ロボット工学第一
3 D グラフィックによる 3 次元運動解析
(Mathematica を用いて)

平成 28 年 1 月 4 日
山北昌毅

Mathematica 入門

この章では Mathematica の簡単な使い方を解説します。Mathematica は S.Wolfram によって開発された数式処理言語で、一般のプログラム言語と同じように数値計算を行なうとが出来ますが、特に数式をそのまま扱う場合に威力を発揮する言語です。数式処理言語としては業界標準と言ってよいもので、東工大ではサイトライセンスを持っており、大学内では自由に使うことが出来ます。本演習は Mathematica を用いて行列演算と運動解析について習得することを目的とします。そこで、Mathematica を用いるに当たり必要となる最低の機能を説明します。(本演習で用いる Mathematica の機能は限定的なものですので、他の数値計算言語と 3 D グラフィック環境でも本演習の内容は簡単に行なえます。)

Mathematica の起動はデスクトップで Mathematica のアイコンをクリックするだけです。起動すると「Wolfram Mathematica へようこそ」というウェルカムスクリーンが立ち上がりますので、そこにある「新規ドキュメント」のボタンを押してください。新規作製で製作されるウインドウ内でコマンドが実行できますが、コマンドを入れたあとは**シフトボタンとリターンを同時に**押してください。(リターンだけではないことに注意してください。)

1　実行・環境設定とファイル作成

Mathemaitca の起動はユーティリティ・ホールダ **Mathematica8** のアイコンをダブルクリックするだけです。Mathematica の基本機能についてはウェルカムスクリーンのドキュメントボタンを押して、ドキュメントセンターの「コアとなる言語と構成」内の「言語の概要」の「ラーニングソース」の「How Tos」の How to : 変数と関数を使うなどの参考ページで自習してください。

Mathematica を実行すると、ワーキングディクトリは各自のホームディレクトリになります。適当なフォールダに移動するには SetDirectory[] コマンドを使います。使い方は

$$SetDirectory[\text{"移動したいディレクトリ "}];$$

です。(Mathematica での文字列は表現したい文字列を ""で囲みます。)また、コマンドの行は基本的にセミコロンで終わります。Mathematica が立ち上がった時点では、カレントディレクトリは各自のドキュメントホールダになります。

実行の最初にいちいちこのコマンドを使うのは面倒なので、適当な初期化用のコマンド列を書いた実行ファイルを用意しておくと便利でしょう。ファイルの作成は、Mathematica を立ち上げて、ファイルのプルダウンメニューから「新規作成」を選択すればエディターが起動します。適当なコマンドを記述して、同じプルダウンメニュー内の「保存」によりファイルを保存することができます。Mathematica の実行ファイルの拡張子は**.nb** です。

ファイルを読み込み、同時に実行するには、

```
<<"ファイル名";
```

とタイプするだけです。演習で使う関数は robot1.m というファイル内で定義されていますので、まずはこれを各自のドキュメントホールダにダウンロードしてください。(標準設定の場合)

2　ベクトル・行列の表現

数は普通の言語と同じように記述しますが、縦ベクトルと横ベクトルは少し表現が異なります。横ベクトルは要素をカンマで区切って {{}} で囲みます。例えば、

$$[1, 2, 3]$$

は

```
{{1, 2, 3}}
```

と表現します。縦ベクトルの方は各要素を {} で囲って、それらをカンマで区切って更に {} で囲みます。例えば、

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

は

```
{{1}, {2}, {3}}
```

とします。（縦ベクトルと横ベクトルの区別をする必要がない場合は要素をカンマで区切って {} で囲むだけでいいです。制御の分野ではそれらを区別しますので、上記の表現の方が無難です。）ただし、横ベクトルと縦ベクトルの積はスカラー量（実数）になりませんので後に述べるような注意が必要です。Mathematica では、要素をカンマで区切って {} で囲ったものをリストと呼びます。リストの要素はリストでも構いません。

行列は行毎に要素をカンマで区切って {} で囲み、それらをさらにカンマで区切って {} で囲みます。例えば、

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

は

```
{{1, 2, 3}, {4, 5, 6}}
```

のように表現します。行列は常にこのような形で表現されますが、結果を２次元的に表現したい場合は MatrixForm[] という関数を使うと見慣れた形で表示することができます。

変数は基本的に大域変数で型がなく、宣言をせずに使用することができます。（局所変数を宣言することはできますが、ここでは説明しません。）ただし、式を変数に代入する時に注意が必要です。C 言語などでは代入演算は右辺の値が計算されて左辺の変数に代入されますが、Mathematica の場合は必ずしもそうはなりません。変数への式の代入には２種類の演算子があり、'=' と ':=' があります。これは、Mathematica が本来数式を処理する言語であることから来ています。この違いは例を見る方が早いでしょう。

```
x = 1;
y = x;
```

とすると y の値は 1 となります。（文の終りは C 言語と同様；です。）ここで、x の値を未定とすると、つまり、

```
x=.;
```

とすると x の値は未定になりますが（. を代入すると代入された変数の値が未定となります)、y の値は１のままです。ここで、

```
x=2;
```

としても y の値に変化はありません。=の演算では右辺で値の決まっている変数はその数値が変数に代わって演算に使われます。(C 言語などではこれしかありません。) 一方、:=を用いた場合は値ではなくて、形式的な式が左辺に代入されたように動きます。従って、

```
x = 1;
y := x;
```

とした場合、y は x で、x の値が 1 ですので、この時点では y の値は上と同じく 1 となりますが、

```
x=.;
```

とすると、y の値は式 x のままで値は未定となります。ここで、

```
x=2;
```

とすると、y の値は x が 2 ですので 2 となります。=と:=の違いが分かってもらえたでしょうか。（右辺が定数である場合は両者の違いはありません。） :=を用いた方が間違いが少ないですが、変数の値を評価する際にいちいち式を計算しますので効率の悪いプログラムとなってしまいます。また、同じ変数を違う場所で違う値を入れて使うような場合はバグを混入させてしまう恐れがありますので注意が必要です。

3　ベクトル・行列の演算

前節のように縦ベクトルと横ベクトルを区別すると、ベクトルは行列の特殊なものとなりますので、ベクトルと行列は区別する必要がなくなります。したがって、ベクトルと行列はサイズの規則が合っている限り、掛け算は全て. で行なうことができます。（サイズが合っていないと当然演算はできません。）

例えば、

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \qquad v = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

とした場合、Mathematica では

```
M = {{1,2,3},{4,5,6}};
v = {{1}, {2}, {1}};
```

として、

```
M.v
```

とすると行列と縦ベクトルの計算 Mv が行なわれ、結果は縦ベクトルの

```
{{8},{20}}
```

となります。

また、ベクトルや行列から (i,j) 要素を取り出すときは、ベクトル、行列の後に [[i,j]] を付けます。

ここで用いている縦ベクトル、横ベクトルの表現を用いていると、内積に相当する演算や二次形式演算を行なうと結果は実数を {{}} で囲んだ 1 行 1 列の行列となります。このようなものから中の実数を取り出すには [[1,1]] を付けることになります。

Mathematica での行列演算とその関数は次のようになります。ただし、m は行列。

転置	Transpose[m]
共役転置	ConjugateTranspose[m]
逆行列	Inverse[m]
行列式	Det[m]
トレース	Tr[m]
ランク (階数)	MatrixRank[m]
小行列式	Minors[m]
k 番目の小行列式	Minors[m,k]

表 1: Mathematica での行列演算

3.1 ファイル操作

ファイルを読み込み、同時に実行するには前にも説明したように、

```
<<"ファイル名";
```

とタイプします。また、変数の値をファイルに保存するには

```
Save["ファイル名", 変数名 1, 変数名 2,...];
```

と入力します。すでにファイルが存在する場合には、変数名の内容がファイルの最後に追加されます。

C 言語で作ったような数値がテキストとして羅列されたファイルからその数値を読み込むには次のようにします。数値をリストとして読み込むには

```
ReadList["ファイル名", Number]
```

とします。ここで、Number はファイルの中を数値として読み込むように指定するオプションです。また、ReadList 自体が読み込まれたデータに対応するリストになります。一方、ファイル内の数値を行列として読み込むには

```
ReadList["ファイル名", Number, RecordLists->True]
```

を使います。

例えば、data.m というファイルの中身が

```
1.0  2.0  3.0
4.0  5.0  6.0
```

の場合、

```
ReadList["data.m", Number]
```

は {1.0, 2.0, 3.0, 4.0, 5.0, 6.0} を返します。一方、

```
ReadList["data.m", Number, RecordLists->True]
```

は {{1.0, 2.0, 3.0}, {4.0, 5.0, 6.0}} となります。

[練習]

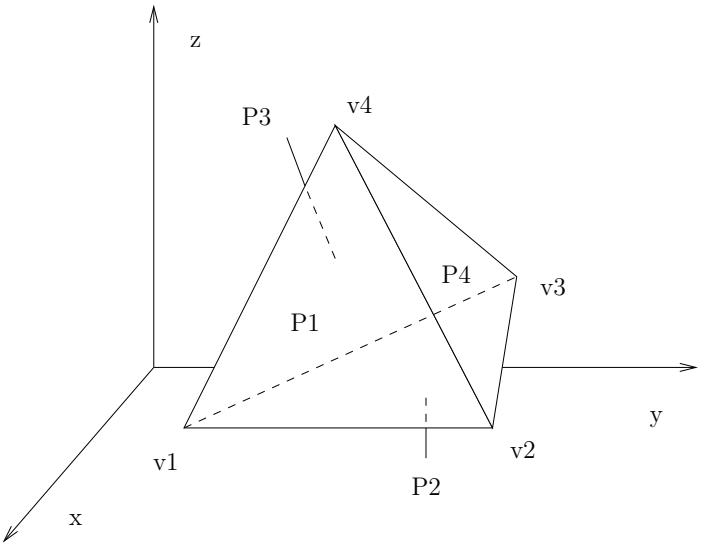
文字列を含む２つのサイズの等しい行列を定義して、４則演算を計算させて MatrixForm により表示しなさい。

[練習]

適当な文字を含む行列を定義して、表 1 の全ての演算を実行しなさい。

[練習]

Mathematica には数式を処理を行う非常に多くの機能が用意されている。紙面や授業時間の関係でその多くをここで紹介することはできません。しかし、Mathematica のヘルプメニューの「ドキュメントセンター」内に各項目について詳しい説明があります。特に、「数学とアルゴリズム」の部分の「数学関数」と微積分の「微分」と関連するチュートリアルの「全微分」は重要ですので、一回はチュートリアルを各自行ってください。(式を修正し、[shift+return] で実行できます)



4　3D グラフィックオブジェクト

本演習では、３次元空間内での剛体を表現する 3D グラフィックオブジェクトとして、表面が凸多面体で構成されたものを考えます。

例として次の三角錐を考えます。頂点を v1,v2,v3,v4 としてその値をそれぞれリストとして

```
v1 = {4, 2, 1};
v2 = {4, 4, 1};
v3 = {1, 3, 1};
v4 = {2, 2, 4};
```

とします。ここでリストの第１要素が x 座標、第２要素が y 座標、第３要素が z 座標です。また、各頂点から構成される３角形の面を P1, P2, P3, P4 とします。頂点のデータから凸多角形を定義するには次のようにします。

```
P1 = Polygon[{v1, v2, v4}];
P2 = Polygon[{v1, v2, v3}];
P3 = Polygon[{v1, v3, v4}];
P4 = Polygon[{v2, v3, v4}];
```

この多角形から多面体を定義するには、多面体を構成する凸多角形をリストにするたけです。今、定義したい三角錐の変数名を Tri とすれば、

```
Tri = {P1, P2, P3, P4};
```

とするだけです。

定義した３次元グラフィックオブジェクトを画面に表示するには次のようにします。

```
Show[Graphics3D[object]];
```

もし表示したいものが複数あれば、object の代わりにそれらをリストにしたものを Graphic3D の引数とすればいいです。

演習で使う関数群は下記 URL の「ロボット工学第一 Mathematica 演習基本ライブラリ」からダウンロード出来ますので、ダウンロードして robot1.m というファイル名で演習用のディレクトリに保存してください。読み込みには次のコマンドを実行してください。

```
Get["robot1.m"];
```

Tri や直方体のデータ Rec がすでに定義されています。また、それらを座標系と一緒に表示する関数 disp が用意されていますので、それらを表示するには

```
disp[Tri];
```

とか、

```
disp[{Tri, Rec}];
```

とタイプするだけでグラフィックオブジェクトが画面に表示されます。

本演習で使用する Mathematica のグラフィックス関係の機能は非常に限定されたものです。より高度な機能を使いたい人はヘルプメニューの「Graphics」の項目を参照してください。

5 ホモジニアス（同次）変換

授業では基本の回転を表す同次変換行列を Rot(x, theta), Rot(y, theta), Rot(z, theta) のように表記しましたが、演習で用いる関数名は回転する軸方向で関数名が異なり、次のように対応しています。

```
Rot(x, theta) -> Rx[theta]
Rot(y, theta) -> Ry[theta]
Rot(z, theta) -> Rz[theta]
```

授業では回転の角度はラジアン（rad) を使っていますが、演習で用いる回転の関数は回転角度の引数が手入力しやすいように度 (degree) になっていますので注意してください。

また、並進は授業では Trans(x, d), Trnas(y, d), Trans(z, d) のように記述していましたが、プログラムでは次のように対応します。

```
Trans(x, d) -> Tx[d]
Trans(y, d) -> Ty[d]
Trans(z, d) -> Tz[d]
```

グラフィックオブジェクトの回転、移動には変換したいオブジェクトの変数と回転や移動に使いたい同次変換行列を trans という関数に渡します。これによって、オブジェクトを構成している全ての頂点のデータが渡された同次変換行列によって変換され、その結果が trans 自身の値として返されます。

例えば、Tri を x 軸周りに 90 度回転したものを Tri2 として得たければ、

```
Tri2 = trans[Rx[90.0], Tri];
```

とすれば良いです。また、行列の積は. ですから、Tri を x 軸周りに 90 度回転したものを、さらに絶対座標系で y 軸周りに 90 度回転したものを Tri3 として得たければ、

```
Tri3 = trans[Ry[90.0].Rx[90.0], Tri];
```

とすればよいです。（もちろん、Tri3 = trans[Ry[90.0], Tri2]; としても良いですが。）

[演習 1]

1. 最初絶対座標系と直方体に取り付けた動座標系は一致していると仮定します。(disp[Rec] で表示される状態です。) Rec を x 軸周りに 90 度回転し、さらに絶対座標系で y 軸周りに 90 度回転したものを表示しなさい。また、Rec を x 軸周りに 90 度回転し、さらに回転された座標系 (物体) の y 軸周りに 90 度回転したものを表示し、プリントアウトしなさい。(画面のプリントアウトは、ウインドウの右側の範囲指定括弧を選択し、部分印刷でプリントアウトすることができます)

2. 同次変換行列をロール・ピッチ・ヨー角 (phi, theta, psi) と直行座標 (x,y,z) によって定義しなさい。定義した変数をリスト形式又は行列表現で表示しなさい。

3. 直方体 Rec は z 方向に厚さが１である。一つの直方体の上に z 方向にもう一つの直方体がくっついたような絵を描きたい。（以後下の直方体を Rec1, 上のものを Rec2 とする。） Rec1, Rec2 とともに移動する座標系を前問と同じように同次変換行列として定義しなさい。ただし、Rec1 の変数を操作すると、Rec2 はあたかも Rec1 にくっついているように座標系が変化するようにしなさい。Rec1 を適当に移動または回転させたとき、Rec2 がそれに伴って移動することが分かるものを表示し、プリントアウトしなさい。

[演習２]

disp[Tri] で４面体を表示しなさい。この時、カメラ座標系は図示された座標系と一致しているものとする。このとき次の問に答えよ。ただし、どのように計算したかが分かるようにしなさい。また、逆行列を計算する Mathematica のコマンドは Inverse である。例えば、行列 M の逆行列は Inverse[M] により計算される。

1. カメラ座標系を y 軸方向に-2 だけ動かした時、カメラ座標系で Tri がどのように見えるか表示しなさい。（ヒント：適当に Tri に座標変換を施し、disp コマンドにより表示すれば良い）

2. 更にカメラ座標系を y 軸周りに 10 度回転し、回転された z 軸周りに２０度回転した。この時、カメラ座標系で Tri がどのように見えるか表示しなさい。

Mathematicaによる関数の微分

Mathematicaでは数式 (行列も含む) をそのまま加減乗除や微分積分を行うことが出来ます。ここでは、数式や関数を定義して、それの微分を計算する方法について説明します。

数式の微分を行うMathematicaのコマンドには大きく分けて以下のような2つのものがあります。

表現	意味
D[f,x]	数式fをxにより偏微分する
Dt[f, x]	数式fをxにより全微分する

（多変数の数式を同時に複数の変数で微分を行う操作に関してはマニュアルを参照のこと）

例えば、数式fを

```
f = x^2 + y^2
```

とします。fをxによって偏微分するには

```
D[f,x]
```

とすると2xという結果を得ます。一方、fをxによって全微分するには

```
Dt[f,x]
```

とします。そうすると結果は

```
2x + 2 Dt[y,x]
```

となります。全微分の場合には、yがxと独立でない可能性があるためyのxによる全微分の項が残ります。

一方、一変数の未知な関数fのその変数による一階の微分は次のように表します。

表現	意味
f’	**一変数の関数**の微分の表現

また、微分を取ったあとに変数にxを代入したものを表現するには

```
f'[x]
```

とします。ここで、fは具体的に決定されている必要がないことに注意します。また、式の表現から類推できますが、fの二階の微分は

```
f''[x]
```

と表現されます。

例として、平面内の質点の座標を (x,y) として、(x,y) が時刻tの関数として

$$x = t, y = sin(t)$$

で決まるとき、それぞれの点に対して関数Uが

$$U = x^2 + y^2$$

で与えられるとします。このときの、点の時間変化に伴う関数Uの時間変化を計算したいとします。そのためには、次のように計算すれば良いことになります。

```
x = t;
y = Sin[t];
U = x^2 + y^2;
D[U,t]
2 t + 2 Cos[t] Sin[t]
```

ただし、Sin[] は Mathematica の組み込み関数です。また、この場合 Dt を使っても結果は同じです。

関数の偏微分を求めることは関数 D[] により計算することができますが、微分の連鎖公式を用いて次のように求めることもできます。U の x, y それぞれの偏微分を求めたい時は、時間 t で全微分して、x, y の時間微分の微係数を取り出すことによっても求めることができます。x の多項式 f があるとき、f から x の 1 次の係数を取り出すには

```
Coefficient[f, x]
```

とすれば取り出すことができます。これを用いて、Ux, Uy をそれぞれ、U の x, y それぞれの偏微分とすると、

```
Ux = Coefficient[Dt[U, t], Dt[x,t]];
Uy = Coefficient[Dt[U, t], Dt[y,t]];
```

によっても計算することができます。この考え方は、多変数関数のヤコビ行列を全微分の関係から計算するときや、微分方程式の係数を計算するのに便利です。

また、微分演算や行列演算を行うと式の項数が非常に多くなり、式が見にくくなってきます。消去可能な項や三角関数の公式などにより式の項数を少なくして式を見やすくするコマンドとして **Simplify**[] があります。また、行列の転置を計算するコマンドとして、**Transpose**[] がありますので、必要に応じて使用してください。

[演習 3]

回転行列 R をロール・ピッチ・ヨー角 (phi, theta, psi) を用いて表したものとします。このとき、次のものを計算しなさい。

1. 時刻 t での角速度ベクトル ω を次の定義式に従って計算しなさい。

$$\omega\times = \dot{R}R^T$$

   [ヒント] 右辺を Dt[] を用いて計算し、角速度ベクトルの要素に対応する部分を取り出す。

2. 今 $\dot{\alpha}$ を

$$\dot{\alpha} = [\dot{phi}, \dot{theta}, \dot{psi}]^T$$

   と定義する。ここで、

$$\omega = J_a\dot{\alpha}$$

   となる行列 J_a を求めなさい。また、J_a のそれぞれの列ベクトルが $\dot{phi}, \dot{theta}, \dot{psi}$ のそれぞれの瞬間回転軸になっていることを確かめなさい。

   [ヒント] ω から J_a のそれぞれの列ベクトルは、ω の $Dt[phi, t]$ などの係数を取り出すことにより求まる。

3. 上で求めた J_a はロール・ピッチ・ヨー角の時間微分からそれに対応する角速度ベクトルを求める式となっており、それぞれの角度の時間変化の寄与分はそれぞれの瞬間回転軸成分の和で求められることを示している。このことを角速度ベクトルの座標変換などを使って手計算による式によって示しなさい。

DH パラメータと変換方程式による動座標系で見た物体の表示

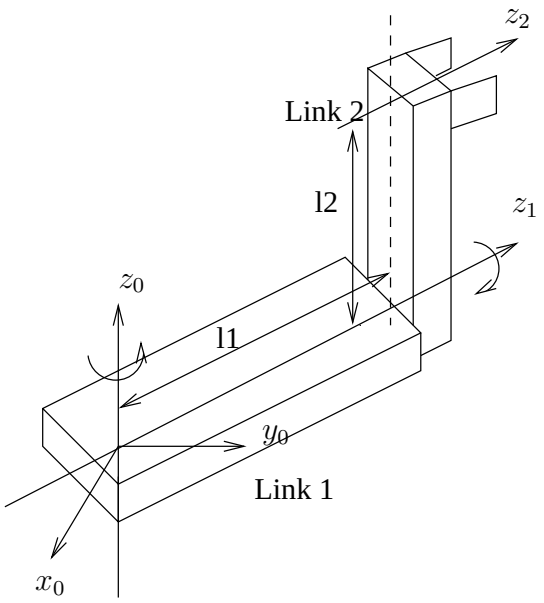
[演習 4]

下の図のような 2 リンクマニピュレータを考える。この時、以下の設問に答えなさい。

1. DH 表記法にしたがって座標系を設定した場合の座標系を図に書き込みなさい。ただし、z_0, z_1, z_2 は図のように取り、変数以外の DH パラメータは正の値になるようにしなさい。

2. リンク 1、リンク 2 の回転角を theta1, theta2 とし、それぞれを変数とするリンクの相対運動を表す同次変換行列 $A1(theta1), A2(theta2)$ を DH パラメータを用いて定義しなさい。ただし、Mathematica では π は Pi である。

3. $theta1 = 0, theta2 = 0$ の場合のマニピュレータの姿勢を図に書き込みなさい。



4. disp[Rec] で直方体を表示しなさい。（Rec のそれぞれの辺の長さは 1,1,3 である）Rec を x,y 方向にそれぞれ 3 だけ並行移動したものを Rec2 とし、disp[Rec2] で移動後の直方体を表示しなさい。上のマニピュレータで l1=1, l2=1 とする。$theta1 = \pi/2, theta2 = 0.0$ の時、手先の座標系 $O_2 - x_2y_2z_2$ で見た Rec2 を表示しなさい。ただし、逆行列を計算する Mathematica のコマンドは **Inverse**[] である。例えば、行列 M の逆行列は Inverse[M] により計算される。

速度関係（ヤコビ行列）を用いた逆運動学計算

Mathematica での制御構造は次のように記述します。（ここでは、最低限のことを説明しますので、詳しくはヘルプの式の評価の条件子、ループと制御構造体の部分を参照してください）

繰り返し 繰り返しは次のように記述します。

```
For[start, test, incr, body]
```

ここで、start は初期化で、test を評価した結果が真であれば body を実行して、incr を実行する。

条件分岐 条件によりプログラムの実行を制御するには次のように記述します。

```
If[p, then, else]
```

ここで、p は関係式で p が真なら then の式を実行し、偽であれば else の式を実行します。関係式は C 言語の関係演算式と同じようなものが使用可能です。

また、Mathematica でデータからグラフをプロットするには ListPlot という関数を利用します。（関数を与えてグラフをプロットする方法は次回に説明します）例えば、横軸 x と縦軸 y のデータの組み (xi, yi)（i=1,...,N) が与えられているとき、それらのデータからリストを作り、次のように呼び出します。

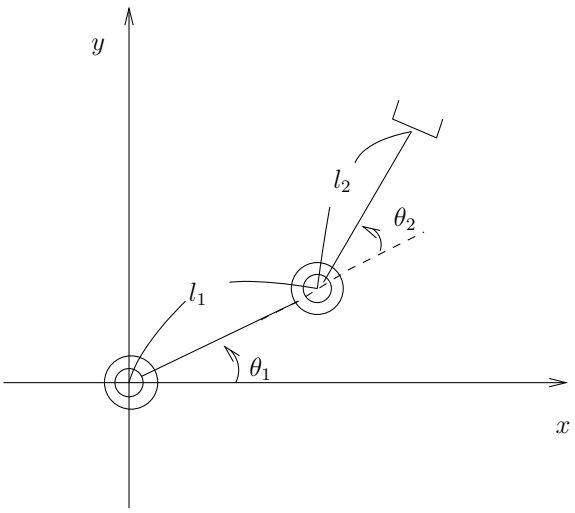
```
ListPlot[{{x1,y1},{x2, y2},..., {xN, yN}]
```

For 文と ListPlot を併用して、y=2x+1 のグラフを x が 0 から 10 までを 0.1 刻みでプロットするには次のようにプログラミングすればいいです。ただし、データをリストとして追加するのに **Append** というリストを操作する関数を用いています。

```
For[i=0.0;data={},i<=10.0,i++,data=Append[data,{i, 2*i+1}]]
```

Mathematica では C 言語のカンマとセミコロンの役割が反転していることに注意してください。

[演習５]



1. 関節角度ベクトルを $q := [\theta_1, \theta_2]^T$、マニピュレータの手先の x, y 座標を x_h, y_h とし、作業座標ベクトルを $x := [x_h, y_h]^T$ とする。このとき、$\dot{q}$ と $\dot{x}$ の速度関係を表すマニピュレータ・ヤコビヤン (ヤコビ行列) を求めなさい。

2. 逆運動学を数値的に解くことを考える。x の目標値 xd を与えたとき、xd を実現する qd を最急降下法によって求め、繰り返しの回数 i と求められた qi, xi をグラフにプロットしなさい。そうして、求められた解が十分な精度で xd を実現していることを確かめなさい。ただし、$l_1 = 0.5[m], l_2 = 1[m], x$ で $xd = [0.7, 0.0]$ とし、q の初期値を $[0.1, 0.0]^T$ と $[-0.1, -0.0]^T$ の 2 通りで実行した結果を求めなさい。また、最急降下法のパラメータは適当に決めなさい。(このアルゴリズムは収束が遅いので繰り返しの回数は多くとる必要があることに注意)

３Ｄグラフィックによる可操作性の可視化

Mathematica には１変数、２変数の関数に対して、関数の値を知りたい定義域の集合（変数の取り得る範囲）を指定して、対応する値を様々な形の２次元または３次元内のグラフとして表示する機能があります。ここでは、２変数関数の値を３次元のグラフとして表示する方法について説明します。

ここでは、２変数関数である次の Gauss 分布関数を考えます。

$$p(x,y) = \exp\{-x^2 - y^2\} \tag{1}$$

この関数の $-2 \leq \quad x \quad \leq 2, -2 \leq \quad y \quad \leq 2$ での範囲の値を、横軸を x, 縦軸を y, とし $p(x,y)$ の値を z 軸方向にプロットしたグラフを表示するには Mathematica では次のようにします。

```
p := Exp[-x^2-y^2];
Plot3D[p,{x,-2,2},{y,-2,2}];
```

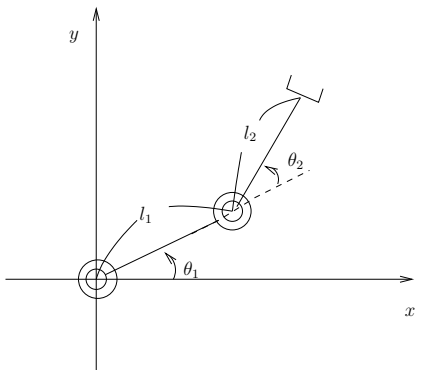
また、表示される平面の分割数を指定するには、関数にオプションとして分割数 PlotPoints を次のように指定します。例えば、分割数を４０にしたい時は

```
Plot3D[p,{x,-2,2},{y,-2,2}, PlotPoints -> 40];
```

とします。

[演習 6]

ロボットマニピュレータが物体などを操作するのに都合のよい姿勢を判定する指標の一つに可操作性というものがある。ここでは、その指標を視覚的に表示し、作業に適した姿勢の概形を取得するものとする。以下の図で示す平面内２リンクマニピュレータを考え、その可操作性を３次元グラフで表示することを考える。以下の設問にしたがってグラフを完成させなさい。



1. $-\pi \leq \theta_1 \leq \pi, -\pi \leq \theta_2 \leq \pi$ の範囲での可操作性を３次元グラフ表示しなさい。ただし、リンクの長さはそれぞれ、$l_1 = 0.5[m], l_2 = 1[m]$ とする。ただし、Mathematica での行列式、平方根は Det[], Sqrt[] で計算できる。
2. 可操作性のもっとも大きい姿勢の概形を手書きで描きなさい。(この場合はユニークではない)
3. (オプション) 余裕のある人は、手先の位置 (x_h, y_h) に対して、可操作性がどのような値になるかを３次元グラフで表示しなさい。全領域ではなくて、手先が到達可能な適当な矩形領域で良い。また、Mathematica で $sin^{-1}(y)$,$cos^{-1}(x)$, $tan^{-1}(y/x)$ はそれぞれ ArcSin[y], ArcCos[x], ArcTan[x,y] で計算できる。

Mathematica による Euler-Lagrange の運動方程式の導出

考えている対象の機械系に対して、一般化座標ベクトル q を導入して Lagragian $L(q,\dot{q})$ を記述すると形式的に Lagrange の運動方程式を導出することができ、他節の手法と組み合わせると対象の数値シミュレーションを簡単にすることができることになります。Lagrangian から、Lagrange の運動方程式は

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = F_i \quad (i = 1, .., n)$$

を形式的に計算するだけですので、原理的には非常に簡単です。この方程式の左辺を更に詳しく書くと

$$\sum_{j=1}^{n} \{d_{ij}(q)\ddot{q}_j\} + h_i(q,\dot{q}) = F_i \quad (i = 1, ..., n)$$

となります。しかし、複雑なシステムに対して左辺の計算を手作業で行なおうとすると、計算の途中で式が複雑になりあまり現実的ではありません。これに対して、Mathematica では、数式で表された関数の微分を行なう機能がありますので、比較的簡単に Lagrange の運動方程式を導くことができます。(ただし、Mathematica を用いて Lagrange の運動方程式を導いても、システムが複雑になると式の中に現れる項数が膨大になり、これを数値シミュレーションにそのまま用いるのは実用的ではなく、純粋に数値的な解法に頼る方が賢明だと考えます。) ただ、3 リンク程度のマニピュレータの運動方程式を導出するのには便利ですので、ここではそのやり方を説明します。(手計算で導出したものの確認にも使えます) 以下の説明では、$q_i(t)$ などの関数や変数の下付きの数字は Mathematica のプログラムの中では下付きにしない関数名や変数名で表します。例えば、q_i は qi と表記します。

Mathematica の中で時間 t の関数 $q_i(t)(i = 1, ..., n)$ を定義するには、**qi[t]** と記述します。(関数の引数を書く所が'(' ではなくて'[' を使う) また、qi[t] の時間での偏微分は **D[q[t],t]** あるいは、**qi'[t]** と記述します。ここで考えている関数は t のみの関数ですので、t での偏微分は t での全微分に相当します。**D[] は１番目の引数の関数を２番目の引数の変数で偏微分したものを計算する関数です。**(Mathematica には全微分を行なう Dt[] という関数もありますが、ここでは説明しません) L を定義するには、qi[t] と qi'[t] を用いて、具体的に運動エネルギとポテンシャルエネルギを定義するだけです。(この部分は省略します。簡単ですが、一番面倒なのはこの部分です。)

L が qi[t] と qi'[t] を用いて記述されると $\frac{\partial L}{\partial \dot{q}_i}$ は

```
D[L, qi'[t]]
```

と計算されます。$\frac{\partial L}{\partial \dot{q}_i}$ を

```
pLpqi = D[L,qi'[t]];
```

とおきます。これを更に時間 t で全微分すると $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right)$ が計算できますので、それは

```
D[pLpdqi, t]
```

により計算することができます。これで、Lagrange の運動方程式の左辺第一項が計算できました。左辺第二項の $\frac{\partial L}{\partial q_i}$ は

```
D[L, qi[t]]
```

により計算できます。したがって、Lagrange の運動方程式の i 番目の変数に関する運動方程式の左辺を eqi とすると

```
eqi = D[pLpqi,t] - D[L,qi[t]]; (* eqi = D[D[L,qi'[t]],t] - D[L,qi[t]]; *)
```

により計算できます。eqi は $\ddot{q}_i(t), \dot{q}_i(t), q_i(t)$ に相当する qi"[t], qi'[t], qi[t] の関数になっています。ただし、**qi"[t] は qi[t] を時間 t で２回微分したものです**。数値シミュレーションには慣性行列の要素である d_{ij} が必要ですから、eqi の中の $\ddot{q}_j$ つまり、qj"[t] の係数を取り出す必要があります。そのためには、式の中から指定した項の係数を取り出す **Coefficient[]** という関数を使います。Coefficient[] は第 1 引数の式から、第 2 引数で表される項の係数を取り出す関数です。eqi から dij を取り出すには全ての j について

```
dij = Coefficient[eqi, qj''[t]];
```

とすれば良い事になります。一方、Lagrange の運動方程式の左辺残りの項に相当する $h_i(q,\dot{q})$ を取り出すには、計算された dij を使って

```
hi = eqi - di1*q1''[t] - di2*q2''[t]-...-din*qn''[t];
```

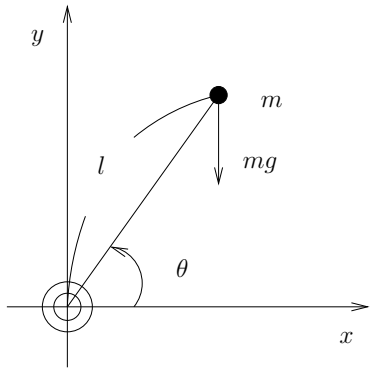
を計算すれば良い事になります。あるいは、全ての i について

```
qi''[t]=0;
```

とすると hi の中身が求めるものになります。

これで、シミュレーションに必要な全ての式が得られます。

例えば、授業で説明した下図のような系では次のように計算できます。




```
x = l*Cos[q1[t]];
y = l*Sin[q1[t]];

dxdt = D[x,t];
dydt = D[y,t];

K = m*(dxdt*dxdt+dydt*dydt)/2;
U = m*g*l*Sin[q1[t]];

L = K - U;

pLpdq1 = D[L, q1'[t]];
eq1 = D[pLpdq1,t] - D[L,q1[t]];

eq1 = Simplify[eq1];

d11 = Coefficient[eq1, q1''[t]];
h1 = eq1 - d11*q1''[t];
h1 = Simplify[h1];
```

これで d11 = ml ^{2}, h1 = mgl Cos[q1[t]] として求まります。上のプログラムで **Simplify[]** は得らえる式が簡潔になるように適宜使用しています。

[注意] 簡単なシステムでよいので、手計算による Lagrange の運動方程式の導出に慣れてから Mathematica などの数式処理システムを用いるようにしてください。くれぐれも Mathematica などの数式処理システムに頼り過ぎないように。

Mathematica による数値シミュレーション

多変数の常微分方程式を統一的に解くために, まず方程式を状態のベクトルの 1 階の微分方程式（状態方程式表現）で表します。例えば, 一般の n 自由度のメカニカルシステムでは, 一般化座標ベクトルを $q := [q_1, q_2, \cdots, q_n]^T$ とした場合, 運動方程式は

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = F \tag{2}$$

のように表現できます。このとき, 状態ベクトルを

$$x(t) := \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}, x_1(t) := q(t), \quad x_2(t) := \dot{q}(t) \tag{3}$$

とすると, 状態ベクトルの時間微分 $\dot{x}$ は

$$\dot{x}(t) = \frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ D^{-1}(x_1)(-C(x_1,x_2)x_2 - G(x_1) + F) \end{bmatrix} \tag{4}$$

となり, 状態ベクトルの一階の常微分方程式が得られます。

Mathematica ではこのような状態方程式に対して, 初期状態 $x(t_s)$ と求めたい解の時間区間 $[t_s, t_e]$ を与えて, $x(t), (t_s \leq t \leq t_e)$ を数値的に求める関数 NDSolve 関数が用意されています。

状態方程式が

$$\frac{d}{dt}\begin{bmatrix} x1[t] \\ x2[t] \\ . \\ xN[t] \end{bmatrix} = \begin{bmatrix} f1(x1[t], x2[t], \cdots, xN[t]) \\ f2(x1[t], x2[t], \cdots, xN[t]) \\ . \\ fN(x1[t], x2[t], \cdots, xN[t]) \end{bmatrix}, x1[0] = x10, x2[0] = x20, \cdots, xN[0] = xN0 \tag{5}$$

として表現されていた場合, 時間区間 $[0, T]$ での解を求めたい場合次のようにします。

```
NDSolve[{x1'[t]==f1(x1[t], x2[t], ..., xN[t]), ...,xN'[t]==fN(x1[t],x2[t], ..., xN[t]),
        x1[0]==x10, ..., xN[0]==xN0},{x1,x2, ...,xN}, {t, 0, T}]
```

' や==の演算子の使い方に注意してください。' は微分を表し、==は定義を行なっています。つまり,NDSolve の引数は NDSolve[微分方程式と初期状態, 状態変数のリスト, 求解時間区間] のように指定します。これにより,x1[t],.., xN[t] が t の時間関数として計算されます。計算結果をプロットしたい場合には,

```
sol = NDSolve[{x1'[t]==f1(x1[t], x2[t], ..., xN[t]),...,xN'[t]==fN(x1[t],x2[t], ..., xN[t]),
            x1[0]==x10, ...,xN[0]==xN0},{x1,x2, ...,xN}, {t, 0, T}]
```

として,

```
Plot[Evaluate[{x1[t], x2[t], ..., xN[t]} /. sol], {t, 0, T}]
```

とすれば x1[t],...,xN[t] がグラフ表示されます。(Evaluate や/. の使い方はここではこのように使うものだと思って使ってください)

例えば, 振り子の運動方程式が

$$ml^2\ddot{\theta} + \mu\dot{\theta} + mglcos(\theta) = 0 \tag{6}$$

として与えられていた場合, この状態方程式表現は $x_1 := \theta, x_2 = \dot{\theta}$ とすると

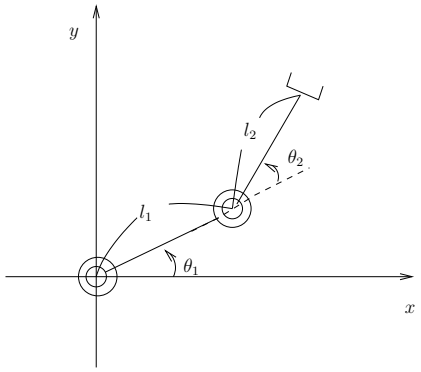
$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ (-\mu x_2 - mglcos(x_1))/(ml^2) \end{bmatrix} \tag{7}$$

と書き直せるので,$m = 1.0, \mu = 0.1, l = 0.5, g = 9.8$, 初期状態を $\theta(0) = 0, \dot{\theta}(0) = 1$ として時間区間 $[0, 10]$ での $x(t)$ を求めてプロットしたい場合は次のようにプログラムすることになります

```
f1 := x2[t];
f2 := (-mu*x2[t]-m*g*l*Cos[x1[t]])/(m*l*l);
m = 1.0;
mu = 0.1;
l = 0.5;
g = 9.8;
sol = NDSolve[{x1'[t]==f1, x2'[t]== f2,x1[0]==0.0, x2[0]==1.0},{x1,x2},{t,0,10}];
Plot[Evaluate[{x1[t],x2[t]} /. sol] ,{t,0,10}];
```

[演習 7]

下の図で示す垂直平面 内 2 リンクマニピュレータを考え, 動作を数値シミュレーションすることを考える。この時, 設問に答えながらシミュレーションを完成させなさい。



1. マニピュレータの根元からリンク 1 , リンク 2 とし, それぞれのリンクの質量, 慣性モーメントを m_i, I_i, 各リンクの重心はリンクの中央にあるとし、重力加速度を g としたとき,Lagrange の運動方程式を求めなさい。ただし,θ_1, θ_2 に双対なトルクを τ_1, τ_2 とする。

2. 入力トルクをゼロとし,$l_1 = 0.5, l_2 = 1.0, m_1 = 2.0, m_2 = 1.0, I_1 = 0.1, I_2 = 0.01, g = 9.8$, 初期状態を $\theta_1(0) = 0.0, \theta_2(0) = 0.2, \dot{\theta}_1(0) = 0.0, \dot{\theta}_2(0) = 0.0$ としたときの t が $[0, 10]$ での応答をプロットしなさい。

Mathematicaによる動画表示

講義では運動学に関するトピックスを一通り学習しました。これまでの学習の成果を利用することにより、各人が機構設計したロボットを３ D グラフィックとして画面内で動かすことができます。今回はその方法について演習を行います。

手順は次のようになります。

1. マニピュレータの各リンクに DH 表記などにより座標系を取り付ける。
2. 各リンクを簡単な形状の多面体で近似し、その頂点の座標を取り付けられた座標系で決定し、取り付けられた座標系での多面体を定義する。(多面体の頂点の数などを増やすとリアルな表示となるが、本演習では簡単のため直方体とする）
3. ジョイントの変数（回転ジョイントなら θ, 直動ジョイントなら d) を一般化座標に選び、それを Mathematica の変数とする。
4. 一定時間毎の変数の値を決定する。適当なポーズの離散的なデータから時間関数を適当な方法で補間するか、運動方程式に基づいて数値シミュレーションを行う。
5. 得られた時間関数（又はデータ列）に対して、適当な時間間隔での一般化座標値を用いて、全ての多面体を DH 表記などに基づいてプロットする。つまり、DH 表記に従って、各リンクに取り付けられた座標系の絶対座標系での同次変換行列表現を求め、その変換行列を用いて多面体データを変換して 3D グラフィックで表示する。

具体的に３ D グラフィックデータを用いてアニメーションするには、Append 関数により幾つかの３ D フラフィックデータのリストを作り、ListAnimate 関数によりそれを表示します。例えば、空のリスト view を用意して、これに object という多面体の表示データを追加するには

```
view = {};
view = Append[view, Graphics3D[object]];
```

とすればいいです。

例えば、振り子の角度を $sin(2\pi t)$ として、３ D グラフィック表示するプログラムは次のように与えられます。(ただし、１秒毎の状態をコマ送りで表示しています。)

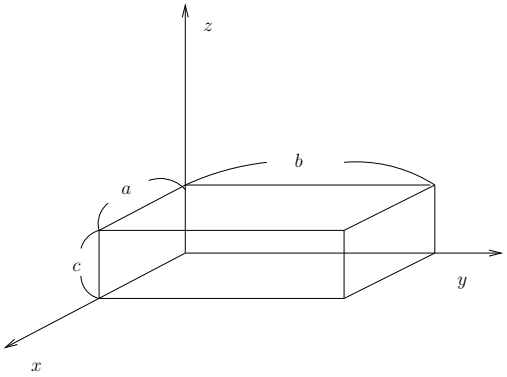
```
view = {};

Rec = DefRec[1,3,1];

For[i=1,i<=10,i++,
   theta = Sin[2.0*Pi*i];
   Recs1 = trans[Rx[theta],Rec];
   view = Append[view, Graphics3D[{coord, Recs1},
          DisplayFunction -> Identity, PlotRange ->{{-2,12},{-2,12},{-2,12}}]];
]

ListAnimate[view]
```

ただし、この例では回転角を x 軸周りに取り、直交座標系も同時に表示するよう、coord オブジェクトもリスト内で指定しています。Graphics3D コマンドに与えている DisplayFunction -> Identity, PlotRange ->{{-2,12},{-2,12},{-2,12}} は、表示に関するオプションで、DisplayFunction -> Identity とすることによりグラフィックの表示を禁止しデータだけの作製をさせることができます。また、PlotRange ->{{-2,12},{-2,12},{-2,12}} で対象の３次元空間のどの領域を表示するかを指定しています。

プログラムでは直交座標系内でそれぞれ３辺の長さ a,b,c を指定して、直方体の多面体データを定義する関数 DefRec を用いています。

```
Rec = DefRec[1,2,3];
disp[Rec];
```

により、指定した３辺の長さをもつ直方体が表示されるのを確認してください。



Mathematicaによる数値シミュレーション結果の動画表示

前の演習で、運動方程式と入力の時間関数が与えられた場合、システムの状態がどのように時間応答するかを計算する方法を学びました。また、この演習の初期に直方体などで表される剛体が定義されている時、座標系を適当に導入することにより 3D グラフィックで表示する方法を学びました。これらを組み合わせることにより、運動方程式で与えられたシステムが、３次元空間内でどのような挙動をしているのかを 3D グラフィックで表示することができます。

手順は次のようになります。

1. マニピュレータの各リンクに DH 表記などにより座標系を取り付ける。
2. 各リンクを簡単な形状の多面体で近似し、その頂点の座標を取り付けられた座標系で決定し、取り付けられた座標系での多面体を定義する。(多面体の頂点の数などを増やすとリアルな表示となるが、本演習では簡単のため直方体とする）
3. ジョイントの変数（回転ジョイントなら θ, 直動ジョイントなら d) を一般化座標に選び、運動方程式を導出する。
4. 上で得られた運動方程式に対して、想定する条件で数値シミュレーションする。
5. 得られたシミュレーション結果に対して、適当な時間間隔での一般化座標値を用いて、全ての多面体を DH 表記などに基づいてプロットする。つまり、DH 表記に従って、各リンクに取り付けられた座標系の絶対座標系での同次変換行列表現を求め、その変換行列を用いて多面体データを変換して 3D グラフィックで表示する。

先の演習の例では、数値シミュレーションデータをプロットする方法を以下のように説明しました。

```
sol = NDSolve[{x1'[t]==f1, x2'[t]== f2,x1[0]==0.0, x2[0]==1.0},{x1,x2},{t,0,10}];
Plot[Evaluate[{x1[t],x2[t]} /. sol] ,{t,0,10}];
```

数値シミュレーションで得られた解は sol という変数に格納され、解は多項式データとして表されています。このような状況で時刻 t での状態の値 x1[t], x2[t] はそれぞれ、**(x1[t] /. sol)[[1]], (x2[t] /. sol)[[1]]** として表されます。また、幾つかの３ D グラフィックデータを用いてアニメーションするには、Append 関数により幾つかの３ D フラフィックデータのリストを作り、ListAnimate 関数によりそれを表示します。例えば、空のリスト view を用意して、これに object という多面体の表示データを追加するには

```
view = {};
view = Append[view, Graphics3D[object]];
```

とすればいいです。

振り子のシミュレーションデータを３ D グラフィック表示するプログラムは次のように与えられます。(ただし、１秒毎の状態をコマ送りで表示しています。)

```
f1 := x2[t];
f2 := (-mu*x2[t]-m*g*l*Cos[x1[t]])/(m*l*l);
m = 1.0;
mu = 0.01;
l = 0.5;
g = 9.8;
sol = NDSolve[{x1'[t]==f1, x2'[t]== f2,x1[0]==0.0, x2[0]==1.0},{x1,x2},{t,0,10}];

view = {};

Rec = DefRec[1,3,1];
```

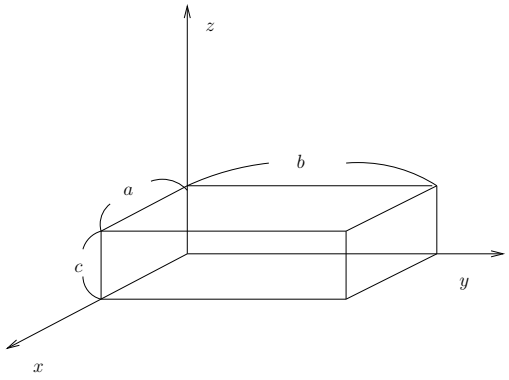
```
For[i=1,i<=10,i++,
   theta = (x1[i] /. sol)[[1]];
   Recs1 = trans[Rx[theta],Rec];
   view = Append[view, Graphics3D[{coord, Recs1},
          DisplayFunction -> Identity, PlotRange ->{{-2,12},{-2,12},{-2,12}}]];
]

ListAnimate[view]
```

ただし、この例では回転角を x 軸周りに取り、直交座標系も同時に表示するよう、coord オブジェクトもリスト内で指定しています。Graphics3D コマンドに与えている DisplayFunction -> Identity, PlotRange ->{{-2,12},{-2,12},{-2,12}} は、表示に関するオプションで、DisplayFunction -> Identity とすることによりグラフィックの表示を禁止しデータだけの作製をさせることができます。また、PlotRange ->{{-2,12},{-2,12},{-2,12}} で対象の３次元空間のどの領域を表示するかを指定しています。

プログラムでは直交座標系内でそれぞれ３辺の長さ a,b,c を指定して、直方体の多面体データを定義する関数 DefRec を用いています。

```
Rec = DefRec[1,2,3];
disp[Rec];
```

により、指定した３辺の長さをもつ直方体が表示されるのを確認してください。



また、Mathematica での for ループの指定は次のようになっています。

```
For[ループ変数の初期化, 繰り返し条件, ループ変数の更新, 繰り返しの本体]
```

[演習 8]（演習８の結果は試験期間終了日までに山北に提出）

上に挙げた手順に従って、演習７での数値シミュレーション結果を３ D グラフィックにより表示しなさい。