e-learningを用いた自習システムにおける問題の作成について

平成 23 年 2 月 14 日

情報電子工学科
渡辺 崇寛

目 次

概要

大学の数学の講義では高校の授業に比べて問題演習が少なく、「もう少し演習をやりたい」と考える学生は、図書館の本や演習書などを利用して演習する必要がある。しかし、似た問題を大量に演習するには適さない場合があるし講義のレベルや種類に合っているとは限らない。そこで、値をランダムに変化させ、解法方法が似た問題を作成できる e-learning を用いた、自習をするための問題作成の部分について考察する。学習内容に合う問題が解けることが可能となるような数学の自習システムについて、どのようにプログラムを表現すれば自習に向くか考察していく。今回は自習システムの問題作成の部分について LaTeX を主としたプログラムの作成をおこない、そこから浮かび挙がった問題点を改善した Perl を主としたプログラムを作成し考察をおこなった。

1　はじめに

数学の学習において、計算をするための解法方法を身に付けるには多くの演習が必要となる。しかし、本学を含め大学の講義では演習をあまり多くおこなっておらず、演習が少ないと感じる人もいる。自習をするために通常は演習書を買う、図書館で本を借りるなどをするだろうが、講義のレベルや種類に合っているとは限らないため、多くの演習をするのには不向きである。だが、教員個人に数学の問題と解答を大量に用意してもらうには問題作成者の負担が大きすぎる。

そこで、問題の値を静的にではなく動的に変化させ、解法方法が似た問題を作成できる e-learnig を用いた、Web ベースでの自習をおこなう人のための自習システムの開発を目標とする。また、問題や解法、解答をどのように表現すれば自習に向くシステムなのかを考え、次に自習システムの問題作成の部分について考察する。

2　e-learning システム

2.1　e-learnig とは

Wikipedia[2] によると e-learning とはパソコンやコンピュータネットワークなどを利用して教育を行ない、教室で学習を行なう場合と比べて遠隔地にも教育を提供できる点や、コンピュータならではの教材が利用できる点などが特徴である。e ラーニングは企業の社内研修などで用いられているほか、英会話学校などがインターネットを通じて教育サービスを提供している例などがある。e ラーニングシステムの利用者には、「学習者」と「教師」が想定されており、学習者用の機能と、教師用の機能は異なっている。また、多くの e ラーニングシステムには、e ラーニングシステムの「システム管理者」がおかれ、システム管理者によって、学習活動・教育活動に対する支援が行われる場合もある。自習システムの特殊な例として、コンピュータソフトウェアのチュートリアル機能があげられ、チュートリアルは画面の指示にしたがって操作などをしながら、ソフトウェアの使い方が学習できることを意図して作成されているものである。

2.2　STACK とは

STACK[3] とは System for Teaching and Assessment using a Computer algebra Kernel の略であり、e-learning システム Moodle のオンラインテスト（小テストモジュール）で、数式による解答を受け付け、学生が送信した数式による解答の正誤評価、自動採点可能にするシステムです。数学のためのオンラインテストシステムである。コンピューター教育でよく使われる、教師が用意した選択肢から正解を選ぶもの、解答を数値として要求する

ものなどがあるが、STACK では教師は学生に解答を数式として要求することが出来き、また、数式処理システムを利用してその正誤評価を行うことが出来る。STACK には、以下のように全てオープンソースソフトウェアで構成されている。

- Apache (Web サーバーソフト)
- PHP (スクリプト言語)
- MySQL (データベース)
- Maxima (数式処理システム)
- gnuplot (グラフを表示するソフト)
- Moodle (授業用の Web ページを作るためのソフト)
- jsMath (TEX の書式で入力された数式をブラウザに表示されるツール)

また、STACK には教師が問題を作成・管理する様々な機能がある。

- 係数を変化させて、同形式での数値の異る問題をランダムに自動作成できる。
- 数式入力, 二者択一など様々なタイプの問題を作成できる。
- 「ポテンシャル・レスポンス・ツリー」[3] を利用して、学生の解答に対して、種々のフィードバックを与えることができる。具体的には、正誤評価、コメント、採点結果である。
- 小問を含む問題を作成することができる。
- 解答に対して部分点を与えることができる。
- 問題に応じて図やグラフが動的に作成され、学生の解答をもとに、フィードバック中でも図やグラフを表示できる。

2.3 Maxima とは

STACK では様々な数式評価に、数式処理システム Maxima を利用している。Maxima は、Lisp で記述された数式処理システムである。微分、積分、テーラー級数展開、ラプラス変換、常微分方程式、集合、リスト、行列などを扱うことができ、STACK は、これらの機能を駆使することにより、様々な問題を作成することが可能となる。例えば「x^2+3x+2 を因数分解せよ」という問題の場合、正解は $(x+1)(x+2)$ であるが、もし、学生が $(x+2)(x+1)$

と入力したとき、それも正解である。あらゆる解答を準備しておき、そのどれかに一致した場合に正解にする方針となるが、困数が多くなるなど正解が複雑になってくると、正解の候補を全て準備することは容易ではない。一方「教師の解答 − 学生の解答 = 0」となるとき、学生の解答は正解であるという判定が出来れば、正解の候補を準備する必要がない。しかし、そのためには数式の代数処理が必要となる。

2.4 STACK を使わない理由

STACK は e-learning のシステムとして問題をランダムに自動作成するなどの有効な点もあるが、今回の目標である自習システムとして考えた場合、解答を入力してもらい、それを判定させる必要はなく、以下の点があるため、STACK は自習システムには不向きである。

- 汎用の e-learning システムでは必要であるが、全体を含めるとかなり重い。
- Maxima は自習システムならば、解答を事前に用意するため不要。
- 専用のシステムならば簡単な cgi でも提供できる。

3 使用するソフト

e-ランニングは、Web サイトを用いたものを考え、使用するソフトは Perl と LaTeX である。今回ウェブアプリケーションに優れている Perl を使用し、LaTeX ファイルとして問題を出力する。

3.1 Perl とは

Perl[1)] とはラリー・ウォールによって開発されたインタプリタ形式のプログラミング言語である。CGI を実現するためのプログラミング言語でもある。文字を扱う点で非常に優れている。また、コンパイルというコンピューターが理解できる機械言語に翻訳処理する作業が必要無く、プログラムがテキスト方式なので作成や実行、修正が簡単に出来る。C,awk,sed, シェルスクリプトなどのほとんど全ての機能を取り込んでいるため、それらの言語で出来ることで Perl で出来ないことはほとんどないが、一つのことを実現するのに何通りでも出来てしまうと言った「副作用」がある。なお、Perl は UNIX の他にも、Macintosh,Windows などでも動作できる移植性の高い言語である。

3.2 LaTeX とは

LaTeX とは、コンピュータ科学者のレスリー・ランポート (Leslie Lamport) によって機能強化された TeX である。TeX とは、コンピュータコンピューター科学者であるドナルド・クヌースにより作られた電子組版ソフトウェアである。LaTeX では文書の論理的な構造と視覚的なレイアウトを分けて考えるのが特徴である。長所として、

- 章・節・図・表・数式などの番号を自動で付け、参照箇所には番号やページを自動で挿入でき、目次・索引・引用文献の処理も自動でおこなえる。
- 文書の他に数式や作表のコマンドがあり、多用途での使用が可能である。
- テキストベースなのでデータの確認が簡単である。

などが挙げられる。しかし、コマンドが多数ありレイアウトが自由であることは熟練者には扱いやすいが、慣れていない者には難しい作業になるという欠点がある。

4 自習における学習の流れ

4.1 大学で学習する数学の内容

今回は、本大学での必修科目でもある基礎数理 I,II と数学演習 I,II に注目し、本学校の講義概要 [4] を参考にして、クラス別の講義で共通して学ぶ項目と、演習が必要な項目および、e-learning の需要がありそうな項目の選択を行う。

4.2 基礎数理 I, 数学演習 II

基礎数理 I は学科によっては必修科目であり、学科を問わず学習することが可能な科目である。また、クラス別に分かれており、内容は表 4.1 の通りである。

基礎数理 I で A クラスと B クラスの人は数学演習 I を受ける。 主に、基礎数理の内容にそった演習をする科目である。内容は表 4.2 の通りである。

4.3 基礎数理 II, 数学演習 II

基礎数理 I と同様に学科によっては必修科目であり、学科を問わず学習することが可能な科目である。講義はクラス別で表 4.3 の通りである。A クラスは微分法の公式が 2 回行

A クラス	B クラス	C クラス
学力試験	復習	高校の復習テスト
式と計算	三角関数	数列と極限，無限級数
2 次関数	累乗の一般化	関数と極限
2 次方程式	指数法則	導関数の定義と意味
2 次等式	対数法則	微分法の公式
ベクトル	中間試験	指数、対数関数とその微分
中間試験	微分係数関数	三角関数とその微分
三角関数	導関数	n 次導関数
三角関数のグラフ	不定積分	不定形の極限値
加法定理	定積分	関数の増減
指数法則	定期試験	試験
対数の性質		
微分法		
積分法		
期末試験		

Table 4.1 クラス別 基礎数理 I の内容

われる。対して、B クラスの講義は 2 回行う内容と 3 回に分けて行う内容が多かった。

数学演習 I と同じく、今回は基礎数理 II の演習を行う科目である。数学演習 II での A クラスの人が受ける科目である。内容は表 4.4 の通りである。

A クラス	B クラス
学力試験	学力試験
数と式の演習	指数関数とそのグラフ
式の展開と因数分解	弧度法と三角関数
式の変形と方程式	逆三角関数・極座標
指数計算	中間試験
座標、2 点間の距離	極限値
直線の式	微分係数
直線の式とグラフ	導関数
中 間 試 験	導関数の応用
2 次曲線（放物線）	定期試験
極大極小	
直線と 2 次曲線	
三角比	
弧度法と三角比の拡張	
三角関数のそのグラフ	
期 末 試 験	

Table 4.2 クラス別数学演習 I の内容

4.4 共通項目

基礎数理 I でのクラスで共通している項目は

- 三角関数 (グラフを含む)
- 指数、対数関数とその微分
- 導関数 (n 次導関数を含む)

である。基礎数理 II でのクラスで共通している項目は

- 微分及び導関数の復習
- テイラー展開（テイラー級数）
- 積分（不定積分、定積分）

であり、数学演習でも同じ内容が行われており、時間を 2 時間も使っている。また、演習科目で基礎数理 I で学習した微分と積分にまた時間を割いている。

A クラス	B クラス
関数と極限値	微分の復習
微分係数と導関数	テイラー展開
微分法の公式	極座標
様々な関数の導関数	積分の定義
高階導関数	基本的な積分の計算
テイラー級数	置換積分法
近似式	部分積分法
関数の値の変化	有理関数の積分法
中間試験	定積分
不定積分	試験
不定積分の計算	
定積分	
定積分の計算	
面積・体積	
曲線の長さ	
定期試験	

Table 4.3 クラス別基礎数理 II の内容

A クラス
指数・対数の復習
三角関数の復習
極限
導関数
微分の計算
微分法の応用
中間試験
不定積分
積分の計算
定積分
積分法の応用
定期試験

Table 4.4 数学演習 II の内容

4.5 自習で提供するもの

自習をする上で、問題と解答が必要である。今回自習をするための流れは、

1. 問題を用意する。
2. 問題を解く。
3. 問題の答え合わせをする。

だが、自習の場合、問題を解き、解いた解答と答えを照らし合わせる時、解答をパソコンから入力するのではなく、答えを作成したページと照らし合わせ学生自ら自己採点をする形が効率がよいと思われる。
そこで、Web ページを利用し、自習をする人に問題を提供し、解法と解答も同時に提供する形のプログラムを作成する。また、値などを動的に変化できるものを提供出来れば自習を多くおこなえ、問題作成者の負担が増えずに済み、便利であると考えた。今回そのプログラムの問題を作成する部分について考察する。また、問題を作成する部分のみなので、HTML に変換する必要はないため LaTeX ファイルを作成し、コンパイルする方法を今回はおこなっている。

5 LaTeX を主としたプログラム

Perl を用いて問題を動的に変化させるためのプログラムを作成した。これは問題を出題する人が主に LaTeX が出来る人を対象にしたプログラムであり、問題を変更する場合に Perl のソースを変えずに問題の変更を可能とするために作ったものである。今後 Web ページにするために、複数人が使用できるように時間も取り入れている。

5.1 基本構成

Perl のソースおよび構成は以下の通りである。

- ranmon.pl – 全体を管理する部分
- monsaku.pl –問題の作成をする部分
- ./bin/ –問題作成をおこなう部分および LaTeX ファイルが格納されているディレクトリ
- ./texmon/ –ベースとなる問題の LaTeX ソースファイルを格納するディレクトリ
- ./cash/ –問題作成が完了した LaTeX ファイル コンパイルしたファイルが格納されるディレクトリ

読み込まれるファイルおよび作成されるファイルは以下のとおりである。*は実行時間。bibun-mon.tex の問題以外の tex ファイルとは、タイトルや問題文を記載しているファイルであり、式の雛形ファイルと別している。理由は値を変更するときに、文章を間違って値に変更されないようにするためである。

- lis*.dat -使用する問題の番号
- bibun-mon.tex -問題の式以外の LaTeX ファイル
- bibun-1.tex -問題の式 (値が変わる前) の雛形ファイル
- *.dat -使われた値が入るファイル
- mondai*.tex -問題文と値が入った式の LaTeX ファイル

5.2 参照する LaTeX ファイルについて

今回の微分の問題においては、上記に示した bibun-mon.tex,bibun-1.tex を使っており、以下のような内容が入っている。

bibun-mon.tex

```
\documentclass{jarticle}
\begin{document}
\section{微分の基礎}
\subsection{導関数}
\begin{enumerate}
\item 次の関数を微分せよ。
\end{enumerate}

&mondai

\end{document}
```

bibun-1.tex

```
[1]\[y=x^&ma-&lax^&ma+&lax-&la\]
[2]\[y=(&lax+&la)^&ma \]
[3]\[y=(x^&ma+&lax-&la)^&ma \]
[4]\[y=\frac{&lax}{&lax^&ma+&la} \]
[5]\[y=\sqrt[&ma]{\mathstrut x^&ma} \]
[6]\[y=\frac{1}{\sqrt[&ma]{(x+&ma)^&ma}} \]
[7]\[y=\frac{1}{1+\sqrt{x^&ma}} \]
[8]\[y=\sqrt{(x+&ma)(x-&ma)} \]
[9]\[y=\frac{x}{\sqrt{x^&ma+&ma}} \]
[10]\[y=\sqrt{\frac{&ma-x}{&ma+x}} \]
```

Perl のソースはまず、ランダムで問題番号を選択し、「bibun-1.tex」の内容にある問題番号の式を読み取る。次に、bibun-1.tex ファイルの式の中にある&ma,&la&ma,&na,&oa を認識し書きの表 5.1 の通りにランダムを生成し、その部分に生成したランダムの値に書き換える。 式の値の変更が終了すると、次に「bibun-mon.tex」の&mondai を見つけ出し、そこにさきほど作成した問題を入れて、「mon*.tex」を作成する。

&ka	0 から 1000
&la	0 から 100
&ma	0 から 10
&na	10 から 100
&oa	100 から 1000

Table 5.1 ランダムの範囲対応表

以下は bibun-1.tex の問題全てに値を入力し、数式に変換した場合の参考例である。

(1) $y = x^3 - 4x^2 + 3x - 2$ (2) $y = (2x+3)^3$

(3) $y = (x^2 + 2x - 1)^3$ (4) $y = \frac{3x}{2x^2+1}$

(5) $y = \sqrt[4]{x^3}$ (6) $y = \frac{1}{\sqrt[3]{(x+1)^2}}$

(7) $y = \frac{1}{1+\sqrt{x}}$ (8) $y = \sqrt{(x+1)(x-2)}$

(9) $y = \frac{x}{\sqrt{x^2+1}}$ (10) $y = \sqrt{\frac{1-x}{1+x}}$

5.3 プログラムの構成、フローチャート

流れは、以下のとおりである。フローチャートは図 5.1 を参照。

1. 時間の取得
   プログラムの起動した時間を読み込む。作成される、ログファイルおよび、問題のファイル名としても使われる。

2. 式の選択
   今回使用する式をランダムで選び、選んだ番号を (lis*.dat) に書き込む。

3. 問題文の雛形の読み込み
   問題文の雛形 (bibun-mon.tex) を読み込む。

4. 式の雛形を選択
   (bibun-1.tex) から (lis*.dat) に格納されている番号とに対応する式を読み込む。

5. ランダムの値を作成
   問題に値を入力するためにランダムで値を生成し、配列に格納する。

6. 雛形の式に値を振り分ける。

```
[1]\[y=x^&ma-&lax^&ma+&lax-&la\]
```

の式が選択された場合、Perl で&ma,&la などがある場所を見つけ、書き換える値の配列を指定する。

7. 振り分けられた部分に値を入力
さきほど指定された配列の値と&ma,&la などを

```
[1]\[y=x^4-43x^2+64x-36\]
```

のように書き換える。

8. 入力した値の出力
入力された値は (*.dat) へも入力される。

9. 問題を作成し LaTeX を作成、コンパイル
作成された問題は (mon*.tex) となり、同時にコンパイルされるようになっており、さきほどのような式の場合は以下のように表示される。

$$[1]y = x^4 - 43x^2 + 64x - 36$$

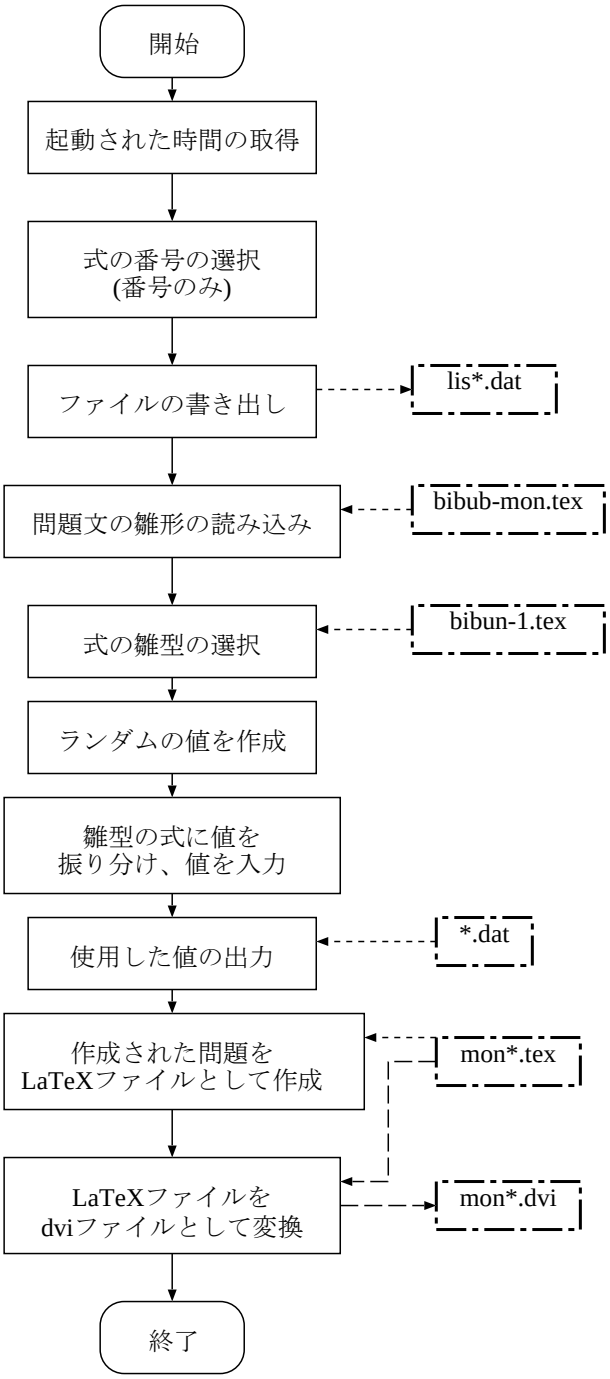

Fig. 5.1 LaTeX を主としたフローチャート

5.4 作成した問題の考察

以下は、今回作成したプログラムで作成した問題である。[] の番号は、前節で説明した bibun-1.tex から使用された式の番号である。

- 1 回目

1. 次の関数を微分せよ。

[5]

$$y = \sqrt[3]{x^6}$$

[4]

$$y = \frac{3x}{86x^7 + 51}$$

[3]

$$y = (x^8 + 29x - 1)^1$$

[4]

$$y = \frac{86x}{94x^1 + 13}$$

[8]

$$y = \sqrt{(x+1)(x-3)}$$

- 2 回目

1. 次の関数を微分せよ。

[10]

$$y = \sqrt{\frac{1-x}{2+x}}$$

[8]

$$y = \sqrt{(x+8)(x-6)}$$

[1]

$$y = x^9 - 64x^8 + 6x - 10$$

[8]

$$y = \sqrt{(x+8)(x-3)}$$

[2]

$$y = (11x + 86)^7$$

問題の式の値が変化可能になり、動的に問題の作成には成功したが、以下のような問題がある。

1. 平方根が取れてしまい意図した解法方法で解けない場合がある。
2. 問題の中に 1 乗が出てしまう。
3. 公約数を考えていない。
4. 値が大きすぎる (又は小さすぎる) 場合がある。
5. 値のみが変化したため、係数の変化に乏しい。
6. 入力した値の管理が難しい。

1 点目の問題点は、bibun-1.tex の [5] の問題の場合、以下のような式と解法が想定される。

$$y = \sqrt[4]{x^3}$$

この場合は x の分数乗であり、

$$(\sqrt[4]{x^3})' = (x^{\frac{3}{4}})' = \frac{3}{4}x^{-\frac{1}{4}} = \frac{3}{4\sqrt[4]{x}}$$

このように計算してもらうことを意図している。しかし 1 回目の [5] の問題は

$$y = \sqrt[3]{x^6}$$

である、よって解法は、

$$(\sqrt[3]{x^6})' = (x^2)' = 2x$$

も可能になってしまい分数乗の形に直す必要がなくなってしまう。

2 点目の問題点は 1 回目の [3] のように、$y = (x^8 + 29x - 1)^1$ は、$y = x^8 + 29x - 1$ となり、[1] の解答方法と同一になる。

3 点目と 4 点目の問題点は、1 回目の [4],[5] の問題において、例えば、

$$y = \sqrt[6]{x^4}$$

の場合、$\sqrt[6]{x^4}$ の 6 と 4 はお互い 2 の公約数を持っているため、3 と 2 にしてから、

$$y = \sqrt[3]{x^2}$$

として値を小さくするべきである。

また、[4] の問題において、以下の問題

$$y = \frac{5x}{5x^2 + 5}$$

を出題された場合、5 の公約数をもっているため、

$$y = \frac{x}{x^2 + 1}$$

となり、難易度が変わってしまう。

5 点目の問題点は符号が + なら +,− なら − と、値以外の変化をしていない。問題の式を見ただけで答えが出てしまう場合がある。

6 点目は、生成された値は lis*.dat の中に格納されているが値を入れているだけであり、何処の式の何処の場所に何の値を入れたかが不明であり、今後解答や解法を作る際に管理がしにくい。

よって、出題者側の意図した解法で解かれずに答えが出てしまい、自動的に作成される解答が普通の正答にはならず、想定した解答とは別解が存在する、問題が易しくなる、などの問題がおきてしまう。改善点として以下のものが挙げられる。

- 約分が出来ないように、個々に偶数と奇数の指定。
- 公約数を判定させる。
- 個々の数値の大きさ (適正値) を指定し、問題毎に値を生成させる。
- +,− といった符号に変化をつけさせる。
- 生成された乱数の値は、プログラム内で管理させる。

といった改善をする必要がある。これら多くの問題は、値を先に生成させ、それを式の値に割り振るといった方法が良くなかったのではと思われる。

6　Perl を主としたプログラム

6.1　前回から変更された内容

前章の場合は問題に使われる値は先に値を生成、格納してから、値を割り振っていたため、細かい値の指定が出来なかった。なので LaTeX を主としたものを、Perl で問題の使用したい場所に値を指定し生成、格納する方法に作り直し、プログラム内で値などの管理をおこない、問題と値の調整をできるようにした。また、今後解答を作成する場合、どの式のどの部分の値なのかが認識でき、解答を作成する場合においても有効だと思われるためである。前章同様、問題作成の部分のみの考察のため HTML に変換せず、LaTeX として出力している。

- ランダムの値を作成するとき、偶数と奇数を必要に応じて変更
  乱数を生成するサブルーチンを作り、偶数なのか、奇数なのか、どちらでもかまわないのかをそのサブルーチンに渡し、値に数偶、奇数の指定があれば、範囲の最大値は半分にし、乱数を生成、生成された値を 2 倍にして値を偶数にする。奇数なら、それに加えて、−1 をして奇数にするようになっている。偶数、奇数の指定がない場合はそのまま乱数を発生させている。
- 値の適正値を設ける
  先程のサブルーチンに、入力する値の範囲を指定できるようにし、細かく範囲を指定できるようになっている。
- +や-の変化に対応
  符号を付ける場所に符号を選択させるサブルーチンを用意した。これは 2 までのランダムを発生し、それが 1 以下なら −、1 以上なら + が選択されるようになっている
- 2 数が公約数を持つかどうか公約数をだすためにユークリッドの互除法 [5] を使用した。ここで、答えが 1 になればこの 2 つの値は公約数を 1 以外は持たないことになる。

6.2　ユークリッドの互除法

ユークリッドの互除法 [5] は 2 つの自然数 x，y の最大公約数を効率的に計算する方法であり、a と b の 2 つの数 ($a > b$ とする) の最大公約数を求める場合、a を b で割った余りを求め、これを b' とする。次に、b' を b で割った余りを求め、 b'' とする。さらに、b' を b'' で割った余りを求め…と交互に相手を割った余りを割っていき、余りが 0 になった時点の除数 (割る方の数、小さい方の数) が最大公約数になるというものである。例えば、100 と 42 の公約数をだすためには、以下のようになる。 この余りが 0 になったときの答

えが最小公約数である。そして、答えが 0 であれば 2 つの値は公約数を持っていないことになる。

ユークリッドの互除法の例

$100 \div 42 = 2 \cdots 16$

$46 \div 16 = 2 \cdots 10$

$16 \div 10 = 1 \cdots 6$

$10 \div 6 = 1 \cdots 4$

$6 \div 4 = 1 \cdots 2$

$4 \div 2 = 2 \cdots 0$

よって、最大公約数は 2 となる。

6.3 基本構成

このプログラムは、問題の作成を終わらせてから LaTeX ファイルを作成している。ディレクトリとソースは以下に示す。

- ranmon2.pl –プログラム本体
- ./cash/ –作成されたファイルを格納するディレクトリ
- mon*.tex –作成した問題の tex ファイル
- mon*.dvi –作成された tex ファイルを dvi に変換したもの

問題文、および問題の式は前章の (bibun-mon.tex),(bibun-1.tex) の内容を使用しているが、Perl のソース内に入っており、LaTeX ファイルは読み込まない。また、プログラムを起動するうえでオプションを設定し、表 6.1 のように問題の数と難易度を設定できるようにした。これは問題の数を自習をする数や範囲を調整可能にしている。問題数は 10 から 50 まで可能にし、難易度は 4 段階設け、その範囲は表 6.2 に示す。

-d	デフォルト	問題数：10 難易度:全て
-r	ランダム	問題数:10 から 50 難易度 6 段階
-o	回数を指定	問題数の指定 1 から 50 まで
-l	レベルの指定	難易度 0 から 6 段階まで

Table 6.1 オプション内容

難易度	問題範囲
0	範囲全て
1	(1) から (3) までの範囲
2	(1) から (7) までの範囲
3	(3) から (10) までの範囲
4	(5) から (10) までの範囲
5	(7) から (10) までの範囲

Table 6.2 レベルの構成

6.4 プログラムの構成、フローチャート

流れは以下のとおりである。フローチャートは図 6.1 を参照。

1. オプションの判定および設定
   問題の数、および難易度の設定を読み込む。。
2. mon*.tex に問題文を入力
   前章の bibun-mon.tex の内容が入る。
3. 式の選択
   難易度の範囲から式を選択する。
4. 指定された数まで問題を作成
   指定された数まで 5,6,7 を繰り返す。
5. 式の選択
   難易度の範囲から式を選択する。
6. 値の作成
   指定された値、符号を生成する。
7. 問題を mon*.tex に入力
   式に値を加え、それを LaTeX ファイルとして出力する。

8. mon*.tex を dvi ファイルに変換
   pLaTeX を使い、dvi に変換を行なう。

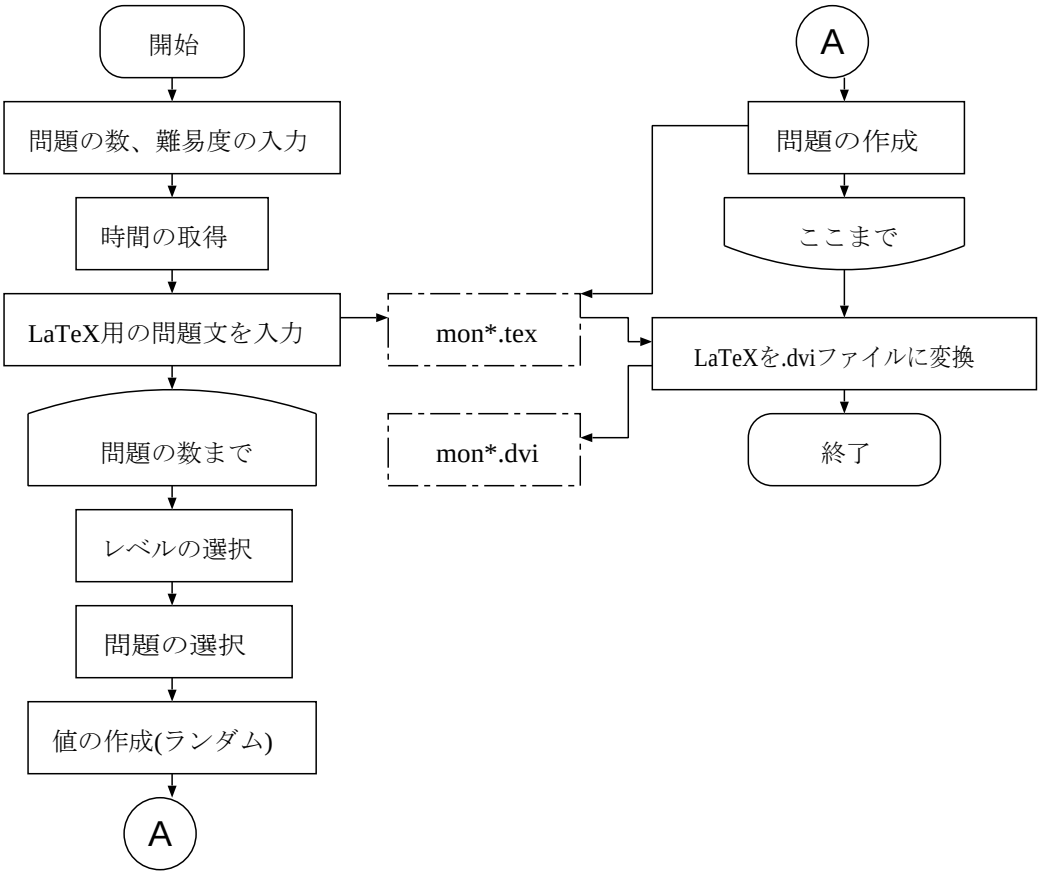

Fig. 6.1 Perl を主としたフローチャート

プログラムの問題式の作成をするソースは以下のとおりである。[1] の問題の場合、乱数を使用する配列 (@mon1),(@monbin1) に乱数を格納し、必要であれば、その配列の値を調整する。その例が [5] のソース内の (&koya) というサブルーチンである。そこには 2 つの値の最大公約数を出すユークリッド互除法を使用しており、1 以外の値が返ってきた場合は公約数を持っているので乱数を格納し直し、1 が返ってくるまで乱数を格納する。また、そのときに 2 つの値の大小を固定するようにしている。平方根に使われる値が $\frac{1}{2}$ 乗であれば、根号で表現するときに 2 を省略している。そして配列 (&ca1) には符号 +,− をランダムに格納される。値の調整、格納が終了すると、その式を print で LaTeX ファイルとして出力するようになっている。

[1] の問題作成をする部分のソース

```
if($a == 1 ){
        for($i=0;$i<5;$i++){
            $mon1[$i] =  &ranpon(10,1,0);
            $monbin1[$i] = &ranpon(5,1,0);
        }
        for($i=0;$i<3;$i++){
            $ca1[$i] =&pm;
        }
        #\[y=x^3-4x^2+3x-2\]
        print (OUT "\\[y=x^$monbin1[0]$ca1[0]$mon1[0]x^$monbin1[1]
        +$mon1[1]x$ca1[1]$monbin1[2]\\]\n");
}
```

[5] の問題作成をする部分のソース

```
if($a == 5 ){
        local$fa;
        $mon5[0] =  &ranpon(10,2,0);
        $monbin5[0] = &ranpon(10,2,0);
        $fa = &koya($mon5[0],$mon5[0]);
        until( $fa = 1){
            while($mon5[0]<$monbin5[0]){
                $monbin5[0]= &ranpon(10,2,0);
            }
            $fa = &koya(
                $mon5[0],$mon5[0]);
            }
        if ($mon5[0] == $monbin5[0]){
                $monbin5[0] = $monbin[0]+1;
        }
        if( $mon5[0] == 2){
                $mon5[0] = " ";
        }
        #\[y=\sqrt[4]{\mathstrut x^3} \]
 print (OUT "\\[y=\\sqrt[$mon5[0]]{\\mathstrut x^$monbin5[0]} \\]\n");
}
```

6.5 作成した問題の考察

今回作成したプログラムで問題を作成した。前章の LaTeX を主としたプログラムで起きていた問題点、改良点を加え、公約数を考慮した。下記のように、[5] をいくつか作成したが、平方根が取れないようになっており、$\frac{1}{2}$ 乗のような問題には根号を使用するとき、2 を省略して表示されているのがわかる。また、問題を抽選したとき の (1) と (2) は両方 [1] の問題を使用しているが、+ と − の符号が変化している。しかし、今回のプログラムは LaTeX を意識しなくても出来るようになっている訳ではなく、Perl の知識が必要である。なぜなら「[1] の問題作成をする部分のソース」から $y = x^3 - 4x^2 + 3x - 2$ などを作成する場合、print をする部分には LaTeX の数式モードのソースを入力し、その値や、符号を書き換えて LaTeX ファイルを作成しているため、LaTeX と Perl の両方を知らなくては問題を追加したり修正することは難しいからである。

- 公約数を考慮した [5] の例

1. 次の関数を微分せよ。

(1)

$$y = \sqrt[5]{x^6}$$

(2)

$$y = \sqrt{x^2}$$

(3)

$$y = \sqrt{x^5}$$

(4)

$$y = \sqrt[8]{x^9}$$

(5)

$$y = \sqrt[7]{x^4}$$

- 問題を抽選した場合

1. 次の関数を微分せよ。

(1)

$$y = (x^4 + 1x + 5)^2$$

(2)

$$y = (x^2 + 1x - 1)^4$$

(3)

$$y = \frac{1}{\sqrt[5]{(x+4)^2}}$$

(4)

$$y = x^3 - 7x^4 + 9x - 3$$

(5)

$$y = (2x + 1)^2$$

7　まとめ

本研究の最終目標は自習するための e-learnig システムの作成であるが、本稿では問題の作成について考察し、問題をどのように作成し、問題を作成する際の問題点について考察した。

問題の作成方法として LaTeX を主とした作成方法と Perl を主とした作成方法に関して考察したが、始めは LaTeX を主として問題を作成できれば出題者が容易に問題を作れると思い作成した。LaTeX を主とした場合、問題の追加、修正についてはプログラムを意識せずに行なえるが、微分などの値の調整が必要な問題に関してはその調整方法が難しくなり、さらに解答や解法を作成する場合はどこにどの値が使われたかの管理が難しいため、プログラムを主体とする Perl を主とした方法をとるようにした。そのため、Perl を主とした場合においては、使用した値がどこの場所に使用されたかが管理でき、解法や解答を作るときに有効である。また、LaTeX を主とした場合におこなえなかった、値の適正値を設けることができ、解法が異なることを防げるため、問題作成は Perl を主としておこなうのが有効な手段である。

しかし、Perl を主とした場合の問題点は、Perl 内のソースに LaTeX のソースを入れる方式なので、LaTeX と Perl の両方を知っている必要があることである。値や符号の変化はプログラム内で処理しているため、LaTeX を意識することを少なくする方法を考えるか、数学の式が表現出来るものを見つけることが今後の問題作成の課題である。

また、今回は問題の作成の考察のみに終わってしまったが、解答と解法を提示するプログラムの作成についても今後の課題である。

参考文献

[1] Perl の基礎入門
http://www.kent-web.com/perl/

[2] e-ラーニング -Wikipedia
http://ja.wikipedia.org/wiki/E%E3%83%A9%E3%83%BC%E3%83%8B
%E3%83%B3%E3%82%B0

[3] 中村恭之：数学 e-ラーニング (東京電機大学出版局,2010)

[4] 講義概要｜新潟工科大学
http://www.niit.ac.jp/syllabus/2010/102/index.html

[5] 拡張ユークリッド互除法
http://www2.cc.niigata-u.ac.jp/∼takeuchi/tbasic/BackGround/ExEuclid.html

[6] ユークリッドの互除法とは - 意味/解説/説明/定義 ： IT 用語辞典
http://e-words.jp/w/E383A6E383BCE382AFE383AAE38383E38389E3
81AEE4BA92E999A4E6B395.html