MATLAB による 2 リンクロボットマニピュレータ制御のシミュレーション

● 2 リンクロボットマニピュレータモデル

ロボットダイナミクスは次式のとおりである.

$$\boldsymbol{\tau} = M(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + V_m(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) + F(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) + G(\boldsymbol{\theta})$$

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}, \quad \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

$$M(\boldsymbol{\theta}) = \begin{bmatrix} (m_1+m_2)l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 \cos\theta_2 & m_2 l_2^2 + m_2 l_1 l_2 \cos\theta_2 \\ m_2 l_2^2 + m_2 l_1 l_2 \cos\theta_2 & m_2 l_2^2 \end{bmatrix}$$

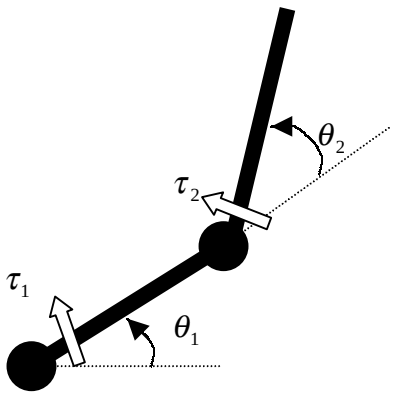
$$V_m(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) = \begin{bmatrix} -m_2 l_1 l_2 (2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_1^2)\sin\theta_2 \\ m_2 l_1 l_2 \dot{\theta}_1^2 \sin\theta_1 \end{bmatrix}$$

$$G(\boldsymbol{\theta}) = \begin{bmatrix} (m_1+m_2)gl_1\cos\theta_1 + m_2 g\cos(\theta_1+\theta_2)l_2 \\ m_2 g l_2 \cos(\theta_1+\theta_2) \end{bmatrix}$$

$$F(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) = \begin{bmatrix} b_1\dot{\theta}_1 \\ b_2\dot{\theta}_2 \end{bmatrix}$$

ロボットの各パラメータを次のように設定する.

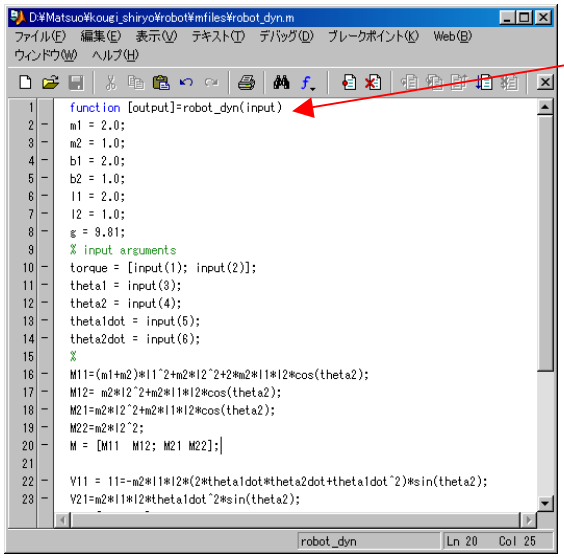
$$g = 9.81$$
$$l_1 = 2.0,\ l_2 = 1.0$$
$$m_1 = 2.0,\ m_2 = 1.0$$
$$b_1 = 1.0,\ b_2 = 2.0$$



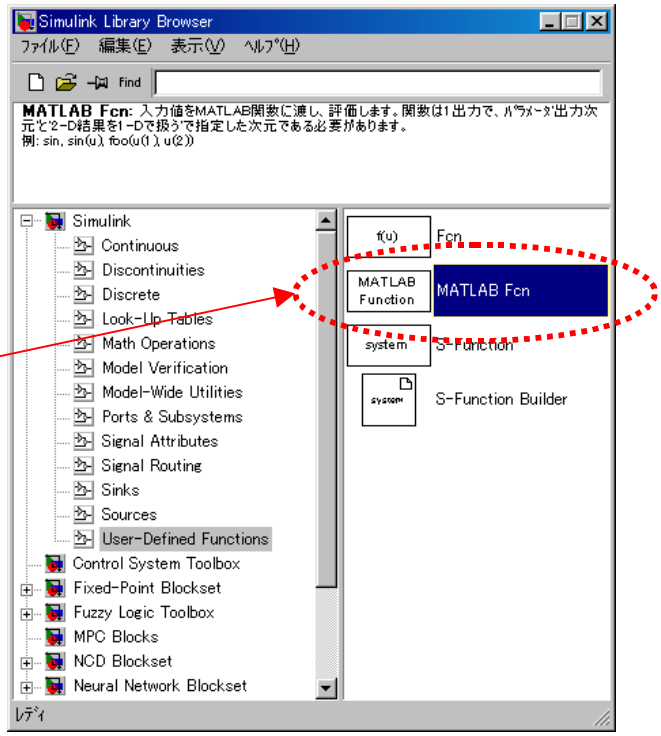
このモデルを MATLAB function M ファイルで実現しよう.

関数名：**robot_dyn.m**，入力引数 **input →** $\boldsymbol{\tau, \theta, \dot{\theta}}$ ：，出力引数 **output →** $\boldsymbol{\dot{\theta}, \ddot{\theta}}$

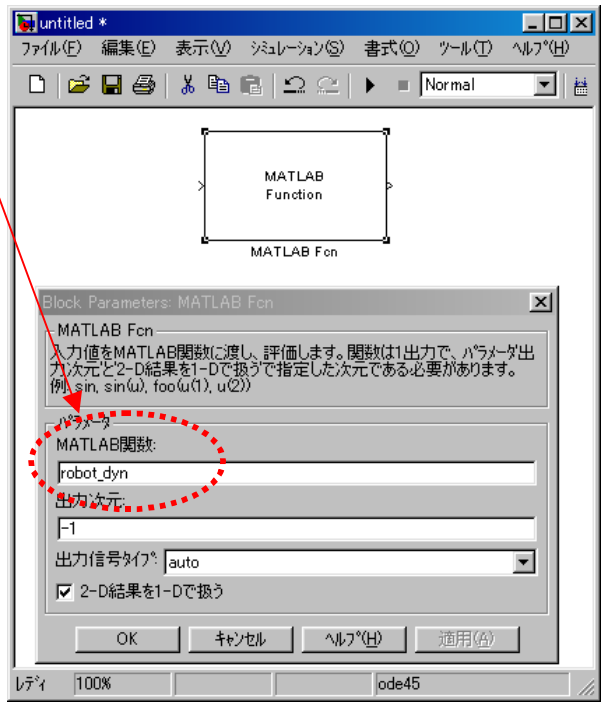
```
function [output]=robot_dyn(input)
m1 = 2.0;
m2 = 1.0;
b1 = 2.0;
b2 = 1.0;
l1 = 2.0;
l2 = 1.0;
g = 9.81;
% input arguments
torque = [input(1); input(2)];
theta1 = input(3);
theta2 = input(4);
theta1dot = input(5);
theta2dot = input(6);
%
M11=(m1+m2)*l1^2+m2*l2^2+2*m2*l1*l2*cos(theta2);
M12= m2*l2^2+m2*l1*l2*cos(theta2);
M21=m2*l2^2+m2*l1*l2*cos(theta2);
M22=m2*l2^2;
M = [M11    M12; M21 M22];

V11=-m2*l1*l2*(2*theta1dot*theta2dot+theta1dot^2)*sin(theta2);
V21=m2*l1*l2*theta1dot^2*sin(theta2);
V = [V11;V21];

G11=(m1+m2)*g*l1*cos(theta1)+m2*g*cos(theta1+theta2)*l2;
G21=m2*g*l2*cos(theta1+theta2);
G=[G11;G21];

F = [b1*theta1dot;b2*theta2dot];

thetaddot=inv(M)*(torque-F-V-G);
output=[input(5) input(6) thetaddot(1) thetaddot(2)];
```



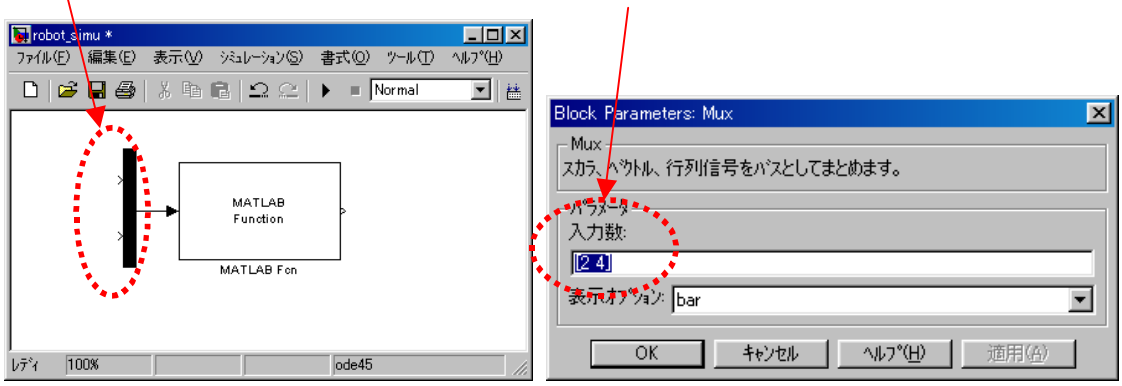
保存ファイル名は，関数名と同じ，
robot_dyn.m にすること．



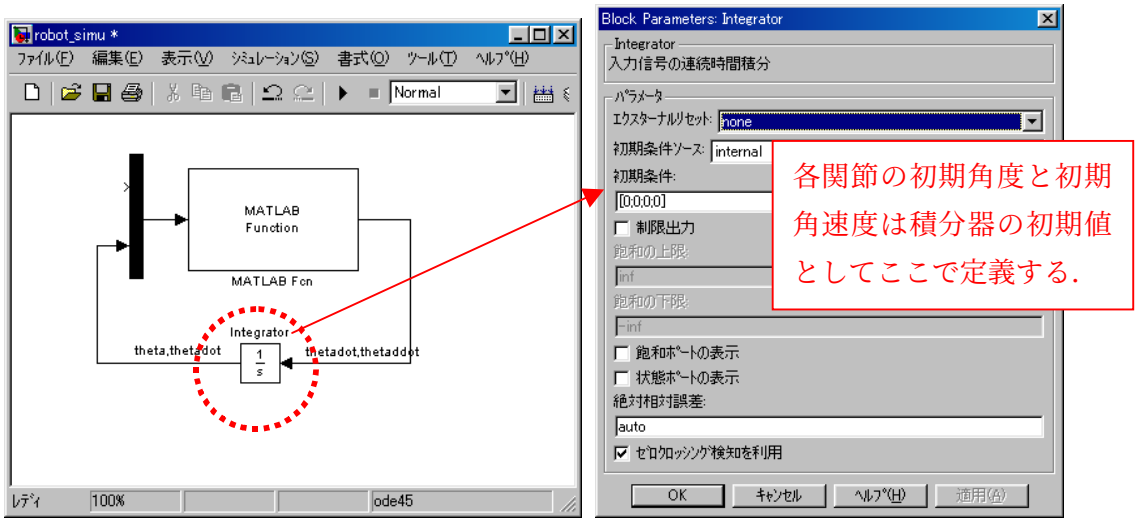
さらに，これを Simulink における MATLAB Fuction ブロックで定義する．ブロックをダブルクリックして，関数名に **robot_dyn** を書き込む．



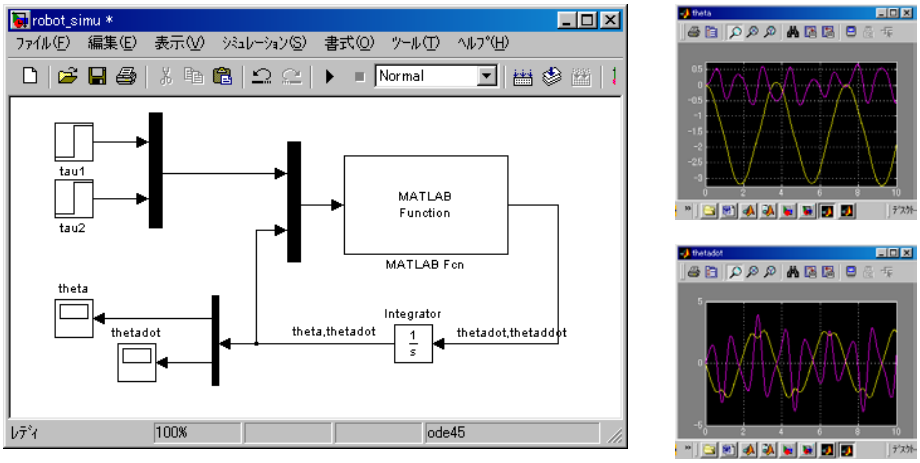
Mux を用いて，入力ベクトルを $\boldsymbol{\tau}$ と $\boldsymbol{\theta},\dot{\boldsymbol{\theta}}$ に分割する．Mux をダブルクリックして表示されるダイアログボックスの入力数に，入力ベクトルの分割したい次元を指定する．



さらに入力引数のうち $\boldsymbol{\theta},\dot{\boldsymbol{\theta}}$ と出力引数 $\dot{\boldsymbol{\theta}},\ddot{\boldsymbol{\theta}}$ と積分器で結合することにより，ロボットダイナミクスを支配する微分方程式を実現することができる．



応答をシミュレーションするために，各軸のトルク入力としてステップ関数を入れて，グラフ表示用に Scope を入れて，シミュレーションを行うとつぎのようになる．



● 逆運動学による希望軌道の計算方法

手先が中心$(1,1)$, 半径 0.5 の円軌道を描くような関節角の目標起動を導出してみよう．ただし，そのような目標軌道を生成するトルクをつくるためには，関節角，角速度，角加速度の目標軌道が必要になる．

手先の目標軌道の直交座標表現は次式のようになる．

$$(x-1.5)^2+(y-1.5)^2=0.25$$
$$x(t)=0.5\cos(\omega t)+1.5$$
$$y(t)=0.5\sin(\omega t)+1.5$$

直交座標での手先の速度と加速度は次式のようになる．

$$\dot{x}(t)=-0.5\omega\sin(\omega t),\ \dot{y}(t)=0.5\omega\cos(\omega t)$$
$$\ddot{x}(t)=-0.5\omega^2\cos(\omega t),\ \ddot{y}(t)=-0.5\omega^2\sin(\omega t)$$

手先の直交座標と関節角の座標変換式は次式で与えられる．

$$x=l_1\cos\theta_1+l_2\cos(\theta_1+\theta_2)$$
$$y=l_1\sin\theta_1+l_2\sin(\theta_1+\theta_2)$$

これから，位置と角度の逆運動学は次式で与えられる．

$$\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}=\begin{bmatrix}\arctan(y,x)+\arctan(k,x^2+y^2+l_1^2-l_2^2)\\ -\arctan(k,x^2+y^2-l_1^2-l_2^2)\end{bmatrix}$$

ただし，

$$k=\sqrt{(x^2+y^2+l_1^2+l_2^2)^2-2\left((x^2+y^2)^2+l_1^4+l_2^4\right)}$$

で， $\arctan(y,x)$ は x,y の 4 象限逆正接値であり， $-\pi\le\arctan(y,x)\le\pi$ である．

速度と角速度は Jacobi 行列で関係付けられる．

$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix}=J(\boldsymbol{\theta})\begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\end{bmatrix}$$

ただし， $J(\boldsymbol{\theta})$ は Jacobi 行列で，次式で与えられる．

$$J(\boldsymbol{\theta})=\begin{bmatrix}-l_1\sin\theta_1-l_2\sin(\theta_1+\theta_2) & -l_2\sin(\theta_1+\theta_2)\\ l_1\cos\theta_1+l_2\cos(\theta_1+\theta_2) & l_2\cos(\theta_1+\theta_2)\end{bmatrix}$$

これから速度の逆運動学は次式で得られる．

$$\begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\end{bmatrix}=J^{-1}(\boldsymbol{\theta})\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix}$$

さらに，速度と角速度の関係式を時間微分すると，加速度と角加速度の関係が次式のように得られる．

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\end{bmatrix}=J\begin{bmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\end{bmatrix}+\begin{bmatrix}-l_1\dot{\theta}_1^2\cos\theta_1-l_2\left(\dot{\theta}_1+\dot{\theta}_2\right)^2\cos(\theta_1+\theta_2)\\ -l_1\dot{\theta}_1^2\sin\theta_1-l_2\left(\dot{\theta}_1+\dot{\theta}_2\right)^2\sin(\theta_1+\theta_2)\end{bmatrix}$$

これから加速度の逆運動学は次式で得られる．

$$\begin{bmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\end{bmatrix}=J^{-1}\left\{\begin{bmatrix}\ddot{x}\\ \ddot{y}\end{bmatrix}-\begin{bmatrix}-l_1\dot{\theta}_1^2\cos\theta_1-l_2\left(\dot{\theta}_1+\dot{\theta}_2\right)^2\cos(\theta_1+\theta_2)\\ -l_1\dot{\theta}_1^2\sin\theta_1-l_2\left(\dot{\theta}_1+\dot{\theta}_2\right)^2\sin(\theta_1+\theta_2)\end{bmatrix}\right\}$$

これを MATLAB function として実現する．

・ 順運動学を計算する function M-file **: xytrajectory.m**

入力引数：時間変数t（クロック）

出力引数：直交座標位置，速度，加速度$x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y}$

```
function [output]=xytrajectory(input)
t1=input;
omega = 0.5;
phi=omega*t1;
x=0.5*cos(phi)+1.5;
y=0.5*sin(phi)+1.5;
xdot=-0.5*sin(phi);
ydot=0.5*cos(phi);
xddot=-0.5*cos(phi);
yddot=-0.5*sin(phi);
output=[x;y;xdot;ydot;xddot;yddot];
```

・ 位置の逆運動学を計算する function M-file **angle.m**

入力引数：直交座標位置x, y

出力引数：関節角θ_1, θ_2

```
function [output]=angle(input)
x=input(1);
y=input(2);
l1=2.0;
l2=1.0;
a=(x^2+y^2+l1^2+l2^2)^2;
b=2*((x^2+y^2)^2+l1^4+l2^4);
k=sqrt(a-b);
theta1=atan2(y,x)+atan2(k,x^2+y^2+l1^2-l2^2);
theta2=-atan2(k,x^2+y^2-l1^2-l2^2);
output=[theta1;theta2];
```

・ 速度の逆運動学を計算する function M-file **angledot.m**

入力引数：関節角θ_1,θ_2，直交座標速度$\dot{x},\dot{y}$

出力引数：関節角速度$\dot{\theta}_1,\dot{\theta}_2$

```
function [output]=angledot(input)
theta1=input(1);
theta2=input(2);
xdot=input(3);
ydot=input(4);
xydot=[xdot;ydot];
l1=2.0;
l2=1.0;
A=[-l1*sin(theta1)-l2*sin(theta1+theta2)   -l2*sin(theta1+theta2)
     l1*cos(theta1)+l2*cos(theta1+theta2)   l2*cos(theta1+theta2)];
if det(A)<=0.001
  thetadot=[0;0];
else
    thetadot=inv(A)*xydot;
end;
output=thetadot;
```

・ 加速度の逆運動学を計算する function M-file **angleddot.m**

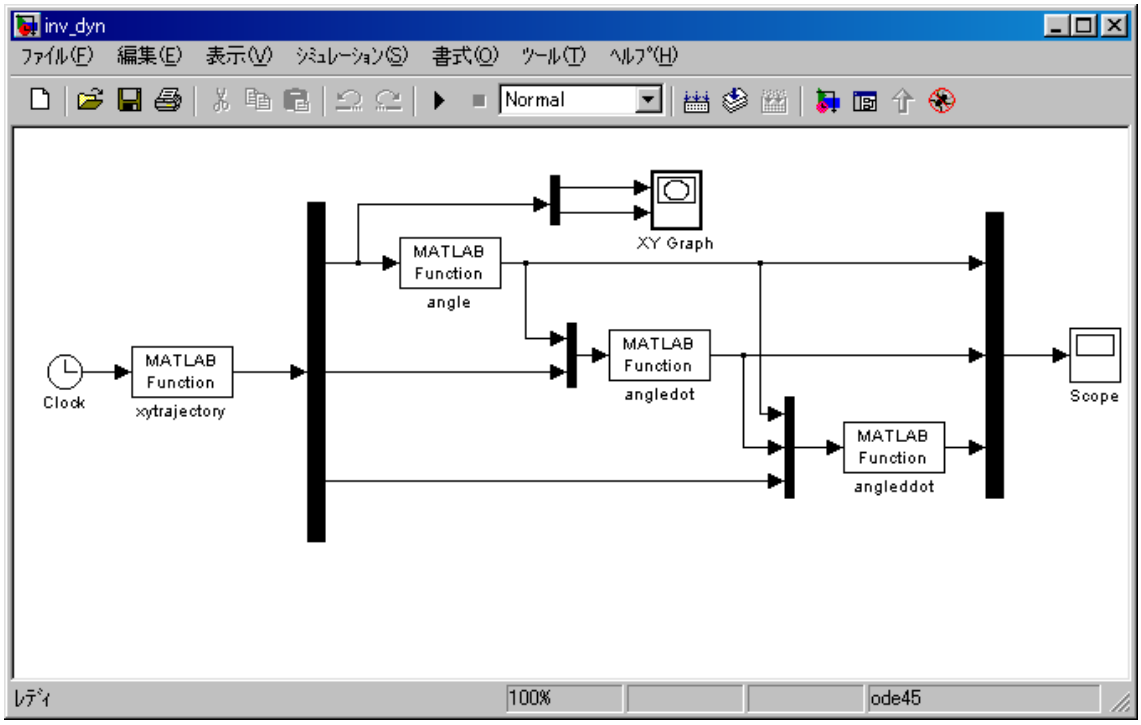
入力引数：関節角θ_1,θ_2，関節角速度$\dot{\theta}_1,\dot{\theta}_2$，直交座標角速度$\ddot{x},\ddot{y}$

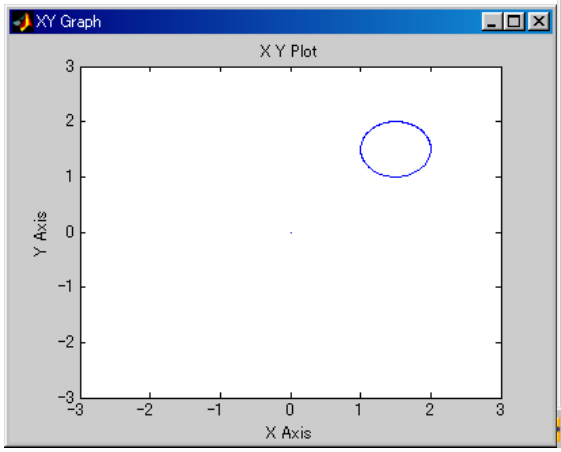
出力引数：関節角加速度$\ddot{\theta}_1,\ddot{\theta}_2$

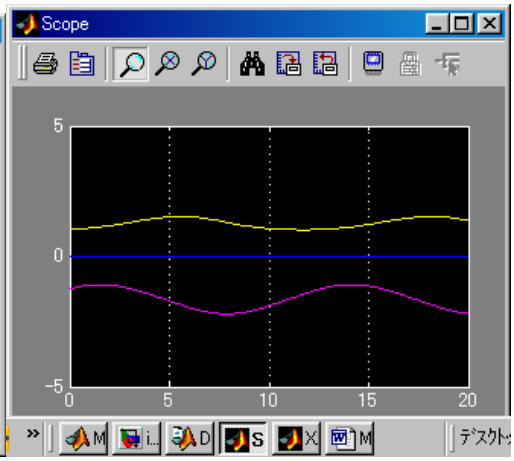
```
function [output]=angleddot(input)
theta1=input(1);
theta2=input(2);
theta1dot=input(3);
theta2dot=input(4);
xddot=input(5);
yddot=input(6);

xyddot=[xddot;yddot];
l1=2.0;
l2=1.0;
A=[-l1*sin(theta1)-l2*sin(theta1+theta2)   -l2*sin(theta1+theta2)
     l1*cos(theta1)+l2*cos(theta1+theta2)   l2*cos(theta1+theta2)];
B11=-l1*cos(theta1)*theta1dot^2-l2*cos(theta1+theta2)*(theta1dot+theta2dot)^2;
```

```
B21=-l1*sin(theta1)*theta1dot^2-l2*sin(theta1+theta2)*(theta1dot+theta2dot)^2;
B=[B11;B21];
% to avoid matrix Asingularity
if det(A)<=0.001
    thetaddot=[0.001;0.001];
else
    thetaddot=inv(A)*(xyddot-B);
end;
output=thetaddot;
```

これらの関数を Simulink の MATLAB Function に登録し，それを結合することにより，位置，速度，加速度の逆運動学がつぎのように構成できる．







● **Computed Torque Control**

前述の逆運動学から生成される関節の目標軌道を定める目標角度，目標角速度，目標角加速度を，各々$\boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}_d, \ddot{\boldsymbol{\theta}}_d$とおく．実際の関節角と目標角との誤差を次式のように定義する．

$$\mathbf{e} = \boldsymbol{\theta}_d - \boldsymbol{\theta}$$

速度と加速度の誤差は次式のようになる．

$$\dot{\mathbf{e}} = \dot{\boldsymbol{\theta}}_d - \dot{\boldsymbol{\theta}}$$
$$\ddot{\mathbf{e}} = \ddot{\boldsymbol{\theta}}_d - \ddot{\boldsymbol{\theta}}$$

ロボットマニピュレータのダイナミクスは次式である．

$$\boldsymbol{\tau} = M(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + V_m(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + F(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + G(\boldsymbol{\theta})$$

トルクが入力であるので，これを次式の形で構成するのが，Computed Torgue Control である．これは，希望の目標値になるトルクを逆算してそれを入力に用いることからきている．

$$\boldsymbol{\tau} = M(\boldsymbol{\theta})u + V_m(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + F(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + G(\boldsymbol{\theta})$$
$$u = \ddot{\boldsymbol{\theta}}_d + K_v\dot{\mathbf{e}} + K_p\mathbf{e}$$

このような入力を実現するためには，関節角，角速度はすべて測定可能であり，まさつなどの要素やシステムパラメータはすべて既知でなければならない．閉ループ系の誤差方程式は上式の 2 つからトルクを消去することにより，次式のように得られる．

$$M(\boldsymbol{\theta})\left(\ddot{\mathbf{e}} + K_v\dot{\mathbf{e}} + K_p\mathbf{e}\right) = 0$$

ここで，$M(\boldsymbol{\theta})$は正則であるので，誤差方程式は次式のようになる．

$$\ddot{\mathbf{e}} + K_v\dot{\mathbf{e}} + K_p\mathbf{e} = 0$$

これは，線形微分方程式になり，K_v, K_pを適切に選ぶことにより，漸近安定，つまり次式を満足するようにできる．

$$\lim_{t\to\infty}\mathbf{e} = 0$$

たとえば，

$$K_v = \begin{bmatrix} k_{v1} & 0 \\ 0 & k_{v2} \end{bmatrix}, \quad K_p = \begin{bmatrix} k_{p1} & 0 \\ 0 & k_{p2} \end{bmatrix}, \quad k_{v1}, k_{v2}, k_{p1}, k_{p2} > 0$$

と選ぶと，つぎのように独立の 2 本の安定な 2 階微分方程式になる．

$$\ddot{e}_1 + k_{v1}\dot{e}_1 + k_{p1}e_1 = 0$$
$$\ddot{e}_2 + k_{v2}\dot{e}_2 + k_{p2}e_2 = 0$$

目標角，角速度，角加速度を実現するトルクは次式の MATLAB function M ファイルで計算できる．

関数名 **comp_torque.m**

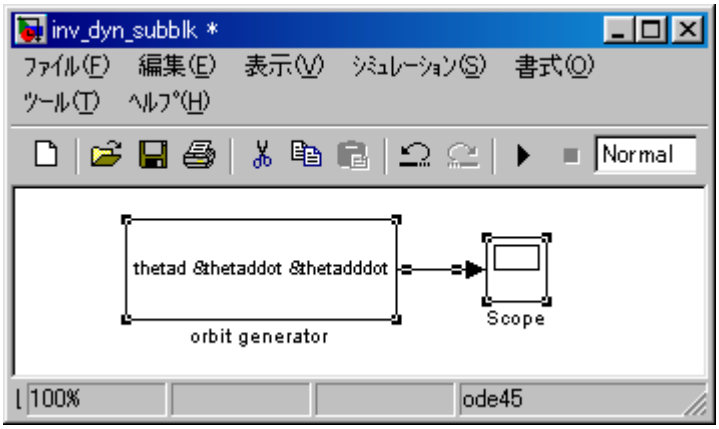
入力引数 **input →** $\theta, \dot{\theta}, \theta_d, \dot{\theta}_d, \ddot{\theta}_d$ ：，出力引数 **output →** τ

```
function [output]=computed_torque(input)
theta1=input(1);
theta2=input(2);
theta1dot=input(3);
theta2dot=input(4);
thetad1=input(5);
thetad2=input(6);
thetad1dot=input(7);
thetad2dot=input(8);
thetad1ddot=input(9);
thetad2ddot=input(10);
Kv=[2 0;0 2];
Kp=[6 0;0 6];
m1 = 2.0;
m2 = 1.0;
b1 = 2.0;
b2 = 1.0;
l1 = 2.0;
l2 = 1.0;
g = 9.81;

M11=(m1+m2)*l1^2+m2*l2^2+2*m2*l1*l2*cos(theta2);
M12= m2*l2^2+m2*l1*l2*cos(theta2);
M21=m2*l2^2+m2*l1*l2*cos(theta2);
M22=m2*l2^2;
M = [M11    M12; M21 M22];

V11=-m2*l1*l2*(2*theta1dot*theta2dot+theta1dot^2)*sin(theta2);
```

```
V21=m2*l1*l2*theta1dot^2*sin(theta2);
V = [V11;V21];

G11=(m1+m2)*g*l1*cos(theta1)+m2*g*cos(theta1+theta2)*l2;
G21=m2*g*l2*cos(theta1+theta2);
G=[G11;G21];

F = [b1*theta1dot;b2*theta2dot];

theta = [theta1; theta2];
thetadot = [theta1dot; theta2dot];
thetad = [thetad1; thetad2];
thetaddot = [thetad1dot; thetad2dot];
thetadddot = [thetad1ddot; thetad2ddot];
taucon = M*(thetadddot+Kv*(thetaddot-thetadot)+Kp*(thetad-theta))+V+G+F;
output=[taucon(1) taucon(2)];
```

このコントローラを使った閉ループ系の応答シミュレーションを Simulink を用いて行う．

１）逆運動学のよる軌道生成 Simulink をサブシステム化



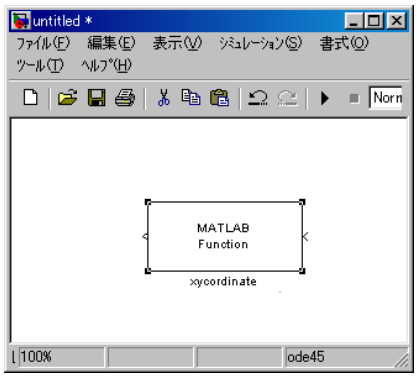
２）順運動学を計算する M ファイルをつくり，MATLAB Function で Simulink ブロック化する．

関数名 **xycord.m**

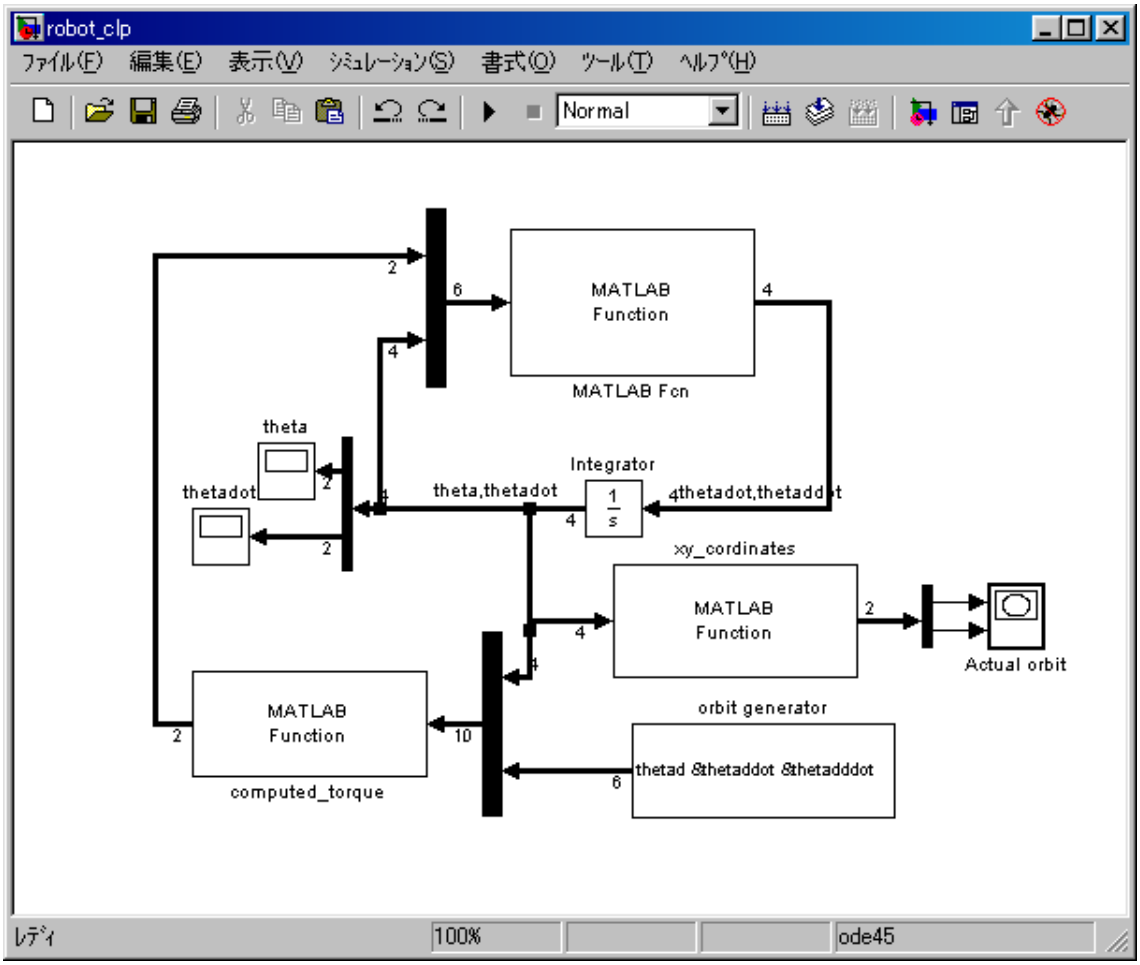
入力引数 **input** → θ ：，出力引数 **output** → x, y

```
function [output]=xycord(input)
theta1=input(1);
theta2=input(2);
l1=2.0;
```

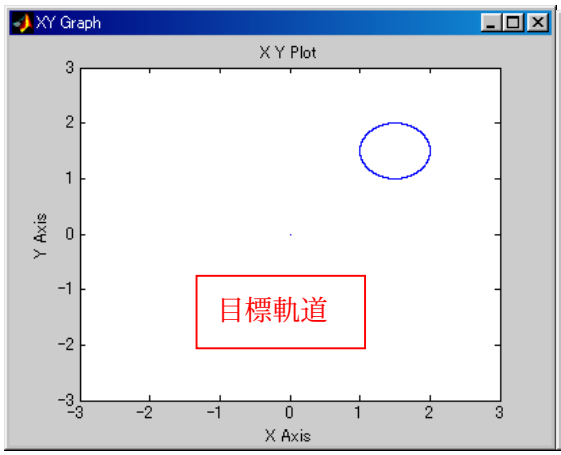
```
l2=1.0;
x=l1*cos(theta1)+l2*cos(theta1+theta2);
y=l1*sin(theta1)+l2*sin(theta1+theta2);
output=[x;y];
```

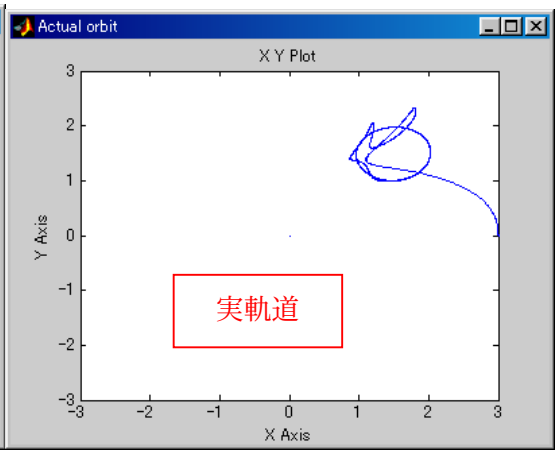


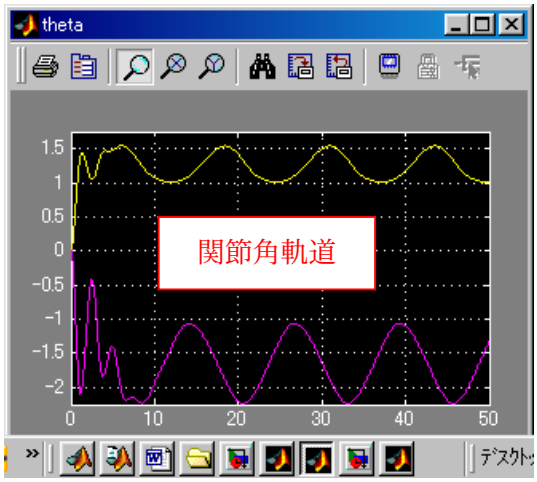
3）rob_dyn，computed_torque と逆運動学サブシステムとの結合



４）応答波形の確認







● **Computed Torque Control のロバスト性**

リンクの質量や粘性摩擦係数が正確に測れていない状況として，Computed Torque Control を行った場合のロバスト性をシミュレーションにより確認する．ここでは，コントローラを構成する際に用いるパラメータをノミナル値とよび，実際の値よりも 25%ずれているとする．

	m1	m2	b1	b2
実際の値	2.0	1.0	2.0	1.0
ノミナル値	2.5	1.25	2.5	1.25

このときの M ファイルはつぎのとおりである．

関数名 **ct_perturb.m**

入力引数 **input →** $\theta, \dot{\theta}, \theta_d, \dot{\theta}_d, \ddot{\theta}_d$ ：，出力引数 **output →** τ

```
function [output]=ct_perturb(input)
theta1=input(1);
```

```
theta2=input(2);
theta1dot=input(3);
theta2dot=input(4);
thetad1=input(5);
thetad2=input(6);
thetad1dot=input(7);
thetad2dot=input(8);
thetad1ddot=input(9);
thetad2ddot=input(10);
Kv=[2 0;0 2];
Kp=[6 0;0 6];
m1 = 2.5;
m2 = 1.25;
b1 = 2.5;
b2 = 1.25;
l1 = 2.0;
l2 = 1.0;
g = 9.81;

M11=(m1+m2)*l1^2+m2*l2^2+2*m2*l1*l2*cos(theta2);
M12= m2*l2^2+m2*l1*l2*cos(theta2);
M21=m2*l2^2+m2*l1*l2*cos(theta2);
M22=m2*l2^2;
M = [M11    M12; M21 M22];

V11=-m2*l1*l2*(2*theta1dot*theta2dot+theta1dot^2)*sin(theta2);
V21=m2*l1*l2*theta1dot^2*sin(theta2);
V = [V11;V21];

G11=(m1+m2)*g*l1*cos(theta1)+m2*g*cos(theta1+theta2)*l2;
G21=m2*g*l2*cos(theta1+theta2);
G=[G11;G21];

F = [b1*theta1dot;b2*theta2dot];

theta = [theta1; theta2];
```

thetadot = [theta1dot; theta2dot];

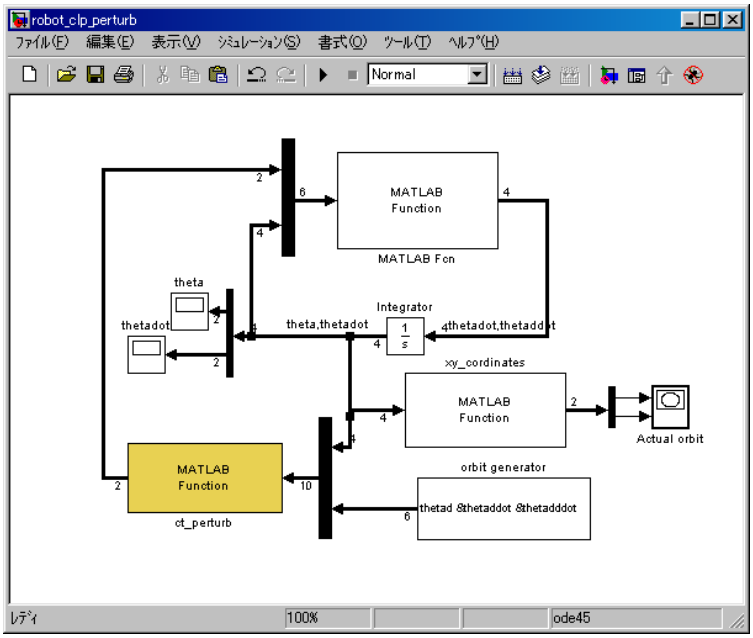
thetad = [thetad1; thetad2];

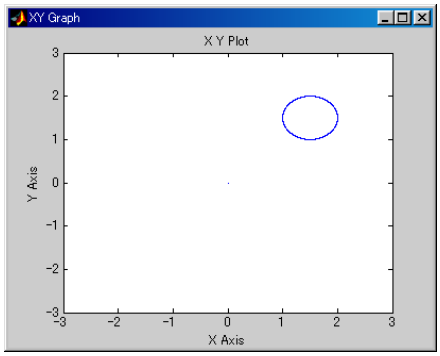
thetaddot = [thetad1dot; thetad2dot];

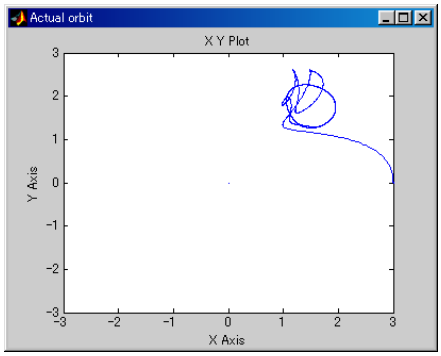
thetadddot = [thetad1ddot; thetad2ddot];

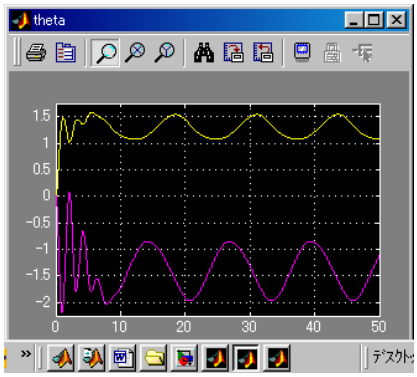
taucon = M*(thetadddot+Kv*(thetaddot-thetadot)+Kp*(thetad-theta))+V+G+F;

output=[taucon(1) taucon(2)];









● **PD Plus Gravity Controller**

重力補償項のついたＰＤコントローラによりトルクを生成して，軌道制御することを考える．この場合のコントローラは次式で与えられる．

$$\boldsymbol{\tau} = K_v\dot{\mathbf{e}} + K_p\mathbf{e} + G(\boldsymbol{\theta})$$

このとき，リアプノフ関数を用いると，誤差の有界性（発散しないこと）を証明することができる．

関数名 **PDGtorque.m**

入力引数 **input →** $\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}_d, \ddot{\boldsymbol{\theta}}_d$ ：，出力引数 **output →** $\boldsymbol{\tau}$

```
function [output]=PDGtorque(input)
theta1=input(1);
theta2=input(2);
theta1dot=input(3);
theta2dot=input(4);
thetad1=input(5);
thetad2=input(6);
thetad1dot=input(7);
thetad2dot=input(8);
thetad1ddot=input(9);
thetad2ddot=input(10);
Kv=[2 0;0 2];
Kp=[6 0;0 6];
m1 = 2.0;
m2 = 1.0;
b1 = 2.0;
b2 = 1.0;
l1 = 2.0;
l2 = 1.0;
g = 9.81;

G11=(m1+m2)*g*l1*cos(theta1)+m2*g*cos(theta1+theta2)*l2;
G21=m2*g*l2*cos(theta1+theta2);
G=[G11;G21];

theta = [theta1; theta2];
```

thetadot = [theta1dot; theta2dot];

thetad = [thetad1; thetad2];

thetaddot = [thetad1dot; thetad2dot];

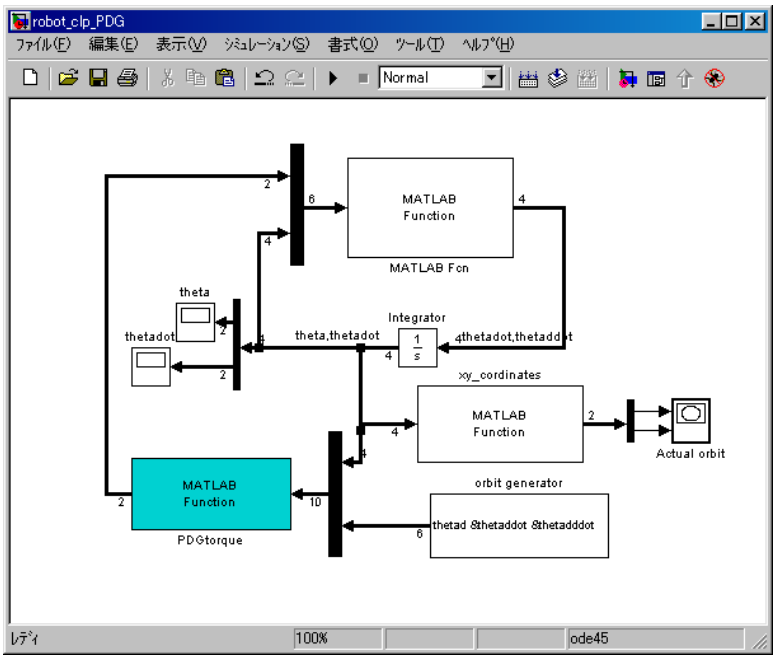
thetadddot = [thetad1ddot; thetad2ddot];

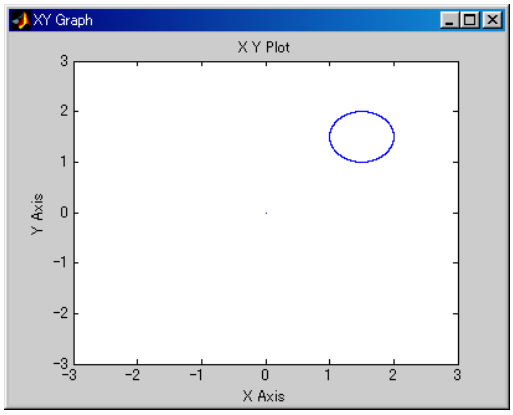
e = thetad - theta;

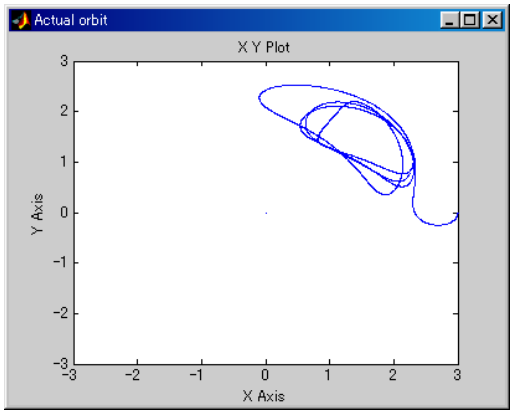
edot = thetaddot - thetadot;

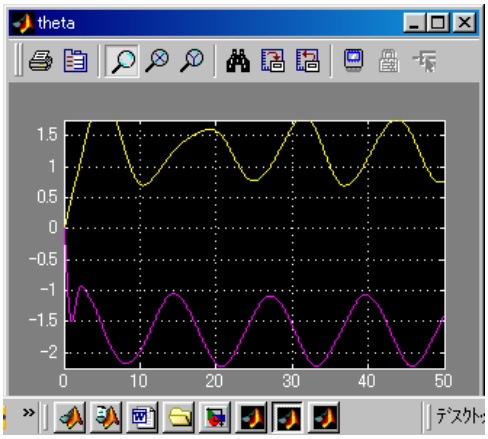
taucon = Kv*edot+Kp*e+G;

output=[taucon(1) taucon(2)];









● **Adaptive Computed Torque Control**

リンク質量と粘性摩擦係数を未知パラメータとして，つぎのようにおく．

$$\boldsymbol{\Phi} = \begin{bmatrix} m_1 \\ m_2 \\ b_1 \\ b_2 \end{bmatrix}$$

ロボットダイナミクスをこのパラメータとそれ以外の積で表すことができ，次式のように書ける．

$$\boldsymbol{\tau} = M(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + V_m(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) + F(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) + G(\boldsymbol{\theta}) = W(\boldsymbol{\theta},\dot{\boldsymbol{\theta}},\ddot{\boldsymbol{\theta}})\boldsymbol{\Phi}$$

未知パラメータの推定値を次式のようにおく．

$$\hat{\boldsymbol{\Phi}} = \begin{bmatrix} \hat{m}_1 \\ \hat{m}_2 \\ \hat{b}_1 \\ \hat{b}_2 \end{bmatrix}$$

上式のロボットダイナミクスを未知パラメータを推定値で置換えたものを次式のようにおく．

$$\hat{\boldsymbol{\tau}} = \hat{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \hat{V}_m(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) + \hat{F}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) + \hat{G}(\boldsymbol{\theta}) = W(\boldsymbol{\theta},\dot{\boldsymbol{\theta}},\ddot{\boldsymbol{\theta}})\hat{\boldsymbol{\Phi}}$$

適応コントローラは次式で与えられる．

$$\begin{aligned} \boldsymbol{\tau} &= \hat{M}(\boldsymbol{\theta})\left(\ddot{\mathbf{e}} + K_v\dot{\mathbf{e}} + K_p\mathbf{e}\right) + \hat{\boldsymbol{\tau}} \\ &= \hat{M}(\boldsymbol{\theta})\left(\ddot{\boldsymbol{\theta}}_d + K_v\dot{\mathbf{e}} + K_p\mathbf{e}\right) + \hat{V}_m(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) + \hat{F}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) + \hat{G}(\boldsymbol{\theta}) \end{aligned}$$

このとき，閉ループ系は次式のようになる．

$$\ddot{\mathbf{e}} + K_v\dot{\mathbf{e}} + K_p\mathbf{e} = \hat{M}^{-1}(\boldsymbol{\theta})W(\boldsymbol{\theta},\dot{\boldsymbol{\theta}},\ddot{\boldsymbol{\theta}})\tilde{\boldsymbol{\Phi}}$$

ただし，未知パラメータの推定誤差を次式で定義している．

$$\tilde{\boldsymbol{\Phi}} = \boldsymbol{\Phi} - \hat{\boldsymbol{\Phi}}$$

これを状態方程式に直すと，次式のようになる．

$$\dot{\boldsymbol{\varepsilon}} = A\boldsymbol{\varepsilon} + B(\hat{M}(\boldsymbol{\theta})W(\boldsymbol{\theta},\dot{\boldsymbol{\theta}},\ddot{\boldsymbol{\theta}})\tilde{\boldsymbol{\Phi}})$$

$$\boldsymbol{\varepsilon} = \begin{bmatrix}\mathbf{e}\\ \dot{\mathbf{e}}\end{bmatrix},\ A = \begin{bmatrix}0 & I\\ -K_p & -K_v\end{bmatrix},\ B = \begin{bmatrix}0\\ I\end{bmatrix}$$

これは，適応制御における標準の誤差方程式である．適応制御では，パラメータの推定値を時変で更新していくが，ここでは次式のようなパラメータ調整則を与える．

$$\dot{\hat{\boldsymbol{\Phi}}} = \Gamma W^T(\boldsymbol{\theta},\dot{\boldsymbol{\theta}},\ddot{\boldsymbol{\theta}})\hat{M}^{-1}(\boldsymbol{\theta})B^T P\boldsymbol{\varepsilon}$$

ただし， A は安定行列で， P は次式のリアプノフ不等式をみたす正定解である．

$$A^T P + PA < 0$$

Γ は次式の正定対称行列を選定する．

$$\Gamma = \begin{bmatrix}r_1 & 0 & 0 & 0\\ 0 & r_2 & 0 & 0\\ 0 & 0 & r_3 & 0\\ 0 & 0 & 0 & r_4\end{bmatrix},\quad r_1, r_2, r_3, r_4 > 0$$

パラメータ調整則を実現する M ファイルはつぎのとおりである．

関数名 **updatelaw.m**

入力引数 **input →** $\boldsymbol{\theta},\dot{\boldsymbol{\theta}},\boldsymbol{\theta}_d,\dot{\boldsymbol{\theta}}_d,\ddot{\boldsymbol{\theta}}_d,\boldsymbol{\Phi}$ ：，出力引数 **output →** $\dot{\hat{\boldsymbol{\Phi}}}$

```
function [output]=updatelaw(input)
theta1=input(1);
theta2=input(2);
theta1dot=input(3);
theta2dot=input(4);

theta1dotm=input(5);
theta2dotm=input(6);
theta1ddot=input(7);
theta2ddot=input(8);
```

```
thetad1=input(9);
thetad2=input(10);
thetad1dot=input(11);
thetad2dot=input(12);
thetad1ddot=input(13);
thetad2ddot=input(14);

m1 = input(15);
m2 = input(16);
b1 = input(17);
b2 = input(18);

l1 = 2.0;
l2 = 1.0;
g = 9.81;

gamma = [1 0 0 0
         0 1 0 0
         0 0 1 0
         0 0 0 1];
B=[ 0 0
    0 0
    1 0
    0 1 ];

kp = eye(2);
kv = 2*eye(2);
P = 0.5*[ kp*0.5*kv     0.5*eye(2)
          0.5*eye(2)    eye(2) ];

M11=(m1+m2)*l1^2+m2*l2^2+2*m2*l1*l2*cos(theta2);
M12= m2*l2^2+m2*l1*l2*cos(theta2);
M21=m2*l2^2+m2*l1*l2*cos(theta2);
M22=m2*l2^2;
M = [M11    M12; M21 M22];
```

```
W11 = l1^2*theta1ddot+g*l1*cos(theta1)
W12 = (l1^2+l2^2+2*l1*l2*cos(theta2))*theta1ddot ...
      -(l2^2+l1+l2*cos(theta2))*theta2ddot-l1*l2*(2*theta1dot*theta2dot...
      +theta1dot^2)*sin(theta2)+g*l1*cos(theta1)+g*l2*cos(theta1+theta2)
W13 = theta1dot
W14 = 0;
W21 = 0;
W22 = (l2^2+l1*l2*cos(theta2))*theta1ddot+l2^2*theta2ddot ...
      +l1*l2*theta1dot^2*sin(theta2)+g*l2*cos(theta1+theta2)
W23 = 0;
W24 = theta2dot
W = [W11 W12 W13 W14
     W21 W22 W23 W24]
ep = [thetad1-theta1
      thetad2-theta2
      thetad1dot-theta1dot
      thetad2dot-theta2dot]

phidot = gamma*W'*inv(M)*B'*P*ep
output=[phidot(1) phidot(2) phidot(3) phidot(4)]
```

トルクを計算する M ファイルはつぎのとおりである.
関数名 **adaptive_torque.m**

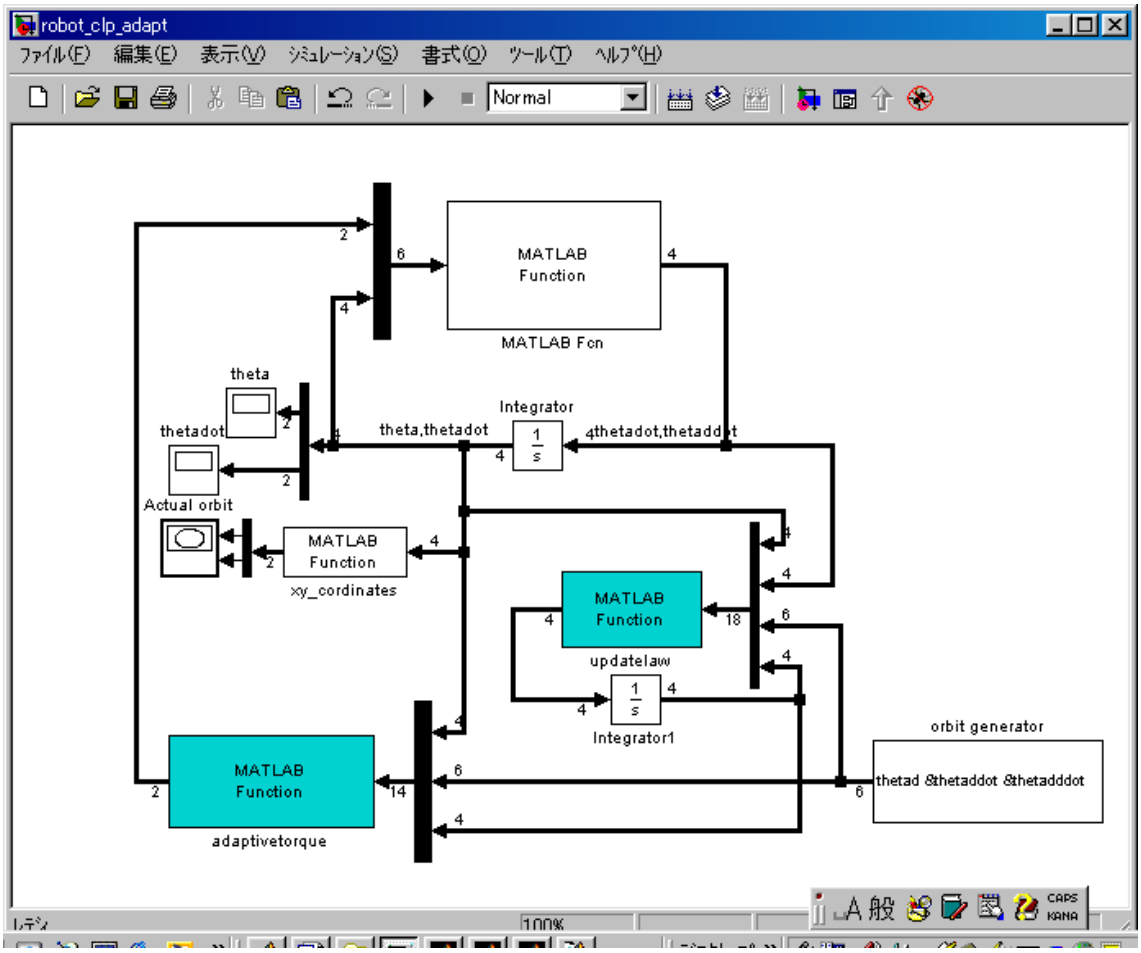
入力引数 **input →** $\theta, \dot{\theta}, \theta_d, \dot{\theta}_d, \ddot{\theta}_d, \Phi$ ：, 出力引数 **output →** τ

```
function [output]=adaptive_torque(input)
theta1=input(1);
theta2=input(2);
theta1dot=input(3);
theta2dot=input(4);

thetad1=input(5);
thetad2=input(6);
thetad1dot=input(7);
thetad2dot=input(8);
thetad1ddot=input(9);
```

```
thetad2ddot=input(10);

m1 = input(11);
m2 = input(12);
b1 = input(13);
b2 = input(14);

l1 = 2.0;
l2 = 1.0;
g = 9.81;

kp=eye(2);
kv=2*eye(2);

M11=(m1+m2)*l1^2+m2*l2^2+2*m2*l1*l2*cos(theta2);
M12= m2*l2^2+m2*l1*l2*cos(theta2);
M21=m2*l2^2+m2*l1*l2*cos(theta2);
M22=m2*l2^2;
M = [M11    M12; M21 M22];

V11=-m2*l1*l2*(2*theta1dot*theta2dot+theta1dot^2)*sin(theta2);
V21=m2*l1*l2*theta1dot^2*sin(theta2);
V = [V11;V21];

G11=(m1+m2)*g*l1*cos(theta1)+m2*g*cos(theta1+theta2)*l2;
G21=m2*g*l2*cos(theta1+theta2);
G=[G11;G21];

F = [b1*theta1dot;b2*theta2dot];

theta = [theta1; theta2];
thetadot = [theta1dot; theta2dot];
thetad = [thetad1; thetad2];
thetaddot = [thetad1dot; thetad2dot];
thetadddot = [thetad1ddot; thetad2ddot];
e = thetad-theta;
```

edot = thetaddot-thetadot;

taucon = M*(thetadddot+kv*edot+kp*e)+V+G+F;
output=[taucon(1) taucon(2)];



参考文献

He, Chengli：Motion Control, Midterm Project.
Slotine,W.Li.：Applied Nonlinear Control