98/11/24 法政大学工学部システム制御工学科　小林

自律走行車用プログラムの仕様

MATLAB によるリアルタイム制御のための MEX インターフェース環境はほぼ整いつつある．ここでは、これらインターフェース仕様をまとめさらに自律走行車を開発する上で効率よく開発するインターフェースを再定義する．提案する仕様は、各自それぞれの要素開発に際し必要な情報のみを公開することで全体のプログラム構成を抽象化できる．それぞれの機能のチューニングに専念できる環境を整えることが目的である．

MATLAB による自律走行プログラムの概要

現在、想定している自律走行車は、

1.シミュレーションによるもの

2.Rug-Warrior による実機モデル

3.遊歩パワーチェアによる実機モデル

である。これら３つにおいてそれぞれをモジュール化し相互に使用できるようにする。

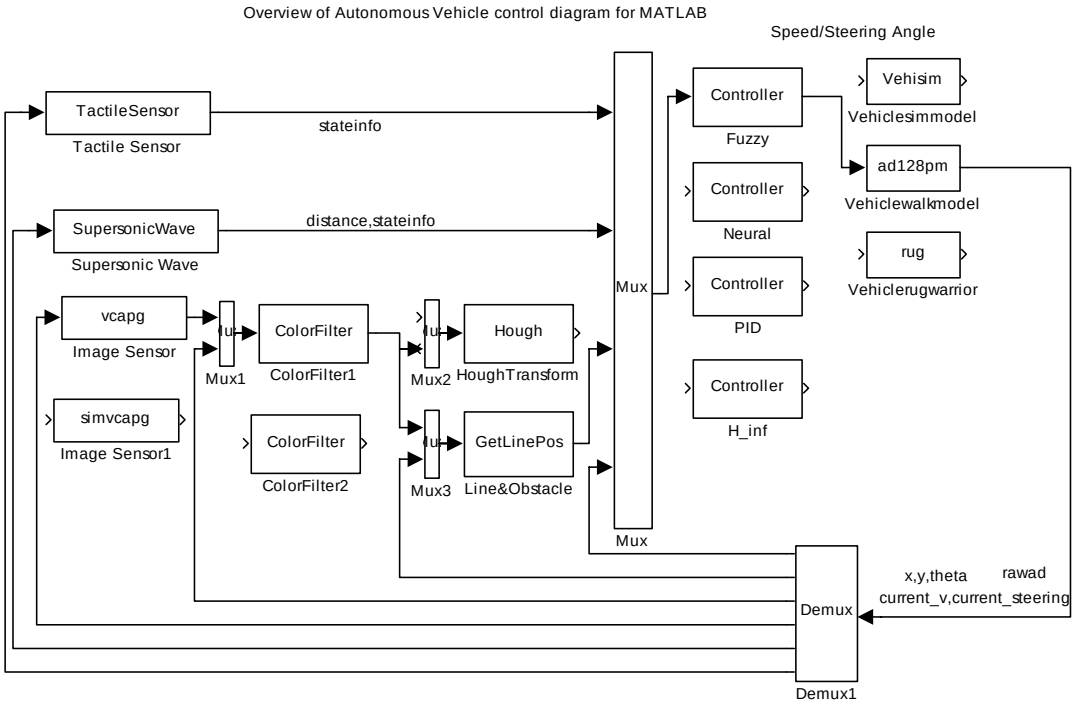


Figure 1 Schematic diagrams for Autonomous unmanned vehicle control for MATLAB environment.

Fig.1 に MATLAB で記述するブロックの概要を示す．ブロックは、左の列から順に Tactile センサ、image Sensor に代表される**センサブロック**、Color filter,Hough Transform, Getlinepos に代表される**状況把握ブロック**．それらの情報により状況に応じた的確な車両速度、ステアリング角を計算する**コントロールブロック**．そして 2 つの情報を伝え実際の

車両に信号を伝えたり、そのシミュレーションを行う**車両ブロック**の 4 つの役割に分類される．車両ブロックから得られる現在の情報は、またそれぞれのブロックに状態フィードバックされるような構造になる．
実用上は、これら 4 つのグループのどれかを関数化し設計すれば良い．しかし、ここで区分けした状況把握ブロックと、コントロールブロックは、研究単位では分けて考えているが、これらは非常に密接な関係があると考えられ、これらのインターフェース仕様を固定してしまうとそれぞれの設計の自由度が低くなってしまう恐れがある．そこでここで提案する仕様では、これら状況把握ブロックとコントロールブロックについては、2 つで 1 つのブロック情報処理ブロックと呼ぶことする．よって、まとめると、便宜上、**ハードウェア関連**の**センサ関連ブロック**、**車両ブロック**を独立したブロックとし、それ以外の状況把握ブロック、コントロールブロックを 1 つにまとめにし**情報処理ブロック**とする．つまり、メインのプログラムでは、センサ関連ブロック→情報処理ブロック→車両ブロックで一巡する大きな関数ループにより構成されるようにする．

ブロック設計の基本的な考え方

標準的に宣言する変数

多人数のプロジェクトでプログラムを使う場合、グローバル変数を多用することは、バグを生むもとであり好ましいことではない．そこでここでは、グローバル変数の使用を完全に排除し関数（ブロック）間の変数のやり取りのみとする。ここで定義するのは、メインスクリプトファイルで使用する標準的な予約変数である。各ブロックは、関数 M ファイルとして宣言し、必要な予約変数は、その関数の引数として渡す。この方法は、データ構造を定義した MATLAB のツールボックスと同じである。
ここで使用する予約変数を以下に示す。
sampling:サンプリング周期[sec]
x,y:車両の絶対座標を表す[m]
theta:車両の絶対進行方向[m]
current_v:現在の車両速度[m/sec]
current_steering:現在のステアリング角[rad]
raw_ad:AD コンバータより得られた生信号
（これは、ad128pm 関数で DA を行うと AD も同時に行う仕様になっているため、車両ブロックで計測したデータをセンサ関連ブロックへ送るための措置.）
map: シミュレーションの時使用　マップデータ
以上これらの変数は、メインスクリプトファイル内で他の目的には使わないようにする。

センサ関連ブロック

役割　このブロックは、現在の状況を返す関数である．出力形式は、センサにより異なる．

入力:　センサをシミュレーションする時に必要な x, y, theta,current_v, current_steering, raw_ad などが入力変数の対象になる。ただし、個々のセンサにより必要な変数は異なると考えられるため、必要最低限な変数を入力とすればよい。例えば、vcapg は、入力は無しで出力変数は RGB 形式のカラー3 次元配列を出力するセンサブロックである。センサ関連ブロックは、ハードウェア関連の差異を吸収するブロックである。情報処理ブロック内でセンサ関連ブロックを呼ぶ可能性もあるためインターフェース間のオーバーヘッドを少なくするようにした方がよい。

情報処理ブロック

役割　センサ関連の情報を受け状況に応じたステアリング角、車両速度を求める.
入力：入力は、使用するセンサブロックにより異なってくる．また、情報処理内部でセンサを使用する例のサンプルは、後でその実例を挙げる。

車両ブロック

役割　車両のダイナミックスのシミュレーションとその状況の表示、実際の車両の駆動、現在の x,y 座標の計算等を行う．このブロックは、主に駆動関連のハードウェア扱う。
入力：入力は、速度 v,ステアリング角 steering、それに現在の座標 x,y 進行角度 theta、サンプリング時間 sample である。
出力：車両ブロックの情報は、情報処理、センサのシミュレーション等にも必要と思われる情報、x,y:車両の絶対座標を表す[m]、theta:車両の絶対進行方向[m]、current_v:現在の車両速度[m/sec]、current_steering:現在のステアリング角[rad]、raw_ad:AD コンバータより得られた情報である。

MATLAB プログラムの基本仕様とそのサンプルプログラム

メインスクリプトファイル

MATLAB メインスクリプトプログラムの構造を示す．while 文による無限ループ内にある時間のウェイトは、リアルタイム制御のためのものである．シミュレーションのみの場合には、コメントアウトすれば高速化できる．

```
while(1)
時間によるウェイト（実機の場合）
センサ関連ブロック
情報処理ブロック
車両ブロック
end
```

センサ関連ブロック及び､車両ブロックは、主にハードウェア関連の個所である．特にセンサ関連のブロックと情報処理ブロックが密接に関わって使用している場合には、次のように情報処理ブロックの引数として使用する．このようにすれば、センサ関連のブロックは、情報処理ブロック内部を変更することなくセンサを変更することができる．

```
while(1)
時間によるウェイト（実機の場合）
情報処理ブロック（センサ関連ブロック）
車両ブロック
end
```

時間によるウェイト(tictoc.m)

MATLAB で管理可能なサンプリング時間は、100 分の 1 秒単位でカウントを読むことができる．ただし、Windows 上で管理しているため正確に時間管理ができる訳ではないのでその点は留意して使用する必要がある．時間を読むには、tic,toc 関数を使用する．tic 関数は、カウンタを初期化スタートさせるのに用い、toc 関数でその経過時間を読む．

右に 0.1 秒間隔で実際の経過時間を 100 点分サンプリングしたのちグラフ化するプログラムを示す．実際のリアルタイムコントロールのプログラムでは、このプログラムの thistory(1,ii)=pretime;のところに関数を挿入すればよい．

```
% Timer sample program by Gerox(c) 1998
% Please wait 10 sec
sampling=0.1;%[sec]
%count start!
thistory=zeros(1,100);tic;
newtoc = toc;
pretime = newtoc;
for ii=1:100
  while(newtoc < pretime)
    newtoc = toc;
  end
  pretime = newtoc+sampling;
  thistory(1,ii)=pretime;
end
plot((1:99)*sampling,diff(thistory))
```

センサ関連ブロック(例えば、vcapg.dll)

現在ある、センサ関連の関数としては、ビデオキャプチャ関数 vcapg,AD コンバータ ad128pm(但し、この関数は、使用すると AD/DA が同時に実行される。通常は DA として車両ブロック用として使用するので AD されたデータは関数の引数とする。)などがある．他にも例えば、232C 経由でデータを転送する光ファイバージャイロなどが考えられる．センサ関連のブロックは、次の情報処理ブロック内で呼ばれる可能性があるのでなるべくオーバーヘッドがないように設計するとよい。

情報処理ブロック(sensing.m)

```
function
[info1,info2,info3]=sensing(line1,line2,line3,sensfunc);
a=feval(sensfunc);
info1=a(:,line1,:);
info2=a(:,line2,:);
info3=a(:,line3,:);
```

ここでの処理は、センサ関連ブロックから得られたデータを処理し、目標とする車両速度、ステアリング角度を計算するのが目的である．これらの中には、画像処理、カラー変換、エッジ処理、hough 変換などの画像処理テクニックも含まれる．この情報処理ブロックの目的は、カラーフィルタリング、hough 変換などの画像処理、他センサからの情報を総合し、例えば、ファジィロジック、ニューラルネット、PID、H∞などにより車両の状況に応じた判断が含まれる．右に示したサンプルプログラムは、情報処理ブロックを呼び出す際に変数としてセンサ関連ブロックを入れた例である．feval 関数により情報処理内でセンサ関連のブロックを呼び出している．（これが **eval 系関数の効果的な使い方の例**！！）このようにすることで、情報処理ブロック内部を変更することなく容易にセンサを変更することが可能になる．実は、同様のことが eval 関数を使用しても出来る。しかし MATLAB コンパイラでは、feval 関数をサポートしているが eval 関数はサポートしていない。そのためできるかぎりコンパイルして高速化できる可能性がある feval 関数を使用する。

```
》 [info1,info2,info3]=sensing(10,10,30,'vcapg');
》
```

また、feval 関数で複数の引数つきの関数

[out1,out2]=sensfunc(in1,in2);

を使う場合には、以下のようにする。

[out1,out2]=feval('sensfunc',in1,in2);

車両ブロック(vehisim.m)

車両ブロックは、実際の車両の動きのシミュレーションまたは、実際に車両を駆動させるブロックである．また、シミュレーションで必要な、x,y,theta などの情報は、実際の車両を駆動させるためには必要ないが、実走行した後の解析用等に有用であると考えられるのでインプリメントする．ここでは、まずシミュレーションの例を示す．シミュレーションでは、情報処理ブロックで得られた目標速度、目標角度よりまず、車両の応答のダイナミクスを考慮し実際の車両速度、ステアリング角を計算する．その速度、角度を基に車両の位置を更新すればよい．実際の x,y 座標,theta の具体的な計算式は、

$$current\_steering = transfer\_function(steering);$$
$$current\_v = transfer\_function(speed);$$
$$theta(k+1) = current\_steering \cdot sampling + theta(k)$$
$$x(k+1) = \cos(theta(k)) \cdot current\_v \cdot sampling + x(k)$$
$$y(k+1) = \sin(theta(k)) \cdot current\_v \cdot sampling + y(k)$$

となる．ここで定義されるダイナミクスを記述するための伝達関数を記述するためには、persistent 変数を使用すると簡潔に表現できる．

```
》 clear all
》 smpfil(1)
ans =
0.0100
》 smpfil(1)
ans =
0.0198
```

ここでは persistent 変数の効果的な使用例としてフィルタの例を示す．MATLAB の persistent 変数とは、関数内のみで使用できる変数（C でいうところのスタティク変数のこと）である．

```
function y=smpfil(u)
persistent preu
persistent prey
if(isempty(preu)) preu=u;end
if(isempty(prey)) prey=0;end
[num den]=c2dm([1],[1 1],0.01);
y = (-prey*den(2)+u*num(1)+preu*num(2))/den(1);
preu = u;
prey = y;
```

ここでは，差分方程式の必要な１回前の状態変数の保持に使用している．これら関数内の persistent 変数の内容を初期化するには，clear 関数名を使用する．

ＭＡＴＬＡＢでのコメントアウトのテクニック

MALTAB でのコメント文は、%であるが、この方法では、複数行のコメントアウトするのは、非常に面倒である。そこで複数行のコメントアウトするには、

```
if 0

end
```

リファレンスプログラム

このプログラムでは、

pvtest.m：メインスクリプトプログラム

vehisim.m：車両シミュレーションプログラム

sensim1.m:センサブロック vcapg を利用した情報処理ブロックの例（もし vcapg 互換のセンサシミュレーション関数を作った場合でも容易に変更可能にできる.）

sensim2.m：マップ情報を用いたシミュレーション用情報処理ブロック

（但し、車両シミュレーションにダイナミックスが入った場合にはうまく動作しない.）

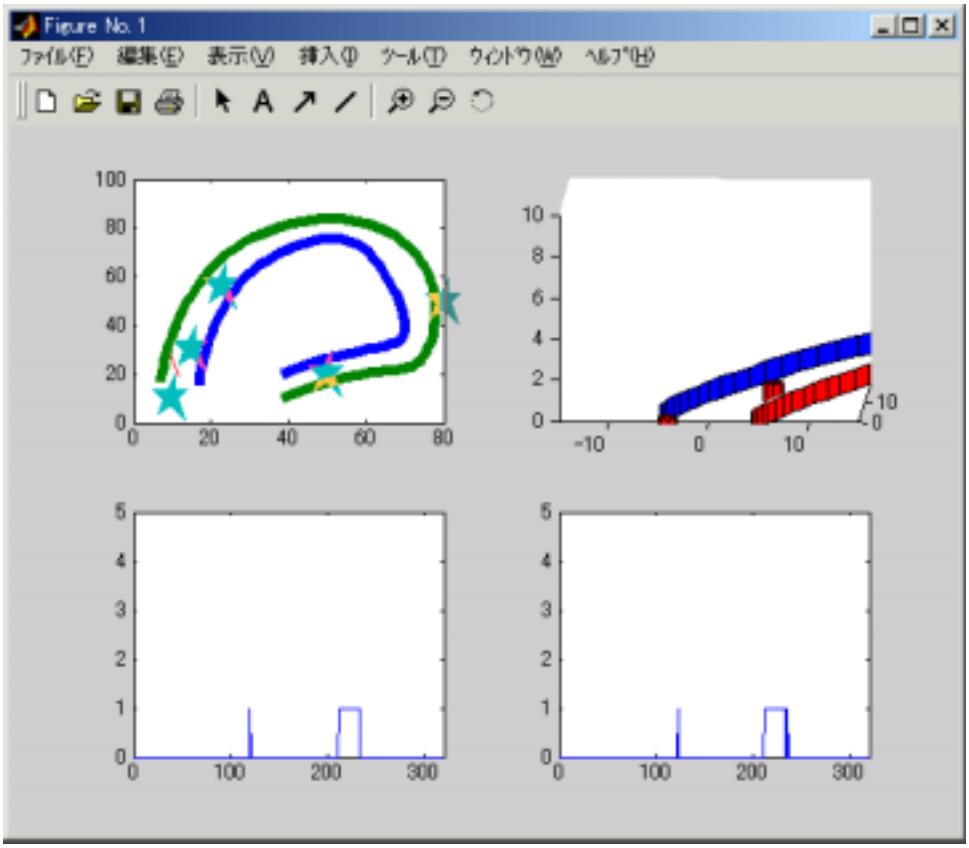
sensim3.m：マップ情報を基に比例制御で車両を動かす情報処理ブロック

makemapsim.m：マップ情報を作り出すプログラム.

の関数で構成される.

pvtest とタイプすると以下のような画面が表示され実行される。

左上の図が、全体のコースを上から見た図で星印が障害物を示している。赤の棒は、車両の進行方向を指している。右上の図は、３次元立体表示した図である。下段の図は、画像センサを想定したセンサから得られたある特定の距離の白線と障害物を判別し表示したものである。

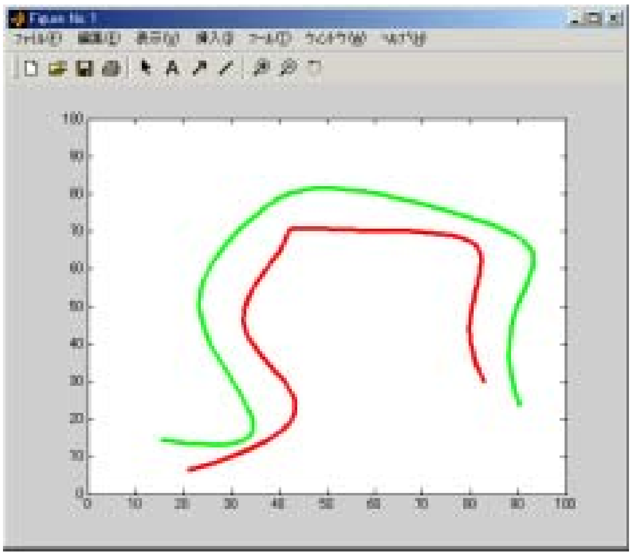


ちなみに、pvtest のコメントにも書いてあるが、

[map,x,y,theta]=makemapsim(1);

とすると独自のマップを作成することも出来るようになっている。

>> [map,x,y,theta]=makemapsim(1);




```
---------------------------------------pvtest.m---------------------------------------------------------------------------------------------------------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reference Sample Program for Autonomous Unmanned Vehicle
% SET INITIAL PARAMETERS
% Gerox (c) 1998 10 23
close all;% close all figures
clear functions;
sampling=0.1;%[sec]
raw_ad=[0 0 0 0 0 0 0 0];
x=0;y=0;theta=0;current_v=0;current_steering=0;v=0;steering=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CREATE MAPINFO
[map,x,y,theta]=makemapsim(x,y,theta); % If you want to make new map
                    % Please use   [map,x,y,theta]=makemapsim(1);
close all
if 1
% CHECK MAP
   theta=atan2(map(2,2)-map(1,2),map(2,1)-map(1,1));
   newvect=[sin(pi/2-theta) -cos(pi/2-theta);cos(pi/2-theta) sin(pi/2-theta)]*[0 10;0 0];
   figure(1);subplot(2,2,1);
   aa=plot(map(:,1),map(:,2),map(:,3),map(:,4),...
   newvect(1,:)+[x x],newvect(2,:)+[y y],'linewidth',5);
   set(aa(3),'linewidth',1);set(gca,'drawmode','fast');set(aa,'erasemode','xor');
   R=[sin(theta) -cos(theta);cos(theta) sin(theta)];
   rmrr = (R*([map(:,1)-x map(:,2)-y])');
   lmll = (R*([map(:,3)-x map(:,4)-y])');
   subplot(2,2,2);bb=plot(rmrr(1,:),rmrr(2,:),lmll(1,:),lmll(2,:));
   axis([-10 10 0 20]);
   set(gca,'drawmode','fast');set(aa,'erasemode','xor');
   for i=2:length(map)
      theta=atan2(map(i,2)-map(i-1,2),map(i,1)-map(i-1,1));
      newvect=[sin(pi/2-theta) -cos(pi/2-theta);cos(pi/2-theta) sin(pi/2-theta)]*[0 10;0 0];
      x=(map(i,1)-map(i,3))/2+map(i,3);y=(map(i,2)-map(i,4))/2+map(i,4);
      set(aa(3),'xdata',newvect(1,:)+[x x],'ydata',newvect(2,:)+[y y]);
      R=[sin(theta) -cos(theta);cos(theta) sin(theta)];
      rmrr = (R*([map(:,1)-x map(:,2)-y])');
      lmll = (R*([map(:,3)-x map(:,4)-y])');
      set(bb(1),'xdata',rmrr(1,:),'ydata',rmrr(2,:));
      set(bb(2),'xdata',lmll(1,:),'ydata',lmll(2,:));
      drawnow;
   end
end
if 1
%%%%set Global variable%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

生成したマップのチェックプログラムである.
これで 1 周走行した時の様子が把握できる.

```
theta=atan2((map(1+1,2)-map(1,2)),(map(1+1,1)-map(1,1)));
x=(map(1,1)-map(1,3))/2+map(1,3);
y=(map(1,2)-map(1,4))/2+map(1,4);
steering=theta;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic;
newtoc = toc;pretime = newtoc + sampling;
close all
newvect=[sin(pi/2-theta) -cos(pi/2-theta);cos(pi/2-theta) sin(pi/2-theta)]*[0 10;0 0];
aa=plot(map(:,1),map(:,2),map(:,3),map(:,4),...
newvect(1,:)+[x x],newvect(2,:)+[y y],'linewidth',5);
set(aa(3),'linewidth',1);set(gca,'drawmode','fast');set(aa,'erasemode','xor');
while(1)
   while(newtoc < pretime+sampling);newtoc = toc;end
   pretime = newtoc;
   [v,steering]=sensim1(0.7,70,'vcapg');% autodrive with vcapg
%   [v,steering]=sensim2(map,x,y,theta,sampling);% view map mode use no dynamics vehisim
%   [v,steering]=sensim3(map,x,y,theta); % autodrive mode;
   [x,y,theta,current_v,current_steering]=vehisim(v,steering,x,y,theta,sampling);

%%%%%%%%%%%%%%This is for plot%%%%%%%%%%%%%%%%%%%%%%%%%%
   newvect = [sin(pi/2-theta) -cos(pi/2-theta);cos(pi/2-theta) sin(pi/2-theta)]*[0 10;0 0];
   set(aa(3),'xdata',newvect(1,:)+[x x],'ydata',newvect(2,:)+[y y]);
   drawnow;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   end
end
```

ここがメインループ
基本的には、この関数
名を変更するだけ対応
できるようにする.

```
--------------------------------------vehisim m------------------------------------------------------------------------------------------------------
function [x,y,theta,current_v,current_steering]=vehisim(v,steering,x,y,theta,sampling)
persistent Avd Bvd Cvd Dvd;
persistent Asd Bsd Csd Dsd;
persistent vx sx;
if isempty(vx)
   % Calculate discrete transfer function parameters
   [Av,Bv,Cv,Dv] = tf2ss(1,[1 1]);[a,b]=size(Av);
   vx=zeros(a,1);
   [Avd,Bvd,Cvd,Dvd]=c2dm(Av,Bv,Cv,Dv,sampling,'tustin');
   [As,Bs,Cs,Ds] = tf2ss(1,[1 1]);[a,b]=size(As);
   [Asd,Bsd,Csd,Dsd]=c2dm(As,Bs,Cs,Ds,sampling,'tustin');
   sx=zeros(a,1);
   current_v = 0;
   current_steering = 0;
end
vx          = Avd * vx + Bvd * v;
current_v= Cvd * vx + Dvd * v;

sx                   = Asd * sx + Bsd * steering;
current_steering= Csd * sx + Dsd * steering;
% if 1 then sensim2
if 0
current_v = v;
current_steering = steering;
end
theta = theta + current_steering * sampling;
x = x + current_v * cos(theta) * sampling;
y = y + current_v * sin(theta) * sampling;
```

ステアリング、速度の伝達関数
を状態方程式にし変換した後、
離散値化している.

```
--------------------------------------sensim1.m------------------------------------------------------------------------------------------------------
function [v,steering]=sensim1(threshold,lineno,viewfunc)
a=feval(viewfunc,1);
b=(double(a(:,lineno,1))+double(a(:,lineno,2))+double(a(:,lineno,3)))/3;
figure(2)
plot(b);
steering=-mean(find(b>threshold)-length(b)/2);
v=3;
--------------------------------------sensim2.m------------------------------------------------------------------------------------------------------
function [v,steering]=sensim2(map,x,y,theta,sampling)

persistent count;
persistent px py ppx ppy ptheta pptheta;
if isempty(count)
   count = 1;
```

```
   pptheta=atan2((map(count+1,2)-map(count,2)),(map(count+1,1)-map(count,1)));
   ppx=(map(count,1)-map(count,3))/2+map(count,3);
   ppy=(map(count,2)-map(count,4))/2+map(count,4);
   x=ppx;
   y=ppy;
   theta=pptheta;
end
count = count + 1;
ptheta=atan2(map(count,2)-map(count-1,2),map(count,1)-map(count-1,1));
px=(map(count,1)-map(count,3))/2+map(count,3);
py=(map(count,2)-map(count,4))/2+map(count,4);
steering=(ptheta-pptheta)/sampling;
v=sqrt((px-ppx)^2+(py-ppy)^2)/sampling;
pptheta=ptheta;
ppx=px;
ppy=py;
   R=[sin(theta) -cos(theta);cos(theta) sin(theta)];
   rmrr = (R*([map(:,1)-x map(:,2)-y])');
   lmll = (R*([map(:,3)-x map(:,4)-y])');
figure(2)
   plot(rmrr(1,:),rmrr(2,:),lmll(1,:),lmll(2,:));
   axis([-10 10 0 20])
----------------------------------------sensim3.m-----------------------------------------------------------
function [v,steering]=sensim3(map,x,y,theta)
persistent hdl;
   viewrange=15;
   lookahead=9;
   eyeheight = 2;
   R=[sin(theta) -cos(theta);cos(theta) sin(theta)];
   rmrr = (R*([map(:,1)-x map(:,2)-y])');
   lmll = (R*([map(:,3)-x map(:,4)-y])');
%    x1 = rmrr(:,1)./(rmrr(:,2));
%    x2 = lmll(:,1)./(lmll(:,2));
%    y1 = -eyeheight./(rmrr(:,2));
%    y2 = -eyeheight./(lmll(:,2));
if isempty(hdl)
   figure(2)
   hdl=plot(rmrr(1,:),rmrr(2,:),lmll(1,:),lmll(2,:),'linewidth',5);
   axis([-viewrange viewrange 0 lookahead+10]);
   set(gca,'drawmode','fast');set(hdl,'erasemode','xor');
%    view([0,10]);
else
   set(hdl(1),'xdata',rmrr(1,:),'ydata',rmrr(2,:));
   set(hdl(2),'xdata',lmll(1,:),'ydata',lmll(2,:));
%    set(gca,'pos',[0 0 1 0.7],'color',[0 0.25 0],...
%    'xtick',[],'ytick',[],'box','off')
%    set(gcf,'color',[0 0 .5]);
end
%%%%%%%%%%%%%search x axis limit %%%%%%%%%%
   tmp=find(rmrr(1,:)<viewrange);
   rtmp=rmrr(:,tmp);
   tmp=find(rtmp(1,:)> -viewrange);
   rtmp=rtmp(:,tmp);
%%%%%%%%%%%%%search y axis limit%%%%%%%%%%
   tmp=find(rtmp(2,:)< viewrange);
   rtmp=rtmp(:,tmp);
   tmp=find(rtmp(2,:)> 0);
   redge =rtmp(:,tmp)';
%%%%%%%%%%%%%search x axis limit%%%%%%%%%%
   tmp=find(lmll(1,:)<viewrange);
   ltmp=lmll(:,tmp);
   tmp=find(ltmp(1,:)> -viewrange);
   ltmp=ltmp(:,tmp);
%%%%%%%%%%%%%search y axis limit%%%%%%%%%%
   tmp=find(ltmp(2,:)< viewrange);
   ltmp=ltmp(:,tmp);
   tmp=find(ltmp(2,:)> 0);
   ledge =ltmp(:,tmp)';
   [m n]=size(ledge);
   medge=[];
%%%%%%%%%%serch cross section between edge adn lookahead%%%%%%%%%%%%
```

このスクリプトでは、マップ情報と x,y,theta の情報から観測できるビューを計算しその情報を基にステアリング角を比例制御している.

```
   for ii=2:m
      if ((ledge(ii-1,2) > lookahead) & (ledge(ii,2) <= lookahead)) medge=[medge; ledge(ii-1,:) ;ledge(ii,:)];end
      if ((ledge(ii-1,2) <= lookahead) & (ledge(ii,2) > lookahead)) medge=[medge; ledge(ii-1,:);ledge(ii,:)];end
   end
   [m n]=size(redge);
   for ii=2:m
      if ((redge(ii-1,2) > lookahead) & (redge(ii,2) <= lookahead)) medge=[medge; redge(ii-1,:) ;redge(ii,:)];end
      if ((redge(ii-1,2) <= lookahead) & (redge(ii,2) > lookahead)) medge=[medge; redge(ii-1,:);redge(ii,:)];end
   end
   mmedge=[];
   [m n]=size(medge);
   for ii=1:2:m
      mmedge = [mmedge spline([medge(ii,2) medge(ii+1,2)],[medge(ii,1) medge(ii+1,1)],lookahead)];
   end
   lininfo=zeros(1,320);
   linmmedge = round(mmedge./viewrange*160+160);
   linmmedge = linmmedge.*(linmmedge>0).*(linmmedge<321);
   lininfo(1,linmmedge)=1;
v = 3;
   hoge=find(lininfo>0.5);
   if isempty(hoge),steering = 0;
   else
      steering=-mean(find(lininfo>0.5)-160)/160*pi/2;
   end
   if isnan(steering), steering=0;end
----------makemapsim.m----------
function [map,x,y,theta]=makemapsim(x,y,theta,options)
close all;figure(1);plot([0 0 100 100],[0 100 100 0]);
if nargin==1
        disp('終わりは、リターン');
        [mx,my]=ginput
else
        mx =[ 12.2120   14.9770   25.5760   44.0092   62.4424   72.8111   72.1198   62.2120   38.2488
22.5806];
        my =[ 15.7895   35.6725   63.1579   78.0702   75.1462   52.3392   31.2865   25.4386   14.9123
15.4971];
end
mr=[];ml=[];haba=5;
ka=(my(2)-my(1))¥(mx(2)-mx(1));
kb=my(1)+ka*mx(1);
sita = atan(1/ka)+pi/2;
sita2 = atan(1/ka)-pi/2;
for i=1:length(mx)-1
        ka=(my(i+1)-my(i))¥(mx(i+1)-mx(i));
        kb=my(i)+ka*mx(i);
        sita = atan2(my(i+1)-my(i),mx(i+1)-mx(i))+pi/2;
        mr=[mr ;[mx(i)+haba*cos(sita) my(i)+haba*sin(sita)]];
        sita2 = atan2(my(i+1)-my(i),mx(i+1)-mx(i))-pi/2;
        ml=[ml ;[mx(i)+haba*cos(sita2) my(i)+haba*sin(sita2)]];
end
mrr=[spline(1:length(mr(:,1)),mr(:,1),1:0.1:length(mr(:,1)));spline(1:length(mr(:,2)),mr(:,2),1:0.1:length(mr(:,2)))]';
mll=[spline(1:length(ml(:,1)),ml(:,1),1:0.1:length(ml(:,1)));spline(1:length(ml(:,2)),ml(:,2),1:0.1:length(ml(:,2)))]';
close all;
plot(mrr(:,1),mrr(:,2),'g',mll(:,1),mll(:,2),'r','linewidth',3);
axis([0 100 0 100]);
% for global initial condition
x=(mll(1,1)+mrr(1,1))/2;
y=(mll(1,2)+mrr(1,2))/2;
theta=atan2((my(2)-my(1)),(my(2)-my(1)));
map=[mll mrr];
----------makemapsim.m----------
```

グローバル変数の利用について(gltest.m,glfunc.m)

一般的にプログラミングする場合、オブジェクト指向の考えからグローバル変数を使用することは、プログラムのモジュール化、可読性を低くすることになるため好ましいことではない．MATLAB では、Version 5.2 になり C 言語の関数内で宣言する static 変数に相当する persistent 変数が導入されたためほとんどの場合、この変数を用いれば対処できる．

```
% global variable sample program by Gerox(c) 1998
clear functions; % pretty important!
global sampling raw_ad;
global x y theta current_v current_steering
sampling = 0.01;
x=0;y=0;theta=0;current_v=1;current_steering=1;
% initial condition
[x y theta current_v current_steering]
glfunc; % 1 step after
[x y theta current_v current_steering]
glfunc; % next step
[x y theta current_v current_steering]
```

しかしながらそれでもどうしてもグローバル変数を使用したい場合も存在する．ここでは、まず、一般的なグローバル変数の使用法について説明した後、グローバル変数の使用が効果的な例を挙げて説明する．

グローバル変数とは？

スクリプト M ファイルの変数名は、スクリプトファイル内、MATLAB ワークスペース内で有効であるが、関数 M ファイル内部で宣言された変数は呼び出したスクリプトファイルワークスペース上ともに読むことはできない．スクリプト内変数、ワークスペース内変数を関数 M ファイル内でも有効にするのがグローバル変数である．

宣言する際、

global x,y

とカンマで変数を区切ると区切られた後の変数は、グローバル変数にならないので注意すること．正しくは、

global x　y

とカンマで区切るのではなくスペースを入れ区切る必要がある．また、関数スクリプトファイル内ともグローバル変数を使用する前にグローバル変数宣言する必要がある．

```
function glfunc
global x y theta current_v current_steering sampling;
theta = theta+current_steering*sampling;
x=x+cos(theta);
y=y+sin(theta);
```

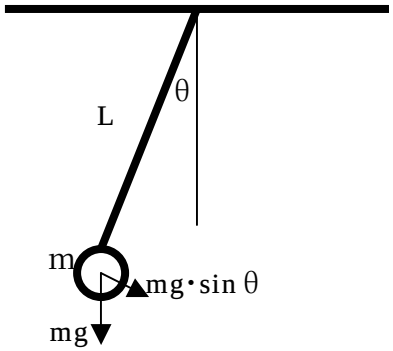
グローバル変数の効果的な使用法

通常の MATLAB プログラミングでの関数は、C 言語の関数とは異なり、複数の引数、戻り

値を取ることができるため、グローバル変数を使うより、関数の引数、戻り値を使用したほうが、プログラムのモジュール性，可読性の意味からも都合が良い．しかし、例えば、関数を引数に取るような関数の場合には、一般的にその関数の引数の数、戻り値の数が決められており、それを変更することはできない．このような場合にグローバル変数は非常に有効に活用できる．例えば、関数を引数にとるような関数
fmin,fmins,fzero,ode23,ode45,quad,quad8 などの関数がそれにあたる．ここでは、具体的例として ode45 を用い、まず ode45 の使用法、次になぜグローバル変数と良いのかを挙げて説明する．ode45 とは、ルンゲクッタ法により与えられた関数（微分方程式）の解を求める関数である．この関数の引数は、[T,Y,TE,YE,IE] = ODE45('F',TSPAN,Y0,OPTIONS)となっている．微分方程式は、線形、非線形問わず常に次のような一階の微分方程式で近似できる．

$$\frac{dx}{dt} = f(t,x)$$

そこでこの ODE45 では、この f を関数の引数'F'として取ることが出来るようになっている．要するに、ユーザは、引数 2 の関数 M ファイルり適正なパラメータを与えれば、その応答がシミュレーションできるようになっている．
例えば、振子を例に非線形連立微分方程式をシミュレーションしてみよう．
右図に単振動系を考える．
θが大きく変化する場合には、
非線形微分方程式でモデル化したほうが良い．
この場合角θに関する運動方程式は、



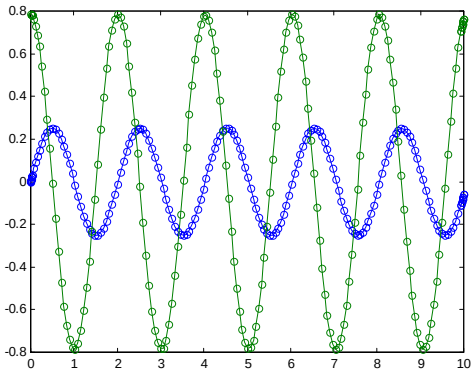
$$mL\frac{d^2\theta}{dt^2} = -mg\sin\theta$$

である．この方程式を、先ほどの
一階の微分方程式に変換する．今各周波数ωを導入すると,次のように表すことが出来る．

$$\frac{d}{dt}\begin{bmatrix}\theta\\ \omega\end{bmatrix} = \begin{bmatrix}\omega\\ -\frac{g}{L}\sin\theta\end{bmatrix}$$，ただし $\theta(0)=\theta_0, \omega(0)=\omega_0$ である．この非線形微分方程式を MATLAB 関数ファイル pend.m で表現すると、以下のような入力が時間 t と状態変数 x、出力が dx/dt となる関数 M ファイルとなる．
この関数では、g=9.8m/s^2,L=1m と設定した．
この非線形関数を ode45 でシミュレーションするには、次のようにすればよい．

```
function dxdt = pend(t,x)
g=9.8;L=1;
theta=x(1);omega=x(2);
dxdt = [omega;-g/L*sin(theta)];
```

》 clear functions;ode45('pend',[0 10],[0;45*pi/180])



このようにシミュレーションできるわけであるが、例えば、もし、さまざま長さでシミュレーションしてみたいと考えてみよう．要するに、単振子の長さ L を 10 センチ置きに 2m まで変えたいとしたらどうしたら良いであろうか？一般的には、関数の汎用性を高めるため関数の引数を増やし、長さ L も関数の引数にしてしまえば良いのであるが、ode45 関数を使いシミュレーションする場合には、引数が時間と状態変数のみで設計されているためうまくいかない．このような場合に有効に使えるのがグローバル変数である．それでは、長さ L を可変できるようにスクリプト M ファイルと関数 M ファイルを変更してみよう．

```
function dxdt = pend2(t,x)
global L;
g=9.8;
theta=x(1);omega=x(2);
dxdt = [omega;-g/L*sin(theta)];
```

グローバル変数を使えるようにするには、スクリプト M ファイル内で、global 変数と宣言し、かつ、関数 M ファイル内でも global 変数宣言をする必要がある．

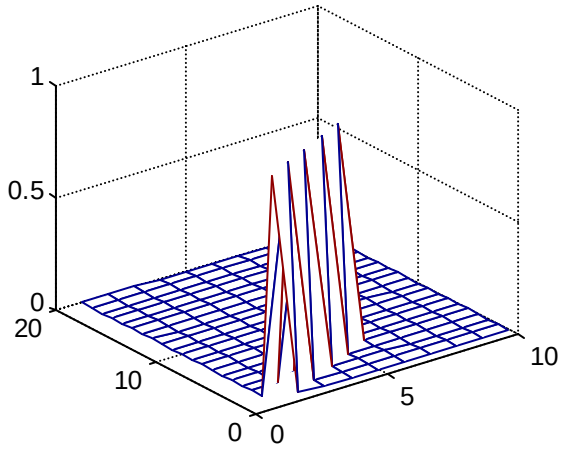
```
》 clear all;global L;for L=1:0.1:2;figure;ode45('pend2',[0 10],[0;45*pi/180]);end
```

この例では、変数を全てクリアした後 L をグローバル変数で宣言し、for 文により L を 1 から 10cm 間隔で 2m まで変更し ode45 でシミュレーションしている例である．

このように、グローバル変数を活用すれば，パラメータを関数の外で変更することが出来るので便利な場合もある．

2 次元ベクトルの扱い方の違いによる相違

MATLAB では，ベクトルでデータを扱うと高速にかつ簡潔に表現できる．ここでは，しかし，2 次元ベクトルの扱いと 1 次元ベクトルの扱いは，若干異なるため注意が必要である．ここでは，for 文による例とベクトル化し表示した例の違いを見てみる．

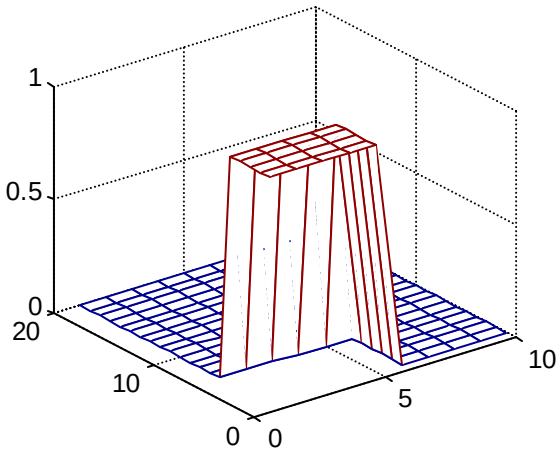



```
ab=zeros(20,10);
for i=1:5;
ab(i,i)=1;
end
figure(1);mesh(ab);
```

このスクリプトが for 文で書いたプログラムである．この場合，ab(1,1),ab(2,2),...ab(5,5)までの配列に 1 を書き込んでいる．

このプログラムを高速化しようとしてベクトル化した例を以下に示す．

```
ab=zeros(20,10);
i=1:5;
ab(i,i)=1;
figure(2);mesh(ab);
```



この例では，for 文をベクトル化し直接ベクトルを入れた例である．
1 次元配列の場合には，ベクトル化しても問題なく同様に動作するが，2 次元配列の場合には，注意が必要である。