Macaulay2 による計算代数幾何

安田 健彦 (大阪大学)

2012 年 10 月 16 日

第 1 章

はじめに

これはノートは、2012 年 10 月、首都大学東京での集中講義のためのものだ。

1.1 目標

この講義の目標は以下の通り。

1. Macaulay2 に慣れ親しむ。
2. 何が計算できるのか、おおざっぱに理解する。
3. 関数を書いて簡単なプログラミングができるようになる。
4. マニュアルの調べ方を学び、今後自分でスキルを磨けるようになる。

私自身は Macaulay2 を体系的に勉強したことはなく、見よう見まねでやっているうちに少しずつ覚えて、今では研究でもよく使うようになった。概して、このような方法がプログラミングを習得する近道なのだろう。（本格的なプログラマーになろうという人は別だろうが。）この講義でも、なるべく代数幾何や可換環論で実際に行われそうな計算例を扱いながら、スキルを身につけるように計画した。

個々人の必要とする計算は異なるし、時間の制約もあり、教えられることは限られる。この講義の後は、マニュアルを読んだり、他人のプログラムを参考にしたりしてスキルを磨いて欲しい。参考文献として、[2, 5] を挙げておく。また日本語の文献には、Knoppix/Math にも入っている [7] がある。

1.2 実験のススメ

数学研究における、コンピューターの使用は少しずつ増えてきているが、まだまだ「数学は紙と鉛筆でひたすら証明を積み上げる」というイメージが強いと思う。しかし、私はコンピューターを使った数値実験を大いにお勧めしたい。コンピューターを使ってできることとして、以下のことが挙げられる：

1. 予想を確かめる：特に、計算すれば間違っていることがすぐに分かる予想を、証明しようと時間を無駄にすることがなくなる。
2. 予想を立てる：計算結果を見て、法則を見つけるのは数学の醍醐味。未知の領域を開拓するには、観察するしかない。
3. 証明を補強する：現代数学の多くの証明は、難しく、本当に正しいのか不安が残る。数値実験がそれを補う。

これらは、すべて手計算についても言えることだが、コンピューターの守備範囲と手計算のそれでは全く異なる。一方で、コンピューターの計算には載らない理論も多い。数学では多くの無限を扱うが、計算では上手く無限を避けなければならない。今後は、数学も通常科学に近づき、理論と実験が 2 本の柱になるだろう。

1.3 予備知識

可換環論、代数幾何の基礎知識を仮定している。分からないところは、適宜標準的な教科書を見て欲しい。

1.4 演習の解答例

ノートの中の演習問題で☆マークのあるものには、ノート末尾に解答例がある。

1.5 起動、終了

この講義では、Macaulay2 は Emacs 上で使うことにする。

Macaulay2 を起動するためには、(本講義で受講生が使っているはずの) Knoppix/Math では数学ソフトのメニューから Macaulay2 を選ぶと Emacs と Macaulay2 が起動する。Emacs だけを起動して、F12 キーで Macaulay2 を起動しても良い。

Macaulay2 を起動すると、次のような画面が現れ、`i1 :` の右にカーソルが点滅し、入力待ちの状態になる。

```
+ M2 --no-readline --print-width 88
Macaulay2, version 1.4
with packages: ConwayPolynomials, Elimination, IntegralClosure, LLLBases,
               PrimaryDecomposition, ReesAlgebra, TangentCone

i1 :
```

たとえば、1+1 と入力しリターンキーを押すと、計算結果が以下のように出力される。

```
i1: 1+1

o1 = 2
```

プログラムを終了するためには `quit` と打てば良い。

```
i2 : quit

Process M2 finished
```

演習 1 F12 で Macaulay2 を再起動して、`quit` で終了せよ。

1.6 マニュアル

M2（以下、Macaulay2 をこのように略す）を使っていると、関数名の使い方を調べたり、やりたい計算をしてくれる関数を探したりする必要が出てくる。そのためにマニュアルを見なければならない。`viewHelp` でブラウザが起動し、オンラインマニュアルを見ることができる。このページをブックマークしておくと便利だ。また、特定の関数名、例えば `quit` について調べたいときは、`viewHelp quit` とすれば良い。また、マニュアルの Index ページ（マニュアルのトップページ上部のリンクをたどる）は関数を探すのによく使う。

また、M2 のウェブサイト（url: http://www.math.uiuc.edu/Macaulay2/）にも有益な情報があるので、チェックすると良い。オンラインマニュアルからリンクをたどっても行ける。

演習 2

1. `viewHelp quit` で `quit` のオンラインマニュアルを見よ。`help quit` と比べよ。
2. オンラインマニュアルから、二項係数を計算する関数を見つけよ。また、整数の割り算の余りを求める関数を見つけよ。
3. M2 ウェブサイトの左上サーチボックスから、Emacs に関する情報を見つけよ。

第 2 章

体、多項式環、イデアル、商環 – 必要最小限の準備

2.1 体

M2 の多くの計算は、まず考える環（通常、多項式環の商環）を設定しなければならない。商環の設定の仕方を順番に見ていこう。まず、係数体となるべき体について見ていく。QQ と入力してみよう。

```
i1 : QQ

o1 = QQ

o1 : Ring
```

これは有理数体を表している。以下のように有理数体内で計算ができる。

```
i2 : 2/3+6/7

     32
o2 = --
     21

o2 : QQ
```

有限体も使える。例えば 5 元体は

```
i3 : ZZ/5

     ZZ
o3 = --
      5
```

```
o3 : QuotientRing
```

この体の中で計算をするには、以下のようにすれば良い。

```
i10 : R = ZZ/5

o10 = R

o10 : QuotientRing

i11 : 4_R

o11 = -1

o11 : R

i12 : 3_R^5

o12 = -2

o12 : R
```

（おっと番号が飛んでしまった。今後は少々不格好だが、入出力の番号が飛んだり戻ったりしても気にしないで欲しい。）ここでは、`_R` **を付けることで、R の元を考えている。**

演習 3 有限体を用いて、様々な値でフェルマーの小定理（$a^{p-1} \equiv 1 \mod p$）を確かめよ。

係数体についての注意

M2 では大体、基礎体として素体 $\mathbb{Q}, \mathbb{F}_p$ を使うことが多い。これらの有限次拡大なども使えるが、使う頻度は低いだろう。実数体、複素数体も使えるが、多くの計算はこれらの体上ではできないため、あまり使わない。このように、多くの計算は、代数閉でない体上で行うことになるため、少し注意が必要。可換環論やスキーム論の基礎知識があれば、望ましい。

また私はあまり使わないが、標数 0 の問題に興味がある場合でも、M2 では大きな素数 p に対し、$\mathbb{F}_p$ 上で計算することがある。これは、計算を早くするため措置だ。多くの場合、計算結果は標数 0 の場合と一致する。p として Schenck の本 [5] では主に 101、[2] の中の Eisenbud の記事では、使える最も大きい素数 32749 が使われている。

演習 4 オンラインマニュアルを調べて、素体以外の有限体をどのように扱うか調べよ。

2.2 多項式環

多項式環を考えよう。

```
i14 : S  = QQ[x,y,z]

o14 = S

o14 : PolynomialRing
```

有理数係数の 3 変数多項式環だ。この環の元である多項式は以下のように入力する。

```
i19 : f = (2*x*y^3*z^2-5*x^6)*(x-y+z)^2

          8      7      6 2      7       6        6 2     3 3 2     2 4 2       5 2     2 3 3
o19 = - 5x  + 10x y - 5x y  - 10x z + 10x y*z - 5x z  + 2x y z  - 4x y z  + 2x*y z  + 4x y z  - 4x*y
      ------------------------------------------------------------------------------------------------
          3 4
      2x*y z

o19 : S
```

係数、変数の間にある**積の記号*が省略できない**ことに注意。次は有限体を係数とした計算。

```
i20 : T = ZZ/7[x,y,z];

i21 : g = (3*x+y-z)^7

        7    7    7
o21 = 3x  + y  - z

o21 : T
```

セミコロン; で出力を省略した。**もう一度、前の多項式環に戻したいときは** `use S` **とすれば良い。**

```
i22 : use S; (x+y)^7

       7     6       5 2      4 3      3 4      2 5       6    7
o23 = x  + 7x y + 21x y  + 35x y  + 35x y  + 21x y  + 7x*y  + y

o23 : S
```

セミコロンで、複数の文をつなげることができる。一番最後にセミコロンがなければ、最後の文が評価され出力される。

演習 5　1. `x^3+3*x^2*y+3*x*y^2+y^3 == (x+y)^3` と打ち、`true` と出力されることを確認せよ。マニュアルで==について調べよ。

2. 多項式の次数を返す関数をマニュアルから見つけ、使用せよ。☆

2.3　イデアル

設定した記号がごちゃごちゃしてきたら、**リセット**しよう。

```
i24 : clearAll
```

多項式環のイデアルを考える。

```
i25 : S = QQ[x,y]; I = ideal (x+y)^3

                3     2         2    3
o26 = ideal(x  + 3x y + 3x*y  + y )

o26 : Ideal of S
```

このイデアルの根基イデアルを計算してみる。

```
i27 : radical I

o27 = ideal(x + y)

o27 : Ideal of S
```

これを 3 乗したら元のイデアルに戻ることを確かめる。

```
i28 : I == oo^3

o28 = true
```

ここで、根基イデアルに名前を付け忘れたので、oo **で前の結果を再利用した。**oo **の代わりに、**o27 **などとして番号を明示すれば、もっと前の結果も利用できる。また、**ooo、oooo **で 2 つ前、3 つ前の結果も利用できる。**

演習 6　1. イデアル $(xy, x^2) \subset k[x, y]$ が根基イデアルでないことを M2 を使い確かめよ。☆

2. `isSubset` の使い方を調べて、イデアル (xy, x^2) が (x) に含まれることを確かめよ。

3. イデアルからグレブナー基底を計算する方法を調べよ。多項式の単項式順序はどのように変更したら良いか？

2.4 商環

多項式環をイデアルで割った商環を使おう。

```
i29 : S/I

                  S
o29 = ----------------------
       3     2        2    3
      x  + 3x y + 3x*y  + y

o29 : QuotientRing

i30 : dim oo

o30 = 1
```

この商環の次元は 1 だそうだ。いきなり、以下のように書いても良い。

```
i31 : R = ZZ/13[x,y,z]/( x+y-z^3, y^4-8*x^2+z^2)

o31 = R

o31 : QuotientRing
```

最後の環では $x+y-z^3=0$ となる。

```
i32 : x+y-z^3

o32 = 0

o32 : R
```

演習 7　1. `describe S` として、多項式環 S に含まれているデータを確認せよ。

2. いろいろな商環を定義して、その次元を `dim` で計算し、期待通りの値になることを確かめよ。また、それらが正規環かどうか M2 を使い確かめよ。(ヒント：パッケージ)
3. 関数 `ambient` について調べよ。この関数は商環に対してどのように働くか？
4. 同様に `Spec` について調べよ。

2.5 計算結果の保存と再利用

Emacs 上で M2 の計算結果を保存したい場合は C-x, C-w でファイルにテキストファイルを保存する。または、ツールバーをクリックして保存する。例えば aiueo.m2 などのように、拡張子は.m2 にする。

保存したファイルを Emacs で再度開くと色つきで表示される。計算結果（というより前回の入力）を再利用するには以下のようにする。F12 で M2 を別フレームで起動する。開いたファイルの任意の入力行で F11 を押すと、現在起動している M2 セッションに入力される。

演習 8 以上のことを試せ。

第 3 章

平面曲線の特異点と変曲点 – 手慣らし

3.1 特異点

まずは、いきなり Pascal の Limaçon(蝸牛形) という曲線を考える。

```
i1 : limacon = QQ[x,y]/((x^2+y^2-2*x)^2-(x^2+y^2))

o1 = limacon

o1 : QuotientRing
```

この平面曲線の特異点を計算する。

```
i2 : singularLocus limacon

                                                   QQ[x, y]
o2 = ------------------------------------------------------------------------------------------
       4     2 2    4     3       2     2    2    3       2      2     2         2      3
     (x  + 2x y  + y  - 4x  - 4x*y  + 3x  - y , 4x  + 4x*y  - 12x  - 4y  + 6x, 4x y + 4y  - 8x*y - 2

o2 : QuotientRing
```

これは商環の形をしていて、しかも分母のイデアルは根基でないかもしれない。

```
i3 : radical ideal oo

o3 = ideal (y, x)

o3 : Ideal of QQ[x, y]
```

これで、特異点は原点のみであることが分かった。ここで、oo で前の出力を再利用できる。radical ideal は oo に関数 ideal を適用してから、radical を適用している。次に、この曲線の射影閉包を考えよう。

```
i5 : Plimacon = QQ[x,y,z]/((x^2+y^2-2*x*z)^2-(x^2+y^2)*z^2)

o5 = Plimacon

o5 : QuotientRing
```

変数 z で斉次化した。また特異点を計算しよう。今度は一気に 3 つの関数を適用する。

```
i6 : radical ideal singularLocus Plimacon

                        2    2
o6 = ideal (y*z, x*z, x  + y )

o6 : Ideal of QQ[x, y, z]

i7 : dim oo

o7 = 1
```

次元は 1 。これはアフィン錐を考えているからで、つじつまは合う。singularLocus は長い名前で打つのが面倒だが、sing まで打って残りは **TAB キーで補完できる**。候補が複数ある場合は、別フレームに候補が表示されるので、希望のものをクリックすれば良い。（これらは、Emacs を使っている場合の話。）また **Control+上下キーでこれまでの入力を再利用できる。**

既約分解してみよう。

```
i8 : decompose o6

               2    2
o8 = {ideal (x  + y , z), ideal (y, x)}

o8 : List
```

o6 で第 6 出力を再利用した。イデアル (y, x) は原点の特異点に対応。もう片方のイデアルは、無限遠直線 $z = 0$(斉次座標 x, y) の $x^2 + y^2 = 0$ で定義される 2 点を与える。2 点は実数点ではない。

演習 9

1. 定義式の斉次化を自動で行え。☆
2. Cayley の 6 次曲線 $4(x^2 + y^2 - x)^3 = 27(x^2 + y^2)^2$ の射影閉包の特異点を全て求めよ。
3. Roman 曲面 $x^2y^2 + y^2z^2 + z^2x^2 + xyz = 0$ の non-normal locus を M2 で求めよ。（ヒント：パッケージ、斉次化）☆

3.2 変曲点

平面曲線 $C = (f = 0) \subset \mathbb{P}^2$ を考える。点 $p \in C$ が**変曲点**とは、非特異点であり、点 p での C の接線を L とすると、L と C が P で 3 重以上に接する（交点の重複度が 3 以上）となること。変曲点は Hessian determinant

$$\det\left(\frac{\partial f}{\partial x_i \partial x_j}\right)_{i,j=0,1,2}$$

で求められる。偏微分を計算するのに使える関数をマニュアル内で検索すると、`jacobian` が見つかる。

```
i37 : jacobian ideal Plimacon

o37 = {1} | 4x3+4xy2-12x2z-4y2z+6xz2 |
      {1} | 4x2y+4y3-8xyz-2yz2       |
      {1} | -4x3-4xy2+6x2z-2y2z      |

                           3                   1
o37 : Matrix (QQ[x, y, z])  <--- (QQ[x, y, z])

i38 : jacobian transpose oo

o38 = {-3} | 12x2+4y2-24xz+6z2 8xy-8yz          -12x2-4y2+12xz |
      {-3} | 8xy-8yz           4x2+12y2-8xz-2z2 -8xy-4yz       |
      {-3} | -12x2-4y2+12xz    -8xy-4yz         6x2-2y2        |

                           3                   3
o38 : Matrix (QQ[x, y, z])  <--- (QQ[x, y, z])

i39 : hess = det oo

            6       4 2       2 4       6        5         3 2           4         4 2        2 2 2
o39 = - 288x  - 864x y  - 864x y  - 288y  + 1152x z + 2304x y z + 1152x*y z - 1440x z  - 1728x y z
      ----------------------------------------------------------------------------------------------
          3 3         2 3       2 4      2 4
      576x z  + 576x*y z  + 216x z  - 72y z

o39 : QQ[x, y, z]
```

曲線上で Hessian determinant が消えている点を計算する。

```
i41 : J = ideal Plimacon + ideal hess; decompose J
```

```
o41 : Ideal of QQ[x, y, z]

                                 2    2                    2     2
o42 = {ideal (y, x), ideal (z, x  + y ), ideal (x - 3z, y  + 5z )}

o42 : List
```

得られたリストのうち、最初の 2 つは特異点に対応する。最後の 1 つが 2 つの変曲点に対応する。

演習 10　1. Cayley の 6 次曲線の変曲点を求めよ。

2. $f = x - 2(xz + y^2)y - (xz + y^2)^2 z$, $g = y + (xz + y^2)z$ とする。このとき、$x \mapsto f$, $y \mapsto g$, $z \mapsto z$ で定義される $\mathbb{C}[x, y, z]$ の自己準同型のヤコビ行列式を計算せよ。(キーワード：永田の同型、ヤコビ予想)

第 4 章

Frobenius 写像、Kunz の定理と Peskine-Szpiro の定理 – 環準同型と加群を使う

4.1 Frobenius 写像と Kunz の定理

この章では Frobenius 写像と Kunz の定理を題材に、Macaulay2 での環準同型や加群の扱いを練習しよう。まず、標数 3 における Whitney の傘という曲面を考える。

```
i89 : whitney = ZZ/3[x,y,z, Degrees => {3,2,2}]/(x^2-y^2*z);
```

後のために、次数環とするために x, y, z の次数を $3, 2, 2$ と設定した。

演習 11 これの特異点集合は $x = y = 0$ となることを `singularLocus` を使い確認する。

この特異点集合を別のやり方で計算してみよう。

一般に標数 p の環 R に対して、Frobenius 写像 $F : R \to R,\ f \mapsto f^p$ は環準同型となる。また素体 $\mathbb{F}_p$ の Frobenius 写像は恒等写像であることに注意する。Macaulay2 では、環の準同型は次のように定める。

```
i90 : whitney' = ZZ/3[x,y,z, Degrees => {9,6,6}]/(x^2-y^2*z);

i91 : use whitney; F = map(whitney, whitney', {x^3,y^3,z^3})

                                2    3   3
o92 = map(whitney,whitney',{x*y z, y , z })

o92 : RingMap whitney <--- whitney'

i95 : isWellDefined F

o95 = true
```

```
i96 : isHomogeneous F

o96 = true
```

後のために、環準同型が次数を保つように、もうひとつの環 `whitney'` を用意して次数を変えた。x, y, z の行き先 x^7, y^7, z^7 を与えることで環準同型を定める。ただし、商環の場合は well-defined かどうかは別にチェックする必要がある。また `isHomogeneous` で斉次準同型であることも確かめた。

Kunz の定理は**「環 R が正則（非特異）⇔Frobenius 写像が平坦」**というもの。これは、写像 $F : R \to R$ において、右の R を左の R 上の加群とみたときに（つまり、push-forward F_*R）、それが平坦（局所自由）ということだ。加群の push-forward を計算する関数 `pushForward` があるが、これは環、環準同型、加群が全て斉次の場合の適用できる。（斉次でなくても Frobenius 写像による pushforward を計算する（ただし、係数体は素体）関数が私のホームページに置いてある。url: http://takehikoyasuda.jimdo.com/）F_*R を計算しよう。

```
i97 : FR = pushForward (F, whitney^1)

o97 = cokernel {0} | 0  0  0  0  0  0  0  0  0  0  0  0  0  0  |
               {3} | x  0  0  0  0  0  0  yz 0  0  0  0  0  0  |
               {5} | 0  x  0  0  0  0  0  0  yz 0  0  0  0  0  |
               {7} | 0  0  0  0  0  0  -z 0  0  0  0  0  0  -x |
               {7} | 0  0  0  -y 0  0  0  0  0  0  -x 0  0  0  |
               {9} | 0  0  0  0  -y 0  0  0  0  0  0  -x 0  0  |
               {5} | 0  0  -y 0  0  0  0  0  0  -x 0  0  0  0  |
               {7} | 0  0  0  0  0  -y 0  0  0  0  0  0  -x 0  |
               {2} | 0  0  x  0  0  0  0  0  0  yz 0  0  0  0  |
               {4} | 0  0  0  x  0  0  0  0  0  0  yz 0  0  0  |
               {6} | 0  0  0  0  x  0  0  0  0  0  0  yz 0  0  |
               {8} | 0  -y 0  0  0  0  0  0  -x 0  0  0  0  0  |
               {4} | 0  0  0  0  0  x  0  0  0  0  0  0  yz 0  |
               {6} | -y 0  0  0  0  0  0  -x 0  0  0  0  0  0  |
               {2} | 0  0  0  0  0  0  0  0  0  0  0  0  0  0  |
               {4} | 0  0  0  0  0  0  x  0  0  0  0  0  0  y2 |

                                                    16
o97 : whitney'-module, quotient of whitney'
```

`whitney^1` は環 `whitney` 上の階数 1 の自由加群を表す。行列は自由加群の間の写像を表し、その余核として加群が表示されている。

演習 12 この加群の階数が $3^2 = 9$ であることを M2 で確認せよ。

この加群 FR が平坦でない場所を特定できれば、それが特異点集合と一致するはずだ。そのために、Fitting イデアルを使えば良い。一般に加群 M と非負整数 i に対し、M の i 次 Fitting イデアルは M が局所的に i 個の元で生成されない場所を定める。我々の例で F_*R は階数 $3^2 = 9$ なので、9 次 Fitting イデアルが平坦でない場所を定める。

```
i98 : radical fittingIdeal(9,FR)

o98 = ideal (y, x)

o98 : Ideal of whitney'
```

期待したとおりの結果が出た。この計算は時間がかかる。非力なコンピュータなら計算がなかなか終わらないかもしれない。その場合は、**Control+C を 2 回押して、計算を中断しよう。**

演習 13 1. 標数 3 のカスプ曲線 $y^2 = x^3$ で、M2 と Kunz の定理を用いて、特異点の位置を確かめよ。

2. 標数 0 の曲面 $y(y-x)(y-2x) - xz^2 = 0$（単純楕円型特異点）に対し、微分形式の加群 $\Omega_{R/k}$ を M2 で構成せよ。また、$\bigwedge^2 \Omega_{R/k}$ を計算せよ。(ヒント：`jacobian`、`exteriorPower` を使う。)

4.2 Peskine-Szpiro の定理

定理 1 (Peskine-Szpiro の定理) R を正標数のネーター環、M を有限生成 R 加群とする。M の射影次元は有限と仮定する。このとき、$\mathrm{Tor}_i^R(M, F_*R) = 0,\ i > 0$。

この定理を具体例で確かめる。

```
i1 : cusp = ZZ/3[x,y,Degrees=>{2,3}]/(y^2-x^3); cusp' = newRing(cusp,Degrees=>{6,9});
```

とし、カスプ特異点を定義する。そしてフロベニウス写像を前回同様に定義。F_*R を計算する。(途中の計算は省略する)

```
i8 : FR = pushForward(F,cusp^1)

o8 = cokernel {0} | y  0  0  x2 0  0  |
              {2} | 0  y  0  0  x2 0  |
              {4} | 0  0  y  0  0  x2 |
              {7} | 0  0  -x 0  0  -y |
              {5} | 0  -x 0  0  -y 0  |
              {3} | -x 0  0  -y 0  0  |

                                     6
o8 : cusp'-module, quotient of cusp'
```

ここで 2 つの加群を考える。

```
i9 : use cusp'; M1 = coker matrix{{x,y}}; M2 = coker matrix{{x}};

i15 : res(M1,LengthLimit => 10)

           1          2          2          2          2          2          2          2          2
o15 = cusp'  <-- cusp'  <-- cusp'  <-- cusp'  <-- cusp'  <-- cusp'  <-- cusp'  <-- cusp'  <-- cusp'

      0          1          2          3          4          5          6          7          8

o15 : ChainComplex

i16 : res(M2,LengthLimit => 10)

           1          1
o16 = cusp'  <-- cusp'  <-- 0

      0          1          2

o16 : ChainComplex
```

M1 は射影次元が無限であることが推測され、M2 は射影次元が 1 だと分かる。LengthLimit **を設定しないと、自由分解が変数の数までの長さで打ち切られるので注意が必要。同じ理由で射影次元を計算する** pdim **にも注意。**多項式環上で考える場合は、気にする必要はない。

演習 14

1. 可換環論の教科書を見て、実際に M2 の射影次元が無限であること理解する。
2. 2 つの加群の Tor 加群を計算して Peskine-Szpiro の定理が成り立っていることを確認する。Tor の計算についてはマニュアルを調べる。
3. ChainComplex C に対して、C.dd で境界作用素を見ることができる。M1 の自由分解の境界作用素の周期性を観察せよ。
4. 同様の計算を標数 2、3 次元の簡単な特異点で行う。

第 5 章

イデアルの冪と随伴素イデアル – 関数作成

この章では、M2 によるプログラミングをごく簡単に学ぶ。プログラミングを体系的に最初から学ぼうとすると、時間ばかりかかって嫌になってしまう。そこで、本講義では準備は最小限にとどめて、実用的な例でどんどん慣れていくことを目指す。さらに、学びたい人はマニュアルを読んだり、[2, 5] に載っている例の真似をしたりして、少しずつスキルを上げていけば良いと思う。

5.1 関数を作ろう

もし、研究で同じような計算を、少しずつ違うデータで繰り返す場面によく出くわす。毎回、同じような操作を繰り返すのは億劫なので、コンピュータに自動でやってもらいたい。そのために、**関数**を書けば良い。M2 のプログラミングは基本的に関数をたくさん書いていくことになる。

まずは、計算対象のセッティング。

```
i36 : R = ZZ/3[t,x,y]/(x^4+t*x^2*y^2+y^4);

i41 : I = ideal(x,y);

o41 : Ideal of R
```

(この例は論文 [6] から取った。)

演習 15 `isPrime` を使い、`R` は整域、`I` は素イデアルであることを確認せよ。

では、簡単な関数を書いてみよう。2 乗を計算する関数を下のように定義する。

```
i17 : square = i -> i^2

o17 = square

o17 : FunctionClosure
```

`square` が関数名で `i` が与えられたら、`i^2` を返す関数だ。このように**新しい関数は、**

```
functionName = input -> output
```

という形で定義する。関数名や一般に変数名（多項式環の変数という意味ではなく、cusp など、計算対象に付けた名前）は、(半角) 英数字とアポストロフ “’”が使える。最初の文字は数字以外でなければならない。環や加群に R や M と名付けるのは別にして、fittingIdeal のように、小文字で初めて、単語の区切りは大文字で表すのが慣例のようだ。

定義した関数は、以下のように利用できる。

```
i21 : square(-3)

o21 = 9

i42 : square I

               2        2
o42 = ideal (x , x*y, y )

o42 : Ideal of R
```

演習 16　1. 与えられた整数を 5 で割った余りを求める関数を作れ。☆

2. 与えられたイデアルの根基の 2 乗を計算する関数を作れ。

5.2 関数の合成

複数の関数を組み合わせることで、より複雑な関数を定義することができる。

自分で定義した別の関数を合成しても良いが、ここではイデアルの随伴素イデアルを求める既存の関数 associatedPrimes を使おう。イデアルを 2 乗してからその随伴素イデアルを求めるのを、一度に計算してくれる関数は、少し冗長だが以下のように定義できる。

```
i46 : f = I -> (
      J := square I;
      associatedPrimes J);

i47 : f I

o47 = {ideal (y, x)}

o47 : List
```

この関数では、まず I の 2 乗を計算し J :=で局所変数（関数内だけで有効）J に割り当てる。これは関数の出力とならずに、セミコロン; で次の計算へと続く。最後の associatedPrimes J の結果が関数の出力となる。

同じ関数を、関数の合成演算子@@を使い、次のように定義することもできる。

```
i50 : f = associatedPrimes @@ square;
```

しかし、この方法は個人的にはあまり使わない。少し複雑な関数になると、@@を使うより、セミコロンで次々と局所変数を定めていく方法の方が簡単なようだ。

演習 17 1. 与えられた商環の特異点集合の既約成分を求める関数を書け。☆

2. 多項式から Hessian determinant を求める関数を書け。
3. 与えられたイデアルの随伴イデアルの数を求める関数を作れ。(ヒント：List の元の数を求める関数は？)
4. 与えられた加群の台の次元をもとめる関数を作れ。

5.3 多変数関数

複数の入力を持つ関数も定義できる。n 乗の随伴素イデアルを求める関数を書こう。

```
i55 : assPrimePower = (I,n) -> associatedPrimes (I^n);

i57 : assPrimePower (I,3)

o57 = {ideal (y, x)}

o57 : List
```

演習 18 この関数を使い、n を変化させて、現れる随伴素イデアルを調べよ。(Brodmann により、n を大きくしていくと、現れる随伴素イデアルの集合は一定になることが証明されている。)

次に通常の冪ではなく、Frobenius 冪を考えよう。イデアル I の e 次 Frobenius 冪 $I^{[p^e]}$ とは f^{p^e}, $f \in I$ で生成されるイデアルのことをいう。I が $f_1, \ldots, f_l$ で生成されるとき、$I^{[p^e]}$ は $f_1^{p^e}, \ldots, f_l^{p^e}$ で生成される。Frobenius 冪を計算する関数を書こう。

```
i69 : frobeniusPower = (I,e) -> (
      p := char ring I;
      G := flatten entries gens I;
      G' := apply(G, i -> i^(p^e));
      ideal G');
```

G はイデアルの生成元のリスト。apply(**リスト , 関数) はリストの各元に、関数を適用したリストを返す。**

演習 19 1. 上の関数定義に使われた関数の働きをマニュアルで調べ、ちゃんと期待通りの定義になっていることを確認せよ。特に apply を自分の例で試してみよ。(関数 gens は generators の省略形。)

2. assPrimePower のように、フロベニウス冪の随伴素イデアルを求める関数を定義せよ。そして、e が大きくなると随伴素イデアルの数を増えていくことを確認せよ。([6] により、表れる随伴素イデアルの数は無限であることが知られている。フロベニウス冪に無限の随伴素イデアルが表れる例は最初に

Katzman により与えられた。)

5.4 関数の保存と再利用

せっかく作った関数を、Macaulay2 のセッションを一度終了して、再開した後も使いたいだろう。そのためには、たとえば以下のようなテキストファイルを作り、MyFuncs.m2 などという名前で適当な場所に保存する。

```
assPrimePower = (I,n) -> associatedPrimes (I^n);

frobeniusPower = (I,e) -> (
      p := char ring I;
      G := flatten entries gens I;
      G' := apply(G, i -> i^(p^e));
      ideal G');
```

そして、セッション再開後、以下のようにファイルをロードすれば、関数を使える。

```
i7 : load "/Users/highernash/MyFuncs.m2"
```

Macaulay2 の動いているディレクトリを `currentDirectory()` で調べて、そこにファイルを保存しておけば、`load "MyFuncs.m2"`とファイル名だけで良い。

演習 20 やってみよ。

第 6 章

$\mathbb{Q}$-Cartier 因子 – もう少し関数。そして、if、for、while

6.1 if, for, while – 条件分岐とループ

多くのプログラミング言語と同様に、M2 でも if, for, while は基本的な構文だ。if で条件分岐することができ、for と while で同じ計算を繰り返すループを行うことができる。ごく簡単に、これらの使い方を見てみよう。

if

イデアル $I \subset R$ が与えられたときに、I が素イデアルならば整域 R/I の分数体を求め、そうでなければエラーメッセージを返す関数は以下のように書ける。

```
i24 : quotField = I -> (
      if isPrime I
      then frac(ring I / I)
      else "This is not a prime ideal.");
```

ただし、改行は見やすくするためのもので、しなくても良い。if p then x else y という構文では、p が評価されその値が Bool 値 true なら x を評価して返し、false なら y を評価して返す。else 以降は無くても良い。(その場合、p が false なら null を返す。)

演習 21 入力 k に対し、それが体なら 1 変数多項式環 $k[t]$ を返し、そうでなければ「これは体ではない」と返す関数を作れ。

for

まずは一例。

```
i5 : for i from 1 to 10 list 2^i

o5 = {2, 4, 8, 16, 32, 64, 128, 256, 512, 1024}
```

```
o5 : List
```

見れば分かると思うが、i が 1 から 10 まで 2^i を並べたリストを得る。

次の例。

```
i11 : j = 1; for i to 10 do (j = j^2+1); j

o13 = 2067332040424216771816347187340368893772743793833577167934058658233170953687756572895428933 7
      6781635914484312173817557988370878548961245582641826131216636641404794790451610519266609830 0
      2113611309425141433175164838870296570198482099598937133539604307050417213011286620929104962 2
      8032616591791135372780375257785844367023767613423607865994296575429774169891418160743 30
```

$j = 1$ からスタートし 2 乗して 1 足すという操作を 11 回繰り返した。(from を省略すると 0 からになるので 11 回。)

演習 22 1. 300 以下の素数のリストを作れ。(ヒント：continue) ☆

2. 2 つの 1 変数多項式 f, g にユークリッドの互除法を n 回適用後に得られる多項式の組 f', g' を返す関数を求めよ。n 回までに最大公約数が得られる場合はそれを返すようにせよ。

while

```
i22 : i = 0; while i < 10^10 list (i=i^2+1;i)

o23 = {1, 2, 5, 26, 677, 458330, 210066388901}

o23 : List
```

最初に $i = 0$ とおき、$i < 10^{10}$ であるかぎり、2 乗して 1 足すという操作を繰り返し、得られた数のリストを返す。while p list x do y は条件 p が満たされる限り、x を評価した結果をリストにし、y を実行する。

次の例では、条件部分を true にしているので、常に条件が満たされている。ループを抜けるのは、if ... break で行う。

```
i30 : smoothSingularCenter = I -> (
      R := ring I;
      currentI := I;
      while true do (
        singI := radical ideal singularLocus currentI;
        if singI == ideal (1_R) then break;
        currentI = singI);
      currentI);
```

演習 23 この関数は何を計算しているのか理解せよ。

6.2 2-Cartier 判定

次の 2 次錐とそのイデアルを考える。

```
i101 : qCone = QQ[x,y,z]/ideal(x*y-z^2)

o101 = qCone

o101 : QuotientRing

i102 : I = radical ideal(y)

o102 = ideal (z, y)

o102 : Ideal of qCone
```

このイデアルの定める既約因子 D は Cartier ではない。これは、以下のように確かめられる。

```
i105 : M = module I

o105 = image | z y |

                                        1
o105 : qCone-module, submodule of qCone

i106 : radical fittingIdeal(1,M)

o106 = ideal (z, y, x)

o106 : Ideal of qCone
```

イデアルを加群と見たときに、原点で平坦でないことが分かった。しかし $2D$ は Cartier である。これは $(M^{\otimes 2})^{\vee\vee}$ が平坦であることを以下のように確かめれば良い。

```
i117 : dual dual (M**M)

o117 = image {1} | 1 |

                                        1
o117 : qCone-module, submodule of qCone
```

演習 24 3 次 Fermat 型超曲面 $R = \mathbb{C}[x, y, z]/(x^3 + y^3 + z^3)$ において、イデアル $\sqrt{(x + y)} \subset R$ で定義され

る既約因子は 2-Cartier でないことを確認せよ。

2-Cariter かどうかチェックする関数 `is2Cartier` を書く。因子を与えるイデアル `I` が与えられたときに、上と同様の操作で 2-Cartier かどうかを確かめる。

```
i10 :  is2Cartier = I -> (
       M := module I;
       M2 := M**M;
       DDM2 := dual dual M2;
       J := fittingIdeal(1,DDM2);
       R := ring M;
       J == ideal 1_R);
```

このように、途中の計算結果を`:=`で分かり易い変数名に割り当てて、次々と計算していくと良い。最後の`==`は両辺が等しいかどうかを Bool 値 (`true` or `false`) で判定する。両辺が同じ環のイデアルの場合は、イデアルの表示（生成元）が違っても等しいイデアルなら等しいと判定する。ここでは、グレブナー基底を使うとこのようなことができるのだが、その理論的背景は適当な教科書を読んで欲しい。

以下は上の関数の適用例。

```
i11 : is2Cartier I

o11 = true

i23 : R = QQ[x,y,z]/ideal(x*y-z^5); I = radical ideal y;

o24 : Ideal of R

i25 : is2Cartier I

o25 = false
```

6.3 Cartier 指数の計算

次に各自然数 r に対し、r-Cartier かどうかを順番に確かめていく関数を書こう。

```
i40 :  cartierIndex = (I,N) -> (
  M := module I;
  r:=1;
  while r <= N do (
    Mr := M^**r;
    DDMr := dual dual Mr;
    J := fittingIdeal(1,DDMr);
```

```
    R := ring M;
    if J == ideal 1_R then break;
    r = r+1);
  if r <= N then r else 0);

i41 : cartierIndex(I,6)

o41 = 5
```

上の関数では `while` **ループ**を用いた。これは

```
while condition do Z
```

という形をしている。`condition` が満たされる限り、`Z` が評価され続ける。上の具体例では、$r = 1$ から初めて、一回ループが回るごとに r を 1 増やす。$r = N + 1$ となるか、途中で r-Cartier になれば `break` でループを抜ける。このループが終了後、$r \leq N$ なら r の値を返し、$r = N + 1$ なら 0 を返す。

演習 25

1. `cartierIndex` を while の代わりに for を使い書き直せ。
2. 正規商環（または正規射影多様体（次章））が与えられたときに、その Gorenstein 指数を求める関数を書け。
3. 代数多様体 $X \subset \mathbb{C}^n$ が与えられたとき、射影 $\mathbb{C}^n \ni (x_1, \ldots, x_n) \mapsto (x_1, \ldots, x_{n-1}) \in \mathbb{C}^{n-1}$ により X の像を取るという操作を繰り返し、最初に特異になる像を求める関数を作れ。（射影は代数的には変数の消去に対応する。）☆

第 7 章

曲線の特異点解消 – アルゴリズム

7.1 カスプ特異点の解消

Spec$R$ のイデアル $I \subset R$ に関する爆発は $\mathrm{Proj}(\bigoplus_{n=0}^{\infty} I^n)$ で定義されるのだった。ここに現れる環（次数付き R 代数）を Rees 環という。カスプの原点での爆発を計算してみよう。

```
i2 : R = ZZ/101[x,y]/(y^2-x^3);

i3 : reesAlgebra ideal(x,y)

                   R[w , w ]
                      0   1
o3 = ------------------------------------
                     2              2    2
     (y*w  - x*w , x w  - y*w , x*w  - w )
         0      1     0      1     0    1

o3 : QuotientRing
```

このままでは、爆発の性質（非特異かどうかなど）を調べにくいので、アフィン被覆を取ろう。この例では、2 つのアフィン開集合で被われていて、それぞれのアフィン開集合は $w_i = 1$ と置くことで得られる。例えば $w_0 = 1$ とすると。

```
i6 : Bl = o3; use Bl; R0 = Bl / ideal(w_0 - 1)

o8 = R0

o8 : QuotientRing

i9 : describe R0
```

```
                    R[w , w ]
                       0   1
      -------------------------------------
                     2              2    2
      (y*w  - x*w , x w  - y*w , x*w  - w )
          0      1     0      1     0    1
o9 = ----------------------------------------
                        w  - 1
                         0

i27 : minimalPresentation R0

       ZZ
o27 = ---[w ]
      101  1

o27 : PolynomialRing
```

このアフィン開集合はアフィン直線に同型、特に非特異であることが分かる。

特異点解消のデータとしては、アフィン開集合の座標環だけでなく、アフィン開集合から元の曲線への射のデータも欲しい。

```
i35 : f = map(R0,R)

o35 = map(R0,R,{x, y})

o35 : RingMap R0 <--- R

i42 : g = R0.minimalPresentationMap

           ZZ                  2   3
o42 = map(---[w ],R0,{1, w , w , w })
          101  1           1   1   1

               ZZ
o42 : RingMap ---[w ] <--- R0
              101  1

i43 : g*f

           ZZ         2   3
```

```
o43 = map(---[w ],R,{w , w })
          101  1      1   1

               ZZ
o43 : RingMap ---[w ] <--- R
              101  1
```

演習 26 イデアル $I \subset R$ が与えられたら、I での爆発のアフィン被覆の座標環 S_i に対し、準同型 $R \to S_i$ のリストを返す関数を作れ。このときに、`minimalPresentation` ☆

7.2 アルゴリズムの作成と実装

爆発してアフィンチャートを計算して、特異点が残っていたらまた爆発する。これを繰り返せば、特異点解消ができる。ただし、これをコンピューターで計算するには、もっと細かい手順をきちんと書き下さなければならない。いきなり、プログラムを書き始めるのは大変なので、まずは日本語でアルゴリズムを書き下そう。

入力：商環 R（整域）
出力：環準同型 $f_i : R \to R_i$ の集まり $\{f_1, \ldots, f_l\}$。ただし、$\operatorname{Spec} R_i$ はある特異点解消のアフィン被覆、f_i はその特異点解消から定まるものとなっている。
アルゴリズム：

1. SmoothCharts $= \emptyset$, SingularCharts $= \emptyset$, WaitingCharts $= \{(R, id_R)\}$ と置く。
2. WaitingCharts が空ならばステップ 4 へ。空でなければ WaitingCharts から最初の $f : R \to S$ を選び、WaitingCharts から取り除く。特異点集合を定める根基イデアル $J \subset S$ を計算、$J = (1)$ なら f を SmoothCharts に加え、そうでなければ (f, J) を SingularCharts に加える。ステップ 3 へ。
3. SingularCharts が空ならば、ステップ 2 へ。そうでなければ、SingularCharts からひとつの (f, I) を選んで、SingularCharts から取り除く。S を I で爆発。そのアフィン被覆、そして写像を計算。得られた写像全てを WaitingCharts に加える。
4. SmoothCharts を出力する。

演習 27 上のアルゴリズムを M2 の関数として実装せよ。☆

演習 28 できた関数を以下のように修正せよ。

1. 一度に 1 つの極大イデアルで爆発させる
2. 各アフィンチャートが何回目の爆発で得られたか記録する。

第 8 章

射影多様体と層のコホモロジー

8.1 Hodge 数

計算の題材として Fermat 型 4 次超曲面 $(x_0^4 + \cdots + x_3^4 = 0) \subset \mathbb{P}^3$ を考えよう。これの斉次座標環を、少し今までと違うやり方で書いてみよう。

```
i5 : R = QQ[x_0..x_3]/(sum(4,i->x_i^4))

o5 = R

o5 : QuotientRing
```

このように変数の多い多項式環は、添え字を使うことで短く書くことができる。`sum(n,f(i))` は $f(0) + f(1) + \cdots + f(n-1)$ を計算する。

さて、斉次環が与えられると、`Proj` で対応する射影多様体を作ることができる。

```
i6 : X = Proj R

o6 = X

o6 : ProjectiveVariety
```

環のままの方が扱いやすいこともあるが、以下のように層のコホモロジーの計算は射影多様体にしたほうが良い。

まず X が非特異であることを確認しよう。

```
i7 : singularLocus X

         /QQ[x , x , x , x ]\
         |    0   1   2   3 |
o7 = Proj|------------------|
         \         1        /
```

```
o7 : ProjectiveVariety
```

分母が 1 なので、特異点集合が空集合であることが分かる。

X の Hodge 数を計算しよう。`hh^(p,q)` で計算できる。全ての Hodge 数を同時に求めるには以下のようにすれば良い。

```
i10 : for p to 2 list (for q to 2 list hh^(p,q)(X))

o10 = {{1, 0, 1}, {0, 20, 0}, {1, 0, 1}}

o10 : List
```

これは $H^q(\Omega_X^p)$ の次元を計算している。これらの適当な交代和を取れば位相的 Euler 標数が求まるが、M2 では `euler` で計算できる。ただし、上の理由で X が非特異であることは自分で確かめなければならない。また `genus` で算術種数が計算できる。これは、X が非特異である必要はない（と思う）。

演習 29

1. 多様体の次元、次数、定義式をいろいろ変えて Hodge 数を計算してみよう。どのような例ではすぐに計算が終わり、どのような例では計算が終わらないか見てみよう。
2. `hh` を改良して、多様体が特異の場合にはエラーメッセージを返すような関数を書け。

8.2 連接層のコホモロジー

いくつかの標準的な連接層は簡単に射影多様体から作ることができる。まず、（捻れ）構造層。

```
i13 : OO_X

o13 = OO
        X

o13 : SheafOfRings

i14 : OO_X(3)

         1
o14 = OO  (3)
        X

o14 : coherent sheaf on X, free
```

記法から、その意味は明らかだろう。

次に、接層、余接層、余接層の外冪。

```
i20 : TX = tangentSheaf X

o20 = image {-2} | -x_0x_2^3   x_1x_2^3     x_2^4+x_3^4 0           x_1x_3^3     -x_0x_3^3   |
            {-2} | x_1^4+x_2^4 x_0^3x_1     x_0^3x_2    -x_1x_3^3   0            x_2x_3^3    |
            {-2} | x_0x_1^3    -x_1^4-x_3^4 -x_1^3x_2   -x_0x_3^3   x_2x_3^3     0           |
            {-2} | -x_1^3x_3   -x_0^3x_3    0           x_2^4+x_3^4 x_0^3x_2     x_1^3x_2    |
            {-2} | x_2^3x_3    0            x_0^3x_3    x_1x_2^3    x_0^3x_1     x_1^4+x_3^4 |
            {-2} | 0           x_2^3x_3     -x_1^3x_3   x_0x_2^3    -x_1^4-x_2^4 x_0x_1^3    |

                                                 6
o20 : coherent sheaf on X, subsheaf of OO  (2)
                                                X


i21 : cotangentSheaf X;

i22 : cotangentSheaf(2,X);
```

また、任意の連接層を捻りたければ、例えば TX(3) などとすれば良い。(接層と $\mathcal{O}_X(3)$ のテンソル積。)

また得られた連接層 F,G から、F ++ G、F ** G、F ^ n、exteriorPower(n,F)、dual F、sheafHom(F,G)、sheafExt^n(F,G) で新しい連接層を作ることができる。最初の 2 つは直和、直積、n コピーの直和を意味する。あとは記法から意味を推測して欲しい。

連接層のコホモロジー群は HH^n で計算できる。

```
i130 : HH^1(TX)

         20
o130 = QQ

o130 : QQ-module, free
```

演習 30 Serre の双対定理と消滅定理、小平の消滅定理を好きな例で確認せよ。

8.3 加群から連接層を作る

もっと別の連接層を作りたければ、斉次座標環上の次数付き加群から作る方法がある。例として、2 つの一般の超平面切断 $H_1, H_2 \subset X$ の交わり $H_1 \cap H_2$ の構造層（$\mathcal{O}_X$ 加群として）を構成しよう。以下のように斉次座標環 R から、次数 1 の斉次元をランダムに 2 つ取ってくる。

```
i31 : f=random(1,R)

            1      5
```

```
o31 = x  + -x  + -x  + 8x
       0   2 1   8 2     3

o31 : R

i32 : g=random(1,R);
```

次に、行列 (f, g) で定義される自由加群の写像 $R^2 \to R$ の余核を定義して、それに付随する連接層と取れば求めるものが得られる。

```
i34 : M = coker matrix{{f,g}}

o34 = cokernel | x_0+1/2x_1+5/8x_2+8x_3 1/5x_0+7/3x_1+2/9x_2+5/4x_3 |

                                 1
o34 : R-module, quotient of R

i38 : F = sheaf M

o38 = cokernel | x_0+1/2x_1+5/8x_2+8x_3 1/5x_0+7/3x_1+2/9x_2+5/4x_3 |

                                              1
o38 : coherent sheaf on X, quotient of OO
                                             X

i39 : HH^0(F)

        4
o39 = QQ

o39 : QQ-module, free
```

演習 31 1. 座標環 R を次数 3 以上の部分だけ取り出した加群をつくれ。`truncate` を使う。

2. この加群から連接層 `F` を作り `HH^0(F(*))` を計算せよ。
3. `module F` と比べよ。

演習 32 この章の扱っている、X の標準層 $\omega_X = \Omega^2_X$ が自明な可逆層であることを、同伴公式 (adjunction formula) を用いずに以下の手順で示せ。

1. ω_X から `module` で対応する斉次座標環上の次数加群を作る。
2. この加群をもちいて、ω_X の一般の超平面への制限を求める。
3. Riemann-Roch の公式とコホモロジーの計算より、ω_X の超平面への制限が次数 0 であることを示す。

4. $\dim H^0(\omega_X) = 1$ を確かめる。

演習 33（研究課題）与えられた 2 つの連接層の同型か判定する関数を作成せよ。（有限次元加群の場合はアルゴリズムがあるみたいなので、それを参考にすればできると思うが：[4][1]）

第 9 章

その他

この章では、講義では扱わなかったが、知っておくと良いかも知れない内容にいくつか触れておく。詳しく知りたい人は、マニュアルの「The Macaulay2 language」のところを読んで下さい。

9.1 ハッシュテーブル

ハッシュテーブルは鍵と値の組の集まりで、鍵を指定して値を取り出すことができる。

```
i1 : legends = new HashTable from
     {"Hilbert" => "David", "Euler"=>"Leonhard", "Newton"=>"Issac"}

o1 = HashTable{Euler => Leonhard}
               Hilbert => David
               Newton => Issac

o1 : HashTable

i2 : legends#"Hilbert"

o2 = David
```

演習 34 国を鍵として、首都を値とするハッシュテーブルを作れ。

9.2 パッケージを使う

専門的な計算を扱う関数はパッケージにまとめられている。M2 のバージョン 1．4 では、起動時点で ConwayPolynomials, Elimination, IntegralClosure, LLLBases, PrimaryDecomposition, ReesAlgebra, TangentCone のパッケージが読み込まれるので、これらに含まれる関数は使うことができる。ただし、関数の説明はマニュアルの各パッケージの部分にあり、Macaulay2Doc の Index には載っていないので注意。

これら以外のパッケージを使いたければ、それを読み込むようにコマンドを打たなければならない。たとえば深度や Cohen-Macaulay 性を扱うパッケージ Depth を使い、商環が Cohen-Macaulay かどうか以下のよ

うに確かめられる。

```
i1 : loadPackage "Depth";

i3 : isCM (QQ[x,y]/ideal(x*y))

o3 = true

i4 : isCM (QQ[x,y,z]/intersect(ideal(x,y),ideal(z)))

o4 = false
```

自分でパッケージを作ることも出来る。私はやり方を知らないがマニュアルに書いてある。

9.3 クラス

M2 では全てのものが**クラス（型）**を持っている。

```
i5 : class (QQ[x])

o5 = PolynomialRing

o5 : Type

i6 : class {1,2,3}

o6 = List

o6 : Type
```

関数の振る舞いは、変数 (入力) のクラスに依存する。例えば関数 `ideal` は商環に適用すると、商環を定めるイデアルを返すし、多項式に適用すると、それが生成するイデアルを返す。

自分でクラスを作ることも出来る。良い例が「Macaulay2Doc > mathematical examples > Tutorial: Divisors」にある。ここでは、代数多様体上の因子を扱う新しいクラスを作っている。

9.4 ファイルの読み書き

ファイルに計算結果を書き出したり、ファイルを読み込んで関数に適用したり出来る。でも、私はやったことがない。マニュアルを読んで下さい。

第 10 章

演習解答例

演習 5.2

関数 degree を使う。

演習 6.1

```
i10 : QQ[x,y]; I = ideal(x*y,x^2);

o11 : Ideal of QQ[x, y]

i12 : I == radical I

o12 = false
```

演習 9.1

（例）２変数多項式環の元 f を substitute を使い３変数多項式環に移し、substitute(f,z) で新しい変数 z で斉次化：

```
i13 : QQ[x,y]; f = x^2+y+1;

i15 : f = substitute(f,QQ[x,y,z]);

i16 : homogenize (f,z)

       2          2
o16 = x  + y*z + z

o16 : QQ[x, y, z]
```

演習 9.3

整閉包の写像のコンダクター・イデアルが non-normal locus を定義する。しかし、関数 `conductor` は斉次ケースしか使えないので斉次化した Roman 曲面を考える。

```
i36 : homoRoman = QQ[x,y,z,w]/(x^2*y^2+y^2*z^2+z^2*x^2+x*y*z*w);

i37 : I = conductor icMap homoRoman

o37 = ideal (y*z, x*z, x*y)
```

$w = 1$ とすれば、希望のイデアルが得られる。

演習 16.1

```
i1 : waru5 = n -> n % 5;
```

演習 17.1

```
i32 : singularComponents = R -> (
        singR := singularLocus R;
        singIdeal := ideal singR;
        decompose singIdeal);
```

演習 22.1

```
i2 : for i to 300 list (if isPrime i then i else continue)

o2 = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61,
     ------------------------------------------------------------------------
     67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137,
     ------------------------------------------------------------------------
     139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211,
     ------------------------------------------------------------------------
     223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283,
     ------------------------------------------------------------------------
     293}

o2 : List
```

演習 25.3

```
isSmooth = I -> (
     R := ring I;
     singI := radical ideal singularLocus I;
     singI == ideal 1_R);

singularProjection = I -> (
     R := ring I;
     J := I;
     while true do (
          R := ring J;
          Vs := flatten entries vars R;
          if length Vs == 1 then break;
          lastVar := last Vs;
         J = radical eliminate(lastVar,J);
         Vs' := drop(Vs,-1);
         KK = coefficientRing R;
         R = KK[Vs'];
         J = sub(J,R);
         if not isSmooth J then break;
         );
     J);
```

演習 26

```
blowupCharts = {Variable => w} >> o -> I -> (
     R := ring I;
     rees := reesAlgebra(I,Variable => o.Variable);
     reesAmb := ambient rees;
     G := gens reesAmb;
     l := length G;
     charts := for w in G list rees/ideal(w-1_rees);
     RToCharts := for T in charts list map(T,R);
     for T in charts list (minimalPresentation T);
     minMaps := for T in charts list T.minimalPresentationMap;
     for i to l-1 list ((minMaps_i) * (RToCharts_i))
     );
```

次の演習のために変数名をオプションで変えられるようにした。

演習 27

```
desing = R -> (
     smCharts := {};
     singCharts := {};
     waitingCharts := {id_R};
     numBlowups := 0;

  while waitingCharts != {} do (
   f := waitingCharts_0;
   waitingCharts = drop(waitingCharts,1);
   S := target f;
   singIdeal := radical(sub(ideal singularLocus S,S));

   if singIdeal == ideal(1_S)
     then smCharts = append(smCharts,f)
     else singCharts = append(singCharts, (f,singIdeal));

  while singCharts != {} do (
       (g,J) := singCharts_0;
       singCharts = drop(singCharts,1);
       numBlowups = numBlowups + 1;

       newMaps = for h in blowupCharts(J,Variable=> vars(numBlowups))
                            list (h*g);
       waitingCharts = join(waitingCharts, newMaps);
       );

       );

     smCharts
     );
```

参考文献

[1] P. A. Brooksbank and E. M. Luks. Testing isomorphism of modules. **J. Algebra**, 320(11):4020–4029, 2008.

[2] D. Eisenbud, D. R. Grayson, M. Stillman, and B. Sturmfels, editors. **Computations in algebraic geometry with Macaulay 2**, volume 8 of **Algorithms and Computation in Mathematics**. Springer-Verlag, Berlin, 2002.

[3] D. R. Grayson and M. E. Stillman. Macaulay 2, a software system for research in algebraic geometry. Available at http://www.math.uiuc.edu/Macaulay2/.

[4] K. M. Lux and M. Szőke. Computing decompositions of modules over finite-dimensional algebras. **Experiment. Math.**, 16(1):1–6, 2007.

[5] H. Schenck. **Computational algebraic geometry**, volume 58 of **London Mathematical Society Student Texts**. Cambridge University Press, Cambridge, 2003.

[6] A. K. Singh and I. Swanson. Associated primes of local cohomology modules and of Frobenius powers. **Int. Math. Res. Not.**, (33):1703–1733, 2004.

[7] 横田博史. Macaulay2 の紹介 Available at http://durian2.math.kobe-u.ac.jp/KnoppixMath-doc/ponpoko/Macaulay2.pdf