数理解析研究所講究録
第1951巻 2015年 223-229

線形代数用演算 TEX マクロの作成と利用

大阪府立大学　吉冨　賢太郎 (Kentaro Yoshitomi)
Faculty of Liberal Arts and Sciences,
Osaka Prefecture University

1　動機と経緯・概要

所属の大阪府立大学において, 微積分学や線形代数の授業を担当する過程で, さまざまな演習や小テストの問題を作成しているうちに TEX 内で乱数の利用 (生成・保存) やさらに行列の計算処理ができれば便利であろうと考えるようになった. 通常, 計算問題の作成には, Mathematica や Maxima, Maple などの数式処理ソフトを用いて, 行列を設定し, その計算結果や過程を TeXForm (Mathematica の場合) で TEX 用に整形出力してはりつける, という手順を踏む. あるいは, 最近では KETpic[1] のように scilab[2] 等の CAS を呼び出して出力した tex ファイルを`\input` で読み込んで処理するという方法もある. しかし, 微妙な調整をしたり, 適切な乱数が生成されるまで何度か試行錯誤を試みるという手順や問題コンテンツデータの可搬性[*1], 問題データの保守性[*2], 編集操作の一貫性[*3]を総合して考えると, すべてを TEX 内で処理するのが最適であると考えたのである.

TEX 内部での計算というと, 未だ違和感を持つ TEX ユーザも少なからずいるかもしれない. しかし, 実際は整数演算は 31 ビット[*4]まで演算を内部変数である counter を用いて行うことができ, また, そのようなコマンドを自作することも容易である. 仮に `\ZAdd` を整数の加法マクロ[*5] とすると,

```
\def\a{1}\def\b{2}\ZAdd\a\b\c
$\a + \b = \c$
```

のような表記が可能である. さらには, 必要に応じて Shell Escape 機能[*6]を使うことによって, Python, Ruby, Perl といった処理言語や Pari/GP や maxima などの数式処理システムを呼び出して使うことも可能である[*7].

[*1] 当時からeラーニングとの問題コンテンツデータの供用の着想がすでにあり, 共通化して使用するには汎用性を持ち, かつ自身がサンプル PDF を生成する TEX ファイルを作っておくべきという発想があった (後述).

[*2] 複数ファイルから構成されるとファイルをコピーして利用する際に, プログラムなどの副ファイル群もコピーする必要が生じる.

[*3] 平たく言えば Emacs の中で C-c C-c だけで全てが済むこと.

[*4] 処理系に寄るかもしれない

[*5] 例えば: `\def\ZAdd#1#2#3{\@tempcnta=#1\advance\@tempcnta #2\xdef#3{\number\@tempcnta}}` `\@tempcnta` は TEX の持つデフォルトのカウンタ. この実装は自作後 emath の `\IAdd` と比較するとほぼ同一であり, 自然な定義であると考えられる.

[*6] ある時期の (おそらく pTEX の) 実装では, オプション '-shell-escape' を default では禁止していたこともあるので注意が必要

[*7] PariGP の場合例えば
`\def\CallPariGP#1#2{%`

したがって, 浮動小数演算も行うことができる. 実際, そのような機能を実現したものとして emath[4] がある. emath は計算以外にもグラフ描画など中高等学校で用いられるような図形を多用する教材を容易に作成できる[*8] マクロ集である. 乱数生成機能を含むさまざまな計算機能の他, 列挙機能の拡張など計算機能以外にも有用なマクロが総合的に含まれた巨大なマクロ集[*9]である. 線形代数プリント作成をもっと楽にできないかと考えていた頃, この emath を知り, 計算機能と乱数生成機能に着目した[*10] のである.

2 マクロの概要と利用法

行列の演算マクロを実装するにあたって工夫を要した点は行列の表現方法であった. TEX のマクロ名には数字を用いることができないので, \Mat12 のようなマクロ名は使えず, この方法で行列を表現することはできない[*11]

そこで, '1', '2' , ... の代わりに 'a', 'b' を用いて\MATaa, \MATab で (1,1), (1,2) 成分を表すという方法を考案した. マクロ名の生成は例えば\@I が 'b', \@J が 'c' であれば, \csname MAT{\@I}{\@J}\endcsname によって作成でき, マクロ \MATbc を参照することができる. これにより, \MATxx の形のマクロの集合を仮想的に行列の表現としたのである[*12]. 図 1 に行列の積の演算マクロを実際に示す.

\Ifor は emath が提供するループ機能である[*13]. これを用いて行列の積 $AB = C$ を計算している. \num@to@abc が数値 '0','1',... を'a','b',... に変換している.

また, 乱数機能[*15]については図 2 のように emath の乱数生成機能を利用し, さらに, 乱数のシードを保存して再現できる機構を加えたマクロ\RannsuuInit を定義して利用している. マクロ\RansuuInit{200}はシードを保存したファイル \jobname.ran[*16]があればそれを読み込み, 同じ乱数パターンを生成できるようにする. 存在しなければ引数で指定する長さの乱数シードを生成し, 同ファイルに保存する. 引数が負数であれば, 保存・読み込みは行わず初期化のみ行う. また, 引数が 0 ならば乱数は使用しない.

乱数保存機能により, 例えば, 問題のみのモードと解答モードを\if 文で場合分け[*17]して

```
\begingroup\makeatletter\endlinechar=\m@ne\everyeof{\noexpand}
\edef\@@x{\endgroup\def\noexpand#2{\@@input|"echo '#1'|gp -q" }
}\@@x
```

このような定義は Stackoverflow 等に記載がある (例えば [3]) なお, 一部の処理系ではバナーなどが消せない (Silent モードがない) 場合があるので, Shell スクリプトなどで stderr にリダイレクトするなど wrapper を用意する必要があるかもしれない (gp -q の '-q' はサイレントモード).

*8 KETpic もこの点は同じであり, ketcindy という Cinderella2 と連携する機能も開発が始まっている.

*9 ファイル数は 60 を越える

*10 実際には整数成分の演算のみであったので, emath を使わずとも実装は可能であったが開発開始当初 (2010 年頃) はまだそのような見識がなく, また, 乱数の生成も必須であったため emath の計算機能・乱数生成機能を利用する形で行列の演算マクロ開発を始めた.

*11 \Mat というマクロを引数が#1=1,#2=2 のときに (1, 2) 成分を返すという仕様にうまく工夫すればできることが最近わかり, 開発中の改良版 (後述) においてはその方式を採用している.

*12 この方法には行・列の数が 26 までしか扱えないという欠点がある. が, 教育利用上では制限にはならない.

*13 LATEX にもトークン列や CSV[*14]形式のトークン列をループして利用できる tfor や for が提供されている. C 言語に代表される for 文のような数値の増大・減少列によるループはカウンタによる \@whilenum 文を使えばよいが, カウンターの使用宣言など準備が若干必要である.

*15 ここで言う乱数は暗号論的乱数ではなく, シードが同じであれば同じ乱数列を生成する疑似乱数である

*16 \jobname はファイル名が abc.tex であれば abc に展開される.

*17 筆者の場合は作成中は\newif で生成した if 文を用いて分岐し, 本番用は\jobname.amd というファイルがあれば解答モード, なければ非解答モードのようにしてスクリプトで 2 回コンパイルしている

```
  \def\MatMul#1#2#3#4#5#6{%#1(size#4x#5)x#2(size#5x#6)=#3
    \Ifor\@I{0}{#4}\Do{\num@to@abc{\@I}{@@I}%
      \Ifor\@J{0}{#5}\Do{\num@to@abc{\@J}{@@J}%
        \edef\tmp@copy{\csname #1\@@I\@@J\endcsname}%
        \expandafter\edef\csname tmp@A@\@@I\@@J\endcsname{\tmp@copy}%
      }}%
    \Ifor\@I{0}{#5}\Do{\num@to@abc{\@I}{@@I}%
      \Ifor\@J{0}{#6}\Do{\num@to@abc{\@J}{@@J}%
        \edef\tmp@copy{\csname #2\@@I\@@J\endcsname}%
        \expandafter\edef\csname tmp@B@\@@I\@@J\endcsname{\tmp@copy}%
      }}%
    \Ifor\@I{0}{#4}\Do{\num@to@abc{\@I}{@@I}%
      \Ifor\@J{0}{#6}\Do{\num@to@abc{\@J}{@@J}%
        \def\@@tmp{0}%
        \Ifor\@K{0}{#5}\Do{\num@to@abc{\@K}{@@K}%
          \edef\tmp@a{\csname tmp@A@\@@I\@@K\endcsname}%
          \edef\tmp@b{\csname tmp@B@\@@K\@@J\endcsname}%
          \IMul\tmp@a\tmp@b\tmp@c\IAdd\@@tmp\tmp@c\@@tmp%
        }\expandafter\xdef\csname#3\@@I\@@J\endcsname{\@@tmp}%
      }}%
}
```

図 1　行列の積を計算するマクロ

```
\def\RansuuInit#1{
\ifnum#1>0
\IfFileExists{\jobname.ran}{\input{\jobname.ran}
\setransuuretu{\rtmp}}{
\ransuuretu{#1}\r@tmp%
\write18{/bin/echo "\gdef\ rtmp{\r@tmp}"|sed 's/ //g' > \jobname.ran}}
\else\ifnum#1<0 \IMul{#1}{-1}\ransuu@init@tmp
\ransuuretu{\ransuu@init@tmp}\rtmp\fi\fi}% #1=0 : no ransuu
\RansuuInit{200}
\Ransuu[d]{Int(X*3)+1}\Tmp@A
\Ransuu[d]{2*Int(X*5)+1}\Tmp@B
\Ransuu[d]{Int(X*5)-2}\Tmp@C
```

図 2　乱数初期化コマンドの拡張 (乱数シードの設定と保存)

問題を授業で配布し解答を授業や授業支援システム (LMS) 上に掲載するというような運用が可能である. 図 3 に行列生成・演算・表示を用いたサンプルを示す (数値パラメータの設定部分は省略).

これらの機能を提供するマクロ emLinAlg.sty は [5] の URL からダウンロードできる. 自由に改変して利用可能である. ただし, これまで述べたように emath が使える状態にしておくことが必要である[*18]. また, 利用方法については同じく [5] からダウンロードできるサ

[*18] emath のすべてのスタイルファイルが必要なわけではないが, 依存関係でどこまで読みこんでいるか明確に把握していないので利用者の判断でファイルを選択して欲しい. 問題なければ /usr/local/share/texmf-local/tex/latex/emath/ や~/texmf/tex/latex/emath などに配置して mktexlsr しておけば問題なく使えるだろう. ただし, emath.pl などの perl ライブラリも perl -V で表示される @INC のいずれかの Path に配置する必要がある.

```
\RansuuInit{200}
\makeatletter
\Ransuu[d]{Int(X*3)+1}\Tmp@Aa% one of 1,2,3,4
        (中略)
\SetMatrix{MPA}4{% 4x4 上半三角
  1{\Tmp@Aa}{\Tmp@Ab}{\Tmp@Ac}01{\Tmp@Ad}{\Tmp@Ae}001{\Tmp@Af}0001}
\SetMatrix{MPB}4{% 4x4 下半三角
  1000{\Tmp@Ba}100{\Tmp@Bb}{\Tmp@Bc}10{\Tmp@Bd}{\Tmp@Be}{\Tmp@Bf}1}
\MatMul{MPA}{MPB}{MP}444% 行列式 1 の行列
\makeatother
行列 $A = \DspMatrix{MP}44$ の逆行列を求めよ.
```

図 3　マクロ利用例 (逆行列の問題)

ンプルソースを参照されたい[*19].

3　今後の課題と予定

TEX 内部で計算することのメリットはもちろんコンパイルするだけで問題パターンを新規生成できるという点にある. 追加課題はファイルをコピーして別ファイルにするか, 乱数保存ファイルを削除かリネームすれば違う数値パターンで再作成できる.

一方, 元々 TEX ファイルでこのような計算問題コンテンツを作成した動機のもう一つとして, 可搬性の高い問題コンテンツのデータベースの作成が念頭にあった. 大阪府立大学で運用している e-Learning システムの webMathematica ベースの MathOnWeb や最近では Moodle[*20]のプラグインとして動作する Maxima ベースの STACK など同様の問題コンテンツを必要とするシステムは多い.

そのようなシステムでの利用も考えたとき, それらのコンテンツの共通仕様メタデータがあれば, そこから, TEX や各システムへのコンテンツのデータ変換が可能となる. そのコンテンツデータの雛形の位置付けとして TEX ソース単独でコンテンツのサンプル生成機能を持たせることを意図していた[*21].

emath は冒頭にも述べたように巨大なマクロ集であるが, CTAN[6][*22]には収録されておらず別途 [4] からダウンロードする必要がある. また, ファイル数が多いため, 作業ディレクトリに毎回展開しておくことは合理的ではないので, 脚注*18 に述べたように /usr/local/share/texmf-local/ のようなグローバルな PATH に展開して別途設置する必要がある[*23]. また, 本稿で紹介しているマクロは乱数生成機能さえ別途実装すれば emath から独立させることも可能である. そこで, 現在, emath があれば利用し, また, pgf[*24] があれば pgf の計算機能を用い, どちらもなければ自前で整数演算と乱数機能を持

[*19] 開発中の後述の **iMath.sty** と **LinearAlg.sty** についても同サイトからダウンロード可能であるが, 近日中に github に移行予定であり, その場合の移行先も上記 URL にリンクを設置する.

[*20] LMS の一つ. 講義支援システムとして採用される大学・高等教育機関が増えている.

[*21] 開発当初はここまで明確に意図していたわけではないが, 最近, コンテンツの仕様案を提案しており, この発想は TEX データと合わせて一貫性を持つようになってきたと考えている. また, この仕様案の中で例えば乱数生成を指定する変数部分を TEX マクロとして実装するにはやはり, 本稿のような機能が必要である.

[*22] Comprehensive TeX Archive Network. TeX の標準配布に通常含まれ, さまざまなマクロが含まれる.

[*23] 一旦やってしまえば澄むが, 実際この作業をすべての TEX ユーザに強いるのはしばしば困難を伴う (った).

[*24] beamer.sty は数学者におなじみのスタイルファイルであるが pgf をグラフィックエンジンに使用している.

ち, かつ, 次世代の TEX(LATEX3 や LuaTEX) でも利用可能であることを目指した独立性と共存性, 将来性を高めた形での改良版のパッケージを作成中で, それが iMath.sty と LinearAlg.sty である[*25]. これらも前述のように開発中のものは現在 [5] から取得できる. この改良版における行列の新しい表現方法については次節の付録で解説する.

今後はこの改良版を完成させるとともに, 公開が中座している [5] のコンテンツの公開を完了させ, さらに, 改良版マクロに合わせた改訂も随時進めて行く予定である[*26].

4 付録:改良版マクロについて

本節では改良版の LinearAlg.sty における行列の表現方法について解説する. この方法はオブジェクト指向的であり, TEX の\makeatletter ～ \makeatother 内で作成すれば \Mat:1,2 というようなマクロ名が生成できることを利用し, emLinAlg.sty において内在していたいくつかの問題点を解消するものである.

図 4 において, 例えば\NewMatrix35{Mat} は 3×5 行列 Mat を宣言する. マクロ \Mat:type や \Mat:typeid は将来の拡張用に成分が整数であることを表す[*27]. \Mat:0,0 はサイズ "3,5", \Mat:0,1 は列数 "3" などが定義されている. 各成分は \Mat{1}{2}[*28] のような形でアクセスでき, 宣言時は 0 で初期化されている.

このようにして宣言された '行列クラス' の 'インスタンス' Mat は行列の型を内部で持っているので, 型を指定する必要がない. したがって, 型不一致による計算エラーなど問題開発時にしばしば遭遇するバグを事前に察知することできる. また, 現行マクロで問題作成時に予期せず起きた\MatAab のようなマクロを別の目的で上書き定義してしまうというような問題が回避される. 何故ならば\Mat12 は $(1,2)$ 成分に展開されるが, $(1,2)$ 成分を変更するには, 直接マクロ \Mat:1,2 を変更しなければならず, これは通常の \makeatother な TEX の状態では参照できないからである. つまり, このようなマクロは C++ 等で言うところの Private 変数に該当し内部成分の変更は特別なマクロ \setMat12{...}を用いて明示的に行う必要があるため, 安全である.

このように現行の emLinAlg.sty パッケージで発生していた問題開発時に起こる種々の問題はすべて解消されている[*29]. さらには, 小行列の取得など 'a', 'b' を用いた数値表現では困難のあった機能も実装が容易になっている. TEX は再帰呼び出しに向かないとされるが, 理論上再帰呼び出しによる行列式の計算も可能である[*30].

開発中の iMath.sty では乱数を emath 読み込んでいるときは emath を利用し, そうでないときは線形合同法 [8][*31] を用いる方針で現在開発中である. 乱数シードを Windows 以外では /dev/uraondom から, Windows では echo %RANDOM% を利用して初期値 x_0 を取得し[*32], そこから固定した a, c, m を用いて $x_n = ax_{n-1} + c \pmod{m}$ で疑似乱数を生成

[*25] いずれも小さいファイルなので今後統合する可能性もある

[*26] 本来の予定であった e ラーニングコンテンツデータベースとの連携も進めたいと考えているが, まだ時間がかかる見込である. [5] は本来その雛形の予定であった.

[*27] 筆者の場合, 授業では整数ですべて計算できるような問題を出すという' 親切設計' であるが, 必ずしも教育的でない場合もあるので成分には '1/2' や '1/3' のようなちょっとした分数が出ることも想定すべきである.

[*28] \Math12 と書けるので, より直感的である

[*29] 行・列の数が 26 までという制限も一応撤廃されている.

[*30] 実際には Bareiss による Integer Preserving Gaussian Elimination [7] を用いる予定だが, 未実装.

[*31] 教育目的の利用上は 8 ビット前後の素数を 2～3 つかけあわせた整数で乱数を生成すれば十分であろう.

[*32] もちろん, 手動で設定することも可能にする予定である

```
\def\NewMatrix#1#2#3{% Declare #1x#2 matrix #3
  \la@i=0\la@j=0\edef\@@i{\number\la@i}\edef\@@j{\number\la@j}
  \la@k=1\la@l=1\edef\@@k{\number\la@k}\edef\@@l{\number\la@l}
  \expandafter\xdef\csname #3:type\endcsname{IntegerMatrix}% Upper class?
  \expandafter\xdef\csname #3:typeid\endcsname{11}%
  \expandafter\xdef\csname #3:\@@i,\@@j\endcsname{#1,#2}% 0,0=row,col
  \expandafter\xdef\csname #3:\@@i,\@@l\endcsname{#2}% 0,1 = col
  \expandafter\xdef\csname #3:\@@k,\@@j\endcsname{#1}% 1,0 = row
  \la@row=#1\advance\la@row\@ne\advance\la@i\@ne
  \la@col=#2\advance\la@col\@ne\advance\la@j\@ne
  \@whilenum \la@i<\la@row \do{%
    \la@j=1
    \@whilenum \la@j<\la@col \do{%
      \edef\@@i{\number\la@i}\edef\@@j{\number\la@j}
      \expandafter\xdef\csname #3:\@@i,\@@j\endcsname{0}
      \advance\la@j\@ne
    }
    \par
    \advance\la@i\@ne
  }
  \expandafter\gdef\csname #3\endcsname##1##2{%
    \csname #3:##1,##2\endcsname
  }
  \expandafter\gdef\csname get#3\endcsname##1##2{%
    \csname #3:##1,##2\endcsname
  }
  \expandafter\gdef\csname set#3\endcsname##1##2##3{
    \expandafter\xdef\csname #3:##1,##2\endcsname{##3}
  }
}
```

図 4　TEX による '行列クラス'(インスタンス) の宣言

する. この方法で, NTL により試験的に乱数を生成して検証したが, 教育利用目的には十分であると考えられる.

また, 置換群の演算など置換に関する機能も改良されており, 乱数機能と合わせれば, 問題群からランダムに複数選択する機能なども実装可能で, 一般的な問題コンテンツ作成への応用もできる.

なお, この改良版マクロにおける行列の実装で用いた方法は, 言わば多重配列や連想配列を含む配列機能を実現していると見ることもできる. また, クラスのインスタンスの疑似的実装として見れば $\mathbb{Z}$ 上の楕円曲線の射影モデルやその上の点の 'インスタンス' の実装も可能であると思われる. これが実装できれば簡単な代数の講義などでの講義資料作成などでも活用できると期待される[*33].

[*33] ただし, 本格的に研究利用で使用するには多倍精度の計算が必要であるから, 結局 Pari/GP などを利用することになるという点では研究目的での利用にはあまりメリットはないかもしれない.

参考文献

[1] http://ketpic.com/
[2] http://www.scilab.org/
[3] http://tex.stackexchange.com/questions/16790/write18-capturing-shell-script-output-as-command-variable
[4] http://emath.s40.xrea.com/
[5] http://bg(または bg2).las.osakafu-u.ac.jp/emath/
[6] http://ctan.org/
[7] http://www.ams.org/journals/mcom/1968-22-103/S0025-5718-1968-0226829-0/S0025-5718-1968-0226829-0.pdf
[8] http://en.wikipedia.org/wiki/Linear_congruential_generator