『魔方陣の基礎的研究』： 摂田 寛二

序論： 魔方陣研究の基礎

基礎論こそは，本来その道の大家が書くべきものである。小生のごとき市井の一介の初心者が書くべきものでないことは，重々わかっている(つもりである)。
だが，これといったものがほとんど見られない状況では，今更きけない基本的な疑問がいつまでもそのままに残される。専門家の間でも良い議論が起こらない。たとえ起きても，声高に独断論を主張し合うばかりで，全然噛み合わない。
生産的な論争をあえて起こすためにも，いわばその「たたき台」となる基本的な議論を小生が僭越を承知の上で提示して，一般の注意を喚起したいと思う。

小生，魔方陣の学問的研究をかって志すも，最初は何をどう研究していいのか，さっぱりわからなかった。調べても五里霧中で，ずいぶんと当惑したものだ。
魔方陣の研究史は相当古いのに，先人たちの研究は，対象も方法も成果もてんでんバラバラな印象で，統一感がない。学界にまとまった，厳密で決定的な著作がまだ乏しい感じで，学問の形成はどうやら緒に就いたばかりという印象を受けた。
顧みれば，小生の最初の教科書は，阿部楽方老師の共著書『方陣の研究』だった。何を研究したらよいのかをそれで教わった。その後は，鈴木睦前教授のホームページが教本になった。ネットサーフィンして，研究の新しい潮流や世界の動静を感覚的に把握した。御両所の格別の厚遇を得た小生は，真に幸運な学徒だったと言える。
だが，幸運が全てを順調に運ぶとは限らない。不思議を不思議と思う知性には，他人にはわからない独自の苦難が待ち受けている。いざ研究を自分で進めて見ると，すぐに様々な疑問にぶつかった。何をどうすべきか，迷うことが多かった。トライ＆エラー。自分で大胆に選択し，失敗を恐れず進んでみた。失敗したら，元に戻る。そしてまた進む。
試行錯誤は，自分の足で歩み始めた誰もが通る自然の道筋である。
理論上の深刻な疑問は，鈴木前教授にメールで相談して解決した。しかし，どんな偉い人に相談しようとも，中には相手が実際にそれで苦労した経験が無いとすぐには答えられない問題というものがある。結局，認識の食い違いも生じた。最終的には「見解の相違」・「趣味の選択」という結論になる問題さえも，残念ながら二，三はあった。
自分だけが抱えた難問は，結局自力で解決するしかなかった。そんな時は，数学的に自明の事柄だけを組み合わせて前提とし，粘り強く推理・推論して決めた。
最終的には，何をどう研究するか課題と方法そのものを自分で発明・発見するしかない場面にも遭遇した。先端に出てしまうと，もう誰にも教わることができない。
そういう時には，不遜にも「伝統」や「権威」に従うことを敢えて拒否した。

ここで，小生が苦労した諸問題のうち基礎的で一番微妙な問題だと思うものを，少し詳しく具体的に論じておくことにしよう。

第１章　何を研究するのか

「魔方陣とは何か」の導入は，もうしない。それがどんなものなのかは，既に概略を知っているという前提で話を進める。
研究全体の進み具合から見て，まだ「総論」を書くべき時期には来ていないと，小生は考える。やはり次元数・次数を特定して議論する「各論」を，まだ引き続き辛抱強く積み重ねなければならない必要があると思う。

１．どんな魔方陣を研究するのか

世界中には，様々なタイプの魔方陣を研究している人が多数いる。古くからのスペシャリストもいれば，ニュータイプの発見・発明者という人もいる。インターネットを見て回ると，驚かされるページが多い。何という多様で豊かな世界なのだろうか。

魔方陣のタイプを大きく括れば，基本方陣と応用方陣の２つに分類できるだろう。

（１）「**基本方陣**」は，昔からある三次から九次ぐらいの正方形あるいは立方体の魔方陣で，構造は「標準型」・「汎対角線型」・「補数対称型」(それに「正方連結型」(「相結陣」とも言う)を加えてもよいと小生は考える)。古典方陣の場合は，いずれも１から始まる連続した自然数を用いる(近代方陣では，０から始まる整数列を用いる)。演算は加法のみを採用し，各部の定和を求めるオーソドックスな条件が付く。昔から良く研究されてきた分野だ(と誰もが思っていた)。

（２）「**応用方陣**」は，その他のものすべてを含む。たいていは，ユニークな形と仕掛けを持ち，使う数列も独特なら，条件式も独特だ。「阿部方陣」や「力石方陣」など作者の個性が色濃く出たアート的作品が多い。力作は，たいてい発明・発見者の名を取って紹介される例が多い。

小生はてぇと，**基本方陣**のスペシャリストだ。昔から知られていて，特に誰が作ったとも言われない，いわば「神の造りたまいし」古典的な方陣ばかりを，専ら古典的な条件下でのみ研究している。そうした方陣群がどんな「世界」を作っているのかをただ知りたいと，常々思っているからに他ならない。

基本方陣は，名にし負う大家たちによって昔から研究され尽くした分野と誰にも思われて来たが，小生にはどうしてもそう思えなかった。むしろ「ほとんど手つかずの沃地」とさえ思った。これは，小生独自の直観であるが，我々は万物創造神の造りたまいし「仕掛け」の様々な秘密をまだ十分に掘り起こしていないのではないかと，密かに感じている。

２．何を研究するのか

各基本方陣について神の造りし「世界」を解き明かすとは，どういうことか。

一言で言えば，その世界を支配する様々な美しい法則を発見するということ。その発見報告ができれば，それだけで最高の幸せだと心底思っている。

具体的には，各タイプの方陣の解集合を集めて，その総数・相互関連・有意味な構造を知りたい。他のタイプとの緊密な対応関係も，知りたいのである。

基本方陣の場合，一個や二個の作例を報告するだけでは，今は「新発見」とは言われない。コンピュータによってそのタイプの総数が簡単に決定できる時代になったからだ。

自らの手でできる新しい製作・構成法を見つけた場合ならば有望だが，すぐに大騒ぎはできない。それでそのタイプの方陣の全数を作れないような方法では，価値はだいぶ下がるから。もし数個の製法をまとめて使えば全数を作ることができるという発見であれば，皆に報告してみる価値はあると思う。解集合の構造解析に役に立つかもしれないから。いろいろテストして大知識に化ける可能性が無いとは決して言い切れない。

方陣解の「相互関連」を知るには，先ず解をいくつかのグループに分類してみるのがよい。グループ間の相似な特徴を発見できれば，ゴールは確実に近づいていると言える。

「基本形」と「変換法」を考え，それによって解の総数を再構成するという工夫も試みるとよい。出来上がれば，もうそれは立派な「構造解析」になるから。

例えば，Ｎ進法表現を用いて，基本方陣の解析・構成を試みるのもよい。

その研究法・構成法が全体にあまねく通用するかを，系統的に次数を上げて次々とテストすることも，重要だ。ぜひ試みて，自分なりの体験的知識を獲得・蓄積されよ。

違ったタイプや違った次元数の魔方陣との相互関連も，見つけるように工夫・努力してみたらよい。だんだんと難しい仕事にはなるが，思わぬ結びつきを発見することができれば，次の大発見・大発明を生む端緒になると思われる。

３．各魔方陣のタイプ名と定義について

何の研究でも構成法でも，成果の報告は，自分の考え方や方法を明記した上で，すべてを包み隠さずに発表すべきである。数学的真理の世界は，実用世界ではない。ここには保護されるべき個人的利益や権利などというものは元々無い。特許の取れる方法などというものもあり得ない。あるのはせいぜい，お金になりそうもない発見の名誉だけだ。

あなたは，知っていただろうか。ノーベル賞に数学部門が無いのは，実は数学者たちの

密かな誇りなのである。たとえ賞創設の裏に贖罪の意味があろうとも，爆薬を作った人から賞金を金輪際貰わねぇてぇのだから，数学者たぁ何とも豪毅な人たちではないか。
　冗談はともかく，問題は定義である。この難題をめぐって，議論がしばしば生じる。
　ご存知の通り，小生の定義は，いつも決まって多元連立方程式である。これは，皆の定義と違う。皆と違ったものを作っているのではないかと，いぶかられる。

先人たちの定義

　例えば,「対称型」。欧米では，オランダのハーベイ・ハインツ老の定義に従って対称型を「補数変換に関して自己相似の魔方陣」と言う人が多い。名前からしてそうなのだから，定義も大概そうする。ちなみに故ハインツ老は，非常に良心的な研究者として世界中から尊敬されていた。彼は平素，誰の研究でもオリジナルを尊重して再録・継承し，先達の名誉をなるべく守ろうと献身・努力しておられた奇特な人だ。
　図で説明すると，こうなる。一番簡単な三次の場合で示そう。

＊対称三方陣図＊

2	9	4
7	5	3
6	1	8

〔原形〕

→

＊補数変換＊

10−2=8;
10−9=1;
10−4=6;
10−7=3;
10−5=5;
10−3=7;
10−6=4;
10−1=9;
10−8=2;

→

＊点対称移動形
あるいは180°回転形＊

8	1	6
3	5	7
4	9	2

［自己相似形］

　元の方陣の各項にそれぞれ補数変換を施す。そして数値を入れ替えて新たな方陣を作る。その方陣は，内容が自分自身と同じである（対称移動しただけ，あるいは180°回転移動しただけ)。そういうふうになるような方陣〔原型〕を求めるのである。
　次のように説明する人もいる。

＊対称三方陣図＊

2	9	4
7	5	3
6	1	8

〔原形〕

＋

＊対称移動＊

8	1	6
3	5	7
4	9	2

［対称形］

＝

＊補数和表＊

10	10	10
10	10	10
10	10	10

［項和表］

　元になる方陣の各項を点対称移動して，新たな方陣を作る。原形と対称移動形の対応する各項を１つ１つ足すと，どの場合も定和の10 になる。そういう方陣を求める。
　ａ＋ｂ＝Ｃ(定数)になるとき，我々は“(ａ，ｂ)をＣの補数”と呼ぶ。あるいは“ａはｂの補数”とも“ｂはａの補数”とも呼ぶ。

　上の２つの説明は，手順が逆になっているが，引き算を使うか足し算を使うかだけの差だ。作ろうとしている方陣の概念は，同じである。そのことにお気づきだろうか。

私流の定義：　補数対称型三方陣の定義式

n1	n2	n3
n4	n5	n6
n7	n8	n9

〔基本図〕

n1+n9=C;
n2+n8=C;
n3+n7=C;
n4+n6=C;
n5+n5=C;
n6+n4=C;
n7+n3=C;
n8+n2=C;
n9+n1=C;

→

n1+n9=n2+n8=
　n3+n7=n4+n6=10;
n5=5;
〔対称方陣条件式〕

この他に次の基本定義式も加える

n1+n2+n3=15
n4+n5+n6=15
n7+n8+n9=15
n1+n4+n7=15
n2+n5+n8=15
n3+n6+n9=15
n1+n5+n9=15
n3+n5+n7=15

　小生の定義は，いつも上図のようにしている。

左の基本図を仮定し，右の条件式を添えて，「補数対称型」の定義としている。概念は同じだ。これで，補数和も対称移動も全部入っている。

何故にこうするのかと言えば，後が楽だからだ。プログラムも簡単に書けるし，条件のチェックも簡単に済む。それでいて，出来上がるものに変わりは無い。全く別のものが出来たりはしない。奇を衒っているわけでも決してない。

数学研究の場合，常に「**オッカムの剃刀**」が厳格に働く。内容が同じならば，なるべく簡単な形式のものが優先的に採用されるのである。

魔方陣を，人類文化史の一つの遺産として追跡するのでなく，あくまで「数学研究」の素材として追究したいというのであれば，なるべく合理的な方法を採用した方が良い。

必ずしもオリジナルの古拙なスタイルを忠実に守らなくてもよいのではないか。

対象によって自ずから決まる方法的定義

小生は，概念と定義を分ける立場を取る。定義は，方法上の便宜を考えてなるべく簡単な形式のものを選んでいる。チェックがしやすいように，プログラムが組みやすいようにしている。

次は，四次の場合で具体的に「汎対角線型魔方陣（**汎魔方陣**)」を説明しよう。

「**汎魔方陣**」の概念は，既に序章で説明した。再述しない。

行列の逐次移動を可能にする条件は，何だろうか。下図のように方陣空間を方陣枠の外まで拡張して調べ，いろいろ研究した結果，次のように定義するのが良いとわかった。

下の基本図を仮定し，２組の連立方程式(全16本)を立てるのである。

＊　汎四方陣の定義式群　＊

```
＃１．四方陣の基本条件式： K＝34;
  n1+n2+n3+n4=K     ...rw1;     n1+n6+n11+n16=K...pd1;
  n5+n6+n7+n8=K     ...rw2;     n4+n7+n10+n13=K...pb4;
  n9+n10+n11+n12=K  ...rw3;
  n13+n14+n15+n16=K ...rw4;
  n1+n5+n9+n13=K    ...cl1;
  n2+n6+n10+n14=K   ...cl2;
  n3+n7+n11+n15=K   ...cl3;
  n4+n8+n12+n16=K   ...cl4;
```

［基本図：拡張方陣空間］

```
＃２．汎四方陣の汎対角線定和式： K＝34;
  n1+n6+n11+n16=K   ...pd1;     n1+n8+n11+n14=K   ...pb1;
  n2+n7+n12+n13=K   ...pd2;     n2+n5+n12+n15=K   ...pb2;
  n3+n8+n9+n14=K    ...pd3;     n3+n6+n9+n16=K    ...pb3;
  n4+n5+n10+n15=K   ...pd4;     n4+n7+n10+n13=K   ...pb4;
```

これで，行列の逐次移動が可能になる。汎四方陣を設計できる。

こうしたスタイルの定義は，方陣そのものの性質によって決定される。今後使う方法によっても，決定される。いわば概念を表現するための「方法的定義」と言える。

「完全四方陣」や「正方連結完全四方陣」も，同様に連立方程式で定義する。
連立方程式は，プログラムに組み込みやすい最良の方法だと，小生は信じている。

第２章　どう研究するのか

先ずは，目的の方陣を作成する方法だが，小生はパーソナル・コンピュータを積極的に活用している。プログラムを自分で組んで，構成・計数・チェック・作表・検索など，何でも我が愛機 Power Mac にやらせている。
三次・四次くらいなら手計算でもできるが，五次以上ともなると，手計算は正直に言って，辛い。立方体は，三次でも辛い。何しろ，解の全数を数えようというのだから。
高次の魔方陣では，調べる「場合の数」が急激に増える。解の総数も多い。計数に一生をかけても，まだ時間が足りない。それに人間は，疲れると，簡単な計算をすぐに間違う。間違いをチェックする時間が，またかかる。その上に報酬が無いと，誰もこの単調な作業をやりたがらない。
望遠鏡を使わない天文観測など，今は考えられない。コンピュータを使わずデータを高速に処理する方法も，もう考えられない。どの研究分野でも，ＩＴ化は必至である。
なのに一流の数学者たちは，何故にコンピュータを使いたがらないのだろう。
魔方陣研究は，純粋な理論数学というより，今はまだ実測を基にする経験科学に近い側面を持っている。有理数も無理数も使わない整数値だけの「離散数学」ではあるけれども，多変数関数に与える条件式が少なめなので，答が予想を超えた多様さになる。今のところ，調査・検証・分類の作業が必要不可欠だ。演繹的理論の出番はまだ少ない。
何しろ，この学際的な数学は，オイラー先生が活躍して以来もう約二百五十年も経つというのに，進歩らしい進歩がほとんど見られない「変な」分野なのである。公式も法則も理論も，まともなものは，片手で数える程度しか見つかっていない。
もしかしたら，基本的なデータ量が足りないのではないか。それで遅れているのではないか。ＰＣでも何でも使って，手持ちのデータ量をどんどん増やしてはどうだろうか。研究材料が増えれば，理論も次々と見つかって来るのではあるまいか。
そんな単純な期待と予感に導かれて，小生は先ずデータの蓄積・分類を心がけた。幸い，ＰＣは近年急激に進化した。大いにそのおかげをこうむる成果を上げることができた。

１．コンピュータによる計数・作表

では，あるタイプの魔方陣の「総数」を求めるには，具体的にどうするか。
汎用の理論がまだ見つからないので，今はまだ実測が必要不可欠だ。理論的にある程度予測はできても，別の有効な手段によって厳密なチェックが現実にできないからだ。
だが，コンピュータにいったいどうやって数えさせたら良いのだろうか。

三次の標準型魔方陣で，やり方の実例を示そう。

＊　三次の標準型魔方陣の定義　＊

［基本図］

n1	n2	n3
n4	n5	n6
n7	n8	n9

［連立方程式群］

```
n1+n2+n3=C ... eq1;
n4+n5+n6=C ... eq2;
n7+n8+n9=C ... eq3;
n1+n4+n7=C ... eq4;
n2+n5+n8=C ... eq5;
n3+n6+n9=C ... eq6;
n1+n5+n9=C ... eq7;
n3+n5+n7=C ... eq8;  [C=15]
```

先ず上のような基本図と連立方程式群で定義をする。もちろん，使う数値は，自然数の１，２，３，...，８，９のみ。どの数値の欠落も重複使用も許されない。
この連立方程式を解くのである。いわゆる「加減法」でもなく「代入法」でもなく，いわんや「掃き出し法」でもなく，一番原始的な「数値検査法」で。

n1，n2，n3，．．．，n8，n9 の各変数に具体的に１，２，３，．．．，８，９の数値を順次欠落無く重複無く入れる。それでもって上の連立方程式が成り立つかどうかを調べる。そして，成り立った場合の各変数の値の組を，求める解とするのである。

（１） まず n1 に自然数１，２，３，．．．，８，９を順に入れる。（２）に進む。
　n1 の値を最後まで試し終わったら，作業を終了する。

（２） 次に n2 に１，２，３，．．．，８，９を順に入れる。ただし，n1 の値と重複しないように注意する。（３）に進む。
　n2 に入れる最後の数値９まで試したならば，（１）に戻って n1 に次の候補を入れる。そして，（２）に進む。

（３） 次に n3 に１，２，３，．．．，８，９を順に入れてゆく。ただし，n1，n2 の値と重複しないように注意する。
　そしてここで条件式 n1+n2+n3=15 ．．． eq1 が成り立つかどうかをチェックする。成り立てば，（４）に進む。
　成り立たなければ，n3 の次の値を試す。
　最後の数値の９まで試して成り立たなければ，（２）に戻る。

（４） 次に n4 に１，２，３，．．．，８，９を順に入れてゆく。ただし，n1，n2，n3 の値と重複しないように注意する。
　．．．（以下同様）．．

こんなふうに煩雑な手順を次々と踏んで成り立つ解を探すのだが，理屈もへったくれも無い，何とも愚直で泥臭い印象を受けるかと思う。しかし，この無技巧の素朴な方法しか今のところ有効な手だてが無いのである。

素朴で愚直な方法だが，全部の可能性を尽くして一々丁寧に調べてくれる。これならばいずれは「総数」を数え尽くすであろうと，誰もが推理・想像することができる。

幸いコンピュータは，こういう素朴だが愚直な作業を，命令すれば快くやってくれる。非常な高速度でやり遂げてしまう。機械がそういうふうに作られているのだ。

プログラムに書くと，こうなる。一番初歩的なスタイルで示そう。

プログラムは，いわばコンピュータに与える命令書だ。仕様（約束）に従ってクドいくらい懇切丁寧に，正確無比に書いてやらなければならない。

```
//** First PPROGRAM #1 for Magic Squares of Order 3 **
//** File: 'MS33First.c' Dictated by Kanji Setsuda  **
//**  on Jan.20, '05, May 17, '14 and Jun. 3, '15  **
//**  working with MacOSX 10.10.3 and Xcode 6.3.2  **
//
#include <stdio.h>
//
short cnt, LSM;
short n1,n2,n3,n4,n5,n6,n7,n8,n9;
short uflg[10];
//
//** Main Program **
int main(){
 short n;
  printf("\n* Magic Squares of Order 3 *\n");
  for(n=0;n<10;n++){uflg[n]=0;}
  LSM=15; cnt=0;
  for(n1=1;n1<10;n1++){
    if(uflg[n1]==0){uflg[n1]=1;
      for(n2=1;n2<10;n2++){
        if(uflg[n2]==0){uflg[n2]=1;
          for(n3=1;n3<10;n3++){
            if((n1+n2+n3==LSM)&&(uflg[n3]==0)){uflg[n3]=1;
              for(n4=1;n4<10;n4++){
                if(uflg[n4]==0){uflg[n4]=1;
                  for(n7=1;n7<10;n7++){
                    if((n1+n4+n7==LSM)&&(uflg[n7]==0)){uflg[n7]=1;
                      for(n5=1;n5<10;n5++){
```

```
                                if((n3+n5+n7==LSM)&&(uflg[n5]==0)){uflg[n5]=1;
                                  for(n6=1;n6<10;n6++){
                                    if((n4+n5+n6==LSM)&&(uflg[n6]==0)){uflg[n6]=1;
                                      for(n8=1;n8<10;n8++){
                                        if((n2+n5+n8==LSM)&&(uflg[n8]==0)){uflg[n8]=1;
                                          for(n9=1;n9<10;n9++){
                                            if((n1+n5+n9==LSM)&&(n3+n6+n9==LSM)&&(n7+n8+n9==LSM)){
                                              uflg[n9]=1;
                                              cnt++;
                                              printf("[%d]\n",cnt);
                                              printf("%3d%3d%3d\n",n1,n2,n3);
                                              printf("%3d%3d%3d\n",n4,n5,n6);
                                              printf("%3d%3d%3d\n",n7,n8,n9);
                                              uflg[n9]=0;}}
                                        uflg[n8]=0;}}
                                    uflg[n6]=0;}}
                                uflg[n5]=0;}}
                            uflg[n7]=0;}}
                        uflg[n4]=0;}}
                    uflg[n3]=0;}}
                uflg[n2]=0;}}
            uflg[n1]=0;}}
  printf("[Count = %d]  OK!\n\n",cnt);
 return 0;
}
//
```

＊ 標準型三方陣の解のリスト ＊

```
[1]          [2]          [3]          [4]
  2  7  6      2  9  4      4  3  8      4  9  2
  9  5  1      7  5  3      9  5  1      3  5  7
  4  3  8      6  1  8      2  7  6      8  1  6
[5]          [6]          [7]          [8]
  6  1  8      6  7  2      8  1  6      8  3  4
  7  5  3      1  5  9      3  5  7      1  5  9
  2  9  4      8  3  4      4  9  2      6  7  2
[Count = 8]  OK!
```

今書いているこの一文は，プログラミングの解説書ではないので，立ち入った詳しい解説はしない。慣れない人にはわかりにくいかもしれないが，一応Ｃ言語で‘ミニPascal’風に書いている。重複検査などに結構高級な技法も使っている。

こういう愚直で粘り強い泥臭い計数法を決してあなどってはいけない。そして，この技法をあなたもぜひ身に付けてほしい。何かという時に役立つからだ。例えば，他に良い方策が浮かばない時，解の総数だけは知りたい時，或いは解の実例をぜひ入手したい時に，あなたのアシスタントとして何かと研究を助けてくれる。

こんな小さなプログラムでも，実行させれば，上のように解の総数の８個をきちんと全部数え尽くして出力する。

この結果があれば，基礎データとして研究者に渡すことができる。

２．データ分析

方陣研究者は，このデータを良く観察して解析する。

まず各解の検査を行う。行列の和が一定になっているか。主対角線や汎対角線の和はどうなっているか。補数ペアの和が一定で，対称的に配置されているか，などを調べる。

＊ 各解の定和になるべき項目を全て検査する ＊

```
[1]
   .---.---.---.  Rows & Columns    Pan-diagonals
 6 | 2 | 7 | 6 | 2    2+7+6=15;     2+5+8=15;
   |---+---+---|      9+5+1=15;     7+1+4=12;
 1 | 9 | 5 | 1 | 9    4+3+8=15;     6+9+3=18;
   |---+---+---|      2+9+4=15;     2+1+3= 6;
 8 | 4 | 3 | 8 | 4    7+5+3=15;     7+9+8=24;
   '---'---'---'      6+1+8=15;     6+5+4=15;
* Complementary Pairs:
    2+8=10;  7+3=10;  6+4=10;  9+1=10;  5+5=10;
  ... This is a Self-complementary Magic Square!
[2]
   .---.---.---.  Rows & Columns    Pan-diagonals
 4 | 2 | 9 | 4 | 2    2+9+4=15;     2+5+8=15;
   |---+---+---|      7+5+3=15;     9+3+6=18;
 3 | 7 | 5 | 3 | 7    6+1+8=15;     4+7+1=12;
   |---+---+---|      2+7+6=15;     2+3+1= 6;
 8 | 6 | 1 | 8 | 6    9+5+1=15;     9+7+8=24;
   '---'---'---'      4+3+8=15;     4+5+6=15;
* Complementary Pairs:
    2+8=10;  9+1=10;  4+6=10;  7+3=10;  5+5=10;
  ... This is a Self-complementary Magic Square!
[3]
   .---.---.---.  Rows & Columns    Pan-diagonals
 8 | 4 | 3 | 8 | 4    4+3+8=15;     4+5+6=15;
   |---+---+---|      9+5+1=15;     3+1+2= 6;
 1 | 9 | 5 | 1 | 9    2+7+6=15;     8+9+7=24;
   |---+---+---|      4+9+2=15;     4+1+7=12;
 6 | 2 | 7 | 6 | 2    3+5+7=15;     3+9+6=18;
   '---'---'---'      8+1+6=15;     8+5+2=15;
* Complementary Pairs:
    4+6=10;  3+7=10;  8+2=10;  9+1=10;  5+5=10;
  ... This is a Self-complementary Magic Square!
[4]
   .---.---.---.  Rows & Columns    Pan-diagonals
 2 | 4 | 9 | 2 | 4    4+9+2=15;     4+5+6=15;
   |---+---+---|      3+5+7=15;     9+7+8=24;
 7 | 3 | 5 | 7 | 3    8+1+6=15;     2+3+1= 6;
   |---+---+---|      4+3+8=15;     4+7+1=12;
 6 | 8 | 1 | 6 | 8    9+5+1=15;     9+3+6=18;
   '---'---'---'      2+7+6=15;     2+5+8=15;
* Complementary Pairs:
    4+6=10;  9+1=10;  2+8=10;  3+7=10;  5+5=10;
  ... This is a Self-complementary Magic Square!
[5]
   .---.---.---.  Rows & Columns    Pan-diagonals
 8 | 6 | 1 | 8 | 6    6+1+8=15;     6+5+4=15;
   |---+---+---|      7+5+3=15;     1+3+2= 6;
 3 | 7 | 5 | 3 | 7    2+9+4=15;     8+7+9=24;
   |---+---+---|      6+7+2=15;     6+3+9=18;
 4 | 2 | 9 | 4 | 2    1+5+9=15;     1+7+4=12;
   '---'---'---'      8+3+4=15;     8+5+2=15;
* Complementary Pairs:
    6+4=10;  1+9=10;  8+2=10;  7+3=10;  5+5=10;
```

```
  ... This is a Self-complementary Magic Square!
[6]
   .---.---.---.  Rows & Columns    Pan-diagonals
 2 | 6 | 7 | 2 | 6    6+7+2=15;     6+5+4=15;
   |---+---+---|      1+5+9=15;     7+9+8=24;
 9 | 1 | 5 | 9 | 1    8+3+4=15;     2+1+3= 6;
   |---+---+---|      6+1+8=15;     6+9+3=18;
 4 | 8 | 3 | 4 | 8    7+5+3=15;     7+1+4=12;
   '---'---'---'      2+9+4=15;     2+5+8=15;
* Complementary Pairs:
    6+4=10;  7+3=10;  2+8=10;  1+9=10;  5+5=10;
  ... This is a Self-complementary Magic Square!
[7]
   .---.---.---.  Rows & Columns    Pan-diagonals
 6 | 8 | 1 | 6 | 8    8+1+6=15;     8+5+2=15;
   |---+---+---|      3+5+7=15;     1+7+4=12;
 7 | 3 | 5 | 7 | 3    4+9+2=15;     6+3+9=18;
   |---+---+---|      8+3+4=15;     8+7+9=24;
 2 | 4 | 9 | 2 | 4    1+5+9=15;     1+3+2= 6;
   '---'---'---'      6+7+2=15;     6+5+4=15;
* Complementary Pairs:
    8+2=10;  1+9=10;  6+4=10;  3+7=10;  5+5=10;
  ... This is a Self-complementary Magic Square!
[8]
   .---.---.---.  Rows & Columns    Pan-diagonals
 4 | 8 | 3 | 4 | 8    8+3+4=15;     8+5+2=15;
   |---+---+---|      1+5+9=15;     3+9+6=18;
 9 | 1 | 5 | 9 | 1    6+7+2=15;     4+1+7=12;
   |---+---+---|      8+1+6=15;     8+9+7=24;
 2 | 6 | 7 | 2 | 6    3+5+7=15;     3+1+2= 6;
   '---'---'---'      4+9+2=15;     4+5+6=15;
* Complementary Pairs:
    8+2=10;  3+7=10;  4+6=10;  1+9=10;  5+5=10;
  ... This is a Self-complementary Magic Square!
 [Count = 8]  OK!
```

そして，研究者は，次のことに気づく。

（1）定義では n5＝5 を仮定しなかったのに，全ての解が n5＝5 になっている。

（2）定義でそう仮定しなかったのに，どれも n1+n9＝n2+n8＝n3+n7＝n4+n6＝10 になっている。

つまり，n5 に関して相互に点対称の位置にあるどの２つの数値の組み合わせも，みな「10の補数」を成すのである。

この（1），(2)は「補数対称型」の定義と同じだ。これは，何を意味するか。

三次の標準型魔方陣は，全てが「補数対称型」なのだ。「補数対称型」しか無いのである。

（3）８つの解の各行・各列・各対角線の内容を調べてみると，共通性が多い。

いや，全部が同じパターンになっている。ただ，裏返したり，時計回りに90度・180度・270度回転して出来た形ばかりではないか。

これらは，本来**一個**の解の「異形」と見なすべきではないか。

そこで研究者は，「基本解」１個とほんの少数の「変換法」を念頭に置き，それらを見出そうとする。そして，その２つだけで解の全数個を再現しようと企てる。

３．基本解と変換法のプログラム

では，上の８個のうちのどの方陣が「基本解」なのだろうか。

その前に，どうして同じ顔の兄弟解が８個も出来てしまうのだろうか。
これは,同じカードが裏返ったり回転したりして床上に落ちたと想像すればよい(透明なカードで裏からも自由に読めるとする)，８通りの仕方で。
試しに「反転」と「回転」という２種類の変換法を８個の方陣のそれぞれに実行してみる。結果はどれもが同じ解集合に戻る。８個はいずれも，資格も能力も平等なのだ。
ならば，任意に「代表解」を選べばよい。要するに，どれであってもよいのである。
小生は，こういう選別の時には，例えば次のような「不等式」をプログラムに加える。

n1＜n9; n1＜n3＜n7; ... **標準解リスト出力用の「正規化」不等式**

次のプログラムは，たった１個の「代表解」を出力し，さらにそれを変換して自分自身を含めて８個の「兄弟解」を作り出すように設計したものだ。
最初から「補数対称型」の三方陣を企図して設計し，構成している。
変換法には,「**鏡映反転**」と「**90度回転**」の２種類を用意した。後者を繰り返して使用すれば，180度や 270度の回転形は容易に得られる。

```
//** Make the Self-Complementary MS33 and Transform **
//** File:'SCMS33Trnsf.c' Dictated by Kanji Setsuda **
//**  on Jan.20, '05, Aug.16, '13 and Jun. 3, '15 **
//**  working with MacOSX 10.10.3 and Xcode 6.3.2 **
//**
//* 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);
//* Conditions for 'Self-Complementary Pairs':
//   n1+n9=10; n2+n8=10; n3+n7=10; n4+n6=10 ... eq11~eq14;
//* Normalizing Inequality Conditions for the Standard Solution:
//   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 on\n");
  printf("    Jan.20, 2005, Aug.16, 2013 and Jun. 3, 2015\n");
  printf("     working with MacOSX 10.10.3 and Xcode 6.3.2; **\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; .--.--.--.
// |n1|n2|n3| m2=n4;   | m2=n4; |n7|n4|n1|
// |--+--+--| m3=n7;   | m3=n1; |--+--+--|
// |n4|n5|n6| m4=n2;   | m4=n8; |n8|n5|n2|
// |--+--+--| m5=n5;   | m5=n5; |--+--+--|
// |n7|n8|n9| m6=n8;   | m6=n2; |n9|n6|n3|
// '--'--'--' m7=n3;   | m7=n9; '--'--'--'
//     \/     m8=n6;   | m8=n6;
// .--.--.--. m9=n9;   | m9=n3;
// |n1|n4|n7|
// |--+--+--|
// |n2|n5|n8|
// |--+--+--|
// |n3|n6|n9|
// '--'--'--'
//
//* 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[l4][0],ans[l4+1][0],ans[l4+2][0],ans[l4+3][0]);
    for(m=0;m<4;m++){printf("  .---.---.---.");}; printf("\n");
    for(m=0;m<4;m++){
      printf("  |%2d |%2d |%2d |",ans[l4+m][1],ans[l4+m][2],ans[l4+m][3]);}
    printf("\n");
    for(m=0;m<4;m++){printf("  |---+---+---|");}; printf("\n");
    for(m=0;m<4;m++){
      printf("  |%2d |%2d |%2d |",ans[l4+m][4],ans[l4+m][5],ans[l4+m][6]);}
    printf("\n");
    for(m=0;m<4;m++){printf("  |---+---+---|");}; printf("\n");
    for(m=0;m<4;m++){
      printf("  |%2d |%2d |%2d |",ans[l4+m][7],ans[l4+m][8],ans[l4+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]
  .---.---.---.
  | 2 | 9 | 4 |
  |---+---+---|
  | 7 | 5 | 3 |
  |---+---+---|
  | 6 | 1 | 8 |
  '---'---'---'
  [Count = 1]  OK!
```

* 鏡映反転と90度回転の組合せによる変形一覧 *

```
             1/  ->         2/  ->         3/  ->         4/
  .---.---.---.  .---.---.---.  .---.---.---.  .---.---.---.
  | 2 | 9 | 4 |  | 6 | 7 | 2 |  | 8 | 1 | 6 |  | 4 | 3 | 8 |
  |---+---+---|  |---+---+---|  |---+---+---|  |---+---+---|
  | 7 | 5 | 3 |  | 1 | 5 | 9 |  | 3 | 5 | 7 |  | 9 | 5 | 1 |
  |---+---+---|  |---+---+---|  |---+---+---|  |---+---+---|
  | 6 | 1 | 8 |  | 8 | 3 | 4 |  | 4 | 9 | 2 |  | 2 | 7 | 6 |
  '---'---'---'  '---'---'---'  '---'---'---'  '---'---'---'
       |    5/  ->         6/  ->         7/  ->         8/
  .---.---.---.  .---.---.---.  .---.---.---.  .---.---.---.
  | 2 | 7 | 6 |  | 4 | 9 | 2 |  | 8 | 3 | 4 |  | 6 | 1 | 8 |
  |---+---+---|  |---+---+---|  |---+---+---|  |---+---+---|
  | 9 | 5 | 1 |  | 3 | 5 | 7 |  | 1 | 5 | 9 |  | 7 | 5 | 3 |
  |---+---+---|  |---+---+---|  |---+---+---|  |---+---+---|
  | 4 | 3 | 8 |  | 8 | 1 | 6 |  | 6 | 7 | 2 |  | 2 | 9 | 4 |
  '---'---'---'  '---'---'---'  '---'---'---'  '---'---'---'
  [Count = 8]  OK!
```

** Calculated and Listed by Kanji Setsuda on
    Jan.20, 2005, Aug.16, 2013 and Jun. 3, 2015
     working with MacOSX 10.10.3 and Xcode 6.3.2; **

出来た最後の８個が，元の兄弟解の８個と同じ集合を成すことを確かめてほしい。
現れる順番が少し違うが，メンバーが別の方陣に変わったりはしていない。

この１個から８個を得る**変換構成法**は,正方形の平面方陣ならば次数･タイプを問わず，どれにも普遍的に使える。つまり，どんな方陣でも８個の「兄弟解」があり，それぞれについて１個の「代表解」を選べるのである。元の兄弟解は，いつでもこのやり方で代表解から再現することができる。
ちなみに**立方体**方陣では，同じ顔の48個の兄弟解がある。そのうちの任意の１個を代表解に選んで良いのである。行列・対角線の内容がみんな同じなのだから。
兄弟解を含む原始集合を，小生は「**原始解**の母集合」と呼んでいる。そして代表解のみを集めた解集合のことを「**標準解**のリスト」と呼ぶ。一般の眼にふれるリストは，簡潔な後者だ。印刷された出力表は，たいてい代表解のリストの方だと思ってよい。
高次魔方陣の**原始解**の母集合の出力は，いかにも冗長だ。見ていて疲れる。だから，普段は，その出力を忘れていてよい。だが，存在を忘れてはいけない。思わぬ時に**原始解**の出番が来るからだ。別タイプへの変換の時だ。変換法一般を「写像」と捉えた時，それについて「閉じている集合」は**原始解**の集合になるケースが多い。大半の方陣でそうなる。

４．定義と解集合

三次の場合，標準型と対称型の三方陣は同じ解集合を持つことがわかっている。
四次の場合は，どうであろうか。同じような解集合の重複が起きるのだろうか。それを調べてみよう。
鈴木前教授のデータベースには，解の総数表がある。一般型は，880個。汎型・完全型・正方連結型・対称型は，各48個とある。重複の可能性のあるのは，後者の48個の各タイプだ。これは，個別に構成・計数して，一々実地に調べて見るしかない。
以下に各タイプの定義の仕方を示す。その相違に注目してほしい。

＊　四次の魔方陣の各タイプの定義条件式　＊

```
＃１　四方陣の基本条件式（各タイプ共通）:K=34;
  n1+n2+n3+n4=K       ...rw1;    n1+n6+n11+n16=K...pd1;
  n5+n6+n7+n8=K       ...rw2;    n4+n7+n10+n13=K...pb4;
  n9+n10+n11+n12=K    ...rw3;
  n13+n14+n15+n16=K   ...rw4;
  n1+n5+n9+n13=K      ...c|1;
  n2+n6+n10+n14=K     ...c|2;
  n3+n7+n11+n15=K     ...c|3;
  n4+n8+n12+n16=K     ...c|4;

 14 15 16 13 14 15 16 13 14 15
          ,-----------.
  2  3  4 | 1| 2| 3| 4 | 1  2  3
          |--+--+--+--|
  6  7  8 | 5| 6| 7| 8 | 5  6  7
          |--+--+--+--|
 10 11 12 | 9|10|11|12 | 9 10 11
          |--+--+--+--|
 14 15 16 |13|14|15|16 |13 14 15
          `-----------'
  2  3  4  1  2  3  4  1  2  3

＃２　汎四方陣の汎対角線定和式：K=34;
  n1+n6+n11+n16=K...pd1;    n1+n8+n11+n14=K...pb1;
  n2+n7+n12+n13=K...pd2;    n2+n5+n12+n15=K...pb2;
  n3+n8+n9+n14=K  ...pd3;   n3+n6+n9+n16=K  ...pb3;
  n4+n5+n10+n15=K...pd4;    n4+n7+n10+n13=K...pb4;
```

```
＃３　完全四方陣の補数定和式：K=34, C=17;
  n1+n11=n2+n12=n3+n9=n4+n10
  =n5+n15=n6+n16=n7+n13=n8+n14=C=K/2    ...cmp|tcnd

＃４　対称四方陣の補数定和式：K=34, C=17;
  n1+n16=n2+n15=n3+n14=n4+n13
  =n5+n12=n6+n11=n7+n10=n8+n9=C=K/2     ...SCcnd

＃５　正方連結四方陣の正方連結四数定和式：K=34;
  n1+n2+n5+n6=K        ...cs1;
  n2+n3+n6+n7=K        ...cs2;
  n3+n4+n7+n8=K        ...cs3;
  n5+n6+n9+n10=K       ...cs4;
  n6+n7+n10+n11=K      ...cs5;
  n7+n8+n11+n12=K      ...cs6;
  n9+n10+n13+n14=K     ...cs7;
  n10+n11+n14+n15=K    ...cs8;
  n11+n12+n15+n16=K    ...cs9;
```

汎四方陣は上の＃１と＃２，完全四方陣は＃１と＃３，対称四方陣は＃１と＃４，そして正方連結型の魔方陣は＃１と＃５の連立方程式群を組み合わせて，定義する。

以下に**汎四方陣**の原始解 384個を求める基本プログラムとその成果のリストを示す。

```
//** 'Pan-diagonal' Magic Squares of Order 4: **
//** 384 'Primitive' Solutions Pre-normalized **
//** File: 'PDMS44P384.c' Dictated by Kanji Setsuda **
//**  in 2004, 2006, 2014 and on Jun. 5, 2015; **
//**  with MacOSX 10.10.3 and Xcode 6.3.2  **
//*
//* Basic Form and Simultaneous Equations for Definition *
//   14 15 16 13 14 15 16 13 14 15
//           .-----------.
//    2  3  4| 1| 2| 3| 4| 1  2  3
//           |--+--+--+--|
//    6  7  8| 5| 6| 7| 8| 5  6  7
//           |--+--+--+--|
//   10 11 12| 9|10|11|12| 9 10 11
//           |--+--+--+--|
//   14 15 16|13|14|15|16|13 14 15
//           '-----------'
//    2  3  4  1  2  3  4  1  2  3
//
//     [Basic Conditions]              [Pandiagonal Conditions]
//     n1+n2+n3+n4=C        ...rw1;    n1+n6+n11+n16=C   ... pd1;
//     n5+n6+n7+n8=C        ...rw2;    n2+n7+n12+n13=C   ... pd2;
//     n9+n10+n11+n12=C     ...rw3;    n3+n8+n9+n14=C    ... pd3;
//     n13+n14+n15+n16=C    ...rw4;    n4+n5+n10+n15=C   ... pd4;
//     n1+n5+n9+n13=C       ...cl1;    n1+n8+n11+n14=C   ... pb1;
//     n2+n6+n10+n14=C      ...cl2;    n2+n5+n12+n15=C   ... pb2;
//     n3+n7+n11+n15=C      ...cl3;    n3+n6+n9+n16=C    ... pb3;
//     n4+n8+n12+n16=C      ...cl4;    n4+n7+n10+n13=C   ... pb4;
//
#include <stdio.h>
//
short cnt, cnt2, cnt3;
short LSM;
short nm[17], uflg[17];
short anm[9][17];
//
void stp01(void), stp02(void), stp03(void), stp04(void);
void stp05(void), stp06(void), stp07(void), stp08(void);
void stp09(void), stp10(void), stp11(void), stp12(void);
```

```
void stp13(void), stp14(void), stp15(void), stp16(void);
void ansprint(void);
void printans(short x);
//
//* Main Program *
int main(){
 short n;
  printf("\n");
  printf("*** 'Pan-diagonal' Magic Squares of Order 4 ***\n");
  printf("** List of the 384 'Primitive' Solutions Pre-normalized **\n");
  for(n=0;n<17;n++){nm[n]=0; uflg[n]=0;}
  LSM=34; cnt=0; cnt3=0;
  stp01();   //* Begin the Calculations *
  if(cnt3>0){printans(cnt3);}
  printf(" [Count = %d] OK!\n",cnt);
  printf("\n");
  printf("** Calculated and Listed by Kanji Setsuda on\n");
  printf("    Jun. 5, 2015 with MacOSX 10.10.3 and Xcode 6.3.2 **\n");
  printf("\n");
 return 0;
}
//
//* Begin the Calculations *
//* Set n1 *
void stp01(){
 short a;
  for(a=1;a<17;a++){
    if(uflg[a]==0){
      nm[1]=a; uflg[a]=1;
      stp02();
      uflg[a]=0;}
  }
}
//* Set n2 *
void stp02(){
 short a;
  for(a=1;a<17;a++){
    if(uflg[a]==0){cnt2=0;
      nm[2]=a; uflg[a]=1;
      stp03();
      uflg[a]=0;}
  }
}
//* Set n3 *
void stp03(){
 short a;
  for(a=1;a<17;a++){
    if(uflg[a]==0){
      nm[3]=a; uflg[a]=1;
      stp04();
      uflg[a]=0;}
  }
}
//* Set n4=LSM-n1-n2-n3 *
void stp04(){
 short a;
  a=LSM-nm[1]-nm[2]-nm[3];
  if((0<a)&&(a<17)&&(uflg[a]==0)){
    nm[4]=a; uflg[a]=1;
    stp05();
    uflg[a]=0;}
}
//* Set n5 *
void stp05(){
```

```
 short a;
  for(a=1;a<17;a++){
    if(uflg[a]==0){if(nm[1]==1){cnt2=0;}
      nm[5]=a; uflg[a]=1;
      stp06();
      uflg[a]=0;}
  }
}
//* Set n6 *
void stp06(){
 short a;
  for(a=1;a<17;a++){
    if(uflg[a]==0){
      nm[6]=a; uflg[a]=1;
      stp07();
      uflg[a]=0;}
  }
}
//* Set n7 *
void stp07(){
 short a;
  for(a=1;a<17;a++){
    if(uflg[a]==0){
      nm[7]=a; uflg[a]=1;
      stp08();
      uflg[a]=0;}
  }
}
//* Set n8=LSM-n5-n6-n7 *
void stp08(){
 short a;
  a=LSM-nm[5]-nm[6]-nm[7];
  if((0<a)&&(a<17)&&(uflg[a]==0)){
    nm[8]=a; uflg[a]=1;
    stp09();
    uflg[a]=0;}
}
//* Set n9 *
void stp09(){
 short a;
  for(a=1;a<17;a++){
    if(uflg[a]==0){
      nm[9]=a; uflg[a]=1;
      stp10();
      uflg[a]=0;}
  }
}
//* Set n13=LSM-n1-n5-n9 *
void stp10(){
 short a;
  a=LSM-nm[1]-nm[5]-nm[9];
  if((0<a)&&(a<17)&&(uflg[a]==0)){
    nm[13]=a; uflg[a]=1;
    stp11();
    uflg[a]=0;}
}
//* Set n10=LSM-n4-n7-n13 *
void stp11(){
 short a;
  a=LSM-nm[4]-nm[7]-nm[13];
  if((0<a)&&(a<17)&&(uflg[a]==0)){
    nm[10]=a; uflg[a]=1;
    stp12();
    uflg[a]=0;}
```

```
}
//* Set n14=LSM-n2-n6-n10 *
void stp12(){
 short a,b;
  a=LSM-nm[2]-nm[6]-nm[10];
  b=LSM-nm[3]-nm[8]-nm[9];
  if((0<a)&&(a<17)){
    if((a==b)&&(uflg[a]==0)){
      nm[14]=a; uflg[a]=1;
      stp13();
      uflg[a]=0;}}
}
//* Set n11=LSM-n1-n8-n14 *
void stp13(){
 short a;
  a=LSM-nm[1]-nm[8]-nm[14];
  if((0<a)&&(a<17)&&(uflg[a]==0)){
    nm[11]=a; uflg[a]=1;
    stp14();
    uflg[a]=0;}
}
//* Set n12=LSM-n9-n10-n11 *
void stp14(){
 short a,b;
  a=LSM-nm[9]-nm[10]-nm[11];
  b=LSM-nm[2]-nm[7]-nm[13];
  if((0<a)&&(a<17)){
    if((a==b)&&(uflg[a]==0)){
      nm[12]=a; uflg[a]=1;
      stp15();
      uflg[a]=0;}}
}
//* Set n15=LSM-n3-n7-n11 *
void stp15(){
 short a,b,c;
  a=LSM-nm[3]-nm[7]-nm[11];
  b=LSM-nm[4]-nm[5]-nm[10];
  c=LSM-nm[2]-nm[5]-nm[12];
  if((0<a)&&(a<17)){
    if((a==b)&&(a==c)&&(uflg[a]==0)){
      nm[15]=a; uflg[a]=1;
      stp16();
      uflg[a]=0;}}
}
//* Set n16=LSM-n13-n14-n15 *
void stp16(){
 short a,b,c,d;
  a=LSM-nm[13]-nm[14]-nm[15];
  b=LSM-nm[4]-nm[8]-nm[12];
  c=LSM-nm[1]-nm[6]-nm[11];
  d=LSM-nm[3]-nm[6]-nm[9];
  if((0<a)&&(a<17)&&(uflg[a]==0)){
    if((a==b)&&(a==c)&&(a==d)){
      nm[16]=a; uflg[a]=1;
      ansprint();
      uflg[a]=0;}}
}
//
//* Print the Answers *
void ansprint(){
 short n;
  cnt++; cnt2++;
  if(cnt2==1){
    anm[cnt3][0]=cnt;
```

```
    for(n=1;n<17;n++){anm[cnt3][n]=nm[n];}
    cnt3++; if(cnt3==6){printans(6); cnt3=0;}
  }
}
//
//* Print the Answers *
void printans(short x){
 short l,l4,m,n;
  for(m=0;m<x;m++){
    printf("%12d#",anm[m][0]);
    if(m+1<x){printf(" ");}
  }
  printf("\n");
  for(l=0;l<4;l++){l4=l*4;
    for(m=0;m<x;m++){printf(" ");
      for(n=1;n<5;n++){printf("%3d",anm[m][l4+n]);}
      if(m+1<x){printf(" ");}
    }
    printf("\n");
  }
}
//
```

** 汎対角線型四方陣の原始解のリスト **

** List of the 384 'Primitive' Solutions Pre-normalized **

1#	2#	3#	4#	5#	6#
1 8 10 15	1 8 10 15	1 8 11 14	1 8 11 14	1 8 13 12	1 8 13 12
12 13 3 6	14 11 5 4	12 13 2 7	15 10 5 4	14 11 2 7	15 10 3 6
7 2 16 9	7 2 16 9	6 3 16 9	6 3 16 9	4 5 16 9	4 5 16 9
14 11 5 4	12 13 3 6	15 10 5 4	12 13 2 7	15 10 3 6	14 11 2 7

7#	8#	9#	10#	11#	12#
1 12 6 15	1 12 6 15	1 12 7 14	1 12 7 14	1 12 13 8	1 12 13 8
8 13 3 10	14 7 9 4	8 13 2 11	15 6 9 4	14 7 2 11	15 6 3 10
11 2 16 5	11 2 16 5	10 3 16 5	10 3 16 5	4 9 16 5	4 9 16 5
14 7 9 4	8 13 3 10	15 6 9 4	8 13 2 11	15 6 3 10	14 7 2 11

13#	14#	15#	16#	17#	18#
1 14 4 15	1 14 4 15	1 14 7 12	1 14 7 12	1 14 11 8	1 14 11 8
8 11 5 10	12 7 9 6	8 11 2 13	15 4 9 6	12 7 2 13	15 4 5 10
13 2 16 3	13 2 16 3	10 5 16 3	10 5 16 3	6 9 16 3	6 9 16 3
12 7 9 6	8 11 5 10	15 4 9 6	8 11 2 13	15 4 5 10	12 7 2 13

19#	20#	21#	22#	23#	24#
1 15 4 14	1 15 4 14	1 15 6 12	1 15 6 12	1 15 10 8	1 15 10 8
8 10 5 11	12 6 9 7	8 10 3 13	14 4 9 7	12 6 3 13	14 4 5 11
13 3 16 2	13 3 16 2	11 5 16 2	11 5 16 2	7 9 16 2	7 9 16 2
12 6 9 7	8 10 5 11	14 4 9 7	8 10 3 13	14 4 5 11	12 6 3 13

25#	31#	37#	43#	49#	55#
2 7 9 16	2 11 5 16	2 13 3 16	2 16 3 13	3 6 9 16	3 10 5 16
11 14 4 5	7 14 4 9	7 12 6 9	7 9 6 12	10 15 4 5	6 15 4 9
8 1 15 10	12 1 15 6	14 1 15 4	14 4 15 1	8 1 14 11	12 1 14 7
13 12 6 3	13 8 10 3	11 8 10 5	11 5 10 8	13 12 7 2	13 8 11 2

61#	67#	73#	79#	85#	91#
3 13 2 16	3 16 2 13	4 5 10 15	4 9 6 15	4 14 1 15	4 15 1 14
6 12 7 9	6 9 7 12	9 16 3 6	5 16 3 10	5 11 8 10	5 10 8 11
15 1 14 4	15 4 14 1	7 2 13 12	11 2 13 8	16 2 13 3	16 3 13 2
10 8 11 5	10 5 11 8	14 11 8 1	14 7 12 1	9 7 12 6	9 6 12 7

```
        97#             103#            109#            115#            121#            127#
 5  4  9 16     5 10  3 16     5 11  2 16     5 16  2 11     6  3 10 15     6  9  4 15
10 15  6  3     4 15  6  9     4 14  7  9     4  9  7 14     9 16  5  4     3 16  5 10
 8  1 12 13    14  1 12  7    15  1 12  6    15  6 12  1     7  2 11 14    13  2 11  8
11 14  7  2    11  8 13  2    10  8 13  3    10  3 13  8    12 13  8  1    12  7 14  1

        133#            139#            145#            151#            157#            163#
 6 12  1 15     6 15  1 12     7  2 11 14     7  9  4 14     7 12  1 14     7 14  1 12
 3 13  8 10     3 10  8 13     9 16  5  4     2 16  5 11     2 13  8 11     2 11  8 13
16  2 11  5    16  5 11  2     6  3 10 15    13  3 10  8    16  3 10  5    16  5 10  3
 9  7 14  4     9  4 14  7    12 13  8  1    12  6 15  1     9  6 15  4     9  4 15  6

        169#            175#            181#            187#            193#            199#
 8  1 12 13     8 10  3 13     8 11  2 13     8 13  2 11     9  4  5 16     9  6  3 16
10 15  6  3     1 15  6 12     1 14  7 12     1 12  7 14     6 15 10  3     4 15 10  5
 5  4  9 16    14  4  9  7    15  4  9  6    15  6  9  4    12  1  8 13    14  1  8 11
11 14  7  2    11  5 16  2    10  5 16  3    10  3 16  5     7 14 11  2     7 12 13  2

        205#            211#            217#            223#            229#            235#
 9  7  2 16     9 16  2  7    10  3  6 15    10  5  4 15    10  8  1 15    10 15  1  8
 4 14 11  5     4  5 11 14     5 16  9  4     3 16  9  6     3 13 12  6     3  6 12 13
15  1  8 10    15 10  8  1    11  2  7 14    13  2  7 12    16  2  7  9    16  9  7  2
 6 12 13  3     6  3 13 12     8 13 12  1     8 11 14  1     5 11 14  4     5  4 14 11

        241#            247#            253#            259#            265#            271#
11  2  7 14    11  5  4 14    11  8  1 14    11 14  1  8    12  1  8 13    12  6  3 13
 5 16  9  4     2 16  9  7     2 13 12  7     2  7 12 13     6 15 10  3     1 15 10  8
10  3  6 15    13  3  6 12    16  3  6  9    16  9  6  3     9  4  5 16    14  4  5 11
 8 13 12  1     8 10 15  1     5 10 15  4     5  4 15 10     7 14 11  2     7  9 16  2

        277#            283#            289#            295#            301#            307#
12  7  2 13    12 13  2  7    13  2  7 12    13  3  6 12    13  8  1 12    13 12  1  8
 1 14 11  8     1  8 11 14     3 16  9  6     2 16  9  7     2 11 14  7     2  7 14 11
15  4  5 10    15 10  5  4    10  5  4 15    11  5  4 14    16  5  4  9    16  9  4  5
 6  9 16  3     6  3 16  9     8 11 14  1     8 10 15  1     3 10 15  6     3  6 15 10

        313#            319#            325#            331#            337#            343#
14  1  8 11    14  4  5 11    14  7  2 11    14 11  2  7    15  1  8 10    15  4  5 10
 4 15 10  5     1 15 10  8     1 12 13  8     1  8 13 12     4 14 11  5     1 14 11  8
 9  6  3 16    12  6  3 13    15  6  3 10    15 10  3  6     9  7  2 16    12  7  2 13
 7 12 13  2     7  9 16  2     4  9 16  5     4  5 16  9     6 12 13  3     6  9 16  3

        349#            355#            361#            367#            373#            379#
15  6  3 10    15 10  3  6    16  2  7  9    16  3  6  9    16  5  4  9    16  9  4  5
 1 12 13  8     1  8 13 12     3 13 12  6     2 13 12  7     2 11 14  7     2  7 14 11
14  7  2 11    14 11  2  7    10  8  1 15    11  8  1 14    13  8  1 12    13 12  1  8
 4  9 16  5     4  5 16  9     5 11 14  4     5 10 15  4     3 10 15  6     3  6 15 10
 [Count = 384] OK!

** Calculated and Listed by Kanji Setsuda on
    Jun. 5, 2015 with MacOSX 10.10.3 and Xcode 6.3.2 **
```

同様に完全四方陣や正方連結四方陣の構成プログラムも示したいのだが，紙数に余裕が無いので省略する。結論だけ言うと，同じ原始解 384個の出力をする。

概念も定義も違うのに，それぞれが同じ解集合を構成してしまうのである。

リスト出力用の正規化不等式（共通のもの）を加えたプログラムで代表解 48個を出力させても，いずれも同じ解集合が出来てしまう。これは，何と考えたら良いのだろう。

対称四方陣の場合も，解の総数は同じとなる。だが，こちらは中身が違う。

では，試みに三連立型の正方連結完全四方陣を構成してみよう。条件式の＃１と＃３と＃５を３組同時に使うのである。全部成り立つようにする。条件式を増やしたから，解の総数はたぶん減るだろう。スマート・サイズの「標準解のリスト」をフルに出力しよう。

以下にそのプログラムを示す。

```
//** 'Simultaneous' Magic Squares of Order 4: 'Composite & Complete'; **
//** File: 'MS44Sml.c' Dictated by Kanji Setsuda **
//**  in 2004, 2006, 2013; and on June 5, 2015  **
//**  Working with MacOSX 10.10.3 and Xcode 6.3.2  **
//*
//* Basic Form and Simultaneous Equations for Definition: C=34; *
//   16 13 14 15 16 13 14 15      n1+n2+n5+n6=C;          .... (1)
//     .--.--.--.--.              n2+n3+n6+n7=C;          .... (2)
//   n4|n1|n2|n3|n4|n1 n2 n3      n3+n4+n7+n8=C;          .... (3)
//     |--+--+--+--|              n5+n6+n9+n10=C;         .... (4)
//   n8|n5|n6|n7|n8|n5 n6 n7      n6+n7+n10+n11=C;        .... (5)
//     |--+--+--+--|              n7+n8+n11+n12=C;        .... (6)
//   12|n9|10|11|12|n9 10 11      n9+n10+n13+n14=C;       .... (7)
//     |--+--+--+--|              n10+n11+n14+n15=C;      .... (8)
//   16|13|14|15|16|13 14 15      n11+n12+n15+n16=C;      .... (9)
//     '--'--'--'--'              n1+n11=n2+n12=n3+n9=
//   n4 n1 n2 n3 n4 n1 n2 n3      n4+n10=n5+n15=n6+n16=
//   [Extended Form]              n7+n13=n8+n14=CC=C/2;   ... (10)
//* Basic Conditions: C=34;
//      n1+n2+n3+n4=C;  ... (11),       n1+n5+n9+n13=C;    ... (12)
//
#include <stdio.h>
//
short cnt, cnt2;
short LSM, PSM;
short nm[17], uflg[17];
short anm[9][17];
//
void stp01(void), stp02(void), stp03(void), stp04(void);
void stp05(void), stp06(void), stp07(void), stp08(void);
void ansprint(void);
void printans(short x);
//
//* Main Program *
int main(){
 short n;
  printf("\n");
  printf("** 'Simultaneous' Magic Squares of Order 4: 'Composite & Complete'; **\n");
  printf(" ** The 48 Standard Solutions Normalized under n1<n4<n13 and n1<n16 **\n");
  for(n=0;n<17;n++){nm[n]=0; uflg[n]=0;}
  LSM=34; PSM=17; cnt=0; cnt2=0;
  stp01();      //* Begin the Calculations *
  if(cnt2>0){printans(cnt2);}
  printf(" [Count = %d]  OK!\n",cnt);
  printf("\n");
  printf("** Calculated and Listed by Kanji Setsuda on\n");
  printf("    June 5, 2015 with MacOSX 10.10.3 & Xcode 6.3.2 **\n");
  printf("\n");
 return 0;
}
//
//* Begin the Calculations *
//* Set n1 & n11 *
void stp01(){
 short a,b;
  for(a=1;a<17;a++){b=PSM-a;
```

```
    if((uflg[a]==0)&&(uflg[b]==0)){
      uflg[a]=1; uflg[b]=1;
      nm[1]=a; nm[11]=b;
      stp02();
      uflg[b]=0; uflg[a]=0;}
  }
}
//* Set n2 & n12 *
void stp02(){
 short a,b;
  for(a=1;a<17;a++){b=PSM-a;
    if((uflg[a]==0)&&(uflg[b]==0)){
      uflg[a]=1; uflg[b]=1;
      nm[2]=a; nm[12]=b;
      stp03();
      uflg[b]=0; uflg[a]=0;}
  }
}
//* Set n3 & n9 *
void stp03(){
 short a,b;
  for(a=1;a<17;a++){b=PSM-a;
    if((uflg[a]==0)&&(uflg[b]==0)){
      uflg[a]=1; uflg[b]=1;
      nm[3]=a; nm[9]=b;
      stp04();
      uflg[b]=0; uflg[a]=0;}
  }
}
//* Set n4=LSM-n1-n2-n3 & n10 & n1<n4 *
void stp04(){
 short a,b;
  a=LSM-nm[1]-nm[2]-nm[3];
  if((nm[1]<a)&&(a<17)){b=PSM-a;
    if((uflg[a]==0)&&(uflg[b]==0)){
      uflg[a]=1; uflg[b]=1;
      nm[4]=a; nm[10]=b;
      stp05();
      uflg[b]=0; uflg[a]=0;}}
}
//* Set n5 & n15 *
void stp05(){
 short a,b;
  for(a=1;a<17;a++){b=PSM-a;
    if((uflg[a]==0)&&(uflg[b]==0)){
      nm[5]=a; nm[15]=b;
      uflg[a]=1; uflg[b]=1;
      stp06();
      uflg[b]=0; uflg[a]=0;}
  }
}
//* Set n6=LSM-n1-n2-n5 & n16 & n1<n16 *
//* & Check n11+n12+n15+n16==LSM *
//* & Check n5+n6+n9+n10==LSM *
void stp06(){
 short a,b,c;
  a=LSM-nm[1]-nm[2]-nm[5];
  b=LSM-nm[11]-nm[12]-nm[15];
  c=LSM-nm[5]-nm[9]-nm[10];
  if((0<a)&&(a<17)&&(a==c)){
    if((nm[1]<b)&&(a+b==PSM)){
      if((uflg[a]==0)&&(uflg[b]==0)){
        uflg[a]=1; uflg[b]=1;
        nm[6]=a; nm[16]=b;
```

```
          stp07();
          uflg[b]=0; uflg[a]=0;}}}
}
//* Set n7=LSM-n2-n3-n6 & n13 *
//* & Check n1+n5+n9+n13==LSM & n4<n13 *
//* & Check n6+n7+n10+n11==LSM *
void stp07(){
 short a,b,c;
   a=LSM-nm[2]-nm[3]-nm[6];
   b=LSM-nm[1]-nm[5]-nm[9];
   c=LSM-nm[6]-nm[10]-nm[11];
   if((0<a)&&(a<17)&&(a==c)){
     if((a+b==PSM)&&(b>nm[4])){
       if((uflg[a]==0)&&(uflg[b]==0)){
         uflg[a]=1; uflg[b]=1;
         nm[7]=a; nm[13]=b;
         stp08();
         uflg[b]=0; uflg[a]=0;}}}
}
//* Set n8=LSM-n3-n4-n7 & n14 *
//* & Check n9+n10+n13+n14==LSM *
//* & Check n7+n8+n11+n12==LSM *
//* & Check n10+n11+n14+n15==LSM *
void stp08(){
 short a,b,c,d;
   a=LSM-nm[3]-nm[4]-nm[7];
   b=LSM-nm[9]-nm[10]-nm[13];
   c=LSM-nm[7]-nm[11]-nm[12];
   d=LSM-nm[10]-nm[11]-nm[15];
   if((0<a)&&(a<17)){
     if((a==c)&&(b==d)&&(a+b==PSM)){
       if((uflg[a]==0)&&(uflg[b]==0)){
         uflg[a]=1; uflg[b]=1;
         nm[8]=a; nm[14]=b;
         ansprint();
         uflg[b]=0; uflg[a]=0;}}}
}
//
//* Print the Answers *
void ansprint(){
 short n;
   cnt++;
   anm[cnt2][0]=cnt;
   for(n=1;n<17;n++){anm[cnt2][n]=nm[n];}
   cnt2++; if(cnt2==6){printans(6); cnt2=0;}
}
//
//* Print the Answers *
void printans(short x){
 short l,l4,m,n;
   for(m=0;m<x;m++){printf("%13d#",anm[m][0]);}
   printf("\n");
   for(l=0;l<4;l++){l4=l*4;
     for(m=0;m<x;m++){printf("  ");
       for(n=1;n<5;n++){printf("%3d",anm[m][l4+n]);}
     }
     printf("\n");
   }
}
//
```

** 標準解のリスト：連立型四方陣：正方連結型かつ完全型 **
** The 48 Standard Solutions Normalized under n1<n4<n13 and n1<n16 **

```
          1#              2#              3#              4#              5#              6#
 1  8 11 14    1  8 13 12    1  8 13 12    1 12  7 14    1 12 13  8    1 12 13  8
12 13  2  7   14 11  2  7   15 10  3  6    8 13  2 11   14  7  2 11   15  6  3 10
 6  3 16  9    4  5 16  9    4  5 16  9   10  3 16  5    4  9 16  5    4  9 16  5
15 10  5  4   15 10  3  6   14 11  2  7   15  6  9  4   15  6  3 10   14  7  2 11

          7#              8#              9#             10#             11#             12#
 1 14  7 12    1 14 11  8    1 14 11  8    1 15  6 12    1 15 10  8    1 15 10  8
 8 11  2 13   12  7  2 13   15  4  5 10    8 10  3 13   12  6  3 13   14  4  5 11
10  5 16  3    6  9 16  3    6  9 16  3   11  5 16  2    7  9 16  2    7  9 16  2
15  4  9  6   15  4  5 10   12  7  2 13   14  4  9  7   14  4  5 11   12  6  3 13

          13#             14#             15#             16#             17#             18#
 2  7 12 13    2  7 14 11    2  7 14 11    2 11  8 13    2 11 14  7    2 11 14  7
11 14  1  8   13 12  1  8   16  9  4  5    7 14  1 12   13  8  1 12   16  5  4  9
 5  4 15 10    3  6 15 10    3  6 15 10    9  4 15  6    3 10 15  6    3 10 15  6
16  9  6  3   16  9  4  5   13 12  1  8   16  5 10  3   16  5  4  9   13  8  1 12

          19#             20#             21#             22#             23#             24#
 2 13  8 11    2 13 12  7    2 13 12  7    2 16  5 11    2 16  9  7    2 16  9  7
 7 12  1 14   11  8  1 14   16  3  6  9    7  9  4 14   11  5  4 14   13  3  6 12
 9  6 15  4    5 10 15  4    5 10 15  4   12  6 15  1    8 10 15  1    8 10 15  1
16  3 10  5   16  3  6  9   11  8  1 14   13  3 10  8   13  3  6 12   11  5  4 14

          25#             26#             27#             28#             29#             30#
 3  6 15 10    3  6 15 10    3 10 15  6    3 10 15  6    3 13  8 10    3 13 12  6
13 12  1  8   16  9  4  5   13  8  1 12   16  5  4  9    6 12  1 15   10  8  1 15
 2  7 14 11    2  7 14 11    2 11 14  7    2 11 14  7    9  7 14  4    5 11 14  4
16  9  4  5   13 12  1  8   16  5  4  9   13  8  1 12   16  2 11  5   16  2  7  9

          31#             32#             33#             34#             35#             36#
 3 13 12  6    3 16  5 10    3 16  9  6    3 16  9  6    4  5 16  9    4  5 16  9
16  2  7  9    6  9  4 15   10  5  4 15   13  2  7 12   14 11  2  7   15 10  3  6
 5 11 14  4   12  7 14  1    8 11 14  1    8 11 14  1    1  8 13 12    1  8 13 12
10  8  1 15   13  2 11  8   13  2  7 12   10  5  4 15   15 10  3  6   14 11  2  7

          37#             38#             39#             40#             41#             42#
 4  9 16  5    4  9 16  5    4 14  7  9    4 14 11  5    4 14 11  5    4 15  6  9
14  7  2 11   15  6  3 10    5 11  2 16    9  7  2 16   15  1  8 10    5 10  3 16
 1 12 13  8    1 12 13  8   10  8 13  3    6 12 13  3    6 12 13  3   11  8 13  2
15  6  3 10   14  7  2 11   15  1 12  6   15  1  8 10    9  7  2 16   14  1 12  7

          43#             44#             45#             46#             47#             48#
 4 15 10  5    4 15 10  5    5  4 15 10    5 16  3 10    6  3 16  9    6 15  4  9
 9  6  3 16   14  1  8 11   16  9  6  3    4  9  6 15   15 10  5  4    3 10  5 16
 7 12 13  2    7 12 13  2    2  7 12 13   14  7 12  1    1  8 11 14   13  8 11  2
14  1  8 11    9  6  3 16   11 14  1  8   11  2 13  8   12 13  2  7   12  1 14  7
 [Count = 48]  OK!

** Calculated and Listed by Kanji Setsuda on
    June 5, 2015 with MacOSX 10.10.3 & Xcode 6.3.2 **
```

これは**原始解**ではなくて**標準解**のリストだ。全 48 個は，他の３タイプと変わらない。総数が減らないどころか，やはり同じ解集合を作ってしまったのである。

八次の場合だと，汎八方陣と完全八方陣は，一部共通部分がダブるが，違った解集合を構成する。総数も違う。正方連結汎八方陣も正方連結完全八方陣も，各個に他とは違った解集合を構成する。連立型は，ずっと小さな「積集合」を形成する。

ところが，**四次**の場合は，「和集合」も「積集合」もみな同じで変わらない！

・・・・・・

全てが論理学的「同値」を指し示している。

確かに各定義式にはお互いに内的関連があり，相手の性質を簡単な代数計算で導き出すことができる。汎四方陣は，同時に完全四方陣であり，同時に正方連結完全四方陣でもあるのだ。四次の正方連結型魔方陣は，同時に汎四方陣であり，同時に完全四方陣でもある。実に不思議な現象であると言わなければならない。

小生は，これをあえて四次の汎魔方陣の「**奇跡の一致**」と呼んでいる。創造の神の仕掛けた偉大な「謎」の１つではないか，と思っている。

ここでは，一見**定義**が無意味に思える。我々は，一体何を作って来たのだろうかと。

だが，定義そのものは，決して失敗していない。**存在**そのものが不思議なのである。

良く考えれば，様々な定義の総体が，この現象の不思議さを説明するのに，それこそ大いに役立っているではないか。相手は，複雑な構造物なのである。その特徴は，多様なアプローチを試みずには，総体的に解き明かせない。知のたくましい運動こそが，存在を解明するのである。そう思うことにしている。

完全四方陣と**対称四方陣**の間の緊密な相互連関も，不思議の１つだ。

５．相互「型変換」の方法

対称四方陣からでもよい。完全四方陣からでもよい。３列目と４列目を相互に交換してから，その後で３行目と４行目を相互に交換する。そうすると，対称型は完全型に変わり，完全型は対称型に変わる。しかも，いずれも魔方陣には変わりがない。

この可能性に気づいていた人は昔から多数居たようだが，小生が自分で初めて気づいた時には，ずいぶんと喜んだものだった。以前の記事で，詳しく報告している。

1/SC	→型変換→	1/C
1 8 15 10	1 8 10 15	1 8 10 15
12 13 6 3	12 13 3 6	12 13 3 6
14 11 4 5	14 11 5 4	7 2 16 9
7 2 9 16	7 2 16 9	14 11 5 4

1/C	→型変換→	1/SC
1 8 10 15	1 8 15 10	1 8 15 10
12 13 3 6	12 13 6 3	12 13 6 3
7 2 16 9	7 2 9 16	14 11 4 5
14 11 5 4	14 11 4 5	7 2 9 16

（縦の３列目と４列目を交換してから，その結果の横の３行目と４行目をさらに交換している。）

ところが，その後 48個のリストの全数個で対応関係を逐一詳しく調べてみると，後半の24個について，相手が見つからないケースが 12個もあることがわかった。せっかくの**一対一対応**の関係が，残念ながら崩れているのであった。

＊ 対称四方陣とそれを型変換した完全四方陣のリスト：元のリストの解番号を参照 ＊

1/S 5/C		2/S 6/C		3/S 3/C	
1 8 12 13	1 8 13 12	1 8 12 13	1 8 13 12	1 8 14 11	1 8 11 14
14 11 7 2	14 11 2 7	15 10 6 3	15 10 3 6	12 13 7 2	12 13 2 7
15 10 6 3	4 5 16 9	14 11 7 2	4 5 16 9	15 10 4 5	6 3 16 9
4 5 9 16	15 10 3 6	4 5 9 16	14 11 2 7	6 3 9 16	15 10 5 4

4/S 4/C		5/S 1/C		6/S 2/C	
1 8 14 11	1 8 11 14	1 8 15 10	1 8 10 15	1 8 15 10	1 8 10 15
15 10 4 5	15 10 5 4	12 13 6 3	12 13 3 6	14 11 4 5	14 11 5 4
12 13 7 2	6 3 16 9	14 11 4 5	7 2 16 9	12 13 6 3	7 2 16 9
6 3 9 16	12 13 2 7	7 2 9 16	14 11 5 4	7 2 9 16	12 13 3 6

7/S 9/C		8/S 10/C		9/S 8/C	
1 12 8 13	1 12 13 8	1 12 8 13	1 12 13 8	1 12 14 7	1 12 7 14
14 7 11 2	14 7 2 11	15 6 10 3	15 6 3 10	15 6 4 9	15 6 9 4
15 6 10 3	4 9 16 5	14 7 11 2	4 9 16 5	8 13 11 2	10 3 16 5
4 9 5 16	15 6 3 10	4 9 5 16	14 7 2 11	10 3 5 16	8 13 2 11

10/S 7/C

1	12	15	6	1	12	6	15
14	7	4	9	14	7	9	4
8	13	10	3	11	2	16	5
11	2	5	16	8	13	3	10

11/S 12/C

1	14	8	11	1	14	11	8
15	4	10	5	15	4	5	10
12	7	13	2	6	9	16	3
6	9	3	16	12	7	2	13

12/S 11/C

1	14	12	7	1	14	7	12
15	4	6	9	15	4	9	6
8	11	13	2	10	5	16	3
10	5	3	16	8	11	2	13

13/S 17/C

2	7	11	14	2	7	14	11
13	12	8	1	13	12	1	8
16	9	5	4	3	6	15	10
3	6	10	15	16	9	4	5

14/S 18/C

2	7	11	14	2	7	14	11
16	9	5	4	16	9	4	5
13	12	8	1	3	6	15	10
3	6	10	15	13	12	1	8

15/S 15/C

2	7	13	12	2	7	12	13
11	14	8	1	11	14	1	8
16	9	3	6	5	4	15	10
5	4	10	15	16	9	6	3

16/S 16/C

2	7	13	12	2	7	12	13
16	9	3	6	16	9	6	3
11	14	8	1	5	4	15	10
5	4	10	15	11	14	1	8

17/S 13/C

2	7	16	9	2	7	9	16
11	14	5	4	11	14	4	5
13	12	3	6	8	1	15	10
8	1	10	15	13	12	6	3

18/S 14/C

2	7	16	9	2	7	9	16
13	12	3	6	13	12	6	3
11	14	5	4	8	1	15	10
8	1	10	15	11	14	4	5

19/S 21/C

2	11	7	14	2	11	14	7
13	8	12	1	13	8	1	12
16	5	9	4	3	10	15	6
3	10	6	15	16	5	4	9

20/S 22/C

2	11	7	14	2	11	14	7
16	5	9	4	16	5	4	9
13	8	12	1	3	10	15	6
3	10	6	15	13	8	1	12

21/S 20/C

2	11	13	8	2	11	8	13
16	5	3	10	16	5	10	3
7	14	12	1	9	4	15	6
9	4	6	15	7	14	1	12

22/S 19/C

2	11	16	5	2	11	5	16
13	8	3	10	13	8	10	3
7	14	9	4	12	1	15	6
12	1	6	15	7	14	4	9

23/S 24/C

2	13	7	12	2	13	12	7
16	3	9	6	16	3	6	9
11	8	14	1	5	10	15	4
5	10	4	15	11	8	1	14

24/S 23/C

2	13	11	8	2	13	8	11
16	3	5	10	16	3	10	5
7	12	14	1	9	6	15	4
9	6	4	15	7	12	1	14

25/S ??/C

3	6	13	12	3	6	12	13
10	15	8	1	10	15	1	8
16	9	2	7	5	4	14	11
5	4	11	14	16	9	7	2

26/S 26/C

3	6	13	12	3	6	12	13
16	9	2	7	16	9	7	2
10	15	8	1	5	4	14	11
5	4	11	14	10	15	1	8

27/S ??/C

3	6	16	9	3	6	9	16
10	15	5	4	10	15	4	5
13	12	2	7	8	1	14	11
8	1	11	14	13	12	7	2

28/S 25/C

3	6	16	9	3	6	9	16
13	12	2	7	13	12	7	2
10	15	5	4	8	1	14	11
8	1	11	14	10	15	4	5

29/S 30/C

3	10	13	8	3	10	8	13
16	5	2	11	16	5	11	2
6	15	12	1	9	4	14	7
9	4	7	14	6	15	1	12

30/S 29/C

3	10	16	5	3	10	5	16
13	8	2	11	13	8	11	2
6	15	9	4	12	1	14	7
12	1	7	14	6	15	4	9

31/S 34/C

3	13	6	12	3	13	12	6
16	2	9	7	16	2	7	9
10	8	15	1	5	11	14	4
5	11	4	14	10	8	1	15

32/S 33/C

3	13	10	8	3	13	8	10
16	2	5	11	16	2	11	5
6	12	15	1	9	7	14	4
9	7	4	14	6	12	1	15

33/S ??/C

4	5	14	11	4	5	11	14
9	16	7	2	9	16	2	7
15	10	1	8	6	3	13	12
6	3	12	13	15	10	8	1

34/S 36/C

4	5	14	11	4	5	11	14
15	10	1	8	15	10	8	1
9	16	7	2	6	3	13	12
6	3	12	13	9	16	2	7

35/S ??/C

4	5	15	10	4	5	10	15
9	16	6	3	9	16	3	6
14	11	1	8	7	2	13	12
7	2	12	13	14	11	8	1

36/S 35/C

4	5	15	10	4	5	10	15
14	11	1	8	14	11	8	1
9	16	6	3	7	2	13	12
7	2	12	13	9	16	3	6

37/S 40/C

4	9	14	7	4	9	7	14
15	6	1	12	15	6	12	1
5	16	11	2	10	3	13	8
10	3	8	13	5	16	2	11

38/S 39/C

4	9	15	6	4	9	6	15
14	7	1	12	14	7	12	1
5	16	10	3	11	2	13	8
11	2	8	13	5	16	3	10

39/S 44/C

4	14	5	11	4	14	11	5
15	1	10	8	15	1	8	10
9	7	16	2	6	12	13	3
6	12	3	13	9	7	2	16

```
        40/S 43/C                41/S ??/C                42/S ??/C
 4 14  9  7   4 14  7  9    5  4 16  9    5  4  9 16    5  4 16  9    5  4  9 16
15  1  6 12  15  1 12  6   10 15  3  6   10 15  6  3   11 14  2  7   11 14  7  2
 5 11 16  2  10  8 13  3   11 14  2  7    8  1 12 13   10 15  3  6    8  1 12 13
10  8  3 13   5 11  2 16    8  1 13 12   11 14  7  2    8  1 13 12   10 15  6  3
        43/S ??/C                44/S ??/C                45/S ??/C
 5 10 11  8   5 10  8 11    5 11 10  8    5 11  8 10    6  3 15 10    6  3 10 15
16  3  2 13  16  3 13  2   16  2  3 13   16  2 13  3    9 16  4  5    9 16  5  4
 4 15 14  1   9  6 12  7    4 14 15  1    9  7 12  6   12 13  1  8    7  2 11 14
 9  6  7 12   4 15  1 14    9  7  6 12    4 14  1 15    7  2 14 11   12 13  8  1
        46/S ??/C                47/S ??/C                48/S ??/C
 6  3 15 10   6  3 10 15    6  9 12  7    6  9  7 12    6 12  9  7    6 12  7  9
12 13  1  8  12 13  8  1   15  4  1 14   15  4 14  1   15  1  4 14   15  1 14  4
 9 16  4  5   7  2 11 14    3 16 13  2   10  5 11  8    3 13 16  2   10  8 11  5
 7  2 14 11   9 16  5  4   10  5  8 11    3 16  2 13   10  8  5 11    3 13  2 16

 * Monitor List of Solution Correspondences between /SC and /Complete *
  ??: 12
   1:  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  25:  1  1  0  0  1  1  0  0  1  1  1  1  0  0  1  1  0  0  1  1  0  0  0  0

** Calculated and Listed by Kanji Setsuda on
   June 7, 2015 with MacOSX 10.10.3 and Xcode 6.3.2 **
```

上のデータは，対称四方陣の標準解を48個出力し，さらに上の変換法を各個に実行して完全四方陣としたもの。元のリストにあるか一々相手をさがし，モニター出力してみた。
御覧の通り，後半の 24個の内，相手が見つからない解が 12個もある。変換後の完全四方陣が，n1<n16，n1<n4<n13 の範囲を逸脱して「標準解」ではなくなっているのだ。

この結果には頭を痛めてしまった。何故なら，この相互変換法を前提にした後の議論の一切が成り立たないことになってしまったから。汎型で起こる現象を対称型の方から説明できるとした主張が，無意味になってしまったから。…
実に惜しい気がした。大半は良い感じで成り立っているのに…。
この問題をどう解決するか。
… 似たような現象を数学の分野でさがしてみた。
これは，**集合**と**写像**の問題である。考えている写像変換について「閉じている集合」をさがす時に，数学者は大概要素のもう少し多いものを求める。例えば，自然数の集合は加法について「閉じている」が，減法については「閉じていない」。閉じさせるためには，要素の範囲を0や負数を含む整数にまで広げなければならない。
この考え方を利用できないか。標準解48個のリストで型変換するのではなくて，**原始解**384個の母集合で型変換して対応関係を調べて見てはどうだろうか。

以下のデータがその追試である。完全四方陣も対称四方陣も，原始解の母集合を用意する。そして，384個の１つ１つについて型変換して相手をさがす。相手が見つかればその番号を参照してリストに載せた。「お見合い」モニターの方は省略無しに全部掲げた。

＊　対称四方陣と完全四方陣と間の型変換を「原始解」の集合で調べる　＊

```
         1/S  5/C                 2/S  6/C                 3/S  3/C
 1  8 12 13   1  8 13 12    1  8 12 13   1  8 13 12    1  8 14 11   1  8 11 14
14 11  7  2  14 11  2  7   15 10  6  3  15 10  3  6   12 13  7  2  12 13  2  7
15 10  6  3   4  5 16  9   14 11  7  2   4  5 16  9   15 10  4  5   6  3 16  9
 4  5  9 16  15 10  3  6    4  5  9 16  14 11  2  7    6  3  9 16  15 10  5  4
```

4/S 4/C

1	8	14	11	1	8	11	14
15	10	4	5	15	10	5	4
12	13	7	2	6	3	16	9
6	3	9	16	12	13	2	7

5/S 1/C

1	8	15	10	1	8	10	15
12	13	6	3	12	13	3	6
14	11	4	5	7	2	16	9
7	2	9	16	14	11	5	4

6/S 2/C

1	8	15	10	1	8	10	15
14	11	4	5	14	11	5	4
12	13	6	3	7	2	16	9
7	2	9	16	12	13	3	6

7/S 11/C

1	12	8	13	1	12	13	8
14	7	11	2	14	7	2	11
15	6	10	3	4	9	16	5
4	9	5	16	15	6	3	10

8/S 12/C

1	12	8	13	1	12	13	8
15	6	10	3	15	6	3	10
14	7	11	2	4	9	16	5
4	9	5	16	14	7	2	11

9/S 9/C

1	12	14	7	1	12	7	14
8	13	11	2	8	13	2	11
15	6	4	9	10	3	16	5
10	3	5	16	15	6	9	4

10/S 10/C

1	12	14	7	1	12	7	14
15	6	4	9	15	6	9	4
8	13	11	2	10	3	16	5
10	3	5	16	8	13	2	11

11/S 7/C

1	12	15	6	1	12	6	15
8	13	10	3	8	13	3	10
14	7	4	9	11	2	16	5
11	2	5	16	14	7	9	4

12/S 8/C

1	12	15	6	1	12	6	15
14	7	4	9	14	7	9	4
8	13	10	3	11	2	16	5
11	2	5	16	8	13	3	10

13/S 17/C

1	14	8	11	1	14	11	8
12	7	13	2	12	7	2	13
15	4	10	5	6	9	16	3
6	9	3	16	15	4	5	10

14/S 18/C

1	14	8	11	1	14	11	8
15	4	10	5	15	4	5	10
12	7	13	2	6	9	16	3
6	9	3	16	12	7	2	13

15/S 15/C

1	14	12	7	1	14	7	12
8	11	13	2	8	11	2	13
15	4	6	9	10	5	16	3
10	5	3	16	15	4	9	6

16/S 16/C

1	14	12	7	1	14	7	12
15	4	6	9	15	4	9	6
8	11	13	2	10	5	16	3
10	5	3	16	8	11	2	13

17/S 13/C

1	14	15	4	1	14	4	15
8	11	10	5	8	11	5	10
12	7	6	9	13	2	16	3
13	2	3	16	12	7	9	6

18/S 14/C

1	14	15	4	1	14	4	15
12	7	6	9	12	7	9	6
8	11	10	5	13	2	16	3
13	2	3	16	8	11	5	10

19/S 23/C

1	15	8	10	1	15	10	8
12	6	13	3	12	6	3	13
14	4	11	5	7	9	16	2
7	9	2	16	14	4	5	11

20/S 24/C

1	15	8	10	1	15	10	8
14	4	11	5	14	4	5	11
12	6	13	3	7	9	16	2
7	9	2	16	12	6	3	13

21/S 21/C

1	15	12	6	1	15	6	12
8	10	13	3	8	10	3	13
14	4	7	9	11	5	16	2
11	5	2	16	14	4	9	7

22/S 22/C

1	15	12	6	1	15	6	12
14	4	7	9	14	4	9	7
8	10	13	3	11	5	16	2
11	5	2	16	8	10	3	13

23/S 19/C

1	15	14	4	1	15	4	14
8	10	11	5	8	10	5	11
12	6	7	9	13	3	16	2
13	3	2	16	12	6	9	7

24/S 20/C

1	15	14	4	1	15	4	14
12	6	7	9	12	6	9	7
8	10	11	5	13	3	16	2
13	3	2	16	8	10	5	11

25/S 29/C

2	7	11	14	2	7	14	11
13	12	8	1	13	12	1	8
16	9	5	4	3	6	15	10
3	6	10	15	16	9	4	5

26/S 30/C

2	7	11	14	2	7	14	11
16	9	5	4	16	9	4	5
13	12	8	1	3	6	15	10
3	6	10	15	13	12	1	8

27/S 27/C

2	7	13	12	2	7	12	13
11	14	8	1	11	14	1	8
16	9	3	6	5	4	15	10
5	4	10	15	16	9	6	3

28/S 28/C

2	7	13	12	2	7	12	13
16	9	3	6	16	9	6	3
11	14	8	1	5	4	15	10
5	4	10	15	11	14	1	8

29/S 25/C

2	7	16	9	2	7	9	16
11	14	5	4	11	14	4	5
13	12	3	6	8	1	15	10
8	1	10	15	13	12	6	3

30/S 26/C

2	7	16	9	2	7	9	16
13	12	3	6	13	12	6	3
11	14	5	4	8	1	15	10
8	1	10	15	11	14	4	5

31/S 35/C

2	11	7	14	2	11	14	7
13	8	12	1	13	8	1	12
16	5	9	4	3	10	15	6
3	10	6	15	16	5	4	9

32/S 36/C

2	11	7	14	2	11	14	7
16	5	9	4	16	5	4	9
13	8	12	1	3	10	15	6
3	10	6	15	13	8	1	12

33/S 33/C

2	11	13	8	2	11	8	13
7	14	12	1	7	14	1	12
16	5	3	10	9	4	15	6
9	4	6	15	16	5	10	3

34/S 34/C

2	11	13	8	2	11	8	13
16	5	3	10	16	5	10	3
7	14	12	1	9	4	15	6
9	4	6	15	7	14	1	12

35/S 31/C

2	11	16	5	2	11	5	16
7	14	9	4	7	14	4	9
13	8	3	10	12	1	15	6
12	1	6	15	13	8	10	3

36/S 32/C

2	11	16	5	2	11	5	16
13	8	3	10	13	8	10	3
7	14	9	4	12	1	15	6
12	1	6	15	7	14	4	9

37/S 41/C

2	13	7	12	2	13	12	7
11	8	14	1	11	8	1	14
16	3	9	6	5	10	15	4
5	10	4	15	16	3	6	9

38/S 42/C

2	13	7	12	2	13	12	7
16	3	9	6	16	3	6	9
11	8	14	1	5	10	15	4
5	10	4	15	11	8	1	14

39/S 39/C

2	13	11	8	2	13	8	11
7	12	14	1	7	12	1	14
16	3	5	10	9	6	15	4
9	6	4	15	16	3	10	5

40/S 40/C

2	13	11	8	2	13	8	11
16	3	5	10	16	3	10	5
7	12	14	1	9	6	15	4
9	6	4	15	7	12	1	14

41/S 37/C

2	13	16	3	2	13	3	16
7	12	9	6	7	12	6	9
11	8	5	10	14	1	15	4
14	1	4	15	11	8	10	5

42/S 38/C

2	13	16	3	2	13	3	16
11	8	5	10	11	8	10	5
7	12	9	6	14	1	15	4
14	1	4	15	7	12	6	9

43/S 47/C

2	16	7	9	2	16	9	7
11	5	14	4	11	5	4	14
13	3	12	6	8	10	15	1
8	10	1	15	13	3	6	12

44/S 48/C

2	16	7	9	2	16	9	7
13	3	12	6	13	3	6	12
11	5	14	4	8	10	15	1
8	10	1	15	11	5	4	14

45/S 45/C

2	16	11	5	2	16	5	11
7	9	14	4	7	9	4	14
13	3	8	10	12	6	15	1
12	6	1	15	13	3	10	8

46/S 46/C

2	16	11	5	2	16	5	11
13	3	8	10	13	3	10	8
7	9	14	4	12	6	15	1
12	6	1	15	7	9	4	14

47/S 43/C

2	16	13	3	2	16	3	13
7	9	12	6	7	9	6	12
11	5	8	10	14	4	15	1
14	4	1	15	11	5	10	8

48/S 44/C

2	16	13	3	2	16	3	13
11	5	8	10	11	5	10	8
7	9	12	6	14	4	15	1
14	4	1	15	7	9	6	12

49/S 53/C

3	6	10	15	3	6	15	10
13	12	8	1	13	12	1	8
16	9	5	4	2	7	14	11
2	7	11	14	16	9	4	5

50/S 54/C

3	6	10	15	3	6	15	10
16	9	5	4	16	9	4	5
13	12	8	1	2	7	14	11
2	7	11	14	13	12	1	8

51/S 51/C

3	6	13	12	3	6	12	13
10	15	8	1	10	15	1	8
16	9	2	7	5	4	14	11
5	4	11	14	16	9	7	2

. . . . . .

94/S 94/C

4	15	9	6	4	15	6	9
14	1	7	12	14	1	12	7
5	10	16	3	11	8	13	2
11	8	2	13	5	10	3	16

95/S 91/C

4	15	14	1	4	15	1	14
5	10	11	8	5	10	8	11
9	6	7	12	16	3	13	2
16	3	2	13	9	6	12	7

96/S 92/C

4	15	14	1	4	15	1	14
9	6	7	12	9	6	12	7
5	10	11	8	16	3	13	2
16	3	2	13	5	10	8	11

97/S 101/C

5	4	10	15	5	4	15	10
11	14	8	1	11	14	1	8
16	9	3	6	2	7	12	13
2	7	13	12	16	9	6	3

98/S 102/C

5	4	10	15	5	4	15	10
16	9	3	6	16	9	6	3
11	14	8	1	2	7	12	13
2	7	13	12	11	14	1	8

99/S 99/C

5	4	11	14	5	4	14	11
10	15	8	1	10	15	1	8
16	9	2	7	3	6	12	13
3	6	13	12	16	9	7	2

100/S 100/C

5	4	11	14	5	4	14	11
16	9	2	7	16	9	7	2
10	15	8	1	3	6	12	13
3	6	13	12	10	15	1	8

101/S 97/C

5	4	16	9	5	4	9	16
10	15	3	6	10	15	6	3
11	14	2	7	8	1	12	13
8	1	13	12	11	14	7	2

102/S 98/C

5	4	16	9	5	4	9	16
11	14	2	7	11	14	7	2
10	15	3	6	8	1	12	13
8	1	13	12	10	15	6	3

103/S 107/C

5	10	4	15	5	10	15	4
11	8	14	1	11	8	1	14
16	3	9	6	2	13	12	7
2	13	7	12	16	3	6	9

104/S 108/C

5	10	4	15	5	10	15	4
16	3	9	6	16	3	6	9
11	8	14	1	2	13	12	7
2	13	7	12	11	8	1	14

105/S 105/C

5	10	11	8	5	10	8	11
4	15	14	1	4	15	1	14
16	3	2	13	9	6	12	7
9	6	7	12	16	3	13	2

. . . . . .

214/S 214/C

9	16	6	3	9	16	3	6
7	2	12	13	7	2	13	12
4	5	15	10	14	11	8	1
14	11	1	8	4	5	10	15

215/S 211/C

9	16	7	2	9	16	2	7
4	5	14	11	4	5	11	14
6	3	12	13	15	10	8	1
15	10	1	8	6	3	13	12

216/S 212/C

9	16	7	2	9	16	2	7
6	3	12	13	6	3	13	12
4	5	14	11	15	10	8	1
15	10	1	8	4	5	11	14

217/S 221/C

10	3	5	16	10	3	16	5
8	13	11	2	8	13	2	11
15	6	4	9	1	12	7	14
1	12	14	7	15	6	9	4

218/S 222/C

10	3	5	16	10	3	16	5
15	6	4	9	15	6	9	4
8	13	11	2	1	12	7	14
1	12	14	7	8	13	2	11

219/S 219/C

10	3	8	13	10	3	13	8
5	16	11	2	5	16	2	11
15	6	1	12	4	9	7	14
4	9	14	7	15	6	12	1

220/S 220/C

10	3	8	13	10	3	13	8
15	6	1	12	15	6	12	1
5	16	11	2	4	9	7	14
4	9	14	7	5	16	2	11

221/S 217/C

10	3	15	6	10	3	6	15
5	16	4	9	5	16	9	4
8	13	1	12	11	2	7	14
11	2	14	7	8	13	12	1

222/S 218/C

10	3	15	6	10	3	6	15
8	13	1	12	8	13	12	1
5	16	4	9	11	2	7	14
11	2	14	7	5	16	9	4

. . . . .

286/S 286/C

12	13	6	3	12	13	3	6
7	2	9	16	7	2	16	9
1	8	15	10	14	11	5	4
14	11	4	5	1	8	10	15

287/S 283/C

12	13	7	2	12	13	2	7
1	8	14	11	1	8	11	14
6	3	9	16	15	10	5	4
15	10	4	5	6	3	16	9

288/S 284/C

12	13	7	2	12	13	2	7
6	3	9	16	6	3	16	9
1	8	14	11	15	10	5	4
15	10	4	5	1	8	11	14

289/S 293/C

13	2	3	16	13	2	16	3
8	11	10	5	8	11	5	10
12	7	6	9	1	14	4	15
1	14	15	4	12	7	9	6

290/S 294/C

13	2	3	16	13	2	16	3
12	7	6	9	12	7	9	6
8	11	10	5	1	14	4	15
1	14	15	4	8	11	5	10

291/S 291/C

13	2	8	11	13	2	11	8
3	16	10	5	3	16	5	10
12	7	1	14	6	9	4	15
6	9	15	4	12	7	14	1

292/S 292/C

13	2	8	11	13	2	11	8
12	7	1	14	12	7	14	1
3	16	10	5	6	9	4	15
6	9	15	4	3	16	5	10

293/S 289/C

13	2	12	7	13	2	7	12
3	16	6	9	3	16	9	6
8	11	1	14	10	5	4	15
10	5	15	4	8	11	14	1

294/S 290/C

13	2	12	7	13	2	7	12
8	11	1	14	8	11	14	1
3	16	6	9	10	5	4	15
10	5	15	4	3	16	9	6

. . . . .

334/S 334/C

14	11	4	5	14	11	5	4
7	2	9	16	7	2	16	9
1	8	15	10	12	13	3	6
12	13	6	3	1	8	10	15

335/S 331/C

14	11	7	2	14	11	2	7
1	8	12	13	1	8	13	12
4	5	9	16	15	10	3	6
15	10	6	3	4	5	16	9

336/S 332/C

14	11	7	2	14	11	2	7
4	5	9	16	4	5	16	9
1	8	12	13	15	10	3	6
15	10	6	3	1	8	13	12

337/S 341/C

15	1	4	14	15	1	14	4
6	12	9	7	6	12	7	9
10	8	5	11	3	13	2	16
3	13	16	2	10	8	11	5

338/S 342/C

15	1	4	14	15	1	14	4
10	8	5	11	10	8	11	5
6	12	9	7	3	13	2	16
3	13	16	2	6	12	7	9

339/S 339/C

15	1	6	12	15	1	12	6
4	14	9	7	4	14	7	9
10	8	3	13	5	11	2	16
5	11	16	2	10	8	13	3

340/S 340/C

15	1	6	12	15	1	12	6
10	8	3	13	10	8	13	3
4	14	9	7	5	11	2	16
5	11	16	2	4	14	7	9

341/S 337/C

15	1	10	8	15	1	8	10
4	14	5	11	4	14	11	5
6	12	3	13	9	7	2	16
9	7	16	2	6	12	13	3

342/S 338/C

15	1	10	8	15	1	8	10
6	12	3	13	6	12	13	3
4	14	5	11	9	7	2	16
9	7	16	2	4	14	11	5

. . . . .

376/S 376/C

16	5	3	10	16	5	10	3
9	4	6	15	9	4	15	6
2	11	13	8	7	14	1	12
7	14	12	1	2	11	8	13

377/S 373/C

16	5	9	4	16	5	4	9
2	11	7	14	2	11	14	7
3	10	6	15	13	8	1	12
13	8	12	1	3	10	15	6

378/S 374/C

16	5	9	4	16	5	4	9
3	10	6	15	3	10	15	6
2	11	7	14	13	8	1	12
13	8	12	1	2	11	14	7

```
      379/S 383/C                   380/S 384/C                   381/S 381/C
 16  9  2  7  16  9  7  2    16  9  2  7  16  9  7  2    16  9  3  6  16  9  6  3
  3  6 13 12   3  6 12 13     5  4 11 14   5  4 14 11     2  7 13 12   2  7 12 13
  5  4 11 14  10 15  1  8     3  6 13 12  10 15  1  8     5  4 10 15  11 14  1  8
 10 15  8  1   5  4 14 11    10 15  8  1   3  6 12 13    11 14  8  1   5  4 15 10
      382/S 382/C                   383/S 379/C                   384/S 380/C
 16  9  3  6  16  9  6  3    16  9  5  4  16  9  4  5    16  9  5  4  16  9  4  5
  5  4 10 15   5  4 15 10     2  7 11 14   2  7 14 11     3  6 10 15   3  6 15 10
  2  7 13 12  11 14  1  8     3  6 10 15  13 12  1  8     2  7 11 14  13 12  1  8
 11 14  8  1   2  7 12 13    13 12  8  1   3  6 15 10    13 12  8  1   2  7 14 11

[Count = S/384:384/C]

* Monitor List of Solution Correspondences between /SC and /Cmplt *
 ??:  0
  1:  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
 25:  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
 49:  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
 73:  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
 97:  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
121:  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
145:  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
169:  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
193:  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
217:  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
241:  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
265:  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
289:  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
313:  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
337:  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
361:  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
[OK!]

** Calculated and Listed by Kanji Setsuda on
   June 7, 2015 with MacOSX 10.10.3 and Xcode 6.3.2 **
```

完璧な**一対一対応**だ。原始解の母集合は，相互型変換の方法について「閉じている」。
これならば，使えるではないか。
何故に対称四方陣もまた同数の解集合を持つのか，その構造的説明も付くではないか。
完全四方陣と表裏一体の構造が，全数において解き明かされたと考えている。

表側から見れば汎四方陣でもあり完全四方陣でもあり正方連結完全四方陣でもあるものが，裏側から見るとどれもが対称四方陣に変換される。四方陣のこの 384個は，一体全体何ものなのだろうか？
緊密な構造物だ。
神の仕掛け賜うた謎の１つではないのか。...
哲学好きの血が騒いでならない。

こんな素敵な発見でも，議論の仕方によっては混乱の原因になることがある。
我々が魔方陣を構成する際，定義を目的にすべきか，集合を目的にすべきか。
同じ解集合を作り出す様々な定義の各個を「別物」と考えるべきか，それとも結果から見て本来「同一」と考えるべきか，どれが「基本形」なのか，等々。
こういう生産的な議論は，一々喧しくすべきである。私は，厳密な議論を歓迎する。

であるからこそ，あなたがレポートを書くとき，何を前提にしてその結論を得たのかを全部明記すべきなのである。皆にフェアに議論してもらうためなのだ。

第３章　さらに何を研究すればよいのか

昔は，魔方陣と言えば，ただ「**標準型**」を研究していれば良かった。**三次**の場合は，標準解が１個（原始解は８個）しかない。だが，**四次**になると，標準解が 880個にも増える（原始解は，その８倍の7040個もある)。それを特定するだけで，一大事業だった。
幾つかの奇跡的な発見が，昔の偉大な研究者によって手計算で成し遂げられている。
ところが，**五次**になると，標準型の標準解は，なんと２億7530万5224個にもなる（コンピューター時代になって初めてわかった総数だ)。
こんなに多いと，およそ分類とか分析とかの研究には適さない。
六次・七次の標準型が幾つあるのかは，まだ誰も計数し尽くせていない。
高次の魔方陣では，解の総数が飛躍的に増える。これを恐れて「インフレーション」と我々は呼ぶ。研究のやりやすい方法がおよそ無いのである。

そこで，**標準型**を諦めて，別のもっと珍しいタイプの魔方陣を研究対象に選ばざるを得ない：例えば，**補数対称型**，**汎対角線型**，**完全型**，**正方連結型**など。また，**連立型**の**対称汎魔方陣**や**多重型**の魔方陣なども，研究対象にする。
しかし，**六次**以上になると，それもしだいに困難になる。**六次・十次・十四次**などでは，対称型も汎型も本来構成が不可能で，珍しいものを作りようが無いのだ。**七次**以上では，珍しいタイプでも解の総数が多い。コンピューターでも容易には答が出せない。
もっとずっと珍しいものを探さなければならない。

例えば一つは，「基本解・大基本解」を探す。もう一つは，「多重連立型」を作る方法が工夫されている。前者の一例が，「**ラテン方陣**」・「**オイラー方陣**」だ。これを構成法として用いることで，**七次～十三次**くらいまでがようやく射程圏内に入って来た。…

魔方陣の研究には，その他にも面白い研究課題がいろいろある。各次数・各タイプ別の個性的な話題もあれば，全てに共通して言える現象・法則・方法もある。
後者は，説明するにも理解するにも難しい話題が概して多いのだが，その中で小生が特別に気に入っているのは，第一に「**ひな形変換法**」と「**原初方陣**」。第二に「**完全オイラー方陣**」が来る。
詳しい報告を既に書いているので，どうか腰を据えて読んでみてほしい。魔方陣の世界ではそんなことまで成り立つのかと，あなたも不思議な思いにとらわれるに違いない。そう感じてもらえることを，小生は願っている。
どちらも，基本的な発想は，オイラー先生など先人が言い出したものだが，小生はそれを普遍的な現象と見て，三次から四次，五次，七次，八次，九次と系統的に調べて研究報告している。結局のところ元々のアイデアを相当変更する結果になったが，それを小生の僭越な「修正主義」と見るか，それとも理論的「必然」と見るべきかは，後世の評価に委ねることにしたい。

もう１つのトピック。次元の壁を超えて平面方陣・立体方陣・**超立体方陣**の相互連関を論じ，新発見を報告した数件は，これは最初から最後まで小生のオリジナルである。
例えば，汎四方陣の例の「**奇跡の一致**」の背後には，**四次元超立体二方陣**の根本存在が隠れていることがわかった。…
ただし，事柄の性質上，あまりにも抽象的な議論が続く。万人向きの話題とは必ずしも言い難い。追試・検証をしようにも，いかんせん理解を超えた部分があるかも知れない。
しかし，それでもこれは，訓練を受けた数学者ならば当然慣れている程度の初歩的な抽象的議論だと思う。どうか，最初から疑って眉ツバの怪しいものを見るような眼で見ないでほしい。小生の単なるタイプミスを，どうか致命的欠陥のように見ないでほしい。
小生としては，神の仕掛け賜うた謎に挑戦する使命を感じて興奮していた折から，超感

覚的な存在についてあえて論じざるを得なかったのである。
　恐れを知らぬアマチュアだからこそできる特権的な冒険だ。多少の哲学臭は，どうかご堪忍いただきたい。

（初稿：Dec.24, 2004;　改訂稿：Feb.23, 2006, May 18. 2014 and Jun. 7, 2015;
by 摂田 寛二(Kanji Setsuda)；計数・作表：with MacOSX 10.10.3 & Xcode 6.3.2）

E-Mail Address: jag12001@nifty.com
Homepage: http://kanjisetsuda.la.coocan.jp