# 論文形式の LaTeX 文書のスライド発表形式の LaTeX 文書への変換プログラム

日本 IBM(株)　東京基礎研究所，金沢工業大学　客員教授　寒川　光

2007 年 9 月 20 日

- 『LaTeX & PostScript スーパーユーザのテクニック』で，論文形式の LaTeX 文書を，Perl による簡単な変換プログラムによって，スライド発表形式の LaTeX 文書に変換する方法を紹介した[a].
- スライド発表用には LaTeX $2_\varepsilon$ の seminar クラスを使用し，変換には Perl プログラムに対する指示文（ディレクティブ）を使用する.
- 本資料は「数学ソフトウェアとフリードキュメント V」での講演資料として，このアプローチを発想した周辺事情を含めて紹介し，講義や勉強会などで数式を多用する方々にこの方法を利用していただく目的で用意した.

[a] 共立出版，2006 年 11 月. “http://www.kyoritsu-pub.co.jp/shinkan/shin0611_07.html”

**指示文の利用** Fortran や C 言語によるプログラムでは，原始（ソース）プログラムにコンパイラに対する指示文（言語規格上ではコメント行だがコンパイル時には特別な指定をできるステートメント）によって，2 通り，3 通りにコンパイルする方法が利用されている．この方法と同様に，LaTeX のソーステキストを原始プログラムと見立てて，シングルソースで論文用と発表用を 2 通りに変換されるようにした．この場合，変換後のテキストには直接手を加えない方針とする．両者に共通のソースを用いて，改訂はそれに対してだけ行えば両方に反映されるようにする．

**スクリプト言語プログラミング** 数値計算には Fortran，システムプログラミングには C 言語，Web サービスでは Java+XML というように，処理内容に適したプログラミング言語がある．文書の変換には Perl の正規表現を利用すると，意外に簡単に変換プログラムを実現できる．

**仕様の決定と改良** 実際に使用しながら，自分のプレゼンテーションに適した形に仕様と変換プログラムを改良してゆく（**日曜大工プログラミング**の楽しみ）．

```
      implicit double precision (a-h,o-z)
      character*30 charg
      call getarg(1,charg)           !  ./a.exe 20000
      read(charg,'(i10)') nmax       !  nmax =  20000
#ifdef TXT
      open(8,file='Fern_txt.dat')
#elif BIN
      open(9,file='Fern_bin.dat',form='unformatted')
#endif
      call srand48(68111)
      x=0.5; y=0.
       do i=1,nmax
        r=drand48()
         if(r.le.0.02d0) then
          xn=0.5d0;                          yn=0.27d0*y
         else if((r.gt.0.02d0) .and. (r.le.0.17d0)) then
          xn=-0.139d0*x+0.263d0*y+0.57d0;   yn= 0.246d0*x+0.224d0*y-0.036d0
         else if((r.gt.0.17d0) .and. (r.le.0.3d0)) then
          xn=0.17d0*x -0.215d0*y+0.408d0;   yn=0.222d0*x+0.176d0*y+0.0893d0
         else
          xn= 0.781d0*x+0.034d0*y+0.1075d0; yn=-0.032d0*x+0.739d0*y+0.27d0
         endif
        x=xn; y=yn
#ifdef TXT
        write(8,*) x,y
#elif BIN
        write(9) x,y
#else
        write(*,*) x,y
#endif
       enddo
#ifdef TXT
      close (8)
#elif BIN
      close (9)
#endif
      end
```

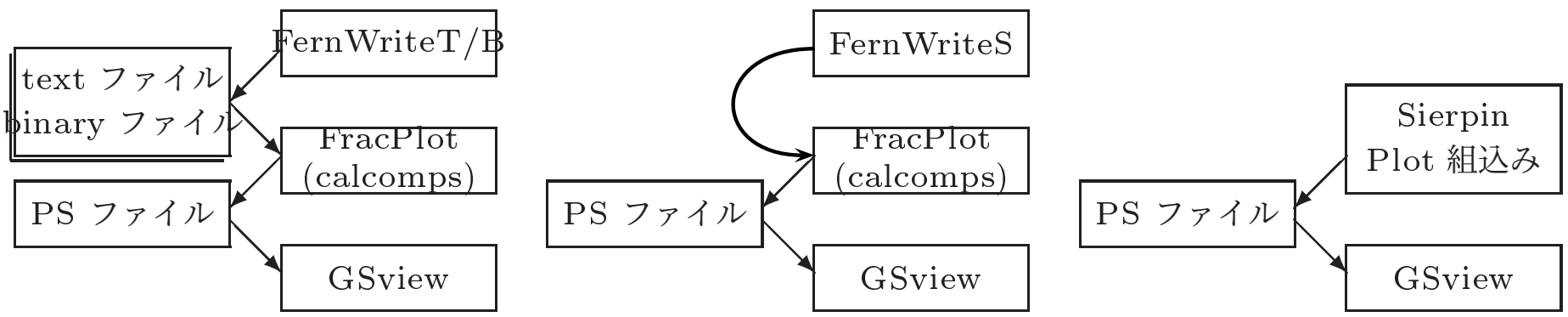


Figure 1: ファイル渡し（左），パイプ渡し (中)，プロットルーチンの埋め込み（右）

# seminarクラスの使い方

---

- クラス名を seminar としてスライドにしたいものを “\begin{slide}” と “\end{slide}” で囲う.
- 図 2 は，ドキュメントクラスに “seminar” を指定して,「\LaTeX の picture 環境での作図機能は次のように纏められる.」の 1 センテンスの後に enumerate 環境によって 4 項目を箇条書きした例である.
- ソーステキストのファイルを semitest.tex とすれば，“platex semitest” と “dvipsk -t a3 semitest” で semitest.ps を作成できる.
- オプションの “-t a3” は dvipsk に与える用紙サイズである.

```
\documentclass{seminar}
\pagestyle{empty}
\begin{document}
\begin{slide}
\LaTeX の picture 環境での作図機能は次のように纏められる.
\begin{enumerate}
\item 任意の太さの縦横の罫線を引く.
\item 限定された傾きの 10pt 以上の長さの斜め線と矢印を描く.
\item 任意の位置に非常に正確にフォント（図形）を配置する.
\item フォントをマトリックス構造に配置する（配置される場所は tabular 環境や
array 環境のように，置かれるものの大きさにぴったり揃えられる）.
\end{enumerate}
\end{slide}
\end{document}
```

LaTeX の picture 環境での作図機能は次のように纏められる.

1. 任意の太さの縦横の罫線を引く.
2. 限定された傾きの 10pt 以上の長さの斜め線と矢印を描く.
3. 任意の位置に非常に正確にフォント（図形）を配置する.
4. フォントをマトリックス構造に配置する（配置される場所は tabular 環境や array 環境のように，置かれるものの大きさにぴったり揃えられる）.

Figure 2: seminar クラスによるスライドの例

- “\slideframe{none}” を指定すれば外側の枠を消すことができる.

# 論文用テキストからスライド用テキストへの変換

---

- ソーステキストが **1 センテンス 1 レコード原則**で書かれていると，変換プログラムは例えば Perl なら “while(<F>){...}” のループ構造の中で処理しやすい.
- 入力ファイルの 1 レコード単位に文脈と指示行に従ってレコードを加工すればよいからである.

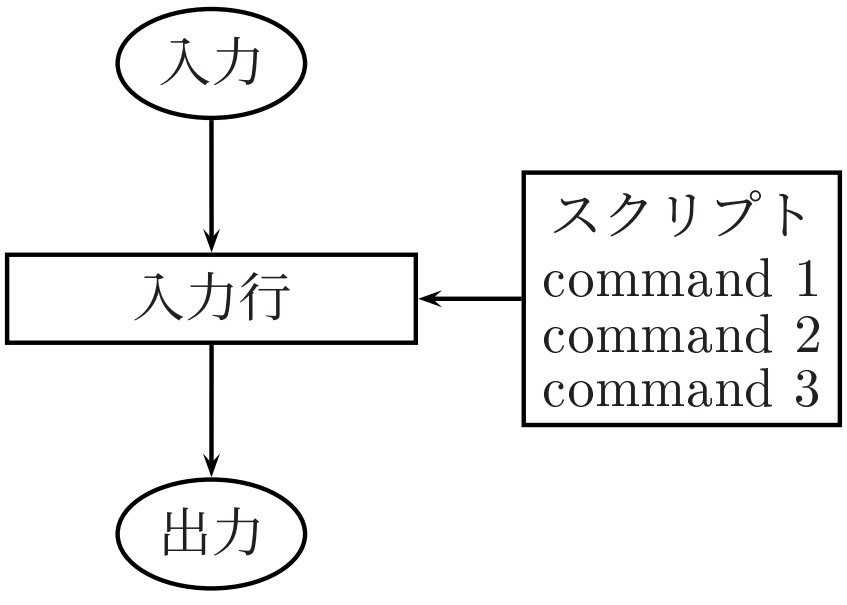


Figure 3: sed と awk の動作（スクリプト言語の起点）

- プログラム名は texslide.pl とした.
- hoge.tex ファイルを変換して，slhoge.tex を作成する場合には “perl texslide.pl hoge.tex > slhoge.tex” とする.
- このあと “platex slhoge” と “dvipsk -t a3 slhoge” とするので，makefile を用意して “make” で論文用が，“make slide” でスライド用ができるようにしている.

# 移行対象の環境

---

- figure 環境，equation 環境，table 環境などを検出し，この環境をまとめて “\begin{slide}” と “\end{slide}” の中に埋め込む.
- 変換対象の環境は figure，table，equation，displaymath，eqnarray，align，gather，quote，itemize，enumerate，description，thebibliography 環境としている.
- 検出は入力行から \begin{xxx} を探しており，1 文字目にブランクを入れた場合は変換対象から除外される.

# ダミー環境

---

- 論文には書かれていないが，プレゼンテーションでは必要な（式の細かな変換など）補足説明のためのスライドや，論文の紙数制限で割愛した数式を表示する目的で使用する．
- また大学の講義では，先週の講義で使用した図をその日の講義テーマの前に復習で見せるなど，別ファイルの図を説明に使いたいということがある．
- これらの目的で，上記の環境の行頭に%を 1 つだけ付加した行は，%を外してスライドには現われる機能を設けた．

# 見出しの表示

---

- section，subsection，subsubsection，paragraph に対しては，その引数（見出し）をそれぞれ Huge，huge，LARGE，Large のフォントサイズでスライドの中央に center 環境で出力するように変換している[a].

[a] スタイルファイルとして jarticle.cls を対象としている.

# ディレクティブ

- 上記の変換だけでは不十分であるが，次の 4 種類のディレクティブを挿入することで，実用的なスライドの加工が可能になる．

**%term** キーセンテンスの直前に “%term” の 1 行を加える．これによって次のレコード（行）を itemize 環境のアイテムとして埋め込む．1 センテンスではなく，センテンスの一部や**単語**を指定する場合は，1 センテンス 1 レコード原則を諦めて，改行を細かく入れて，表示に回す部分を 1 レコードとして切り出し，その直前に%term を指定する．指定されたキーセンテンスは見出しの下に箇条書きされる．

**%newslide** 指定したキーセンテンスが多いと，1 枚のスライドに入りきらない．そこで “%newslide” を入れて強制的に**スライドを改める**ようにした．

**%copystart と%copyend** このディレクティブに挟まれたソーステキストは，スライド用のソーステキストに**コピー**する．複数の数式を 1 枚のスライドに収めるために使用する．講義などで，式の展開を確認しながら説明する場合には必須である．

**% SlideReplace** seminar 環境での図のサイズ調整のために用意した．図を縮小する必要がある場合，これを変換プログラムで自動的に行うのは無理と考えた[a]．ここでは図ごとにサイズを，論文用とスライド用の 2 通りに指定して，“\psset{unit=1cm}
% SlideReplace\psset{unit=8mm}” のように**置き換え**る．

---

[a] 変換対象の環境の終わりまでを読み込んで，スライドに変換された場合の縦横のサイズを予測する必要がある．

# 『LaTeX & PostScript スーパーユーザのテクニック』例

---

- 共立出版さんのご厚意によりサンプルをダウンロード可能としていただいた.
- これらは金沢工業大学でのプログラミング演習の説明や，雑談などの目的で用意したものが大半であるが，数式が含まれるものが多く，LaTeX 以外のツールで説明文章を用意することに気が進まないものである.
- 説明文章とプレゼンテーション用スライドを 2 重に用意するのは手間と考え，これらの資料の作成と並行して，texslide.pl を改良していった.
- 変換プログラムに print ”\\slideframe{none}\n”; の 1 行を加えることで，%size ディレクティブを不要にできる.

**ベジェ曲線** コンピュータグラフィックスの内容で，フォントデザインには欠かせない知識である．内容に数式の展開が多いので，スライドは %copystart と %copyend を多用して，1 枚のスライドで式の展開が追える形にした．

**RSA 暗号の根拠となる表** $x^k \pmod m$ を，$m$ や $k$ を変更して作表する TEX マクロ（アライメントタブを生成するマクロ）を解説した．後半は $x^k \pmod m$ に関する話題（フェルマーの定理）と，RSA 暗号，またユークリッドの互除法と連分数の幾何学的解釈について述べる．

**Calcomp インターフェース** 古典的なペンプロッタのインターフェースを，PostScript（calcomps.c）と X-Window（calcompx.c）で実装する．どちらも Fortran と C から呼び出し可能である．

**対話型プログラム** Calcomp インターフェースでは計算結果のポスト処理をしたので，X-Window の対話型の機能を利用しなかった．ここではその延長で対話形式の機能（イベント駆動型のループ構成）を経験してみる．

**フラクタル** Cygwin や Linux 環境でのプログラミングと計算結果の出図の方法を実習する．題材はフラクタル図形で，その語源である分数（fractional）次元の説明も加えた．

**疎行列のプロット** 数値解析プログラムが差分法と有限要素法を使用した場合の，疎行列の非ゼロ要素の位置と，その行列を三角分解したときに現れるフィルインと呼ばれる要素の位置を作画する．

**MPI ライブラリ** IBM の超並列型スーパーコンピュータ Blue Gene で，MPI ライブラリを MPICH2 をベースに実装した方法を解説する．ここでは説明文に記述された以上の図をスライドに用意して，MPICH2 に少しずつコンポーネントを追加してゆく様子を紙芝居でプレゼンテーションする．

**2 次元配列分割・分散** 線形代数の並列計算では行列を 2 次元配列分割して，並列計算のノードに分散させる．密行列を 2 次元ブロックサイクリックに配列分割して MPI プログラミングする人には便利なデバッグツールでもある．

**写真機のレンズ** レンズ面を増やすと収差を除去するためのパラメータが増えるので，収差の少ないレンズを設計することができる．初等関数の多項式近似で，多項式の次数（パラメータ）を増やすと精度が良くなるのに似ている．Calcomp インターフェースの演習問題の背景の説明でもある．

**住所録から宛名変換** ソーステキストの変換プログラムの概要を述べる．宛名の郵便番号は正確に置く必要があるので，TEX の picture 環境で座標値を指定している．

**MusiXTEX** 楽譜を書くための MusiXTEX を紹介し解説する．「楽譜とその演奏」を「文書テキストとその印刷」のアナロジで考えたエッセイを付加した．「音律は音楽のアーキテクチャ」か．

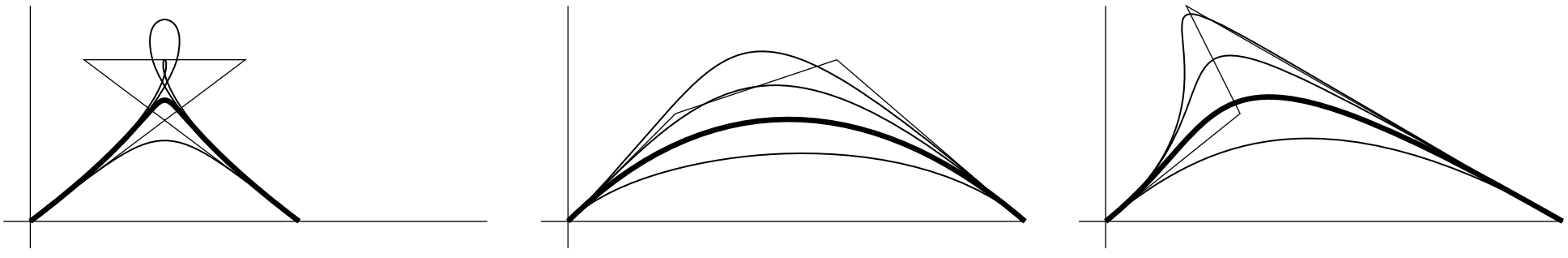

Figure 4: ベジェ曲線の根拠

```
\begin{pspicture}(0,1)(5,3)
\rput(3.3,0){\includegraphics[100,70][500,200]{bez3rat.ps}}
\end{pspicture}
```

Table 1: $x^k$ (mod 10) by \PowerModTable(1,11,10,12)

| $x$ \ $k$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 2 | 4 | 8 | 6 | 2 | 4 | 8 | 6 | 2 | 4 | 8 |
| 3 | 1 | 3 | 9 | 7 | 1 | 3 | 9 | 7 | 1 | 3 | 9 | 7 |
| 4 | 1 | 4 | 6 | 4 | 6 | 4 | 6 | 4 | 6 | 4 | 6 | 4 |
| 5 | 1 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 6 | 1 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| 7 | 1 | 7 | 9 | 3 | 1 | 7 | 9 | 3 | 1 | 7 | 9 | 3 |
| 8 | 1 | 8 | 4 | 2 | 6 | 8 | 4 | 2 | 6 | 8 | 4 | 2 |
| 9 | 1 | 9 | 1 | 9 | 1 | 9 | 1 | 9 | 1 | 9 | 1 | 9 |
| 10 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Figure 5: fern と tree

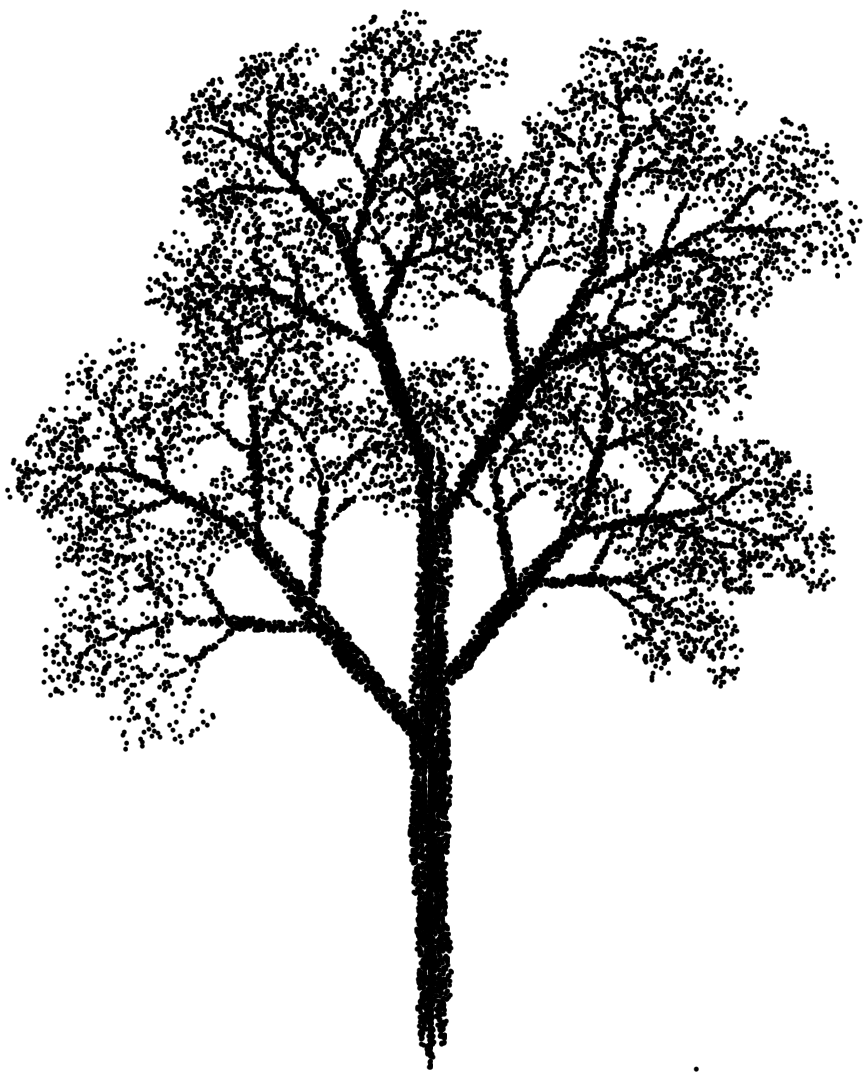

Figure 6: fern と tree

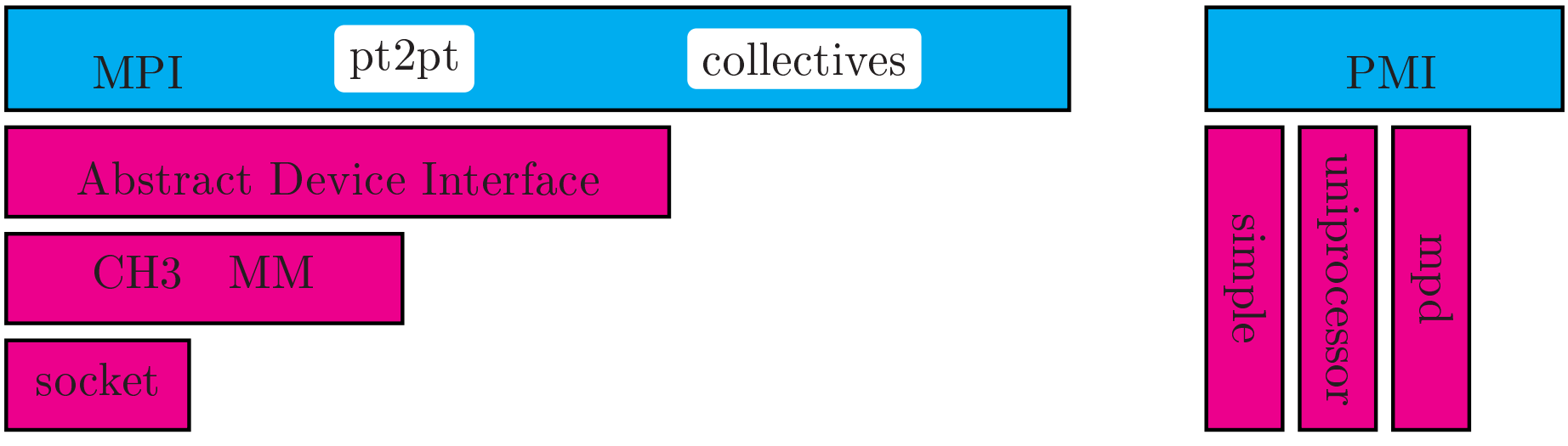


Figure 7: MPICH2

```
\newcommand{\MPICH}{%
%%%%%%  Top layer painted with Cyan   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\psframe[fillcolor=cyan,fillstyle=solid](0,4.4)(8,5.2)
 \rput(1,4.7){MPI}
 \rput(3,4.8){\psframebox*[framearc=0.3]{pt2pt}}
 \rput(6.0,4.8){\rnode{Col}{\psframebox*[framearc=0.3]{collectives}}}
%%%%%%%%% PMI %%%%%%%%%%
\psframe[fillcolor=cyan,fillstyle=solid](9,4.4)(11.7,5.2)
 \rput(10.4,4.7){PMI}
%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%  Second layer painted with Magenta %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\psframe[fillcolor=magenta,fillstyle=solid](0,3.6)(5,4.3)
 \rput(2.5,3.9){Abstract Device Interface}
%%%%%%%%% PMI %%%%%%%%%%
\psframe[fillcolor=magenta,fillstyle=solid](9,2)(9.6,4.3)
 \rput(9.3,3.15){\rotateright{simple}}
\psframe[fillcolor=magenta,fillstyle=solid](9.7,2)(10.3,4.3)
 \rput(10.0,3.15){\rotateright{uniprocessor}}
\psframe[fillcolor=magenta,fillstyle=solid](10.4,2)(11.0,4.3)
 \rput(10.7,3.15){\rotateright{mpd}}
%%%%%%%%%%%%%%%%%%%%%%%%
\psframe[fillcolor=magenta,fillstyle=solid](0,2.8)(3,3.5)
 \rput(1,3.2){CH3}
 \rput(2,3.2){MM}
\psframe[fillcolor=magenta,fillstyle=solid](0,2)(1.4,2.7)
 \rput(0.7,2.4){socket}
}
```

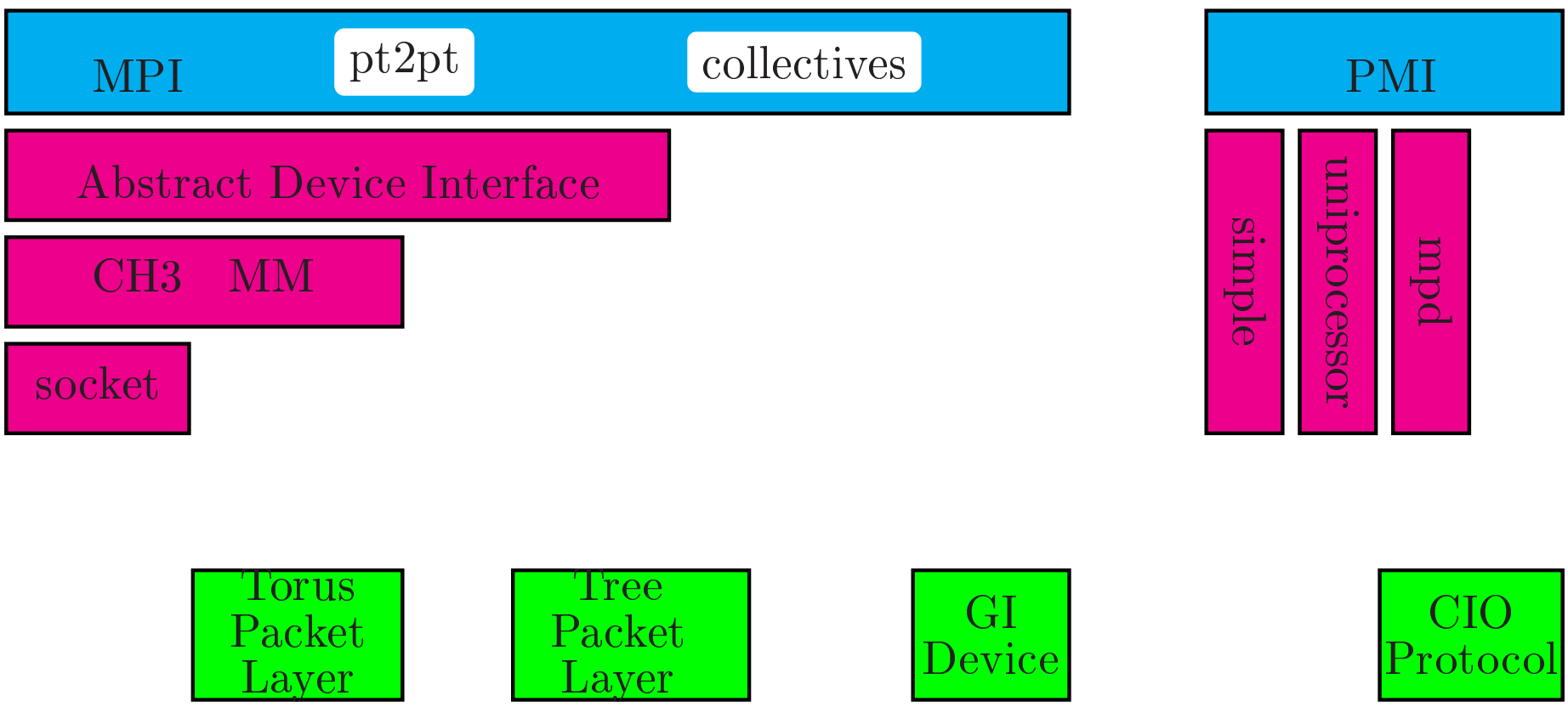


Figure 8: MPICH2 + BG/L network H/W

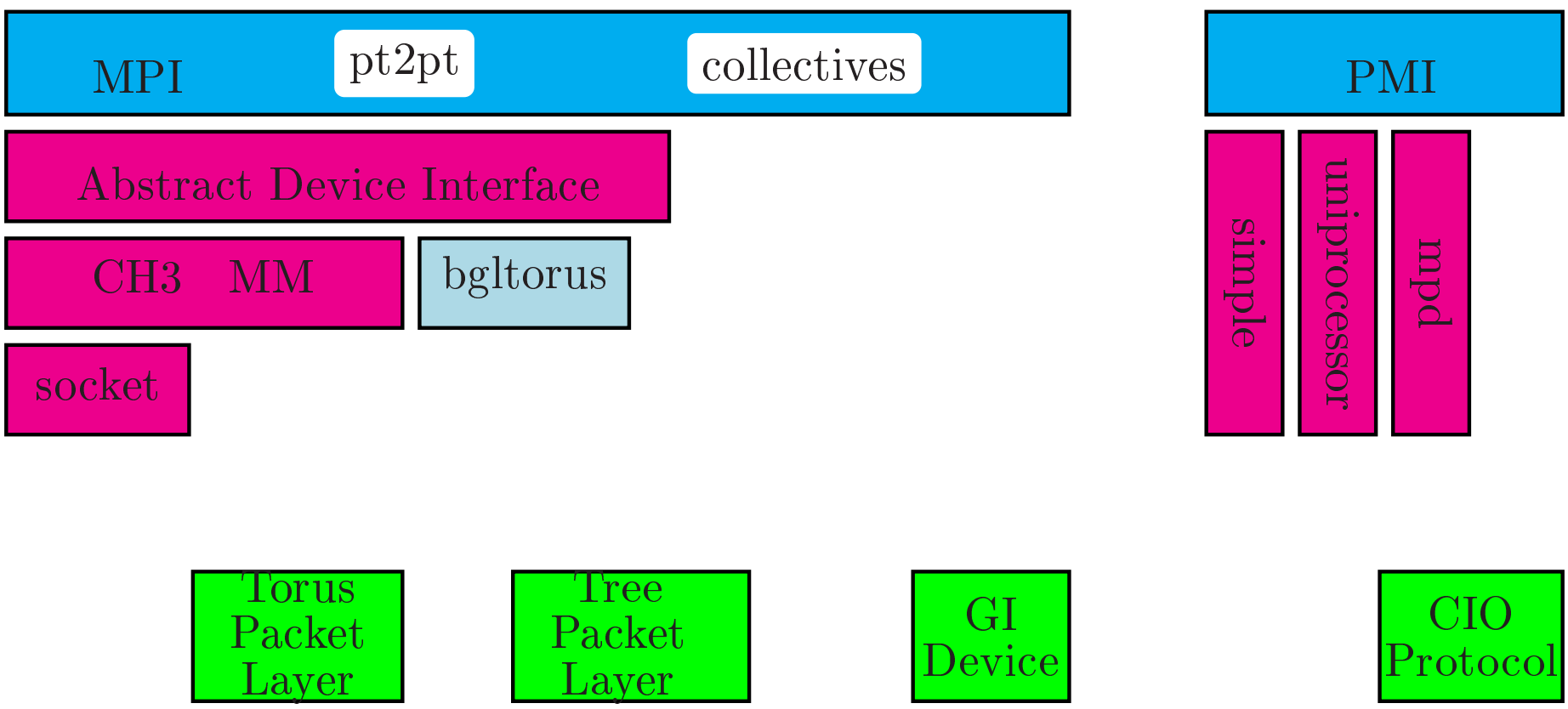


Figure 9: Torus interface

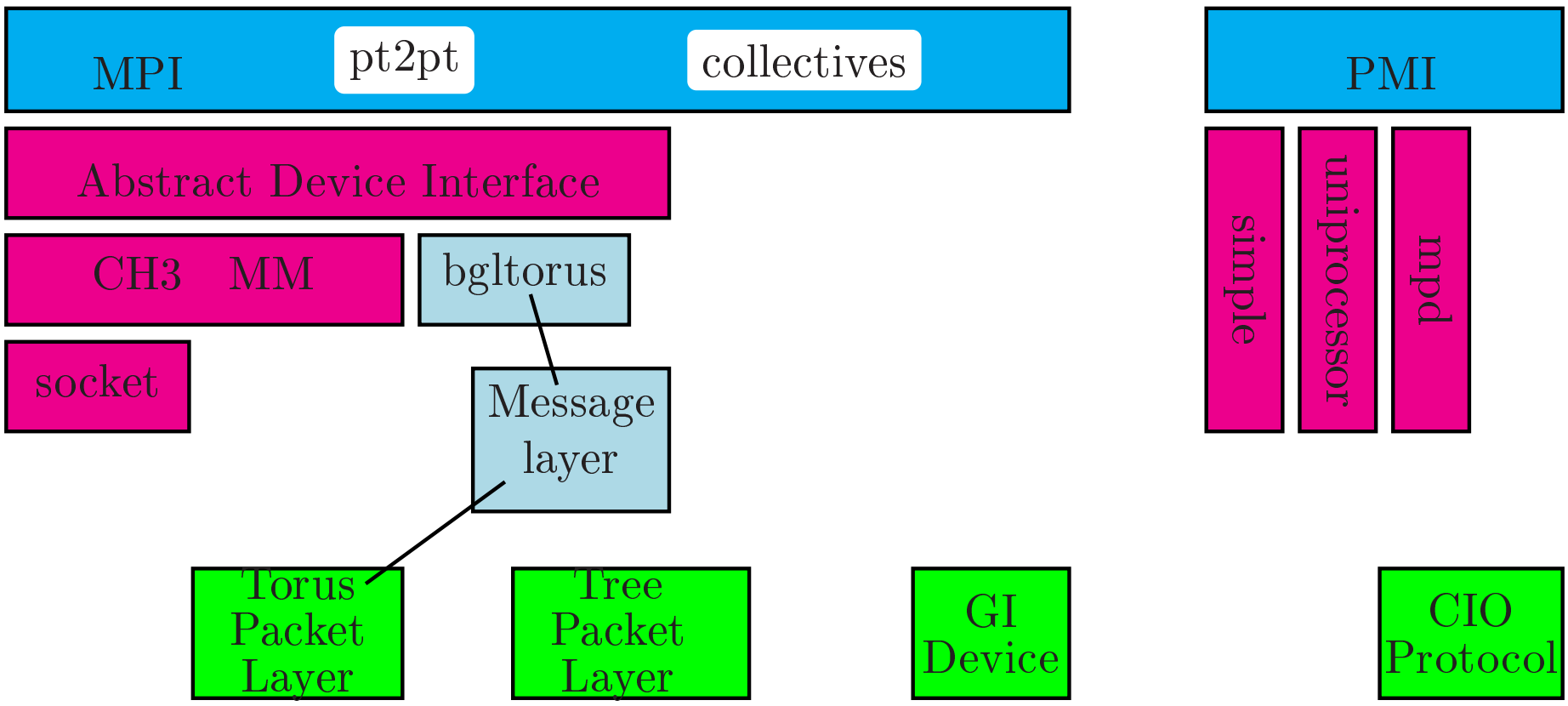


Figure 10: Torus implementation

```
\MPICH
\NetworkHW

\psframe[fillcolor=lightblue,fillstyle=solid](3.1,2.8)(4.7,3.5)
 \rput(3.9,3.2){\rnode{Btor}{bgltorus}}
\psframe[fillcolor=lightblue,fillstyle=solid](3.5,1.4)(5,2.5)
 \rput(4.25,2){\rnode{ML}{\shortstack{Message\\layer}}}
   \ncline{Btor}{ML}
   \ncline{ML}{TP}
```

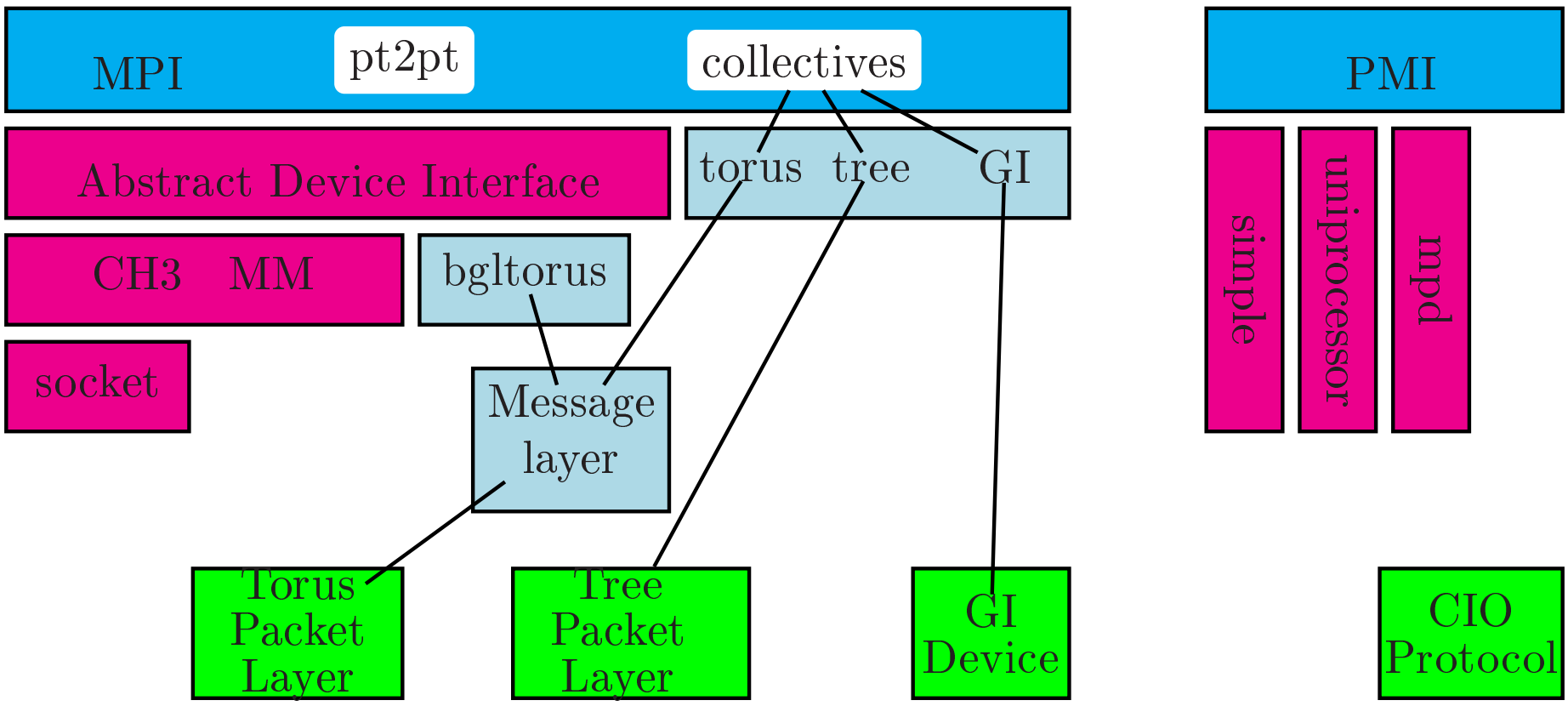


Figure 11: Collective communications

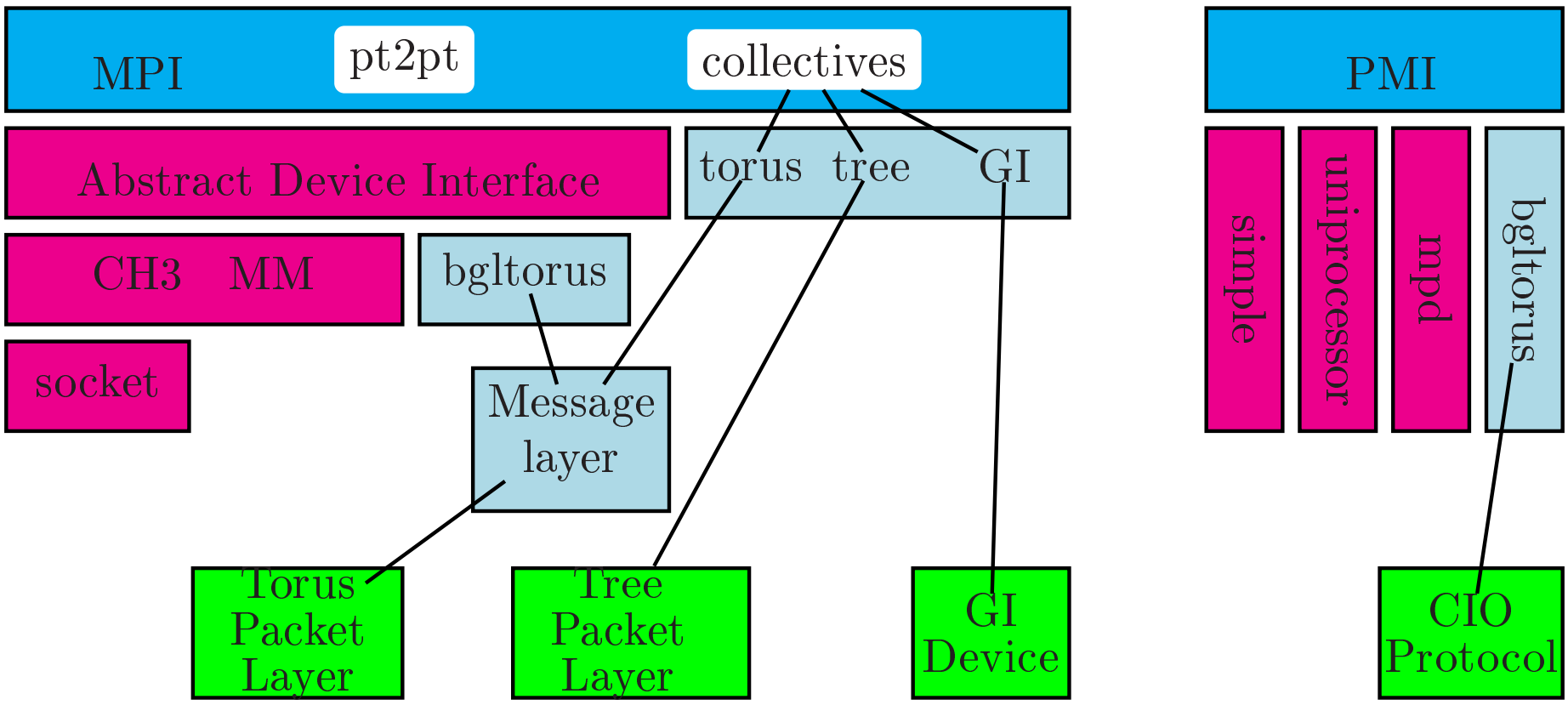


Figure 12: MPICH2 BG/L Roadmap

$$\int_\alpha^\beta vLudx = \int_\alpha^\beta \{vqu'' + vbu' + vcu\}dx \tag{1}$$

$$= [vau' + vbu]_\alpha^\beta + \int_\alpha^\beta [-(va)'u' - (vb)'u + vcu]\,dx \tag{2}$$

$$= [vau' + vbu - (va)'u]_\alpha^\beta + \int_\alpha^\beta [(va)''u - (vb)'u + vcu]\,dx \tag{3}$$

$$= [avu' - (av)'u + bvu]_\alpha^\beta + \int_\alpha^\beta u\,\underbrace{[(av)'' - (vb)' + cv]}\,dx \tag{4}$$

$$= L^* v$$

```
\begin{align}
 \int_{\alpha}^{\beta} v L u dx &= \int_{\alpha}^{\beta} \{\rnode{A}{vau''}
   + \rnode{B}{vbu'} + vcu \} dx \\
   &= [\rnode{Aone}{vau'} + \rnode{Bone}{vbu}]_{\alpha}^{\beta}
  + \int_{\alpha}^{\beta} \left[ \rnode{Atwo}{-(va)'u'} \rnode{Btwo}{-(vb)'u} +
 \ncline{->}{A}{Aone}  \ncline{->}{A}{Atwo}  \ncline{->}{B}{Bone}  \ncline{->}
   &= [vau' + vbu -\rnode{Athree}{(va)'u}]_{\alpha}^{\beta}
  + \int_{\alpha}^{\beta} \left[ \rnode{Afour}{(va)''u} -(vb)'u + vcu\right] d
  \ncline{->}{Atwo}{Athree} \ncline{->}{Atwo}{Afour} \\
   &= [avu' -(av)'u +  bvu ]_{\alpha}^{\beta}
     + \int_{\alpha}^{\beta} u\underbrace{\left[ (av)''-(vb)' + cv\right]}_{=L^*
\end{align}
```

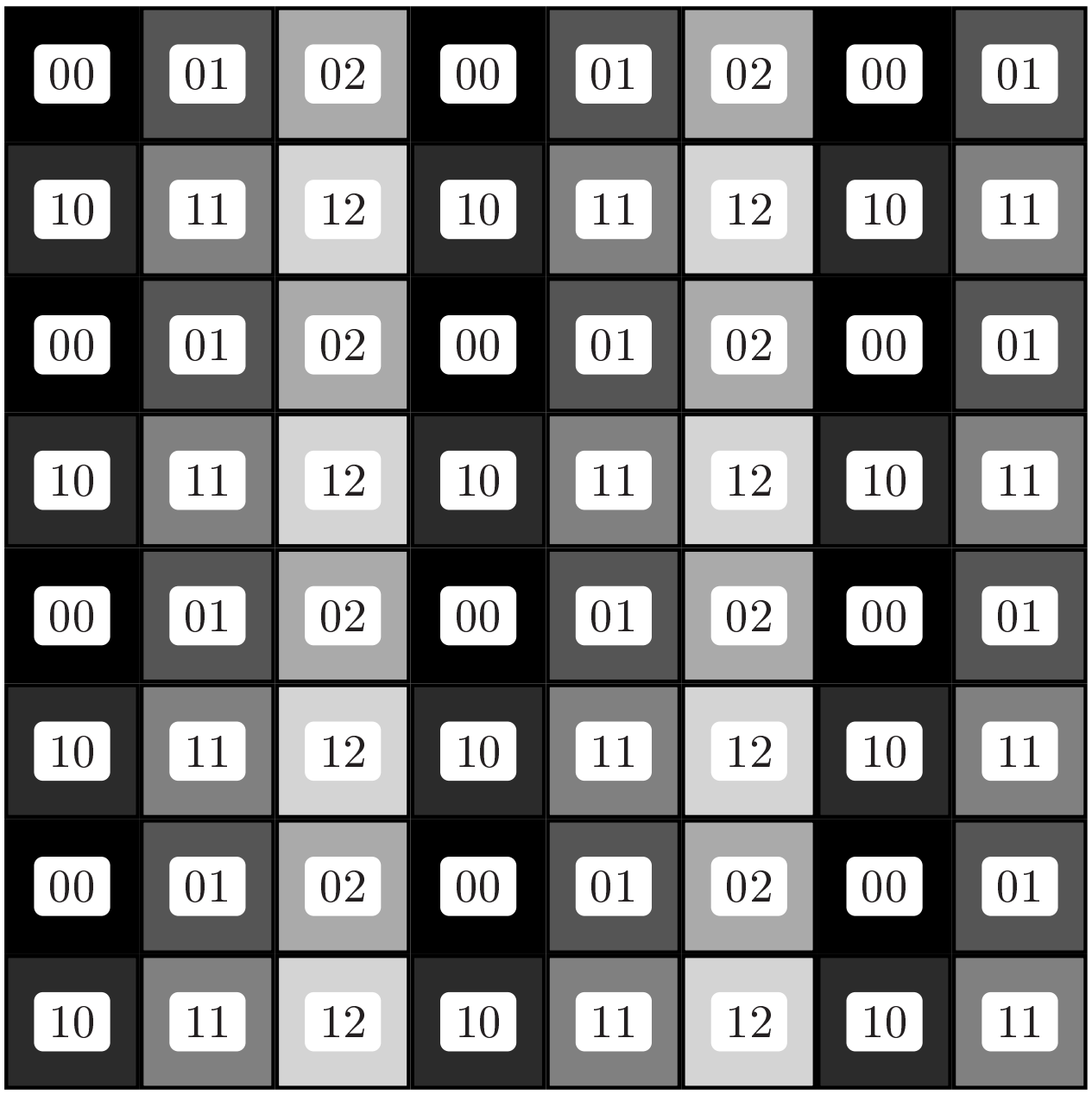


Figure 13: 2 次元配列分割（小行列を所有するノード番号）

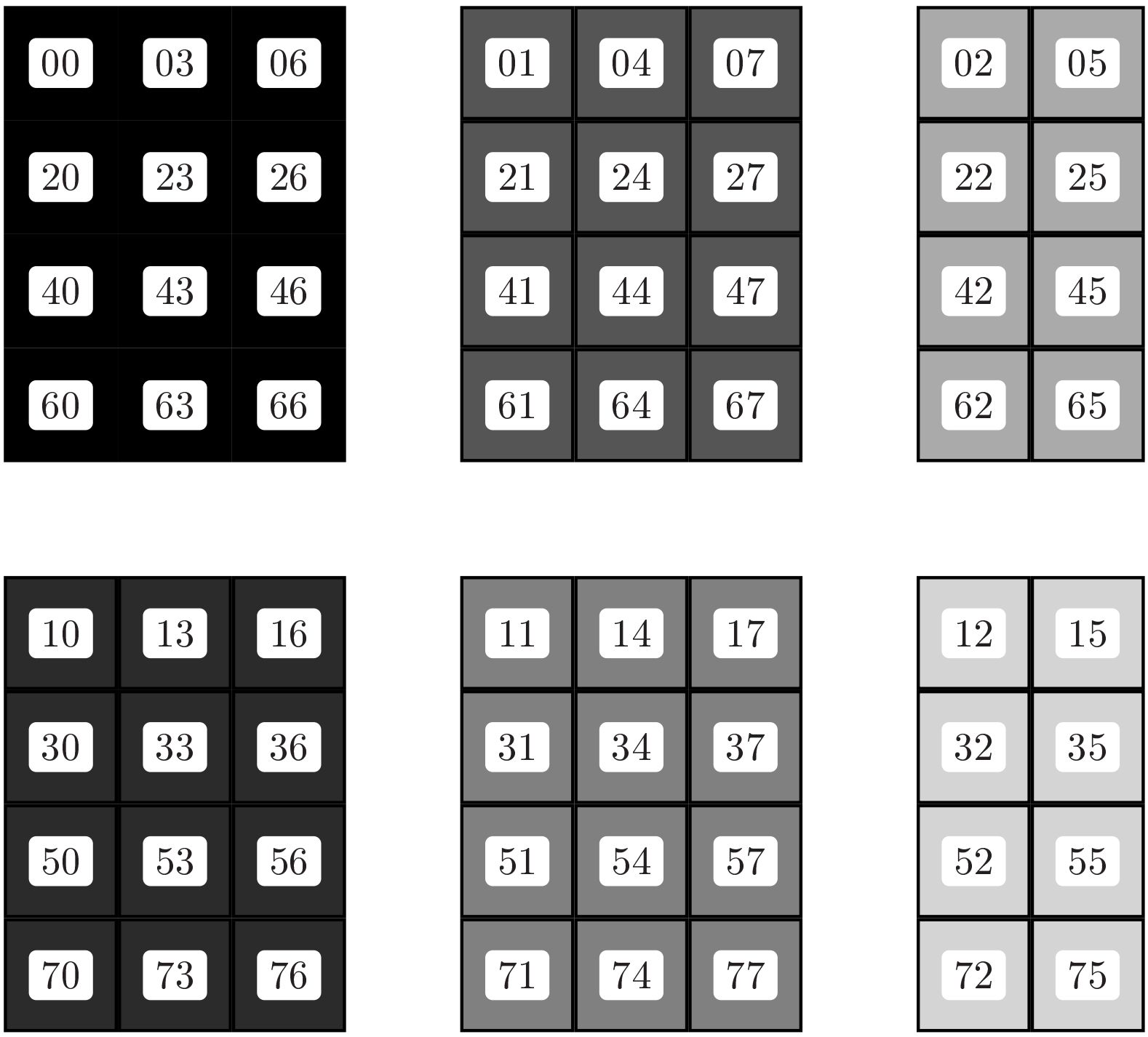


Figure 14: ブロックサイクリック分割・分散のローカルの視野（小行列 $A_{IJ}$）

```
\RecBlkCyc{8}{8}{2}{3}{1}

\RecBlkCycL{8}{8}{2}{3}{1}{0}{0}
\put(4,0){
 \RecBlkCycL{8}{8}{2}{3}{1}{0}{1}
}
\put(8,0){
 \RecBlkCycL{8}{8}{2}{3}{1}{0}{2}
}
\put(0,-5){
 \RecBlkCycL{8}{8}{2}{3}{1}{1}{0}
}
\put(4,-5){
 \RecBlkCycL{8}{8}{2}{3}{1}{1}{1}
}
\put(8,-5){
 \RecBlkCycL{8}{8}{2}{3}{1}{1}{2}
}
```


Figure 15: Leitz Minolta CL (Leica CL) 40mmF2, 90mmF2, 28mmF2.8

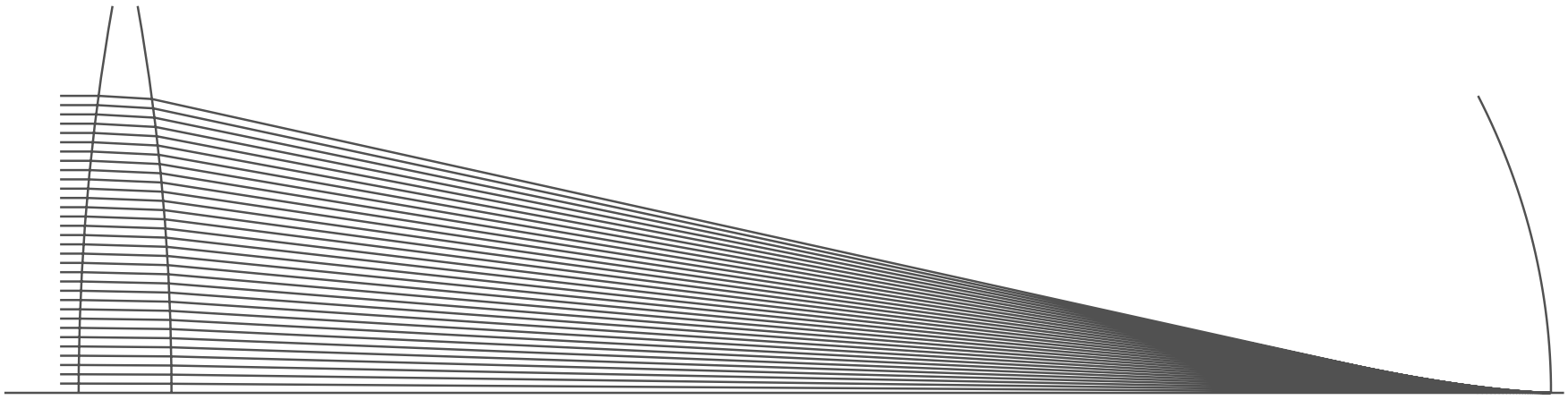

Figure 16: レンズを通過する光線と球面収差

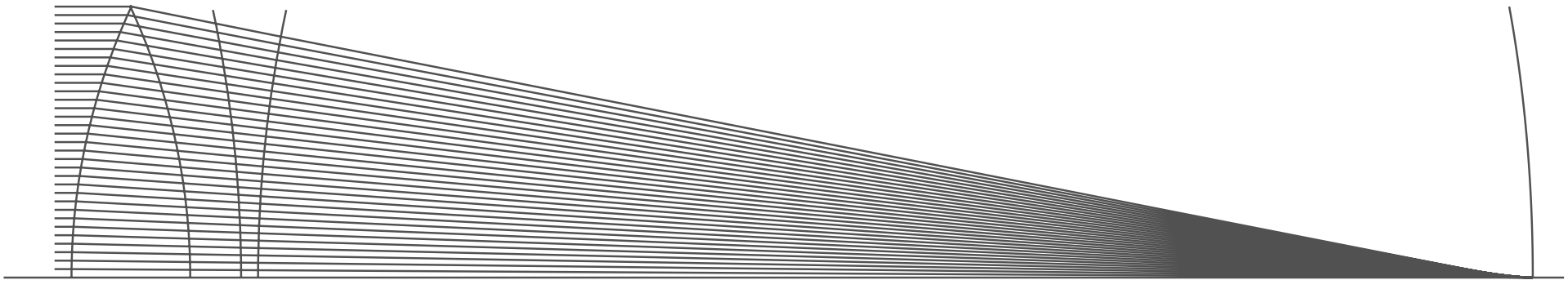

Figure 17: 2 枚のレンズを通過する光線と球面収差

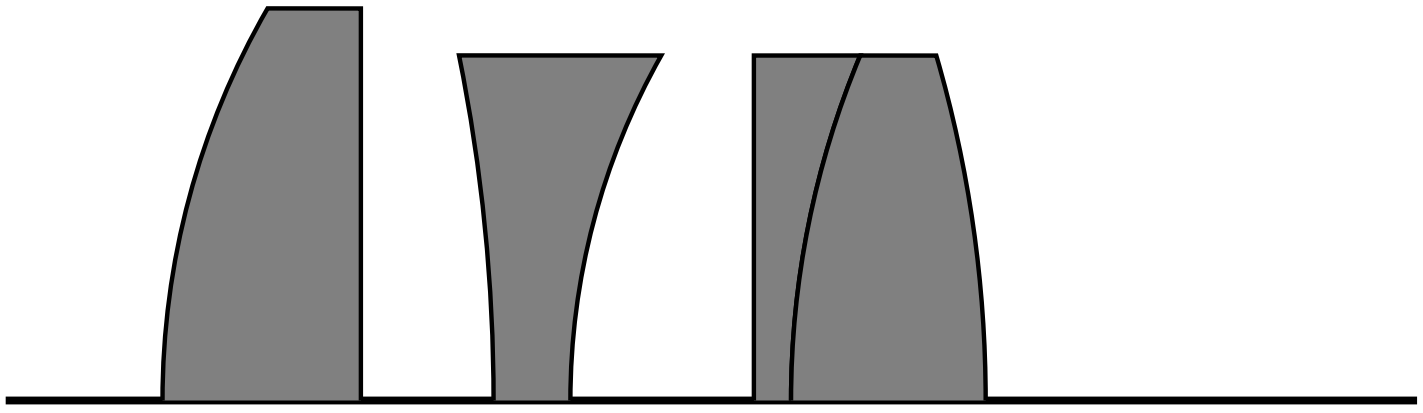

Figure 18: テッサー型のレンズ:ロッコール 45mmF2.8（ミノルタハイマチック）

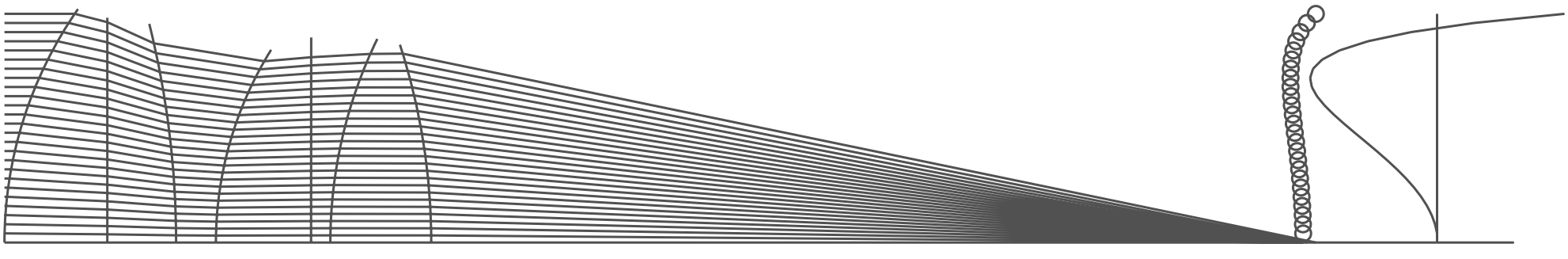

Figure 19: 4 枚のレンズを通過する光線と球面収差

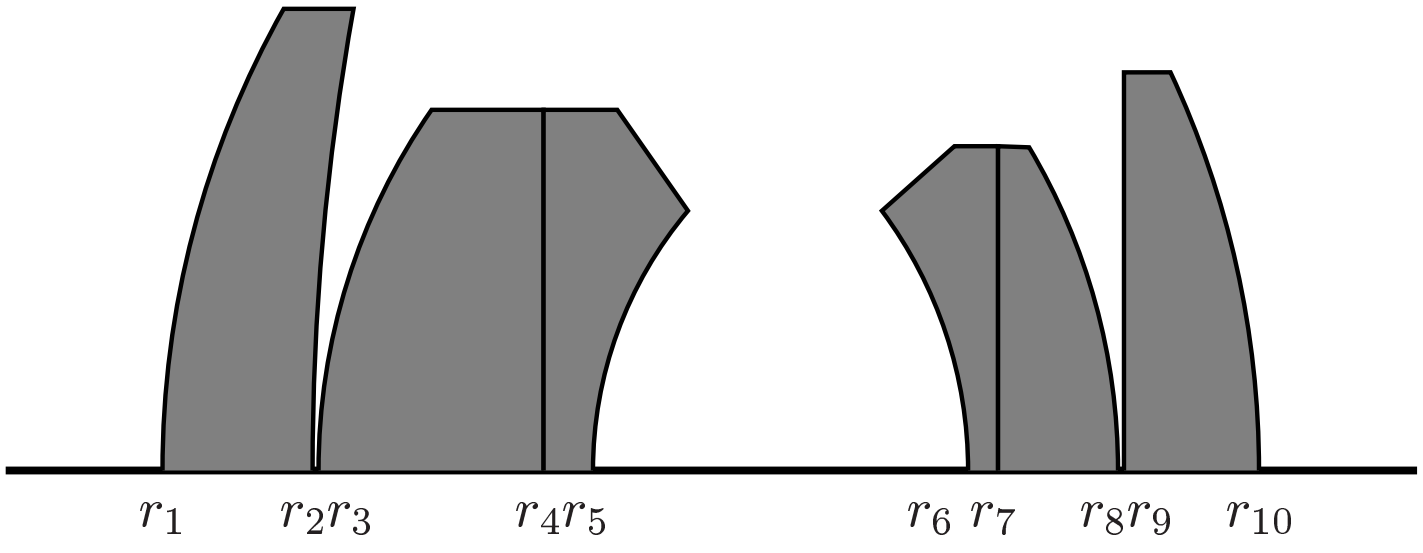


Figure 20: ガウス型のレンズ:ライカ summicron F2.0

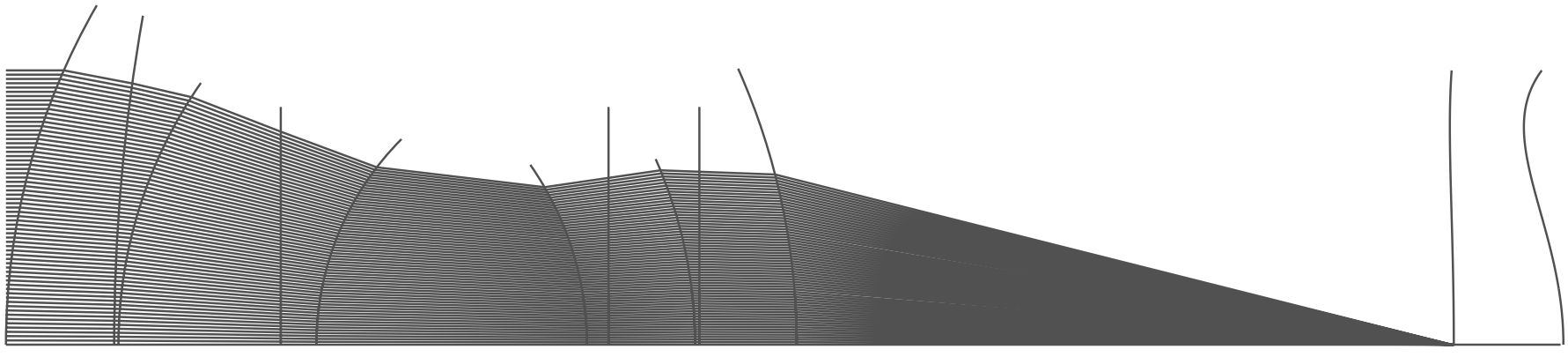

Figure 21: 6 枚のレンズを通過する光線と球面収差

# Minuetto in G-dur

J. S. Bach

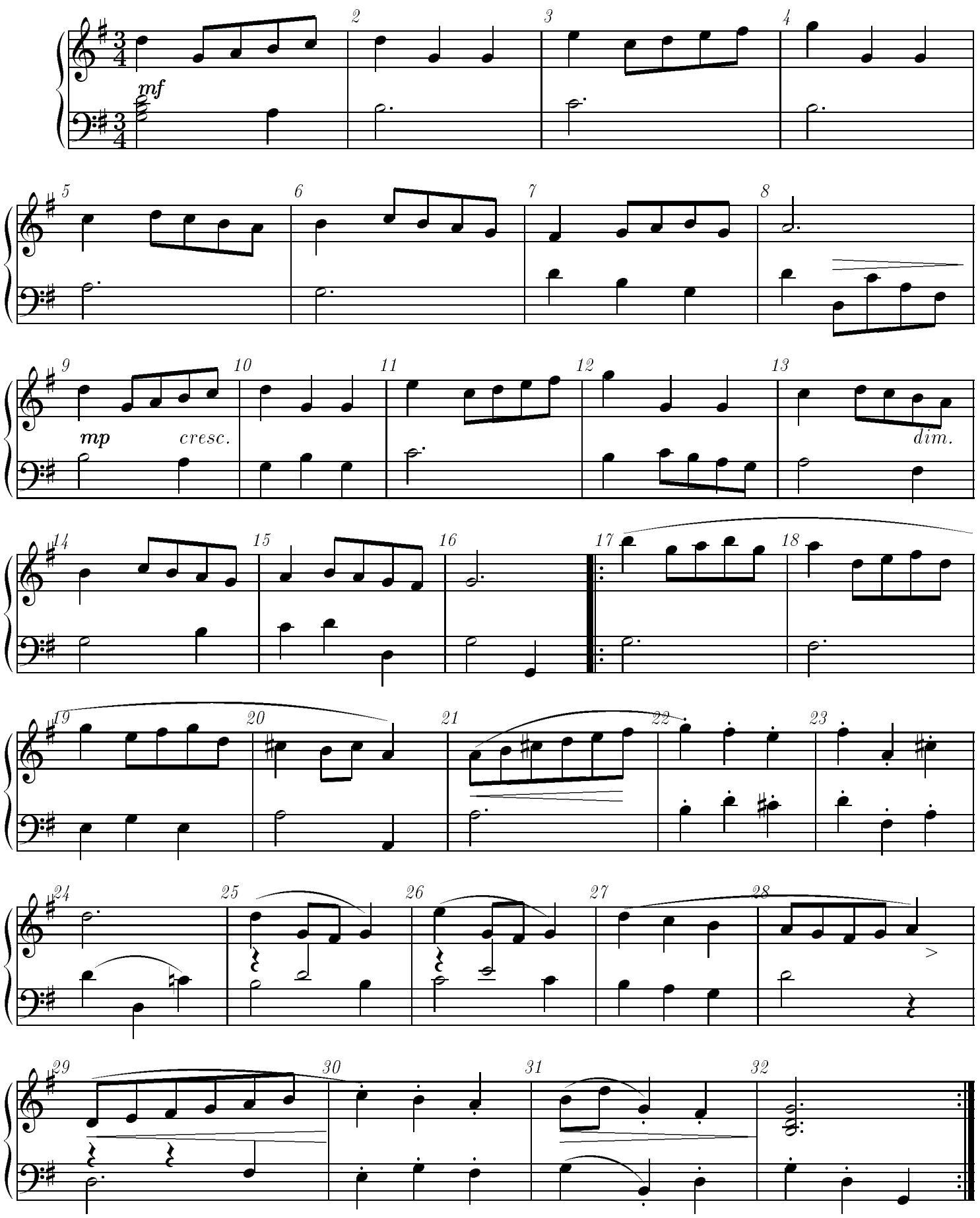

Table 2: 音律の比較

| | ピタゴラス音律 | | | ツァルリーノ音律 | | | 中全音律 | | | 平均律 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 音程比 | 小数 | セント | 音程比 | 小数 | セント | 音程比 | 小数 | セント | 音程比 | 小数 | セント |
| C | 1 | 1.0000 | 0.0000 | 1 | 1.0000 | 0.0000 | 1 | 1.0000 | 0.0000 | 1 | 1.0000 | 0 |
| D | 9/8 | 1.1250 | 203.91 | 9/8 | 1.1250 | 203.91 | $\sqrt{5}/2$ | 1.1180 | 193.16 | $2^{\frac{1}{6}}$ | 1.1225 | 200 |
| E | 81/64 | 1.2656 | 407.82 | 5/4 | 1.2500 | 386.31 | 5/4 | 1.2500 | 386.31 | $2^{\frac{1}{3}}$ | 1.2599 | 400 |
| F | 4/3 | 1.3333 | 498.04 | 4/3 | 1.3333 | 498.04 | $2/5^{\frac{1}{4}}$ | 1.3375 | 503.42 | $2^{\frac{5}{12}}$ | 1.3348 | 500 |
| G | 3/2 | 1.5000 | 701.96 | 3/2 | 1.5000 | 701.96 | $5^{\frac{1}{4}}$ | 1.4953 | 696.58 | $2^{\frac{7}{12}}$ | 1.4983 | 700 |
| A | 27/16 | 1.6875 | 905.87 | 5/3 | 1.6667 | 884.36 | $5^{\frac{3}{4}}/2$ | 1.6719 | 889.74 | $2^{\frac{3}{4}}$ | 1.6813 | 900 |
| H | 243/128 | 1.8984 | 1109.8 | 15/8 | 1.8750 | 1088.3 | $5^{\frac{5}{4}}/4$ | 1.8692 | 1082.9 | $2^{\frac{11}{12}}$ | 1.8877 | 1100 |
| C’ | 2 | 2.0000 | 1200.0 | 2 | 2.0000 | 1200.0 | 2 | 2.0000 | 1200.0 | 2 | 2.0000 | 1200 |

# おわりに

- 情報は上流から下流へとコンパイルされ変形されるたびに形を変え，最終的には美しい図を含む読みやすい文書になるが，ある種の情報，例えば図形の持つ論理的な意味をプログラムで抽出することは難しくなる．
- ここでの「意味」は，対称性や繰返し，曲線の次数などを指している．
- このような意味は，資料の最上流では可読な形で存在するが，下流へ進むたびにそれはプログラムの実行結果に置き換えられてしまうので，上流のほうが意味を正確に扱うことができる．
- 最終的には PDF ファイルが表示したビットマップを人間が眺めて判定するしか方法がなくなる．

- この段階まで進んでしまうと，GUI を使ってカットアンドペーストでパワーポイントファイルに貼りつけたりしなくてはならなくなる[a].
- 論文などの文書に描かれた図をスライド用にするには，拡大して配置し直すだけでよいので，ソーステキストのまま取り出せれば簡単にできそうである.
- 数式や表も同様である.
- LaTeX の場合，figure，table，equation 環境で書かれた図，表，数式を，拡大してスライドの中央に置くだけでよい.

[a] LaTeX のソーステキストで eps ファイルで取り込まれた図を，LaTeX でコンパイルして PS ファイルに変換し，さらに PDF 変換してからそれを巨大なビットマップファイルとして取り出してパワーポイントに貼りつけたりすると，**クリック猿**と呼ばれかねないので注意しよう.

- 『LaTeX & PostScript . . .』は金沢工業大学の大学院で「特別講義」として行ったテキストを元にしている.
- 2002 年に始めた講義内容で，当時はグリッドコンピューティングが脚光をあびていた時期で IT 革命の進行とともにシステムが複雑化・巨大化してきた時期である.
- この複雑化を支えてきたシステムの階層構造と**マルチプログラミングリンガル**の重要性を痛感していた.

- 論文や書籍出版などを扱う「文書処理」アプリケーションでも，ハードウェア（写植機），制御言語，その上に構築される LaTeX 2$_\varepsilon$ と PostScript の世界があり，文書はその上に書かれてゆく．
- これらの階層を縦に繋げて，どの階層で何がおこっているかを仔細に分析する経験は，システムエンジニアとしてエンジニアリングセンスを磨く上で役立つのではないかと考えた．
- ここには普段はあまり目にしないマクロプログラミングや PostScript 言語に触れる新鮮さもある．
- また，プログラムの仕様を決める経験もできる．
- この題材では，自然にプログラミングを原理的な姿から考え直す機会を与える．

最もプリミティブな，電卓のキーを叩いてでも行えるような計算を行うプログラムから始めて，判断と繰返しを記述したコードに制御を与えることの意味を実感してもらう．このようなプログラミングでも，プログラマは繰返しの過程を自分の頭に構築する．これは 1 レベルの間接実行である．PostScript なので仮想的なペンプロッタのようなマシンも組み立てられるので，インターフェースも自分で定義可能である．変数の型やスコープ，サブルーチンへの引数の渡し方，データ構造の意義，ルールベースのプログラミングなど，改めて考えてみたい題材がいろいろ出てくる．このようなプログラミング言語の発達も，文書処理の中でクイックに体験できる．また，文字コードの規格の復習もできる．

情報系の学生には，工学部の他の学科の学生よりも IT 技術の階層ごとの知識や階層間のインターフェースについて正確な知識をもっていてほしい．思えば IT 革命は Windows 95 に，パソコンの OS としては初めて TCP/IP が標準で装備されてから，Web ブラウザとインターネットで急速に加速された．

- インターネットの上に展開された IT システムの階層構造はとても深くなり，各階層に現在でも次々と新しい技術が投入されている．
- アセンブリ言語で機械語を並べるプログラミングを平屋にたとえるなら，現代の IT システムは高層ビルディングにたとえられよう．
- その最上階からアプリケーションを叩く（に要求を出す）と，下の階でアルゴリズムが回って複雑な処理がなされる．
- システム全体をブラックボックスとして見るのではなく「なぜこれが機能するのか」という問題意識を忘れずに，アルゴリズムは何階で回っているのだろうかと考える習慣を忘れないでほしい．
- 1 つ下の階のことは考えれば分るが，2 つ下の階になると，十分な知識がなくては分らないことが多い（各階のアルゴリズムは意外に単純な場合が多いにもかかわらず）．

知識の次はプログラミング能力である．IT 革命の進行とともに，すべての分野で利用できるツールはとても便利になった[a]．この結果，自分でアプリケーションの骨格となるループ構造をプログラミングする機会は少なくなっている．大学で（社会人を対象とする講習会のように）よくできたツールの利用技術を教えてみても楽しくない．せめて学生時代は，原理的な内容を勉強して，プログラミングに時間をかけてみたい．問題を与えられると反射的に Google を検索するのではなくて，まず自分の頭で考え，自分の創意工夫を自分のコードに埋め込んで，デバッグして稼動させられたときの喜びを体験できれば，プログラミングを趣味にできる人が増えてくれるのではないかと考えた．また数学アレルギーを多少なりとも解決させられればと考え，数学の例題も増やした．自分で仕様を考え，自分で作り，自分で使うことを繰り返して，すこしずつ良いものに洗練させてゆくのも楽しみである[b]．

[a] 文書処理のツールも然りで，論文もワープロで書ける時代になった．その結果，文書処理システムのインフラ技術に触れる機会が少なくなってしまった．

[b] 本書に述べたスライド変換プログラムは，はじめ簡単に作ってはみたものの，実際に非常勤講師として数値計算の講義の中で使ってみると機能が不足しており，ディレクティブを追加して本書の姿になった．

日常の筆記のベースを LaTeX に持つと，ノート代わりのメモに現れる数式を清書できるので便利である．これを授業の資料，論文，講演のスライド，さらにマニュアル，教科書，参考書の出版にまで何度も利用することが可能になる．そのような LaTeX とのお付き合いをしている方々に，本書で紹介したスライド変換や索引語の自動挿入プログラムがお役に立てれば光栄である．