

第3部：『魔方陣の新研究』： 摂田 寛二

第1章：三方陣の代数的研究（新）

第1節：一般型の三方陣を作る

0. 三方陣については、既に『序論』でも『基礎的研究』でも論じたので、特に新しい研究成果というものは無いが、そこで論じた基礎的な事柄が今後とも依然重要なので、もう一度取り上げて丁寧に復習しておこう。

1. 先ず一般型の三方陣の構成を志し、コンピュータに数えさせよう。プログラミングそのものに慣れておく必要があるからだ。今回与える定義式は、『序論』の時と同じだが、変数の決定のスタイルを今までとは少し変える。それと `for(){ };` の多重入れ子スタイルを止め、常に下位の手続きをサブルーチンとして押し出すスタイルを取ることにする。

先ず変数値を、`n2`, `n4`, `n6`, `n8` という順序で決める。何故ならば、数値1が来る場所はそこしか無いから。先ず1から決めたいと思うのは、自然だ。`n2` から決めても、結果が変わらないことをぜひ確認してほしい。

先ずは、わがリストを見ていただこう。冗長な部分が多い低速プログラムだが、わかりやすさを優先したためにあえてそうしたので、かかる教育的配慮が無ければ、ここはもう少しコンパクトで高速なプログラムを組めるところだ。

```
/** Advanced PROGRAM #2 for Magic Squares of Order 3 **
/** File: 'MS33AP22.c' Built by Kanji Setsuda **
/** on Jan.20, 2005 and Aug.16, 2013; **
/** with MacOSX 10.8.4 & Xcode 4.6.3 **
//
/** Basic Form & Simultaneous Equations:
// .--.--.--.   n1+n2+n3=C ... eq1;
// |n1|n2|n3|   n4+n5+n6=C ... eq2;
// |--+--+---|   n7+n8+n9=C ... eq3;
// |n4|n5|n6|   n1+n4+n7=C ... eq4;
// |--+--+---|   n2+n5+n8=C ... eq5;
// |n7|n8|n9|   n3+n6+n9=C ... eq6;
// '--'--'--'   n1+n5+n9=C ... eq7;
//                n3+n5+n7=C ... eq8;
//
#include <stdio.h>
//
short LSM;
short cnt, cnt2;
short nm[10], uflg[10];
short tn[5][10];
//
void stp1(void), stp2(void), stp3(void);
void stp4(void), stp5(void), stp6(void);
void stp7(void), stp8(void), stp9(void);
void ansrecord(void);
void prans(short x);
//
/** Main Program *
int main(){
short i;
printf("\n");
printf("*** Magic Squares of Order 3 **\n");
for(i=0;i<10;i++){nm[i]=0; uflg[i]=0;}
LSM=15; cnt=0; cnt2=0;
stp1();      /* Begin the Calculations *
if(cnt2>0){prans(cnt2);}
```

```

printf(" [Count = %d] OK!\n",cnt);
printf("\n");
printf("*** Calculated and Listed by Kanji Setsuda\n");
printf(" on Aug.16, 2013 with MacOSX & Xcode 4.6.3 **\n");
printf("\n");
return 0;
}
/* Begin the Calculations *
/* Set n2 *
void stp1(){
short i;
for(i=1;i<10;i++){
if(uflg[i]==0){
nm[2]=i; uflg[i]=1;
stp2();
uflg[i]=0;}
}
}
/* Set n4 *
void stp2(){
short i;
for(i=1;i<10;i++){
if(uflg[i]==0){
nm[4]=i; uflg[i]=1;
stp3();
uflg[i]=0;}
}
}
/* Set n6 *
void stp3(){
short i;
for(i=1;i<10;i++){
if(uflg[i]==0){
nm[6]=i; uflg[i]=1;
stp4();
uflg[i]=0;}
}
}
/* Set n8 *
void stp4(){
short i;
for(i=1;i<10;i++){
if(uflg[i]==0){
nm[8]=i; uflg[i]=1;
stp5();
uflg[i]=0;}
}
}
/* The Second Level *
/* Set n1 *
void stp5(){
short i;
for(i=1;i<10;i++){
if(uflg[i]==0){
nm[1]=i; uflg[i]=1;
stp6();
uflg[i]=0;}
}
}
/* Set n3 & Check(n1+n2+n3=LSM) *

```

```

void stp6(){
short i;
for(i=1;i<10;i++){
if((nm[1]+nm[2]+i==LSM)&&(uflg[i]==0)){
nm[3]=i; uflg[i]=1;
stp7();
uflg[i]=0;}}
}
}
/* The Third Level *
/* Set n7 & Check(n1+n4+n7=LSM) *
void stp7(){
short i;
for(i=1;i<10;i++){
if(nm[1]+nm[4]+i==LSM){
if(uflg[i]==0){
nm[7]=i; uflg[i]=1;
stp8();
uflg[i]=0;}}}
}
}
/* Set n5 & Check(n2+n5+n8=LSM)&(n4+n5+n6=LSM)&(n3+n5+n7=LSM) *
void stp8(){
short i;
for(i=1;i<10;i++){
if((nm[2]+i+nm[8]==LSM)&&(nm[4]+i+nm[6]==LSM)){
if((nm[3]+i+nm[7]==LSM)&&(uflg[i]==0)){
nm[5]=i; uflg[i]=1;
stp9();
uflg[i]=0;}}}
}
}
}
/* Set n9 & Check(n7+n8+n9=LSM)&(n3+n6+n9=LSM)&(n1+n5+n9=LSM) *
void stp9(){
short i;
for(i=1;i<10;i++){
if((nm[7]+nm[8]+i==LSM)&&(nm[3]+nm[6]+i==LSM)){
if((nm[1]+nm[5]+i==LSM)&&(uflg[i]==0)){
nm[9]=i; uflg[i]=1;
ansrecord();
uflg[i]=0;}}}
}
}
}
//
/* Record the Answers *
void ansrecord(){
short i;
tn[cnt2][0]=cnt+1;
for(i=1;i<10;i++){tn[cnt2][i]=nm[i];}
cnt++; cnt2++;
if(cnt2==4){prans(4); cnt2=0;}
}
//
/* Print 4 Answers *
void prans(short x){
short l,l3,n;
for(n=0;n<x;n++){printf("%14d/",tn[n][0]);}; printf("\n");
for(n=0;n<x;n++){printf(" .----.----.----.");}; printf("\n");
for(l=0;l<3;l++){l3=l*3;
for(n=0;n<x;n++){

```

```

    printf("  %2d %2d %2d  ",tn[n][l3+1],tn[n][l3+2],tn[n][l3+3]);}
    printf("\n");
    if(l<2){for(n=0;n<x;n++){printf("  |---+---+---|");}; printf("\n");}
}
for(n=0;n<x;n++){printf("  '---'---'---'");}; printf("\n");
//printf("\n");
}
//

```

* 一般型三方陣の原始解 8 個のリスト *

1/	2/	3/	4/
8 1 6	6 1 8	8 3 4	4 3 8
3 5 7	7 5 3	1 5 9	9 5 1
4 9 2	2 9 4	6 7 2	2 7 6
5/	6/	7/	8/
6 7 2	2 7 6	4 9 2	2 9 4
1 5 9	9 5 1	3 5 7	7 5 3
8 3 4	4 3 8	8 1 6	6 1 8

[Count = 8] OK!

** Calculated and Listed by Kanji Setsuda
on Aug. 16, 2013 with MacOSX & Xcode 4.6.3 **

定義段階では対象の補数対称型を宣言して居なかったにもかかわらず、結果は8解の全てが対称型になった。5が中央にいて動かない。5をはさんで点対称の位置にある2数の和は、どれも10に等しい。1+9=2+8=3+7=4+6=10

つまり、{1, 9}, {2, 8}, {3, 7}, {4, 6} が「10の補数」であるばかりでなく、ポジションを表す変数 {n1, n9}, {n2, n8}, {n3, n7}, {n4, n6} のペア自身も「10の補数」を成す。

(このように和が一定の10になる2数のことを「10の補数」と言う。「n1 と n9 は、10の補数を成す」と言う。10が周知の定数ならば、単に「n9 は、n1の補数である」という言い方もする。

どの補数ペアも n5をはさんで点対称の位置にあるので、必ず「補数対称型方陣」になる。)

三方陣には補数対称型しか無い。補数対称型8個の魔方陣が三次の魔方陣の全てである。

2. このことは、定義式の代数計算によっても、証明することができる。以前の説明を再録しよう。

基本条件を図と式によって次のように定義する。

D1. 基本図と変数名

n1 n2 n3
n4 n5 n6
n7 n8 n9

D2. 連立方程式

- n1+n2+n3=C ... (1);
- n4+n5+n6=C ... (2);
- n7+n8+n9=C ... (3);
- n1+n4+n7=C ... (4);
- n2+n5+n8=C ... (5);
- n3+n6+n9=C ... (6);
- n1+n5+n9=C ... (7);
- n3+n5+n7=C ... (8);

(1)~(6)式は、行和および列和に関する式で、いずれも一定の値になる。また(7)~(8)式は、対角線の上にある3数の和が行列の和と等しくなることを示している。1~9の自然数9個を重複無

く全個用いて、これら8本の連立方程式を満たさなくてはならない。
 Cは変数ではなくて定数なので、先ずこの行列和の値を決めておこう。

$$(1)+(2)+(3)$$

$$n1+n2+n3+\dots+n9=3*C$$

左辺には1～9の自然数が重複なく入る。1～9の自然数の総和は45だから、

$$45=3*C$$

$$\therefore C=15 \quad \dots (9)$$

行和、列和、対角線の和が、いずれも15になることがわかる。

あらためて8本の連立方程式を良く観察すると、方程式によって各変数がアクセスされる度合いが必ずしも平等ではないことに気付く。

変数	n1	n2	n3	n4	n5	n6	n7	n8	n9
度数	3	2	3	2	4	2	3	2	3

こういうふうに不均一がある時は、度数が特別に高い変数や低い変数の値には、特徴的な数値が入るものだ。先ずn5の値を追求してみよう。

n5を含む4式の辺々を加えて、整理してみる。

$$(2)+(5)+(7)+(8)$$

$$n4+n5+n6=C \quad \dots (2)$$

$$n2+n5+n8=C \quad \dots (5)$$

$$n1+n5+n9=C \quad \dots (7)$$

$$+) \quad n3+n5+n7=C \quad \dots (8)$$

$$n4+n5+n6+n2+n5+n8+n1+n5+n9+n3+n5+n7=4*C$$

$$3*n5+(n1+n2+n3+n4+n5+n6+n7+n8+n9)=4*C$$

ところが、(n1+n2+n3+n4+n5+n6+n7+n8+n9)の値は、1～9の総和に等しい。

$$3*n5+45=60$$

$$\therefore n5=5 \quad \dots (10)$$

方陣の中央には、必ず定数5が来ることがわかった。

これを(2),(5),(7),(8)式に戻すと、

$$n4+5+n6=15 \quad \dots (2) \quad \text{から} \quad n4+n6=15-5=10$$

$$\text{同様に} \quad n2+n8=10, \quad n1+n9=10, \quad n3+n7=10$$

$$\text{まとめて} \quad n1+n9=n2+n8=n3+n7=n4+n6=n5+n5=10 \quad \dots (11)$$

要するに、一般型の定義式から補数対称型の定義式を導けるのだ。こういう場合には、最初から補数対称型の定義式を一緒に加えて魔方陣の構成を試みても、同じ結果を得る。

条件式を増やすと、普通なら解の総数が減るものだが、この場合は減らない。何故か？論理的に「同値」なのである。

3. ところで、上の8つの解は、どれも相互にとても良く似ている。いや、「同一解」と言ってもよい。位置こそ違え、行列・主対角線の内容が全く同じだからだ。

主対角線は、どの解も $2+5+8=15$; $4+5+6=15$; の2本。1は、いつも6と8とによって挟まれている。

形の違いは、回転したり裏返ったりして出来るものばかりだ。

このことには昔から気づいていて、多くの人が「解は1つしかない」と言っていた。我々も、1つと見る。8個の中の任意の解1個を以て「標準解」とも「代表解」ともしている。

下の図を見てほしい。1番解には、任意に選んだ「代表解」をおいた。上の段の右の3つは、それを順次時計回りに90度ずつ回転させたものだ。

5番解は、主対角線 $2+5+8=15$ を軸として1番解を対称的に反転させたものだ。

6, 7, 8番解は、5番解を順次時計回りに90度ずつ回転させていったものだ。

1番解は、「原形」だから、回転も反転もさせていない。

* 回転と反転による変形 *

	1/ →	2/ →	3/ →	4/
8 1 6	4 3 8	2 9 4	6 7 2	
3 5 7	9 5 1	7 5 3	1 5 9	
4 9 2	2 7 6	6 1 8	8 3 4	
↓	5/ →	6/ →	7/ →	8/
8 3 4	6 1 8	2 7 6	4 9 2	
1 5 9	7 5 3	9 5 1	3 5 7	
6 7 2	2 9 4	4 3 8	8 1 6	

こうして作った8解が、今までに求めた8解と同じものであることを確かめてほしい。

このように「回転」と「反転」という2種類の変換法を仮定すれば、代表解の1個からいつでも原始解の8個を再現することができる。

このことを知っていれば、我々はもう8個の原始解をいちいち記録・保存しておく必要は無いわけで、1個だけ代表解を持っていればよいことになる。

事情は、四方陣でも五方陣でも何方陣でも、同じだ。回転も反転も、本来次数・タイプに無関係に可能だから、いつでも1個から8個を再現することができる。

そんなわけで、代表解のリストは、いつも原始解の8分の1の規模で済む。そういう理由で、一般に見られるリスト出力は大概がスマートな「代表解のリスト」になっている。

今後は、この種の代表解を「標準解」と呼ぶことにしよう。

4. では、どうやって標準解のみを選んで出力するか。

リスト整形出力用の不等式を使うのである。

上の例では、 $n_2 = 1$ から始めているから、 $n_2 < n_8$; $n_2 < n_4 < n_6$; で第1番解を得ている。 n_2 を一番軽くできるから、回転が封じられる。対称反転を封じているのは、 $n_4 < n_6$;

n_2 ではなく、 n_1 の位置に一番小さな数字2を置いて n_3, n_7, n_9 よりも軽くしたい時には、正直に $n_1 < n_3$; $n_1 < n_7$; $n_1 < n_9$; と書いて解を選別する。これで、解の「回転」を封じることができる。

$n_1 + n_5 + n_9$ の主対角線を軸とした「反転」を封じたいときには、さらに $n_3 < n_7$; という不等式を用いればよい。

全部合わせれば、 $n_1 < n_3 < n_7$; $n_1 < n_9$; でよいわけである。

第2節： 補数対称型の三方陣を作る

1. 次のプログラム・リストは、最初から「補数対称三方陣」を目指して設計・構成し、標準解1個のみを出力させるものだ。今度は n_1, n_2, n_3, \dots という順序で数値を決める。それで、前には無かったリスト整形用の不等式を用いている。 $n_1 < n_3 < n_7$; $n_1 < n_9$; だ。

```

/** Make the Self-Complementary MS33 and Transform **
/** File:'SCMS33Trnsf.c' Built by Kanji Setsuda **
/** on Jan.20, 2005 and August 16, 2013 **
/** with MacOSX 10.8.4 and Xcode 4.6.3 **

```

```

/**
/** Definitions: Basic Form and
/** .---.---.---. Simultaneous Equations:
/** |n1|n2|n3| n1+n2+n3=C ... eq1;
/** |---+---+---| n4+n5+n6=C ... eq2;
/** |n4|n5|n6| n7+n8+n9=C ... eq3;
/** |---+---+---| n1+n4+n7=C ... eq4;
/** |n7|n8|n9| n2+n5+n8=C ... eq5;
/** |'---'---'---'| n3+n6+n9=C ... eq6;
/** | n1+n5+n9=C ... eq7;
/** | n3+n5+n7=C ... eq8;
/** | C=15; n5=5 ... eq9~eq10(Constant Values);
/** 'Self-Complementary' Conditions:
/** n1+n9=10; n2+n8=10; n3+n7=10; n4+n6=10 ... eq11~eq14;
/** 'List Forming' Inequality Conditions:
/** n1<n9; n1<n3<n7;... cdt1~cdt2;
/**
#include <stdio.h>
/**
short cnt, lsm;
short nm[10], tnm[10];
short flg[10];
short ans[9][10];
/**
void stpin01(void), stpin02(void), stpin03(void), stpin04(void);
void recordans(void), ansprint1(void), prtrns(void);
void reflct(void), rotat90(void), trnsf(void);
/**
/** Main Program *
int main(){
short n;
printf("\n");
printf("*** Self-Complementary Magic Squares of Order 3 **\n");
printf(" ** List of the only Standard Solution **\n");
for(n=0;n<10;n++){flg[n]=0;} /* Reset All UsedFlags *
lsm=15; cnt=0; /* Reset All Counters *
nm[5]=5; flg[5]=1; /* Set n5=5 *
stpin01(); /* Begin the Calculations *
printf(" [Count = %d] OK!\n",cnt);
printf("\n");
printf("* Transform by Rotations and Reflection *\n");
trnsf();
prtrns();
printf(" [Count = %d] OK!\n",8);
printf("\n");
printf("*** Calculated and Listed by Kanji Setsuda\n");
printf(" on on Jan.20, 2005 and Aug.16, 2013;\n");
printf(" with MacOSX 10.8.4 and Xcode 4.6.3 **\n");
printf("\n");
return 0;
}
/**
/** Begin the Calculations *
/** Set n1 & n9 & n1<n9 *
void stpin01(){
short i,j;
for(i=1;i<5;i++){j=10-i;
if((flg[i]==0)&&(flg[j]==0)){
nm[1]=i; nm[9]=j;
flg[i]=1; flg[j]=1;
}
}
}

```

```

        stpin02();
        flg[j]=0; flg[i]=0;
    }
}
}
//
/* Set n2 & n8 *
void stpin02(){
    short i,j;
    for(i=1;i<10;i++){j=10-i;
        if((flg[i]==0)&&(flg[j]==0)){
            nm[2]=i; nm[8]=j;
            flg[i]=1; flg[j]=1;
            stpin03();
            flg[j]=0; flg[i]=0;
        }
    }
}
//
/* Set n3=lsm-n1-n2 & n7 & n1<n3 & n3<n7 *
void stpin03(){
    short i,j;
    i=lsm-nm[1]-nm[2]; j=10-i;
    if((nm[1]<i)&&(i<j)){
        if((flg[i]==0)&&(flg[j]==0)){
            nm[3]=i; nm[7]=j;
            flg[i]=1; flg[j]=1;
            stpin04();
            flg[j]=0; flg[i]=0;
        }
    }
}
//
/* Set n4=lsm-n1-n7 & n6 *
void stpin04(){
    short i,j;
    i=lsm-nm[1]-nm[7]; j=10-i;
    if((0<i)&&(i<10)){
        if((flg[i]==0)&&(flg[j]==0)){
            nm[4]=i; nm[6]=j;
            flg[i]=1; flg[j]=1;
            ansprint1();
            flg[j]=0; flg[i]=0;
        }
    }
}
//
void ansprint1(){
    short n;
    cnt++; nm[0]=cnt;
    printf("[%d]\n",nm[0]);
    printf("----.\n");
    printf("%2d %2d %2d |\n",nm[1],nm[2],nm[3]);
    printf("|---+---+---|\n");
    printf("%2d %2d %2d |\n",nm[4],nm[5],nm[6]);
    printf("|---+---+---|\n");
    printf("%2d %2d %2d |\n",nm[7],nm[8],nm[9]);
    printf("'---'---'---'\n");
    for(n=0;n<10;n++){ans[0][n]=nm[n];}
}

```

```

//
/* Transform by Rotations and Reflection *
void trnsf(){
short i,j;
for(j=1;j<10;j++){nm[j]=ans[0][j];}
for(i=1;i<4;i++){
rotat90();
for(j=1;j<10;j++){ans[i][j]=tnm[j];}
ans[i][0]=i+1;
}
for(j=1;j<10;j++){nm[j]=ans[0][j];}
reflct();
for(j=1;j<10;j++){ans[4][j]=tnm[j];}
ans[4][0]=5;
for(i=5;i<8;i++){
rotat90();
for(j=1;j<10;j++){ans[i][j]=tnm[j];}
ans[i][0]=i+1;
}
}
//
/* Mirror Reflection | Rotation by 90 degrees *
// .-.-.-.-. m1=n1; | m1=n7; .-.-.-.-.
// ln1ln2ln3| m2=n4; | m2=n4;ln7ln4ln1|
// |---+---+---| m3=n7; | m3=n1;|---+---+---|
// ln4ln5ln6| m4=n2; | m4=n8;ln8ln5ln2|
// |---+---+---| m5=n5; | m5=n5;|---+---+---|
// ln7ln8ln9| m6=n8; | m6=n2;ln9ln6ln3|
// '---'---'---' m7=n3; | m7=n9;'---'---'---'
// \ m8=n6; | m8=n6;
// .-.-.-.-. m9=n9; | m9=n3;
// ln1ln4ln7|
// |---+---+---|
// ln2ln5ln8|
// |---+---+---|
// ln3ln6ln9|
// '---'---'---'
//
/* Mirror Reflection *
void reflct(){
short n;
tnm[1]=nm[1]; tnm[2]=nm[4]; tnm[3]=nm[7];
tnm[4]=nm[2]; tnm[5]=nm[5]; tnm[6]=nm[8];
tnm[7]=nm[3]; tnm[8]=nm[6]; tnm[9]=nm[9];
for(n=1;n<10;n++){nm[n]=tnm[n];}
}
//
/* Rotation by 90 degrees *
void rotat90(){
short n;
tnm[1]=nm[7]; tnm[2]=nm[4]; tnm[3]=nm[1];
tnm[4]=nm[8]; tnm[5]=nm[5]; tnm[6]=nm[2];
tnm[7]=nm[9]; tnm[8]=nm[6]; tnm[9]=nm[3];
for(n=1;n<10;n++){nm[n]=tnm[n];}
}
//
/* Print the Transformed Objects
void prtrns(){
short l,l4,m;
for(l=0;l<2;l++){l4=l*4;

```

```

printf("%14d/ ->%10d/ ->%10d/ ->%10d/\n",
      ans[14][0],ans[14+1][0],ans[14+2][0],ans[14+3][0]);
for(m=0;m<4;m++){printf(" .----.----.----.");}; printf("\n");
for(m=0;m<4;m++){
  printf("  !%2d !%2d !%2d !",ans[14+m][1],ans[14+m][2],ans[14+m][3]);};
printf("\n");
for(m=0;m<4;m++){printf(" |----+----+----|");}; printf("\n");
for(m=0;m<4;m++){
  printf("  !%2d !%2d !%2d !",ans[14+m][4],ans[14+m][5],ans[14+m][6]);};
printf("\n");
for(m=0;m<4;m++){printf(" |----+----+----|");}; printf("\n");
for(m=0;m<4;m++){
  printf("  !%2d !%2d !%2d !",ans[14+m][7],ans[14+m][8],ans[14+m][9]);};
printf("\n");
for(m=0;m<4;m++){printf(" '---'---'---'");}; printf("\n");
printf("\n");
}
}
//

```

* 補数対称型三方陣の標準解 1 個 *

[1]

2	9	4
7	5	3
6	1	8

[Count = 1] OK!

* 回転・反転による変換の実際 *

	1/ →	2/ →	3/ →	4/																																				
<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>2</td><td>9</td><td>4</td></tr> <tr><td>7</td><td>5</td><td>3</td></tr> <tr><td>6</td><td>1</td><td>8</td></tr> </table>	2	9	4	7	5	3	6	1	8	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>6</td><td>7</td><td>2</td></tr> <tr><td>1</td><td>5</td><td>9</td></tr> <tr><td>8</td><td>3</td><td>4</td></tr> </table>	6	7	2	1	5	9	8	3	4	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>8</td><td>1</td><td>6</td></tr> <tr><td>3</td><td>5</td><td>7</td></tr> <tr><td>4</td><td>9</td><td>2</td></tr> </table>	8	1	6	3	5	7	4	9	2	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>4</td><td>3</td><td>8</td></tr> <tr><td>9</td><td>5</td><td>1</td></tr> <tr><td>2</td><td>7</td><td>6</td></tr> </table>	4	3	8	9	5	1	2	7	6	
2	9	4																																						
7	5	3																																						
6	1	8																																						
6	7	2																																						
1	5	9																																						
8	3	4																																						
8	1	6																																						
3	5	7																																						
4	9	2																																						
4	3	8																																						
9	5	1																																						
2	7	6																																						
↓	5/ →	6/ →	7/ →	8/																																				
<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>2</td><td>7</td><td>6</td></tr> <tr><td>9</td><td>5</td><td>1</td></tr> <tr><td>4</td><td>3</td><td>8</td></tr> </table>	2	7	6	9	5	1	4	3	8	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>4</td><td>9</td><td>2</td></tr> <tr><td>3</td><td>5</td><td>7</td></tr> <tr><td>8</td><td>1</td><td>6</td></tr> </table>	4	9	2	3	5	7	8	1	6	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>8</td><td>3</td><td>4</td></tr> <tr><td>1</td><td>5</td><td>9</td></tr> <tr><td>6</td><td>7</td><td>2</td></tr> </table>	8	3	4	1	5	9	6	7	2	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>6</td><td>1</td><td>8</td></tr> <tr><td>7</td><td>5</td><td>3</td></tr> <tr><td>2</td><td>9</td><td>4</td></tr> </table>	6	1	8	7	5	3	2	9	4	
2	7	6																																						
9	5	1																																						
4	3	8																																						
4	9	2																																						
3	5	7																																						
8	1	6																																						
8	3	4																																						
1	5	9																																						
6	7	2																																						
6	1	8																																						
7	5	3																																						
2	9	4																																						

[Count = 8] OK!

** Calculated and Listed by Kanji Setsuda
on on Jan. 20, 2005 and Aug. 16, 2013;
with MacOSX 10. 8. 4 and Xcode 4. 6. 3 **

プログラム中「回転」や「反転」の変換法も記述して使用している。どういう風にやるのか、よく読んでほしい。他の次数でも、同様に使える方法だからだ。
整形不等式は、昔からこれに決まっているわけではない。不等号の向きは逆でもよい。

n_1, n_3, n_7, n_9 で整理したいのならば, $n_1 < n_3 < n_7; n_1 < n_9$; だが,
 n_2, n_4, n_6, n_8 で整理したいのならば, $n_2 < n_4 < n_6; n_2 < n_8$; がよい。
 この辺は, 各作成者の裁量範囲なので, 小生はあえて押し付けない。

2. 何故 1 は n_2, n_4, n_6, n_8 以外には, 来ることができないのか。

不思議な感じがするが, それが三方陣の特質なのだ。背理法で証明しておこう。

奇数・偶数の組み合わせで論じる。

行列の定和である 15 は, 奇数である。よって行列を構成する 3 数は, {奇数, 奇数, 奇数}, {奇数, 偶数, 偶数} の組み合わせのいずれかでなくてはならない。例えば {奇数, 奇数, 偶数} では, 和が偶数になって, 15 にはなり得ない。

いま $n_1 = 1; n_5 = 5$; とおくと, 1 も 5 も奇数であるから, n_9 は奇数でなくてはならない。なぜなら, $n_1 + n_5 + n_9 = 15$; だから。

<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>n1</td><td>n2</td><td>n3</td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td>n4</td><td>n5</td><td>n6</td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td>n7</td><td>n8</td><td>n9</td></tr> </table>	n1	n2	n3				n4	n5	n6				n7	n8	n9	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>1</td><td>n2</td><td>n3</td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td>n4</td><td>5</td><td>n6</td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td>n7</td><td>n8</td><td>n9</td></tr> </table>	1	n2	n3				n4	5	n6				n7	n8	n9	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>奇</td><td>n2</td><td>n3</td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td>n4</td><td>奇</td><td>n6</td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td>n7</td><td>n8</td><td>n9</td></tr> </table>	奇	n2	n3				n4	奇	n6				n7	n8	n9	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>奇</td><td>偶</td><td>偶</td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td>偶</td><td>奇</td><td>n6</td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td>偶</td><td>n8</td><td>奇</td></tr> </table>	奇	偶	偶				偶	奇	n6				偶	n8	奇	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>奇</td><td>偶</td><td>偶</td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td>偶</td><td>奇</td><td>偶</td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td>偶</td><td>偶</td><td>奇</td></tr> </table>	奇	偶	偶				偶	奇	偶				偶	偶	奇
n1	n2	n3																																																																													
n4	n5	n6																																																																													
n7	n8	n9																																																																													
1	n2	n3																																																																													
n4	5	n6																																																																													
n7	n8	n9																																																																													
奇	n2	n3																																																																													
n4	奇	n6																																																																													
n7	n8	n9																																																																													
奇	偶	偶																																																																													
偶	奇	n6																																																																													
偶	n8	奇																																																																													
奇	偶	偶																																																																													
偶	奇	偶																																																																													
偶	偶	奇																																																																													

また n_2, n_3 は, いずれも偶数でなければならない。 $n_1 + n_2 + n_3 = 15$; だから。

同様に n_4, n_7 も, 偶数でなければならない。 $n_1 + n_4 + n_7 = 15$; だから。

残りの n_6, n_8 も, 偶数となる。 $n_4 + n_5 + n_6 = 15; n_2 + n_5 + n_8 = 15$; だから。

ところが, それでは最右図でわかる通り, 奇数が 3 個と偶数が 6 個要ることになってしまう。しかるに用いる自然数は, 厳密に 1, 2, 3, 4, 5, 6, 7, 8, 9 でなければならない。奇数が 5 個, 偶数が 4 個しか無い。これでは, 魔方陣の作りようがない。

かかる不合理は, $n_1 = 1; n_5 = 5$; としたために起こったのであるから, この最初の仮定は破棄しなければならない。つまり, n_1 に 1 は来れないのである。

n_3, n_7, n_9 にも 1 が来れないのは, おわかりだろう。回転させれば, どれも n_1 の位置に持って来れるからである。 $n_5 = 5$ は, どうやっても動かさない。

よって, 数値 1 は, $\{n_2, n_4, n_6, n_8\}$ のいずれかに登場せざるを得ない。

3. そうとわかってみれば, 後はもうコンピュータを動かさずとも, 手計算でできるというものだ。試みに下の最左図を出発点として, 4 隅の数字を決定しよう。

使う条件は, 次の補数対称型の定義式である。つまり, 5 を挟む 2 数を「10の補数」にすればよいのである。

$$n_1 + n_9 = n_2 + n_8 = n_3 + n_7 = n_4 + n_6 = 10$$

<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>偶</td><td>9</td><td>偶</td></tr> <tr><td>7</td><td>5</td><td>3</td></tr> <tr><td>偶</td><td>1</td><td>偶</td></tr> </table>	偶	9	偶	7	5	3	偶	1	偶	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>8</td><td>9</td><td>偶</td></tr> <tr><td>7</td><td>5</td><td>3</td></tr> <tr><td>偶</td><td>1</td><td>2</td></tr> </table>	8	9	偶	7	5	3	偶	1	2	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>6</td><td>9</td><td>偶</td></tr> <tr><td>7</td><td>5</td><td>3</td></tr> <tr><td>偶</td><td>1</td><td>4</td></tr> </table>	6	9	偶	7	5	3	偶	1	4	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>4</td><td>9</td><td>偶</td></tr> <tr><td>7</td><td>5</td><td>3</td></tr> <tr><td>偶</td><td>1</td><td>6</td></tr> </table>	4	9	偶	7	5	3	偶	1	6	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>2</td><td>9</td><td>4</td></tr> <tr><td>7</td><td>5</td><td>3</td></tr> <tr><td>6</td><td>1</td><td>8</td></tr> </table>	2	9	4	7	5	3	6	1	8
偶	9	偶																																															
7	5	3																																															
偶	1	偶																																															
8	9	偶																																															
7	5	3																																															
偶	1	2																																															
6	9	偶																																															
7	5	3																																															
偶	1	4																																															
4	9	偶																																															
7	5	3																																															
偶	1	6																																															
2	9	4																																															
7	5	3																																															
6	1	8																																															
スタート	(1)	(2)	(3)	(4)																																													

(1) 先ず $n_1 = 8$ を仮定する。 $n_9 = 2$ となる。

だが, これは不可能である。最上行の和がすでに $9 + 8$ だけで 15 を超えている。

(2) $n_1 = 6$ と仮定する。 $n_9 = 4$ となる。

これも不可能である。最上行の和がすでに $9 + 6$ だけで 15 になる。 n_3 に 0 を入れることはできない。使える数は, 1 以上の自然数なのだから。

(3) $n_1 = 4$ と仮定する。 $n_9 = 6$ となる。残りの {2, 8} は, 残りの $\{n_3, n_7\}$ に入れるのだが, 次の条件式を満たすように配置する。

$$n1+n2+n3=15; n3+n6+n9=15; n1+n4+n7=15; n7+n8+n9=15;$$

だが、どう配置しても、矛盾が生じてしまう。不可能とわかる。

(4) $n1=2$ と仮定する。 $n9=8$ となる。残りの $\{4, 6\}$ を残りの $\{n3, n7\}$ に入れるのだが、次の条件式を満たすように配置する。

$$n1+n2+n3=15; n3+n6+n9=15; n1+n4+n7=15; n7+n8+n9=15;$$

$$n3=4; n7=6; \text{とすると、うまくゆくことがわかる。}$$

結局のところ、これが唯一の正解である。

何故他の7解が出て来ないのか。不思議に思われるかもしれない。

これは、最初のスタートに理由がある。不等式 $n1 < n3 < n7; n1 < n9;$ が課せられているからで、標準解のみが求まるのだ。

第3節：汎対角線型の三方陣を作ることはできない

汎三方陣は構成が不可能であることが既にわかっている。その論証を再録する。

汎三方陣と言うのは、「汎対角線」を持った三方陣のことだが、この性質は、もともと方陣の行列が1行1列ずつ移動可能であるという約束から来ている。

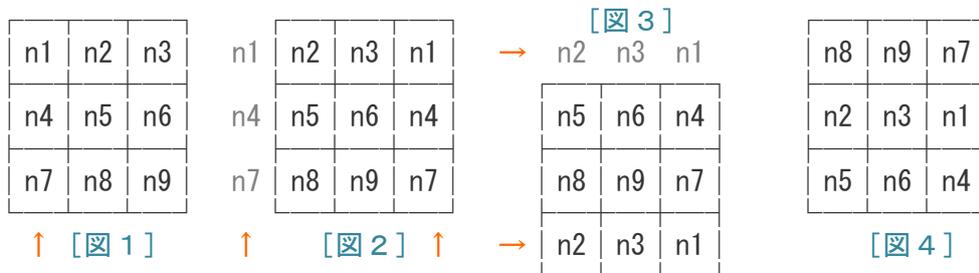


図1の最左列を最右列の右側に持って来て、新しい方陣[図2]を作る。こうしても行列の和と対角線3数の定和が成り立つものを「汎三方陣」と言う。図2で

$$n1+n6+n8=C \text{ および } n2+n6+n7=C \text{ が成り立たなければならない。}$$

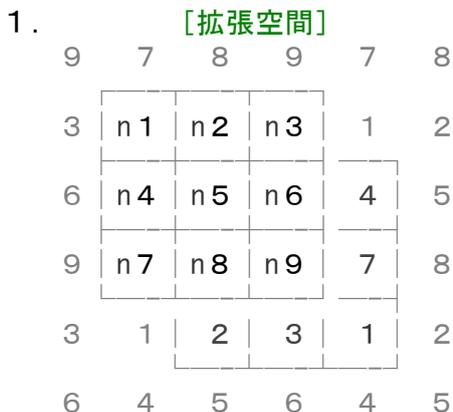
また図2の最上行を最下行の下に持って来て新たに作った方陣が図3である。ここでもまた行列の定和と新対角線の定和が成り立つ。そういうものを「汎三方陣」と言う。つまり、

$$n5+n9+n1=C \text{ および } n4+n9+n2=C \text{ が成り立たなければならない。}$$

或いは、図2の最下行を最上行の上に持って来た図4でも、

$$n8+n3+n4=C \text{ および } n7+n3+n5=C \text{ が成り立たなければならない。}$$

移動に伴って出来るこれら新対角線のことを「汎対角線」と言う。汎魔方陣の名の由来である。いったい汎対角線は、何本あるのであろうか。次のような拡張空間を用意し、連立方程式によって記述しよう。



下図のように「拡張空間」を作成する。拡張空間は、方陣の行列の内容をただ1行1列ずつ左右上下にコピーしただけのものだが、いろいろ便利に使える。行列の逐次移動の代わりに方陣枠の方をこの拡張空間の中で移動させても、同じ結果が得られる。派生方陣を見つけるのに便利だ。汎対角線も見やすい。

$n1$ を通る対角線は、
 $n1+n5+n9=C$ だけでなく、
 $n1+n6+n8=C$ もある。
 $n2$ を通る対角線は、
 $n2+n6+n7=C$
 $n2+n4+n9=C$ の2本ある。……

各数についてそれぞれ2本ずつあるから、結局 $9 \times 2 = 18$ 本あることになる。ところが、1つの対角線を3数が必ず共有しているから、重複を省くと、 $18 \div 3 = 6$ 本の汎対角線がある勘定になる。この中には「主対角線」も含まれる。

2. 定義の連立方程式をあらためて立てる。

[(7) ~ (12) は、汎対角線の定和式である。]

$$\begin{array}{l|l}
 n_1 + n_2 + n_3 = C \dots (1) & n_1 + n_5 + n_9 = C \dots (7) \\
 n_4 + n_5 + n_6 = C \dots (2) & n_1 + n_6 + n_8 = C \dots (8) \\
 n_7 + n_8 + n_9 = C \dots (3) & n_2 + n_6 + n_7 = C \dots (9) \\
 n_1 + n_4 + n_7 = C \dots (4) & n_2 + n_4 + n_9 = C \dots (10) \\
 n_2 + n_5 + n_8 = C \dots (5) & n_3 + n_4 + n_8 = C \dots (11) \\
 n_3 + n_6 + n_9 = C \dots (6) & n_3 + n_5 + n_7 = C \dots (12)
 \end{array}$$

もちろん三方陣の一般的な性質である $C = 15$ および補数和の $CC = 10$ 、さらに $n_5 = 5$ は、そのまま受け継ぐものとする。

3. 汎三方陣は構成不可能であることが知られている。その理由を背理法によって証明しよう。

$$\begin{array}{l}
 (1) \text{式から } n_2 + n_3 = CC - n_1, \quad (4) \text{式から } n_4 + n_7 = CC - n_1, \\
 (7) \text{式から } n_5 + n_9 = CC - n_1, \quad (8) \text{式から } n_6 + n_8 = CC - n_1 \\
 \text{ゆえに } n_2 + n_3 = n_4 + n_7 = n_5 + n_9 = n_6 + n_8 = CC - n_1 \dots (13) \\
 \text{同様に } n_1 + n_3 = n_4 + n_9 = n_5 + n_8 = n_6 + n_7 = CC - n_2 \dots (14) \\
 n_1 + n_2 = n_4 + n_8 = n_5 + n_7 = n_6 + n_9 = CC - n_3 \dots (15) \\
 n_1 + n_7 = n_2 + n_9 = n_3 + n_8 = n_5 + n_6 = CC - n_4 \dots (16) \\
 n_1 + n_9 = n_2 + n_8 = n_3 + n_7 = n_4 + n_6 = CC - n_5 \dots (17) \\
 n_1 + n_8 = n_2 + n_7 = n_3 + n_9 = n_4 + n_5 = CC - n_6 \dots (18) \\
 n_1 + n_4 = n_2 + n_6 = n_3 + n_5 = n_8 + n_9 = CC - n_7 \dots (19) \\
 n_1 + n_6 = n_2 + n_5 = n_3 + n_4 = n_7 + n_9 = CC - n_8 \dots (20) \\
 n_1 + n_5 = n_2 + n_4 = n_3 + n_6 = n_7 + n_8 = CC - n_9 \dots (21)
 \end{array}$$

これらの式は、それぞれ異なる系列の補数4ペアを要請している。しかも、各式の内部では同じ数値の重複使用は許されていない。

つまり、汎三方陣は、相互に和が異なる補数4ペアを9セットも必要としている。

ところが1~9の自然数を重複なく組み合わせて実際に出来る補数のペアをチェックしてみると、9セットも無いことがわかる。和によって分類した表を掲げる。

和：補数セット

2:

3: (1, 2)

4: (1, 3)

5: (1, 4) (2, 3)

6: (1, 5) (2, 4)

7: (1, 6) (2, 5) (3, 4)

8: (1, 7) (2, 6) (3, 5)

9: (1, 8) (2, 7) (3, 6) (4, 5)

10: (1, 9) (2, 8) (3, 7) (4, 6)

11: (2, 9) (3, 8) (4, 7) (5, 6)

12: (3, 9) (4, 8) (5, 7)

13: (4, 9) (5, 8) (6, 7)

14: (5, 9) (6, 8)

15: (6, 9) (7, 8)

16: (7, 9)

17: (8, 9)

18:

4ペアも出来るのは、和が9, 10, 11の場合の3通りしかない。よって、1~9の自然数を重複なく用いて(13)~(21)式の全てを満たすことはできない。つまり、「汎三方陣」を構成することは不可能である。
(Q. E. D.)

(初稿: May 26, 2000;

改定稿: December 15, 2000;)

(改訂新稿: January 22, 2005;

清書稿: August 16, 2013;

MacOSX 10.8.4 and Xcode 4.6.3)

著者: 摂田 寛二(Kanji Setsuda)

Email Address: <jag12001@nifty.com>