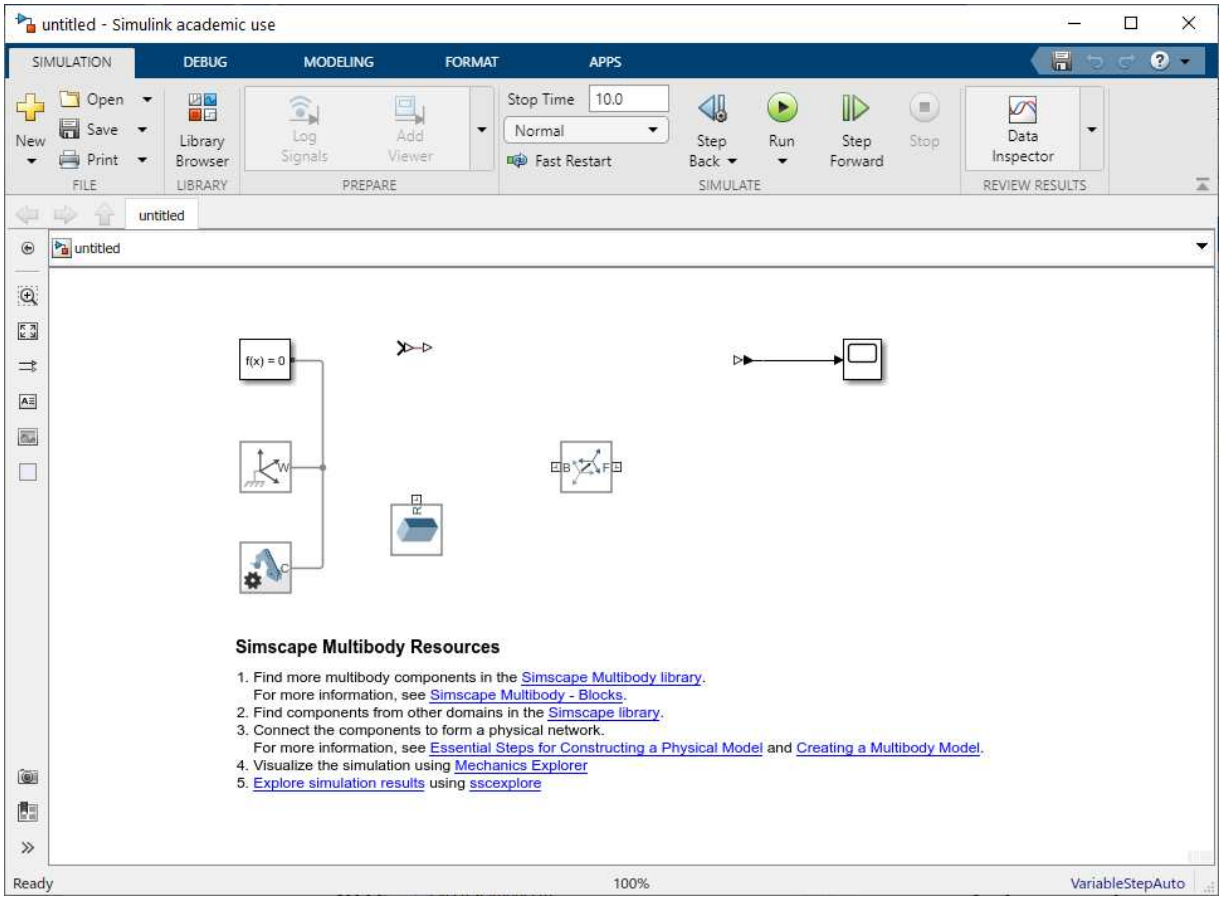
SimMechanics がバージョンアップし、Simscape＋Simscape Multibody の構成に変更になったため、ロボットモデリングのサンプルファイルが実行できなくなってしまいました。最新版の構成で動くように修正しました。
実行は、MATLAB2020a で確認しています。

7.8 Simscape Multibody によるモデル作成とその検証

Simscape は、Simulink 上で動作する Add-on のツールボックス群のひとつであり、Simulink 上でマルチドメインの物理現象をシミュレーションできる。Simscape Multibody は、マルチドメインの扱いに、空間的な座標関係を扱えるようにしたもので、剛体の物理的な原理の確認が可能になる。ここでは、vectest9demo.m と同じモデルを作成、シミュレーションを行い、加速度、速度などの比較を行う。このように複数のやり方で同じシミュレーションをすることで、勘違いや間違いを軽減し、特に Simscape Multibody の使い方を深めることができる。
Simscape Mutibody では、smnew コマンドを使うと、必要な環境のセットアップができる。

```
>>smnew
```

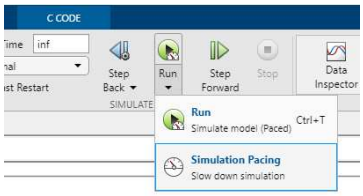


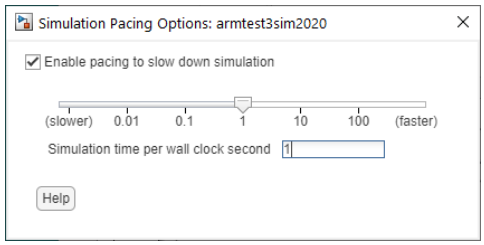
左の３つブロックは、Simscape　Multibody でシミュレーションをするために、必要なブロックである。F(x)=0 は、微分方程式を解くための Solver　Configuration Block, ワールド座標系の原点を定める World Frame Block、重力の方向などを設定する Mechanism　Configuration Block である。

Simscape Multibody を用いた 2 軸ロボットアームの実装

　Simscape を使った、スライダーで可変可能な 2 軸ロボットアームの例を示す。

図のブロック図に示すように、左側の３つのブロックが、基点となり、グランドからロボットアームを作成する。MATLAB では、併進と回転を同次変換行列として、扱ってきたが、この例では、あえて別々の Rigid Transform Block により実現している。Revolute Joint Block の設定を変更することで、角度入力を可能とした。なお、デフォルトの設定では、スライダーを動かしてもロボットアームは、インタラクティブに動かすことができない。（MATLAB で for 文の中に drawnow を入れないとインタラクティブアニメーションできないイメージ）スライダーゲインによる調整を有効にするには、



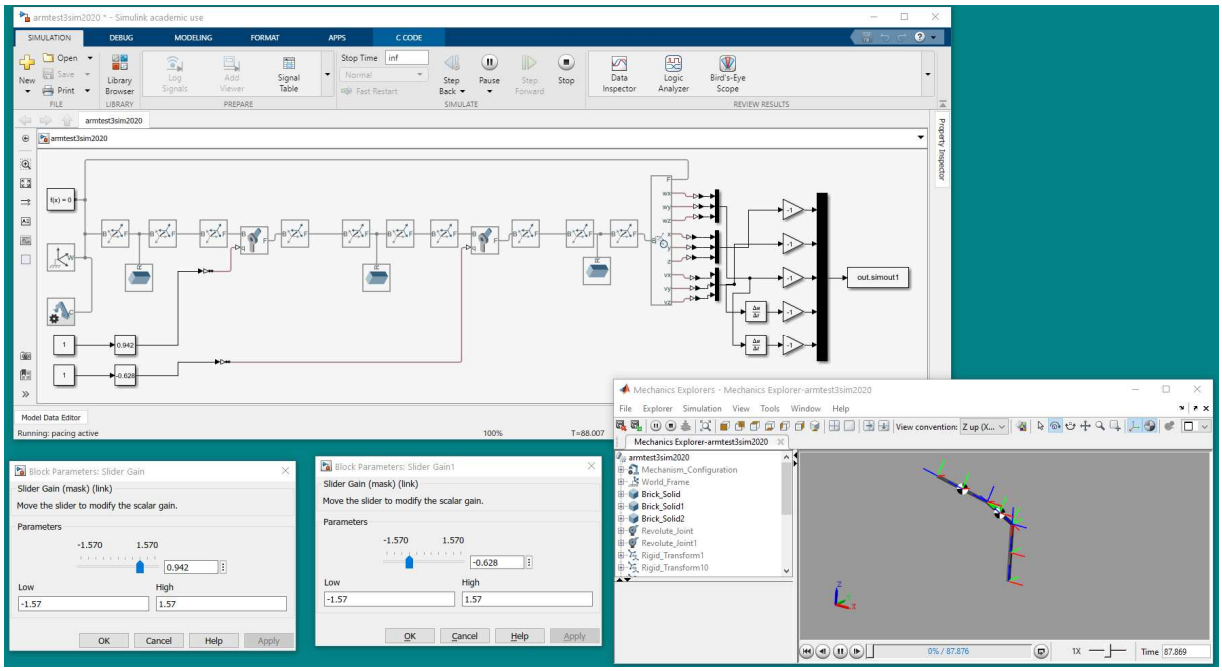


のチェックを入れる必要がある。

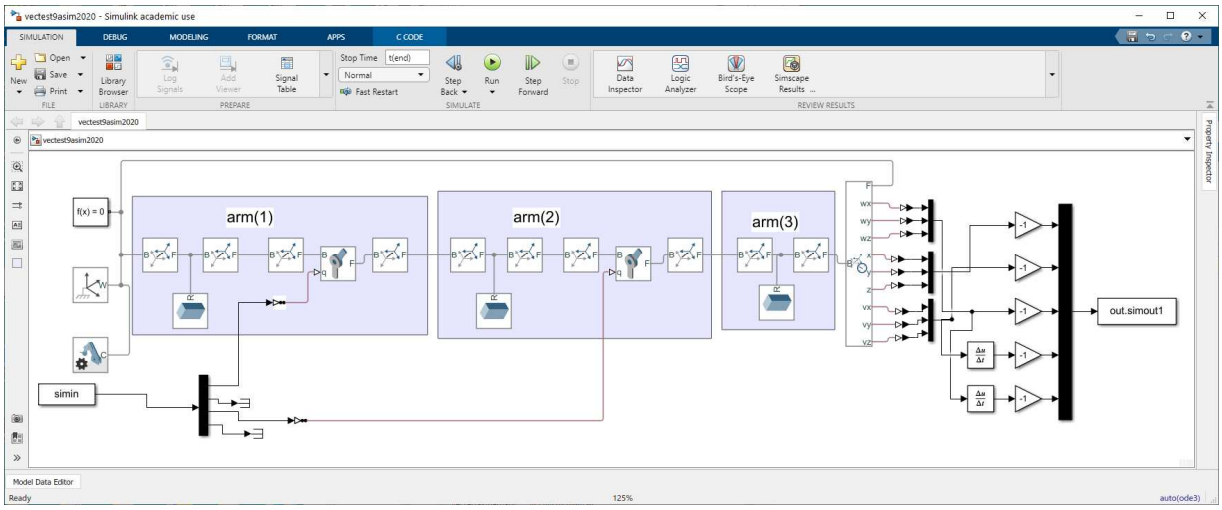
実行には、

```
>>vectest9demo
>>armtest3sim2020
```

とする。

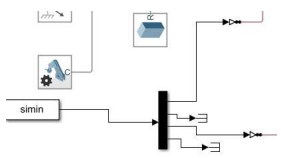


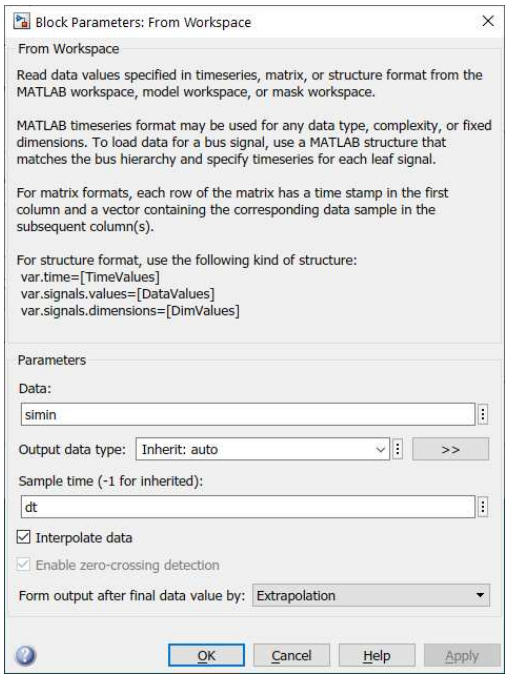
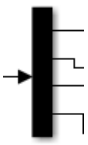
以下に MATLAB でのシミュレーション結果と比較するための Simscape モデルを示す。

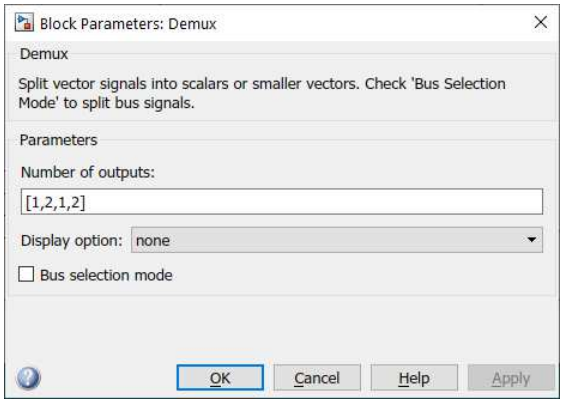


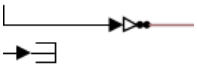
左側のブロックから順に設定の詳細を説明していく。

角度データのインポート

	MATLAB ワークスペースから角度データをインポートする。

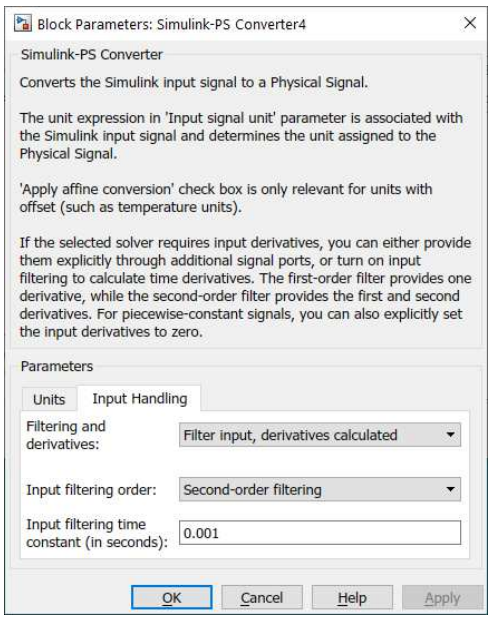
<table>
<tr>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td> 

simin には、th1,th2 の角度、角速度、角加速度も入っている。必要なのは、角度のみであるため、[1,2,1,2]とした。1 は、角度、2 は、角速度と角加速度の 2 つを表している。</td>
<td> 
</td>
</tr>
</table>

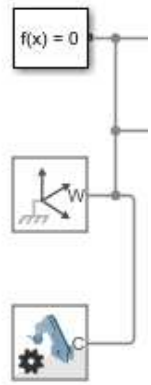


Terminator ブロックは、ワーニングを抑えるためのブロックである。

Simlink-PS　ブロックは、Units は、デフォルトのままでよいが、Input Handling は、右図のように変更する。ワークスペースから与えられる信号は、離散的なデータであるのに対し、Simscape 内では、微分方程式を解くため、信号の微分値が必要になる。そのため、2 次のフィルタをかけて、離散データを微分計算が可能になるように連続データに変換する必要がある。



座標設定



左の 3 つは、Simscape でシミュレーションをするために、必要なブロックである。

F(x)=0 は、微分方程式を解くための Solver　Configuration ブロックであり次は、ワールド座標系の原点を定める World Frame ブロックであり、そして最後のブロックは、重力の方向などを設定する Mechanism　Configuration ブロックである。

基本、これらのブロックの設定の変更は、必要ない。

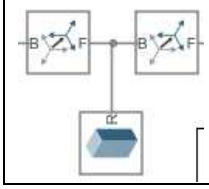
MATLAB シミュレーションモデルでは、重力を考慮していないので、

Mechanism Configuration Block

Uniform Gravity	None

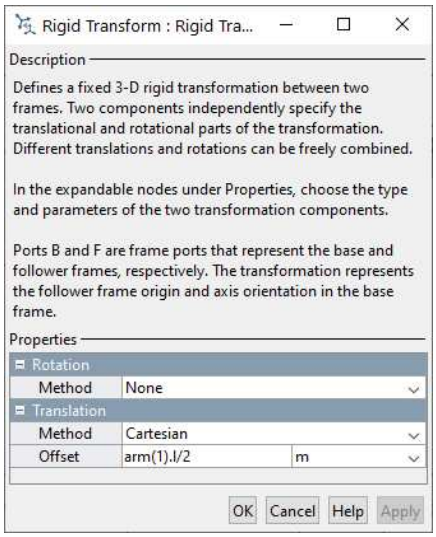
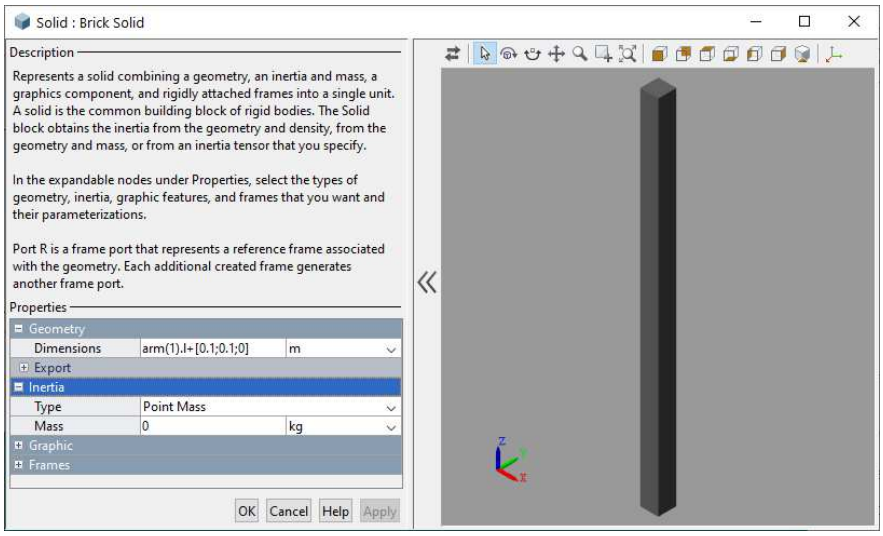
とする。

arm(1)の設定

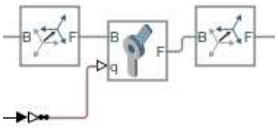


このブロックは、重心の位置と、その形状を指定するブロックである。

MATLAB のシミュレーションと同様に、棒状の形状の半分の長さのところに重心（形状ブロック）が来るように設定する。

Rigid Transform Block  前後とも同じパラメータ設定とした。	
 設定変更は、 Dimenstions Inertia のみで、他は、デフォルトとした。 MATLAB シミュレーションでは、慣性モーメントは、考慮していないので、重さを 0 とした。	

ジョイント部分の設定

	Revolute Joint Block であるが、z 軸周りしか回転しないため、前後に回転ブロックを入れて回転軸を変更している。

最初

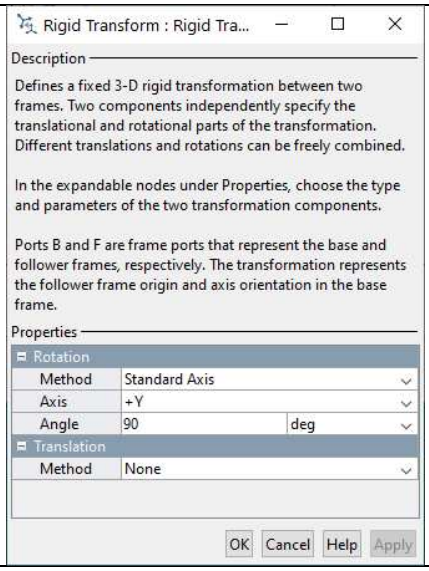
＋Y で 90 度

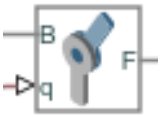
で設定後

Revolute Joint Block

その後、座標系を元に戻すため

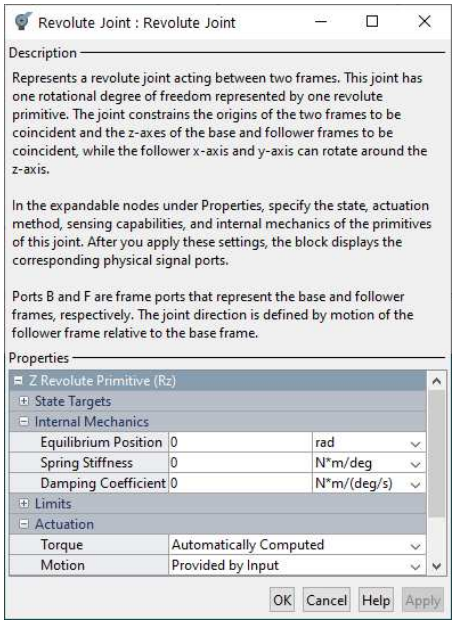
＋Y で―90 度とした。





Actuation の部分のみ設定を変更した。

Motion を Provided by Input とすると、左側に q の入力端子が出てくる。入力端子 q の単位は、MATLAB の変数と合わせるため rad とした。



arm(2)の設定

形状＋座標変換

Rigid Transform	Rotation	Method	**None**
	Translation	Method	**Cartesian**
		Offset	**arm(2).l/2　　m**

Brick Solid	Geometry	Dimension	**arm(2).l+[0.1;0.1;0]**
	Inertia	Type	**Point Mass**
		Mass	**0 kg**

Rigid Transform	Rotation	Method	**None**
	Translation	Method	**Cartesian**
		Offset	**arm(2).l/2 m**

ジョイント部分の設定

Rigid Transform	Rotation	Method	**Standard Axis**
		Axis	**-X**
		Angle	**90 deg**
	Translation	Method	**None**

Revolute Joint	Z Revolute Primitive (Rz)	Internal Mechanics/ Equilibrium Position	**0 rad**
		Actuation/Torque	**Automatically Computed**
		Actuation/Motion	**Provided by Input**

Rigid Transform	Rotation	Method	**Standard Axis**
		Axis	**-X**
		Angle	**-90 deg**
	Translation	Method	**None**

arm(3)の設定

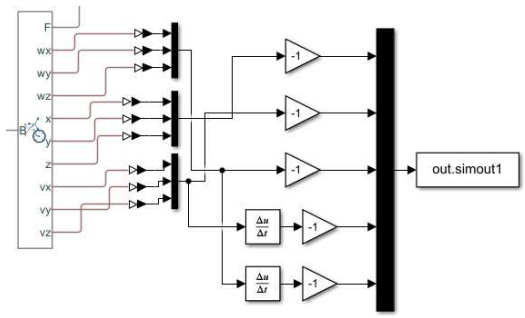
形状＋座標変換

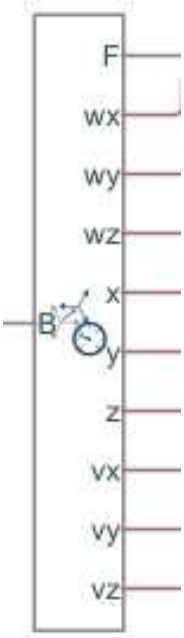
Rigid Transform	Rotation	Method	**None**
	Translation	Method	**Cartesian**
		Offset	**arm(3).l/2 m**

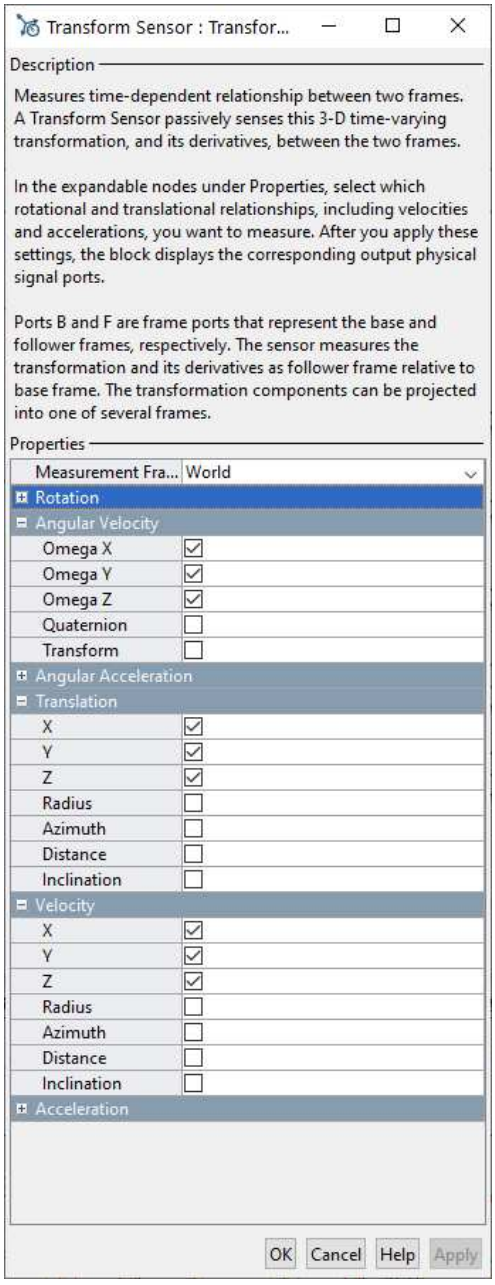
Brick Solid	Geometry	Dimension	**arm(2).l+[0.1;0.1;0]**
	Inertia	Type	**Point Mass**
		Mass	**0 kg**

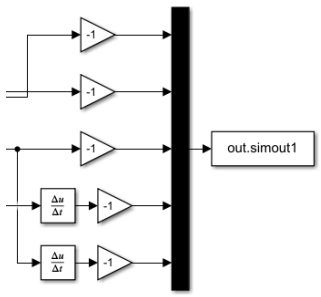
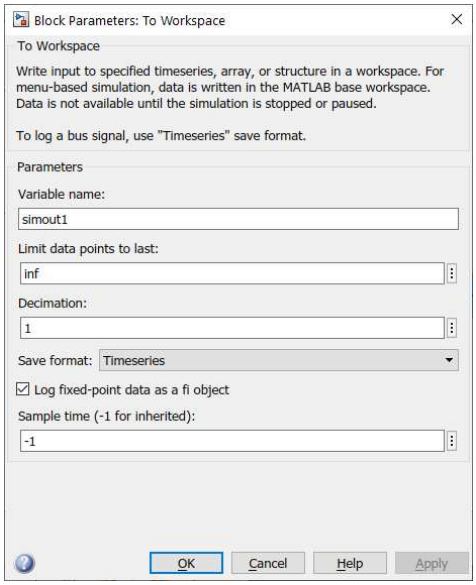
Rigid Transform	Rotation	Method	**None**
	Translation	Method	**Cartesian**
		Offset	**arm(3).l/2 m**

Transform Sensor

	ロボットアームの先端の座標、速度などを取り出し、MATLAB ワークスペースに保存する。

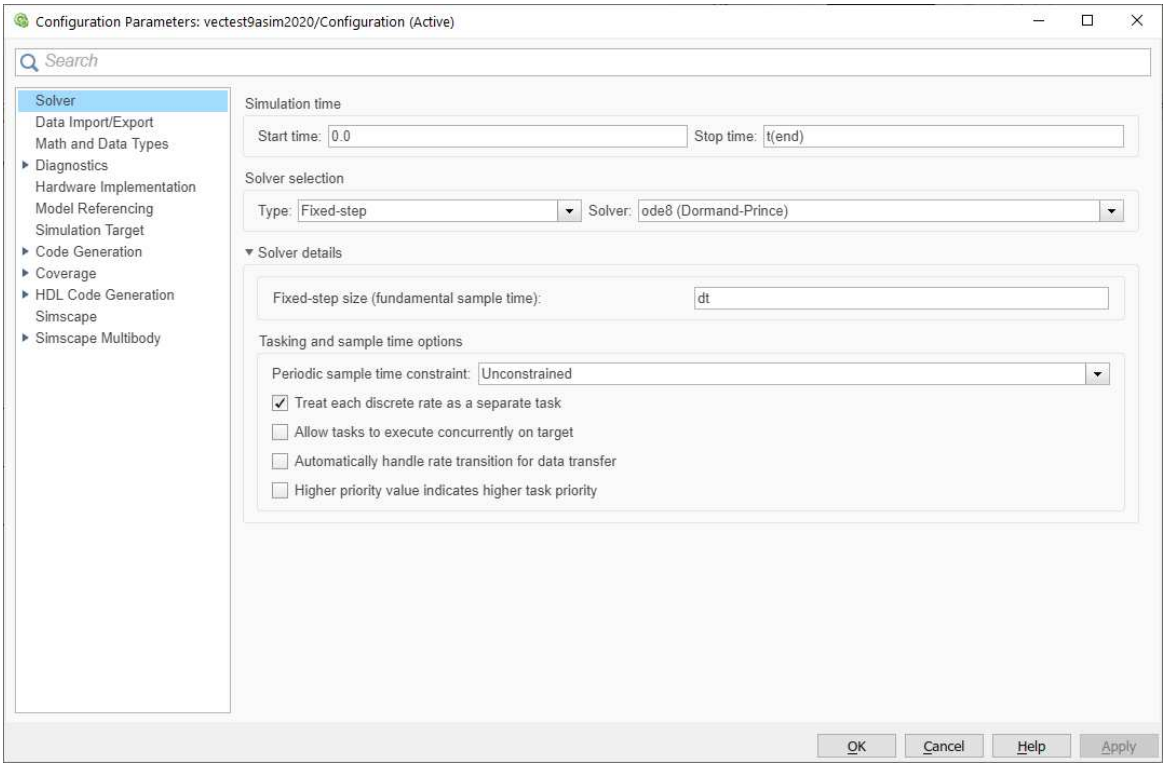
<table>
<tr>
<td> 

World Frame での位置 X,Y,Z　速度 vx,vy,vz
角速度 wx,wy,wz を出力する。</td>
<td> 
</td>
</tr>
<tr>
<td>PS-Simulink と Mux Block
  </td>
<td>PS-Simulink ブロックにより、位置や速度を Simulink で読める形式に変換している。3 次元データであるため、Mux block で、1 本の線にまとめて表現している。</td>
</tr>
</table>

	MATLAB での座標系に合わせるため、符号を反転している。なお、Transform Sensor ブロックで、加速度、角加速度も出力することもできるが、数値に問題があるようであったため、それぞれ代替で、速度、角速度を微分ブロックで計算するようにし、さらに Mux ブロックで、信号をまとめて、To Workspace ブロックにより、データをエクスポートする。
	

Solver の設定

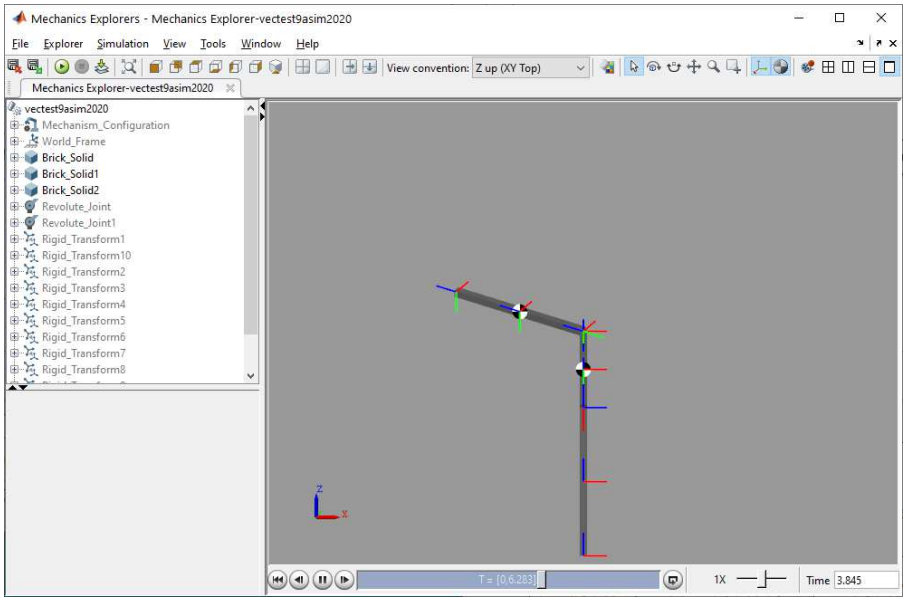
Configuration Parameter の設定は、つぎのようにした。



検証は、

```
>>vectest9asim2020demo
```

で実行できる。



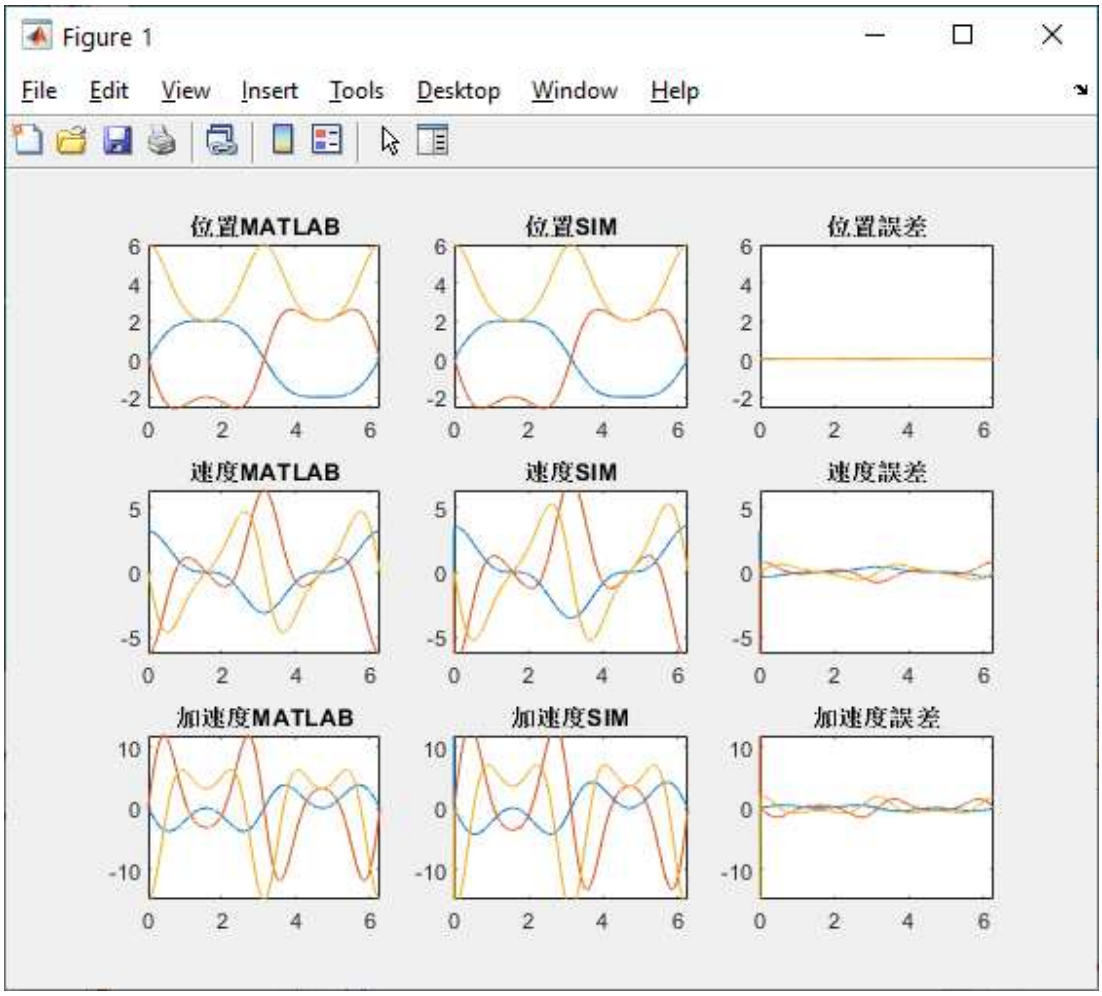


図　MATLAB/Simscape Multibody の位置、速度、加速度のシミュレーション比較

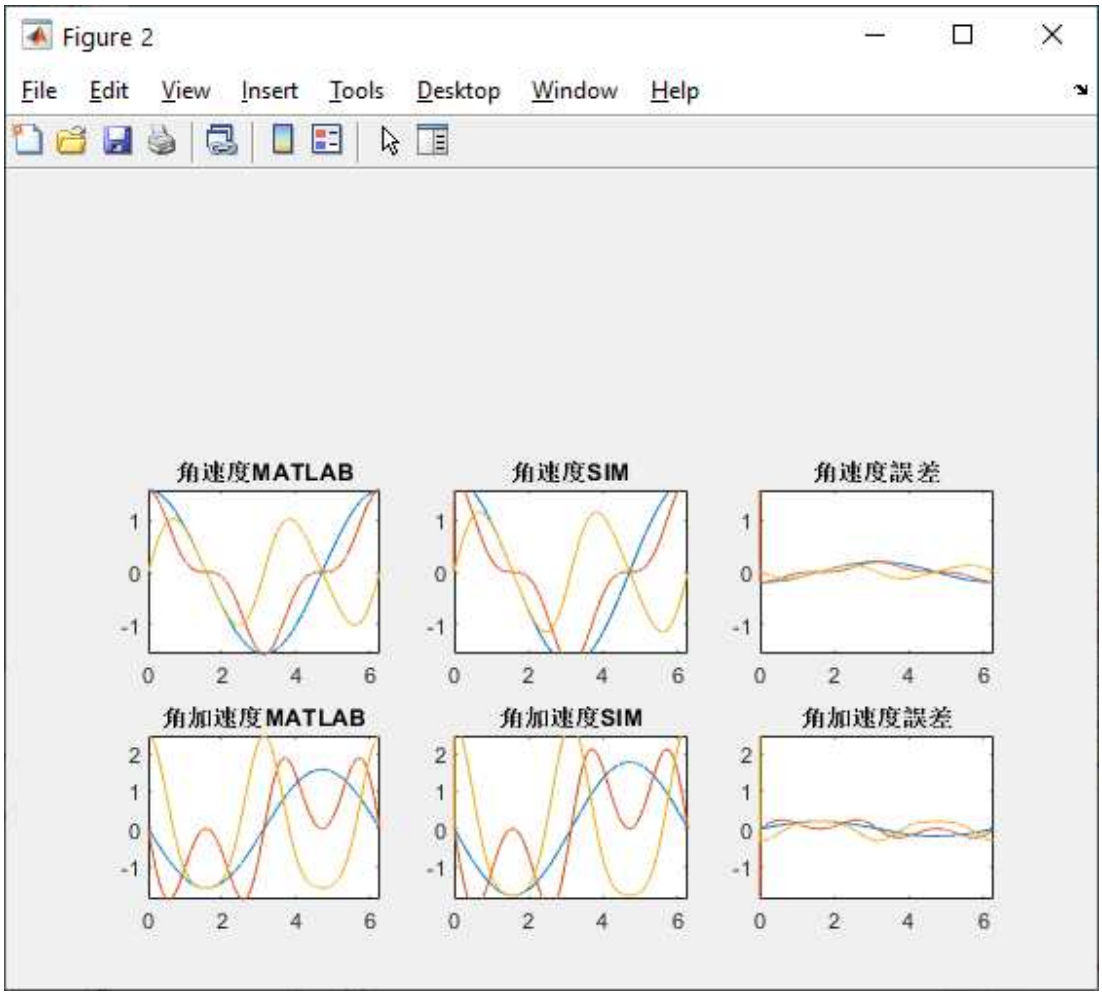


図　MATLAB/Simscape Multibody の角速度、角加速度のシミュレーション比較

なお、加速度誤差、角加速度誤差の計算は、微分ブロックを使っているため、最初のデータは、かなり大きな値になってしまう。そのため、加速度誤差、角加速度誤差の軸の設定は、それぞれ、加速度、角加速度の軸の設定を使用した。

速度、加速度とも差分結果を見てみると、ほぼ、0 であり結果が一致していることがわかる。このように、複数のシミュレーションにより検証することは、重要である。

vectest9asim2020demo.m

```
close all;clear all
if 1
viewagl=[-37.5,30;0,90;90,0;0,0];
n=3;
l=[zeros(2,n);2*ones(1,n)];
th=zeros(1,n);
lg=l/2;%重心位置
%同次変換座標行列の定義より、arm(1)の重心 lg は、関節が動いても lg の位置には作用しない。
%
arm(1).th=th(1);arm(1).a=[1;0;0];arm(1).l=l(:,1);arm(1).lg=lg(:,1);
arm(2).th=th(2);arm(2).a=[0;1;0];arm(2).l=l(:,2);arm(2).lg=lg(:,2);
arm(3).th=th(3);arm(3).a=[1;0;0];arm(3).l=l(:,3);arm(3).lg=lg(:,3);

% define arm list structure
arm(1).parent=0;arm(2).parent=1;arm(3).parent=2;
% 子リンクを接続し双方向リンクとする。
arm=vectest9makechildlink(arm);
ind=vectest9findallchildren(arm);

dt=0.001;t=0:dt:2*pi;
th  = pi/2*sin(t);
dth =diff(th)/dt;dth =[dth(:,1), dth];
ddth=diff(dth)/dt;ddth=[ddth(:,1),ddth];
%Simulink モデルは、収縮しないモデルと仮定
l = 2 + 0*sin(t);
dl = diff(l)/dt;dl=[dl(:,1),dl];
ddl=diff(dl)/dt;ddl=[ddl(:,1),ddl];

lg = 1 + 0*sin(t);
dlg = diff(lg)/dt;dlg=[dlg(:,1),dl];
ddlg=diff(dlg)/dt;ddlg=[ddlg(:,1),ddl];

plot1=[];
P{length(arm)}=[];DP{length(arm)}=[];DDP{length(arm)}=[];
Pg{length(arm)}=[];DPg{length(arm)}=[];DDPg{length(arm)}=[];
R{length(arm)}=[];W{length(arm)}=[];DW{length(arm)}=[];
for ii=1:length(t)
        for i=1:length(arm)
                arm(i).th=th(ii);
                arm(i).dth=dth(ii);
                arm(i).ddth=ddth(ii);
                arm(i).l=[0;0;l(ii)];
                arm(i).dl=[0;0;dl(ii)];
                arm(i).ddl=[0;0;ddl(ii)];
                arm(i).lg=[0;0;lg(ii)];
                arm(i).dlg=[0;0;dlg(ii)];
                arm(i).ddlg=[0;0;ddlg(ii)];
        end
        ind=vectest9findallchildren(arm);
        arm=vectest9kinematics3(arm,ind);
        for i=1:length(arm)
                P{i}=[P{i},arm(i).T(1:3,4)];
                DP{i}=[DP{i},arm(i).dT(1:3,4) ];
                DDP{i}=[DDP{i},arm(i).ddT(1:3,4)];
                if(~isempty(arm(i).Tg))
                        Pg{i}  =[Pg{i}  ,arm(i).Tg(1:3,4)  ];
                        DPg{i} =[DPg{i} ,arm(i).dTg(1:3,4) ];
```

```
                        DDPg{i}=[DDPg{i},arm(i).ddTg(1:3,4)];
                end
                dR = arm(i).V(1:3,1:3);
                w = [dR(3,2);dR(1,3);dR(2,1)];% or w = -[dR(2,3);dR(3,1);dR(1,2)];
                ddR = arm(i).A(1:3,1:3);
                dw = [ddR(3,2);ddR(1,3);ddR(2,1)];% or dw = -[ddR(2,3);ddR(3,1);ddR(1,2)];
                R{i}  = [R{i} ,r2ath(arm(i).T(1:3,1:3))];
                W{i}  = [W{i} ,w];
                DW{i} = [DW{i},dw];
        end
        plot1 = vectest9plot(arm,plot1);
        drawnow;
end
simin.time=t(:);
simin.signals.values=[th(:),dth(:),ddth(:),th(:),dth(:),ddth(:)];
simin.dimenstions=6;
save vectest9mat2020
else
    load vectest9mat2020
end
out=sim('vectest9asim2020','ReturnWorkspaceOutputs', 'on')
simout1=out.simout1;
close all
figure
subplot(3,3,1);plot(t,P{3}');axis tight;title('位置 MATLAB');
paxis=axis;
subplot(3,3,2);plot(t,simout1.Data(:,1:3));title('位置 SIM');
axis(paxis);
subplot(3,3,3);plot(t,P{3}'-simout1.Data(:,1:3));title('位置誤差');
axis(paxis);
subplot(3,3,4);plot(t,DP{3});axis tight;title('速度 MATLAB')
dpaxis=axis;
subplot(3,3,5);plot(t,simout1.Data(:,4:6));title('速度 SIM')
axis(dpaxis);
subplot(3,3,6);plot(t,DP{3}'-simout1.Data(:,4:6));title('速度誤差');
axis(dpaxis);
subplot(3,3,7);plot(t,DDP{3}');axis tight;title('加速度 MATLAB')
accaxis=axis;
subplot(3,3,8);plot(t,simout1.Data(:,10:12));title('加速度 SIM')
axis(accaxis);
subplot(3,3,9);plot(t,DDP{3}'-simout1.Data(:,10:12));title('加速度誤差');
axis(accaxis);

figure
subplot(3,3,4);plot(t,W{2}');axis tight;title('角速度 MATLAB')
waxis=axis;
subplot(3,3,5);plot(t,simout1.Data(:,7:9));title('角速度 SIM')
axis(waxis);
subplot(3,3,6);plot(t,W{2}'-simout1.Data(:,7:9));title('角速度誤差');
axis(waxis);

subplot(3,3,7);plot(t,DW{2}');axis tight;title('角加速度 MATLAB')
accwaxis=axis;
subplot(3,3,8);plot(t,simout1.Data(:,13:15));title('角加速度 SIM')
axis(accwaxis);
```

```
subplot(3,3,9);plot(t,DW{2}'-simout1.Data(:,13:15));title('角加速度誤差');
axis(accwaxis);
```