数値計算入門

http://kurasawa.c.ooco.jp

目 次

0 GCC の使い方など **1**

1 超越方程式：ニュートン法 **3**

2 直交多項式：エルミート, ルジャンドル多項式 **7**

3 数値積分：台形則とシンプソン則 **9**

4 連立一次方程式：ガウスの消去法 **11**

5 常微分方程式：ルンゲ・クッタ型公式 **14**

6 行列の固有値問題 : ヤコビ法 **18**

7 シュレディンガー方程式の数値解法 **22**

7.1 微分方程式を解く 22

7.2 行列の対角化 26

8 モンテカルロ法 **28**

付録 A 数値計算の結果を図で表わす：TEX の tpic special **34**

付録 B 数値計算の結果を図で表わす：PostScript **40**

付録 C Mule(Emacs) のコマンド一覧 **45**

はサンプルプログラムにバグがあることを示す。

meadow (emacs, mule) の使い方は**付録 C** を参考

© 倉澤 治樹

参考書

- 柴田望洋, 新版 明解 C 言語 入門編 (ソフトバンク)
- 浦 昭二, 原田 賢一：C 入門 (培風館)
- 川上 一郎：理工系の数学入門コース 8 数値計算 (岩波書店)
- 戸川 隼人：UNIX ワークステーションによる科学技術計算ハンドブック［基礎篇 C 言語版］(サイエンス社)
- NUMERICAL RECIPES in C [日本語版] (技術評論社)

0 GCC の使い方など

GCC の使い方

test.c というファイルをコンパイルするには

```
gcc -o test test.c
```

と入力する。UNIX の場合にはオプション -lm が必要かもしれない。正常にコンパイルが終了したら, cygwin 版の GCC の場合には実行ファイル test.exe, Unix の場合は test が生成される。

GCC は実行するときに多くのオプションがある。よく使うオプションを挙げておく。

-v	コンパイルの途中経過を表示する。
-c	リンクしないで xxxx.o を作る。
-o xxx	最終出力ファイルを xxx にする。cygwin 版の GCC では xxx.exe を生成する。
-g	デバッグ情報を含める。
-lxxx	ライブラリ libxxx.a をリンクする。例えば -lm は libm.a をリンクする。libm.a は数学関数のライブラリである。
-s	実行ファイルからシンボルテーブルと再配置情報を全て削除する。実行ファイルは小さくなる。
-Wall	全ての警告メッセージ (あるいはエラー) を出す。

他に実行速度に関連するオプションとして

-On	O はオーでありゼロではない。n は 0 ～ 3。最適化のレベルを指定。数字が大きいほどレベルが高く, 生成した実行ファイルの実行速度は高速になる。
-ffast-math	実行速度最適のため ANSI や IEEE の規則や仕様を破ることを GCC に許す。例えば、sqrt 関数の引数が負にならないと仮定する。
-malign-double	double, long double, long long の計算が Pentium で若干速くなる
-funroll-loops	ループを展開する
-fno-math-errno	errno をセットしない（多少速くなる）

高速で実行サイズの小さな実行ファイルを作るには

```
gcc -s -O3 -ffast-math -o test test.c
```

とすればよい。cygwin 版の場合 -mno-cygwin を付けると更に高速化する。

Windows の場合, 次の 1 行

```
gcc -s -O3 -ffast-math -mno-cygwin -o %1 %1.c
```

からなる拡張子が bat であるテキストファイル, 例えば cc.bat を作成すれば

```
cc test
```

と入力すると

```
gcc -s -O3 -ffast-math -mno-cygwin -o test test.c
```

が実行され test.exe が生成される。入力の手間が軽減する。

コマンド行の引数をプログラムに渡す

C 言語では main 関数で

```
main( int argc, char *argv[] )
```

とすると argc にプログラムを呼び出したコマンド行の引数の個数, argv に引数の文字列を指すポイ

ンタが入る (argc, argv は argment count, argment vector の意味で慣例的に使われるが, 別の変数名でもよい)。例えば, test.exe というプログラムを test abc 12 xyz で起動すると argc は 4 で

```
argv[0]="test"   argv[1]="abc"    argv[2]="12"    argv[3]="xyz"
```

になる。規約により argv[0] は起動したプログラム名であり, 実際の引数は argv[1] から始まる。

下記のプログラムを引数を付けて実行してみよ。

```
#include <stdio.h>

int main( int argc, char *argv[] )
{
    int i;
    for(i=0; i<=argc-1; i++ )
        printf("argv[%d]=%s¥n",i,argv[i] );
    return 0;
}
```

プログラムに数値を渡したい場合, 引数に例えば 12 を与えても, プログラムに渡されるのは文字列であるため, 文字列を数値に変換する必要がある。このための関数が C 言語では用意されている。

文字列 → 整数　int atoi(char *s)
文字列 → 実数　double atof(char *s)

argv[1]="12" のとき n=atoi(argv[1]) とすると n は整数値 12 になる。これらの関数を使うときはヘッダ stdlib.h を include する。

プログラム中で別のプログラムを実行する

数値結果を TEX ファイルに出力した場合, プログラム終了後コマンド行で例えば platex test と入力して TEX ファイルをコンパイルする必要がある。プログラム作成中には, これを何度も実行することになり結構面倒くさい。そこで, system 関数 int system(char *s) を使ってこの手間を省こう。system 関数を使う場合にもヘッダ stdlib.h を include する。実行したいコマンドを文字列 s に渡すだけである。使い方は, 例えば

```
#include <stdio.h>
#include <stdlib.h>
.....
int main(){
    .....
    fptpic=fopen("test.tex","w");
    .....
    fclose(fptpic);
    system("platex test");
    .....
```

である。このプログラムを実行すれば, 自動的に platex test も実行される。

1　超越方程式：ニュートン法

方程式 $f(x)=0$ を解くためには, 漸化式

$$x_0 = a, \quad x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{1.1}$$

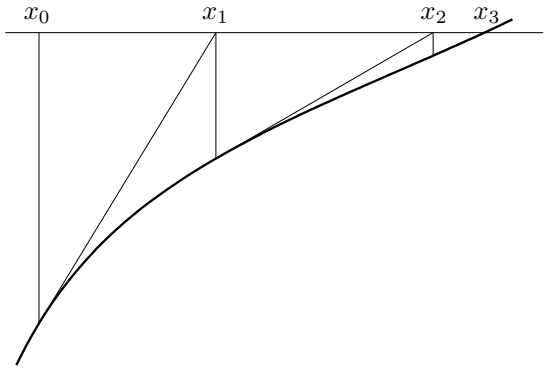
を適当な初期値 a に対して解いて, 数列 $\{x_n\}$ を構成する。初期値 a がうまく与えられれば, この数列は上述の方程式の１つの解 α に収束する。この方法をニュートン法あるいはニュートン・ラフソン法という。漸化式 (1.1) の意味とその収束の様子は, 次の図と式を見れば一目瞭然である。点 $(x_0, f(x_0))$ を通る接線の方程式は

$$y - f(x_0) = f'(x_0)(x - x_0)$$

である。この接線と x 軸との交点の x 座標を x_1 とすると

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{1.2}$$

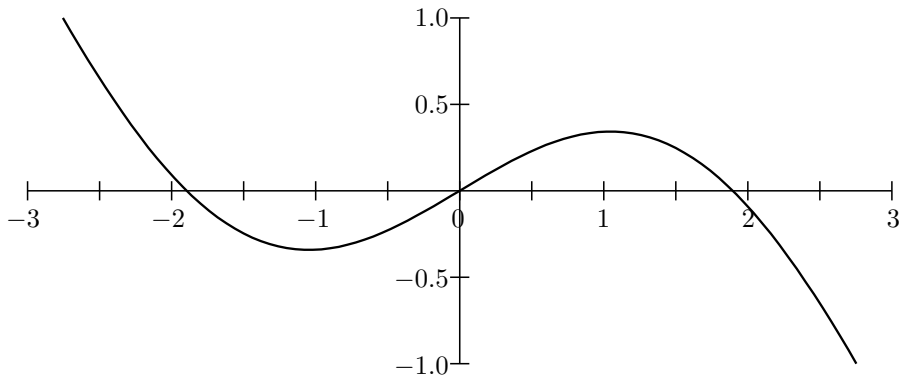
これを繰り返せば上記の漸化式 (1.1) が得られる。



x_1 を x_0 に代入して再び (1.2) を使うと, 得られた x_1 は実は x_2 である。これから分かるように, プログラムする場合 $x_0, x_1, \cdots, x_n$ のすべての変数を記憶する必要はなく, 2 つの変数を用意すれば十分である。

収束の判定は相対誤差 $|(x_n - x_{n-1})/x_n| < \varepsilon$ で判断する。10 進で m 桁の精度が欲しいときは $\varepsilon = 10^{-m}$ である。ところで, $|x_n|$ が非常に小さいときには, 左辺が桁溢れを起こすため判定できない。この場合には絶対誤差 $|x_n - x_{n-1}| < \varepsilon_A$ で判定する。ε_A としては 0 以外で扱える絶対値最小の値を使う。$\varepsilon_A = 10^{-70}$ としておけばよい。以上から $|x_n - x_{n-1}| < \varepsilon|x_n| + \varepsilon_A$ を収束の条件とする。多くの場合, $n \sim 10$ で eps$= 10^{-8}$ 程度の精度が得られるが, 解が求まらない場合も考慮して, n がある値 (例えば 20) 以上になったらプログラムを強制終了する。

問 1.1　$\sin x - x/2 = 0$ の解を求めるサンプルプログラムを次に示す。プログラムソースの各行を理解した上で実行させてみよ。下図に $y = \sin x - x/2$ を図示する。初期値 x_0 に依存してどの解に収束するか, 予想してみよ。また, $\tan x - 1/x = 0$ の解を求める様にサンプルプログラムを変更せよ。この場合, 解は無限個ある。



```
#include <stdio.h>
#include <math.h>
#define EPSA 1.e-70
double f( double x );
double df( double x );
int main()
{
    double x0, x1, eps;
    int n, nmax;
    eps=1.e-8;
    nmax=20;
    printf("初期値 x0 = ");
    scanf("%lf",&x0);   /*  l は一ではなくエル  */
    for( n=1; n<=nmax; n++ ){
        x1 = x0 - f(x0)/df(x0);
        if( fabs(x1-x0) < EPSA + eps*fabs(x1) ){
            printf("---------------------------------------¥n");
            printf("%2d %15.8e %15.8e¥n", n, x1, f(x1) );
            return 0;
        }
        x0=x1;
        printf("%2d %15.8e %15.8e¥n", n, x1, f(x1) ); /* 途中経過を表示 */
    }
    printf("解が見つかりません !!¥n");
    return 0;
}

double f( double y )
{
    return sin(y)-y/2;
}

double df( double y )
{
    return cos(y)-1/2;
}
```

問 1.2 上のサンプルプログラムでは, ニュートン法は main 関数に埋め込まれている。この部分を独立した関数にすると次のようになる。このようにすると, 解を求めたい関数 $F(x)$, $F'(x)$ を用意すれば, ニュートン法の部分を修正せずに, 様々な方程式の解をえることができる。1 つのプログラムで $\sin x - x/2 = 0$ と $\tan x - 1/x = 0$ の解を求めよ。

`double (*func)( double x )` は `func` が「1 つの `double` 型引数をもち `double` の値を返す関数」へのポインタであることを表す。`double *func( double x )` は `func` が `double` 型変数のポインタを返す関数ということになり意味は全く違う。最初は理解しずらい点であるが, こんなものと思って使いましょう。

```
#include <stdio.h>
#include <math.h>
#define EPSA 1.e-70
double f( double y );
double df( double y );
double newton( double (*func)( double x ), double (*dfunc)( double x ),
               double x0, double dx );
```

```
int main()
{
    double y0, y1, eps;
    eps=1.e-8;
    printf("¥n初期値 ==> ");
    scanf("%lf",&y0);
    y1 = newton( f, df, y0, eps );
    printf("%15.8e  %15.8e¥n",y1, f(y1) );
    return 0;
}

double f( double x )
{
    return sin(x)-x/2;
}

double df( double x )
{
    return cos(x)-1/2;
}

double newton( double (*func)( double x ), double (*dfunc)( double x ),
               double x0, double dx )
{
    int n;
    double x1;
    n=0;
    while( n++ < 20 ){
        x1 = x0 - func(x0)/dfunc(x0);
        if( fabs(x1-x0)<EPSA+dx*fabs(x1) )
            return x1;
        x0=x1;
    }
    printf("解が見つかりません !!¥n");
    return x1;
}
```

問 1.3　(1.1) で微分 $f'(x)$ を差分

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

で近似して

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

という漸化式で解を求める方法を**セカント法**という。セカント法は $f'(x)$ を定義しなくてよいのが利点であるが, 初期値として x_0, x_1 の 2 点を与える必要がある。与えられた 2 点 x_0, x_1 から上の式により x_2 を求め, 次に x_1, x_2 から x_3 を求める, これを収束するまで続ける。サンプルプログラムをセカント法に変更して解を求めよ。

問 1.4　[量子力学] ポテンシャルが井戸型 $V(x)$

$$V(x) = \begin{cases} -V_0\,, & |x| < R \\ 0\,, & |x| > R \end{cases}, \quad V_0 > 0$$

の場合, エネルギー固有値 E (この場合 $-V_0 < E < 0$) は

$$\begin{aligned} &\psi(x) \text{ が偶関数のとき} \quad f(k) = k^2 - v_0^2 \cos^2 k = 0\,, \quad \tan k > 0 \\ &\psi(x) \text{ が奇関数のとき} \quad g(k) = k^2 - v_0^2 \sin^2 k = 0\,, \quad \tan k < 0 \end{aligned} \tag{1.3}$$

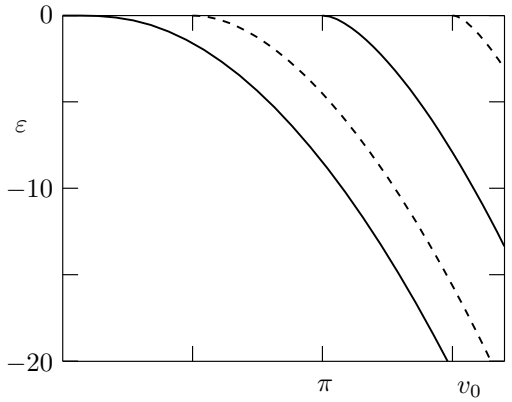
の解で与えられる [量子力学 A の講義ノート]。ただし

$$k = \sqrt{\frac{2m(V_0+E)}{\hbar^2}}\,R\,, \qquad v_0 = \sqrt{\frac{2mV_0}{\hbar^2}}\,R$$

v_0 を与えたときニュートン法により (1.3) を解き固有値 $\varepsilon = 2mR^2E/\hbar^2 = k^2 - v_0^2$ を求めよ。解は 1 つとは限らない。$f(k) = 0$ の解は直線 k/v_0 と曲線 $|\cos k\,|$ の交点で与えられるから

$$(n-1)\pi < v_0 \le n\pi\,, \quad n = 1,\, 2,\, \cdots$$

のとき n 個の解が存在する。$g(k) = 0$ についても同様である。$v_0 \to \infty$ の場合, (1.3) の k は解析的に求まる。これと数値結果を比較せよ。



(1.3) の $f(k)$ と $g(k)$ は k だけでなく v_0 にも依存するから関数 `newton` をこれに合わせて修正してもよいが, ここでは次のようにして無修正で済むようにする。

```
.....
double v0;
int main()
{
     .....
     v0 = ... ;
     .....
     newton( f, df, x0, eps );
     .....
}

double f( double x )
{
     return x*x-v0*v0*cos(x)*cos(x);
}
```

関数内 (`main` も一種の関数) で宣言した変数は, その関数内だけで有効で, 他の関数からは直接扱えない。このような変数を局所変数または自動変数という。一方, 関数の前 (上) で, 通常 `main` の前で, 変数を宣言することもできる。これを外部変数という。外部変数は全ての関数で共通に扱える変数である。上の例では, v_0 を表す変数 `v0` を外部変数として宣言し, `v0` に依存する関数 `f` と `df` が `newton` で使用される前に `v0` の値を与える。これにより `f` と `df` 内の `v0` も与えた値になる。

なお, 不必要に外部変数にすると, 関数ごとに本来は独立であるべき変数が共通になったりして, 思わぬ副作用をもたらす。また, 外部変数の変数名を簡単な変数名にすると, 局所変数と共通になることがあるから, 何を表しているか分かるような名前にするとよい。

2　直交多項式：エルミート, ルジャンドル多項式

エルミート (Hermite) 多項式 $H_n(x)$, ルジャンドル (Legendre) 多項式 $P_n(x)$ は漸化式

$$H_0(x) = 1\,, \quad H_1(x) = 2x\,, \quad H_n(x) - 2xH_{n-1}(x) + 2(n-1)H_{n-2}(x) = 0\,,$$

$$P_0(x) = 1\,, \quad P_1(x) = x\,, \quad nP_n(x) - (2n-1)xP_{n-1}(x) + (n-1)P_{n-2}(x) = 0$$

を満たす。これらの漸化式を用いて $H_n(x)$, $P_n(x)$ を求める関数

`double hermite( int n, double x )`　　`double legendre( int n, double x )`

を作成する。`hermite( int n, double x )` のサンプルを次に示す。

```
double hermite( int n, double x )
{
    int k;
    double y0, y1, y2;

    y0=1.;
    if( n==0 )
        return y0;
    y1=2.*x;
    for( k = 2; k<=n; k++ ){
        y2 = 2.*( x*y1 - (k-1)*y0 );
        y1 = y2;
        y0 = y1;
    }
    return y1;
}
```

問 2.1　$x = 0.05k$ ($k = 0, 1, 2, \cdots, 20$) に対して, 下記の具体例と数値が一致することを確かめよ。

$$H_5(x) = 32x^5 - 160x^3 + 120x\,, \qquad P_5(x) = \frac{1}{8}\Big(63x^5 - 70x^3 + 15x\Big)$$

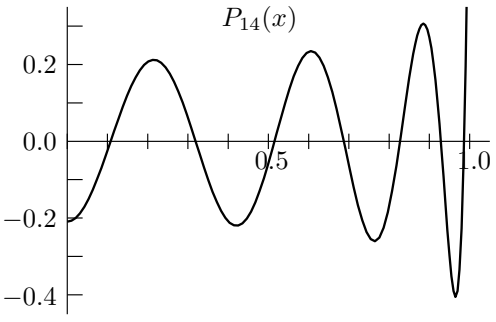
問 2.2　$P_n(x)$ は $-1 < x < 1$ に n 個の零点をもつ。$0 < x < 1$ における零点を次の方法ですべて求める。$0 \le x \le 1$ を $k_{\max}$ 個に分割する。$x_k = k/k_{\max}$, ($k = 0, 1, 2, \cdots, k_{\max}$) とするとき

$$P_n(x_k)P_n(x_{k+1}) < 0\,, \quad k = 0, 1, 2, \cdots, k_{\max} - 1$$

ならば $x_k < x < x_{k+1}$ に零点が存在する。初期値を $(x_k + x_{k+1})/2$ としてニュートン法で零点を求める。求まった解が $[n]/2$ 個より少ないなら $k_{\max}$ を大きくする。導関数 $P_n'(x)$ は

$$P_n'(x) = n\frac{P_{n-1}(x) - xP_n(x)}{1 - x^2}$$

である。$P_n(x)$ を図示し, 求めた解が $P_n(x) = 0$ を満たすことをグラフ上で確かめよ。1 章で作成したニュートン法のプログラム `newton` を無修正で使用するためには, **問 1.4** と同様にして, $P_n(x)$ の n を次のように外部変数として宣言すればよい。



```
.....
int nl;
int main()
{
     .....
     nl = 5;
     .....
     newton( f, df, x0, eps );
     .....
}

double f( double x )
{
     return legendre(nl,x);
}

double df( double x )
{
     return nl*( legendre(nl-1,x) - x*legendre(nl,x) )/(1-x*x);
}
.....
```

問 2.3［量子力学］規格化した一次元調和振動子の固有関数は

$$\psi_n(x) = \sqrt{\frac{\alpha}{\sqrt{\pi}\,2^n n!}}\, H_n(q)\, e^{-q^2/2}\,, \qquad q = \alpha x\,, \quad \alpha = \sqrt{m\omega/\hbar} \tag{2.1}$$

である。x と $x+dx$ に粒子を見出す確率は $\psi_n^2(x)\,dx$ になる。一方, 古典力学では x と $x+dx$ に粒子を見出す確率 $P_{\rm cl}(x)\,dx$ は, この区間を通過する時間 dt に比例するから

$$P_{\rm cl} \propto \frac{dt}{dx} = \frac{1}{v}\,, \qquad v = \text{粒子の速さ}$$

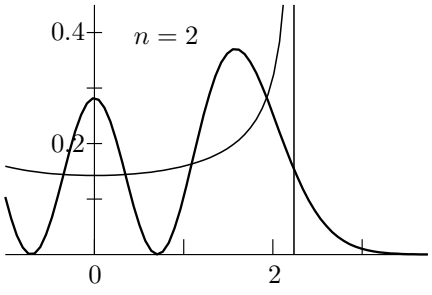
調和振動子の場合, $E = mv^2/2 + m\omega^2x^2/2$ より規格化定数を C として

$$P_{\rm cl} = \frac{C}{\sqrt{E - m\omega^2x^2/2}}\,, \quad \text{ただし} \quad E > m\omega^2x^2/2$$

したがって E が量子力学の固有値 $\hbar\omega(n+1/2)$ に等しいとき, 規格化した $P_{\rm cl}$ は

$$P_{\rm cl} = \frac{\alpha}{\pi\sqrt{2n+1-q^2}}\,, \qquad q^2 < 2n+1$$

になる [量子力学 A の講義ノート]。$n = 2$ の場合 $\psi_n^2(q)/\alpha$ と $P_{\rm cl}(q)/\alpha$ を図示すると右図のようになり q 依存性は異なる。$n \gg 1$ (例えば $n = 50$) の場合, 両者を比較せよ。



整数を 4 バイト (32 ビット) で表すシステムでは

$$13! = 6,227,020,800 > 2^{32} = 4,294,967,296$$

であるから, $n \geq 13$ では $n!$ を整数として扱うと不正確な値になる。$n \geq 13$ の階乗も扱う場合を考慮して, 整数 n に対して実数で $n!$ を返す関数を作成する。$n \leq 20$ に対して $n!$ を整数と実数で扱った場合を比較せよ。$x^y = \texttt{pow(x,y)}$

3　数値積分：台形則とシンプソン則

関数 $F(x)$ を a から b まで積分するとき, 区間 $[a,b]$ を n 等分して, その分点を

$$x_0 = a,\ x_1 = a+h,\ \cdots,\ x_{n-1} = a+(n-1)h,\ x_n = a+nh = b$$

とする。$F(x_k) = F_k$ で表わす。

台形則　区間 $[x_k, x_{k+1}]$ の積分値を台形の面積 $\dfrac{h}{2}\Big(F_k + F_{k+1}\Big)$ で近似し, これをすべての区間で足しあわせれば

$$\int_a^b F(x)\,dx \approx h\left(\frac{F_0+F_n}{2} + \sum_{k=1}^{n-1} F_k\right)$$

である。

シンプソン則　n を偶数にとる。$x_k \le x \le x_{k+2}$ (k は偶数) における $F(x)$ を 3 点 (x_k, F_k), (x_{k+1}, F_{k+1}), (x_{k+2}, F_{k+2}) を通る 2 次式

$$\begin{aligned} y(x) = {} & \frac{(x-x_{k+1})(x-x_{k+2})}{(x_k-x_{k+1})(x_k-x_{k+2})}F_k + \frac{(x-x_k)(x-x_{k+2})}{(x_{k+1}-x_k)(x_{k+1}-x_{k+2})}F_{k+1} \\ & + \frac{(x-x_k)(x-x_{k+1})}{(x_{k+2}-x_k)(x_{k+2}-x_{k+1})}F_{k+2} \end{aligned}$$

で近似する。$[x_k, x_{k+2}]$ での積分値は

$$\int_{x_k}^{x_{k+2}} dx\, y(x) = \int_{-h}^{h} dx\, y(x+x_{k+1}) = \frac{h}{3}\Big(F_k + 4F_{k+1} + F_{k+2}\Big)$$

である。したがって

$$\int_a^b F(x)\,dx \approx \frac{h}{3}\Big(F_0 + \sum_{k=1}^{n-1} c_k F_k + F_n\Big), \quad c_k = \begin{cases} 4 & k\text{が奇数} \\ 2 & k\text{が偶数} \end{cases}$$

台形則, シンプソン則により積分値を与える関数

```
double  daikei( double (*func)( ), double a, double b, int n )
double sympson( double (*func)( ), double a, double b, int n )
```

を作成せよ。台形則のプログラム例

```
double daikei( double (*func)( ), double a, double b, int n )
{
    int i;
    double dx, s;
    dx=( b - a )/n;
    s=0.5*func( a );
    for( i=1; i<n; i++)
        s+=func( a+dx*i );
    s+=0.5*func( b );
    return s*dx;
}
```

問 3.1　$n = 2, 4, 8, 16, \cdots, 1024$ として, 台形則とシンプソン則により次の積分値を求めよ。

$$\int_0^1 \frac{dx}{1+x} = \log 2\,, \qquad \int_0^1 \sqrt{1-x^2}\,dx = \frac{\pi}{4}$$

問 3.2　$n = 2, 4, 8, 16, \cdots, 1024$ として, 台形則とシンプソン則により次の直交性を確かめよ。

$$\frac{2k+1}{2}\int_{-1}^{1} dx\, P_k(x)\, P_\ell(x) = \delta_{k\ell}\,, \quad \frac{1}{\sqrt{\pi}\, 2^k\, k!}\int_{-\infty}^{\infty} dx\, \exp(-x^2) H_k(x) H_\ell(x) = \delta_{k\ell}$$

k と ℓ を外部変数として宣言し `daikei` と `sympson` は修正しない。2 番目の積分区間では被積分関数が実質的に 0 になる $|x| = x_{\max}$ で置き換える。たとえば $e^{-x_{\max}^2} \sim 10^{-16}$, つまり, $x_{\max} \sim 6$ である。$x_{\max} = 3$, 4, 5, 6, 7, 8 としたとき, 積分値がどう変化するか調べよ。この積分の場合, 台形則はシンプソン則より早く収束することに注意せよ。

問 3.3　[量子力学]　任意の関数 $\varphi(x)$ は調和振動子の固有関数 (2.1) を用いて

$$\varphi(x) = \sum_{n=0}^{\infty} c_n\, \psi_n(x)\,, \qquad c_n = \int_{-\infty}^{\infty} dx\, \psi_n(x)\varphi(x)$$

と展開できる。

$$\varphi(x) = \frac{\sqrt{\alpha}}{\pi^{1/4}}\, \exp\left(-\frac{\alpha^2(x-x_0)^2}{2}\right) = \frac{\sqrt{\alpha}}{\pi^{1/4}}\, \exp\left(-\frac{(q-q_0)^2}{2}\right)\,, \quad q = \alpha x\,, \quad q_0 = \alpha x_0$$

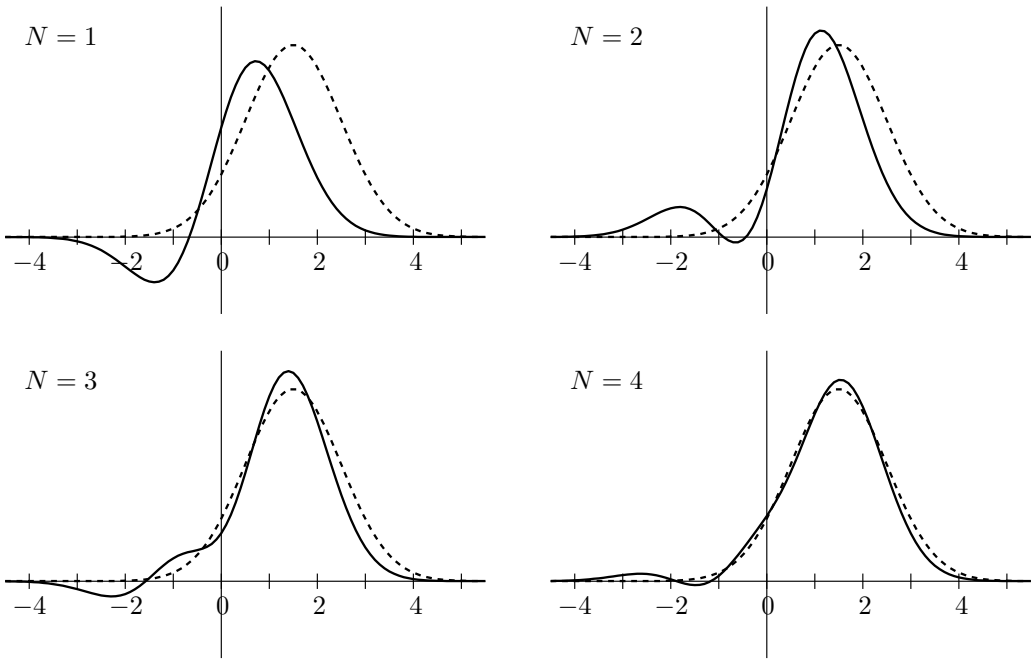
とする。ただし, α は (2.1) と同じである。c_n を台形則で数値積分して求めよ。この積分は解析的に行えて

$$c_n = \sqrt{\frac{\rho^n\, e^{-\rho}}{n!}}\,, \quad \text{ただし} \quad \rho = \frac{q_0^2}{2}\,, \qquad \sum_{n=0}^{\infty} c_n^2 = 1$$

である [量子力学 A の講義ノート]。

$$\varphi_N(x) = \sum_{n=0}^{N} c_n\, \psi_n(x)$$

とする。$q_0 = 3$ のとき, N を変化させて $\varphi_N(q)$ と元の関数 $\varphi(q)$ を図示し比較せよ。下図は $q_0 = 1.5$ の結果である。



4　連立一次方程式：ガウスの消去法

n 個の未知数 $x_1, x_2, \cdots, x_n$ についての連立一次方程式

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \cdots\cdots & \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned} \tag{4.1}$$

をガウスの消去法で解く。

最初に, 2 行目以下の方程式から x_1 を消去する。1 行目を p_i 倍して i 行目に加えると

$$\begin{aligned} &(a_{i1} + a_{11}p_i)x_1 + (a_{i2} + a_{12}p_i)x_2 + \cdots + (a_{ij} + a_{1j}p_i)x_j + \cdots + (a_{1n} + a_{1n}p_i)x_n \\ &= b_i + b_1p_i \end{aligned}$$

となる。したがって $p_i = -a_{i1}/a_{11}$ とすれば i 番方程式から x_1 が消去でき

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ a_{32}x_2 + \cdots + a_{3n}x_n &= b_3 \\ \cdots\cdots & \\ a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned} \tag{4.2}$$

となる。ただし

$$a_{ij} = a_{ij} - \frac{a_{i1}a_{1j}}{a_{11}} \quad (i, j = 2, 3, \cdots, n),, \qquad b_i = b_i - \frac{a_{i1}b_1}{a_{11}} \quad (i = 2, 3, \cdots, n)$$

上の 2 式は左辺の値を右辺で置き換えることを意味する。(4.2) の 2 行目以下の係数の値は (4.1) の係数とは異なっている。

第二段の消去は (4.2) の 3 行目以下から x_2 を消去する。(4.2) を導いたのと同様にすると

$$\begin{aligned} a_{11}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_1 \\ a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \\ a_{33}x_3 + \cdots + a_{3n}x_n &= b_3 \\ \cdots\cdots & \\ a_{n3}x_3 + \cdots + a_{nn}x_n &= b_n \end{aligned} \tag{4.3}$$

ただし

$$a_{ij} = a_{ij} - \frac{a_{i2}a_{2j}}{a_{22}} \quad (i, j = 3, 4, \cdots, n), \qquad b_i = b_i - \frac{a_{i2}b_2}{a_{22}} \quad (i = 3, 4, \cdots, n)$$

以下 $x_3, x_4, \cdots, x_{n-1}$ を消去することができる。

以上の消去の手順をまとめると

$$
\begin{aligned}
&k = 1, 2, \cdots, n-1 \\
&\{ \\
&\quad i = k+1, k+2, \cdots, n \\
&\quad \{ \\
&\quad\quad p = a_{kk} \qquad (4.4) \\
&\quad\quad j = k+1, k+2, \cdots, n \\
&\quad\quad \{ \\
&\quad\quad\quad a_{ij} = a_{ij} - a_{ik} a_{kj} / p \qquad (4.5) \\
&\quad\quad \} \\
&\quad\quad b_i = b_i - a_{ik} b_k / p \\
&\quad \} \\
&\}
\end{aligned}
$$

である。

$a_{kk} = 0$ になると (4.5) からこれ以上計算は継続できない。a_{kk} が厳密に 0 でなくても非常に小さな数の場合, (4.5) の右辺の第二項が非常に大きくなるため第一項 a_{ij} の情報が失われ (桁落ち), 新たに計算した a_{ij} は無視できない誤差を含んでしまうことがある (計算機上では数値は有限桁で表現されている)。この誤差をできるだけ小さくするには, p として $a_{ij}\,(i, j \geq k)$ の中で絶対値が最大の要素が k 行 k 列にくるように行と列を入れ替えればよい。p として選ぶ行列要素は重要な働きをする。この行列要素をピボットという。

通常は k 列だけの要素

$$a_{kk}\,,\ a_{k+1\,k}\,, \cdots,\ a_{nk}$$

の中から絶対値最大のものを探す。$a_{mk}\ (m > k)$ の絶対値が最大であったなら, k 行と m 行を入れ替える。この入れ替えはもとの連立方程式の k 行目の方程式と m 行目の方程式を交換することになるが, 方程式の順序を入れ替えても解は変わらない。この方法を部分選択ピボット法という。実際のプログラムでは (4.4) の前にピボット選択のルーチンが必要である。

x_{n-1} まで消去を繰り返すと, 連立方程式は

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\
a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \\
a_{33}x_3 + \cdots + a_{3n}x_n &= b_3 \qquad (4.6) \\
\cdots\cdots & \\
a_{nn}x_n &= b_n
\end{aligned}
$$

となる。したがって $x_n = b_n / a_{nn}$ と直ちに求まる。x_n が求まれば x_{n-1} は

$$x_{n-1} = \frac{b_{n-1} - a_{n-1\,n}\,x_n}{a_{n-1\,n-1}}$$

で与えられる。同様にして x_n から x_1 に向かって計算を行うとすべての解

$$x_k = \frac{1}{a_{kk}} \left(b_k - \sum_{j=k+1}^{n} a_{kj}\,x_j \right)$$

が求まる。x_n, x_{n-1}, $\cdots$, x_{k+1} はすでに求まっているから x_k を決定できる。

ガウスの消去法により連立一次方程式を解く関数プログラム

```
void gauss( double a[], double b[], int n )
```

を作成せよ。ただし

$$\texttt{n} = n\,, \qquad \texttt{a[i+n*j]} = a_{ij}\,, \qquad \texttt{b[i]} = b_i$$

である。解 x_i は `b[i]` に返す。a_{ij} 用に 2 次元配列を用いてもよいが, 1 次元配列ならば関数に引き渡すとき配列の大きさを明示する必要がないので, プログラムに柔軟性を持たせることができる。ただし, `a[]` の大きさは n^2+1 以上確保すること。また, `main` において n, a_{ij}, b_i の値を

$$\begin{array}{ccccc} n & & & & \\ a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array} \tag{4.7}$$

という形式で標準入力 (キーボード) から入力できるようにする。

問 4.1 次の連立方程式をガウスの消去法で解け。

$$\begin{cases} 8x_1+3x_2+x_3=14 \\ 3x_1+4x_2+x_3=13.8 \\ x_1+x_2+2x_3=9 \end{cases} \qquad \begin{cases} x_1+2x_2+x_3+5x_4=20.5 \\ 8x_1+x_2+3x_3+x_4=14.5 \\ x_1+7x_2+x_3+x_4=18.5 \\ x_1+x_2+6x_3+x_4=9 \end{cases}$$

得られた解が連立方程式を満たすことを確かめよ。

問 4.2 次のデータ

$$a_{ij} = \begin{cases} 4 & i=j \\ 1 & i=j\pm 1 \\ 0 & \text{その他} \end{cases}\,, \qquad b_i = \sum_{j=1}^{n} a_{ij} j$$

を (4.7) の形式で標準出力 (ディスプレイ) に出力するプログラムを作成せよ。ただし, n の値をコマンド行の引数として渡せるようにする。つまり, 実行プログラム名を `program` とした場合

```
program 20
```

と入力すると, n の値が 20 になるようにする。このプログラムの出力を DOS あるいは UNIX のパイプ機能によりガウス消去法のプログラムにデータとして読み込ませよ。ガウス消去法の実行プログラム名が `gauss` ならば

```
program 30 | gauss
```

と入力する。コマンド行の引数をプログラムに渡す方法は 1 ページを参照。

5　常微分方程式：ルンゲ・クッタ型公式

次のような 1 階の n 元連立微分方程式を考える。

$$\begin{aligned}\frac{dX_1}{dt} &= F_1(\,X_1,\,X_2,\,\cdots,\,X_n,\,t\,)\\ \frac{dX_2}{dt} &= F_2(\,X_1,\,X_2,\,\cdots,\,X_n,\,t\,)\\ &\vdots\\ \frac{dX_n}{dt} &= F_n(\,X_1,\,X_2,\,\cdots,\,X_n,\,t\,)\end{aligned}$$

この微分方程式はただ 1 つの変数 t の微分を含む。これを常微分方程式という。物理でよく出てくる 2 階の微分方程式, 例えば

$$\frac{d^2x}{dt^2} = -\,\omega^2\,x \tag{5.1}$$

は $X_1 = x$, $X_2 = dx/dt$ とすると

$$\frac{dX_1}{dt} = X_2\,,\quad \frac{dX_2}{dt} = -\,\omega^2\,X_1$$

と表わせるから, $F_1(\,X_1,\,X_2,\,t\,) = X_2$, $F_2(\,X_1,\,X_2,\,t\,) = -\,\omega^2 X_1$ とした 1 階の 2 元連立微分方程式に還元できる。独立変数 t を t_0 から一定の刻み幅 h で $t_k = t_0 + kh\,(\,k = 1, 2, 3, \cdots\,)$ と増やしていき, 各分点 t_k における未知の関数 $X_i(t_k)$ を数値的に求める。

t_k における導関数を

$$\left.\frac{dX_i}{dt}\right|_{t_k} \approx \frac{X_i(t_{k+1}) - X_i(t_k)}{h}$$

で近似すると, 元の微分方程式から

$$X_i(t_{k+1}) = X_i(t_k) + hF_i(\,X_1(t_k),\,X_2(t_k),\,\cdots,\,X_n(t_k),\,t_k\,)$$

となる。すべての $X_i(t_0)\,,(\,i = 1, 2, \cdots, n\,)$ を初期条件として与えれば, この式から

$$\begin{aligned}X_i(t_1) &= X_i(t_0) + hF_i(\,X_1(t_0),\,X_2(t_0),\,\cdots,\,X_n(t_0),\,t_0\,)\\ X_i(t_2) &= X_i(t_1) + hF_i(\,X_1(t_1),\,X_2(t_1),\,\cdots,\,X_n(t_1),\,t_1\,)\\ &\vdots\end{aligned}$$

と次々に求められる。これは常微分方程式の最も簡単な数値解法であり, **オイラー法**と呼ばれる。オイラー法は h のオーダーまで正しい結果を与える。

t_{k+1} の関数を求めるとき, t_k での傾き F_i より中間点の傾きを用いた方が正確である。そこで, ステップ幅 $h/2$ のオイラー法を組み合わせて ($t_{k+1/2} = t_k + h/2$)

$$X_i(t_{k+1/2}) = X_i(t_k) + \frac{h}{2}\,F_i(X_1(t_k), X_2(t_k), \cdots, X_n(t_k), t_k) \tag{5.2}$$

$$X_i(t_{k+1}) = X_i(t_k) + hF_i(X_1(t_{k+1/2}), X_2(t_{k+1/2}), \cdots, X_n(t_{k+1/2}), t_{k+1/2}) \tag{5.3}$$

として, $X_i(t_k)$ から $X_i(t_{k+1})$ を求める。これを**改良オイラー法**という。改良オイラー法は h^2 のオーダーまで正確である。

$X_i(t_k)$ から $X_i(t_{k+1})$ を求める関数プログラムを次のようにして作成する。

- $X_i(t_k)\,(\,i = 1, 2, \cdots, n\,)$ の値を 1 次元配列 `x[]` に入れる。`x[i]` $= X_i(t_k)$ である。`x[]` のサイズは $n + 1$ 以上である。

- $X_i(t_k)\,(i=1,2,\cdots,n)$ から $F_i(X_1(t_k),X_2(t_k),\cdots,X_n(t_k),t_k)$ を求める関数を別途用意する。この関数名を例えば func とすると

```
void func( double x[], double f[], double t )
```

である。f[] はサイズが $n+1$ 以上の 1 次元配列で, 求めた F_i を f[i] に返す。

- x[], f[] から $X_i(t_k+h)$ を求め, その値を x[i] に返す。改良オイラー法では $X_i(t_{k+1/2})$ 用にもう一つ 1 次元配列が必要である。以下の例では wrk[] がそのための配列である。

作成する関数プログラムの形式は, オイラー法では

```
void euler( void (*func)(), double x[], double f[], double t, double h, int n )
```

改良オイラー法では

```
void meuler( void (*func)(), double x[], double f[], double wrk[],
             double t, double h, int n )
```

となる。ただし, n = 連立微分方程式の個数 n , h = 刻み幅 h , t = t の初期値 t_k である。

改良オイラー法のサンプル

```
void meuler( void (*func)(), double x[], double f[], double wrk[],
             double t, double h, int n )
{
    int i;
    func( x, f, t );
    for(i=1; i<=n; i++ )
        wrk[i]=x[i]+h*f[i]/2;

    func( wrk, f, t+h/2 );
    for(i=1; i<=n; i++ )
        x[i]+=h*f[i];
}
```

最初の func(x, f, t) で $X_i(t_k)$ が与えられたとき (5.2) の右辺で必要になる F_i を f[i] に求める。次の for ループで (5.2) の左辺の $X_i(t_{k+1/2})$ を計算しこれを wrk[i] に保存する。(5.3) に対しても同様の処理をする。ただし, F_i を求めるとき $X_i(t_k)$ ではなく $X_i(t_{k+1/2})$, つまり wrk[i] を使う。

func は, (5.1) の場合は

```
void vib( double x[], double f[], double t )
{
    f[1]=x[2];
    f[2]=-w*w*x[1];
}
```

(5.4) の場合は

```
void damped( double x[], double f[], double t )
{
    f[1]=x[2];
    f[2]=-2*gamma*x[2]-x[1]+force*cos( w*t );
}
```

とすればよい。これらの関数を例えば euler(vib, ⋯) として使う。w, gamma, force は外部引数とし, 引数として渡さなくてもよいようにする。

問 5.1 h = 0.02 として

$$\frac{d^2x}{dt^2}=-x\,,\quad x(0)=1\,,\quad \left.\frac{dx}{dt}\right|_{t=0}=0\,,\qquad 0\le t\le 20$$

をオイラー法と改良オイラー法で解け。解析解 $\cos t$ との差を図示せよ。euler, meuler では

t での x[i] を与えて t+h での x[i] が求まることに注意すること。

問 5.2 [古典力学] 次の微分方程式 (減衰強制振動) を改良オイラー法で解く。

$$\frac{d^2x}{dt^2} + 2\gamma\frac{dx}{dt} + x = f\cos\omega t \tag{5.4}$$

1. $f = 0$ のとき, $\gamma = 0.1, 0.2, 0.5, 1, 2$ として解析解と数値解を比較せよ。
2. 例えば $f = 0.2$, $\gamma = 0.1$ のとき, $\omega = 0.5, 0.6, \cdots, 1.4, 1.5$ として時間が十分経過したときの振幅の ω 依存性を調べよ。

古典的ルンゲ・クッタ法

オイラー法と改良オイラー法は, ルンゲ・クッタ型公式と総称されるものの特別な場合である。実際の数値計算ではオイラー法や改良オイラー法はあまり使われない。より精度の良い結果を与える古典的ルンゲ・クッタ法がよく使われる。古典的ルンゲ・クッタ法は

$$X_i(t_{k+1}) = X_i(t_k) + \frac{1}{6}\Big(k_i + 2l_i + 2m_i + n_i\Big) + O(h^5)$$

ただし

$$\begin{aligned}
k_i &= hF_i\Big(X_1(t_k), \cdots, X_n(t_k), t_k\Big) \\
l_i &= hF_i\Big(X_1(t_k) + k_1/2, \cdots, X_n(t_k) + k_n/2, t_{k+1/2}\Big) \\
m_i &= hF_i\Big(X_1(t_k) + l_1/2, \cdots, X_n(t_k) + l_n/2, t_{k+1/2}\Big) \\
n_i &= hF_i\Big(X_1(t_k) + m_1, \cdots, X_n(t_k) + m_n, t_{k+1}\Big)
\end{aligned}$$

である。$X_i(t_{k+1})$ を $t = t_k$ でテイラー展開すると, 真の解 $X_i(t_k + h)$ のテイラー展開と h^4 のオーダーまでは一致する。上式のプログラム例を示す。

```
void runge( void (*func)(), double x[], double f[], double wrk1[],
            double wrk2[], double t, double h, int n )
{
    int i;
    func( x, wrk2, t );           /* k_i = h*wrk2[i] */
    for(i=1; i<=n; i++ ) {
        f[i] = x[i] + h*wrk2[i]/6;
        wrk1[i] = x[i] + h*wrk2[i]/2;
    }
    func( wrk1, wrk2, t+h/2 );    /* l_i = h*wrk2[i] */
    for(i=1; i<=n; i++ ) {
        f[i] += h*wrk2[i]/3;
        wrk1[i] = x[i] + h*wrk2[i]/2;
    }
    func( wrk1, wrk2, t+h/2 );    /* m_i = h*wrk2[i] */
    for(i=1; i<=n; i++ ) {
        f[i] += h*wrk1[i]/3;
        wrk1[i] = x[i] + h*wrk2[i];
    }
    func( wrk1, wrk2, t+h );      /* n_i = h*wrk2[i] */
    for(i=1; i<=n; i++ )
        x[i] = f[i] + h*wrk2[i]/6;
}
```

問 5.3［古典力学］中心力 $-r^\alpha$ の 2 次元ニュートン方程式

$$\frac{d^2x}{dt^2} = -r^\alpha \frac{x}{r}\,, \quad \frac{d^2y}{dt^2} = -r^\alpha \frac{y}{r}\,, \quad r = \sqrt{x^2+y^2}$$

を改良オイラー法とルンゲ・クッタ法で解け。この場合, 1 階の 4 元連立微分方程式になる。初期条件としては, 例えば $x(0)=2$, $y(0)=0$, $v_x(0)=0$, $v_y(0)=0.5$, $h=0.1$

1. 様々な α の値に対して軌道を図で表せ。$\alpha=-2$ または $\alpha=1$ のとき, 束縛運動の軌道は初期条件に依らずに閉じる。
2. 力学的エネルギーと角運動量の z 成分 L_z

$$E = \frac{1}{2}\left(v_x^2+v_y^2\right) + V(r)\,, \qquad L_z = xv_y - yv_x\,, \qquad V(r) = \begin{cases} \dfrac{r^{\alpha+1}}{\alpha+1}\,, & \alpha \neq -1 \\ \log r\,, & \alpha = -1 \end{cases}$$

   は時間的に一定であるが, 数値計算の結果ではどうなるか。
3. $\alpha > -3$ の場合, 軌道は同心円で限られた領域 $r_{\min} \le r \le r_{\max}$ になる。ところで, 中心力ポテンシャル $V(r)$ の場合, 力学的エネルギー E は極座標で表すと

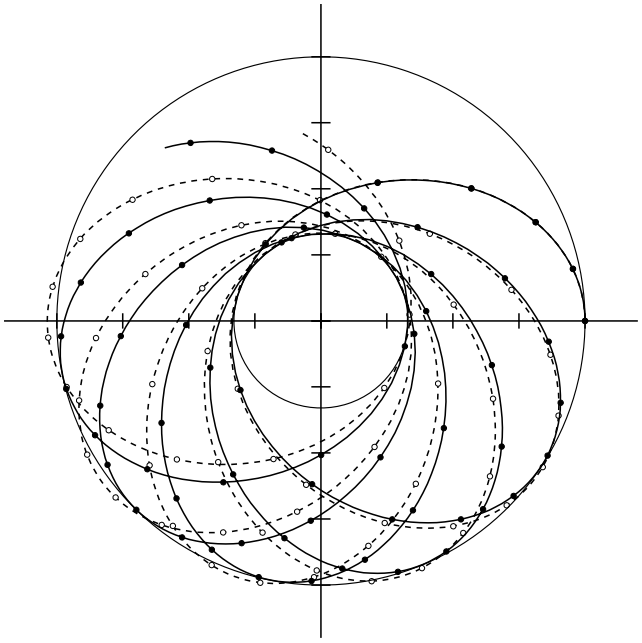
$$E = \frac{1}{2}\dot{r}^2 + \frac{L_z^2}{2r^2} + V(r)$$

   これから運動可能な r は

$$\frac{1}{2}\dot{r}^2 = E - \frac{L_z^2}{2r^2} - V(r) \ge 0$$

   を満たす領域である。上式 $=0$ を満たす r をニュートン法で求め, この半径の円を軌道の図に描いてみよ。E と L_z は初期条件を与えれば決まる。

実線はルンゲ・クッタ, 破線は改良オイラー



6　行列の固有値問題：ヤコビ法

行列成分 a_{ij} が実数で $a_{ij}=a_{ji}$ を満たす $n\times n$ 実対称行列 $\boldsymbol{A}_0=(a_{ij})$ の固有値 λ と固有ベクトル $\boldsymbol{X}$

$$\boldsymbol{A}_0\boldsymbol{X}=\lambda\boldsymbol{X}\,,\quad \boldsymbol{X}=\begin{pmatrix} x_1\\ x_2\\ \vdots\\ x_n\end{pmatrix} \tag{6.1}$$

をヤコビ法で求める。$\boldsymbol{P}_1=(p_{ij})$ として 2 次元の回転に対応する

$$\boldsymbol{P}_1=\begin{array}{cl}\begin{pmatrix} 1 & 0 & 0 & \cdots & & & & & \\ 0 & 1 & 0 & \cdots & & & \text{\Large 0} & & \\ & & \ddots & & & & & & \\ & & & 1 & & & & & 0\\ & & & & \cos\phi & \cdots & \sin\phi & \cdots & 0\\ & & & & \vdots & & \vdots & & \vdots\\ & & & & -\sin\phi & \cdots & \cos\phi & \cdots & 0\\ & \text{\Large 0} & & & \vdots & & \vdots & \ddots & \\ & & & & 0 & & 0 & & 1\end{pmatrix} & \begin{array}{l} \\ \\ \\ \\ p\text{ 行}\\ \\ q\text{ 行}\\ \\ \\ \end{array}\\ \begin{array}{ccccccccc} & & & & p\text{ 列} & & q\text{ 列} & & \end{array} & \end{array}$$

つまり

$$p_{ij}=\delta_{ij}\,p_i+(\delta_{ip}\,\delta_{jq}-\delta_{iq}\,\delta_{jp})\sin\phi\,,\quad p_i=\begin{cases}1 & i\neq p,\,q\\ \cos\phi & i=p,\,q\end{cases}$$

を考える。$\boldsymbol{P}_1$ は $\boldsymbol{P}_1^{-1}=\boldsymbol{P}_1^{\mathrm{T}}$ を満たすから直交行列である ($\boldsymbol{P}_1^{\mathrm{T}}$ は $\boldsymbol{P}_1$ の転置行列を表わす)。ここで $(a'_{ij})=\boldsymbol{A}_1=\boldsymbol{P}_1^{-1}\boldsymbol{A}_0\boldsymbol{P}_1$ とすると

$$a'_{ij}=a_{ij}\,,\qquad i\neq p,\,q\quad j\neq p,\,qr \tag{6.2}$$

$$a'_{pk}=a'_{kp}=a_{pk}\cos\phi-a_{qk}\sin\phi\,,\qquad k\neq p,\,q \tag{6.3}$$

$$a'_{qk}=a'_{kq}=a_{pk}\sin\phi+a_{qk}\cos\phi\,,\qquad k\neq p,\,q \tag{6.4}$$

$$a'_{pp}=a_{pp}\cos^2\phi+a_{qq}\sin^2\phi-a_{pq}\sin 2\phi \tag{6.5}$$

$$a'_{qq}=a_{pp}\sin^2\phi+a_{qq}\cos^2\phi+a_{pq}\sin 2\phi \tag{6.6}$$

$$a'_{pq}=a'_{qp}=\frac{a_{pp}-a_{qq}}{2}\sin 2\phi+a_{pq}\cos 2\phi \tag{6.7}$$

になる。非対角成分の中で絶対値が最大のものを a_{pq} とし, 変換後の行列 $\boldsymbol{A}_1$ の pq 成分 a'_{pq} が 0 になるように ϕ を選ぶ。(6.7) から ϕ は

$$\tan 2\phi=-\frac{2a_{pq}}{a_{pp}-a_{qq}}$$

である。このとき非対角成分の二乗和に対して不等式

$$\sum_{i\neq j}{a'_{ij}}^2\leq r\sum_{i\neq j}a_{ij}^2\,,\qquad r=1-\frac{2}{n(n-1)} \tag{6.8}$$

が成り立つ。証明はこの章の最後にある。以上の変換を行列 $\boldsymbol{A}_1$ に対して再度行い $\boldsymbol{A}_2$ を求める。この操作を繰り返し行う。変換に番号を付け

$$\boldsymbol{A}_m=\boldsymbol{P}_m^{-1}\boldsymbol{A}_{m-1}\boldsymbol{P}_m\,,\qquad m=1,\,2,\,\cdots$$

とすると

$$\boldsymbol{A}_m = \boldsymbol{P}_m^{-1}\boldsymbol{P}_{m-1}^{-1}\boldsymbol{A}_{m-2}\boldsymbol{P}_{m-1}\boldsymbol{P}_m = \cdots = \boldsymbol{T}_m^{-1}\boldsymbol{A}_0\boldsymbol{T}_m\,, \qquad \boldsymbol{T}_m = \boldsymbol{P}_1\cdots\boldsymbol{P}_{m-1}\boldsymbol{P}_m$$

になる。$\boldsymbol{A}_m$ の非対角成分の二乗和を S_m とすると (6.8) より $S_m \le rS_{m-1} \le \cdots \le r^m S_0$ であり $0 \le r < 1$ であるから, $m \to \infty$ のとき $S_m \to 0$ になる。したがって, 非対角成分はすべて 0 になるから

$$\boldsymbol{\Lambda} = \boldsymbol{T}^{-1}\boldsymbol{A}_0\boldsymbol{T}\,, \qquad \boldsymbol{T} = \lim_{m\to\infty}\boldsymbol{T}_m$$

は対角行列である。$\boldsymbol{\Lambda}$ の対角成分を $\lambda_1, \lambda_2, \cdots, \lambda_n$ とし, $\boldsymbol{T}$ の k 列成分からなる列ベクトルを $\boldsymbol{X}_k$, つまり, $(\boldsymbol{X}_k)_i = t_{ik}$ とすると, $\boldsymbol{A}_0\boldsymbol{T} = \boldsymbol{T}\boldsymbol{\Lambda}$ は

$$\boldsymbol{A}_0\boldsymbol{X}_k = \lambda_k\boldsymbol{X}_k$$

と書ける。したがって, $\boldsymbol{\Lambda}$ の対角成分 λ_k が $\boldsymbol{A}_0$ の固有値であり, $\boldsymbol{X}_k$ が固有値 λ_k に属する固有ベクトルになる。以上のようにして固有値と固有ベクトルを求める方法を**ヤコビ法**という。なお, $\boldsymbol{P}_m = (p_{ij})$, $\boldsymbol{T}_{m-1} = (t_{ij})$, $\boldsymbol{T}_m = (t'_{ij})$ とすると $\boldsymbol{T}_m = \boldsymbol{T}_{m-1}\boldsymbol{P}_m$ より

$$t'_{ij} = t_{ij}\ (j \ne p,\, q)\,, \quad t'_{ip} = t_{ip}\cos\phi - t_{iq}\sin\phi\,, \quad t'_{iq} = t_{ip}\sin\phi + t_{iq}\cos\phi$$

である。

実対称行列 $\boldsymbol{A}_0 = (a_{ij})$ が与えられたとき, ヤコビ法で固有値 λ_k と $\boldsymbol{T}$ を求める関数を作成する。$\boldsymbol{A}_m$ の非対角成分の最大絶対値がある値 (例えば 10^{-8}) 以下になったら対角化されたとする。$1 \le i, j \le n$ の場合, 配列の大きさは $(n+1)\times(n+1)$ 以上確保する。プログラムの流れは

```
#define NDIM 100 /** 必要な配列の大きさ **/
double a[NDIM][NDIM], t[NDIM][NDIM];
void yacobi( int n, double aa[NDIM][NDIM], double tt[NDIM][NDIM], double eps );
int main(){
   n=10;
   a に行列 A0 を与える
   yacobi( n, a, t, 1.e-8 );
   .....
}
void yacobi( int n, double aa[NDIM][NDIM], double tt[NDIM][NDIM], double eps )
{
   tt を単位行列として初期化
   count = 0;  maxvalue = 1;
   while( ++count < 10000 && maxvalue > eps ){
      aa の非対角成分で絶対値が最大の aa[p][q] を検索。maxvalue = |aa[p][q]|
      角度 φ を求め A' = P^{-1}AP, T' = TP を計算する。
      A' を A に, T' を T に代入 ( いきなり A = P^{-1}AP, T = TP としてはダメ )
   }
}
```

(6.2)∼(6.7) は $i \ne p, q$ などの条件を `if` 文で正直にプログラムすると結構複雑になるが, $i \ne p, q$ などの条件は無視して (6.2)∼(6.7) の順番でプログラムすればよい。例えば, (6.2) で $i = p$ の場合は (6.3) で上書きされる。

問 6.1 次の行列の固有値と固有ベクトルを求めよ。また, $(\boldsymbol{A}_0\boldsymbol{T}_m)_{ik} = \lambda_k(\boldsymbol{T}_m)_{ik}$ が数値的に成り立つことを確かめよ。

$$\begin{pmatrix} 6.0 & 5.0 & 4.0 & 3.0 & 2.0 & 1.0 \\ 5.0 & 5.0 & 4.0 & 3.0 & 2.0 & 1.0 \\ 4.0 & 4.0 & 4.0 & 3.0 & 2.0 & 1.0 \\ 3.0 & 3.0 & 3.0 & 3.0 & 2.0 & 1.0 \\ 2.0 & 2.0 & 2.0 & 2.0 & 2.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \end{pmatrix}$$

固有値は 17.20686, 1.98816, 0.77472, 0.44622, 0.31886, 0.26519 になる。

問 6.2 [量子力学] 角運動量 $\boldsymbol{J}^2$, J_z の同時固有状態 $|j\,m\rangle$ を基底にとる (ただし $\hbar = 1$)。j を与えたときハミルトニアン

$$H = J_z^2 - \omega J_x$$

の行列を対角化することにより, H の固有値と固有状態における J_x の期待値を求める。$j = 13/2$ のとき, 固有値と J_x 期待値を ω の関数として図示せよ。$j = 13/2$ の場合 $2j+1 = 14$ 次元の行列になる。ω が小さいときは縮退がある場合の 1 次の摂動, 一方, ω が非常に大きいときは $H \approx -\omega J_x$ としてよいから, 固有値と J_x の期待値がどうなるか予想できる。これと数値計算の結果を比較せよ。

$|\,n\,\rangle = |\,j\,m{=}n{-}j{-}1\rangle$, ($n = 1, 2, \cdots, 2j+1$) とする。$H|E\rangle = E|E\rangle$ である $|E\rangle$ は

$$|E\rangle = \sum_{n=1}^{2j+1} X_n\,|\,n\,\rangle$$

と展開できる。したがって

$$\sum_{n'} H|\,n'\,\rangle\,X_{n'} = E\sum_{n'} X_{n'}|\,n'\,\rangle$$

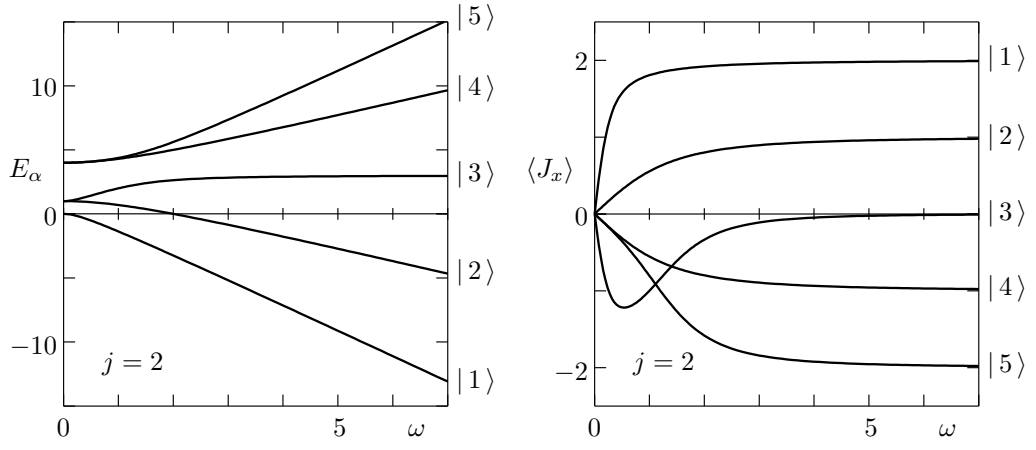
である。$\langle\,n\,|$ をかけ直交性 $\langle\,n\,|\,n'\,\rangle = \delta_{nn'}$ を使うと

$$\sum_{n'} h_{nn'}X_{n'} = E\,X_n\,, \qquad h_{nn'} = \langle\,n\,|H|\,n'\,\rangle$$

$(\boldsymbol{A}_0)_{nn'} = h_{nn'}$ とすれば (6.1) である。k 番目の固有値を E_k, 対応する X_n を X_{kn} とすると $X_{kn} = (\boldsymbol{T})_{nk}$ であるから, 期待値は

$$\langle E_k|J_x|E_k\rangle = \sum_{nn'} X_{kn}X_{kn'}\langle\,n\,|J_x|\,n'\,\rangle = \sum_{nn'}(\boldsymbol{T})_{nk}\langle\,n\,|J_x|\,n'\,\rangle(\boldsymbol{T})_{n'k} = \left(\boldsymbol{T}^{\mathrm{T}}\boldsymbol{J}_x\boldsymbol{T}\right)_{kk}$$

ただし $\boldsymbol{J}_x$ は行列 $(\boldsymbol{J}_x)_{nn'} = \langle\,n\,|J_x|\,n'\,\rangle$ である。



(6.8) の証明

$\boldsymbol{P}_1^{-1} = \boldsymbol{P}_1^{\mathrm{T}}$, $\boldsymbol{A}_0^{\mathrm{T}} = \boldsymbol{A}_0$ であるから

$$\boldsymbol{A}_1^{\mathrm{T}} = \left(\boldsymbol{P_1^{-1}A_0P_1}\right)^{\mathrm{T}} = \boldsymbol{P}_1^{\mathrm{T}}\boldsymbol{A}_0^{\mathrm{T}}\boldsymbol{P}_1 = \boldsymbol{P}_1^{-1}\boldsymbol{A}_0\boldsymbol{P}_1 = \boldsymbol{A}_1$$

$\boldsymbol{A}_1$ も実対称行列である。$a'_{ij} = a'_{ji}$ より

$$\sum_{i,\,j} {a'_{ij}}^2 = \sum_{i,\,j} a'_{ij}a'_{ji} = \mathrm{Tr}(\boldsymbol{A}_1\boldsymbol{A}_1) = \mathrm{Tr}(\boldsymbol{P}_1^{-1}\boldsymbol{A}_0\boldsymbol{P}_1\boldsymbol{P}_1^{-1}\boldsymbol{A}_0\boldsymbol{P}_1) = \mathrm{Tr}(\boldsymbol{A}_0\boldsymbol{A}_0) = \sum_{i,\,j} a_{ij}^2$$

となる。非対角成分の二乗和は

$$\sum_{i\neq j} {a'_{ij}}^2 = \sum_{i,\,j} {a'_{ij}}^2 - \sum_{i} {a'_{ii}}^2 = \sum_{i,\,j} a_{ij}^2 - \sum_{i} {a'_{ii}}^2 \tag{6.9}$$

(6.5), (6.6), (6.7) より ${a'_{pp}}^2 + 2{a'_{pq}}^2 + {a'_{qq}}^2 = a_{pp}^2 + 2a_{pq}^2 + a_{qq}^2$ になるが, $a'_{pq} = 0$ であるから

$${a'_{pp}}^2 + {a'_{qq}}^2 = a_{pp}^2 + 2a_{pq}^2 + a_{qq}^2$$

$i \neq p,\, q$ のとき $a'_{ii} = a_{ii}$ であるから

$$\sum_{i} {a'_{ii}}^2 = \sum_{i\neq p,\,q} a_{ii}^2 + {a'_{pp}}^2 + {a'_{qq}}^2 = \sum_{i\neq p,\,q} a_{ii}^2 + a_{pp}^2 + 2a_{pq}^2 + a_{qq}^2 = \sum_{i} a_{ii}^2 + 2a_{pq}^2$$

これを (6.9) に代入すると

$$\sum_{i\neq j} {a'_{ij}}^2 = \sum_{i,\,j} a_{ij}^2 - \sum_{i} a_{ii}^2 - 2a_{pq}^2 = \sum_{i\neq j} a_{ij}^2 - 2a_{pq}^2$$

になる。a_{pq} は非対角成分の中で絶対値が最大であるから

$$\sum_{i\neq j} a_{ij}^2 \leq \sum_{i\neq j} a_{pq}^2 = n(n-1)a_{pq}^2\,, \quad i.e. \quad a_{pq}^2 \geq \frac{1}{n(n-1)} \sum_{i\neq j} a_{ij}^2$$

したがって

$$\sum_{i\neq j} {a'_{ij}}^2 = \sum_{i\neq j} a_{ij}^2 - 2a_{pq}^2 \leq \left(1 - \frac{2}{n(n-1)}\right) \sum_{i\neq j} a_{ij}^2$$

7　シュレディンガー方程式の数値解法

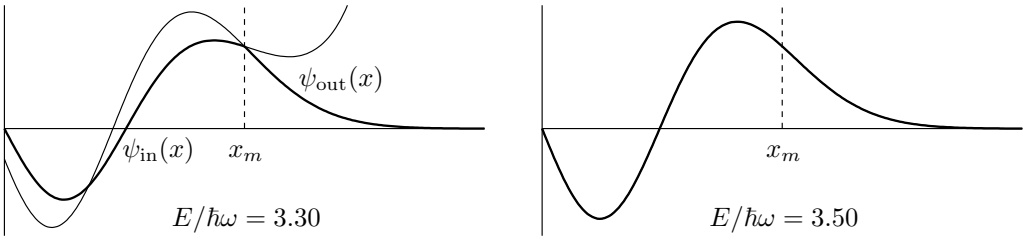
7.1　微分方程式を解く

境界条件がある 1 点だけで与えられる場合 (例えば, 初期条件を与えてニュートン方程式を解く), 境界条件を出発点にとりルンゲ・クッタ公式を繰り返し適用すれば, 微分方程式の数値解を逐次得ることができる。一方, 例えば, 1 次元シュレディンガー方程式

$$\left(-\frac{\hbar^2}{2m}\frac{d^2}{dx^2} + V(x) \right) \psi(x) = E\psi(x)$$

の束縛状態を求める場合, 区間 $a \le x \le b$ の両端で境界条件が与えられる。ただし, 境界が無限遠の場合には波動関数がほぼ 0 となる適当な有限区間で置き換える。適当な E を与えた場合, 片側の境界条件から出発して求めた解 $\psi(x)$ は, 一般にはもう 1 つの境界条件を満たさない。E がある値のときだけ両方の境界条件を満たす。このときの値が離散的固有値を与える。シュレディンガー方程式の束縛状態を数値的に求める問題は, 微分方程式を解くと同時に, 固有値も求めなければならない。

1 次元調和振動子ハミルトニアンの場合, $x = 0$ で $\psi(x) = 0$ として $x \to \infty$ に向けてルンゲ・クッタ公式で解いた波動関数 $\psi_{\rm in}(x)$ と x が十分大きい点で $\psi(x) = 0$ として $x \to 0$ に向けて解いた解いた波動関数 $\psi_{\rm out}(x)$ を下図に示す。ただし, 適当な点 x_m で $\psi_{\rm in}(x_m) = \psi_{\rm out}(x_m)$ になるように定数倍した。$E \ne \hbar\omega(n + 1/2)$, (以下の例では $n = 3$) の場合, $\psi_{\rm in}$ は $x \to \infty$ での境界条件を, $\psi_{\rm out}$ は $x = 0$ での境界条件を満たさない。また, $x = x_m$ で両者は連続であるが, 導関数 $\psi'(x)$ は不連続である。一方, $E = \hbar\omega(n + 1/2)$ の場合, $\psi_{\rm in}$ と $\psi_{\rm out}$ は一致し 2 つの境界条件を満たす。



離散的固有値を求める 1 つの方法を示す。[*)] $x = a$ での境界条件から x を増加させて得られる解を $\psi_a(x, E)$, $x = b$ での境界条件から x を減少させて得られる解を $\psi_b(x, E)$ とし, これらを $x = x_m$ でつなぐことを考える。一般に $\psi_a(x_m, E) \ne \psi_b(x_m, E)$ であり $x = x_m$ で連続ではないが, 定数倍した関数も微分方程式を満たすから

$$\phi_a(x, E) = \frac{\psi_a(x, E)}{\psi_a(x_m, E)}, \quad \phi_b(x, E) = \frac{\psi_b(x, E)}{\psi_b(x_m, E)}$$

とすれば $\phi_a(x_m, E) = \phi_b(x_m, E) = 1$ になり $x = x_m$ で連続である。しかし, パラメータ E が固有値に等しい場合を除けば, 導関数は $x = x_m$ で連続にならない。したがって

$$\phi'_a(x_m, E) - \phi'_b(x_m, E) = \frac{W(E)}{\psi_a(x_m, E)\psi_b(x_m, E)}$$

ただし

$$W(E) = \psi'_a(x_m, E)\,\psi_b(x_m, E) - \psi_a(x_m, E)\,\psi'_b(x_m, E)$$

とし, $W(E) = 0$ の解を求めれば固有値が得られる。$W(E) = 0$ の解はセカント法で求める。n 番目の近似解を E_n とすると, さらに精度の良い $n + 1$ 番目の近似解は

$$E_{n+1} = E_n - \frac{W(E_n)}{\dfrac{W(E_n) - W(E_{n-1})}{E_n - E_{n-1}}}$$

[*)] 櫻井捷海, コンピュータで学ぶ量子力学 [基礎編](裳華房, 1992)

である。最初の E_1 と E_2 を与えて, $|(E_{n+1}-E_n)/E_n| < \varepsilon$ (ε は非常に小さい定数) になるまで繰り返し実行する。最終的に得られた E_n が固有値, その時のルンゲ・クッタ公式の数値解が固有関数になる。$W(E)$ はロンスキー行列式 (あるいはロンスキアン) である。$W(E)$ は x_m に依存しない。

3 次元中心力ポテンシャル $V(r)$ の場合に, 以上の方法をもう少し具体的に考えよう。

$$\psi(\boldsymbol{r}) = \frac{u(r)}{r}\, Y_{\ell\, m}(\theta,\, \phi)$$

とおくと $u(r)$ は

$$\frac{d^2u}{dr^2} = \left(\frac{\ell(\ell+1)}{r^2} + \frac{2M}{\hbar^2}\Big(V(r)-E\Big)\right) u(r) \tag{7.1}$$

を満たす。$X_1 = u$, $X_2 = u'$ とすると

$$\frac{dX_1}{dr} = X_2\,, \qquad \frac{dX_2}{dr} = \left(\frac{\ell(\ell+1)}{r^2} + \frac{2M}{\hbar^2}\Big(V(r)-E\Big)\right) X_1$$

である。したがって, ルンゲ・クッタ型公式の関数は

$$F_1(X_1, X_2, r) = X_2\,, \qquad F_2(X_1, X_2, r) = \left(\frac{\ell(\ell+1)}{r^2} + \frac{2M}{\hbar^2}\Big(V(r)-E\Big)\right) X_1$$

とすればよい。

- $r = 0$ から r を増加させて解く場合
  $r = 0$ とすると $\ell(\ell+1)/r^2$ は発散するから, $r_{\min}$ を適当な微小値として $r = r_{\min}$ から解き始める。$r \to 0$ のとき $r^2V(r) \to 0$ ならば

  $$u(r) \to C\, r^{\ell+1}\,, \qquad C = \text{定数}$$

  である。そこで

  $$u(r_{\min}) = r_{\min}^{\ell+1}\,, \qquad u'(r_{\min}) = (\ell+1) r_{\min}^{\ell}$$

  を初期条件としてルンゲ・クッタを用いる。
- $r = \infty$ から r を減少させて解く場合
  $r = r_{\max}$ から解き始める。$r_{\max}$ としては, 多くの場合 $r_{\max} = (3 \sim 4) \times r_{\mathrm{T}}$ とすれば問題ない。ただし, $E - V(r_{\mathrm{T}}) = 0$ であり, 古典力学的には r_{T} より外側では運動は禁止される。u, u' の初期値は $r \to \infty$ での漸近形が解析的に求まるならば漸近形の値を使う。求まらない場合は

  $$u(r_{\max}) = 0\,, \quad u'(r_{\max}) = \varepsilon\,, \quad \varepsilon \text{は微小量}$$

  として解く。

2 つの数値解の接続点 r_m は何処にとってもよいわけだが, 実際に数値計算すると r_m の取り方により, 求まるはずの固有値が得られなかったりする。r_m の取り方には注意すべきである。

具体的なプログラム例　ℓ と E は `main` の前で宣言しておく。これらの変数を参照するとき引数として渡さなくて済む。また, 5 章で作成した関数 `runge` を変更しないために, $V(r)$ に含まれる定数も `main` の前で宣言する。区間 $[\,r_{\min}\,,\, r_{\max}\,]$ を $i_{\max}$ 等分する。$h = (r_{\max} - r_{\min})/i_{\max}$ とし, 接続点 r_m を $r_m = r_{\min} + i_m h$ にとる。これらの変数も `main` の前で宣言し, 具体的値は `main` の中で与える。以下では

$$\texttt{ll} = \ell\,, \quad \texttt{eigen} = E\,, \quad \texttt{hbarm} = 2M/\hbar^2$$

$$\texttt{hh} = h\,, \quad \texttt{rmin} = r_{\min}\,, \quad \texttt{rmax} = r_{\max}\,, \quad \texttt{imax} = i_{\max}\,, \quad \texttt{im} = i_m\,,$$

とする。

`wronskian` ℓ, E を与えたときロンスキアンを返す関数。ただし `pot(r)` $= V(r)$。

`get_eigen` 初期値 E_1, E_2 を与えてセカント法で固有値を求める関数。`w1`, `w2` という変数を用いずに

```
de = - wronskian(e2)*(e2-e1)/( wronskian(e2)-wronskian(e1) );
```

でもよいが, ロンスキアンの計算回数が 3 倍, したがって, 計算時間も 3 倍程度かかる。

```
void schr( double *x, double *f, double r )
{
     f[1]=x[2];
     f[2]=( ll*(ll+1)/(r*r) + hbarm*( pot(r) - eigen ) )*x[1];
}

double wronskian( double energy )
{
    int i;
    double x[3], f[3], wrk1[3], wrk2[3], r, wf, dwf;
    eigen = energy;
/*** rmin --> rm  ***/
    x[1] = pow(rmin, ll+1);     /**  u(rmin)の値   **/
    x[2] = (ll+1)*pow(rmin,ll); /**  u'(rmin)の値  **/
    for(i=0; i<im; i++ ){       /**  i=im は含まない   **/
        r = rmin+hh*i;
        runge( schr, x, f, wrk1, wrk2, r, hh, 2 );
    }
    wf = x[1]; dwf = x[2];
/*** rmax --> rm ***/
    x[1] = 0;      /**  u(rmax)の値  **/
    x[2] = 1.e-6;  /**  u'(rmax)の値 **/
    for(i = imax; i>im; i-- ){
        r = rmin+hh*i;
        runge( schr, x, f, wrk1, wrk2, r, - hh, 2 );
    }
    return dwf*x[1] - x[2]*wf;
}

double get_eigen( double e1, double e2 )
{
     int n;
     double de, eps, w1, w2;
     de  = 1.e6;     /**  適当な大きな値  **/
     eps = 1.e-8;    /**  収束条件  **/
     w1  = wronskian( e1 );
     n = 0;
     while( fabs(de/e1) > eps ){
         if( n++>20 ){
              printf("収束せず¥n");
              return 1.e20;  /**  無意味な値  **/
         }
         w2 = wronskian( e2 );
         de = - w2*(e2-e1)/(w2-w1);
         e1 = e2; w1 = w2;
         e2+= de;
     }
     return e2;
}
```

問 7.1　合流型超幾何関数 $M(\beta,\gamma,x)$ は

$$M(\beta,\gamma,x) = 1 + \frac{\beta}{\gamma}\frac{x}{1!} + \frac{\beta(\beta+1)}{\gamma(\gamma+1)}\frac{x^2}{2!} + \frac{\beta(\beta+1)(\beta+2)}{\gamma(\gamma+1)(\gamma+2)}\frac{x^3}{3!} + \cdots$$

で定義される。$n = 0,\, 1,\, 2,\, \cdots$ のとき $M(-n,\gamma,x)$ は n 次の多項式になる。$M_n = M(-n,\gamma,x)$ とおくと M_n は漸化式

$$(n+\gamma)M_{n+1} = (2n+\gamma-x)M_n - nM_{n-1}\,, \qquad M_0 = 1\,, \quad M_1 = 1 - x/\gamma$$

を満たす。$M(-n,\gamma,x)$ を求める関数 `hyper( int n, double gamma, double x )` を作成せよ。エルミート多項式が

$$H_{2n}(x) = (-1)^n \frac{(2n)!}{n!} M(-n, 1/2, x^2\,)$$

$$H_{2n+1}(x) = (-1)^n \frac{(2n+1)!}{n!}\, 2x M(-n, 3/2, x^2\,)$$

になることを数値的に確かめよ。

問 7.2　ポテンシャルが

$$V(r) = \frac{1}{2} M\omega^2\, r^2$$

の場合, 3 次元シュレディンガー方程式を数値的に解き, 固有値が

$$E_{n\ell} = \hbar\omega \left(\ell + 2n + \frac{3}{2} \right), \qquad \ell = 0,\, 1,\, 2,\, \cdots, \quad n = 0,\, 1,\, 2,\, \cdots \tag{7.2}$$

になることを確かめる。パラメータの値は

$$\hbar\omega = 41/A^{1/3}\,\mathrm{MeV}\,, \quad Mc^2 = 939\,\mathrm{MeV}\,, \quad \hbar c = 197.33\,\mathrm{MeV\,fm} \tag{7.3}$$

とする。

$$\frac{2M}{\hbar^2} = \frac{2Mc^2}{(\hbar c)^2}\,, \qquad V(r) = \frac{Mc^2(\hbar\omega)^2}{2(\hbar c)^2}\, r^2$$

であるから, Mc^2 , E , $V(r)$ を MeV 単位, r を fm $= 10^{-15}$ m 単位で表せばよい。A は核子数であり適当な値 (例えば $A = 100$) を用いる。

1. $0 \le E \le 10\hbar\omega$ を 50 等分して $E_i = \hbar\omega \times 0.2\, i$, ($0 \le i \le 50$) とする。$W(E_i)\, W(E_{i-1}) < 0$ になる E_{i-1}, E_i を求め $E_{i-1} < E_{n\ell} < E_i$ を確かめよ。
2. 1. で求めた E_{i-1}, E_i を初期値としてセカント法により $W(E) = 0$ となる E を求め, $E/\hbar\omega$ と $\ell + 2n + 3/2$ を比較せよ。n は $u(r) = 0$ になる点の個数 ($r = 0$ は除く) である。
3. 数値的に求めた波動関数を規格化し

$$u_{n\ell}(r) = \sqrt{\alpha}\, N_{n\ell}\, q^{\ell+1} e^{-q^2/2} M(-n\,,\, \ell + 3/2\,,\, q^2\,) \tag{7.4}$$

ただし

$$q = \alpha r\,, \quad \alpha = \sqrt{\frac{M\omega}{\hbar}} = \frac{\sqrt{Mc^2\,\hbar\omega}}{\hbar c}\,, \quad N_{n\ell} = \sqrt{\frac{2\Gamma(\ell+3/2+n)}{n!\,\left(\Gamma(\ell+3/2)\right)^2}}$$

と一致することを確かめよ。ただし, $M(\beta,\gamma,x)$ は合流型超幾何関数, Γ はガンマ関数で n を負でない整数とするとき

$$\Gamma(n+1/2) = \frac{(2n)!}{2^{2n} n!}\sqrt{\pi}\,, \qquad \Gamma(n+1) = n!$$

である。

問 7.3　調和振動子ポテンシャルの代わりに Woods–Saxon ポテンシャル

$$V(r) = -\frac{V_0}{1+\exp((r-R)/a)} \tag{7.5}$$

を考える。ただし, パラメータの値は (7.3) 及び

$$V_0 = 51\,\mathrm{MeV}\,, \quad R = 1.25\,A^{1/3}\,\mathrm{fm}$$

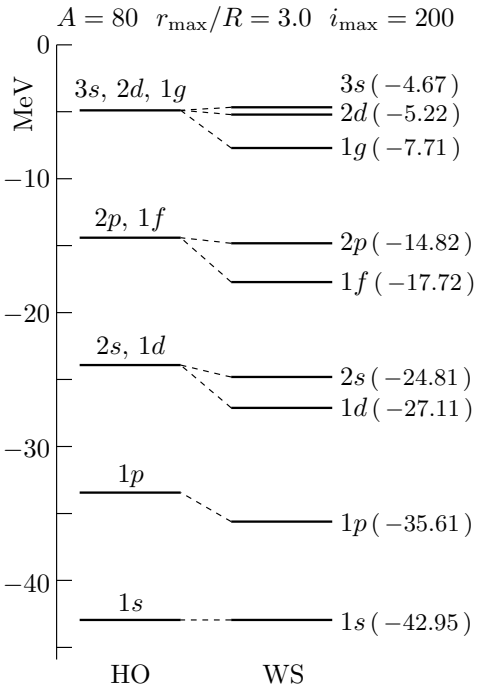
とする。束縛状態の固有値 E は $V(0) \approx -V_0 < E < 0$ であるから, この区間を適当に等分し, $W(E_i)\,W(E_{i-1}) < 0$ である E_{i-1}, E_i を初期値としてセカント法を用いる。

1. $\ell = 0$ のとき (7.1) はポテンシャルが $V(r)$ である 1 次元シュレディンガー方程式になる。ところで, $a \to +0$ とすると (7.5) は井戸型ポテンシャル

$$V(r) = \begin{cases} -V_0\,, & 0 \le r < R \\ 0\,, & r > R \end{cases}$$

になるから, $\ell = 0$ の固有値 E は (1.3) の $g(k) = 0$ で与えられる。$a = 0.01\,\mathrm{fm}$ 及び $a = 0.1\,\mathrm{fm}$ のとき (7.1) をルンゲ・クッタで解いて得られた固有値と $g(k) = 0$ から求めた固有値を比較せよ。

2. $a = 0.65\,\mathrm{fm}$ とする。$E_{n\ell}$ と (7.2) を比較して図示せよ。ただし, (7.2) に定数を加えて $n = 0$, $\ell = 0$ を一致させる。(7.2) では $\ell + 2n$ が同じ (n, ℓ) の状態は縮退している。また, 得られた固有関数を (7.4) と比較してみよ。



7.2　行列の対角化

行列の対角化によりシュレディンガー方程式を解く。あるハミルトニアン H_0

$$H_0 = -\frac{\hbar^2}{2m}\boldsymbol{\nabla}^2 + V_0(\boldsymbol{r})$$

の固有値 $E_0(i)$ と固有関数 $|\,i\,\rangle$ が分かっているとする:

$$H_0|\,i\,\rangle = E_0(i)|\,i\,\rangle$$

ここで i は一般に量子数の組である。ハミルトニアン H

$$H = -\frac{\hbar^2}{2m}\boldsymbol{\nabla}^2 + V(\boldsymbol{r}) = H_0 + \delta V\,, \quad \delta V = V - V_0$$

の固有値 E と固有関数 $|\psi\rangle$ を求める。

$$|\psi\rangle = \sum_i c_i\,|\,i\,\rangle$$

と展開できるから, これを $H|\psi\rangle = E|\psi\rangle$ に代入すると

$$\sum_j c_j\,H|\,j\,\rangle = E\sum_j c_j\,|\,j\,\rangle$$

ブラ $\langle i|$ を掛け直交性 $\langle i\,|\,j\rangle = \delta_{ij}$ を使うと

$$\sum_j H_{ij}\,c_j = E\,c_i\,,\quad H_{ij} = \langle\, i\,|H_0+\delta V|\,j\,\rangle = E_0(i)\,\delta_{ij} + \langle\, i\,|\delta V|\,j\,\rangle$$

となる。これはエルミート行列 H_{ij} の固有値と固有ベクトルを求めることに他ならない。

数値計算する上での注意

- 行列 H_{ij} が無限次元の場合, 適当な次元数で計算し, 次元数を変化させても固有値が求めたい精度内で不変であることを確かめる。
- 行列 H_{ij} を求めるとき対称性を考慮し, できるだけ行列の次元を小さくする。
  例えば, 球対称の場合 (V, V_0 が r だけの関数) を考える。この場合, H_0, H は角運動量演算子 $\boldsymbol{\ell}$ と交換するから, H_0, H の固有状態は $\boldsymbol{\ell}^2$, ℓ_3 の固有状態でもある。$|\,i\,\rangle$ として H_0, $\boldsymbol{\ell}^2$, ℓ_3 の同時固有状態 $|\,n\ell m\,\rangle$ を採用すれば, 一般に

  $$|\psi\rangle = \sum_{n\ell m} c_{n\ell m}\,|\,n\ell m\,\rangle\,,\quad H_0|\,n\ell m\,\rangle = E_0(n\ell)\,|\,n\ell m\,\rangle$$

  であるが, $|\psi\rangle$ が $\boldsymbol{\ell}^2$, ℓ_3 の固有状態であることを要請すると

  $$|\psi\rangle = \sum_{n} c_{n\ell m}\,|\,n\ell m\,\rangle$$

  となり, ℓ, m について和をとる必要はない。$|\,n\ell m\,\rangle$ の波動関数を

  $$\frac{u_{n\ell}(r)}{r}\,Y_{\ell\,m}(\theta,\,\phi)$$

  とすると, 必要な行列要素は

  $$\langle n\ell m|\delta V|n'\ell m\rangle = \int_0^\infty dr\,u_{n\ell}(r)\,u_{n'\ell}(r)\,\delta V(r)$$

  である。

問 7.4　$\hbar\omega = 41/A^{1/3}\,\mathrm{MeV}$ である調和振動子ポテンシャルの固有関数を基底にとり, **問 7.3** のハミルトニアンを対角化して固有値を求めよ。

$$\delta V(r) = -\,\frac{V_0}{1+\exp((r-R)/a)} - \frac{1}{2}M\omega^2 r^2$$

である。δV の行列要素を求めるとき, 積分は台形則でよい。(7.4) より

$$\begin{aligned}&\langle n\ell m|\delta V|n'\ell m\rangle\\ &= N_{n\ell}\,N_{n'\ell}\int_0^\infty dq\,e^{-q^2}q^{2\ell+2}\,\delta V(q/\alpha)\,M\left(-n\,,\,\ell+3/2\,,\,q^2\right)M\left(-n'\,,\,\ell+3/2\,,\,q^2\right)\end{aligned}$$

ただし n, $n' = 0,\,1,\,2,\,\cdots$ である。q の積分範囲は**問 3.2** を参照せよ。
$\delta V = 1$ とすれば $\langle n\ell m|\delta V|n'\ell m\rangle = \delta_{nn'}$ になるはずである。行列要素を計算する関数が正しいか, これでチェックすること。

8 モンテカルロ法

- 独立な確率変数 x_1, x_2, $\cdots$ が同じ確率分布 (平均 $\mu = \langle x \rangle$, 分散 $\sigma^2 = \langle x^2 \rangle - \langle x \rangle^2$) に従うとき，$\lim_{n\to\infty} \sum_{k=1}^{n} \dfrac{x_k - \mu}{\sqrt{n}}$ は平均 0, 分散 σ^2 の正規分布 (ガウス分布) に収束する (**中心極限定理**)。証明は 31 ページ。
- $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$, $\boldsymbol{y} = (y_1, y_2, \cdots, y_n)$ の確率密度をそれぞれ $\rho(\boldsymbol{x})$, $\rho(\boldsymbol{y})$ とする。$x_k = x_k(\boldsymbol{y})$ の場合

$$\rho(\boldsymbol{x})\, dx_1 \cdots dx_n = \rho(\boldsymbol{y})\, dy_1 \cdots dy_n\,, \qquad dy_1 \cdots dy_n = \left| \frac{\partial(y_1, \cdots, y_n)}{\partial(x_1, \cdots, x_n)} \right| dx_1 \cdots dx_n$$

である。$\partial(y_1, \cdots, y_n)/\partial(x_1, \cdots, x_n)$ はヤコビアンである。これから

$$\rho(\boldsymbol{x}) = \rho(\boldsymbol{y}) \left| \frac{\partial(y_1, \cdots, y_n)}{\partial(x_1, \cdots, x_n)} \right|, \qquad 1 \text{ 変数の場合} \quad \rho(x) = \rho(y) \left| \frac{dy}{dx} \right| \tag{8.1}$$

になる。特に，$\rho(\boldsymbol{y})$ が $[0,1]^n$ で一様分布ならば $\rho(\boldsymbol{y}) = 1$ である。
- 独立な x_1, x_2 の確率密度を $\rho_1(x_1)$, $\rho_2(x_2)$ とするとき，$x = x_1 + x_2$ の確率密度 $\rho(x)$ は

$$\rho(x) = \int_{-\infty}^{\infty} dx_1\, dx_2\, \rho_1(x_1)\rho_2(x_2)\delta(x_1 + x_2 - x) = \int_{-\infty}^{\infty} dx_1\, \rho_1(x_1)\rho_2(x - x_1) \tag{8.2}$$

である。x_1, x_2 が ρ_1, ρ_2 で分布する乱数のとき，$x_1 + x_2$ は $\rho(x)$ で分布する。
- 区間 $[0, 1)$ の一様乱数の生成
  以下の関数を使うには `#include <stdlib.h>` , `time()` の場合 `#include <time.h>`
  1. `#define drand() ((double)random()/(RAND_MAX+1.))`
     として `drand()` を使う (`RAND_MAX` $= 2^{31} - 1$)。$16 \times (2^{31} - 1) \approx 3.4 \times 10^{10}$ 個の異なる擬似乱数を生成する。初期化は `void srandom(unsigned int)` で行う。ソースの最初で 1 度だけ `srandom((unsigned int)time(NULL))` とすると，プログラム実行ごとに異なる乱数になる。
  2. `double drand48(void)`, 初期化は `void srand48(long int)` で行う。$2^{48} \approx 2.8 \times 10^{14}$ 個の異なる乱数を生成する。

モンテカルロ法の適用例として区間 $[0,1]$ の積分を考える。この区間を M 個に等分割すると

$$I = \int_0^1 dq\, f(q) \approx \frac{1}{M} \sum_{i=0}^{M-1} f(q_i)\,, \qquad q_i = \frac{i}{M}$$

である。x_1, x_2, $\cdots$, x_N を区間 $[0,1]$ での一様乱数とする。$q_i < x_k < q_{i+1}$ である x_k は N/M 個存在する。この x_k を x_{ij} で表す ($j = 1, 2, \cdots, N/M$)。M が十分大きければ $f(x_{ij}) \approx f(q_i)$ になるから

$$I \approx \frac{1}{M} \sum_{i=0}^{M-1} \frac{1}{N/M} \sum_{j=1}^{N/M} f(x_{ij}) = \frac{1}{N} \sum_{i=0}^{M-1} \sum_{j=1}^{N/M} f(x_{ij}) = \frac{1}{N} \sum_{k=1}^{N} f(x_k) \tag{8.3}$$

である。n 次元の領域 $V = [0,1]^n$ での積分は

$$I = \int_V d^n x\, f(\boldsymbol{x}) \approx \frac{1}{N} \sum_{k=1}^{N} f\big(x_1^{(k)}, \cdots, x_n^{(k)}\big)$$

になる。ここで $x_1^{(k)}$, $\cdots$, $x_n^{(k)}$ は独立に選んだ区間 $[0,1]$ の一様乱数であり Nn 個の乱数を生成する。

問 8.1　3 次元の半径 1 の球の体積は

$$I = 8\int_0^1 dxdydz\, f(x,y,z) \approx \frac{8}{N}\sum_{k=1}^{N} f(x_k,y_k,z_k)\,, \qquad f(x,y,z) = \theta\big(1-x^2-y^2-z^2\big)$$

である。I を求めよ。

特定の分布をする乱数

区間 $[\,a,b\,]$ で $\rho(x)$ で分布する N 個の乱数 $x^{(k)}$ を考える。$a \le x \le b$ を間隔 Δx で等分し

$$\rho_N(x) = \frac{[\,x-\Delta x/2, x+\Delta x/2\,]\ \text{の}\ x^{(k)}\text{の個数}}{N\Delta x}$$

とする。$x = a + n\Delta x$ のとき $(n-1/2)\Delta x \le x^{(k)} - a < (n+1/2)\Delta x$ より

$$\rho_N(a+n\Delta x) = \frac{N_n}{N\Delta x}\,, \qquad N_n = \frac{x^{(k)}-a}{\Delta x} + \frac{1}{2}\ \text{の整数部分が}\ n\ \text{になる}\ x^{(k)}\text{の個数} \tag{8.4}$$

端点の場合 $2N_n/(N\Delta x)$ にする。$N\to\infty$, $\Delta x \to 0$ で $\rho_N(x) \to \rho(x)$ になる。(8.3) と同様にして

$$\int_a^b dx\, f(x)\rho(x) \approx \sum_n f(a+n\Delta x)\rho_N(a+n\Delta x)\Delta x = \frac{1}{N}\sum_n f(a+n\Delta x)N_n \approx \frac{1}{N}\sum_{k=1}^{N} f\big(x^{(k)}\big)$$

である。

逆関数法

確率密度 $\rho(x)$, $a \le x \le b$ のとき

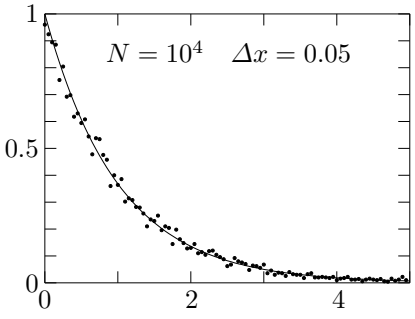
$$y(x) = \int_a^x dx'\rho(x')\,, \qquad \int_a^b dx'\rho(x') = 1 \tag{8.5}$$

とする。(8.1) より $[\,0,1\,]$ の一様乱数 y に対して ($\rho(y) = 1$), $y = y(x)$ で決まる x は $\rho(x)$ で分布する。逆関数法は $x = x(y)$ が簡単に求まる場合に適用できる。

問 8.2　$\rho(x) = \lambda e^{-\lambda x}$, $\lambda > 0$, $0 \le x < \infty$ の場合

$$y = \int_0^x dx'\rho(x') = 1 - e^{-\lambda x}\,, \quad \therefore \quad x = -\frac{\log(1-y)}{\lambda} \ \text{ または } \ x = -\frac{\log y}{\lambda}$$

である。(8.4) の $\rho_N(x)$ と $\rho(x)$ を図示して比較せよ。右図は $\lambda = 1$ の場合である。曲線は $\rho(x) = e^{-x}$ を示す。



$$\frac{1}{N}\sum_{k=1}^{N} x_k^n \approx \int_0^\infty dx\, x^n\rho(x) = \frac{n!}{\lambda^n}$$

を確かめよ。

正規分布 (ガウス分布)

平均 μ, 分散 σ^2 の正規分布

$$\rho(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \qquad -\infty < x < \infty \tag{8.6}$$

の場合，(8.5) の $y(x)$ はエラー関数になり $x = x(y)$ を求めることは複雑である。別の方法を考える。

y_k を $[-1,1]$ の一様乱数とし

$$x=\sqrt{\frac{3}{n}}\,\sigma\sum_{k=1}^{n}y_k+\mu=\frac{1}{\sqrt{n}}\sum_{k=1}^{n}Y_k+\mu\,,\qquad Y_k=\sqrt{3}\,\sigma y_k \tag{8.7}$$

とする。$|\,x-\mu\,|\le\sqrt{3n}\,\sigma$ である。$\langle\,y_k\,\rangle=0\,,\ \langle\,y_k^2\,\rangle=1/3$ であるから $\langle\,Y_k\,\rangle=0\,,\ \langle\,Y_k^2\,\rangle-\langle\,Y_k\,\rangle^2=\sigma^2$ になる。$n\gg1$ の場合，中心極限定理より $x-\mu$ の分布は近似的に平均 0, 分散 σ^2 の正規分布になるから，x の分布 $\rho^{(n)}(x)$ は平均 μ, 分散 σ^2 の正規分布 (8.6) になる。

上の方法は多数の一様乱数が必要で効率が悪い。これを回避する方法にボックス・ミューラー法がある。$r,\,\theta$ を $[0,1]$ での一様乱数として ($\rho(r,\theta)=1$)

$$x=\sqrt{-\,2\sigma^2\log r}\,\cos(2\pi\theta)+\mu\,,\qquad y=\sqrt{-\,2\sigma^2\log r}\,\sin(2\pi\theta)+\mu \tag{8.8}$$

とする。$0\le\sqrt{-\log r}<\infty$, $0\le2\pi\theta\le2\pi$ であるから，(x,y) は平面上の任意の点になる。

$$r=\exp\!\left(-\,\frac{(x-\mu)^2+(y-\mu)^2}{2\sigma^2}\right),\qquad \tan(2\pi\theta)=\frac{y-\mu}{x-\mu}$$

より

$$\rho(x,y)=\rho(r,\theta)\left|\frac{\partial(r,\theta)}{\partial(x,y)}\right|=\frac{e^{-(x-\mu)^2/2\sigma^2}}{\sqrt{2\pi\sigma^2}}\,\frac{e^{-(y-\mu)^2/2\sigma^2}}{\sqrt{2\pi\sigma^2}}$$

になるから，x, y は独立にガウス分布する。ボックス・ミューラー法は 2 個の一様乱数から 2 個の正規乱数を生成するが，log と三角関数の計算が必要になる。三角関数の計算を回避する方法もある。

問 8.3　$n=2$ と $n=12$ の場合，**問 8.2** と同様に $\rho_N^{(n)}(x)$ を図示し (8.6) と比較せよ。$\rho^{(2)}(x)$ が二等辺三角形の分布になることを (8.2) より示せ。(8.8) と $n=12$ の (8.7) と計算速度を比較せよ。

棄却法

逆関数法の適用が困難な $\rho(x)\,,\ a\le x\le b$ に対して，$f(x)\ge\rho(x)$ である簡単な関数 $f(x)$ を考える。

$$F(x)=\int_a^x dx\,f(x)\,,\qquad F_{\max}=\int_a^b dx\,f(x)\,,\qquad F_{\max}>\int_a^b dx\,\rho(x)=1$$

とする。逆関数法より $[\,0,F_{\max}\,]$ の一様乱数を u とすると，$u=F(x)$ である x の分布は $f(x)$ になる。v を $[\,0,1\,]$ の一様乱数とするとき，$\rho(x)/f(x)\ge v$ を満たす x だけ採用する。$(x,vf(x))$ は $a\le x\le b\,,\ 0\le y\le\rho(x)$ に一様に分布するから，ランダムな点を $[\,x,x+dx\,]$ に見出す確率は $\rho(x)\,dx$ になる。したがって，$\rho(x)$ で分布する乱数が求まる。

問 8.4　ガンマ分布 $\rho(x)=x^ne^{-x}/n!$ を考える ($x\ge0\,,\ n=0,\,1,\,\cdots$)。$a_n=\sqrt{2n+1}$ として

$$f(x)=\frac{\rho(n)}{1+(x-n)^2/a_n^2}\,,\qquad F_{\max}=\int_0^\infty dx\,f(x)=a_n\rho(n)\big(\pi/2+\theta\,\big)\,,\quad \theta=\tan^{-1}\frac{n}{a_n}$$

でローレンツ分布する x は，y を $[\,0,1\,]$ の一様乱数とすると

$$x=a_n\tan\Big(\big(\pi/2+\theta\big)y-\theta\Big)+n$$

になることを示せ。$f(x)\ge\rho(x)$ である。棄却法により $\rho(x)$ で分布する乱数を生成する。棄却前の乱数が $f(x)$, 棄却後の乱数が $\rho(x)$ で分布することを確かめよ。なお，y_k を $[\,0,1\,]$ の一様乱数とするとき $x=-\log\big(y_1y_2\cdots y_{n+1}\big)$ は $\rho(x)$ で分布する。

メロトポリス法

一般に n 次元の確率密度 $\rho(\boldsymbol{x})$ で分布する点列 $\boldsymbol{x}^{(1)}$, $\boldsymbol{x}^{(2)}$, $\cdots$ の発生方法を考える。$\boldsymbol{x}^{(1)}$ を適当に与えたとき，$\boldsymbol{x}^{(k)}$ から $\boldsymbol{x}^{(k+1)}$ を次のようにして決める。$\boldsymbol{x}^{(k)}$ を中心とした 1 辺 $2\,\delta x$ の箱 V を考え

$$\boldsymbol{x}'=\boldsymbol{x}^{(k)}+\delta x\big(2\boldsymbol{y}-1\big)\,,\qquad y_i=\text{独立な}\,[\,0,1\,]\,\text{の一様乱数}$$

とする。$\boldsymbol{x}'$ は V に一様に分布する。$r=\rho(\boldsymbol{x}')/\rho(\boldsymbol{x})$ とするとき

$$r\geq 1 \text{ のとき } \boldsymbol{x}^{(k+1)}=\boldsymbol{x}'\,,\qquad r<1 \text{ のとき確率 } r \text{ で } \boldsymbol{x}^{(k+1)}=\boldsymbol{x}'$$

にする。$[0,1]$ の一様乱数 v が $r\geq v$ になる確率は，$r\geq 1$ のとき 1, $r<1$ のとき r であるから

$$r=\rho(\boldsymbol{x}')/\rho(\boldsymbol{x})\geq v \text{ のとき } \boldsymbol{x}^{(k+1)}=\boldsymbol{x}'\,,\qquad \rho(\boldsymbol{x}')/\rho(\boldsymbol{x})<v \text{ のとき } \boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)} \tag{8.9}$$

である。

$\rho(\boldsymbol{y})>\rho(\boldsymbol{x})$ の場合を考える ($\rho(\boldsymbol{y})<\rho(\boldsymbol{x})$ の場合も同様)。$\boldsymbol{x}$ 近傍の乱数の個数を $N(\boldsymbol{x})$ とする。$\boldsymbol{x}\to\boldsymbol{y}$ になる確率は 1，$\boldsymbol{y}\to\boldsymbol{x}$ になる確率は $\rho(\boldsymbol{x})/\rho(\boldsymbol{y})$ であるから

$$\boldsymbol{x}\to\boldsymbol{y} \text{ に移動する乱数 } N(\boldsymbol{x})\,,\qquad \boldsymbol{y}\to\boldsymbol{x} \text{ に移動する乱数 } N(\boldsymbol{y})\frac{\rho(\boldsymbol{x})}{\rho(\boldsymbol{y})}$$

したがって $\boldsymbol{x}\to\boldsymbol{y}$ に移動する正味の個数は

$$N(\boldsymbol{x})-N(\boldsymbol{y})\frac{\rho(\boldsymbol{x})}{\rho(\boldsymbol{y})}=N(\boldsymbol{y})\left(\frac{N(\boldsymbol{x})}{N(\boldsymbol{y})}-\frac{\rho(\boldsymbol{x})}{\rho(\boldsymbol{y})}\right),\qquad \therefore\quad \frac{N(\boldsymbol{x})}{N(\boldsymbol{y})}\gtrless\frac{\rho(\boldsymbol{x})}{\rho(\boldsymbol{y})} \text{ のとき } \boldsymbol{x}\rightleftarrows\boldsymbol{y} \text{ に移動}$$

これから，最終的には $N(\boldsymbol{x})/N(\boldsymbol{y})=\rho(\boldsymbol{x})/\rho(\boldsymbol{y})$ になり $N(\boldsymbol{x})$ は $\rho(\boldsymbol{x})$ で分布する。

メトロポリス法では δx のとり方が問題になる。δx が小さいと棄却 ($\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}$) 回数は少ないが，$\boldsymbol{x}^{(k)}$ が $\boldsymbol{x}^{(1)}$ から離れる確率は低くなり，効率よく $\rho(\boldsymbol{x})$ を生成できない。一方，δx が大きすぎると棄却回数が多くなり，やはり非効率である。目安としては，棄却率が $1/2$ 程度になる δx である。

問 8.5　(8.9) を N 回繰り返して 1 次元の正規分布する乱数 x を生成する。**問 8.2** と同様の分布図を求め，δx 依存性を調べよ。

中心極限定理の証明

x_k の確率密度 $\rho(x)$ をフーリエ変換して

$$\rho(x)=\frac{1}{2\pi}\int_{-\infty}^{\infty}dq\,e^{iq(x-\mu)}\varphi(q)\,,\qquad \varphi(q)=\int_{-\infty}^{\infty}dx\,e^{iq(\mu-x)}\rho(x)$$

とする。全確率は 1 であるから $\varphi(0)=1$ になる。また

$$\varphi'(q)=\int_{-\infty}^{\infty}dx\,i\big(\mu-x\big)e^{iq(\mu-x)}\rho(x)\,,\qquad \varphi''(q)=-\int_{-\infty}^{\infty}dx\,\big(\mu-x\big)^2e^{iq(\mu-x)}\rho(x)$$

より $\varphi'(0)=i\mu-i\langle\,x\,\rangle=0$, $\varphi''(0)=\mu^2-\langle\,x^2\,\rangle=-\,\sigma^2$ であるから $\varphi(q)=1-\sigma^2q^2/2+\cdots$ になる。$y=\big(x-\mu\big)/\sqrt{n}$ とすると

$$\rho(y)=\frac{dx}{dy}\rho(x)=\sqrt{n}\,\rho(x)=\frac{\sqrt{n}}{2\pi}\int_{-\infty}^{\infty}dq\,e^{i\sqrt{n}\,qy}\varphi(q)\,,\quad \therefore\quad \int_{-\infty}^{\infty}dy\,e^{-ipy}\rho(y)=\varphi\big(\,p/\sqrt{n}\,\big)$$

である。(8.2) より $X=y_1+\cdots+y_n$ の確率密度は

$$\begin{aligned}\rho(X)&=\int_{-\infty}^{\infty}dy_1\cdots dy_n\,\rho(y_1)\cdots\rho(y_n)\,\delta\big(X-\sum_k y_k\big)\\&=\int_{-\infty}^{\infty}dy_1\cdots dy_n\,\rho(y_1)\cdots\rho(y_n)\,\frac{1}{2\pi}\int_{-\infty}^{\infty}dp\,\exp\Big(ipX-ip\sum_k y_k\Big)\\&=\frac{1}{2\pi}\int_{-\infty}^{\infty}dp\,e^{ipX}\left(\int_{-\infty}^{\infty}dy\,e^{-ipy}\rho(y)\right)^n=\frac{1}{2\pi}\int_{-\infty}^{\infty}dp\,e^{ipX}\Big(\varphi\big(\,p/\sqrt{n}\,\big)\Big)^n\end{aligned}$$

になる。

$$\Big(\varphi\big(p/\sqrt{n}\big)\Big)^n = \left(1-\frac{\sigma^2p^2}{2n}+\cdots\right)^n, \quad \therefore \quad \rho(X) \xrightarrow{n\to\infty} \frac{1}{2\pi}\int_{-\infty}^{\infty} dp\, e^{ipX}e^{-\sigma^2p^2/2} = \frac{e^{-X^2/2\sigma^2}}{\sqrt{2\pi\sigma^2}}$$

である。

無限に深い井戸型ポテンシャルの基底状態エネルギー

$H = -\dfrac{\hbar^2}{2M}\dfrac{d^2}{dx^2}$, $0<x<L$ のとき

$$\psi(x,t) = e^{-Ht/\hbar}\psi_0(x), \qquad \psi_0(0)=\psi_0(L)=0, \quad 0<x<L \text{ では } \psi_0(x)>0$$

とする。H の固有値 E_n の固有関数を $\varphi_n(x)$ とすると ($E_0<E_1<\cdots$)

$$\psi_0(x) = \sum_{n=0}^{\infty} C_n\varphi_n(x), \qquad C_n = \int_0^L dx\,\varphi_n^*(x)\psi_0(x)$$

と展開できるから

$$P(t) \equiv \frac{1}{L}\int_0^L dx\,\psi(x,t) = \sum_{n=0}^{\infty} D_n e^{-E_nt/\hbar} \xrightarrow{t\to\infty} D_0e^{-E_0t/\hbar}, \qquad D_n = \frac{C_n}{L}\int_0^L dx\,\varphi_n(x)$$

になる。$\psi_0(x)$ は基底状態 $\varphi_0(x)$ と直交しないから $D_0>0$ にできる。$\log P(t)$ を t の関数としてプロットしたとき，直線の傾きから基底状態のエネルギー E_0 が求まる。

$\psi(x,t)$ は $\hbar\,\partial_t\psi = -H\psi$ の解である。これを無次元の変数 $q=x/L$, $\tau=\hbar t/(ML^2)$ で表すと

$$\frac{\partial\psi(q,\tau)}{\partial\tau} = \frac{1}{2}\frac{\partial^2\psi(q,\tau)}{\partial q^2}, \qquad \psi(0,\tau)=\psi(1,\tau)=0 \tag{8.10}$$

である。この方程式の解析解は求まるが，ここでは使わない。$\Delta\tau \ll 1$ のとき

$$\psi(q,\tau+\Delta\tau) \approx \frac{1}{\sqrt{2\pi\Delta\tau}}\int_0^1 dq'\exp\left(-\frac{(q-q')^2}{2\Delta\tau}\right)\psi(q',\tau) \tag{8.11}$$

になる。実際，$\Delta\tau\ll1$ の場合 $q'\approx q$ 近傍だけが寄与するから，テイラー展開

$$\psi(q',\tau) = \psi(q,\tau) + \sqrt{2\Delta\tau}\,u\,\psi'(q,\tau) + \Delta\tau\,u^2\psi''(q,\tau)+\cdots, \qquad u=\frac{q'-q}{\sqrt{2\Delta\tau}}$$

を代入すると

$$\psi(q,\tau+\Delta\tau) = \frac{1}{\sqrt{\pi}}\int_{-u_0}^{u_1} du\,e^{-u^2}\left(\psi(q,\tau)+\sqrt{2\Delta\tau}\,u\,\psi'(q,\tau)+\Delta\tau\,u^2\psi''(q,\tau)+\cdots\right)$$

ただし $u_0=q/\sqrt{2\Delta\tau}$, $u_1=(1-q)/\sqrt{2\Delta\tau}$ である。$0<q<1$ のとき u_0, $u_1 \xrightarrow{\Delta\tau\to0} \infty$ より

$$\psi(q,\tau+\Delta\tau) = \psi(q,\tau) + \frac{\Delta\tau}{2}\psi''(q,\tau)+\cdots$$

になり (8.10) を満たす。$\psi(0,\tau)=\psi(1,\tau)=0$ ではないから，$q\le0$, $q\ge1$ では $\psi(q,\tau)=0$ とする。

$\psi(q,\tau)$ で分布する N_τ 個の乱数を q_k とする ($0<q_k<1$)。$\psi(q,\tau)$ は規格化されていないから

$$P(\tau) = CN_\tau, \qquad C = \tau\text{に依存しない定数}$$

である。$\psi_0(q)$ で分布する乱数が N_0 個のとき，$C=1/N_0$ とすれば $P(0)=1$ になる。(8.11) の $\psi(q,\tau+\Delta\tau)$ は，平均 0, 分散 $\Delta\tau$ の正規分布と分布 $\theta(q)\theta(1-q)\psi(q,\tau)$ の畳み込みである。(8.2) より

$$Q_k = q_k + \sqrt{\Delta\tau}\,v_k, \qquad v_k = \text{平均 0, 分散 1 の正規乱数}$$

は $\psi(q, t+\Delta\tau)$ で分布する。$Q_k \le 0$, $Q_k \ge 1$ の Q_k を棄却すると $P(\tau+\Delta\tau) = \dfrac{Q_k \text{の個数}}{N_0}$ である。

$$\log P(\tau) \xrightarrow{\tau\to\infty} -E_0 t/\hbar + \text{const.} = -\varepsilon_0 \tau + \text{const.}\,, \qquad \varepsilon_0 = ML^2E_0/\hbar^2$$

になる。解析解は $\varepsilon_0 = \pi^2/2$ である。n が奇数のとき $\varphi_n(x)$ の $0 \le x \le L$ での積分は 0 になるから

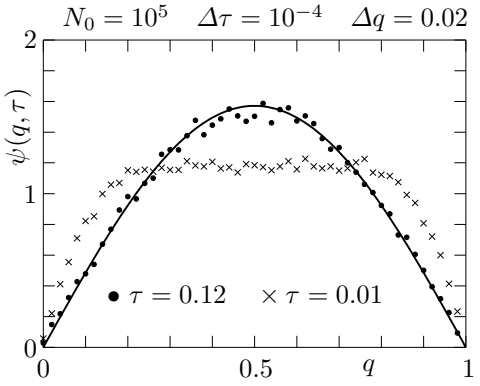
$$P(t) = D_0 e^{-E_0 t/\hbar}\left(1 + \frac{D_2}{D_0} e^{-(E_2-E_0)t/\hbar} + \cdots\right), \qquad E_n = (n+1)^2 E_0$$

したがって

$$\frac{(E_2-E_0)t}{\hbar} = \frac{\tau}{\tau_0} \gg 1\,, \qquad \tau_0 = \frac{1}{8\varepsilon_0} = \frac{1}{4\pi^2} \approx 0.025$$

では $P(t) \approx D_0 e^{-E_0 t/\hbar}$ になる。

問 8.6　$\psi_0(q) = 1$ とする。この分布をする乱数は $[0,1]$ の一様乱数である。ε_0 と $\psi(q,\tau)$ を求め，解析解と比較せよ。下図に波動関数を示す。曲線は $\varphi_0(q) = (\pi/2)\sin\pi q$ である。



付録 A　数値計算の結果を図で表わす：TEX の tpic special

数値計算の結果を図で表わすにはいろいろな方法があるが, ここでは TEX の tpic ¥special を用いて表現しよう。TEX については当面何も知らなくてよい。以下のようなテキストファイル (拡張子は tex) を作成する (数値結果を出力する場合, 計算プログラム自体に tex ファイルを作成させる)。なお, 円記号 ¥ とバックスラッシュ \ は同じである。

```
¥documentclass{jarticle}
¥begin{document}
¥unitlength 0.001in
¥begin{picture}(x_size,y_size)(x_move,y_move)
この間にtpic命令が入る
¥end{picture}
¥end{document}
```

x_size, y_size, x_move, y_move は数値である。tpic では長さの単位はミリインチであるため, TEX 側の単位もミリインチを指定する (¥unitlength 0.001in)。picture 命令は横幅 x_size ミリインチ, 縦 y_size ミリインチの図を挿入するための領域を確保する。x_move と y_move は図全体の位置を移動するためのパラメータである。x_move は左方向に x_move ミリインチ, y_move は下方向に y_move ミリインチずらす (負の値を指定すると逆方向にずれる)。

例えば, 次のような内容の test.tex というファイルを作成する。

```
¥documentclass{jarticle}
¥begin{document}
¥unitlength 0.001in
¥begin{picture}(1100,600)(0,-550)
¥special{pn 20}%
¥special{pa 0 0}%
¥special{pa 1000 0}%
¥special{pa 1000 500}%
¥special{fp}%
¥special{pn 8}%
¥special{pa 1000 500}%
¥special{pa    0 500}%
¥special{pa    0   0}%
¥special{fp}%
¥end{picture}
¥end{document}
```

pn, pa, fp が tpic の命令で

¥special{tpic命令 必要ならパラメータ}%

という形式である。tpic 命令の具体的意味は次節を参照せよ。test.tex を LATEX で処理する。処理法はシステムで異なる。Windows ならばコマンド・プロンプト (MS–DOS プロンプト) を起動し

platex test

と入力して Enter キーを押す。拡張子 .tex は省略できる。各種のファイルが新たに作成されるが, 拡張子が dvi というファイルに出力イメージが格納されている。今の場合 test.dvi である。dvi ファイルはテキストファイルではないので, 出力イメージを画面上で見るためにはプレビュアと呼ばれるプログラムを起動する。Windows の場合 dviout.exe である。画面左上に右図が現れるはずである。

曲線の表現　$x_{\min} \le x \le x_{\max}$, $y_{\min} \le y \le y_{\max}$ の範囲を横 xsize, 縦 ysize ミリインチの領域にマップしよう。tpic 座標では x 軸は右が正, y 軸は**下向きが正方向**である (次節参照)。したがって

$$(x_{\min},\, y_{\min}) \longrightarrow (0, \mathtt{ysize})\,, \quad (x_{\max},\, y_{\max}) \longrightarrow (\mathtt{xsize}, 0)$$

となればよいから, 点 $(x,\, y)$ の tpic 座標 (X, Y) は

$$X = \frac{\mathtt{xsize}}{x_{\max} - x_{\min}}(x - x_{\min})\,, \quad Y = \mathtt{ysize} - \frac{\mathtt{ysize}}{y_{\max} - y_{\min}}(y - y_{\min}) \qquad \text{(付録 A.1)}$$

である。曲線 $y = f(x)$ ($a \le x \le b$) を描くには, 区間 $[a, b]$ を n 等分して, $n + 1$ 個の点

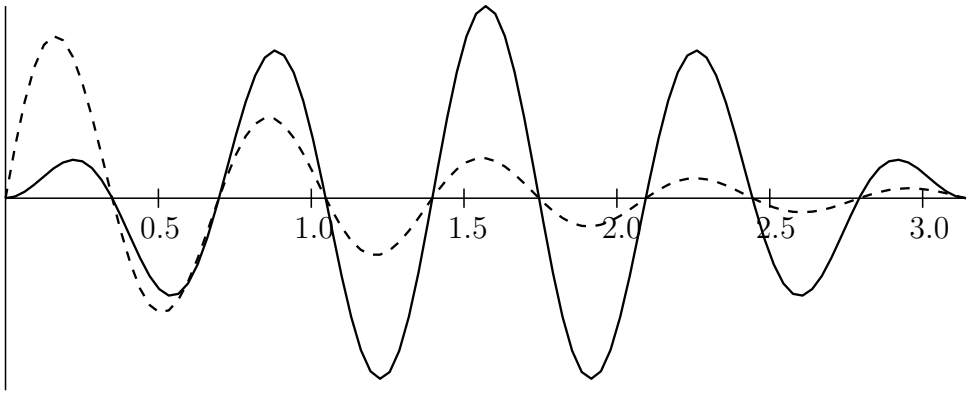
$$\big(x_k,\, f(x_k)\big)\,, \quad x_k = a + \frac{b-a}{n}k \;\; (\, k = 0, 1, 2, \cdots, n\,)$$

を tpic 座標に変換しこれらを直線で結ぶ。n をある程度大きくとれば曲線に見える。

次ページの C プログラムは $f(x) = \sin x \, \sin 9x$ と $g(x) = \exp(-x) \, \cos 9x$ を図示する TEX ファイル sin.tex を出力する。このファイルを LATEX で処理すると下図を得る。C プログラムのファイルは

http://physics.s.chiba-u.ac.jp/~kurasawa/fig_tpic.c

にある。



この例では, 横 5 インチ, 縦 2 インチの領域に幅 20 ミリインチの実線で $f(x)$, 破線で $g(x)$ を描き, x 軸と y 軸を幅 8 ミリインチの直線で描く。上の曲線は各々 100 個の線分からなる。

- 関数 tp_conv は (付録 A.1) 式に従い x, y を tpic 座標に変換する。return では 1 つの値しか返せないので, ここでは tpic 座標 (X, Y) の 2 つのポインタを引数にしている。
- tp_line は 2 点 $(x_1,\, y_1)$, $(x_2,\, y_2)$ を結ぶ線分を引く。また, tp_func は配列 x[], y[] が与えられたとき, 点 (x[imin], y[imin]), $\cdots$,(x[imax],y[imax]) を線分で結ぶ。tp_line, tp_func では描画命令を文字列 fmt に与える。例えば, 実線なら "fp" である。
- 図に文字列を貼り付けるには, picture 環境の ¥put 命令を使う。¥put 命令の原点と tpic 原点は同じだが, y 軸の向きは逆である。したがって, tpic 座標 (X, Y) に文字列を表示するには ¥put(X, $-Y$){文字列} とする。ただし, ¥unitlength=0.001in として, picture 環境での長さの単位をミリインチにしておく。
- C の文字列中では ¥ と % はエスケープキャラクタであり特別な意味をもつ。¥, % それ自身はそれぞれ ¥¥, %% で表わす。
- TEX は改行を space 1 つと解釈することがある。これを行わせないため, ¥specail{　}% として行末に % を付加し, 継続行としてある。

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#ifndef PI
#define PI 3.14159265358979323846
#endif
#define NMAX 100
void tp_conv( double x, double y, double *tpx, double *tpy );
void tp_pa( double x, double y );
void tp_line( double x1, double y1, double x2, double y2, char *fmt );
void tp_func( double *x, double *y, int imin, int imax, char *fmt );
FILE *fp_tpic;
double xd[NMAX+1], yd1[NMAX+1], yd2[NMAX+1];
double cx, cy, xmin, ymin, ysize;
int main()
{
    double h, x, y, xmax, ymax, xsize, tpx, tpy;
    int k;
    char *filename, string[128];
    xmin = 0;
    xmax = PI;
    ymin = -1;
    ymax = 1;
    h = ( xmax - xmin )/NMAX;
    for(k=0; k<=NMAX; k++){
        xd[k] = xmin + h*k;
        yd1[k] = sin(xd[k])*sin(9*xd[k]);
        yd2[k] = exp( -xd[k] )*cos(9*xd[k]);
    }
/*********  TeX出力  **********/
    xsize = 5000;
    ysize = 2000;
    cx = xsize/( xmax - xmin );
    cy = ysize/( ymax - ymin );
    filename = "sin.tex";
    fp_tpic = fopen(filename,"w");
    fprintf(fp_tpic,"¥¥documentclass{jarticle}¥n");
//    fprintf(fp_tpic,"¥¥usepackage[dvips]{color}¥n");
//    fprintf(fp_tpic,"¥¥ExecuteOptions{usenames}¥n");
    fprintf(fp_tpic,"¥¥begin{document}¥n");
    fprintf(fp_tpic,"¥¥unitlength 0.001in¥n");
    fprintf(fp_tpic,"¥¥begin{picture}(5000,2000)(0,-2000)¥n");
/*******  関数描画  ******/
    fprintf(fp_tpic,"¥¥special{pn 16}%%¥n");
    tp_func( xd, yd1, 0, NMAX, "fp" );
    tp_func( xd, yd2, 0, NMAX, "da 0.05 0.05" );
//    tp_func_color( xd, yd1, 0, NMAX, "fp", 1, 0, 0 );
//    tp_func_color( xd, yd2, 0, NMAX, "fp", 0, 1, 0 );
//    tp_func_ncolor( xd, yd1, 0, NMAX, "fp", "Cyan" );
//    tp_func_ncolor( xd, yd2, 0, NMAX, "fp", "VioletRed" );
/********  座標軸  *******/
    fprintf(fp_tpic,"¥¥special{pn 8}%%¥n");
    tp_line( xmin, 0., xmax, 0., "fp" );      /*** x-軸 ***/
    tp_line( 0., ymin, 0., ymax, "fp" );      /*** y-軸 ***/
```

```
    for(k=1; k<=6; k++ ){                    /*** x-軸目盛り ***/
        x=0.5*k;
        y=0;
        tp_line( x, -0.05, x, 0.05, "fp" );
        tp_conv( x, y, &tpx, &tpy );
        fprintf(fp_tpic,"¥¥put(%.0f,%.0f){¥¥large$%.1f$}%%¥n",
        tpx-100, -tpy-200, x );
    }
    fprintf(fp_tpic,"¥¥end{picture}¥n");
    fprintf(fp_tpic,"¥¥end{document}¥n");
    fclose(fp_tpic);
/********* pLaTeX 実行 ********/
    sprintf(string,"platex %s", filename );
    system(string);
    return 0;
}

void tp_conv( double x, double y, double *tpx, double *tpy )
{
    *tpx = cx*( x - xmin );
    *tpy = ysize - cy*( y - ymin );
}

void tp_pa( double x, double y )
{
    double tpx, tpy;
    tp_conv( x, y, &tpx, &tpy );
    fprintf(fp_tpic,"¥¥special{pa %.0f %.0f}%%¥n",tpx, tpy);
}

void tp_line( double x1, double y1, double x2, double y2, char *fmt )
{
    tp_pa( x1, y1 );
    tp_pa( x2, y2 );
    fprintf(fp_tpic,"¥¥special{%s}%%¥n", fmt );
}

void tp_func( double *x, double *y, int imin, int imax, char *fmt )
{
    int i;
    for( i=imin; i<=imax; i++ )
        tp_pa( x[i], y[i] );
    fprintf(fp_tpic,"¥¥special{%s}%%¥n", fmt );
}
```

カラー

color パッケージを使うと文字や曲線に色を付けて出力できる。色の指定には幾つか方法がある。

- 色指定を赤 (red), 緑 (g), 青 (b) の強度指定で行う。tpic の出力コマンド (fp, da 等) 前に ¥textcolor[rgb]{c_r,c_g,c_b} を挿入して, 例えば

  ¥textcolor[rgb]{0.5,0.9,0.8}{¥special{fp}}

  とする。c_r, c_g, c_b は 0 から 1 の間の数値で, 基本的な色の組み合わせ (c_r, c_g, c_b) は黒 $(1,1,1)$, 青 $(0,0,1)$, 赤 $(1,0,0)$, 緑 $(0,1,0)$, 水色 $(0,1,1)$ などである。上のサンプルプログラムで

```
//   fprintf(fp_tpic,"¥¥usepackage[dvips]{color}¥n");
```

のコメントを外し, tp_func を下記のものに置き換えてみよ。

```
void tp_func_color( double *x, double *y, int imin, int imax, char *fmt,
                    double r, double g, double b )
{
    int i;
    for( i=imin; i<=imax; i++ )
        tp_pa( x[i], y[i] );
    fprintf(fp_tpic,"¥¥textcolor[rgb]{%f,%f,%f}{%%¥n", r, g, b );
    fprintf(fp_tpic,"¥¥special{%s}%%¥n", fmt );
    fprintf(fp_tpic,"}%%¥n" );
}
```

- color パッケージで定義済みの 68 色を使う。¥textcolor[rgb]{...}{...} の代わりに

  ¥textcolor[named]{定義済みの色}{...}

とする。

```
void tp_func_ncolor( double *x, double *y, int imin, int imax, char *fmt,
                     char *color )
{
    int i;
    for( i=imin; i<=imax; i++ )
        tp_pa( x[i], y[i] );
    fprintf(fp_tpic,"¥¥textcolor[named]{%s}{%%¥n", color );
    fprintf(fp_tpic,"¥¥special{%s}%%¥n", fmt );
    fprintf(fp_tpic,"}%%¥n" );
}
```

使いたい色名を文字列で指定する。例えば, GreenYellow を使いたければ

```
tp_func_ncolor( ..... , "GreenYellow" );
```

である。定義済みの色名については dviout に含まれる color.dvi を参照。

GreenYellow	Yellow	Goldenrod	Dandelion	Apricot
Peach	Melon	YellowOrange	Orange	BurntOrange
Bittersweet	RedOrange	Mahogany	Maroon	BrickRed
Red	OrangeRed	RubineRed	WildStrawberry	Salmon
CarnationPink	Magenta	VioletRed	Rhodamine	Mulberry
RedViolet	Fuchsia	Lavender	Thistle	Orchid
DarkOrchid	Purple	Plum	Violet	RoyalPurple
BlueViolet	Periwinkle	CadetBlue	CornflowerBlue	MidnightBlue
NavyBlue	RoyalBlue	Blue	Cerulean	Cyan
ProcessBlue	SkyBlue	Turquoise	TealBlue	Aquamarine
BlueGreen	Emerald	JungleGreen	SeaGreen	Green
ForestGreen	PineGreen	LimeGreen	YellowGreen	SpringGreen
OliveGreen	RawSienna	Sepia	Brown	Tan
Gray	Black	White		

tpic 2.2 コマンドセット仕様

TEX が保持している「ページ上の現在位置」を原点にとり, x 軸は右が正、y 軸は下が正方向になる (y 軸は picture 環境と逆向き)。座標は整数で表し, 単位はミリインチ。角度は時計回りが正で, ラジアン単位で記述する。

pn s　線幅を s ミリインチに設定する。

pa x y　点 (x, y) をパスに付け加える。x と y の単位はミリインチ。

fp　それまでに定義したパスを現在の線幅で実際に描く。パス内の点の数はゼロにリセット。シェーディングが設定され, かつパスが閉じていれば, パス内部を塗りつぶす。

ip　fp と同じだがパスを描かない。シェーディングは条件が満たされれば行う。

da f　パスを破線で描く。f は実数でダッシュ当たりの長さ (インチ単位)。

dt f　パスを点線で描く。f は実数で点の間隔 (インチ単位)。

sp d　パスをスプライン曲線で描く。d は実数で, 曲線の種類を指定する。$d = 0$ か d が省略された場合, 実線で描く。$d > 0$ の場合, 破線で描く。d はダッシュの長さ。$d < 0$ の場合, 点線で描く。$-d$ は点間隔。

ar x y r_x r_y s e
　中心 (x, y) の弧を描く。s は開始角度, e は終了角度である。完全な円か楕円である場合は, r_x, r_y はそれぞれ x, y 半径を表す。そうでない場合は $r_x = r_y$ であり, s から e へ弧を描く。条件が満たされればシェーディングも行う。x, y, r_x, r_y の単位はミリインチ。

ia x y r_x r_y s e
　これは ar と同じだが, 弧は描かず, 条件が満たされればシェーディングのみ行う。

sh s　シェーディングを行なう。このコマンドの次に定義される閉じた図形の内部を塗りつぶす。s は 0(白) 以上 1(黒) 以下の実数である。$0 < s < 1$ のとき, 図形の下のものは消去せずに灰色で塗り加える。s が指定されていなければ 0.5 の値を用いる。シェーディングは図形の内側のみに対して行い, 境界線はそれとは独立に現在の線幅を用いて描かれる。

Windows 用プレビュア dviout は以下の拡張機能をサポートしている。一般に dviout 以外ではサポートされないので、これらを使用して作成した dvi ファイルは、互換性の点で問題がある。

da コマンドの拡張
　da コマンドを拡張し, 複数の引数が取れる。引数は交互に実線部分, 空白部分の長さをインチ単位の実数で指定する。この場合, 破線はこの指定パターンのくり返しで描かれ, パスの各点を「連続に」通過する。例えば

```
¥special{da 0.1}%      o___    ___     ___o___    ___
¥special{da 0.1 0.1}% o___    ___    ___ o ___    ___
¥special{da 0.1 0.01 0.01 0.01}          % 一点鎖線
```

rt　回転や拡大, 鏡映などの線形変換を施した描画を行う。詳しくは dviout の HELP 参照。

　rt h v
　カレントポイントを原点として、以後の全てのテキストおよびグラフィックス出力を横方向に h 倍、縦方向に v 倍にスケール変換する。負の値も可能。rt 1 1 でリセット

　rt x y f
　以後の全てのテキストおよびグラフィックス出力を (x, y) の回りに角度 f だけ回転させる。rt x y 0 でリセット。

Bz n　sp コマンドで描画する曲線をベジェ ($n = 0$) またはスプライン ($n = 1$) にする。

付録 B　数値計算の結果を図で表わす：PostScript

PostScript 言語は汎用ページ記述言語であり, DTP では事実上の標準である。通常, PostScript 言語を直接プログラミングすることはなく, お絵描きソフトなどのアプリケーションが PostScript コードを生成する。ユーザは PostScript 言語について知る必要はない。しかし, 前節の tpic 程度のことならば, 自分で PostScript コードを出力するプログラムを書くことは簡単である。ここでは tpic に対応する PostScript の演算についてまとめておく。これは PostScript 全体の極一部である。なお, 他の言語にない PostScript 言語の特徴は次の 2 点である。

- **逆ポーランド法 (後置記法)**
  数学で用いられる記法には, 演算子 (オペレータ) とその引数 (オペランド) の並び方により, ポーランド法 (前置記法), 中間記法, 逆ポーランド法 (後置記法) がある。ポーランド法は引数列の前に演算子を記す方法で, $\sin\theta$, $f(x,y)$ などがポーランド法である。中間記法の典型は四則演算子 $\pm$, $\times$, $\div$ である。多くのプログラミング言語は, 数学で使われている記法に近い設計がなされ, 引数列の後に演算子くる逆ポーランド法はめったにないが, PostScript は逆ポーランド法の言語である。$a+b$ は `a b add` と書き, $\sin a$ も `a sin` と表す (これは電卓で `sin` を計算するときと同じ)。
- **オペランド・スタック**
  PostScript では, 演算子への引数の引き渡しにスタックを陽に使う。引数は一旦 1 次元配列のスタックに格納される (push する)。次々に値を追加するとき, 後から格納する値は先に格納した値の上になる。演算子はこのスタックから必要な個数の引数を取り出す (pop する)。この時, 原則として一番上にあるものから取り出す。取り出された値はスタックから消去される。演算の実行時に引数がスタックに正しい順序で格納されていればよい。それさえ正しければ, PostScript のソースコードで演算子と引数の間に関係のない演算子や引数があってもよい。一般に演算の結果はスタックに格納される。

表記形式　引数 1 引数 2 $\cdots$ `operator` 結果 1 結果 2 $\cdots$。引数, 結果がない場合は - で表す。

座標系

デフォルトの座標系は, 原点が左下隅, x 軸は水平右向き, y 軸は垂直上向きである。長さの単位は $1\,\mathrm{pt} = 1/72\,\mathrm{inch} = 0.3528\,\mathrm{mm}$。角度は度単位で正のとき反時計周り。

x y `translate` -　新しい原点を (x, y) に移動する。

x y `scale` -　x は x 軸方向, y は y 軸方向の拡大率。デフォルトに対して `72 72 scale` とするとインチ単位になる。

θ `rotate` -　座標系を θ 度回転させる。

描画

- `showpage` -　カレントページを出力し, ページをクリアして初期化する。

- `newpath` -　パスを初期化する。

x y `moveto` -　パスに初めてカレントポイントを加える。

x y `lineto` -　パスに新しいカレントポイントを加え, 古いカレントポイントと新しいカレントポイントを線分で結ぶ。`moveto` を実行していない場合エラーになる。

x y r θ_1 θ_2 `arc` -　中心 (x, y), 半径 r の円弧 (開始角 θ_1, 終了角 θ_2) をパスに追加する。カレントポイントがあれば, この点から円弧の開始点までの線分も付加する。

- closepath -　パスに始点 (moveto で指定した点) とカレントポイントを結ぶ直線を加えてパスを閉じる。

- stroke -　実際にパスに沿って線を描く。色, 線幅, パターンについては次のグラフィック状態を参照。描画後, newpath を実行する。

- fill -　カレントカラーでパスで囲まれた領域を塗りつぶし, newpath を実行する。パスが閉じてないときは始点と終点を結ぶ。

グラフィック状態

r g b setrgbcolor -　カレントカラーを赤 r, 緑 g, 青 b で指定した色にする。パラメータは 0 から 1 の範囲。線, 領域, 文字を描く際に用いる色を指定する。

x setgray -　カレントカラーを x で指定した明度の灰色にする。パラメータは 0(黒) から 1(白) の範囲である。線, 領域, 文字を描く際に用いる色を指定する。

pattern offset setdash -　stroke で描く破線を設定する。pattern は [x_1 x_2 $\cdots$] という形式で, 左から右へ順に線, 隙間の長さを表し, 右端にきたら再び左端に戻る。奇数個指定した場合, パターンは逆転する。offset は破線パターンを offset だけ線方向にずらす。実線に戻すには [] 0 setdash とする。

x setlinewidth -　stroke で描く線の幅を設定する。

フォント

fontname findfont font　fontname はフォント名に / を付けたもので /Times-Roman など。font は選択したフォントに関する情報を含む。具体的にどのようなものかは知らなくてよい。スタックに置いておき, 次の scalefont で使う。フォント名については, ファイル Fontmap を見よ。

font s scalefont font　フォントの大きさを s にする。

font setfont -　指定したフォントをカレントフォントにする。実際のフォント指定は

fontname findfont s scalefont setfont

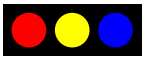
string show -　カレントポイントを印字位置として文字列 string をカレントフォントで出力する。カレントポイントを moveto などで最初に定義しておく必要がある。文字列は括弧で区切る。特殊文字は括弧と ¥。文字列中の括弧は ¥(, ¥) で表す。

簡単な例

次のソースを適当なファイル名で保存し GhostScript で表示させてみよ。ただし, % より右側はコメント文である。GhostScript については http://www.cs.wisc.edu/~ghost/index.html を参照。Windows 用の GhostScript が入手できる (Freeware)。

線幅を変えながら破線を描く。

```
2 setlinewidth     % 線幅を2ptにする 。
newpath
100 100 moveto     % ( x, y ) = ( 100, 100 )
450 100 lineto     % ( x, y ) = ( 450, 100 )
stroke             % ( 100, 100 )と( 450, 100 )を結ぶ実線を引く 。
3 setlinewidth
[10 20] 0 setdash  % 破線 - - - - の設定
newpath 100 140 moveto 450 140 lineto stroke
showpage
```

信号機 

```
newpath
 40  40 moveto 360 40 lineto
360 160 lineto 40 160 lineto
closepath fill                  % 黒い長方形(デフォルトは黒)
1 0 0 setrgbcolor               % 色を赤に設定
newpath 100 100 40 0 360 arc    % 中心 (100, 100 ), 半径 40 の円
fill                            % 円の内部を赤で塗りつぶす
1 1 0 setrgbcolor newpath 200 100 40 0 360 arc fill  % 黄色い円
0 0 1 setrgbcolor newpath 300 100 40 0 360 arc fill  % 青い円
showpage
```

フォント

```
/Times-Roman findfont 24 scalefont setfont
100 100 moveto (Times-Roman) show
/Courier findfont 24 scalefont setfont
(Courier) show
showpage
```

計算結果を PostScript で表示する

次の C プログラムは $f(x) = \sin x \sin 9x$, $g(x) = \exp(-x)\cos 9x$ のグラフを描く PostScript ファイル `sin.ps` を出力する。方法は tpic の場合と同じである。座標 (x, y) から PostScript 座標への変換は `ps_conv` で行なう。PostScript 座標を (X, Y) とすると

$$X = \frac{\texttt{xsize}}{x_{\max} - x_{\min}}(x - x_{\min}), \quad Y = \frac{\texttt{ysize}}{y_{\max} - y_{\min}}(y - y_{\min})$$

である。ファイル `sin.ps` を GhostScript で表示してみよ。

```
#include <stdio.h>
#include <math.h>
#ifndef PI
#define PI 3.14159265358979323846
#endif
#define NMAX 100
void ps_line( double x1, double y1, double x2, double y2 );
void ps_func( double *x, double *y, int imin, int imax );
void ps_conv( double x, double y, double *psx, double *psy );
FILE *fp_ps;
double cx, cy, xmin, ymin, xd[NMAX+1], yd1[NMAX+1], yd2[NMAX+1];
int main()
{
    double h, x, y, xmax, ymax, xsize, ysize, ux, uy;
    int k;
    xmin = 0;
    xmax = PI;
    ymin = -1;
    ymax =  1;
    h = ( xmax - xmin )/NMAX;
    for(k=0; k<=NMAX; k++){
        xd[k] = xmin + h*k;
```

```
        yd1[k] = sin(xd[k])*sin(9*xd[k]);
        yd2[k] = exp(-xd[k])*cos(9*xd[k]);
    }
/***** PS出力 ****/
    xsize = 5*72;   /* 横 5inch */
    ysize = 2*72;   /* 縦 2inch */
    cx = xsize/( xmax - xmin );
    cy = ysize/( ymax - ymin );
    fp_ps=fopen("sin.ps","w");
    fprintf(fp_ps,"50 50 translate 1.5 setlinewidth¥n");
    ps_func( xd, yd1, 0, NMAX );
    fprintf(fp_ps,"[5] 0 setdash¥n");
    ps_func( xd, yd2, 0, NMAX );
    fprintf(fp_ps,"[ ] 0 setdash¥n");
    fprintf(fp_ps,"0.5 setlinewidth¥n");
    ps_line( xmin, 0., xmax, 0 );         /**  x-軸 **/
    ps_line( 0., ymin, 0, ymax );         /**  y-軸 **/
    fprintf(fp_ps,"/Helvetica-Narrow findfont 14 scalefont setfont¥n");
    for( k=1; k<=6; k++ ){                /**  x-軸目盛り **/
        x=0.5*k;
        y=0.;
        ps_line( x, -0.05, x, 0.05 );
        ps_conv( x, y, &ux, &uy );
        fprintf(fp_ps,"%.3f %.3f moveto (%.1f) show¥n",ux-8, uy-18, x );
    }
    fprintf(fp_ps,"showpage¥n");
    fclose(fp_ps);
    return 0;
}

void ps_line( double x1, double y1, double x2, double y2 )
{
    double ux1, uy1, ux2, uy2;
    ps_conv( x1, y1, &ux1, &uy1 );
    ps_conv( x2, y2, &ux2, &uy2 );
    fprintf(fp_ps,"newpath %.3f %.3f %.3f %.3f moveto lineto stroke¥n",
            ux2, uy2, ux1, uy1 );
}

void ps_func( double *x, double *y, int imin, int imax )
{
    double ux, uy;
    int i;
    ps_conv( x[imin], y[imin], &ux, &uy );
    fprintf(fp_ps,"newpath %.3f %.3f moveto¥n",ux, uy);
    for(i=imin+1; i<=imax; i++ ){
        ps_conv( x[i], y[i], &ux, &uy );
        fprintf(fp_ps,"%.3f %.3f lineto¥n",ux, uy);
    }
    fprintf(fp_ps,"stroke¥n");
}

void ps_conv( double x, double y, double *psx, double *psy )
{
    *psx = cx*( x - xmin );
    *psy = cy*( y - ymin );
}
```

EPS(Encapsulated PostScript) 形式

複数の PostScript ファイルを合成する場合の形式として EPS 形式がある。EPS 形式は PostScript ファイルが他の PostScript ファイルに挿入されたときに副作用を起こさないための規約である。例えば, `showpage` があるとその部分でページ出力されてしまうから, EPS ファイルは `showpage` を含んではいけない。また, 色などのグラフィック状態を変更したら元に戻す。そのためには EPS ファイルの先頭に `gsave`, 終わりに `grestore` を置く。`gsave` はカレントのグラフィック状態を複製して保存する演算, `grestore` は `gsave` で保存したグラフィック状態を回復する演算である。EPS ファイルで, 必ず含まなければならないコメント文として

`%%BoundingBox:` x_1 y_1 x_2 y_2

がある。これは EPS ファイルの描画領域を指定するコメント文で, x_1, y_1 は領域の左下の座標, x_2, y_2 は右上の座標である。TEX ファイルに PostScript の図を挿入するとき, `sty` ファイルによっては, このコメント文で図の位置や大きさ合わせをする。

EPS ファイルの基本形は

```
%%BoundingBox: x_1 y_1 x_2 y_2
gsave
   PostScriptコード
grestore
```

である。

付録 C　Mule(Emacs) のコマンド一覧

C-x　CTRL キーを押しながら x キーを押す。　M-x　ESC キーを押してから x キーを押す。
RET　Enter キーを押す。　SPC　スペース

特殊キー

矢印キー (→, ←, ↑, ↓)　1 文字移動
Insert キー　上書きモード ⟷ 挿入モード
Delete キー　カーソル位置の文字削除　Back space キー　カーソル位置前の文字削除

ファイル操作など

C-x C-f　ファイル読み込み
C-x C-v　現ファイルと置き換えてファイル読み込み。
C-x i　現在位置にファイルを挿入する。
C-x C-s　ファイルを保存する。
C-x C-w　バッファを指定ファイルに書き出す。
C-x C-c　終了する。
C-l　画面描画しカーソルを画面中央に移動
C-z　一時停止しコマンドプロンプトに移行する。exit か fg で戻る。
C-g　現在のコマンドを中止する。
C-h　オンラインヘルプに入る。
C-x (　キーボードマクロ定義開始。　C-x) 定義終了　C-x e マクロ実行
M-n command　command を n 回繰り返す。
C-u n command　command を n 回繰り返す。n を指定しないと 4 回繰り返す。

カーソル移動

C-p　前行へ	C-n　次行へ	C-b　1 文字左へ
C-f　1 文字右へ	C-a　行頭へ	C-e　行末へ
M-b　ワード単位で戻る。	M-f　ワード単位で進む。	M-a　センテンス先頭に
M-e　センテンス末尾に	M->　ファイル末尾に	M-<　ファイル先頭に

M-x goto-line　ファイル先頭から n 行目に移動

削除

C-d　カーソル位置の文字	M-d　カーソル位置からワードの最後
C-k　カーソル位置から行末	M-0 C-k　行頭からカーソル位置 (0 はゼロ)
M-k　カーソル位置からセンテンス最後	C-y　直前に削除した内容を復元

カット & ペースト

C-@, C-SPC　領域の始端 (終端) をマーク	M-h　パラグラフをマーク
C-x h　バッファ全体をマーク	C-w　マークした領域を削除
M-w　マークした領域のコピー	C-y　削除・コピーした部分をペースト
C-x C-x　カーソルとマークの位置を交換	

その他の編集機能

M-c　ワード先頭を大文字　M-u　ワード全体を大文字　M-l　ワード全体を小文字
M-x overwrite-mode　上書きモード ⟷ 挿入モード

ミニバッファ

TAB　補完　　SPC　1 ワード補完　　?　補完候補一覧
M-p　入力履歴 up　　M-n　入力履歴 down

検索

C-s, C-r　順方向 (C-s), 逆方向 (C-r) にインクリメンタルサーチを開始。
文字を入力せず RET で単純サーチ。更に C-w でワードサーチ。

C-s, C-r : 次のサーチ文字列を検索。　RET : 終了
DEL : サーチ文字列から文字を削除　　C-g : 中断して開始位置に戻る。

置換

M-x replace-string　カーソル位置以降に表れるすべてのワードを確認なしに置換する。

M-%　対話的置換
SPC : 置換する　n : 置換しない　! : 残りの置換を確認なしに行う　M : 終了
C-r : リカーシブエディット (置換を中断して編集)
M-C-c : リカーシブエディットを抜け置換再開
C-] : リカーシブエディットと置換の両方を抜ける。

マルチプルバッファ

C-x b　バッファ変更 (TAB でバッファ一覧)　　C-x k　バッファ削除。
C-x s　複数バッファのセーブ

マルチウィンドウ

C-x 0　現ウィンドウ削除 (0 はゼロ)　　C-x 1　現ウィンドウ以外をすべて削除
C-x 2　現ウィンドウを上下に分割　　C-x 3　現ウィンドウを左右に分割
C-x o　他のウィンドウに移動　　C-x 4 f　他ウィンドウにファイル読込み
C-x ^　現ウィンドウを 1 行増やす　　C-x 4 b　他ウィンドウのバッファ変更

シェル

M-!　Unix コマンドを実行　　M-| マークした領域を Unix コマンドに渡して実行
M-x shell　Unix シェル
C-c C-c　ジョブ強制終了　　C-c C-z　ジョブ一時停止
C-c C-y　直前の入力コマンド表示　　C-c C-o　直前のコマンド出力削除
C-c C-r　直前のコマンド出力の最初に移動

言語モード

M-;　コメント挿入　　M-j　コメントを次行に続ける。
M C-¥　マーク領域をインデント　　M-m　行の最初の非ブランク文字に移動
M-^　前行と現在行を結合　　C-x ‘　エラー行に移動 (‘ はバッククォート)
M-x compile　コンパイル

C 言語モード

M C-a　関数先頭に移動　　M C-e　関数末尾に移動　　M C-h　関数全体をマーク

YaTeX

プロセス

C-c t j　コンパイル。実行コマンドは (setq tex-command "platex") 等で指定。
C-c t r　領域を指定してコンパイル
C-c t p　プレビューア起動。プレビューアは (setq dvi2-command "dviout")
C-c ’　エラー行に移動。タイプセットバッファでエラー表示行にカーソルを合わせて SPC を押すとエラー行に移動。
%#記法　本文に %#!<command> と書くと <command> をそのまま shell に渡す。
例 : %#!latex main.tex
%#BEGIN と %#END で囲むと C-c tr でその領域だけコンパイルできる。

補完

C-c b	¥begin{xxx} 型の補完	C-c B	マーク領域を ¥begin{xxx} 型で囲む
C-c s	¥command{xxx} 型の補完	C-c S	マーク領域を ¥command{xxx} に
C-c l	{¥command } 型の補完	C-c L	マーク領域を {¥command xxx} に
C-c e	¥end 補完	C-c a	アクセント記号補完

C-c SPC　先頭が ¥ で始まる文字列を入力中に補完する
C-c e　¥end 補完
C-c a　アクセント記号補完
;　数式モード中で矢印や記号を入力
:　数式モード中でギリシャ文字を入力。:a で ¥alpha

コメントアウト

C-c >　マーク領域をコメントアウト。
カーソルが ¥begin あるいは ¥end にあるときは ¥begin ～ ¥end をコメントアウト
C-c <　マーク領域のコメントを外す。
カーソルが ¥begin あるいは ¥end にあるときは ¥begin ～ ¥end のコメントを外す。
C-c .　カーソルのあるパラグラフ全体をコメントアウト
C-c ,　カーソルのあるパラグラフのコメントを外す。

ジャンプ

C-c g　対応する場所にジャンプ。
¥begin{} ⟷ ¥end{}　¥label{} ⟷ ¥ref{}　¥bibitem{} ⟷ ¥cite{}
¥include{} ⟶ 対応するファイル
C-c 4 g　対応する場所にジャンプ。ただし、ジャンプ先を別ウィンドウに表示
M C-a　¥begin へジャンプ
M C-e　¥end へジャンプ
M C-@　¥begin ～ ¥end をマーク

その他

C-c c　変更したいコマンドにカーソルを合わせて C-c c を押す。
C-c k　コマンドの削除
C-c &　tabular, array 環境で現在のカラム数を表示
M RET　tabular, array, itemize, enumerate, tabbing 環境でその環境に応じたエントリーを挿入する。
C-c ?　オンラインヘルプ。デフォルトではカーソル位置のコマンドについて。
C-c /　オンライン apropos。入力したキーワードを説明文に含む項目をすべて表示