Mathematica, Python and R言語

Linear Algebra and Geometric Algebra

－コンピュータで線型代数を学ぶ－

1 数学学習・研究のためコンピュータ

数学研究においてコンピュータを活用することは，現在では日常的になっている．そもそも数学とコンピュータは歴史的に非常に関係が深い．コンピュータの黎明期に活躍した数学者としてフォン・ノイマン，アラン・チューリングやアロンゾ・チャーチなどが有名であるし，情報理論であればクロード・シャノン，ウォーレン・ウィーバーが創始者でもある．歴史的にライプニッツの2進法まで遡れば，学問として深い関係があることがわかる．ただ，1970年から1980年代の多くの大学では，コンピュータ（当時は電子計算機と呼んでいた）に関連する分野といえば，応用面としての数値解析や数学基礎論分野の計算論であり，代数分野，解析分野，幾何分野の学生および研究者のコンピュータ（電子計算機）の利用はあまり多くなかった．当時の自身の経験でいうと学部の授業の一環でFortranの演習を経験した程度である．数学の研究にコンピュータを活用としたものとして，エポックメイキングな出来事として記憶に残っていることは四色問題のコンピュータによる解決である．その後，数式処理ソフトウェアで記号計算が可能になり，数学の各分野の研究や学習に寄与してきたのは1990年代以降であろう．最近ではパーソナルコンピュータ（PC）用の数学ソフトウェアが充実し，使いやすくなっている．特に中学生・高校生でも活用している事例が多くなってきた．ここでは，個人がPCを用いて数学の学習や研究する場合について述べる．

2 線型代数とコンピュータ

新入生が初年級で学ぶ数学は，微分積分（解析学）と線型代数である．高等学校数学との関係でいうと，当然，微分積分は高等学校での微分積分の延長として位置付けているので，大学新入生にとっても比較的取り組みやすい科目である．線型代数は，高等学校の数学の中で関係の深い分野と言えるのがベクトルになるであろう．かつては高等学校の数学で行列や線型変換（一次変換）も扱っていたが，2012年度以降，全く扱ってこなかった（2022年度からの高等学校新学習指導要領では数学Cで行列が復活する）．このため高等学校の数学と大学の線型代数には少しギャップがある．大学での線型代数では，一般的にはベクトルから行列・行列式そして線型空間へと徐々に抽象化されていく．私の学部時代在籍した数学科には，1年生を対象とした「数学概論1」という科目があり，風変わりな線型代数をオリジナルな講義録で学んだ．この講義録は後年，「線型代数」（小島順著：日本放送出版協会）[1]として世に出る．特徴を一言で述べると，公理主義的かつ圏論的に扱った内容にも拘わらず，1次元線型空間を扱ったり，線型変換の分類としての標準化問題を扱う中に，二次形式の分類や

線型ベクトル場(線型微分方程式の入門)を扱うというかなり幾何学的なものだった.今思うと,この線型代数を実現できる数学ソフトウェアがあればと悔やまれる.

そこでここでは,

・ベクトルと行列およびそれに関係する演算
・連立方程式
・固有値問題
・二次形式とその標準化
・線型微分方程式

を中心に取り上げる.

3　数学用ソフトウェアの使用目的と選び方

目的や選び方は個人に依存するが,まず目的としては複雑な計算,決まりきった手順で行う計算そして3次元までのグラフィックス描画であろう.選び方としては,有償か無償(フリーソフトウェア)かの判断が必要になる.有償ソフトウェアの多くは,数学の内容を統合的にサポートするシステムが多く,多くの大学ではサイトライセンスにより学生は自由に使える.また,無償ソフトウェアは,純粋なプログラミング言語(+ライブラリ)や特定の分野に特化した内容のものが多いようだ.

本稿では,有償な数式処理システム(数学統合的な環境を提供しているシステム)としてMathematica,無償なものとして,Wolfram Alpha ,Python(+ライブラリ),R言語,そして数学の特定の分野に特化したものとして,幾何代数の計算環境を提供しているGAViewerを紹介する.今回は紹介しないものとして,数値解析システムとして有名でベクトル・行列計算に定評のMATLAB(Matrix Laboratory)がある.

4　数学用ソフトウェア

(1)　Mathematica

Mathematicaは英国の物理学者スティーブン・ウルフラムが開発した数式処理システムである.MathematicaはプログラミングALGOL・LISP・APLおよび数式処理システムMacsymaの影響を受けており,プログラミング言語としても強力である.Mathematicaには高度な組み込み関数が多数用意され,システムとしては「Wolfram言語」[2]を解釈し実際に計算を実行する「カーネル」とその計算結果を表示する「フロントエンド」の2つの部分から構成されている.そしてカーネルとフロントエンドの間の通信にはMathLinkというプロトコルが使われている.

Wolfram言語の基本は,関数[引数](f[x, y])という形式となっている.

例:1+2 → Plus[1,2]

(Expression → FullForn)

そして,データの形式は「リスト」という概念で統一されている.

・リスト:{1,2,3} → List[1,2,3]

(2)　Python

Pythonはインタプリタ型で汎用プログラミング言語である．オランダ生まれのグイド・ヴァン・ロッサムによって開発された．1991年にリリースされオフサイドルールの使用によってコードの可読性を重視しているのが特徴的である．初心者にとっても読みやすく効率のよいコードを簡単に書けるように設計されている．まず，データの形式について，Pythonには要素をもつオブジェクトとして，

・集合：{1,2,3}

・タプル（tuple）:(1,2,3)

・リスト:[1,2,3]

がある．タプルとリストの違いはオブジェクトとして immutable（変更不能） かmutable（変更可能)かの違いである．Python では配列というクラスは特に用意されておらず，行列はリストを入れ子にすることで実現する．また，これら基本機能以外に外部ライブラリが充実しているのが Python の特徴である．まず NumPy（行列計算）である．NumPyでは多次元配列オブジェクト ndarrayが強力でベクトルや行列計算に非常に便利な環境を提供する．その他としてSciPy(科学技術計算)，SymPy（数式記号処理)，Matplotlib(グラフィックス機能)，Pandas（統計処理）などの外部ライブラリが用意され，これらを用いることでPython単体よりも高速処理を実現し，統計や機械学習などにも使われる．特に SymPy は記号計算を実現し，MathematicaやMATLABに近い環境を提供する．

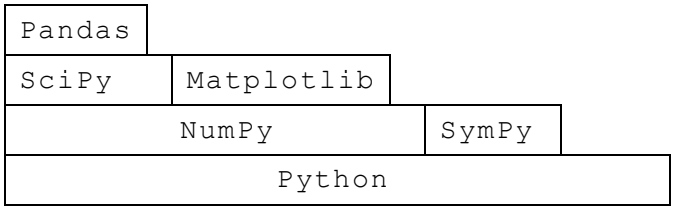


図1　Pythonと外部ライブラリの概要

(3)　R言語

R言語（以下R）はオープンソースの統計解析向けのプログラミング言語で，ニュージーランドのオークランド大学のロス・イハカとロバート・ジェントルマンにより1990年代前半に作られた．ソースコードは主にC言語、FORTRANなどによって開発された.R はデータ解析を行ってグラフなどに出力する機能に優れている．R も関数[引数]を基本として，

・ベクトル：c(1,2,3)

・行列：matrix(c(1:6,2,3)　　# 2行3列の行列

・配列：arrary(1:6,dim= c(2,3))　# 2行3列の配列

・リスト：list(1,2,a,b)

というデータ構造をもっている．ただし，配列やリストは異なるデータ形式にも対応する．また，関数型プログラミング言語の環境も提供する．

5　ベクトル・行列の四則演算（データ形式）

(1)　Mathematica での四則演算

（In[1]:=入力,Out[1]:出力，;は出力抑制）

```
In[1]:= a={1,2,3};          # リストの生成
In[2]:= a + b;              # 和
In[3]:= a.b;                # 内積
In[4]:= Cross[a,b];         # 外積
In[5]:= aa={{1,2},{3,4}};   # 行列
In[6]:= aa//MatrixForm;     # 行列表示
```

(2)　Python での四則演算（対話モード入力 >>>）

```
>>> import numpy as np
      # Numpy をインポートして np とする
>>> a = np.array([3,4])      # 配列の生成
>>> a + b                    # 加法
>>> a * b                    # 成分ごとの乗法
>>> a / b                    # 成分ごとの除法
>>> a.dot(b)                 # 内積
>>> a@b (ver3.5 以降)        # 内積
>>> np.cross(a,b)            # 外積
```

(3)　R での四則演算（対話モード入力 >）

```
> a <- c(1,2,3,4)           # ベクトルの生成
> a + b                     # 和
> a * b                     # 成分ごとの積
> a %*% b                   # 内積
```

3 つともほぼ同等なスタイルである.

6　関数の定義

ここでは，3n+1 問題の関数定義を例に，3 つの言語の特徴を比較する．Wolfram 言語は関数型言語の構造を持つので,条件分岐は下記のようなパターンマッチングを用いる．これに対して，Python も R もインタプリタ型の手続き言語であるが，Python の特徴はオフサイドルールを用いて，if 文などを { } を用いずにブロックを字下げ（インデント）で指定する.

(1)　Mathematica：

```
In[1]:= collatz[i_?OddQ]:= 3*i + 1;
        collatz[i_?EvenQ]:= i/2;
        collatz[3]
Out[1]:= 10  # 出力
```

(2)　Python：

```
>>> def collatz(x):
      if x%2 == 0:
         return x/2
      else:
         return 3*x+1
>>> collatz(3)
      10  # 出力
```

(3)　R：

```
 >collatz<-function(x){
  if (x%%2 == 0){
       x<-x/2
     }
   else{
       x <-3*x+1
   }
   return(x)
 }
> collatz(3)
  [1] 10  # 出力
```

7　線型代数のトピックス

(1)　3元連立1次方程式の解法

次の例で説明する.

$$x+y+z=1,$$
$$2x-y+z=-1,$$
$$3x-y+2z=2.$$

○Mathematica：

```
In[1]:=
   aaa = {{1,1,1},{2,-1,1},{3,-1,2}};
   xyz = {{x},{y},{z}};
   bbb = {{1},{-1},{2}};
   MatrixForm[aaa].MatrixForm[xyz]==
                          MatrixForm[bbb];
```

$$\text{Out[1]:=}\quad \begin{pmatrix}1 & 1 & 1\\ 2 & -1 & 1\\ 3 & -1 & 2\end{pmatrix}.\begin{pmatrix}x\\ y\\ z\end{pmatrix} == \begin{pmatrix}1\\ -1\\ 2\end{pmatrix}$$

・逆行列 Inverse を利用して解を求める方法.

```
In[2]:=Inverse[aaa].bbb;
Out[2]:={{-6},{-2},{9}}   # 解
```

・LineaSolve を利用する方法.

```
In[3]:=LinearSolve[aaa, bbb];
Out[3]:= {{-6},{-2},{9}}    # 解
```

・掃き出し法で RowReduce を利用する方法.

```
In[4]:= aaabbb
={{1,1,1,1},{2,-1,1,-1},{3,-1, 2,2}};
xyz1 = {{x},{y},{z},{-1}};
MatrixForm[aaabbb].MatrixForm[xyz1] ==
          MatrixForm[{{0},{0},{0},{0}}];
```

$$\text{Out[4]:=}\quad \begin{pmatrix}1 & 1 & 1 & 1\\ 2 & -1 & 1 & -1\\ 3 & -1 & 2 & 2\end{pmatrix}.\begin{pmatrix}x\\ y\\ z\\ -1\end{pmatrix} == \begin{pmatrix}0\\ 0\\ 0\\ 0\end{pmatrix}$$

```
In[5]:=RowReduce[aaabbb]//MatrixForm;
```

$$\text{Out[5]:=}\quad \begin{pmatrix}1 & 0 & 0 & -6\\ 0 & 1 & 0 & -2\\ 0 & 0 & 1 & 9\end{pmatrix}$$

・グレブナー基底を利用する方法.

Wolfram 言語において，方程式の求解アルゴリズムはブッフバーガー氏が開発した Gro¨bner 基底のアルゴリズムを使用している．少し大げさだが，この例で Gro¨bner 基底を計算する．数式処理システムの内部アルゴリズムを意識することも重要である.

```
In[1]:=GroebnerBasis[
       {x+y+z-1,2x-y+z+1,3x-y+2z-2},
       {x,y}
        ];
Out[1]:= {-9 + z, 2 + y, 6 + x}
```

○Python：

$$x+y=1,$$
$$2x-y=1.$$

を例にする.

NumPy では.

```
>>> from numpy.linalg import solve
>>> aaa = [[1,1],[2,-1]]
>>> solve(aaa,[1,1])
array([0.66666667, 0.33333333])
```

結果は小数点表示である.

SymPy では.

```
>>> from sympy import solve
>>> from sympy.abc import x,y,z
>>> solve([x+y-1,2*x-y-1],{x,y});
 {x: 2/3, y: 1/3}
```

と分数表示になる.

```
>>> solve([x+y+z-1,2*x-y+z+1,
          3*x-y+2*z-2],[x,y,z]);
 {x: -6, y: -2, z: 9}
```

このように Sympy は記号処理などに適しており Numpy での配列・数値計算との違いが際立っている.

(2)　固有値の計算

○Mathematica：

```
In[1]:= AA = {{2,2},{2,5}};
         Eigenvectors[AA];
Out[1]:={{1, 2}, {-2, 1}} # 固有ベクトル
In[2]:= Eigenvalues[AA];
Out[2]:= {6, 1}  # 固有値
```

○Python：

```
>>> import sympy as sp
>>> import sympy as A, B
>>> A = [[2,2],[2,5]]
>>> B = sp.Matrix(A).eigenvects();B
   [(1,1,[Matrix([[-2],[ 1]])]),
    (6,1,[Matrix([[1/2],[ 1]])])]
```

固有値は重複度も含めて出力する.

(3)　二次形式の標準化（Mathematica）

例として，$2x^2-4xy+5y^2=9$ を取り上げる．対称行列 A を用いて，次のように表現できる.

$$\begin{pmatrix} x & y \end{pmatrix}\begin{pmatrix} 2 & 2 \\ 2 & 5 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}=9.$$

さらに直交行列 P を用いて，A を対角化すると，

$${}^tPAP=B=\begin{pmatrix} 6 & 0 \\ 0 & 1 \end{pmatrix}.$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = P\begin{pmatrix} X \\ Y \end{pmatrix}, P = \frac{1}{\sqrt{5}}\begin{pmatrix} 1 & -2 \\ 2 & 1 \end{pmatrix}.$$

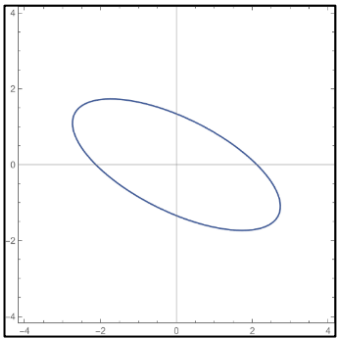
$$\begin{pmatrix} x & y \end{pmatrix} A\begin{pmatrix} x \\ y \end{pmatrix} = {}^{t}\left(P\begin{pmatrix} X \\ Y \end{pmatrix}\right) A\left(P\begin{pmatrix} X \\ Y \end{pmatrix}\right)$$

$$= (X,Y)\,{}^{t}PAP\begin{pmatrix} X \\ Y \end{pmatrix} = (X,Y)\begin{pmatrix} 6 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} X \\ Y \end{pmatrix}$$

$$= 6X^2 + Y^2 = 9$$

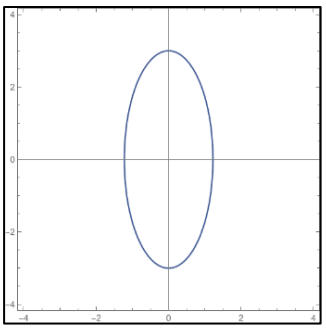
と標準化できる．これを Mathematica で計算する．

```
In[1]:= QQ={x,y}.AA.{x,y}//Simplify;
Out[1]:=  2x^2-4xy+5y^2
In[2]:= ContourPlot[
          QQ == 9,{x,-3,3},{y,-3,3},
          Axes -> True, AxesStyle -> Gray
          ];（図 2）
```



そして，固有ベクトルを用いて標準化する．

```
In[3]:=
        NN = Norm[Eigenvectors[AA][[1]]];
        PP = Eigenvectors[AA]/NN;
        P = Inverse[PP].{x, y};
        RR = P.AA.P//Simplify;
Out[3]:= 6X^2+Y^2=9
In[4]:= ContourPlot[
          RR == 9,{x,-10,10},{y,-10,10},
          Axes->True,AxesStyle->Gray
        ];（図 3）
```



(4)　線型微分方程式（Mathematica）

$$\frac{dx(t)}{dt} = x(t) - y(t), \frac{dy(t)}{dt} = x(t) + y(t),$$
$$x(0) = 1, y(0) = 1$$

$$\begin{pmatrix} x'(t) \\ y'(t) \end{pmatrix} = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$$

の例で考える．

ここでは，Dsolve を使う．

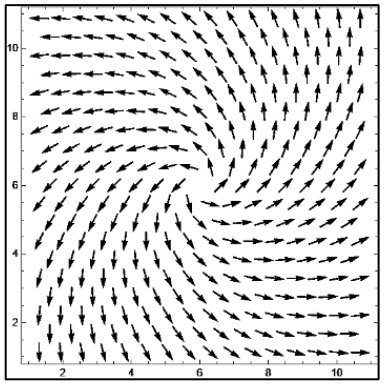
```
In[1]:= sol = DSolve[
    {x'[t]== x[t]-y[t],
     y'[t]== x[t]+y[t], x[0]==1,y[0]==1},
     {x[t],y[t]},t
    ];
```

その解は，

```
Out[1]:= {{x[t]->E^t(Cos[t]-Sin[t]),
           y[t]->E^t(Cos[t]+Sin[t])}}
```

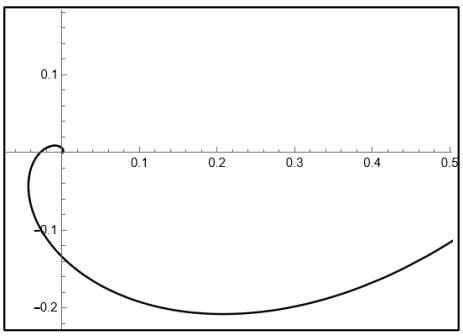
この解曲線を表示するには，

```
In[2]:= ParametricPlot[
        {x[t],y[t]}/.sol,{t,-3Pi,0},
        AspectRatio -> Automatic
        ];（図 4）
```



ベクトル場をグラフ化するには，

```
In[3]:= ListVectorPlot[
       Table[{x - y, x + y},
            {x,-5, 5},{y,-5, 5}
       ]
      ];（図 5）
```



(5)　線型回帰分析（R）

線型代数とは直接関係しないが，R は統計処理に適した言語であり，例えば散布データを線型近似する方法をもっている．ここでは人口数の経年変化のデータを pop として説明する．

```
> pop.lm <- lm(number~year,data = pop)          # pop.lm を定義
                                  # lm（目的変量~説明変量，data=データ
> plot(pop)                      # 散布図の描画
> abline(pop.lm)（図 6）       # 回帰直線の追加
> coefficients(pop.lm)     # 直線の切片と傾き
  (Intercept)    year
  -74678.9545    100.9545
 > cor(pop$year,pop$number)  # 相関係数
  0.9189531
```

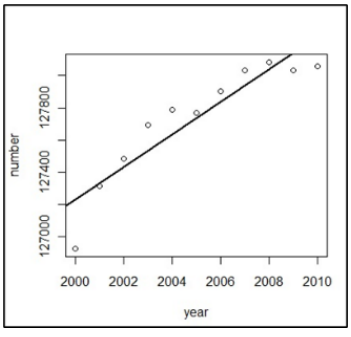


図 6　R による回帰直線

7　Wolfram Alpha について

Wolfram Alpha[3]はウルフラム・リサーチ社が開発した数理科学専用の検索システムで 2009 年に公開された．ユーザはテキストフィールドに質問や計算リクエストを入力し，Wolfram Alpha は知識ベースの蓄積された構造化データから情報を回答する．これはコンピュータ代数，記号および数値計算などの機能を有する Mathematica（Wolfram 言語）上に構築されている（図 7）．

線型代数のコーナーには，

「ベクトル」,「行列」,「線型独立」,「ベクトル空間」

という項目があり，例えば，

「(1,-1），(1,1)の固有値は？」

と入力すると，次のように返す（図 8）．

図 7　Wolfram Alpha の画面

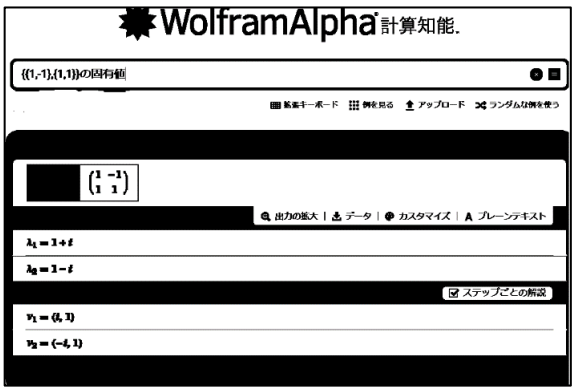


図 8　固有値を求める画面

8　GAViewer

幾何代数（Geometric Algebra）はハミルトンの四元数を拡張したクリフォード代数の流れを汲む線型代数の拡張である[4]．最近では米国の物理学者であるデビット・ヘステネスが中心的な役割を果たしている．大きな特徴はコーディネートフリーで幾何学的側面を重視した線型代数といえる．特に，幾何積として，

$$uv := u \cdot v + u \wedge v$$

と内積と外積を使って定義する．ここで紹介する GAViewer[5]は幾何代数用に開発されたアプリケーションソフトである．幾何積の計算において手動で大変な場合，GAViewer を用いることでいろいろな計算が簡単に出来る．

ここでは 2 次元での回転と 3 次元での回転例を紹介しよう．一般的には 3 次元回転は行列計算を行いることが多く，かなり計算が大変なのだが，この幾何積を用いると単純な公式で計算できる[6]．最近ではロボット設計など工学分野でも使用されている．

○2 次元回転：e1,e2:正規直交基底

u = e1+e2 を原点 O を回転軸として 90 度回転させる場合：

```
>> u = e1 + e2                          # 回転前
 u = 1.00*e1+1.00*e2
```

```
>> r = sqrt(2)/2*(1+gp(e1,e2))# 作用素 R
 r = 0.71+0.71*e1^e2
>> reverse(r)                          # R^-1
  ans = 0.71-0.71*e1^e2
>> gp(gp(reverse(r),u),r)    # 公式 R^-1AR
  ans = -1.00*e1 + 1.00*e2       # 回転後
```

○3 次元回転：e1,e2,e3:正規直交基底

e1 を法線方向 e1＋e2＋e3 を回転軸として 120 度回転させる場合：

```
>> I = e1^e2^e3              # 単位擬スカラー
    I = 1.00*e1^e2^e3
>> n = sqrt(3)/3*(e1+e2+e3)      # 回転軸
 n = 0.58*e1+0.58*e2+0.58*e3
>> gp(I,n)                                  # nI
  ans = 0.58*e2^e3+0.58*e3^e1+0.58*e1^e2
>> u = e1                                  # 回転前
   u = 1.00*e1
>> r = (1/2+gp(I,n)*sqrt(3)/2) # 作用素 R
r=0.50+0.50*e2^e3+0.50*e3^e1+0.50*e1^e2
>> gp(gp(reverse(r),e1),r)    # 公式 R^-1AR
 ans = 1.00*e2                    # 回転後
```

上の 2 つの状況を図示すると，次のようになる．

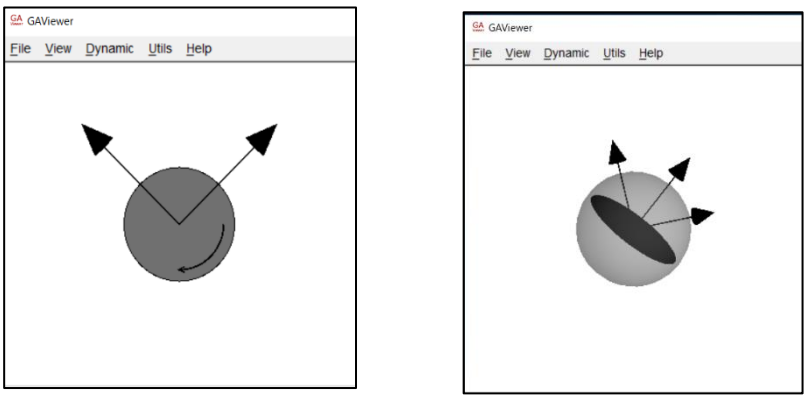


図 9：GAViewer の画面

9　おわりに

今回，紹介した数学のアプリケーションソフトは一部でしかないが，一つのシステムや言語をしっかり習得すれば，他の環境での習得は比較的容易である．ともかく自分の学習や研究に適したものを選択することが重要であろう．

【参考文献】

[1] 小島順：「線型代数」，日本放送出版協会，1976.

[2] Wolfram.S：
An Elementary Introduction to the Wolfram Language,2015.

[3] Wolfram Alpha:https://www.wolframalpha.com/

[4] A.Macdonald：Linear and Geometric Algebra,2010.

[5] GAViewer:https://geometricalgebra.org/gaviewer_download.html

[6] 武沢護：「幾何積ノート」，2021. ：
http://www.f.waseda.jp/takezawa/lab/paper/ga.pdf

「2022 年 5 月　数理科学」より

早稲田大学高等学院　武　沢　護
takezawa@waseda.jp