mao 言語仕様

佐藤晴郎

2003.2.17

1 mao とは

mao とはこの修士論文の筆者である佐藤晴郎が論文の題材として設計したプログラミング言語のことであり、またその設計に基づき製作した精度保証付き数値計算が可能な処理系 (ソフトウェア) のことでもある。

mao は制御フローをもつインタプリタ型言語であり、その文法、言語仕様は MATLAB や Octave を参考に作られている。MATLAB、Octave との相違点は機械区間演算を完全にサポートしたところにある。

本章では mao の言語としての仕様を記述する。

2 mao の文

mao の文は行単位で書く。それを list と記し、その中には以下のようなものが存在する。

```
list :=  (nothing)
 | \n
 | list \n
 | list expr \n
 | list asgn \n
 | list stmt \n
 | list defn \n
 | list error \n
```

- $\backslash n$ *:* 改行コード、リターン。
- *expr:* 数字、変数、四則演算の数式　等
- *asgn:* 代入文
- *stmt:* ループ文、自作関数の呼び出し、プリント文等
- *defn:* 自作関数の定義

- *error:* エラー文

3 mao のデータ型

mao のデータ型には次のようなものがある。

- スカラー区間
- 区間行列
- 文字列

4 スカラー区間

mao では全ての (スカラーの) 値を機械区間として扱う。機械区間は二つの倍精度浮動小数点数 (以後 double 型と書く場合もある) をもち、二つの倍精度浮動小数点数で機械区間の下限、上限を表す。この機械区間には必ず真の値が含まれている。また、実数世界の区間も値になりうる。その場合には区間を含む機械区間を値として扱う。

5 定数

mao では機械区間の上限、下限。また点区間の入力に定数を用いる。mao の定数は次のようなものである。

1234

567.89

現段階の mao では整数部、小数部がそれぞれ 15 桁まで入力できる。

6 区間行列

区間行列とは要素にスカラー区間をもつ型である。区間ベクトルという型は用意されておらず、行列で代用することとなる。

7 識別子

変数名とユーザー定義関数名は先頭に英字 1 文字とそれに続く英字または数字、アンダーバー “_”での列で構成される。

しかし if 等の予約語、組み込み関数名、組み込み定数名とは同じに出来ない。

8 真偽

mao は制御フローをもっている。具体的に言えば、他のプログラミング言語と同じように if 文や while 文を用いることが出来る。その制御を司るものとして、普通、関係演算子を用いる。関係演算子とは二つの値がどのような関係にあるかを示すもので、関係演算子の結果は真または偽のどちらか一方に評価される。mao の言語仕様では偽を $[0,0]$ を含む区間とし、真を $[0,0]$ を含まない区間としている[1]。

9 その他の命令文

print 文で任意の値もしくは文字列を出力できる。

mao ではこのように特別な命令文が幾つか装備されている。(詳しくは後述するが) 以下にそれを記す。

load, 設定変更に関連する命令, lu, norm, whos, free, plot

10 関数と手続き

mao ではユーザーが関数や手続きを作ることが出来る。関数と手続きの違いは関数が値を返し、手続きは値を返さないことである。関数または手続きでは引数を使うことが可能で引数はその関数および手続き内でのみ参照することができる。関数と手続きは再帰的にもちいてもよい。

11 コメント

mao ではコメントを記すことができる。コメントは “#” および “%” から行の最後までである。

[1] mao では論理をあらわすデータ型を用意していない。この仕様は C 言語のそれを参考に作った。点区間 $[0,0]$ とせず、$[0,0]$ を含む区間としたのは計算上で区間幅は拡大されるものであり、点区間が現れることは希少であると考えたからである。

12　文字列

mao では文字列を扱うことが出来る。文字列は二重引用符 “"”で囲んだものである。

13　expr (式)

ここからは第 2 節で記述したものの詳細を追っていくこととする。まず expr とは式のことであり次のようなものである。

```
expr::= suu
| [ expr @ expr ]
| VAR
| asgn
| BLTIN ( expr )
| 基本的な演算子を用いた演算
| 行列の演算
| 関係演算子、論理演算子を用いた演算
| matrix
| 行列の要素の摘出
| 特別な行列の生成
| suu ++
| suu --
| ARG
| FUNCTION begin ( arglist )
| READ ( VAR )
| expr \ expr
| 組み込み定数
| 絶対値
| Max, Min
| ノルム
| ユーザー定義関数の呼び出し
```

であり、各要素の詳細は

- *suu :* 定数をあらわす。実際は点区間である
- *VAR :* 変数のこと
- *asgn :* 代入文のこと
- *BLTIN :* 組み込み関数
- **基本的な演算子を用いた演算** *:* 第 18 節参照
- **行列の演算** *:* 第 26 節参照
- **関係演算子、論理演算子を用いた演算***:* 第 19 節、第 20 節参照
- *matrix :* 行列
- **行列の要素の摘出** *:* 第 27.4 節参照

- **特別な行列の生成** *:* 第 28 節参照
- *suu* ++ *, suu* −− *:* VAR++,VAR−− に似せて作った。suu に $[1, 1]$ を足す、または引く。
- *ARG :* $ ではじまる関数の引数
- *FUNCTION :* ユーザー定義関数
- *READ :* READ という関数
- *expr* \ *expr :* 連立一次方程式を解く際に利用。
- **ユーザー定義関数の呼び出し** *:* ユーザー定義関数の呼び出し
- **組み込み定数** *:* 組み込み定数のこと
- **絶対値** *:* 絶対値の演算のこと
- *VAR :* max, min の演算のこと
- *VAR :* ノルムの演算のこと

である。

14 asgn (代入)

asgn とは代入文のことであるが、一方で expr (式) でもある。asgn とは以下のようなものである。

```
asgn::=  変数への代入
| 行列の要素への代入
| 引数への代入
| LU 分解の結果の代入
```

である。

- **変数への代入文** *:* 第 23 節 第 25 節参照
- **行列の要素への代入文** *:* 第 27 節参照
- **引数への代入文** *:* $ で始まる引数への代入文。第 31 節
- *LU* **分解に関するもの** *:* LU 分解に関するもの。第 34 節参照

15 stmt (文)

stmt とは文のことであり、以下のようなものである。

```
stmt: expr
| RETURN 文
| PRINT 文
| ユーザー定義手続きの呼び出し
```

| 制御コード
| WHOS 文
| FREE 文
| シェルへ命令を引渡す命令文
| 既存ファイルの読み込み命令

である。各要素の詳細は以下に示すとおりである。

- *return :* 自作関数と手続きの区別
- *print* **文 :** 出力命令
- **ユーザー定義手続きの呼び出し :** ユーザー定義手続きの呼び出し
- **制御コード :** if、while、for
- *WHOS* **文 :** 使用している変数を知るための命令
- *FREE* **文 :** 変数、およびメモリの解放命令
- **シェル[2]へ命令を引き渡す命令文 :** C 言語の system 関数を呼び出す[3]
- **既存ファイルの読み込み命令 :** 既存ファイルからの入力に切り替える

16 defn (定義)

defn とは定義のことであり次のようなものである

defn: 自作関数の定義
| 手続きの定義

である。

17 エラー文

mao では実行中に何らかのエラーが存在した時、その旨を画面に出力している。

18 基本的な演算子

mao の基本的な演算子の使用法を記す。

$$expr1, expr2 \in expr \quad (expr1, expr2 \text{ はスカラー})$$

とする。下は二つの expr の和を表す。

[2]UNIX で使用される用語。コマンド・インタプリタのこと
[3]例えば多くのシステムで ls (ファイルの検索命令) や pwd (現在の位置の表示) 等の命令が使えるであろう。しかしこれらの命令は mao を起動するシステムによって使用不可能になる場合もある。

```
expr1 + expr2
```

+ の部分は他にも − (差) , ∗ (積) , / (商) , ^ (べき乗) を用いることが出来る。(ただし商の場合は expr2 に 0 を含まない事が必要)

```
- expr1
```

上は単項マイナスの入力である。

その他、任意の深さの括弧を用いることが出来る。括弧は丸括弧 “(”, “)”で入力する。

19 関係演算子

19.1 関係演算子とは

関係演算子とは二つの値がどのような関係にあるかを示すもので、関係演算子の結果は真または偽のどちらか一方に評価され、主に for、if、while 等で利用される。ところで mao の仕様では偽を $[0,0]$ を含む区間とし、真を $[0,0]$ を含まない区間としている。このことから関係演算子の結果はその関係が真ならば $[1,1]$、関係が偽ならば $[0,0]$ を返している。

19.2 不等式

$a, b \in \mathrm{IR}$ の時

$$a > b$$

は $\forall x \in a, \forall y \in b$ に対して以下の式が常に成り立つということと定義する。これは実は次と同値である。

a, b を $a = [a_1, a_2], \quad b = [b_1, b_2]$ と書くと、$a > b$ が成り立つということは $a_1 > b_2$ が常に成り立っているということである。

同様に $<, \geq, \leq$ も定義する。mao では $>$ の演算を

```
expr1 > expr2
```

と入力し、その関係が正しい時には 1 ($[1,1]$)、正しくない時には 0 ($[0,0]$) を結果として返す。

$>$ の部分は他にも $<$ ($<$), $>=$ ($\geq$), $<=$ ($\leq$) を用いることが出来る。

上の不等式の定義に関して注意しなければならないことがある。それは

$$\geq \quad \neq \quad (> \text{ または } =)$$
$$\leq \quad \neq \quad (< \text{ または } =)$$

(=は第 19.4 節で定義しているもの)

ということである。

19.3 包含関係

集合の包含関係と同じ意味で機械区間演算の包含関係を考える。

expr2 が expr1 に完全に含まれているかを判定するのに mao では

```
expr1 >> expr2
```

と入力し、その関係が正しい時には 1 ([1, 1])、正しくない時には 0 ([0, 0]) を結果として返す。

その逆で expr1 が expr2 に完全に含まれているかを判定するのに mao では

```
expr1 << expr2
```

と入力し、その関係が正しい時には 1 ([1, 1])、正しくない時には 0 ([0, 0]) を結果として返す。

$$expr1 \supset (or \supseteq)\ expr2$$

を判定するのに mao では

```
expr1 >>= expr2
```

と入力し、その関係が正しい時には 1 ([1, 1])、正しくない時には 0 ([0, 0]) を結果として返す。>>= の部分は他にも <<= ($\subset$ *or* $\subseteq$) を用いることが出来る。

19.4 等号

$a = [a_1, a_2],\ b = [b_1, b_2] \in \mathrm{IR}$ の時

$$a = b$$

は以下の式が成り立つということである。

$$\begin{aligned} a_1 &= b_1 \\ a_2 &= b_2 \end{aligned}$$

mao ではこれを

```
expr1 == expr2
```

と入力し、その関係が正しい時には 1 ([1, 1])、正しくない時には 0 ([0, 0]) を結果として返す。

また等号が成り立たない場合。つまり $a \neq b$ の場合を mao では

```
expr1 != expr2
```

と入力し、その関係が正しい時には 1 [1, 1]、正しくない時には 0 [0, 0] を結果として返す。

20　論理演算子

mao では論理演算子の and (論理積) , or (論理和) , not (否定) を用いることが出来る。and を例に取って説明する。

$$a\ and\ b$$

は a が真であり且つ b が真の場合に限り真を返す。これを mao 上の真偽にあてはめれば a に 0 ($[0,0]$) を含まない、且つ b に 0 ($[0,0]$) を含まない時、0 ($[0,0]$) を含まない区間を返すの意を表す。実際の mao での入力は

```
expr1 && expr2
```

と入力し、その関係が正しい時には 1 ($[1,1]$)、正しくない時には 0 ($[0,0]$) を結果として返す。&&の変わりに|| (or) を用いることができる。

```
! expr1
```

mao で上の入力は $expr1 = [0,0] = 0$ であるかどうかの判定をする。その関係が正しい時には 1 ($[1,1]$)、正しくない時には 0 ($[0,0]$) を結果として返す。

参考までに以下に論理演算の真偽表を記しておく。この表では 1 は真、0 で偽、zero で $[0,0]$ を含む区間、not zero で $[0,0]$ を含まない区間を表す。

a	b	a && b	a || b	!a
zero	zero	0	0	1
zero	not zero	0	1	1
not zero	not zero	1	1	0
not zero	zero	0	1	0

21　組み込み関数

後述するようにユーザが関数を定義することも可能なのだが、はじめから、定義されている関数がある。$x \in expr$　(x はスカラー) とする。

$$\sin(x),\ \cos(x),\ \tan(x),\ \cot(x),$$

$$\sinh(x),\ \cosh(x),\ \tanh(x),\ \coth(x),$$

$$\arcsin(x),\ \arccos(x),\ \arctan(x),\ \text{arccot(x)},$$

$$e^x,\ \log_e(x),\ \log_{10}(x),\ |x|,$$

$$x^2,\ \sqrt{x}$$

22　組み込み定数

mao では下の表に記す組み込み定数を用意している。もちろん、値は区間である。

組み込み定数名	意味		mao での値
DEG	$180/\pi$	1 ラジアンあたりの度数	$[57.29577951308232, 57.29577951308233]$
E	e	自然対数の底	$[2.718281828459045, 2.718281828459046]$
GAMMA	γ	Euler-Mascheroni の定数	$[0.577215664901532, 0.577215664901533]$
PHI	$(\sqrt{5}+1)/2$	黄金分割比	$[1.618033988749894, 1.618033988749895]$
PI	π	円周率	$[3.141592653589793, 3.141592653589794]$

これらは第 23 節で説明する変数の様に expr として扱うことが出来る。しかしユーザーが新たに値を代入することは禁止している。

23　変数

mao では変数をグローバル変数として扱う。変数 (VAR) に値を代入したい時は次のようにする。

```
VAR = expr
```

=の部分は他にも`+=,-=,*=,/=` (意味は C 言語のそれと同じ) を用いることが出来る。上の式は代入文 (asgn) に分別され、結果は何も表示されない。また後置インクリメント、デクリメントも使うことが出来る。[4] 入力法は次のとおりである。

```
VAR ++(or --)
```

また変数 (VAR) は expr の一種である。上節までで扱ってきたような expr を含んだ式のなかに変数 (VAR) を入力すると VAR は代入されている値に変換される (もしその際に VAR に何も値が代入されていなければ、mao は error を出力する) 。

24　行列

行列は次のように入力する。

$$a, b, c, d, e, f, g, h, i \in expr \quad (\text{ただし } a, \cdots, i \text{ はスカラー})$$

```
[ a, b, c ; d, e, f ; g, h, i]
```

[4] C 言語と異なり値は先にインクリメント (またはデクリメント) される。

上の入力で

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

としていることと同じである。

つまり、行列の始まりと終わりを'[' と']' でくくる。列は、',' でくぎり、行は ';' で区切る。

mao ではベクトルを一行もしくは一列の行列であると考え、行列と同様な処理を行っている。

25　行列を変数へ代入

変数への入力は次のようにする。

```
A = [ a, b, c ; d, e, f ; g, h, i]
```

上の入力で

$$A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

としていることとなる。

26　行列の演算

行列の演算（足し算、引き算、掛け算、単項マイナス）がサポートされており A, B を行列とするとそれぞれ

$$A + B,\ A - B,\ A * B,\ -A$$

と入力する。

また、a をスカラーとすると

$$a * A,\ A * a$$

との入力で A の各要素に a を掛けた結果を返す。

転置行列は次の様にする

$$A'$$

27　行列の要素単位での操作

27.1　行列の要素の書き換え（上書き）

行列の要素の書き換え（上書き）を

> 変数　[書き換えのスタート地点の行番号 , 書き換えのスタート地点の列番号] = 式

とする。

例えば

$$A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \tag{1}$$

$$B = j \tag{2}$$

$$C = \begin{pmatrix} k & l & m \end{pmatrix} \tag{3}$$

(英小文字は $expr$ でスカラー)

の時

```
A[1,2]=B
```

とすると

$$A = \begin{pmatrix} a & j & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

となり、

```
A[1,1]=C
```

とすると

$$A = \begin{pmatrix} k & l & m \\ d & e & f \\ g & h & i \end{pmatrix}$$

となる。

27.2　行列の要素の書き換え（和）

行列の要素の書き換え（和）を

変数 [書き換えのスタート地点の行番号 , 書き換えのスタート地点の列番号] + = 式

例えば A, B, C が (1)、(2)、(3) の場合

```
A[1,1]+=B
```

とすると

$$A = \begin{pmatrix} a+j & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

となり

```
A[1,1]+=C
```

とすると

$$A = \begin{pmatrix} a+k & b+l & c+m \\ d & e & f \\ g & h & i \end{pmatrix}$$

となる。

27.3　行列の要素の書き換え（差）

行列の要素の書き換え（差）を

変数 [書き換えのスタート地点の行番号 , 書き換えのスタート地点の列番号] − = 式

例えば A, B, C が (1)、(2)、(3) の場合

```
A[1,1]-=B
```

とすると

$$A = \begin{pmatrix} a-j & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

となり

```
A[1,1]-=C
```

とすると

$$A = \begin{pmatrix} a-k & b-l & c-m \\ d & e & f \\ g & h & i \end{pmatrix}$$

となる。

27.4 行列の要素を摘出

変数 [取り出しのスタート地点の行番号：取り出し終わりの行番号, 取り出しのスタート地点の列番号：取り出し終わりの列番号]

上の操作で、行列の要素の摘出または、もとの行列よりも小さな行列の摘出を行うことが出来る。

例えば A が (1) の場合

mao での入力を

```
A[2:3,2:3]
```

とすると、出力は

$$\begin{pmatrix} e & f \\ h & i \end{pmatrix}$$

となる

また一つの要素だけを取り出すことも出来る。その場合は一行一列の行列ではなく、一つの区間 (スカラー) として取り出される。

```
A[2:2,2:2]
```

とすると出力は e となる。摘出の式は expr に分類されることから次のように変数に代入することも出来る

```
D = A[2:2,2:2]
```

とすると $D = e$ となる。

2:2 と書いてあるところは、':' の左と右が同じ値の場合の限り、一つでかくことができる。つまり、次のように省略することも出来る。

```
A[2,2]
```

A から 2 行目全体を取りたいこともあるだろう。その場合は

```
A[2,:]
```

とすればよい。
':' の右と左に何も入力しなければ一行目から A の最終行までということになる。
列に関しても同様のことが出来る。

28 特別な行列を作る

対角成分が 1 の行列、ゼロ行列、行列の要素が全て 1 の行列、を作ることが出来る。

```
eye(m,n)
zeros(m,n)
ones(m,n)
```

上はそれぞれ、その性質の m 行 n 列の行列を作る。
引数は一つにすることも可能である。その場合、本来、二つ必要である引数に同じ値を入れていることと同値となる。下はその例である。

```
eye(m)
```

m 行 m 列の単位行列を作る。

29 print

出力は print 文で作られる。print 文の引数は、C と同じように二重引用符でくくった文字列、あるいは式をコンマで区切った並びを使う。また改行などを表す記号としてバックスラッシュと英小文字を組み合わせたエスケープシーケンス (拡張表記) を用いることが出来る。

30 制御フロー

制御フローとして if、if-else、while、for を用いることが出来る。しかし C 言語と異なり break 文、continue 文はない。mao ではセミコロンが特別な意味をもたない。文と文は改行で区切られる。具体的な mao での入出力例は第**??**節 mao manual で記述する。

31　関数と手続き

mao ではユーザーが関数や手続きを作ることが出来る。関数と手続きの違いは関数が値を返し、手続きは値を返さない。上がみたされない場合はエラーとなる。関数と手続きには起動するときにコンマで区切った引数を書き加えても良い。引数は関数または手続き内で参照することができ、$2 と書けば 2 番目の引数という意となる。関数と手続きは再帰的にもちいてもよい。具体的な mao での入出力例は第**??**節 mao manual で記す。

32　ファイルの読み込み

mao ではあらかじめ作っておいたファイルからのプログラムの読み込み (入力) をすることが出来る。

入力方法は

```
load (" ファイル名 ")
or
load " ファイル名 "
```

とする。

33　変数、関数、手続きへの操作

whos という命令で自分の設定した変数、関数、手続きの名前の確認をすることが出来る。設定した変数は上書き可能だが、ユーザー定義関数、ユーザー定義手続きは上書きすることができない。このことから変数、関数、手続きを削除することを可能にした。実行方法は

```
free（変数名、ユーザー定義関数名、ユーザー定義手続き名）
```

また全ての変数、関数、手続きを削除するには

```
free
```

とすればよい。上を行うと、mao では記憶している変数、関数、手続きのメモリを削除し、さらにそれ以外にも不要となったメモリを解放する。よって、これはメモリを節約するためにも効果的な方法である。

34　LU 分解

mao では LU 分解に対する命令を持っている。

正方行列 A の LU 分解とは

$$A = LU$$

ただし L は下三角行列、U は上三角行列 と分解することである。

次に部分ピボット交換ありの LU 分解とは正方行列 A を以下の様にすることである。

$$PA = LU$$

P は置換行列、L は下三角行列、U は上三角行列である。

35 その他の命令

逆行列を導く命令、連立一次方程式を解く命令、最大値ノルムを計算する命令、max, min を計算する命令絶対値を計算する命令を mao は持つ。

36 設定の変更

mao では幾つかの設定を変更することが出来る。詳しくは第**??**節 mao manual に記述しておく。

37 シェルモード

シェルでの命令 (コマンド) を起動することが出来る。

使えるコマンドは用いているシステムによって異なるが C 言語における system 関数で呼び出せるものとする。

38 plot モード

gnuplot 等で応用できるファイルを作り出す。