計算数学 II　作業記録

05 班 “epsilon”，北川 弘典

2008/01/31 20:49

目次

1 注意事項

- 成果物の ε-pTEX は `http://www.ms.u-tokyo.ac.jp/~kitagawa/`で公開しております．
- 角藤さんが 2007/12/30 に pTEX + ε-TEX の実験[1)]を公開されたが，これは本プロジェクトとは全く関係のないものである．
- この文書は概ね着手した順番に並んでいるが，「形式化」以降は順番が若干入り乱れている．また後になるほど実際の作業と同時進行して書いた部分が多くなってくる．
- 本文書は計算数学 II の発表に用いるには長すぎるということで，発表当日 (2/1) に使うのは別に作ったスライドだけとした．しかし，もちろん，この文書は残すつもり．
- **一部の version の Adobe Reader ではヘブライ文字がちゃんとでないようですが，原因はまだよくわかりません．**
- 阿部 紀行さん，角藤 亮さん，「教職男」さん，「通りすがり 2」さんをはじめとする方々に（このようなところで申し訳ないが）深くお礼を申し上げます．

[1)] こちらは peTEX と呼称．正式にはどう logo を組むか分からないので，この表現で許してもらおう．p と e の位置が反対なのがおもしろいと共に幸いであるけれども，こちらの名称の由来は p(ε-TEX) という実装方法であったからだろう（と推測してみる）．peTEX の登場でこちらもやる気がかき立てられ，ε-pTEX がより完成度が高いものとなりえた．

2 テーマについて

2.1 背景

Knuth 教授の開発された狭義の TEX では，日本語が通らない．それを日本語に対応させようという試みとして，NTT JTEX[2] とアスキーの pTEX[3]が有名であり，後者の pTEX は縦組みもできるということもあって，日本では標準的になったといってもよい．（狭義の）TEX に対応する LATEX マクロについても，JTEX, pTEX のそれぞれに対応版が存在し，接頭辞を拝借して pLATEX などのように呼ぶ．

一方，それ以外の TEX の拡張として，例えば次のようなものがある．

- ε-TEX (eTeX). 各種機能が拡張されている．詳細は後述．
- Ω (Omega). TEX を Unicode に対応させて多言語処理を実装させたものである．これに対応する LATEX マクロは Λ (Lambda) と呼ばれる．公式ページは `http://omega.enstb.org/`.
- ℵ(Aleph). Ω に ε-TEX 相当の機能を取り込んだもの．対応する LATEX マクロは ℶ (Lamed) と呼ばれる．
- pdfTEX. TEX source から直に pdf を作成できるようにしたもの．今は ε-TEX がベースとのこと．公式ページは `http://www.tug.org/applications/pdftex/`.

さて，TEX はある種のプログラミング言語であり，変数，つまりレジスタも `\count`（4 byte 整数），`\dimen`（2^{-16} pt 単位で，整数部 14 bit，小数部 16 bit の固定小数点数），`\box` などのいくつかの種類があり，各種類 0 ～ 255 の index で，つまり 1 種類につき 256 個ずつ使えるようになっていて，pTEX でもそれを引きずっている．しかし，MusiXTEX などのような巨大なマクロパッケージでは，全く同じ source でも（pLATEX では日本語まわりで余計にレジスタを使うので）欧文 LATEX では処理できるが．pLATEX では処理できないといった事態がすでに発生している．一方，前出の ε-TEX による拡張では，レジスタは各種 2^{15} 個使えるように改良されているので，拡張機能を使う状況では実質的には起こりえないと思ってよい..

欧文の LATEX を使おうとして端末上で `latex` と打つと，そこで起動するのは元々の TEX ではなく，大幅に機能拡張された pdfTEX が互換モードで動いているという状況が最近では発生している，現に，作業を行った PC では TEX Live 2007 という TEX 関連のファイルをまとめたもの（pTEX は入ってない）に，自分で pTEX を追加して使っているのだが，以下のように pdfTEX が起動した．このことから考えると，世界的には ε-TEX の機能が利用できることが当たり前になりつつあるといってもよいのかもしれない．つまり，特に，各種 256 個というレジスタの制限はもはや obsolute になりつつあるといえるだろう．

```
[h7k doc]$ latex
This is pdfTeX, Version 3.141592-1.40.3 (Web2C 7.5.6)
（後略）
```

2.2 テーマの提示

前置きが長くなったが，以上の背景から，本班では **pTEX に ε-TEX の機能の一部，特にレジスタ数の話についての機能を組み込む**ということを第 1 の目標とした．

第 2 の目標は半分ネタである．TEX ソース内で使用できる算術演算はすべて整数（と固定小数点数）の四則演算であるため，例えば 3 次方程式の解の近似計算などは Fortran とか C とかの他の言語でやらなければ

[2] 千葉大の桜井さんがメンテ．`http://www.math.s.chiba-u.ac.jp/~sakurai/software.html`.

[3] `http://www.ascii.co.jp/pb/ptex/`が公式ページである．

ならなかった．これを TEX の内部でできたら，いちいち計算用プログラムの結果を貼り付けたり \input で読んだりする必要が無くて，楽になるに違いない[4]，と僕は考えたので，（速度はある程度は度外視することにして）**TEX への浮動小数点演算の実装**を次の目標にした[5]．

3 TEX 周辺の compile 方法

pTEX の改造をするためには，それがどのように compile されているかを知らなければもちろんできない．僕は昔から TEX 関係のプログラムを自分でソースからコンパイルしていた（Unix 系統なら土村さんによる ptetex3 のおかげで，だいぶ簡単になった）ので知っていたのでこれを調べる時間はほとんど 0 で済んだ．

TEX の一番元々のソースは tex.web というファイルであり，はっきり冒頭に

```
% This program is copyright (C) 1982 by D. E. Knuth; all rights are reserved.
% Copying of this file is authorized only if (1) you are D. E. Knuth, or if
% (2) you make absolutely no changes to your copy. (The WEB system provides
% for alterations via an auxiliary file; the master file should stay intact.)
% See Appendix H of the WEB manual for hints on how to install this program.
% And see Appendix A of the TRIP manual for details about how to validate it.

% TeX is a trademark of the American Mathematical Society.
% METAFONT is a trademark of Addison-Wesley Publishing Company.
```

とあるように，これが Knuth 教授の製作したファイルそのままである．このファイルは WEB という言語/書式で書かれている．

3.1 WEB の説明

ここについては松山 [1] を読んだ方が，以下の僕の下手な説明よりわかりやすいかもしれないが，まあ，お付き合いください．

WEB の詳細は Knuth [2] とかから得られるが，大雑把に言うと，Pascal でかかれたプログラムのソースを小さく分割して，各分割の断片は TEX ソースによる説明文を必要なら付属させて，それらを（ほとんど任意の順序で）並べたものである．上の説明では非常にわかりにくいので，実例を別に挙げるこおにしておこう（付録参照）．

さて，WEB で書かれたファイル（例えば fuga.web）を tangle というプログラムに通すと，そこから Pascal で書かれたプログラムのソースだけが抜き出され，fuga.pas とか fuga.p ができる．一方で，weave というプログラムに通すと，fuga.web 中に書かれた解説文が Pascal ソースとともに整形され，TEX ソース fuga.tex を得る．これを TEX で処理すると，綺麗に整形されたソース付きのドキュメントが得られる，という寸法である[6]．

また，WEB には change file という機能が用意されており，ちょうどパッチファイルの役割を果たす．構造も似ており，

---

[4] そう考えるのは多分僕だけだ．

[5] Beebe[13] の title は Extending TEX and METAFONT with Floating-Point Arithmetic であるが，そちらは「Lua 等のスクリプト言語による（IEEE 754r に基づくような）実装」というアプローチを示しているだけである．

[6] ちなみに，tangle は「もつれる/させる」，weave /wíːv/は「織る」という意味である．tangle は WEB ソース中に適当な順序で置かれたプログラムの各断片をちゃんと並べ直すが，吐かれた Pascal ソースは読もうとする気が失せるほどぐちゃぐちゃである．

```
@x
変更前の部分
@y
変更後の部分
@z
```

の繰り返しからなっている．@x の行が@x l.1701 hogehoge... などのようになっている（注釈が書かれている）ことも多い．tangle や weave の引数にはこの change file を WEB ソースとともに指定することができて，例えば，

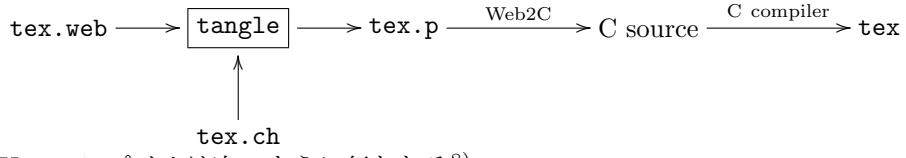
```
> tangle fuga.web fuga.ch
```

とすると，fuga.web の内容を fuga.ch でパッチしたものについて，tangle 処理が行われる．また，tie というプログラムを使えば，WEB ソースと change file から，change file 適用後の WEB ソースを作ったり，複数の change file をまとめることができる[7]．

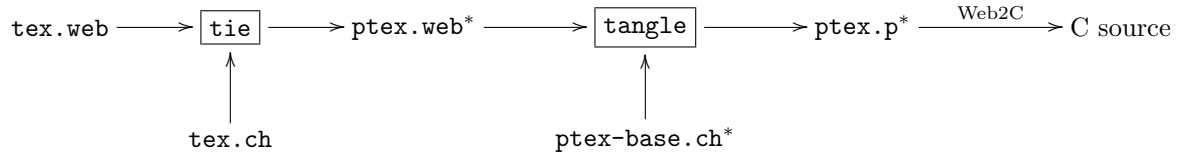
3.2 Web2C と TEX のコンパイル

さて，tangle で tex.web を処理すると tex.p が生成されるが，現在は Web2C という実装方法でこれを処理している．詳細は松山 [1] に譲るが，tex.web に tex.ch という Web2C 実装特有の拡張やシステム依存部分（入出力とか，時計）を納めたパッチを適用して，それを tangle にかけて tex.p を生成する．その後，web2c というプログラム（と付属する小ツール）によって，tex.p は texini.c, tex0.c, ... という C ソースにまで変換される．それらと（もともと C ソースで準備されているシステム依存部分である）texextra.c とかをコンパイル，リンクして，ようやく実行可能形式が出来上がるというわけである．

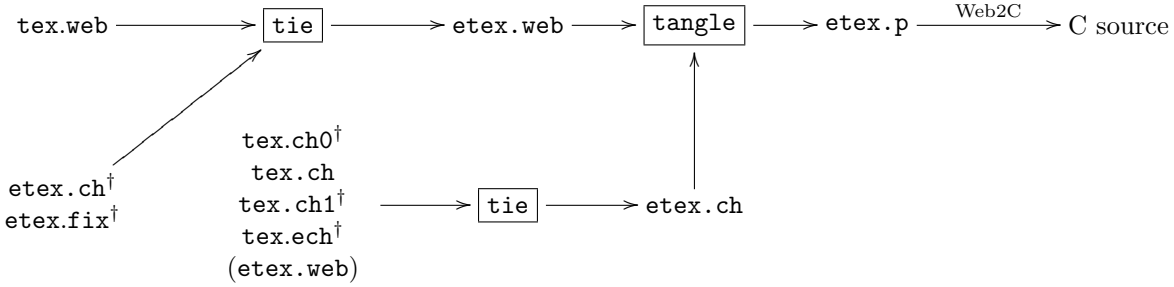
これを図にしてみよう：



同様に，pTEX のコンパイルは次のように行われる[8]：



さらに，今回の準主役とも言える ε-TEX では，こうなっている：



ここで，∗, † とかはファイルのあるディレクトリを表している．なお，etex.fix は実際には中身はなかった．

---

[7] 厳密には，これは CWEB のプログラムらしい．名前の由来は英単語が「結ぶ」という意味だからであろう．Star Wars の某帝国軍戦闘機とは関係はないだろう．

[8] 最後の C compiler のところは省略した．また，厳密には ptex-base.ch と ptex.web から ptex.ch を作り，それを適用して tangle に食わせているが，そこら辺は省略．

ここから，第 1 の目標，つまり pTEX に ε-TEX の機能の一部をマージすること，を実現させるためには，大雑把に言えば，tex.web＋tex.ch＋ptex-bash.ch* ＋etex.ch† とするか，tex.web＋tex.ch＋etex.ch† ＋ptex-bash.ch* と（ここでの加法はもちろん非可換である！）すればよいだろう，と想像がつく．どちらを選んでもよかったが，ここでは pTEX をベースとしたいという意味で，前者を採用することにした[9]．

4 ε-TEX のマージ

4.1 pTEX の準備

というわけで，主に編集するのは etex.ch である．しかし，pTEX を元に作業をするので，pTEX がコンパイルできることを確かめておかねばならない．その手順は大まかには TEXwiki [3] の後半部にある「古い情報」にある通りである．しかし，teTEX 3.0 下では無事にあるようにコンパイルができるのだが，TEX live 2007 だと Web2C のバージョンが新しく，途中で以下のようなエラーを出して止まってしまった：

```
gcc -DHAVE_CONFIG_H  -I. -I.. -I../.. -I../..   -O2 -s -march=pentium-m -fomit-f
rame-pointer  -c ptexextra.c -o ptexextra.o
ptexextra.c: In function 'parse_options':
ptexextra.c:956: error: too few arguments to function 'printversionandexit'
ptexextra.c: In function 'getjobname':
ptexextra.c:1722: error: number of arguments doesn't match prototype
texcoerce.h:908: error: prototype declaration
make: *** [ptexextra.o] Error 1
```

これは printversionandexit 関数と getjobname 関数の宣言が Web2C の version up に伴って変わってしまったのが原因である．修正箇所は ptex756.diff に diff の形で書いている．また，tftopl とかについては tangle の段階でエラーが出るようであるが，これらは本 project と関係ないので省略する：

```
This is TIE, CWEB Version 2.4. (Web2C 7.5.6)
Copyright (c) 1989,1992 by THD/ITI. All rights reserved.
(../tftopl.web)
(../tftopl.ch)
....500....1000....1500
(No errors were found.)
../tangle ./tftopl.web tftopl.ch
This is TANGLE, Version 4.5 (Web2C 7.5.6)
*1*6*18*26*44*88*96*100
! Hmm... 1 of the preceding lines failed to match. (change file l.378)
@y
   *114*126
Writing the output file.....500..
Done.
(Pardon me, but I think I spotted something wrong.)
make: *** [tftopl.p] Error 1
```

さて，とりあえずのたたき台として pTEX の WEB ソース，つまり tex.web, tex.ch, ptex-base.ch を 1 つにまとめた ptex-orig.web を tie によって作成した．その後，ptex-orig.web 中の最初の部分である「This is TeX」の部分などを若干修正した．修正箇所は diff の形で ptex-orig.diff としてある．

[9] これにより project 名が **ε-pTEX** と決まった．ε-(pTEX) のつもりである．もちろんこの文書も ε-pTEX で組版している．

4.2 ε-TEX の機能

ε-TEX のマニュアル [4] を元に ε-TEX の主な機能を見てみることにする．ε-TEX には Compatibility mode と Extended mode の 2 つが存在し，前者では ε-TEX 特有の拡張は無効になるのでつまらない．後者がおもしろい．

拡張機能を使うにはファイル名を渡すときに * をつけるかコマンドラインオプションとして -etex スイッチをつければいいが，ε-TEX 拡張に関わる追加マクロ[10]は当然ながらそれだけでは駄目である．「plain マクロ for ε-TEX」（etex.fmt というのが一番マシかな）では自動的に追加マクロである etex.src が呼ばれる．LATEX 下ではちょうど etex.src に対応した etex パッケージを読み込む必要がある．

■レジスタの増加

最初に述べたように，TEX では 6 種類のレジスタが各 256 個ずつ利用できる．それぞれのレジスタには \dimen75 などのように 0 ～ 255 の番号で指定できる他，予め別名の定義をしておけばそれによって指定することもできる[10]．これらのいくつかは特殊な用途に用いられる（例えば \count0 はページ番号などのように）ことになっているので，さらに user が使えるレジスタは減少する．

ε-TEX では，追加のレジスタとして番号で言うと 256 ～ 32767 が使用できるようになった．上の pdf によると最初の 0 ～ 255 と違って若干の制限はあるようだが，それは些細な話である．追加された（各種類あたり）$32768 - 256 = 32512$ 個のレジスタは，メモリの効率を重視するため sparse register として，つまり，必要な時に始めてツリー構造の中で確保されるようになっている．

■式が使用可能に

TEX における数量の計算は充実しているとは言い難い．例えば，

\dimen123 ← (\dimen42 + \@tempdima)/2

という計算を元々の TEX で書こうとすると，

```
\dimen123=\dimen42 \advance\dimen123 by \@tempdima \dimen123=0.5\@tempdima
```

のように書かないといけない．代入，加算代入，乗算代入，除算代入ぐらいしか演算が用意されていない状態になっている（上のコードのように，$d_2 \mathrel{+}= 0.8d_1$ というような定数倍を冠することは平気）．

ε-TEX では，そのレジスタの演算に，他のプログラミング言語で使われているような数式の表現が使えるようになった．上の PDF では実例として

```
\ifdim \dimexpr (2pt-5pt)*\numexpr 3-3*13/5\relax + 34pt/2<\wd20
```

が書かれている．これは，

$$32\,\mathrm{pt} = (2\,\mathrm{pt} - 5\,\mathrm{pt})\,(3 - \mathrm{div}(3 \cdot 13, 5)) + \frac{34\,\mathrm{pt}}{2} < \texttt{\box20 の幅}$$

が真か偽かを判定していることになる[11]．

[10] 例えば，TEX ではレジスタに \@tempcnta などの「別名」をつけることが非常によくある．この機能については我々が実際に使うのはは plain マクロや LATEX マクロによって提供されており，当然 256 個制限の環境下を前提として組まれている．

[11] div は整数除算を表したつもりである．整数除算は余りが被除数の符号と等しくなるように計算される．

■ **\middle primitive**

TEX に \left, \right という primitive があり，それを使えば括弧の大きさが自動調整されるのはよく知られている．ε-TEX では，さらに \middle primitive が追加された．

具体例を述べる．

$$\left\{ n+\frac{1}{2} \middle| n\in\omega \right\} \left\{ n+\frac{1}{2} \middle| n\in\omega \right\}$$

これは以下の source で出力したものである：

```
\def\set#1#2{\setbox0=\hbox{$\displaystyle #1,#2$}%
\left\{\, \vphantom{\copy0}#1 \,\right|\!\left.\, \vphantom{\copy0}#2 \,\right\}}
\def\eset#1#2{\left\{\, #1 \,\middle|\, #2 \,\right\}}
\[ \set{n+\frac12}{n\in \omega} \eset{n+\frac12}{n\in \omega} \]
```

両方とも集合の表記を行うコマンドである．TEX 流の \set では 2 つの \left, \right の組で実現させなければならず，そのために | の左側と右側に入る式の最大寸法を測定するという面倒な方法を使っている．その上，この定義では \textstyle 以下の数式（文章中の数式とか）ではそのまま使えず，それにも対応させようとすると面倒になる．一方，ε-TEX 流の \eset では，何も考えずに \left, \middle, \right だけで実現できる．

■ **TEX--XETEX** (TeX--XeT)

left-to-right と right-to-left を混植できるという機能であるらしい．ヘブライ語あたりの組版に使えるらしいが，よく知らない．ここでの RtoL は LtoR に組んだものを逆順にしているだけのような気がする．

とりあえず一目につきそうな拡張機能といったらこれぐらいだろうか．他にも tracing 機能や条件判断文の強化などあるが，そこら辺はパッとしないのでここで紹介するのは省略することにしよう．

この中で，TEX--XETEX機能だけは，「TEX から pTEX への変更部分とかちあって，作業に非常に手間がかかる」と予想した（根拠はなし．あえて言えば勘）．それに，僕は右→左の言語をアラビア語やヘブライ語ぐらいしか知らず，これらは ArabTEX などを使うことで pLATEX でも入力できるので，ひとまず TEX--XETEX 機能だけは実装を見送り[12]，他はできる限り実装することにした．

4.3 etex.ch の改変

ここから実際の作業であり，ひどく退屈な作業である．まず etex.ch 先頭の著作権等に関する部分を書き換えておく．

etex.ch 中の@x ... @y ... @z のところにはありがたいことにコメントがついていて，どういう理由でパッチが当たるのかと，当たる行番号までわかるようになっている[13]．次は etex.ch の抜粋である：

```
%-----------------------------------
@x [15] m.208 l.4087 - e-TeX saved_items
@d un_vbox=24 {unglue a box ( \.{\\unvbox}, \.{\\unvcopy} )}
@y
@d un_vbox=24 {unglue a box ( \.{\\unvbox}, \.{\\unvcopy} )}
  {( or \.{\\pagediscards}, \.{\\splitdiscards} )}
@z
%-----------------------------------
```

[12] 目次からも分かる通り，最終的には実装することになるが．

[13] 但し，ここに書かれている行番号は tex.web に対してのそれであろう．しかし ptex-orig.web でもあまり行番号は変わらない．

```
  @x [15] m.208 l.4097 - e-TeX TeXXeT
  @d valign=33 {vertical table alignment ( \.{\\valign} )}
  @y
  @d valign=33 {vertical table alignment ( \.{\\valign} )}
    {or text direction directives ( \.{\\beginL}, etc.~)}
  @z
  %---------------------------------------
  @x [15] m.208 l.4113 - e-TeX middle
  @d left_right=49 {variable delimiter ( \.{\\left}, \.{\\right} )}
  @y
  @d left_right=49 {variable delimiter ( \.{\\left}, \.{\\right} )}
    {( or \.{\\middle} )}
  @z
%---------------------------------------
```

TEX--XETEX機能の実装はとりあえず見送ったと書いたので，上で言うところの e-TeX TeXXeT などとコメントが振られている部分をバッサリ削除するとともに，ptex-orig.web にそのまま etex.ch が適用できないところを調べ，tex.web と ptex-orig.web も見ながら修正していく．ε-TEX 拡張の本質的な部分は，（pTEX でもそうだったが）etex.ch の後半部の以下の部分に集中して現れるので，そこでも TEX--XETEX関連と思われる場所を削除した．

```
  %---------------------------------------
  @x [54] m.1379 l.24945 - e-TeX additions
  @* \[54] System-dependent changes.
  @y
  @* \[53a] The extended features of \eTeX.
  （中略）
  @* \[54] System-dependent changes.
  @z
  %---------------------------------------
```

削除しすぎたような場合は tangle 時に警告メッセージが表示される（断片が欠けていた場合）か，C source のコンパイルでエラーが出るので，退屈ではあったが，そんなに厳しくは無かった．

なお，ptex-orig.web にそのまま etex.ch が適用できないところを調べるのは簡単である．そのようなところがあった場合，tie で適用後の WEB ソースを作ろうとすると，以下のようなメッセージが表示されるからである．

```
  [h7k eptex]$ tie -m ep.web  ptex-orig.web ../etexdir/etex.ch
  This is TIE, CWEB Version 2.4. (Web2C 7.5.6)
  Copyright (c) 1989,1992 by THD/ITI. All rights reserved.
  (ptex-orig.web)
  (../etexdir/etex.ch)
  ....500....1000....1500....2000....2500....3000....3500....4000....4500....5000.
  ...5500....6000....6500....7000....7500....8000....8500....9000....9500....10000
  ....10500....11000....11500....12000....12500....13000....13500....14000....1450
  0....15000....15500....16000....16500....17000....17500....18000....18500....190
  00....19500....20000....20500....21000....21500....22000....22500....23000....23
  500....24000....24500....25000....25500....26000....26500....27000....27500....2
  8000....28500..
  ! Change file entry did not match (file ../etexdir/etex.ch, l.200).

  (Pardon me, but I think I spotted something wrong..)
```

ちょっとずつ etex.ch を改変するごとに上のようなコマンドを走らせれば，次に修正するべき場所がわかるという話である．

4.4 TRIP test, e-TRIP test

TRIP test とは，Knuth 教授による TEX の debug の補助ツールである．これは TEX のソースコードの全部分を渡るように作られたテスト入力であり，大量の log を出力する．一度自分の計算機で TRIP test を行い，その結果が「正しい」出力と異なっていたなら，それはどこかに bug が潜んでいるに違いないということを意味する．Knuth 教授自身も，Knuth [5] の中で以下のように述べている．

> If somebody claims to have a correct implementation of TEX, I will not believe it until I see that TRIP.TEX is translated properly. I propose, in fact, that a program must meet two criteria before it can justifiably be called TEX: (1) The person who wrote it must be happy with the way it works at his or her installation; and (2) the program must produce the correct results from TRIP.TEX.

その意味では，pTEX の段階でももはや TRIP test はパスしない．TEX で出ていた

```
! Bad character code (256).
```

という error はもはや出ないし，内部モードが出力される所でもいちいち yoko direction などが出力される．しかし我々は pTEX をベースとしているので，ε-pTEX の動作をチェックするという今の文脈では．pTEX による出力を TRIP test の「正しい」出力と考えてもよいだろう．

さらに，今回は ε-TEX の機能も組み込んでいるのだから，ε-TEX による TRIP test の出力とも比較する必要があった．Extended mode の出力は当然 TEX による TRIP test の出力とは異なる．幸い，その内容は文献 [6] に記述されている．また，ε-TEX では，ε-TEX 特有の拡張機能をテストするための，同様の e-TRIP test というものが別に用意されている．

ひとまず TEX--XeTeX 以外を追加した ε-pTEX により TRIP, e-TRIP test を行わせてみた．出力結果は，pTEX による結果とも ε-TEX による結果とも若干異なっていたが，その差異は，TEX → pTEX による出力結果の差異と TEX → ε-TEX による出力結果の差意を合わせたものでしかない．また，ε-pTEX による e-TRIP test の結果は ε-TEX のそれと大きく異なるが，それは「ε-pTEX が TEX--XeTeX を実装していないことによるもの」「pTEX に合うように etex.ch を変更したことによる，内部パラメタの若干の変更」「pTEX の機能による，yoko direction とかの追加出力など」がほとんどであり，十分に予測ができたものであった．その意味では，（日本語処理以外には）critical な bug はこの時点でなかったと言っていいだろう[14]．

この TRIP, e-TRIP test は，実際に ε-pTEX の debug に役に立った．11/1 近傍で，すでに「通常のソースファイルを入力しても出力は同じ」状況にはなっていた．しかし，TRIP test をかけると，異常な penalty 値[15]の出力が見られた．そこで，ε-TEX による penalty 拡張を ε-pTEX に実装するところに bug が埋まっていると見当を付けて，そこを見直した結果，etexdir/etex.ch（つまり，もともとの TEX → ε-TEX の change file）にある 2 箇所の修正箇所が，ε-pTEX 用の etex.ch には何故か抜け落ちていたことが分かり，この ε-pTEX の bug を修正することができたのである．

なお，この時点でも ε-pTEX は ε-TEX の TEX--XeTeX 以外の全ての機能を実装していたことを注意しておこう．

---

[14] なお，第 6 章でさらなる TRIP, e-TRIP test をパスさせるための debug を行っている．

[15] penalty （罰金）とは，改行/改ページの制御に使われる量であって，box, glue とともに TEX での組版において重要な役割を果たしている．

5　浮動小数点演算の実装

もう一つのテーマの柱，浮動小数点演算の実装に入ろう．

まず，サポートする精度についてであるが，計算速度から言えば 4 byte の所謂「単精度」が良く，コードも簡単になる．しかし単精度は有効桁数が 7 桁前後しかない．しかし，実際の数値計算では，この 7 桁程度の精度では不足し，倍精度以上が使われる場合が多いと思われる．浮動小数点演算の実装の目的が「TEX に数値計算をやらせる」ことであるので，倍精度以上は必須である．

5.1　基本方針

何よりも，TEX の既存部分の WEB ソースを改変するところを最小限に留めることを最優先にした．使い勝手から言えば，新しいレジスタの種類を（例えば \real とかいう名前で）定義して，他のレジスタと同様に扱えるようにしたほうがよいのだが，それは TEX の内部処理を大きく変更することとなり，bug も混じる可能性も増える．

ここで，TEX の数量にまつわるレジスタの種類を思い出そう：

- \count：−2147483647 から +2147483647 までの整数 (number) を格納する．
- \dimen：$1\,\mathrm{sp} = 2^{-16}\,\mathrm{pt}$ を刻み幅として，絶対値が $2^{30}\,\mathrm{sp} = 16384\,\mathrm{pt}$ より小さい長さ (dimension) を格納する．
- \skip：いわゆる glue．自然長 (width part)，伸び量 (stretch part)，縮み量 (shrink part) を持っており，後者 2 つは無限大も指定できる（しかもその無限大には rank があったりする）．
- \muskip：数式モードでの \skip に相当する．

前者 2 つは内部でも 4 byte であり，後者 2 つは 4 byte 変数 3 つ分を記憶領域として使っている．TEX のソースを見ると，レジスタがこれら（と \toks, \box）だけであることに依存したコードが書かれていたので，新たに浮動小数点数用のレジスタと専用の記憶領域を確保したりするのは労力がかかるだけだと判断した．そのように考えてみると，\skip が 12 byte 以上を使っていることになる．

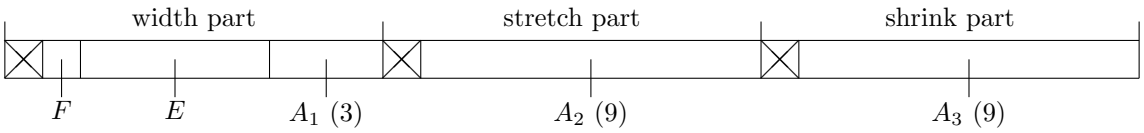
すると，1 つの glue に浮動小数点数 1 つを納めるようにするのが楽である．せっかく 12 byte あるのに 8 byte 分しか使わないのは勿体ないので，12 byte 分にできる限り収まる限りの精度とすることにした．なお，Web2C でサポートされている Pascal の機能は限定的なものであるので，Pascal 規格内にある浮動小数点演算のサポートについても不安である．そこで，ここでは**整数演算のみを用いて実装**することとした．そのため，仮数部の演算は本質的には多倍長整数の演算であるので速度は非常に遅いことを覚悟しなければならない..

高レベルな機能は TEX のマクロ機能を使い実装することを考えると，とりあえず実装すべき機能として，以下を決めた：

- 浮動小数点定数値を表現する glue を返す：\real[16]．
- 2 項演算：\fpadd, \fpsub, \fpmul, \fpdiv, \fppow．結果は第 1 引数のレジスタに上書き．
- 単項演算：\fpneg，その他初等関数達．
- 浮動小数点数の出力，及び型変換．

[16] 他のコマンドが fp を接頭辞にしているのにこれだけそうじゃないのは変かもしれないが，かといって \fp にするのは短すぎる気がするし，演算コマンド側を \realadd などとすると長すぎると思ったからである．当然ながら，Fortran の REAL*x が根底にある．

格納形式は色々と悩んだが，最終的には以下に述べるように落ち着いた．2 進で格納すると基数変換とかいう変なものを組む必要がある[17]ので，10 進で行った．但し指数部については諸事情により 16 bit 分としている．



一つの glue の自然長，伸び量，縮み量は TEX 内部ではそれぞれ 32 bit 整数として扱われている．これらをこの順に並べて，1 つの 96 bit 領域とみなして書いたのが上の図である．

1. 各 32 bit 部分では上位 2 bit は未使用とする[18]．それぞれの残り 30 bit で 10 進 9 桁を格納する．
2. 仮数部は図の A_1, A_2, A_3 で示した領域とする．これらの後ろに () でくくった数字はそこの 10 進での桁数を表す．合計で 10 進 21 桁分の領域となり，（図の）左側を上位として 21 桁を格納する．小数点が A_1 での 100 の位と 10 の位の間に来た状態を「正規化されている」と称する．
3. 指数部は E で示した部分で，2 進数で 16 bit 分の領域をもっている．ここには実際の指数（−32767–32767）を 32767 をバイアスした値が格納される．但し，この E の値が 65535 のときは，（正負の）無限大として扱う．
4. F で表した部分は 0, 1, 2 のどれかが格納され，符号 bit としての役割を持つ．0 は正，1 は負，2 は NaN を表す．
5. 以上の説明では見にくいので，この glue が表現している値を述べると，以下のようになる．

$$\begin{cases}(1-2F)(10^{18}A_1+10^9A_2+A_3)\cdot 10^{E-32767-20}, & \text{if } F\neq 2;\\ \text{NaN}, & \text{if } F=2.\end{cases}$$

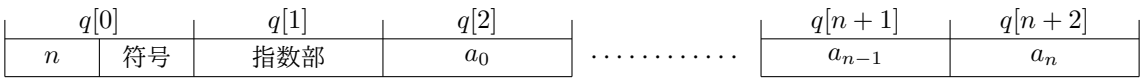
また，width part に実際に格納されている値は $1000(E+65536F)+A_0$ である．

TEX ソース中での浮動小数点数の表記は，他のプログラミング言語でもよく用いられているごく普通の表記を採用するが[19]，TEX 本来の読み取りルーチンを一部流用した部分があるため，小数点は大陸式の ‘,’ も認め，符号は複数あってもいい（その場合，実際の符号は全部掛けたものになる）ことになった．

5.2 内部表現と四則演算

実際の演算を glue に「パックされた」状態で行うのは余り良くないように感じた．なぜなら，初等関数の実装を最初から念頭に置いていたが，それを 21 桁精度のままで計算するのは精度の面で非常に不安だったからである．そうすると，内部での計算の実装は，21 桁とは限らない任意長[20]の計算に対応できるようにした方がよい．

上を考慮に入れて，浮動小数の内部表現である 4 byte 整数型の配列 q を以下のように定める：



ここで，各 a_i, $0\le i\le n+1$ は 0 ～ 999 の範囲の自然数であって，符号部は（前節のように）0 が正，1 が負，2 が NaN を表す．即ち，仮数部を基数 1000 で表現している[21]ことになる．a_n は桁上がりなどの際

[17] もちろん debug 時であっても．僕は暗算は異常に弱いし．

[18] dimension の表現では最上位 bit は符号，次の bit は Overflow 検出用のような意味を持っている．そのため，これらの 2 bit も使ってしまうと何かと面倒になるように思われた．

[19] Fortran では指数部に `D` や時には `Q` などを許容したりしているけれども，それらは ε-pTEX では認めない．

[20] とはいっても，さすがに数千桁とか数万桁とかはたかが 21 桁の計算の内部処理としてはしないだろう．個人的には 60 桁もあれば十分と感じている．

[21] 乗法を行う場合，基数の平方が処理系の内部で扱える必要はないように感じるが，それを考えると，基数に 32768 とか

に用いられ，所謂「正規化された状態」では $a_n = 0$, $a_{n-1} \geq 100$ である．また，指数部は（正規化された状態では）$-32767 \sim 32767$ の範囲の整数が通常の浮動小数点数を，32768 が無限大を表す．NaN や 0 に符号の区別はなく，当然ながら IEEE 754/754r 規格には全然適合しない．n を「浮動小数点数の長さ」と以降呼ぶことにする．

このような配列は，最初の実装では，その都度動的に確保，解放することにした[22]．TEX でのメモリ管理はすべて TEX 自身が行う[23]ことが伝統的になっている ([8]) ので，ここでも当然それに従う．

浮動小数点の四則演算を大きくわけると，仮数部同士の計算と，指数部の調整に 2 分される．前者は前に書いたように多倍長整数の四則計算となるので，Knuth [7] にある「古典的な」アルゴリズムをそのまま用いることができる．

これらのアルゴリズムは（割り算を除けば）そんなに長くはなく，指数部や符号の処理，無限大や NaN の処理の方が分量的には多い．整数演算として得られた仮数部を正規化する処理や，加法や減法で行われる 2 引数の指数部を揃える処理，また乗法や除法で行われる，列の長さを延長したり縮小したりする処理の方にいっそう気を使った．

5.3 入出力と型変換

演算はできるようになっても，それを入出力することができなければどうしようもない．浮動小数点は glue として `\skip` レジスタに格納されるので，`\skip` レジスタに対して直に pack された値を指定したり，あるいは `\the` で出力をしたりすれば一応はいいのだが，それでは非常に使い勝手が悪い．

入力側についてのほとんどの処理は浮動小数点数を読み込み，内部表現の配列の形に変換して返す *scan_float* procedure によって行う．これは整数を読み込む *scan_int* や長さを読み取る *scan_dimen*，glue を読み取る *scan_glue* の各 procedure に対応するものとして考えたもので，コードも大いに参考にさせてもらった．2 項演算のコマンドの 2 つめの引数を読み取る部分もこの procedure「だけ」で行われる．

scan_dimen では整数部分を読むのに *scan_int* を用いて，次に小数部を独自に読み取るという方法をとっていたが，同じ方法では 21 桁の数値を読み取ることはできない．そこで，*scan_float* では，先に（すぐ後で述べる）内部表現の配列を 21 桁分確保し，それに上の方から数字を詰めていくという方法をとった．22 桁め以降が存在した場合は，Knuth [7] に従い「基本的には四捨五入だが，丸める桁が 5 である場合は最も近い偶数になるように」丸める[24]．こうして仮数部を読めるだけ読んだ後，もし `e` または `E` があれば，その後に指数部が続くものと認識して，指数部を *scan_int* で読み取る．こうして仮数部も指数部も両方読み込んだ後は，*scan_float* はこれを glue として pack し，それを返す．

また，「浮動小数点数を読み取る」とはいっても，文字列で書かれた `-1.701E5` だけを「読み取る」のではあまり便利ではないので，既存の skip レジスタや，count レジスタや dimen レジスタも指定できるようにしてある．skip レジスタの場合は，ある浮動小数点数を表現していない可能性も考えられるので，そのチェックを行うようにしてある．一方，count レジスタや dimen レジスタはそのようなことについてはチェックす

10000 とかを使った方が，メモリも計算の手間も減ることになる．しかし，glue に格納するのが 21 桁であるから，1000 にするのもそんなに悪くないように思う．

[22] このポリシーは 5.6 節で変更される．

[23] 起動時に巨大な 4 byte 整数型の配列を確保して，あとは全部その枠内で行われる．配列確保と解放はそれぞれ $\langle pointer \rangle :=$ *get_node*($\langle length \rangle$) と *free_node*($\langle pointer \rangle, \langle length \rangle$) で行われる．ここで $\langle pointer \rangle$ は先頭要素の index であり，$\langle length \rangle$ は確保された配列の要素数である．

[24] この丸め方は内部演算で共通である．全部の下位の桁を見てるわけでは無いので，丸め誤差は最大で 0.6 ulp ぐらいになるはずである．

る必要はない．なお，dimension を直に，例えば `1.2mm` とか指定された場合は，最初に出会う文字が 1 であるから，浮動小数点数が文字列表記されているものと見なされるので，意図した結果は絶対に得られないことに注意．

出力部で問題になるのは「どの書式で出力するか」である．ご存知のように Fortran では `FORMAT` 文が，C 言語では `printf` 文の第一引数が出力書式の設定を担っており，何桁出力するだとか，指数形式にするのかそうでないのかとか，いくつかの設定項目が用意されている．今回ではそういう本格的な出力機能を実装することはせずに，primitive としては以下に絞ることにした：

1. `\fpfrac`：仮数部を $\pm a.bcd\cdots$ の形で出力する．最後に 0 が続かない範囲内でできるだけ多くの桁を出力する．正数の場合は符号は出力されない．
2. `\fpexpr`：指数部をそのまま出力する．

浮動小数点数全体を一度に出力する primitive を準備しなかったのは，「そういうのはマクロ機能があれば用が足りる」ことである．また，TEX の数式組版能力を生かさないのは勿体ないことではないか，とも考えたからである．例えば -1234567890123456.4 を `-1.2345678901234564E15` と出力するよりかは，$-1.2345678901234564\times10^{15}$ などと表示させる方が見栄えが良い．桁数指定については（dimension の表示とかにもそのような機能は無いので）見送ることにした．

count レジスタや dimen レジスタから浮動小数点数の変換が実装されたのだから，逆の，即ち浮動小数点数から整数や dimension への変換も実装されるべきである．しかし，dimension の TEX での内部表現を考えると，2 つの処理はあまり違わないようにできる．dimension への変換は浮動小数点数を sp 単位に（つまり，65536 をかけて）したものの整数部をとれば終わってしまう．丸め方については，0 に近い方に丸めるとしたが，dimension への変換は一連の計算の最後で行うことと決めておけば．あまり不都合は起こらないように感じる．

5.4 虫取り

以上の機能を実装した所で（実際は実装と半分平行した感じで）debug を行った．もはや `TRIP` test 等には頼ることはできず，全部自分でなんとかしなければならない．元から typo が多い方なので bug は多量に及び，かなりの時間と気力を debug にとられてしまった．debug は主に加算や平方根を求める Newton 法，e の Taylor 展開による計算などの繰り返しで行った．演算アルゴリズムの部分にも，タイプミスや僕の詰めの甘さ（加減算での符号の処理など）が見られたが，厄介だったのは TEX 自身のメモリ管理による動的確保/解放の処理であった．

先の脚注に書いたように，TEX でのメモリ確保は最初に巨大な配列が確保された後はすべて TEX 内部の *get_node* 関数などによって行われるのであった．動的確保で避けて通れない問題として，初期化忘れと解放忘れ，それに 2 重の解放がある．前者については対処方法は簡単なのだが，後者 2 つが曲者だった．浮動小数点数を内部で表現する配列は自分で作ったことがよくわかっているから，それについては問題ない．

一方，実は glue の格納についても動的確保/解放が使われている．glue は width, stretch, shrink の 3 つの部分があると書いたが，内部ではさらに *glue_ref_count* という量が各 glue について設けられている．これは名前の通り「何個の変数がこの glue を指しているか」というのを示す値であって，glue を解放するように *delete_glue_ref* procedure で指令しても，*glue_ref_count* が 0 でない限り[25]実際には解放されないようになっている．そして，skip レジスタに実際に格納されているのは実際の glue の値へのポインタとなっている．

動作確認をしたところ，`\fpadd` などである skip レジスタの値が演算結果によって書き換えられるような

[25] 本来は null 値基準なのだが，そこまで細かい所はどうでもいい．

場合に，書き換え元の glue は（他にそれを参照している変数が無い限り）解放されるべきであるにも関わらず，なぜか解放されないまま，いわば次々と過去の遺物だけが増えていってしまうという事態が起こった[26]．1 回あたりでは些細な量としかならないが，しかし bug は bug である．これを除去するために，解放指令を 1 回多く入れたら，今度は無限ループやら segmentation fault が出て強制終了するという事態が発生した．

結局，*scan_float* procedure の内部で既存の glue を取り出す所で reference count が 1 増加しているのではないかという結論に達し，*glue_ref_count* だけを 1 減少させる処理をそこに加えた所，うまく動くようになった．

5.5 初等関数

次に，初等関数の実装である．基本的には，関係式を用いて引数を 0 近傍のある値にまで持っていって，そこから先を Taylor 展開などを用いて計算させ，関係式を逆に用いて元に戻すというものである．平山 [11] での実装と奥村 [12] を大いに参考にした．

ここで，関係式の適用と逆向きの適用により誤差が入り込む余地があるので，そこをなんとかするために内部の四則演算を任意精度でできるように実装したのである．本節の関数の計算では．n 桁の引数が与えられた時に，内部では $n+15$ 桁で計算を行っている．この分量は「まあこれぐらい桁を増やしておけばいいか」という勝手な推測で選んだものであって，適切かどうかは知らない．

■平方根 (*fp_sqrt*)

a の平方根 $\sqrt{a}$ の計算は，方程式 $1-a/x^2=0$ から導かれる Newton 法 $x_{n+1}:=(3-ax_n^2)x_n/2$ を用いることにした．この Newton 法では $1/\sqrt{a}$ が 2 次収束で計算され，収束後 x_∞ に a をかければ，$\sqrt{a}$ が得られる．a は $[0.1,10)$ の範囲内に入るように変換して計算することとして，初期値 x_0 は，a の上位桁をいくつかとったものの平方根を奥村 [12] に載っているアルゴリズムを使用して計算したものとした．

なお，この関数の debug 中に，ずっと前に書いた「列の長さの縮小」の丸め処理に bug が見つかった．原因は **while** $i<n+1$ **do** ... のループで，i も n も更新せずに回していたことであった．このように，初等関数の debug 中に，既に組んだ四則演算等の bug が発見されることが結構あったが，以降はいちいちそれを書かないことにする．

■指数関数 (*fp_exponent*)

指数関数は Taylor 展開 $\exp x=\sum_{i=0}^{\infty}x^i/i!$ の有限項の部分和で計算する．収束を速めるためと，交代級数の計算に桁落ちの恐れがあることから，Taylor 級数の計算は $x\in[0,1]$ の範囲内だけで行い，一般は $\exp 2x=(\exp x)^2$, $\exp(-x)=1/\exp x$ でそれに帰着させている．

■三角関数周辺 (*fp_tri_hyp*, *fp_ceil_floor*)

ご存知のように，三角関数と双曲線関数の Taylor 展開は符号が異なるだけである．さらに，$\tan x=\sin x/\cos x$ であり，sin の展開は奇数次のみが，cos の展開には偶数次のみが現れるので，$\sin x$ と $\cos x$ を（あるいは，$\sinh x$ と $\cosh x$ を）同時に計算するのも悪くは無い．$\cos x$, $\cosh x$ の Taylor 展開の計算では，桁落ちを防ぐために 2 次の項から先の和を計算して，すべての計算の後に 1 を足すという方法をとった．

いつものように収束を速めるため，三角関数では引数を $[0,\pi/4]$ に，双曲線関数では引数を $[0,1]$ に帰着させる．三角関数では，引数 x に対して $a:=x-2\pi\lfloor x/2\pi\rfloor$ として，$a/4$ に対して Taylor 展開の計算を行い，

$$\sin 2x=2\sin x(\cos x-1)+2\sin x,\qquad \cos 2x-1=-2\sin^2 x$$

を 2 回用いて答えを計算する．

---

[26] `\tracingstats` に例えば 3 ぐらいの値を指定しておけば，TEX でのメモリ消費量がページ出力時に表示される用になる．

双曲線関数では，

$$\sinh 2x = 2\sinh x(\cosh x - 1) + 2\sinh x, \qquad \cosh 2x - 1 = 2\sinh^2 x$$

を同様に用いる．

また，前段落の a の式から分かるように，$\lfloor\cdot\rfloor$ の計算が必要となる．これと同じようなコードで $\lceil\cdot\rceil$ も計算できるので，ついでに両者を行う *fp_ceil_floor* procedure もここで書いた．

■対数関数周辺 (*fp_log_inner*, *fp_log*, *fp_arc_hyp*)

対数関数は，奥村 [12] によれば

$$\frac{1}{2}\log\frac{1+u}{1-u} = u + \frac{u^3}{3} + \frac{u^5}{5} + \frac{u^7}{7} + \cdots$$

を計算する方が速いとあるが，明らかに分かるようにこの式は arctanh の Taylor 展開式そのものである．$x = (1+u)/(1-u)$ を解くと $u = (x-1)/(x+1)$ となり，これは $x > 0$ では単調増加で $x = 1$ のときに $u = 0$ となる．当然収束を速めるためには $|u|$ が小さい方がよいので，x を 1 の近傍の範囲に帰着させて計算をおこなうようにする．帰着方法は単純で，まず適切な 10^n を掛けて $x \in [1, 10)$ とさせ，以下のようにさらに変換する：

$$x \in [1, 1.5) \Longrightarrow (\text{そのまま}), \qquad x \in [1.5, 3) \Longrightarrow x \leftarrow x/2,$$
$$x \in [3, 6) \Longrightarrow x \leftarrow x/4, \qquad x \in [6, 10) \Longrightarrow x \leftarrow x/8.$$

こうすると，$|u| \le 0.2$ となり，収束速度はかなり速いものと予想される．逆変換に必要な $\log 10$, $\log 2$, $\log 4$, $\log 8$ については，$\log 10$ と $\log 64$ を定数として内部に持っておくこととした．

対数関数 $\log x$ を実装してしまえば，逆双曲線関数は以下の式で計算可能である：

$$\operatorname{arcsinh} x = \log(x + \sqrt{x^2+1}),$$
$$\operatorname{arccosh} x = \log(x + \sqrt{x^2-1}), \qquad x \ge 1$$
$$\operatorname{arctanh} x = \frac{1}{2}\log\frac{1+x}{1-x}, \qquad |x| < 1$$

この中で arctanh の実装は log のそれの内部に入り込む形になっているので，引数が小さい場合には $\log x$ の内部計算部（即ち，本節の最初に示した Taylor 展開の計算）にそのまま引き渡してしまうようにした．

しかし，この式では $x \simeq 0$ であるとき，$x + \sqrt{x^2+1} \simeq 1$ となるので桁落ちが発生する．そこで，x の有効桁数が $3n$ であるとき，$|x| < 10^{-n-5}$ であれば，次の Taylor 展開式を用いて arcsinh を計算することにした：

$$\operatorname{arcsinh} x = x - \frac{x^3}{6} + \frac{3x^5}{40} - \cdots.$$

■逆三角関数 (*fp_arc_tri*)

逆三角関数は多価だが，常識的に考えれば $\arcsin x$, $\arctan x \in [-\pi/2, \pi/2]$, $\arccos x \in [0, \pi]$ とするのが適当だろう．$\arcsin x = \arctan(x/\sqrt{1-x^2})$ であるから，$\arctan x$ だけ計算すればよいが，これはいつものように級数展開

$$\arctan x = \sum_n \frac{(-1)^n x^{2n+1}}{2n+1}, \qquad |x| \le 1$$

を用いる．$\arctan x = \pi/2 - \arctan x^{-1}$, $\arctan x = \pi/4 - \arctan((1-x)/(1+x))$ を用いれば，$|x| \le \tan\pi/8 = \sqrt{2} - 1 = 0.414\cdots$ までできて，収束は保証される．

■べき乗 (*fp_pow_int*)

一般には a^b の計算は $\exp(b\log a)$ とすればよいが，それでは $(-2)^5$ などの $a \in [-\infty, 0)$, $b \in \mathbb{Z} \setminus \mathbb{N}$ のときが計算できない．このため，指数部分が整数のときの累乗を計算する関数 *fp_pow_int* を作った．方針は奥

村 [12] に従い，例えば $b = 1701$ のときは，$1701 = 1024 + 512 + 128 + 32 + 4 + 1$ であるから，$x_0 := a$, $x_{n+1} := x_n^2$ としたとき，$a^b = x_0 x_2 x_5 x_7 x_9 x_{10}$ として求めれば良い．

気になるのは 0^0 の扱いだが，本来は NaN とするのが正当な扱いであろう．しかし，ここでは手抜きし，*fp_pow* でも *fp_pow_int* でも 1 とした．後から考えたり見つけたりしたこじつけを 3 つばかり：

- $a^b = \exp(b \log a)$ を無理やり $a = b = 0$ のときにも適用して，“$0^0 = \exp(0 \log 0) = \exp(0 \cdot -\infty) = \exp 0 = 1$”．
- $m^n = \#\{f \colon n \longrightarrow m\}$ である[27]から，$0^0 = \#\{f \colon 0 \longrightarrow 0\} = \#\{f \colon \emptyset \longrightarrow \emptyset\} = 1$ となる．
- 二項定理 $(1+m)^n = \sum_{i=0}^{n} \binom{n}{i} m^i$ を $m = n = 0$ のとき成立させるために必要．

5.6 高速化

一応以上に挙げた関数の実装は行ったが，これらの関数は演算の途中経過が求まる度に動的確保/解放を行っており，arcsin を求めるだけで 800 回以上の動的確保を行っていた．これは速度的に非常に不利であるので，できる限り動的確保を行わずにすむように書き直す必要があった．

方針としては次の通りである：各演算では，当然引数と結果を格納するための領域と，一時領域が必要になるが，引数の有効桁数が（つまり，引数を表現するのに必要なメモリ領域の量が）わかっていれば，一時領域の大きさはそこから計算することができる．元々の呼び出し元は *do_float_operation* という 1 つの procedure である[28]．よって，必要な一時領域の上限がわかるので，演算を実際に呼び出す前に適当な所で確保しておけば良い．

演算の種類によって一時領域の量は次の様になった：

演算	**procedure** 名	引数	返り値	一時領域
$+, -$	*fp_add_or_sub*	n, n	n	$n+3$
$\cdot$	*fp_mul*	m, n	m	$m+n+3$
$/$	*fp_div*	n, n	n	$5n+15$
$/$ (short)	*fp_short_div*	n	n	$2n+10$
a^2	*fp_square*	n	n	$2n+3$
$\lfloor\cdot\rfloor, \lceil\cdot\rceil$	*fp_ceil_floor*	n	n	$n+3$
$\sqrt{}$	*fp_sqrt*	n	n	$4n+32$
exp	*fp_exponent*	n	n	$4n+32$
sin, cosh etc.	*fp_tri_hyp*	n	$n+5$	$5n+40$
log（内部）	*fp_log_inner*	n	n	$4n+12$
log	*fp_log*	n	n	$6n+48$
arcsin etc.	*fp_arc_tri*	n	$n+5$	$8n+120$
arcsinh etc.	*fp_arc_hyp*	n	$n+5$	$8n+120$
a^b if $b \in \mathbb{Z}$	*fp_pow_int*	n	n	$3n+24$

一時領域の欄の単位は 4 byte であり，引数と返り値はそれぞれを表す浮動小数点数の「長さ」を表している[29]．但し，下半分の演算については，上半分の四則演算等の一時領域は除いて計算している．

ここで，初等関数の演算のために，π, $\log 10$, $\log 64$ を定数としてもっておくことは前節に書いた．この定数の初期化と一時領域の確保は同時に行っても良いだろう，しかし，一時領域と定数たちだけでメモリの $80 + 184 + 3 \cdot 23 = 333$ 要素が使われることになり，常時確保しておくのは気が引ける[30]．そこで，この「初

[27] 集合論的定義に従い，$m = \{0, 1, \ldots, m-1\}$ として考えている．

[28] ここで TEX のコマンドで渡された演算指令が最終的に実行され，結果を表現する glue が得られる．

[29] 動作切り替えのブール値や整数の引数はすべて省略している．

[30] 現在のように贅沢にメモリが使える環境ではごく僅かの量である．なお，内部演算は最大 $3(8+5) = 39$ 桁で行われる

期化」を行う \fpinit コマンドと，逆に一時領域や定数たちを解放する \fpdest コマンドを作り，浮動小数点演算を使用する場面の前後にこれらのコマンドを実行させるようにした．なお，実装の簡略化のため，二重確保や二重解放のチェックは全く行っておらず，一時領域が確保されてない状態で演算を行おうとしたときのエラー処理も全く行っていない．確保されてない状態で演算を行えば，最悪の場合はセグメンテーション違反で落ちることになるので気をつけること．

6 形式化

浮動小数点演算を実装したところで，説明文をつけたり install script をつけたりする段階へと入ることにした．ここら辺を行う直前になって，角藤さんの peTEX が登場した^^;.

一番苦労したのは，「ε-pTEX のためにどのファイルを修正したのか分からない」ということであった．ptexextra.c の変更は，コマンドラインオプション -etex をつけるためにやったことを覚えていたが，結構それ以外にも変更したファイルがあったものだった．なお，僕は Make の使い方が分からないので，コンパイルとインストールは基本的には（自分の環境用の）bash script として準備して，何をやっているかは別途テキストファイルに書いておいた．

pTEX から ε-pTEX への修正部分を分類してみると，その性格の違いからいくつものファイルに分けた方が気分がいいと思ったので，そのようにした．実質的に何もしてない ptex-orig.diff を削除したり，ptex-hack-1.diff（次章参照）を入れたり取ったりとかで整理した結果，本プロジェクトの最終結果としては以下のようになった：[31].

■ファイル構成

```
-------------------------------------------------------------------------
  README.txt          このファイル.
  Changelog           更新履歴
  HOWTOINST.txt       インストール方法について
  doc/                浮動小数点演算についての簡易説明書
  ks2/                「計算数学 II」での詳細な作業記録
  build               build するための（自分用の）bash script
  install             install するための（自分用の）bash script
  trip                TRIP, e-TRIP test を実行させる bash script
  eptex.src           etex.src を e-pTeX 用に改変したもの．ほとんど同じ
  eptexdefs.lib       etexdefs.lib を e-pTeX 用に改変したもの．ほとんど同じ
  ep1.diff            \
  pconvert.diff       |- e-TeX 機能のマージに使用する patch 群
  etex-sysdep.ch      |
  etex.diff           /
  fp.ch               浮動小数点演算の実装部
  ptex756.diff        pTeX を Web2C-7.5.6 に対応させる patch
  uptex.diff          \_ upTeX とマージしようとするときに使用する patch 群
  up1.diff            /
```

ので，四則演算に必要な一時領域は 80 要素あれば充分となるのだが，80113.22 より前の版では，長さ 18 で四則演算されてしまうバグがあった．

[31] 以下は README.txt からの引用である．

7 細かい機能追加

7.1 TEX--XETEXの実装

角藤さんによる peTEX に触発されて[32]，ε-pTEX でも TEX--XETEX機能の実装を試みることにした．浮動小数点演算の部分は ε-TEX 拡張とは完全に切り離せるので，`fp.diff` を組み込まない状態でも作業を行える．

作業方法は「ε-TEX のマージ」で行ったものとほとんど同じである．オリジナルの `etex.ch` と前に作った対 pTEX 用の `etex.ch` を見比べ，TEX--XETEX機能で後者に抜けているものがあれば加えたりした．これにより，新たな `etex.diff` は TEX から pTEX への変更に伴う部分だけとなり，分量が減った．

`\beginR` と `\endR` ではさまれた区間は右から左に文字が組まれる区間であるが，そのまま `etex.ch` の方法を使うと日本語が通らずに，強制終了したり dvi が読めなくなったりする．これの原因を次のように考えた：

> right-to-left を実現させる部分に，「全ての」構成要素を逆順にするという *reverse* function がある．例えば LtoR の状況で「`...abcdefABghCDEFij...`」：小文字は欧文文字，大文字は 2 つで和文文字を表すとなっていたものが，この procedure により `...jiFEDChgBAfedcba...` と反転されてしまい，それによってエラーが発生する．従って，和文文字のところは反転せずに `...jiEFCDhgABfedcba...` という結果を得るようにすれば，エラーは起きないのではないか？

pTEX のソースで `if font_dir[f]<>dir_default then` とかいうフレーズが何回かあったので，これを 2 byte 文字と 1 byte 文字の判定と考えることで上の処理を実装したところ，一応はうまくいくようになった．しかし，組版規則はどうか，というと当然のことながら「怪しい」．ただ単に全部左右が入れ替えられるだけであるからだ．

7.2 副産物

TEX--XETEX関連の debug を行っていたときに，和欧文混在の文章を right-to-left にしたら上下位置がおかしくなることに気づいた．「全てを逆順に」がここでも悪さをしており，pTEX により和欧文間に挿入される baseline 補正の指標 *disp_node* までも逆順にされてしまうからだった．

pTEX では，和文と欧文の境目に *disp_node* が挿入され，この node 以降の文字の baseline 補正がそこで示されるようになっている．したがって，和文→欧文と切り替わるときの node では `\ybaselineshift` とかが，逆に切り替わるときの node では 0 が対応している．よって，一時変数を使って，*reverse* function で反転させるときに，*disp_node* の値も入れ替えてしまえばいいのではないか，と推測される．それで実際大抵の場合にはうまくいったように思えた[33]．

こうしてなんとか解決した直後に，たまたま TEX フォーラム[34]を覗いたら，奥村さんによる「pTEX への提案もどき」と称した次の投稿 [15] があった（以下引用）：

---

[32] 対抗して，と言った方が実状にあっているが．

[33] 1/30 追記：実際には，こんな簡単な処理ではダメだった．段落全体が RtoL の状態のとき，どうやら，各行は displacement node で始まっており，その後に TEX--XETEX関連の node とかがくるようで，こういう場合が曲者だったりする．global 変数 *revdisp* を作り，その変数に「現在の displacement の値」を代入するようにしておくことで，何とか対処療法的に片付けて，**「大抵の場合にはうまく動くのではないか」という段階にはなったけれども，pTEX の内部動作を良く理解しているわけではないので，非常に怪しい．使うときには注意してください．**

[34] `http://oku.edu.mie-u.ac.jp/tex/`

その 1: \ybaselineshift の仕様変更

和文と欧文のベースラインを調節するための \ybaselineshift を使っても数式部
分のベースラインが変わらない。数式も欧文なみと見て移動したい。

例：

\ybaselineshift=0.12zw
第 5 章第$n=5$節

その 2: 段落の頭での引用符の挙動の改善

デフォルトの pTeX では引用符で始まる段落の頭が \parindent+0.5zw 下がりになる。
jsarticle，jsbook では \everypar を使ったハックで補正しているが，pTeX 本体に
段落頭の JFM グルーをなくすスイッチを付けられないか。

（後略）

この「その 1」というのは，僕も散々悩まされてきた問題であった．古くは高校の部活の部誌（縦書き）の組版で，最近では数学科オリパンフ [14] の組版でも遭遇し，非常に腹が立った．「その 2」は現状の jsclasses でのハックで十分だと思い，またこちらは難しそうだった．

「その 1」のパッチを作るのは，*disp_node* が挿入される位置を調べ，そこでいかほどの値がセットされるかを調べればいいので，さほど困難な作業ではなかった．互換性のために従来の pTEX の動作も残し，スイッチで切り替えるようにしたが，スイッチの定義は \TeXXeTstate（TEX--XETEX が on ならば 1，off ならば 0）のそれが非常に参考になった．

最初は ε-pTEX 内部に本パッチを取り込んでいたが，（後に述べるように）角藤さんが ε-pTEX を W32TEX に導入するときに外されたと聞いたので，こちらでも外しておくことにした．

7.3 さらなる虫取り

本稿では，e-TRIP test をしたときの ε-TEX との差異をもっと少なくすることを行う．まず，通常の TRIP test を行ったときに，そのままでは http://oku.edu.mie-u.ac.jp/%7Eokumura/texfaq/qa/50546.html と同様の症状，すなわち，direction がいくつか余計に残ってしまうことから対処を始めた．これの原因は pTEX で box の中身を dump したときに，yoko direction の表示部である．box が yoko でも tate でも DtoU 方向[35]でもないときは yoko などが表示されず，ただ，direction と表示されてしまうのだった．，が出てくる場所は 1 箇所だったので，ここだけ direction 表示のコードを別にすることで解決．

次に，\lastnodetype と \currentiflevel の調整に入った．ε-TEX でコマンドがどう処理されているかを調べると，各 node にはソース中で型を表す自然数が割り振られており，その値に 1 を足すとかの若干の補正をして出力するだけだった．そうすると話は速く，TEX → pTEX の拡張で内部パラメータがどれだけ変更されているかを調べ，それに合うように両コマンドを hack するだけであった．

これらのコマンドの正しい動作は，ε-TEX のマニュアル [4] に与えられている．pTEX で追加された node や条件判断文には，既存の番号より大きい値を与えるようにした．具体的には，下の表のようになった．上が \currentiflevel，下が \lastnodetype の動作で，番号が太字なのは pTEX 特有のものである[36]．なお，これに合わせるように etexdefs.lib を更新した eptexdefs.lib も作ってある：

[35] DtoU 方向ってなんだろうね．

[36] 当然ながら，\iffp は本プロジェクトの浮動小数点演算の実装に特有である．

1:	\if	8:	\ifmmode	15:	\iftrue	**22:**	\ifydir
2:	\ifcat	9:	\ifinner	16:	\iffalse	**23:**	\ifmdir
3:	\ifnum	10:	\ifvoid	17:	\ifcase	**24:**	\iftbox
4:	\ifdim	11:	\ifhbox	18:	\ifdefined	**25:**	\ifybox
5:	\ifodd	12:	\ifvbox	19:	\ifcsname	**26:**	\iffp
6:	\ifvmode	13:	\ifx	20:	\iffontchar		
7:	\ifhmode	14:	\ifeof	**21:**	\iftdir		

-1:	none (empty list)	6:	adjust node	13:	penalty node
0:	char node	7:	ligature node	14:	unset node
1:	hlist node	8:	disc node	15:	math mode nodes
2:	vlist node	9:	whatsit node	**16:**	direction node
3:	rule node	10:	math node	**17:**	displacement node
4:	ins node	11:	glue node		
5:	mark node	12:	kern node		

これでだいぶ TRIP, e-TRIP test の動作はうまくいくようになった．状況を述べると，以下のようになる．

- **Compatibility mode での TRIP test の，pTEX との差異**
  最後の方の 1440 strings of total length 25223 等が異なるだけ．ソース中の文字列が多くなっていたりすることなどによるものだろう．
- **TRIP test の，ε-TEX との差異（両方とも Compatibility mode）**
  ↑を除けば，pTEX での TRIP test でも出ている差異なので，問題はないと考えられる．
- **TRIP test の，ε-TEX との差異（両方とも Extended mode）**
  「両方とも Compatibility mode」のときの差異と同じような雰囲気．
- **e-TRIP test の，ε-TEX との差異**
  - \lastnodetype のチェックで 3 箇所失敗する．pTEX の仕様に関わるのではないか？
  - Memory usage が多かったり，direction が表示されたりとか，明らかに pTEX 拡張によると考えられるもの．

8 他環境でのコンパイル

今まで TEX Live 2007 上で開発を行ってきたが，TEX Live は非常に巨大であり，これを気楽にダウンロード，コンパイルということはやはりし辛いものである．一方，これ以外に主に使われている（と信じる）TEX distribution としては，以下がある．

- **teTEX 3.0**：Thomas Esser さんが作っていた Unix 系 OS 用のもの．適度な大きさで便利だったのだが，残念ながら 3.0 をもって開発中止に．ソースは CTAN/systems/unix/teTeX/3.0/distrib/からダウンロードできる．おそらく大半の linux distribution がこれをベースにした TEX 環境を導入している．
- **ptetex3**：土村さんによる teTEX3.0 + 日本語環境を簡単に整えるための patch 集．公式ページは http://www.nn.iij4u.or.jp/~tutimura/tex/ptetex.html.
- **upTEX**：ttk さんによる，pTEX の内部コードを Unicode にする実験．ptetex3 内の Unicode サポート活動 (ptexenc) と少なからず関係．下の W32TEX でも使えるが，本来は ptetex3 へのパッチ．公式ページは http://homepage3.nifty.com/ttk/comp/tex/uptex.html.
- **W32TEX**：角藤さんの作られている Windows 用 TEX distribution の通称．おそらく Windows 上で TEX を利用している人ならほとんどが入れてるはず[37]．

[37] cygwin でコンパイルしたものを使って，W32TEX を使ってない人という人が若干いる模様．

公式ページは http://www.fsci.fuk.kindai.ac.jp/kakuto/win32-ptex/web2c75.html.

ここから一番近いミラーは数理の大島研．インストールには TA の阿部さんの「TEX インストーラ」が非常に便利である．

とりあえず上 2 つはなんとかサポートするようにするべきだろう．W32TEX については，TA の阿部さんからの要望である．

8.1 W32TEX

W32TEX のソースは前節の大島研の FTP などからダウンロードすることができる．「どうやら Visual C++ がコンパイルに必要らしい」ということはわかっていたので，Microsoft のサイトから Microsoft Visual C++ 2005 Express Edition と Platform SDK をインストールし，また最新の Cygwin もインストールをして，環境を整えた[38]．

しかし，ここから先が大変だった．pTEX のソースは 756/texk/web2c/ptexdir/にある．どうやら土村さんによる ptexenc 拡張を取り入れている様子であった．しかし，なぜかその直上のディレクトリの make がうまくいかず，tangle とか web2c とかは元々のバイナリのものを使わなければならなかった．これらでいろいろ悪戦苦闘したあげく，結局 ptexdir のソースをベースにすることは諦めることにした．

その後，いろいろと慣れない Windows 環境のことを調べ，なんとかコンパイルできるところまでこぎつけた．しかし，阿部さんから作った binary が動作不良という報告を受け，結構悩むことになった．おそらく DLL まわりの事情だったと思うのだが，「W32TEX に取り込む」という角藤さんからの報告が来たので，こちらでの Windows バイナリの開発はやらなくても良いような状況になってしまった．

8.2 teTEX 3, ptetex3 (ptexenc), upTEX

これらはみな Web2C-7.5.4 環境がオリジナルである．従って，ptex756.diff を使わないことが必要になるが，前者 2 つについては，それぐらいだけであまり注意は必要なかった．ptetex3 独自拡張の ptexenc についても，コンパイル時にライブラリを追加するぐらいであった．upTEX についても，WEB ソースについては同じであったが，できるプログラム名を euptex などとしてみたため，C header などの修正が少し面倒になった．

後者の ptexenc, upTEX で心配であることは，日本語が TEX--XETEX で通るか，ということであった．僕が先に行った TEX--XETEX への和文対応では，漢字などの 2 byte 文字が，内部では 2 つの node で表現されていることを前提としたものであった．そのため「Unicode 拡張によって 1 文字 3 node とかになっていないだろうか」と心配した．WEB change file をざっとざっと見てみた所，どうやらそのような事態は起こっていないように思えた．どうやら動いてはいるようである．

このときついでに，fp.diff を change file の形式に書き直したりもした．そこで *fp_sub_absolute_value* procedure 内で $\infty - \infty \neq \mathrm{NaN}$ という bug を発見して，修正することができた．

[38] Platform SDK が必要であることを知るまでに数時間を無駄にした．

9　今後の課題と，それにまつわる雑感

今まで書いたような作業をしてきて，最初に気づいたことは，**自分の英語力の無さ**である．英語の数学書とかは（英文については）たいした苦もなく読めるのだが，一方でいざ自分で書こうとすると表現が全く思い浮かばない．WEB ソースを書き換えるにあたっては，とりあえずの英語で間に合わせたけれども，意味の余り取れないような文章になっているであろうことは確実である．院試までになんとかしたいものだ^^;

内容面について．浮動小数点の実装をしたはいいが，Pascal の，それもごく基本的な仕様に沿っただけの実装であり，速度的にも不利であると思われる．Beebe [13] にあるように，C 製の何か信頼性のおける高速多倍長数値計算ライブラリを利用するというのが手っ取り早くまた安全な方策であろう．しかし，僕は自分で浮動小数点演算を実装したことがなかったので，一回は自分でコードを書いてみたかった．

また，ε-TEX で長さ量とかには `\dimexpr` などがあり式の表現が使えるのに，浮動小数点演算ではまだその機能がないとか，やっぱり出力書式指定の機能がないというのも気になる話である．後者はマクロを頑張っていじればなんとかできそうな気もする．

最後に，避けて通れない問題であるが，bug の問題がある．特に浮動小数点演算の方は自分で書き起こしたものであるから，どんな bug が潜んでいるかは分からない．だいぶ bug は取れたような気もするが，それは現時点での思い込みであって，やはり心配である．

ひょっとすると，pTEX の寿命はもはやあまり長くないのではないかと心配している．最初に紹介した TEX の拡張たちや，pdfTEX を拡張してスクリプト言語 Lua を組み込んだ LuaTEX，また XETEX なんかの存在を聞いてると，純粋な TEX に日本語を対応させただけの pTEX の存在意義は過去の遺産の継承ぐらいしかなくなり，主流は世界的にもこれらの拡張のどれか or その発展形 ——LuaXETEX とでも呼べばいいのだろうか？ ——になっていきそうな予感がする．ちょうど，日本電気の PC-9800 シリーズと所謂「DOS/V パソコン」の関係のように．

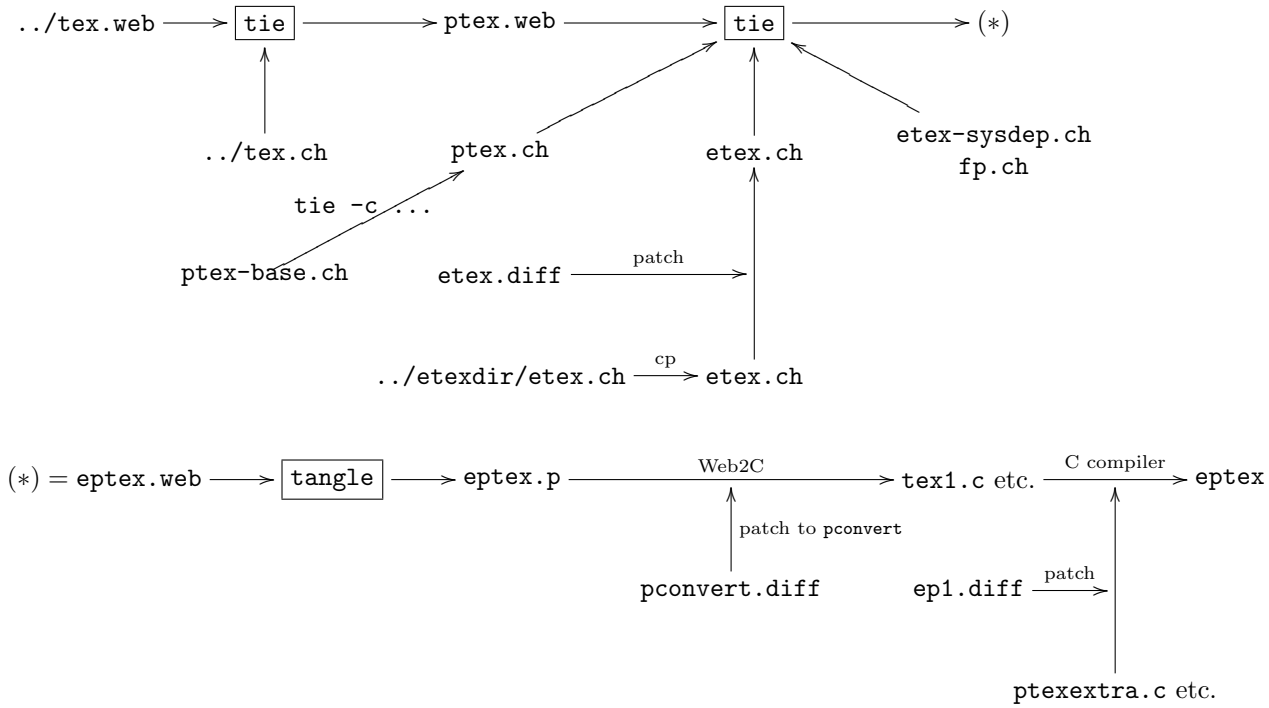
しかし，せっかくこの計算数学 II で「pTEX に対して，ε-TEX の機能のマージと浮動小数点演算の実装を行う」ということが大っぴらにできたのだから，このまま終わるのは非常に虚しいと思っている．こんなことを言うと「何を言っているんだ」と言われそうだが，upTEX などと共に，pTEX の地位向上に寄与できたらいいな，と思っている．

以上，いろいろ書いてきたが，冬休み中やそれ以降に，この ε-pTEX まわりがとんでもないおおごとになってしまったことには驚きである．取り上げてもらえるのはうれしいことではあるが，ε-pTEX がどんどん自分の中に抱えきれないものになっているような気もしている．

10 付録

10.1 ε-pTEX のコンパイル方法のイメージ

eptex-xxxxx.xx.tar.bz2 内の HOWTOINST.txt とかを見れば分かる話ではあるが，図にしておくとなにかと便利である．



10.2 WEB の実例：ks1.web

このファイルがあるディレクトリ，もしくは公開場所にある ks1.web は，僕が計算数学 I のレポートに書いた Carmichael 数を探索するプログラム（ks1.f90, Fortran 90）を WEB に翻訳したものである．

ここでは，（本来の WEB では日本語が通らないので）松山さんの日本語版 WEB [9] を使わせてもらうことにした．この日本語版 WEB による jtangle, jweave コマンドがそれぞれ tangle, weave の代わりとなる．

僕は Pascal のコンパイラを所持していなかったので，Web2C を利用してコンパイルを行った．

```
[h7k doc]$ web2cdir=/home/h7k/ks2/texk/web2c/web2c
[h7k doc]$ cat $web2cdir/common.defines $web2cdir/texmf.defines ks1.p|$web2cdir/
web2c -t -hks1-a.h -cks1 |$web2cdir/fixwrites > ks1.c
[h7k doc]$ gcc -o ks1 ks1.c
```

jtangle, jweave コマンドでできた ks1.p, ks1.tex や，上のように C 言語に変換して得られた ks1.c, ks1.h も同じディレクトリに置いておいた．なお，ks1-a.h というのは Web2C が吐く C ソースをきちんとコンパイルできるようにするための最小限の設定である．

参考文献

[1] 松山 道夫,「TEX 読み物その 3　TEX の成立ちに関する諸々」.
http://www.matsuand.com/downloads/texread/texread3.pdf

[2] Donald E. Knuth. *The WEB System of Structured Documentation*. Stanford Computer Science Report CS980, 1983.
例えば http://www.ctan.org/tex-archive/systems/knuth/WEB/WEBman.tex とかにある.

[3] 奥村氏の TEX wiki の「Make」の項.
http://oku.edu.mie-u.ac.jp/~okumura/texwiki/?Make

[4] The NTS Team. *The ε-TEX manual*.
http://www.tug.org/texlive/Contents/live/texmf-dist/doc/etex/base/etex_man.pdf

[5] Donald E. Knuth. *A torture test for TEX*. Stanford Computer Science Report CS1027, 1984.
http://ftp.yz.yamagata-u.ac.jp/pub/CTAN/systems/knuth/tex/tripman.tex

[6] The NTS Team. *A torture test for ε-TEX*.
旧版 (Ver. 2) が以下にある．現在は Ver. 2.2 で，TEX Live 2007 とかには入っている.
http://tug.ctan.org/cgi-bin/getFile.py?fn=/systems/e-tex/v2/etrip/etripman.tex

[7] Donald E. Knuth. *The Art of Computer Programming*, Volume 2: Seminumerical Algorithms. Addison-Wesley, third edition 1998.

[8] 有澤 誠 編,『クヌース先生のプログラム論』，共立出版，1991

[9] 松山 道夫，日本語版 WEB（試供版）.
http://www.matsuand.com/downloads/jWEB/jWEB-0.0.7.tar.gz

[10] K. イェンゼン，N. ヴィルト 著，A. B. ミケル，L. F. マイナー 改訂，原田 賢一 訳,
『情報処理シリーズ 2　PASCAL　原書第 4 版』，培風館，1981

[11] 平山 弘，高精度計算プログラム MPPACK 1.0. http://phase.hpcc.jp/phase/mppack/

[12] 奥村 晴彦,『C 言語による最新アルゴリズム辞典』，技術評論社，1991

[13] Nelson H. F. Beebe. Extending TEX and METAFONT with Floating-Point Arithmetic. *TUGboat*, 28 (3), November 2007. http://www.math.utah.edu/~beebe/talks/2007/tug2007/tug2007.pdf

[14] 数学科 3 年オリエンテーション係,『2007 年度 数学科オリパンフ』，2007.9
（数理 266（学部生用控え室）に結構残部がある模様.）

[15] 奥村 晴彦,「pTeX への提案もどき」，2008.1.4.
http://oku.edu.mie-u.ac.jp/tex/mod/forum/discuss.php?d=29