The 32nd Annual Conference of the Japanese Society for Artificial Intelligence, 2018

解答解説機能付き微積分ソルバの開発

Development of *Calculus Solver* software which provides solving hints and answer steps

大塚　基広 *1　　平　博順 *1　　真貝　寿明 *1
Motohiro Otsuka　　Hirotoshi Taira　　Hisaaki Shinkai

大阪工業大学大学院 情報科学研究科
Faculty of Information Science & Technology, Osaka Institute of Technology

We develop a python-based software which provides hints and answers for freshmen-level calculus interactively. Our solver covers multi-variable differentiations, Taylor expansions, graph drawings, and single-variable definite/indefinite integrations. A user can input a problem using GUI interface, and the output comes as a LaTeX-typed pdf. We find that differentiation processes can be coded with four tasks, while integration requires more. In order to prepare "readable" answers, we have to adjust output expressions. Although this software cannot handle some problems, the provided answers pass tests as the grade B to D, so that it helps freshmen for their self-education.

1. はじめに

これまでまでに Mathematica，Maple，Matlab といった優れた数式処理ソフトウェアが開発され，利用されてきている．数式処理ソフトウェアは，人間が手計算で行なっていたものを自動で計算してくれることから，様々な研究開発の分野で，その研究開発スピードを高めている．また，多くの数式処理ソフトウェアでは，様々な可視化機能が付属しており，ユーザに処理結果を分かりやすく提示することができる．

このような可視化機能を利用して，大学における数学教育で数式処理ソフトウェアを導入することで，微積分や線形代数における基本的な概念を学生に理解させやすくする試みはこれまでいくつかの報告がある．例えば木村は，数式処理ソフト Sage を学部生の線形代数の授業内で利用し，ベクトルの線形変換の様子などを可視化することで，数式だけだとイメージしにくい線形変換などの線形代数の概念を学生に理解させる上で一定の効果をあげている [1].

また，中島らは，数式処理ソフト Maple を微積分の講義で利用し，2 変数関数のグラフ，接平面，グラフの極値と鞍点，2 重積分などを可視化することで学生の数式や数式処理について具体的なイメージを持たせる試みを行なっている [2].

以上は，概念の可視化を通じて，学生の理解を助ける取り組みである．他方，微積分や線形代数の計算自体に習熟させるために，効率的な e-learning 教材の利用も提案されている．例えば中村は，STACK と呼ばれる数学解答評価システムを開発し，問題の数式の係数などをランダムな値として類似問題を自動生成することで，教員側が大きな労力をかけなくても，学生が繰り返し計算問題に挑戦できる仕組みを作っている [3]. ただし，この取り組みでは扱える問題が係数が異なる程度の類題にとどまっている．

一方，先に述べた数式処理ソフトでは，入力された関数や関数に対する微積分などの処理や可視化は簡単に行なえるものの，途中の数式処理については提示されず，またその解き方について解説してくれるわけでもないので，入力されたものに対する結果しかユーザは得ることが出来ない．

また,「ロボットは東大に入れるか」プロジェクト（東ロボ）では，東大に合格できるような試験問題ソルバが開発されており，センター試験の数学の模擬試験では偏差値が 70 を越えるようなソルバが開発されている．しかしながら，これらのソルバの出力する解答は数学的に正しい解答であっても，普通の受験生が書くような答案とは大きくかけ離れている．問題文に対して形態素解析，係り受け解析，照応解析などを行なったあと，量化記号を含む論理式を立式し，限量記号消去法（Quantifier Elimination 法; QE 法）を用いて計算処理が行なわれている．この結果を用いて解答を生成しているため，受験生が授業や参考書などで学ぶ処理方法とはかけ離れており，学生にとって参考になる解答とはいえない．

受験生が解くような解き方で入力された数学の問題を解くソフトウェアとして，フリーウェア・フリーのアプリとして，Mathway や Microsoft 社の Math 3.0，Photomath などがある．これらは，中学校レベルの数学の問題では途中の解き方を出力できるが，大学レベルの微積分の問題では，基本公式に当てはまる問題程度しか解答を出力できず，置換積分のような問題については解き方を出力することはできない．

そこで，本研究では大学の数学教育において，学生が自ら微積分などの計算問題を入力し，置換積分や部分積分など高度な解き方を含む問題についても，自動的にその解答と解説を生成することで，学生が多様な計算について興味を持ち，自分の解き方も確認できるような数式処理ソフトウェアの開発を試みた．

開発したシステムについて，大学の微積分の講義における期末試験問題の過去問についてどの程度正しく，解答と解説が自動生成できるかについて評価したところ，3 年分の試験で試験の解答として 70%が正しいものが出力されることが確認できた．

2. 実装へのアプローチ

2.1 微分に関して

例えば関数 $f(x) = x\cos(2x)$ を微分するとき，人間の思考では次のような手順で考える．

1. 関数 $f(x)$ は 2 つ関数の積 $f(x) = g(x)\,h(x)$ からできている．

連絡先:　大塚　基広，大阪工業大学情報科学部,m18.aiandnip.m.o@gmail.com

2. 積の微分公式を用いることになる．

$$f'(x) = g'(x)\,h(x) + g(x)\,h'(x)$$

3. $g(x) = x$, $h(x) = \cos(2x)$ と置く．

4. $h(x)$ は合成関数である．$h_1(x) = 2x$ とおいて，$h(x) = \cos(h_1(x))$ となる．

5. 合成関数の微分公式を用いることになる．

$$h'(x) = \frac{\partial h(h_1(x))}{\partial (h_1(x))} \cdot \frac{\partial h_1(x)}{\partial x}$$

6. $h'(x) = 2\sin(2x)$ である．したがって，与式の微分は，

$$f'(x) = \cos(2x) - 2x\sin(2x).$$

以上の過程を，計算をするときに必要な部分と，解答を作るときに必要な部分をまとめて簡略化すると，次の 4 つのタスクにまとめることができる．

(A) 合成関数としての分離，ヒエラルキーの把握

(B) 定数・変数の把握

(C) 式全体の積の微分の演算結果の保存

(D) 1 つの演算単位での微分結果の保存

これらに加え，関数をすべて積の関数としてみなして扱うことにすれば，微分演算の手順をまとめることができる．

それぞれ 4 つのタスクを，上記の 6 つの手順と照らし合わせてみると，次のようになる．

1 は，認識のレベルでタスクの（B）に相当する．2 は，関数の積と解釈できればいいので，タスク（B）に相当する。3 は，公式を使用し，新たに関数を定義しなおしているので．タスクの（A），（C）に相当する。4 は，同様にして，（A），（C）に相当する．5 は，定数と変数が把握できれば良いので，タスクの（B）である．6 はすべての結果を合わせるのですべてのタスクに相当する．

このようにして，合成関数と積関数が入り混じった計算でも，この 4 つの情報を保持できれば解答を付けることが可能になる．

2.2 積分について

積分も微分と同様に上記 4 つのタスクを使用すれば，ある程度計算はできる．しかし，積分演算では，部分積分・置換積分・漸化式を用いる解法・部分分数分解を必要とする積分などさまざまな技法がある．また，解析的には計算できない積分もある．したがって，それらを個別に実装する必要があった．

本研究で実装できなかったものに関しては，解答のヒントを示すようにしている．なお, 部分積分 (漸化式解放を除く) と，部分分数分解を使用する計算に関しては実装ができた．部分分数分解を必要とする積分は sympy の apart を使用した．

積の関数を積分するときには，部分積分

$$\int g'(x)\,h(x)\,dx = g(x)\,h(x) - \int g(x)\,h'(x)\,dx$$

を用いることになるが，どちらを先に積分して $g'(x)$ としておくのか，という点については，必ずしも一意ではない．場合によってはできないこともあるし，一方に三角関数があれば，部分積分を 2 回行う手法もある．

以下に擬似コードを示す．

擬似コード

```
str = ’xCos_{x}’ #input
list = str.split(’*’) #list=[x,Cos_{x}]
if(list[0] == 三角関数):
    if(list[1] == 三角関数):
        print 漸化式解法（未実装）
    else:
        list[0] を h(x), list[1] を g(x) として部分積分
else:
    list[0] を g(x), list[1] を h(x) として部分積分
```

3. ソフトウェアの概要

GUI での操作ができるようにした．起動すると，まず，計算するカテゴリー（微分・積分・Taylor 展開）を選択する画面になる (図 1)．

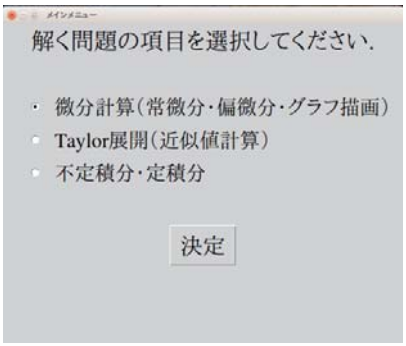

図 1: GUI のメイン画面

決定ボタンを押すと，式や条件などの入力画面になる．入力の方法は，入力例ボタンをクリックすると表示される．ヒントや解説を表示するかどうか，グラフを描く場合の範囲の指定，級数展開の場合は展開次数などの選択項目を指定すると，計算が実行される．処理のフローを図 2 に示す．

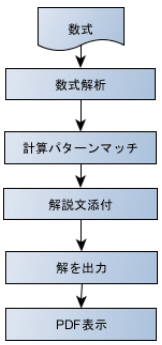

図 2: 処理のフロー

3.1 式の入力方法に関する工夫

ここで入力インタフェイスについて簡単に述べる．入力の仕様は以下のようにした．

- 四則演算子
  乗算を表す記号（∗）は不要
- べき関数
  ^ を使用し，べきの部分は { } で囲む．
  例 `x^{2}` $\rightarrow x^2$
- 三角関数
  最初の文字を大文字にして，_ (下線) を付与した関数名とし，引数部分を { } で囲む．
  例 `Cos_{x}` $\rightarrow \cos(x)$
- 対数関数
  最初の文字を大文字にして，_ (下線) を付与した関数名とし，底と真数部を { } で囲む．底を省略すると，底を e として扱わる．
  例 `Log_{x}` $\rightarrow \log(x)$

このインタフェイスにした理由は、手書きに近く，利用者の多い Tex や Mathematica に似た入力仕様にしたかったためである．

例えば，$x \cdot \cos(2x)$ を微分するときには「`xCos_{ 2x }`」と入力し，微分する変数と，微分階数を指定することになる．

3.2 アルゴリズム

処理のアルゴリズムを図 3 で示す．上記の入力例は

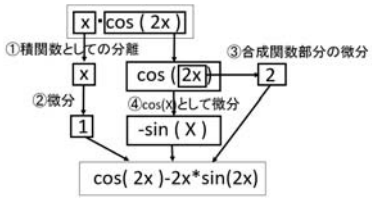

図 3: 処理のアルゴリズム

```
List [ [Cos,_,{,[2x],},],]
```

のように分解される．List の中身を探索することによって，[Cos,_,{,[2x],},] が出現する．この時点で入力された関数が，三角関数 cos と，その引数が $2x$ であることが分かり，合成関数であることがわかる．そこで，cos の処理を行うための関数に処理を渡す．その後，合成関数の処理をし計算結果を返す．結果を格納するリストを result と定義すると，

```
result[2,*,Sin,_,{,2x,}]
```

となる．Taylor 展開や偏微分の計算は，微分と同じものを使用し，その関数を繰り返し適応する計算を行うことになる．

3.3 出力に関する工夫

一般的に，PC で処理された数式は，∗ やがそのまま使用されていることが多い．また，整数で書くべきところが小数点付で表示することもある．出力される答案では，これらの記号を使わないように工夫した．

Taylor 展開を表示する場合，例えば，

$$e^x = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \cdots$$

のような表示をしたいとき，展開係数は階乗記号で書かれる方が好ましい．通常の処理をしてしまうと，分母の階乗記号が計算されてしまう（例えば，$4! = 24$）．しかし，本プログラムでは，$\frac{1}{4!}$ とった形で出力されるように，n 階の簡略化された微分結果と，階乗の文字列を繋げるようにし，階乗記号が計算されないように工夫した．実際の出力を示す．

[問題]
微分する関数を $f(x)$ と定義する．

$$f(x) = x\cos(2x) \tag{1}$$

[基礎知識] ．積の微分公式

$$f(x) = g(x)h(x) \rightarrow f'(x) = g'(x)h(x) + g(x)h'(x) \tag{2}$$

商の微分公式

$$f(x) = \frac{g(x)}{h(x)} \rightarrow f'(x) = \frac{g'(x)h(x) - g(x)h'(x)}{h(x)^2} \tag{3}$$

合成関数の微分公式

$$f(x) = g(h(x)) \rightarrow f'(x) = \frac{\partial h(x)}{\partial x}\frac{\partial g(h(x))}{\partial (h(x))} \tag{4}$$

[解答]
関数 $f = \cos(2 \cdot x)$ は $f(x) = \cos(g(x))$ の形を取っている．すなわち合成関数の微分を行う必要がある．
この関数の場合 $g(x) = 2 \cdot x$ である．
$g(x)$ を合成関数として考えると $g'(x) = 2$ となる．
したがって関数関数 $f = \cos(2 \cdot x)$ の微分結果は $f' = -2 \cdot \sin(2 \cdot x)$ となる
以上より $f(x)$ を x で微分させた結果は次の通りになる．

$$f(x)_x = -2x\sin(2x) + \cos(2x) \tag{5}$$

3.4 グラフ出力に関する工夫

グラフは，python の matplotlib モジュールを用いて表示した．微分計算の答案を作成するにあたっては，微分して極大・極小を与える場所を求め，増減表を作成する必要がある．これらの各ステップを表示するようにプログラムする必要がある．

また，グラフの描画域も整数値で指定できるようにして，ユーザの答案作成の視点に近づけた．また，解答にグラフが記述される前に，どのようなグラフになるか見られるプレビュー機能も作成した．この機能により，ユーザはグラフを何度も調整することができる．

4. ソルバの性能

4.1 対象のテストの結果

このソルバを使用し，本学の微積分学 I の期末試験を解かせた結果を示す．なお，過去問は Web[*1] 後悔されてている．

表 1: 過去問の大問ごとの採点結果

年度	大問 1 微分	大問 2 積分	大問 3 Taylor 展開	大問 4 偏微分	合計
配点	30 点	30 点	20 点	20 点	100 点
2017 年度	25 点	25 点	17 点	14 点	81 点
2016 年度	20 点	13 点	11 点	14 点	59 点
2015 年度	25 点	19 点	12 点	14 点	70 点

なお，この試験では 100 点スコアがつくとともに，A〜D（合格），F（不合格）の 5 段階評価がおこわなれる．5 段階評価の基準は以下の通りである．

D 判定　大問 1（微分とグラフ，ライプニッツ）と大問 2（積分と，求積問題）の得点の和が一定以上

C 判定　D 判定の判定と同条件（ボーダーラインが上がる）

∗1 `http://www.oit.ac.jp/is/~shinkai/lecture/`

B 判定　C 判定を取得できる者のうち，大問 3（Taylor 展開）の得点が一定以上

A 判定　B 判定を取得できる者のうち，大問 4（偏微分）の得点が一定以上

この基準を元に，システムの出力について，微積分の講義の担当教員に評価してもらった結果，2017 年度は B，2016 年度は D，2015 年度は A という判定であった．

このように，合格はできるものの一定以上の点数が取れないのは，文章題や積分計算の一部がまだ解けないことが原因である．なかには「$(x^2+x)\sin x$ の n 階微分を求めよ（ライプニッツの公式を用いる）」のようにまだプログラミングが対応できていない問題もある．置換積分でどの部分を置換したらよいのかが正しく判定できない場合や，

- 「○○の回転体の表面積を求めよ」
- 「$(1-x)^n$ の級数展開を用いて，$\sqrt[3]{0.9}$ を小数第 3 位まで求めよ」

のような，計算だけでは求まらない問題があり，今後の課題である．

4.2　汎用的な問題を解いた結果

実際に，どの程度の範囲であるならば正しく解答が得られるのかを評価した．講義で使用されている教科書 [8] をであげられている項目の内，実装が完了した計算を表 2 に挙げる．

表 2: 実装できた計算ツール

微分	べき関数，指数関数，対数関数，三角関数，商関数，積関数，Taylor 展開，偏微分
積分	一変数のべきの関数，指数関数，対数関数，周期関数を 1 つだけ含む積の関数，特殊な置換積分，分母の微分が分子になる商関数，一部の部分分数分解可能な分数，定積分（一部除く）

現時点で対応できていない問題としては，以下のようなものがある．

1. 二重指数を含む微積分　例　$\dfrac{d}{dx}x^x$
   これは対数微分法で解けるものであるが，対数を取って微分をした後，再び対数を取って変数を元にもどす処理が複雑で実装が完了していない．
2. 複数回の部分積分が必要な積分
   例　$\int e^x \cdot \sin mx\, dx$
   部分積分自体は実装できているので，計算をしてくれるが無限ループに入る．そのループを止める基準である，「与式」と「部分積分しているうちに出てくる関数」が一致しているという判定が難しい．

4.3　解けない問題への対応

積分の問題は，解析的にできないものも多い．たとえば

$$\int \frac{1}{1-x^2}dx$$

は置換積分で解けるが，

$$\int \frac{1}{\sqrt{(1-x^2)(1-k^2x^2)}}dx$$

は，楕円積分の知識がないと解けない．そして，一般には解析で解けないものは級数解法か数値積分を行うことになる．

学生にとってのヒントとしては，できる限り，解ける道筋を示すべきだが，ソルバとしては『お手上げ』状態になることも避けられない．そのようなときは，「このソルバでは解けません．○○か○○の方法で解けるかもしれません.」あるいは「入力された関数は解析的に積分できない可能性があります.」などと表示させることにした．Mathematica などでも，解析的にできない積分はそのまま返されるが，それよりは親切な表示といえる．

5.　まとめと展望

本研究では，解答解説付き微積分ソルバを開発した．本ソフトウェアの特徴は，学生の答案作成の見本を示すように，答案として途中経過を表示させることができることである．なるべく，人間として自然な答案を示すために，計算結果を表示するだけではなく，分数の形式で表示をする，階乗記号は計算させない，整数は整数の形を保ったまま表示するなどの工夫が必要であった．

実装する過程で，微分の計算は 4 つのタスクと，積の関数として微分することで大半の問題が解けることが分かった．しかし，二重指数の問題や，有理化が必要な計算に関しては，別ルーチンを組む必要があった．一方で，積分では，4 つのタスクだけでは解ける問題が少なく，部分積分，置換積分，漸化式解法，商の積分などを別ルーチンで組む必要があった．

量化記号が必要になるほどの論理が必要な問題には，まだ対応できていない．今後の課題である．

参考文献

[1] 木村　巌,「free な数式処理ソフト Sage の学部教育での活用事例」，数理解析研究所講究録第 1865 巻 p101-109（2013）

[2] 中島匠一，水谷明,「数式処理ソフト Maple による微積分の教育支援」計算機センター Vol.31(2010)

[3] 中村泰之,「数学 e ラーニング　数学解答評価システム STACK と Moodle による理工系教育」

[4] 伊藤巧，松崎拓也，佐藤理史，名古屋大学大学院工学研究科,「数学問題テキストにおけるさまざまな照応現象の解析」　人工知能学会全国大会（2017）

[5] 岩根秀直，穴井宏和「数式処理による入試問題への挑戦」，FUJITSU(2015)

[6] 岩根秀直，松崎拓也，穴井宏和，新井紀子，辻通研究所，国立情報学研究所，九州大学,「数式処理による入試問題の解法と言語処理との接合点における課題」　人工知能学会全国大会 (2013)

[7] 長岡亮介,「総合研究論理学で学ぶ数学-思考ツールとしてのロジック」,2017 年　旺文社,

[8] 真貝寿明,「徹底攻略　微分積分」　2013 年　共立出版