数式処理ソフトMaple入門(III)

ー3DCGと線形代数ー

関西学院大学理工学部 西谷滋人

2次元平面での写像

CG(Computer graphics)は，三次元に配置した物体の輪郭を画面に投影することです．もっとも基本となる２次元平面での写像をお見せします．

写像の表示

適当に作った行列によって点を移動させる写像の様子を示すスクリプトは以下の通りです．

```
> restart;
  with(LinearAlgebra):with(plots):with(plottools):
> A:=Matrix([[1,2],[2,1]]);
  a0:=Vector([1,1]);
```

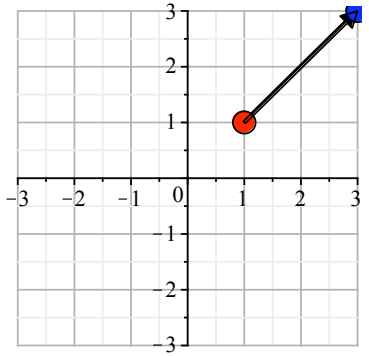
$$A := \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$$

$$a0 := \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{2.1.1}$$

```
> a1:=A.a0;
```

$$a1 := \begin{bmatrix} 3 \\ 3 \end{bmatrix} \tag{2.1.2}$$

```
> p0:=convert(a0,list):p1:=convert(a1,list):
  point1:=[disk(p0,0.2,color=red),disk(p1,0.2,color=blue)]:
  line1:=arrow(p0,p1,0.05,0.3,0.1):
  display(point1,line1,view=[-3..3,-3..3],gridlines=true);
```



a0(1,0)の赤点が，a1(1,2)へ移動していることがわかると思います．

回転写像

次に原点周りでの回転の様子です．tに回転角を入れています．

```
> Matrix([[cos(theta),sin(theta)],[-sin(theta),cos(theta)]])
  ;
```

$$\begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \tag{2.2.1}$$

```
> t:=Pi/3;
  A:=Matrix([[cos(t),sin(t)],[-sin(t),cos(t)]]);
  a0:=Vector([3,0]);
```

$$t := \frac{1}{3}\pi$$

$$A := \begin{bmatrix} \frac{1}{2} & \frac{1}{2}\sqrt{3} \\ -\frac{1}{2}\sqrt{3} & \frac{1}{2} \end{bmatrix}$$

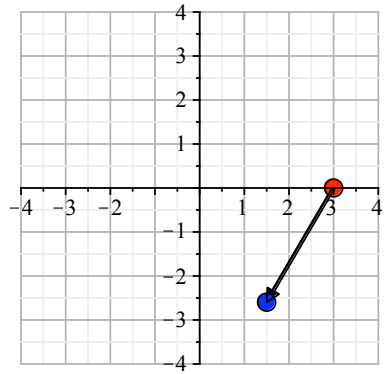
$$a0 := \begin{bmatrix} 3 \\ 0 \end{bmatrix} \tag{2.2.2}$$

```
> a1:=A.a0;
```

(2.2.3)

$$a1 := \begin{bmatrix} \frac{3}{2} \\ -\frac{3}{2}\sqrt{3} \end{bmatrix} \quad (2.2.3)$$

```
> p0:=convert(a0,list):p1:=convert(a1,list):
  point1:=[disk(p0,0.2,color=red),disk(p1,0.2,color=blue)]:
  line1:=arrow(p0,p1,0.05,0.3,0.1):
  display(point1,line1,view=[-4..4,-4..4],gridlines=true);
```



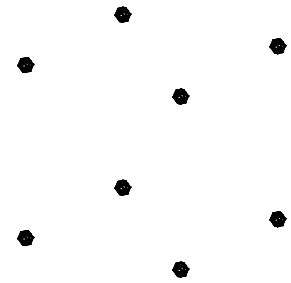
最も簡単な平行投影図の作成

ではもう少し複雑な対象物として，ここでは立方体の表示を考えます．まず３次元座標を打ち込みます．

```
> restart;
  with(plots):
  with(plottools):
  p:=[[0,0,0],[1,0,0],[1,1,0],[0,1,0],
  [0,0,1],[1,0,1],[1,1,1],[0,1,1]]:
```

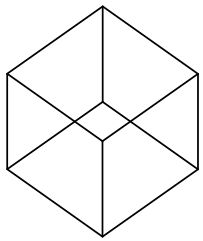
次にこれを表示させます．

```
> points:= { seq(p[i],i=1..8) }:
  pointplot3d(points,symbol=circle,symbolsize=40,color=black);
```



もうすこし見やすいように頂点を結んでおきましょう．

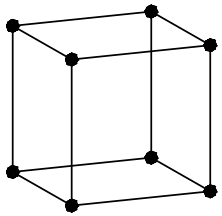
```
> ll:=[[1,2],[2,3],[3,4],[4,
  1],[1,5],[2,6],[3,7],[4,
  8],
  [5,6],[6,7],[7,8],[8,5]]:
  lines:=[seq(line(p[ll[i]
  [1]],p[ll[i][2]]),i=1..
  nops(ll))]:
> display(lines,scaling=
  constrained,color=black);
```



```
> l3:=display(lines,scaling=
  constrained,color=black):
  p3:=pointplot3d(points,
  symbol=circle,symbolsize=
  40,color=black):
  display([p3,l3],scaling=
  constrained,color=black);
```

スクリプトの中身

こちらの欄では，Mapleスクリプトの中身を解説していきます．最初読むときは無視してもらって結構です．
まず，点と点を結ぶ線の引き方を見ておきます．たとえば，p[1]とp[2]との間を線で結ぶには，
line(p[1],p[2]);
とすればできます．



>
>

これで点が表示されているはずです．画像をつまんでぐるぐる回してみてください．Mapleではこんな操作は簡単にできます．ただ，よく見ればわかるように，この3次元表示では透視図ではなく，平行投影図といわれるものを書いています．そこで本当の透視図を作るのがこの課題の目的です．

透視図

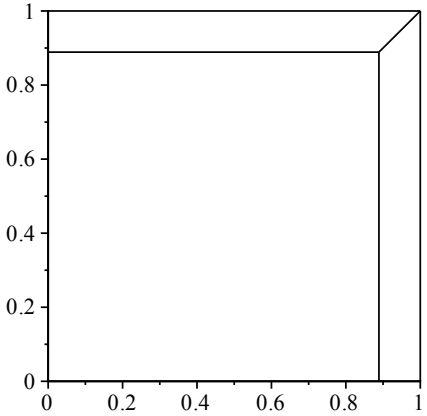
さていよいよ透視図の作成です．もっとも簡単な変換法は

```
> proj2d:=proc(x,z)
   local x1,y1;
   x1:=x[1]*z/(z-x[3]);
   y1:=x[2]*z/(z-x[3]);
   return [x1,y1];
  end proc:
```

です．zに視点の距離を入れて，xで座標を受け取って変換した結果を[x1,y1]として返しています．この関数を前の表示と組み合わせれば透視図の描画が完成です．

```
> z_p:=-8;
  lines:=[seq(line(proj2d(p[ll[i][1]],z_p),
                   proj2d(p[ll[i][2]],z_p)),
   i=1..nops(ll))]:
  display(lines);
```

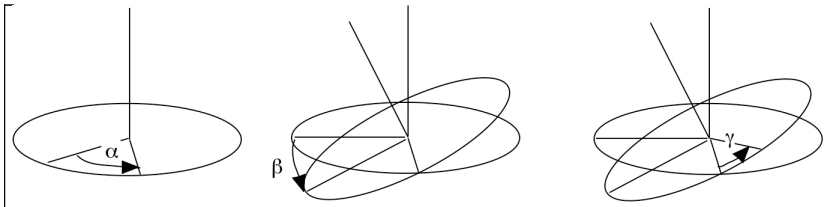
$$z\_p := -8$$



なんか物足りないでしょ．もっとまともなのを造りたい人は以下の記述を参照して工夫してください．

課題

３次元上での物体の回転を表す“オイラー角”は次のような変換マトリックスからなります．これを参照して立方体を回した状態での透視図を描いてください．（参考：ベクトル・テンソルと行列，ジョージ・アルフケン著，講談社1977 ）



オイラー角を示す模式図．二つの軸の回転とその軸周りの回転の３個の独立なパラメータで示される．

```
> with(LinearAlgebra):
> RZ1:=Matrix(3,3,[[cos(alpha),sin(alpha),0],[-sin(alpha),cos
  (alpha),0],[0,0,1]]);
```

$$RZ1 := \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
> RY:=Matrix(3,3,[[cos(beta),0,-sin(beta)],[0,1,0],[sin(beta),
  0,cos(beta)]]);
```

$$RY := \begin{bmatrix} \cos(\beta) & 0 & -\sin(\beta) \\ 0 & 1 & 0 \\ \sin(\beta) & 0 & \cos(\beta) \end{bmatrix}$$

```
> RZ2:=Matrix(3,3,[[cos(gamma),sin(gamma),0],[-sin(gamma),cos
  (gamma),0],[0,0,1]]);
```

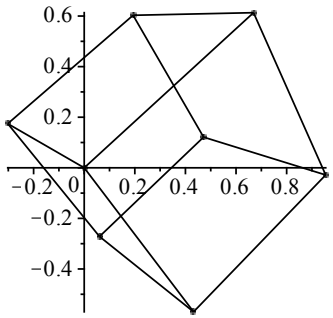
$$RZ2 := \begin{bmatrix} \cos(\gamma) & \sin(\gamma) & 0 \\ -\sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
> Trans1:=unapply(RZ2.RY.RZ1,(alpha,beta,gamma)):
```

解答例

```
> c:=[0,0,0,0,0,0,0,0]:
  for i from 1 to 8 do
  c[i]:=convert(Trans1(Pi*20/180,Pi*30/180,Pi*30/180).Vector
  (p[i]),list);
  od:
```

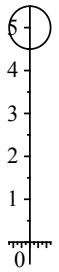
```
> z_p:=-2:
  points:=pointplot([seq(proj2d(c[i],z_p),i=1..8)]):
  lines:=[seq(line(proj2d(c[(ll[i][1])],z_p),proj2d(c[(ll[i]
  [2])],z_p)),i=1..nops(ll))]:
  display(points,lines,scaling=constrained);
```



アニメーションの作成

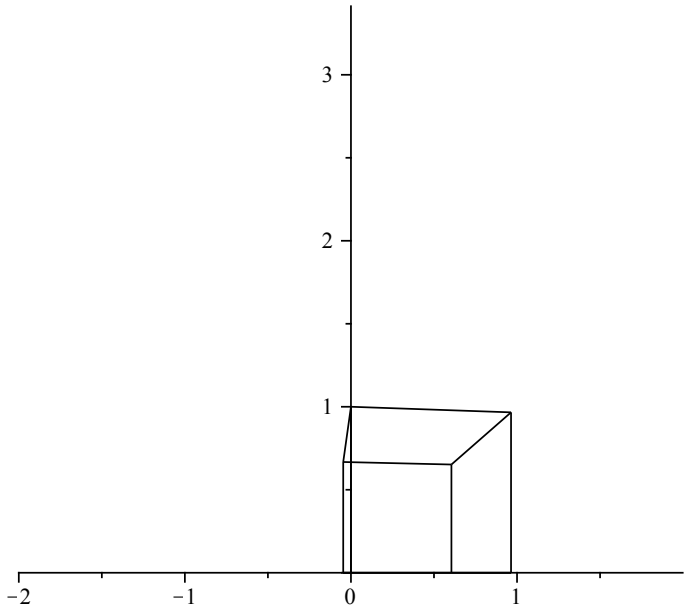
Mapleで動画を作るには次のようにします.

```
> tmp:=[]:
  for i from 0 to 5 do
    coord:=[0,5-i];
    tmp:=[op(tmp), circle(coord,0.5)];
  end do:
  display(tmp,insequence=true,scaling=constrained);
```



先ほどのオイラー角と組み合わせると立方体が原点を中心に回転している様子が見れます.

```
> z_p:=-2:
  tmp:=[]:
  for j from 1 to 90 do
    for i from 1 to 8 do
      c[i]:=convert(Trans1(0,Pi*4/180*j,0).Vector(p[i]),list);
    od:
    points:=pointplot([seq(proj2d(c[i],z_p),i=1..8)]):
    lines:=[seq(line(proj2d(c[(ll[i][1])],z_p),proj2d(c[(ll[i]
  [2])],z_p)),i=1..nops(ll))]:
    tmp:=[op(tmp),display(lines)]:
  od:
  display(tmp,insequence=true,scaling=constrained);
```



Mapleの描画関数の覚書

mapleにはいくつかの描画レベルに合わせた関数が用意されている．どの時にどの関数を使うかの選択指針として，それぞれの関数・パッケージがどのような意図で作られ，どのような機能(library)を担っているか，その依存関係をメモしておく．

描画の下位関数

plot[structure]にあるPLOT,PLOT3Dデータ構造が一番下でCURVES,POINTS,POLYGONS,TEXTデータを元に絵を描く．

plotsパッケージ

簡単にグラフを書くための道具．たとえばここで使ったpointplotはまさに，pointを使って関数を表示する事を当初の目的としている．

PLOTよりもう少し上位で，グラフィックスの基本形状を生成してくれる関数群．arc, arrow, circle, curve, line, point, sphereなどの関数があり，PLOT構造を吐く．表示にはplots[display]を使う．

plottoolsパッケージ