書き換えたものです。教員の E-mail address を <Teacher@server> としてありますが、もちろんこれは正しい address ではありません。メールでのサポート等はなく、”As is” で提供します。

数値計算演習で標準的に行う事柄

演習時間に行なう事柄のリスト

1. PC の電源を入れる。
2. Browser を立ち上げて、演習のページを開く。HTML files `CompPhys_html-HUOCW.zip` を入手して展開し、Browser から「ファイルを開く」により数値計算演習のトップページを開く。

   `file:///home/Your_userid/CompPhys_html-HUOCW/index.html`

3. Unix コンピュータに login する。
4. プログラムを編集する。(ファイル名の yy の部分は週によって違います。)

   `mule progyy.f &`

   完成したら mule の中で C-x C-s でファイルに書き込んで、端末画面の Window にうつる。
5. プログラムをコンパイルする。

   `f90 progyy.f` (a.out ができます。)

   あるいは

   `f90 progyy.f -o progyy` (実行ファイル名は progyy)

   失敗していたら、どこかに間違いがあるはず。4. に戻って確かめる。
   北大のサーバでは fortran プログラムのコンパイルコマンドは `f90` ですが、コマンド名はサーバによって異なります。(`g77`、`f77` など)。環境にあわせて読みかえてください。
6. コンパイルに成功したら、プログラムを実行する。

   `./a.out`

   あるいは

   `./progyy`

   失敗していたら、どこかに間違いがあるはず。4. に戻って確かめる。
7. 実行がうまくいったことを確かめた後、担当教官までメールを送る。

   `mail Teacher@server s0300xxy < progyy.f`

   (`s0300xxy` は、あなたの userid のことです。最後の `xxy` の部分は、人により異なります。)
8. 余裕があれば、プリントの「やってみよう」や「演習問題」の部分を試したり、演習のホームページの他の部分をみて、”数値計算” やコンピュータの使い方に慣れていく。
9. 最後に
   mule を終了 (C-x C-c)、Unix コンピュータから logout、Windows を終了、
   の後、PC の電源を切る。
10. Good bye !

Contents

1　実数の和

整数 n と実数 x を入力して

$$\mathrm{sum} = \sum_{k=0}^{n} x^k = 1 + x + x^2 + \cdots + x^n$$

を出力するプログラムを作りましょう。もちろん答は、$x \neq 1$ のとき $\mathrm{sum} = \dfrac{1-x^{n+1}}{1-x}$，$x = 1$ のとき $\mathrm{sum} = 1 + n$ です。

1.1　最も単純なプログラム

```
 1  C234567890
 2        program prog01
 3        write(*,*) 'n, x = ?'
 4        read(*,*) n, x
 5        write(*,*) 'n, x = ',n,x
 6  c
 7        sum = 0.0
 8        do k = 0,n
 9           sum = sum + x**k
10        end do
11  c
12        write(*,*) 'sum   = ',sum
13  c
14        if(x.eq.1.0) then            ! IF (条件式) THEN
15           write(*,*) 'Exact = ',n+1
16        else                         ! ELSE
17           write(*,*) 'Exact = ',(1.0-x**(n+1))/(1.0-x)
18        endif                        ! ENDIF
19  c
20        stop
21        end
```

さて、例として n=100, x=0.02 を代入して計算してみて下さい。答えは一致しますか？

```
% n, x = ?
% ?
% 100 0.02
% n, x =           100  0.199999996E-01
% sum   =    1.02040803
% Exact =    1.02040815
```

一致しないのはなぜでしょう？

1.2 漸化式 + 倍精度実数を用いたプログラム

```
 1        program prog012
 2        real*8 x, sum
 3        write(*,*) 'n, x = ?'
 4        read(*,*) n, x
 5        write(*,*) 'n, x = ',n,x
 6  c
 7  c 漸化式を利用した計算の例
 8  c
 9  c     S(0)=1
10  c     S(k)=S(k-1)*x + 1
11  c     sum =S(n)
12  c
13        sum = 1.0d0 ! 1.0 のことだが、ずっと 0 が続いていることを示す。
14        do k = 1, n
15           sum = sum*x + 1.0d0
16        end do
17  c
18        write(*,*) 'sum   = ',sum
19        if(x.eq.1.0d0) then
20           write(*,*) 'Exact = ',n+1
21        else
22           write(*,*) 'Exact = ',(1.0d0-x**(n+1))/(1.0d0-x)
23        endif
24        stop
25        end
```

このプログラムでは、次の漸化式を用いて計算しています。

$$\begin{aligned} S_0 &= 1 \\ S_k &= S_{k-1}x + 1 \end{aligned}$$

この式から、$\mathrm{sum} = S_n$ となることがわかります。

1.3 今日のポイント

- プログラムは 7 桁目から
- read 文、write 文
- FORTRAN プログラムにおける左辺と右辺の意味
- DO ループ、END DO 文
- FORTRAN における暗黙の型宣言 $\cdots$ i $\sim$ n は整数, a $\sim$ h, o $\sim$ z は単精度実数
- IF ループ

```
 IF（条件式）THEN
      条件にあっている場合の動作
ELSE
      条件にあっていない場合の動作
ENDIF
```

- 桁落ちと精度、倍精度実数
- プログラムの名前
- コメント行、文中のコメント

1.4 余裕があれば、試してみて下さい。

- 漸化式を使わないでも、精度良く答えを出す方法があります。考えてみて下さい。
  (ヒント: 小さいものから順に足せばよいので、|x| が 1 よりも小さい時には、DO loop で足し合わせる順序を逆にすれば精度は上がります。)

1.5 プログラムが出来たら...

prog012.f (漸化式 + 倍精度実数を用いた計算精度の高いプログラム) までできたら、プログラムをメールして下さい。

```
mail Teacher@server s0300xxy < prog012.f
```

2　数値積分 (その 1, function 副プログラム)

区分数 n と下限 a, 上限 b を入力して

$$S = \int_a^b \frac{1}{\sqrt{x}}\,\mathrm{d}x$$

の近似値を中点則で計算するプログラムを作りましょう。もちろん真の値は $S = 2(\sqrt{b} - \sqrt{a})$ です。

2.1　中点公式を用いたプログラム (prog02.f)

```
 1        program prog02
 2        implicit real*8 (a-h,o-z)
 3  c
 4  c     This program calculate the integral
 5  c       sum = integral_a^b 1/sqrt(x) dx
 6  c     by using the Mid-Point formula.
 7  c
 8        write(*,*) 'n, a, b = ?'
 9        read(*,*)   n, a, b
10        write(*,*) 'n = ', n,' a = ',a,' b = ',b
11  c
12        sum = 0.0d0
13        dx = (b - a)/n
14  c
15        do i = 1, n
16           x = a + (i - 0.5d0)*dx
17           sum = sum + f(x)
18        end do
19           sum = sum*dx
20  c
21        exa=2*(sqrt(b)-sqrt(a))
22        write(*,*) 'Sum    = ',sum
23        write(*,*) 'Exact  = ',exa
24        stop
25        end
26  c ***********************************************************************
27        function f(x)
28  c ***********************************************************************
29        implicit real*8 (a-h,o-z)
30        f = 1.0d0/sqrt(x)
31        end
```

さて、例として n=10, a=10, b=20 を代入して計算してみて下さい。有効数字何桁位一致しますか？

```
n, a, b = ?
?
10, 10, 20
n =           10  a =    10.0000000000000000       b =    20.0000000000000000
Sum    =    2.61929135954643399
Exact  =    2.61971658966240017
```

2.2 シンプソン公式を使ったプログラム (prog022.f)

前回と同様に、精度を上げる努力をしてみましょう。ここではシンプソン公式を使ってみます。

```
 1        program prog022
 2        implicit real*8 (a-h,o-z)
 3  c
 4  c     This program calculate the integral
 5  c       sum = integral_a^b 1/sqrt(x) dx
 6  c     by using the Simpson formula.
 7  c
 8        write(*,*) 'n, a, b = ?'
 9        read(*,*)   n, a, b
10        write(*,*) 'n = ', n,' a = ',a,' b = ',b
11  c
12           sum = 0.0d0
13           dx = (b - a)/n
14  c
15        do i = 1, n
16           x1 = a + (i - 1.0d0)*dx ! 短冊の左端
17           x2 = a + (i - 0.5d0)*dx ! 短冊の中央
18           x3 = a +  i         *dx ! 短冊の右端
19           sum = sum + (f(x1) + 4*f(x2) + f(x3))/6 ! 1:4:1 の比で足す。
20        end do
21           sum = sum*dx
22  c
23        exa=2*(sqrt(b)-sqrt(a))
24        write(*,*) 'Sum    = ',sum
25        write(*,*) 'Exact  = ',exa
26        stop
27        end
28  c ********************************************************************
29        function f(x)
30  c ********************************************************************
31        implicit real*8 (a-h,o-z)
32        f = 1.0d0/sqrt(x)
33        end
```

さて、n を増やしていったとき、誤差はどの程度まで小さくなるでしょう？

2.3 今日のポイント

- implicit 文 (暗黙の型宣言を変更する)
  ··· 今の場合、i ～ n は整数, a ～ h, o ～ z は倍精度実数

- function 副プログラム
  以下の例では、function f(x) は、メインプログラム (program ... からはじまり、end で終っている最初の部分) から呼び出して、 1.0d0/sqrt(x) の値を返します。

```
      program prog02
      implicit real*8 (a-h,o-z)
      ...
      sum = sum + f(x)*dx
      ...
      end
c
      function f(x)
      implicit real*8 (a-h,o-z)
      f=1.0d0/sqrt(x)
      end
```

- FORTRAN で使える関数

- 積分公式 ··· 中点公式、台形公式、シンプソン公式

2.4 演習問題

2.4.1 演習１

次の定積分を求めるプログラムを作り、分点の数を増やしたときその値が真の値にどのように近付くかを調べてみよう。

$$(1)\quad \int_0^1 \pi \sin \pi x \, \mathrm{d}x \qquad (2)\quad \int_0^1 4\sqrt{1-x^2} \, \mathrm{d}x \qquad (3)\quad -\int_0^1 \log x \, \mathrm{d}x$$

(注：$\pi = 3.14159\ 26535\ 89793\ \cdots$)

2.4.2 演習２

自分でいろいろな関数の定積分を考え、それを計算するプログラムを作ってみよう。

2.5 FORTRAN で使える主な数学関数 (実数型)

必要となるかもしれない関数です。

関数名	数式	意味
`sqrt(x)`	$\sqrt{x}$	平方根
`exp(x)`	e^x	指数関数
`log(x)`	$\log_{\mathrm{e}}(x)$	自然対数
`log10(x)`	$\log_{10}(x)$	常用対数
`sin(x)`	$\sin(x)$	正弦関数
`cos(x)`	$\cos(x)$	余弦関数
`tan(x)`	$\tan(x)$	正接関数
`asin(x)`	$\sin^{-1}(x)$	逆正弦関数
`acos(x)`	$\cos^{-1}(x)$	逆余弦関数
`atan(x)`	$\tan^{-1}(x)$	逆正接関数
`atan2(x,y)`	$\tan^{-1}(x/y)$	逆正接関数
`sinh(x)`	$\sinh(x)$	双曲正弦関数
`cosh(x)`	$\cosh(x)$	双曲余弦関数
`tanh(x)`	$\tanh(x)$	双曲正接関数
`abs(x)`	$\lvert x\rvert$	絶対値

2.6 プログラムが出来たら...

プログラム `prog02.f` をメールで送って下さい。また、演習問題のプログラムを次回の演習の (終りの) 時間までに作っておいて下さい。

3 数値積分 (その 2)

前回と同じ問題設定ですが、subroutine を使ったプログラムを作ってみましょう。

3.1 サブルーチンを用いたプログラム (prog03.f)

```
 1        program prog03
 2        implicit real*8 (a-h,o-z)
 3        write(*,*) 'a, b = ?'
 4        read(*,*)   a, b
 5        write(*,*) 'a = ',a,' b = ',b
 6        exa=2*(sqrt(b)-sqrt(a))
 7        do n=10,100,10
 8           call chuten(a,b,n,sum)
 9           dx=(b-a)/n
10           err=abs(sum-exa)
11           write( *,800) n, dx, sum, exa, err, err/dx/dx
12           write(16,800) n, dx, sum, exa, err, err/dx/dx
13        enddo
14  c234567
15   800  format(1x,i3,5(1x,f10.7))
16  c
17        end
18  c ***********************************************************************
19        subroutine chuten(a,b,n,sum)
20  c ***********************************************************************
21        implicit real*8 (a-h,o-z)
22        sum = 0.0d0
23        dx = (b - a)/n
24  c ここから４行は中点公式で実際に積分値を求める部分です。
25  c 考えて書いてみましょう。
26  c ---------------------------------------------------------------------*
27
28
29
30
31  c ---------------------------------------------------------------------*
32        sum = sum*dx
33        end
34  c ***********************************************************************
35        function f(x)
36  c ***********************************************************************
37        implicit real*8 (a-h,o-z)
38        f = 1.0d0/sqrt(x)
39        end
```

3.2 積分する function の名前を変えられる subroutine を使ったプログラム (prog032.f)

上のプログラムでは、関数 f(x) しか積分出来ません。いくつかの別の関数を同様のやり方で積分出来るようにしてみましょう。

```
 1        program prog032
 2        implicit real*8 (a-h,o-z)
 3        external f1
 4        a=0.0d0
 5        b=1.0d0
 6        exa=2.0d0
 7        do n=10,100,10
 8           call chutenf(f1,a,b,n,sum)
 9           dx=(b-a)/n
10           err=abs(sum-exa)
11           write( *,800) n, dx, sum, exa, err, err/dx/dx
12           write(16,800) n, dx, sum, exa, err, err/dx/dx
13        enddo
14   800  format(1x,i4,5(1x,f10.7))
15        end
16  c ***********************************************************************
17        subroutine chutenf(FUNC,a,b,n,sum)
18  c ***********************************************************************
19        implicit real*8 (a-h,o-z)
20        external FUNC
21  c
22        sum = 0.0D0
23        dx = (b - a)/n
24  c ここから４行は中点公式で実際に積分値を求める部分です。
25  c 考えて書いてみましょう。
26  c -----------------------------------------------------------------*
27
28
29
30
31  c -----------------------------------------------------------------*
32        sum = sum*dx
33        end
34  c ***********************************************************************
35        function f1(x)
36  c ***********************************************************************
37  c     function f1(x)=pi*sin(pi*x)
38  c ***********************************************************************
39        implicit real*8 (a-h,o-z)
40        pi=4*atan(1.0d0)
41        f1=pi*sin(pi*x)
42        end
```

3.3 今日のポイント

- subroutine 副プログラムの使い方
  まとまった一つの操作の組合せを行なうのが subroutine です。
  以下の例では、subroutine chuten(a,b,n,sum) は、メインプログラム (program ... からはじまり、end で終っている最初の部分) から呼び出されて、与えられた a,b,n の値を使って積分値 sum の値を返します。

```
c234567
      program prog03
      implicit real*8 (a-h,o-z)
      ...
      call chuten(a,b,n,sum)
      ...
      end
c
      subroutine chuten(a,b,n,sum)
      implicit real*8 (a-h,o-z)
      ...
      sum = sum + f(x)
      ...
      end
c
      function f(x)
      implicit real*8 (a-h,o-z)
      ...
      f = ...
      end
```

- do n=初期値，終り値，きざみ

```
      DO K = 初期値，終り値 (，きざみ)
            (演算内容)
      END DO
```

(例) k = 10 から 100 まで 10 ずつ増やしてて繰り返す場合。

```
  DO  K = 10, 100, 10
      (演算の内容)
  END DO
```

きざみを省略すると、1 と見なされます。(K が 1 ずつ増える)

- write(unit,format): ファイルへの出力
  例: write(16,800): ft.16 に文番号 800 に示された format で出力

- 800 format(1x,i3,5(1x,f10.7))
  - 文番号は 1 ～ 5 桁の間に。
  - nx は n 個の空白、in は n 桁の整数、
    5(...) は、(...) を 5 回繰り返し、f10.7 は、実数で少数点以下 7 桁で出力

- external 文
  メインプログラムと副プログラムでともに、external f と宣言することにより、「関数の名前」を引数として副プログラムにわたすことができます。

```
c234567
      program prog032
      implicit real*8 (a-h,o-z)
      external f
```

```
      ...
      call chutenf(f,a,b,n,sum)
      ...
      end
c
      subroutine chutenf(f,a,b,n,sum)
      implicit real*8 (a-h,o-z)
      external f
      ...
      sum = sum + f(x)
      ...
      end
c
      function f(x)
      implicit real*8 (a-h,o-z)
      ...
      f = ...
      end
```

3.4 課題 その 1

問題1、2プログラム (ex031.f 及び ex032.f) を作って下さい。課題はメイルにして送ること。

- 問題1

  関数名 FUNC, 下限 a, 上限 b, 分割数 N を与えて台形則による積分 sum を計算するサブルーチン

  subroutine daikeif(FUNC,a,b,N,sum)

  を作り、これを利用して、分割数を 10 から 10 おきに 100 まで増やしていって、それぞれに対応する Δx と積分値を出力するプログラム

  ex031.f

  を作って下さい。

- 問題2

  関数名 FUNC, 下限 a, 上限 b, 分割数 N を与えて Simpson 則による積分 sum を計算するサブルーチン

  subroutine simpsonf(FUNC,a,b,N,sum)

  を作り、これを利用して、分割数を 10 から 10 おきに 100 まで増やしていって、それぞれに対応する Δx と積分値を出力するプログラム

  ex032.f

  を作って下さい。

問題1、2のどちらの場合でも定積分は答えを解析的に求められるもの (例えば先週の演習1) を設定して下さい。またプログラムは入力無しで動作するようにしてください。(read 文無しのプログラム。必要な値はプログラム中に書いておく。) また、積分値と解析解の差は Δx と予想された関係にあるかどうか調べて下さい。どのような場合 (関数、区間) に、予想された関係からずれるでしょうか？

3.5 Double Exponential 公式を用いた高精度積分プログラム (prog033.f)

参考文献: 森 正武、「FORTRAN77 数値計算プログラミング」(岩波コンピュータサイエンス)

変数変換を行って高精度の積分を行うプログラムです。

$$
\begin{aligned}
S &= \int_a^b f(x)\,dx = \int_{-\infty}^{\infty} f(\varphi(t))\,\frac{d\varphi}{dt}\,dt \\
\varphi(t) &= \frac{b-a}{2}\,\tanh\left(\frac{\pi}{2}\sinh t\right) + \frac{b+a}{2}
\end{aligned}
$$

上の 2 重指数関数変換を行うことにより、積分を行う変数の領域を $-\infty$ から ∞ に変換し、積分を有限の和によって近似するときの誤差がなるべく打ち消すようになっています。

```
 1        program prog033
 2        implicit real*8 (a-h,o-z)
 3  c
 4  c     This program gives the integral
 5  c     integral_0^1 FUNC(x) dx
 6  c     by using the Double Exponential Formula
 7  c     by Takahashi-Mori
 8  c
 9  c     Functions integrated in this program are
10  c
11  c        f0(x)=1.0d0/sqrt(x)
12  c        f1(x)=pi*sin(pi*x)
13  c        f2(x)=4*sqrt(1.0d0-x**2)
14  c        f3(x)=-log(x)
15  c
16        external f1,f2,f3,f0
17        a=0.0d0
18        b=1.0d0
19  c
20        hmax=4.7d0
21  c
22        exa1=2.0d0
23        exa2=4.0d0*atan(1.0d0)
24        exa3=1.0d0
25  c
26        write( *,'(a)') '#   n  dx          Err1        Err2        Err3'
27        write(16,'(a)') '#   n  dx          Err1        Err2        Err3'
28        do n=5,50,5
29           call DEint(f1,a,b,n,sum1)
30           call DEint(f2,a,b,n,sum2)
31           call DEint(f3,a,b,n,sum3)
32  c
33           err1=abs(sum1-exa1)
34           err2=abs(sum2-exa2)
35           err3=abs(sum3-exa3)
36  c
37        write( *,800) n,hmax/n,err1,err2,err3
38        write(16,800) n,hmax/n,err1,err2,err3
```

```
 800  format(1x,i4,4(1x,g11.4))
      enddo
c
      end
c **********************************************************************
      subroutine DEint(FUNC,a,b,n,sum)
c **********************************************************************
c
c     This subroutine calculate the integral
c       sum = integral_a^b FUNC(x) dx
c     by using the Double Exponential formula (Takahashi-Mori).
c
c       sum= integral_{-infinity}^infinity FUNC(phi(t)) d(phi)/dt dt
c       phi = (b-a)/2*tanh(pi/2*sinh(t))+(b+a)/2
c
c **********************************************************************
      implicit real*8 (a-h,o-z)
      external FUNC
c
      tmax=4.7d0
      dt=tmax/n
      pi=4.0d0*atan(1.0d0)
c
      sum = 0.0d0
      do i = -n,n
         t=i*dt
         s=sinh(t)
         phi=(b-a)/2*tanh(pi/2*s)+(b+a)/2
         dphidt=pi/2*cosh(t)/(cosh(pi/2*s))**2*(b-a)/2
         sum=sum+FUNC(phi)*dphidt
      end do
         sum = sum*dt
c
      end
c **********************************************************************
      function f0(x)
c **********************************************************************
c     function f0(x)=1.0d0/sqrt(x)
c        0 < x < eps it returns 1.0d0/sqrt(eps)
c        x < 0       it returns 0
c **********************************************************************
      implicit real*8 (a-h,o-z)
c
      eps=1.0d-32
      if(x.lt.0) then
         f0=0.0d0
      else if(x.lt.eps) then
         f0=1.0d0/sqrt(eps)
      else
         f0=1.0d0/sqrt(x)
      endif
```

```
 90  c
 91        end
 92  c **********************************************************************
 93        function f1(x)
 94  c **********************************************************************
 95  c     function f1(x)=pi*sin(pi*x)
 96  c **********************************************************************
 97        implicit real*8 (a-h,o-z)
 98        pi=4.0d0*atan(1.0d0)
 99  c
100        f1=pi*sin(pi*x)
101  c
102        end
103  c **********************************************************************
104        function f2(x)
105  c **********************************************************************
106  c     function f2(x)=4*sqrt(1.0d0-x**2)
107  c        |x| > 1          it returns 0
108  c **********************************************************************
109        implicit real*8 (a-h,o-z)
110  c
111        if(abs(x).gt.1.0d0) then
112           f2=0.0d0
113        else
114           f2=4*sqrt(1.0d0-x**2)
115        endif
116  c
117        end
118  c **********************************************************************
119        function f3(x)
120  c **********************************************************************
121  c     function f3(x)=-log(x)
122  c        0 < x < eps  it returns -log(eps)
123  c        x < 0        it returns 0
124  c **********************************************************************
125        implicit real*8 (a-h,o-z)
126  c
127        eps=1.0d-32
128        if(x.lt.0) then
129           f3=0.0d0
130        else if(x.lt.eps) then
131           f3=-log(eps)
132        else
133           f3=-log(x)
134        endif
135  c
136        end
```

計算実行例

```
ap1 47: f90 prog033.f
f90: compile start : prog033.f
```

```
*OFORT90 V01-04-/A 開始
*プログラム名 = PROG033
*プログラム名 = DEINT
*プログラム名 = F1
*プログラム名 = F2
*プログラム名 = F3
*プログラム名 = F0
*プログラム数 = 0006 , エラーはありません。
ap1 48: a.out
#   n  dx          Err1        Err2        Err3
    5  0.9400      0.4048      0.9390E-01  0.1450E-01
   10  0.4700      0.2161E-02  0.4287E-04  0.3120E-05
   15  0.3133      0.4557E-05  0.5213E-08  0.2562E-09
   20  0.2350      0.5803E-08  0.7003E-12  0.1377E-13
   25  0.1880      0.5318E-11  0.8882E-15   0.000
   30  0.1567      0.3553E-14  0.1332E-14  0.2220E-15
   35  0.1343      0.2220E-15   0.000      0.1110E-15
   40  0.1175       0.000      0.8882E-15  0.1110E-15
   45  0.1044      0.4441E-15   0.000       0.000
   50  0.9400E-01  0.4441E-15  0.4441E-15   0.000
```

3.6 代表的な質問

どうやってエラーのある場所を探せばよいですか？

ホームページの ”How to Run” の部分
file:///home/Your_userid/CompPhys_html-HUOCW/LESSON/compile.html
を見て下さい。

よくあるのは、ある行で . (ピリオド) と , (コンマ) を間違っている場合です。ある行での間違いは、メッセージで KCHF と書いてある行で最後が数字が行番号を表しています。この場合は 15 行目に間違いがあります。(分類不可能な文があります (命令のスペルミス)、不当な区切り記号が使用されました (. と , の間違い)、などのメッセージがこれに対応します。)

```
*OFORT90 V01-03-/B 開始
  KCHF049K  12            15
            分類不可能な文があります。
*プログラム名 = MAIN
*プログラム数 = 0001, エラー総数 = 0001, 最大エラーレベル = 12
```

次によくあるのは、それぞれの行は間違っていないが、変数や関数が定義されていないばあいです。この場合には「エラーはありません」、と出てくるのに、実行ファイル ”a.out” が出来ません。下の例の場合は、F1 が Unsatisfied symbols である (変数、あるいは関数が未定義である) とのメッセージが最後に出ていますね。F1 (あるいは f1, fortran では大文字、小文字が区別されない。) が使われているところを (mule では C-s で) 探して下さい。

```
*OFORT90 V01-04-/A 開始
*プログラム名 = REIDAI
*プログラム名 = F
*プログラム数 = 0002 , エラーはありません。
/usr/ccs/bin/ld: Unsatisfied symbols:
   F1 (code)
```

Mule の使い方について

Q．mule で日本語を書くにはどうすればよいのか？ また、文節を伸ばしたり縮めたりするにはどうしたらよいのか？
A．C-o で日本語を書くモードに入ります。 またフェンスモード (|...| となっているモード) のときには、 C-o で文節を伸ばし、C-i で文節を縮められます。

Q．mule でプログラムを編集し、新しいプログラムを書きたい。今のプログラムをもとにして書くにはどうすればよいのか？
A．C-x C-w で別のファイル (新しいファイルでもよい) に save できます。 (ここも参照。) それから出発すれば、書き換えの手間が少ないですね。 このファイルに書き込むという操作は、 メニューの File → Save Buffer As... を選んでもできます。

Q．mule 変なキーをおして、抜けられなくなりました。
A．困った時はとにかく何度か C-g です。

キー操作のまとめ、初心者の練習方法については、
file:///home/Your_userid/CompPhys_html-HUOCW/LESSON/mule-short.html
file:///home/Your_userid/CompPhys_html-HUOCW/LESSON/mule-first.html
も参照して下さい。

4　微分方程式 その 1: １変数１階微分方程式

区分数 n を入力して、初期条件 $y(1) = 1$ のもとに、次の微分方程式を解くプログラムを作りましょう。もちろん答えは $y(x) = \sqrt{x}$ です。

$$\frac{dy}{dx} = \frac{y}{2x}$$

次のプログラムでは、差分法を使って次のように解いています。本当の”仕事” をしているのは 26 行目だけですね。

$$y_k = y_{k-1} + f(x_{k-1}, y_{k-1})\,\Delta x$$

4.1　オイラー法で微分方程式を解くプログラム (prog04.f)

```
 1        program prog04
 2  C
 3  C       This program solves the differential equation
 4  C           dy/dx = y/2x
 5  C       by using the Euler method.
 6  C
 7        implicit real*8(a-h,o-z)
 8  C
 9        write(*,*) 'n=?'
10        read(*,*) n
11  C
12        xi = 1.0d0         ! Initial value of x
13        xf = 2.0d0         ! Final   value of x
14        yi = 1.0d0         ! Initial value of y
15        dx = (xf-xi)/n     ! Mesh size
16  C
17        x = xi             ! Initial Conditions
18        y = yi             ! Initial Conditions
19  C
20        open(16,file='prog04.dat')     ! open(unit,file='filename')
21        write( *,*) x,y,abs(y-sqrt(x))
22        write(16,*) x,y,abs(y-sqrt(x))
23        do k = 1,n
24           dy = f(x,y)*dx
25           x = x + dx         ! x(k) = x(k-1) + dx
26           y = y + dy         ! y(k) = y(k-1) + dx * f(x(k-1), y(k-1))
27           write( *,*) x,y,abs(y-sqrt(x))
28           write(16,*) x,y,abs(y-sqrt(x))
29        end do
30  C
31        stop
32        end
33  c ***********************************************************************
34        function f(x,y)
35  c ***********************************************************************
36        implicit real*8 (a-h,o-z)
37        f = 0.5d0*y/x
38        end
```

4.2 今日のポイント

- 常微分方程式の (解析的な) 解き方
  1. 変数分離形
  $$\frac{dy}{dx} = f(x)\,g(y) \quad \rightarrow \frac{dy}{g(y)} = f(x)\,dx$$
  2. 同次形 $\cdots$ $u = y/x$ と置き換えて変数分離形へ
  $$\frac{dy}{dx} = f(y/x) \quad \rightarrow x\,\frac{du}{dx} + u = f(u) \quad (u = y/x)$$
  3. 1 階線形常微分方程式 $\cdots$ 一般解 = 特解 + 同次方程式の一般解
  $$\frac{dy}{dx} + p(x)\,y = q(x)$$
  $$\rightarrow y = \exp\left[-\int^x p(x')\,dx'\right]\ \left[C + \int^x q(x')\,\exp\left\{\int^{x'} p(x'')\,dx''\right\}\,dx'\right]$$
  (C は積分定数)
  4. $y'' = f(y)$ の形 $\cdots$ $u = dy/dx$ とおくと 変数分離形へ
  $$\frac{d^2y}{dx^2} = f(y) \quad \rightarrow u\,\frac{du}{dy} = f(y) \quad \rightarrow u = \frac{dy}{dx} = \pm\sqrt{2\int^y f(y')\,dy\ +\ C}$$

- 常微分方程式の (数値的な) 解き方
  1. オイラー法:
  $$y_k = y_{k-1} + f(x_{k-1}, y_{k-1})\,\Delta x$$
  2. 修正オイラー法: 半分進ませた点での微係数を使う。
  $$\begin{aligned} y_k &= y_{k-1} + f(x', y')\,\Delta x\ , \\ x' &= x_{k-1} + \Delta x/2\ , y' = y_{k-1} + f(x_{k-1}, y_{k-1})\,\Delta x/2 \end{aligned}$$
  3. 改良オイラー法
  4. Heun の公式

- `open(unit, file='filename')`

4.3 演習問題

(1) 今回のプログラムで刻み幅 `dx` を小さくする（`n` を大きくする）と、どのように真の値に近付くかを確かめてみよう。$x = 2$ の時の近似値と真の値の差を `dx` の関数としてプロットするのに便利なデータを出力するように改造し、その様子を調べて下さい。

(2) 上で改造したプログラムを修正オイラー法を用いるものに変更し、どの程度改良されるかを調べてみよう。

(3) 空気中を速さ v で落下する物体には、空気抵抗の力 αv^2 が働くものとします。このとき、$v \geq 0$ とすれば、重力加速度 g のもとで落下する物体の運動方程式は

$$m\frac{\mathrm{d}v}{\mathrm{d}t} = mg - \alpha v^2$$

となります。$m = 1, \alpha = 0.1, g = 9.8$ として $v(t)$ を求めるプログラムを作ってみよう。ただし初期条件 $v(t = 0) = 0$ とします。

4.4 Gnuplot でグラフを描こう

4.4.1 コマンドラインからグラフを描く

今週は gnuplot というコマンドを使って、グラフを描いてみましょう。まず、次にある下線部をタイプしてみて下さい。

```
ap1:  gnuplot

      G N U P L O T
      Unix version 3.7
      patchlevel 0
      last modified Thu Jan 14 19:34:53 BST 1999

      Copyright(C) 1986 - 1993, 1998, 1999
      Thomas Williams, Colin Kelley and many others
      ....

Terminal type set to 'x11'
gnuplot> plot sin(x)                (plot 関数 で関数のグラフ)
gnuplot> plot "prog04.dat"          (plot "ファイル名" で 1, 2 カラム目のグラフ)
gnuplot> plot "prog04.dat" , sqrt(x)
                                  (plot "ファイル名", 関数, ...  で重ねたグラフ)
gnuplot> quit                       (gnuplot を終る)
```

4.4.2 スクリプトファイルを用いてグラフを描く

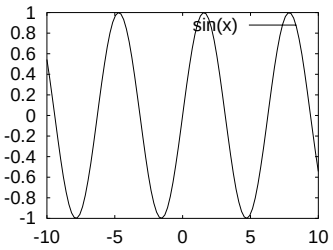
次に、下のような2行を書いたファイル (prog04.plt) を作りましょう。

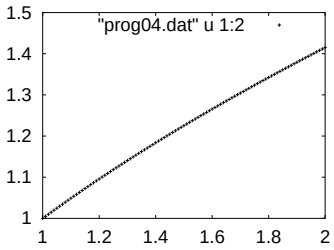
```
plot "prog04.dat", sqrt(x)
pause -1
```

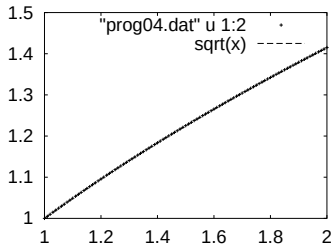
そして、このスクリプトファイル (操作をまとめて行なうためのファイル) で gnuplot を動かします。

```
ap1: gnuplot prog04.plt
```

先程と同じグラフが見えましたか？pause N は、N 秒間待ってからグラフを消せ、という命令です。N が負の場合は (コマンドをタイプした Window に戻って) どれかのキーを押すとグラフが消えます。







5　微分方程式 その 2: １変数２階微分方程式 (改良オイラー法)

今週は「簡単には解析的に解けない」微分方程式を解いてみましょう。振り子の問題です。

区分数 n を入力して、初期条件 $x(0) = 2.0,\ \dfrac{dx}{dt}(0) = 0$ のもとに、

$$m\ell \frac{d^2x}{dt^2} = -mg \sin x$$

を、時刻 $0 \leq t \leq 10$ において、改良オイラー法で解き、各時刻 t における位置 (実際は角度) x、速度 (実際は角速度) v、エネルギー E を出力するプログラムを作りましょう。ただし、$g/\ell = 9.8$ として下さい。

このプログラムでは $v = \dfrac{dx}{dt}$ とおいて、

$$\frac{dx}{dt} = v, \qquad \frac{dv}{dt} = -\frac{g}{\ell} \sin x$$

と変形し、x, v を連立して解いています。

5.1　改良オイラー法で微分方程式を解くプログラム (prog05.f) の中心部分

ここではプログラムの中心部分のみを示します。前後の部分は自分で書いてみて下さい。Homepage にはプログラム全体を載せてありますので、つまったら見ても構いませんが、まずは自分で考えて書いてみましょう。

```
 1        program prog05
 2        .....
 3  c Parameter inputs
 4        write(*,*) '# n=?'
 5        read(*,*) n
 6  c
 7        dt = .....
 8  c
 9           write( *,800) t,x,v,energy(x,v)
10        do i = 1,n
11           v0 = v                ! dx/dt at (x,v)
12           f0 = f(x)             ! dv/dt at (x,v)
13  c
14           x1 = x + dt*v0
15           v1 = v + dt*f0        ! dx/dt at (x1,v1)
16           f1 = f(x1)            ! dv/dt at (x1,v1)
17  c
18           dx = (v0 + v1)*dt/2 ! Improved Euler
19           dv = (f0 + f1)*dt/2 ! Improved Euler
20  c
21           t = t + dt
22           x = x + dx
23           v = v + dv
24           write( *,800) t,x,v,energy(x,v)
25        end do
26        .....
27  c
```

```
 800  format(4(1x,f15.7))
      end
c ***********************************************************************
      function f(x)
c ***********************************************************************
      ....
      f = .....
      end
c ***********************************************************************
      function energy(x,v)
c ***********************************************************************
      implicit real*8 (a-h,o-z)
      g = 9.8d0          ! gravitation constant
      energy = ....
      end
```

5.2 今日のポイント

- 2階常微分方程式は、2変数の1階常微分方程式へ。
- 常微分方程式の (数値的な) 解き方
  1. オイラー法
  2. 修正オイラー法
  3. 改良オイラー法: オイラー法で進ませた点での微係数との平均

$$\frac{dx}{dt} = f(x) \ \rightarrow \ x_k = x_{k-1} + \frac{f(x_{k-1}) + f(x')}{2}\,\Delta t\ , \quad x' = x_{k-1} + f(x_{k-1})\,\Delta t\ .$$

  4. Heun の公式

5.3 ファイルからの入力とファイルへの出力、gnuplot でグラフを描く

プログラムができたら、ファイルから入力して計算を実行してみましょう。

- まず、n の値を書いたファイル prog05.in を mule 等で作って下さい。例えば n=2000 であれば、2000 と1行だけ書きます。(その後で改行だけはしておいて下さい。) これを save してから、プログラムをファイルから入力して実行してみましょう。下の例では、通常はキーボードから入力する n の値を prog05.in というファイルに書いてある内容から read し、prog05.dat というファイルに write しています。
  既に prog05.dat というファイルがある場合には、消して (rm prog05.dat) から実行して下さい。

```
ap1 xx: f90 prog05.f
ap1 xx: a.out < prog05.in > prog05.dat
```

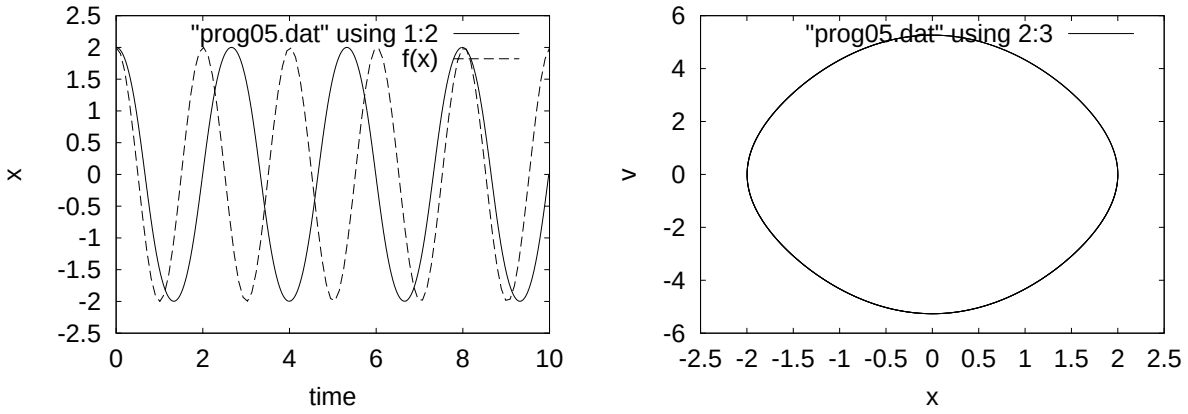
- さて、prog05.dat というファイルに、結果が出力されていますか？これが確認できれば、グラフを書いてみましょう。今週の最後のページにある gnuplot のスクリプト・ファイル prog05.plt を作って下さい。
  最初の数字は行数なのでファイルに書いてはいけません。また、これは fortran program ではないので、最初の 6 つのスペースは必要はありません。また、# で始まる行はコメントですから、書かなくても構いません。
  ファイルが出来たら save してから、グラフを描いてみましょう。

```
ap1 xx: gnuplot prog05.plt
```

ファイルを書くのが面倒であれば、今回はコピーして使っても構いません。(ただしファイルの中身は読んで下さい。)

```
ap1 xx: cp /www1/s-0007/PRINT/lesson05/prog05.plt .
ap1 xx: gnuplot prog05.plt
```

グラフは見えましたか？



このように redirect (<, > によって入力・出力先 (direction) を変える) によるファイルへの出力を使うと

- プログラム中に open 文を書かなくてもよい、
- 自分の好みのファイル名を実行時に選べる、
- テスト段階では画面に、完成すればファイルに、という出力の切替えが簡単、

などのメリットがあります。 一方、

- 出力ファイル名が場合によって変わるので、どのファイルがどのプログラムの出力かを覚えておく (あるいは、出力時に計算の内容まで書かせておく) 必要がある、
- 既に `prog05.dat` というファイルがある場合には、いったん消して (`rm prog05.dat`) から出力する必要がある。

というデメリットもあります。場合により、open 文と使い分けて下さい。

5.4 演習問題

(1) 今回のプログラムの結果はグラフに描けましたか。いろいろな初期条件でプログラムを走らせてみよう。また、どの程度エネルギーが保存されているか確かめてみよう。ファイル名は `ex051.f` として下さい。

(2) 次の 2 階微分方程式を解くプログラムを作ってみよう。

$$m\frac{d^2x}{dt^2} + \gamma\frac{dx}{dt} + m\omega^2 x = F\cos t$$

どっかでみたことある方程式ですね。$m = F = 1$ として、初期条件と ω, γ をいろいろ変えて解いた結果をグラフに描いてみよう。力学で勉強したような答えになりましたか。ファイル名は `ex052.f` として下さい。

(3) (2) で作ったプログラムを、サブルーチンを使って書き換えてみよう。位置 x, 速度 v, 時刻 t, 微小時間 dt, 摩擦係数 gam, 固有振動数 omg を与えて、1 段階の時間発展を求めるサブルーチン、

```
subroutine impeul(x,v,t,dt,gam,omg)
```

を作り、これを使って上の微分方程式を作るプログラムを作って下さい。ファイル名は ex053.f として下さい。

prog05.plt

```
 1  # このシャープマークの後はコメントです。無くても構いません。
 2  # また空行があっても構いません。
 3
 4  # 1 番目のグラフ: 時間の関数として x がどのように変化するかグラフを描きます。
 5  # set xlabel 'label' で x 軸の名前を入れられます。
 6  set xlabel 'time'
 7  set ylabel 'x'
 8
 9  # 高校の時に習ったように、sin(x) -> x と近似した時の解と比較するため、
10  # 近似解 f(t) を定義します。
11  # グラフでは横軸は x なので、f(x) として描く必要があります。
12  g=9.8
13  xi=2.0
14  f(t) = xi*cos(sqrt(g)*t)
15
16  # plot 'file 名' using n:m で、n カラム目と m カラム目が x, y となります。
17  # plot 'file 名' with lines で、線でグラフを描きます。
18  plot "prog05.dat" using 1:2 with lines, f(x)
19
20  # pause -1 は、キーボードから何かの入力（改行キーでよい）があるまで
21  # グラフを消さないことを意味します。
22  pause -1
23
24  # 2 番目のグラフ: 位置（実際は角度）と速度（実際は角速度）が
25  # どのような関係にあるか調べます。
26  set xlabel 'x'
27  set ylabel 'v'
28  plot "prog05.dat" using 2:3 with lines
29  pause -1
30
31  # 3 番目のグラフ: エネルギーがどの程度保存されているか調べます。
32  set xlabel 'time'
33  set ylabel 'Energy'
34  plot "prog05.dat" using 1:4 with lines
35  pause -1
```

6　微分方程式 その 3: ２変数２階微分方程式 (ケプラー問題)

Kepler 運動の方程式

$$\frac{\mathrm{d}^2\vec{r}}{\mathrm{d}t^2} = -\frac{\vec{r}}{r^3}$$

を改良 Euler 法で解くプログラムを作りましょう。平面上での運動を仮定し、初期条件は

$$\vec{r}_0 = (4, 0), \quad \vec{v}_0 = (0, 0.4)$$

としてください。また、時刻 t = 50 までを n = 2000 ステップに分割して解いて下さい。

6.1　ケプラー問題を改良オイラー法で解くプログラム (prog06.f) の中心部分

このプログラムではで dt の時間だけ 1 ステップ進める subroutine onestep を使って運動方程式を解いています。また、力の 2 つの成分 (fx, fy) を求める必要があるので、function の代わりに subroutine force によってまとめて計算しています。

ここではプログラムの中心部分のみを示します。... の部分は自分で書いてみて下さい。Homepage にはプログラム全体を載せてありますので、つまったら見ても構いませんが、まずは自分で考えて書いてみましょう。

```
 1        program prog06
 2  c
 3  c     Kepler Motion in the x and y coordinate system
 4  c     by using the improved Euler method
 5  c     subroutines:
 6  c        onestep(x,y,vx,vy,dt)
 7  c        force(x,y,fx,fy)
 8  c        engang(x,y,vx,vy,eng,ang)
 9  c
10        .......
11
12  c Initial Condition
13        t  = ti
14        .......
15  c
16           call engang(x,y,vx,vy,eng,ang)
17           write(*,800) x,y,t,eng,ang
18        do i = 1,n
19           call onestep(x,y,vx,vy,dt)
20           t = t + dt
21           .......
22        end do
23  c
24   800  format(5(1x,f15.7))
25        end
26  c **********************************************************************
27        subroutine onestep(x,y,vx,vy,dt)
28  c **********************************************************************
29        implicit real*8 (a-h,o-z)
30  c
31  c Force at x,y
32        call force(x,y,fx,fy)
```

```
33  c
34  c Trial Propagation and Force at x,y
35        x1  = x + vx*dt
36        y1  = y + vy*dt
37        vx1 = vx + fx*dt
38        vy1 = vy + fy*dt
39        call force(x1,y1,fx1,fy1)
40  c
41  c Calculate the shifts of x,y,vx,vy in Improved Euler method
42        dx =  (vx + vx1)*dt/2   ! Improved Euler
43        dy =  (vy + vy1)*dt/2   ! Improved Euler
44        dvx = (fx + fx1)*dt/2   ! Improved Euler
45        dvy = (fy + fy1)*dt/2   ! Improved Euler
46  c
47  c Real Propagation in Improved Euler method
48        x = x + dx
49        y = y + dy
50        vx = vx + dvx
51        vy = vy + dvy
52  c
53        end
54  c ***********************************************************************
55        subroutine force(x,y,fx,fy)
56  c ***********************************************************************
57        implicit real*8 (a-h,o-z)
58        r = ......
59        fx = .....
60        end
61  c ***********************************************************************
62        subroutine engang(x,y,vx,vy,eng,ang)
63  c ***********************************************************************
64        implicit real*8 (a-h,o-z)
65        eng = (vx*vx+vy*vy)/2.....
66        ang = ....
67        end
```

6.2 今日のポイント

今週のプログラムには、FORTRAN の文法上、特に新しい所はありません。これまでの知識を使って取り組んでみましょう。

6.3 gnuplot でグラフを描く

この例では質量や重力定数は 1 とおいています。したがって全エネルギー E と角運動量の大きさ L は

$$E = \frac{|\vec{v}|^2}{2} - \frac{1}{r}, \quad L = |\vec{r} \times \vec{v}| = x \cdot v_y - y \cdot v_x$$

となります。これらは保存量ですから（式の上で時間微分をとって確かめましょう)、数値解がどの程度正しいかの目安になります。

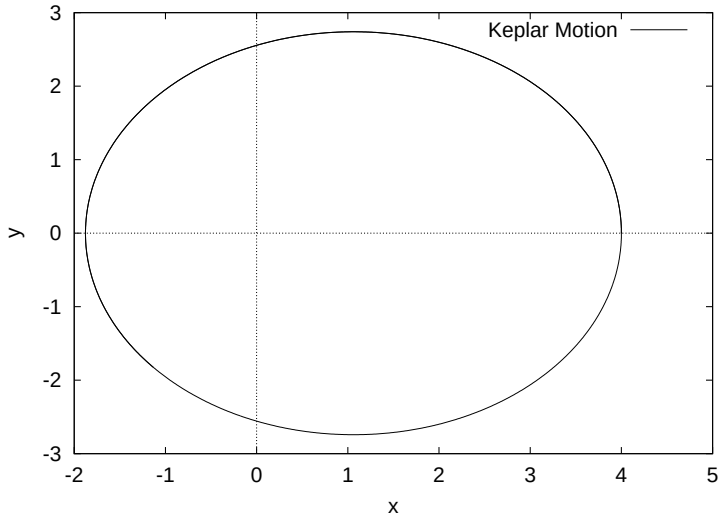
力学の教えるところでは、$E > 0$ のとき双曲線、$E = 0$ のとき放物線、$E < 0$ のとき楕円の軌道を描きます。今週作ったプログラムでの初期条件ではどのような軌道を描くのか、グラフをかいて調べてみましょう。まず、先週と同様にプログラムの計算結果をファイル (prog06.dat) に出力して下さい (今回は入力が無いので a.out > prog06.dat です)。その後、Homepage の指示に従うか、あるいは次のスクリプト・ファイルを作って、

```
ap1 xx: gnuplot prog06.plt
```

とすればグラフが見えます。

prog06.plt

```
 1  set xzeroaxis # x 軸を描きます。
 2  set yzeroaxis # y 軸を描きます。
 3  # のマークは行の途中でもその後がコメントになります。つまり # 以降はなくてよい。
 4  set xlabel "x" # x 軸の説明として x と書きます。
 5  set ylabel "y" # y 軸の説明として y と書きます。
 6
 7  # u で using を、w l で with lines を表します。
 8  # using と with の間に title "..." といれると、線の説明を入れられます。
 9  plot "prog06.dat" u 1:2 title "Keplar Motion" w l
10  pause -1
```



6.4 余裕があればやってみよう。

ちなみに、この運動方程式を極座標表示を用いて解くプログラムも Homepage 上に用意しました。解読できるでしょうか。

6.5 演習問題

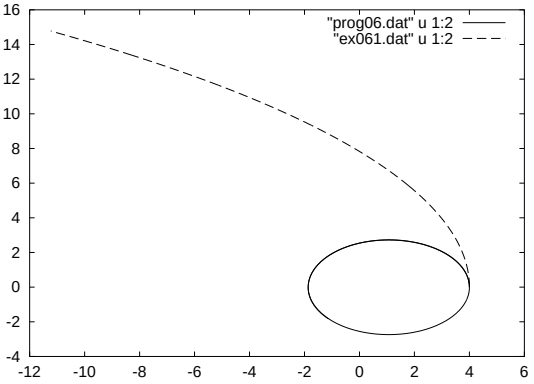
(1) 今回のプログラムの結果から、軌道をグラフに描いてみましょう。いろいろな初期条件でプログラムを走らせると期待どおりになりますか。また、どの程度 E や L が保存されているか確かめてみよう。

(2) いろいろな中心力を考えて解いてみよう。例えば

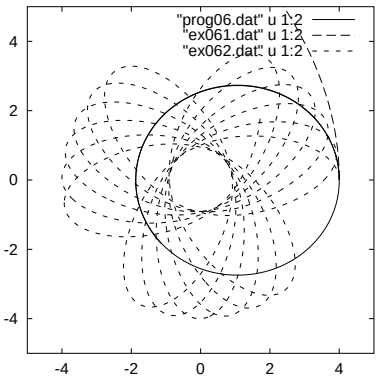
$$\frac{\mathrm{d}^2\vec{r}}{\mathrm{d}t^2} = -r^n\frac{\vec{r}}{r}$$

で $n = -2$ が Kepler 運動でした。$n = 1$ なら調和振動子です。他の n では？

`x = 4, vyi = 0.7` の場合との比較



`n = -1, x = 4, vyi = 0.4` の場合との比較。`t = 200` まで解いています。



6.6 プログラムが出来たら...

今日は `mule` から直接メールを送ってみて下さい。

(1) `mule` の画面から `C-x m` (mail mode) とすると、メールを送る画面になります。

(2) まず、`To:` の行に私の E-mail アドレスを書きます。次に改行して自分のアドレスに cc (carbon copy) を送る行 (`cc: s0300xxy`) を付け加えます。`Subject:` の行には `Prog06` と書いて下さい。

(3) `--text follows this line--` の行の下にカーソルを移動し、`C-x i` (insert file) から `prog06.f` を選択すると、プログラムが読み込まれます。

(4) 質問等を書くときには、プログラムの前 (ただし、 `--text follows this line--` の行の下) に書いて下さい。

(5) さて、送る直前の画面の状態は次のようになります。

```
To: Teacher@server
cc: s0300xxy
Subject: Prog06
--text follows this line--
質問。あそことここが分かりにくいです。
感想。北海道の冬は暑いですね。
      program prog06
c
c     Kepler Motion in the x and y coordinate system
c     by using the improved Euler method
c     subroutines:
c        onestep(x,y,vx,vy,dt)
c        force(x,y,fx,fy)
      ...
```

(6) メールを送るコマンドは `C-c C-c` です。取り消したい時には `C-c C-q` として下さい。

7　一般の方程式 — 2 分法と Newton 法

x に関する次の方程式を様々な $c(>0)$ に対して解くプログラムを作りましょう。

$$cx = \mathrm{e}^{-x}$$

7.1　2 分法で方程式を解くプログラム

解は 1 つで $0 < x < 1/c$ は明らかなので、その間を半分づつに狭くして、解の範囲をしぼっていく方法です。

```
 1        program prog07
 2  c
 3  c This program solves the equation c*x = exp(-x) by using NI-Bun-Ho.
 4        implicit real*8 (a-h,o-z)
 5  c Statement function (文関数)
 6        f(x)=c*x - exp(-x)
 7
 8  c 初期設定 ----------------------------------------------------------*
 9        eps = 1.0d-10       ! 数値解の誤差
10        do m = 1,50
11           c = m*0.1d0      ! さまざまな c について調べる。
12           a = 0.0d0        ! f(0) = -1, f(1/c)= 1 - exp(-1/c) > 0
13           b = 1.0d0/c      ! なので、f(x)=0 の解は 0<x<1/c の間にある
14  c (b-a) の区間を n 回 2 分割して解の範囲を探すので、
15  c 解の存在する領域の範囲は
16  c        (b-a)/2**n
17  c これを eps (解の誤差) より小さくするには
18  c        n > log((b-a)/eps)/log(2.0d0)
19           n = int(log((b-a)/eps)/log(2.0d0))+1
20           fa = f(a)
21  c 初期設定終り。 ----------------------------------------------------*
22
23  c 実際の計算 --------------------------------------------------------*
24           do i = 1,n
25              x = (a + b)/2.0d0
26              fx = f(x)
27              if(fx .eq. 0.0d0) goto 20    ! fx = 0 なら解が見付かった。
28              if(fx*fa .gt. 0.0d0) then           ! IF (...) THEN
29                 a = x            ! f(x)*f(a) > 0 ならば、解は x と b の間
30                 fa = fx          ! よって領域を (x, b) とする。
31              else                                ! ELSE
32                 b = x            ! f(x)*f(a) < 0 ならば、解は a と x の間
33              end if                              ! END IF
34           end do
35   20      continue                ! goto 20 でこの行に飛ぶ
36           write(*,100) c,(a+b)/2.0d0             ! 最終的な答えを書く
37        end do
38  c 実際の計算終り。 --------------------------------------------------*
39   100  format(2(1x,f15.7))
40        end
```

7.2 Newton 法で方程式を解くプログラム

与えられた x の値でのテイラー展開を用いると、真の解が $x + dx$ として、

$$f(x + dx) \simeq f(x) + f'(x)dx$$

となるので、dx の大体の大きさがわかります。これにより次の漸化式を用いて $n \to \infty$ の極限で真の解が得られるとするのが Newton 法です。

$$x_n = x_{n-1} - f(x_{n-1})/f'(x_{n-1})$$

```
 1        program prog072
 2  c
 3  c This program solves the equation c*x = exp(-x)
 4  c by using Newton's method.
 5        implicit real*8 (a-h,o-z)
 6  c
 7  c Statement function (文関数)
 8        f(x)=c*x - exp(-x)
 9        df(x)=c + exp(-x)
10
11  c 初期設定 ----------------------------------------------------------------*
12        eps = 1.0d-10              ! 数値解の誤差
13  c 初期設定終り。 ----------------------------------------------------------*
14
15  c 実際の計算 --------------------------------------------------------------*
16        do m = 1,50
17           c = m*0.1d0             ! さまざまな c について調べる。
18           x = 0.0d0
19           n1 = 1
20             do n=1,1000
21                fx = f(x)
22                n1 = n
23                if(abs(fx).lt.eps) goto 20 ! 誤差が小さければ終り。
24                dx = -fx/df(x)   ! Newton 法
25                x = x + dx
26             enddo
27   20        continue
28             write(*,100) c, x, n1 ! 最終的な答えと何回で収束したかを書く
29        end do
30  c 実際の計算終り。 --------------------------------------------------------*
31
32   100  format(2(1x,f15.7),1x,i4)
33        end
```

7.3 今日のポイント

- 方程式の数値的解き方 — 2 分法と Newton 法
- FORTRAN の if 文で使える主な関係演算子
  数値同士の比較を行います。x と y に変数や数値を記述します。

関係	数式	意味
`x.lt.y`	$x < y$	less than
`x.le.y`	$x \leq y$	less than or equal to
`x.gt.y`	$x > y$	greater than
`x.ge.y`	$x \geq y$	greater than or equal to
`x.eq.y`	$x = y$	equal to
`x.ne.y`	$x \neq y$	not equal to

- 文関数 (statement function)
  プログラムの中に直接関数の形を書く形式の関数文です。宣言文 (`implicit ...` や `real*8 a, b` などの文) の後、実行文 (`write, read, x=x+a, ...` などの文) の前に書かなければなりません。また、一行で書かなければならないので、複雑な関数には使えません。

- `goto` 文

7.4 演習問題

(1) Newton 法を用いて

$$y = \tanh x = \frac{\mathrm{e}^x - \mathrm{e}^{-x}}{\mathrm{e}^x + \mathrm{e}^{-x}}$$

の逆関数 $\tanh^{-1} y$ を求めるプログラムを Newton 法を用いて作ってみよう。但し、y の範囲を $0 < y < 1$ とします。

ヒント y を定数とみなして $f(x) = y - \tanh x$ とおけば、$f(x) = 0$ の解 x が $\tanh^{-1} y$ の値となります。初期値は $x = 0$ とすればよいでしょう。$|f(x)| < \epsilon$ を満たしたら解が見つかったことにします。ϵ は各自で与えて下さい。

(2) 空気抵抗を考慮した落体の運動を考えます。適当な条件の下では、速さ v で運動する半径 a の球形の物体には、速度の向きと逆方向に $\gamma a^2 v^2$ の空気抵抗が働きます（γ は定数)。このとき、高さ h から静かに物体を離すと着地するまでにどのくらいの時間がかかるでしょうか。

鉛直下向きに座標軸を取ると、この物体の運動方程式は、

$$m\frac{\mathrm{d}v}{\mathrm{d}t} = -\gamma a^2 v^2 + mg$$

となります。ここで m は物体の質量です。

1. $v(t)$ を初期条件 $v_0 = 0$ で解析的に解き、さらにそれを時間積分して、落下した距離 $x(t)$ の厳密解を求めましょう。
2. 着弾する時刻 t_1 は $x(t_1) = h$ を解くことによって得られます。これを数値的に求めるプログラムを作ってみよう。$g = 9.8, \gamma = 1$ として、あとはいろいろな a, m, h について解いてみよう。高さ 20m のビルから野球のボールとサッカーボールを落とすとどちらが何秒先に着地すると思いますか。

7.5 プログラムが出来たら...

今週も `mule` から直接メールを送ってみて下さい。メールの中身は `prog072.f` と質問、感想などとして下さい。

8　空気抵抗のある場合の物体の運動: 課題 その２

8.1　問題設定: 空気抵抗のある場合の物体の運動

空気抵抗を考慮した大砲の砲撃の問題を考えます。水平方向、垂直方向をそれぞれ x, y 方向とします。仰角 θ で大砲を初速度の大きさ v_0 で打ち出すとき、弾の運動方程式は

$$\begin{aligned} \frac{dx}{dt} &= v_x \ , & m\frac{dv_x}{dt} &= -\gamma a^2 v_x v \ , \\ \frac{dy}{dt} &= v_y \ , & m\frac{dv_y}{dt} &= -\gamma a^2 v_y v - mg \ , \end{aligned} \tag{1}$$

となるとします。ここで、γ は定数、a と m は弾の半径と質量で、v は速度の大きさ $v = \sqrt{v_x^2 + v_y^2}$ です。初期条件は

$$\begin{aligned} x(0) &= 0, & y(0) &= 0 \\ v_x(0) &= v_0 \cos\theta, & v_y(0) &= v_0 \sin\theta \end{aligned}$$

で与えられます。

1. 上の微分方程式を修正オイラー法で解き、y が負の値になるまで時間発展させるプログラム（`prog081.f`）を作ってください。ただし、$g = 9.8, m = 1.0$ として下さい。

   例えば、$\theta = \pi/4$, $v_0 = 20$, $a = 0.1$, $\gamma = 1$ としてプログラムを走らせると、弾の軌跡はどのようになりますか。$\gamma = 0$ としたとき、放物線になりますか？グラフを描いて確かめて下さい。

2. プログラムを変更して、θ を入力して、$y = 0$ となるときの x の値（これを $x_{\rm f}$ とします）を出力するようにしてください (`prog082.f`)。$y < 0$ になる直前の位置と直後の位置 (あるいはその一方) をうまく利用するとよいでしょう。

3. さらに、θ をいろいろ変えて、その都度の $x_{\rm f}$ を求め、θ と $x_{\rm f}$ の値を次々に出力するように変更してください (`prog083.f`)。そうすれば到達距離 $x_{\rm f}$ を仰角 θ の関数としてプロットしてみることができます。到達距離を最大にする仰角は、空気抵抗がない時 ($\theta = \pi/4$) に比べてどうなりますか。

8.2　課題レポート問題

8.2.1　課題内容

課題 その２
上記の問題設定のもとに、$v_0 = 30$, $a = 0.2$, $m = 1$, $\gamma = 1$ として、到達距離を最大にする仰角 θ を有効数字 3 桁まで求めて下さい。単位は degree でお願いします。
締め切りは 1 月 23 日 (必着) です。

8.2.2　課題レポートで提出するもの

あなたの作った「最終的なプログラム」と「最終的な答え」、そして「その答をどのように求めたか」をまとめてメールして下さい。

- 自動的な方法で動作確認をしますので、「最終的な答え」と「その答をどのように求めたか」は、プログラムの前に書き、かつ一桁目に日本語変換なしの英数字の C をつけて書いて下さい (Fortran でのコメント文になる)。
- 入力無しで動作するプログラム (`read` 文の無いプログラム) を送って下さい。(つまり、入力するべきパラメータなどはプログラム中に書いておいて下さい。)

- Subject （件名）は、日本語変換なしの英数字を使って rep08 として下さい。
  できたところまでで構いません。
  最後まで出来なかった人はどこまで出来たかの詳細を書いてメイルして下さい。

8.3 ヒント

8.3.1 時間発展の部分 (prog081.f)

時間発展 (Week10 の課題設定の 1.) の部分は、prog06.f をもとにして考えると分かりやすい。subroutine onestep, subroutine force を作っておくと後々便利である。

プログラム (prog081.f) の構成例

```
 1        program prog081
 2        .....
 3        ga = gam*a*a
 4        .....
 5  c IF GOTO LOOP Start
 6   10   continue
 7           call onestep(x,y,vx,vy,t,dt,ga)
 8           .....
 9           write(*,....) x,y,t
10        if( y .ge. 0.0d0 ) goto 10
11  c IF GOTO LOOP End
12        .....
13        end
14  c ***********************************************************************
15        subroutine onestep(x,y,vx,vy,t,dt,ga)
16  c ***********************************************************************
17        .....
18        call force(vx, vy, fx, fy, ga)
19        .....
20        call force(vx1,vy1,fx1,fy1,ga)
21        .....
22        dx =  .....
23        .....
24  c Real Propagation in Improved Euler method
25        x = x + dx
26        .....
27        t = t + dt
28        end
29  c ***********************************************************************
30        subroutine force(vx,vy,fx,fy,ga)
31  c ***********************************************************************
32        .....
33        fx = .....
34        .....
35        end
```

8.3.2 $y = 0$ となる $x = x_f$ を求める部分 (prog082.f)

上のプログラムで $y < 0$ となった最初の時間 $t = t_1$ は分かるので、そのあたりから時間を少しもどせばよいだろう。求めるのは $x = x_f$ だが、時間を媒介にして行なうのがよいと思われる。もどす時間のステップは、例えば、Newton 法で求められる。

当然ながら、前に作った subroutine (onestep, force) はそのまま使える。

プログラム (prog082.f) の構成例
(prog081.f のサブルーチン onestep, force を加えることがが必要)

```
1        program prog082
2        .....
3   10   continue
4           call onestep(x,y,vx,vy,t,dt,ga)
5           .....
6        if( y .ge. 0.0d0 ) goto 10
7   c
8        call solvey0(x,y,vx,vy,t,ga)
9        xf=x
10       write(*,100) theta, xf, t
11       ....
12       end
13  c ***************************************************************************
14       subroutine solvey0(x,y,vx,vy,t,ga)
15  c ***************************************************************************
16       .....
17  c
18       eps = 1.0d-10
19       do n = 1, 1000
20         if(abs(y).lt.eps) goto 20
21         dt = .....
22         call onestep(x,y,vx,vy,t,dt,ga) ! 時間を dt だけ進ませる
23       enddo
24         write(*,*) " I cannot find the position where y = 0."
25  20   continue
26  c
27       end
```

8.3.3 θ を変えて、それぞれの $x_{\rm f}$ を求める部分 (prog083.f)

まず、「θ をあたえて x_f を返す subroutine」

```
subroutine solvexf(v0,theta,ga,dt,xf,tf)
```

をつくっておくとよい。これは、大雑把にいえばprog082.f のメイン・プログラムを subroutine にしたものといえる。

プログラムの構成としては、メイン・プログラムで do loop により θ を次々と変化させて上の subroutine solvexf を call すればよい。

再び、当然ながら前に作った subroutine (onestep, force, solvey0) はそのまま使える。

プログラム (prog083.f) の構成例
(prog081.f のサブルーチン onestep, force、prog082.f のサブルーチン solvey0、を加えることがが必要)

```
 1        program prog083
 2        .....
 3        do i = 1, 89
 4          theta = dble(i)
 5          call solvexf(v0,theta,ga,dt,xf,tf)
 6          write(*,100) theta, xf, tf
 7        end do
 8        ...
 9        end
10  c ***********************************************************************
11        subroutine solvexf(v0,theta,ga,dt,xf,tf)
12  c ***********************************************************************
13        .....
14        pi = .....
15         t = 0.0d0
16         x = .....
17        .....
18   10   continue
19          call onestep(x,y,vx,vy,t,dt,ga)
20          .....
21        if( y .ge. 0.0d0 ) goto 10
22        call solvey0(x,y,vx,vy,t,ga)
23        xf = x
24        tf = t
25  c
26        end
```

8.3.4 $x = x_f$ を最大にする θ を求める部分（最終的なプログラム rep08.f）

おそらく $10^\circ < \theta < 90^\circ$ の範囲内で x_f は最大になるだろう。よって、「有効数字 3 桁」ということは、θ の誤差が $\Delta\theta < 0.1^\circ$ であればよい。より具体的には、0.01° 単位で x_f が最大となる θ をさがして、「四捨五入」すれば答えは求まる。

プログラムの構成としては、(1) θ を変化させて、`subroutine solvexf` を `call` し、(2) x_f が最大になる (これまでに求まっている x_f の最大値よりも大きな場合) を次々にさがし、(2) 最後に四捨五入して最終的な答えを出力する、プログラムをつくればよい。

8.4 今週やること。

プログラムが長くなってきたので、演習の Homepage での指示にしたがって `ftnchek` コマンドが使えるようにしておきましょう。また mule で行を表示させるようにしておいて下さい。また、前回と同様にグラフを Homepage に載せてみましょう。最後に、出席確認のために質問・コメントをメールして下さい。方法は問いません。`mule` から、あるいは PC からメールして下さい。

9 乱数発生のプログラム

区間 $(0,1)$ の一様乱数を発生してその分布を調べるプログラムを作りましょう。

9.1 互除法により発生させた乱数の分布を調べるプログラム

```
 1        program prog09
 2  c This program analyzes the distribution of random numbers [0,1)
 3  c generated by
 4  c     function rnd
 5  c
 6        implicit real*8(a-h,o-z)
 7        integer*4 iseed
 8  c dimension = vector
 9        dimension hist(50)
10  c Initialization
11        iseed = 123456789
12        n = 100000
13        dx = 1.0d0/50
14        do j = 1,50
15           hist(j) = 0.0d0
16        end do
17  c Make Histograms
18        do i = 1,n
19           x = rnd(iseed)
20           j = int(x/dx)+1
21           hist(j) = hist(j) + 1.0d0
22        end do
23  c Output
24        do j = 1,50
25           write(*,100) dx*(j-0.5d0),hist(j)/n/dx
26        end do
27   100  format(2(1x,f10.4))
28        end
29  c **********************************************************************
30        function rnd(iseed)
31  c **********************************************************************
32        implicit real*8(a-h,o-z)
33  c...32 bit (=4byte) machine
34        integer*4 iseed,il,ic,ih
35        il = 843314861
36        ic = 453816693
37        ih = 2**30
38        xmax = 2.0d0**31
39  c
40        iseed = iseed*il + ic
41        if(iseed.lt.0) iseed = (iseed + ih) + ih
42        rnd = iseed/xmax
43  c
44        end
```

9.2 今日のポイント

- 互除法による乱数の発生
- 配列の宣言と利用

```
dimension hist(m)
hist(i) = ..
```

- モンテカルロ積分

9.3 演習問題

(1) 今日のプログラムを平均値と分散を求めるように改良してみよう。

(2) ふたつの $(0,1)$ の一様乱数 x_1, x_2 から

$$y = \begin{cases} 1, & \sqrt{x_1^2 + x_2^2} \leq 1 \text{ のとき} \\ 0 & \sqrt{x_1^2 + x_2^2} > 1 \text{ のとき} \end{cases}$$

によって y を発生し、その平均値を求めてみよう。一番簡単なモンテカルロ積分のひとつです。答えはいくらになるべきでしょうか？また、いくらになりましたか？

(3) ふたつの $(0,1)$ の一様乱数 x_1, x_2 から

$$\begin{aligned} y_1 &= \cos(2\pi x_1)\sqrt{-2\log(x_2)} \\ y_2 &= \sin(2\pi x_1)\sqrt{-2\log(x_2)} \end{aligned}$$

によって y_1, y_2 を作り、その分布を調べてみよう。これらは平均 0、分散 1 の正規分布になることが証明できるので、正規乱数を発生させるのに使われる。

(4) 簡単なモンテカルロシミュレーションをしてみましょう。一個のさいころを振って六の目が 5 回でるまでには、平均何回さいころを振らなければならないでしょうか。もちろん答えは手計算ででますが。実際計算機にさいころを振らせてみましょう。

1. $n = 0$,$k = 0$ とする。
2. n を 1 増やす。
3. 確率 $1/6$ で k を 1 増やす。
4. $k = 5$ なら終わり。そうでなければ 2. へ戻る。

これで n がさいころを振った回数に相当します。これを何回か繰り返し、n の平均値を計算するプログラムを作ってください。

9.4 プログラムが出来たら...

演習問題の (2) を解いて `mule` から直接メールを送ってみて下さい。

10　乱数その 2: 熱平衡での運動量分布を求めるプログラム

2 次元の多粒子の衝突により、熱平衡にある粒子の運動量分布を求めるプログラムです。このとき、統計力学によれば運動量分布は

$$P(p_x) = \frac{\exp(-p_x^2/2mk_BT)}{\sqrt{2\pi mk_BT}}$$

となります。ここでは、質量を $m = 1$、ボルツマン定数を $k_B = R/N_A = 1$ (R は気体定数、N_A はアボガドロ数) とし、温度を $E = Nk_BT$ から求めて比較しています。

rnd 関数は先週のものを流用して下さい。

10.1　粒子衝突を考えて Boltzmann 分布を調べるプログラム

```
1         program prog10
2   C
3   C     ２次元の粒子衝突を Monte-Carlo 法でシミュレーションするプログラム
4   C     function rnd は前回のものを使って (コピー・ペースト) 下さい。
5   C
6         implicit real*8(a-h,o-z)
7         integer*4 iseed
8         real*8 hist(-50:50),p(2,10000)
9   C
10  C 準備 ----------------------------------------------------------------*
11  C パラメータ設定
12        iseed = 123456789
13        pi = 4*atan(1.0d0)
14        ncol = 100000
15        n = 10000
16        dx = 0.02d0
17  C ファイルを開く
18        open(16,file='prog10.dat')
19  C
20  C ヒストグラムの初期化
21        do j = -50,50
22           hist(j) = 0.0d0
23        end do
24  C 運動量の初期値設定
25        px = 0.0d0
26        py = 0.0d0
27        do i = 1,n
28           p(1,i) = rnd(iseed)
29           p(2,i) = rnd(iseed)
30           px = px + p(1,i)
31           py = py + p(2,i)
32        end do
33        px = px / n
34        py = py / n
35  C 運動量の平均を 0 にする。
36        do i = 1, n
37           p(1,i) = p(1,i) - px
```

```
38           p(2,i) = p(2,i) - py
39         end do
40   C
41   C 実際の衝突計算: ２次元の弾性散乱を ncol 回繰り返す。-----------------*
42   C
43         do k = 1, ncol
44   C 衝突する粒子対をランダムに選ぶ。
45           i1 =int(rnd(iseed)*n)+1
46           i2 =int(rnd(iseed)*n)+1
47   C
48   C ２次元の弾性散乱
49           px = (p(1,i1) + p(1,i2))/2
50           py = (p(2,i1) + p(2,i2))/2
51           pr = sqrt((p(1,i1)-p(1,i2))**2+(p(2,i1)-p(2,i2))**2)/2
52   C 散乱の方向は乱数で選ぶ
53           theta = 2*pi*rnd(iseed)
54           p(1,i1) = px + pr*cos(theta)
55           p(2,i1) = py + pr*sin(theta)
56           p(1,i2) = px - pr*cos(theta)
57           p(2,i2) = py - pr*sin(theta)
58         end do
59   C
60   C 分布を求めて出力 ---------------------------------------------------*
61         energy = 0.0d0
62         do i = 1, n
63   C Px を dx ごとの短冊に割り当てる。
64           k = nint(p(1,i)/dx)
65            if(k.ge.-50.and.k.le.50) then
66               hist(k) = hist(k) + 1.0d0
67            endif
68           energy = energy + (p(1,i)**2 + p(2,i)**2)/2
69         end do
70         T = energy / n
71   C 結果出力
72         do j = -50, 50
73           x = dx*j
74           write(16,100) x, hist(j)/n/dx, exp(-x**2/2/T)/sqrt(2*pi*T)
75         end do
76   C フォーマット文
77    100  format(3f10.4)
78         end
```

10.2 今日のポイント

- 配列宣言のより一般的な形式

```
dimension hist(n:m)
```

- ２次元配列の宣言と利用

```
dimension p(2,10000)
p(1,i) = ..
```

- ２次元での粒子の弾性散乱

10.3 演習問題

1. 今日のプログラムで、衝突の回数 `ncol` をいろいろ変えて、分布を調べてみよう。粒子あたり何回程度衝突すれば、熱平衡の分布とみなせるでしょうか？
2. 今日のプログラムを 3 次元の衝突問題が扱えるように書き換えてみよう。