§1. 例外群とは何か

リー群

＝群であって、群演算が微分可能であるもの

Sophus Lie

1842〜1899

ノルウェーの数学者

<u>例</u> $GL(n,\mathbb{R}) = n\times n$ 可逆行列の全体

行列の積、逆行列は微分可能

$$(AB)_{ij} = \sum_k a_{ik} b_{kj} \qquad A=(a_{ij}) \quad B=(b_{ij})$$

リー群はこれを抽象化したもの

複素数上でも同様の概念を考えることができる。

<u>例</u> $GL(n,\mathbb{C})$ 、 $SL(n,\mathbb{C})$ ＝ 行列式＝1の $n\times n$ 複素行列全体

一般線型群　特殊線型群

$$SO(n,\mathbb{C}) = \{ g\in GL(n,\mathbb{C}) \mid {}^t g\cdot g = 1_n \}$$

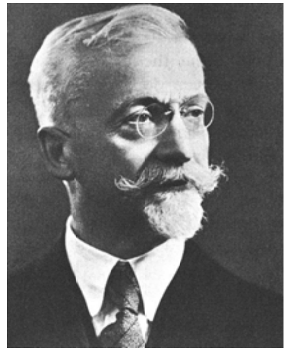
直交群

複素数上で単純リー群（厳密には単純リー環）の分類がある。

ここで 単純 …… 正規部分群を持たない。

それ以上 細かく分けることができないもの
他の群はそれで組み合わせて構成できる。
…… 原子／分子 のような関係

Killing 1847〜1923
ドイツ



Cartan 1869〜1951
フランス

分類表の中に 行列からできる群 ← 古典群と呼ばれる
としては理解できないものが現われた。

これが 例外群である。

古典群は無限系列であらわれる。例. $SL(n,\mathbb{C})$, $SO(n,\mathbb{C})$

A_n , B_n , C_n , D_n　　nは自然数

行列のサイズに対応する。

一方、例外群は有限個しかない。

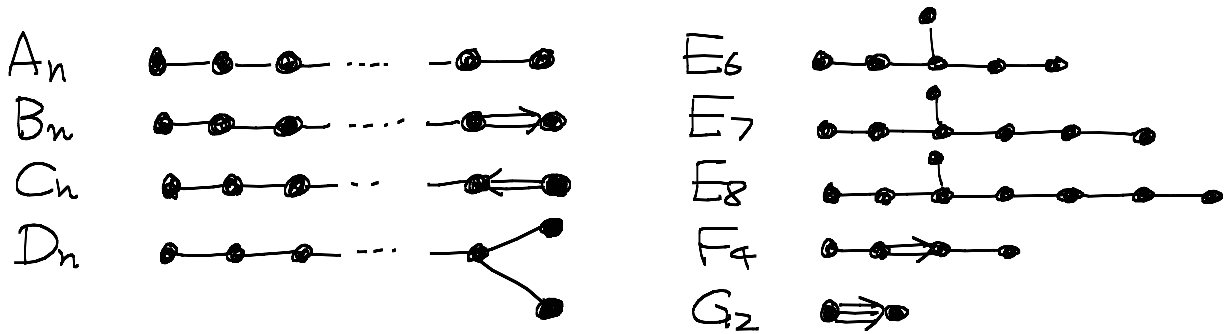
E_6 , E_7 , E_8 , F_4 , G_2

この中で次元が一番高いものが E_8 である。

(248次元!)

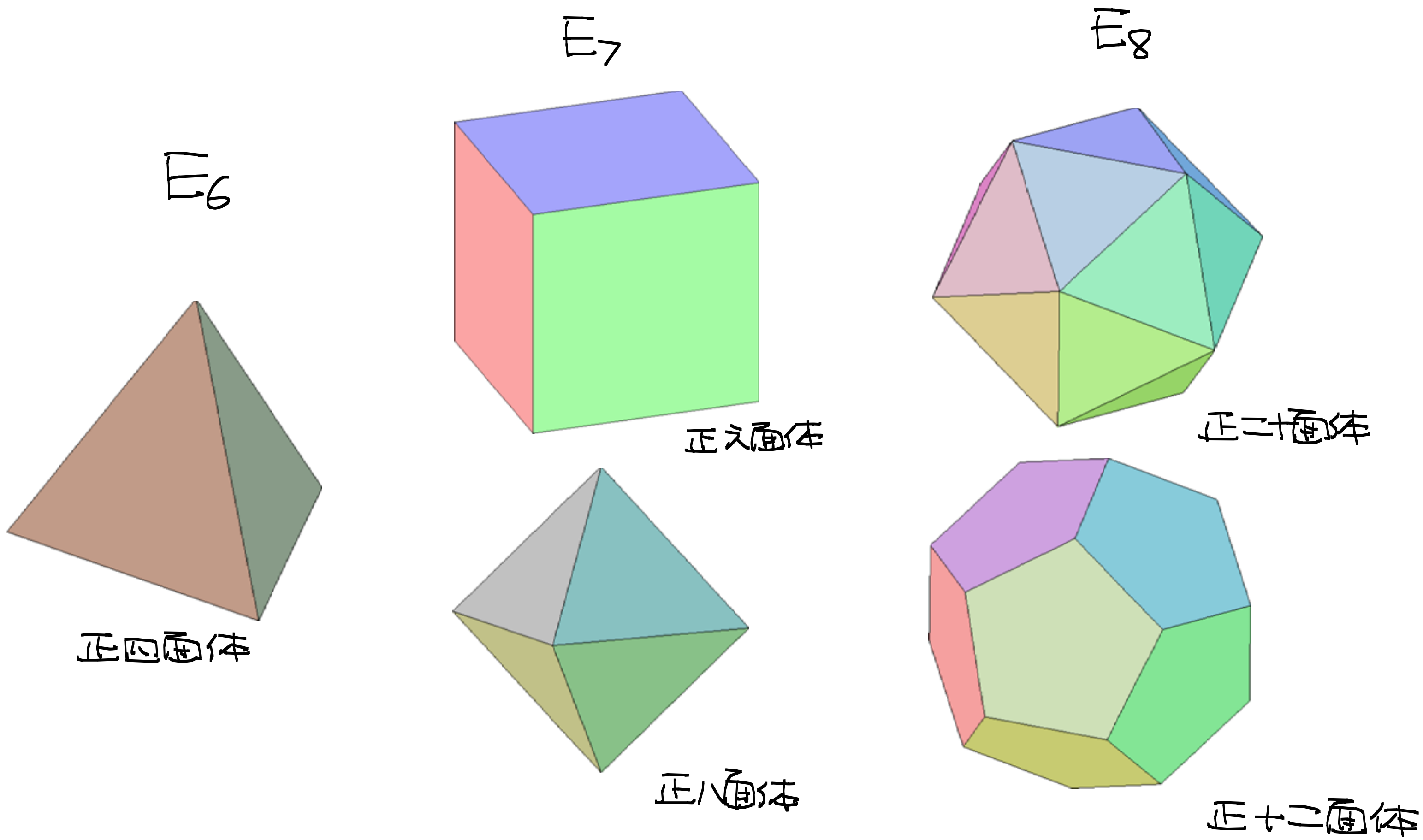
何らかの条件をみたすような 248行248列 の行列の全体となっている……

分類の詳細については，ここでは説明しない。
次のDynkin図式（Coxeter図式とも）が用いられる。

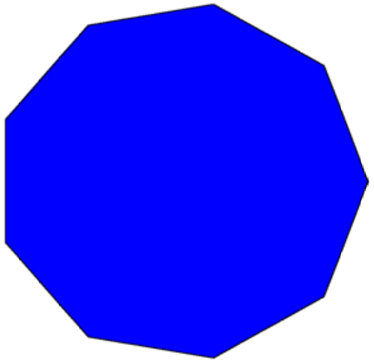


実は，同じ図式が他のところにもよく顔を出すことが知られている。

例えば，正多面体が関係していることが分かっている。
(Klein, Brieskorn, Slodowy, McKay, ---- 長い歴史がある。)



一方，古典群は"退化した"正多面体 = 正二面体 に対応する。



正n角形

— nはいくらでも大きくなりうる。

○ 正多面体が5通りしかないこと（プラトンはこれを知っていたといわれる）と 例外群が有限個しかないことには関係があるのだ。

○ 例外群は のけもの（例外!!）ではなく，美しいものである。

…… ケプラーは 正多面体と惑星（水，金，火，木，土）が対応すると考えたという。

E_8 はポピュラーになりつつある!?

○ Atlas プロジェクト（2007年5月）
http://www.liegroups.org/

Lusztig-Vogan 多項式をスーパーコンピュータで計算した。

（膨大な計算!!）

○ Antony Garrett Lisi : An exceptionally simple theory of everything（2007年11月）
http://en.wikipedia.org/wiki/An_Exceptionally_Simple_Theory_of_Everything

I think the universe is pure geometry - basically, a beautiful shape twisting around and dancing over space-time. Since E8 is perhaps the most beautiful structure in mathematics, it is very satisfying that nature appears to have chosen this geometry.

寄り道　有限群についても"単純"の定義があり，
有限単純群の分類が1980年代に完成している。

これについても　無限個の群からなる系列　と
有限個(26個)の散在群
がある。

散在群の中で，位数が最大(80801742479451287588 6459
90496 17107 57005 75436 8000000000)
のものは　モンスター　と呼ばれている。

ムーンシャイン(月光，バカらしい空想とも)と呼ばれるモジュラー関数との
全く予期されなかった関係が発見され，世界の数学者たちを驚かせた。

Atiyah Bull. of London Math. Soc. (2002)

1984年 インタビュー

Finite groups

This brings us to finite groups, and that reminds me: the classification of finite simple groups is something where I have to make an admission. Some years ago I was interviewed, when the finite simple group story was just about finished, and I was asked what I thought about it. I was rash enough to say I did not think it was so important. My reason was that [illegible] When things get closed down instead of getting opened up, I do not get so excited, but of course a lot of my friends who work in this area were very, very cross. I had to wear a sort of bulletproof vest after that!

There is one saving grace. I did actually make the point that in the list of the so-called 'sporadic groups', the biggest was given the name of the 'Monster'. I think [illegible] It turns out that the Monster is an extremely interesting animal, and is still being understood now. It has unexpected connections, with large parts of the rest of mathematics, with elliptic modular functions, and even with theoretical physics and quantum field theory. This was an interesting by-product of the classification. [illegible]

§2. なぜ例外群が難しいのか?

○ 古典群は基本的に行列と思ってよい。

　　　→ 具体的に計算ができる。

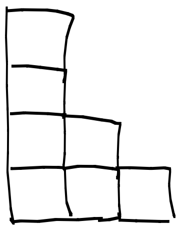
○ 例外群は構成も難しい。

しかし、どちらの場合にも適用する抽象的な構成がある。

(Serre)

○ 古典群を調べるための組合せ論がある。

ヤング図式,盤と呼ばれる



○ 例外群にはそのような特別のものはない

しかし、どちらの場合にも適用するアルゴリズムが

多くの問題の場合にある。

そのため表現論に関する問題で

古典群については ……ヤング図式 etc を使って解答する。

例外群については ……アルゴリズムを使って具体的に答を求めてしまう。

…… しばしば計算機が使われる。

アルゴリズム：計算の仕方を与えるが，本当に計算を実行するわけではない。

注意　古典群のときにもアルゴリズムは有効である。

しかし　$A_1, A_2, A_3, \cdots$　いつまでも終わらない。

$B_1, B_2, B_3, \cdots$

§3. 箙多様体のベッチ数

ここでは箙多様体もベッチ数も説明しない。

複素数体上の単純リー環(より一般にカッツ・ムーディー・リー環)

とその支配的ウェイトの組に箙多様体と呼ばれる空間が定義される。

Dynkin図式の○の上に自然数を乗せたもの

その各連結成分のベッチ数を計算する。

表現論に応用あり.

アルゴリズムがある。(2000年に発見した。)

⟶ 古典型 ---- ヤング盤で答を記述できた。

そこで例外型 ---- プログラムを組んでコンピュータですべて計算してやろうと考えた。

支配的ウェイトは基本ウェイト(どこかの●の上に1があり 他の●の上には0が乗っていて)のみ考えればよい。

何が難しかったのか？

① 計算量が膨大

連結成分の個数 ＝ 6899079264コ ≈ 6.4G

その次元が最大もの … 30次元（笑）

各連結成分ごとにベッチ数（奇数次は消えている）は（次元／2）＋1個ある。

② アルゴリズムは単純ではあるが、再帰的である。

（つまり、順番に計算する必要がある。
逆列計算（部分に分けて同時に計算すること）ができない。

再帰的なアルゴリズムの例 ： $n! = (n-1)! \times n$

```
 // 階乗を求める ( 再帰版 )
long Factorial(int n)
{
    if (n == 0 || n == 1)    // n が 0 か 1 なら
        return (1);          // 1 を返す
    else   // n に自分より 1 小さい自分を掛ける(これが再帰呼び出し)
        return (n * Factorial(n -1));
}
```

スーパーコンピュータは並列計算ができるときに最も威力を発揮する。

Rem. 数学でよく使われるコンピュータ・プログラム (Mathematica, Maple) や、より専門的なコンピュータ言語 (e.g. 代数計算用の Gap) は汎用性を求められており、特別な計算をするのには向いていない。

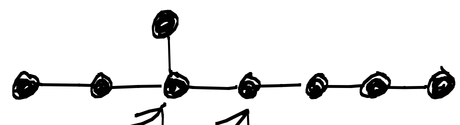
⟶ C言語を用いて自分でプログラミングを行った。

一番苦労したのは、ある単項式でパラメトライズされる各連結成分 (6,4G) をどのようにメモリに格納するか?

2001年に計算を始めたとき：

京大の計算機センターの大型計算機で
一回に使えるメモリの量は最大7GBだった。
（DVD一枚は 4.7GB）

E_8 の基本ウェイトのうち



これ以外は 計算が実行できた。

次に大きなこれは メモリを 3.2GB 必要とした。

一番大きな基本ウェイトについては，
以前の計算結果を覚えておくのには
十分でないので，計算が実行できなかった。

（上の2つの ● 以外については，パーソナルコンピュータでも
計算できる。

2006年に京大の大型計算機が新しくなって計算が実行できた。
200 G bytes までOK

結局　メモリ　：120 G bytes　が必要だった。

計算時間　：350時間
計算結果　：180 GB　のデータ
記憶するのに，パソコンの拡張ハードディスクが
まるまる一台必要！

<u>問題点</u>．
計算結果を"見る"ことができない。
⇝ これを使って何かできるわけでないので
そこから先に進むことができない。