MATLAB プログラミング II

数値計算

目次

1　級数

■テイラー展開　$\sin x$ を $x = 0$ においてテイラー (Taylor) 展開すると次のようになる.

$$\sin x = \sum_{n=0}^{N-1} (-1)^n \frac{x^{2n+1}}{(2n+1)!} + R_N \tag{1}$$

だたし，R_N は剰余項を示す.

■例題 1 >　$\sin x$ を $x = 0$ においてテイラー展開して計算するプログラムを，次の手順に従って作成せよ.

1. プログラム作成に必要な式：　効率よく計算するため，式 (1) を次のようにする.

$$S_0 = T_0 = x \tag{2}$$

$$T_i = T_{i-1} x^2 / \{2i(2i+1)\} \tag{3}$$

$$S_i = S_{i-1} + (-1)^i T_i \tag{4}$$

ただし，S_i は項数 i までで展開したときの値を示し，x は実数とする.

2. サブ関数の作成：　上式より関数を作成する．このとき，収束判定基準を設定すること．例えば T_i が 10^{-6} 以下になれば計算を終了する．
3. メインプログラムの作成：　テイラー展開による値を計算し，同時に組込み関数 $\sin$ による計算もできるようにする．x[deg] の値としては，例えば，0° から 20° きざみで 360° まで．
4. 実行（計算）：　正しく計算できていることを確認する．
5. コメント文の確認：　関数の説明，入力データの説明等をコメント文中に記し完成させる．

■プログラム（M2_ex01.m）　サブ関数の例

```
function M2_ex01
clear all

Q = 180.0/pi;
x = 0.0: 1.0/Q: 20.0/Q;
nx = length(x);
fprintf(1,'%4s %8s %12s %12s %15s\n',...
        'i','x[deg]','My_sin(x)','sin(x)','error');
for I=1:nx
    y = My_sin(x(I));
    fprintf(1,'%4d %8.2f %12f %12f %15.10f\n',...
           I,x(I)*Q,y,sin(x(I)),sin(x(I))-y);
end
end

function s = My_sin(x)
% テイラー展開による sin の計算

EPS = 1.0e-6; % 収束判定条件
er = EPS;
m2 = 1;
s = x;
t = s;
```

```
while er>=EPS
    m1 = m2+1;
    m2 = m1+1;
    t = -t*(x*x/m1/m2);
    s = s+t;
    er = abs(t);
end
end
```

■出力結果

```
   i   x[deg]    My_sin(x)       sin(x)           error
   1     0.00     0.000000     0.000000    0.0000000000
   2     1.00     0.017452     0.017452    0.0000000000
   3     2.00     0.034899     0.034899   -0.0000000000
   4     3.00     0.052336     0.052336   -0.0000000000
   5     4.00     0.069756     0.069756   -0.0000000000
   6     5.00     0.087156     0.087156   -0.0000000000
   7     6.00     0.104528     0.104528   -0.0000000000
   8     7.00     0.121869     0.121869   -0.0000000001
   9     8.00     0.139173     0.139173   -0.0000000002
  10     9.00     0.156434     0.156434   -0.0000000005
  11    10.00     0.173648     0.173648    0.0000000000
  12    11.00     0.190809     0.190809    0.0000000000
```

13	12.00	0.207912	0.207912	0.0000000000
14	13.00	0.224951	0.224951	0.0000000000
15	14.00	0.241922	0.241922	0.0000000000
16	15.00	0.258819	0.258819	0.0000000000
17	16.00	0.275637	0.275637	0.0000000000
18	17.00	0.292372	0.292372	0.0000000000
19	18.00	0.309017	0.309017	0.0000000001
20	19.00	0.325568	0.325568	0.0000000001
21	20.00	0.342020	0.342020	0.0000000002

2　数値微分

■通常の数値微分　解析的に導関数を求めることが困難な場合，導関数を数値計算で代用して近似できる場合がある．いま，関数 $y = f(x)$ が与えられているとき，$x = x_i$ についてテイラー (Taylor) 展開すると，

$$f(x) = f(x_i) + f'(x_i)(x - x_i) + f''(x_i)\frac{(x - x_i)^2}{2} + f'''(x_i)\frac{(x - x_i)^3}{3!} + \cdots \tag{5}$$

ここで，$x \equiv x_i + h \equiv x_{i+1}$ とおき，上式に代入すると，

$$f(x_i + h) = f(x_{i+1})$$

$$= f(x_i) + f'(x_i)h + f''(x_i)\frac{h^2}{2} + f'''(x_i)\frac{h^3}{3!} + O(h^4) \tag{6}$$

逆に，$x \equiv x_i - h \equiv x_{i-1}$ とおけば，

$$f(x_i - h) = f(x_{i-1})$$

$$= f(x_i) + f'(x_i)(-h) + f''(x_i)\frac{(-h)^2}{2} + f'''(x_i)\frac{(-h)^3}{3!} + O(h^4) \tag{7}$$

式 (6)− 式 (7) より，$f''(x_i)$ を消去すると，

$$f(x_{i+1}) - f(x_{i-1}) = 2hf'(x_i) + \frac{h^3}{3}f'''(x_i) + O(h^4) \tag{8}$$

これより，$f'(x_i)$ は次のようになる．

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - \frac{h^2}{6}f'''(x_i) + O(h^4) \tag{9}$$

いま，$f_{i+1} = f(x_{i+1})$，$f_{i-1} = f(x_{i-1})$ が計算できれば，上式の h^2 の項および $O(h^4)$ を無視して $f'(x_i)$ は次のように近似できる．

$$f'(x_i) \simeq \frac{f(x_{i+1}) - f(x_{i-1})}{2h} \tag{10}$$

ただし，計算誤差は h^2 のオーダーとなり，これは通常の数値微分の式である．

■リチャードソンの外挿　計算誤差を小さくするため，サンプル点数を増やした方法について考えてみる．まず，式 (9) において，h の替わりに $2h$ とすると，

$$f'(x_i) = \frac{f(x_i + 2h) - f(x_i - 2h)}{2(2h)} - \frac{(2h)^2}{6}f'''(x_i) + O(h^4) \tag{11}$$

ここで，$f_{i+2} = f(x_i + 2h)$，$f_{i-2} = f(x_i - 2h)$ とおくと，

$$f'(x_i) = \frac{f_{i+2} - f_{i-2}}{4h} - \frac{4h^2}{6} f'''(x_i) + O(h^4) \tag{12}$$

式 (9)×4− 式 (12) より，$f'''(x_i)$ を消去すると，

$$4f'(x_i) - f'(x_i) = 4\,\frac{f_{i+1} - f_{i-1}}{2h} - \frac{f_{i+2} - f_{i-2}}{4h} + O(h^4) \tag{13}$$

よって，上式の $O(h^4)$ を無視すると $f'(x_i)$ は次のように近似できる.

$$f'(x_i) \simeq \frac{-f_{i+2} + 8f_{i+1} - 8f_{i-1} + f_{i-2}}{12h} \tag{14}$$

ただし，誤差は h^4 のオーダーとなり，この式はリチャードソンの外挿と呼ばれる.

■例題２＞　数値微分を行うプログラムを次の要領で作成せよ.

1. 関数の作成：　$f(x)$ を関数として作成する.
2. 数値微分の関数の作成：　通常の式，およびリチャードソンの外挿の式を基にして，各々数値微分を行う関数を作成する.
   (a) 値による呼び出し.

(b) 関数ポインタを用いた参照呼出し.

3. メインプログラムの作成： 通常の式，リチャードソンの外挿，および解析的に導関数を求めた式を用いて計算し，出力して比較する.

■プログラム（M2_ex02a.m） 関数は $f(x) = 1 + x + \sin x$.

```
function M2_ex02a

h = 0.1; % 数値微分のきざみ幅
x = 0.0: 0.1: 2.0;
nx = length(x);
fprintf('%8s %15s %15s %15s %15s %15s\n',...
        'x','Diff1b','error1','Diff2b','error2','dydx');
y1 = Diff1a(x, h);
y2 = Diff2a(x, h);
dydx = 1.0+cos(x); % d/dx(Func)
for I=1:nx
    fprintf('%8.3f %15.11f %15.11f %15.11f %15.11f %15.11f\n',...
            x(I),y1(I),y1(I)-dydx(I),y2(I),y2(I)-dydx(I),dydx(I));
end
end

function d1=Diff1a(x, h)
```

```
% 通常の数値微分
% x: 変数の値, h: 差分の大きさ
% d1: 数値微分値
fp = Func(x+h);
fm = Func(x-h);
d1 = (fp-fm)./2.0./h;
end

function d2 = Diff2a(x, h)
% リチャードソンの外挿
% x: 変数の値, h: 差分の大きさ
% d2: 数値微分値
fp = Func(x+h);
fm = Func(x-h);
fpp = Func(x+h+h);
fmm = Func(x-h-h);
d2 = (-fpp+8.0.*fp-8.0.*fm+fmm)./12.0./h;
end

function f=Func(x)
% 関数の例
f = 1.0+x+sin(x);
end
```

■出力結果

x	Diff1a	error1	Diff2a	error2	dydx
0.000	1.99833416647	-0.00166583353	1.99999667063	-0.00000332937	2.00000000000
0.100	1.99334665398	-0.00165751130	1.99500085254	-0.00000331273	1.99500416528
0.200	1.97843395007	-0.00163262777	1.98006331484	-0.00000326300	1.98006657784
0.300	1.95374505757	-0.00159143156	1.95533330846	-0.00000318067	1.95533648913
0.400	1.91952665971	-0.00153433429	1.92105792745	-0.00000306655	1.92106099400
0.500	1.87612065543	-0.00146190646	1.87757964010	-0.00000292179	1.87758256189
0.600	1.82396074317	-0.00137487174	1.82533286706	-0.00000274785	1.82533561491
0.700	1.76356808752	-0.00127409976	1.76483964084	-0.00000254644	1.76484218728
0.800	1.69554611195	-0.00116059740	1.69670438975	-0.00000231959	1.69670670935
0.900	1.62057446954	-0.00103549873	1.62160789870	-0.00000206957	1.62160996827
1.000	1.53940225217	-0.00090005370	1.54030050700	-0.00000179886	1.54030230587
1.100	1.45284050580	-0.00075561563	1.45359461124	-0.00000151019	1.45359612143
1.200	1.36175412678	-0.00060362770	1.36235654805	-0.00000120642	1.36235775448
1.300	1.26705322011	-0.00044560852	1.26749793802	-0.00000089060	1.26749882862
1.400	1.16968400593	-0.00028313697	1.16996657702	-0.00000056588	1.16996714290
1.500	1.07061936527	-0.00011783640	1.07073696616	-0.00000023551	1.07073720167
1.600	0.97084911924	0.00004864154	0.97080057491	0.00000009722	0.97080047770
1.700	0.87137013918	0.00021463348	0.87115593468	0.00000042897	0.87115550570
1.800	0.77317638617	0.00037848087	0.77279866175	0.00000075644	0.77279790531
1.900	0.67724897974	0.00053854660	0.67671150949	0.00000107635	0.67671043314
2.000	0.58454639481	0.00069323135	0.58385454896	0.00000138551	0.58385316345

■プログラム（M2_ex02b.m）　関数ポインタを用いた参照呼出しの例．ただし，関数は

$f(x) = x \sin x,$

```
function M2_ex02c

f2 = @Func2;

h = 0.01; % 数値微分のきざみ幅
x = 0.0: 0.1: 2.0;
nx = length(x);
fprintf('%8s %15s %15s %15s %15s %15s\n',...
        'x','Diff1a','error1','Diff2a','error2','dydx');
y1 = Diff1b(f2, x, h);
y2 = Diff2b(f2, x, h);
dydx = sin(x)+x.*cos(x); % d/dx(Func2)
for I=1:nx
    fprintf('%8.3f %15.11f %15.11f %15.11f %15.11f %15.11f\n',...
            x(I),y1(I),y1(I)-dydx(I),y2(I),y2(I)-dydx(I),dydx(I));
end
end

function d1=Diff1b(func, x, h)
% 通常の数値微分
% func: 関数, x: 変数の値, h: 差分の大きさ
```

```
% d1: 数値微分値
fp = func(x+h);
fm = func(x-h);
d1 = (fp-fm)./2.0./h;
end

function d2 = Diff2b(func, x, h)
% リチャードソンの外挿
% func: 関数, x: 変数の値, h: 差分の大きさ
% d2: 数値微分値
fp = func(x+h);
fm = func(x-h);
fpp = func(x+h+h);
fmm = func(x-h-h);
d2 = (-fpp+8.0.*fp-8.0.*fm+fmm)./12.0./h;
end

function f=Func2(x)
% 関数の例
f = x.*sin(x);
end
```

■出力結果

x	Diff1b	error1	Diff2b	error2	dydx
0.000	0.00000000000	0.00000000000	0.00000000000	0.00000000000	0.00000000000
0.100	0.19932718321	-0.00000664996	0.19933383298	-0.00000000020	0.19933383317
0.200	0.39466944611	-0.00001320026	0.39468264597	-0.00000000040	0.39468264636
0.300	0.58210160085	-0.00001955255	0.58212115281	-0.00000000059	0.58212115340
0.400	0.75781712878	-0.00002561113	0.75784273914	-0.00000000077	0.75784273991
0.500	0.91818553532	-0.00003128423	0.91821681860	-0.00000000095	0.91821681955
0.600	1.05980735714	-0.00003648520	1.05984384123	-0.00000000111	1.05984384234
0.700	1.17956608461	-0.00004113373	1.17960721708	-0.00000000125	1.17960721834
0.800	1.27467630150	-0.00004515688	1.27472145700	-0.00000000138	1.27472145838
0.900	1.34272739095	-0.00004849012	1.34277587958	-0.00000000149	1.34277588107
1.000	1.38172221248	-0.00005107819	1.38177328909	-0.00000000158	1.38177329068
1.100	1.39011021775	-0.00005287588	1.39016309198	-0.00000000165	1.39016309363
1.200	1.36681454265	-0.00005384868	1.36686838964	-0.00000000170	1.36686839134
1.300	1.31125268934	-0.00005397329	1.31130666091	-0.00000000172	1.31130666263
1.400	1.22335049209	-0.00005323796	1.22340372833	-0.00000000172	1.22340373005
1.500	1.10354914635	-0.00005164275	1.10360078741	-0.00000000170	1.10360078911
1.600	0.95280516775	-0.00004919961	0.95285436571	-0.00000000165	0.95285436736
1.700	0.77258323790	-0.00004593225	0.77262916857	-0.00000000158	0.77262917015
1.800	0.56484198448	-0.00004187595	0.56488385894	-0.00000000149	0.56488386043
1.900	0.33201283349	-0.00003707716	0.33204990927	-0.00000000137	0.33204991065
2.000	0.07697216073	-0.00003159300	0.07700375249	-0.00000000124	0.07700375373

■MATLAB の関数の例

- 差分： `diff(Y)`
- 近似微分値： `diff(Y)./diff(X)`

ただし，`X` はサンプル点での変数値（ベクトル），`Y` はサンプル点での関数値（ベクトル）を示す．

3　数値積分

■ニュートン-コーツの積分公式　関数 $f(x)$ を区間 [a,b] で積分する方法として，補間多項式を用いて近似する補間型の数値積分法がある．その中でも，ニュートン-コーツ (Newton-Cotes) の数値積分は，等間隔のサンプル点で計算する方法で，与えられる積分値 $I(f)$ を，次式で近似するものである．

$$I(f) = \int_a^b f(x)dx \simeq K\Delta x_n \sum_{i=0}^{n} A_i f(a + i\Delta x_n) \tag{15}$$

ただし，

$$\Delta x_n = \frac{b-a}{n} \tag{16}$$

ここで，n は補間公式の次数を示し，KA_i は cotes 数と呼ばれる．表 1 は，K および A_i を次数 n が 1 から 6 までについて示したもので，一般に，区間 [a,b] を小区間に分割して，低次の補間公式 [**?**] が用いられる．

表 1　cotes 数の K，A_i の例

n	1	2	3	4	5	6	$\cdots$
K	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{3}{8}$	$\frac{2}{45}$	$\frac{5}{288}$	$\frac{1}{140}$	$\cdots$
A_0	1	1	1	7	19	41	$\cdots$
A_1	1	4	3	32	75	216	$\cdots$
A_2		1	3	12	50	27	$\cdots$
A_3			1	32	50	272	$\cdots$
A_4				7	75	27	$\cdots$
A_5					19	216	$\cdots$
A_6						41	$\cdots$
$\cdots$							$\cdots$

■台形則　次数 1 の補間公式は台形則と呼ばれ，サンプル点 x_i の間隔を h，計算できる関数値を $f_i \equiv f(x_i)$ とすると，次のようになる．

$$\int_{x_i}^{x_{i+1}} f(x)dx = \frac{h}{2}(f_i + f_{i+1}) + O(h^3 f'') \tag{17}$$

ここで，$O(\alpha)$ は真の値と数値計算による値との差を示したもので，このような誤差は正確にはわからないので α（誤差のオーダー）のある係数倍程度の値であることを意味している.

台形則を基にある程度精度よく数値積分するためには，与えられた積分区間 $[a,b]$ を細かく N 分割し，その区間毎に台形則を次のように適用すればよい.

$$
\begin{aligned}
&\int_a^b f(x)dx = \int_{x_0=a}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \cdots + \int_{x_{N-1}}^{x_N=b} f(x)dx \\
&= \frac{h}{2}(f_0+f_1) + \frac{h}{2}(f_1+f_2) + \cdots + \frac{h}{2}(f_{N-1}+f_N) + O(Nh^3 f'') \\
&= \frac{h}{2}\left[f_0 + 2\left(\sum_{i=1}^{N-1} f_i\right) + f_N\right] + O\left(N\frac{(b-a)^3}{N^3}f''\right) \\
&= \frac{h}{2}(f_0+f_N) + \sum_{i=1}^{N-1} f_i + O\left(\frac{(b-a)^3}{N^2}f''\right)
\end{aligned}
\tag{18}
$$

ここで,

$$
h = \frac{b-a}{N} \tag{19}
$$

これを混合型台形則あるいは拡張台形則という．誤差は $1/N^2$ のオーダーであり，例えば分割数 N を 2 倍にすれば，誤差は $1/4$ に減ることがわかる．ニュートン-コーツとその拡張について説明するため，簡単な例である最低次の台形則を取り上げたが，通常の数値積分では後述するさらに精度のよい方法を用いることが多い．

■シンプソンの 1/3 則　次数 2 の補間公式はシンプソン (Simpson) の 1/3 則と呼ばれ，次のようになる．

$$\int_{x_i}^{x_{i+2}} f(x)dx = \frac{h}{3}(f_i + 4f_{i+1} + f_{i+2}) + O(h^5 f^{(4)}) \tag{20}$$

ただし，h はサンプル間隔，$f^{(4)}$ は $f(x)$ の 4 階微分を示し，誤差は $h^5 f^{(4)}$ のオーダーである．精度よく数値積分するためには，与えられた積分区間 $[a, b]$ を細かく N（偶数）分

割し，その 2 区間毎にシンプソンの 1/3 則を次のように適用すればよい.

$$\int_a^b f(x)dx = \int_{x_0=a}^{x_2} f(x)dx + \int_{x_2}^{x_4} f(x)dx + \cdots + \int_{x_{N-2}}^{x_N=b} f(x)dx$$
$$\simeq \frac{h}{3}(f_0 + 4f_1 + f_2) + \frac{h}{3}(f_2 + 4f_3 + f_4) + \cdots + \frac{h}{3}(f_{N-2} + 4f_{N-1} + f_N)$$
$$= \frac{h}{3}\left[f_0 + 4\left(\sum_{i=1}^{N/2} f_{2i-1}\right) + 2\left(\sum_{i=1}^{N/2} f_{2i}\right) + f_N\right] \tag{21}$$

これを混合型シンプソンの 1/3 則あるいは拡張シンプソン則といい，誤差は $1/N^4$ のオーダーである.

■例題３＞　混合型台形則，あるいは混合型シンプソンの 1/3 則による数値積分のプログラムを作成し，$f(x) = \sqrt{4 - x^2}$，$a = 0$，$b = 2$ のときの積分を数値的に行い，収束を調べよ．　（ヒント）定積分より $I(f) = \pi$ である．

1. 被積分関数 $f(x)$ を計算する関数の作成：
2. 数値積分を実行する関数：　次の (a)，(b) の関数を作成せよ．
   (a) 数値成分のサンプル点における被積分関数値を予め計算して 1 次元配列に格納し，この配列を数値積分の関数の引数で参照せよ．
   (b) この関数の仮引数に上の関数 $f(x)$ を参照して作成せよ（関数ポインタを用いた参照呼出し）．
3. メインプログラムの作成

■プログラム（M2_ex03a.m）　１次元配列による参照呼出しの例

```
function M2_ex03a
% テーマ：数値積分
% 作成日：2010/11/06
% アルゴリズム：混合型シンプソンの 1/3 則
% プログラム，関数の基本構成：１次元配列による参照呼出し

a = 0.0; b = 2.0;
fprintf(1,'%4s %15s %15s %15s %15s \n',...
        'n','deikei','error1','simpson','error2');
for I=2:301
    n = I;
    h = (b-a)/(n-1);
    x = a:h:b;
    ff = Func7(x);
    y1 = Daikei1(ff, a, b);
    y2 = Simpson1(ff, a, b);
    fprintf('%4d %15.11f %15.11f %15.11f %15.11f \n',...
            n,y1,y1-pi,y2,y2-pi);
end
fprintf('\n 定積分 y = %15.11f\n',pi); % 定積分による結果

end
```

```
function y = Simpson1(ff, a, b)
% 混合型シンプソンの 1/3 則
% ff: 1 次元配列，被積分関数値，a,b: 積分区間の上限，下限，n: サンプル点数（奇数）
% y: 解
n = length(ff);
h = (b-a)/(n-1);
if n==2 % 台形則
    y = (ff(1)+ff(n)).*h./2.0;
else
    y4 = 0.0; y2 = 0.0;
    if rem(n,2)==1 % 混合型シンプソンの 1/3 則
        for I=2:2:n-1
            y4 = y4+ff(I);
        end
        for I=3:2:n-1
            y2 = y2+ff(I);
        end
        y = ff(1)+4.0.*y4+2.0.*y2+ff(n);
        y = y.*h./3.0;
    else % 混合型シンプソンの 1/3 則 + 3/8 則
        y = ff(1)+3.0.*(ff(2)+ff(3))+ff(4);
        y = y.*h.*3.0/8.0;
        if n>=6
            for I=5:2:n-1
                y4 = y4+ff(I);
            end
```

```
            for I=6:2:n-1
                y2 = y2+ff(I);
            end
            y = y + (ff(4)+4.0.*y4+2.0.*y2+ff(n)).*h./3.0;
        end
    end
end
end

function y = Daikei1(ff, a, b)
% 混合型台形則
% ff: 1次元配列，被積分関数値，a,b: 積分区間の上限，下限，n: サンプル点数
% y: 解
n = length(ff);
h = (b-a)/(n-1);
y = sum(ff);
y = y-(ff(1)+ff(n))*0.5;
y = h*y;
end

function f=Func7(x)
% 関数の例
f = sqrt(4.0-x.^2);
end
```

■出力結果　　途中は省略している.

```
  n          deikei          error1          simpson          error2
  2   2.00000000000  -1.14159265359   2.00000000000  -1.14159265359
  3   2.73205080757  -0.40954184602   2.97606774343  -0.16552491016
  4   2.91755337878  -0.22403927481   3.03224755112  -0.10934510247
  5   2.99570906810  -0.14588358549   3.08359515495  -0.05799749864
  6   3.03704882888  -0.10454382471   3.10001326583  -0.04157938776
  7   3.06198302229  -0.07960963129   3.11012623680  -0.03146641679
  ....................
  （省略）
  ....................

295   3.14135939672  -0.00023325687   3.14150153820  -0.00009111539
296   3.14136058169  -0.00023207190   3.14150200115  -0.00009065244
297   3.14136175665  -0.00023089694   3.14150246019  -0.00009019340
298   3.14136292174  -0.00022973185   3.14150291538  -0.00008973821
299   3.14136407705  -0.00022857653   3.14150336674  -0.00008928685
300   3.14136522272  -0.00022743087   3.14150381434  -0.00008883925
301   3.14136635885  -0.00022629474   3.14150425821  -0.00008839538

定積分 y =    3.14159265359
```

■プログラム（M2_ex03b.m）　関数ポインタを用いた参照呼出しの例

```
function M2_ex03b
% テーマ：数値積分
% 作成日：2010/11/07
% アルゴリズム：混合型シンプソンの 1/3 則
% 関数の基本構成：関数ポインタを用いた参照呼出し

f7 = @Func7;
a = 0.0; b = 2.0;
fprintf(1,'%4s %15s %15s %15s %15s \n',...
        'n','deikei','error1','simpson','error2');
for I=2:301
    n = I;
    y1 = Daikei(f7, a, b, n);
    y2 = Simpson(f7, a, b, n);
    fprintf('%4d %15.11f %15.11f %15.11f %15.11f \n',...
            n,y1,y1-pi,y2,y2-pi);
end
fprintf('\n 定積分 y = %15.11f\n',pi); % 定積分による結果

end

function y = Simpson(Func, a, b, n)
% 混合型シンプソンの 1/3 則
```

```
% Func: 被積分関数, a,b: 積分区間の上限, 下限, n: サンプル点数（奇数）
% y: 解
h = (b-a)/(n-1);
if n==2 % 台形則
    y = (Func(a)+Func(b)).*h./2.0;
else
    x = a:h:b;
    y4 = 0.0; y2 = 0.0;
    if rem(n,2)==1 % 混合型シンプソンの 1/3 則
        for I=2:2:n-1
            y4 = y4+Func(x(I));
        end
        for I=3:2:n-1
            y2 = y2+Func(x(I));
        end
        y = Func(a)+4.0.*y4+2.0.*y2+Func(b);
        y = y.*h./3.0;
    else % 混合型シンプソンの 1/3 則 + 3/8 則
        y = Func(a)+3.0.*(Func(x(2))+Func(x(3)))+Func(x(4));
        y = y.*h.*3.0/8.0;
        if n>=6
            for I=5:2:n-1
                y4 = y4+Func(x(I));
            end
            for I=6:2:n-1
                y2 = y2+Func(x(I));
```

```
            end
            y = y + (Func(x(4))+4.0.*y4+2.0.*y2+Func(b)).*h./3.0;
        end
    end
end
end

function y = Daikei(Func, a, b, n)
% 混合型台形則
% Func: 被積分関数, a,b: 積分区間の上限, 下限, n: サンプル点数
% y: 解
h = (b-a)/(n-1);
x = a:h:b;
y = sum(Func(x));
y = y-(Func(a)+Func(b))*0.5;
y = h*y;
end

function f=Func7(x)
% 関数の例
f = sqrt(4.0-x.^2);
end
```

■シンプソンの 3/8 則　次数 3 の補間公式はシンプソンの 3/8 則と呼ばれ，次のようになる.

$$\int_{x_i}^{x_{i+3}} f(x)dx = \frac{3h}{8}(f_i + 3f_{i+1} + 3f_{i+2} + f_{i+3}) + O(h^5 f^{(4)}) \tag{22}$$

ただし，誤差は $h^5 f^{(4)}$ のオーダーとなって，シンプソンの 1/3 則と同程度である.

■MATLAB の数値積分の関数の例

- 台形則：　`trapz(X,Y)`
- 適応シンプソン則：　`quad(@func,a,b,tol)`

ただし，`X` はサンプル点での変数の値（ベクトル），`Y` は被積分関数値（ベクトル），`func` は被積分関数の関数名，`a`，`b` は積分範囲の下限値，上限値，`tol` は絶対許容誤差（省略すると 10^{-6}）を示す.

■プログラム（M2_ex03c.m）　数値積分の計算例

```
function M2_ex03c
y = quad(@Func7,0,2); % MATLAB の数値積分の関数
fprintf(' 数値積分   y = %15.11f\n',y); % 数値積分による結果
fprintf('   定積分 pi = %15.11f\n',pi); % 定積分による結果
end

function f=Func7(x)
% 関数の例
f = sqrt(4.0-x.^2);
end
```

■出力結果

```
数値積分   y =    3.14158753070
  定積分 pi =    3.14159265359
```

4　非線形代数方程式

■二分法　二分法 (bisection method) は，$f(x) = 0$ の解 x を数値的に囲い込む一つの方法で，解 x の存在範囲 $[a, b]$ を与え，その両端の関数値の符号を見ながら，繰り返し計算によって範囲を逐次，狭めていく単純な方法である（図 1 参照）．ただし，$f(x)$ は実関数，x は実変数に限られるが，導関数 df/dx の計算が不要な点で非常に有用である．計算手順は次のようになる．

1. 初期条件：　a, b を与える．
2. 解の存在について：　$f(a)f(b) < 0$ であることを確認する．$f(a)f(b) > 0$ の場合は a, b の値を修正する．
3. 逐次計算：　区間 $[a, b]$ の中点 x_m を求める．$f(a)f(x_m) < 0$ なら $[a, x_m)$ に解が存在するので $b = x_m$ と更新し，$f(a)f(x_m) > 0$ なら $(x_m, b]$ に解が存在するので $a = x_m$ と更新し，新区間 $[a, b]$ を決める．ただし，$f(a)f(x_m) = 0$ なら，$f(x_m) = 0$ 故，$x = x_m$ が解となる．
4. 収束判定：　$|a - b| < \epsilon$ まで繰り返す．例えば，$\epsilon = 10^{-9}$．
5. 解の判定：　$f(x_m)$ を計算して，$x = x_m$ が $f(x) = 0$ の解かどうかを判定する．

■例題４＞　二分法を基にして，関数 $f(x) = 0$ の解 x を求めるプログラムを，次の要領で作成せよ.

1. 関数 $f(x)$ を計算する関数の作成：この関数の引数引渡しは，値による呼び出し (call by value) により実現する.

図 1　二分法

2. 二分法の関数：　上の関数を仮引数に参照すること（関数ポインタを用いた参照呼出し）.
3. メインプログラムの作成

■プログラム (M2_ex04.m)　　$f(x) = -(x+1.5)x(x-1.5)$, および $f(x) = \frac{\sin x}{x} + \cos x$

```
function M2_ex04
% テーマ：非線形代数方程式
% 作成日：2010/11/08
% アルゴリズム：二分法
% プログラム，関数の基本構成：関数ポインタを用いた参照呼出し
EPS = 1.0e-9;
f4 = @Func4; f5 = @Func5;

% 例1
Calf(f4, 0.0, 0.2, 21);
disp(' 二分法による零点の計算');
xzero = Bisect(f4, EPS, 0.1, 4.0);
fprintf('x = %15.11f\n',xzero);

% 例2
Calf(f5, 0.0, 0.2, 21);
disp(' 二分法による零点の計算');
xzero = Bisect(f5, EPS, 0.1, 4.0);
fprintf('x = %15.11f\n',xzero);

end

function xm = Bisect(Func, eps,x01,x02)
```

```
% 二分法
% Func: 関数ポインタ，x01: 変数の下限，x02: 上限，eps：収束判定条件
% xm: 解
x1 = x01;
x2 = x02;
y = Func(x1);
while abs(x2-x1)>eps
    xm = (x1+x2)/2.0;
    if y*Func(xm)>0.0
        x1 = xm;
    else
        x2 = xm;
    end
end
end

function Calf(Func, x0, dx, nx)
% 与えられた関数の単なる計算，出力
% Func: 関数ポインタ
% x0: 初期値，dx: きざみ，nx: サンプル点数
disp(' 関数値の計算');
fprintf('%8s %15s\n','x','f(x)');
xstop = x0+dx*(nx-1);
x = linspace(x0,xstop,nx);
for I=1:nx
    y = Func(x(I));
```

```
    fprintf('%8.3f %15.11f\n',x(I),y);
end
end

function y = Func4(x)
% 関数の例
y = -(x+1.5).*x.*(x-1.5);
end

function y = Func5(x)
% 関数の例
if x==0.0
    t = 1.0;
else
    t = sin(x)/x;
end
y = t + cos(x);
end
```

■出力結果

```
関数値の計算
       x              f(x)
   0.000   0.00000000000
```

```
   0.200   0.44200000000
   0.400   0.83600000000
   0.600   1.13400000000
   0.800   1.28800000000
   1.000   1.25000000000
   1.200   0.97200000000
   1.400   0.40600000000
   1.600  -0.49600000000
   1.800  -1.78200000000
   2.000  -3.50000000000
   2.200  -5.69800000000
   2.400  -8.42400000000
   2.600 -11.72600000000
   2.800 -15.65200000000
   3.000 -20.25000000000
   3.200 -25.56800000000
   3.400 -31.65400000000
   3.600 -38.55600000000
   3.800 -46.32200000000
   4.000 -55.00000000000
二分法による零点の計算
x =   1.49999999919
関数値の計算
        x             f(x)
   0.000   2.00000000000
   0.200   1.97341323182
```

```
   0.400   1.89460684977
   0.600   1.76640640390
   0.800   1.59340182297
   1.000   1.38177329068
   1.200   1.13905699278
   1.400   0.87385980718
   1.600   0.59553397960
   1.800   0.31382436691
   2.000   0.03850187687
   2.200  -0.22100275188
   2.400  -0.45595072364
   2.600  -0.65861899498
   2.800  -0.82258371561
   3.000  -0.94295249391
   3.200  -1.01653669562
   3.400  -1.04195734023
   3.600  -1.01968076169
   3.800  -0.95198294637
   4.000  -0.84284424469
二分法による零点の計算
x =   2.02875783856
```

■ニュートン-ラフソン法　ニュートン-ラフソン法 (Newton-Raphson method) では，解 x の初期値 x_0 を与え，その点での接線より解の推定値を求め，繰り返し計算によって解の精度を上げていく．関数 $f(x)$ の $x = x_i$ における接線 $y = y(x)$ は，

$$y = f(x_i) + f'(x_i)(x - x_i) \quad (i = 0, 1, 2, \cdots) \tag{23}$$

ただし，導関数 $f'(x) = df/dx$ は解析的に求めておく必要がある．これにより，x 軸との交点 x_{i+1} を求める（図 2 参照）．このような考え方を基に，繰り返し計算を行い，解の精度を上げていくもので，具体的な計算手順は次のようになる．

1. 準備：　$f(x)$ の導関数を解析的に求める．
2. 初期条件：　解の初期値 x_0 を適切な値に決める．
3. 逐次計算：　解の推定値 x_{i+1} を次式により計算する $(i = 0, 1, 2, \cdots)$．

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \tag{24}$$

4. 収束判定：　$|x_i - x_{i+1}| < \epsilon$ まで繰り返す．例えば，$\epsilon = 10^{-8}$．
5. 解の判定：　$f(x_{i+1})$ を計算して，$x = x_{i+1}$ が $f(x) = 0$ の解かどうかを判定する．

図 2　ニュートン-ラフソン法

■例題５＞　ニュートン-ラフソン法により，関数 $f(x) = 0$ の解 x を計算するプログラムを，次の要領で作成せよ.

1. 関数 $f(x)$ および導関数 $f'(x)$ を計算する関数の作成：　この関数の引数引渡しは，参照による呼び出し (call by reference) により実現すること.

2. ニュートン-ラフソン法の関数：　上の関数を仮引数に参照すること（関数ポインタを用いた参照呼出し）.
3. メインプログラムの作成

■プログラム（M2_ex05.m）　$f(x) = x^2 - 4 = (x+2)(x-2)$

```
function M2_ex05
% テーマ：非線形代数方程式
% 作成日：2010/11/08
% アルゴリズム：ニュートン・ラフソン法
% プログラム，関数の基本構成：関数ポインタを用いた参照呼出し

f6 = @Func6; df6 = @Dfunc6;

% 関数の概形
disp('f(x) = x*x-4.0 = (x-2.0)(x+2.0)');
Calf(f6, 0.0, 0.2, 21);

% 計算，出力
disp(' ニュートン・ラフソン法による零点の計算');
x = Newton(f6, df6, 1.0, 50);
fprintf(1,' 解 x = %15.11f\n',x);
```

```
% 零点かどうかのチェック
fprintf(1,'\n f(x) = %15.11f\n',Func6(x));

end

function xzero = Newton(Func, Dfunc, x0, nmax)
% ニュートン・ラフソン法
% Dfunc: 関数ポインタ, Dfunc: 導関数の関数ポインタ,
% x0: 初期値, nmax: 繰り返し回数の上限
% xzero: 解
% icon: 正常終了あるいはエラーの状態を示すコード

EPS = 1.0e-8; % 収束判定条件
icon = -1;
for i=1:nmax
    x = x0-Func(x0)./Dfunc(x0);
    if abs(x-x0)>EPS
        x0 = x;
    else
        icon = 0;
        break
    end
end
% 標準エラーデバイスへの出力
if icon==-1
    fprintf(2, 'Error in newton: f(x) = %f\n',Func(x));
```

```
end
xzero = x;
end

function Calf(Func, x0, dx, nx)
% 与えられた関数の単なる計算，出力
% Func: 関数ポインタ
% x0: 初期値，dx: きざみ，nx: サンプル点数
disp(' 関数値の計算');
fprintf('%8s %15s\n','x','f(x)');
xstop = x0+dx*(nx-1);
x = linspace(x0,xstop,nx);
for I=1:nx
    y = Func(x(I));
    fprintf('%8.3f %15.11f\n',x(I),y);
end
end

function y = Func6(x)
% 関数の例
y = x.*x-4.0;
end

function y = Dfunc6(x)
% 上の関数の導関数
y = 2.0.*x;
```

```
end
```

■出力結果

```
f(x) = x*x-4.0 = (x-2.0)(x+2.0)
関数値の計算
        x              f(x)
    0.000  -4.00000000000
    0.200  -3.96000000000
    0.400  -3.84000000000
    0.600  -3.64000000000
    0.800  -3.36000000000
    1.000  -3.00000000000
    1.200  -2.56000000000
    1.400  -2.04000000000
    1.600  -1.44000000000
    1.800  -0.76000000000
    2.000   0.00000000000
    2.200   0.84000000000
    2.400   1.76000000000
    2.600   2.76000000000
    2.800   3.84000000000
    3.000   5.00000000000
    3.200   6.24000000000
```

```
   3.400    7.56000000000
   3.600    8.96000000000
   3.800   10.44000000000
   4.000   12.00000000000
ニュートン・ラフソン法による零点の計算
解 x =    2.00000000000

 f(x) =    0.00000000000
```

■MATLAB のゼロ点を求める関数の例　$f(x) = 0$ を満たす x を求める関数

- 区間 $[a, b]$ での解：　`x = fzero(@func,[a,b])`
- 初期値 x_0 近傍での解：　`x = fzero(@func,x0)`
- 一般的な形式：　`[x,fval,exitflag] = fzero(@func,x0)`

ただし，`fval` は得られた x での関数値を示す．また，`exitflag` は次のような終了条件を示す．

1: 関数が解 x に収束した．
-1: 出力関数によってアルゴリズムが終了された．

-3: 符号変化を含む区間のサーチ中に関数値 NaN, または Inf が検出された.

-4: 符号変化を含む区間のサーチ中に複素関数が検出された.

-5: fzero が特異点に収束した可能性がある.

■プログラム（M2_ex05b.m）　関数 fzero の使用例

```
function M2_ex05b
% 関数 fzero の使用例
disp('MATLAB 関数 fzero による零点の計算');

[xzero, fval, exitflag] = fzero(@Func4, [0.1, 4.0]); % 例1
fprintf('（例1）x = %15.11f, f(x) = %15.11f, exitflag = %2d\n',...
        xzero,fval,exitflag);

[xzero, fval, exitflag] = fzero(@Func5, [0.1, 4.0]); % 例2
fprintf('（例2）x = %15.11f, f(x) = %15.11f, exitflag = %2d\n',...
        xzero,fval,exitflag);

[xzero, fval, exitflag] = fzero(@Func6, 1.0); % 例3
fprintf('（例3）x = %15.11f, f(x) = %15.11f, exitflag = %2d\n',...
        xzero,fval,exitflag);

end
```

```
function y = Func4(x)
y = -(x+1.5).*x.*(x-1.5);
end

function y = Func5(x)
if x==0.0
    t = 1.0;
else
    t = sin(x)/x;
end
y = t + cos(x);
end

function y = Func6(x)
y = x.*x-4.0;
end
```

■出力結果

```
MATLAB 関数 fzero による零点の計算
(例1) x =   1.50000000000, f(x) =  -0.00000000000, exitflag =  1
(例2) x =   2.02875783811, f(x) =   0.00000000000, exitflag =  1
(例3) x =   2.00000000000, f(x) =   0.00000000000, exitflag =  1
```

5　ベッセル関数

■第 1 種ベッセル関数の級数展開　実数次 ν の第 1 種ベッセル関数 $J_\nu(z)$ の級数展開の公式は次のようになる.

$$J_\nu(z) = \left(\frac{z}{2}\right)^\nu \sum_{m=0}^{\infty} \frac{(-1)^m \left(\frac{z}{2}\right)^{2m}}{m!\,\Gamma(\nu+m+1)} \qquad (z \neq \text{負の実数}) \tag{25}$$

整数次 n の場合，$\nu = n$ として,

$$\Gamma(n+m+1) = (n+m)! \tag{26}$$

より，次のようになる.

$$J_n(z) = \left(\frac{z}{2}\right)^n \sum_{m=0}^{\infty} \frac{(-1)^m \left(\frac{z}{2}\right)^{2m}}{m!\,(n+m)!} \qquad (z \neq \text{負の実数}) \tag{27}$$

■プログラム作成に必要な式について　級数展開された式は，次のように変形できる.

$$J_n(x) = \left(\frac{x}{2}\right)^n \frac{1}{n!} \sum_{m=0}^{\infty} \frac{\left\{-\left(\frac{x}{2}\right)^2\right\}^m}{m!\,(n+m)(n+m-1)\cdots\quad\cdot(n+1)} \tag{28}$$

いま,

$$a_i \equiv \left(\frac{x}{2}\right)^2 \frac{-1}{i(n+i)} \tag{29}$$

とおくと，次のようになる.

$$J_n(x) = \left(\frac{x}{2}\right)^n \frac{1}{n!}[1 + a_1\{1 + a_2(1 + a_3(1 + \cdots \quad \cdots))\}] \tag{30}$$

これより，効率よくプログラムで計算するため，次のような式を用いて $J_n(x)$ の近似値 S_i を求める.

$$S_0 = T_0 = \left(\frac{x}{2}\right)^n \frac{1}{n!} \tag{31}$$

$$T_i = T_{i-1}\left(\frac{x}{2}\right)^2 \frac{-1}{i(n+i)} \tag{32}$$

$$S_i = S_{i-1} + T_i \tag{33}$$

■第 1 種ベッセル関数の漸近展開　整数次 n の第 1 種ベッセル関数 $J_n(x)$ の漸近展開の公式は次のようになる [**?**].

$$J_n(x) = \sqrt{\frac{2}{\pi x}} \left\{ A_n(x) \cos\left(x - \frac{2n+1}{4}\pi \right) - B_n(x) \sin\left(x - \frac{2n+1}{4}\pi \right) \right\} \tag{34}$$

ここで,

$$A_n(x) \sim \sum_{r=0}^{\infty} (-1)^r \frac{(4n^2-1^2)(4n^2-3^2)\cdots \quad (4n^2-(4r-1)^2)}{(2r)!\,(8x)^{2r}} \tag{35}$$

$$B_n(x) \sim \sum_{r=0}^{\infty} (-1)^r \frac{(4n^2-1^2)(4n^2-3^2)\cdots \quad (4n^2-(4r+1)^2)}{(2r+1)!\,(8x)^{2r+1}} \tag{36}$$

■第 1 種ベッセル関数の漸近展開の補足　式 (35) の $A_n(x)$ および式 (36) の $B_n(x)$ は, 低次の項から書いていくと次のようになる.

$$\begin{aligned} A_n(x) &\sim 1 + (-1)^1 \frac{(4n^2-1^2)(4n^2-3^2)}{2!\cdot(8x)^2} \\ &\qquad + (-1)^2 \frac{(4n^2-1^2)(4n^2-3^2)(4n^2-5^2)(4n^2-7^2)}{4!\cdot(8x)^4} + \cdots \\ &\equiv A_{n,0} + A_{n,1} + A_{n,2} + \cdots \end{aligned} \tag{37}$$

$$B_n(x) \sim \frac{4n^2-1^2}{8x} + (-1)^1 \frac{(4n^2-1^2)(4n^2-3^2)(4n^2-5^2)}{3!\cdot(8x)^3}$$
$$+(-1)^2 \frac{(4n^2-1^2)(4n^2-3^2)(4n^2-5^2)(4n^2-7^2)(4n^2-9^2)}{5!\cdot(8x)^5} + \cdots$$
$$\equiv B_{n,0} + B_{n,1} + B_{n,2} + \cdots \tag{38}$$

これより，低次の項から順次計算していく場合，次のように式を変形すればよい.

$$A_n(x) \sim A_{n,0} - \frac{4n^2-3^2}{2\cdot 8x} B_{n,0} - \frac{4n^2-7^2}{4\cdot 8x} B_{n,1} + \cdots \tag{39}$$

$$B_n(x) \sim B_{n,0} + \frac{4n^2-5^2}{3\cdot 8x} A_{n,1} + \frac{4n^2-9^2}{5\cdot 8x} A_{n,2} + \cdots \tag{40}$$

つまり,

$$A_n(x) \sim \sum_{r=0}^{\infty} A_{n,r}, \qquad B_n(x) \sim \sum_{r=0}^{\infty} B_{n,r} \tag{41}$$

$$A_{n,0} = 1 \tag{42}$$

$$B_{n,0} = \frac{4n^2-1}{8x} \tag{43}$$

$$A_{n,r} = -\frac{4n^2-(4r-1)^2}{2r\cdot 8x}B_{n,r-1} \quad (r=1,2,\cdots) \tag{44}$$

$$B_{n,r} = \frac{4n^2-(4r+1)^2}{(2r+1)\cdot 8x}A_{n,r} \quad (r=1,2,\cdots) \tag{45}$$

■例題６＞　整数次 n の第１種ベッセル関数 $J_n(x)$ を計算するプログラムを作成せよ．ただし，x は負でない実数とする．

■プログラム（M2_ex06.m）

```
function M2_ex06
% テーマ：特殊関数
% 作成日：2010/11/08
X = 0.0:0.5:30;
nx = length(X); % サンプル点数
fprintf('%8s %4s %15s %4s %15s %15s\n',...
        'X','m1','Bessel','m2','Bessel2','besselj');
Y3 = besselj(0,X); % MATLAB の関数
for I=1:nx
    [y1, m1] = Bessel(X(I), 0); % 第 1 種ベッセル関数の級数展開
    [y2, m2] = Bessel2(X(I), 0); % 第 1 種ベッセル関数の級数展開と漸近展開
    fprintf('%8.3f %4d %15.11f %4d %15.11f %15.11f\n',...
            X(I),m1,y1,m2,y2,Y3(I));
end
end

function [y, m] = Bessel(x, n)
% 第 1 種ベッセル関数の級数展開
% x: 変数（実数）, n: 次数（整数）
% y：ベッセル関数の値（実数）, m：展開項数（整数）
EPS = 1.0e-12; % 収束判定条件
a=1.0;
z = x/2.0;
```

```
for I=1:n
    a = a*z/I;
end
y = a;
I = 0;
while abs(a) >= EPS
    I = I+1;
    a = -a*z*z/I/(I+n);
    y = y+a;
end
m = I;
end

function [y,m] = Bessel2(x, n)
% 第１種ベッセル関数の級数展開と漸近展開
% x: 変数（実数）, n: 次数（整数）
% y：ベッセル関数の値（実数）, m：展開項数（整数）
EPS = 1.0e-12; % 収束判定条件
a=1.0;
if abs(x)<14.0 % 級数展開
    z = x/2.0;
    for I=1:n
        a = a*z/I;
    end
    y = a;
    I = 0;
```

```
    while abs(a) >= EPS
        I = I+1;
        a = -a*z*z/I/(I+n);
        y = y+a;
    end
    m = I;
else % 漸近展開
    za = 1.0;
    zb = (4.0*n*n-1.0)/8.0/x;
    an = za;
    bn = zb;
    I = 0;
    I2 = 1;
    while abs(zb)>=EPS
        I = I+1;
        I1 = I2+1;
        I2 = I1+1;
        za = -zb*(4.0*n*n-(2.0*I1-1.0)*(2.0*I1-1.0))/I1/8.0/x;
        zb =  za*(4.0*n*n-(2.0*I2-1.0)*(2.0*I2-1.0))/I2/8.0/x;
        an = an+za;
        bn = bn+zb;
    end
    m = I;
    th = x-(2.0*n+1.0)*pi/4.0;
    y = sqrt(2.0/pi/x)*(an*cos(th)-bn*sin(th));
end
```

```
end
```

■出力結果

```
     X    m1           Bessel    m2          Bessel2          besselj
 0.000     1   1.00000000000     1   1.00000000000   1.00000000000
 0.500     6   0.93846980724     6   0.93846980724   0.93846980724
 1.000     8   0.76519768656     8   0.76519768656   0.76519768656
 1.500     9   0.51182767174     9   0.51182767174   0.51182767174
 2.000    10   0.22389077914    10   0.22389077914   0.22389077914
 2.500    11  -0.04838377647    11  -0.04838377647  -0.04838377647
 3.000    12  -0.26005195490    12  -0.26005195490  -0.26005195490
 3.500    13  -0.38012773999    13  -0.38012773999  -0.38012773999
 4.000    14  -0.39714980986    14  -0.39714980986  -0.39714980986
 4.500    14  -0.32054250899    14  -0.32054250899  -0.32054250899
 5.000    15  -0.17759677131    15  -0.17759677131  -0.17759677131
 5.500    16  -0.00684386942    16  -0.00684386942  -0.00684386942
 6.000    17   0.15064525725    17   0.15064525725   0.15064525725
 6.500    18   0.26009460558    18   0.26009460558   0.26009460558
 7.000    18   0.30007927052    18   0.30007927052   0.30007927052
 7.500    19   0.26633965788    19   0.26633965788   0.26633965788
 8.000    20   0.17165080714    20   0.17165080714   0.17165080714
 8.500    21   0.04193925184    21   0.04193925184   0.04193925184
 9.000    22  -0.09033361118    22  -0.09033361118  -0.09033361118
```

9.500	22	-0.19392874769	22	-0.19392874769	-0.19392874769
10.000	23	-0.24593576445	23	-0.24593576445	-0.24593576445
10.500	24	-0.23664819446	24	-0.23664819446	-0.23664819446
11.000	24	-0.17119030041	24	-0.17119030041	-0.17119030041
11.500	25	-0.06765394811	25	-0.06765394811	-0.06765394811
12.000	26	0.04768931080	26	0.04768931080	0.04768931080
12.500	27	0.14688405470	27	0.14688405470	0.14688405470
13.000	27	0.20692610238	27	0.20692610238	0.20692610238
13.500	28	0.21498916588	28	0.21498916588	0.21498916588
14.000	29	0.17107347611	9	0.17107347611	0.17107347611
14.500	30	0.08754486801	8	0.08754486801	0.08754486801
15.000	30	-0.01422447282	8	-0.01422447283	-0.01422447283
15.500	31	-0.10923065089	8	-0.10923065090	-0.10923065090
16.000	32	-0.17489907397	7	-0.17489907398	-0.17489907398
16.500	32	-0.19638069292	7	-0.19638069294	-0.19638069294
17.000	33	-0.16985425216	7	-0.16985425215	-0.16985425215
17.500	34	-0.10311039819	7	-0.10311039823	-0.10311039823
18.000	34	-0.01335580590	6	-0.01335580572	-0.01335580572
18.500	35	0.07716482125	6	0.07716482142	0.07716482142
19.000	36	0.14662943987	6	0.14662943966	0.14662943966
19.500	37	0.17885382675	6	0.17885382704	0.17885382704
20.000	37	0.16702466460	6	0.16702466434	0.16702466434
20.500	38	0.11509696086	6	0.11509696025	0.11509696025
21.000	39	0.03657906881	6	0.03657907100	0.03657907100
21.500	39	-0.04894204537	6	-0.04894204372	-0.04894204372
22.000	40	-0.12065146429	5	-0.12065147570	-0.12065147570

```
22.500   41  -0.16154031777    5  -0.16154031703  -0.16154031703
23.000   41  -0.16241277065    5  -0.16241278131  -0.16241278131
23.500   42  -0.12392824532    5  -0.12392823156  -0.12392823156
24.000   43  -0.05623024426    5  -0.05623027417  -0.05623027417
24.500   44   0.02369753944    5   0.02369743373   0.02369743373
25.000   44   0.09626663715    5   0.09626678328   0.09626678328
25.500   45   0.14406207320    5   0.14406215755   0.14406215755
26.000   46   0.15599937257    5   0.15599931552   0.15599931552
26.500   46   0.12987740161    5   0.12987762611   0.12987762611
27.000   47   0.07274168414    5   0.07274191801   0.07274191801
27.500   48  -0.00099360547    5  -0.00099222891  -0.00099222891
28.000   48  -0.07315711454    5  -0.07315701055  -0.07315701055
28.500   49  -0.12628967602    5  -0.12629113138  -0.12629113138
29.000   50  -0.14784115220    5  -0.14784876468  -0.14784876468
29.500   50  -0.13314417520    5  -0.13314785830  -0.13314785830
30.000   51  -0.08636246307    5  -0.08636798358  -0.08636798358
```

■MATLAB のベッセル関数の例

- 第一種ベッセル関数：　`besselj(nu,Z)`
- 第二種ベッセル関数：　`bessely(nu,Z)`
- 第一種変形ベッセル関数：　`besseli(nu,Z)`

- 第二種変形ベッセル関数： `besselk(nu,Z)`

ただし，`nu` は次数（実数），`Z` は変数（実数，あるいは実部が正でない複素数）を示す．

■スクリプト（M2_ex06b.m）　第一種ベッセル関数の計算例

```
% 第一種ベッセル関数の計算
X = 0.0:0.2:12;
Y0 = besselj(0,X);
Y1 = besselj(1,X);
Y2 = besselj(2,X);
Y3 = besselj(3,X);
plot(X,Y0,X,Y1,X,Y2,X,Y3); grid on;
axis([-inf inf -0.6 1.0]);
xlabel('z'); ylabel('J_n (z)')
title('function besselj')
legend('J_0(z)','J_1(z)','J_2(z)','J_3(z)',4)
set(gca,'ytick',-0.6:0.2:1.0); set(gca,'xtick',0:1:12)
print -depsc -tiff M2_ex06b_1
```

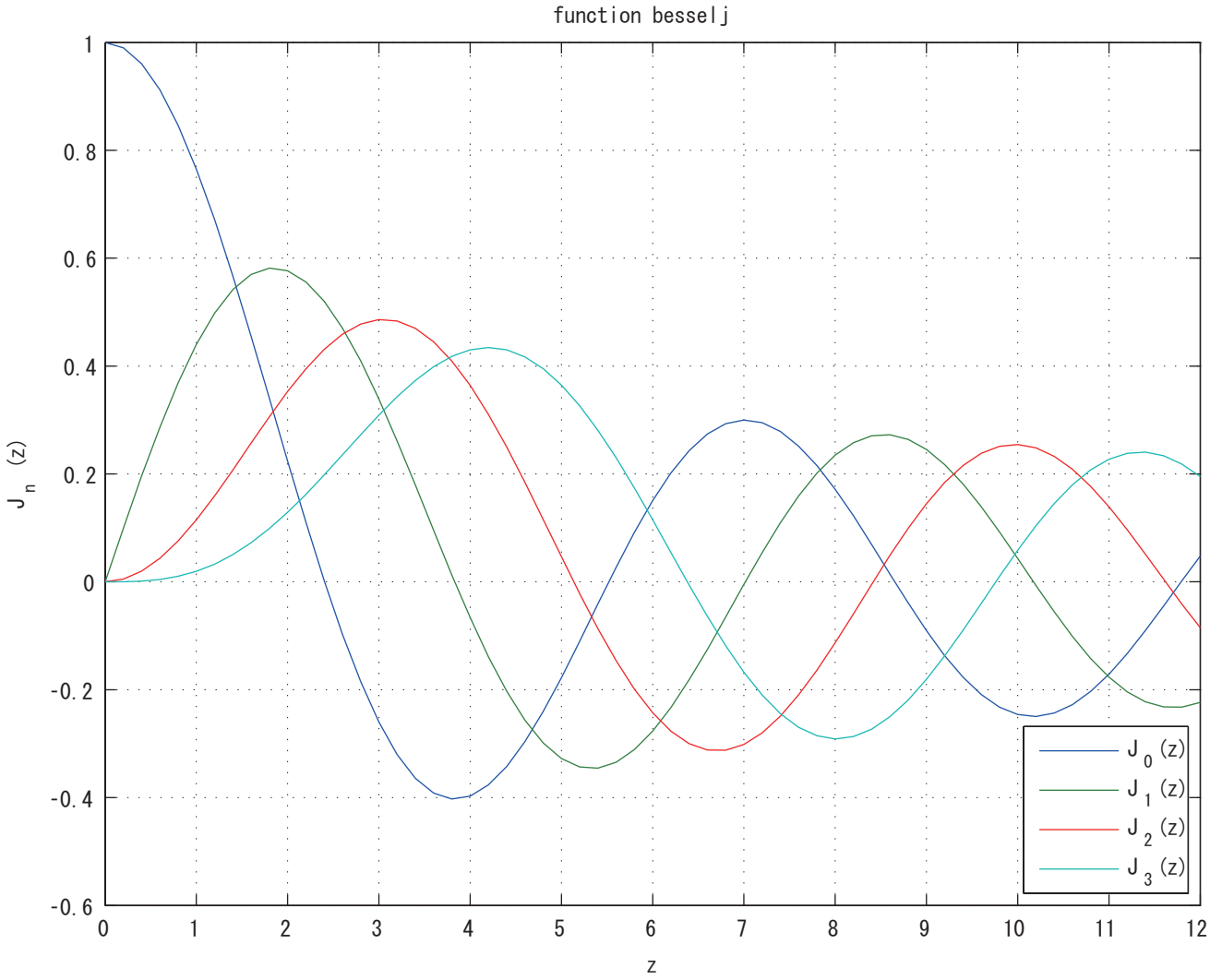


図 3　出力結果（M2_ex06b_1.eps）

■スクリプト（M2_ex06c.m）　第二種ベッセル関数の計算例

```
% 第二種ベッセル関数の計算
X = 0.0:0.2:12;
Y0 = bessely(0,X);
Y1 = bessely(1,X);
Y2 = bessely(2,X);
Y3 = bessely(3,X);
Y4 = bessely(4,X);
plot(X,Y0,X,Y1,X,Y2,X,Y3,X,Y4); grid on;
axis([-inf inf -1.0 0.6]);
xlabel('z'); ylabel('N_n (z)')
title('function bessely')
legend('N_0(z)','N_1(z)','N_2(z)','N_3(z)','N_4(z)',5)
set(gca,'ytick',-1.0:0.1:0.6); set(gca,'xtick',0:1:12)
print -depsc -tiff M2_ex06b_2
```

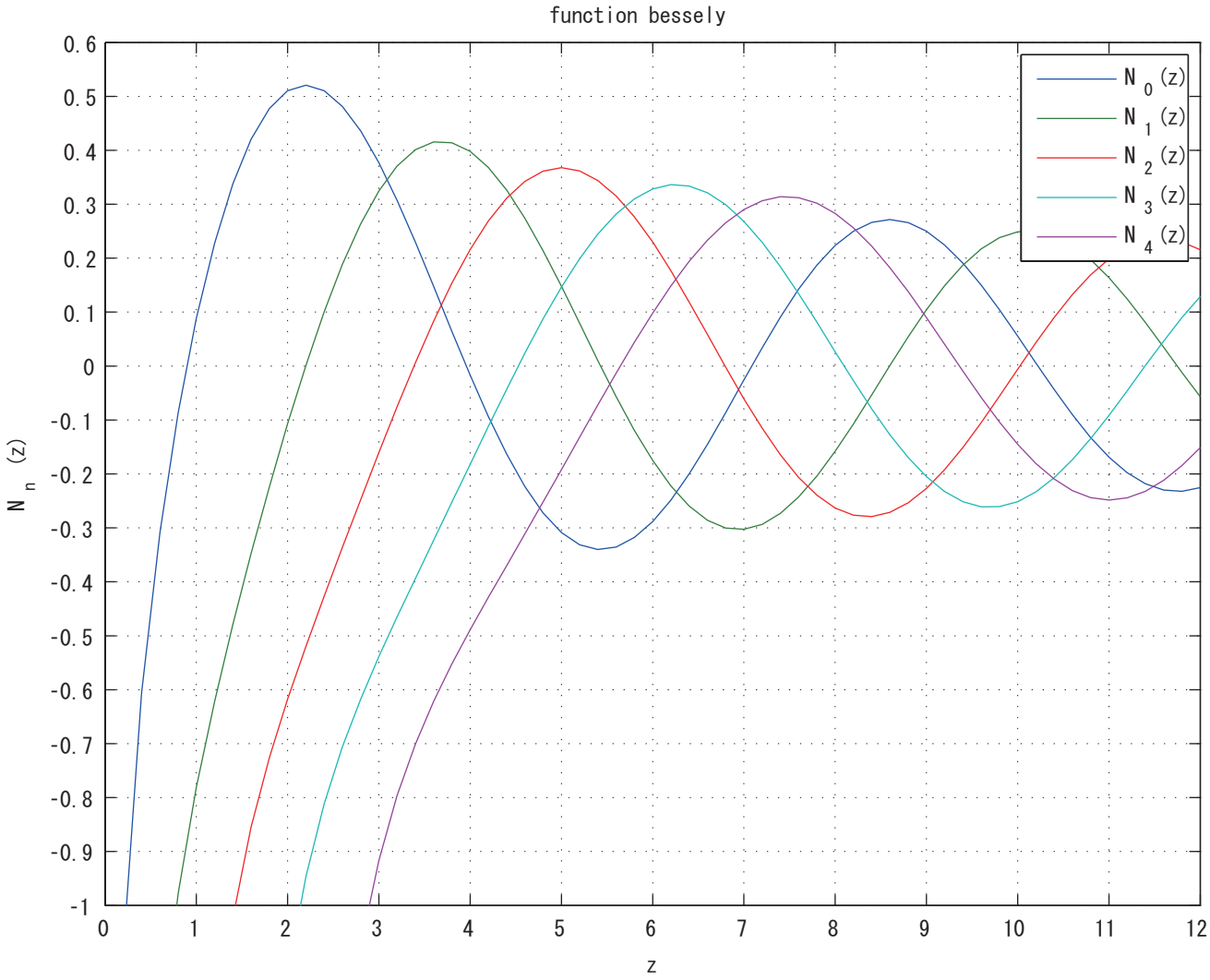


図 4　出力結果（M2_ex06b_2.eps）

■ベッセル関数を含む積分公式　ベッセル関数 $J_\nu(\alpha z)$，$J_\nu(\beta z)$ を満たす微分方程式は $\alpha \neq \beta$ のとき次式で与えられる.

$$zu'' + u' + \left(\alpha^2 - \frac{\nu^2}{z^2}\right) zu = 0 \tag{46}$$

$$zv'' + v' + \left(\beta^2 - \frac{\nu^2}{z^2}\right) zv = 0 \tag{47}$$

ただし，u'，v' は各々 z に関する 1 階微分，u''，v'' は各々 z に関する 2 階微分を示す.
いま，式 (46) $\times v -$ 式 (47) $\times u$ を計算すると次のようになる.

$$\left\{zu'' + u' + \left(\alpha^2 - \frac{\nu^2}{z^2}\right) zu\right\} v - \left\{zv'' + v' + \left(\beta^2 - \frac{\nu^2}{z^2}\right) zv\right\} u = 0$$

$$z\left(u''v - v''u\right) + \left(u'v - v'u\right) = \left(\beta^2 - \alpha^2\right) zuv$$

$$\frac{d}{dz}\left\{z(u'v - v'u)\right\} = \left(\beta^2 - \alpha^2\right) zuv \tag{48}$$

両辺を z で積分すると,

$$z(u'v - v'u) = \left(\beta^2 - \alpha^2\right) \int zuv\, dz \tag{49}$$

ここで，$u = J_\nu(\alpha z)$，$v = J_\nu(\beta z)$ より，

$$u' = \frac{du}{dz} = \alpha J'_\nu(\alpha z), \qquad v' = \frac{dv}{dz} = \beta J'_\nu(\beta z) \tag{50}$$

ただし，$J'_\nu(\alpha z)$，$J'_\nu(\beta z)$ は，各々 $J_\nu(\alpha z)$，$J_\nu(\beta z)$ の αz，βz に関する微分を示す．これより，積分の式は次のようになる．

$$z\{\alpha J'_\nu(\alpha z)J_\nu(\beta z) - \beta J_\nu(\alpha z)J'_\nu(\beta z)\} = \left(\beta^2 - \alpha^2\right)\int zJ_\nu(\alpha z)J_\nu(\beta z)\,dz$$

これは，次のようなよく知られた不定積分公式である．

$$\begin{aligned}&\int zJ_\nu(\alpha z)J_\nu(\beta z)dz\\ &\qquad = \frac{z}{\alpha^2 - \beta^2}\{\beta J_\nu(\alpha z)J'_\nu(\beta z) - \alpha J'_\nu(\alpha z)J_\nu(\beta z)\} \qquad (\alpha \neq \beta)\end{aligned} \tag{51}$$

上式は第 1 種ベッセル関数 $J_\nu(\alpha z)$，$J_\nu(\beta z)$ だけでなく，第 2 種ベッセル関数 $Y_\nu(\alpha z)$，$Y_\nu(\beta z)$，第 1 種ハンケル関数 $H_\nu^{(1)}(\alpha z)$，$H_\nu^{(1)}(\beta z)$，第 2 種ハンケル関数 $H_\nu^{(2)}(\alpha z)$，$H_\nu^{(2)}(\beta z)$ においても成り立つことは言うまでもない．

また，$\alpha = \beta$ のときは，式 (46)$\times 2zu'$ を計算して，次のようになる.

$$\left\{ zu'' + u' + \left(\alpha^2 - \frac{\nu^2}{z^2} \right) zu \right\} 2zu' = 0$$

$$2zu' \left(zu'' + u' \right) + 2 \left(\alpha^2 z^2 - \nu^2 \right) uu' = 0$$

$$2zu' \frac{d}{dz}(zu') + 2 \left(\alpha^2 z^2 - \nu^2 \right) uu' = 0$$

$$\frac{d}{dz} \left\{ (zu')^2 \right\} + \frac{d}{dz} \left\{ \left(\alpha^2 z^2 - \nu^2 \right) u^2 \right\} - 2\alpha^2 zu^2 = 0$$

$$\frac{d}{dz} \left\{ (zu')^2 + \left(\alpha^2 z^2 - \nu^2 \right) u^2 \right\} = 2\alpha^2 zu^2 \tag{52}$$

両辺を z で積分して,

$$(zu')^2 + \left(\alpha^2 z^2 - \nu^2 \right) u^2 = 2\alpha^2 \int zu^2 dz$$

$$2\alpha^2 \int z J_\nu^2(\alpha z) dz = z^2 \alpha^2 J_\nu'^2(\alpha z) + \left(\alpha^2 z^2 - \nu^2 \right) J_\nu^2(\alpha z) \tag{53}$$

これより，次のよく知られた積分公式が得られる.

$$\int z J_\nu^2(\alpha z) dz = \frac{1}{2} \left\{ z^2 J_\nu'^2(\alpha z) + \left(z^2 - \frac{\nu^2}{\alpha^2} \right) J_\nu^2(\alpha z) \right\} \tag{54}$$

■変形ベッセル関数を含む積分公式　変形ベッセル関数についても積分公式が得られるが，こちらはあまり知られていないので，同様であるが求めておこう．変形ベッセル関数 $I_\nu(\alpha z)$，$I_\nu(\beta z)$ を満たす微分方程式は次式で与えられる．

$$zu'' + u' + \left(-\alpha^2 - \frac{\nu^2}{z^2}\right) zu = 0 \tag{55}$$

$$zv'' + v' + \left(-\beta^2 - \frac{\nu^2}{z^2}\right) zv = 0 \tag{56}$$

式 (55) $\times v-$ 式 (56) $\times u$ より

$$\frac{d}{dz}\{z(u'v - v'u)\} = \left(\alpha^2 - \beta^2\right) zuv \tag{57}$$

が得られ，両辺を z で積分すると次のようになる．

$$z(u'v - v'u) = \left(\alpha^2 - \beta^2\right) \int zuv\, dz \tag{58}$$

ここで，$u = I_\nu(\alpha z)$，$v = I_\nu(\beta z)$ より，

$$u' = \frac{du}{dz} = \alpha I'_\nu(\alpha z), \qquad v' = \frac{dv}{dz} = \beta I'_\nu(\beta z) \tag{59}$$

となり，次式が得られる.

$$\int zI_\nu(\alpha z)I_\nu(\beta z)dz$$
$$= \frac{z}{\alpha^2-\beta^2}\{\alpha I'_\nu(\alpha z)I_\nu(\beta z) - \beta I_\nu(\alpha z)I'_\nu(\beta z)\} \quad (\alpha \neq \beta) \tag{60}$$

また，$\alpha = \beta$ のときは，式 (55)$\times 2zu'$ より，

$$\frac{d}{dz}\left\{(zu')^2 + \left(-\alpha^2 z^2 - \nu^2\right)u^2\right\} = -2\alpha^2 zu^2 \tag{61}$$

両辺を z で積分して，

$$\int zI_\nu^2(\alpha z)dz = \frac{1}{2}\left\{-z^2 I'^2_\nu(\alpha z) + \left(z^2 + \frac{\nu^2}{\alpha^2}\right)I_\nu^2(\alpha z)\right\} \tag{62}$$

■ベッセル関数と変形ベッセル関数を含む積分公式　ベッセル関数 $J_\nu(\alpha z)$ と変形ベッセル関数 $I_\nu(\beta z)$ との組合せを含む積分の場合，式 (46) $\times v-$ 式 (56) $\times u$ より

$$\frac{d}{dz}\{z(u'v - v'u)\} = -\left(\alpha^2 + \beta^2\right) zuv \tag{63}$$

が得られ，両辺を z で積分すると次のようになる.

$$z(u'v - v'u) = -\left(\alpha^2 + \beta^2\right) \int zuv\, dz \tag{64}$$

ここで，$u = J_\nu(\alpha z)$，$v = I_\nu(\beta z)$ より，次式が得られる.

$$\int zJ_\nu(\alpha z)I_\nu(\beta z)\, dz = \frac{z}{\alpha^2 + \beta^2}\{\beta J_\nu(\alpha z)I'_\nu(\beta z) - \alpha J'_\nu(\alpha z)I_\nu(\beta z)\} \tag{65}$$

■問題　上で求めた不定積分 I_{JJ}，I_{II}，I_{JI} を用い，積分範囲を与えて計算するプログラムを作成せよ．また，数値積分でも計算し，比較せよ.

6　常微分方程式

次の常微分方程式

$$y' = \frac{dy}{dx} = f(x, y) \tag{66}$$

が与えられたとき，初期条件 $y_0 = y(x_0)$ のもとで，数値的に $y = y(x)$ を計算することを考える．いま，両辺を x について微少区間 $[x_i, x_{i+1}]$ で積分すると，

$$\int_{x_i}^{x_{i+1}} \frac{dy}{dx} dx = \int_{x_i}^{x_{i+1}} f(x, y) dx$$

$$y(x_{i+1}) - y(x_i) = \int_{x_i}^{x_{i+1}} f(x, y) dx$$

よって，$y_i = y(x_i)$ とおくと，$x = x_{i+1}$ のときの y_{i+1} は初期条件をもとに次式で計算できることになる．

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} f(x, y) dx \qquad (i = 0, 1, 2, \cdots) \tag{67}$$

計算精度は，上式の積分項をいかに精度よく計算できるかにかかっている．

■オイラー (Euler) 法　最も簡単な方法は，積分項を $(x_{i+1}-x_i)f(x_i,y_i)$ で近似するもので，オイラー法という．いま，微少区間を等間隔 h にとると，y_{i+1} は与えられた初期値 x_0，y_0 から次式により繰り返し計算で求めることができる．

$$y_{i+1}=y_i+hf(x_i,y_i) \qquad (i=0,1,2,\cdots) \tag{68}$$

ただし，打切り誤差は h^2 のオーダーになる．

■修正オイラー法　積分項を台形公式で近似すると，次のようになる．

$$y_{i+1}=y_i+\frac{h}{2}\{f(x_i,y_i)+f(x_{i+1},y_{i+1})\} \qquad (i=0,1,2,\cdots) \tag{69}$$

上式の右辺をみると，未知数 y_{i+1} があるので，このままでは計算できない．そこで，右辺の y_{i+1} についてオイラー法の式を用いて求めたものを修正オイラー法と呼び，次のようになる．

$$y_{i+1}=y_i+\frac{h}{2}\{f(x_i,y_i)+f(x_{i+1},y_i+hf(x_i,y_i))\} \tag{70}$$

いま,

$$k_1 = hf(x_i, y_i) \tag{71}$$

$$k_2 = hf(x_{i+1}, y_i + k_1) = hf(x_i + h, y_i + k_1) \tag{72}$$

とおくと,

$$y_{i+1} = y_i + \frac{1}{2}(k_1 + k_2) \qquad (i = 0, 1, 2, \cdots) \tag{73}$$

ただし，打切り誤差は h^3 のオーダーになる．この方法は，改良オイラー法あるいは２次のルンゲ-クッタ法ともいう.

■４次のルンゲ-クッタ (Runge-Kutta) 法　さらに精度のよいものとして，４次のルンゲ-クッタ法があり，微少区間の中点での値を巧みに使ったもので，次式で与えられる.

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \qquad (i = 0, 1, 2, \cdots) \tag{74}$$

ここで,

$$k_1 = hf(x_i, y_i) \tag{75}$$

$$k_2 = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right) \tag{76}$$

$$k_3 = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}\right) \tag{77}$$

$$k_4 = hf(x_i + h, y_i + k_3) \tag{78}$$

ただし，打切り誤差は h^4 のオーダーになる.

■MATLAB の常微分方程式（数値計算）の関数の例

- 2 次のルンゲ-クッタ： `[x,y]=ode23(@odefun,xspan,y0)`
- 4 次のルンゲ-クッタ： `[x,y]=ode45(@odefun,xspan,y0)`

ただし，`@odefun` は $y' = f(x, y)$ の関数ポインタ，`xspan` は変数 x の範囲，`y0` は y の初期値を示す．また，`x` は x の計算値，`y` は y の計算値を示す.

■例題 7 ＞　4 次のルンゲ-クッタ法によるプログラムを作成せよ．また，MATLAB の関数を用いた計算も行い，比較せよ.

計算条件）$y' = -4(x-1)y$, 初期条件：$x_0 = 0.0$, $y_0 = e^{-2.0}$

■スクリプト（M2_ex07.m）　ルンゲ-クッタ法による常微分方程式の数値計算例

```
function M2_ex07
% テーマ：常微分方程式
% 作成日：2010/12/14
% アルゴリズム：ルンゲ・クッタ法
n = 1000;
x0 = 0.0; y0 = exp(-2.0); % 初期条件
xe = 2.0; % x の上限値
m = 11;
xs = linspace(x0, xe, m);

fprintf(1,'n =%4d\n',n);
fprintf(1,'%6s %10s %10s %10s %10s %10s %10s %10s \n',...
    'x','Rk4','error(Rk4)','ode23','error','ode45','error','y=y(x)');

[xx, y23] = ode23(@Dfunc7, xs, y0);
[xx, y45] = ode45(@Dfunc7, xs, y0);
for I=1:m
    y = Func7(xs(I));
%    Y1 = Euler(@Dfunc7, x0, y0, xs(I), n);
%    Y2 = Rk2(@Dfunc7, x0, y0, xs(I), n);
    Y3 = Rk4(@Dfunc7, x0, y0, xs(I), n);
```

```
    fprintf(1,'%6.2f %10.6f %10.6f %10.6f %10.6f %10.6f %10.6f %10.6f\n',...
        xs(I),Y3,Y3-y,y23(I),y23(I)-y,y45(I),y45(I)-y,y);
end
end

function y = Euler(Dfunc, x0, y0, xs, n)
% オイラー法
% Dfunc: 関数ポインタ, y'=dy/dx=f(x,y)
% x0, y0: 初期条件, y0 = y'(x0)
% xs: y を求める x の値, n: 区間数
% y：数値解, y = y(xs)

h = (xs-x0)./n;
y = y0;
for I=0:n
    x = x0+h.*I;
    y = y+h.*Dfunc(x,y);
end
end

function y = Rk2(Dfunc, x0, y0, xs, n)
% 修正オイラー法（2次のルンゲ・クッタ法)
% Dfunc: 関数ポインタ, y'=dy/dx=f(x,y)
% x0, y0: 初期条件, y0 = y'(x0)
% xs: y を求める x の値, n: 区間数
% y：数値解, y = y(xs)
```

```
h = (xs-x0)./n;
y = y0;
for I=0:n
    x = x0+h.*I;
    k1 = h.*Dfunc(x,y);
    k2 = h.*Dfunc(x+h,y+k1);
    y = y+(k1+k2).*0.5;
end
end

function y = Rk4(Dfunc, x0, y0, xs, n)
% ４次のルンゲ・クッタ法
% Dfunc: 関数ポインタ, y'=dy/dx=f(x,y)
% x0, y0: 初期条件, y0 = y'(x0)
% xs: y を求める x の値, n: 区間数
% y：数値解, y = y(xs)

h = (xs-x0)./n;
h2 = h.*0.5;
y = y0;
for I=0:n
    x = x0+h.*I;
    k1 = h.*Dfunc(x,y);
    k2 = h.*Dfunc(x+h2,y+k1.*0.5);
    k3 = h.*Dfunc(x+h2,y+k2.*0.5);
```

```
    k4 = h.*Dfunc(x+h,y+k3);
    y = y+(k1+2.0.*(k2+k3)+k4)./6.0;
end
end

function y = Dfunc7(x,y)
% 関数の導関数 y'=f(x,y)
y = -4.0.*(x-1.0).*y;
end

function y = Func7(x)
% 関数の例（解）
y = exp(-2.0.*(x-1.0).*(x-1.0));
end
```

■出力結果

```
n =1000
     x        Rk4 error(Rk4)      ode23      error      ode45      error     y=y(x)
  0.00   0.135335   0.000000   0.135335   0.000000   0.135335   0.000000   0.135335
  0.20   0.278215   0.000178   0.277792  -0.000245   0.278041   0.000004   0.278037
  0.40   0.487220   0.000467   0.486020  -0.000732   0.486757   0.000004   0.486752
  0.60   0.726846   0.000697   0.725011  -0.001138   0.726152   0.000003   0.726149
  0.80   0.923706   0.000590   0.921676  -0.001440   0.923120   0.000004   0.923116
  1.00   0.999998  -0.000002   0.998410  -0.001590   1.000007   0.000007   1.000000
  1.20   0.922228  -0.000888   0.921733  -0.001383   0.923127   0.000011   0.923116
  1.40   0.724521  -0.001628   0.725187  -0.000962   0.726160   0.000011   0.726149
  1.60   0.484884  -0.001868   0.486135  -0.000617   0.486759   0.000006   0.486752
  1.80   0.276439  -0.001599   0.277534  -0.000504   0.278034  -0.000003   0.278037
  2.00   0.134256  -0.001079   0.134665  -0.000671   0.135340   0.000005   0.135335
```

7　乱数

■乱数の発生　ある確率法則に従うでたらめな，予測できない数を**乱数**といい，通常，物理的な確率現象によって得られる．その中でも，ある区間で一様に分布する乱数を特に**一様乱数**といい，例えばさいころの目等がある．また，正規分布に従う乱数は，**正規乱数**という．このような乱数をここでは計算機でつくり出そう（**擬似乱数**）．

■線形合同法　乱数の初期値 x_0（整数）を与え，順次，一様乱数を求めていく一つの方法である線形合同法について説明する．この方法は，整数型の変数がとり得る正の範囲を，一様に擬似的にでたらめな（正）整数をとるようにしたものであり，初期値 x_0 を同じ値にすれば常に同じ乱数が得られることになる．予測できるという点で擬似乱数といわれるわけであるが，逆に考えれば再現性があるのでプログラムの動作確認には適しているといえる．

　この方法では，ある乱数 x_i（整数）から $ax_i + c$（a，c は整定数）を求め，整定数 m（$x_0 < m$）で割ったときの余りを次の乱数 x_{i+1}（整数）とする．

$$x_{i+1} = ax_i + c \quad (\mod m) \tag{79}$$

ただし，m は合同式の法，a は乗数（正整数），c は増分（正整数）を示す．より一様に分

布する乱数を得るためには，整数の範囲を十分大きくとる必要があるが，漸化式はいつかは元にもどり，周期は m 以下となる．したがって，整定数 a，c は乱数の周期が十分大きくなるよう注意して選ぶことが重要である．例えば，

- $m = 2^{31} - 1 = 2147483647$
- $a = 7^5 = 16807$
- $c = 0$

なお，$c = 0$ としたのものは，乗算合同法という．

■例題８＞　b_{min} 以上，b_{max} 未満で一様に発生する乱数を用いたプログラムを，次の要領で作成せよ．

1. 一様乱数に関わる計算
   (a) 0 以上，1 未満の一様乱数 $y_i = x_i/m$．ただし，x_i は区間 $[0, m-1]$ の一様乱数．
   (b) b_{min} 以上，b_{max} 未満の一様乱数 $z_i = (b_{max} - b_{min})y_i + b_{min}$

(c) MATLAB 関数 rand を用いた b_{min} 以上，b_{max} 未満の一様乱数

2. 計算条件ほか

(a) $b_{min} = 0.0$，$b_{max} = 1.0$，初期値 $x_0 = 1$ として乱数を求めて出力し，2 次元座標としてプロット生成する.

(b) $b_{min} = 0.0$，$b_{max} = 6.0$ として，乱数を生成し，区間を 6 等分して各々の区間に含まれる乱数の数を出力する.

■プログラム（M2_rand.m）　一様乱数を用いた計算例

```
function M2_rand
% テーマ：一様乱数
% 作成日：2010/12/08
% アルゴリズム：線形合同法
clear all
n = 6000;
s = 1;
X = zeros(1,n); Y = zeros(1,n);
for I=1:n
    [X(I),s] = F_rand(s);
    [Y(I),s] = F_rand(s);
end
figure('Name',' 一様乱数（線形合同法）','NumberTitle','on')
```

```
plot(X,Y,'bo','MarkerSize',2); axis square;
print -depsc -tiff fig_F_rand % eps ファイルの出力

rand('seed',0); % 初期値のセット
x = rand(1,n); y = rand(1,n);
figure('Name','rand','NumberTitle','on')
plot(x,y,'ro','MarkerSize',2); axis square;
print -depsc -tiff fig_rand % eps ファイルの出力

bmin = 0.0; bmax = 6.0;
k = zeros(1,6); K = zeros(1,6);
rand('seed',0); % 初期値のセット
z = rand(1,n);
z = (bmax-bmin).*z+bmin;
iz = floor(z)+1;
s = 1;
for I=1:n
    [Z,s] = F_rand(s);
    Z = (bmax-bmin).*Z+bmin;
    IZ = floor(Z)+1;
    for J=1:6
        if iz(I)==J
            k(J) = k(J)+1;
        end
        if IZ==J
            K(J) = K(J)+1;
```

```
        end
    end
end
fprintf(1,'          %7s %7s %7s %7s %7s %7s %7s\n',...
        '1','2','3','4','5','6','total');
fprintf(1,'(F_rand) %7d %7d %7d %7d %7d %7d %7d\n',K,sum(K));
fprintf(1,'(rnad)   %7d %7d %7d %7d %7d %7d %7d\n',k,sum(k));
end

function [y,x]=F_rand(s)
% 疑似一様乱数（線形合同法）
% input:  s = initial value of interger randam number
% output: x = interger randam number (-> next initial value)
%         y = randam number of range [0,1]
m = 2^31-1; a = 7^5; c = 0; % Park and Miller (e.g. s=1)

x = a*s+c;
x = mod(x,m);
y = x/m;

end
```

■出力結果

```
              1       2       3       4       5       6   total
(F_rand)    986     985    1028    1027    1002     972    6000
(rnad)      985     986    1027    1027    1002     973    6000
```

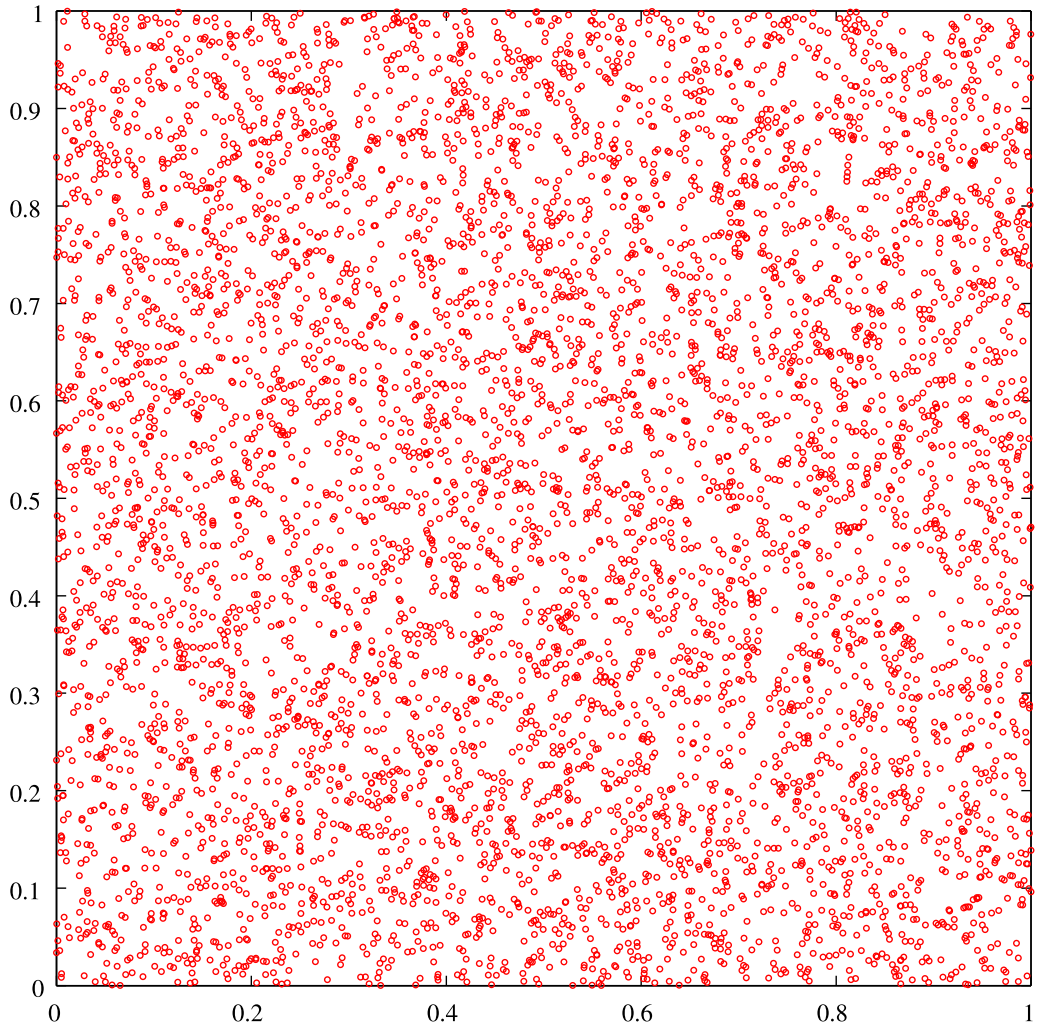


図 5　出力結果（fig_rand.eps）

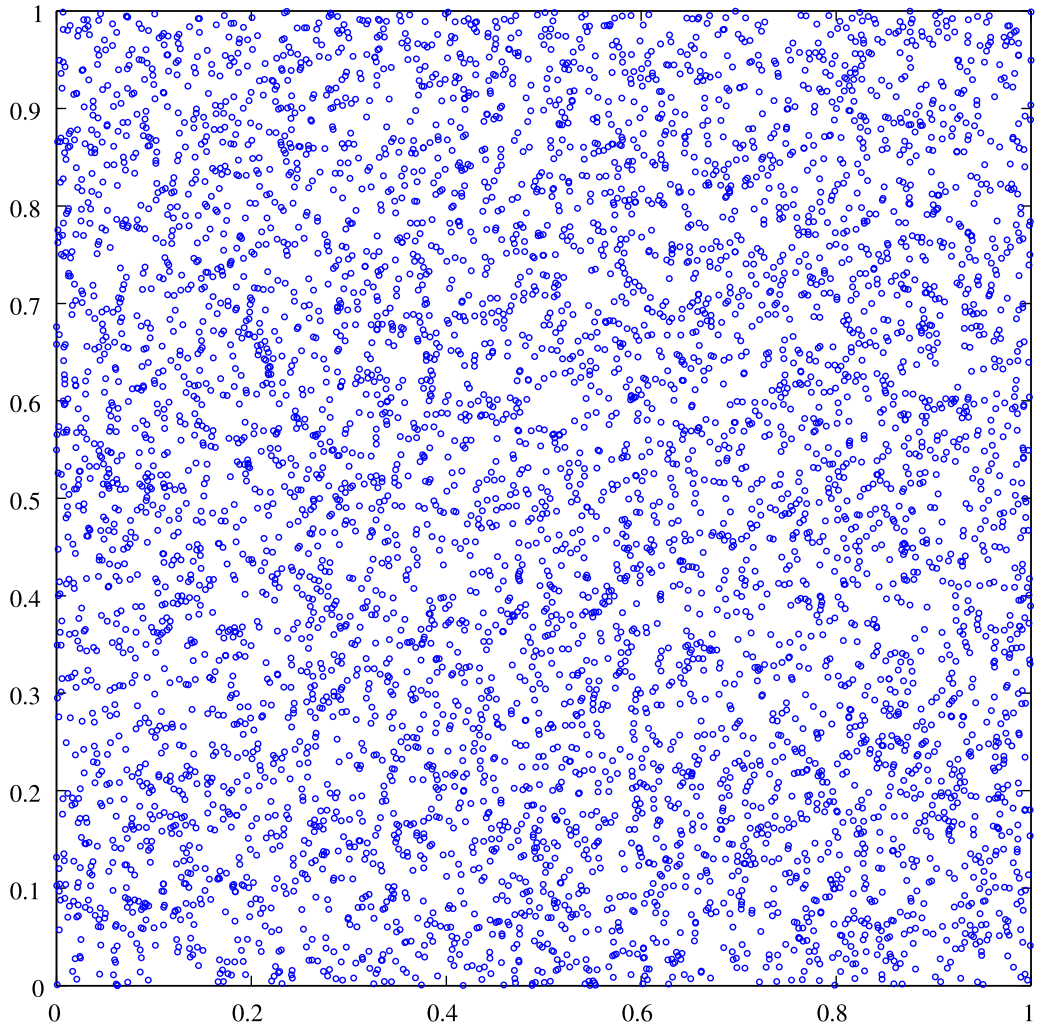


図 6　出力結果（fig_F_rand.eps）

■問題　　投げ矢を，一辺の長さが 2 の正方形に向けて多数回投げた場合を一様乱数（線形合同法）によって模擬し，プロットせよ．そして，中心 $(0,0)$ で半径 1 の円の中に当たる確率を計算し，π の近似値を求めよ．

(ヒント)　確率の推定値は，$\pi \cdot 1^2/2^2 (= \pi/4)$ となる．

8　フーリエ解析

■連続フーリエ変換対　時間 t 領域と周波数 f 領域とのフーリエ変換対は,

$$H(f) = \int_{-\infty}^{\infty} h(t) e^{-j2\pi ft} dt, \qquad h(t) = \int_{-\infty}^{\infty} H(f) e^{j2\pi ft} df \tag{80}$$

■単一のインパルスのフーリエ変換　関数 $h(t)$ として，時間領域における単一のインパルス（デルタ関数）を考えると,

$$h(t) = \delta(t - t') \tag{81}$$

ただし，$\delta(t)$ はデルタ関数であり，次式を満足する.

$$\int_{-\infty}^{\infty} \delta(t - t_0) f(t) dt = f(t_0) \tag{82}$$

これをフーリエ変換した $H(f)$ は,

$$H(f) = \int_{-\infty}^{\infty} h(t) e^{-j2\pi ft} dt = \int_{-\infty}^{\infty} \delta(t - t') e^{-j2\pi ft} dt = e^{-j2\pi ft'} \tag{83}$$

逆フーリエ変換した $h(t)$ は,

$$h(t) = \int_{-\infty}^{\infty} H(f)e^{j2\pi ft}df = \int_{-\infty}^{\infty} e^{-j2\pi ft'}e^{j2\pi ft}df$$
$$= \int_{-\infty}^{\infty} e^{j2\pi f(t-t')}df = \delta(t-t') \tag{84}$$

また，同様にして，関数 $H(f)$ として，周波数領域における単一のインパルスを考えると,

$$H(f) = \delta(f-f') \tag{85}$$

これを逆フーリエ変換した $h(t)$ は,

$$h(t) = \int_{-\infty}^{\infty} H(f)e^{j2\pi ft}df = \int_{-\infty}^{\infty} \delta(f-f')e^{j2\pi ft}df = e^{j2\pi f't} \tag{86}$$

フーリエ変換した $H(f)$ は,

$$H(f) = \int_{-\infty}^{\infty} h(t)e^{-j2\pi ft}dt = \int_{-\infty}^{\infty} e^{j2\pi f't}e^{-j2\pi ft}dt$$
$$= \int_{-\infty}^{\infty} e^{-j2\pi f(f-f')}dt = \delta(f-f') \tag{87}$$

■インパルス列のフーリエ変換　関数 $h(t)$ として，時間領域におけるインパルス列（周期 T）を考えると，

$$h(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT) \tag{88}$$

このような周期関数は，フーリエ級数で表すことができ，

$$h(t) = \sum_{k=-\infty}^{\infty} c_k e^{j2\pi \frac{k}{T} t} \tag{89}$$

ここで，c_k はフーリエ係数を示し，$h(t)$ の任意の一周期より求めると，

$$c_k = \frac{1}{T} \int_{nT-\Delta T}^{(n+1)T-\Delta T} h(t) e^{-j2\pi \frac{k}{T} t} dt = \frac{1}{T} \int_{-\infty}^{\infty} \delta(t - nT) e^{-j2\pi \frac{k}{T} t} dt \tag{90}$$

これより，$h(t)$ は，

$$h(t) = \sum_{k=-\infty}^{\infty} \left\{ \frac{1}{T} \int_{-\infty}^{\infty} \delta(t' - nT) e^{-j2\pi \frac{k}{T} t'} dt' \right\} e^{j2\pi \frac{k}{T} t} \tag{91}$$

これをフーリエ変換した $H(f)$ は,

$$H(f) = \int_{-\infty}^{\infty} \left[\sum_{k=-\infty}^{\infty} \left\{ \frac{1}{T} \int_{-\infty}^{\infty} \delta(t' - nT) e^{-j2\pi \frac{k}{T} t'} dt' \right\} e^{j2\pi \frac{k}{T} t} \right] e^{-j2\pi f t} dt \tag{92}$$

いま，$\bar{t} \equiv t' - nT - t$ とおき，t' を $\bar{t}$ に変数変換すると,

$$\begin{aligned}
H(f) &= \int_{-\infty}^{\infty} \sum_{k=-\infty}^{\infty} \frac{1}{T} \int_{-\infty}^{\infty} \delta(\bar{t} + t) e^{-j2\pi \frac{k}{T}(\bar{t} + nT)} d\bar{t}\, e^{-j2\pi f t} dt \\
&= \frac{1}{T} \sum_{k=-\infty}^{\infty} \int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} \delta(\bar{t} + t) e^{-j2\pi f t} dt \right) e^{-j2\pi \frac{k}{T}(\bar{t} + nT)} d\bar{t} \\
&= \frac{1}{T} \sum_{k=-\infty}^{\infty} \int_{-\infty}^{\infty} e^{j2\pi f \bar{t}} e^{-j2\pi \frac{k}{T} \bar{t}} d\bar{t} \\
&= \frac{1}{T} \sum_{k=-\infty}^{\infty} \int_{-\infty}^{\infty} e^{j2\pi \left(f - \frac{k}{T}\right) \bar{t}} d\bar{t} = \frac{1}{T} \sum_{k=-\infty}^{\infty} \delta\left(f - \frac{k}{T}\right)
\end{aligned} \tag{93}$$

■時間領域における標本化　時間領域の関数 $h(t)$ の標本化（サンプリング）を，次の周期 T のインパルス列 $\Delta_0(t)$

$$\Delta_0(t) = \sum_{l=-\infty}^{\infty} \delta(t - mT) \tag{94}$$

より行う（図 7(b) 左参照）．標本化の結果（図 7(c) 左参照），

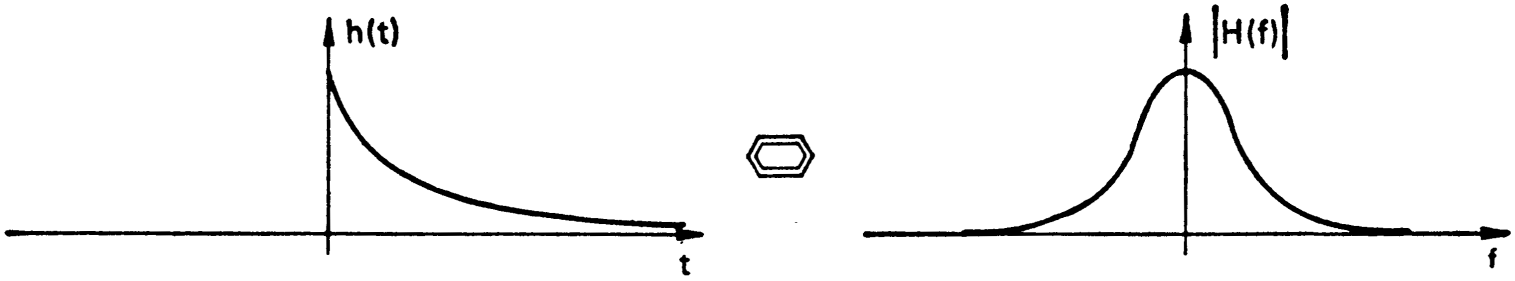
$$h(t)\Delta_0(t) = h(t) \sum_{m=-\infty}^{\infty} \delta(t - mT) = \sum_{m=-\infty}^{\infty} h(mT)\delta(t - mT) \tag{95}$$

このとき，インパルス列 $\Delta_0(t)$ のフーリエ変換 $\widetilde{\Delta}_0(f)$ は（図 7(b) 右参照），

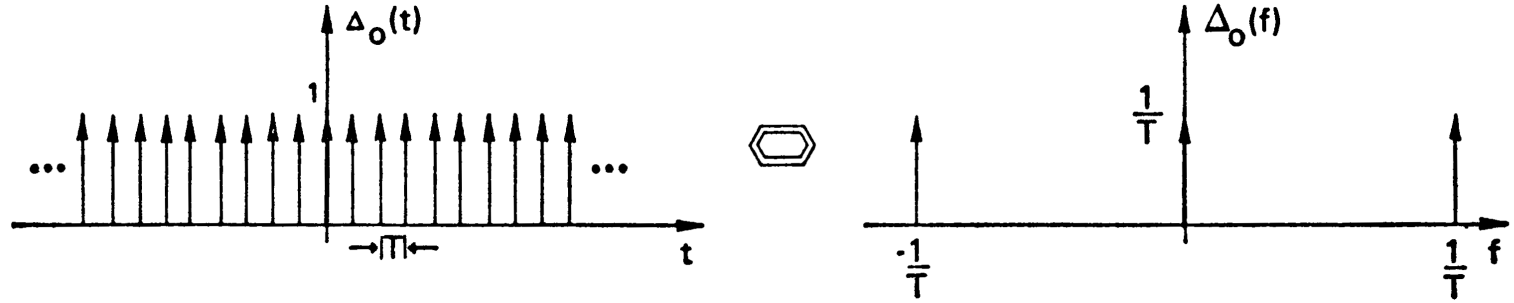
$$\widetilde{\Delta}_0(f) = \frac{1}{T} \sum_{k=-\infty}^{\infty} \delta\left(f - \frac{k}{T}\right) \tag{96}$$

よって，$h(t)\Delta_0(t)$ のフーリエ変換は，$H(f)$ と $\widetilde{\Delta}_0(f)$ の畳み込み積分で求められ，次のようになる（図 7(c) 右参照）.

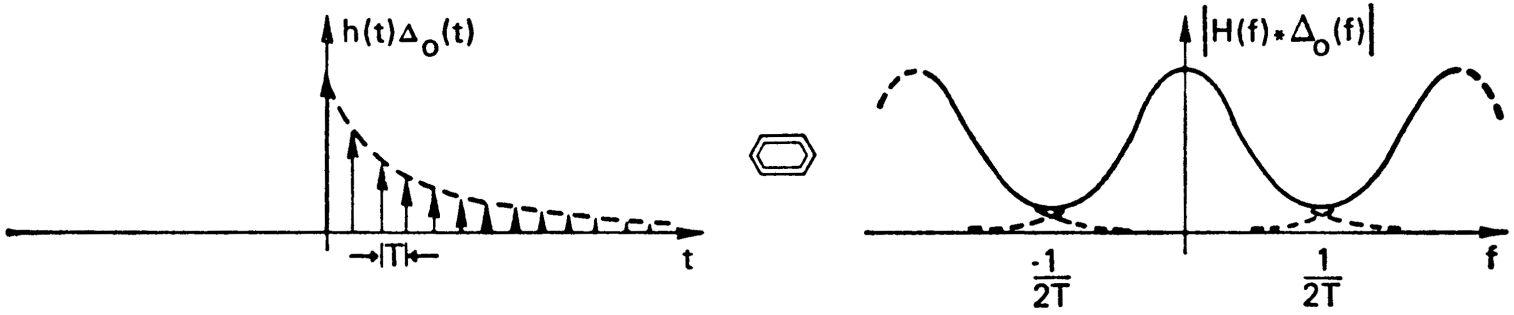
$$
\begin{aligned}
H(f) * \widetilde{\Delta}_0(f) &= \int_{-\infty}^{\infty} H(f')\widetilde{\Delta}_0(f-f')df' \\
&= \int_{-\infty}^{\infty} H(f')\frac{1}{T}\sum_{k=-\infty}^{\infty}\delta\left(f-f'-\frac{k}{T}\right)df' \\
&= \frac{1}{T}\sum_{k=-\infty}^{\infty}\int_{-\infty}^{\infty} H(f')\delta\left(f-f'-\frac{k}{T}\right)df' \\
&= \frac{1}{T}\sum_{k=-\infty}^{\infty} H\left(f-\frac{k}{T}\right)
\end{aligned}
\tag{97}
$$



(a) 連続フーリエ変換対



(b) インパルス列のフーリエ変換対



(c) インパルス列による時間領域の標本化

図 7　離散フーリエ変換対を求める過程（時間領域における標本化）

■時間領域における打ち切り　次に，時間領域の幅 T_0 の方形関数 $x(t)$

$$x(t) = \begin{cases} 1 & \left(-\dfrac{T}{2} < t < T_0 - \dfrac{T}{2}\right) \\ 0 & \left(\text{それ以外}\right) \end{cases} \tag{98}$$

より（図 8(a) 左参照），時間領域における打ち切りを行うと（図 8(b) 左参照），

$$\begin{aligned} h(t)\Delta_0(t)x(t) &= \left\{ \sum_{m=-\infty}^{\infty} h(mT)\delta(t-mT) \right\} x(t) \\ &= \sum_{m=0}^{N-1} h(mT)\delta(t-mT) \equiv h_0(t) \end{aligned} \tag{99}$$

ただし，$N(=T_0/T)$ はサンプル点数を示す．

このとき，方形関数 $x(t)$ のフーリエ変換 $X(f)$ は（図 8(a) 右参照），

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}dt = \int_{-\frac{T}{2}}^{T_0-\frac{T}{2}} e^{-j2\pi ft}dt = \left[\frac{e^{-j2\pi ft}}{-j2\pi f}\right]_{-\frac{T}{2}}^{T_0-\frac{T}{2}}$$

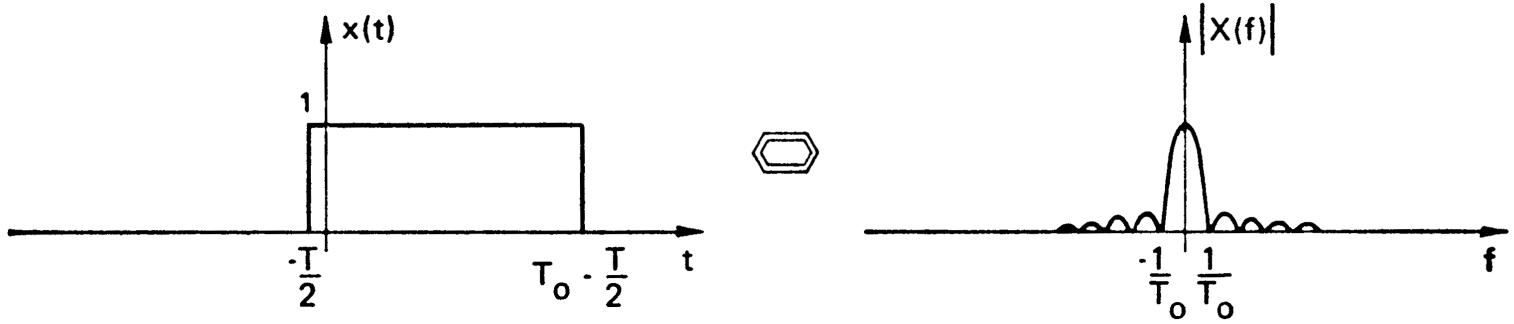
$$= \frac{e^{-j2\pi f\left(T_0-\frac{T}{2}\right)} - e^{j2\pi f\frac{T}{2}}}{-j2\pi f} = e^{-j\pi f\left(T_0-\frac{T}{2}\right)}\frac{e^{-j\pi f\left(T_0-\frac{T}{2}\right)} - e^{j\pi f\left(T_0-\frac{T}{2}\right)}}{-j2\pi f}$$

$$= \left(T_0 - \frac{T}{2}\right)e^{-j\pi f\left(T_0-\frac{T}{2}\right)}\frac{\sin\left(\pi f\left(T_0-\frac{T}{2}\right)\right)}{\pi f\left(T_0-\frac{T}{2}\right)} \tag{100}$$

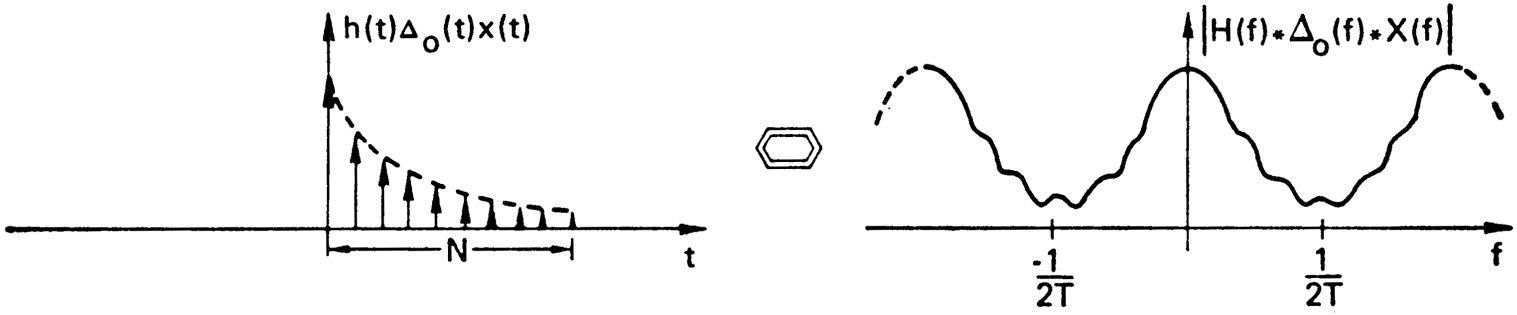
また，$h_0(t)$ のフーリエ変換 $H_0(f)$ は（図 8(b) 右参照）．

$$H_0(f) = \int_{-\infty}^{\infty} h_0(t)e^{-j2\pi ft}dt = \int_{-\infty}^{\infty}\sum_{m=0}^{N-1} h(mT)\delta(t-mT)e^{-j2\pi ft}dt$$

$$= \sum_{m=0}^{N-1} h(mT)\int_{-\infty}^{\infty}\delta(t-mT)e^{-j2\pi ft}dt = \sum_{m=0}^{N-1} h(mT)e^{-j2\pi fmT} \tag{101}$$



(a) 時間領域の方形関数とそのフーリエ変換



(b) 時間領域での打ち切りとそのフーリエ変換

図 8　離散フーリエ変換対を求める過程（時間領域における打ち切り）

■周波数領域における標本化　周波数領域におけるインパルス列の逆フーリエ変換もまた，インパルス列 $\Delta_1(t)$ となり（図 9(a) 左参照）

$$\Delta_1(t) = T_0 \sum_{r=-\infty}^{\infty} \delta(t - rT_0) \tag{102}$$

で表すと，時間領域では次のように $h_0(t)$ と $\Delta_1(t)$ の畳込み積分となる.

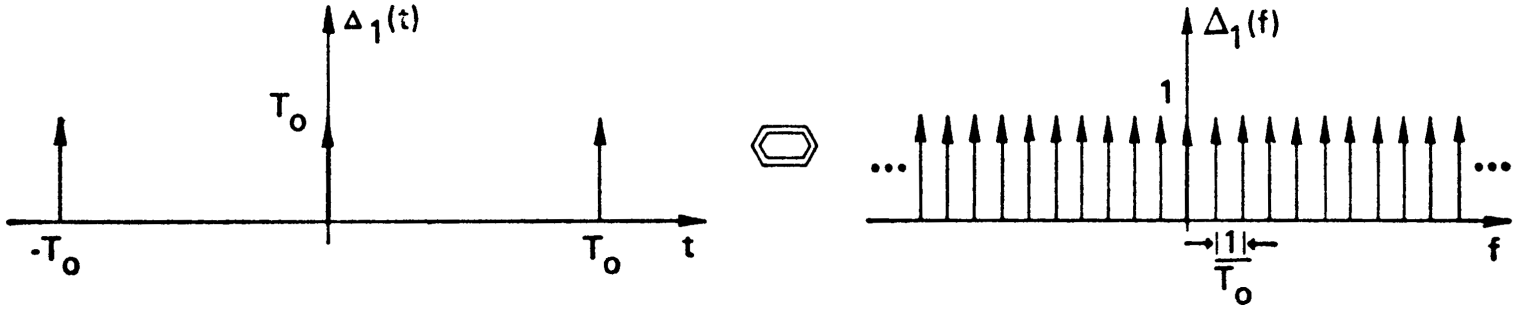
$$\begin{aligned} h_0(t) * \Delta_1(t) &= \int_{-\infty}^{\infty} h_0(\tau)\Delta_1(t-\tau)d\tau \\ &= \int_{-\infty}^{\infty} h_0(\tau) \left(T_0 \sum_{s=-\infty}^{\infty} \delta(t - sT_0 - \tau) \right) d\tau \\ &= T_0 \sum_{s=-\infty}^{\infty} \int_{-\infty}^{\infty} h_0(\tau)\delta(t - sT_0 - \tau)d\tau = T_0 \sum_{s=-\infty}^{\infty} h_0(t - sT_0) \\ &= T_0 \sum_{s=-\infty}^{\infty} \left(\sum_{m=0}^{N-1} h(mT)\delta(t - sT_0 - mT) \right) \equiv h_{\sim}(t) \end{aligned} \tag{103}$$

さらに，これをフーリエ変換すると，次のようになる.

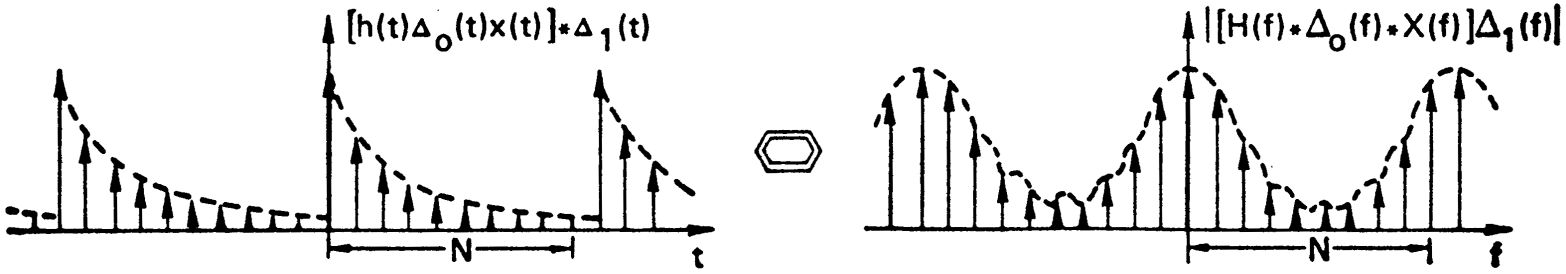
$$
\begin{aligned}
H_{\sim}(f) &= \int_{-\infty}^{\infty} h_{\sim}(t) e^{-j2\pi ft} dt \\
&= \int_{-\infty}^{\infty} T_0 \sum_{s=-\infty}^{\infty} \left(\sum_{m=0}^{N-1} h(mT) \delta(t - sT_0 - mT) \right) e^{-j2\pi ft} dt \\
&= T_0 \sum_{m=0}^{N-1} h(mT) \int_{-\infty}^{\infty} \left\{ \sum_{s=-\infty}^{\infty} \delta(t - sT_0 - mT) \right\} e^{-j2\pi ft} dt \qquad (104)
\end{aligned}
$$

変数変換 $t' \equiv t - mT$ より,

$$
\begin{aligned}
H_{\sim}(f) &= T_0 \sum_{m=0}^{N-1} h(mT) e^{-j2\pi fmT} \int_{-\infty}^{\infty} \left\{ \sum_{s=-\infty}^{\infty} \delta(t' - sT_0) \right\} e^{-j2\pi ft'} dt' \\
&= T_0 \sum_{m=0}^{N-1} h(mT) e^{-j2\pi fmT} \frac{1}{T_0} \sum_{n=-\infty}^{\infty} \delta\left(f - \frac{n}{T_0} \right) \qquad (105)
\end{aligned}
$$



(a) インパルス列のフーリエ変換対



(b) インパルス列による周波数領域の標本化

図 9　離散フーリエ変換対を求める過程（周波数領域における標本化）

ここで，$T_0 = NT$ より，

$$
\begin{aligned}
H_{\sim}(f) &= \sum_{m=0}^{N-1} h(mT) e^{-j2\pi f mT} \sum_{n=-\infty}^{\infty} \delta\left(f - \frac{n}{NT}\right) \\
&= \sum_{n=-\infty}^{\infty} \sum_{m=0}^{N-1} h(mT) e^{-j2\pi \frac{n}{NT} mT} \delta\left(f - \frac{n}{NT}\right) \\
&= \sum_{n=-\infty}^{\infty} \left[\sum_{m=0}^{N-1} h(mT) e^{-j2\pi nm/N} \delta\left(f - \frac{n}{NT}\right) \right]
\end{aligned}
\tag{106}
$$

いま，$f = \frac{n'}{NT}$ $(n' = -\infty, \cdots, -1, 0, 1, \cdots, \infty)$ のとき，

$$
H_{\sim}\left(\frac{n'}{NT}\right) = \sum_{m=0}^{N-1} h(mT) e^{-j2\pi n' m/N} \quad (n' = -\infty, \cdots, -1, 0, 1, \cdots, \infty) \tag{107}
$$

■離散フーリエ変換対　さらに，$n' \equiv n'' + lN$ $(n'' = 0, 1, \cdots, N-1)$, $(l = -\infty, \cdots, -1, 0, 1, \cdots, \infty)$ とおくと，

$$
\begin{aligned}
H_{\sim}\left(\frac{n''+lN}{NT}\right) &= \sum_{m=0}^{N-1} h(mT) e^{-j2\pi(n''+lN)m/N} \\
&= \sum_{m=0}^{N-1} h(mT) e^{-j2\pi n''m/N} e^{-j2\pi lm} \\
&= \sum_{m=0}^{N-1} h(mT) e^{-j2\pi n''m/N} = H_{\sim}\left(\frac{n''}{NT}\right)
\end{aligned}
\tag{108}
$$

これが，**離散フーリエ変換**である．同様にして，離散フーリエ逆変換は次式で与えられる(導出省略)．

$$
h_{\sim}(mT) = \frac{1}{N} \sum_{n=0}^{N-1} H\left(\frac{n}{NT}\right) e^{j2\pi nm/N} \qquad (m = 0, 1, \cdots, N-1) \tag{109}
$$

これより，時間領域のサンプル間隔を $\Delta t \equiv T$，周波数領域のサンプル間隔を $\Delta f \equiv \frac{1}{NT}$ とおくと，離散フーリエ変換対は次のようになる．

$$G(n\Delta f) = \sum_{m=0}^{N-1} g(m\Delta t) e^{-j2\pi nm/N} \qquad (n = 0, 1, \cdots, N-1) \tag{110}$$

$$g(m\Delta t) = \frac{1}{N} \sum_{n'=0}^{N-1} G(n\Delta f) e^{j2\pi nm/N} \qquad (m = 0, 1, \cdots, N-1) \tag{111}$$

ただし，

$$\Delta f = \frac{1}{N\Delta t} \tag{112}$$

■MATLAB の 1 次元高速フーリエ変換（FFT）　$\bar{n} \equiv n+1$，$\bar{m} \equiv m+1$ とおき，$\bar{m}$，$\bar{n}$ 番目の離散データを $g_{\bar{m}}$，$G_{\bar{n}}$ で表わすと，

$$G_{\bar{n}} = \Delta t \sum_{\bar{m}=1}^{N} g_{\bar{m}} \omega_N^{(\bar{n}-1)(\bar{m}-1)} \tag{113}$$

$$g_{\bar{m}} = \Delta f \sum_{\bar{n}=1}^{N} G_{\bar{n}} \omega_N^{-(\bar{n}-1)(\bar{m}-1)} \tag{114}$$

ただし,

$$\omega_N = e^{-j\frac{2\pi}{N}} \tag{115}$$

- フーリエ変換： `G = fft(g)`, `G = fft(g,N)`
- 逆フーリエ変換： `g = ifft(G)`, `g = ifft(G,N)`

ただし，`N` はサンプル点数を示す.

■MATLAB の 2 次元高速フーリエ変換（FFT）

- 2 次元フーリエ変換： `G = fft2(g)`, `G = fft2(g,N)`
- 2 次元逆フーリエ変換： `g = ifft2(G)`, `g = ifft2(G,N)`

■関連する MATLAB 関数

- DC 成分をスペクトルの中心に移動： `Y = fftshift(X)`
- 上の逆のシフト： `Y = ifftshift(X)`

`fftshift(X)` より，ベクトル X（1 次元）の右半分と左半分を入れ替え，あるいは行列 X（2 次元）の第 1 象限と第 3 象限を，第 2 象限と第 4 象限を入れ替えることができる.

- 指定した値 m 以上の最小の 2 のべき乗値 n：　`n = nextpow2(m)`

■例題９＞　次の方形関数（1 次元）$h(t)$ を MATLAB 関数を用いて高速フーリエ変換せよ．また，方形関数の連続フーリエ変換 $H(f)$ も計算し，比較せよ．

■方形関数のフーリエ変換　時間領域の方形関数 $h(t)$ として，$-T_r \leq t \leq T_r$ で大きさ A（定数）の方形関数を考え，フーリエ変換すると，

$$H(f) = \int_{-\infty}^{\infty} h(t)e^{-j2\pi ft}dt = \int_{-T_r}^{T_r} Ae^{-j2\pi ft}dt = A\left[\frac{e^{-j2\pi ft}}{-j2\pi f}\right]_{-T_r}^{T_r}$$
$$= A\frac{e^{-j2\pi fT_r} - e^{j2\pi fT_r}}{-j2\pi f} = A\frac{-j2\sin(2\pi fT_r)}{-j2\pi f} = 2AT_r\frac{\sin(2\pi fT_r)}{2\pi fT_r} \quad (116)$$

■プログラム（M2_fft.m）　フーリエ変換の簡単な計算例

```
function M2_fft
% テーマ：フーリエ解析
% 作成日：2010/12/21
% アルゴリズム：高速フーリエ変換（FFT）
close all % 全てのウインドウを閉じる
clear all % 全ての変数の初期化
format short

N = 32; fprintf(1,'データ数 N = %d\n',N);

% 時間領域のサンプル点の設定
tspan = 20.0; dt = tspan./N;
tmin = -tspan./2.0; tmax = -tmin-dt;
t = linspace(tmin,tmax,N);
fprintf(1,'t の最小値 tmin = %f, 最大値 tmax = %f, データ間隔 dt =%f\n',...
    t(1),t(N),dt)

% 周波数域のサンプル点の設定
df = 1.0./tspan;
fmin = -df.*N./2; fmax = -fmin-df;
f = linspace(fmin,fmax,N);
fprintf(1,'f の最小値 fmin = %f, 最大値 fmax = %f, データ間隔 df =%f\n',...
    f(1),f(N),df)
```

```
% FFT による計算
Tr = 4.0; % 方形関数の範囲
A = 1.25; % 方形関数の振幅
z = A.*(t>=-Tr & t<Tr); % 方形関数値の計算
zs = ifftshift(z); % データ逆シフト
Zs = dt.*fft(zs); % FFT
Z = fftshift(Zs); % データのシフト

% 解析的に方形関数をフーリエ変換した結果
NN = 100*N;
ttmin = tmin; ttmax = -ttmin;
tt = linspace(ttmin,ttmax,NN);
zz = A.*(tt>=-Tr & tt<Tr); % 方形関数値の計算
ffmin = fmin; ffmax = -ffmin;
ff = linspace(ffmin,ffmax,NN);
ff = ff + (ff==0)*eps; % ゼロのときだけ eps を加える
fc = 2.0.*pi.*Tr.*ff;
ZZ = 2.0.*A.*Tr.*sin(fc)./fc; % 方形関数のフーリエ変換

% 方形関数およびフーリエ変換の結果を一つのウインドウに作図
subplot(2,1,1);
stem(t,z,'o'); hold on;
plot(tt,zz,'r');
title('Time domain'); xlabel('t'); ylabel('h(t)');
subplot(2,1,2);
stem(f,Z,'o'); hold on;
```

```
plot(ff,ZZ,'r');
title('Frequency domain'); xlabel('f'); ylabel('H(f)');
print -depsc -tiff fig_fft % eps ファイルの出力

end
```

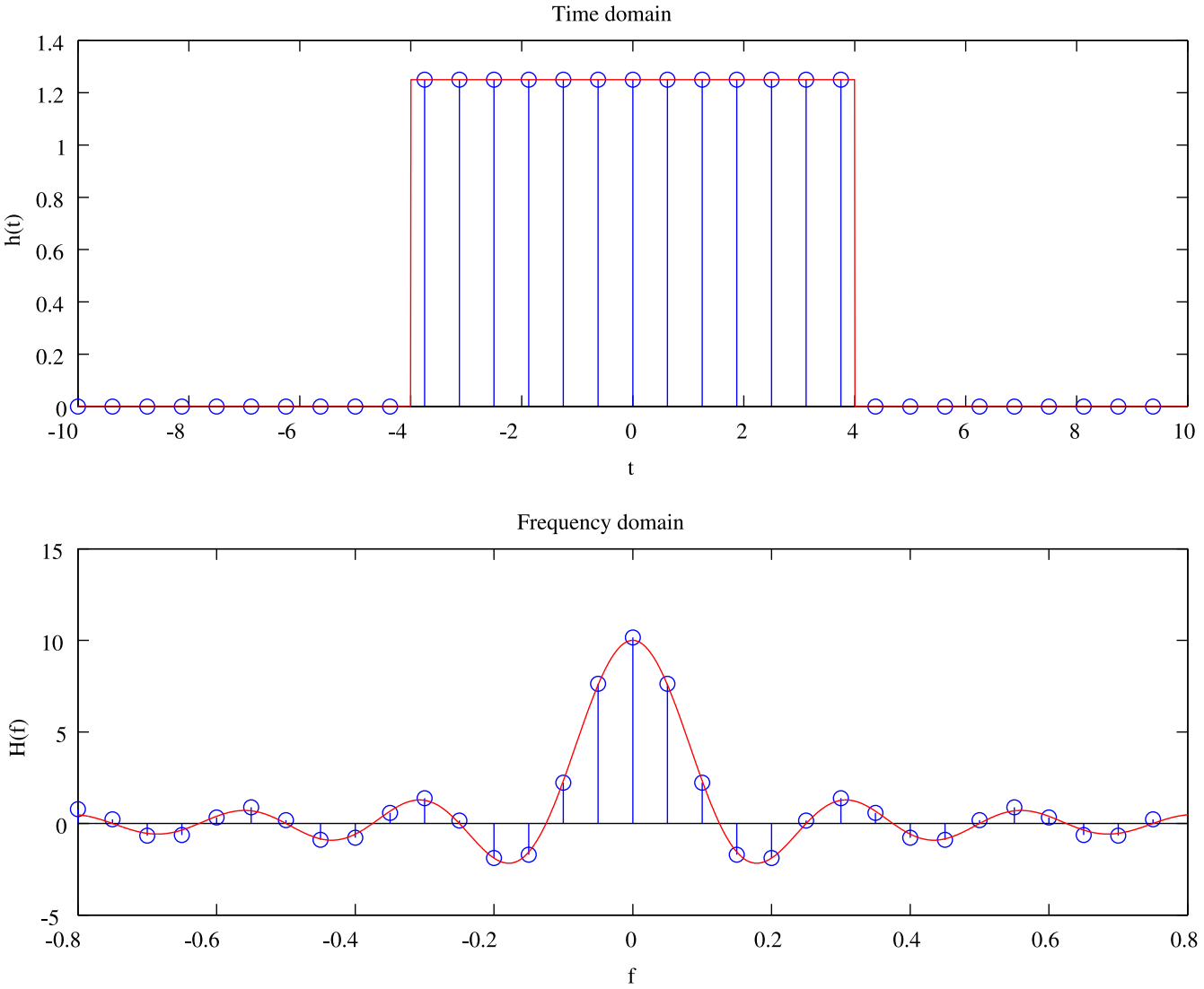


図 10　出力結果（fig_fft.eps）

■フーリエ変換対の別の形　角周波数 $\omega(=2\pi f)$ を用いて連続フーリエ変換対を表わすと，$d\omega = 2\pi df$ より，

$$H(\omega) = \int_{-\infty}^{\infty} h(t)e^{-j\omega t}dt \tag{117}$$

$$h(t) = \int_{-\infty}^{\infty} H(\omega)e^{j\omega t}df = \frac{1}{2\pi}\int_{-\infty}^{\infty} H(\omega)e^{j\omega t}d\omega \tag{118}$$

あるいは，次のようなフーリエ変換対も考えられる.

$$H(\omega) = a_1\int_{-\infty}^{\infty} h(t)e^{-j\omega t}dt, \quad h(t) = a_2\int_{-\infty}^{\infty} H(\omega)e^{j\omega t}d\omega \tag{119}$$

ただし，a_1，a_2 は任意ではない．そこで，定数 a_1a_2 を求めるため，上のフーリエ変換対の第 1 式の $H(\omega)$ を，第 2 式に代入すると，

$$\begin{aligned} h(\tau) &= a_2\int_{-\infty}^{\infty} H(\omega)e^{j\omega\tau}d\omega = a_2\int_{-\infty}^{\infty}\left(a_1\int_{-\infty}^{\infty} h(t)e^{-j\omega t}dt\right)e^{j\omega\tau}d\omega \\ &= a_1a_2\int_{-\infty}^{\infty} h(t)\left(\int_{-\infty}^{\infty} e^{j\omega(\tau-t)}d\omega\right)dt \end{aligned} \tag{120}$$

ここで，デルタ関数の公式を $\omega = 2\pi f$ で変数変換すると，$d\omega = 2\pi df$ より，

$$\delta(\tau - t) = \int_{-\infty}^{\infty} e^{j2\pi f(\tau - t)} df = \int_{-\infty}^{\infty} e^{j\omega(\tau - t)} \frac{d\omega}{2\pi} \tag{121}$$

よって，

$$\int_{-\infty}^{\infty} e^{j\omega(\tau - t)} d\omega = 2\pi\delta(\tau - t) \tag{122}$$

これより，

$$h(\tau) = a_1 a_2 \int_{-\infty}^{\infty} h(t) 2\pi\delta(\tau - t) dt = a_1 a_2 h(\tau) 2\pi \tag{123}$$

したがって，

$$a_1 a_2 = \frac{1}{2\pi} \tag{124}$$

このとき，離散フーリエ変換対は次のようになる.

$$G(n\Delta\omega) = a_1 \sum_{m=0}^{N-1} g(m\Delta t) e^{-jnm/N} \qquad (n = 0, 1, \cdots, N-1) \tag{125}$$

$$g(m\Delta t) = \frac{a_2}{N} \sum_{n=0}^{N-1} G(n\Delta\omega) e^{jnm/N} \qquad (m = 0, 1, \cdots, N-1) \tag{126}$$

ただし,

$$\Delta\omega = \frac{2\pi}{N\Delta t} \tag{127}$$

9　連立 1 次方程式

次式は，$z_1, z_2, \cdots z_n$ を未知数とする**連立 1 次方程式**である.

$$t_{11}z_1 + t_{12}z_2 + t_{13}z_3 + \cdots \quad + t_{1n}z_n = d_1 \tag{128}$$

$$t_{21}z_1 + t_{22}z_2 + t_{23}z_3 + \cdots \quad + t_{2n}z_n = d_2 \tag{129}$$

$$t_{31}z_1 + t_{32}z_2 + t_{33}z_3 + \cdots \quad + t_{3n}z_n = d_3 \tag{130}$$

$$\cdots$$

$$t_{n1}z_1 + t_{n2}z_2 + t_{n3}z_3 + \cdots \quad + t_{nn}z_n = d_n \tag{131}$$

これを行列表示すると,

$$\begin{pmatrix} t_{11} & t_{12} & t_{13} & \cdots & t_{1n} \\ t_{21} & t_{22} & t_{23} & \cdots & t_{2n} \\ t_{31} & t_{32} & t_{33} & \cdots & t_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{n1} & t_{n2} & t_{n3} & \cdots & t_{nn} \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{pmatrix} \tag{132}$$

これを,

$$[T][z] = [d] \tag{133}$$

とおけば，行列 $[T]$ および列ベクトル $[d]$ が既知で，未知の列ベクトル $[z]$ を数値的に求める問題となっている．

■ガウスの消去法　上式の左辺の行列 $[T]$ の形を，行や列を入れ替えたり，線形結合させたりして変形し，最終的に三角行列を得るのが**ガウスの消去法**である．ここでは，**上三角行列** $[U]$ に変形していく方法について説明する．

まず，消去の第一段階は，行列 $[T]$ から上三角行列 $[U]$ の第２列の形になるよう変形することである．連立１次方程式の第１式はそのままで，この第１式を使って第２式以降の式の z_1 の項を消去すればよい．例えば，第２式では式 (129) $-t_{21}/t_{11}\times$ 式 (128) とすればよい．第３式以降も同様に変形すると次のようになる．

$$t_{11}z_1 + t_{12}z_2 + t_{13}z_3 + \cdots + t_{1n}z_n = d_1 \tag{134}$$

$$t_{22}^{(1)}z_2 + t_{23}^{(1)}z_3 + \cdots + t_{2n}^{(1)}z_n = d_2^{(1)} \tag{135}$$

$$t_{22}^{(1)}z_2 + t_{33}^{(1)}z_3 + \cdots + t_{2n}^{(1)}z_n = d_3^{(1)} \tag{136}$$

$$\cdots$$

$$t_{n2}^{(1)}z_2 + t_{n3}^{(1)}z_3 + \cdots + t_{nn}^{(1)}z_n = d_n^{(1)} \tag{137}$$

これを行列表示すると,

$$\begin{pmatrix} t_{11} & t_{12} & t_{13} & \cdots & t_{1n} \\ 0 & t_{22}^{(1)} & t_{23}^{(1)} & \cdots & t_{2n}^{(1)} \\ 0 & t_{32}^{(1)} & t_{33}^{(1)} & \cdots & t_{3n}^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & t_{n2}^{(1)} & t_{n3}^{(1)} & \cdots & t_{nn}^{(1)} \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2^{(1)} \\ d_3^{(1)} \\ \vdots \\ d_n^{(1)} \end{pmatrix} \tag{138}$$

ただし，肩添字の (1) はこの第一段階で得られた値を示し，次のようになる.

$$t_{km}^{(1)} = t_{km} - \frac{t_{k1}}{t_{11}} t_{1m} \qquad (k, m = 2, 3, \cdots \ , n) \tag{139}$$

$$d_k^{(1)} = d_k - \frac{t_{k1}}{t_{11}} d_1 \qquad (k = 2, 3, \cdots \ , n) \tag{140}$$

次に，消去の第二段階では，上三角行列 $[U]$ の第３列の形を得るための変形を行う．上で得られた第１式，第２式はそのままで, 第２式を使って第３式以降の式の z_2 の項を消去

すると，次のようになる.

$$t_{11}z_1 + t_{12}z_2 + t_{13}z_3 + \cdots + t_{1n}z_n = d_1 \tag{141}$$

$$t_{22}^{(1)}z_2 + t_{23}^{(1)}z_3 + \cdots + t_{2n}^{(1)}z_n = d_2^{(1)} \tag{142}$$

$$t_{33}^{(2)}z_3 + \cdots + t_{2n}^{(2)}z_n = d_3^{(2)} \tag{143}$$

$$\cdots$$

$$t_{n3}^{(2)}z_3 + \cdots + t_{nn}^{(2)}z_n = d_n^{(2)} \tag{144}$$

これを行列表示すると,

$$\begin{pmatrix} t_{11} & t_{12} & t_{13} & \cdots & t_{1n} \\ 0 & t_{22}^{(1)} & t_{23}^{(1)} & \cdots & t_{2n}^{(1)} \\ 0 & 0 & t_{33}^{(2)} & \cdots & t_{3n}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & t_{n3}^{(2)} & \cdots & t_{nn}^{(2)} \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2^{(1)} \\ d_3^{(2)} \\ \vdots \\ d_n^{(2)} \end{pmatrix} \tag{145}$$

ただし，肩添字の (2) はこの第二段階で得られた値を示し，次のようになる.

$$t_{km}^{(2)} = t_{km}^{(1)} - \frac{t_{k2}^{(1)}}{t_{22}^{(1)}} t_{2m}^{(1)} \qquad (k, m = 3, 4, \cdots \ , n) \tag{146}$$

$$d_{k}^{(2)} = d_{k}^{(1)} - \frac{t_{k2}^{(1)}}{t_{22}^{(1)}} d_{2}^{(1)} \qquad (k = 3, 4, \cdots \ , n) \tag{147}$$

このような変形を上三角行列が得られるまで続けると，最終的には次のようになる.

$$t_{11}z_1 + t_{12}z_2 + t_{13}z_3 + \cdots \quad + t_{1n}z_n = d_1 \tag{148}$$

$$t_{22}^{(1)}z_2 + t_{23}^{(1)}z_3 + \cdots \quad + t_{2n}^{(1)}z_n = d_2^{(1)} \tag{149}$$

$$t_{33}^{(2)}z_3 + \cdots \quad + t_{2n}^{(2)}z_n = d_3^{(2)} \tag{150}$$

$$\cdots$$

$$t_{nn}^{(n-1)}z_n = d_n^{(n-1)} \tag{151}$$

これを行列表示すると,

$$\begin{pmatrix} t_{11} & t_{12} & t_{13} & \cdots & t_{1n} \\ & t_{22}^{(1)} & t_{23}^{(1)} & \cdots & t_{2n}^{(1)} \\ & & t_{33}^{(2)} & \cdots & t_{3n}^{(2)} \\ & & & \ddots & \vdots \\ 0 & & & & t_{nn}^{(n-1)} \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2^{(1)} \\ d_3^{(2)} \\ \vdots \\ d_n^{(n-1)} \end{pmatrix} \tag{152}$$

最終段階では,

$$t_{nn}^{(n-1)} = t_{nn}^{(n-2)} - \frac{t_{n,n-1}^{(n-2)}}{t_{n-1,n-1}^{(n-2)}} t_{n-1,n}^{(n-2)} \tag{153}$$

$$d_n^{(n-1)} = d_n^{(n-2)} - \frac{t_{n,n-1}^{(n-2)}}{t_{n-1,n-1}^{(n-2)}} d_{n-1}^{(n-2)} \tag{154}$$

このようにして，z_2，z_3，$\cdots$，$z_{(n-1)}$ を含む項を順次消去していくことを，**前進消去** (forward-elimination) という.

■後退代入　ガウスの消去法によって上三角行列 $[U]$ の形を用いて,

$$[U][z] = [e] \tag{155}$$

のように表わすことができたので，列ベクトル $[z]$ は次に示す**後退代入** (backward-substitution) によって容易に計算できる．いま，行列要素を新たに，

$$\begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ \mathbf{0} & & & u_{nn} \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix} = \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix} \tag{156}$$

とおくと，z_k に関する後退代入の式は次のようになる．

$$z_n = \frac{e_n}{u_{nn}} \tag{157}$$

$$z_k = \frac{1}{u_{kk}} \left(e_k - \sum_{m=k+1}^{n} u_{km} z_m \right) \qquad (k = n-1, n-2, \cdots, 1) \tag{158}$$

■部分ピボット選択　各段階の最初の割り算する要素は，丸め誤差を少なくするために，絶対値最大の要素を選んで対角の位置にもってくる必要がある．ただし，列交換は行わないことにする．これを**部分ピボット選択**という．

■例題 10 ＞　ガウスの消去法を基にしてに連立 1 次方程式 $[T][z] = [d]$ （$[z]$ は未知の列ベクトル）を解くプログラムを，次の要領で作成せよ．

1. ガウスの消去法（前進消去）による上三角行列を求める関数の作成．ただし，部分ピボット選択なしでもよい．
2. 後退代入を行う関数の作成
3. メインプログラムの作成．ただし，出力にあたっては，行列，列ベクトルであることが容易にわかるようにすること．
4. プログラムチェック用：　行列 $[T]$ および列ベクトル $[z]$ を与えて，$[d] = [T][z]$ を計算する関数の作成

10　行列計算（LU 分解）

■LU 分解による連立 1 次方程式の解法　行列 $[T]$ を**下三角行列** $[L]$（対角要素とその下側だけ 0 でない行列）と**上三角行列** $[U]$（対角要素とその上側だけ 0 でない行列）の積に分解するのが LU 分解で，連立方程式の解法に用いられる．いま，列ベクトル $[z]$ を未知数とする次のような連立方程式を考えよう．

$$[T][z] = [d] \tag{159}$$

ただし，$[d]$ は既知の列ベクトルを示す．行列 $[T]$ を LU 分解して，

$$[T] \equiv [L][U] \tag{160}$$

すなわち，

$$\begin{pmatrix} t_{11} & t_{12} & \cdots & t_{1n} \\ t_{21} & t_{22} & & \\ \vdots & & \ddots & \\ t_{n1} & \cdots & & t_{nn} \end{pmatrix} = \begin{pmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \mathbf{0} \\ \vdots & & \ddots & \\ l_{n1} & \cdots & & l_{nn} \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & & u_{1n} \\ & u_{22} & & \\ & & \ddots & \vdots \\ \mathbf{0} & & & u_{nn} \end{pmatrix} \tag{161}$$

で表し，次のように分解する．

$$[T][z] = [L]\big([U][z]\big) = [d] \tag{162}$$

これより，上式の ( ) の中を $[e]$ とおくと，次のようになる.

$$[L][e] = [d] \tag{163}$$

$$[U][z] = [e] \tag{164}$$

このように 1 組の連立方程式を 2 組に分解するのは，係数が三角行列の形をした連立方程式は非常に簡単に解くことができるからである.

まず，式 (163) は行列表示すると,

$$\begin{pmatrix} l_{11} & & & \mathbf{0} \\ l_{21} & l_{22} & & \\ \vdots & & \ddots & \\ l_{n1} & \cdots & & l_{nn} \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix} \tag{165}$$

となり，e_k は**前進代入**によって次のようにして求められる.

$$e_1 = \frac{d_1}{l_{11}} \tag{166}$$

$$e_k = \frac{1}{l_{kk}} \left(d_k - \sum_{m=1}^{k-1} l_{km} e_m \right) \qquad (k = 2, 3, \cdots \ , n) \tag{167}$$

同様にして，式 (164) を行列表示すると，

$$\begin{pmatrix} u_{11} & u_{12} & & u_{1n} \\ & u_{22} & & \\ & & \ddots & \vdots \\ \mathbf{0} & & & u_{nn} \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix} = \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix} \tag{168}$$

となり，z_k は**後退代入**によって次のようにして求められる．

$$z_n = \frac{e_n}{u_{nn}} \tag{169}$$

$$z_k = \frac{1}{u_{kk}} \left(e_k - \sum_{m=k+1}^{n} u_{km} z_m \right) \qquad (k = n-1, n-2, \cdots, 1) \tag{170}$$

これより，列ベクトル $[z]$ が求められる．

■LU 分解　次に，LU 分解について説明する．まず，未知数の数が式の数より n 個多いので，

$$l_{ii} = 1 \qquad (i = 1, 2, \cdots, n) \tag{171}$$

と決める．それ以外の未知数は，クラウト (Crout) のアルゴリズムより，$j = 1, 2, \cdots, n$ と次式を繰り返し順次求めていけばよい．

$$u_{ij} = t_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj} \qquad (i = 1, 2, \cdots \ , j) \tag{172}$$

$$l_{ij} = \frac{1}{u_{jj}} \left(t_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj} \right) \qquad (i = j+1, j+2, \cdots \ , n) \tag{173}$$

なお，前進代入で求めた e_k は，$l_{ii} = 1$ より次のようになる．

$$e_1 = d_1 \tag{174}$$

$$e_k = d_k - \sum_{m=1}^{k-1} l_{km} e_m \qquad (k = 2, 3, \cdots \ , n) \tag{175}$$

従って，行列 $[T]$ は次のように LU 分解されることになる．

$$[T] = [L][U] \tag{176}$$

すなわち,

$$\begin{pmatrix} t_{11} & t_{12} & \cdots & t_{1n} \\ t_{21} & t_{22} & & \\ \vdots & & \ddots & \\ t_{n1} & \cdots & & t_{nn} \end{pmatrix} = \begin{pmatrix} 1 & & & \\ l_{21} & 1 & & \mathbf{0} \\ \vdots & & \ddots & \\ l_{n1} & \cdots & & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & & u_{1n} \\ & u_{22} & & \\ & & \ddots & \vdots \\ \mathbf{0} & & & u_{nn} \end{pmatrix} \tag{177}$$

上の行列表示式において，下三角行列 $[L]$ の対角要素が $l_{ii} = 1$ であることに注目してほしい.

■行列式　正方行列 $[T]$ を LU 分解できれば，その行列式 $|T|$ は，次式により求められる.

$$\begin{aligned} |T| &= |L| \cdot |U| = (l_{11} \cdot l_{22} \cdots l_{nn})\,(u_{11} \cdot u_{22} \cdots u_{nn}) \\ &= (1 \cdot 1 \cdots 1)\,(u_{11} \cdot u_{22} \cdots u_{nn}) \\ &= u_{11} \cdot u_{22} \cdots u_{nn} \end{aligned} \tag{178}$$

ただし，ピボット選択がない場合を考えている.

■逆行列　正方行列 $[T]$ を LU 分解できれば，その逆行列 $[T]^{-1}$ を列ごとに求めることができる．つまり，単位行列を $[I]$ としたとき,

$$[Z] \equiv [T]^{-1}, \quad [D] \equiv [I] \tag{179}$$

とおけば，$[T][T]^{-1} = [I]$ は，

$$[T][Z] = [D] \tag{180}$$

で表され，$[D]$ の列ベクトル $[d_i]$ ごとに

$$[T][z_i] = [d_i] \tag{181}$$

を解けば，先に示した LU 分解を基にした計算によって列ベクトル $[z_i]$ が決まり，全ての列について計算すれば $[Z]$ が求められ，逆行列 $[T]^{-1}$ が得られることになる.

■3 重対角行列　行列 $[T]$ が，

$$[T] = \begin{pmatrix} b_1 & c_1 & & & & 0 \\ a_2 & b_2 & c_2 & & & \\ & a_3 & b_3 & c_3 & & \\ & & \ddots & \ddots & \ddots & \\ 0 & & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & & a_n & b_n \end{pmatrix} \tag{182}$$

のように **3 重対角行列**の場合には，クラウト (Crout) のアルゴリズムの一部を考慮して，下三角行列 $[L]$ および上三角行列 $[U]$ は次のようにおける.

$$[L] = \begin{pmatrix} 1 & & & & 0 \\ l_2 & 1 & & & \\ & l_3 & 1 & & \\ & & \ddots & \ddots & \\ 0 & & & l_{n-1} & 1 \\ & & & & l_n & 1 \end{pmatrix} \tag{183}$$

$$[U] = \begin{pmatrix} u_1 & c_1 & & & 0 \\ & u_2 & c_2 & & \\ & & u_3 & c_3 & \\ & & & \ddots & \ddots \\ 0 & & & & u_{n-1} & c_{n-1} \\ & & & & & u_n \end{pmatrix} \tag{184}$$

未知数 l_k, u_k は次のようにして求められる.

$$u_1 = b_1 \tag{185}$$

$$l_k = \frac{a_k}{u_{k-1}} \qquad (k = 2, 3, \cdots, n) \tag{186}$$

$$u_k = b_k - l_k c_{k-1} \qquad (k = 2, 3, \cdots, n) \tag{187}$$

上三角行列に対する後退代入, および下三角行列の前進代入より,

$$z_n = \frac{e_n}{u_n} \tag{188}$$

$$z_k = \frac{1}{u_k}(e_k - c_k z_{k+1}) \qquad (k = n-1, n-2, \cdots, 1) \tag{189}$$

$$e_1 = d_1 \tag{190}$$

$$e_k = d_k - l_k e_{k-1} \qquad (k = 2, 3, \cdots, n) \tag{191}$$

11　関数の離散近似（補間）

■スプライン関数　何らかの連続条件によって多項式を接続した区分的な多項式を**スプライン関数**という（スプラインとは，離散点を滑らかな曲線でつなぐ自在定規を意味する）．このつなぎ目（連続条件を与える点）を節点とよび，通常，実験データを得た点や複雑な関数値を求めた点が節点になる．また，これらの点を滑らかな曲線でつなぎ，その間の点を推定することは**補間**と呼ばれる．スプライン関数は多項式の次数に応じて定義できるが，多項式の次数を上げると余分な振動が生じ，逆に次数が低いと十分な精度が得られないため，ここでは 3 次スプライン関数をとり上げ説明する．

■非周期の 3 次スプライン関数　n 個のデータ (x_j, y_j) $(j = 1, 2, \cdots, n)$ が与えられた場合を考える．3 次スプライン関数 $S(x)$ は，i 番目の区間 $[x_i,\ x_{i+1}]$ $(i = 1, 2, \cdots, n-1)$ において，x に関する 3 次の多項式で表される．ここでは，まず非周期な場合について説明する．

(1) 関数 $S(x)$ が 3 次の多項式であることと，2 次導関数が連続であることから，2 次導

関数 $S''(x)$ が折れ線によって次のように定義される.

$$S''(x) = \frac{M_{i+1}}{h_i}(x - x_i) - \frac{M_i}{h_i}(x - x_{i+1}) \tag{192}$$

だだし,

$$h_i = x_{i+1} - x_i \qquad (i = 1, 2, \cdots, n-1) \tag{193}$$

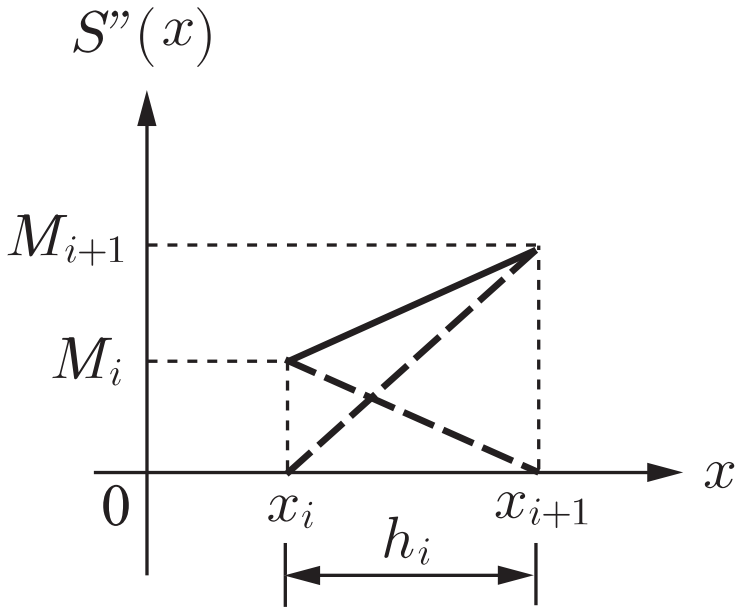


図 11　スプライン関数の 2 次導関数

ここで，M_i は節点 x_i における 2 次導関数の値を示す.

(2) $S''(x)$ を不定積分して，1 次導関数 $S'(x)$ および関数 $S(x)$ を求めると次のようになる.

$$S'(x) = -\frac{M_i}{2}\frac{(x_{i+1}-x)^2}{h_i} + \frac{M_{i+1}}{2}\frac{(x-x_i)^2}{h_i} + \bar{c}_{1,i} \tag{194}$$

$$S(x) = \frac{M_i}{6}\frac{(x_{i+1}-x)^3}{h_i} + \frac{M_{i+1}}{6}\frac{(x-x_i)^3}{h_i} + \bar{c}_{1,i}x + \bar{c}_{2,i} \tag{195}$$

スプライン関数がデータ点を通るための条件，すなわち節点での $S(x)$ の条件：

$$y_i = S(x_i), \qquad y_{i+1} = S(x_{i+1}) \tag{196}$$

より，積分定数 $\bar{c}_{1,i}$，$\bar{c}_{2,i}$ は次のようになる.

$$\bar{c}_{1,i} = \frac{y_{i+1}-y_i}{h_i} - \frac{1}{6}(M_{i+1}-M_i)h_i \tag{197}$$

$$\bar{c}_{2,i} = \frac{x_{i+1}y_i - y_{i+1}x_i}{h_i} - \frac{1}{6}(x_{i+1}M_i - M_{i+1}x_i)h_i \tag{198}$$

このように関数 $S(x)$ は，節点での 2 次導関数値を未知数として含む式によって表される.

(3) 節点において $S''(x)$ および $S(x)$ が連続であることは明らか．残る連続条件は，1 次導関数 $S'(x)$ についてである．まず，両端を除く節点 x_j $(j = 2, 3, \cdots, n-1)$ において，次式を満足しなければならない．

$$S'(x_{j-}) = S'(x_{j+}) \tag{199}$$

ただし，

$$S'(x) = M_j \left\{ -\frac{(x_{j+1} - x)^2}{2h_j} + \frac{h_j}{6} \right\} + M_{j+1} \left\{ \frac{(x - x_j)^2}{2h_j} - \frac{h_j}{6} \right\} + \frac{y_{j+1} - y_j}{h_j} \tag{200}$$

よって，未知数 M_j を決定する連立方程式は次のようになる．

$$a_j M_{j-1} + 2M_j + c_j M_{j+1} = d_j \qquad (j = 2, 3, \cdots, n-1) \tag{201}$$

ただし，

$$a_j \equiv \frac{h_{j-1}}{h_{j-1} + h_j}, \qquad c_j \equiv \frac{h_j}{h_{j-1} + h_j} = 1 - a_j \tag{202}$$

$$d_j \equiv \frac{6}{h_{j-1} + h_j} \left(\frac{y_{j+1} - y_j}{h_j} - \frac{y_j - y_{j-1}}{h_{j-1}} \right) \tag{203}$$

このようにして得られた式をみると，式の数が $(n-2)$ であるのに対し，未知数は n で条件がまだ必要である.

(4) 未知数と式の数が一致する連立方程式を得るためには，両端の節点における端条件を与えなければならない．つまり，n 個のデータ以外に付加的な端条件が必要となる．例えば，両端の 1 次導関数値，あるいは両端の 2 次導関数を与える等が考えられ，次のようになる.

$$2M_1 + c_1 M_2 = d_1, \qquad a_n M_n + 2M_n = d_n \tag{204}$$

両端の 1 次導関数が $S'(a) = y'_1$，$S'(b) = y'_n$ で与えられるとき,

$$\begin{aligned} &c_1 = a_n = 1 \\ &d_1 = \frac{6}{h_1}\left(\frac{y_2 - y_1}{h_1} - y'_1\right), \quad d_n = \frac{6}{h_{n-1}}\left(y'_n - \frac{y_n - y_{n-1}}{h_{n-1}}\right) \end{aligned} \tag{205}$$

両端の 2 次導関数が $S''(a) = y''_1$，$S''(b) = y''_n$ で与えられるとき,

$$c_1 = a_n = 0, \quad d_1 = 2y''_1, \quad d_n = 2y''_n \tag{206}$$

その他に強制的に両端の 2 次導関数値を 0 にしたものは自然スプラインと呼ばれる.

(5) 得られた式を行列表示すると，次のようになる.

$$\begin{pmatrix} 2 & c_1 & & & & & 0 \\ a_2 & 2 & c_2 & & & & \\ & a_3 & 2 & & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & & 2 & c_{n-2} & \\ 0 & & & & a_{n-1} & 2 & c_{n-1} \\ & & & & & a_n & 2 \end{pmatrix} \begin{pmatrix} M_1 \\ M_2 \\ M_3 \\ \vdots \\ \\ M_{n-1} \\ M_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ \\ d_{n-1} \\ d_n \end{pmatrix} \tag{207}$$

左辺の行列は 3 重対角行列であるから，すでに説明した LU 分解による方法によって解ける.

■LU 分解によって得られた非周期の 3 次スプライン関数の表示式

$$\begin{aligned} S(x) = \frac{M_i}{6}\left\{\frac{(x_{i+1}-x)^3}{h_i} - (x_{i+1}-x)h_i\right\} + \frac{x_{i+1}-x}{h_i}y_i \\ + \frac{M_{i+1}}{6}\left\{\frac{(x-x_i)^3}{h_i} - (x-x_i)h_i\right\} + \frac{x-x_i}{h_i}y_{i+1} \end{aligned} \tag{208}$$

だだし,

$$h_i = x_{i+1} - x_i \qquad (i = 1, 2, \cdots, n-1) \tag{209}$$

ここで，上三角行列に対する後退代入より，

$$M_n = \frac{e_n}{u_n} \tag{210}$$

$$M_k = \frac{1}{u_k}\left(e_k - c_k M_{k+1}\right) \qquad (k = n-1, n-2, \cdots, 1) \tag{211}$$

下三角行列の前進代入より，

$$e_1 = d_1 \tag{212}$$

$$e_k = d_k - l_k e_{k-1} \qquad (k = 2, 3, \cdots, n) \tag{213}$$

LU 分解より

$$u_1 = 2 \tag{214}$$

$$l_k = \frac{a_k}{u_{k-1}} \qquad (k = 2, 3, \cdots, n) \tag{215}$$

$$u_k = 2 - l_k c_{k-1} \tag{216}$$

ただし，c_1, a_n, d_1, d_n は端条件により式 (205) あるいは式 (206) 等のように決められ，そ

れ以外は次のようになる.

$$a_k = \frac{h_{k-1}}{h_{k-1} + h_k} \tag{217}$$

$$c_k = 1 - a_k \qquad (k = 2, 3, \cdots, n) \tag{218}$$

$$d_k = \frac{6}{h_{k-1} + h_k} \left(\frac{y_{k+1} - y_k}{h_k} - \frac{y_k - y_{k-1}}{h_{k-1}} \right) \tag{219}$$

12　ラゲルの多項式

ラゲル（Laguerre）の多項式 $L_{n,l}(x)$ は次式で与えられる.

$$L_{n,l}(x) = \sum_{i=0}^{n} \binom{n+l}{n-i} \frac{(-x)^i}{i!} \qquad (l > -1) \tag{220}$$

ここで,

$$\binom{n+l}{n-i} =_{n+l} C_{n-i} = \frac{(n+l)!}{(n-i)!\{(n+l)-(n-i)\}!} = \frac{(n+l)!}{(n-i)!(l+i)!} \tag{221}$$

なお,

$$\binom{n}{k} =_{n} C_{k} = \frac{n(n-1)\cdots(n-k+1)}{1\cdot 2\cdots k} = \frac{n!}{k!(n-k)!} \tag{222}$$

■低次の式の例　ラゲルの多項式の中で低次の式について見ると，$n = 0$ のとき,

$$L_{0,l}(x) = 1 \tag{223}$$

より,

$$L_{0,0}(x) = L_{0,1}(x) = L_{0,2}(x) = \cdots = 1 \tag{224}$$

$n = 1$ のとき,

$$L_{1,l}(x) = (1 + l) - x \tag{225}$$

より,

$$L_{1,0}(x) = 1 - x, \quad L_{1,1}(x) = 2 - x, \quad L_{1,2}(x) = 3 - x, \quad \cdots \tag{226}$$

$n = 2$ のとき,

$$L_{2,l}(x) = \frac{(2 + l)(1 + l)}{2} - (2 + l)x + \frac{x^2}{2} \tag{227}$$

より,

$$L_{2,0}(x) = 1 - 2x + \frac{x^2}{2} \tag{228}$$

$$L_{2,1}(x) = 3 - 3x + \frac{x^2}{2} \tag{229}$$

$$L_{2,2}(x) = 6 - 4x + \frac{x^2}{2} \tag{230}$$

$$\cdots\cdots\cdots\cdots \tag{231}$$

■正規直交系　$y = L_{n,l}(x)$ は，次のラゲルの微分方程式

$$xy'' + (l + 1 - x)y' + ny = 0 \tag{232}$$

の解であり，ロドリゲス表示で示すと次のようになる.

$$L_{n,l}(x) = \frac{e^x x^{-l}}{n!} \frac{d^n}{dx^n}(e^{-x} x^{n+l}) \tag{233}$$

密度関数を $e^{-x}x^l$ とすると，次のような直交性が得られる.

$$\int_0^\infty e^{-x} x^l L_{n,l}(x) L_{n',l}(x) dx = \frac{(n+l)!}{n!} \delta_{n,n'} \tag{234}$$

従って，重み係数を $L_{n,l}(x)$ にかければ直交関数系が得られ，次のようになる.

$$e^{-\frac{x}{2}} x^{\frac{l}{2}} L_{n,l}(x)$$

さらに正規直交系は，式 (234) の直交性より

$$\sqrt{\frac{n!}{(n+l)!}}\, e^{-\frac{x}{2}} x^{\frac{l}{2}} L_{n,l}(x)$$

によって得られる.

いま，$x \equiv X^2$ とおいて変数変換すると，$dx = 2XdX$ より

$$\int_0^\infty e^{-X^2} (X^2)^l L_{n,l}(X^2) L_{n',l}(X^2) 2XdX = \frac{(n+l)!}{n!} \delta_{n,n'} \tag{235}$$

さらに，$X \equiv \gamma/\gamma_0$ とおくと次式が得られる.

$$\int_0^\infty e^{-\frac{\gamma^2}{\gamma_0^2}} \left(\frac{\gamma^2}{\gamma_0^2}\right)^{l+\frac{1}{2}} L_{n,l}\left(\frac{\gamma^2}{\gamma_0^2}\right) L_{n',l}\left(\frac{\gamma^2}{\gamma_0^2}\right) d\left(\frac{\gamma}{\gamma_0}\right) = \frac{(n+l)!}{2n!} \delta_{n,n'} \tag{236}$$

■重みをつけて正規直交系にしたラゲルの多項式（ビームモード）

$$F_{\bar{m},n}(t) = \sqrt{\frac{n!}{(n+\bar{m})!}}\sqrt{2^{\bar{m}}}t^{\bar{m}}L_{n,\bar{m}}(2t^2)e^{-t^2} \qquad \text{ただし，} t \equiv \frac{\rho}{\omega}$$

これより，低次の $F_{\bar{m},n}(t)$ について具体的に考えてみる．まず，$n = 0$ のとき，

$$F_{\bar{m},0}(t) = \sqrt{\frac{1}{\bar{m}!}}\sqrt{2^{\bar{m}}}t^{\bar{m}}L_{0,\bar{m}}\left(2t^2\right)e^{-t^2} = \sqrt{\frac{1}{\bar{m}!}}\sqrt{2^{\bar{m}}}t^{\bar{m}}e^{-t^2} \tag{237}$$

より，

$$F_{0,0}(t) = e^{-t^2} \quad (\text{基本ビームモード}) \tag{238}$$

$$F_{1,0}(t) = \sqrt{2}te^{-t^2} \quad (\text{代表的な高次ビームモード}) \tag{239}$$

$$F_{2,0}(t) = \sqrt{2}t^2e^{-t^2} \tag{240}$$

$n = 1$ のとき，

$$F_{\bar{m},1}(t) = \sqrt{\frac{1}{(1+\bar{m})!}}\sqrt{2^{\bar{m}}}t^{\bar{m}}L_{1,\bar{m}}\left(2t^2\right)e^{-t^2} \tag{241}$$

より,

$$F_{0,1}(t) = L_{1,0}(2t^2)e^{-t^2} = (1-2t^2)e^{-t^2} \tag{242}$$

$$F_{1,1}(t) = tL_{1,1}(2t^2)e^{-t^2} = 2t(1-t^2)e^{-t^2} \quad (\text{代表的な高次ビームモード}) \tag{243}$$

$$F_{2,1}(t) = \frac{2}{\sqrt{6}}t^2 L_{1,2}(2t^2)e^{-t^2} = \frac{2}{\sqrt{6}}t^2(3-2t^2)e^{-t^2} \tag{244}$$

$n = 2$ のとき,

$$F_{\bar{m},2}(t) = \sqrt{\frac{2}{(2+\bar{m})!}}\sqrt{2^{\bar{m}}}t^{\bar{m}}L_{2,\bar{m}}\left(2t^2\right)e^{-t^2} \tag{245}$$

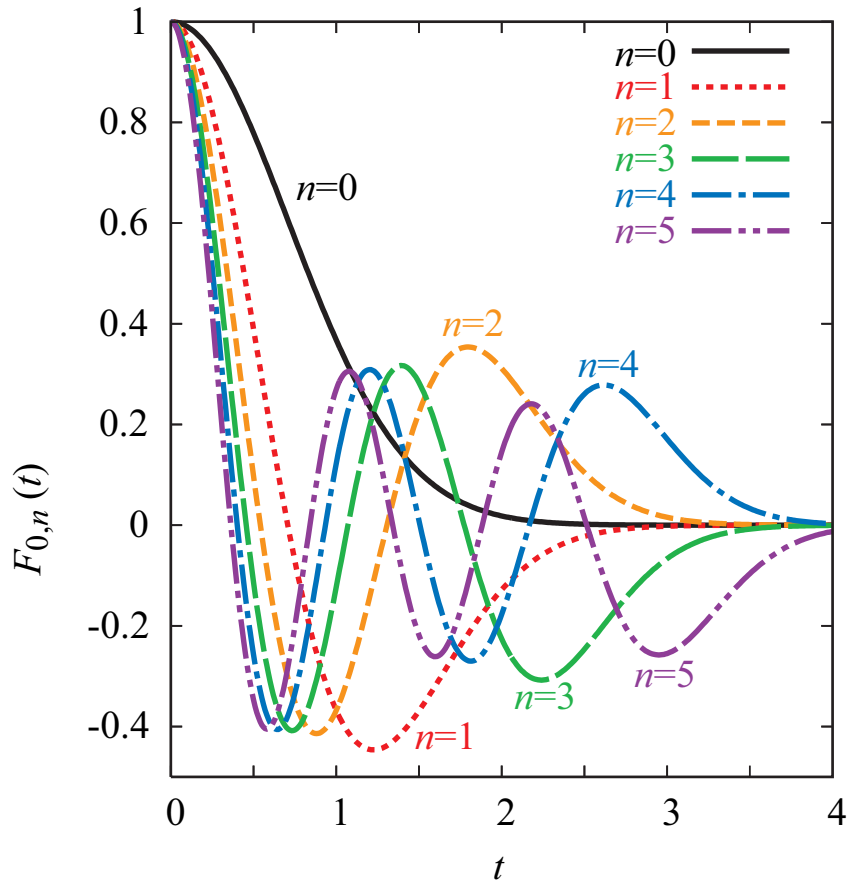
より,

$$F_{0,2}(t) = L_{2,0}(2t^2)e^{-t^2} = (1-4t^2+2t^4)e^{-t^2} \quad (\text{代表的な高次ビームモード}) \tag{246}$$

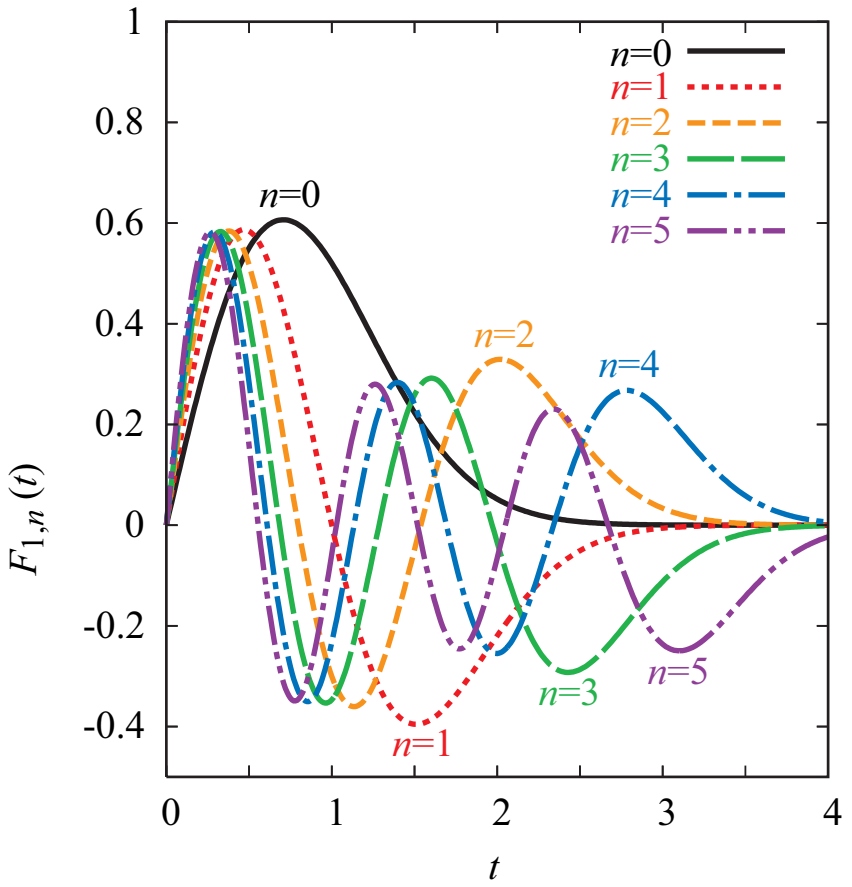
$$F_{1,2}(t) = \frac{2}{\sqrt{6}}tL_{2,1}(2t^2)e^{-t^2} = \frac{2}{\sqrt{6}}t(3-6t^2+2t^4)e^{-t^2} \tag{247}$$

$$F_{2,2}(t) = \frac{1}{\sqrt{3}}t^2 L_{2,2}(2t^2)e^{-t^2} = \frac{2}{\sqrt{3}}t^2(3-4t^2+t^4)e^{-t^2} \tag{248}$$

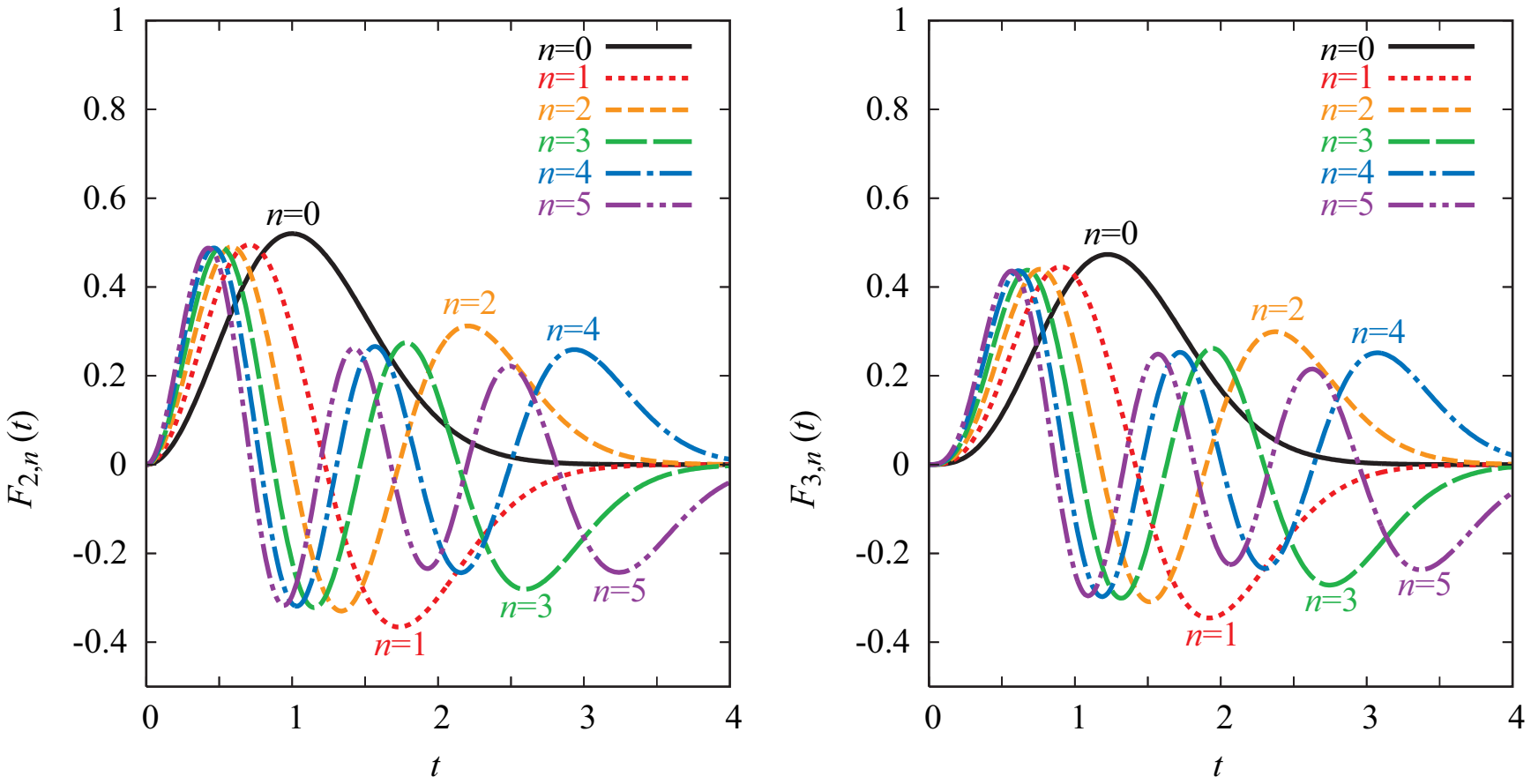
■問題　低次のビームモードの電界分布を等高線および電気力線により図示せよ.



(a) $\bar{m} = 0$



(b) $\bar{m} = 1$



(c) $\bar{m} = 2$　　(d) $\bar{m} = 3$

図 12　重みをつけて正規直交系にしたラゲルの多項式の計算例