MATLAB EXPO 2024

人型遠隔操作ロボットのMBD開発とリアルタイムシミュレーション

2024/5/30

三菱電機ソフトウエア株式会社
神田吉孝

自己紹介

神田（こうだ）吉孝

１９９１年 三菱電機マイコン機器ソフトウエア株式会社 入社

主に衛星通信アンテナ駆動制御装置に従事

２０２２年 三菱電機ソフトウエア株式会社に統合

現所属/職位：通信機事業所　防衛技術第一部　主管技師長

©国立天文台

https://www.nao.ac.jp/en/research/telescope/subaru.html

©国立天文台

https://www.nao.ac.jp/en/research/telescope/alma.html

©宇宙航空研究開発機構

https://www.jaxa.jp/projects/sas/mdss/index_j.html

0 目次

参考資料

三菱電機ソフトウエア技術ライブラリ

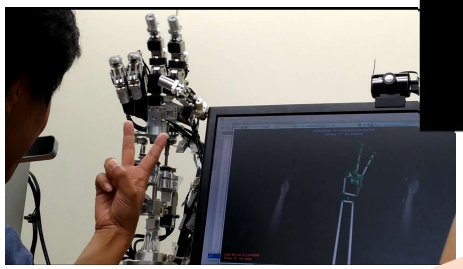
「モデルベース開発手法によるリアルタイムシミュレーション」

https://www.mesw.co.jp/business/report/pdf/mms_29_01.pdf

1 人型遠隔操作ロボット「DiaroiD®」

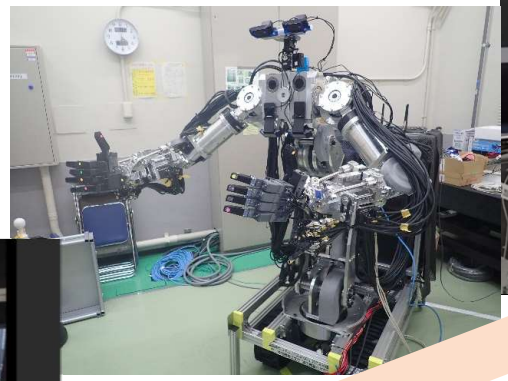
三菱電機(株)が開発中の人型遠隔操作ロボット「**DiaroiD®**」

- 42自由度(試作2号機)
- ステレオカメラによる立体視
- 20kgを超える双腕出力
- 一般的な電動工具を扱えるハンド
- 視覚的触覚フィードバック

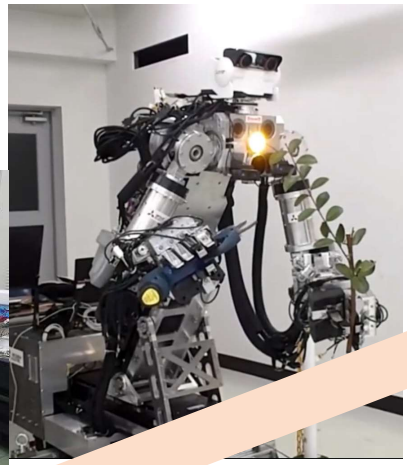


2017年 右手試作

2018年 試作1号機



2019年 試作2号機
2020年 試作2号機改



2021年 試作3号機

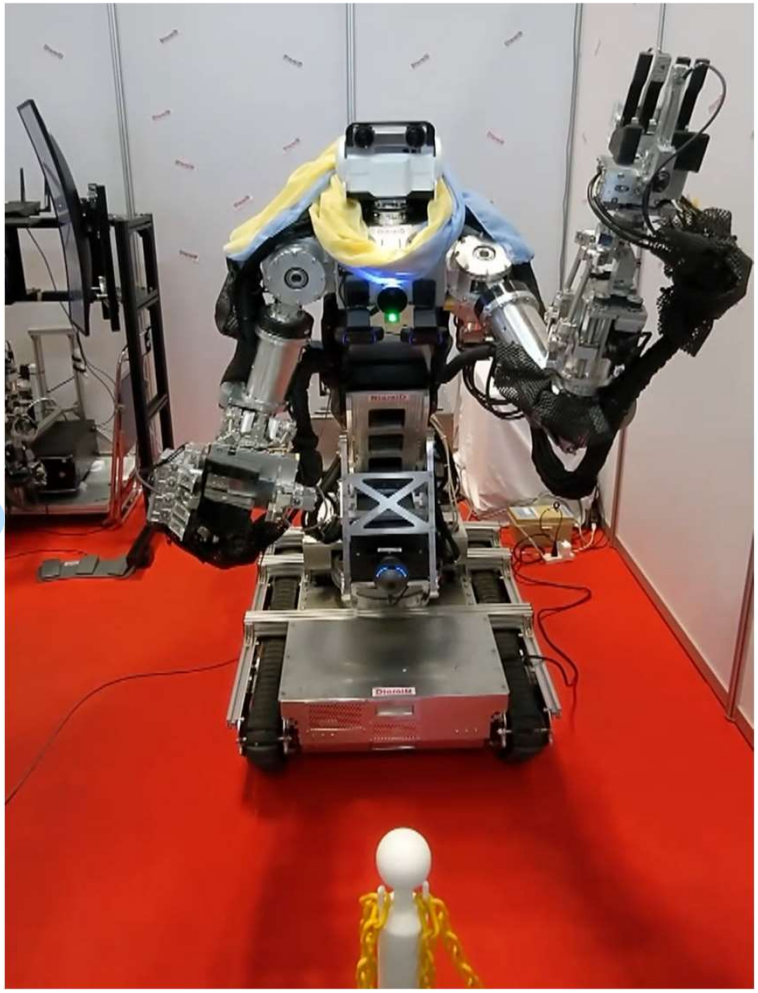
2024年～
量産1号機

1 人型遠隔操作ロボット「DiaroiD®」

２０２４年５月３０日～３１日
関西ロボットワールド2024
出展中！

もしも、明日、大阪に
行く方がいらしゃれば、
お待ちしております。

こちらは2022年のブース展示の様子



関西ロボットワールド2022の様子

1 人型遠隔操作ロボット「DiaroiD®」

(株)クボタ様との協業

DiaroiD® on KATV

先ほど
傾斜不整地向け農業用ロボット「KATV(仮称)」
のご講演がございました。

Model-Based Designツールを活用した傾斜不整地向け農業用ロボットの開発

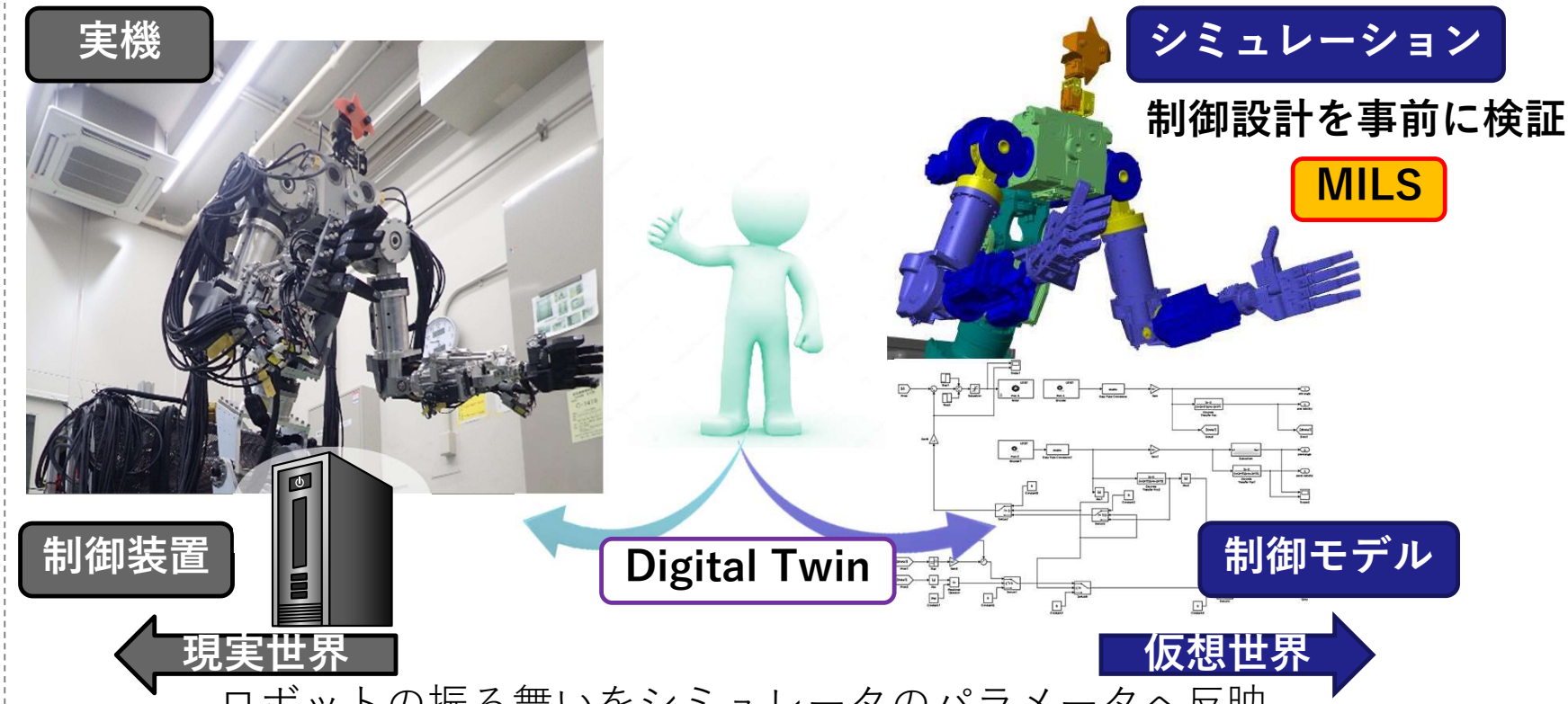
果樹園は既存の4輪車両やクローラでは走行が困難な傾斜不整地である。そこで、動物の脚のような機構と力制御技術を組み合わせて、世界初の傾斜不整地向け農業用ロボットを考案した。本機はハード・ソフト共に新規性が高く、システムも複雑であるため、Model-Based Designツールを活用して開発を進めた。その結果、機能検証サイクルの短縮、実機試験回数の低減に繋がり、スピーディに製品化を実現できた。本講演では、MATLAB®、Simulink®、Simscape™を用いた制御アルゴリズムの構築、不整地の走行シミュレーション事例等を紹介する。

制御　ﾌﾟﾗﾝﾄﾓﾃﾞﾘﾝｸﾞ　ﾛﾎﾞﾃｨｸｽ　ﾓﾃﾞﾙﾍﾞｰｽﾃﾞｻﾞｲﾝ　3d仮想環境

株式会社クボタ
井田 裕介

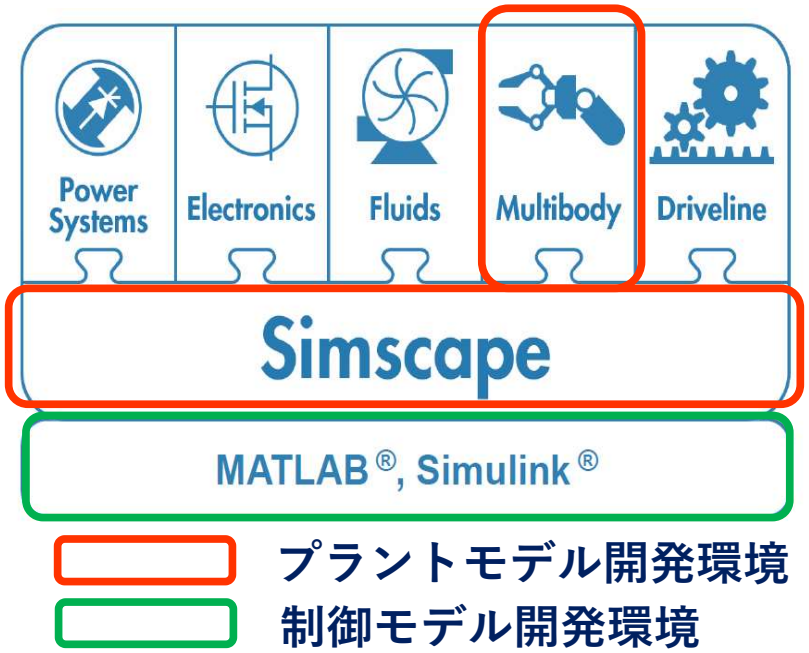
全地形対応プラットフォーム車両
KATV KUBOTA All Terrain Vehicle（仮称）

2 開発概要



ロボットの振る舞いをシミュレータのパラメータへ反映
実機を使わずに適格性確認が可能に

2.1 開発概要　開発ツール

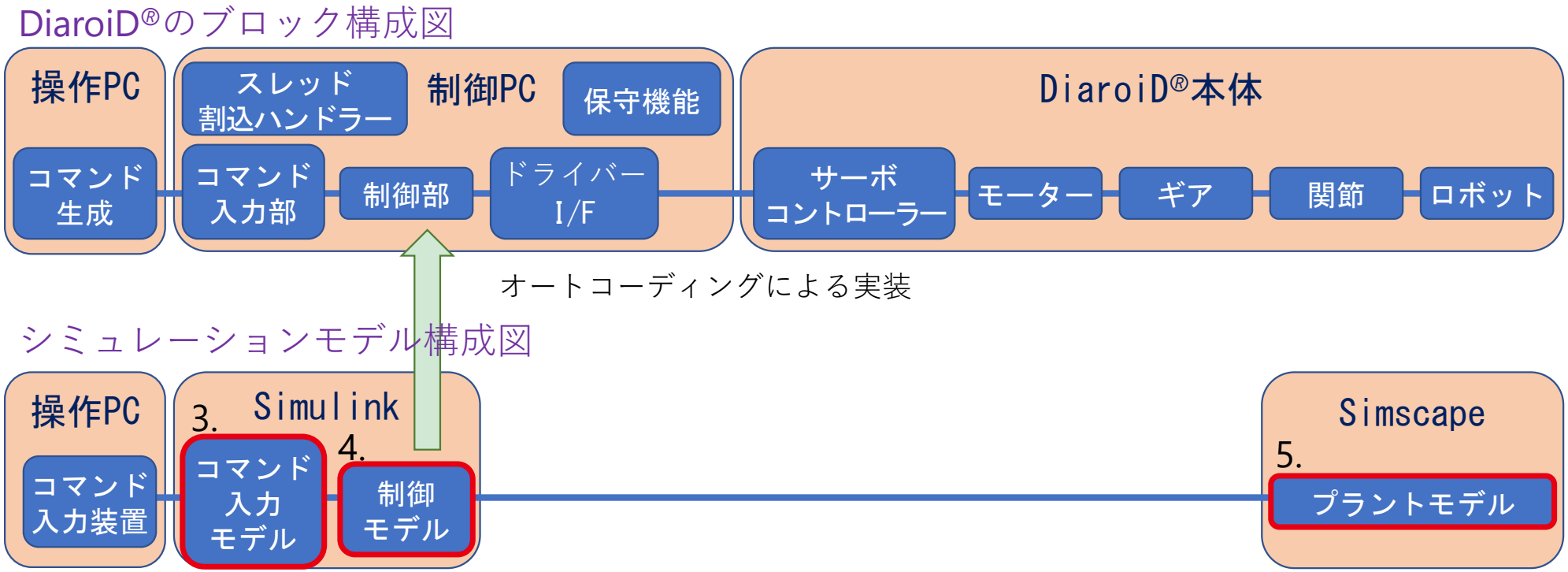


開発ツール

MATLAB®
Simulink®
Simscape™
Simscape™ Multibody™
　Control System Toolbox™
　Symbolic Math Toolbox™
　Simulink® Control Design™
　MATLAB® Coder™
　Simulink® Coder™
　Embedded Coder®
　MATLAB® Support for
　　MinGW-w64 C/C++/Fortran Compiler
　Simscape™ Multibody™ Link

2.2 開発概要　モデル構成

下図は、DiaroiD®のブロック構成図とシミュレーションモデル構成図の比較である。



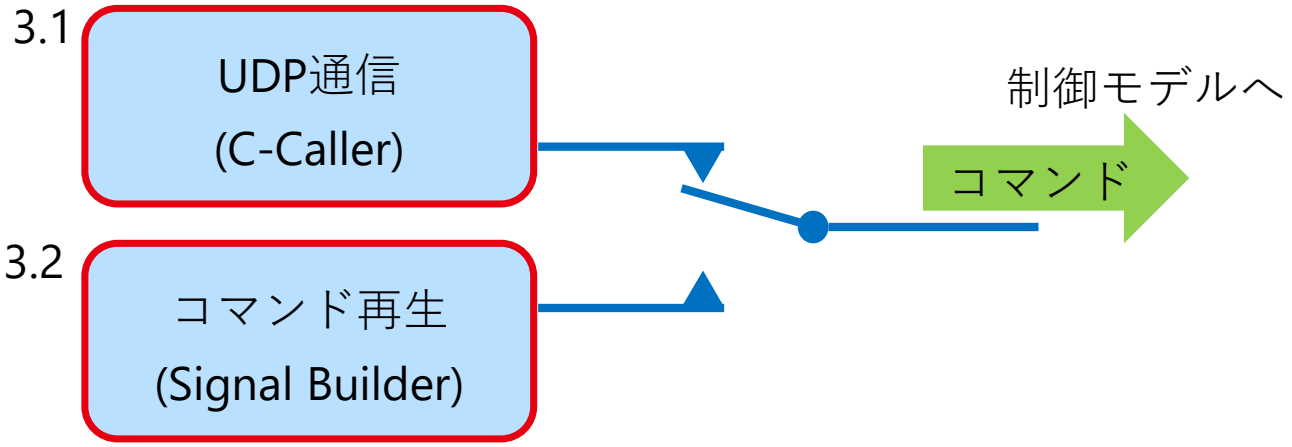
DiaroiD®本体の構成とモデル構成の対応

赤枠 のモデルの制作過程を説明する。

3 コマンド入力モデル

外部から指令入力（UDP通信）は、C Callerを使用した。

定型指令（コマンド再生）には、Signal Builderを使用した。



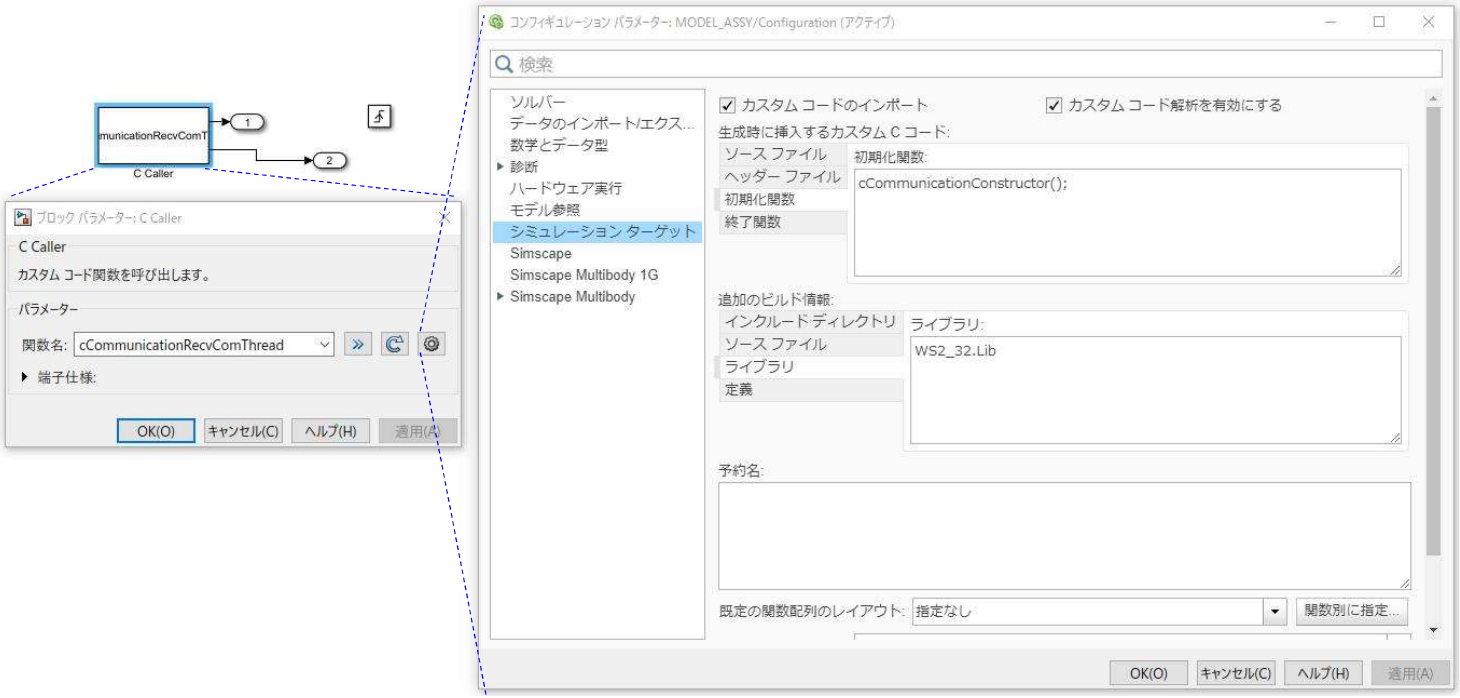
コマンド入力モデル

赤枠のブロックの制作過程を説明する。

3.1 コマンド入力モデル　UDP通信

外部から指令入力は、UDP通信にて実現した。

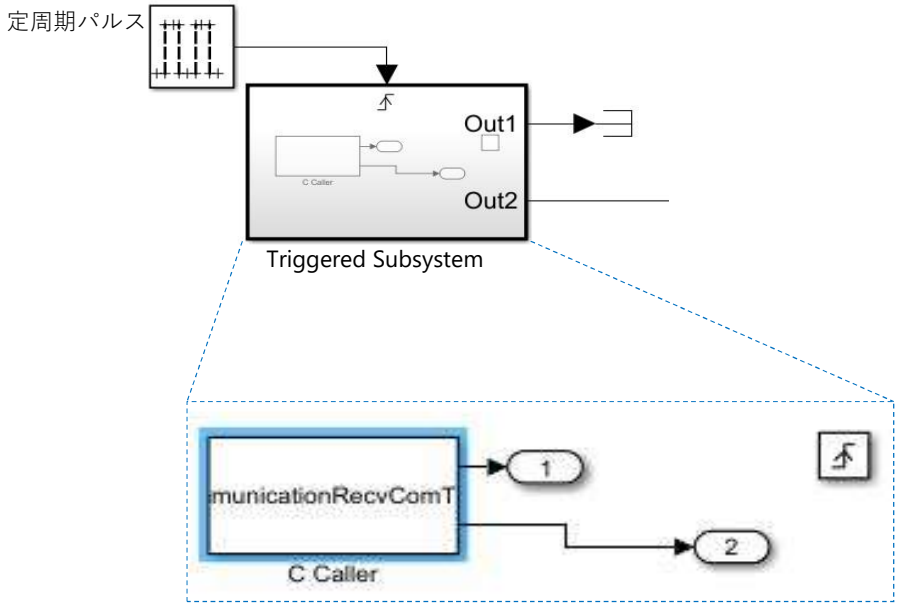
Simulinkに通信用のブロックは実装されておらず、C Callerブロックを利用して自作した。



C Callerのプロパティ

3.1 コマンド入力モデル　UDP通信

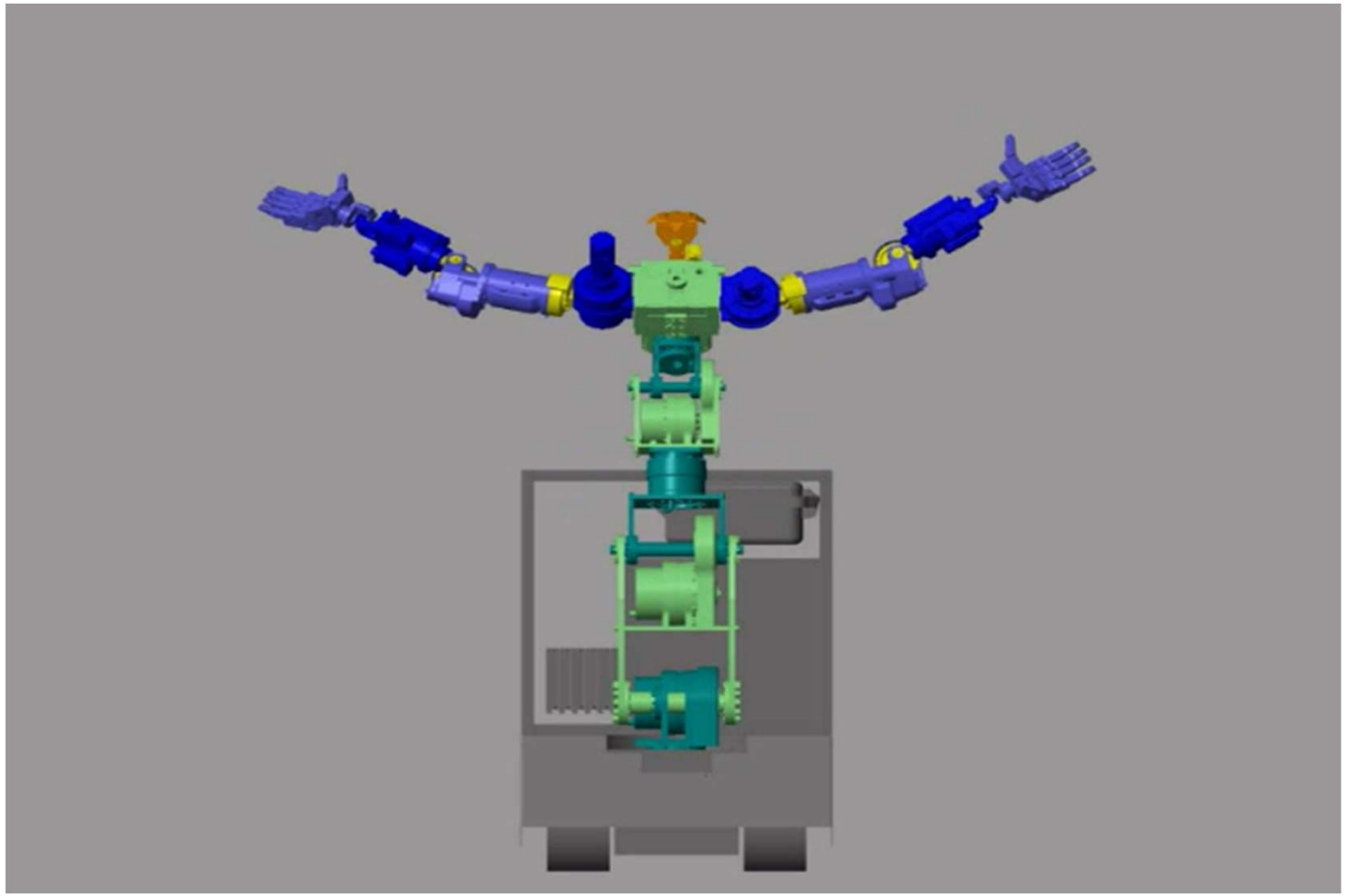
C Caller ブロックが呼び出す関数は、離散化を前提としているため、連続系環境ではシミュレーションできない。
　離散化ブロックを連続系の中で動作させる方法として、Triggered Subsystem が用意されている。



Triggered Subsystemの定義

C Callerブロックを、Triggered Subsystemとして定義。
　定周期のパルスをTriggerとして与えることにより、 C Callerブロックを定周期起動する離散系ブロックとして駆動させることができる。

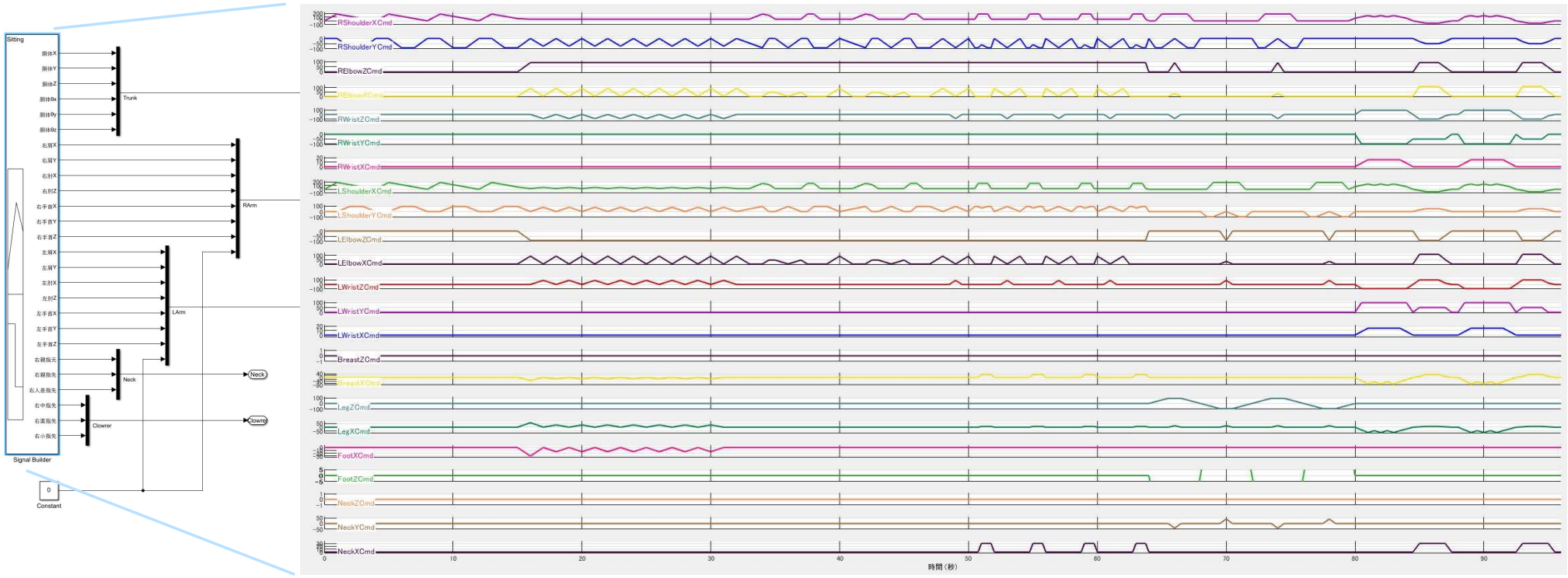
3.2 コマンド入力モデル　コマンド再生



ラジオ体操するDiaroiD®

3.2 コマンド入力モデル　コマンド再生

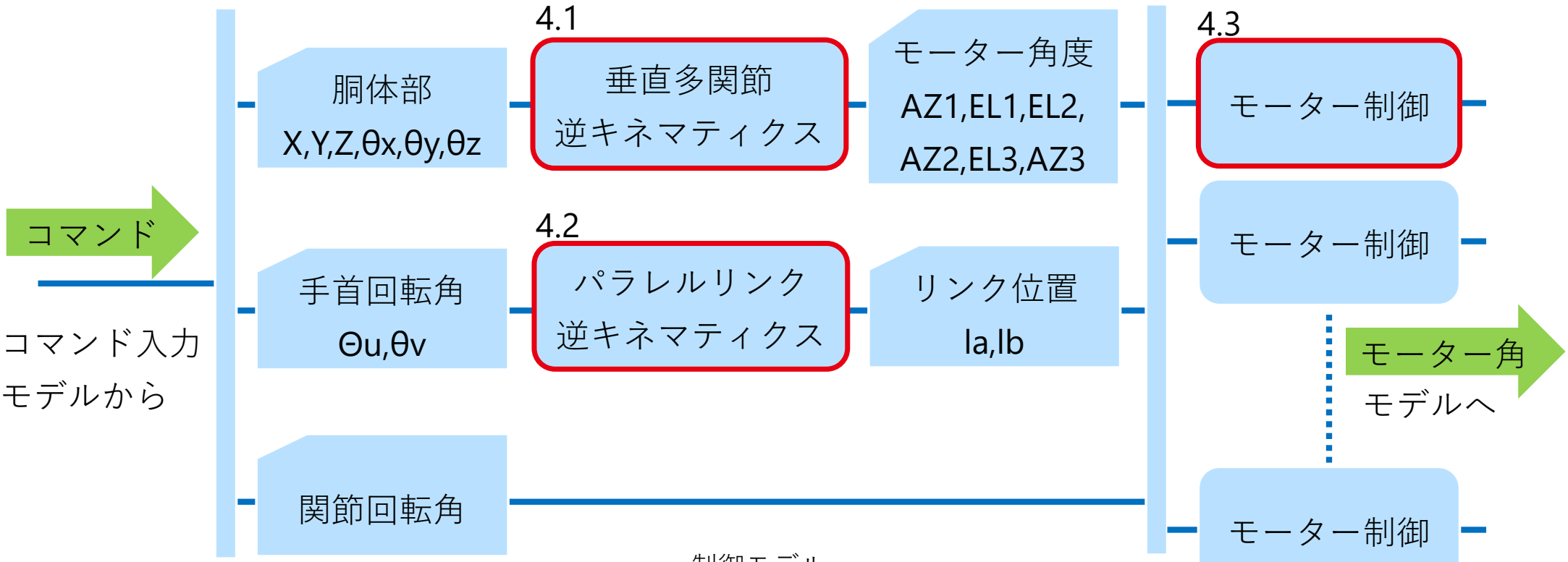
定型コマンドの作成には、Signal Builderを使用した。
カットアンドトライでラジオ体操のコマンド作成した。



Command Builderによるロボット駆動コマンド

4 制御モデル

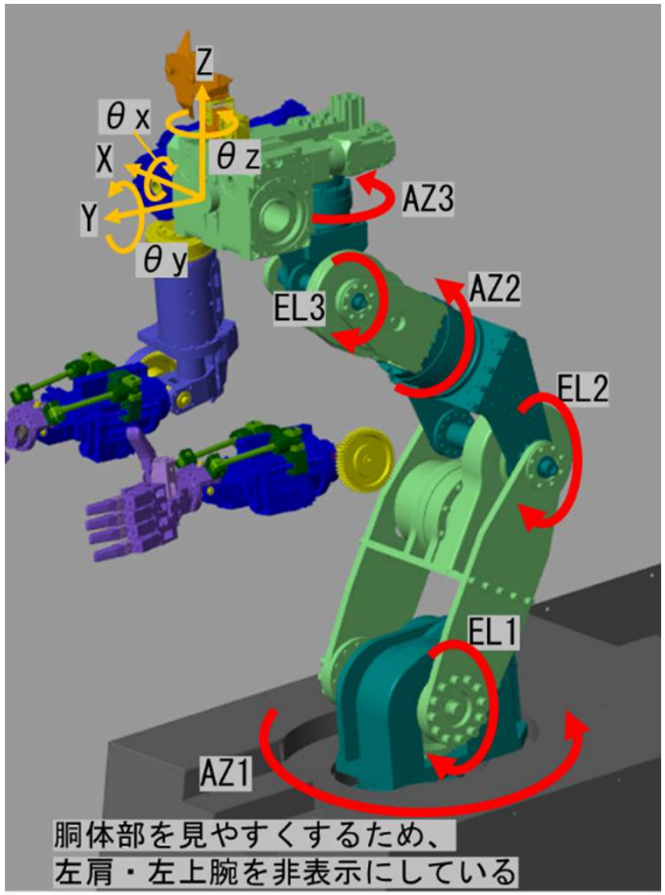
制御モデルはFunction Blockで作成した。下図は制御モデルブロック図である。



制御モデル

赤枠のブロックの制作過程を説明する。

4.1 制御モデル　垂直多関節逆キネマティクス



胴体部の軸構成

胴体部は垂直多関節で構成されておりキネマティクスは以下の行列式で求められる。この逆行列をニュートンラプソン法で求める。

位置

$$[x \quad y \quad z \quad 1] = Rot(\theta AZ1, z) \cdot Trans(P1) \cdot Rot(\theta EL1, x) \cdot Trans(P2) \cdot Rot(\theta EL2, x) \cdot Trans(P3) \cdot Rot(\theta AZ2, z) \cdot Trans(P4) \cdot Rot(\theta EL3, x) \cdot Trans(P5) \cdot Rot(\theta AZ3, z) \cdot Trans(P6) \cdot [0 \quad 0 \quad 0 \quad 1]^t$$

回転

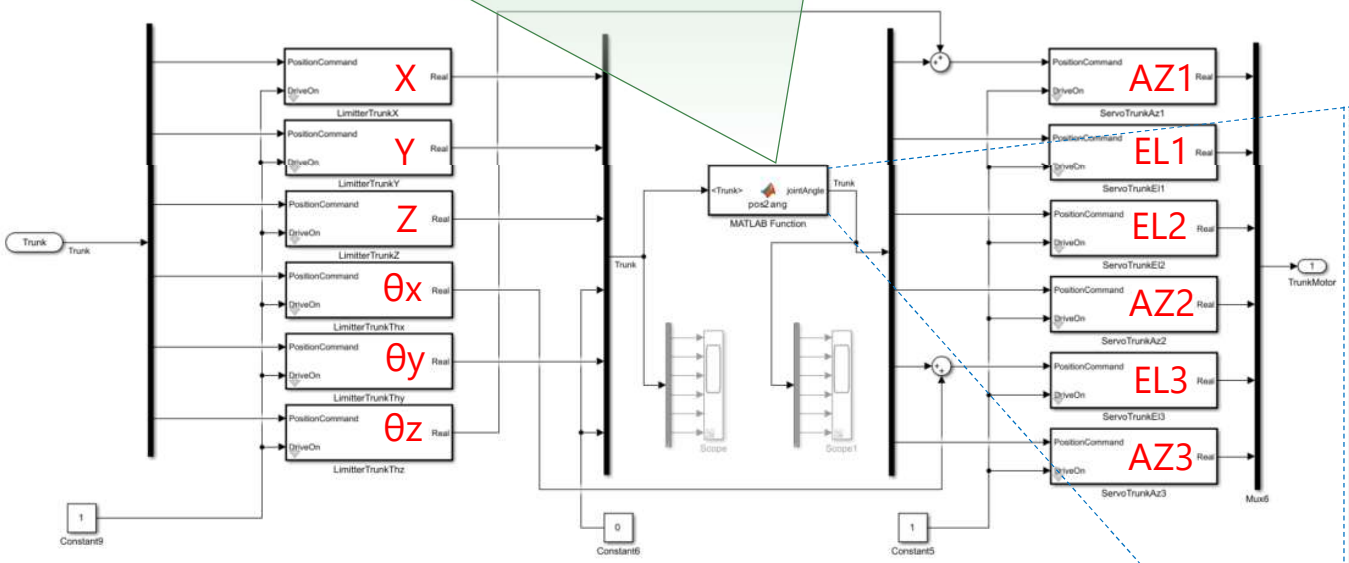
$$[Q] = Rot(\theta AZ1, z) \cdot Rot(\theta EL1, x) \cdot Rot(\theta EL2, x) \cdot Rot(\theta AZ2, z) \cdot Rot(\theta EL3, x) \cdot Rot(\theta AZ3, z)$$

$$[R] = Rot(\theta x, x) \cdot Rot(\theta y, y) \cdot Rot(\theta z, z)$$

$$\theta x = tan^{-1}\frac{-Q_{23}}{Q_{33}}, \qquad \theta y = sin^{-1} Q_{21}, \qquad \theta z = tan^{-1}\frac{-Q_{12}}{Q_{11}}$$

4.1 制御モデル　垂直多関節逆キネマティクス

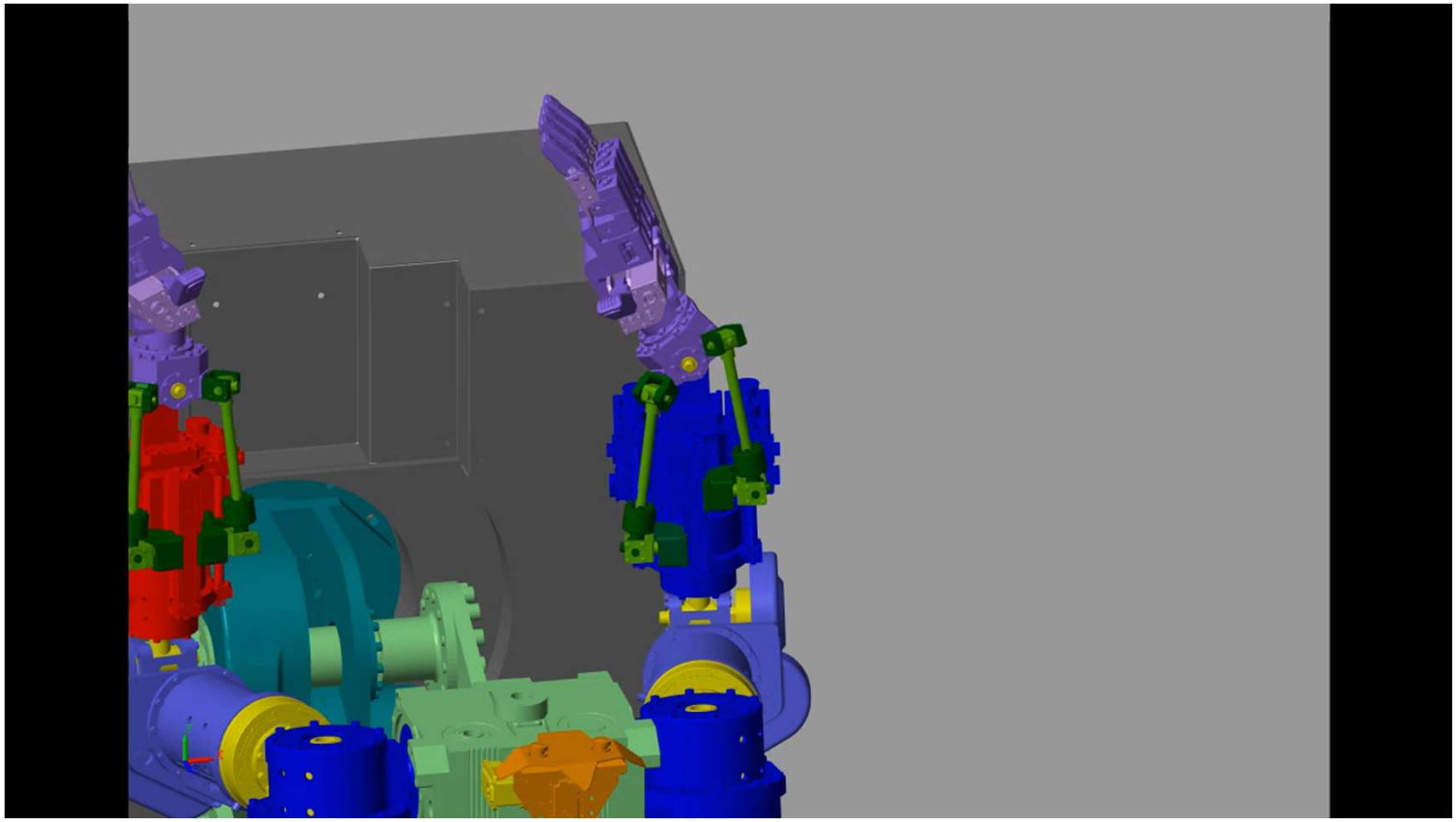
MATLAB言語でプログラムし、MATLAB Functionブロックで実装



MATLAB Functionブロック

```
function jointAngle = pos2ang(posture)

%//============ test_fix_param.m ======================================
%////////////////////////////////////////////////////////////////////////
%// 固定パラメータ
%////////////////////////////////////////////////////////////////////////
%// A点位置の初期値
p0_xyz = [0; 500; 11650];

%// A点向きの初期値
%// ※A点から見た向き
d0_xyz = [0; norm(p0_xyz); 0];

%// 関節(AZk) 角度初期値
tha  = [0, 0, 0];

%// 関節(ELk) 角度初期値
the  = [0, 0, 0];

%// 関節(AZk) 位置初期値
pos_az = [0,0,0; 0,-1000,8100; 0,-1000,11650];

%// 関節(ELk) 位置初期値
pos_ev = [0,0,1150; 0,0,5300; 0,-1000,9950];

%// 逆運動時パラメータ
eps_y = 1e-6;
eps_e = 1e-6;
eps_pp = 1e-6;
n_max = 100;
th_az0_rad = [0.0, 0.0, 0.0];   %// az初期値(基本、何でも良い)
th_ev0_rad = [30.0, -60.0, 30.0] / 180 * pi;   %// ev初期値(ev(3)は自由度変更があるので、0以外)

%//====================================================================

%// 領域確保
pn1_xyz = [0;0;0];
pr1_xyz = [0;0;0];

%// 入力
pn1_xyz(1) = posture(1) * 100;
pn1_xyz(2) = posture(2) * 100;
pn1_xyz(3) = posture(3) * 100;
```

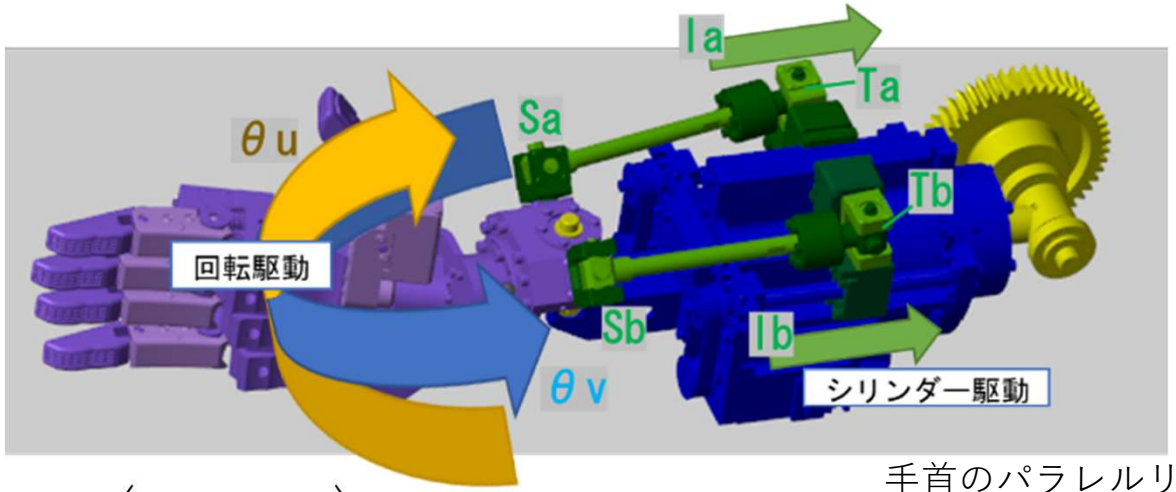
4.2 制御モデル　パラレルリンク逆キネマティクス



パラレルリンク機構の動作イメージ

4.2 制御モデル　パラレルリンク逆キネマティクス

手首はパラレルリンクで構成されておりキネマティクスは以下の行列式で求められる。
この逆行列をニュートンラプソン法で求める。

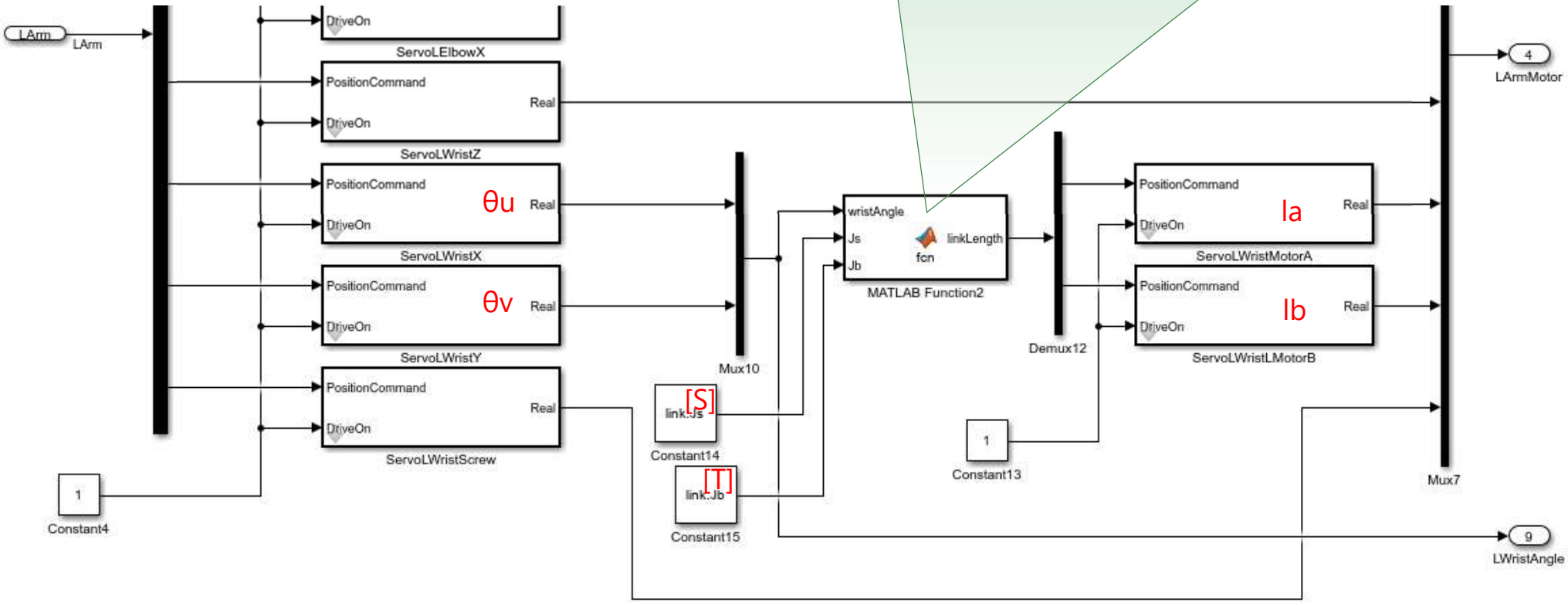


手首のパラレルリンク構成

- $[S] = \begin{pmatrix} S_{ax} & S_{bx} & 0 \\ S_{ay} & S_{by} & 0 \\ S_{az} & S_{bz} & 0 \end{pmatrix}$, $[T] = \begin{pmatrix} T_{ax} & T_{bx} & 0 \\ T_{ay} & T_{by} & 0 \\ T_{az} & T_{bz} & 0 \end{pmatrix}$ $\widehat{T_a} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$, $\widehat{T_b} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$
- $[S'] = Rot(\theta v, y) \cdot Rot(\theta u, x) \cdot [S]$
- $\left|\overrightarrow{S'_a} - \left(\overrightarrow{T_a} + l_a \widehat{T_a}\right)\right| = \left|\overrightarrow{S_a} - \overrightarrow{T_a}\right|$, $\left|\overrightarrow{S'_b} - \left(\overrightarrow{T_b} + l_b \widehat{T_b}\right)\right| = \left|\overrightarrow{S_b} - \overrightarrow{T_b}\right|$

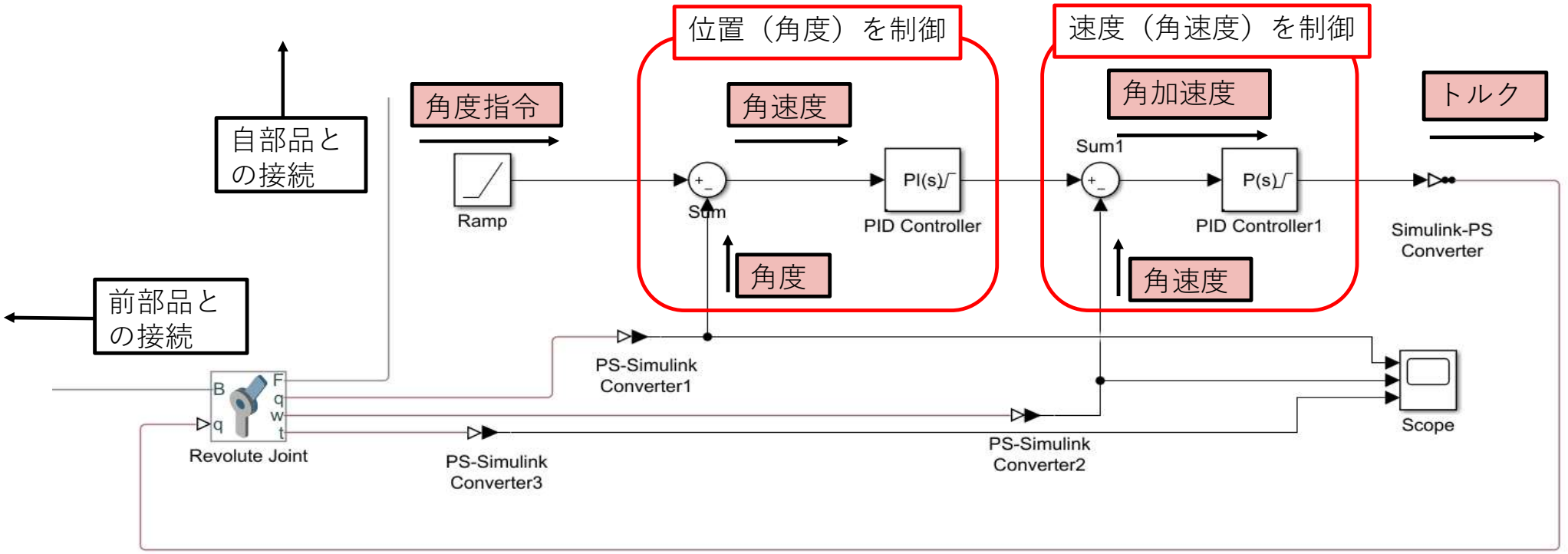
4.2 制御モデル　パラレルリンク逆キネマティクス

MATLAB言語でプログラムし、MATLAB Functionブロックで実装



MATLAB Functionブロック

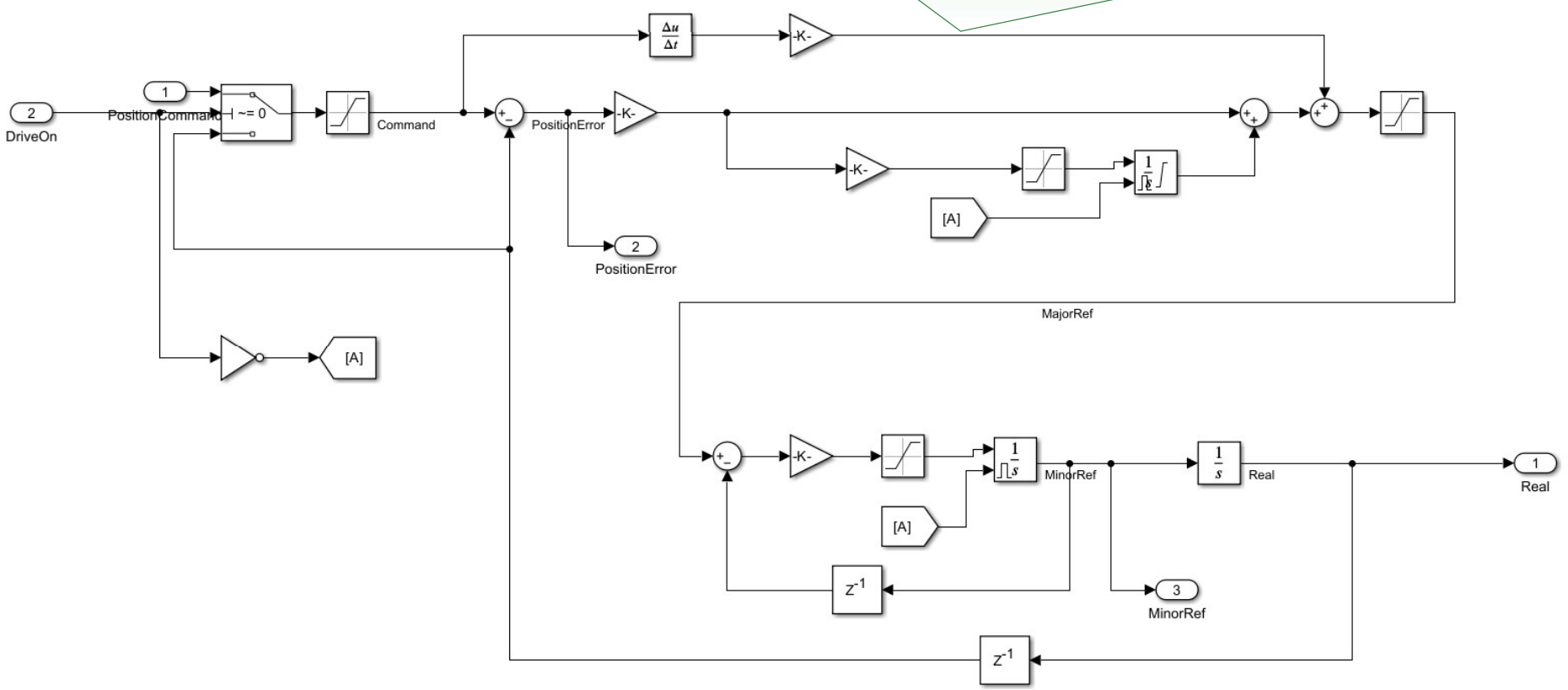
4.3 制御モデル　モーター制御



モーター制御概念ブロック図

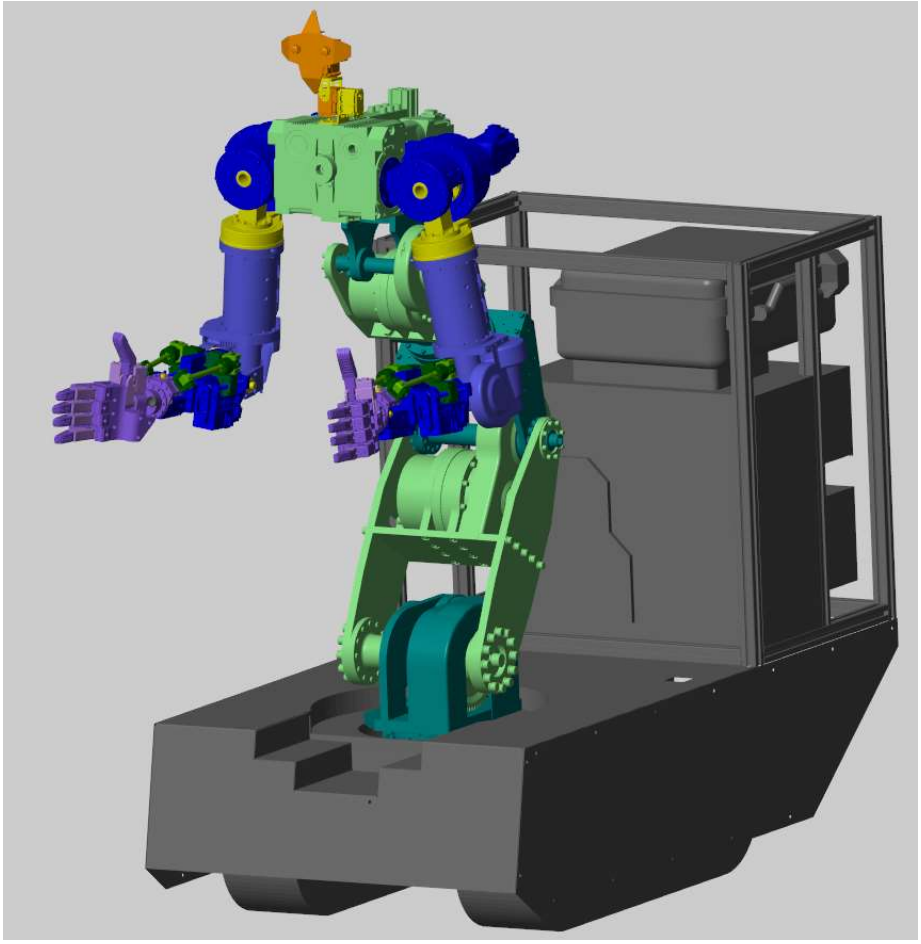
4.3 制御モデル　モーター制御

Simulink定義済みブロックで実装



モーター制御Simulinkブロック図

5 プラントモデル



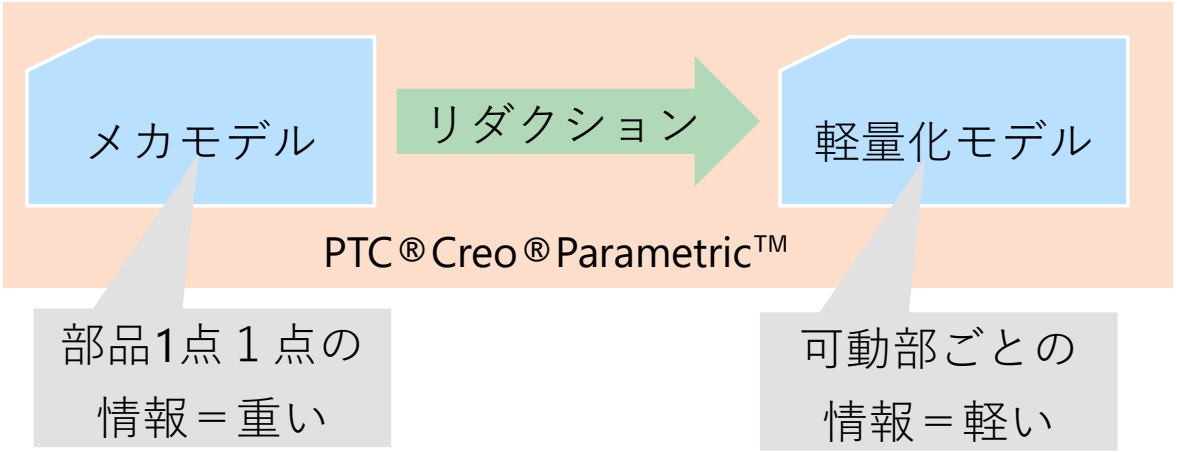
プラントモデル

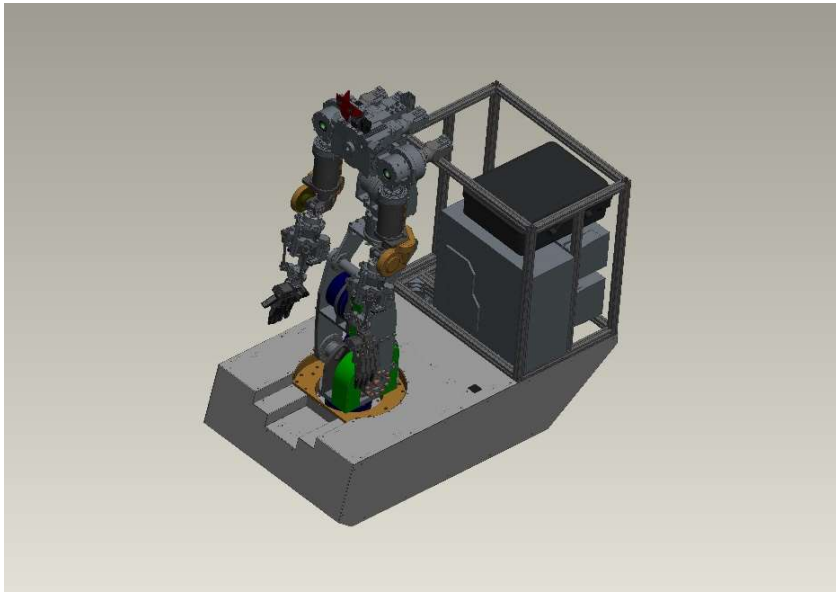
プラントモデルの制作過程を順を追って
説明する。

- 5.1 CADモデルのリダクション(軽量化)
- 5.2 CADモデルのインポート
- 5.3 部品の整列と相対位置の算出
- 5.4 部品の接続
- 5.5 接続点にジョイントを挿入
- 5.6 ジョイントへ位置指令を接続

モデル構築完了

5.1 プラントモデル CADモデルのリダクション(軽量化)





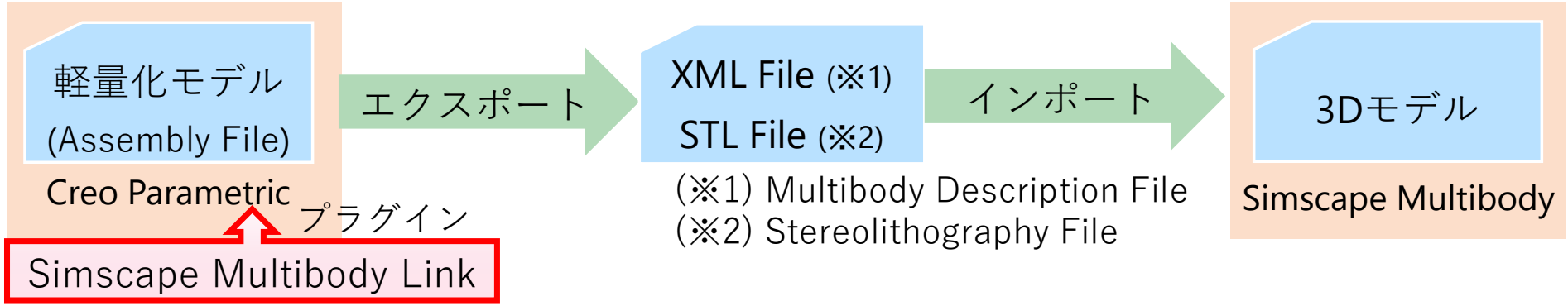
リダクション済みCADモデル

シミュレーション上での処理時間低減のためデータの軽量化（例えば、前腕を構成する複数の部品を「前腕」という一つの部品として定義しなおすこと）が求められる。

これをリダクションという。

三菱電機(株)電子通信システム製作所、および、三菱電機エンジニアリング(株)よりリダクション済みのCADデータを提供頂いた。

5.2 プラントモデル CADモデルのインポート



「Simscape Multibody Link」を使用して、 CADデータ（Assemblyファイル)をSimscape向けデータに変換し、エクスポートする。

ここで、Simscape向けデータとは、Simscape Multibodyが読み込み可能なファイル形式XML(Multibody Description)ファイルとSTL(Stereolithography)ファイルを指す。

Simscape向けデータを、インポートすることにより、Simscape Multibody上で3Dモデルを扱うことが可能となる。

5.3 プラントモデル　部品の整列と相対位置の算出

3Dモデル

絶対座標上に部品が配置されている

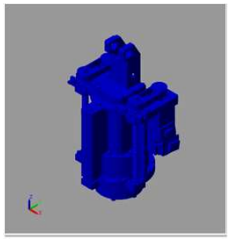
機構順に相対位置を求める

Transform Sensor (Simulink)

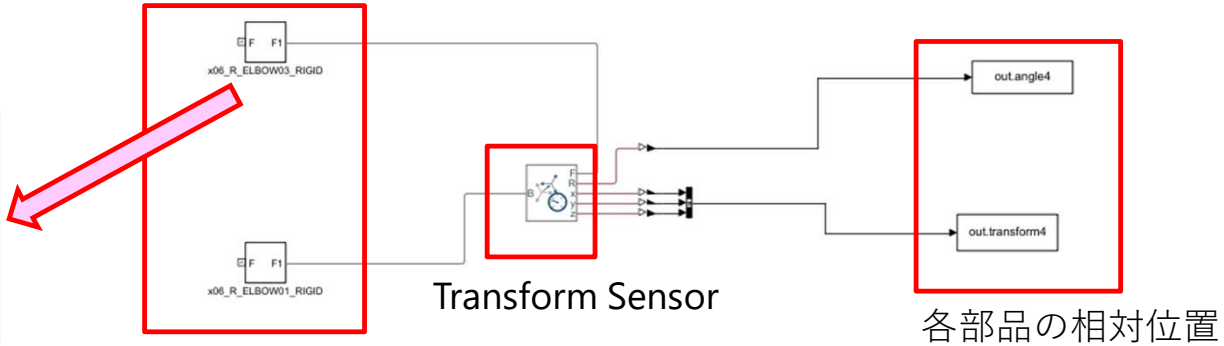
相対位置パラメータ

部品A(上腕等)の回転中心から隣接する部品B(前腕等)の回転中心への相対位置を求める。

- STL原点から回転中心への座標を求めておく必要がある。
- リダクション時に回転中心をSTL原点にしていれば作業は楽になる。



部品を可視化した画像例



各部品（前腕等）

各部品の相対位置

Transform Sensorによる部品間の相対位置を求めるブロック図の例

5.4 プラントモデル　部品の接続

3Dモデル
(絶対座標)

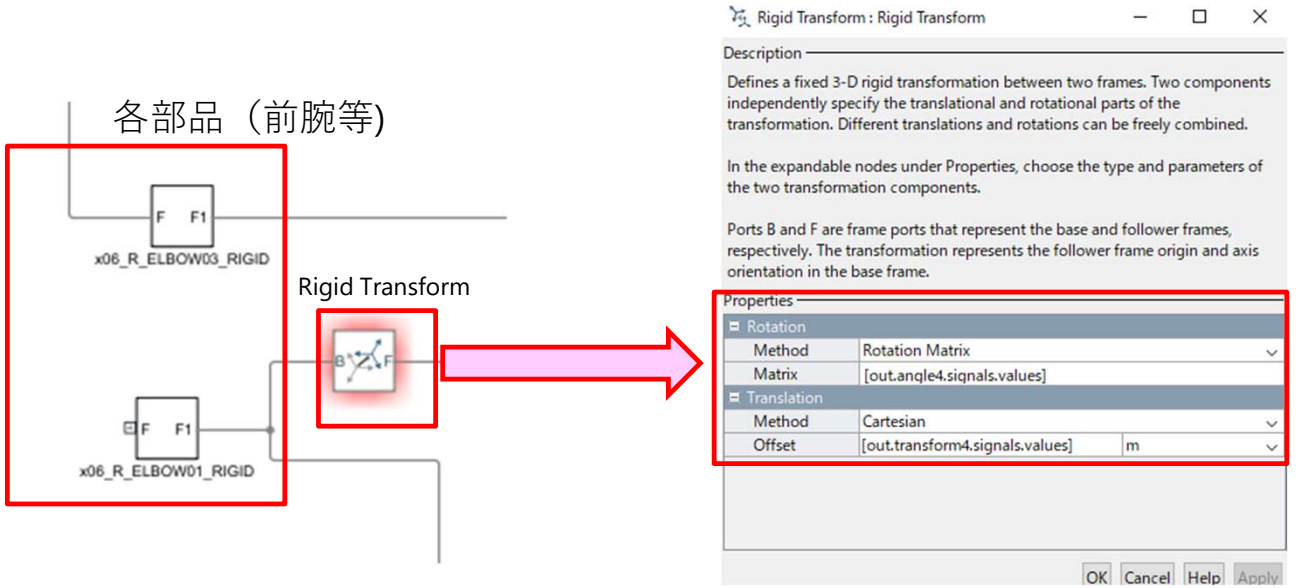
絶対座標上に部品が
配置されている

絶対座標配置をやめて
相対位置で部品同士を接続する
Rigid Transform
(Simulink)

相対位置パラメータ

3Dモデル
(相対座標)

機構順に部品が配置
されている



5.3で求めた部品間の相対位置を
Rigid Transformに定義する

Rigid Transformによる各部品の接続例

5.5 プラントモデル　接続点にジョイントを挿入

3Dモデル
(相対位置固定)

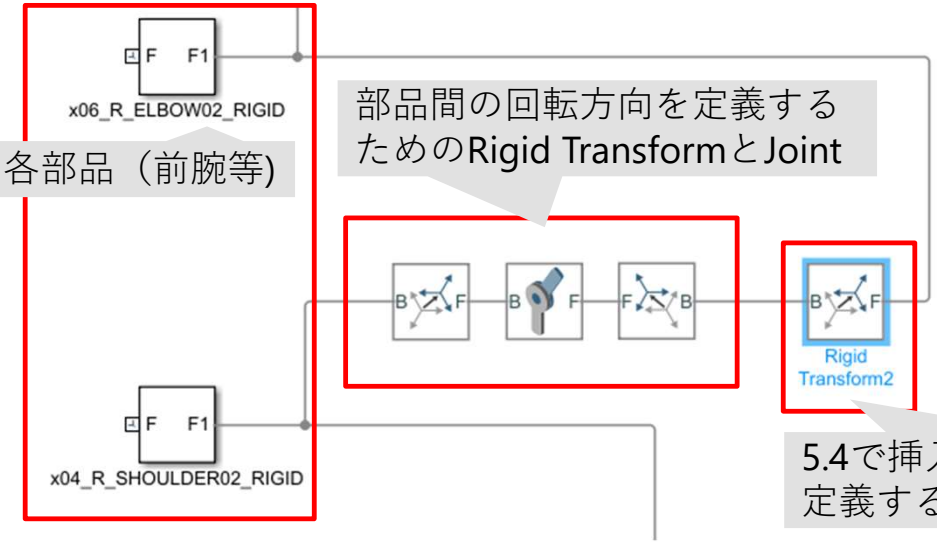
接続点にジョイントを挿入
Joint
(Simulink)

3Dモデル
(姿勢変化可能)

相対位置で部品が
固定されている

ジョイントの回転軸が合っていなければ、
両端をRigid Transform で挟む

関節の回転が可能



部品間へのジョイントの挿入例

5.6 プラントモデル　ジョイントへ位置指令を接続

3Dモデル
(姿勢変化可能)

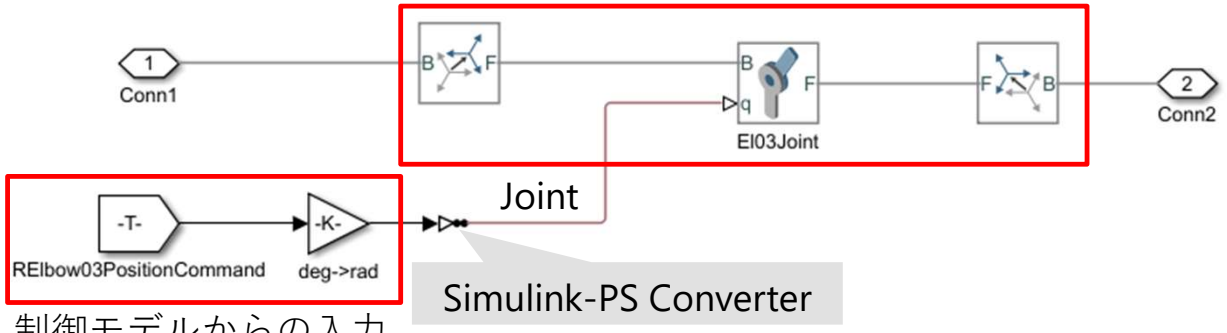
Jointにコマンドを接続
Joint
(Simulink)

3Dモデル
(駆動可能)

静的な姿勢変化が可能

位置指令による駆動が可能

Jointの回転角を外部からの入力と定義し、位置指令によってJointが回転するように指令角度と接続する。



制御モデルからの入力

ジョイントへの位置指令入力例

6 MILS

コマンド入力モデルー制御モデループラントモデルを接続した後、
MILS(Model In the Loop Simulation)環境を構築し、動作確認を行った。
MILSにて動作確認中に気付いた点と工夫した点について述べる。

6.1　特異点の回避

6.2　操縦者の感覚に合わせる工夫

6.3　接触の実現

6.4　駆動順序による衝突回避の検証

6.1 MILS 特異点の回避

逆キネマティクスにおいて、AZ回転軸が一致してしまい、不定解となる（解の個数が無限になる）問題がある。
不定解となるケースは以下の通り、

ケース1： AZ2とAZ3が不定
θEL3=0 の時

EL1の駆動範囲を限定

ケース2： AZ1とAZ2が不定

$\theta EL1 = -\sin^{-1}\frac{b}{a}$ かつ θEL2=-θEL1 の時

EL1の駆動範囲を限定

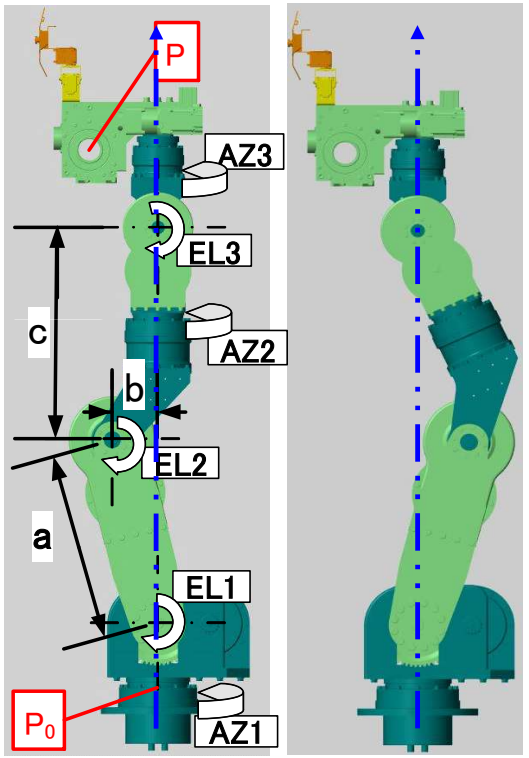
ケース3： AZ1とAZ3が不定

AZ3を固定しAZ1のみ駆動

$$\theta EL2 = -\left\{\sin^{-1}\left(\frac{a\sin\theta EL1}{\sqrt{b^2+c^2}}\right) + \tan^{-1}\frac{b}{c} + \theta EL1\right\}$$

かつ θEL3=θEL1+θEL2 かつ θAZ2=0° の時

※ a,b,cはメカ寸法であり、a=145, b=100, c=280+185

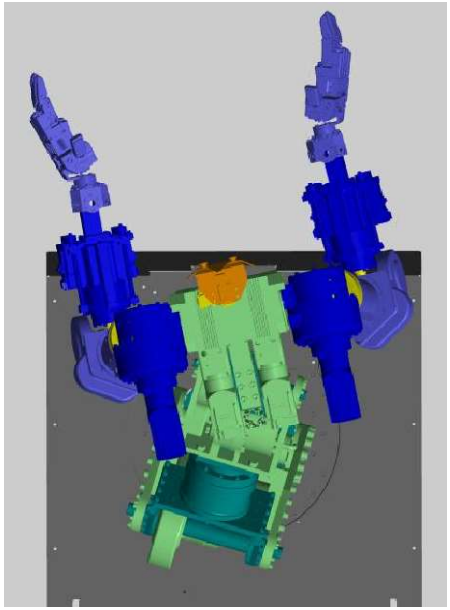


ケース1及び
ケース2の姿勢

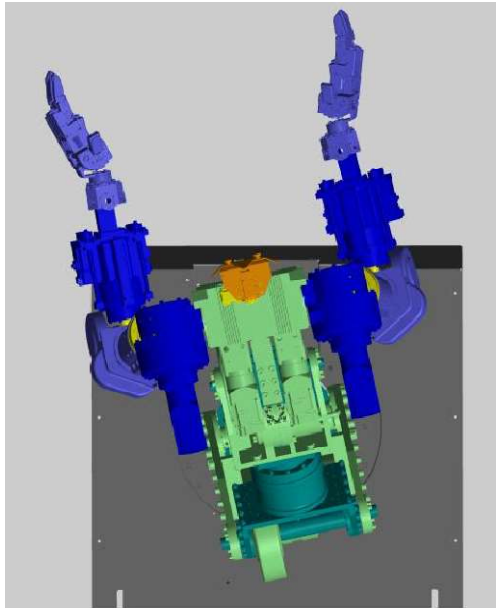
ケース3の姿勢

6.2 MILS 操縦者の感覚に合わせる工夫

「12度左に向く」と指令した場合、逆キネマティクスは胴体を捻る動作をする。
足元から回転する方が操縦者の意図に合っている。

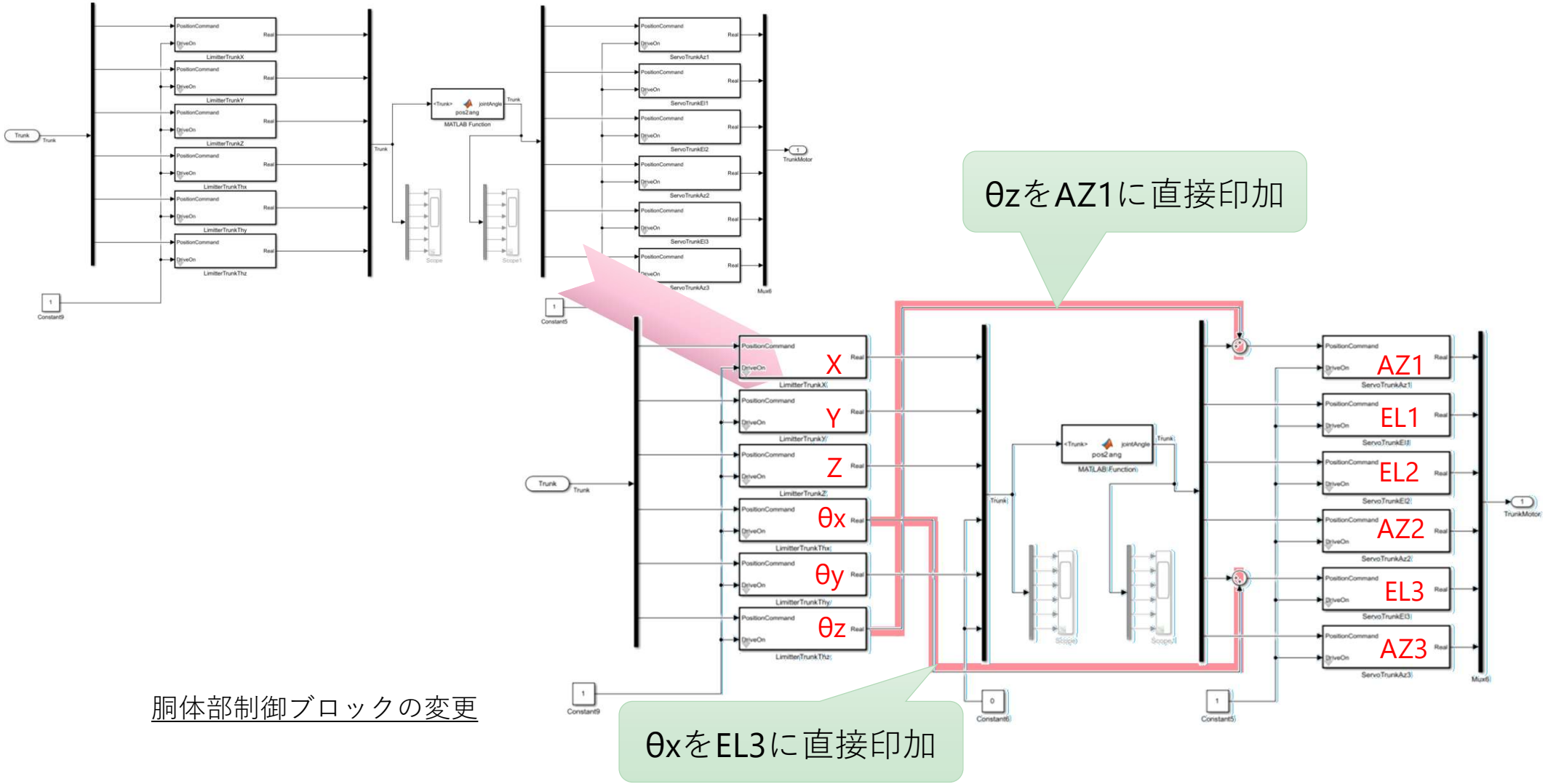


逆キネマティクス
(胸の位置を変えずに左回転)



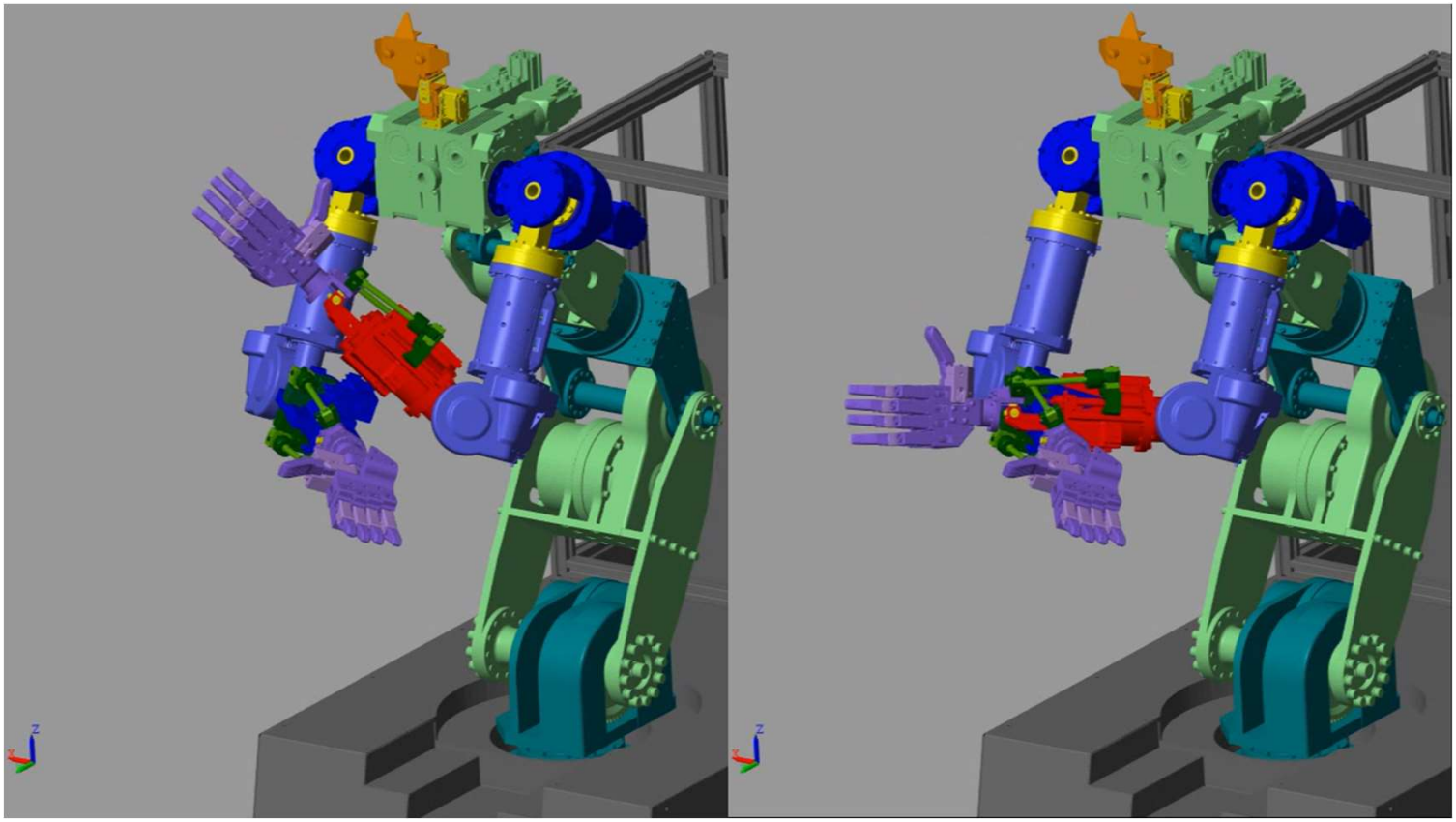
修正版
(足元を軸に左回転)

6.2 MILS 操縦者の感覚に合わせる工夫



胴体部制御ブロックの変更

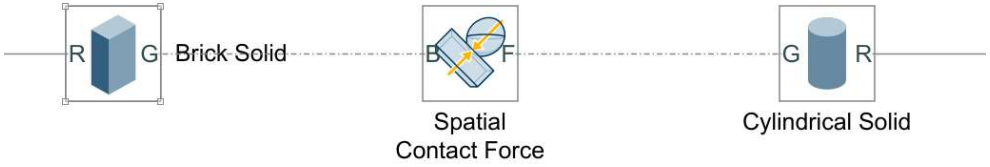
6.3 MILS 接触の実現



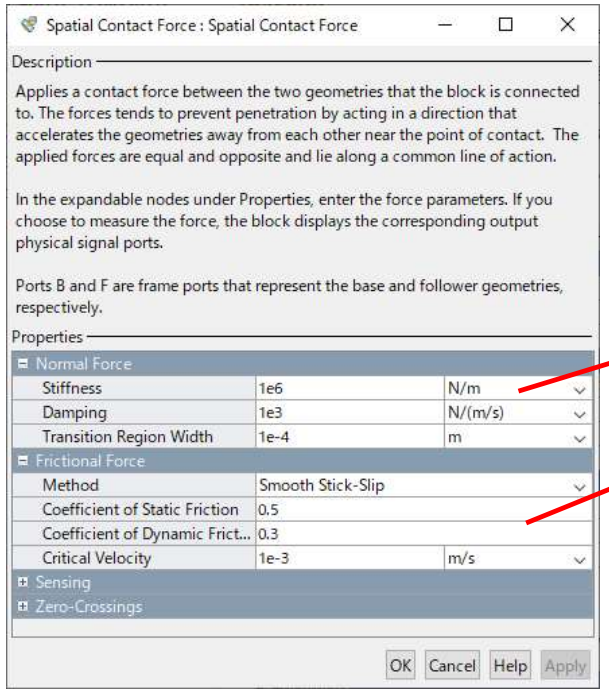
接触の有無

6.3 MILS 接触の実現

Simulinkにおいて物体同士の接触させるためには、Spatial Contact Forceブロックを使用する。
このブロックに接続された物体は、接触時に大きさが等しく逆向きの反力を受ける。



接触判定のブロック図



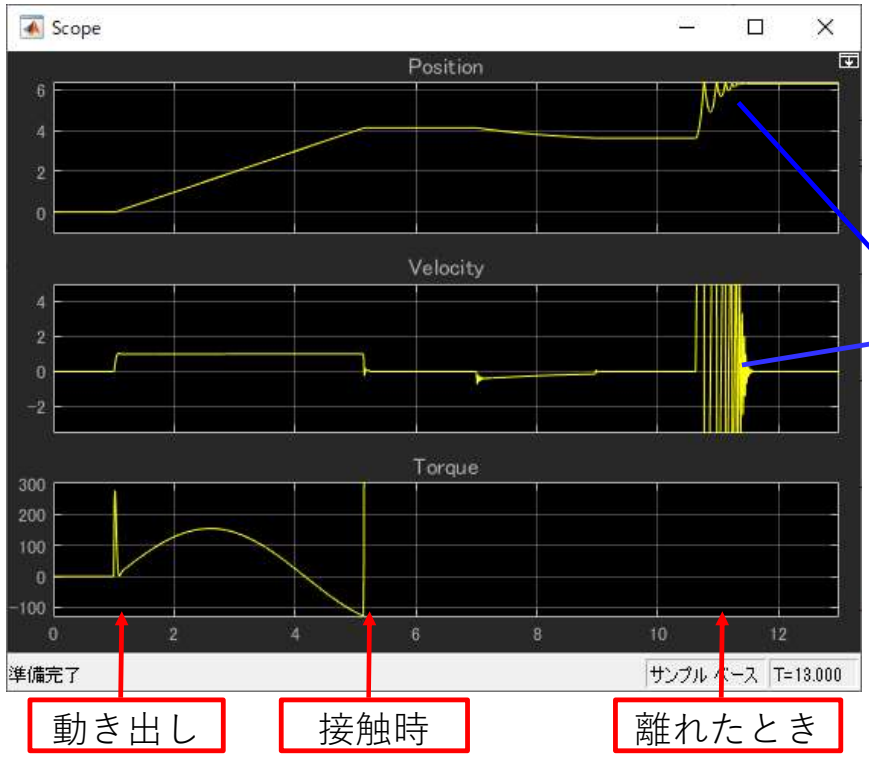
垂直抗力の設定

摩擦力の設定

Spatial Contact Forceブロックの設定

6.3 MILS 接触の実現

接触シミュレーション時の各測定値を示す。



位置、速度、トルク測定値

6.4 MILS 駆動順序による衝突回避の検証

- 一例として、人差し指を何度以上曲げていると、親指と衝突するかなど、機械的な衝突をMILSで検証した。
- 衝突回避のためプログラムで制限を加えた場合も、所望動作の確認だけでなく、望まない副作用がないかどうかも、目視で検証できることを確認した。

7 自動コード生成

制御モデルからSimulink Coder＋Embedded Coderを用いて自動コードを生成した。

その手順と、評価について述べる

- 7.1 離散化
- 7.2 手書きコードとのリンク

→ 手順

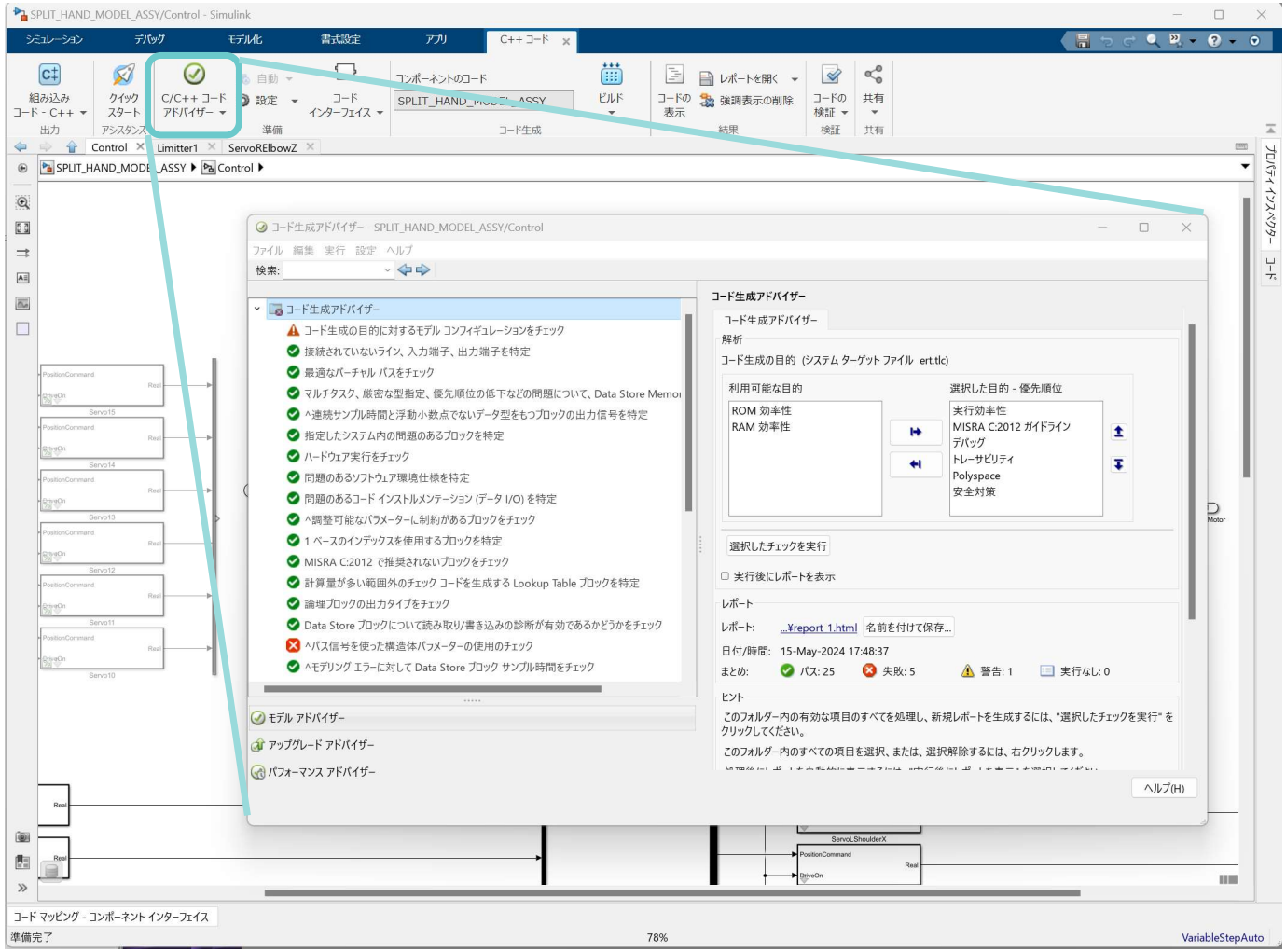
- 7.3 実機動作検証
- 7.4 処理時間比較
- 7.4 評価まとめ

→ 評価

7.1 自動コード生成　離散化

- Simulinkで書かれたモデルの時間系は連続系と離散系の2種類がある。
- シミュレーションモデルはトリガーを必要としない連続系の方が作りやすいが、コード化するモデルは離散系に変更（離散化）しておく必要がある。
- 注意点として、ライブラリーブロックによっては、“Continuous”グループや“Simscape”グループなど連続系でしか動作しないものもあり、再設計が必要になる場合がある。
- MATLABの自動コード生成機能を用いて、検証した制御モデルから実行コードを作成した。実行コード作成には、コード生成を支援する“コード生成アドバイザー”を利用した。
- “コード生成アドバイザー”により、コード化のための手順が提示されるため、容易にコードを生成することができた。

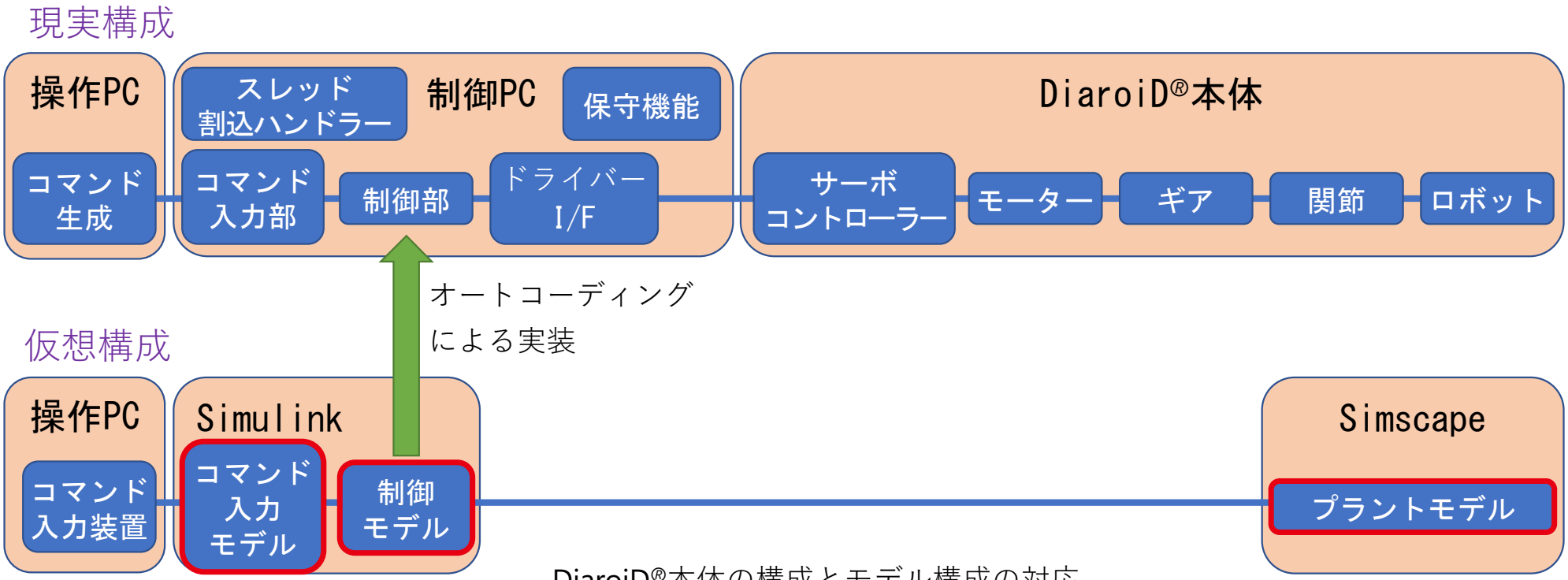
7.1 自動コード生成　離散化



コード生成アドバイザー

7.2 自動コード生成　手書きコードとの結合

- 制御部は制御モデルの自動生成コードを使用した。（比較対象として手書きコードも作成）
- その他の部位は手書きコードを使用した。



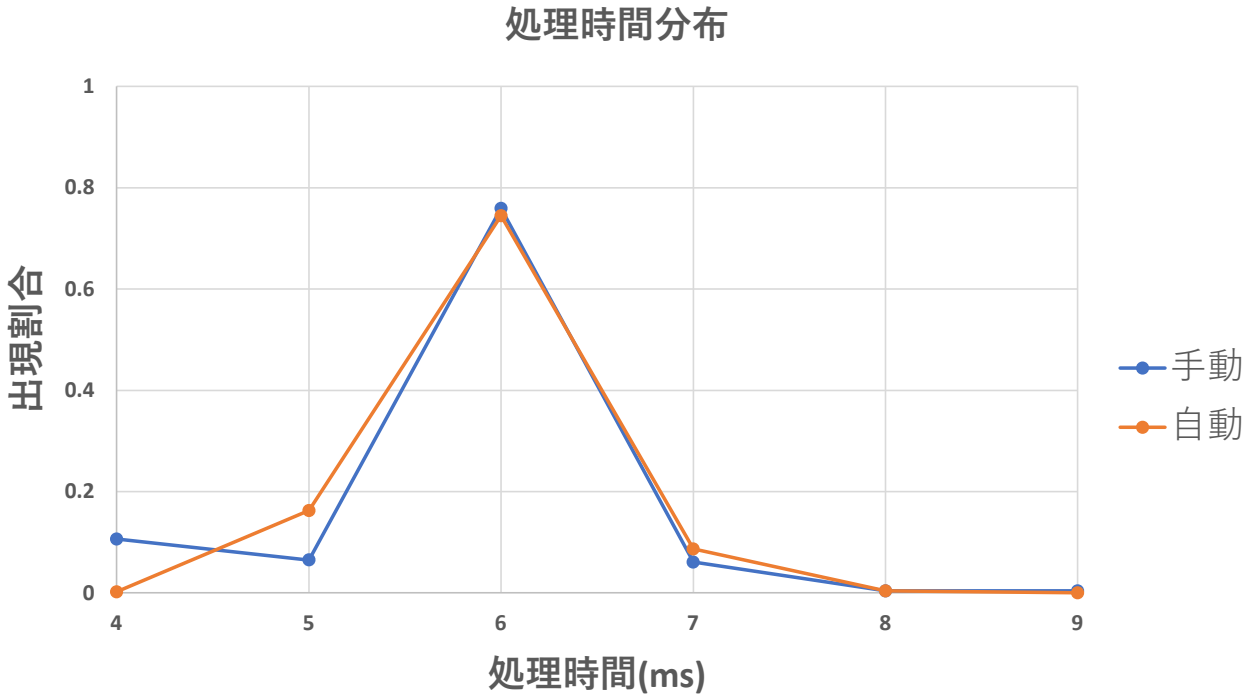
DiaroiD®本体の構成とモデル構成の対応

7.3 自動コード生成　実機動作検証

- ビルドしたプログラムをDiaroiD®に搭載し動作を確認した。
- ここで特筆するべきは、動作確認開始時から問題なく動作したことである。
  MILSで検証したモデルは、コードレビューを必要とせず、コーディングミスを混入する余地もなく、実機で動作した。自動コード生成の優位性を証明する結果であった。

7.4 自動コード生成　処理時間比較

- 制御部の処理時間には動作の遷移により4～8msのばらつきがある。
- 同一の動作をさせたときの手書きコードの平均処理時間は5.80ms、自動生成コードの平均処理時間は5.93msであった。



周期処理の処理時間分布

- 左図は手書きコードと自動生成コードの処理時間のばらつきを、横軸：処理時間、縦軸：出現割合で示したグラフである。
- 処理時間のばらつきも含めて自動生成コードの処理効率は、手書きコードと遜色のないレベルにあることが分かる。

7.5 自動コード生成　評価まとめ

従来手法によるコード（手動）とモデルベース開発による自動生成したコードの比較を以下に示す。

比較項目	従来手法	モデルベース開発	補足
ライン数	３４１１ Line	２４７７ Line	自動生成コードの方が冗長、読み辛い。 一方、人は拡張性やデバッグの容易性等を考慮してしまうことよりライン数は多くなる。
性能 （※１）	ほぼ　差　無し		同じ条件（ロボットに同じ動作をさせる）で計測。平均処理時間、処理時間のばらつきともに、ほぼ差は無かった。
不具合混入	16件	0件	コーディングミス基準値：6.5件／KL 単試の不具合検出基準値：2.5件／KL

（※１）今回は、性能を重視して、自動生成のパラメータをチューニング

8 リアルタイムシミュレーション

MILSとして使えるようになった次は、ロボットの動作と同期させ、操縦支援を行うことを目指した。

そのための、問題解決ならびに追加機能について述べる。

- 8.1 演算負荷軽減
- 8.2 時間調整

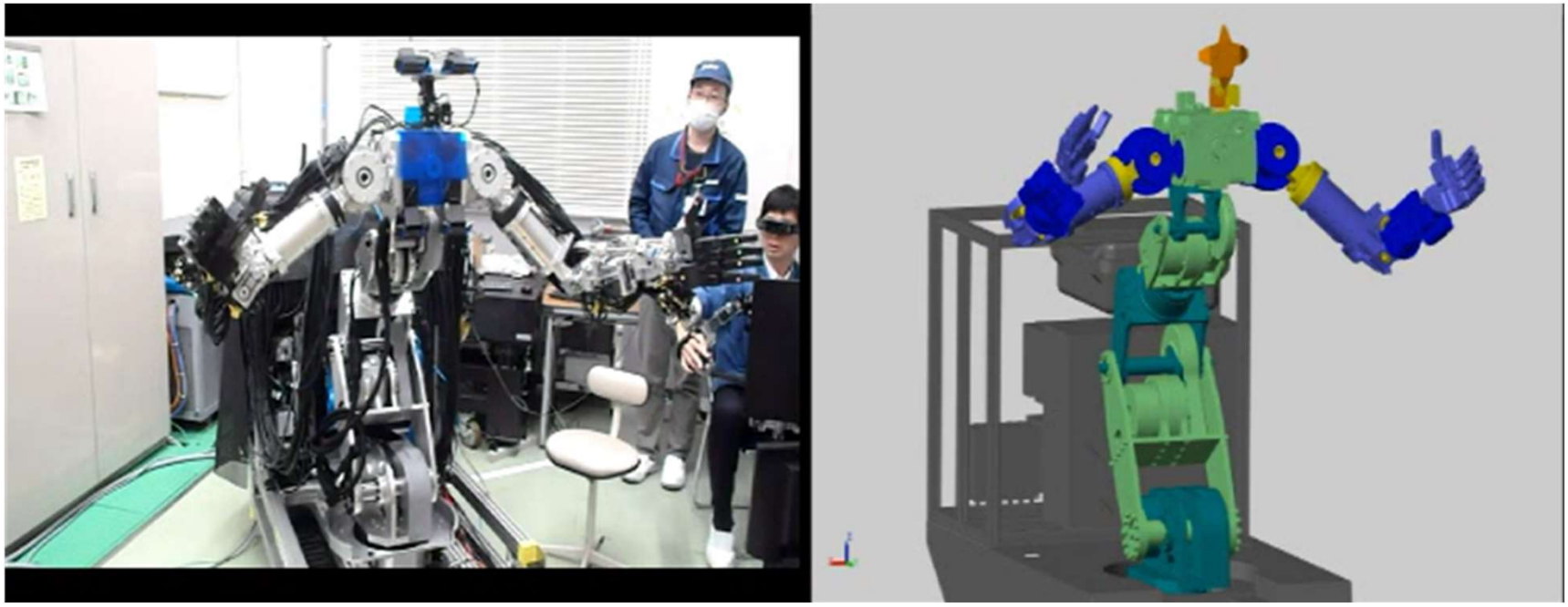
→ 問題解決

- 8.3 LiDAR投影
- 8.4 仮想視点
- 8.5 双腕衝突警告

→ 追加機能

8.2 リアルタイムシミュレーション

ロボットとモデルがシンクロ



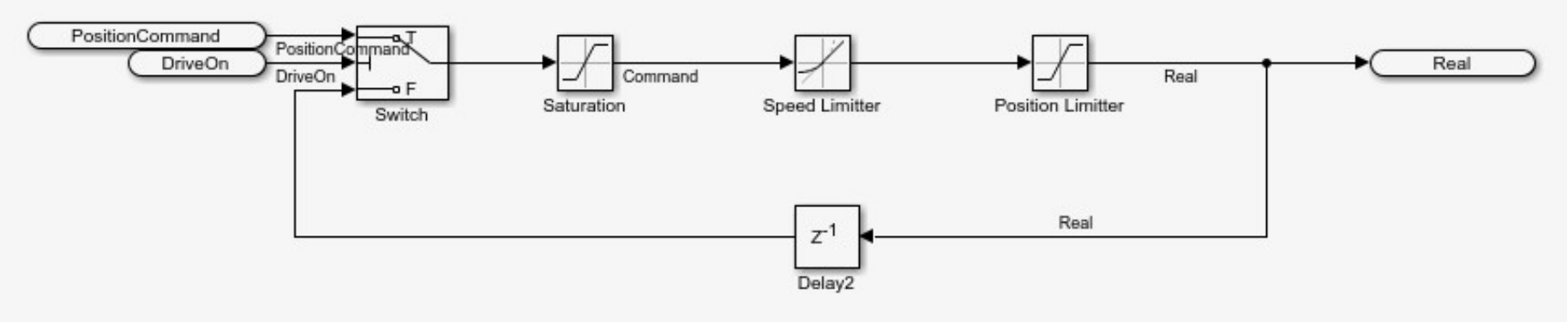
ロボットとプラントモデルの同時動作

8.1 リアルタイムシミュレーション　演算負荷軽減

・1秒間分のシミュレートを1秒以内に演算しなければならない。
以下の処置にて演算負荷を軽減した。

① パラレルリンク機構の簡略化
　手首の駆動にパラレルリンクを使用せず、手首角度をそのまま手首のジョイントに印加した。

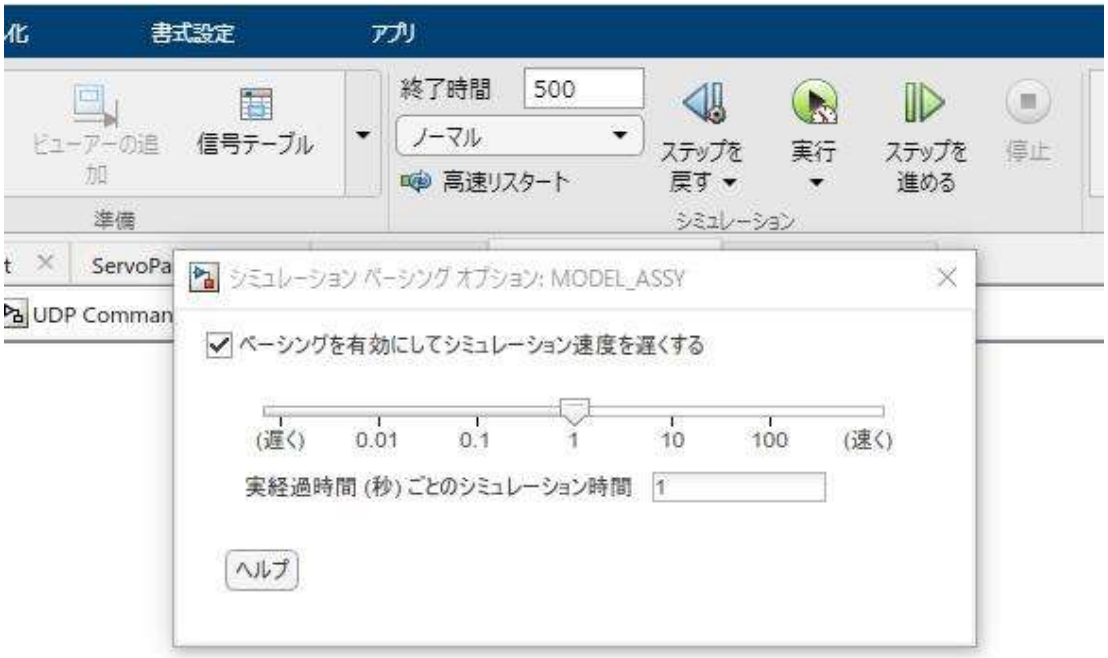
② モーター制御を簡略化した。



簡略版モーター制御ブロック図

8.2 リアルタイムシミュレーション　時間調整

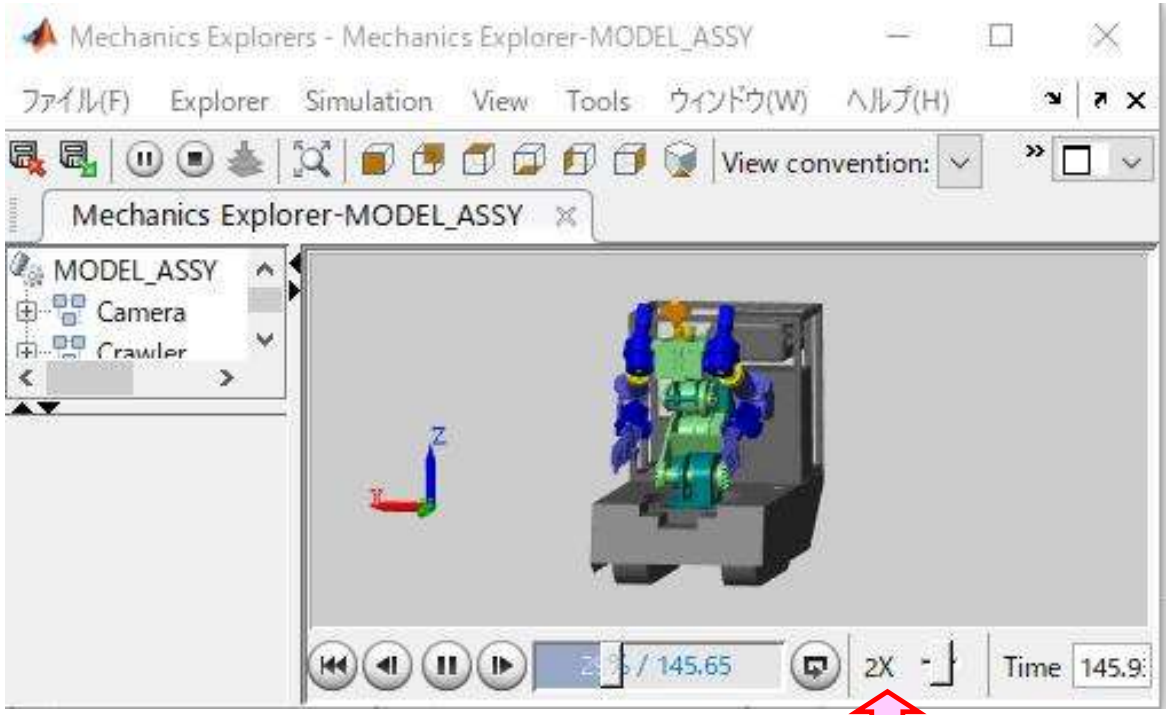
・シミュレーションページングオプションを使用する。
「ページングを有効にして、シミュレーション速度を遅くする。」を選択。
実経過時間1秒でシミュレーション時間も1秒経過するように設定する。



ページングオプション

8.2 リアルタイムシミュレーション　時間調整

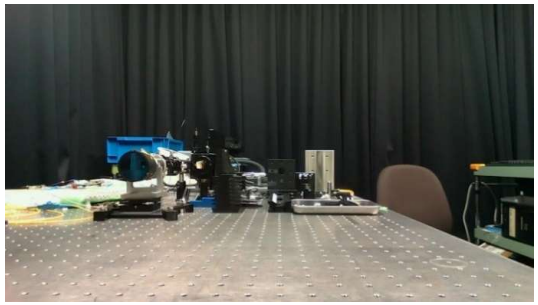
・再生速度を2倍にする。
　シミュレーション起動時に遅れが生じ、再生速度1倍では遅れが解消されることなく、固定時間遅れて表示される。
　再生速度を早くして遅れた分を追いつくようにする。



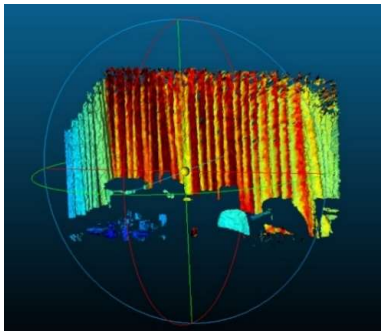
再生速度調整方法

8.3 リアルタイムシミュレーション　LiDAR投影

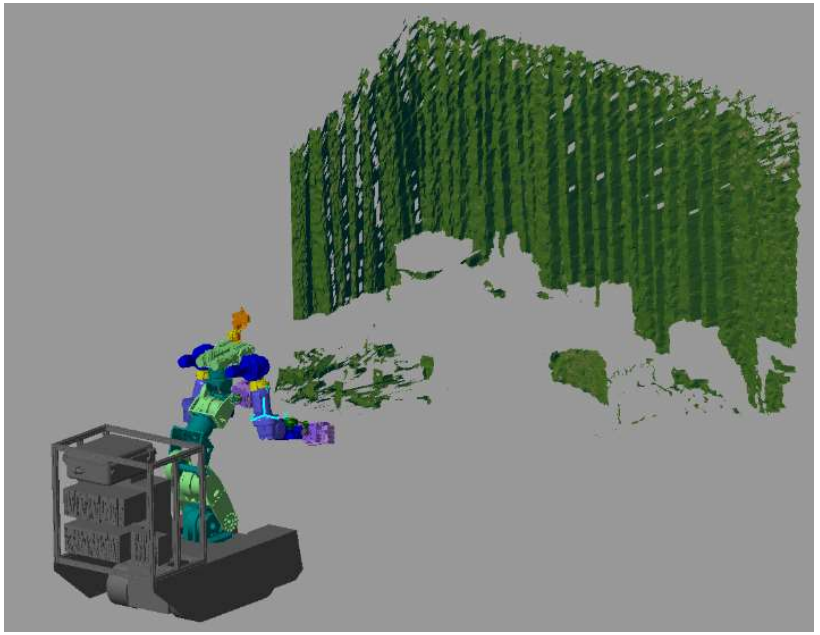
- ロボットの周辺環境をLiDARで取り込み、シミュレーション画面上に再現した。
- 点群データ(ply)をCloudCompare®でSTLファイルに変換し、Simscapeに読み込んだ。
  - ✓ 今では、Lidar Toolbox™により、点群データを扱えるとのこと。



対象の光学写真



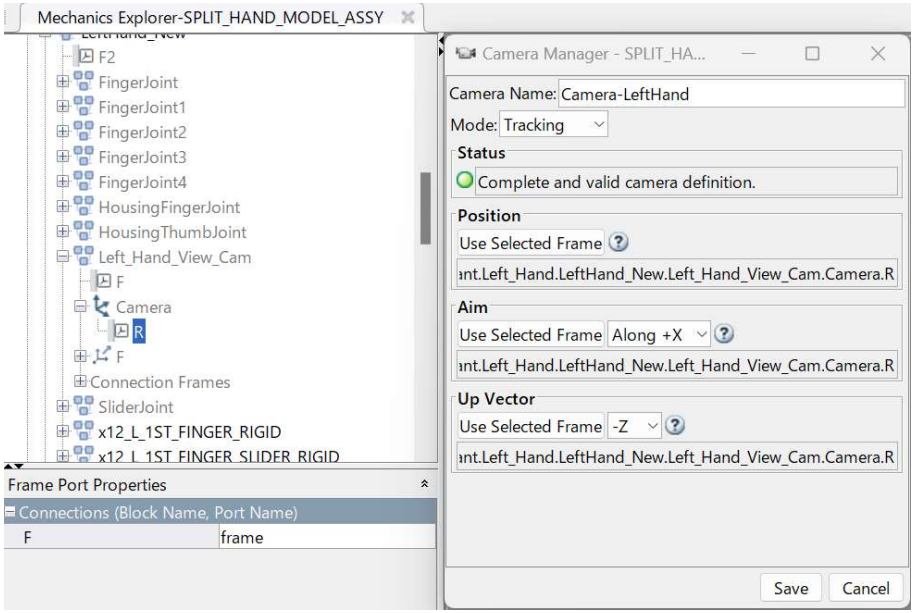
点群データの可視化



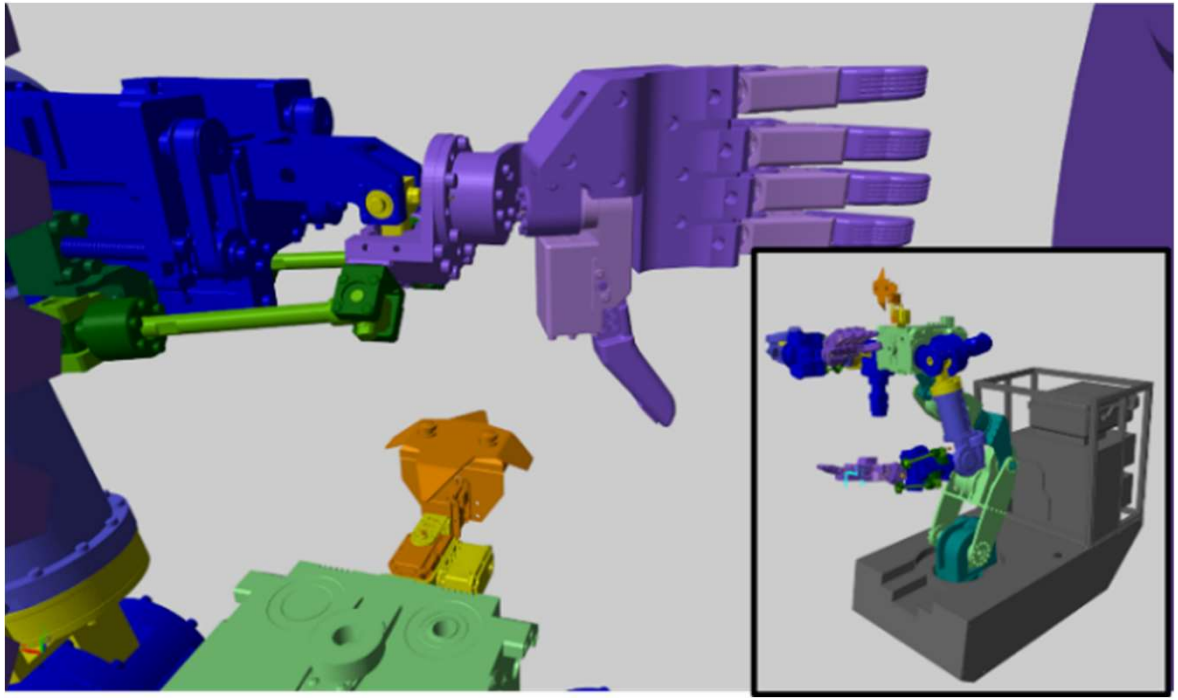
プラントモデルに投影した点群

8.4 リアルタイムシミュレーション　仮想視点

- Camera Managerに任意のFrameを指定して視点を追加することができる。
  - ➢操縦者が見えない視点での動きを確認することができる。

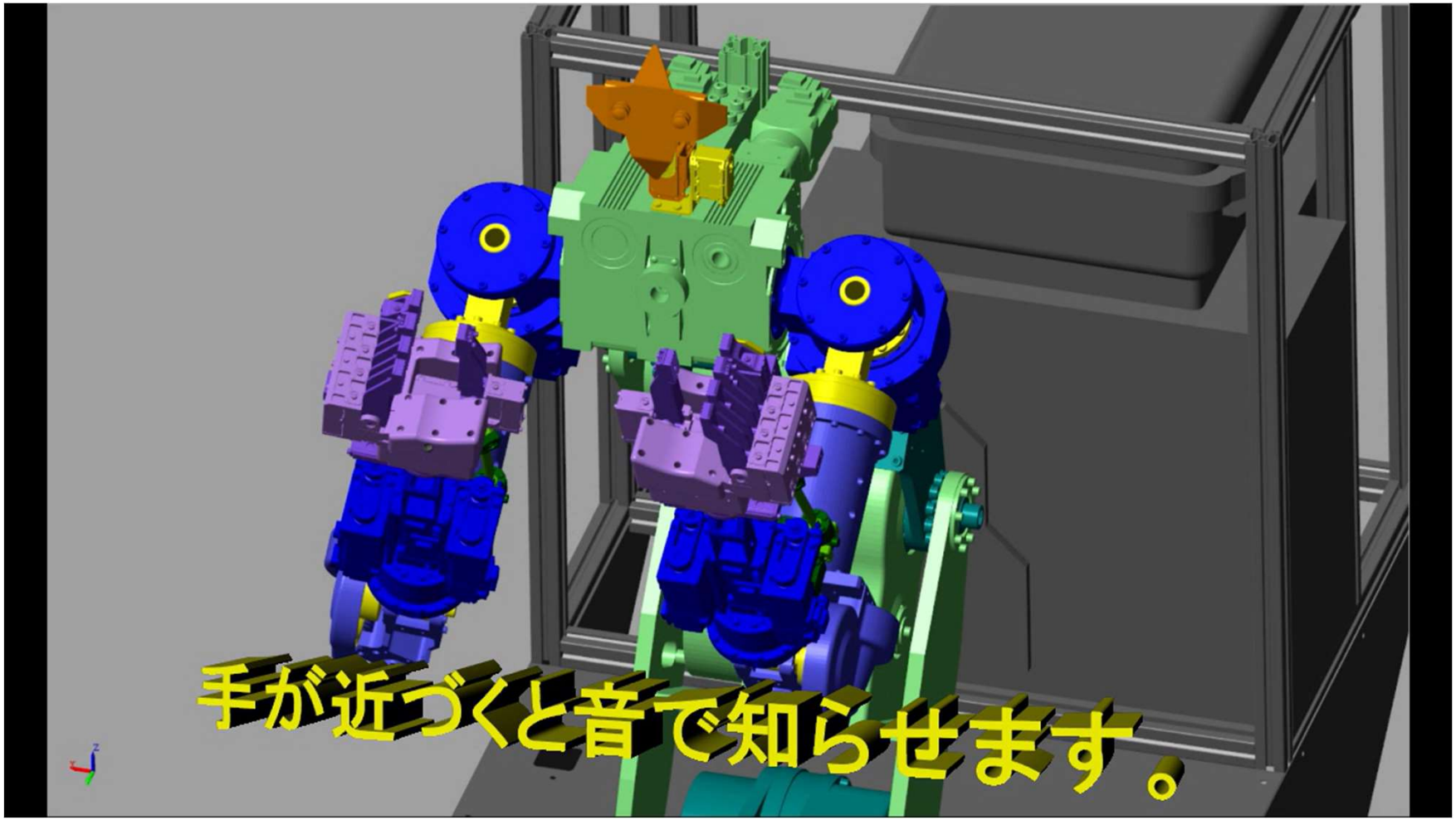


Camera Managerのプロパティ



左手に追加した仮想視点

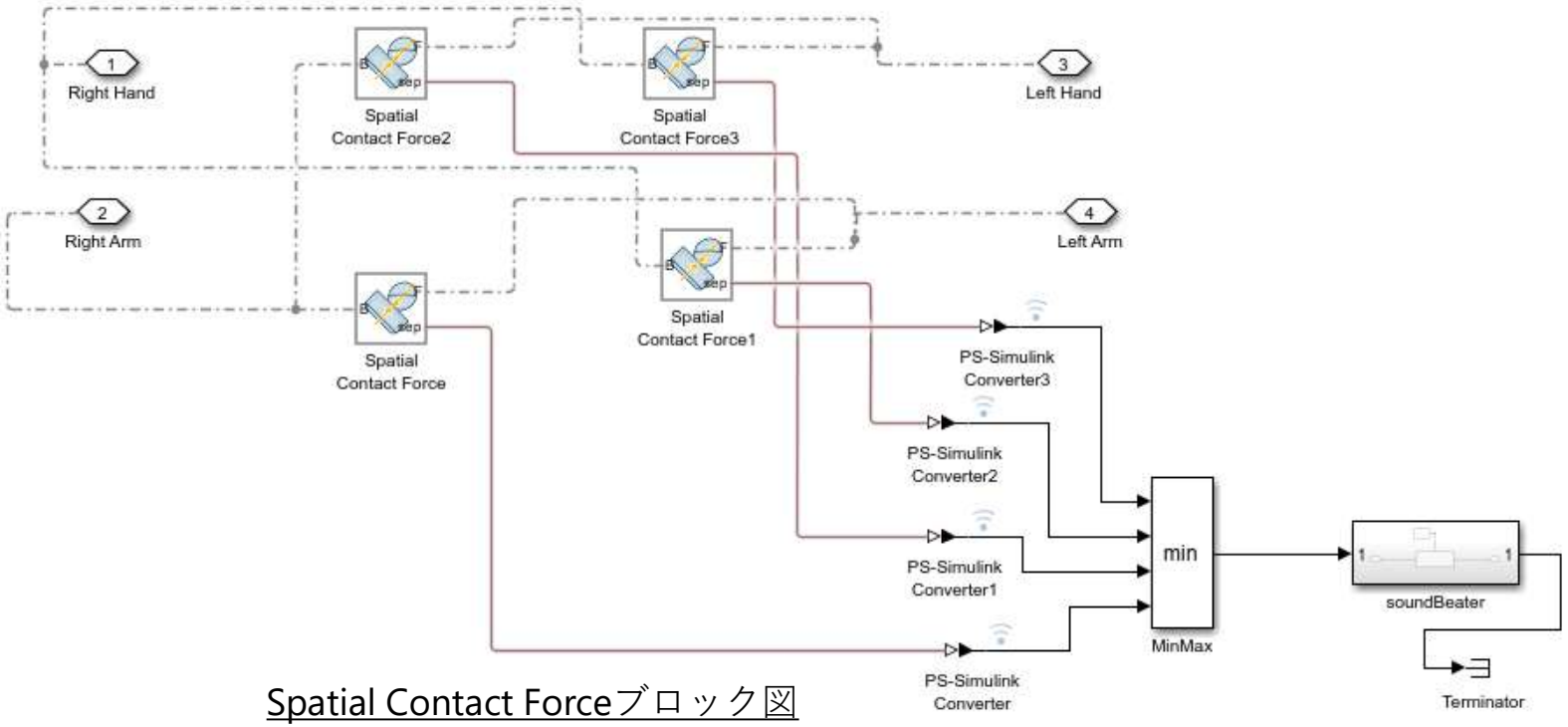
8.4 リアルタイムシミュレーション　双腕衝突警告



音で知らせる双腕衝突警告

8.5 リアルタイムシミュレーション　双腕衝突警告

- 接触の実現と同じく、 Spatial Contact Forceを使用した。
- 接近距離に応じて、警告音を発生させた。



Spatial Contact Forceブロック図

9 MBD導入に際して

これからMBD開発を導入する方に向けて、開発工数や長所／短所を提示したうえで、導入時に検討しておくべき事柄を述べる。

9.1 開発工数

9.2 長所と短所

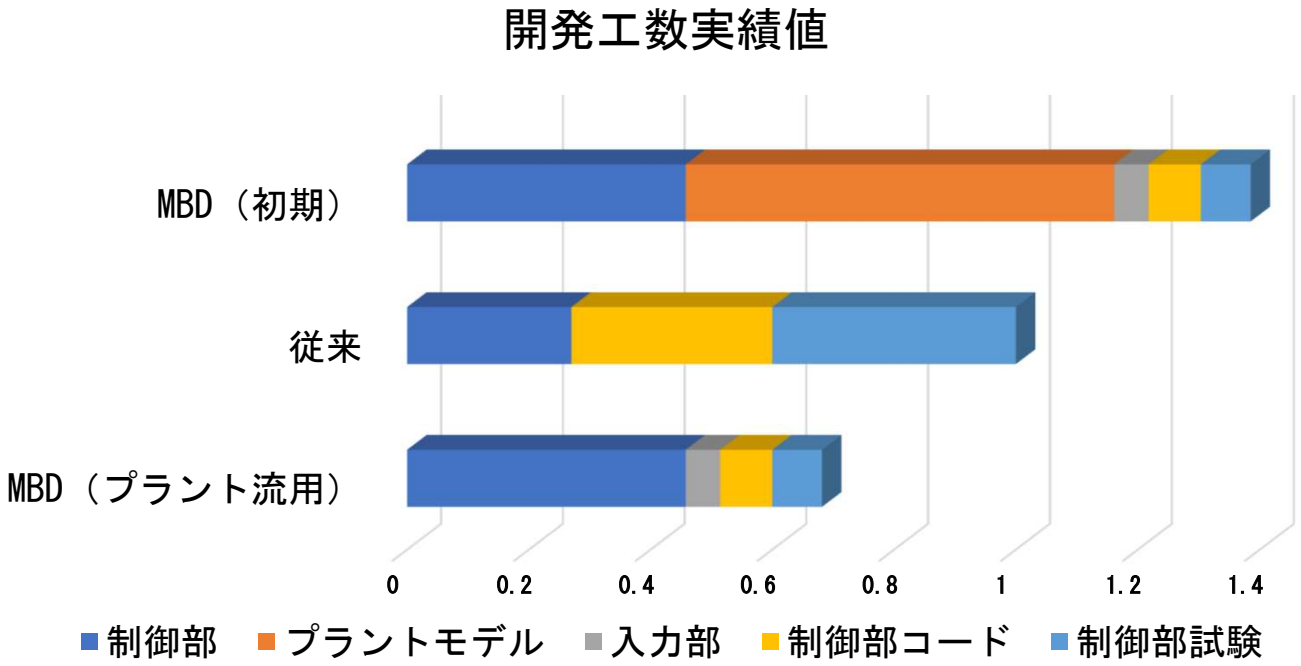
9.3 検討事項

9.1 MBD導入に際して　開発工数

従来手法の開発工数を“１”として、モデルベース開発の開発工数を評価。

プラントモデルの構築に多くの工数が必要になるため、

- モデルベース開発は従来手法より工数増となる結果が得られた。（今回は約4割（38.5%）増）
- 制御対象（＝プラントモデル）を流用する場合は、従来手法より工数減となる。（今回は約４割（37.5%）減。



9.2 MBD導入に際して　長所と短所

MBD開発の長所と短所

長所	短所
明確な仕様伝達が可能である。	ツール操作の習得に手間と時間がかかる。
MILSによる早期の検証ができる。	プラントモデル作成に手間と時間がかかる。
自動コード生成ができる。	従来の品質管理指標が使えない。
モデルが資産になる。	
技術継承に有利である。	

MATLAB/Simulinkの得意分野

得意	不得意
演算処理	ユーザーインターフェース
手続き処理	外部インターフェース
	スレッディング

9.3 MBD導入に際して　検討事項

MBD開発導入時の検討事項

検討事項	解説
制御対象を理解していること	プラントモデルを作成するには、制御対象の構造を理解し振る舞いを定義する見識が必要である。
プラントモデルを流用できること	プラントモデルを一度しか使わなければ、損失になる。パラメーターを変えながら何度も検証を行うような開発や、よく似たハードウエアを繰り返し使用するプロダクトラインに適用するのが良い。
初期費用が確保できること	導入時は技術習得コスト及びプラントモデル作成コストのため費用が増大する。初期費用が確保できない状況で実施すると、有意義なプラントモデルを製作できなくなり、後の改善効果も見込めない事態が予想される。 予算の範囲で、少しずつ適用部位を広げていく方法が適切と考える。
適用部位を広げすぎないこと	ユーザーインターフェース、外部インターフェースとスレッディングなど不得意な分野は従来手法で、演算機能、手続き処理など得意とする分野はモデルベース開発手法で製作し、マージする処置が必要である。

ご清聴ありがとうございました。