数理解析研究所講究録
990巻 1997年 72-81

自然な数学インタフェースを持つプログラミング環境

八巻直一 N.Yamaki (静岡大学)
鳥居達生 T.Torii (名古屋大学)
杉浦 洋 H.Sugiura (名古屋大学)
桜井鉄也 T.Sakurai (筑波大学)
趙 燕結 Y.Zhao (久留米工業大学)
本郷 茂 S.Hongo (青山学院大学)
内田智史 S.Uchida (神奈川大学)

1 はじめに

コンピュータの発明当初は，使用目途としていわゆる科学技術計算に限られていた．しかし，その後急速に進歩を遂げ，経済社会に広く浸透するとともに，その用途は大量データ蓄積と分類集計などのEDPが主流となった．一方で科学技術計算に対しても，飛躍的な計算速度の向上と記憶容量の拡大によって，科学の発展と産業の進歩に大きな寄与をしてきたことも事実ではある．さらに，近年はベクトルプロセッサや並列処理技術が進展しており，ますますコンピュータの潜在的計算能力は高まっている．しかし，なんといってもコンピュータの主たる用途はEDPであり，近年は通信なども含む日常社会への対応である．したがってコンピュータメーカの主たる開発努力は，自ずと科学技術計算系よりもEDP系などとならざるを得ないのが現状であろう．

翻って，科学技術計算に関わる研究者あるいは実務家にとっては，高速計算機構を最大限に活かすアルゴリズムや，プログラミング技術の研究開発が急務である．ところが，各社が開発を競っている高速計算機構は，それぞれに適するプログラミング技術が一般的には異なり，コンピュータの潜在能力を最高に発揮させるには，個々のコンピュータのノウハウを個別に研究し活用する必要がある．そのため，ソフトウェアの移植性は著しく制限される．この作業は極めて局所的であり，一般化は困難である．

科学技術計算は，結局数学的な問題を数値的に解くことであるから，問題と解法およびプログラムがいつも組みとなって扱われる．しかし，現在の環境ではこれらはまったく別の存在である．問題と解法は紙の上に存在し，プログラムは本来解法と同値な内容であるはずであるが，実際にはプログラミング言語の制約，プログラム構造あるいはプログラミ

ングテクニックによって，見た目には全く別の存在となっている．したがって，これらの統一的扱いは一重に人間の手作業に委ねられている．

コンピュータメーカ自らに科学技術計算のユーザインタフェースや，オープン化に対する開発努力を期待したいところであるが，上記のように主たる収益源ではない所以からか，そのような動きは活発には見えない．

本研究ではこのような現況を踏まえて，科学技術計算環境の飛躍的な改善を目指し，以下のような課題の解決を目標とする．なお，本稿の以降では，科学技術計算ソフトウェアを含む，数学が関与するソフトウェアを数学ソフトウェアと呼ぶことにし，数学ソフトウェアの開発や実行環境を対象とする．

- オープンな数学ソフトウェア用のユーザインタフェースを提案する．
- 自然な数学表記を基本とする．
- 問題，アルゴリズムおよびプログラムを統一的に扱う．

以下，第二章ではオープンなユーザインタフェースについて，インターネット上で公開する数学エディタを提案する．第三章では数学ソフトウェアの構造を示し，その統一的取り扱い手法を提案する．第四章では，アルゴリズム記述言語LAMAX[1]を実行形式とするプロトタイプを紹介し、初期の性能評価を行う．最後に，nMathプロジェクトの今後の方針を述べる．

2 nMathシステム

表記の目標を達成するための著者らの研究体制を，nMathプロジェクトと仮によぶことにする．また，nMathプロジェクトの成果であるいくつかのソフトウェアには，nMAthを冠した名前を付与することとする．ちなみに，本稿で実験的に作成したエディタは，nMath-pとよぶことにする．

nMathプロジェクトで開発を目指すソフトウェア全体を，nMathシステムということにすれば，nMathシステムは以下の四つの部分に大別される．

1. エディタ・サブシステム

   nMathシステムとのユーザ・インタフェースである．nMathエディタは，自然な数学表現をそのまま編集可能であり，数学文書，モデル記述，アルゴリズム記述，プログラミングおよび実行環境記述のすべてに亘って，紙の上の数学書式と同じ表現を許容する．エディタが認識する文書は，二次元的な文法を規定した編集機能を持つ．また，nMathエディタの生成する内部形式は公開を前提にしているので，Texなどの文書整形システムへの変換や，FORTRANなどのプログラム言語への変換，あるいは数式処理システムへの受け渡しが制約なしに可能である．nMath-pではTexへの変換機能と，後述のLAMAXへの受け渡し機能が用意されている．

2. 意味解釈・サブシステム

   nMathエディタが生成した内部形式を解析し，文脈に依存した数学表現の意味解釈をおこなう．解釈に際して用いられる知識ベースは，自由に情報の追加やカスタマイズが出来るように，公開を原則としている．解釈結果の形式と内容はやはり公開が前提であるので，より複雑な処理を要する外部のツールへの受け渡しには，解析結果が自由に利用可能である．

3. 計算プログラム生成・サブシステム

   エディタと場合によって意味解釈システムが生成するアルゴリズムを解析し，各種コンピュータに対する計算プログラムの最適コードをつくりだす．各種コンピュータの最適化コードを生成するためのチューニング・ノウハウは、知識ベースを参照することによって獲得されるが，知識ベースそのものも公開を前提としている．

4. 環境定義・サブシステム

   計算する対象は，数学問題の解法アルゴリズムである．数学問題，アルゴリズムおよびプログラム（データも含む場合もある）は，一体で一つの内容を持つ．したがって，nMathシステムではこれらを不可分の全体ととらえ，構造的に管理する．アルゴリズムの実行に際して，問題とデータあるいはユーザとの関係を定義する機能が，環境定義・サブシステムである．環境定義は，アルゴリズムと問題およびデータを連携づけるほか，中間的出力や印刷レイアウトなどの出力情報，特に指定したいライブラリなど木目の細かい定義をおこなうことができる．

2.1 nMathエディタと意味解釈

nMathのフロントエンド・システムは，数学文書エディタを前提としている．nMathエディタは自然な数学表記を許すインタフェースと，数式を含む文書の意味認識能力を持つ数学文書エディタであり，文脈に依存する数学表現の自動解釈機能の実現を目標としている．本稿では，第一次のプロトタイプ版の経験を経て，nMathシステムの基本機能である，アルゴリズム記述の編集とその実行プログラムの生成について，実現可能性の実証を試みた．

さらに，公開性を確保する手段として，インターネット上に存在するエディタとすることを試みた．すなわち，サーバ上のnMath-pをネットワークを介してユーザが使用する．

本稿で開発されたシステム（nMath-p）は，後述のLAMAXプログラムをその実行プログラムとして想定したので，LAMAXの表現形式の制約をある程度受けている．したがって，最終目標の完全に自然な数学表現を許容する段階には，まだ至っていない．

nMath-pが編集した文書は直ちに意味認識され，内部形式に変換される．再表示あるいはその他の処理への変換は，ファイルを介して行われる．

nMathエディタの内部形式は，nMathシステムで定義された二次元的文法（本稿では便宜上これを亀の甲羅文法とよぶ）に従う．

下図は亀の甲羅文法である．

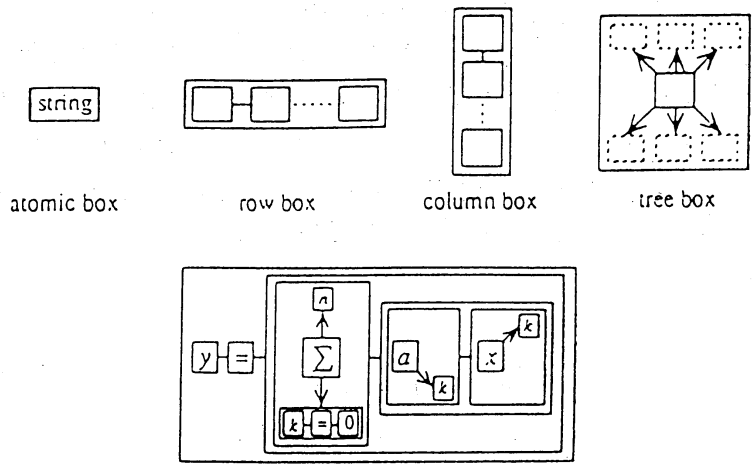

図 1 亀の甲羅文法

下図に，nMath-pの画面の例を示す．

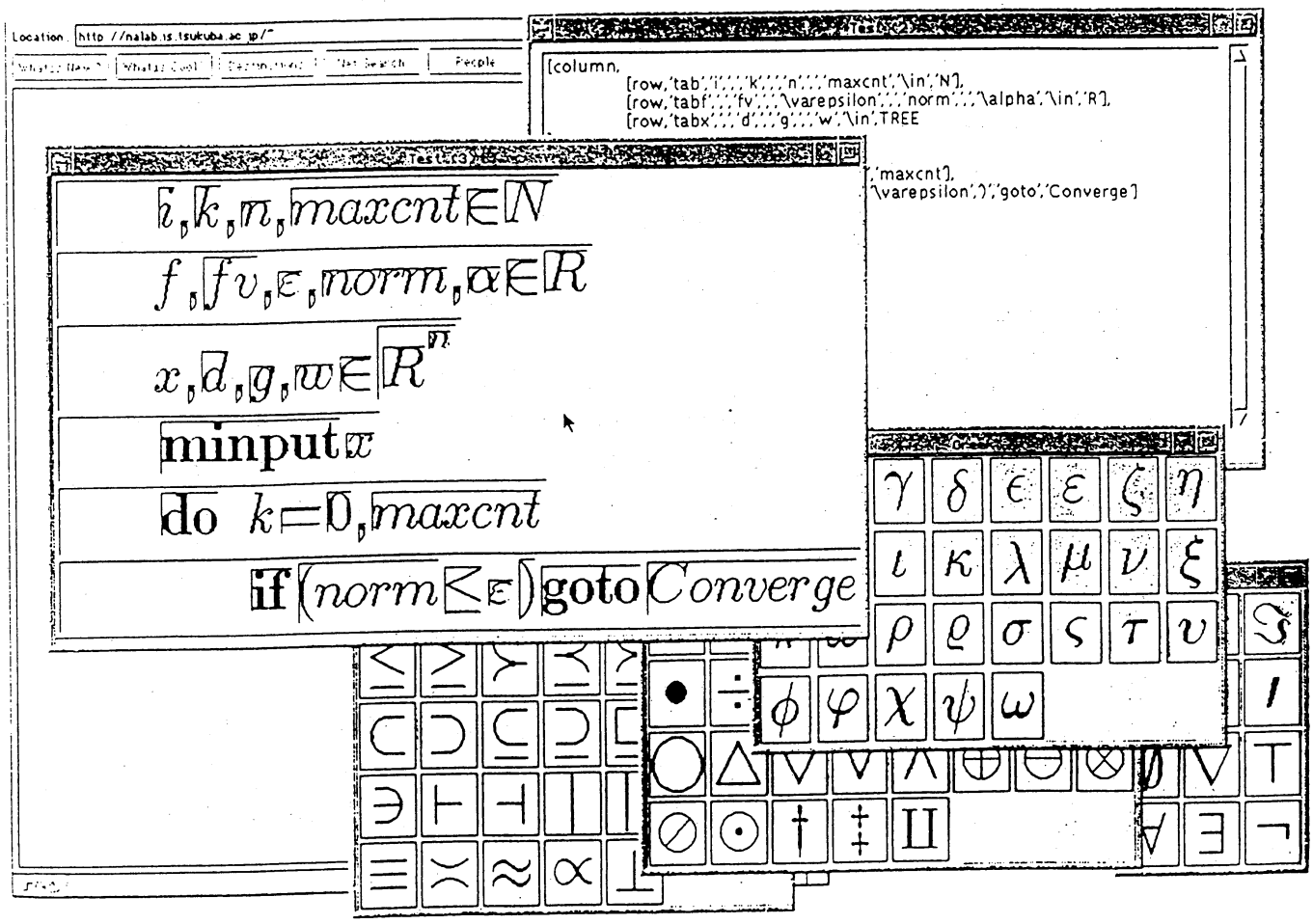

図 2 nMath-p 画面例

ユーザは，メニューから表現形式のテンプレートを選択できる．入力に際しては，画面上に現れる亀の甲羅のボックスに，必要な記号を入力する．ボックスは任意に入れ子を構成できるので，原理上どのような数学表現も編集可能である．

nMath-pはインターネット上で公開を原則としている．そのため，開発言語はJavaとしユーザはサーバ上のnMath-pを，ネットワークを通じて使用する．(公開開始に際しては，いくつかの媒体で公表する予定である) ただし，現在のJavaの制約上編集したファイルの保護や転送方法には，若干の工夫が必要であり，この点の解決を確認した上で公開することにしている．nMAth-pはインターネットに公開されることから，オープンなエディタと定義されることになる．

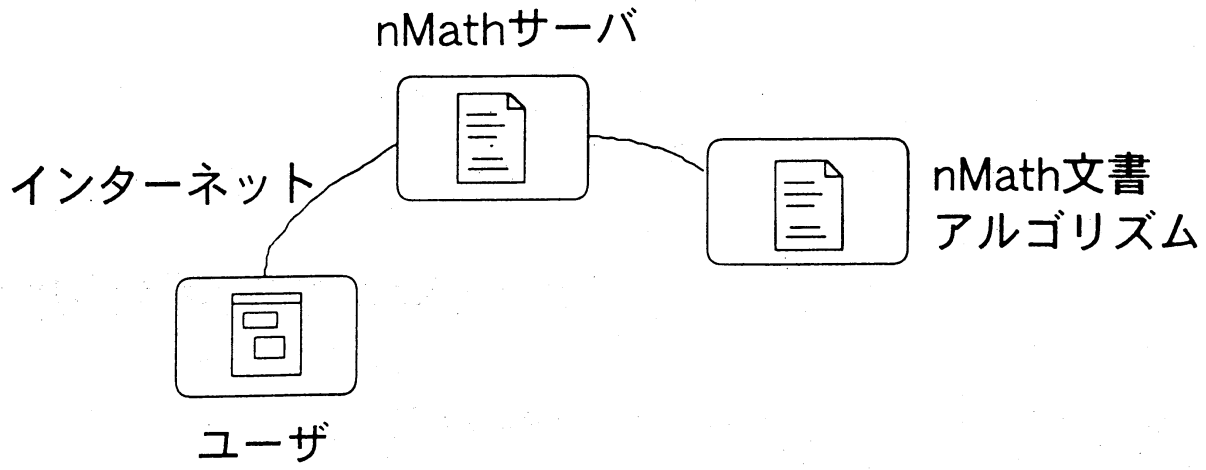

図3 nMath-pの公開方法

2.2 意味認識と内部表現

nMath-pにおける意味認識は，プロトタイプ版の制約から単純化している．本来の研究[2]では知識ベースを拠り所としているが，nMath-pでは数学表現のある単位をオブジェクトと考え，クラスの定義によって認識する．内部では，ユーザが選択するテンプレートとクラスを関連づけて，複数の文章にまたがる複雑な文脈の解釈を回避している．たとえば，

$$c = \sum_{i=1}^{n} a_i b_i$$

という表現では，シグマ記号のおよぶ範囲は，a_iまでか$a_i b_i$までかは曖昧である．nMAth-pでは最初にユーザがシグマテンプレートを選択し，そのおよぶ範囲もユーザによって誘導するようにしている．また，シグマに対応するオブジェクトが生成されるので，プログラム言語への変換では，和を作るコードを用意しておけばよい．以下に亀の甲羅文法で作られた上の式

$$c = \sum_{i=1}^{n} a_i$$

の内部表現を示す.

```
let[row,'c','=',{tree,\sum,lower,{row,'i','=','1'}}
                ,upper,'n'},{tree,'a',back_sub,'i'}]
```

現在のnMath-pでは，LAMAXに依存するため，和をつくるとき上記の表現をとらず，行列表現を用いている．内部表現の形式は以下に示すとおりである．[2]

```
<expression> ::= [ row, <expression list> ] |
                 [ column, <expression list> ] |
                 [ tree, <expression> {, back_sup, <expression> }
                                      {, upper, <expression> }
                                      {, front_sup, <expression> }
                                      {, front_sub, <expression> }
                                      {, lower, <expression> }
                                      {, back_sub, <expression> } ] |
                 ' <morpheme> '

<expression list> ::= <expression> | <expression>, <expression list>

<morpheme> ::= <sign in Normal Code> | <sign in Normal Code> <morpheme>
```

3 数学ソフトウェアの構造

数学ソフトウェアは，問題，アルゴリズムおよび実行環境の三つの要素として認識される．以下に簡単な例を示す．

- 問題
  - 名称 二次形式
  - 内容 $A \in R^{n\times n}, x, b \in R^n, n \in N$,$f : R^n \longrightarrow R$ とするとき，

    $$f(x) = \frac{1}{2}x^T Ax + b^T x$$

    の最小値をもとめよ．
- アルゴリズム

- 名称 ニュートン法
- 定義 $A \in R^{n \times n}, x, b \in R^n, n \in N$,$f : R^n \to R$, D:微分作用素
- **step1** xに初期値を与える.
- **step2** もし　$\|Df(x)\| < \epsilon$ ならば停止
- **step3** $x = x - (DDf(x))^{-1}Df(x)$　としstep2　へ

- 実行環境
  - 問題 二次形式
  - アルゴリズム ニュートン法
  - 入力 step1 でx を"data.dat"から
  - 入力 step1 の前にn, A, bを"data.dat"から
  - 出力 step2 で停止のときxを画面へ
  - 例外 step2 を１００回通過したら画面に"解が得られない"と出力し停止

上記のそれぞれは，nMath-pでは全くこのとおりの形式で入力可能である．nMAth-pは，それぞれの行に対応する文章形式を定形化しているので，文章の形式に式や単語をパラメータとして与えることで，対応するオブジェクトを生成することができる．後述のLAMAXとの連結実験では，LAMAXが持つアルゴリズム記述形式と文章形式のテンプレートを一致させることで，アルゴリズムの入力をあたかもソースプログラムとして受け入れ，実行させることに成功している．

ここで重要なのは，一般にプログラム言語において不可欠であった，入出力や例外処理などの本来のアルゴリズムとは異質な記述の混在が，nMathシステムでは安全に切り離されていることである．本稿のnMath-pではこの点は不完全であり，次の実験で取りくむ予定である．

4　行列演算用言語LAMAXとアルゴリズム記述

LAMAXは，FORTRAN77に強力な行列定義機能を付加した言語である．[1] 行列定義の可能な言語がいくつか知られているが，行列の数学的な性質（正定値性，対称性など）を明示的に取り扱い，適合する連立一次方程式の解法ライブラリと連結したり，各種のコンピュータの最適化ノウハウを取り込んで，最適化FORTRANコードを生成したりできることが，LAMAXの最大の特徴である．LAMAXを用いれば，数学的アルゴリズムの多くが，自然な数学表記に非常に近い形式で表されることが判っている．

LAMAXをnMathシステムの下流に連結すれば，自然なユーザインタフェースとともに，各種コンピュータの性能を最高に引き出す計算性能も獲得できるものと考えられる．

以下では，本稿で行った実験によるnMath-pの画面形式と内部形式，およびLAMAX形式を示す．

nMath-p 画面形式

comment 最急降下法

assign $i, k, n, maxcnt \in N$

assign $f, fv, \epsilon, norm, \alpha \in R$

assign $x, d, g, w \in R^4$

assign $Df \in R^*$

STEP 1 初期点を与える

input x

let $k = 0, n = 4$

STEP 2 収束判定

let $g = Df(x)$

let $fv = f(x)$

let $norm = absmax(g)$

if $norm < \epsilon$ then

output x

STEP 3 探索方向ベクトル

let $d = -g$

STEP 4 直線探索

let $\alpha = 1.$

do $i = 1, 10 \times n$

if $f(x + \alpha d) < fv$ then bleak

let $\alpha = 0.5\alpha$

STEP 5 次の点を求める

let $x = x + \alpha d$

let $k = k + 1$

goto step 2

内部形式

```
comment[row,'最急降下法']
assign[row,'i','','k','','n','','maxcnt','¥in','N']
assign[row,'f','','fv','','¥varepsilon','','norm','','¥alpha','¥in','R']
LAMAX[row,'parameter(n=4, eps=1.d-6, maxcnt=100)']
assign[row,[row,'x','','d','','g','','w','¥in',[row,'R',back_sup,'n']]
assign[row,'Df','¥in',[row,'R',back_sup,'¥ast']]
step[row,'ステップ 1 初期点の設定']
minput[row,'x']
do[row,'k','=','0','maxcnt']
step[row,'ステップ 2 収束の判定']
set[row,'g','=','Df','(','x',')']
set[row,'fv','=','f','(','x',')']
set[row,'norm','=','absmax','(','g',')']
```

```
LAMAX[row,'write(*,*) 'step:',k,' f(x):',fv,'norm:',norm']
if[row,'(','norm','.','le','.','eps',')','goto','収束']
step[row,'ステップ 3 探索方向 d を求める']
set[row,'d','=','-','g']
step[row,'ステップ 4 直線探索']
set[row,'¥alpha','=','1.d0']
do[row,'I','=','1',[row,'10','¥ast','n']]
if[row,'(',[row,'f','(',[row,'x','+','alpha','¥ast','d'],')'],'.','lt','.','fv',')','goto',
set[row,'¥alpha','=','0.5d0','¥ast','¥alpha']
enddo[]
step[row,'直線探索終了']
set[row,'x','=','x','+','¥alpha','¥ast','d']
enddo[]
```

LAMAX 形式

```
c 最急降下法
      integer*4 i,k,n,maxcnt
      real*8 f, fv, eps, norm, alpha
      parameter(n=4, eps=1.d-6, maxcnt=100)
      real*8:vector[n] x, d, g, w
      real*8:vector[*] Df
    1 continue !ステップ1  初期点の設定
      call minput(x)
      do 7 k=0, maxcnt
    2 continue !ステップ2  収束の判定
        g  = Df(x)
        fv = f(x)
        norm =  absmax(g)
        write(*,*) 'step:',k,'  f(x):',fv,'    norm:',norm
        if(norm .le. eps) goto 8
    3 continue !ステップ3  探索方向 d を求める
        d = -g
    4 continue !ステップ4  直線探索
        alpha = 1.d0
        do 5 i=1, 10*n
          if(f(x + alpha*d) .lt. fv) goto 6
          alpha = 0.5d0 * alpha
    5   continue
    6   continue !直線探索終了
        x = x + alpha*d
    7 continue
    8 continue !収束
      write(*,*) ' f= ', fv, ' value of x **********************'
      call mprint(x)
      end
```

5 おわりに

本稿で提案したnMathプロジェクトは，まだ端緒についたばかりである．しかしながら，LAMAXとの連携実験をとおして，その利便性が実証されたものと考えられる．今後は，エディタの性能の向上を図るとともに，資源の配布，印刷，実行ライブラリの拡充などの機能を充実させ，モデリング環境としての機能と数学文書の統合環境としての機能を持つ，名実ともに数学ソフトウェア環境の完成を目指したい．また，エディタの構造や中間形式を公開することによって，nMathプロジェクトへの参加を志す研究者がもし増えれば，これに優る喜びはない．さらに，コンピュータメーカやソフトハウスがnMath環境を発展させて，他の優秀な製品との融合を目指して動くことを希望したい．

参考文献

[1] S.Uchida,N.Yamaki and S.Hongo , "Enhanced Programming Environment for Mathematical Software", Proceedings of APORS'94,pp.469-476,World Scientific(1995)

[2] Y.Zhao,H.Sugiura,T.Torii and T.sakurai, "A Knowledge-Based Method for Mathematical Notations Understanding", Transactions of Information Processing Society of Japan,Vol.35,No.11,pp.2366-2381(1994)