目次

2 文字式の操作

2.1 整式・分数式の計算

展開	expand( )
因数分解	factor( )
単純化	simplify( )
x の多項式 $f(x)$ を x に関し整理	collect(f(x) ,x)
展開し，結果を既約分数にする。	normal( )
整式 $f(x)$ を整式 $g(x)$ で割ったときの商と剰余	divide(f(x),g(x))
多項式 f(x,y) を多項式 g(x,y) で割ったときの商と剰余を x についての整式と見て求める。	divide(f(x),g(x),[x])

注1)

この節は特にたくさんあるので, 軽く流して見て後で必要になったら辞書代わりに使いましょう。例を見て見ましょう。

2.1.1 整式の展開・因数分解

$(x^2-4x+5)(x^2-3)$ は? • `(x ^ 2 - 4x + 5) * (x ^ 2 - 3);` >> $(x^2-4x+5)(x^2-3)$

展開するには expand( ) を使います。

上の式の展開は? • `expand(%);` >> $12x+2x^2-4x^3+x^4-15$

$(a+b+c)^2$の展開は? • `expand((a + b + c) ^ 2);` >> $2ab+2ac+2bc+a^2+b^2+c^2$

a に関し整理したいときは collect(,a) を使います。

a に関し整理すると? • `collect(%, a);` >> $2bc+a^2+b^2+c^2+a(2b+2c)$

因数分解には factor( ) を使います。

上の式の因数分解は? • `factor(%);` >> $(a+b+c)^2$

確かにもとの式に戻りました。いろいろ因数分解してみましょう。

x^4-8x^2-9 の因数分解は?

• `factor(x ^ 4 - 8 * x ^ 2 - 9);` >> $(x-3)(x+3)(x^2+1)$

$x^2+4xy+3y^2+x+5y-2$ の因数分解は?

• `factor(x ^ 2 + 4 * x * y + 3 * y ^ 2 + x + 5 * y - 2);` >> $(x+3y-1)(x+y+2)$

注1) factor は’ 約数’ の意味です。また collect は’ 集める’，divide は’ 割る’ という意味ですね。

検算してみます。2 変数の場合は，次数の低い文字に関し整理するのでした。x に関し整理すると

$$x^2+4xy+3y^2+x+5y-2=x^2+(4y+1)x+(3y^2+5y-2)=x^2+(4y+1)x+(y+2)(3y-1)$$

さらに，たすぎ掛けをして $(x+3y-1)(x+y+2)$ となり一致します。実は，2 次式の場合は解の公式を使って，$ax^2+bx+c=0(a\neq 0)$ の 2 解を α,β とするとき，

$$ax^2+bx+c=a(x-\alpha)(x-\beta)$$

と因数分解できます。(数　) しかし，じつは MuPAD は,5 次式以上でも (5 次以上の方程式では，解の公式は存在しない), 有理数係数の式に因数分解してくれます。
$x^5+x^4+2x^3+2x^2-3x-3$ の因数分解は?

- `factor(x ^ 5 + x ^ 4 + 2 * x ^ 3 + 2 * x ^ 2 - 3 * x - 3);`　　>> $(x-1)(x^2+3)(x+1)^2$

MuPAD は、解の公式を使わずに，どうやって解いているのでしょうか。われわれと同じように代入していっているのでしょうか？どうやら違うみたいです。[注2)] これについては大学へ行ってから勉強してください。

2.1.2　整式の割り算

整式 $f(x)$ を整式 $g(x)$ で割ったときの商と余りは divide(f(x),g(x)) で求めます。

$2x^3-12x+9$ を $(x+3)$ で割ったときの商と余りは?

- `divide(2 * x ^ 3 - 12 * x + 9, x + 3);`　　>> $2x^2-6x+6,\ -9$

これは商が $2x^2-6x+6$, 余りが -9 ということです。

$$\begin{array}{r|l} & 2x^2\ -6x\ \ +6 \\ \hline x+3 & 2x^3\ \ \ \ -12x\ +9 \\ & \underline{2x^3+6x^2\ \ \ \ \ \ } \\ & \ \ \ -6x^2-12x \\ & \ \ \ \underline{-6x^2-18x\ \ } \\ & \ \ \ \ \ \ \ 6x\ +9 \\ & \ \ \ \ \ \ \ \underline{6x+18} \\ & \ \ \ \ \ \ \ \ \ -9 \end{array}$$

2 変数の場合は, どの文字に関する式と考えるかによって結果が変わってきます。このような時は divide(f,g,[x]) のように変数を指定します。

$8x^3+y^3$ を x^2+y^2 で割ったときの商と余りを x の整式と見て計算すると?

- `divide(8 * x ^ 3 + y ^ 3, x ^ 2 + y ^ 2, [x]);`　　>> $8x,\ y^3-8xy^2$

$8x^3+y^3$ を x^2+y^2 で割ったときの商と余りを y の整式と見て計算すると?

- `divide(8 * x ^ 3 + y ^ 3, x ^ 2 + y ^ 2, [y]);`　　>> $y,\ 8x^3-x^2y$

$$\begin{array}{r|l} & 8x \\ \hline x^2+y^2 & 8x^3\ \ \ \ \ \ +y^3 \\ & \underline{8x^3\ \ +8xy^2\ \ \ } \\ & \ \ \ \ \ -8xy^2\ \ +y^3 \end{array}$$

$$\begin{array}{r|l} & y \\ \hline y^2+x^2 & y^3\ \ \ \ \ \ +8x^3 \\ & \underline{y^3\ \ \ \ \ \ +x^2y} \\ & \ \ \ \ \ 8x^3-x^2y \end{array}$$

注2) 立命館大学の倉田氏に教えていただきました。

2.1.3 分数式の計算

分数式を簡単にしたいとき (約分・通分) は normal( ) をつかいます。結果の式は展開されて表示されます。

$\frac{x^2+3x-4}{2x^2-4x+2}$ を約分してみましょう。

- `(x ^ 2 + 3 * x - 4)/(2 * x ^ 2 - 4 * x + 2);` >> $\dfrac{3x+x^2-4}{2x^2-4x+2}$
- `normal(%);` >> $\dfrac{x+4}{2x-2}$

$\frac{3x+x^2-4}{2x^2-4x+2} = \frac{(x+4)(x-1)}{2(x-1)^2} = \frac{x+4}{2(x-1)}$ ですから同じです。通分のときも使えます。

$\frac{2x-3}{x^2-3x+2} - \frac{3x-2}{x^2-4}$ を通分した後，約分してみましょう。

- `(2 * x - 3)/(x ^ 2 - 3 * x + 2) - (3 * x - 2)/(x ^ 2 - 4);` >> $\dfrac{2x-3}{x^2-3x+2} - \dfrac{3x-2}{x^2-4}$
- `normal(%);` >> $\dfrac{4-x}{x+x^2-2}$
- `factor(%);` >> $-\dfrac{x-4}{(x-1)(x+2)}$

実際,

$$\frac{2x-3}{x^2-3x+2} - \frac{3x-2}{x^2-4} = \frac{(2x-3)(x+2)-(3x-2)(x-1)}{(x-1)(x-2)(x+2)} = -\frac{x^2-6x+8}{(x-1)(x-2)(x+2)}$$
$$= -\frac{\cancel{(x-2)}(x-4)}{(x-1)\cancel{(x-2)}(x+2)} = -\frac{x-4}{(x-1)(x+2)}$$

ですから同じです。このように normal( ) は分母・分子を展開形にします。さらに，factor( ) を使うと，分母・分子を因数分解できます。

2.2 関数の定義, 変数の代入

(式) に y という名前をつける	`y := (式)`
関数 $f(x)$ を, y という名前をつけて定義する	`y := x -> f(x)`
関数 $f(x,y)$ を, z という名前をつけて定義する	`z := (x,y) -> f(x,y)`
f という式において, x に a を代入	`subs(f,x = a)`
f という式において, x に a, y に b を代入	`subs(f,x = a,y = b)`
代入した式 f の評価	`eval(f)`
定義 f の消去 (関数, 式とも)	`delete(f)`

注3)

注3) '=' でなく, ':=' であることに注意。':=' は変数に値を代入する。またはその変数を定義するときに使う。(GIDITS:=20 などと同じ) 関数の定義の'- >' は, 矢印 (⟶) を表している。例えば，'$x \longrightarrow x^2$' で，'x に x^2 を対応させる関数' という感じになる。また， subs() は substitute(代入する) の略。eval() は evaluate(評価する) の略。

2.2.1 式に名前をつける.

式に名前をつけるのは':=' を使います。例えば x+y に f という名前をつけたいならば，f:= x+y とします。式に名前をつけるのは，同じ式を繰り返し用いたいとき (いろいろなコマンドを同じ式に作用させたいとき) です。例えば，前節で $\frac{3x+x^2-4}{2x^2-4x+2}$ を normal() を使って，約分させましたがもし，simplify() を使ったらどうなるか知りたいとします。このような時は適当な名前[注4)] を使って，次のようにすると入力が楽です。

$\frac{3x+x^2-4}{2x^2-4x+2}$ に f という名前をつける。

- `bunsu := (3 * x + x ^ 2 - 4)/(2 * x ^ 2 - 4 * x + 2);` >> $\dfrac{3x+x^2-4}{2x^2-4x+2}$

式の名前は出力されません。これを normal() で約分してみましょう。

- `normal(bunsu);` >> $\dfrac{x+4}{2x-2}$

これは前節で normal(%) とした結果と同じです。でも次にもとの分数式を，今度は simplify() を使って簡単にしてみましょう。

- `simplify(bunsu);` >> $\dfrac{x+4}{2x-2}$

二つは，同じ結果となりました。[注5)] もしここで simplify(%) では (直前の式)= $\frac{x+4}{2x-2}$ を simplify する事になりますので，比較したことにはなりません。

2.2.2 関数に名前をつける。

[注6)] 関数を定義しておくと，グラフを描いたり，いろいろな数字を代入したい時に便利です。関数の定義は矢印 '− >' を使います。例えば $f(x) = x^2$ という関数を， nijo という名前をつけて，定義したいならば

- `nijo := x -> x ^ 2;` >> $x \longrightarrow x^2$

2 つ以上の変数のときも同様です。x と y の関数として $\frac{x+y}{2}$ を heikin と名をつけて定義するのは，次のようにします。

- `heikin := (x, y) -> (x + y)/2;` >> $(x, y) \longrightarrow \dfrac{x+y}{2}$

2.2.3 代入

式への代入は，subs (f,x=a),subs(f,x=a,y=b) を用います。[注7)] 一方，関数 f(x) に x=a を代入したい時は，単に f(a); とするだけです。

注4) MuPAD により予約されている名前 (PI,simplify など) と，最初に数字が来る名前 (1x など),_ 以外の記号などを使った名前は定義できません。数字が後に来る場合 (x1 など) や,_を使った名前 (x_1 など) は大丈夫です。

注5) 実は約分に関しては simplify と normal は同じになることが多いみたいです。

注6) 同じ数式 $f(x)$ を，関数としても，式としても定義できます。代入したり，グラフを描いたりしたいときは関数として定義しておくと良いでしょう。一方, 前節のように $f(x)$ そのものをさらに変形したいようなときは，単に式に名前をつけておくのが良いでしょう。

注7) subs(f,x=a,y=b) は a⟶ b,b⟶c などという代入をするのでなければ,subs(f,[x=a,y=b]), subs(f,{x=a,y=b}) と書いても同じです。

$\frac{x+y}{x-y}$ を’式’と考え, f と名前をつけて，いろいろな値を代入してみましょう。

- `f := (x + y)/(x − y);` >> $\frac{x+y}{x-y}$

まず x=1 を代入してみましょう。

- `subs(f, x = 1);` >> $\frac{1+y}{1-y}$

次は x= 0 を代入してみましょう。

- `subs(f, x = 0);` >> -1

$x = 0$ のとき, $\frac{x+y}{x-y} = \frac{y}{-y} = -1$ ですから合っています。次は,x=1,y=2 を代入してみます。

- `subs(f, x = 1, y = 2);` >> -3

文字式を代入することも出来ます。x に a^2, y に a を代入してみましょう。

- `subs(f, x = a ^ 2, y = a);` >> $\frac{a+a^2}{a^2-a}$ • `normal(%);` >> $\frac{a+1}{a-1}$

このように，代入した式は普通の式と同様に扱えます。

これに対し，関数として名前をつけると，代入は普通のコマンドとまったく同様にできます。先ほど定義した nijo := x −> x ^ 2; と heikin := (x,y) −> (x+y)/2; を使ってやってみましょう。

- `nijo(4);` >> 16
- `heikin(4, 8);` >> 6

4 の 2 乗は 16 で，4 と 8 の平均は 6 ですね。このようにまったく普通のコマンドと同じように扱えます。文字を代入することも出来ます。

- `heikin(2 * a + 3, 6 * a + 1);` >> $4a + 2$

確かに，$\frac{(2a+3)+(6a+1)}{2} = 4a + 2$ ですね。

注8)

注8) subs(f,x=a) と f(a) は，ほとんど同じですが次のような場合は違います。

- `subs(sin(x)), x = PI;` >> $sin(PI)$

このような場合は eval() を使って，評価してください。

- `eval(%);` >> 0

2.2.4 関数にするか，式にするか?

今までにも述べましたが，いろいろな変形をしたいときは式として定義し，グラフを描いたり，代入したりしたいときは関数として定義すると良いでしょう。一度，関数として定義すると，その内容は変えることができません。

- kansu := x -> x + 4 - 5; >> $x \longrightarrow x + 4 - 5$
- simplify(%); >> $x \longrightarrow x + 4 - 5$

このように全く変わりません。これに対し式として定義すると

- siki := x + 4 - 5; >> $x \longrightarrow x - 1$

初めから簡単になっています。先に見たようにこの後の変形も自由です。

2.2.5 定義の消去

定義が残っていると，エラーのでることがあります。このような時は，delete() で定義を消去します。

- y := x ^ 2; >> x^2

このように y を定義したことを忘れて，うっかり y を含んだ式を計算させると，結果がおかしくなります。

- expand((y + x) ^ 2); >> $x^2 + 2x^3 + x^4$

$y = x^2$ なので，$(x + y)^2 = x^2 + 2xy + y^2$ にはなりません。$(x + x^2)^2 = x^4 + 2x^3 + x^2$ になります。y の定義を消去してから，もう一度やってみましょう。

- delete(y) : expand((x + y) ^ 2); >> $2xy + x^2 + y^2$

注9)

2.2.6 【参考】継ぎはぎ関数の定義

$f(x) = \begin{cases} x < 0 \text{ のとき} & -x \\ x \geqq 0 \text{ のとき} & x^2 \end{cases}$ のような，いわゆる継ぎはぎ関数の定義には piecewise([f_1],[f_2],$\cdots$) を使います。例えば，上の関数を f という名前をつけて定義するには, 次のようにします。

- f := piecewise([x < 0, - x],[0 <= x, x ^ 2]); >> $-x$ if $x < 0$, x^2 if $0 <= x$

ここで矢印 (– >) を入れて，x– >piecewise としないでください。

注9) いろいろな定義を一度に全て消去したいときは，reset(); を使います。( ) 内は空白。

2.3 【参考】変数の条件設定と変数のタイプ

$x > a$ と条件設定	assume(x > a)
$x \geqq a$ と条件設定	assume(x >= a)
$x < a$ と条件設定	assume(x < a)
$x \leqq a$ と条件設定	assume(x <= a)
$a < x < b$ と条件設定	assume(a < x < b)
条件 B を追加し，B との共通集合に変更	assume(B, _and)
条件 B を追加し，B との和集合に変更	assume(B, _or)
x を, Type :: *name* というタイプに設定	assume(x, Type :: name)
x の設定解除	delete(x);

タイプ	Type :: name
実数	Type :: Real
有理数	Type :: Rational
複素数	Type :: Complex
整数	Type :: Integer
偶数	Type :: Even
奇数	Type :: Odd

注10) MuPAD では変数のタイプがいろいろあり，文字を使った計算では思った結果が出ないことがあります。例えば abs(x ^ 2); としても x にはなってくれません。注11) これは MuPAD のほうで, x が正の数なのか, 負の数なのか, はたまた複素数なのか解らないからです。このような時は,assume() を使って条件を設定します。

$x > 0$ と設定するには?	• assume(x > 0);	>> > 0
このとき, $\|x\|$ は?	• abs(x);	>> x
$x < 0$ と設定するには?	• delete(x) : assume(x < 0);	>> < 0
このとき, $\|x\|$ は?	• abs(x);	>> $-x$

delete() で先の条件を消去してから, 新たに条件を設定しました。注12) 条件を追加するときは _and と _or のオプションを指定します。それぞれ前の条件との共通集合, 和集合を作ります。

- assume(x > 0) : assume(x < 5, _and); >>]0, 5[of Type::Real

]0,5[で $0 < x < 5$ を表します。ちなみに [0,5] で $0 \leqq x \leqq 5$ を表します。Type::Real というのは実数タイプということです。$x > 0$ と $x < 5$ の共通部分ですから $0 < x < 5$ となります。これは次のように設定したのと同じです。

- assume(0 < x < 5); >>]0, 5[of Type::Real

今度は和集合を見て見ます。

- assume(x > 0) : assume(x < 5, _and); >>]0, 5[of Type::Real

注10) 例えば，$x \geqq 0$ は x>=0 とします。x=>0 と順序を逆にするとエラーになります。(コンピュータは頭が固い？) また他にもいろいろタイプがあります。

注11) abs(x) で x の絶対値という意味ですね (前節参照)

注12) 普通は delete(x) なしで，そのまま入力しても大丈夫ですが，まれにうまくいかないときがあります

に続けて次のように打ち込みます。

- `assume(x > 10, _or);` >> > 10 or]0, 5[of Type::Real

これは $x > 10$ または $0 < x < 5$ という事です。

- `assume(x < 20, _and);` >>]0, 5[of Type :: Real or]10, 20[of Type :: Real

これは $0 < x < 5$ または $10 < x < 20$ という事です。

次の例として，x, y 実数のとき，$\frac{1}{x+iy} = \frac{x-iy}{(x+iy)(x-iy)} = \frac{x-iy}{x^2+y^2}$ を考えて見ましょう。この場合は，「x,y が実数」という設定なので，先のようにやるのはちょっと面倒です。このような時は，タイプ変数 (Type::*name*) を使ってタイプを指定します。[注13)]

- `assume(x, Type :: Real) : assume(y, Type :: Real);` >> Type :: Real
- `1/(x + y * I);` >> $\frac{1}{x + \mathrm{I}y}$
- `rectform(%);` >> $\frac{x}{x^2+y^2} + \left(-\frac{y}{x^2+y^2}\right)\mathrm{I}$

z を $x + iy(x, y$ 実数) の形にするのは rectform(z) を使います。(複素数の計算の節参照) この他にもいろいろなタイプ変数があり，和集合, 共通集合の作り方も同じです。

[注13)] assume(x>=0): assume(x<0,_or); とすれば良いはずですが..