

Sim4stamp 取扱い説明書

第 1 版 update 1

2019 年 10 月

by
Keiichi Tsumuta

ドキュメント履歴

Update1 : 2019 年 10 月、 5 値論理の機能追加に伴う記述の追加を行う。

目次

1. はじめに	3
2. 動作環境	4
3. シミュレーション手順	5
4. インストール、実行環境整備	6
4.1. Sim4stamp インストール	6
4.2. Overture 環境の設定	6
5. Sim4stamp の起動	7
6. プロジェクト設定	8
7. モデル生成	10
8. VDM++モデル生成	16
9. 初期値データ設定	18
10. 偏差投入設定	19
11. VDM++の実行	21
11.1. 実行方法その 1	21
11.2. 実行方法その 2	21
11.3. ステップ実行	22
12. 実行ログの確認	24
13. シミュレーション実行結果表示	25
13.1. グラフ形式データ表示	25
13.2. 表形式データ表示	29
13.3. シミュレーション結果の累積、クリア	30
14. 実行結果の外部出力	31
14.1. レポート出力	31
14.2. UCA ヒント出力	32
15. Q&A	33
付録・1 VDM++自動生成例	34
付録・2 5 値論理について	35
付録・2 5 値論理の演算パターン	36

1. はじめに

Sim4stamp は、STAMP/STPA 解析を行うときの思考を補佐する簡易シミュレーションツールとして作成されたものです。

Overture を用いて、VDM++を使用してシミュレーションを作成し、実行することが出来ます。

VDM++は簡易シミュレーションを作成するには適切な手法ですが、データの設定や出力については貧弱な方法しかありません。Sim4stamp は、VDM++の I/O に関する機能を補うために、データの設定、出力、モデリングを、GUI を用いて行うことが出来るようにしたものです。ただし、機能的には STAMP/STPA 用に特化したものとなっています。

2. 動作環境

Sim4stamp の動作環境を以下に示します。

- (1) Overture VDM++モデリング統合ツール
- (2) Java 8 以降の Java ランタイム環境
- (3) Java と Overture が動作する 32Bit または 64Bit OS

Java の実行環境は、内部的な作りの関係上、Java 8 以降が必要になります。

コマンドプロンプトやコンソール上で「`java -version`」をタイプして「Enter」を打つと Java のバージョンが表示されますので、そのバージョンが「1.8.0」以上となっていることを確認します。

ただし、JDK11 以降は未対応です。

(例)

```
java -version
java version "1.8.0_141"
Java(TM) SE Runtime Environment (build 1.8.0_141-b15)
Java HotSpot(TM) 64-Bit Server VM (build 25.141-b15, mixed mode)
```

もし、このコマンドがエラーになる場合は、Java のランタイムがシステムにインストールされていないと思われますので、その場合は、以下の URL よりダウンロードし、Java をインストールしてください。

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

Overture は以下の URL からダウンロードします。

<http://overturetool.org/download/>

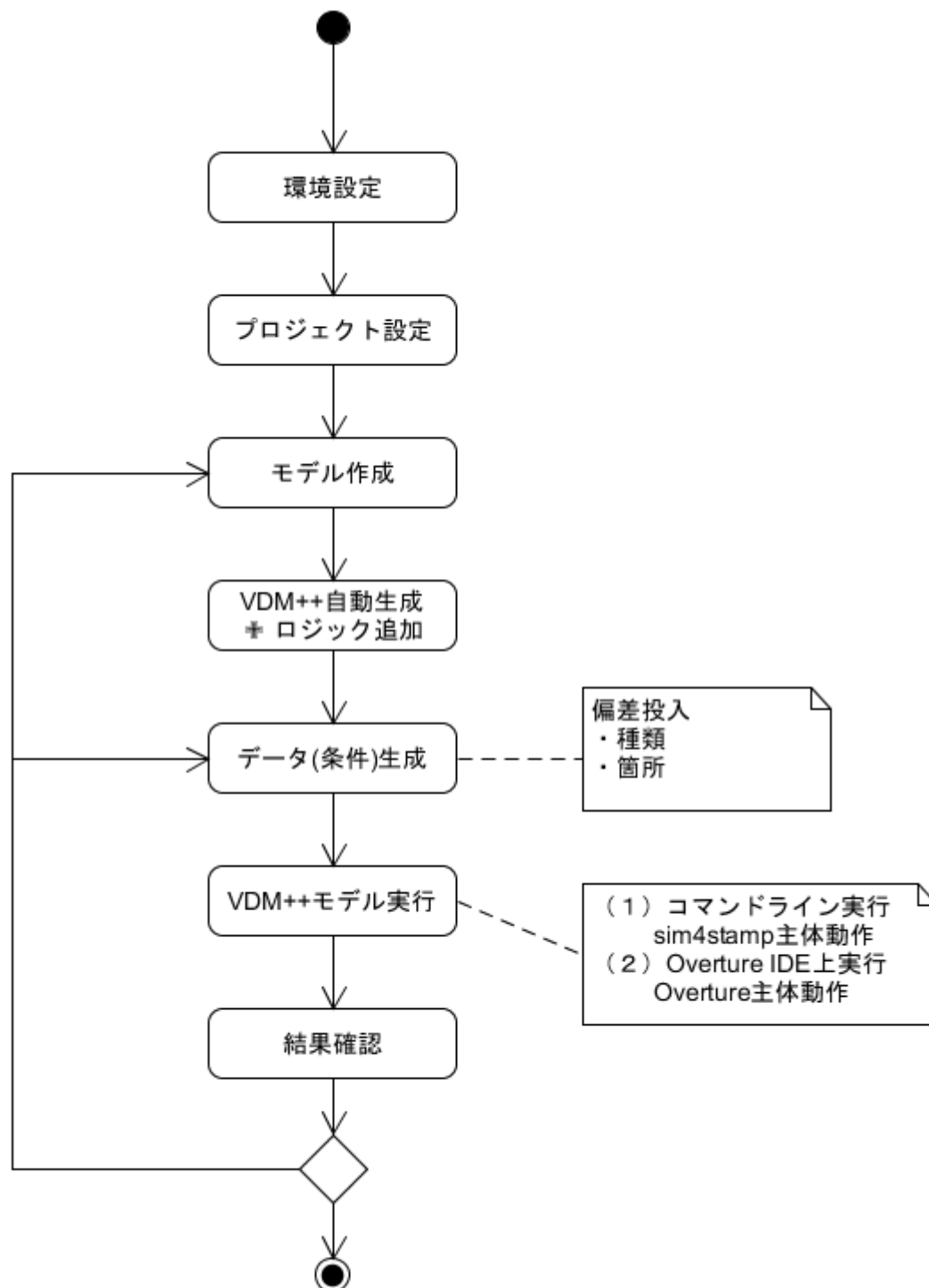
Overture では、デフォルトのエンコーディングは Windows では SJIS となっているので、Sim4stamp と組み合わせて用いるときは、マルチバイトコードが文字化けしてしまいます。そのため、以下の処置を行ってデフォルトのエンコーディングを UTF-8 に変更しておきます。

Overture のインストールフォルダの直下の「Overture.ini」をエディタで開き以下のコードを追加します。

```
-Dfile.encoding=utf-8
```

3. シミュレーション手順

以下に、本ツールの使用手順を示します。



4. インストール、実行環境整備

Sim4stamp を動作させるには、Sim4stamp の実行ファイルと連携して動作させる Overture が必要です。

4.1. Sim4stamp インストール

Sim4stamp の実行ファイルは、「sim4stamp-X.X.X.jar」と言うファイルにまとまっています。

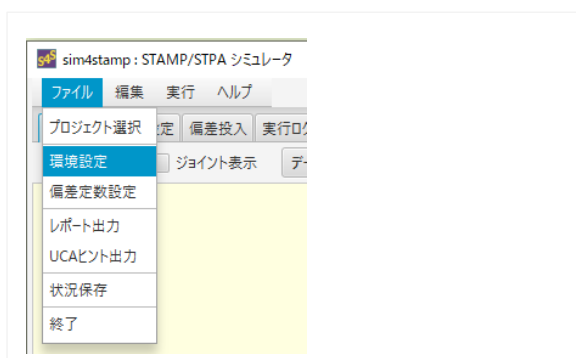
そのため、「sim4stamp-X.X.X.jar」を入手し、任意のフォルダに置くことでインストールは完了します。

Sim4stamp の初回起動時、環境変数等のパラメータを保持するファイルを生成し、以降、そのファイルを使用します。

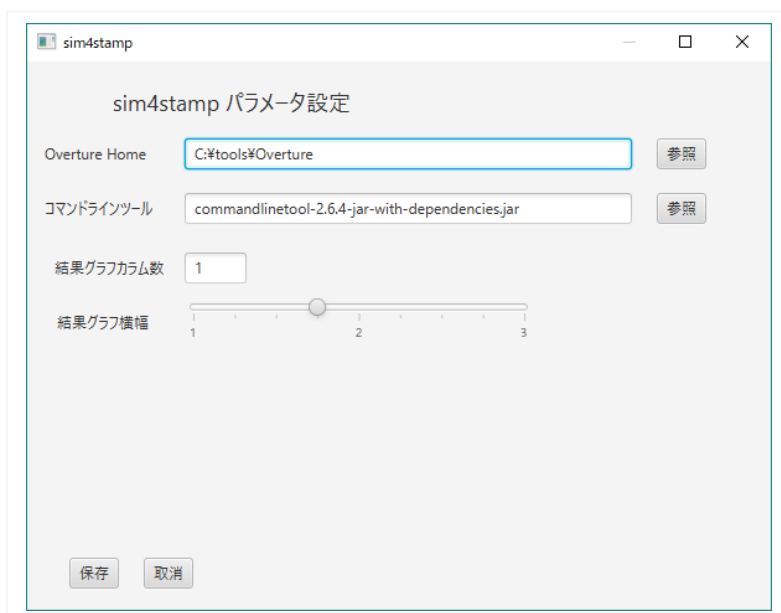
4.2. Overture 環境の設定

次節で示す Sim4stamp の起動方法を用いて、Sim4stamp の起動後、以下の操作を行います。

Overture を連携させるために、あらかじめインストールしてある Overture を Sim4stamp に登録します。



「ファイル」－「環境設定」を開きます。



「Overture Home」には Overture のインストールフォルダを指定します。

注)

コマンドラインツールは、Overture のホームフォルダ下の「commandline」フォルダにあります。

5. Sim4stamp の起動

Sim4stamp の起動は、以下に示す方法（複数あり）で行います。

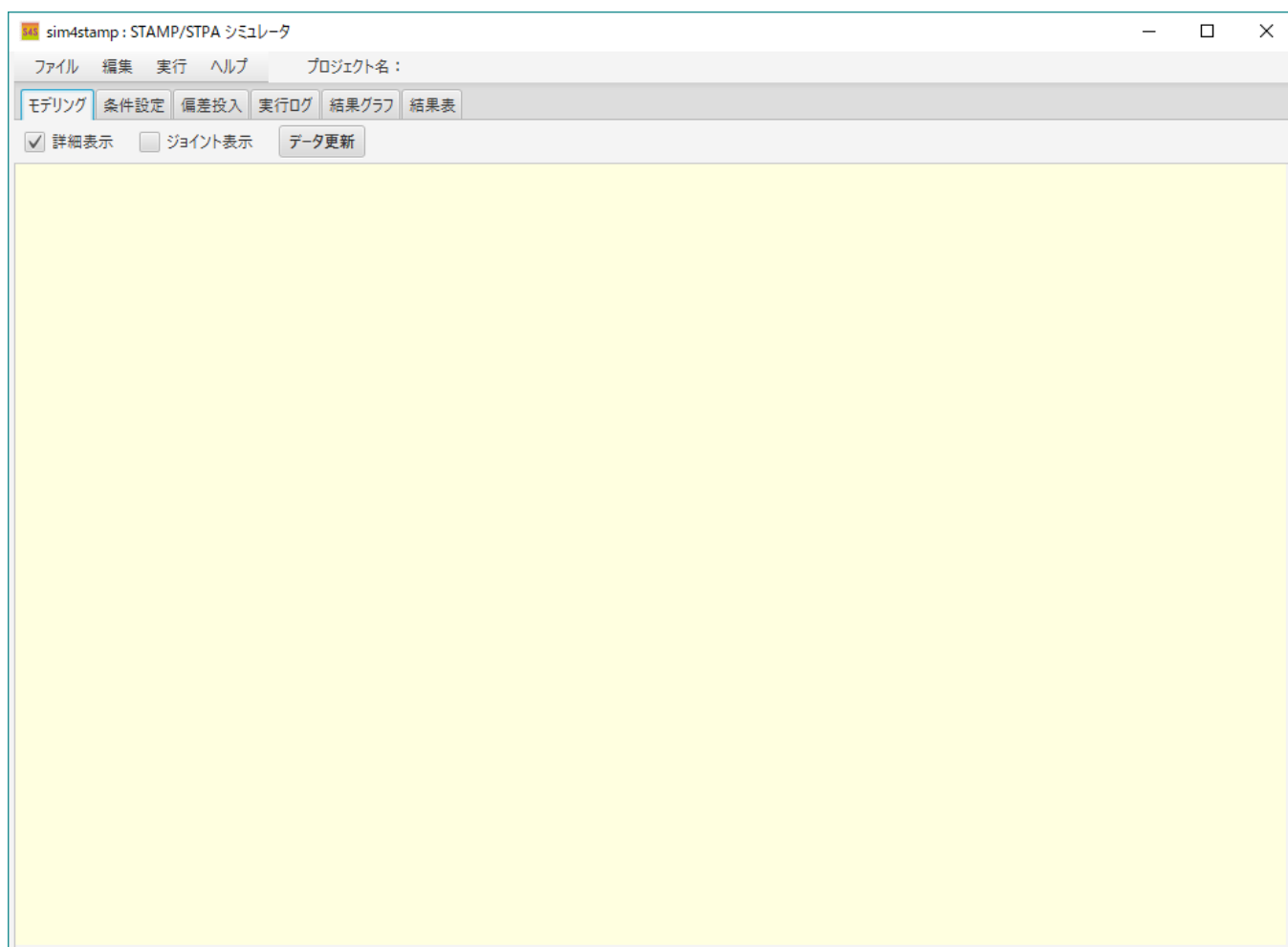
（１）「sim4stamp-X.X.X.jar」をダブルクリックする。

（２）コマンドプロンプトより

```
java -jar sim4stamp-X.X.X.jar
```

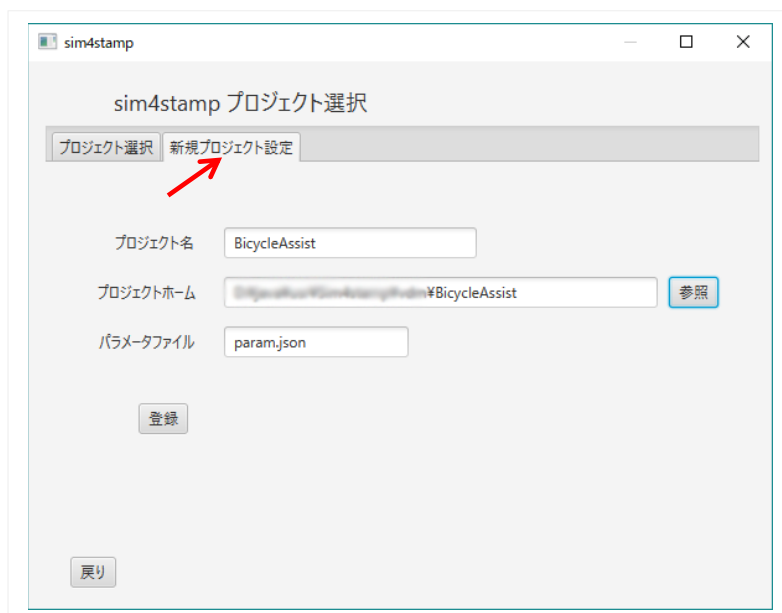
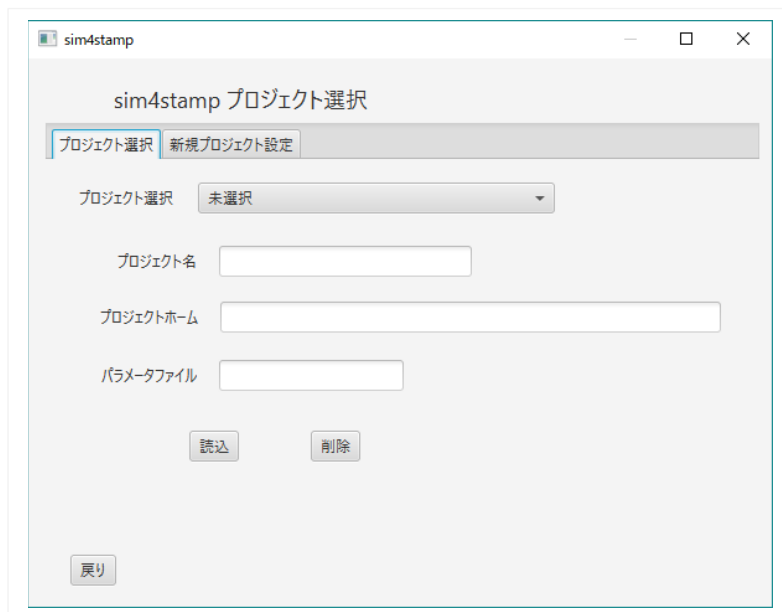
をタイプし、Enter キーを押します。

起動が成功すると以下に示す画面（インストール後の初回起動）を表示します。なお、インストール後の初回起動以外は、前回終了時のプロジェクトが開かれた状態で画面が表示されます。



6. プロジェクト設定

VDM++のプロジェクトを **Overture** 側で作成し、そのプロジェクトを **Sm4stamp** に設定します。
「ファイル」－「プロジェクト選択」を開くと以下のダイアログを表示します。

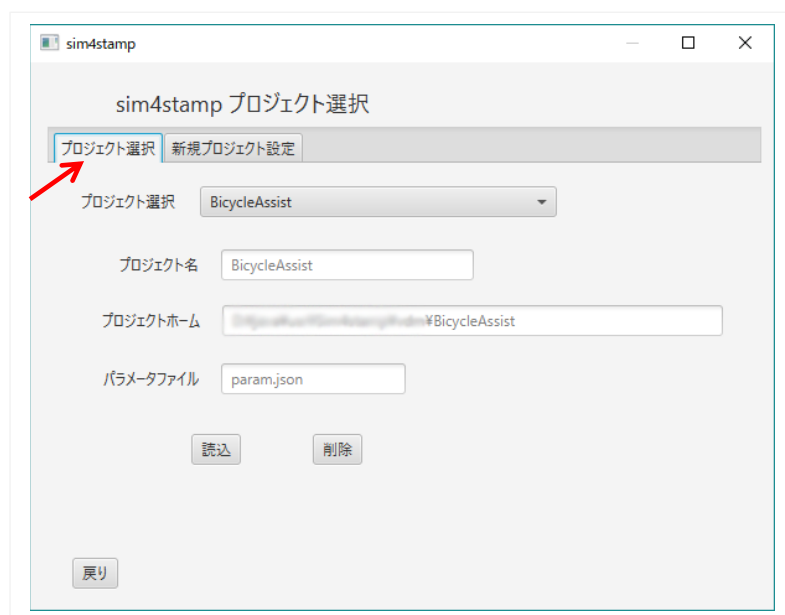


「新規プロジェクト設定」タブに移行し、**Overture** 側のプロジェクトの情報を設定します。

パラメータファイルは、**Sim4stamp** 側で追加するファイルですが、デフォルトのままとします。

なお、プロジェクト名はプロジェクトホームで指定したプロジェクトフォルダ名が自動設定されます。

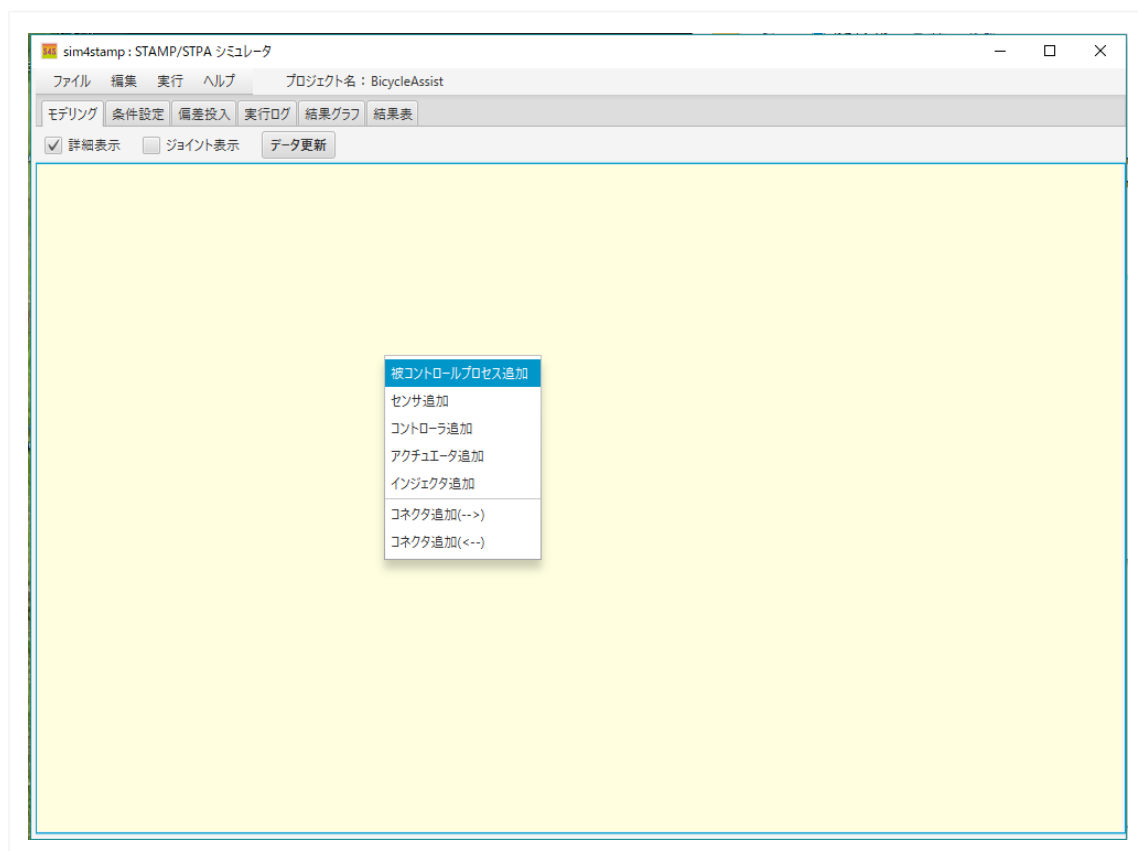
その後、「登録」ボタンをクリックし、登録を行います。



登録後、「プロジェクト選択」タブに戻り、プロジェクト選択（プルダウンメニュー）を行って、「読込」ボタンをクリックすることにより、**Sim4stamp** のプロジェクトが選択プロジェクトになります。

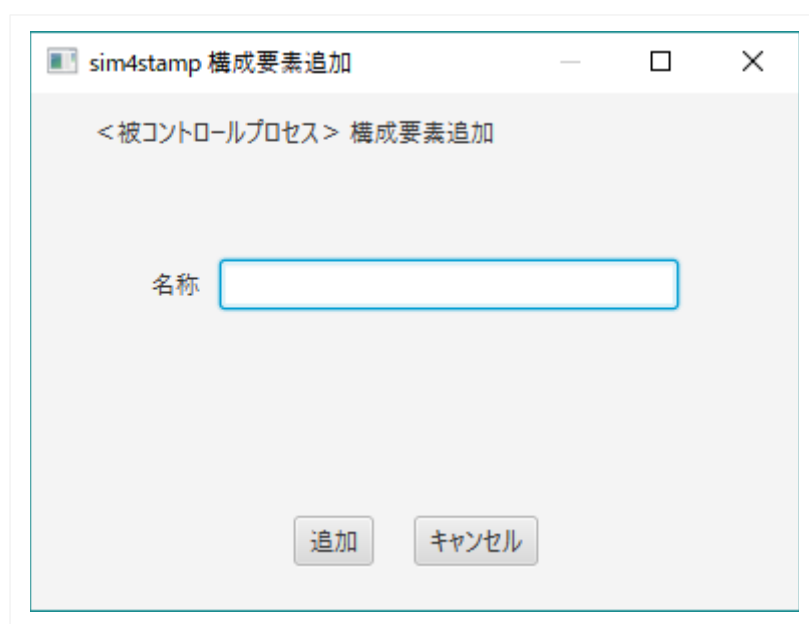
7. モデル生成

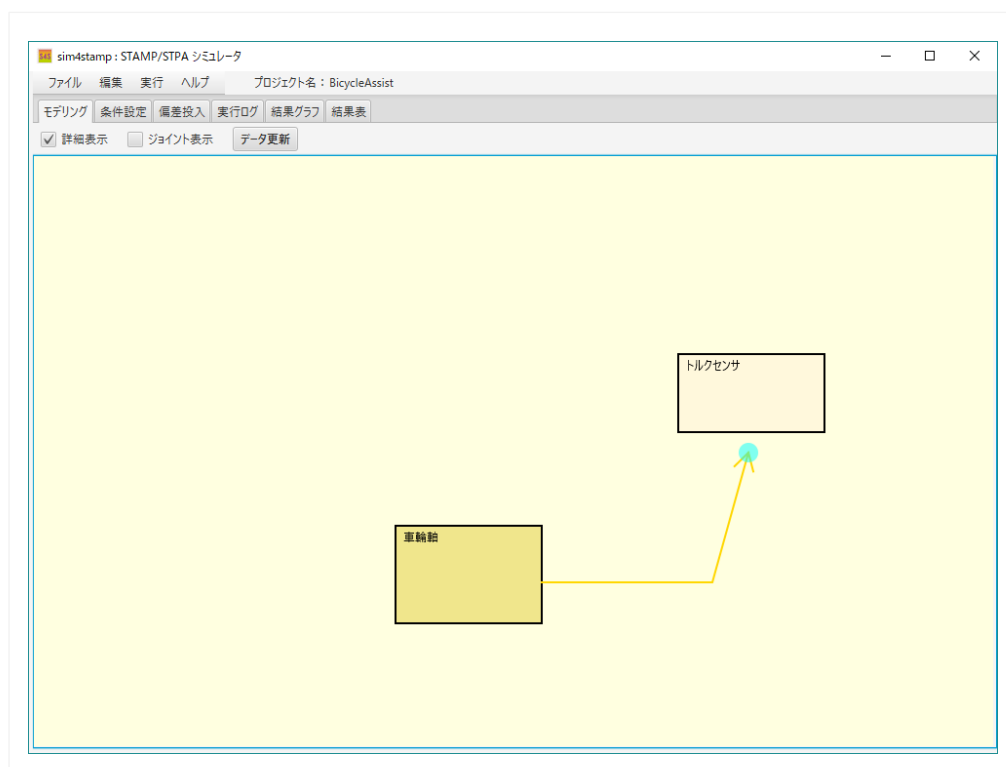
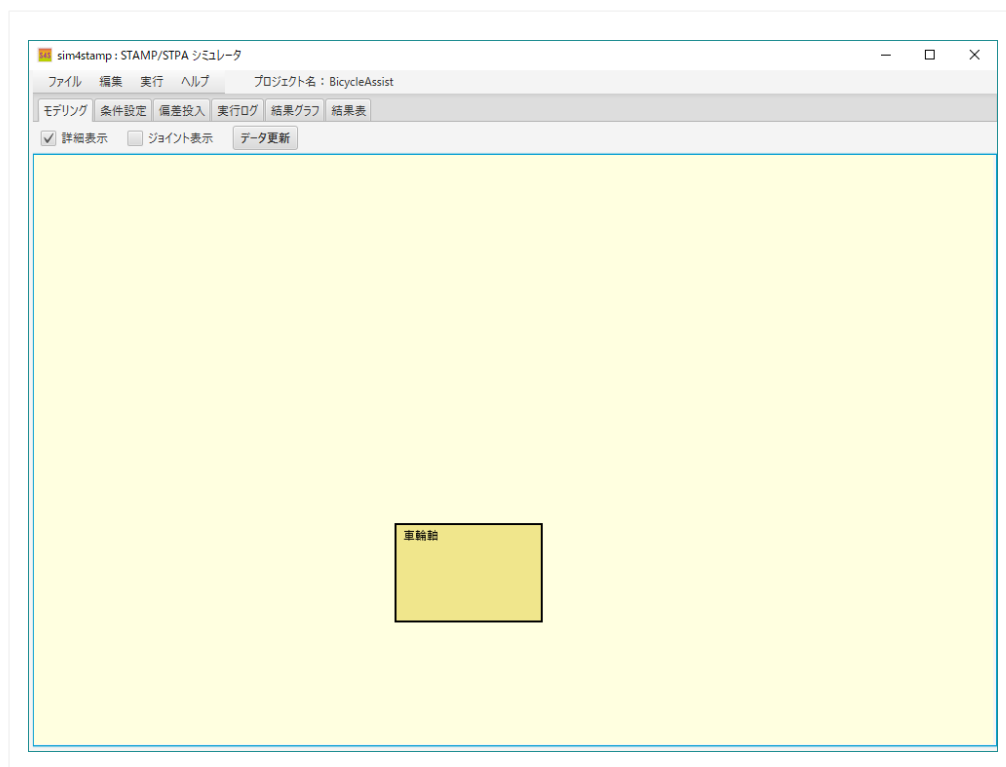
モデルの生成は、「モデリング」タブ内のモデルパネルで行います。



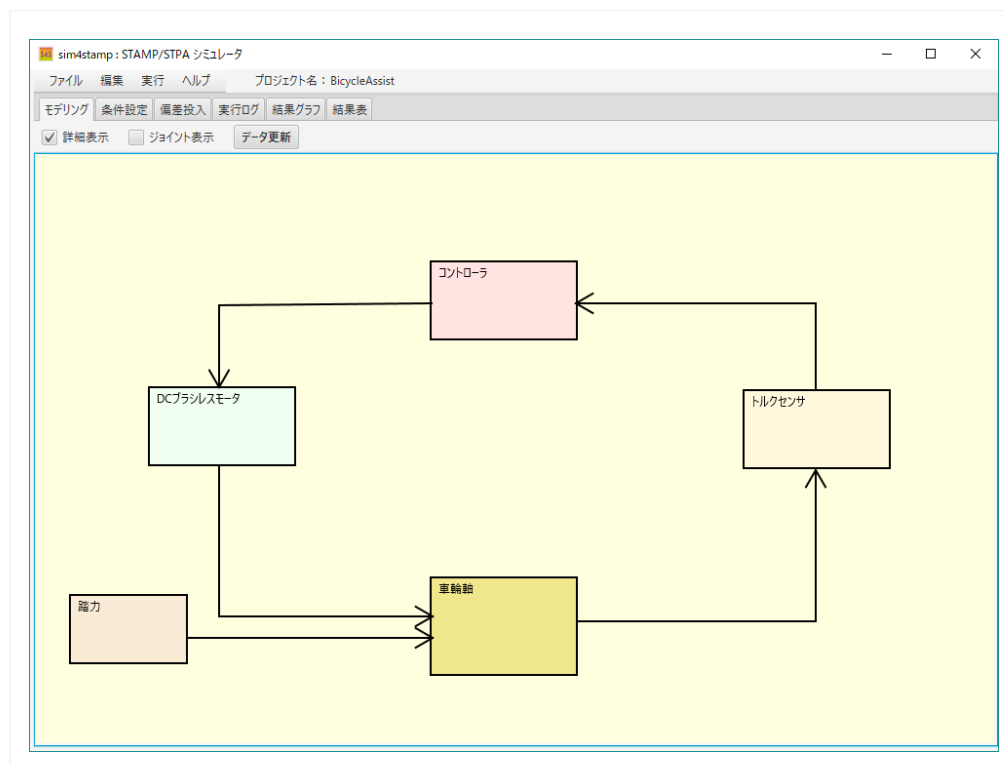
モデルパネル上で、右クリックを行うと追加する部品のリストが表示されます。

部品リストから目的の部品を選択することにより、追加部品のダイアログが表示され、必要事項を入力し、部品をモデルパネルに置きます。





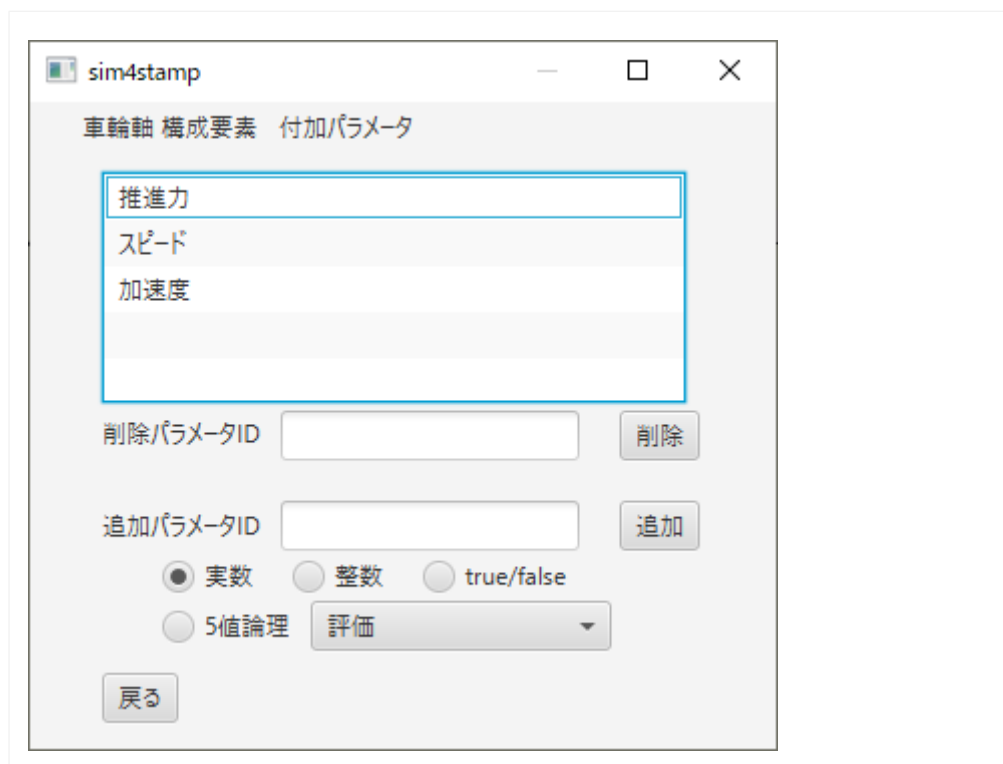
コネクタも設定し、部品要素間のデータの流れを設定します。コネクタは、両端が部品要素に接続されると色が黒に変化します。



部品要素としては、「被コントロールプロセス」、「センサ」、「コントローラ」、「アクチュエータ」、「インジェクタ」の 5 種類があります。「インジェクタ」は本来の STAMP/STPA のモデルにはありませんが、本ツールにてデータの初期値を設定するのに使用します。

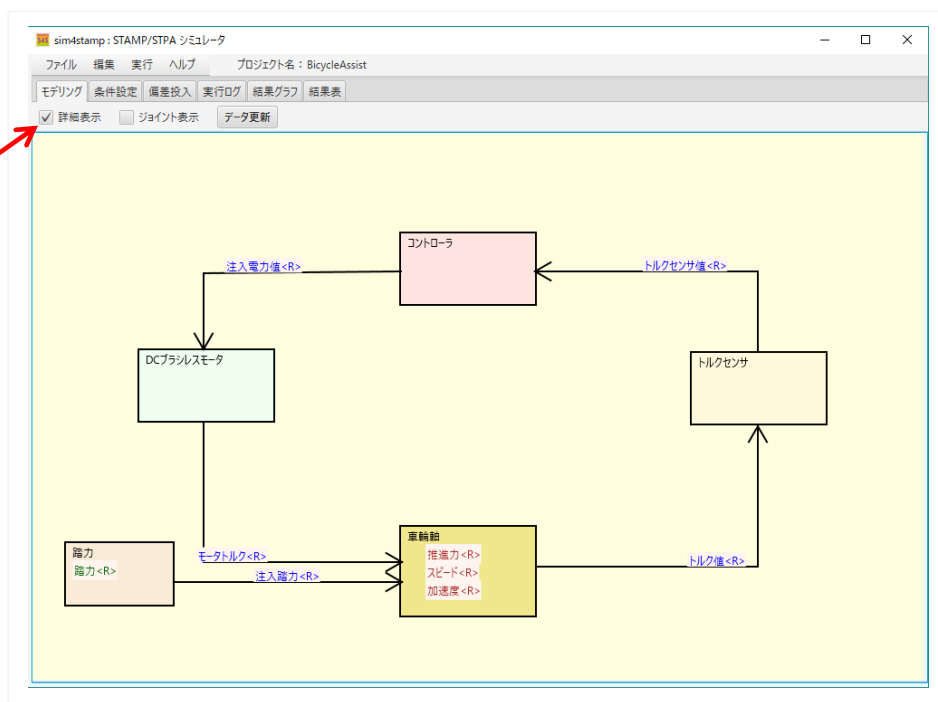
部品要素とその接続関係の設定が完了後、パラメータを設定します。

パラメータの設定は、部品要素であれば、部品要素の内部で右クリックを、コネクタであれば、折れ曲がる頂点上で右クリックします。

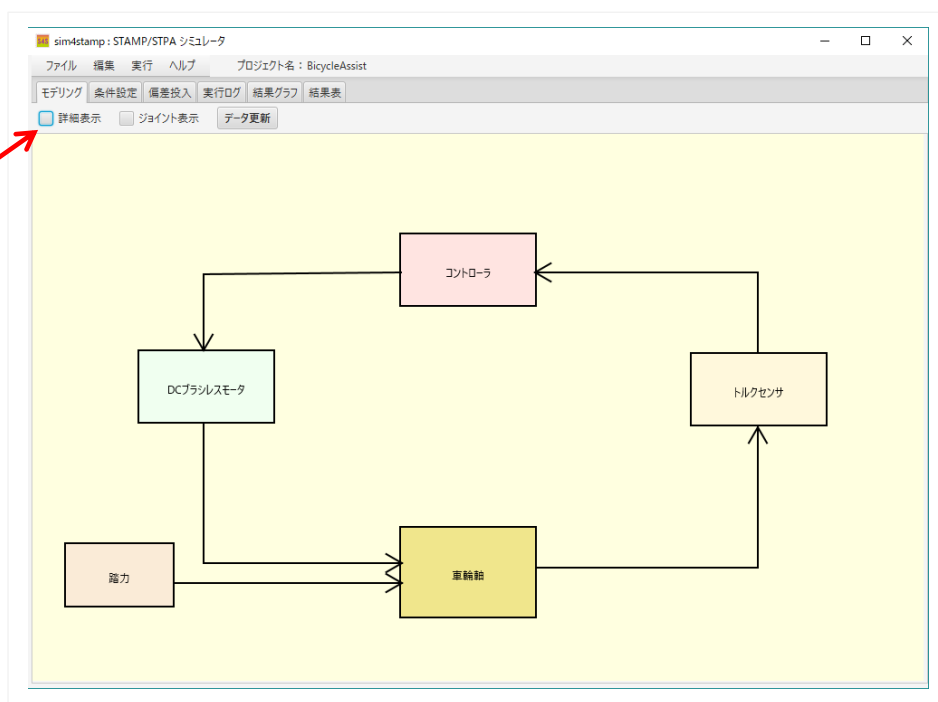


パラメータの型は、実数、整数、論理値（true/false）、5 値論理の 4 種類が選択可能です。

「詳細表示」をチェックすると設定されているパラメータが図上に表示されます。



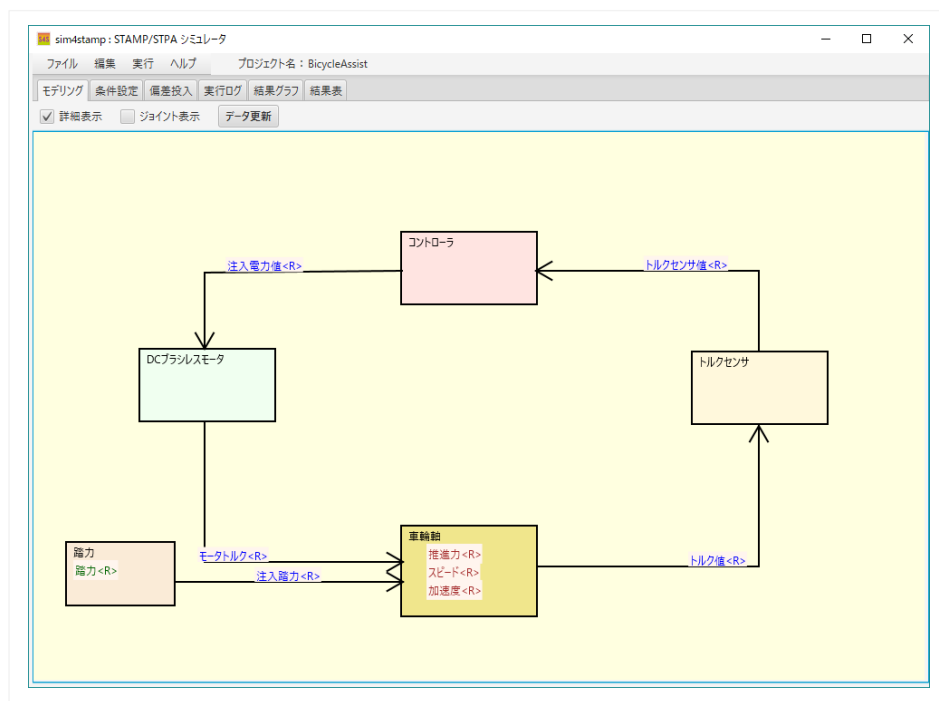
「詳細表示」のチェックを外すと簡易表示となります。



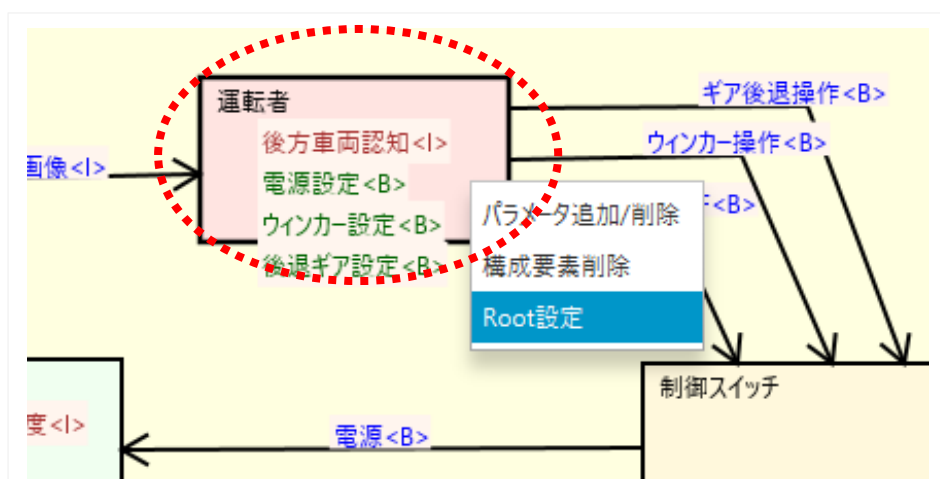
なお、コネクタに設定できるパラメータは1つのみです。複数設定する必要がある場合は、複数コネクタを設定してください。

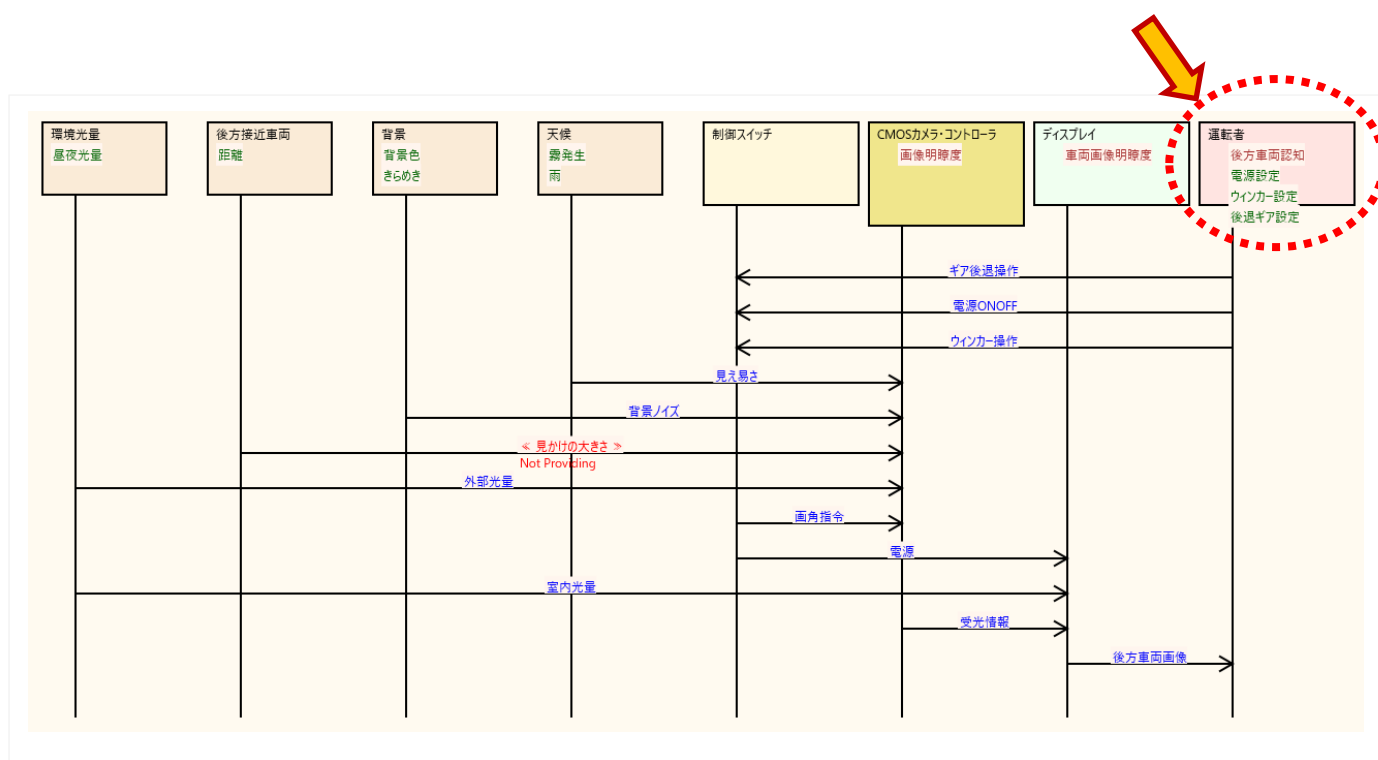
モデル図が完成したら、最後に「データ更新」ボタンをクリックして、内部データの更新を行ってください。
データ更新を行うことにより、次節以降で述べるデータ（設定データ）にデータの種類や設定項目が反映されます。

モデル図を変更した場合、モデルの単なる位置関係以外の変更は「データ変更」ボタンをクリックし、内部データの更新を行ってください。



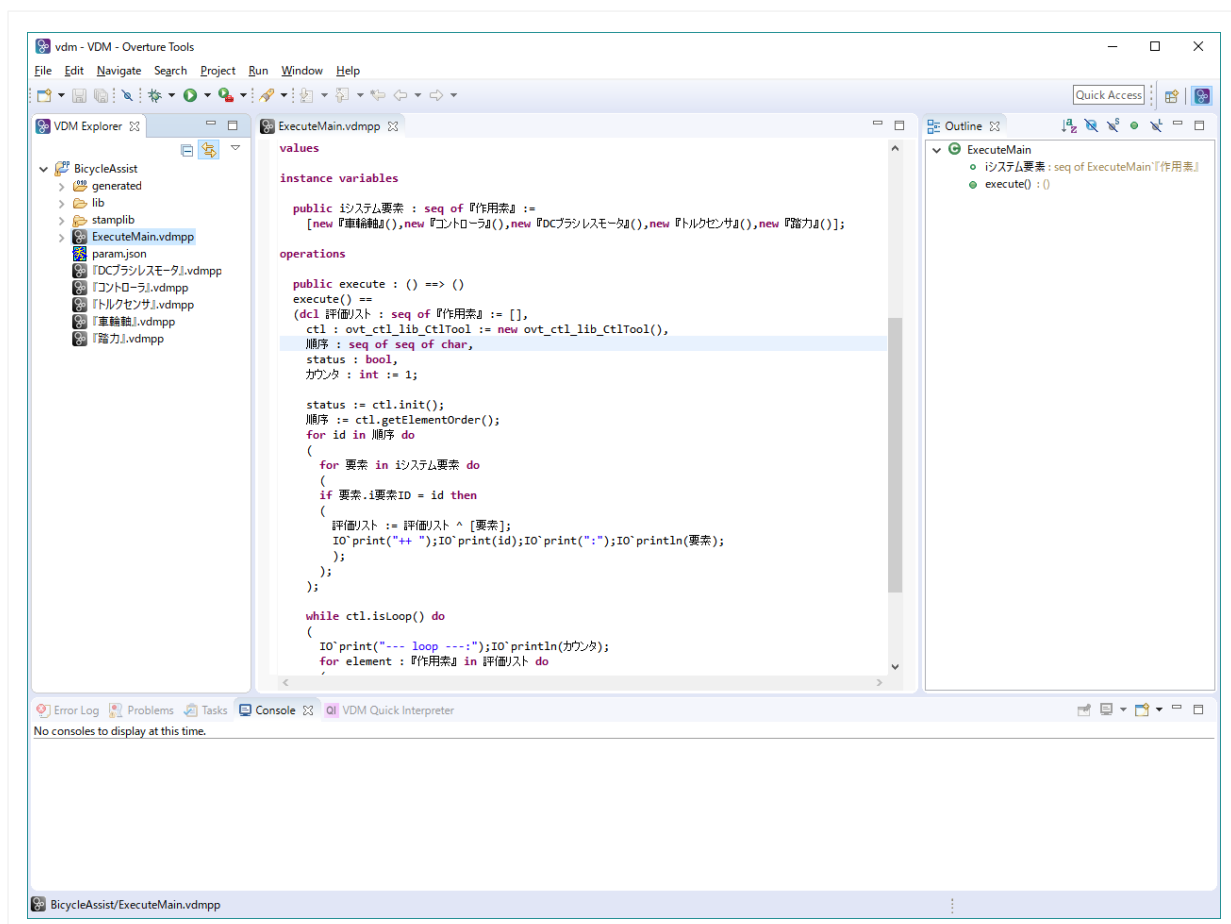
なお、偏差投入タブのシーケンス図は、モデル図の接続関係から自動作成されます。
ただし、自動生成のため、意図するシーケンス図の配置とならないケースも発生することがあります。
その対策として、モデル図上で「Root 設定」を行うことにより、「Root 設定」された構成要素をシーケンス図の最右端に配置することが可能な機能を追加しています。





8. VDM++モデル生成

「データ更新」ボタンクリック後に、「編集」－「VDM++生成」を行うことにより、VDM++のテンプレートコード(エレメント毎に入出力データの setter、getter を生成)が自動生成されます。詳細は付録-1 VDM++ 自動生成例を参照してください。



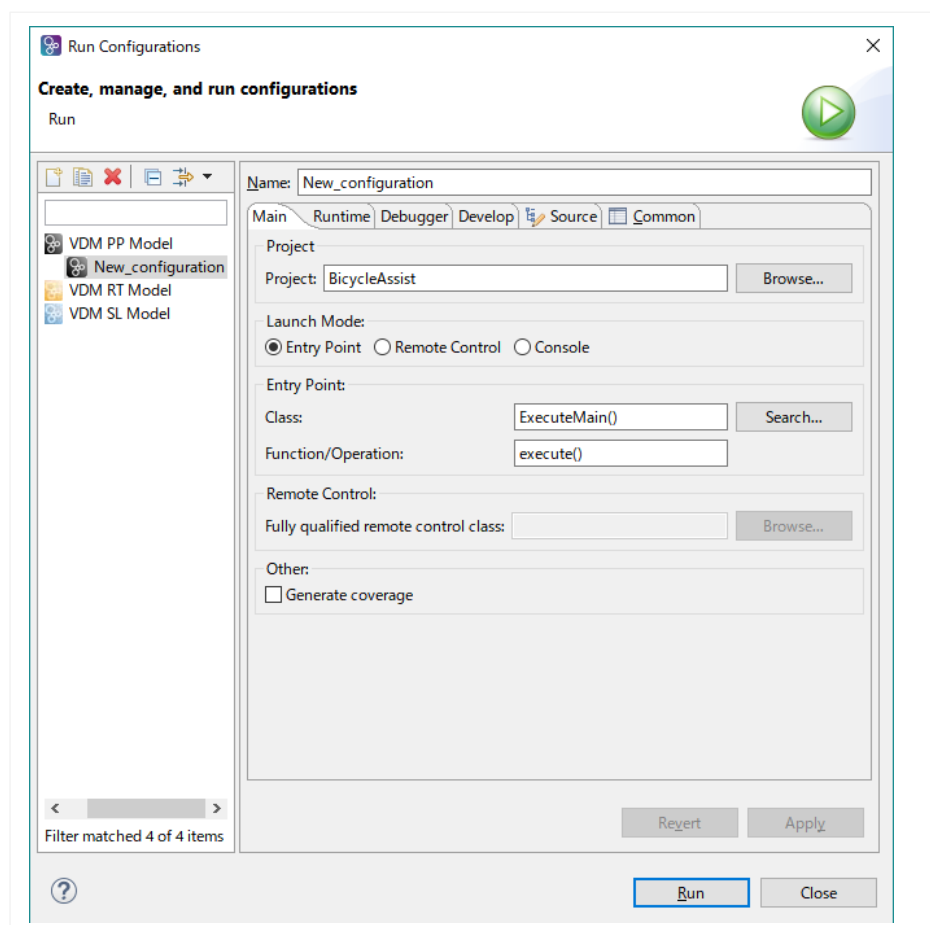
VDM++の生成は、初回は拡張子が「.vdmpp」で生成されますが、同一ファイル名がすでに存在する 2 回目以降は「.vdmpp_ini」で生成し、元のファイルを上書きしないようにします。

次に生成されたテンプレートコードを元に、Overture 側で VDM++のロジックを追加します。

なお、Sim4stamp 側との通信ライブラリも同時にプロジェクトフォルダにインストールされます。

作成された VDM++コードを実行する場合、Overture から実行するには、以下に示すように Entry Point

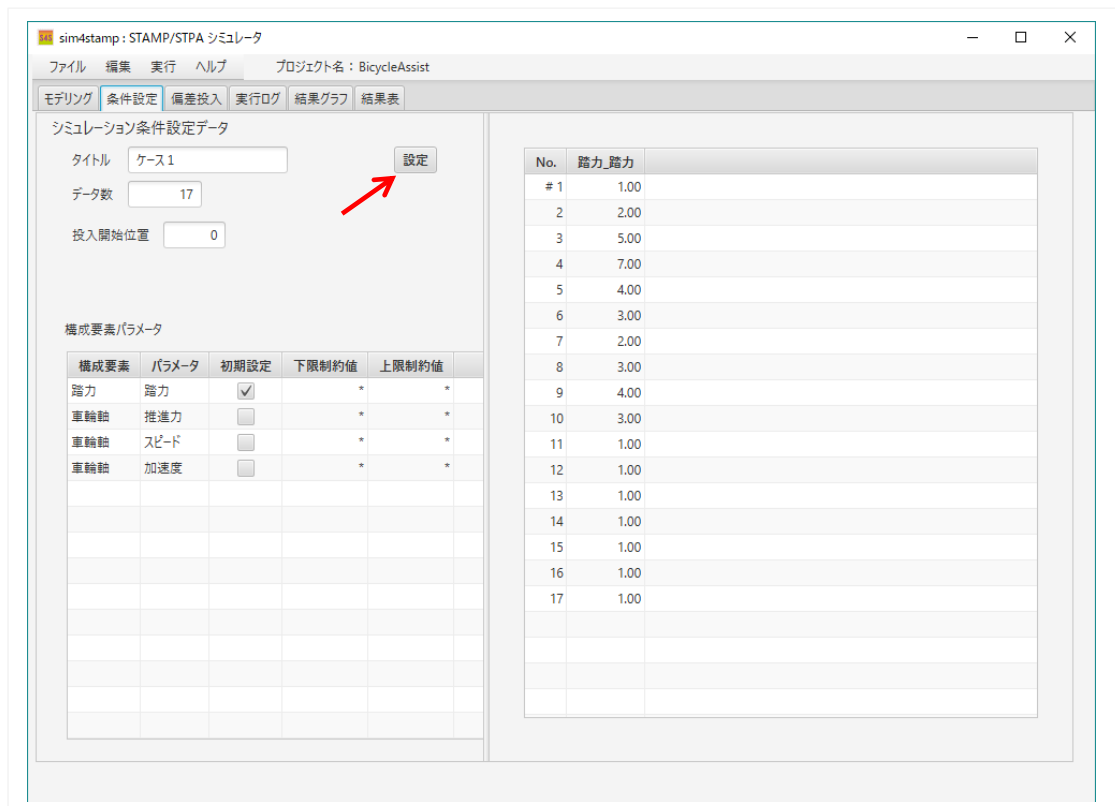
を設定します。



なお、VDM++の実行は Sim4stamp からコマンドラインツールを起動して行う方法でも可能ですが、VDM++のロジックを作成する場合は、Overture 側で Debug 機能を用いて作業を行うことも可能です。

9. 初期値データ設定

シミュレーション条件の初期値設定は、「条件設定」タブで行います。



「データ数」はシミュレーションの各ステップで設定するデータ数です。デフォルトでは 10 個となっています。

初期値設定を行うには、構成要素パラメータの一覧表の「初期設定」カラム欄がチェック状態となったものについて初期値の設定が行えるようになっています。同欄を変更後、「設定」ボタンをクリックすることにより、右側の初期値設定テーブルの設定項目が更新されます。

なお、データの初期値設定を行う際、データの型はモデル設定で行った型に従って行います。

また、構成要素パラメータ欄には、下限制約値、上限制約値が設定することが出来るようになっています。この制約値に設定を行うとグラフ表示時に制約値を同時表示し、逸脱がすぐに分かるようになります。

偏差投入は次節で述べる偏差投入タブを開いて行いますが、偏差投入の開始位置を設定することが変更可能になっています。「投入開始位置」の数値を変更し、「設定」ボタンをクリックして設定します。なお、偏差投入位置は右側の「No.」欄の番号に「#」がついている行で表しています。

偏差投入の設定位置の用途は、電動自転車の例で言うと、自転車が走りはじめ、低速になったときに、上り坂に差し掛かるケースで、坂に差し掛かる位置を偏差投入したい場合等（偏差投入と制約チェックの開始タイミングを調整）に使用します。

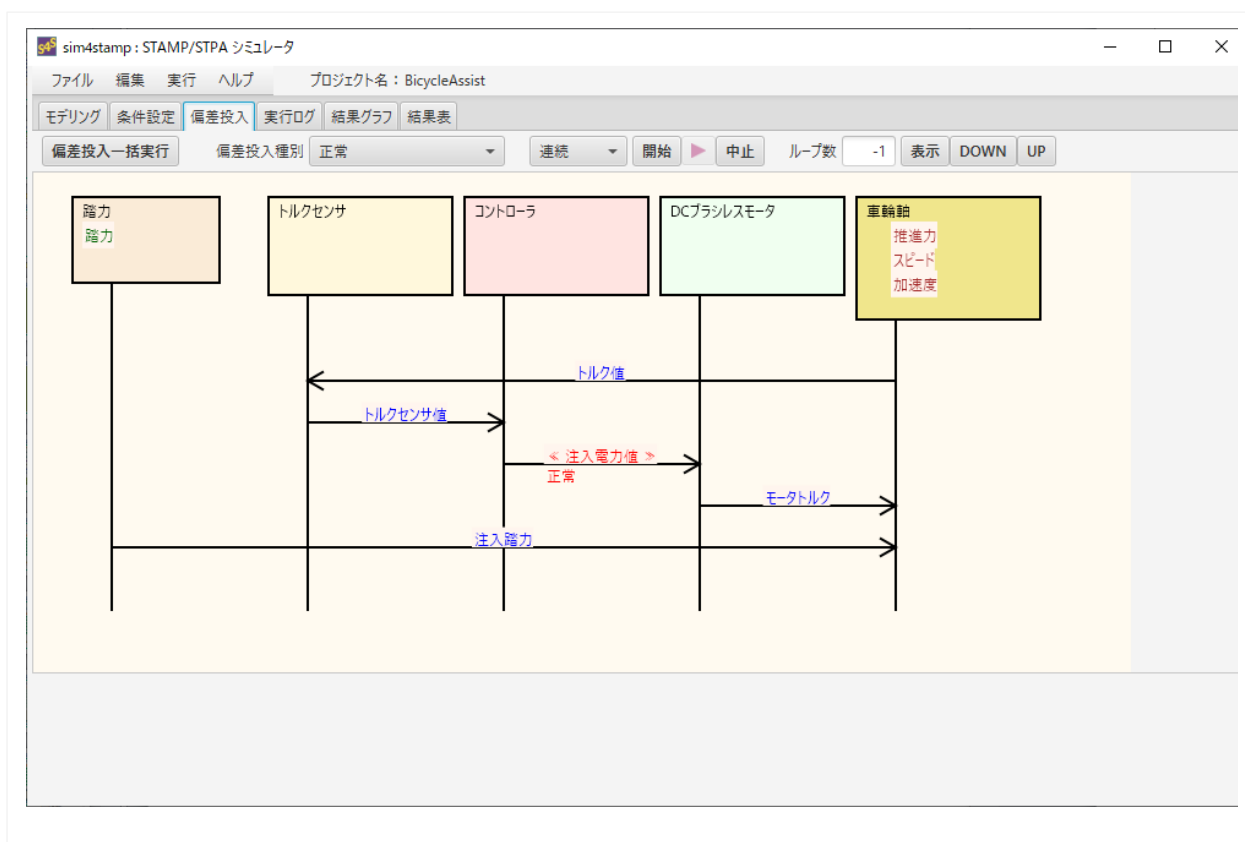
なお、逸脱判定は、「投入開始位置」の次のタイミングから開始します。

投入開始位置に「2」を設定した場合、逸脱判定は「3」以降となります。

また、逸脱については、シミュレーション実行結果をグラフ等で表示する場合について、ビジュアルに上下限値を同時表示することの他に、UCA ヒントの出力対象の判定に使用されます。

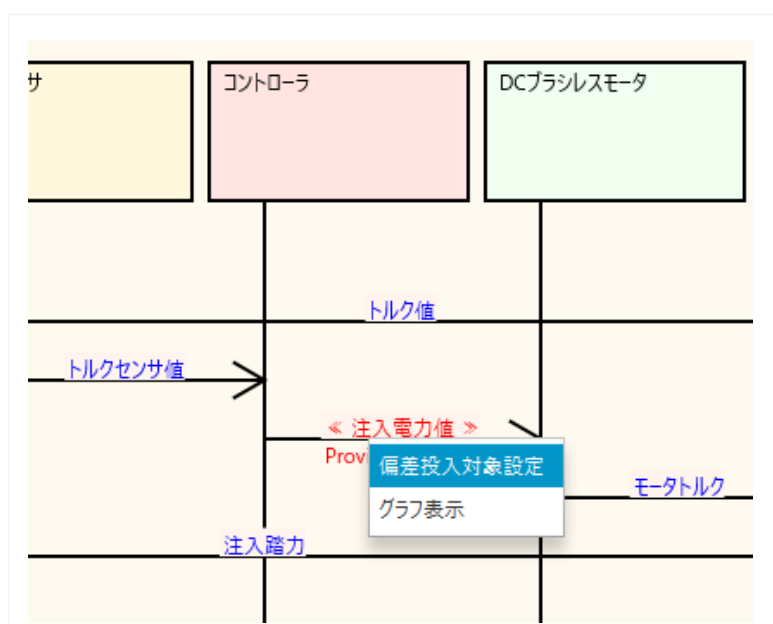
10. 偏差投入設定

偏差投入の設定は、偏差投入タブを開き、偏差投入する場所と種類を設定することにより行います。偏差投入の場所の設定は、本画面上で横矢印の部分（コネクタ）で行います。

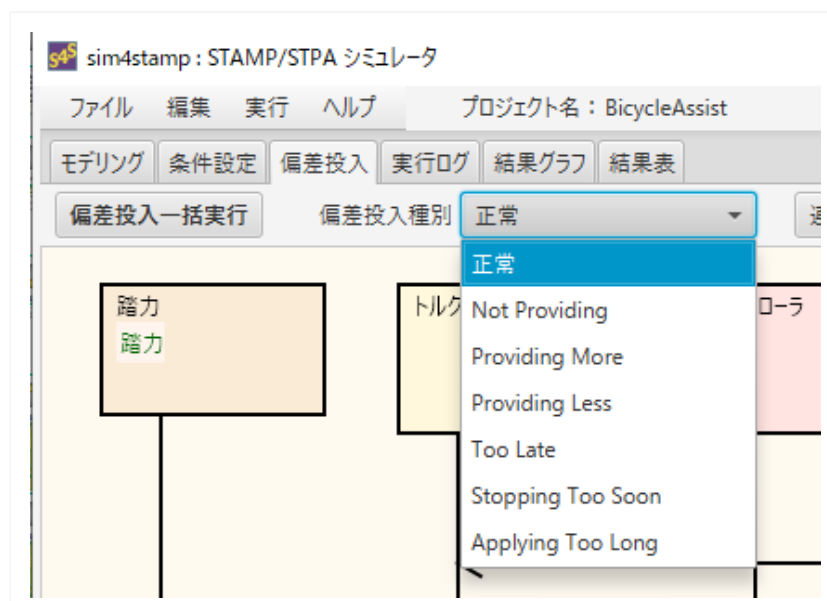


なお、本画面は、設定されたモデル図のデータから Sim4stamp 内で自動生成されます。

横矢印の部分をクリックするとポップアップが出現します。そこで「偏差投入対象設定」を選択すると偏差投入位置を指定することが出来ます。

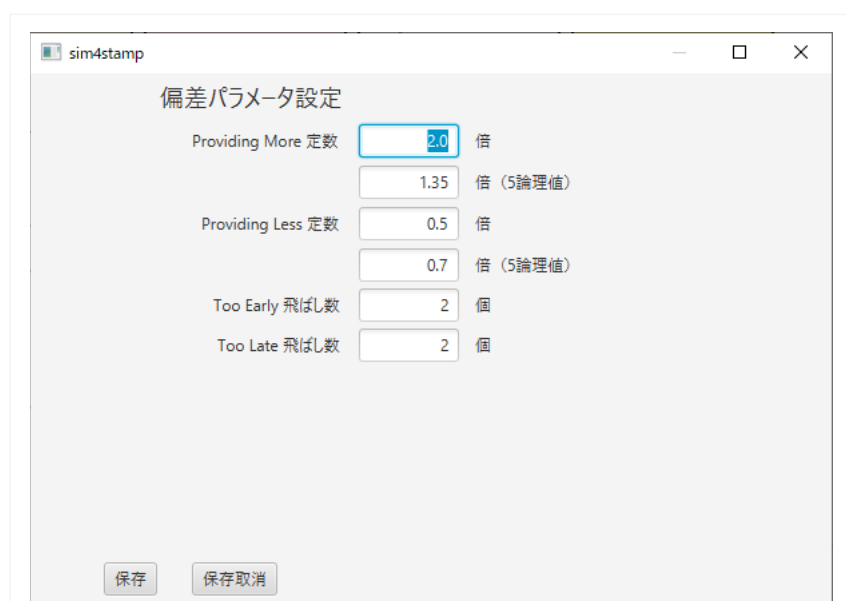


偏差投入の種類を設定するには、ツール画面の上側に表示される偏差投入種別のプルダウンメニューより偏差投入種別を指定します。



最初に「正常」ケースを実行し、その次以降に偏差を指定することにより、正常ケースと偏差投入した結果を比較することが出来ます。

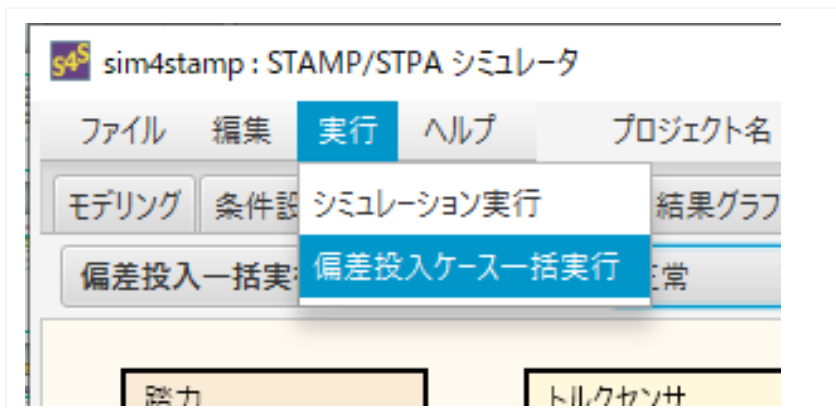
各偏差における設定値は、「ファイル」－「偏差定数設定」により、以下に示す偏差パラメータ設定画面より、各偏差倍率を設定することが出来ます。



11. VDM++の実行

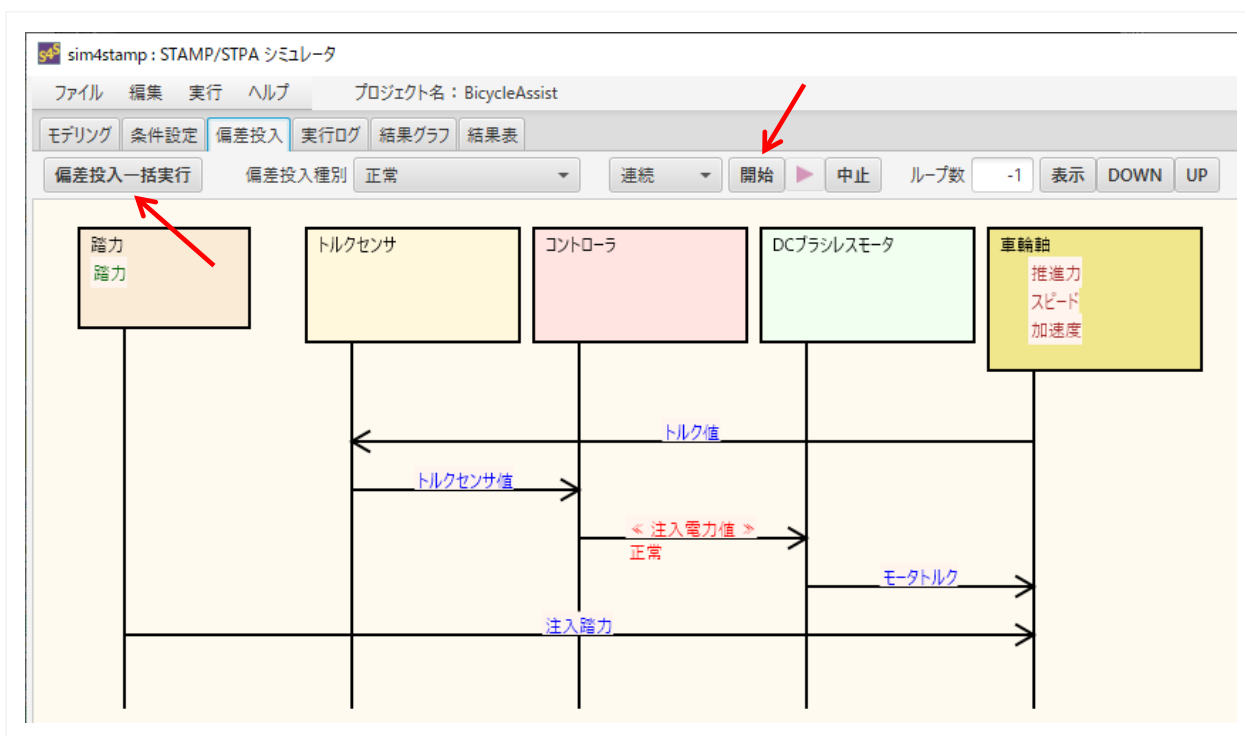
11.1. 実行方法その 1

Sim4stamp 側からの VDM++の実行の一つとして、「実行」－「シミュレーション実行」をクリックする方法があります。また、偏差投入について、一括して実行を行うことも可能です。その場合は、「偏差投入ケース一括実行」をクリックします。偏差投入の一括実行を行った場合は、「正常」から「Applying To Long」までの 7 ケースを繰り返し実行します。



11.2. 実行方法その 2

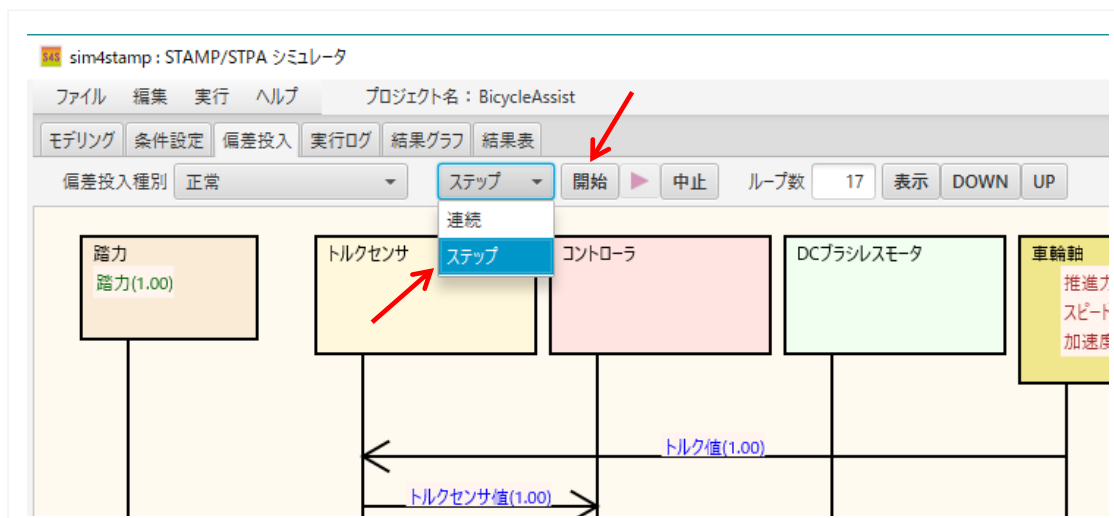
Sim4stamp 側からの VDM++の実行のもう一つの方法として、偏差投入画面から「開始」ボタンのクリックにより行う方法があります。



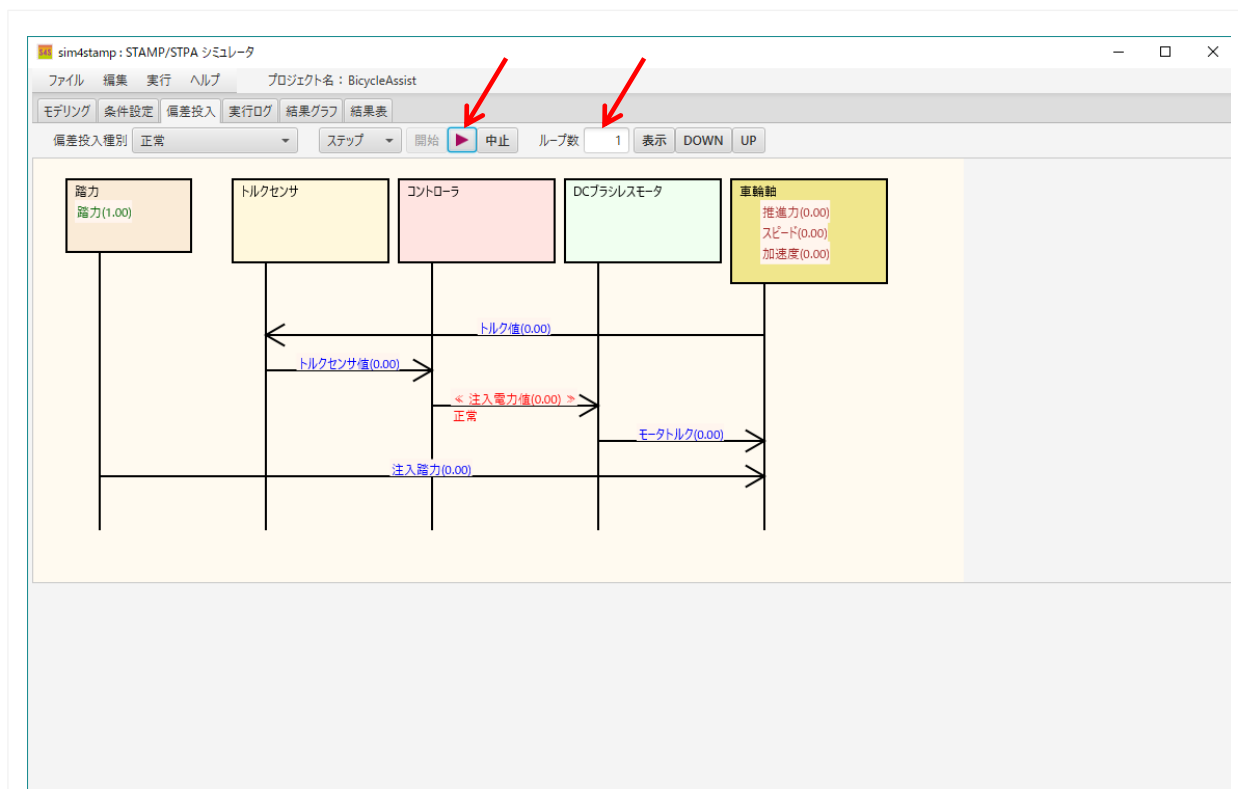
一括実行を行う場合は「偏差投入一括実行」ボタンをクリックします。

11.3. ステップ実行

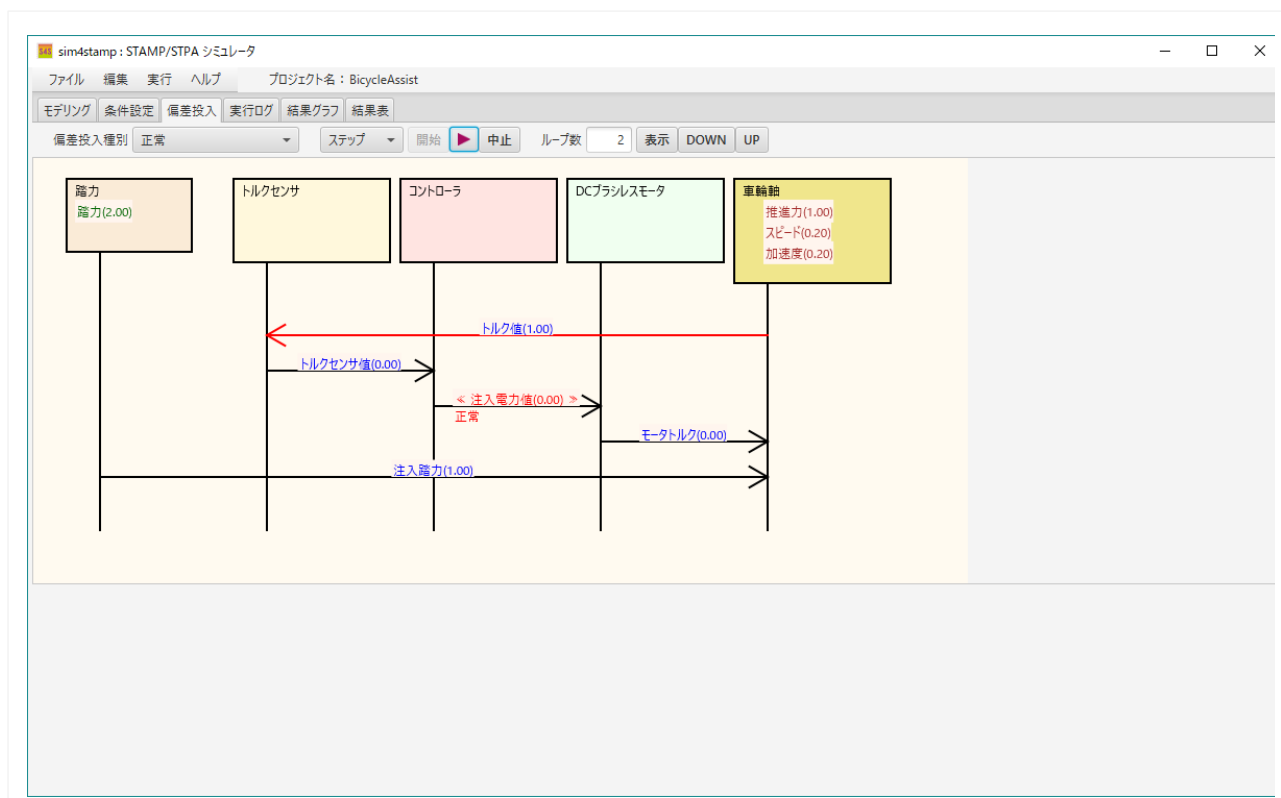
ステップ実行は偏差投入画面より行ないます。ステップ実行を指定し、「開始」ボタンをクリックします。



ステップ実行が開始されると開始ボタンの右隣にある三角マークボタンがクリック可能となります。この三角マークボタンをクリックすることによりステップ実行を行います。ループ数は現在実行された時のループ回数となっています。



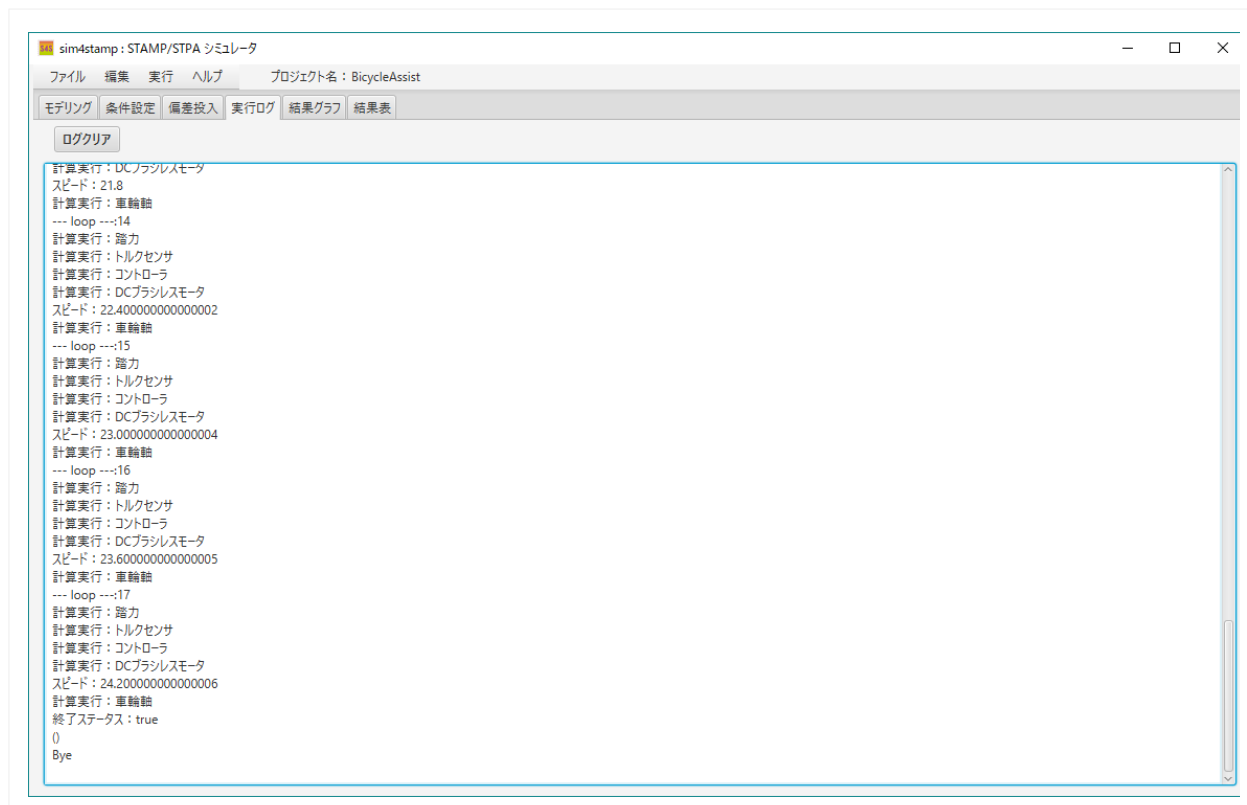
ステップ実行を行っているときには、ステップ実行中のコネクタが赤色表示されます。



ステップ実行を中止する場合は、「中止」ボタンをクリックします。

12. 実行ログの確認

Sim4stamp 側からの VDM++を実行する場合、実行ログ画面で VDM++側のログを確認することが出来ます。



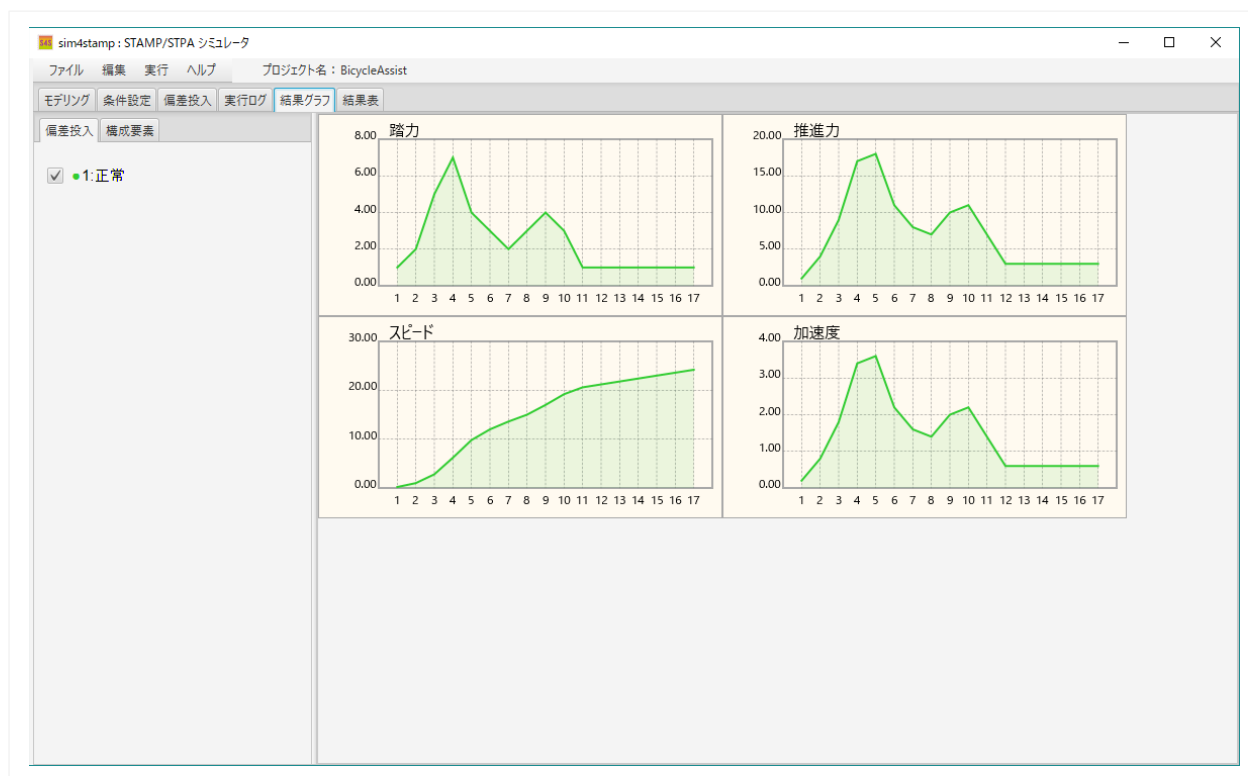
VDM++側で設定しているプリント文等で出力されるログを取り込み表示します。

13. シミュレーション実行結果表示

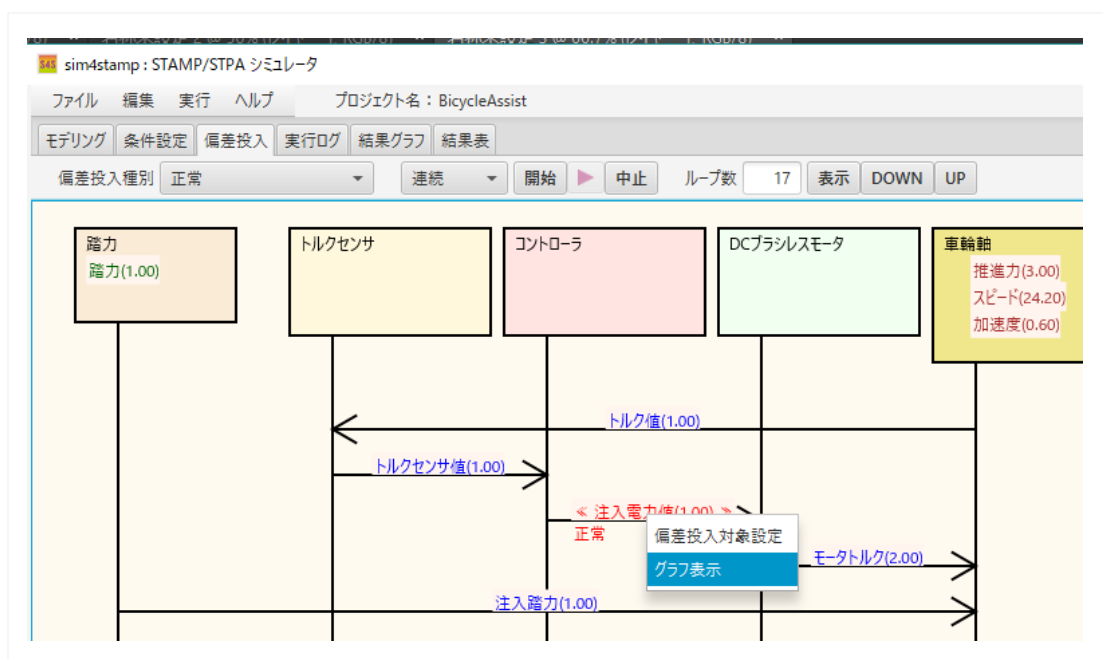
13.1. グラフ形式データ表示

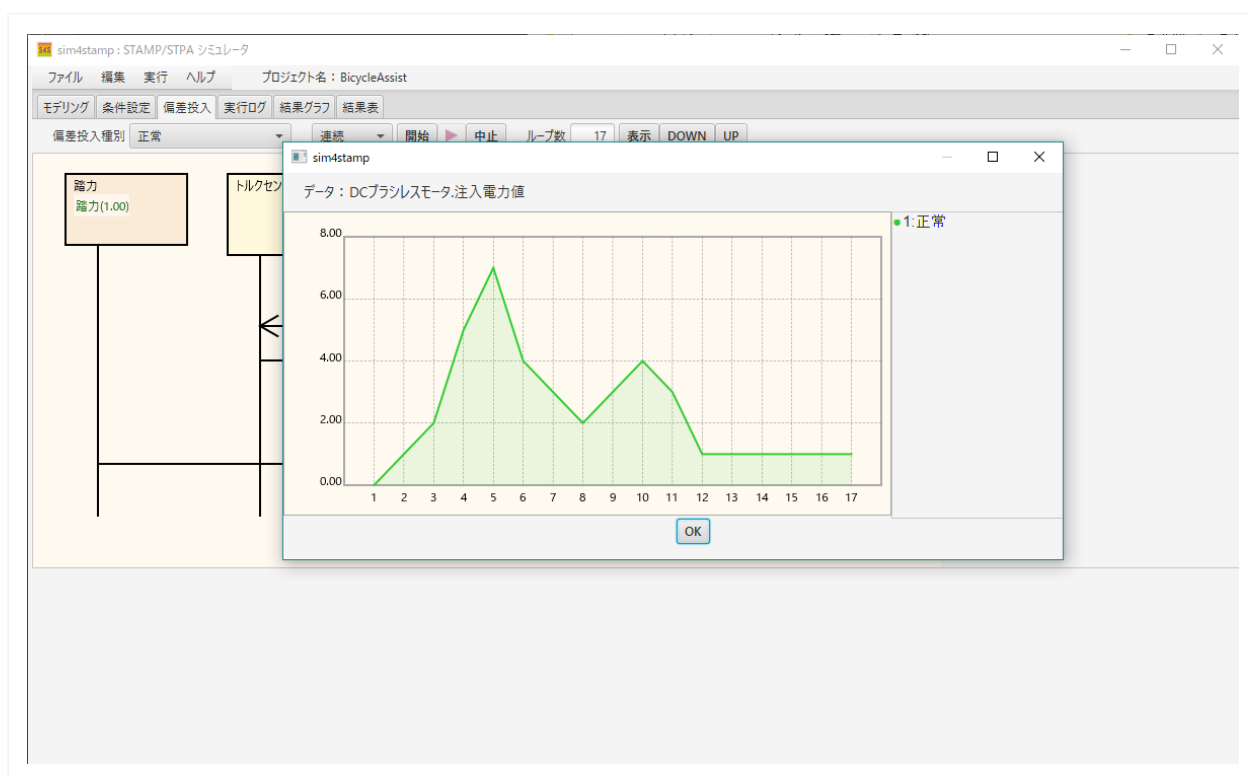
シミュレーション結果について、結果グラフの表示例を以下に示します。

グラフの列の数やグラフの横幅は、sim4stamp パラメータ設定ダイアログ画面（「4.2 Overture 環境の設定」を参照）から変更可能となっています。

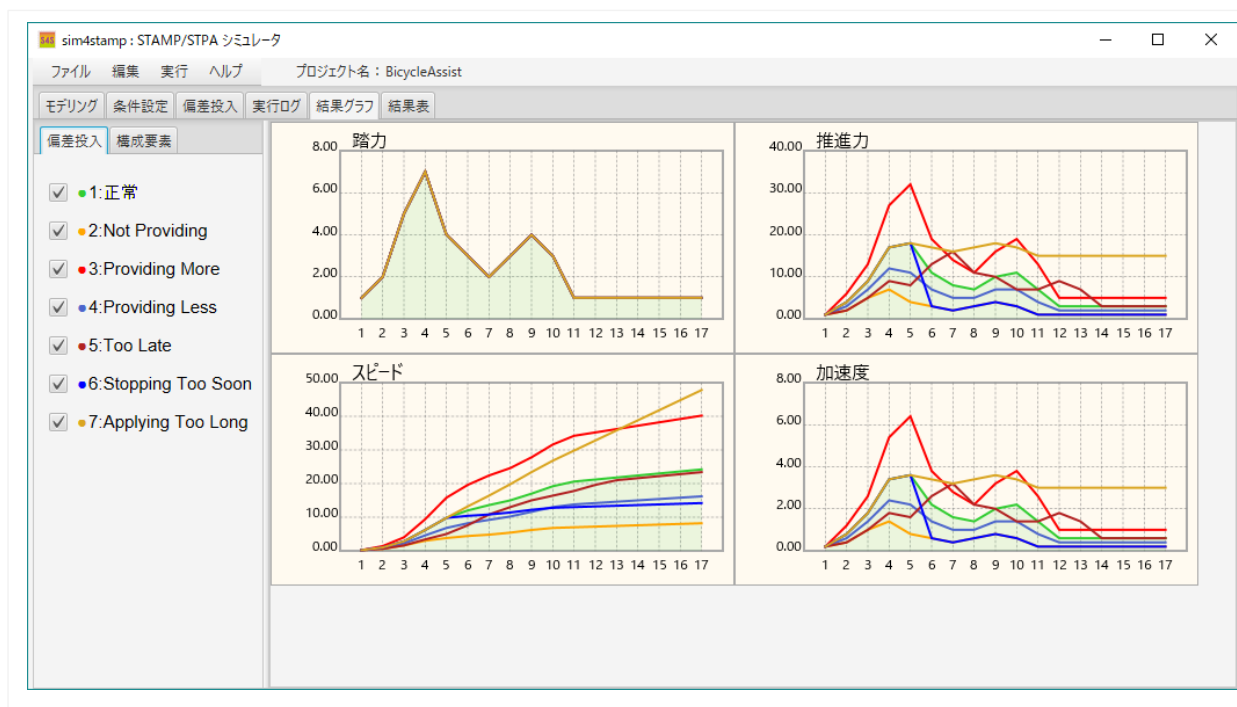


また、グラフ表示は偏差投入画面のコネクタ（横矢印の線）を指定する（クリックし、ポップアップを表示し、グラフ表示を指定する）と、当該コネクタに関するパラメータの値をグラフ表示します。



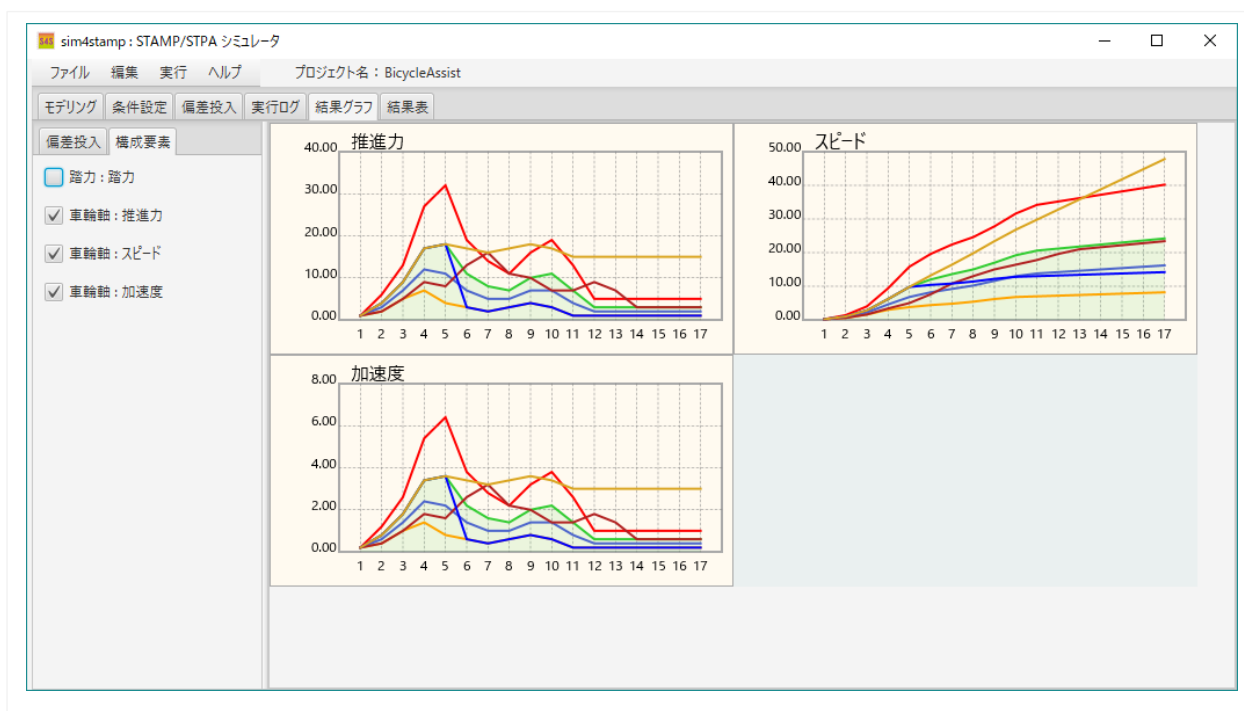


シミュレーションを累積すると、シミュレーションの累積グラフが表示され、正常ケースと偏差投入ケースの比較ができます。



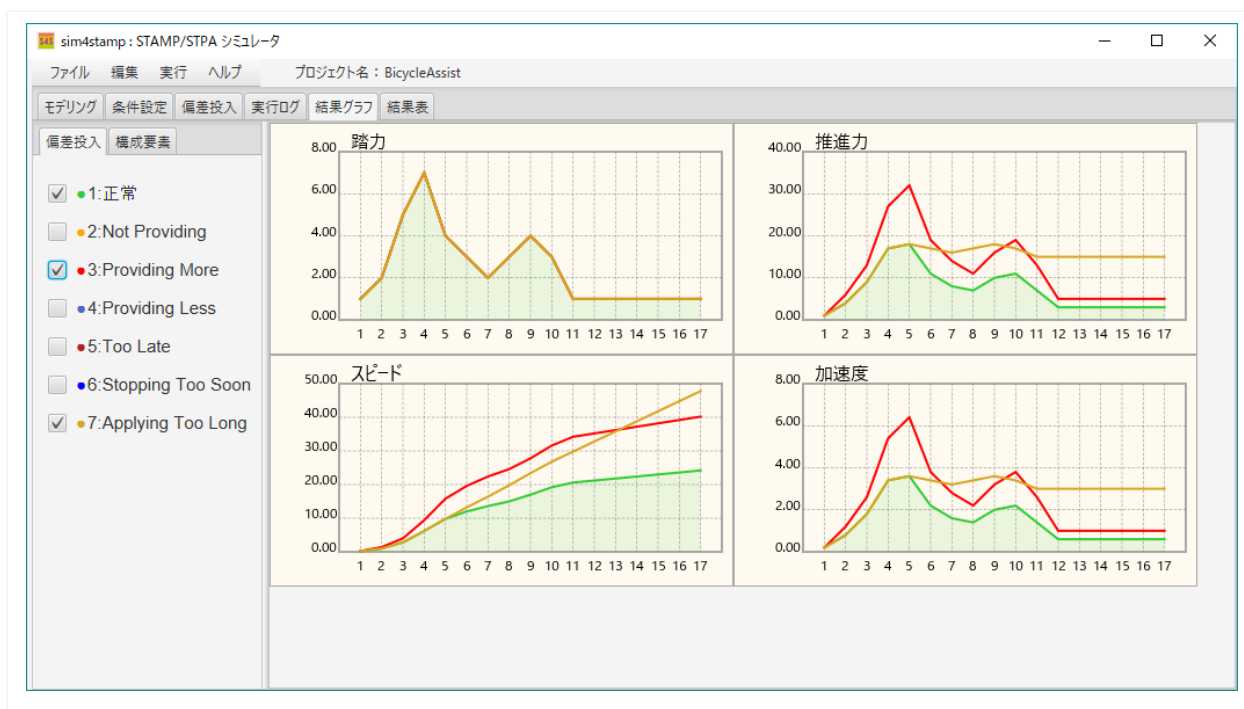
初回のデータは、比較用に最初のデータは面グラフで表示し、その後のデータは折れ線グラフで表示します。

また、構成要素タブのチェックマークを外すと、対応するグラフを非表示にすることが出来ます。こうすることにより、注目すべきグラフに集中できるようになります。

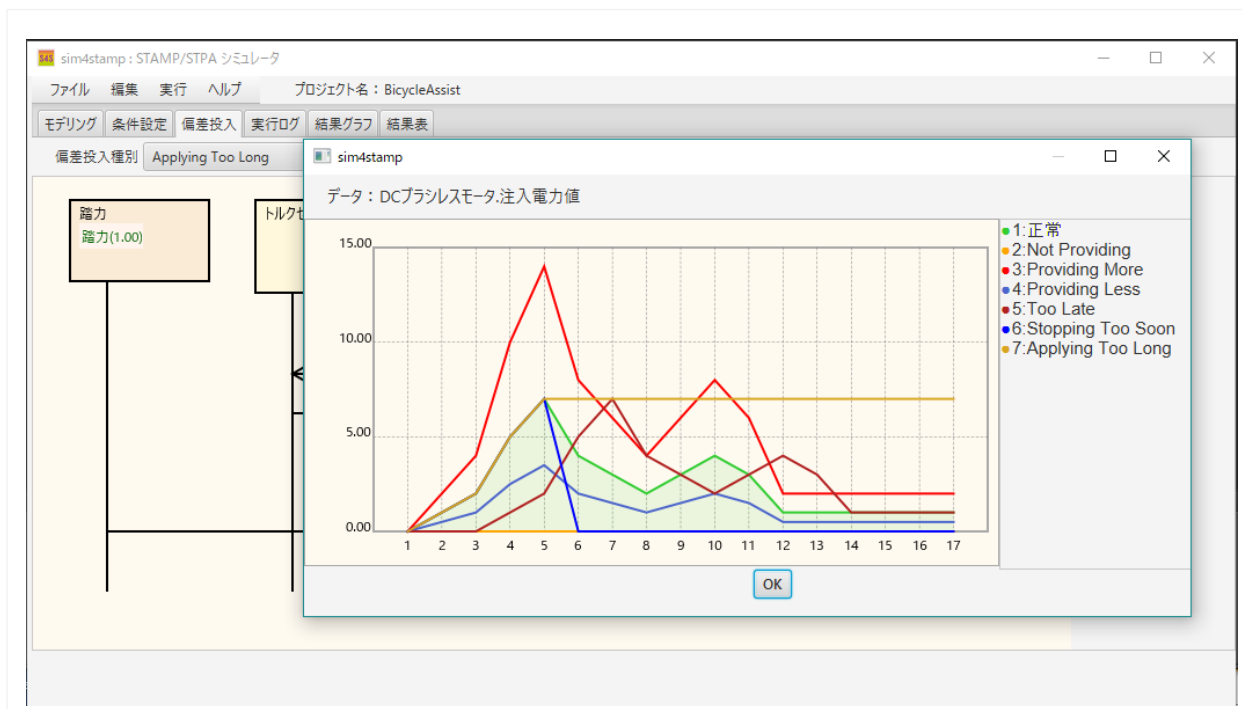


偏差投入タブで、シミュレーションした結果のケースのチェックマークを外すことにより、そのケースのデータ（折れ線グラフ）を非表示にすることが出来ます。

グラフが重なって見難い場合とき等で、任意のグラフを非表示の切り替えを行い、目的のグラフがどれなのかを確認することに使えます。



コネクタ データ グラフについても、複数のシミュレーションを行った場合、同様に重ねて表示します。



13.2. 表形式データ表示

シミュレーション結果を表で見るには、「結果表」タブをクリックします。この表には、シミュレーションを行っているすべてのデータを表示します。

No.	推進力	スピード	加速度	トルク値	モータートルク	注入踏力	トルクセンサ値	注入電力値	DCブラシレスモータ	トルク値	トルクセンサ値	踏力	注入踏力
1	1.00	0.20	0.20	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00
2	4.00	1.00	0.80	2.00	2.00	2.00	1.00	1.00	1.00	2.00	2.00	1.00	2.00
3	9.00	2.80	1.80	5.00	4.00	5.00	2.00	2.00	2.00	4.00	5.00	2.00	5.00
4	17.00	6.20	3.40	7.00	10.00	7.00	5.00	5.00	5.00	10.00	7.00	5.00	7.00
5	18.00	9.80	3.60	4.00	14.00	4.00	7.00	7.00	7.00	14.00	4.00	7.00	4.00
6	11.00	12.00	2.20	3.00	8.00	3.00	4.00	4.00	4.00	8.00	3.00	4.00	3.00
7	8.00	13.60	1.60	2.00	6.00	2.00	3.00	3.00	3.00	6.00	2.00	3.00	2.00
8	7.00	15.00	1.40	3.00	4.00	3.00	2.00	2.00	2.00	4.00	3.00	2.00	3.00
9	10.00	17.00	2.00	4.00	6.00	4.00	3.00	3.00	3.00	6.00	4.00	3.00	4.00
10	11.00	19.20	2.20	3.00	8.00	3.00	4.00	4.00	4.00	8.00	3.00	4.00	3.00
11	7.00	20.60	1.40	1.00	6.00	1.00	3.00	3.00	3.00	6.00	1.00	3.00	1.00
12	3.00	21.20	0.60	1.00	2.00	1.00	1.00	1.00	1.00	2.00	1.00	1.00	1.00
13	3.00	21.80	0.60	1.00	2.00	1.00	1.00	1.00	1.00	2.00	1.00	1.00	1.00
14	3.00	22.40	0.60	1.00	2.00	1.00	1.00	1.00	1.00	2.00	1.00	1.00	1.00
15	3.00	23.00	0.60	1.00	2.00	1.00	1.00	1.00	1.00	2.00	1.00	1.00	1.00
16	3.00	23.60	0.60	1.00	2.00	1.00	1.00	1.00	1.00	2.00	1.00	1.00	1.00
17	3.00	24.20	0.60	1.00	2.00	1.00	1.00	1.00	1.00	2.00	1.00	1.00	1.00

複数のシミュレーション結果を切り替え表示するには、左上側のプルダウンメニューにより行ないます。

No.	加速度	トルク値	モータートルク	注入踏力	トルクセンサ値	注入電力値
1	0.20	1.00	0.00	1.00	0.00	0.0
2	0.80	2.00	2.00	2.00	1.00	1.0
3	1.80	5.00	4.00	5.00	2.00	2.0
4	3.40	7.00	10.00	7.00	5.00	5.0
5	3.60	4.00	14.00	4.00	7.00	7.0
6	17.00	13.20	3.40	3.00	14.00	4.0
7	16.00	16.40	3.20	2.00	14.00	2.0
8	17.00	19.80	3.40	3.00	14.00	2.0
9	18.00	23.40	3.60	4.00	14.00	3.0

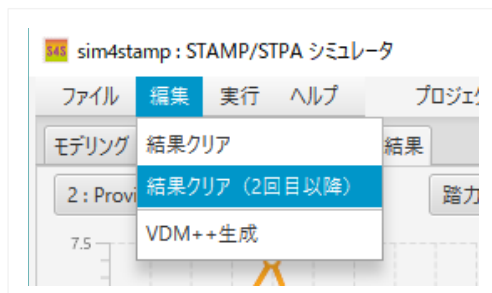
13.3. シミュレーション結果の累積、クリア

シミュレーション実行する毎に、シミュレーションデータが追加され、累積し、それらのデータの比較ができるようになります。

しかし、不要な累積はグラフを見難くするだけです。

その場合、累積データのクリアを行います。

クリアは「編集」－「結果クリア」をクリックします。

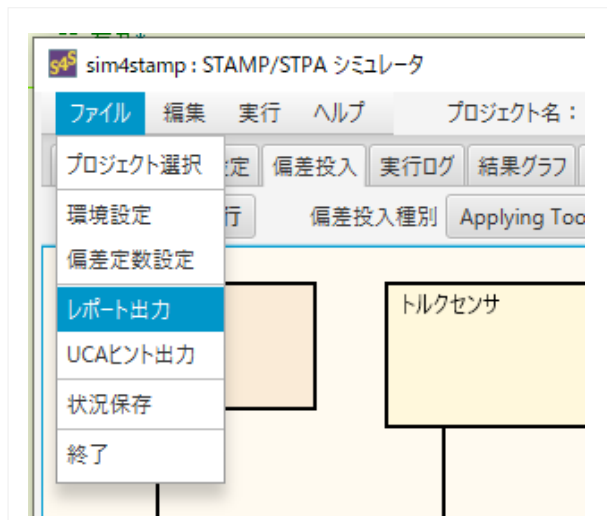


グラフ及び結果の表のデータは、シミュレーションの世代を累積保存していて重ねまたは切り替え表示可能になっていますが、2 世代以降のみをクリアすることが出来ます（上図）。第一世代を偏差投入なしのデータとし、2 世代以降を偏差投入世代にすることにより、試行錯誤を行う上で、正常データを最初に実行する必要がなくなります。

14. 実行結果の外部出力

14.1. レポート出力

シミュレーション結果を外部で利用するために、シミュレーション結果について、CSV ファイル形式でファイル出力が可能です。



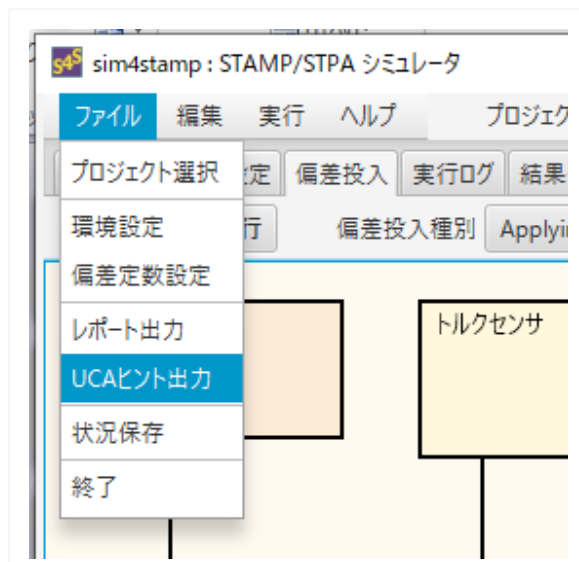
以下に示すように、偏差投入毎の塊で、CSV 出力されます。

s4s_result_20190818_091147.csv		10	20	30	40	50	60	70	80	90	100	110
1	正常,構成要素,温度設定,温度設定,外気温,外気温,温度センサー,温度センサー,室温制御装置,室温制御装置,室温制御装置,室温制御装置,エアコン,エアコン,エアコン,エアコン,エアコン,部屋,部屋,部屋,部屋											
2	パラメータ,設定温度,目標温度,外気温,外気温,計測温度,室温,目標温度,投入電力,冷房指定,計測温度,消費電力量,冷暖SW,エアコン風,投入電力,冷房指定,室内温度,温度差,外気温,エアコン風,室温											
3	1,3,0,3,0,1,0,1,0,1,0,3,0,1,0,0,false,3,0,1,0,true,1,0,1,0,false,1,0,-2.00,1,0,1,0,1,0											
4	2,3,0,3,0,1,0,1,0,1,0,1,0,3,0,5,0,false,1,0,5,0,true,5,0,5,0,false,1,9,4,0,0,1,0,5,0,1,9											
5	3,3,0,3,0,1,0,1,0,1,0,1,9,2,7,3,0,4,0,false,1,9,4,0,true,4,0,4,0,false,2,7,2,0,7,1,0,4,0,2,7											
6	4,3,0,3,0,1,0,1,0,2,7,2,6,3,0,1,0,false,2,7,1,0,true,1,0,1,0,false,2,6,-1.66,1,0,1,0,2,6											
7	5,3,0,3,0,1,0,1,0,2,6,2,9,3,0,2,0,false,2,6,2,0,true,2,0,2,0,false,2,9,-0.59,1,0,2,0,2,9											
8	6,3,0,3,0,1,0,1,0,2,9,2,9,3,0,1,0,false,2,9,1,0,true,1,0,1,0,false,2,9,-1.82,1,0,1,0,2,9											
9	7,3,0,3,0,1,0,1,0,2,9,2,8,3,0,1,0,false,2,9,1,0,true,1,0,1,0,false,2,8,-1.85,1,0,1,0,2,8											
10	8,3,0,3,0,1,0,1,0,2,8,2,7,3,0,1,0,false,2,8,1,0,true,1,0,1,0,false,2,7,-1.78,1,0,1,0,2,7											
11	9,3,0,3,0,1,0,1,0,2,7,2,6,3,0,1,0,false,2,7,1,0,true,1,0,1,0,false,2,6,-1.71,1,0,1,0,2,6											
12	10,3,0,3,0,1,0,1,0,2,6,2,6,3,0,1,0,false,2,6,1,0,true,1,0,1,0,false,2,6,-1.64,1,0,1,0,2,6											
13	11,3,0,3,0,1,0,1,0,2,6,2,9,3,0,2,0,false,2,6,2,0,true,2,0,2,0,false,2,9,-0.57,1,0,2,0,2,9											
14	12,3,0,3,0,1,0,1,0,2,9,2,8,3,0,1,0,false,2,9,1,0,true,1,0,1,0,false,2,8,-1.90,1,0,1,0,2,8											
15	13,3,0,3,0,1,0,1,0,2,8,2,8,3,0,1,0,false,2,8,1,0,true,1,0,1,0,false,2,8,-1.83,1,0,1,0,2,8											
16	14,3,0,3,0,1,0,1,0,2,8,2,7,3,0,1,0,false,2,8,1,0,true,1,0,1,0,false,2,7,-1.76,1,0,1,0,2,7											
17	15,3,0,3,0,1,0,1,0,2,7,2,6,3,0,1,0,false,2,7,1,0,true,1,0,1,0,false,2,6,-1.69,1,0,1,0,2,6											
18	16,3,0,3,0,1,0,1,0,2,6,3,0,3,0,2,0,false,2,6,2,0,true,2,0,2,0,false,3,0,-0.82,1,0,2,0,3,0											
19	17,3,0,3,0,1,0,1,0,3,0,2,9,3,0,1,0,false,3,0,1,0,true,1,0,1,0,false,2,9,-1.05,1,0,1,0,2,9											
20	18,3,0,3,0,1,0,1,0,2,9,2,8,3,0,1,0,false,2,9,1,0,true,1,0,1,0,false,2,8,-1.88,1,0,1,0,2,8											
21	19,3,0,3,0,1,0,1,0,2,8,2,7,3,0,1,0,false,2,8,1,0,true,1,0,1,0,false,2,7,-1.81,1,0,1,0,2,7											
22	20,3,0,3,0,1,0,1,0,2,7,2,7,3,0,1,0,false,2,7,1,0,true,1,0,1,0,false,2,7,-1.74,1,0,1,0,2,7											
23	Not Providing,構成要素,温度設定,温度設定,外気温,外気温,温度センサー,温度センサー,室温制御装置,室温制御装置,室温制御装置,室温制御装置,エアコン,エアコン,エアコン,エアコン,エアコン,部屋,部屋,部屋,部屋											
24	パラメータ,設定温度,目標温度,外気温,外気温,計測温度,室温,目標温度,投入電力,冷房指定,計測温度,消費電力量,冷暖SW,エアコン風,投入電力,冷房指定,室内温度,温度差,外気温,エアコン風,室温											
25	1,3,0,3,0,1,0,1,0,1,0,3,0,1,0,0,false,3,0,1,0,true,1,0,1,0,false,1,0,-2.00,1,0,1,0,1,0											
26	2,3,0,3,0,1,0,1,0,1,0,1,0,3,0,5,0,false,1,0,1,0,true,1,0,1,0,false,1,0,0,0,1,0,1,0,1,0											
27	3,3,0,3,0,1,0,1,0,1,0,1,0,3,0,5,0,false,1,0,1,0,true,1,0,1,0,false,1,0,0,0,1,0,1,0,1,0											
28	4,3,0,3,0,1,0,1,0,1,0,1,0,3,0,5,0,false,1,0,1,0,true,1,0,1,0,false,1,0,0,0,1,0,1,0,1,0											
29	5,3,0,3,0,1,0,1,0,1,0,1,0,3,0,5,0,false,1,0,1,0,true,1,0,1,0,false,1,0,0,0,1,0,1,0,1,0											
30	6,3,0,3,0,1,0,1,0,1,0,1,0,3,0,5,0,false,1,0,1,0,true,1,0,1,0,false,1,0,0,0,1,0,1,0,1,0											

14.2. UCA ヒント出力

UCA ヒントは、シミュレーション実行を行った場合に、あらかじめ設定されている上下限値を逸脱したケースが発生した場合に、安全制約違反として CSV 形式のファイルに出力するものです。

出力する CSV 形式のファイルは、UCA の一覧に準じた表となるようにしています。



室温制御装置-エアコン_投入電力_20190919_212830.csv - Microsoft Excel

	A	B	C	D	E	F
1	No	Control Action	ヒントワードの種類	ハザードシナリオ	安全制約違反	補足
2	0	*****	Not Providing	*****	部屋 > 室内温度 下限逸脱	室温制御装置-エアコン_投入電力偏差投入
3	1	*****	Providing Less	*****	部屋 > 室内温度 下限逸脱	室温制御装置-エアコン_投入電力偏差投入
4	2	*****	Too Late	*****	部屋 > 室内温度 上限 下限逸脱	室温制御装置-エアコン_投入電力偏差投入
5	3	*****	Stopping Too Soon	*****	部屋 > 室内温度 下限逸脱	室温制御装置-エアコン_投入電力偏差投入
6	4	*****	Applying Too Long	*****	部屋 > 室内温度 上限逸脱	室温制御装置-エアコン_投入電力偏差投入
7						
8						
9						

室温制御装置-エアコン_投入電力_20190919_2128

15. Q&A

<Q> Overture 側から実行するのと Sim4stamp 側で実行するのと違いはありますか？

<A> どちらで実行しても同じです。ただし、VDM++でログ等を出している場合は、Overture で実行すると Overture 側に、Sim4stamp で実行すると Sim4stamp の実行ログ画面にログが出ます。

<Q> 実行を一時的に中断し、再開するようなことは可能ですか？

<A> Overture 側のデバック機能を用いる方法と、Sim4stamp のステップ実行を用いる方法の 2 種類があります。

<Q> シミュレーション開始時のパラメータの初期値を設定することは可能ですか？

 自動生成した VDM++に初期値の設定も同時に生成されますので、必要な値に変更することで意図する初期値を設定可能です。

付録-1 VDM++自動生成例

Sim4stamp で自動生成した VDM++のコードは以下の太字の部分です。

```

class 『コントローラ』 is subclass of 『作用素』
types

values
    アシスト係数 : real = 0.25;
    最大アシスト速度 : real = 24.0;
    通常アシスト速度 : real = 10.0;
    ds : real = 最大アシスト速度 - 通常アシスト速度;

instance variables

operations

    public 『コントローラ』 :() ==> 『コントローラ』
    『コントローラ』 () =
    (
        init("コントローラ", <コントローラ>);
        setData("電力", 0.0);
    );

    public func : () ==> ()
    func() =
        let
            トルク : real = getData("トルク"),
            時速 : real = getData("時速値")
        in
            (dcl 注入電力 : real;
                if 時速 < 通常アシスト速度 then
                (
                    注入電力 := トルク * アシスト係数;
                )
                else if 時速 < 最大アシスト速度 then
                (
                    注入電力 := トルク * アシスト係数 * (1.0 - (時速 - 通常アシスト速度)/ds);
                )
                else(
                    注入電力 := 0.0;
                );
                setData("電力", 注入電力);
            );
        );

functions

traces

end 『コントローラ』

```

初期値設定

Getter

Setter

自動生成した VDM++コードを元に、エレメントのコードを書き、完成させます。

付録-2 5 値論理について

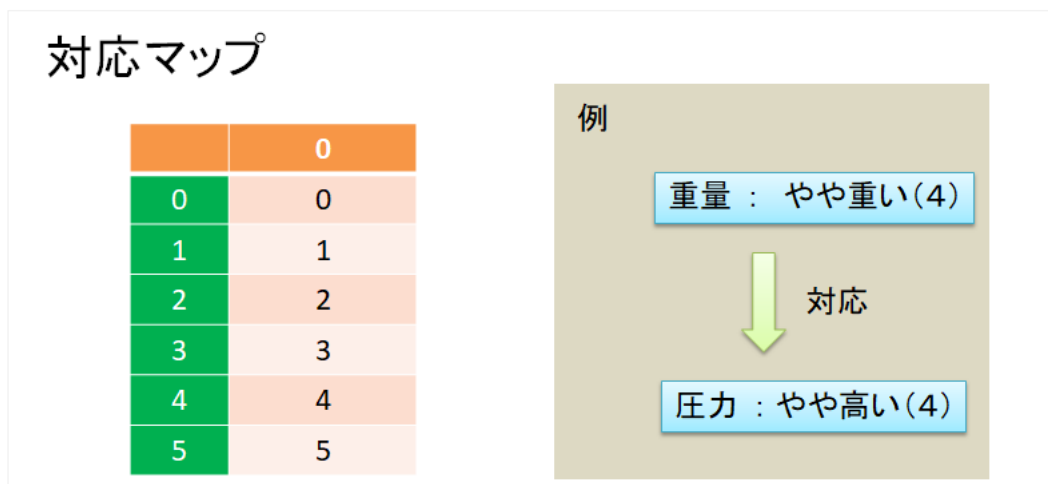
5 値論理は、「0.0」から「5.0」までの数値を以下に示すように意味を割り付け、演算、評価を行うようにしたものです。ただし、「0.0」は「不明」と言う特別な役割を持たせます。

シミュレーション演算は実数で行いますが、最終的な値の評価は得られた数値の範囲で以下の意味に割り付けています。シミュレーションの数値演算を振舞いが損なわれない範囲で簡略化したものです。

単位	0	1	2	3	4	5
評価	不明	とても悪い	悪い	普通	良い	とても良い
大きさ	不明	とても小さい	小さい	適正	大きい	とても大きい
量	不明	とても少ない	少ない	適正	多い	とても多い
力	不明	とても弱い	弱い	適正	強い	とても強い
負担	不明	とても軽い	軽い	適切	高い	とても高い
速度	不明	とても遅い	遅い	適切	早い	とても早い
回転数	不明	とても遅い	遅い	適切	早い	とても早い
温度	不明	とても低い	低い	適温	高い	とても高い
湿度	不明	とても低い	低い	適切	高い	とても高い
容量	不明	とても少ない	少ない	適量	多い	とても多い
重量	不明	とても軽い	軽い	適量	重い	とても重い
質量	不明	とても軽い	軽い	適量	重い	とても重い
圧力	不明	とても低い	低い	適圧	高い	とても高い
電圧	不明	とても低い	低い	適圧	高い	とても高い
電流	不明	とても小さい	小さい	適量	大きい	とても大きい
電力	不明	とても小さい	小さい	適量	大きい	とても大きい
電力量	不明	とても小さい	小さい	適量	大きい	とても大きい
抵抗	不明	とても小さい	小さい	適正	大きい	とても大きい
長さ	不明	とても短い	短い	適正	長い	とても長い
面積	不明	とても小さい	小さい	適正	大きい	とても大きい
体積	不明	とても小さい	小さい	適正	大きい	とても大きい
音量	不明	とても小さい	小さい	適正	大きい	とても大きい
周波数	不明	とても低い	低い	適正	高い	とても高い
風速	不明	とても弱い	弱い	適量	強い	とても強い
摩擦	不明	とても弱い	弱い	適量	強い	とても強い
距離	不明	とても遠い	遠い	適量	近い	とても近い
方向	不明	とても悪い	やや悪い	普通	良い	とても良い
ストレス	不明	とても低い	低い	適切	高い	とても高い
時間	不明	とても遅い	遅い	適切	早い	とても早い
時刻	不明	とても遅れる	遅れる	正確	早すぎる	とても早すぎる
効率	不明	とても悪い	悪い	適正	良い	とても良い
コスト	不明	とても低い	低い	適正	高い	とても高い

付録-2 5 値論理の演算パターン

5 値論理では、以下の演算パターンを定義し、VDM++側のライブラリに実装しました。



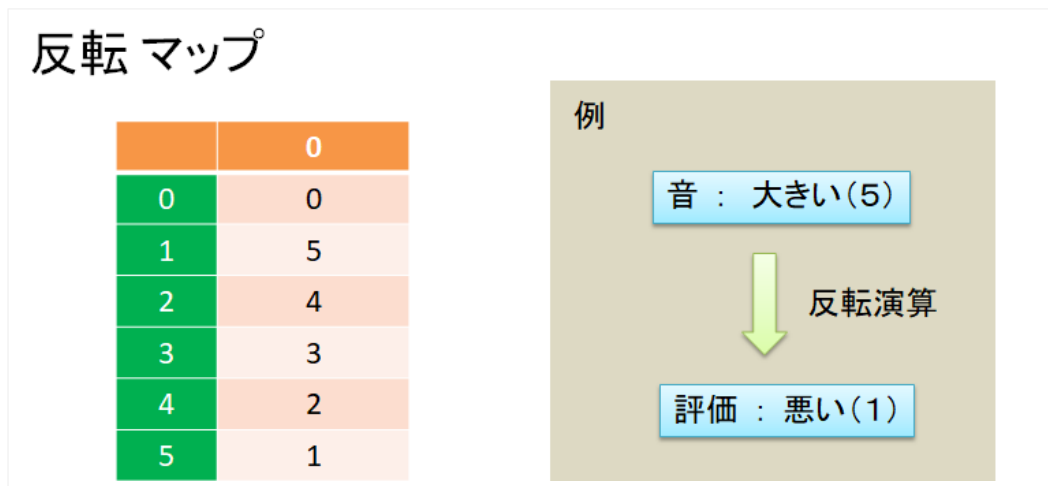
単位が異なるものを対応づけるための値のマッピングを行います。

対応マップを行っても、値には変更がありません。

VDM++ :

```
resultValue = op5Equ(value);
```

ここで、value は対応を行う値。ResultValue は演算結果となります。



反転マップは、対応を行う場合、値を反転させて対応を行うケースです。

VDM++ :

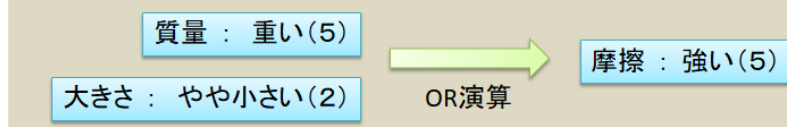
```
resultValue = op5Inv(value);
```

ここで、value は対応を行う値。ResultValue は演算結果となります。

OR マップ

	0	1	2	3	4	5
0	0	0	0	0	0	5
1	0	1	2	3	4	5
2	0	2	2	3	4	5
3	0	3	3	3	4	5
4	0	4	4	4	4	5
5	5	5	5	5	5	5

例：



OR マップは、2つの値の OR を取る演算を定義します。

VDM++：

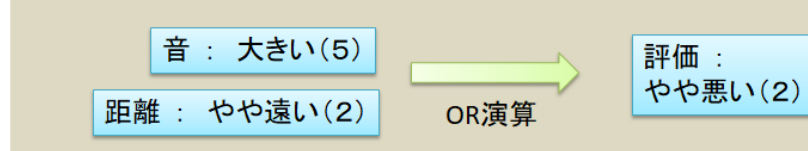
```
resultValue = op5OR(value1, value2);
```

ここで、value1 と value2 は演算に与える前の値、resultValue は演算結果となります。

AND マップ

	0	1	2	3	4	5
0	0	1	0	0	0	0
1	1	1	1	1	1	1
2	0	1	2	2	2	2
3	0	1	2	3	3	3
4	0	1	2	3	4	4
5	0	1	2	3	4	5

例：警告音と距離との関係



AND マップは、2つの値の AND を取る演算を定義します。

VDM++：

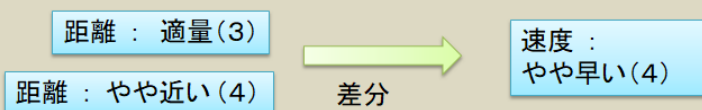
```
resultValue = op5AND(value1, value2);
```

ここで、value1 と value2 は演算に与える前の値、resultValue は演算結果となります。

差分マップ

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	3	4	4	5	5
2	0	2	3	4	4	5
3	0	2	2	3	4	5
4	0	1	1	2	3	4
5	0	1	1	2	2	3

例：距離との関係



差分マップは、2つの値の差分を取る演算を定義します。

VDM++ :

```
resultValue = op5Diff(value1, value2);
```

ここで、value1 と value2 は演算に与える前の値、resultValue は演算結果となります。

正規化

	0
0	0
$0 < x < 1.5$	1
$1.5 \leq x < 2.5$	2
$2.5 \leq x < 3.5$	3
$3.5 \leq x < 4.5$	4
$4.5 \leq x$	5

正規化は、実数で表される値を0から5の整数に対応させるときに使用します。

VDM++ :

```
resultValue = op5map(value);
```

ここで、value は実数値、resultValue は value を上記の規則に沿って整数化した場合の値です。

この他に、VDM++のライブラリには、リミッタを用意しました。
リミッタは、実数値が 1.0 以下の場合、1.0 にし、5.0 以上の場合は 5.0 にします。
この場合、「不明」を表す「0.0」の値のみ、「0.0」のままとします。
VDM++ :

```
resultValue = op5Limit(value);
```

ここで、value は実数値、resultValue は value の上下限を制限した値です。
リミッタは、ユーザ独自の演算規則を作成して演算した場合、1.0~5.0 の範囲を外れた場合、強制的に 1.0 から 5.0 の範囲内に収めるために使用する演算です。