Sim4stamp 取扱い説明書

第 1 版

2008 年 11 月

by
Keiichi Tsumuta

ドキュメント履歴

目次

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 のライ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。

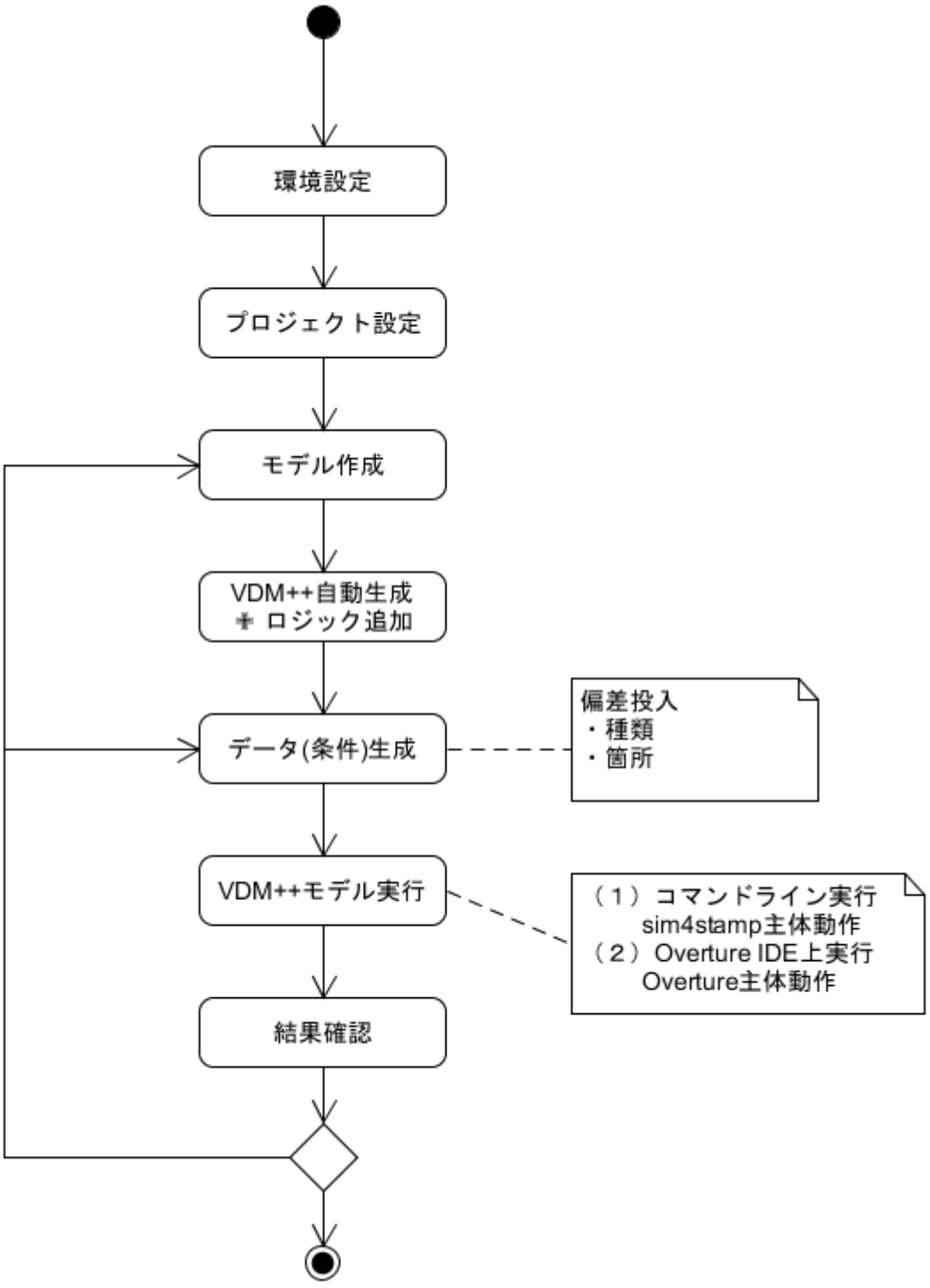
http://overturetool.org/download/

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

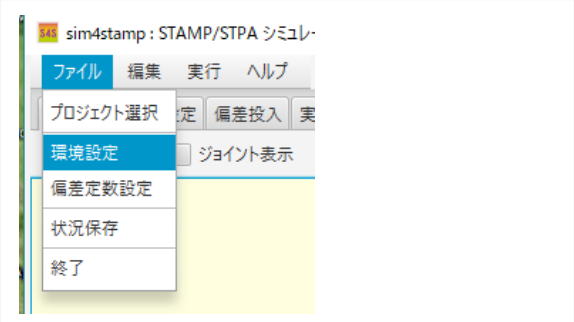
Sim4stamp を動作させるには、Sim4stamp の実行ファイルと連携して動作させる Overture が必要です。

4.1. Sim4stamp インストール

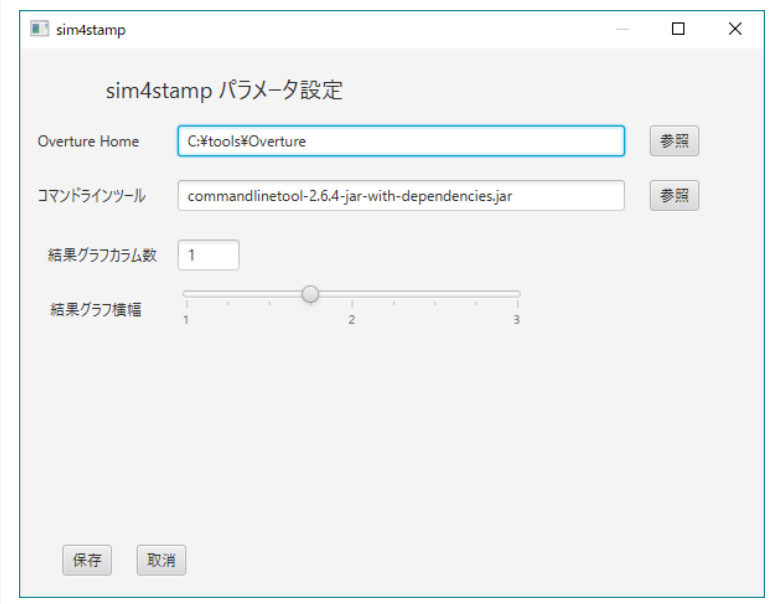
Sim4stamp の実行ファイルは、「sim4stamp-X.X.X.jar」と言うファイルにまとまっています。
そのため、「sim4stamp-X.X.X.jar」を入手し、任意のフォルダに置くことでインストールは完了します。
Sim4stamp の初回起動時、環境変数等のパラメータを保持するファイルを生成し、以降、そのファイルを使用します。

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。

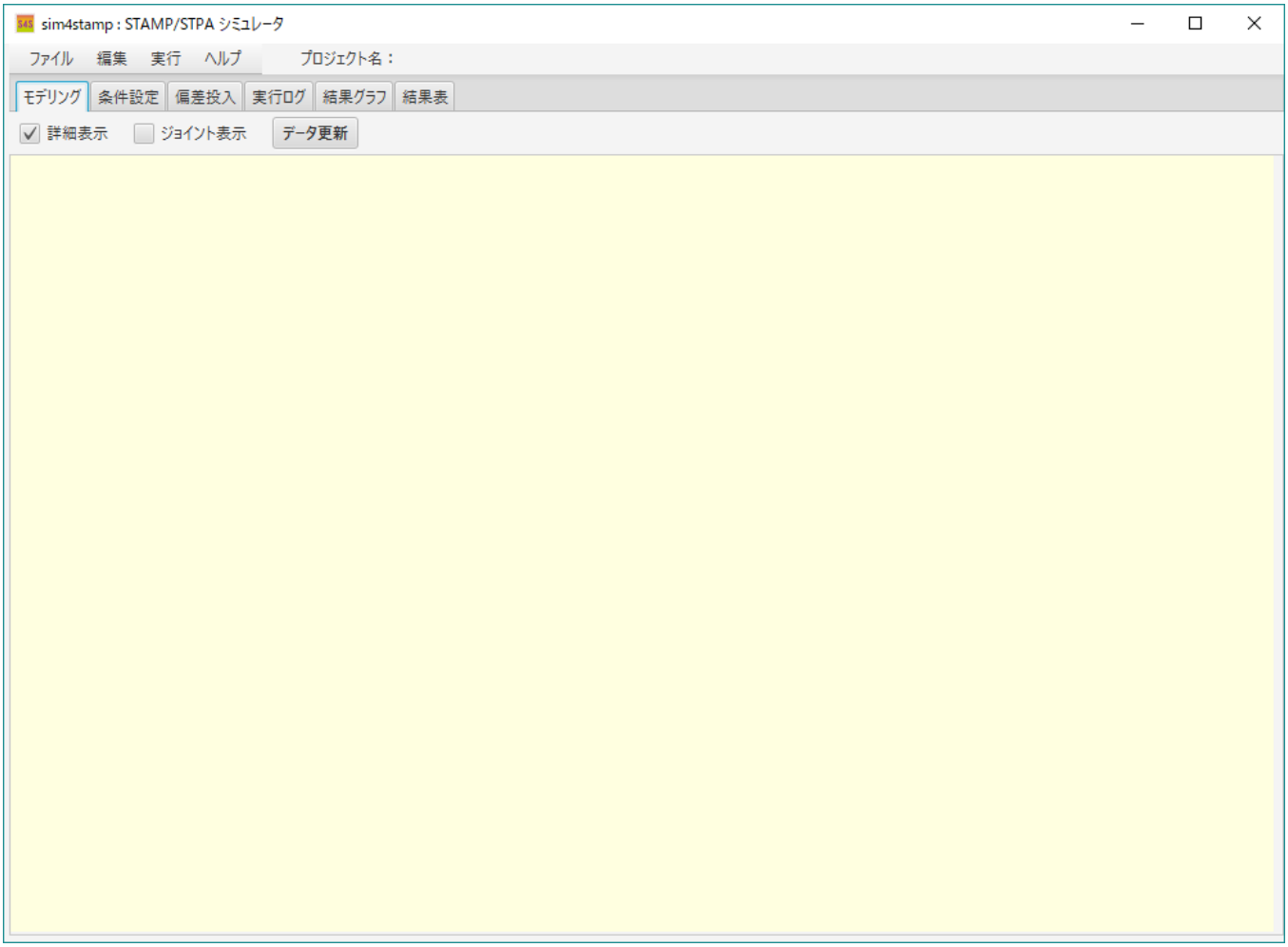
（1）「sim4stamp-X.X.X.jar」をダブルクリックする。

（2）コマンドプロンプトより

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

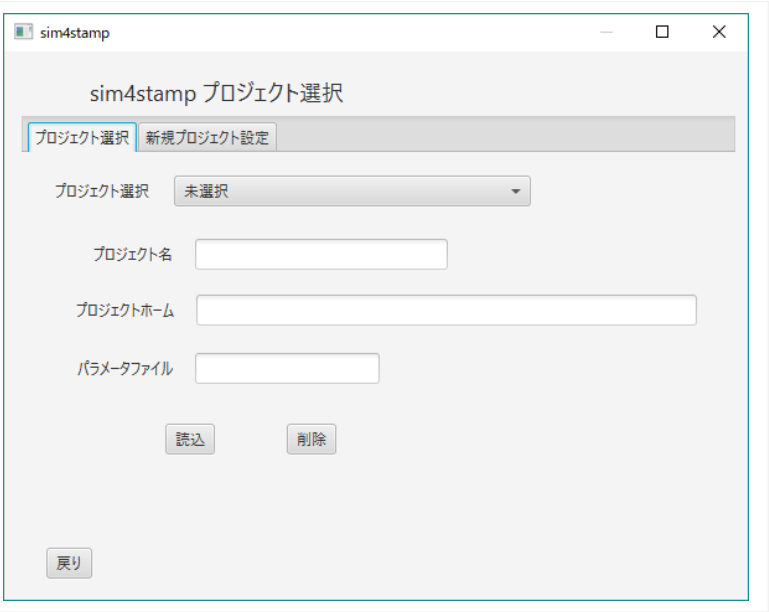
をタイプし、Enter キーを押します。

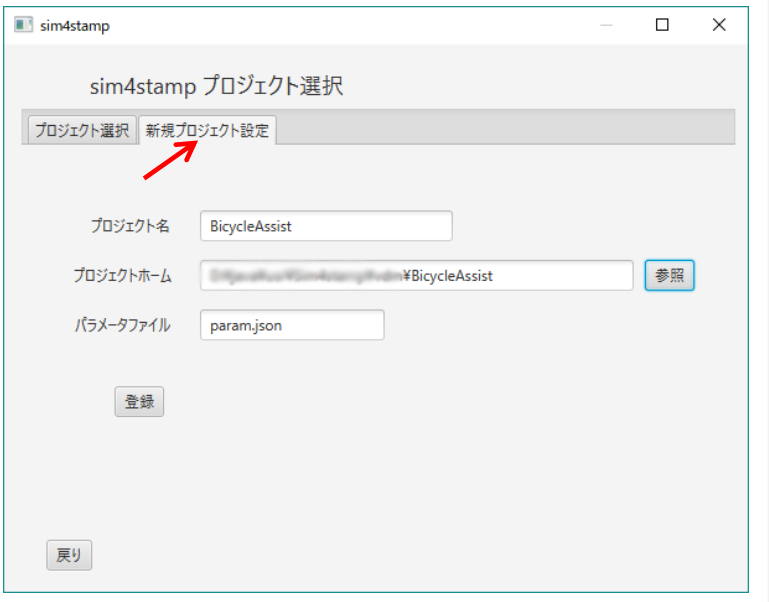
起動が成功すると以下に示す画面（インストール後の初回起動）を表示します。なお、インストール後の初回起動以外は、前回終了時のプロジェクトが開かれた状態で画面が表示されます。



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。



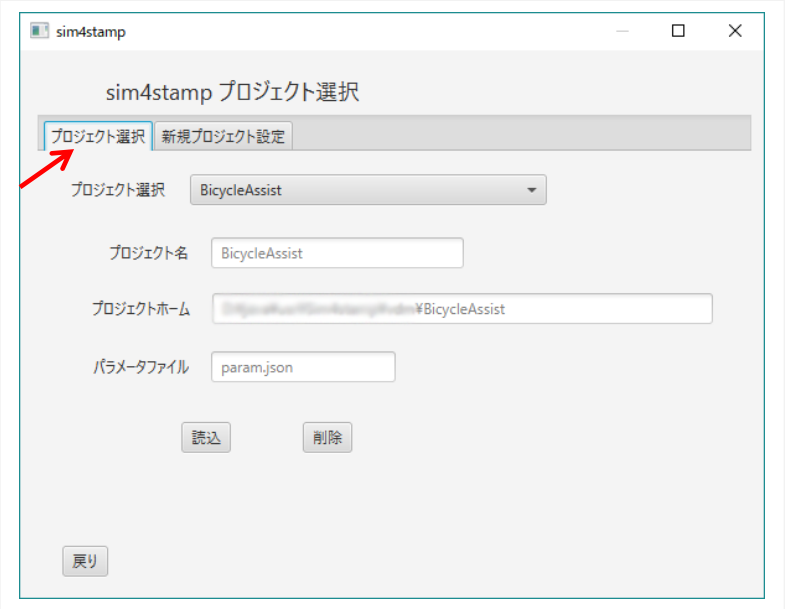


「新規プロジェクト設定」タブに移行し、Overture 側のプロジェクトの情報を設定します。

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。

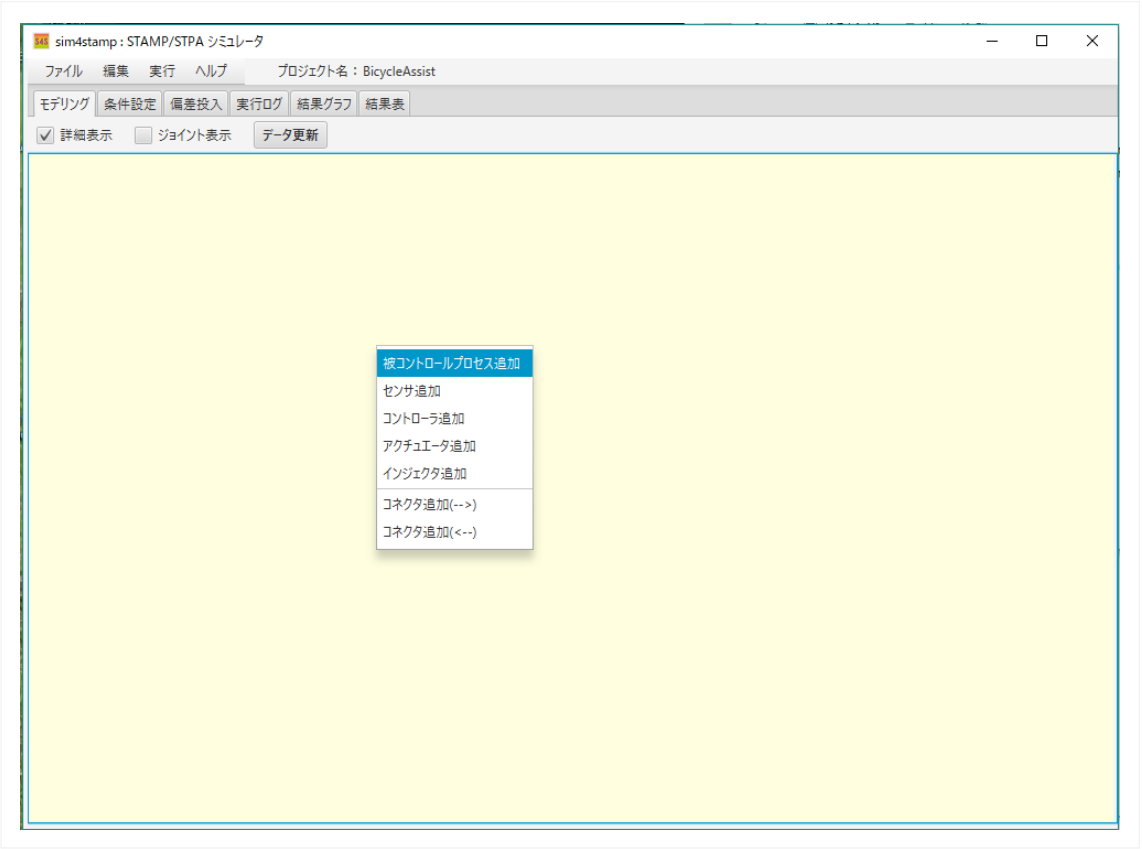
その後、「登録」ボタンをクリックし、登録を行います。



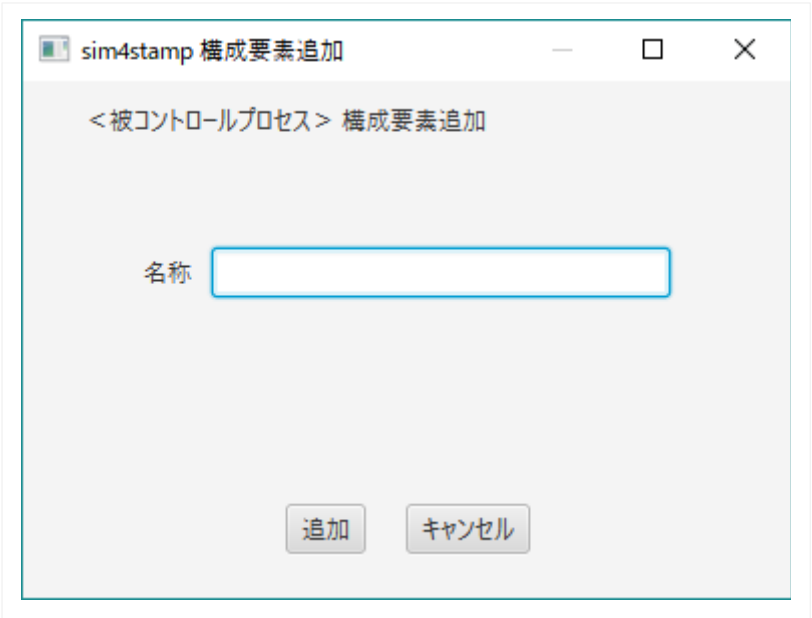
登録後、「プロジェクト選択」タブに戻り、プロジェクト選択（プルダウンメニュー）を行って、「読込」ボタンをクリックすることにより、Sim4stamp のプロジェクトが選択プロジェクトになります。

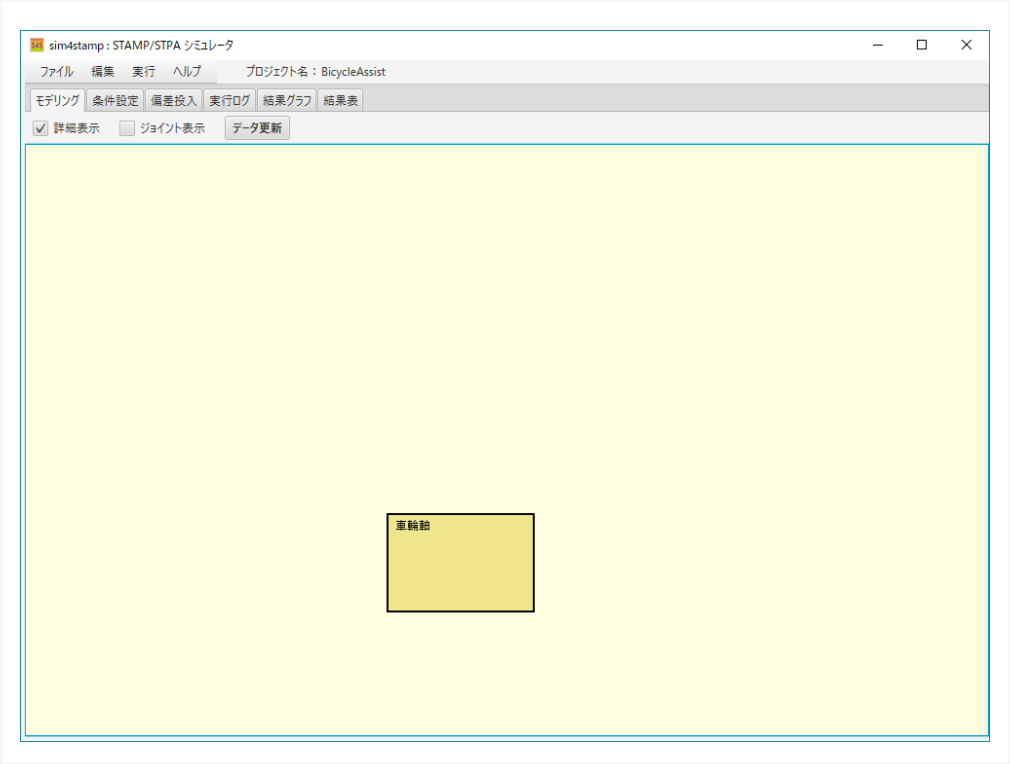
7. モデル生成

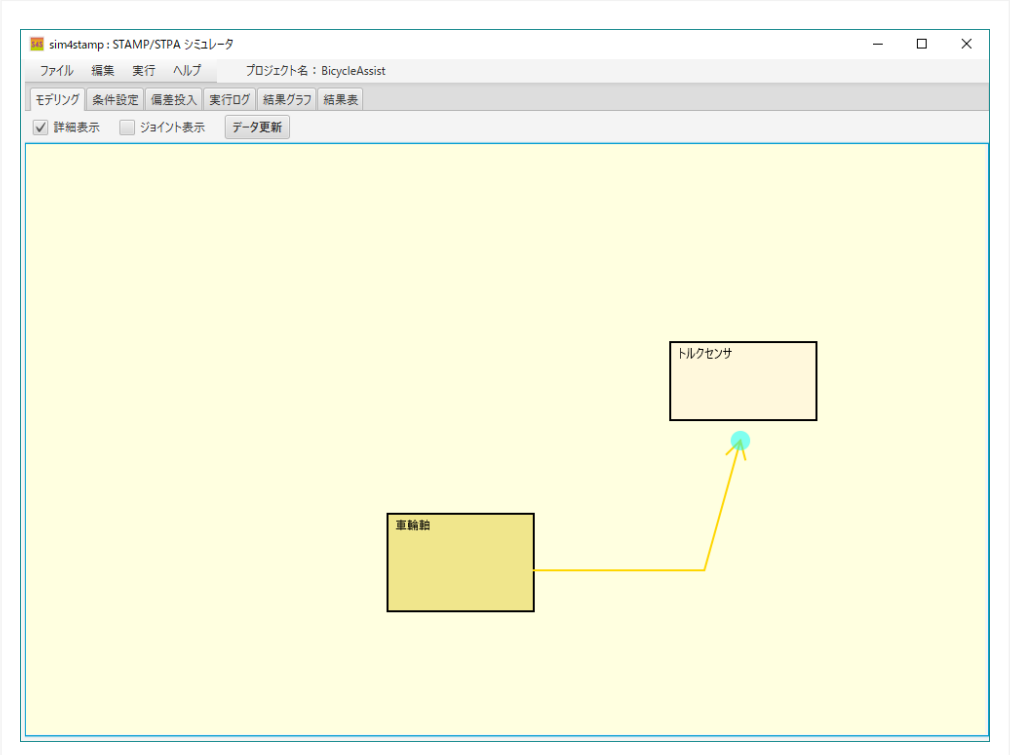
モデルの生成は、「モデリング」タブ内のモデルパネルで行います。



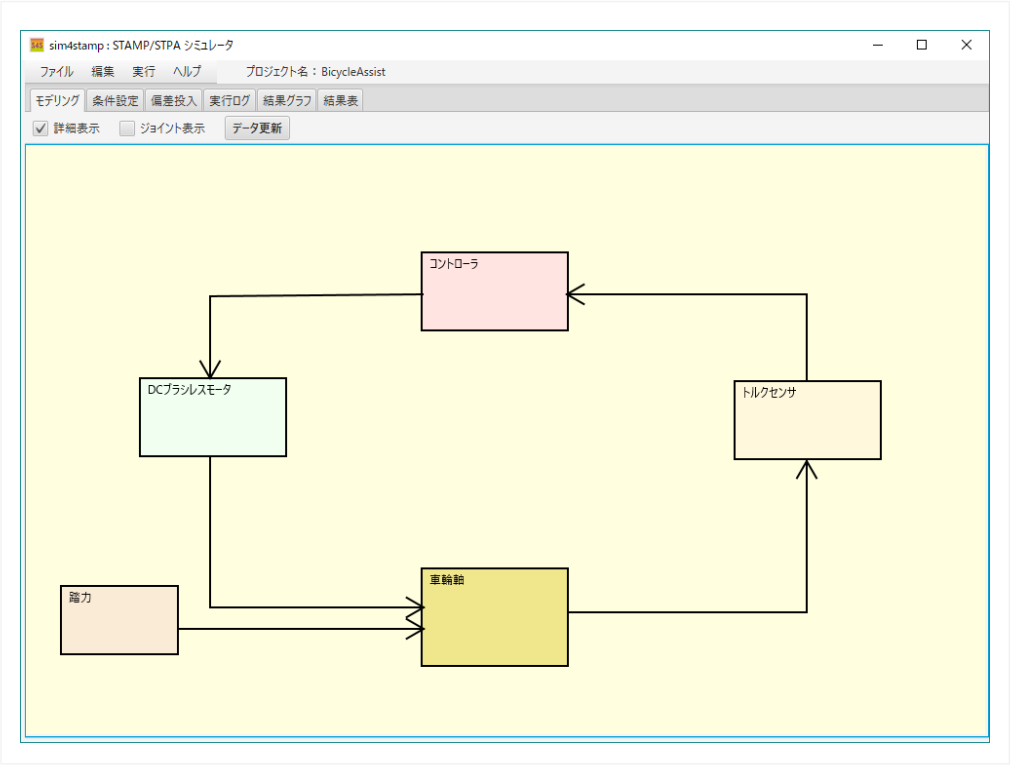
モデルパネル上で、右クリックを行うと追加する部品のリストが表示されます。
部品リストから目的の部品を選択することにより、追加部品のダイアログが表示され、必要事項を入力し、部品をモデルパネルに置きます。







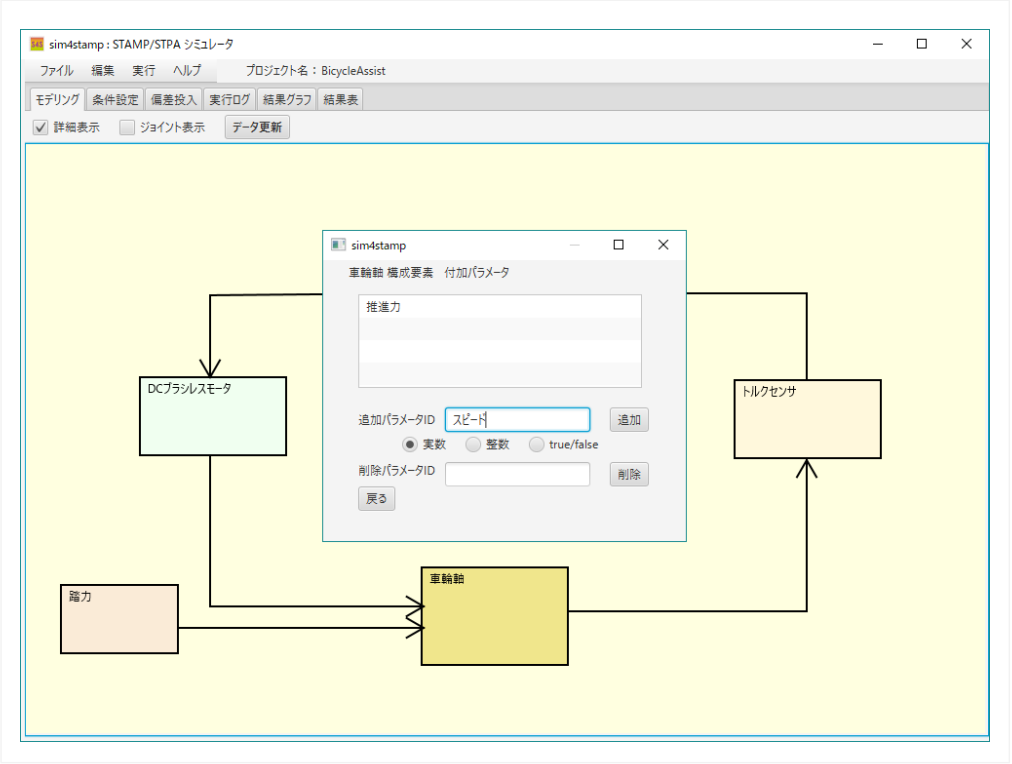
コネクタも設定し、部品要素間のデータの流れを設定します。コネクタは、両端が部品要素に接続されると色が黒に変化します。



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。

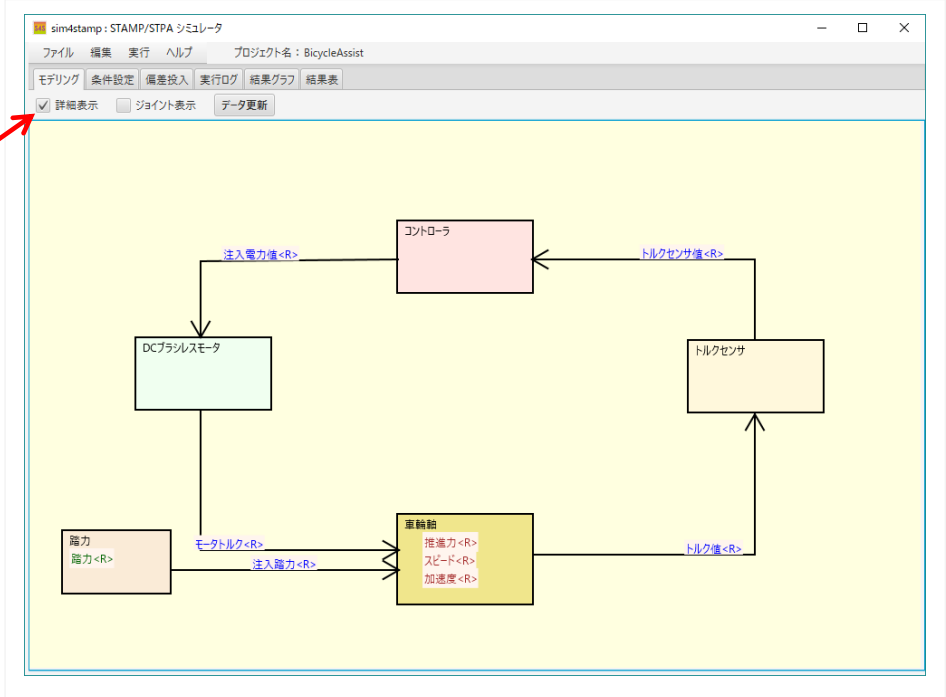
部品要素とその接続関係の設定が完了後、パラメータを設定します。

パラメータの設定は、部品要素であれは、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。

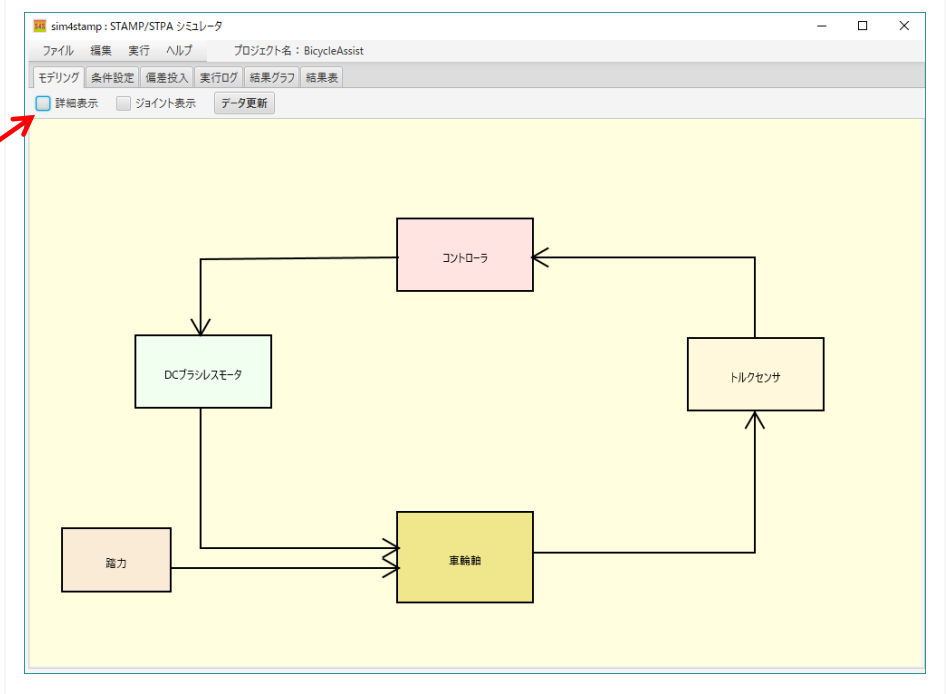


パラメータの型は、実数、整数、論理値（true/false）の 3 種類が選択可能です。

「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。

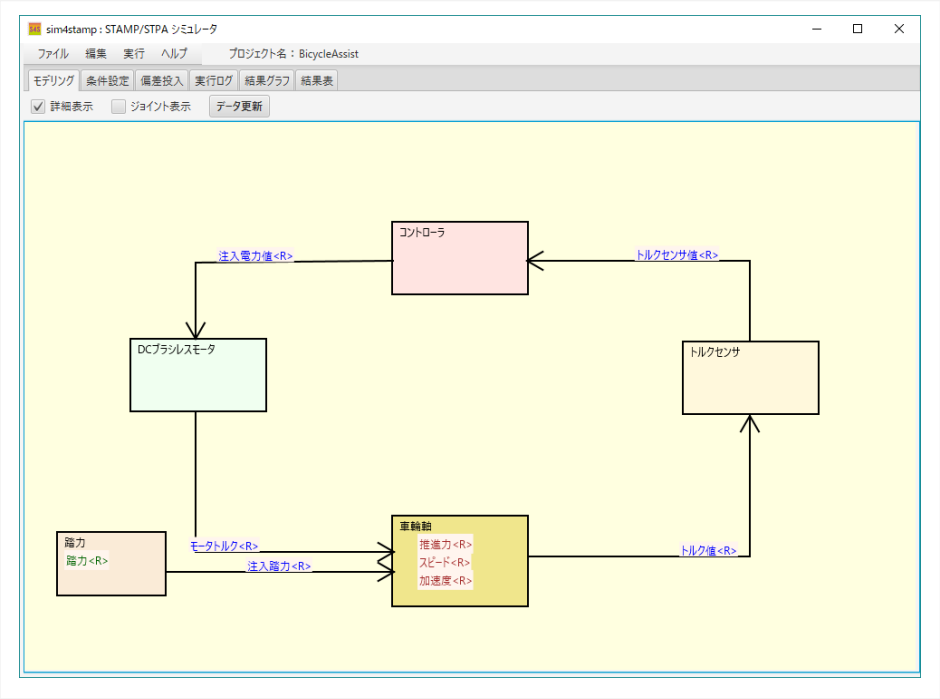


なお、コネクタに設定できるパラメータは１つのみです。複数設定する必要がある場合は、複数コネクタを設定してください。

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。

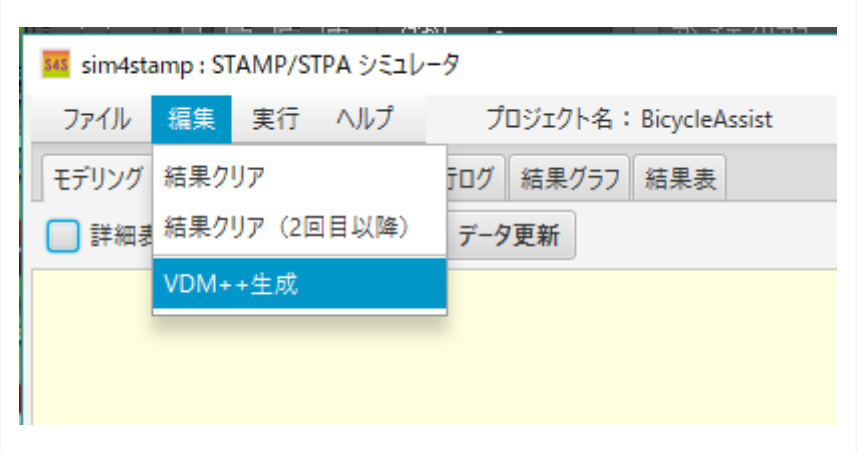
データ更新を行うことにより、次節以降で述べるデータ（設定データ）にデータの種類や設定項目が反映されます。

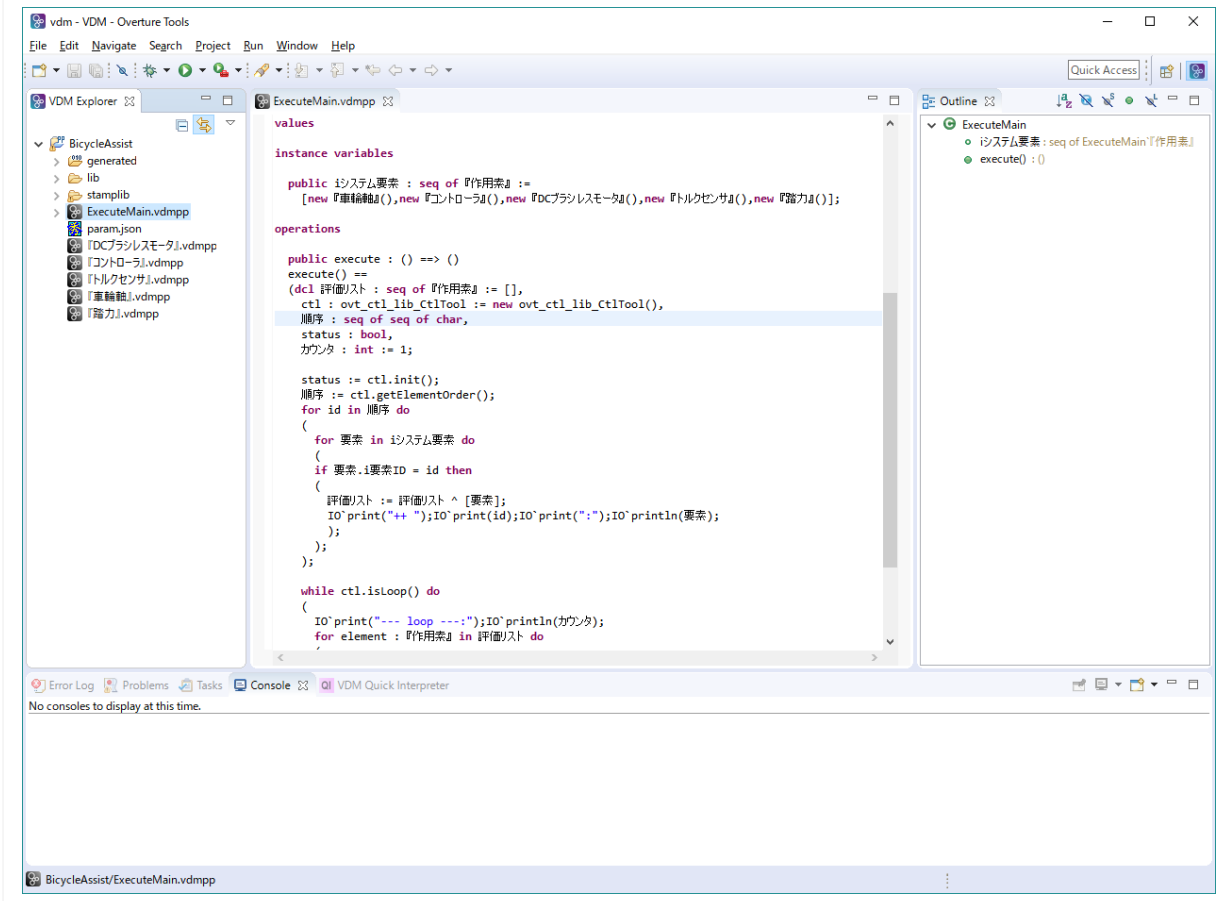
モデル図を変更した場合、モデルの単なる位置関係以外の変更は「データ変更」ボタンをクリックし、内部データの更新を行ってください。



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。





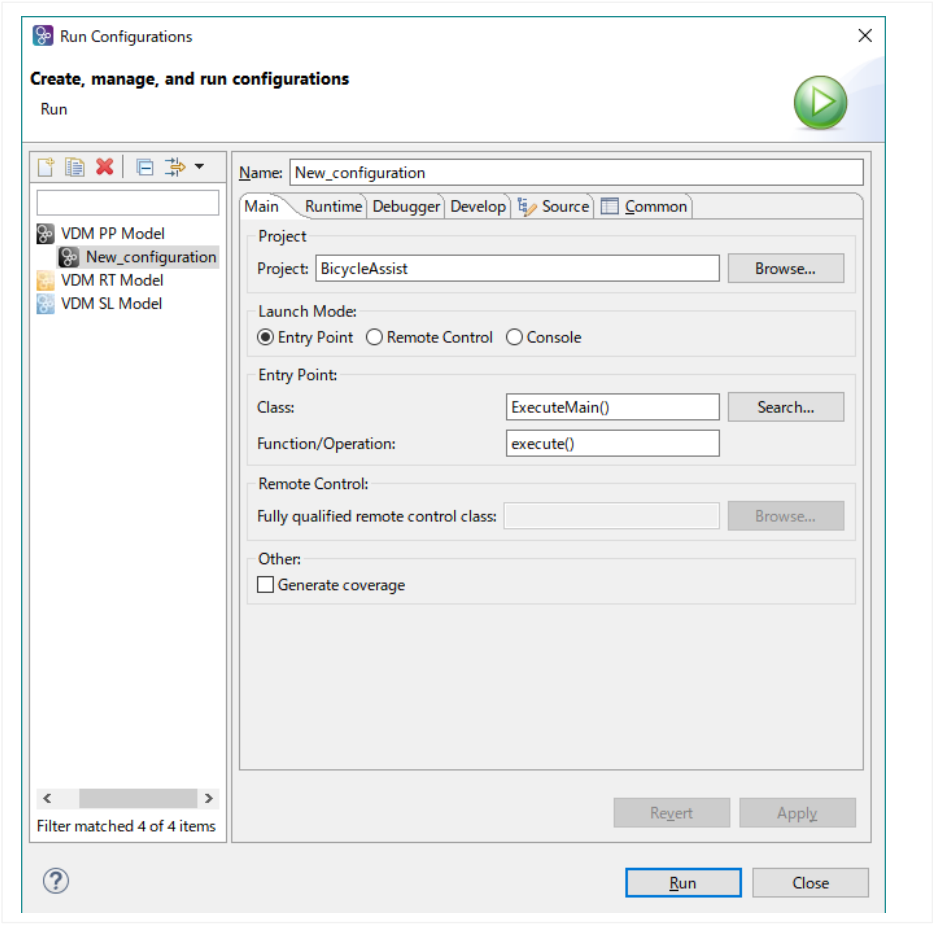
VDM++の生成は、初回は拡張子が「.vdmpp」で生成されますが、同一ファイル名がすでに存在する 2 回目以降は「.vdmpp_ini」で生成し、元のファイルを上書きしないようにします。

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

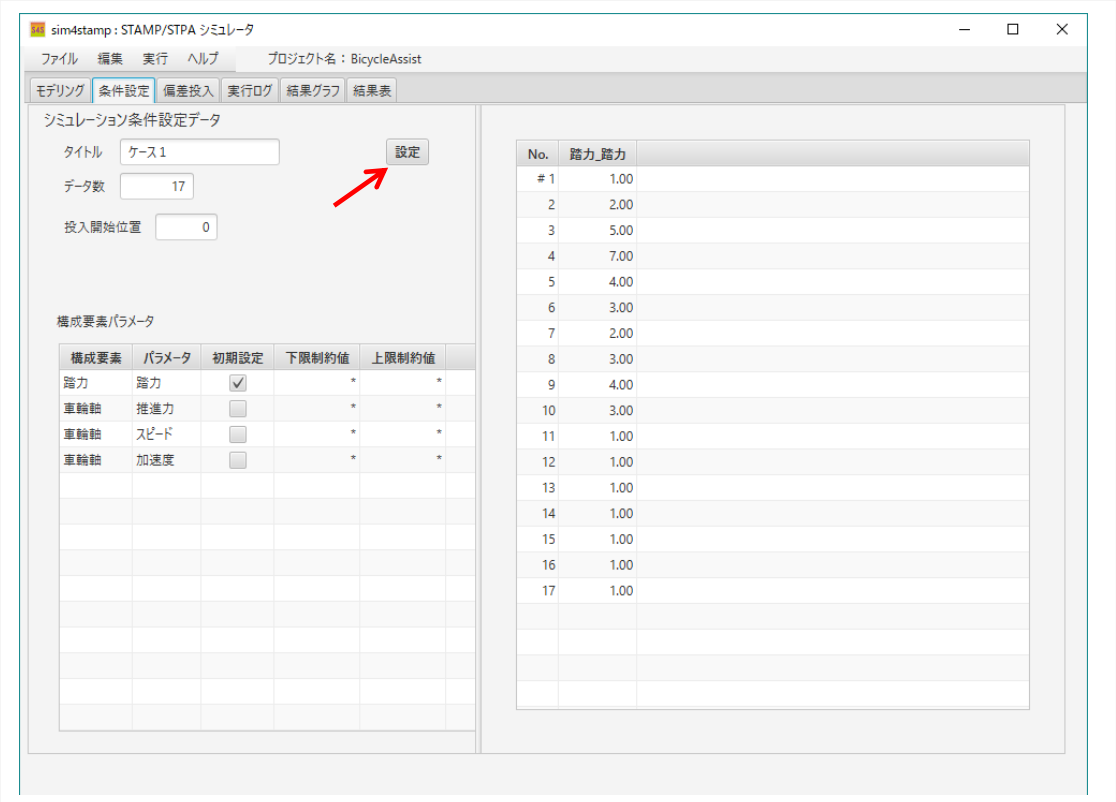
を設定します。



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。

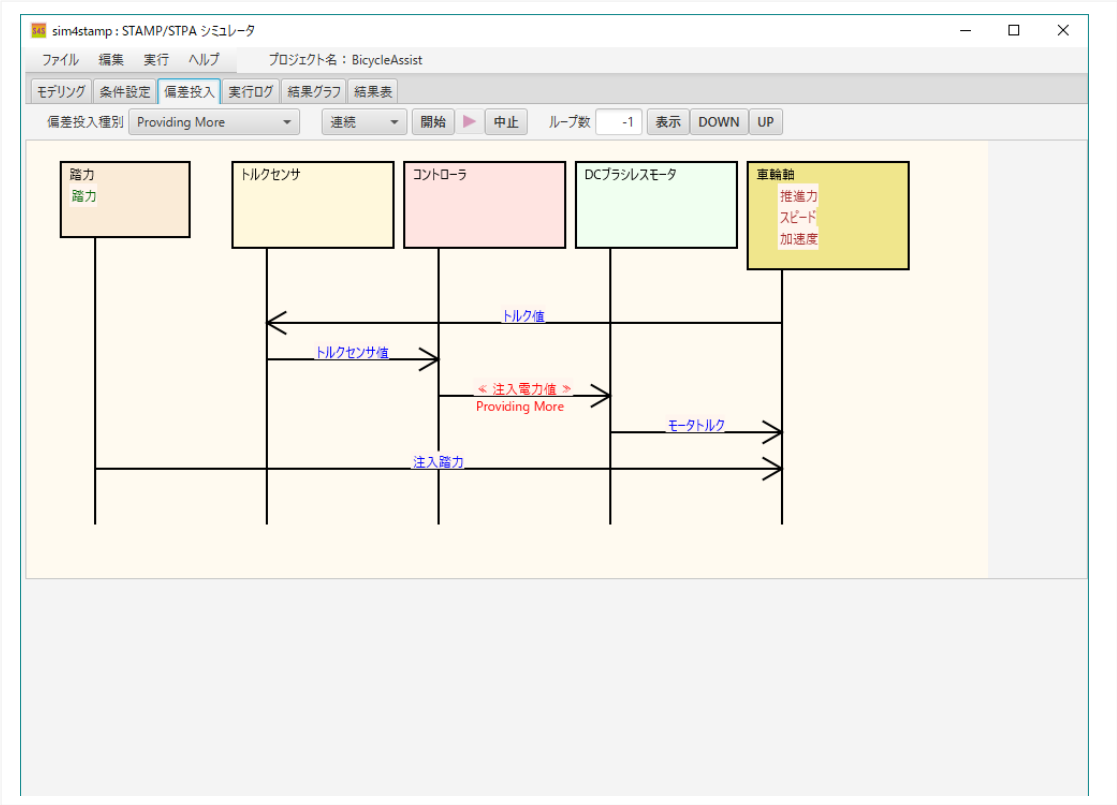
この制約値に設定を行うとグラフ表示時に制約値を同時表示し、逸脱がすぐに分かるようになります。

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。

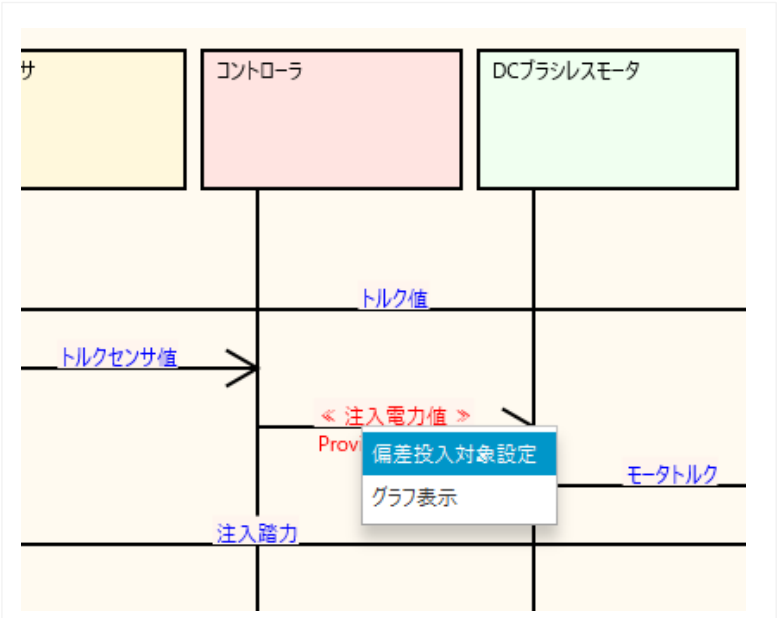
偏差投入の設定位置の用途は、電動自転車の例で言うと、自転車が走りはじめ、低速になったときに、上り坂に差し掛かるケースで、坂に差し掛かる位置を偏差投入したい場合等に使用します。

10. 偏差投入設定

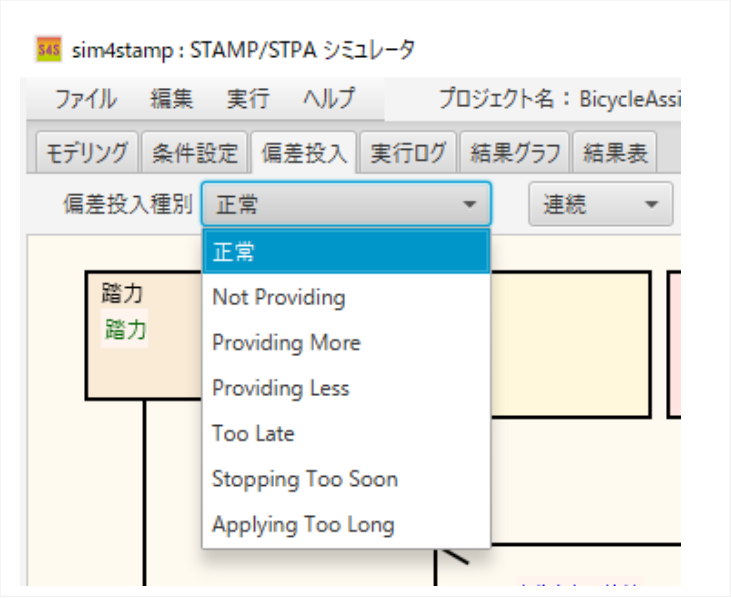
偏差投入の設定は、偏差投入タブを開き、偏差投入する場所と種類を設定することにより行います。
偏差投入の場所の設定は、本画面上で横矢印の部分（コネクタ）で行います。



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。



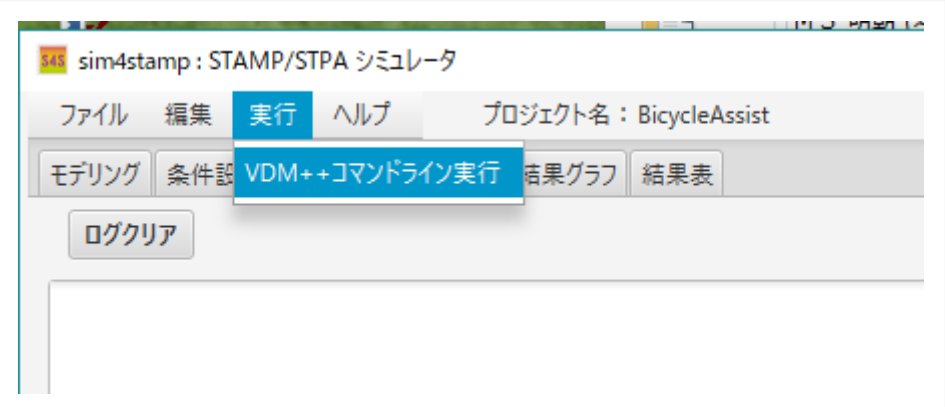
最初に「正常」ケースを実行し、その次以降に偏差を指定することにより、正常ケースと偏差投入した結果を比較することが出来ます。

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

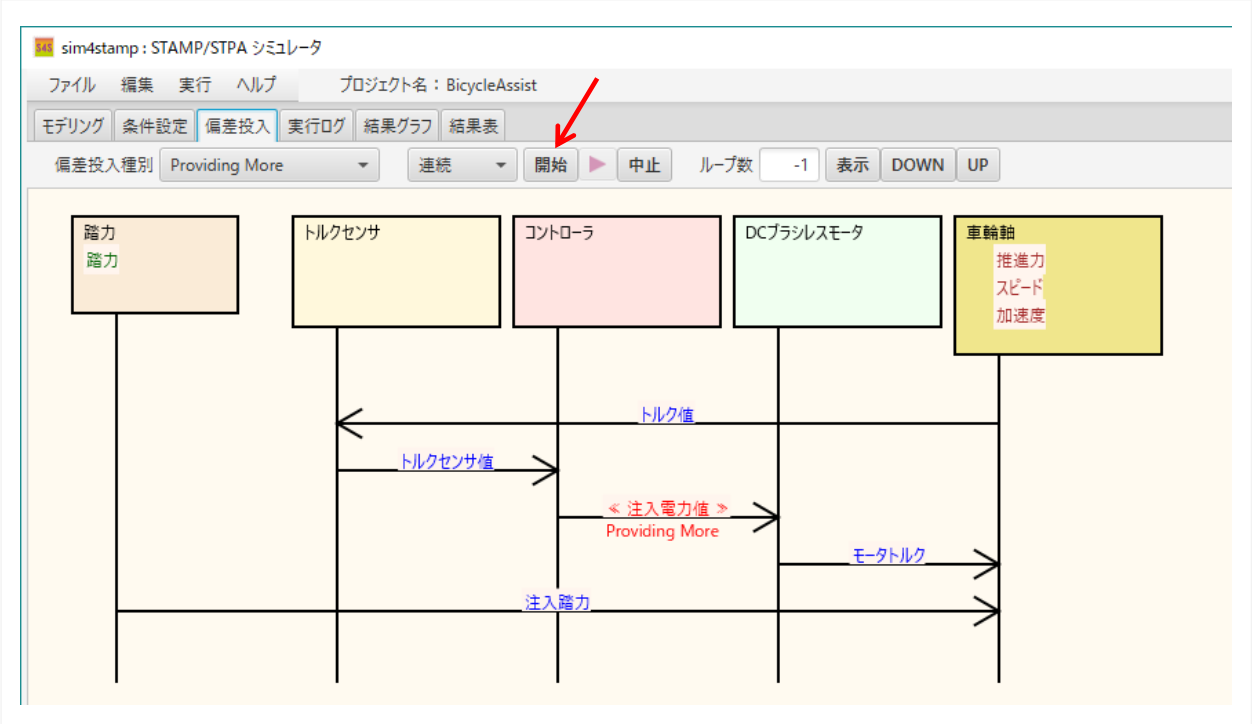
11.1. 実行方法その１

Sim4stamp 側からの VDM++の実行の一つとして、「実行」－「VDM++コマンドライン実行」をクリックする方法があります。



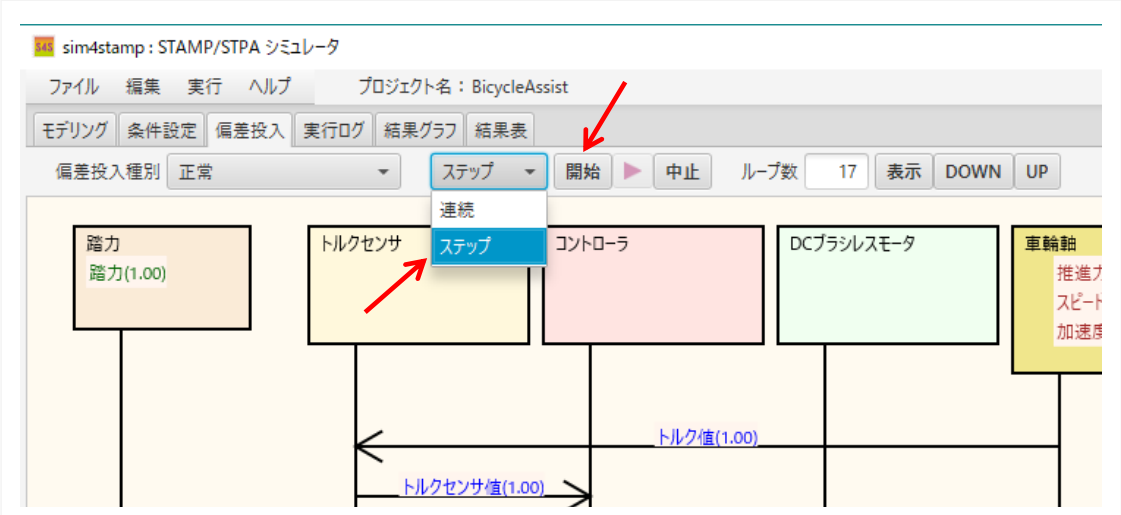
11.2. 実行方法その２

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。



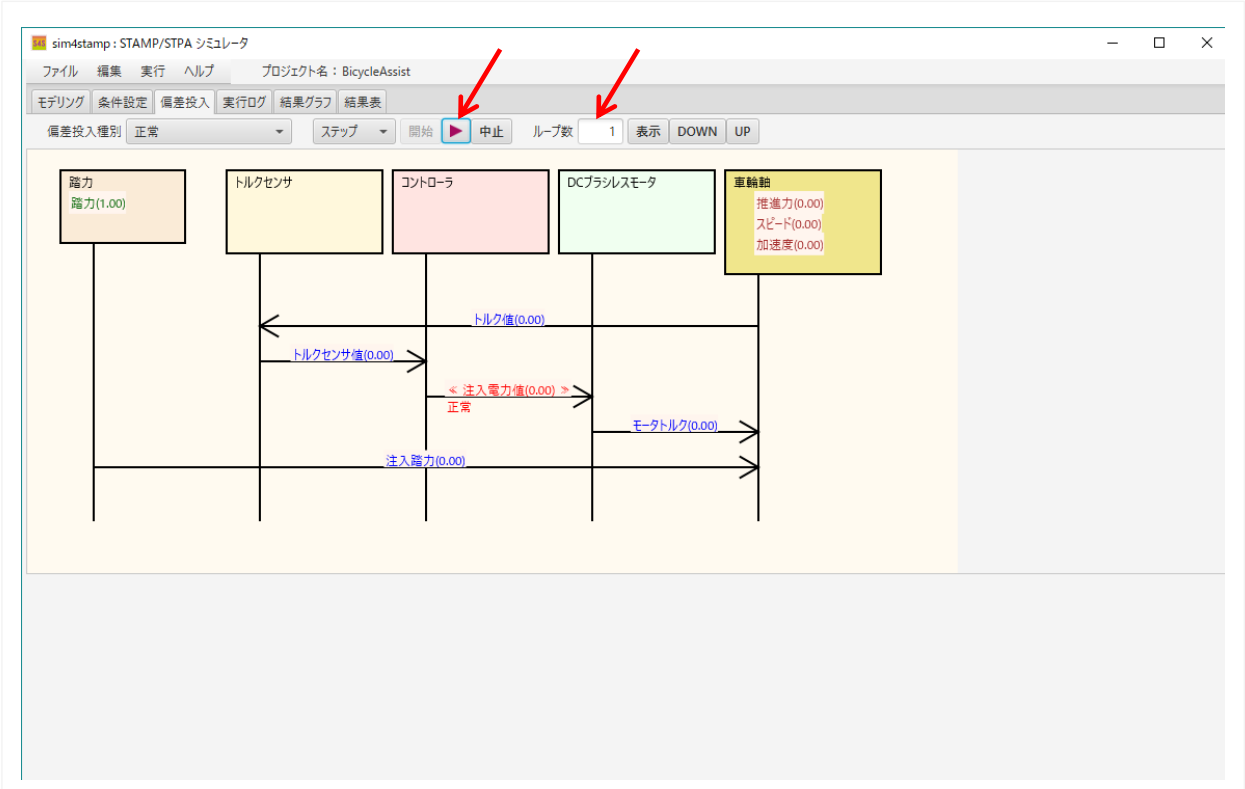
11.3. ステップ実行

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。

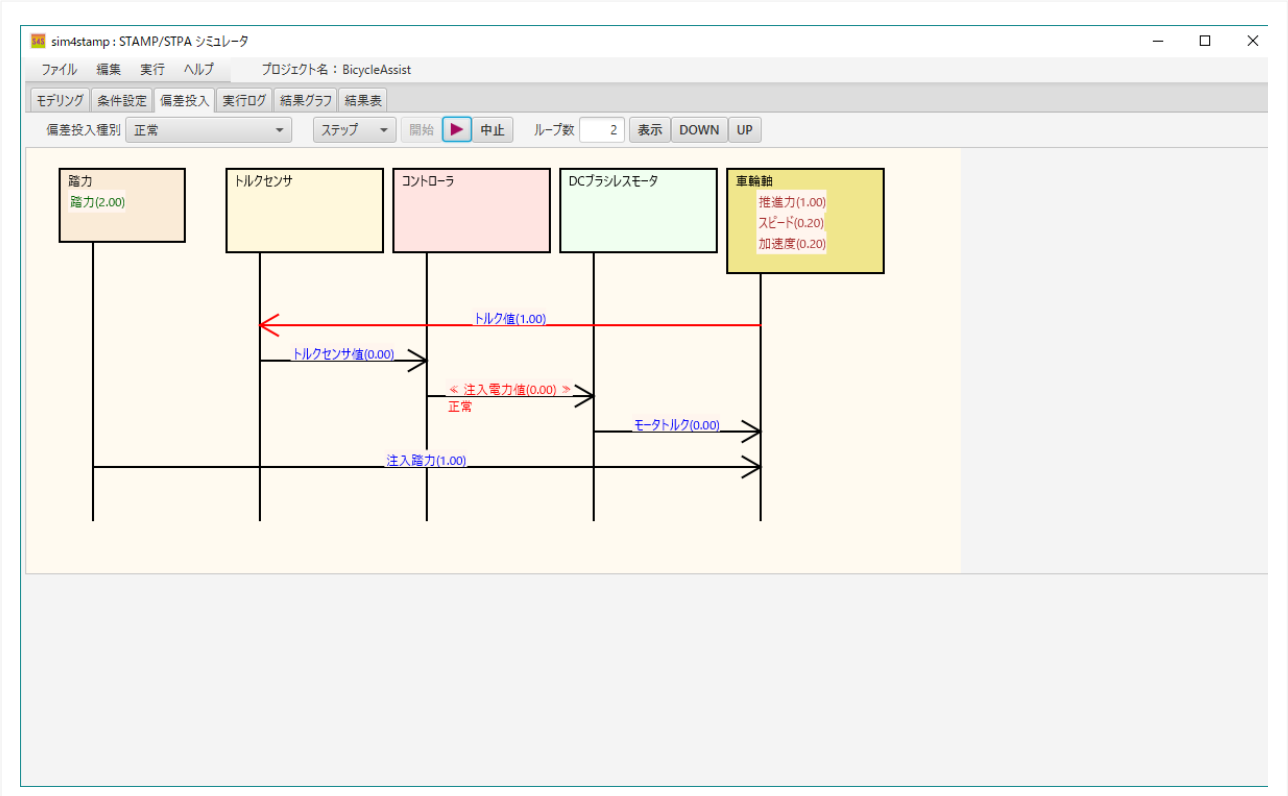


ステップ実行が開始されると開始ボタンの右隣りにある三角マークボタンがクリック可能となります。

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。



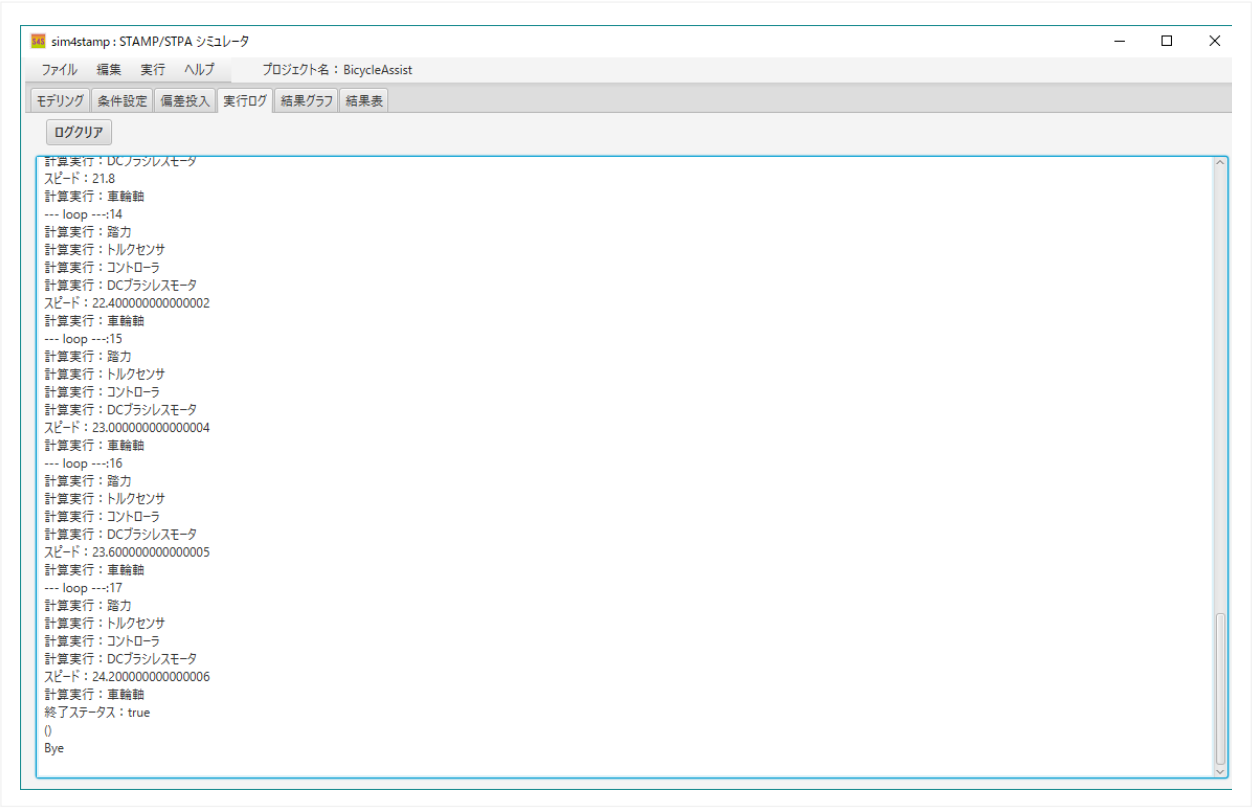
ステップ実行を行っているときには、ステップ実行中のコネクタが赤色表示されます。



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。



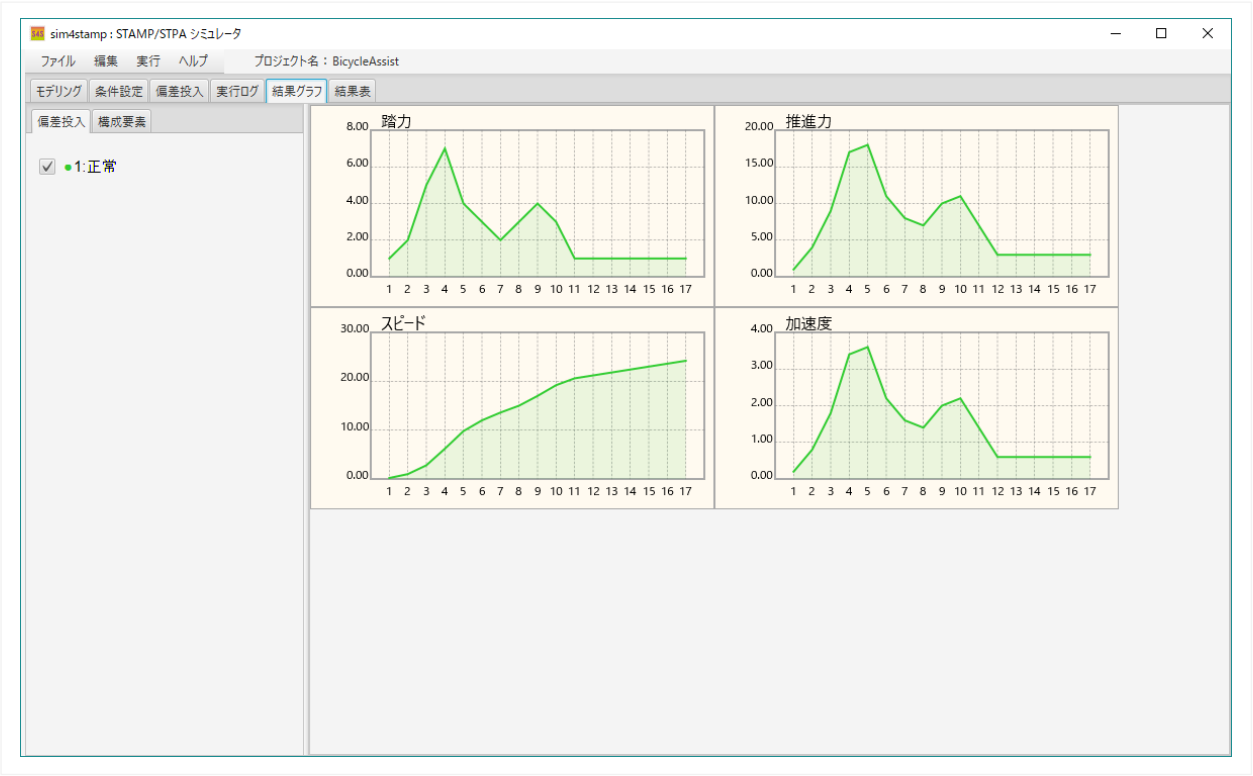
VDM++側で設定しているプリント文等で出力されるログを取り込み表示します。

13. シミュレーション実行結果表示

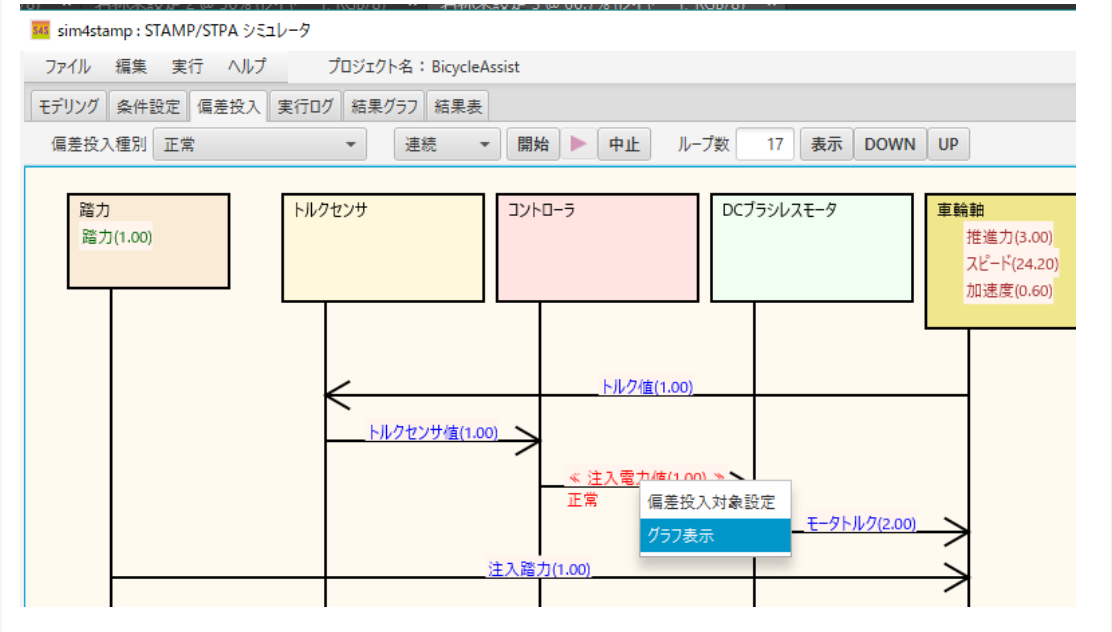
13.1. グラフ形式データ表示

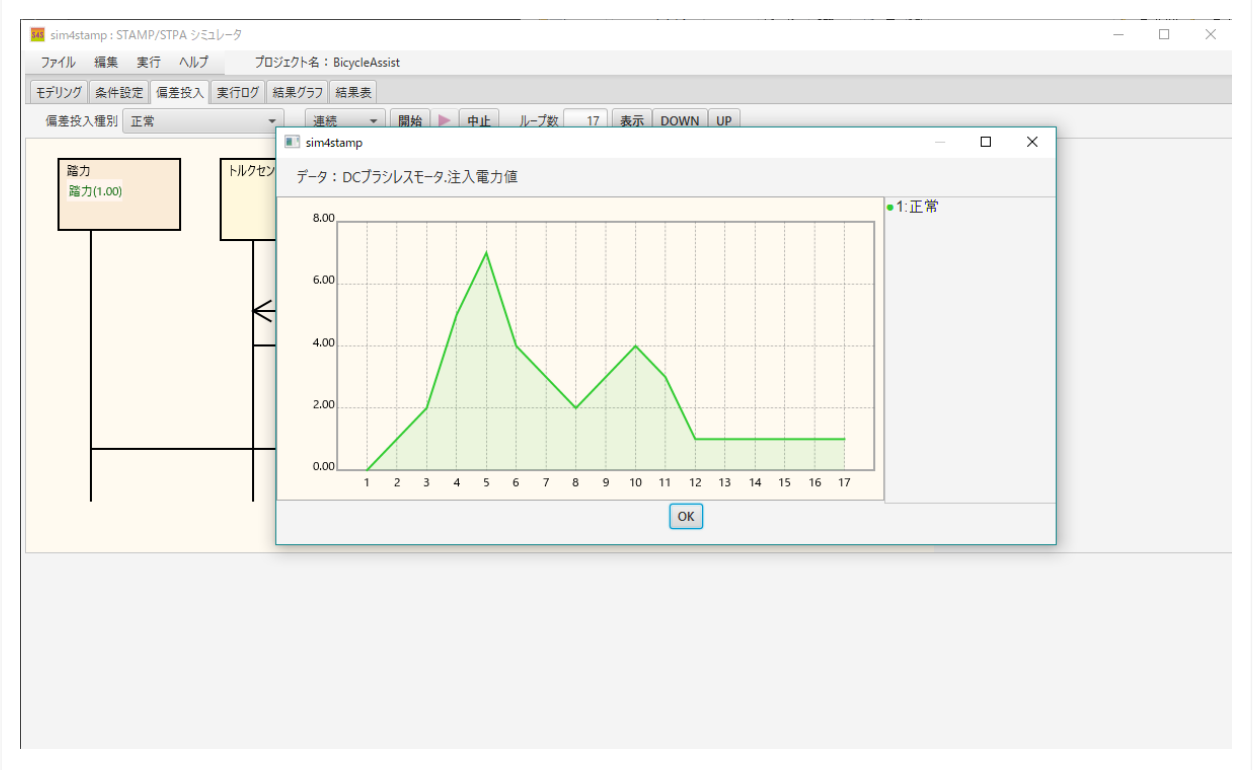
シミュレーション結果について、結果グラフの表示例を以下に示します。

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。

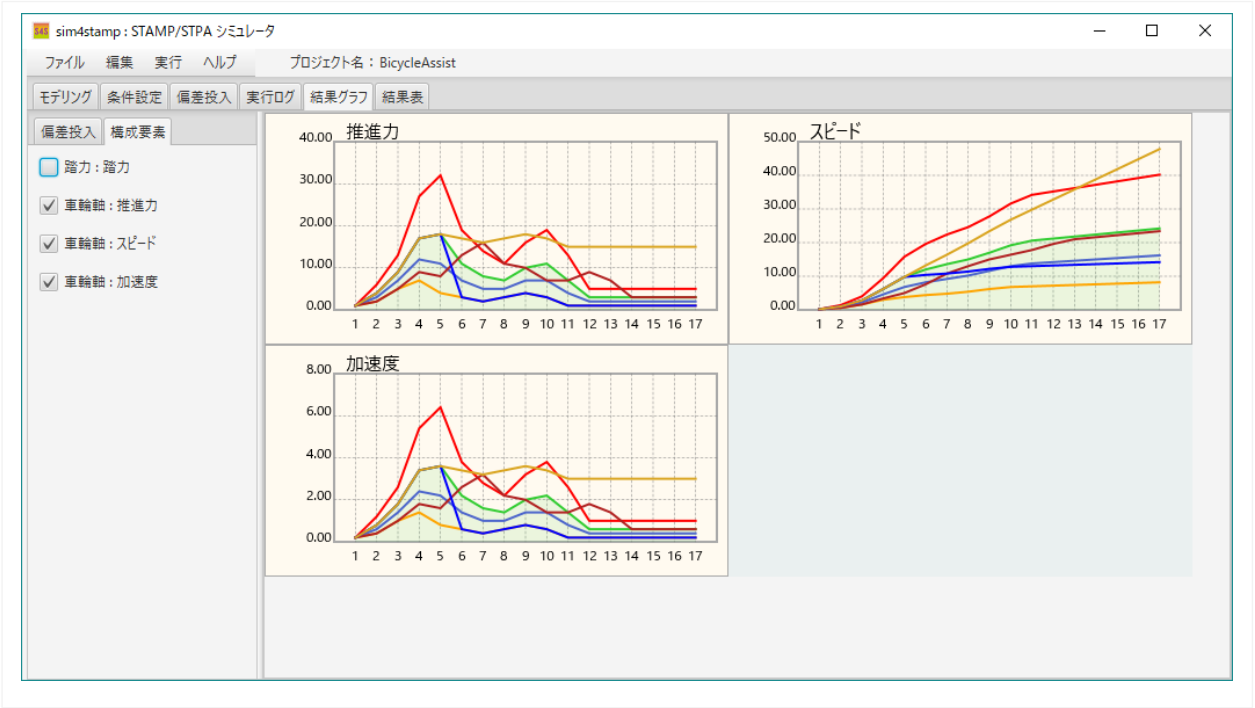




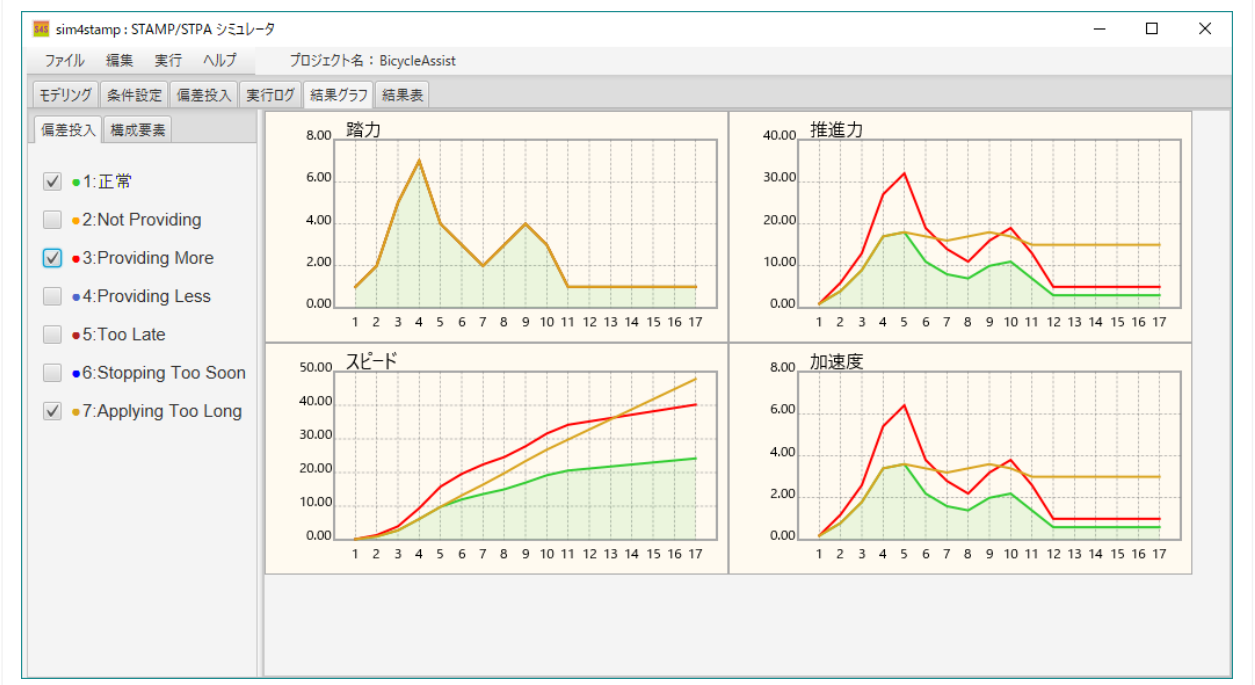
シミュレーションを累積すると、シミュレーションの累積グラフが表示され、正常ケースと偏差投入ケースの比較ができます。

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。

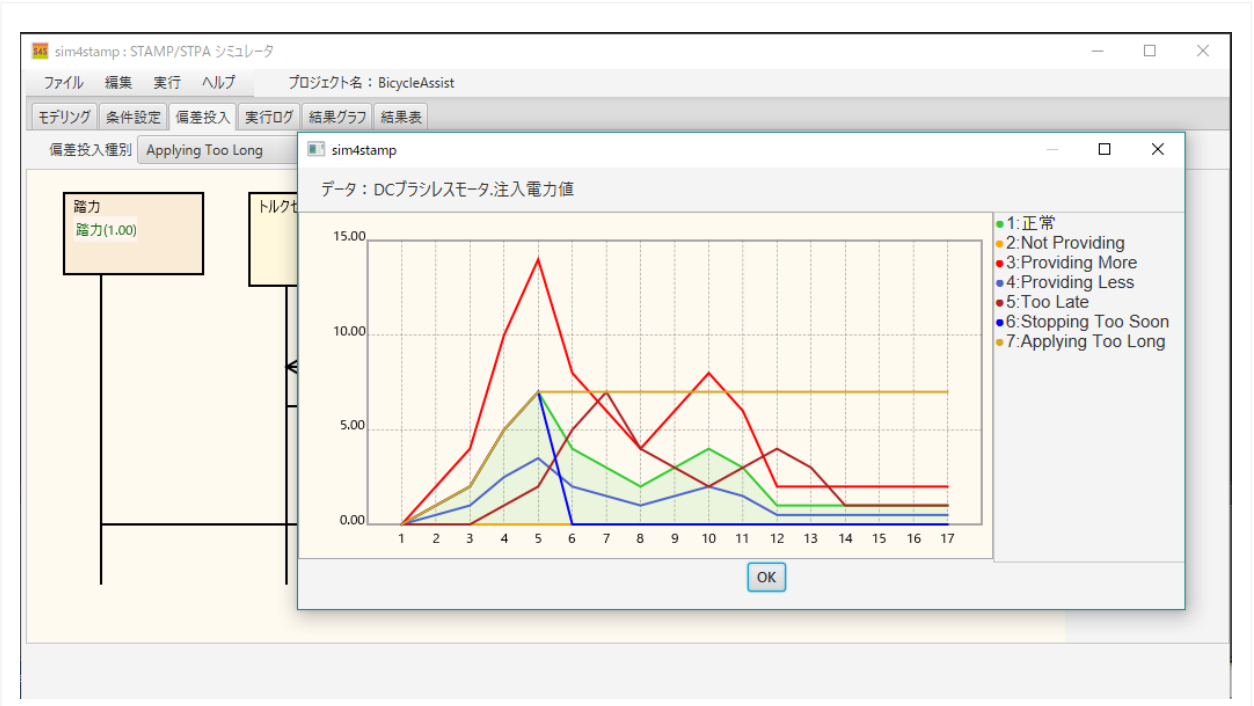
また、構成要素タブのチェックマークを外すと、対応するグラフを非表示にすることが出来ます。
こうすることにより、注目すべきグラフに集中できるようになります。



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。

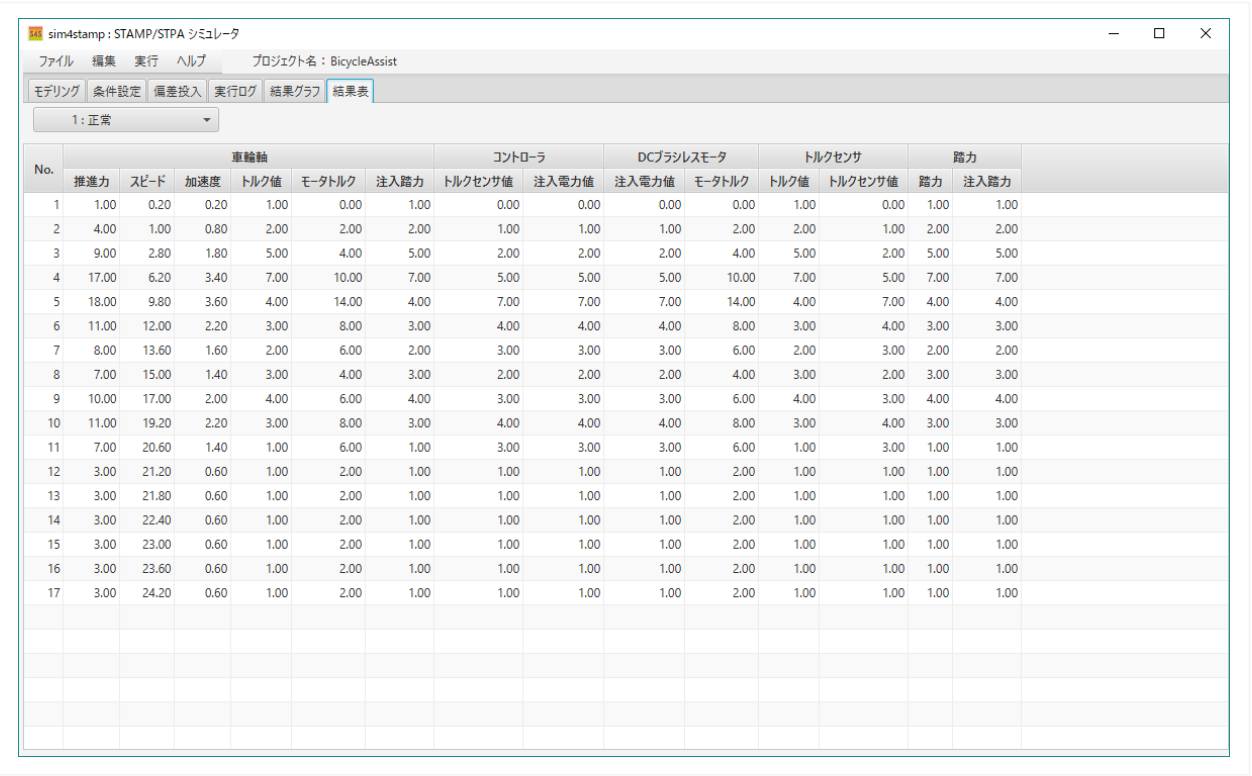


コネクタ データ グラフについても、複数のシミュレーションを行った場合、同様に重ねて表示します。



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。

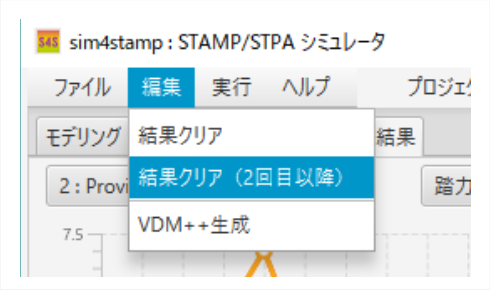


No.	車輪軸						コントローラ		DCブラシレスモータ		トルクセンサ		踏力	
	推進力	スピード	加速度	トルク値	モータトルク	注入踏力	トルクセンサ値	注入電力値	注入電力値	モータトルク	トルク値	トルクセンサ値	踏力	注入踏力
1	1.00	0.20	0.20	1.00	0.00	1.00	0.00	0.00	0.00	0.00	1.00	0.00	1.00	1.00
2	4.00	1.00	0.80	2.00	2.00	2.00	1.00	1.00	1.00	2.00	2.00	1.00	2.00	2.00
3	9.00	2.80	1.80	5.00	4.00	5.00	2.00	2.00	2.00	4.00	5.00	2.00	5.00	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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。

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無くなります。

14. 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。

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>);
        );

        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』
```

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。