数式処理 *Bulletin of JSSAC(2012)*
*Vol. 18, No. 2, pp. 151 - 160*

特集: 数式処理システム 〜いま使うならこれだ〜

# REDUCE のすすめ

加古富志雄*

奈良女子大学理学部

**概　　要**

REDUCE は A.C.Hearn 博士が作成した汎用の数式処理システムである．システムは Lisp 言語で書かれており，システムのソースを含めて配布されてきているので，利用者がシステムの機能を拡張することが比較的容易にできる．

## 1 REDUCE とは

REDUCE は，非常に複雑な代数計算を正確に実行するシステムである．このシステムは，多項式を展開した形式や因数分解した形式など多様な形式で扱うことができ，また，数式の一部を取り出すことが出来るし，微分計算や不定積分の計算も可能である．

REDUCE は A.C.Hearn 博士が素粒子物理学での数式計算を計算機で行わせるために作成された．その後，1970 年に，REDUCE のバージョン 2 が作成され，システムは RLISP と呼ばれる Algol に似た言語（LISP 言語の方言）で書き直された．このシステムは，多くの利用者に配布され，利用者のコミュニティが形成されていった．日本でも，多くの計算機センターでインストールされ利用されていた．当時，数式処理システムとしては，Macsyma や Scrachpad が存在していたが，日本で一般的に利用出来る汎用の数式処理システムとしては REDUCE 位しか無かった．

1983 年に REDUCE3 が現れ，不定積分計算，多変数多項式の因数分解，多倍長精度の実数計算や方程式の求解などの機能が追加されている．REDUCE2 までのシステムはほとんど Hearn 博士一人で作成されていたが，REDUCE3 からは多くの利用者がシステムの拡張に参加する様になって，netlib として，パッケージが配布される様になってきた．

REDUCE2 から，ソースを含めて配布されていたが，REDUCE3（3.3 版以降ぐらいから）オープンな形での配布は行われず，配布手数料としていくらかの金額を取る様になっていた．現在では，2008 年に完全にオープンソースとして，modified BSD ライセンスに基づき，自由に利用出来る様になっている．

REDUCE は，システム自体が Standard Lisp と呼ばれる Lisp 言語で作成されている．システムのソースを含めて配布されてきているので，利用者がシステムの機能を拡張して利用することが比較的容易にできる．

*kako@ics.nara-wu.ac.jp

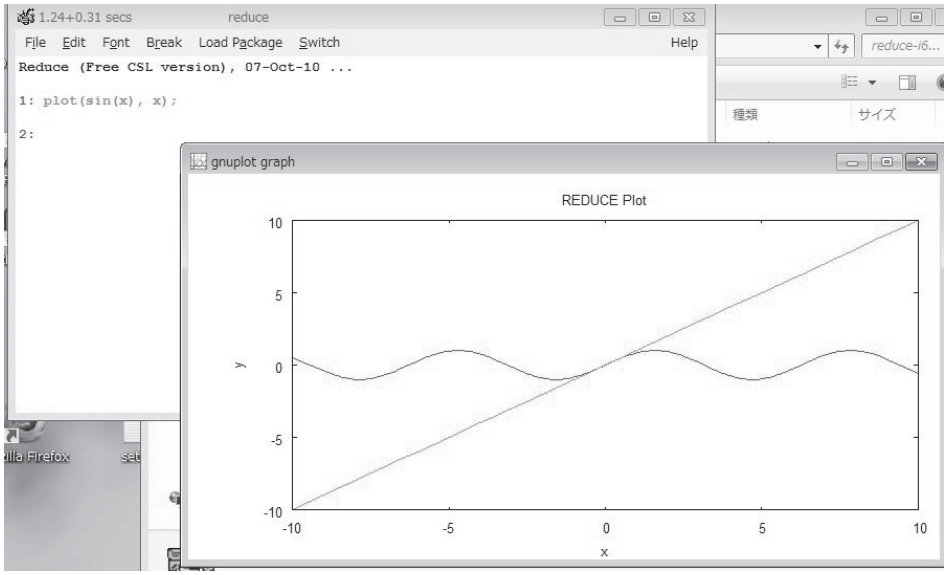
## 2 インストール方法

REDUCE は SourceForge 上のウェブサイト[1)]から入手することができる．コンパイルされたパッケージとして，Windows（32 ビットおよび 64 ビットシステム），Linux（32bit および 64bit Ubuntu 10.10 システム）および MacOS X 10.6 (Snow Leopard) 用のバイナリーファイルが置かれている．

### 2.1 Microsoft Windows

Microsoft Windows では，Windows 用のバイナリーファイル（reduce-windows32-20110414.zip（32 ビット版）と reduce-windows64-20110414.zip（64 ビット版））が用意されているので，これをダウンロードして展開すれば，インストールは終了する．reduce-windows32-20110414 のフォルダ中に実行ファイルがあるので，これをクリックして実行することで REDUCE が起動する．

Microsoft Windows 7 で動かした所では，64 ビット版がうまく動かなかった．gnuplot を使って，グラフを描画することが出来る様になっているのだが，その gnuplot を起動することに失敗しているようだ．32 ビット版を動かしたところでは，正しく動作した（下図参照）．




この REDUCE は CSL(Codemist Standard Lisp) の上で動いているのだが，64 ビット版ではメモリ管理にバグがあるようだ．64 ビット版で $10000 \times 10000$ の行列を作成すると

```
Reduce (Free CSL version), 06-Jan-12 ...

1: matrix m(10000, 10000);

Memory access violation detected
```

[1)] http://reduce-algebra.sourceforge.net

となって，ハングアップした．REDUCE では行列はリストで表現されているので，これで約 1.6G バイトぐらい (64 ビット版では) のメモリを使うはずだが，ちゃんと動かない．

32 ビット版では，使えるメモリは最大で 2G バイトで，10000×10000 の配列を二つ作成することができた．三つ目を作成しようとすると同じく Memory access violation detected のエラーを起こしてハングアップしてしまう．

### 2.2 Linux

Linux あるいは Unix では，ソースからコンパイルするのが簡単である．sourceforge.net には，ソースファイル reduce-src-20110414.tar.bz2 があるので，これをダウンロードして，展開する．あるいは，subversion が使えるならば，それを使って最新のソースを手に入れる方が良い．実際，

```
svn co https://reduce-algebra.svn.sourceforge.net/
           svnroot/reduce-algebra reduce-algebra
cd reduce-algebra/trunk
./configure --with-csl
make
```

で CSL 版の REDUCE が作成される (最初の 2 行は，実際は 1 行で入力する)．

REDUCE を実行するには，スクリプトファイルが reduce-algebra/trunk/bin にあるので

```
bin/redcsl
```

で起動する（使い方は，Windows 版と全く同じ)．

メモリに関しては，CSL 版は Windows と同じで，32 ビット版では，最大 2G バイトまで．64 ビット版では，大きなメモリを (1G バイトぐらい？) を使おうとするとハングアップしてしまう．

Linux あるいは Unix では，PSL(Portable Standard Lisp) を基にした PSL 版の REDUCE を使うことも出来る．最初に作成するときに，`--with-csl` の代わりに`--with-psl` を指定して，

```
./configure --with-psl
make
```

とすると，PSL 版の REDUCE が作成される．これを起動するには，

```
bin/redpsl
```

と入力する．PSL 版でも大きなメモリを使用しようとするとエラーが発生する．

```
Reduce (Free PSL version),  6-Jan-2012 ...


1: matrix m(10000, 10000);
***** Segmentation Violation


2: matrix m(5000, 5000);


3: matrix mn(5000, 5000);
```

ただし，エラーを起こしてもハングアップすることなく，一応計算を続けることが出来るようだ．

### 2.3 Mac OS X

Mac OS 用のバイナリファイルが sourceforge.net に置いてあるものの, CSL 版は Snow Leopard (Mac OS X 10.6) では動かなかった．ソースをダウンロードして，コンパイルしてみましたが，やはり，コンパイルに失敗してしまう．CSL 版では，FOX というユーザインターフェースライブラリを使用しているのだが，これの作成に失敗しているようだ（私は，古い Leopard(Mac OS X 10.5) あるいは Tiger(Mac OS X 10.4)？から環境を引き継いでいるので，古いライブラリ等が残っていて環境がめちゃくちゃになっている為かもしれませんが)．

PSL 版の REDUCE は Mac OS X でコンパイル, 実行することができる．Lion(Mac OS X 10.7) では，CSL 版をソースからコンパイルすることができ，ちゃんと動作する．

### 2.4 マニュアル

マニュアルは，Windows 版では reduce.doc のフォルダ中に TEX のソースファイルがあり，manual.tex をコンパイルすればマニュアルが作成される．Linux では，doc フォルダ中にマニュアル一式のソースが入っており，doc/util/r38.pdf が pdf 形式のマニュアルとなっている．CSL 版の REDUCE では，ヘルプ機能があり，help メニューからマニュアルを見ることができる．

日本語のマニュアルについては，ちょっと古くて 3.7 版のものですが，京都大学のウェブサイト[2] で見ることができ，PDF 形式のマニュアルを奈良女子大学のウェブサイト[3] からダウンロードすることもできる．

## 3 計算例

REDUCE は通常，入力された式を可能な限り展開し，同じ項をまとめ，変数に関してある順番で並べ替えをおこない，その結果を出力する．しかし，展開するかどうかや変数の順番，出力の形式等はユーザが変更することが可能で，これを行うための宣言文等が用意されている．

REDUCE ではまた記号行列の操作ができる．例えば,

```
4: matrix m(2,2);
```

は，m を 2 行 2 列の行列として宣言する．そして，次で行列の各要素の値が代入される．

```
5: m := mat((a,b),(c,d));
```

このようにして定義された m は行列として振る舞うため，m を含む式は行列としての計算が行われる．つまり，1/m は逆行列を，2*m - u*m^2 は行列と通常の式との積及び行列と行列の積を，そして det(m) では行列 m の行列式 (determinant) が計算される．

REDUCE は Standard Lisp という Lisp 言語の一つで書かれている．この REDUCE の基礎となっている Lisp 言語でのプログラムを実行する，記号モード (symbolic モード) というモードも用意されている．このモードでは Lisp の構文に基づいて処理が行われる．例えば,

```
6: symbolic car '(a);
```

また，二つのモード間でデータのやり取りを行うこともできる．

---

[2] https://web.kudpc.kyoto-u.ac.jp/doc/Service/Application/REDUCE/reduce3.7/jman.html

[3] http://kako.ics.nara-wu.ac.jp/~kako/software/reduce/reduce37.pdf

### 3.1 REDUCE での変数

REDUCE は多項式を操作することを基本としている．多項式の場合には，標準形というものが定義されるので，これを使って，簡約することが可能である．多項式でない，例えば，$\sqrt{x}$ 等の代数函数や sin($x$) 等の超越函数を操作するために，変数を拡張して $\sqrt{x}$ 等を変数と考えてその多項式として式を扱っている．このように拡張された変数を kernel と呼んでいる．

REDUCE では，例えば，次のような入力はエラーを起こしてしまう．

```
7: df(f,2*x);
*****  (times 2 x) invalid as Kernel
```

ところが，次の計算はエラーにならずに実行される（ただし，結果は 0 となる）．

```
8: df(f,sqrt(2));
```

sin($x$) と $x$ は独立した変数として処理される．ただし，微分や不定積分を計算する場合には，sin($x$) に対する微分規則等を別に定義 (let 文等で) しているので，正しく計算が行われる．ただし，sin($x$) は $x$ の関数であることを知っているが，逆に $x$ が sin($x$) の関数とは定義されていないので，次の計算結果は 0 となる．

```
9: df(x,sin(x));
```

このように，`sin(x+y)`, `x`, `sqrt(2*x)` 等は kernel になるが，sin($x+y$) に対して，例えば

```
10: forall x,y let sin(x+y)=sin(x)*cos(y)+sin(y)*cos(x);
```

のような定義を与えている場合には，kernel にはならない．また，$x^{2/3}$ は内部では $(x^{1/3})^2$ の形で扱っており，kernel にはならない．

u をオペレーターとして宣言すると，$u(1), u(3,2), \ldots$ 等は kernel として，変数と同じ様に扱うことができ，添字付の変数として計算で使用できる．

### 3.2 ロジステック方程式の周期解

次の差分方程式（ロジステック方程式）を考える．

$$x_{n+1} = \lambda x_n(1 - x_n) \quad (0 \le \lambda \le 4) \tag{1}$$

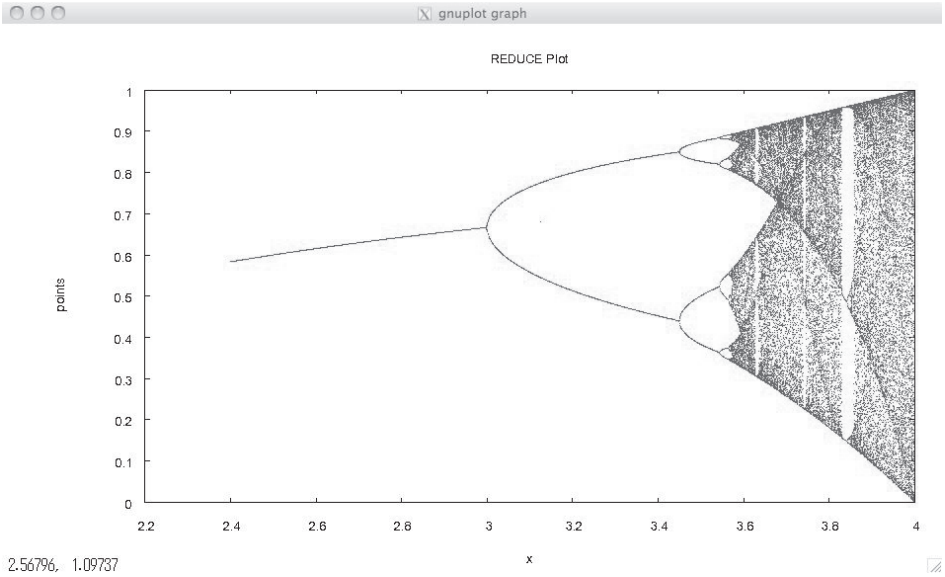
パラメータ $\lambda$ を 0 から 4 まで変化させると，$\lambda = 3$ で 2 周期解が現れる．また，$\lambda = 1+\sqrt{6} \approx 3.449$ で 4 周期解が現れる．どこで，周期解が現れるかをプロットするプログラムを REDUCE で作成すると次の様になる．ここでは，Lisp で関数を定義している．その理由は，algebraic モードでは，rounded モードで計算しないといけないことと，浮動小数点数にはタグ (`!:rd!:`) がつけられるので，単純に結果のリストを gnuplot に渡せないからである．また，結果を REDUCE のリストに変換する為に `'list` を追加している．

```
11: symboic procedure may(a, x);
      begin scalar j;
      for j:= 1 : 1000 do x:= a*x*(1-x);
      return x;
    end;
```

```
12: symbolic procedure listmay();
      begin scalar a, x, l;
      l:=nil;
      for a:=2400:4000 do
        for x:=1 : 100 do
          l := cons(list('list, a*0.001,may(a*0.001, x*0.01)), l);
      return 'list . l;
    end;
```

次により，グラフが表示出来る．

```
13: plot(lisp listmay(), style = dots);
```




ロジステック方程式の周期解を計算する例を以下に示す．

```
14: operator f;
15: forall x let f(x) = a*x*(1-x);
```

REDUCE では，関数を定義するのに operator で宣言する方法と，procedure で定義する方法がある．operator 宣言では，形式的に関数を宣言する．関数の実体は let 文等を使って別に定義する必要がある．procedure 文は C 言語等で定義する関数（手続き）とほとんど同じである．

今の計算例では，上記 2 行の代わりに，

```
14: procedure f(x); a*x*(1-x);
```

として，procedure で宣言しても同じ結果になる．

```
15: f(x);

a*x*( - x + 1)

16: p1:=x-f(x);

p1 := x*(a*x - a + 1)

17: solve(p1, x);

    a - 1
{x=-------,x=0}
      a
```

solve を使って，定常解を求めている．同じ様にして，2 周期解を求める．

```
18: p2:=x-f(f(x));

          3  3      3  2    3      2      2
p2 := x*(a *x  - 2*a *x  + a *x + a *x - a  + 1)

19: solve(p2, x);                    % 周期解
20: resultant(p2, df(p2, x), x);     % 周期解が現れる a を計算
21: solve(ws 20);                    % ws n で n: の入力式の計算結果を参照
```

この結果から 2 周期解は $a = 3$ で現れることが判る．次に 4 周期解を求める．

```
22: p4:=x-f(f(f(f(x))));
23: p4/p2;
24: resultant(ws 23, df(ws 23, x), x);
25: factorize(ws);

   2
{{a  - 2*a - 5,2},
   2
 {a  - 4*a + 5,3},
   2
 {a  + 1,3},
   6      5      4       3      2
 {a  - 6*a  + 3*a  + 28*a  - 9*a  - 54*a - 135,4},
 {a,144}}
```

```
26: g:=gcd(ws(24), df(ws(24), a));
27: ws(24)/g;
28: load roots;
29: realroots(ws 27);

{a=0,
 a= - 1.9601,
 a= - 1.44949,
 a=3.44949,
 a=3.9601}

30: solve(ws 27);

Unknown: a

{a=sqrt(6) + 1,
 a= - sqrt(6) + 1,
             6       5       4        3       2
 a=root_of(a_  - 6*a_  + 3*a_  + 28*a_  - 9*a_  - 54*a_ - 135,a_,tag_2),
 a=i + 2,
 a=i,
 a= - i + 2,
 a= - i,
 a=0}
```

実数解で，$0 \le a \le 4$ の範囲内の解は，$a = 1 + \sqrt{6} \approx 3.449$ と $a \approx 3.9601$ である．

なお，$resultant(f, g, x)$ は $x$ の多項式 $f, g$ に対する終結式を求めている．$f$ と $g$ の終結式がゼロになるのは，$f = 0$ と $g = 0$ が共通の零点を持つ場合である．上の計算では，$f$ とその微分が共通の零点を持つ条件を調べている．

## 4 パッケージ

REDUCE では利用者によって開発されたパッケージが有り，そのリストを以下に示す．

| 名称 | 主な用途 | 名称 | 主な用途 |
|---|---|---|---|
| algint | 平方根を含む式の積分 | applysym | 偏微分方程式の対称性 |
| arnum | 代数的数体上での計算 | assist | 種々の有用なツール |
| avector | ベクトルパッケージ | bibasis | Boolean Involutive Bases の計算 |
| boolean | ブール代数式の計算 | cali | 可換代数演算 |

| 名称 | 主な用途 | 名称 | 主な用途 |
|---|---|---|---|
| camal | 天体力学の計算 | cantens | 添字付きの Object の操作 |
| cdiff | 微分方程式の幾何学計算 | cgb | 包括的 Gröbner 基底の計算 |
| changevr | 微分方程式の独立変数の変換 | compact | 関係式による式の簡約 |
| conlaw | 微分方程式の保存量 | contfr | 連分数近似 |
| crack | 常/偏微分方程式 | cvit | ディラックの $\gamma$ 行列のトレース |
| defint | 定積分 | desir | 線形同次微分方程式 |
| dfpart | 形式的関数の微分 | dummy | ダミー変数を含む式の正準表現 |
| eds | 外微分形式の計算 | excalc | 微分形式の計算 |
| fide | 偏微分方程式の有限要素法 | fps | 形式巾級数の計算 |
| gentran | FORTRAN もしくは C 言語への変換 | | |
| geometry | 平面幾何学の計算 | ghyper | 一般化された超幾何関数の簡約 |
| gnuplot | gnuplot とのグラフ描画 | groebner | Gröbner 基底の計算 |
| guardian | 保護された式の計算 | ideals | 多項式イデアル |
| ineq | 不等式の解法 | invbase | 包含的基底 |
| laplace | ラプラス変換と逆ラプラス変換 | lie | Lie 代数の計算 |
| liepde | 偏微分方程式の無限小対称性 | limits | 極限値 |
| linalg | 線形代数パッケージ | lineneq | 線形不等式 |
| lpdo | 線形偏微分演算子の計算 | mathml | MathML とのインターフェース |
| meijerg | Meijer の G 関数 | mrvlimit | 指数–対数関数の極限値 |
| modsr | 合同演算による方程式の解の計算 | | |
| ncpoly | 非可換多項式イデアル | normform | 行列の標準形の計算 |
| numeric | 数値計算 | odesolve | 常微分方程式の解 |
| orthovec | 直交曲座標での 3 次元ベクトル解析 | | |
| physop | 量子力学における演算子法 | pm | REDUCE のパターンマッチャー |
| polydiv | 拡張された多項式の割算 | qsum | $q$-超幾何級数の和 |
| randpoly | ランダムな多項式の生成 | rataprx | 有理近似 |
| ratint | 有理関数の不定積分 | rdebug | REDUCE のデバッグ |
| reacteqn | 化学反応方程式 | reset | 初期状態へのリセット |
| residue | 留数 | rlfi | LaTeX 形式での出力 |
| roots | 高次方程式の解 | rsolve | 多項式の有理数/整数解の計算 |
| rtrace | REDUCE プログラムの実行履歴の追跡 | | |
| scope | ソースコードの最適化 | sets | 基本的な集合計算 |
| sparse | 疎行列の計算 | spde | 偏微分方程式の対称性 |
| specfn | 特殊関数のパッケージ | sum | REDUCE の和分パッケージ |
| susy2 | 素粒子物理学における超対称性の計算 | | |
| symmetry | 対称行列上の演算 | taylor | テイラー級数の計算 |

| 名称 | 主な用途 | 名称 | 主な用途 |
|---|---|---|---|
| tps | べき級数 | trigint | Weierstrass 置換 |
| tri | TEX と REDUCE のインターフェース | | |
| trigsimp | 三角関数の簡約 | turtle | タートルグラフィック |
| v3tools | 3 次元ベクトルの計算 | wu | Wu の方法による方程式の解法 |
| xcolor | 非アーベルゲージ理論における色因子の計算 | | |
| xideal | 外積代数の Gröbner 基底 | zeilberg | 不定和分および定和分 |
| ztrans | Z-変換 | | |

これらのパッケージ（モジュール）を利用するには，最初にロードする必要がある（パッケージによっては自動的にロードが行われるものもある）．例えば，gnuplot でのグラブ描画を行うには，次のようにパッケージを使用する．

```
31: load gnuplot;
32: plot(1/(x^2+y^2), x=(-0.5 .. 0.5), y=(-0.5 .. 0.5), hidden3d,
          contour, view="70,20");
```

## 参 考 文 献

[1] 広田良悟: REDUCE 入門 – パソコンによる数式処理活用法，Information & computing, Vol. 33，サイエンス社，1989.

[2] 守屋良二: 例題でわかる REDUCE – あなたの数学を応援します，海文堂出版，1992.

[3] F. Brackx: Computer Algebra With Lisp and Reduce: An Introduction to Computer-Aided Pure Mathematics, Matimatics and Its Applications, Kkuwer Academic Plub., 1992.

[4] Gerhard Rayna: Reduce: Software for Algebraic Computation, Springer Series Symbolic Computation Springer-Verlag, 1987.