

**Simscape[®] による
プラントモデリング入門**

Revision

Rev	監修	日時	備考
1	JMAAB Plant Model Workshop 7	2017, Jun	

目次

1. 本書に関して.....	5
1.1 本書を公開する背景.....	5
1.2 本書の範囲.....	5
1.3 本書の公開にあたり.....	6
1.4 著作権.....	6
1.5 本書の取り扱い.....	6
1.6 内容に関する問い合わせ先.....	6
1.7 注意事項.....	6
2. はじめに.....	7
2.1 モデルベースの必要性.....	7
2.2 V字プロセスとプラントモデルの必要性.....	8
2.3 シミュレーションの推移と警鐘.....	9
2.3.3 Simulink との違い ^[1-1]	10
3 簡易 HEV パワートレインモデル.....	12
3.1 目的.....	12
3.2 Simscape による HEV モデル構成例.....	12
3.3 部品モデル.....	16
3.3.1 エンジン.....	16
3.3.2 モータ.....	20
3.3.3 トルクコンバータ.....	25
3.3.4 トランスミッション.....	29
3.3.5 プラネタリギヤ.....	33
3.3.7 バッテリー.....	42
3.3.5 HEV 制御モデル.....	46
3.4 評価.....	51
4 車両運動モデル.....	54
4.1 目的.....	54
4.2 構成.....	54
4.3 部品モデル.....	55
4.3.1 タイヤ.....	55
4.3.2 車体.....	64
4.3.4 ステアリング.....	76
4.3.5 ブレーキ.....	79
4.4 評価.....	85
5. Simscape Tips.....	88
5.1 Simscape モデルの作成.....	88
5.1.1 モデル作成時の便利機能.....	88
5.1.2 Simulink の端子と Simscape の端子の接続方法.....	89

5.1.3 カスタムコンポーネントの作成	90
5.2 シミュレーションのための推奨設定	91
5.2.1 モデルのコンフィギュレーションパラメータ (ソルバー・ゼロクローシング)	91
5.2.2 Solver Configuration ブロック	91
5.3 シミュレーション結果の確認	92
5.3.1 シミュレーション データのプロット	92
5.3.2 Probe ブロック	93
5.4 変数の初期値の確認	94
5.4.1 Variable Viewer	94
5.4.2 初期値と優先度の設定	94

1. 本書に関して

1.1 本書を公開する背景

製品開発において、試作品を製作し、要求仕様や性能を評価することが従来行われてきた。これは非常に時間と費用がかかるだけでなく、人に対する安全機能を検証するには実験が大掛かりになり、開発費用がかさんでしまう。コンピュータ性能の向上、各種シミュレーション技術が発達し、現在ではコンピュータによる機能・性能検証が盛んに行われるようになってきた（コンピュータ内のバーチャル空間に環境や対象とする製品をモデル化し、それらを数式・数値的に解析するモデルベース開発）。

さまざまな企業でモデルベース開発が進む中、プラントモデル（対象とする製品のシミュレーションモデル）の取り扱いについては、一本化された方針（モデリングガイドライン）が決まっておらず、各社、独自に試行錯誤を繰り返して悩んでいるのが現状ではないだろうか。こうした中、プラントモデルのより効率的な作成・活用方法について協議し、それらをまとめることにより広く日本のものづくりにおける技術力向上を目的として、JMAAB（Japan MBD Automotive Advisory Board）の中にプラントモデルワークショップ（以下、PMWSという）を立ち上げた。この論議は実用的な内容が多く、以下の理由により、この成果を広く公開することにした。

PMWSのメンバーは、実務においてモデルベース開発に携わり、プラントモデルに関し様々な経験を持つと同時に目的意識をもった企業人で構成されている。このため、同じプラントモデルを考える際、さまざまなアイデアやデザインがもたらせる。それぞれのモデル作成方法には長所短所があり、正解も不正解もない。他の企業のモデルを見ることは刺激になり、情報や悩みを共有することで、企業を超えた技術の絆が生まれ、絞られた方向性が見えてくる。このようにPMWSで得た方向性をまとめてみると、将来、複数の会社にまたがって作成する大規模モデル構築する際には、様々なモデリング方法を俯瞰しておくことは、非常に有益であることが分かると同時に、今までの書籍では扱われていない内容となっている。このため、ワークショップ内に限らず、多くの技術者や今後もあるベース開発の担い手となる学生にも役に立つようにしたいという考えから、本書を作り情報公開することにした。ただし、PMWSについてはいろいろな論議をしてきたため、情報も広範囲に渡る。ここ3年に絞ってみると、特に以下の3つの項目について論議を重ねてきた、1つはプラントモデリングに関する人材育成、2つめはプラントモデルの標準化、3つめはプラントモデルを組織間で有効活用することについてである。

これらの内容を大きく2部に分け、1および2項目については初級や中級技術者の育成向けに『Simscape によるプラントモデル入門』とし、2および3項目を中心とした内容は、中級や上級技術者向けに『Simscape 実用』と区分けすることとした。

本書は、前者の入門書としてまとめたものであるが、技術者全般が誤解しやすいことや陥りやすい盲点についても配慮しているため、上級技術者にも有益な内容となっている。

1.2 本書の範囲

現状、MATLAB[®]/Simulink[®] 関連書籍は多数出版されており、基礎的な使い方については十分習得できる。しかしながら、市販されている書籍では、実際の開発現場ですぐに利用できるプラントモデルについて若手エンジニアが学ぶには不十分であり、特に、Simscape を使ったモデル設計を具体的に解説したものはほとんどない。一方、大学、大学院の制御理論で紹介されるプラントモデルも限定的であり、倒立振子を取り上げられている。しかし、限定的な倒立振子でさえ、現物を見て触れてみなければ実感がわかない。

このため、PMWS では、具体的な題材として、実用的で実感の湧く、HEV（電気ハイブリッド自動車）と車両のシャーシの2つのプラントモデルを取り上げ、Simscape を使って具体的にどのようにモデルを作ると理解しやすいか、また、誤った設計をしにくい、標準化を意識しながらモデル化を行った事例を紹介している。実践的な入門書を目指すため、本書では、数式をSimscape でどのようにモデル化すればよいのかだけでなく、モデリングに必要な式がどのように算出されたかの理由についても細かく述べるようにした。また、技術者が陥りやすい Simscape 特有の盲点や注意事項についても述べている。

本書で取り上げる 2 つのプラントモデルは、最終的には、制御対象として使われて初めて実践的なモデルとして成立するものと考えている。このため、JMAAB の協調制御ワークショップと連携し、制御可能なモデルとして結合を始めているが、本書を編集している現時点では、制御と結合を行った実績がないことを予め述べておく。

1.3 本書の公開にあたり

今回、PMWS のこれまでの内容をまとめ、その成果を公開することにより、様々な現場でモデル作成やシミュレーションを用いた開発効率の向上に役立つことを願っている。本書としてまとめられたのは、現在の PMWS のメンバーの努力はもちろんのこと、今まで PMWS に属していた方の情報も大いに参考になった。この場でお礼を申し上げる。

なお、読者をお願いしたいのは、我々は非営利で、他の技術者のお役に立てるよう努力してきた。しかし、一方的な情報開示であるため、皆様の意見を測り知ることができない。このため、ぜひ、この入門書を手にして感じたことがあれば、一報頂ければ幸いである。

最後に、PMWS で論議する場、また、Simscape を使って技術情報を共有して頂いた MathWorks[®] 社に感謝する。

1.4 著作権

- ・本書の著作権は JMAAB に帰属する。
- ・JMAAB は、本書の内容に関し、いかなる保証もするものではない。万一本書を利用して不具合等があった場合でも、JMAAB は一切責任を負わない。また、本書に記載されている事項は予告なしに変更または廃止されることがあることを理解し、活用いただきたい。

1.5 本書の取り扱い

- ・本書は、非営利目的、または利用者内部で使用する場合に限り、複製を可能とする。
- ・また、本書を引用する場合は、本書からの引用であることを明示し、引用された著作物の題号や著作者名を明示する等の引用の要件を満たす必要がある。

1.6 内容に関する問い合わせ先

本書に関する質問やご意見は、JMAAB 事務局 (jmaab-office@mathworks.co.jp) に問い合わせさせていただきたい。

1.7 注意事項

本書におけるツールのバージョンは、R2016a を使用している。他のバージョンでは、操作環境、操作結果、表示などが異なる可能性がある。

2. はじめに

2.1 モデルベースの必要性

自動車が生まれてから 100 年以上が経った今でも、機械部品、ギヤ、トランスミッション、エンジン、吸排気、後処理システム、クーリングユニットなどさまざまな技術革新がなされている。このような高度な製品開発においても、製品コスト低減のために開発期間短縮等の効率化が求められている。

Fig 2-1 のような従来の開発プロセスでは、詳細なシミュレーションモデルを作る期間がかかり、設計不具合が発生した場合は手戻りが大きかった。近年、MATLAB/Simulink などを用いたシミュレーション実行可能な仕様書がコンセプト段階で適用できるようになり開発期間が短縮できるような MBD（モデルベース開発）を実現することが可能になってきた。

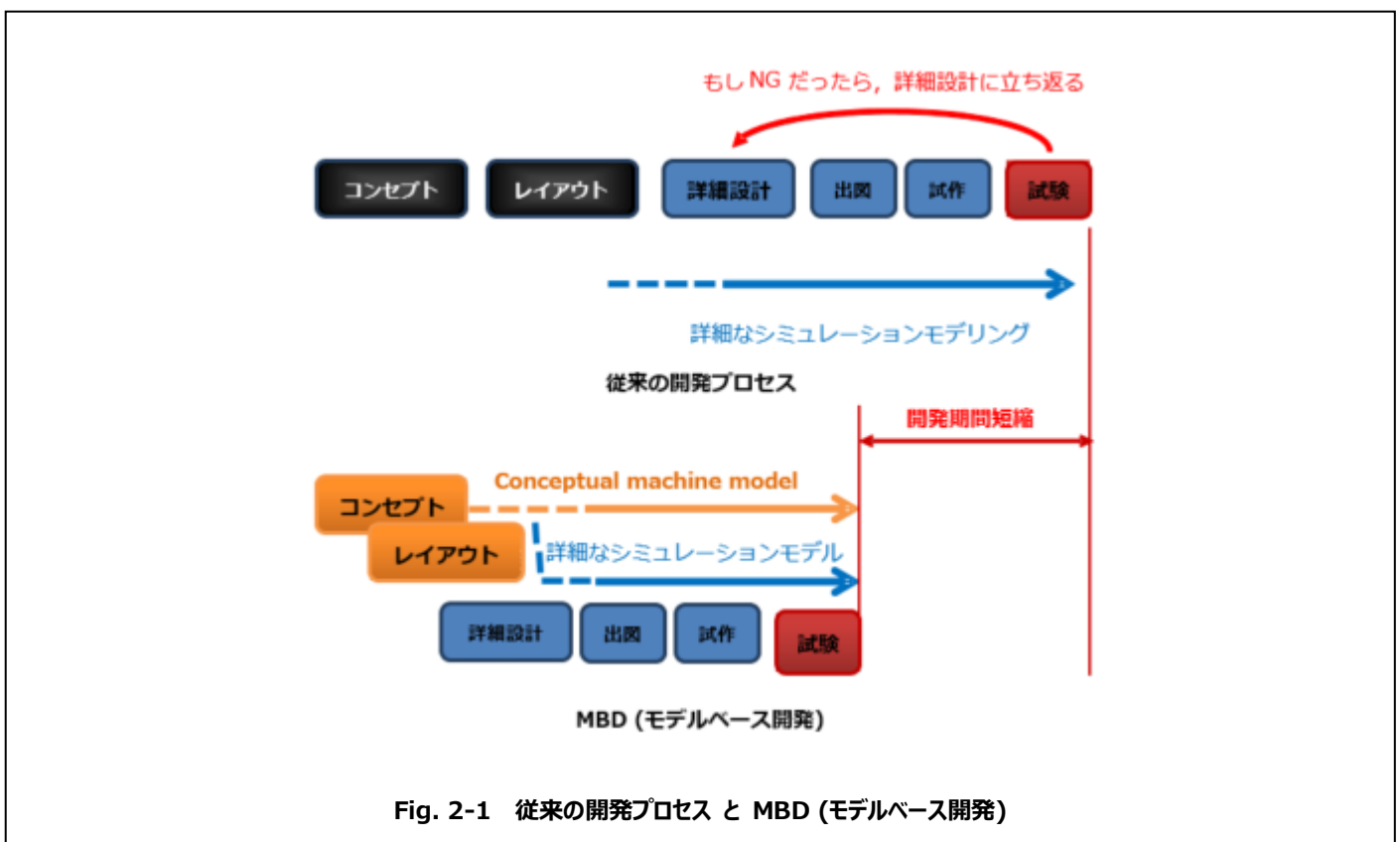


Fig. 2-1 従来の開発プロセス と MBD (モデルベース開発)

2.2 V字プロセスとプラントモデルの必要性

特に、近年においては、技術が大規模化／複雑化する一方、開発期間の短縮が求められている。自動車の制御システム開発においては、制御対象となるプラントモデルの開発が制御の開発と同時進行で行われるなどプラントモデルの開発効率化が強く求められている。

このため、Fig 2-2 に示すようなV字プロセスが適用されている。これは、実装までの前半部分は、“開発フェーズ”で「機能を作り込む段階」で、後半は“検証フェーズ”で「品質を確認する段階」である。開発効率向上と手戻りを小さくするため上述のMBDを使う。しかし、制御システム開発においては、検証フェーズまでは実際の制御対象ハードウェアが無い。

そこで、制御ロジックと制御対象を同時にモデル化し、コンピュータシミュレーションにより動作確認・機能検証を MILS (Model In the Loop Simulation) により行う。また、制御モデルをコード生成し、生成したコードからなるブロックと、制御対象モデルを組み合わせ、シミュレーションにより検証する SILS (Software In the Loop Simulation) という手段もある。一方、制御機器 (ECU) に対するテスト環境として、HILS (Hardware In the Loop Simulation) も広く利用されている。HILS 環境では、制御機器と制御対象モデルから生成した C ソースコードを組み合わせ、リアルタイムに動作させながら振る舞いを確認することができる。

いずれにしても、今や、シミュレーション技術なくして、制御システム開発ができない時代であり、MBDを使ったV字開発を推進することが製品の品質を高め、開発コストを削減することができる有効な開発手法である。

本書では、制御システム開発に不可欠である制御対象のプラントモデルにフォーカスし、プラントモデルのあり方、作成・活用方法を述べる。

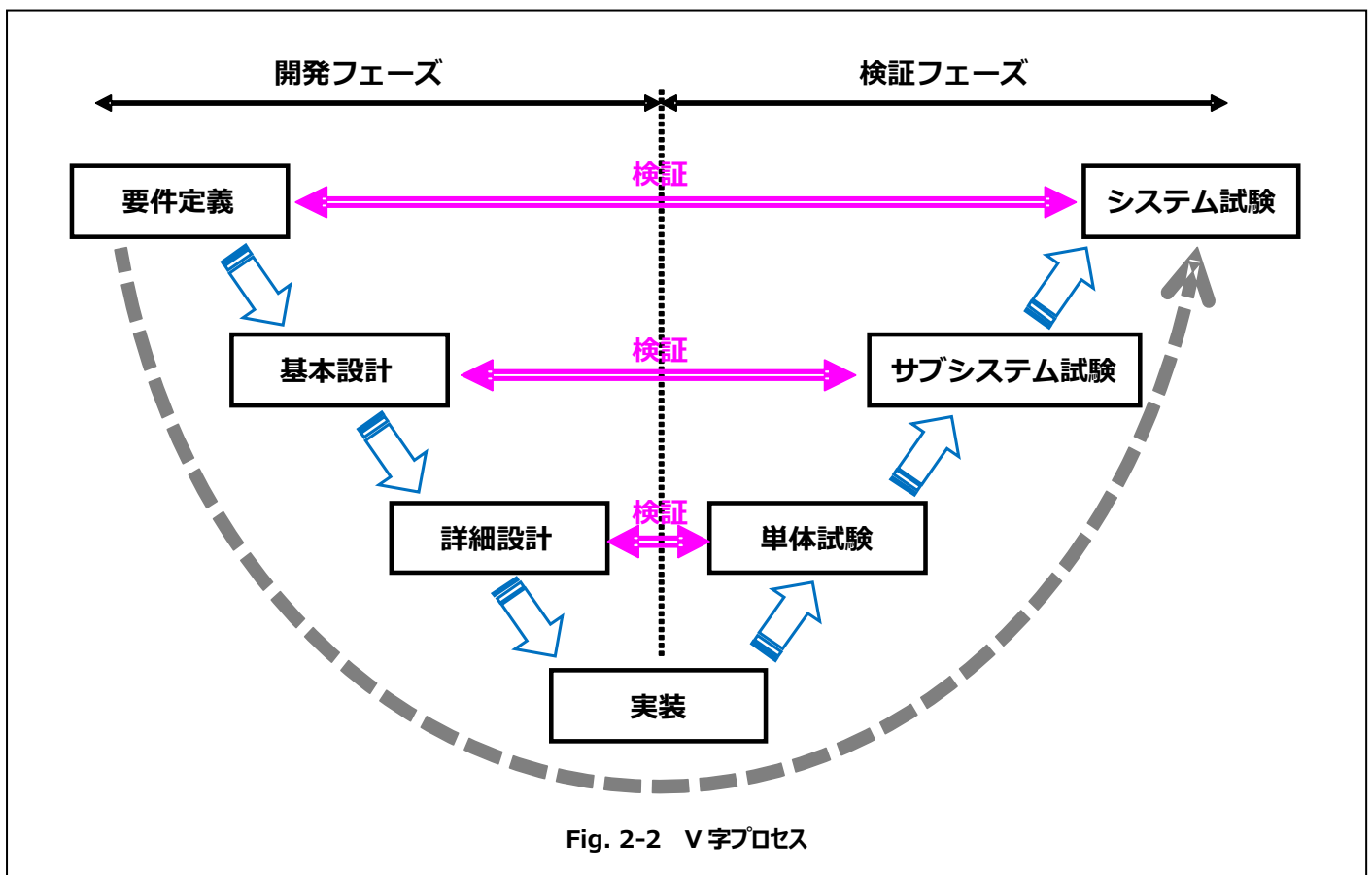


Fig. 2-2 V字プロセス

2.3 シミュレーションの推移と警鐘

コンピュータの発達とともに、下記のように、モデルベースを含むシミュレーション環境が発達してきた。

1990 年ごろから 3D-CAD を筆頭に 3 次元構造解析が飛躍的に普及。

2000 年ごろには、マイコンの処理速度が要求仕様にマッチするようになり、HILS が浸透。

2010 年ごろには、Virtual Reality/Augmented Reality

2015 年リアルタイムシミュレーション、1D-CAE（開発初期におけるコンセプトシミュレーションモデルによる一貫性のある実行可能な仕様書：Executable document）の模索

シミュレーションを使う側としては非常に便利になってきたのは事実である。50 歳を超える技術者はシミュレーションプログラムをアセンブラで記述していた時代もあった。アセンブラから C になり、C++ のオブジェクト指向が使えるようになると部品の再利用も容易になってきた。さらに、Simulink や Simscape などはプログラム言語に触れなくても（テキストエディタを用いたコーディングをしなくても）プラントモデルが作成できるようになってきた。一方で、制御システムを実装する際は、マイコンの固定小数点演算を使うのは変わっていない。アセンブラをやっていた者は、コンピュータはどのように演算するかを熟知している。このため、0.1 は 2 進数では表せないことや、メモリ配置の正順を知ってプログラムを組むためモデルベースを使ってもこの点に関しては慎重になる。また、Simscape をいきなり使ってしまうと、電子回路を CAD のように組み合わせるだけで、微分方程式を全く意識しなくなってしまう。

例えば、図 2-4 の Simscape モデルと図 2-3 の Simulink モデルが等価であることが分からなくても扱えてしまう。これでは、シミュレーションはできても、不具合が出たときにその意味が理解できず、デバックすることができない状態になってしまう。

このため、本書を使ってモデルベースを始めることはお勧めしない、Simulink で簡易なプラントモデルを経験したのちに、本書を利用することをお勧めする。

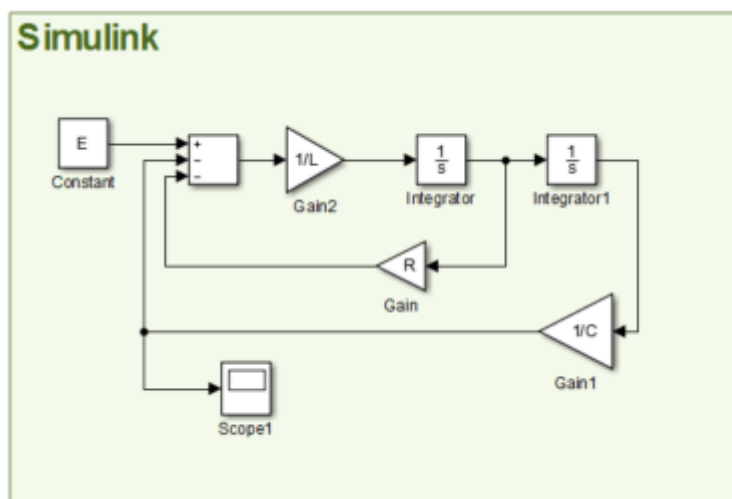


Fig.2-3 Simulink モデル (一例)

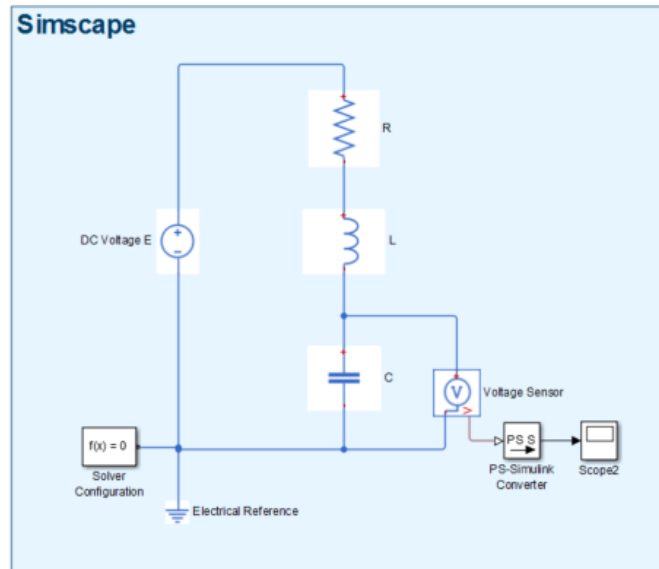


Fig.2-4 Simscape モデル (一例)

2.3.3 Simulink との違い [1-1]

Simscape がリリースされるまでは、一般的なモデリング環境である Simulink が制御モデルの作成やプラントモデリングに用いられてきた。しかしながら、Simulink のソルバーは基本的に常微分方程式を対象とするため、一般的に、微分代数方程式で表現される物理システムをモデル化するには、モデル作成者があらかじめシステムを大幅に単純化しなければならない場合がある。モデル化の対象が比較的小規模なシステムや、大規模であっても物理的には単純なシステムの場合には問題はないかもしれないが、大規模かつ複雑な物理システムを Simulink によりモデル化するのは困難である。

一方、Simscape は微分代数方程式ソルバーを備え、Simscape のコンポーネント内部には微分代数方程式を直接的に記述することができる。また、コンポーネント間の接続には保存則や釣り合いの式が適用される仕組みが搭載されており、大規模で複雑な物理システムのモデル化にも対応可能である。更に、Simscape のコンポーネントと Simulink のブロックをフレキシブルに接続する仕組みも提供されており、コントローラーとの閉ループ構築も容易に可能である。

また物理モデルの作成においても Simulink の場合、信号線の接続に方向性があるのに対し、Simscape は双方向の物理ネットワークで接続するため、各物理コンポーネント間の接続を直接的に表現でき、各物理ドメインの流れを把握しやすく、物理モデルの作成をより容易に検討できる。

このように Simscape は Simulink に対し、ソルバーや Simulink ブロックとの接続性、モデル作成の容易性と物理モデルを作成するのに有益な要素を兼ね備えており、Simulink を用いるよりも有効に活用できる機能を有している。

【参考文献】

[1-1] プラントモデリングのための MuPAD[®] 入門 (一部抜粋), JMAAB Plant Modeling Workshop 3

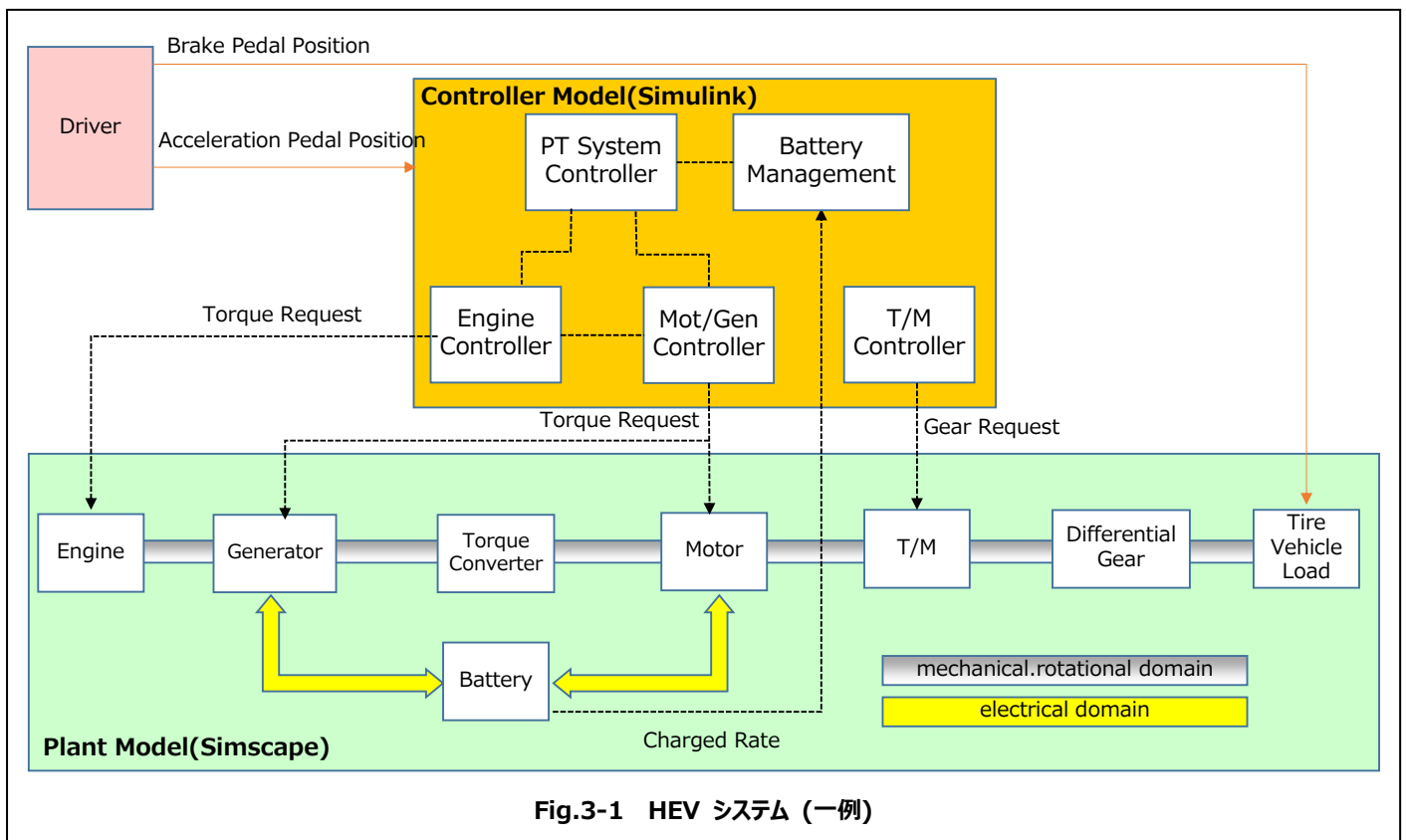
3 簡易 HEV パワートレインモデル

3.1 目的

既述している JMAAB のプラントモデルワークショップにおいて、各メンバーの Simscape の理解、スキルアップを図るためハイブリッド電気自動車（Hybrid Electric Vehicle 以下 HEV）のパワートレインを題材として、Simscape のサンプルモデルを作成した。本章では、そのモデルを取り上げ、Simscape でプラントモデルを構築した場合における各コンポーネント間の接続性や可読性や、Simscape の利点である各コンポーネントを自由に配置し、各種のパワートレインを簡単に構築出来ることを解説する。あわせて、HEV パワートレインモデルで使用している各コンポーネントを Simscape Language で作成したものを解説するので、Simscape によるプラントモデリング手法の参考としてほしい。また、本モデルは、第 4 章の車両運動モデルと接続し車両全体モデルとしての接続性検証や、モデルアーキテクチャ検討用としても利用した。

3.2 Simscape による HEV モデル構成例

一言で HEV といっても、メーカ各社が様々なパワートレイン方式を開発、製品化している。これら様々な HEV パワートレイン方式を、Simscape で作成した個々のコンポーネントモデルを組み合わせることで、シミュレーションモデルの構築が容易に可能となる。以下の図は、PMWS の各メンバーが作成したコンポーネントを持ち寄り、構築した HEV モデルの一例であるが、トルクコンバータの前にエンジンによる発電用モータ、トルクコンバータ後に走行用モータを配置した 2 モータ HEV 方式の場合のシステム図である。



上記システムのプラントモデル部分を Simscape でモデル化した場合は、以下の図の通りとなる。もちろん、モータを一つ削除すれば、1 モータタイプの HEV モデルに変更することも容易である。尚、以下のモデル図中で緑線は機械・回転系の物理ドメイン信号を示しており、青線は電気の物理ドメインの信号を表示している。

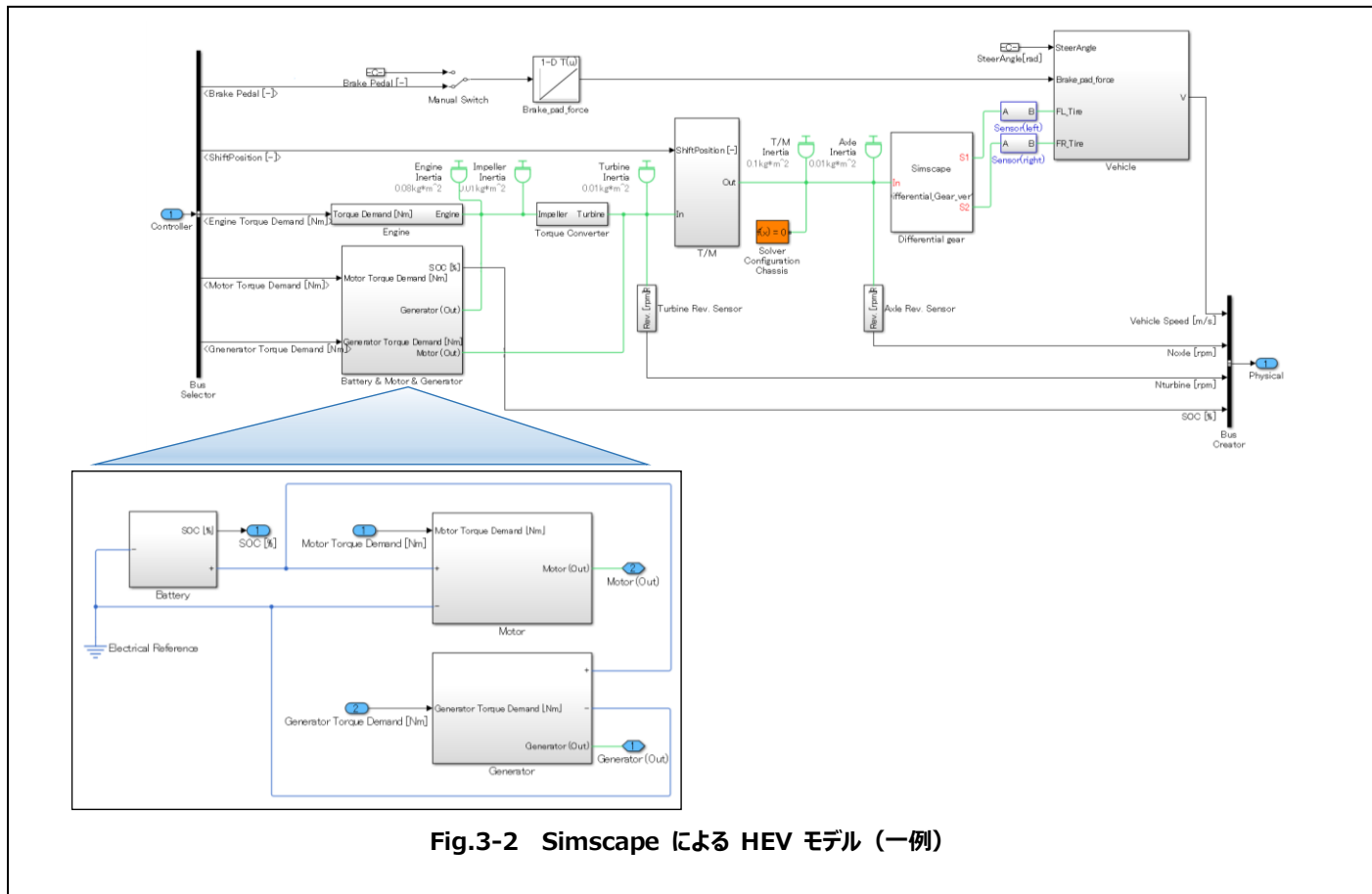


Fig.3-2 Simscape による HEV モデル (一例)

また、別の HEV パワートレイン方式として、トランスミッションの代わりにプラネタリギヤを使用した HEV 方式についても簡単に説明する。前述のモデルから、トルクコンバータとトランスミッションモデルを削除し、プラネタリギヤモデルを追加して、物理ドメインを接続するだけで済むことが、次の図からお分かりいただけると思う。

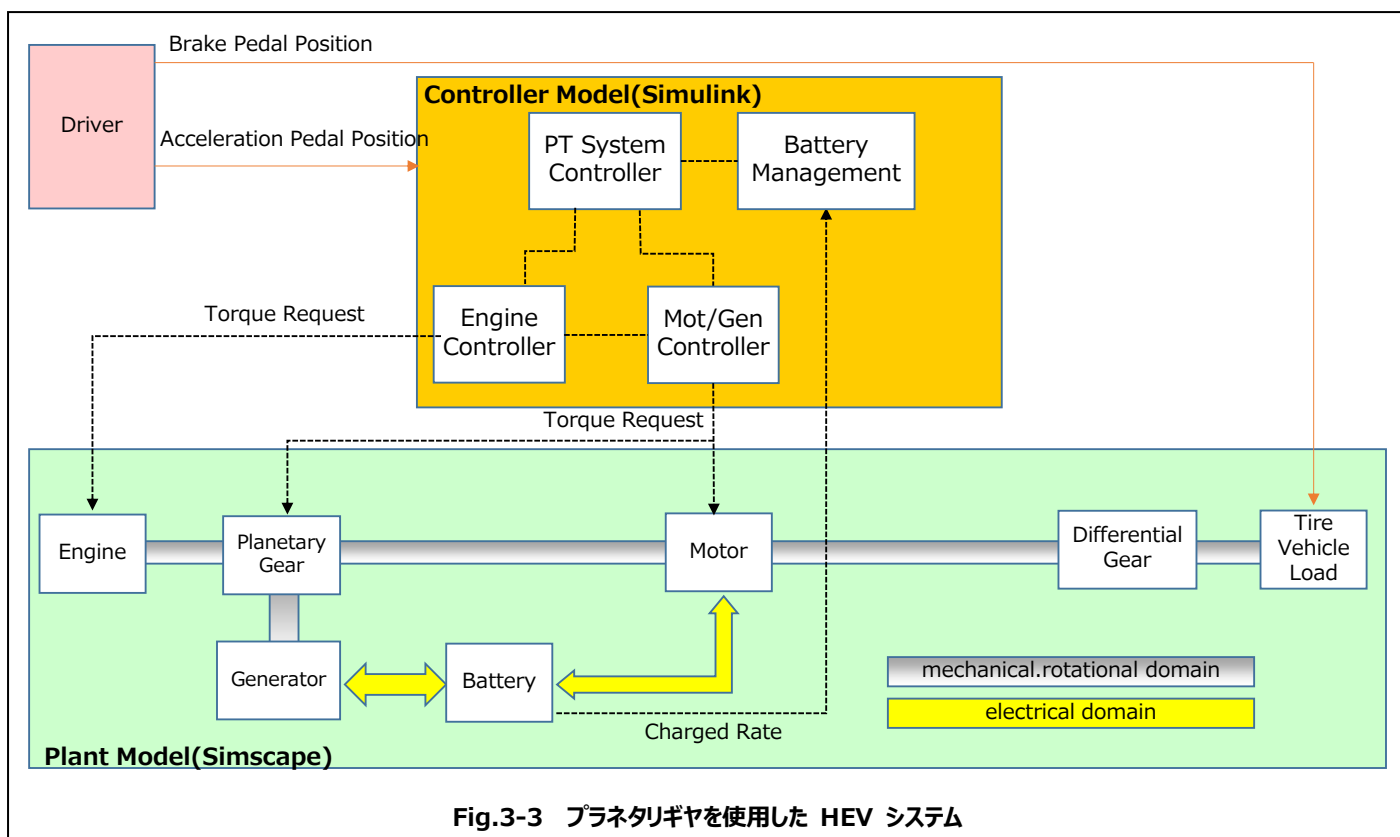


Fig.3-3 プラネタリギヤを使用した HEV システム

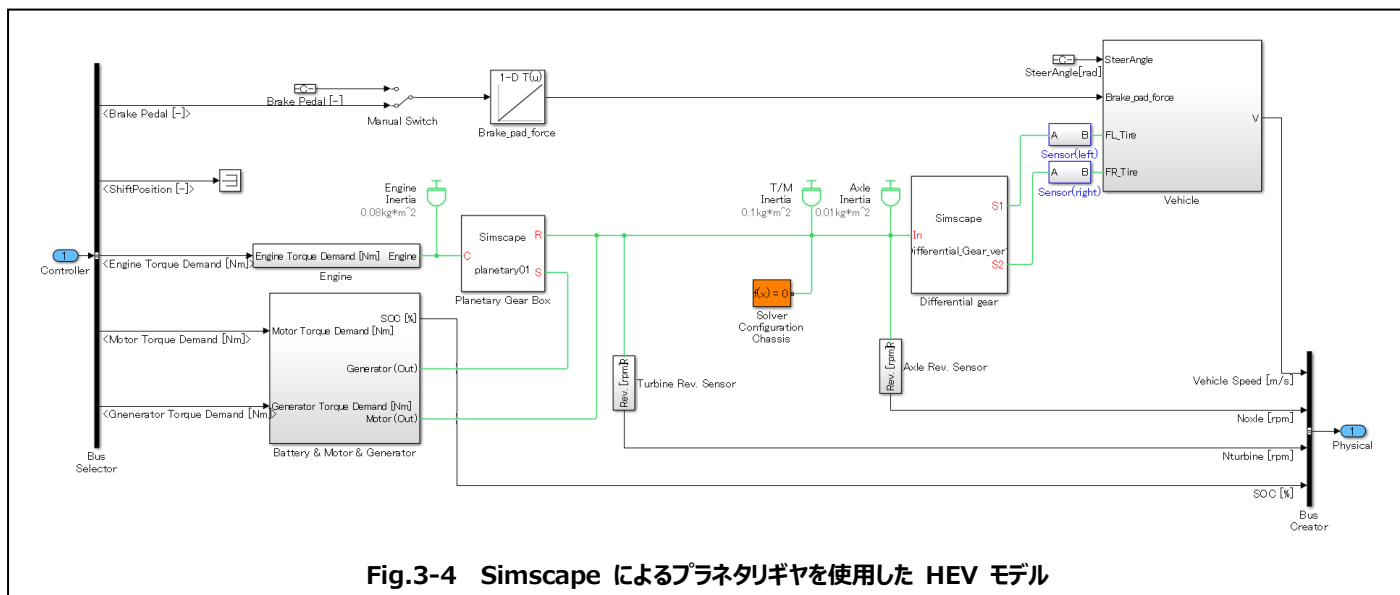


Fig.3-4 Simscape によるプラネタリギヤを使用した HEV モデル

以上の通り、今回紹介したサンプルモデルは、Simscape 標準の機械・回転ドメインと電気ドメインだけを使用した単純なモデルであるが、Simulink で作られたプラントモデルに対して、実際のコンポーネントを組み立てる感覚で接続していくため、非常に操作性がよく、複雑な結線が無くシンプルなため可読性も良い。また、同一のドメイン同士の信号線を接続するだけで済むため、人

為的なミスが起こりにくい。最終的には、Simscape で作成したプラントモデルは、Simulink プラットフォーム上で、Simulink で作成したコントローラモデルとシームレスに接続し、シミュレーションできることが最大の利点である。

3.3 部品モデル

本章では、HEV パワートレインモデルで使用している、Simscape Language で作成した各コンポーネントモデルを簡単に説明する。

3.3.1 エンジン

コントローラから出力されるトルク指示値に基づき、エンジン回転数に応じてあらかじめ設定された最大エンジントルクの範囲内で、エンジントルクを出力する簡易モデルとなる。尚、エンジントルクの応答遅れ、エンジン回転慣性などは考慮されていない。

エンジン最大トルク T_{e_max} は、以下の通りエンジン回転数 W に応じた 1D テーブルで設定する。

$$T_{e_max} = F(W)$$

テーブル内部：線形補間 ($interp_method=1$)

テーブル外部：最終値保持($extrap_method=2$) :

エンジン出力トルク t は、コントローラからの要求トルク T_{demand} と上記最大トルク T_{e_max} を比較して以下の通り出力される。

$$T_{e_max} \geq T_{demand} \text{ のとき、}$$

$$t = T_{demand}$$

$$T_{e_max} < T_{demand} \text{ のとき、}$$

$$t = T_{e_max}$$

(1) モデルブロック

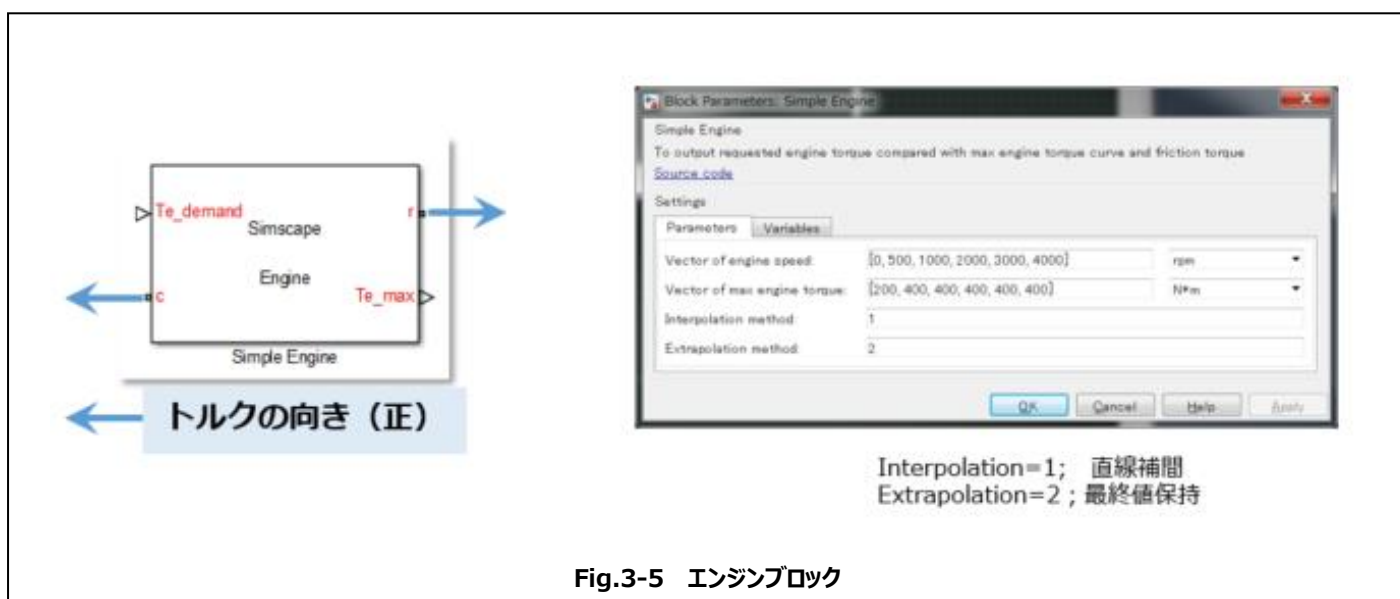
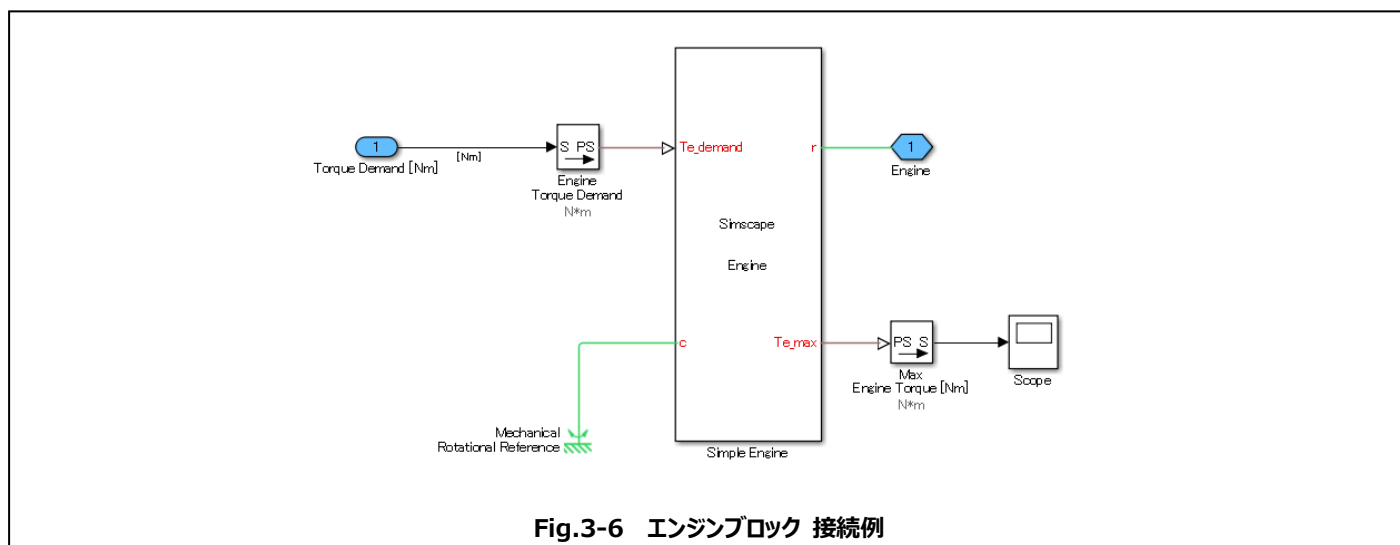


Fig.3-5 エンジンブロック

(2) 接続例



(3) モデル詳細

名称	Engine						
概要	エンジン最大トルクの範囲内で、指示されたエンジントルクを出力						
	トルクの向き ; ノード c to r						
	計算内容 ;						
	①エンジン回転数に応じてエンジン最大トルクを算出						
	②要求トルクがエンジン最大トルクの範囲内であれば、要求トルクを軸トルクとして出力						
③要求トルクがエンジン最大トルクを超えている場合は、最大トルクを軸トルクとして出力							
ノード		ポート名	物理ドメイン	Through/Across	内容	表示位置	
	1	r	機械回転	トルク(Nm), 回転数(rad/s)	エンジン出力 (パワートレーン)	right	
	2	c	機械回転	トルク(Nm), 回転数(rad/s)	入力 (メカニカルリファレンス)	left	
入力		ポート名	内容	範囲	初期値	単位	表示位置
	1	R	要求トルク	-	0	N*m	left
出力		ポート名	内容	範囲	初期値	単位	表示位置
	1	Tmax	エンジン最大トルク	-	0	N*m	right
パラメータ		パラメータ名	内容	範囲	初期設定値	単位	
	1	x_t	X;エンジン回転数	-	外部テーブル(下記参照)	rpm	
	2	y_t	Y;エンジン最大トルク	-	外部テーブル(下記参照)	N*m	
変数		変数名	内容	範囲	初期値	単位	

	1	t	エンジン出力トルク	-	0	N*m	
	2	w	エンジン角速度	-	0	rad/s	

方程式、その他

① 下記テーブルよりエンジン最大トルクを算出

t_{max}

[エンジン最大トルクテーブル]

	x_t (rpm)	0	500	1000	2000	3000	4000	
	y_t (N*m)	200	400	400	400	400	400	

② 要求トルクがエンジン最大トルクの範囲内であれば、要求トルクを軸トルクとして出力

$t_{max} > TrqRq$ のとき $t = TrqRq$

③ 要求トルクがエンジン最大トルクを超えている場合は、最大トルクを軸トルクとして出力

$t_{max} < TrqRq$ のとき $t = t_{max}$

(3) Simscape コード

```
component Engine
% Simple Engine
% Ver. 1.0 Jun/22, 2017 JMAAB Plant Modeling Workshop

parameters(Size = variable)
    Rev = {1 0 500 1000 2000 3000 4000}, 'rpm'; % Vector of engine speed
    Torque = {[200 400 400 400 400 400]}, 'N*m'; % Vector of max engine torque
end

parameters
    interp_method = 1; % Interpolation method
    extrap_method = 2; % Extrapolation method
end

inputs
    % Engine torque demand
    Te_demand = {0, 'N*m'}; % Te_demand : left
end

nodes
    r = foundation.mechanical.rotational.rotational; % r : right
    c = foundation.mechanical.rotational.rotational; % c : left
end

variables
    Te = {0, 'N*m'}; % Engine torque
    W = {0, 'rad/s'}; % Angular velocity
end

outputs
    Te_max = {0, 'N*m'}; % Te_max : right
end

branches
    Te : c.t -> r.t; % Engine torque
end

% The inertia is not included in this component definition.
% It can be added to the physical network in the model.

equations
    W == r.w - c.w;

    if (interp_method == 1)
        if (extrap_method == 1)
            Te_max == tablelookup(Rev, Torque, W, interpolation=linear, extrapolation=linear);
        elseif (extrap_method == 2)
            Te_max == tablelookup(Rev, Torque, W, interpolation=linear, extrapolation=nearest);
        else
            assert(0, 'invalid extrapolation method');
        end
        Te_max == 0;
    elseif (interp_method == 2)
        if (extrap_method == 1)
            Te_max == tablelookup(Rev, Torque, W, interpolation=cubic, extrapolation=linear);
        elseif (extrap_method == 2)
            Te_max == tablelookup(Rev, Torque, W, interpolation=cubic, extrapolation=nearest);
        else
            assert(0, 'invalid extrapolation method');
        end
        Te_max == 0;
    elseif (interp_method == 3)
        if (extrap_method == 1)
            Te_max == tablelookup(Rev, Torque, W, interpolation=spline, extrapolation=linear);
        elseif (extrap_method == 2)
            Te_max == tablelookup(Rev, Torque, W, interpolation=spline, extrapolation=nearest);
        else
            assert(0, 'invalid extrapolation method');
        end
        Te_max == 0;
    else
        assert(0, 'invalid interpolation method');
        Te_max == 0;
    end

    if (Te_demand > Te_max)
        Te == Te_max;
    else
        Te == Te_demand;
    end
end
end
```

最大トルクの 1D テーブル
補間計算

エンジン出力トルク計算

3.3.2 モータ

コントローラからのモータ指示トルクに応じて、トルクを出力する簡易モータモデルとなる。モータ損失は、効率により簡易的に考慮している。また、モータ電流制限に応じて出力トルクを制限する。

コントローラからの目標モータトルク T_{target} (*Targe Torque*) に対して、モータが消費、もしくは発電する電流 I_{temp} を求める。

モータを力行側（正モータトルク）で使用する場合、

$$V * I_{temp} * Efficiency = T_{target} * W$$
$$I_{temp} = \frac{T_{target} * W}{V * Efficiency}$$

モータを充電側（負モータトルク）で使用する場合、

$$V * I_{temp} = T_{target} * W * Efficiency$$
$$I_{temp} = \frac{T_{target} * W * Efficiency}{V}$$

ここで、 V はモータにおける電位差、 W はモータ回転数、 $Efficiency$ はモータ効率。尚、ここでのモータ効率は、力行側は電力から動力への変換効率、充電側は動力から電力への変換効率となり、本モデルでは便宜上同一効率を使用している。

また、本来インバータの機能であるが、本モータモデルでは使用できる上限電流 I_{max} をパラメータで設定することが出来き、実際のモータ出力トルク T はこの電流制限に応じて以下のように制限される。ここで、 I は最終的にモータが消費、もしくは発電する電流値。

$0 \leq I_{temp} < I_{max}$ の時、力行側で電流制限がない場合、

$$T = T_{target}$$
$$I = \frac{T_{target} * W}{V * Efficiency}$$

$I_{temp} \geq I_{max}$ の時、力行側で電流制限される場合、

$$T = \frac{V * I_{max} * Efficiency}{W}$$
$$I = I_{max}$$

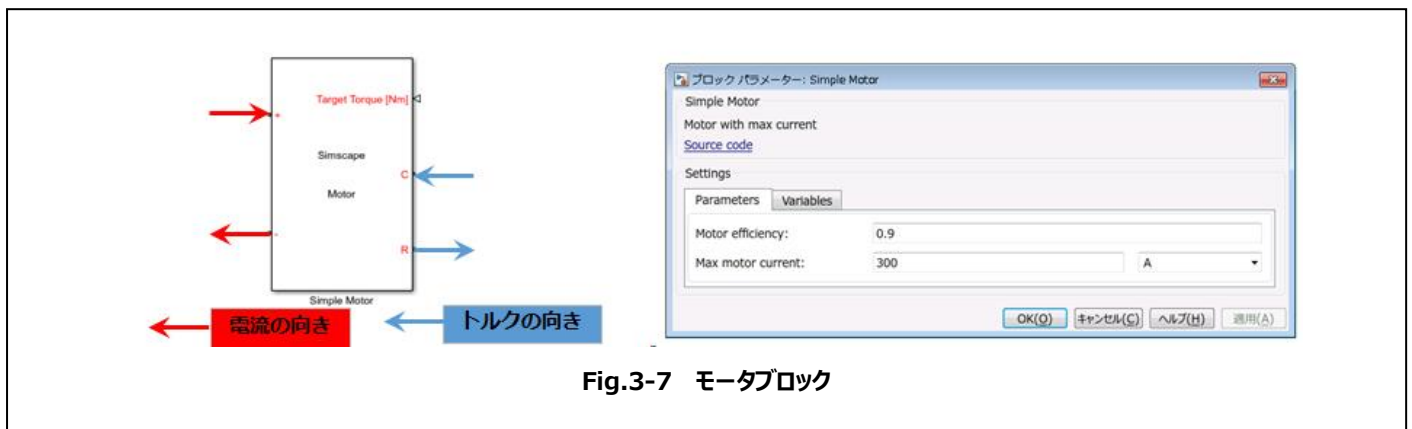
$I_{max} < I_{temp} < 0$ の時、充電側で電流制限がない場合、

$$T = T_{target}$$
$$I = \frac{T_{target} * W * Efficiency}{V}$$

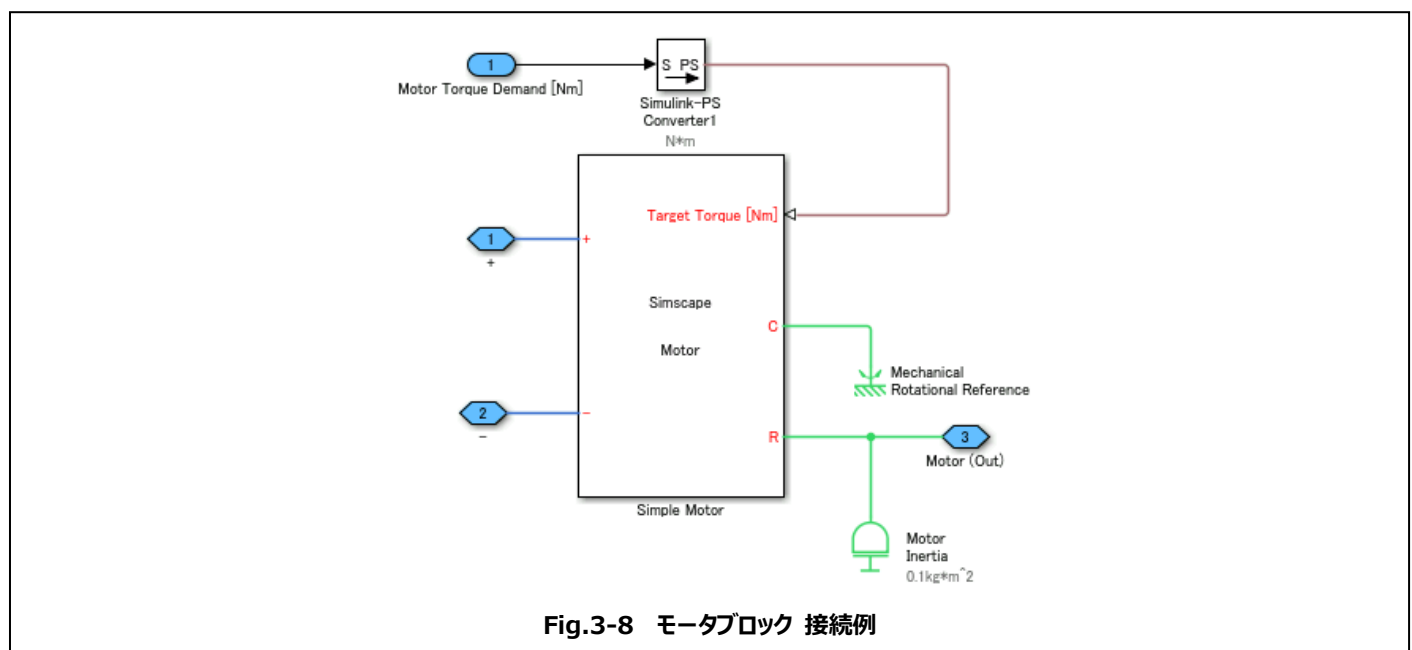
$I_{temp} \leq -I_{max}$ の時、充電側で電流制限される場合、

$$T = \frac{V * -I_{max}}{W * Efficiency}$$
$$I = -I_{max}$$

(1) モデルブロック



(2) 接続例



(3) モデル詳細

名称	Motor						
概要	HEV コントローラからのモータ指示トルクに応じて、トルクを出力						
	電流の向き；モータで消費する電流を正						
	トルクの向き；車両を前進させる時のモータトルクを正						
	①エネルギー保存則； 電気エネルギー（電圧、電流） ⇔ 回転運動エネルギー（回転速度、トルク）						
	②モータ効率のみ考慮(フリクションは未考慮)						
③モータ定格電流を制限するため（本来は Inverter で行う）、ターゲットトルクから電流制限値に応じて出力トルクを制限							
ノード	ポート名	物理ドメイン	Through/Across	内容（接続先）		表示位置	
	1	p(+)	電気	電圧 (V)、電流 (A)	モータ入力側正極 (Battery +)	left	
	2	n(-)	電気	電圧 (V)、電流 (A)	モータ入力側負極 (Battery -)	left	
	3	R	機械回転	回転数(rad/s)、トルク(Nm)	モータ出力 (パワートレイン)	right	
	4	C	機械回転	回転数(rad/s)、トルク(Nm)	入力 (メカニカルリファレンス)	right	
入力	ポート名	内容	範囲	初期値	単位	表示位置	
	1	Target Torque	指示トルク	-	0	N*m	right
パラメータ	ポート名	内容	範囲	初期値	単位		
	1	EMeff	モータ効率	0 ..1	0.9	1	
	2	Ilim	電流制限	>0	300	A	
変数	変数名	内容	範囲	初期値	単位		
	1	i	電流	-	0(High)	A	
	2	i_temp	ターゲットトルクからの電流	-	0	A	
	3	v	電圧	-	0	V	
	4	t	モータ出力トルク	-	0	N*m	
	5	w	モータ角速度	-	0	rad/sec	
方程式、その他							
ターゲットモータトルク(正)に必要な電流 $I_temp = T_target * W / (V * Efficiency)$							
ターゲットモータトルク(負)に必要な電流 $I_temp = T_target * W * Efficiency / V$							

モータ出力トルクを電流上限に応じて制限

① $0 \leq I_temp < I_max$ の時

$$T = T_target \quad , \quad I = T_target * W / (V * Efficiency)$$

② $I_temp \geq I_max$ の時

$$T = (V * I_max * Efficiency) / W \quad , \quad I = I_max$$

③ $-I_max < I_temp < 0$ の時

$$T = T_target \quad , \quad I = T_target * W * Efficiency / V$$

④ $I_temp \leq -I_max$ の時

$$T = (V * -I_max * Efficiency) / W \quad , \quad I = -I_max$$

電位差

$$V = p.v - n.v$$

モータ回転数

$$W = R.w - C.w$$

(4) Simscape コード

```
component Motor
% Simple Motor
% Motor with max current

parameters
    Efficiency = { 0.9, '1' }; % Motor efficiency
    I_max      = { 300, 'A' }; % Max motor current
end

inputs
    T_target = { 0, 'N*m' }; % Target Torque [Nm] : right
end

nodes
    p = foundation.electrical.electrical; % + : left
    n = foundation.electrical.electrical; % - : left
    C = foundation.mechanical.rotational.rotational; % C : right
    R = foundation.mechanical.rotational.rotational; % R : right
end

variables
    I      = { 0, 'A' }; % Current
    I_temp = { 0, 'A' }; % Current (Temporary)
    V      = { 0, 'V' }; % Voltage
    T      = { 0, 'N*m' }; % Torque
    W      = { 0, 'rad/s' }; % Angular velocity
end

branches
    I : p.i -> n.i;
    T : C.t -> R.t;
end
```

equations

V == p.v - n.v;

W == R.w - C.w;

if (T_target >= 0)
V * I_temp * Efficiency == T_target * W;

モータ正トルク、モータは電流を消費する場合の
mechanical、electrical との関係式。

else %T_target < 0
V * I_temp == T_target * W * Efficiency;

モータ負トルク、モータは発電する場合の
mechanical、electrical との関係式。

end

if (I_temp < I_max && I_temp >= 0)
T == T_target;
I == T_target / V / Efficiency * W;

elseif (I_temp >= I_max)
T == V * I_max * Efficiency / W;
I == I_max;

elseif (I_temp > -I_max && I_temp < 0)
T == T_target;
I == T_target / V * Efficiency * W;

else %I_temp <= -I_max
T == V * (-I_max) / Efficiency / W;
I == -I_max;

電流制限値による正負モータトルクのリミット。

end

end

end

3.3.3 トルクコンバータ

トルクコンバータは流体継ぎ手の一種であり、ポンプインペラー（以下インペラー）、タービンランナ（以下タービン）、ステータから構成され、トルク増幅効果を有する。

インペラーが入力軸、タービンが出力軸にあたり、その回転数の比を速度比と呼ぶ。

$$\text{SpeedRatio} = \frac{W_{\text{trb}}}{W_{\text{imp}}}$$

ここで、

SpeedRatio:速度比[-], W_{trb} :タービン回転数[rad/s], W_{imp} :インペラー回転数[rad/s]

一般的に、トルクコンバータの性能は、以下の2式で表される。

$$\text{TorqueRatio} = \frac{T_{\text{trb}}}{T_{\text{imp}}}$$

$$\tau = \frac{T_{\text{imp}}}{W_{\text{imp}}^2}$$

ここで、

TorqueRatio:トルク比[-], T_{trb} :タービントルク[Nm],
 W_{imp} :インペラートルク[Nm], τ :トルク容量係数[Nm/(rad/s)²]

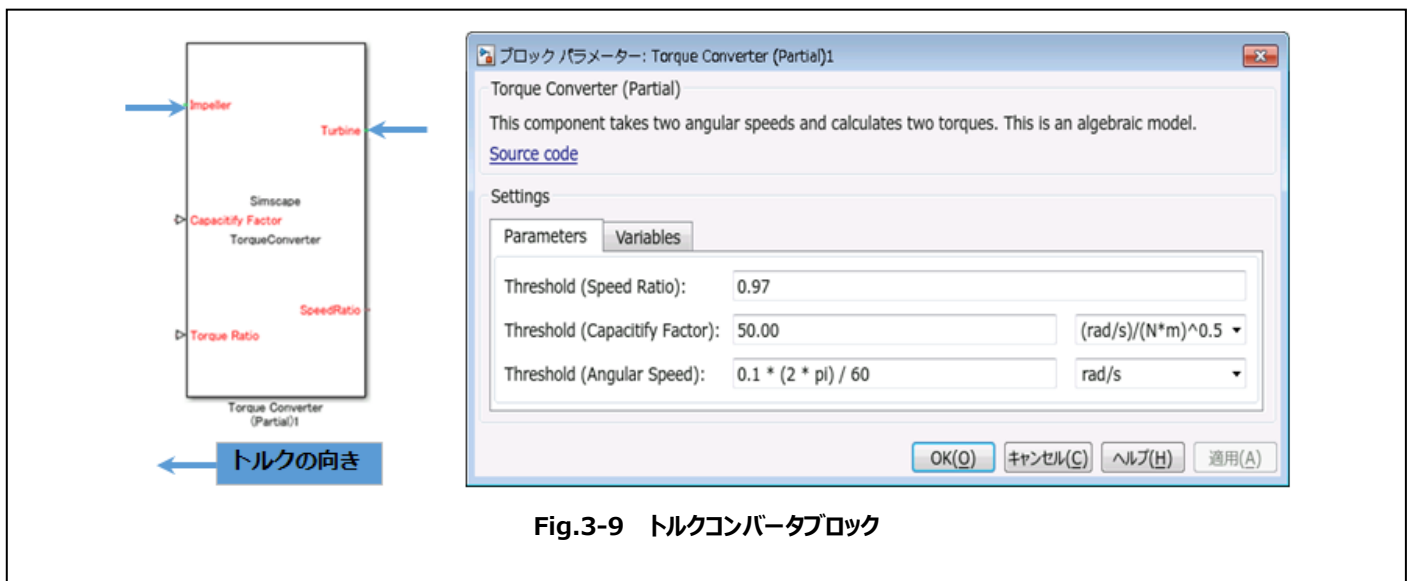
トルク比、トルク容量係数は、速度比に依存した特性として与えられる。例えばトルク比は、速度比が0の場合に最大となり、速度比が1に近づくにつれて減少していき、速度比1近くでは約1（損失があるため、1を下回る）となる。本モデルでも、トルク比と容量係数を、速度比に依存した値として特性データから補間して計算し、外部から入力値として与えるようにしている。

また、容量係数は、様々な係数の取り方があり、本モデルで用いている容量係数 $K[(\text{rad/s})/(\text{Nm})^{-0.5}]$ と τ とは下記の関係がある。

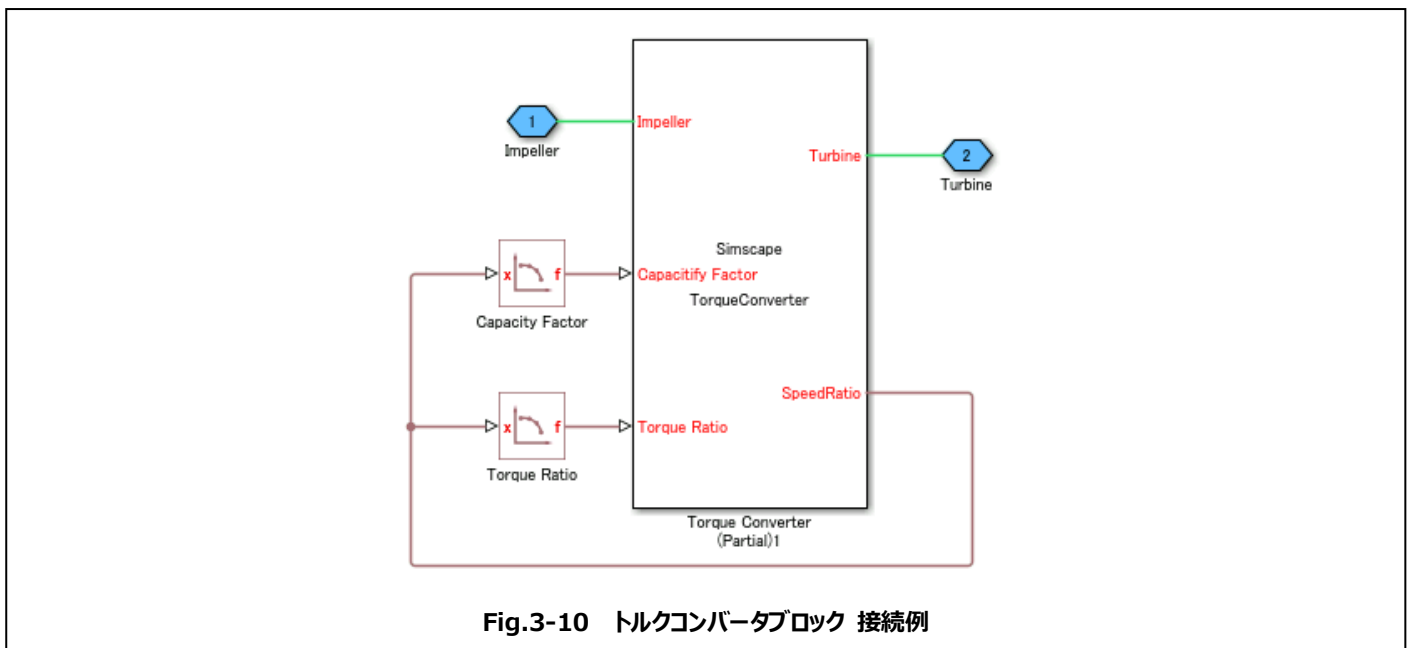
$$\tau = \frac{1 - \text{SpeedRatio}}{K^2}$$

また、本モデルはロックアップ機構を有していない。

(1) モデルブロック



(2) 接続例



(3) モデル詳細

名称	Torque Converter
概要	トルク増幅効果を模擬する簡単なトルクコンバーターモデル トルクの向き； すべてのポートで外から内が正 計算内容； ① 入力された容量係数からインペラー負荷トルクを計算（スレッシュホールドによるガードあり） ② インペラー負荷トルクから、入力されたトルク比をかけてタービン出力トルクを計算 ③ タービン回転数はインペラー回転数に速度比を書けたものになる。（スレッシュホールドによるガードあり）

ノード	ポート名	物理ドメイン	Through/Across	内容	表示位置		
	1	Impeller	機械・回転	トルク(Nm)、回転数 (rad/s)	インペラー側出力	left	
	2	Turbine	機械・回転	トルク(Nm)、回転数 (rad/s)	タービン側出力	right	
入力	ポート名	内容	範囲	初期値	単位	表示位置	
	1	Capacity Factor	容量係数	-	0	(rad/s)/(N*m)^0.5	left
	2	Torque Ratio	トルク比	-	0	-	left
出力	ポート名	内容	範囲	初期値	単位	表示位置	
	1	SpeedRatio	速度比	-	0	-	right
パラメータ	パラメータ名	内容	範囲	初期設定値	単位		
	1	SpeedRatio_th	スレッシュホールド(速度比)	0-1	0.97	-	
	2	K_th	スレッシュホールド(容量係数)	>0	50	(rad/s)/(N*m)^0.5	
	3	W_th	スレッシュホールド(速度)	>0	0.1*(2*pi)/60	rad/s	
	4						
変数	変数名	内容	範囲	初期値	単位	Port	
	1	T_impeller	インペラー負荷トルク	-	0	N*m	Impeller
	2	T_turbine	タービン出力トルク	-	0	N*m	Turbine
	3	W_impeller	インペラー速度	-	0	rad/s	Impeller
	4	W_turbine	タービン速度	-	0	rad/s	Turbine
	5						
	6						
	7						
方程式、その他							
<p>トルクコンバータにおける速度の関係より</p> $W_{turbine} == SpeedRatio * W_{impeller}$ <p>速度比と容量係数を用いてインペラー負荷トルクを計算</p> $T_{impeller} == sign(1 - SpeedRatio) * (W_{impeller} / K)^2$ <p>インペラー負荷トルクからタービン出力トルクを計算</p> $T_{turbine} == -T_{impeller} * TorqueRatio$							

(4) Simscape コード

```
component TorqueConverter
% Torque Converter
% Ver. 1.0 Jun/22, 2017 JMAAB Plant Modeling Workshop

parameters
    SpeedRatio_th = { 0.97, '1' }; % Threshold (Speed Ratio)
    K_th = { 50.00, '(rad/s)/(N*m)^0.5' }; % Threshold (Capacitify Factor)
    W_th = { 0.1*(2*pi)/60, 'rad/s' }; % Threshold (Angular Speed)
end

nodes
    Impeller = foundation.mechanical.rotational.rotational; % Impeller : left
    Turbine = foundation.mechanical.rotational.rotational; % Turbine : right
end

inputs
    K = { 0, '(rad/s)/(N*m)^0.5' }; % Capacitify Factor : left
    TorqueRatio = { 0, '1' }; % Torque Ratio : left
end

outputs
    SpeedRatio = { 0, '1' }; % SpeedRatio : right
end

variables
    T_impeller = { 0, 'N*m' }; % Torque (Impeller)
    T_turbine = { 0, 'N*m' }; % Torque (Turbine)
    W_impeller = { 0, 'rad/s' }; % Angular Speed (Impeller)
    W_turbine = { 0, 'rad/s' }; % Angular Speed (Turbine)
end

branches
    T_impeller : Impeller.t -> *;
    T_turbine : Turbine.t -> *;
end

equations
    W_impeller == Impeller.w;
    W_turbine == Turbine.w;
    if (abs(W_impeller) > W_th)
        W_turbine == SpeedRatio * W_impeller;
    else
        W_turbine == SpeedRatio * W_th;
    end
    if ((abs(W_turbine) + abs(W_impeller)) < W_th)
        T_impeller == 0;
    elseif (SpeedRatio < SpeedRatio_th)
        T_impeller == sign(1 - SpeedRatio) * (W_impeller / K)^2;
    else
        T_impeller == sign(1 - SpeedRatio) * (W_impeller / K_th)^2 * sqrt(abs(1 - SpeedRatio) / (1 - SpeedRatio_th));
    end
    T_turbine == - T_impeller * TorqueRatio;
end
```

タービン回転数計算

インペラー出カトルク計算

タービン出カトルク計算

3.3.4 トランスミッション

コントローラからの要求ギヤ段に応じて、トランスミッションへの入力トルクから、各ギヤ比に応じたトルクを出力するモデルとなる。本モデルにおいては、4速オートマチックトランスミッションとしており、各ギヤ比は 1st: 2.393, 2nd: 1.450, 3rd: 1.000, 4th: 0.577としている。

以下にトランスミッションモデルについて、説明する。

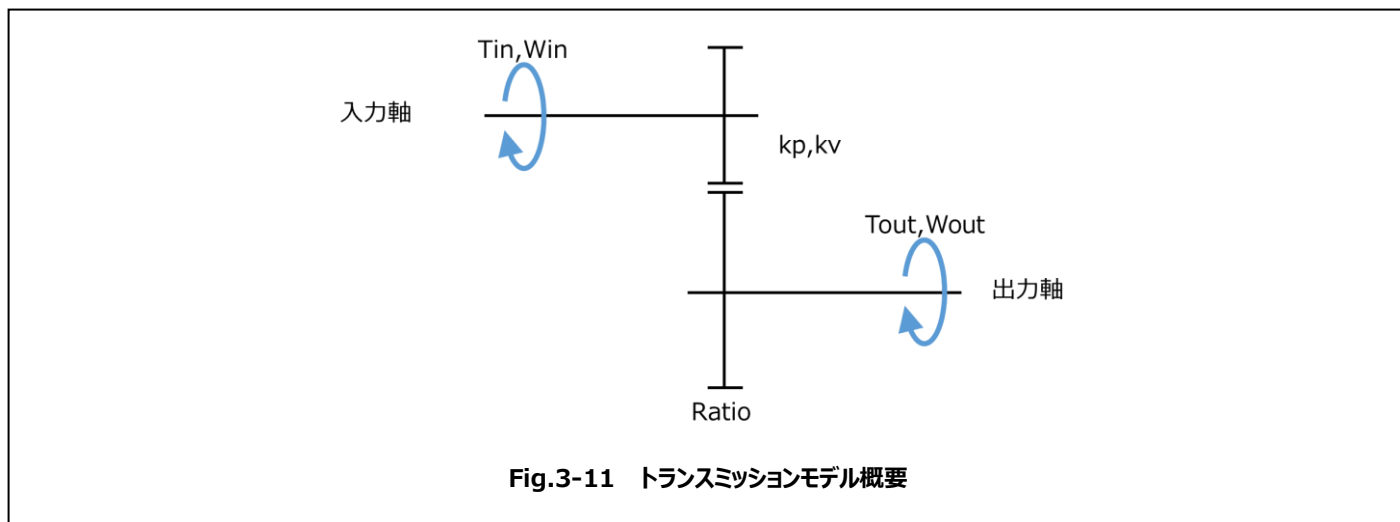


Fig.3-11 トランスミッションモデル概要

T_{in}/W_{in} は入力軸のトルク/回転数であり、 T_{out}/W_{out} は出力軸のトルク/回転数である。また、Ratio はギヤ比、 kp/kv は弾性係数/減衰係数である。

入出力軸の回転数差 $dph = Ratio \times W_{out} - W_{in}$ 、 dph の時間積分を ph と置くと、入力軸の運動方程式は以下である。

$$T_{in} + kp \cdot ph + kv \cdot dph = 0$$

T_{in} について解くと、以下の式が得られる。

$$T_{in} = -kp \cdot ph - kv \cdot dph$$

また、出力軸トルク T_{out} は、ギヤ比の関係から以下となる。

$$T_{out} = Ratio \cdot T_{in}$$

(1) モデルブロック

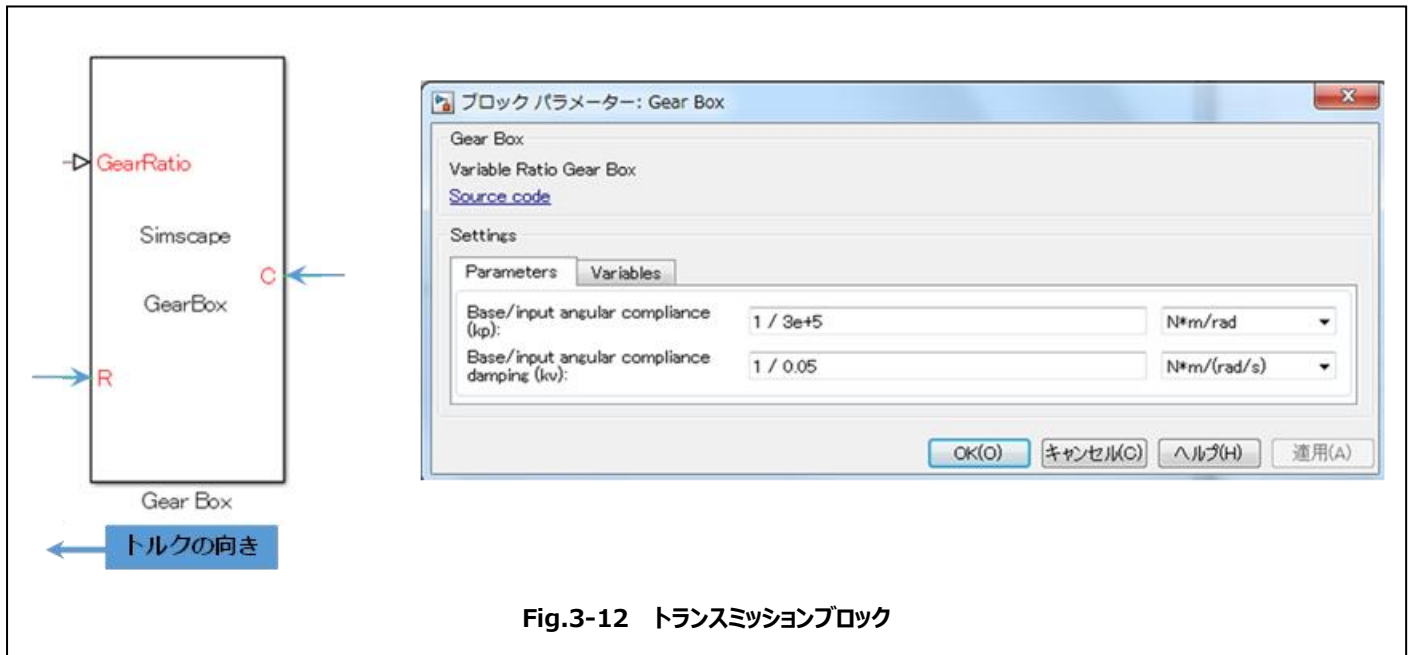


Fig.3-12 トランスミッションブロック

(2) 接続例

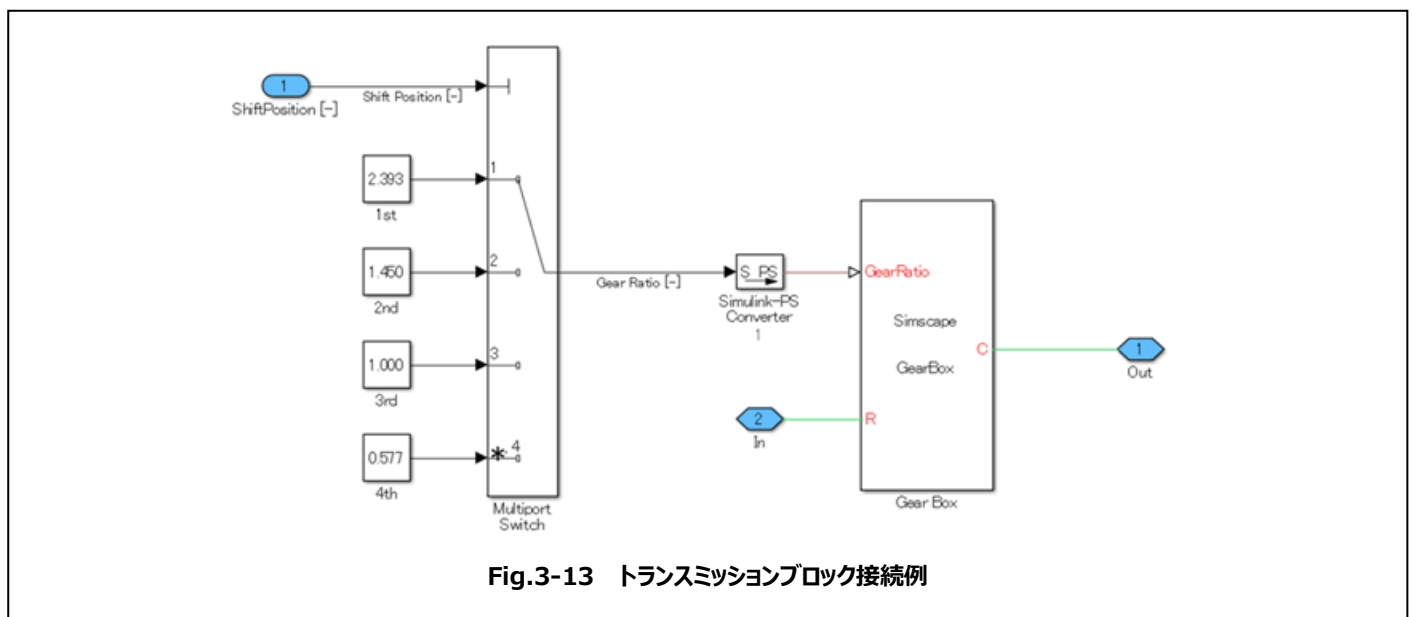


Fig.3-13 トランスミッションブロック接続例

(3) モデル詳細

名称	Gear Box				
概要	入力トルクをギヤ比倍して、出力する。				
	トルクの向き：本コンポーネントに入ってくる方のトルクを正とする。				
	伝達効率は未考慮				
ノード	ポート名	物理ドメイン	Through/Across	内容	表示位置

	1	R	機械回転	トルク(Nm), 回転数(rad/s)	入力軸(トルコン後ろ)	left	
	2	C	機械回転	トルク(Nm), 回転数(rad/s)	出力軸(デフ前)	right	
入力		ポート名	内容	範囲	初期値	単位	表示位置
	1	GearRatio	ギヤ比	-	2.393	-	right
パラメータ		パラメータ名	内容	範囲	初期設定値	単位	
	1	kp	弾性係数	-	1/3e+5	Nm/rad	
	2	kv	減衰係数	-	1/0.05	Nm/(rad/s)	
変数		変数名	内容	範囲	初期値	単位	
	1	T_in	入力軸トルク	-	0	Nm	
	2	T_out	出力軸トルク	-	0	Nm	
	3	W_in	入力軸回転数	-	0	rad/s	
	4	W_out	出力軸回転数	-	0	rad/s	
	5	ph	角度差	-	0	rad	
	6	dph	回転数差	-	0	rad/s	
	7						
方程式、その他							
トルクの関係式 :							
角度差・回転数差から入力軸トルクを算出する							
$T_{in} = -kp \cdot ph - kv \cdot dph$							
入力軸トルクをギヤ比倍して、出力軸トルクを算出する							
$T_{out} = -Ratio \cdot T_{in}$							

(4) Simscape コード

```
component GearBox
% Gear Box
% Ver. 1.0 Jun/22, 2017 JMAAB Plant Modeling Workshop

parameters
    kp = { 1/3e+5, 'N*m/rad' }; % Base/input angular compliance (kp)
    kv = { 1/0.05, 'N*m/(rad/s)' }; % Base/input angular compliance damping (kv)
end

inputs
    Ratio = { 2.393, '1' }; % GearRatio : left
end

nodes
    R = foundation.mechanical.rotational.rotational; % R : left
    C = foundation.mechanical.rotational.rotational; % C : right
end

variables
    T_in = { 0, 'N*m' };
    T_out = { 0, 'N*m' };
    W_in = { 0, 'rad/s' };
    W_out = { 0, 'rad/s' };
    ph = { 0, 'rad' }; % Compliance
    dph = { 0, 'rad/s' };
end

function setup
    ph = 0;
end

branches
    T_in : R.t -> *;
    T_out : C.t -> *;
end

equations
    W_in == R.w;

    W_out == C.w;

    dph == der(ph);

    dph == Ratio*W_out - W_in;

    T_in == -kp*ph - kv*dph;

    T_out == -Ratio*T_in;
end
```

Gear Box へ入ってくるトルクを正

入力トルクは角度差・回転数差から算出する

入力トルクをギヤ比倍して、出力トルクとする

3.3.5 プラネタリギヤ

キャリアへの入力トルクをサンギヤ・リングギヤに分割して出力する簡易モデルとなる。伝達効率は未考慮である。以下にプラネタリギヤモデルについて説明する。

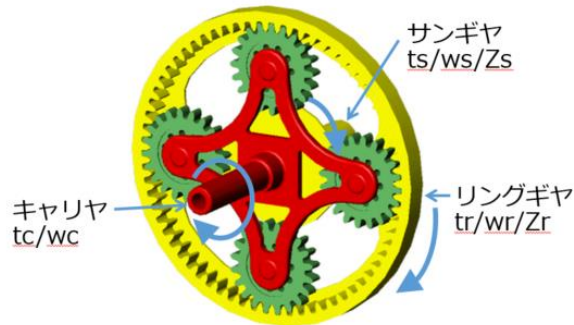


Fig.3-14 プラネタリギヤ

tc/wc はキャリアのトルク/回転数である。 $ts/ws/Zs$ はサンギヤのトルク/回転数/歯数である。 $tr/wr/Zr$ はリングギヤのトルク/回転数/歯数である。

プラネタリギヤのサンギヤ・キャリア・リングギヤの回転数の関係は、いずれか 2 つの回転数が決まれば、残る 1 つの回転数は一意に決まる。以下の共線図を描くと理解しやすい。

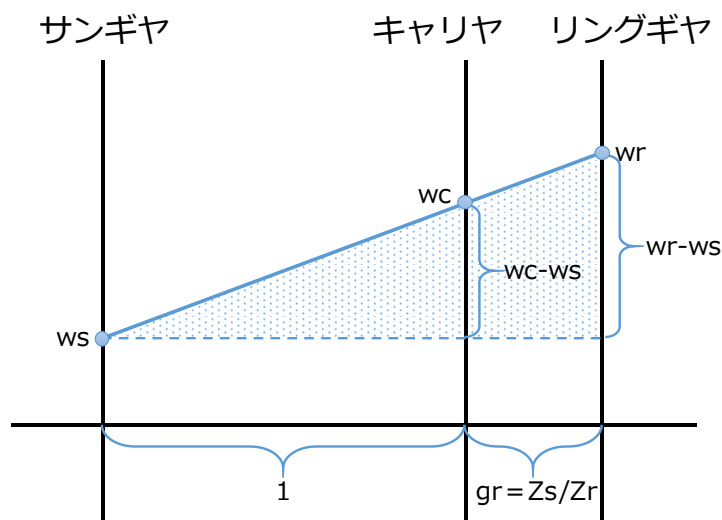


Fig.3-15 プラネタリギヤ共線図 (回転数)

gr はサンギヤの歯数/リングギヤである。回転数の関係式について、導出方法の一例を以下に示す。青で網掛けした三角形の部分の相似の関係から以下の比が得られる。

$$1 : 1 + gr = (w_c - w_s) : (w_r - w_s)$$

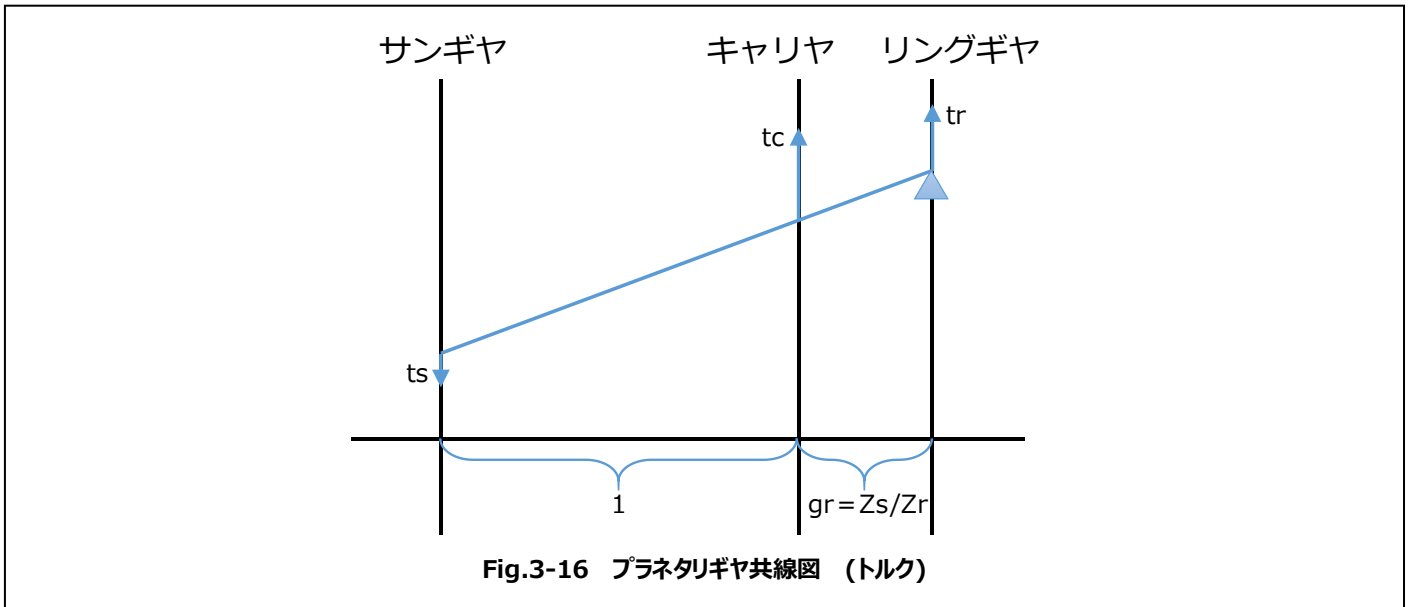
w_c について解く。

$$w_c - w_s = \frac{w_r - w_s}{1 + gr}$$

$$w_c = \frac{w_r - w_s}{1 + gr} + \frac{(1 + \rho)w_s}{1 + gr}$$

$$w_c = \frac{gr}{1 + gr} w_s + \frac{1}{1 + gr} w_r$$

上式がプラネタリギヤのサンギヤ・キャリア・リングギヤの回転数を表す関係式である。



トルクの関係式も共線図を描くと理解しやすい。

プラネタリギヤのトルク配分はテコの原理で考えられる。リングギヤを支点と考えると、以下の式が得られる。

$$(1 + gr) \cdot t_s = gr \cdot t_c$$

t_s について解くと、次のようになる。

$$t_s = \frac{gr}{1 + gr} t_c$$

同様にサンギヤを支点と考えると、次のようになる。

$$(1 + gr) \cdot t_r = t_c$$

更に、 t_r について解くと、次のように表せる。

$$t_r = \frac{1}{1 + gr} t_c$$

(1) モデルブロック

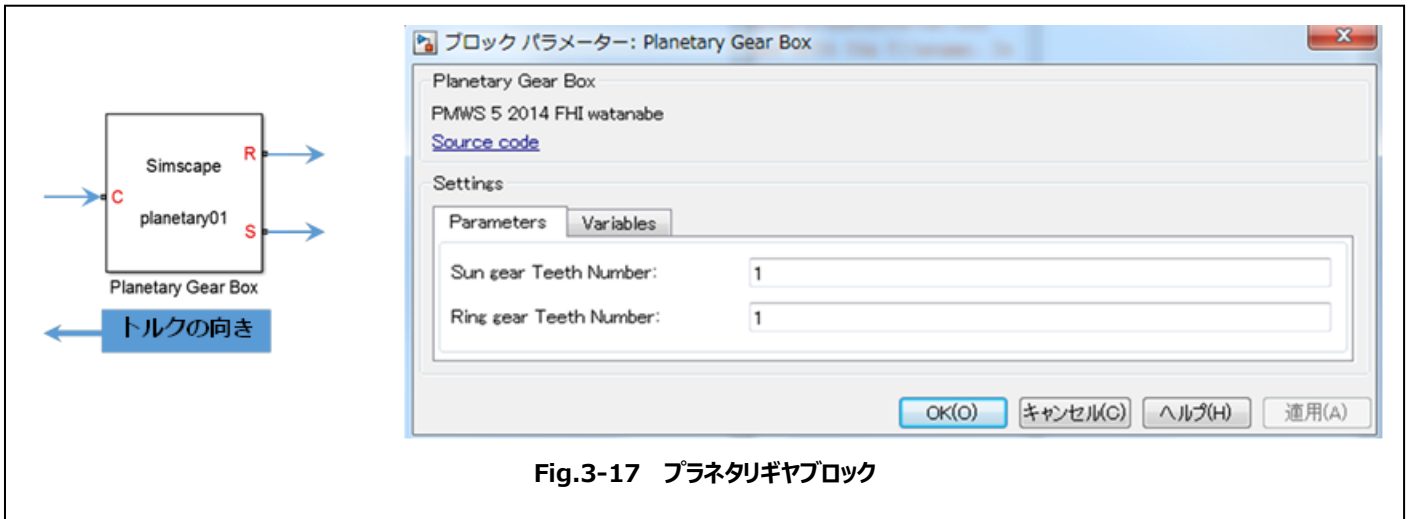


Fig.3-17 プラネタリギヤブロック

(2) 接続例

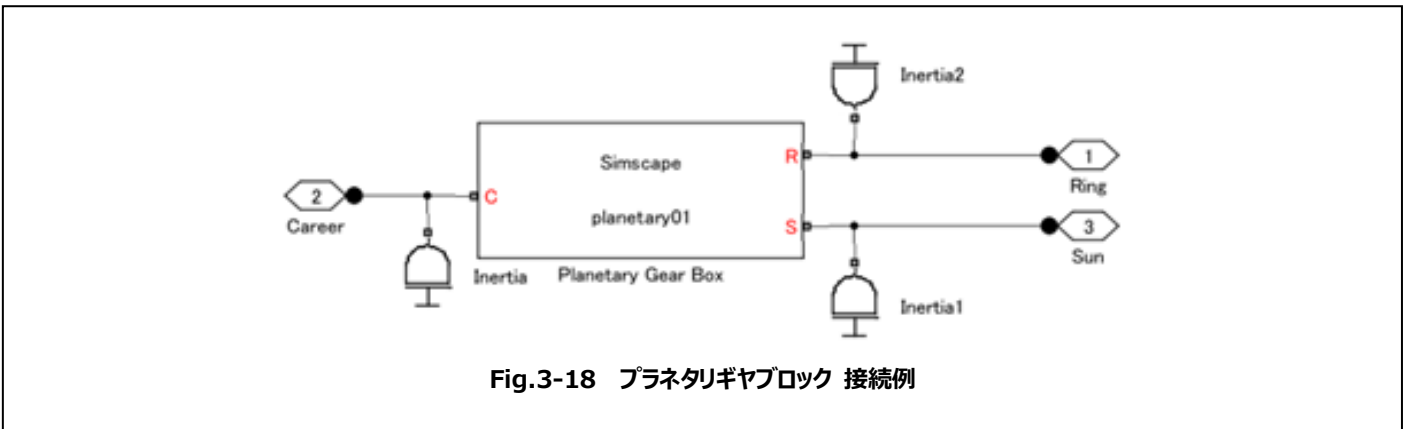


Fig.3-18 プラネタリギヤブロック 接続例

(3) モデル詳細

名称	Planetary Gear					
概要	キャリア(入力)トルクを、サンギヤ・リングギヤ(出力)トルクに分割する。					
	トルクの向き：車両を前進させる時のトルクを正とする。					
	伝達口は未考慮。					
ノード		ポート名	物理ドメイン	Through/Across	内容	表示位置
	1	C	機械回転	トルク(Nm), 回転数(rad/s)	キャリア軸	left
	2	R	機械回転	トルク(Nm), 回転数(rad/s)	リングギヤ軸	right
	3	S	機械回転	トルク(Nm), 回転数(rad/s)	サンギヤ軸	right
パラメータ		パラメータ名	内容	範囲	初期設定値	単位
	1	Zs	サンギヤ歯数	-	1	-

	2	Zr	リングギヤ歯数	-	1	-	
変数		変数名	内容	範囲	初期値	単位	
	1	tc	キャリアトルク	-	0	Nm	
	2	tr	リングギヤトルク	-	0	Nm	
	3	ts	サンギヤトルク	-	0	Nm	
	4	wc	キャリア回転数	-	0	rad/s	
	5	wr	リングギヤ回転数	-	0	rad/s	
	6	ws	サンギヤ回転数	-	0	rad/s	
	7	gr	サンギヤ/リングギヤ比	-	1	-	

方程式、その他

トルクの関係式

$$t_s = \frac{gr}{1+gr} t_c \quad t_r = \frac{1}{1+gr} t_c \quad gr = \frac{Z_s}{Z_r}$$

回転数の関係式

$$w_c = \frac{gr}{1+gr} w_s + \frac{1}{1+gr} w_r$$

(4) Simscape コード

```
component PlanetaryGear
% Planetary Gear
% Ver. 1.0 Jun/22, 2017 JMAAB Plant Modeling Workshop

nodes
    C = foundation.mechanical.rotational.rotational; % C : left
    R = foundation.mechanical.rotational.rotational; % R : right
    S = foundation.mechanical.rotational.rotational; % S : right
end

parameters
    Zs = { 1 '1' }; % Sun gear Teeth Number
    Zr = { 1 '1' }; % Ring gear Teeth Number
end

variables
    tc = { 0 'N*m' };
    tr = { 0 'N*m' };
    ts = { 0 'N*m' };
    wc = { 0 'rad/s' };
    wr = { 0 'rad/s' };
    ws = { 0 'rad/s' };
    gr = { 1 '1' };
end

branches
    tc : C.t -> *;
    ts : * -> S.t;
    tr : * -> R.t;
end

equations
    wc == C.w;
    ws == S.w;
    wr == R.w;

    gr == Zs/Zr;
    ts == gr/(1+gr)*tc;
    tr == 1/(1+gr)*tc;

    wc == gr/(1+gr)*ws + 1/(1+gr)*wr;
end
```

キャリアへの入力を正
サンギヤ・リングギヤへの出力を正

キャリアトルクをサンギヤリングギヤに分割

3.3.6 デイファレンシャル

通称デフと呼ばれるデイファレンシャルは、自動車がスムーズに曲がるために必要な差動機構である。本モデルは、入力トルクに対して左右輪にトルクを分配する挙動を模擬した簡易的なモデルであり、ギヤ効率およびギヤ比はパラメータとして設定できるようにしている。

Fig.3-19 のようなケースを考えると、左側の車輪回転数 w_{s1} [rad/s]は、

$$w_{s1} = (\text{リングギヤ公転回転数}) + (\text{ピニオンギヤ自転回転数}) \times \frac{(\text{ピニオン歯数})}{(\text{サイドギヤ歯数})}$$

$$\therefore w_{s1} = \frac{w_{in}}{Rin} + w_p \cdot Rps$$

ここで、

w_{in} :デフ入力ギヤ角速度 [rad/s], Rin :デフギヤ比 [-],

w_p :ピニオンギヤ角速度 [rad/s], Rps :ピニオン/サイドギヤ比 [-]

となる。また、右側の車輪回転数 w_{s2} [rad/s]は、逆に自転分回転数が減ることになるので、

$$w_{s2} = \frac{w_{in}}{Rin} - w_p \cdot Rps$$

となる。また、入力と出力のトルクの総和の釣り合いから、

$$t_{out1} + t_{out2} + t_{in} \cdot \text{Eta} \cdot Rin = 0$$

ここで、

t_{out1} :左輪出力トルク [Nm], t_{out2} :右輪出力トルク [Nm], t_{in} :デフ入力トルク [Nm], Eta :デフ効率 [-]

となり、入力トルクの内の $(1-\text{Eta}) \cdot 100\%$ 分はロスとして消費されることになる。

また、ピニオンギヤの自転に着目した運動方程式を立てると、左右輪のトルク差により回転数が発生することから、次のようになる。

$$I_p \cdot \frac{dw_p}{dt} = Rps \cdot (t_{out1} - t_{out2})$$

ここで、

I_p : ピニオンギヤ慣性モーメント [kgm²]

である。

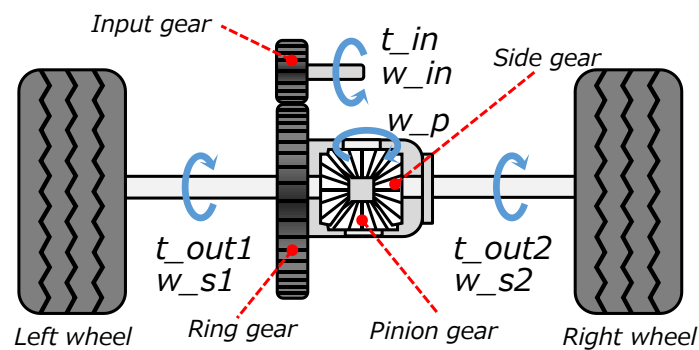


Fig.3-19 デイファレンシャル

(1) モデルブロック

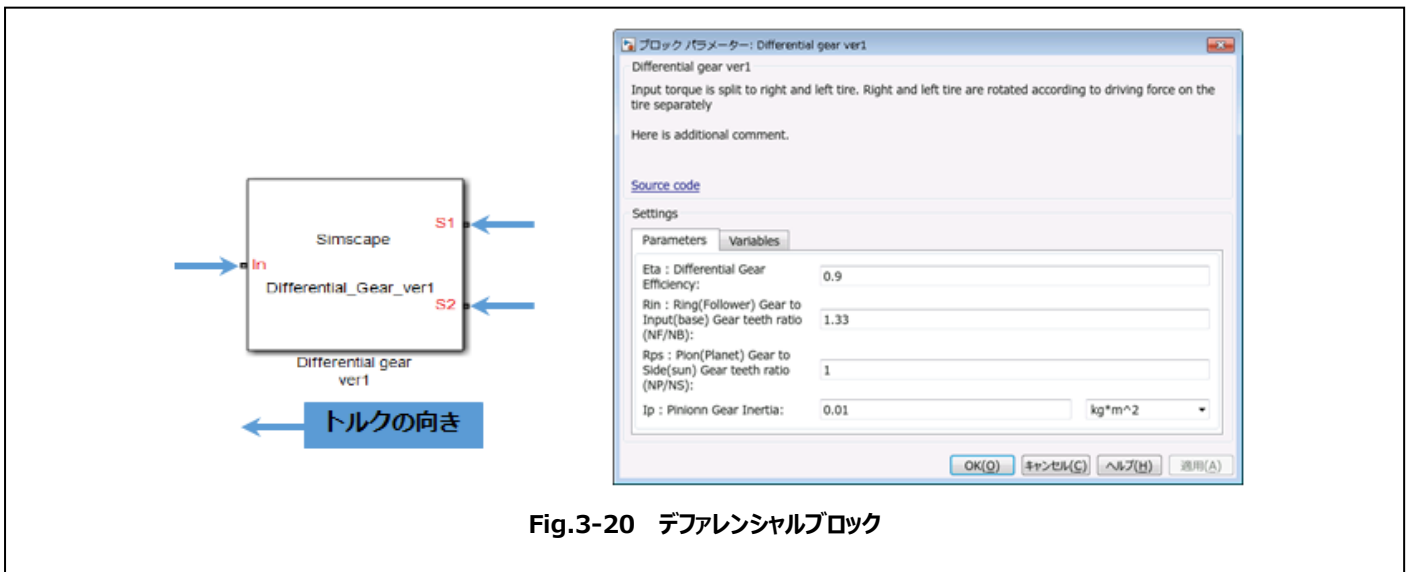


Fig.3-20 デファレンシャルブロック

(2) 接続例

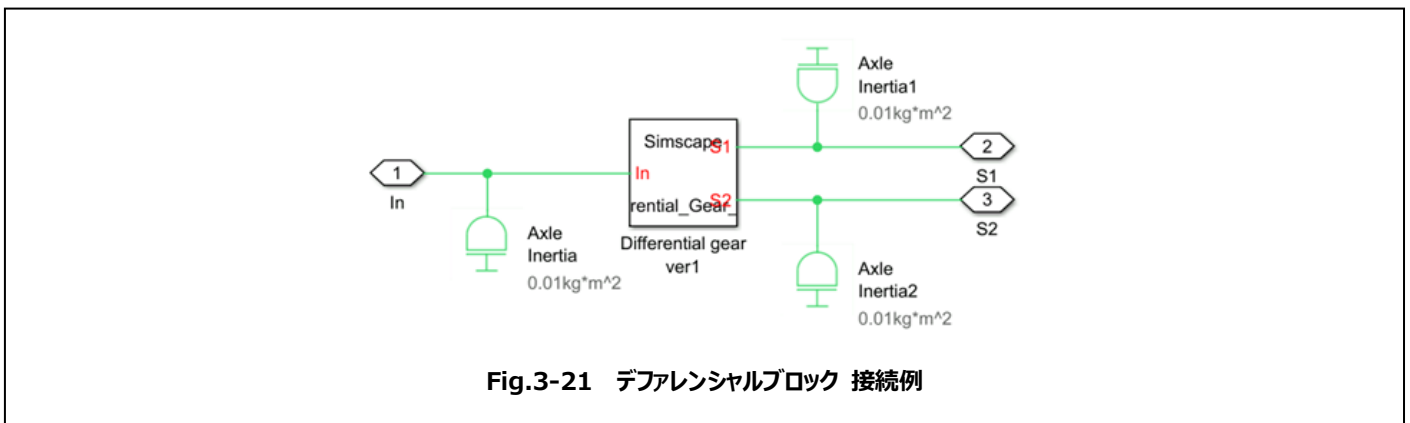


Fig.3-21 デファレンシャルブロック 接続例

(3) モデル詳細

名称	Differential Gear				
概要	回転、トルクのみを考慮した単純な Differential Gear モデル トルクの向き； すべてのポートで外から内が正 計算内容； ① 入力されたパワートレートルから効率分を差し引いて左右輪に分配 ② 左右輪のドライビングトルクの差から、ピニオンギア回転数を計算 ③ ピニオンギア回転数から左右輪の回転数を算出				
ノード	ポート名	物理ドメイン	Through/Across	内容	表示位置
	1 In	機械・回転	トルク(Nm), 回転数(rad/s)	パワートレートルク 入力	left
	2 S1	機械・回転	トルク(Nm), 回転数(rad/s)	左(右)輪への出力	right

	3	S2	機械・回転	トルク(Nm), 回転数(rad/s)	(左)右輪への出力	right	
パラメータ		パラメータ名	内容	範囲	初期設定値	単位	
	1	Eta	デフ効率	0-1	0.9	1	
	2	Rin	デフギア比	>0	1.33	1	
	3	Rps	ピニオン/サイドギア比	>0	1	1	
	4	Ip	ピニオンギアイナーシャ	>0	0.01	kgm ²	
変数		変数名	内容	範囲	初期値	単位	Port
	1	t_in	パワートレーン入力トルク	-	0	Nm	In
	2	t_out1	左(右)輪への出力トルク	-	0	Nm	S1
	3	t_out2	(左)右輪への出力トルク	-	0	Nm	S2
	4	w_in	デフ入力ギア角速度	-	0	rad/s	In
	5	w_s1	左(右)輪角速	-	0	rad/s	S1
	6	w_s2	(左)右輪角速	-	0	rad/s	S2
	7	w_p	ピニオンギア角速度	-	0	rad/s	

方程式、その他

デフにおけるエネルギー保存則より

$$t_out1 + t_out2 + t_in * Eta * Rin = 0$$

左右輪のトルク差からピニオンギアの角速度を計算

$$w_p.der = Rps / Ip * (t_out1 - t_out2)$$

ピニオンギア角速度より、左右輪角速度を計算

$$w_s1 == 1 / Rin * w_in + Rps * w_p$$

$$w_s2 == 1 / Rin * w_in - Rps * w_p$$

(4) Simscape コード

```
component DifferentialGear
% Differential Gear
% Ver. 1.0 Jun/22, 2017 JMAAB Plant Modeling Workshop

nodes
    In = foundation.mechanical.rotational.rotational; % In:left
    Out1 = foundation.mechanical.rotational.rotational; % S1:right
    Out2 = foundation.mechanical.rotational.rotational; % S2:right
end

parameters
    Eta = { 0.9, '1' }; % Eta : Differential Gear Efficiency
    Rin = { 1.33, '1' }; % Rin : Ring(Follower) Gear to Input(base) Gear teeth ratio (NF/NB)
    Rps = { 1, '1' }; % Rps : Pion(Planet) Gear to Side(sun) Gear teeth ratio (NP/NS)
    Ip = { 0.01, 'kg*m^2' }; % Ip : Pinion Gear Inertia
end

variables
    t_in = { 0, 'N*m' };
    t_out1 = { 0, 'N*m' };
    t_out2 = { 0, 'N*m' };
    w_in = { 0, 'rad/s' }; %
    w_s1 = { 0, 'rad/s' }; %
    w_s2 = { 0, 'rad/s' }; %
    w_p = { 0, 'rad/s' }; % w_p : Pinion Gear Speed
end

function setup
    % Parameter range checking
    if Eta < 0
        pm_error('simscape:GreaterThanZero','Eta')
    end
    if Eta > 1
        pm_error('simscape:LessThanOne','Eta')
    end
    if Rin <= 0
        pm_error('simscape:GreaterThanZero','Final gear ratio')
    end
    if Rps <= 0
        pm_error('simscape:GreaterThanZero','PtoS gear ratio')
    end
    if Ip <= 0
        pm_error('simscape:GreaterThanZero','Pinion Gear Inertia')
    end
end

branches
    t_in : In.t -> *;
    t_out1 : Out1.t -> *;
    t_out2 : Out2.t -> *;
end

equations
    t_out2 + t_out1 + t_in * Eta * Rin == 0;
    w_in == In.w;
    w_p.der == Rps / Ip * (t_out1 - t_out2);
    Out1.w == 1 / Rin * w_in + Rps * w_p;
    Out2.w == 1 / Rin * w_in - Rps * w_p;
    w_s1 == Out1.w;
    w_s2 == Out2.w;
end

end
```

3.3.7 バッテリー

モータ、ジェネレータへの入出力電流に応じて、バッテリーの充電率（State of Charge;以下 SOC）を計算して、コントローラへ出力する簡易モデルとなっている。

バッテリー出力電圧は、リチウムイオン電池の SOC-OCV(Open Circuit Voltage;開回路電圧)曲線から OCV を外部より入力し、バッテリーにおける入出力電流値、バッテリー内部抵抗により計算される。

尚、実際のバッテリーは、温度によって内部抵抗が変化して、それに伴い出力電圧、電流が変化するが、本モデルでは一切考慮していない。

バッテリー充電率SOC[%]を求めるために、バッテリーに蓄えられた電荷量を計算する。バッテリーにおける電流収支のエネルギー保存則より、

$$I_b + dQ_{internal}/dt = 0$$

となる。

ここで、 I_b はバッテリー外部との入出力電流、 $Q_{internal}$ は、実際のバッテリー電荷量。尚、放熱によるエネルギーロスは考慮していない。

バッテリー充電率SOC[%] は、求められた現在の電荷量より、以下の式から計算する。

$$SOC = Q_{current} / Q_{max} * 100$$

ここで、 $Q_{current}$ は実際のバッテリー電荷量 $Q_{internal}$ をバッテリー容量 Q_{max} で制限したものである。

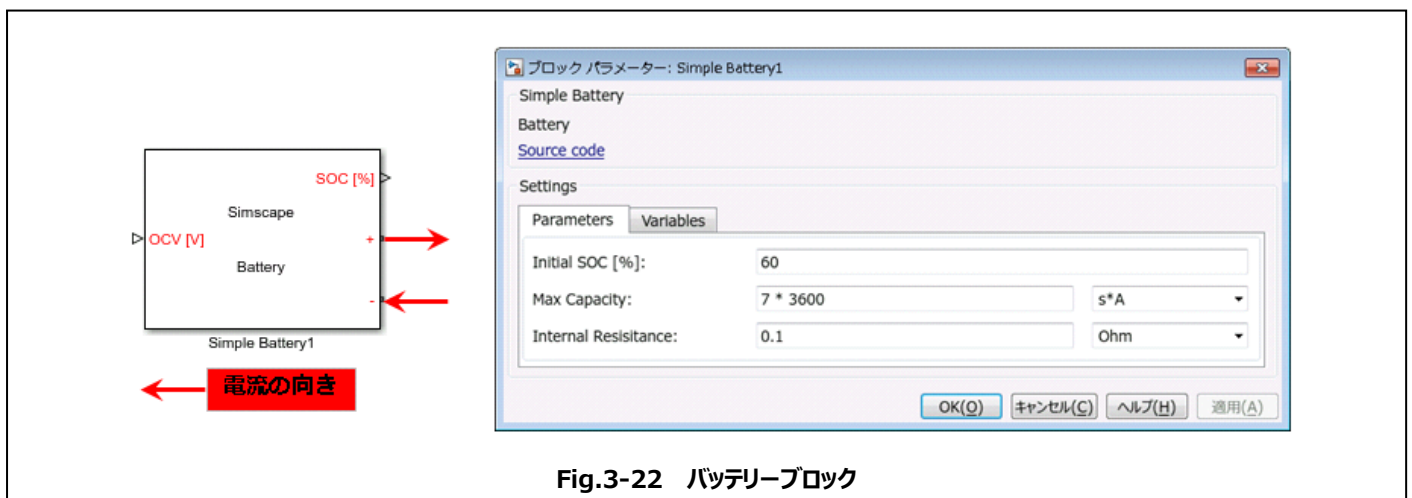
また、バッテリー出力電圧 V_b は、バッテリー開路電圧OCV(Open Circuit Voltage)からバッテリー内部抵抗 R と電流 I_b から求まる電圧成分を加減することにより求める。

$$V_b = OCV - R * I_b$$

[備考]

実際のバッテリーは、電流が供給され続けた場合でも、熱エネルギーなどに変換され SOC100%相当の電荷量を超えないが、本モデルでは計算の制約上バッテリー内部電荷量 $Q_{internal}$ が SOC100%を超えてしまう場合がある。ただシミュレーション上、コントローラで SOC100%にはならないように制御されるため、使用上は問題ないと思われる。

(1) モデルブロック



(2) 接続例

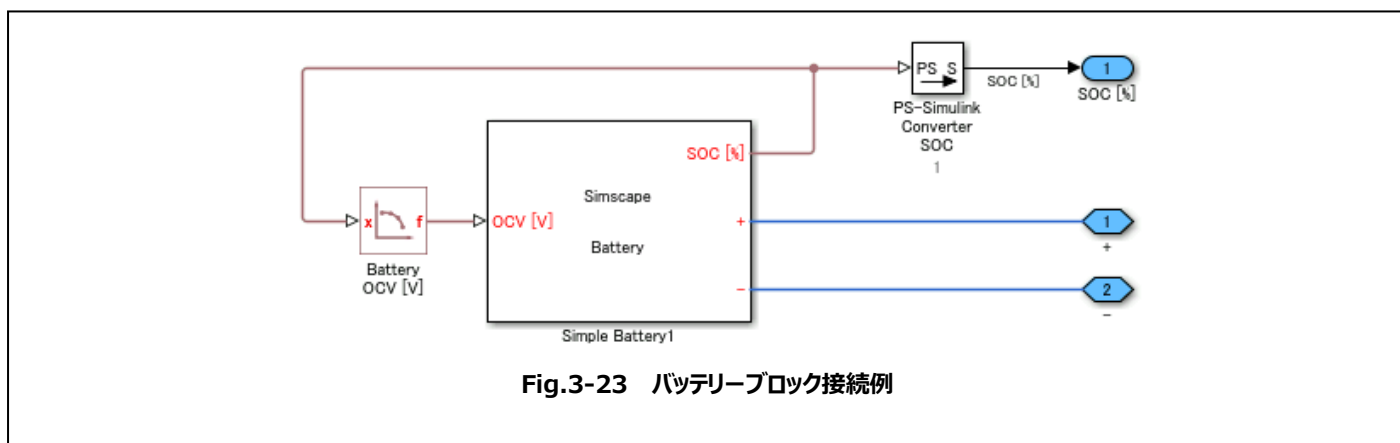


Fig.3-23 バッテリーブロック接続例

(3) モデル詳細

名称	Battery						
概要	バッテリーにおける入出力電流に応じて、バッテリー充電率を計算して、コントローラへ出力						
	電流の向き；バッテリーからの放電電流を正						
	計算内容；						
	①バッテリー出力電圧は、バッテリー内部抵抗、電流によりバッテリー開路電圧から変化						
	②バッテリー充電率 SOC(%)は、初期値から入出力電流に応じて算出、更新						
③バッテリー開路電圧は、SOCに応じて変化（ただし本モデル外部の PS Lookup Table で対応）							
ノード	ポート名	物理ドメイン	Through/Across		内容(接続先)	表示位置	
	1	p(+)	電気	電流 (A) 、電圧 (V)		バッテリー正極端子	right
	2	n(-)	電気	電流 (A) 、電圧 (V)		バッテリー負極端子	right
入力	ポート名	内容	範囲	初期値	単位	表示位置	
	1	OCV	バッテリー開路電圧	0~100	0	V	left
出力	ポート名	内容	範囲	初期値	単位	表示位置	
	1	SOC	バッテリー充電率	0~100	0	%	right
パラメータ	パラメータ名	内容	範囲	初期設定値	単位		
	1	SOC_init	初期充電率	0~100	60	%	
	2	Qmax	最大容量	>0	7*3600	A*s	
	3	R	内部抵抗	>=0	0.1	Ω	
変数	変数名	内容	範囲	初期値	単位		
	1	Ib	電流	-	0	A	
	2	Vb	バッテリー電圧	-	0	V	
	3	Q_internal	電荷量 (中間変数)	-	SOC_init/100*Qmax	A*s	
	4	Q_current	電荷量	0~Qmax	0	A*s	
方程式、その他							

<p>バッテリー電荷保存 (SOC 計算のために電流による電荷量変化量を計算)</p> $I_b + dQ_internal/dt == 0$
<p>Q_{cur} と Q_{in} の関係 (電荷量 (中間変数) を最小値、最大値で制限)</p> $Q_current == \min(Q_{max}, \max(0, Q_internal))$
<p>バッテリー端子電圧 (バッテリー出力電圧は、バッテリー内部抵抗、電流によりバッテリー開路電圧から変化)</p> $V_b == OCV - R * I_b$
<p>バッテリー端子電位差</p> $V_b == p.v - n.v$
<p>バッテリー充電率</p> $SOC == Q_current / Q_{max} * 100$

(4) Simscape コード

```
component Battery
% Simple Battery
% Ver. 1.0 Jun/22, 2017 JMAAB Plant Modeling Workshop

parameters
    SOC_init = { 60, '1' }; % Initial SOC [%]
    Qmax      = { 7*3600, 'A*s' }; % Max Capacity
    R         = { 0.1, 'Ohm' }; % Internal Resistance
end

inputs
    % Battery Open Circuit Voltage (OCV)
    OCV = { 0, 'V' }; % OCV [V] : left
end

outputs
    SOC = { 0, '1' }; % SOC [%] : right
end

nodes
    p = foundation.electrical.electrical; % + : right
    n = foundation.electrical.electrical; % - : right
end

variables
    Ib      = { 0, 'A' }; % Current
    Vb      = { 0, 'V' }; % Voltage
    Q_internal = { 0, 'A*s' }; % Electric Charge (Internal)
    Q_current  = { 0, 'A*s' }; % Electric Charge
end

function setup
    Q_internal = SOC_init / 100 * Qmax; % Q_in initial state
end

branches
    lb : n.i -> p.i;
end

equations
    Vb == p.v - n.v;

    Ib + Q_internal.der == 0; % 電荷保存

    if (Q_internal > 0.9999*Qmax)
        Q_current == Qmax;
    elseif (Q_internal < 0)
        Q_current == 0;
    else
        Q_current == Q_internal;
    end

    Vb == OCV - R*Ib;

    SOC == Q_current / Qmax * 100;
end
end
```

3.3.5 HEV 制御モデル

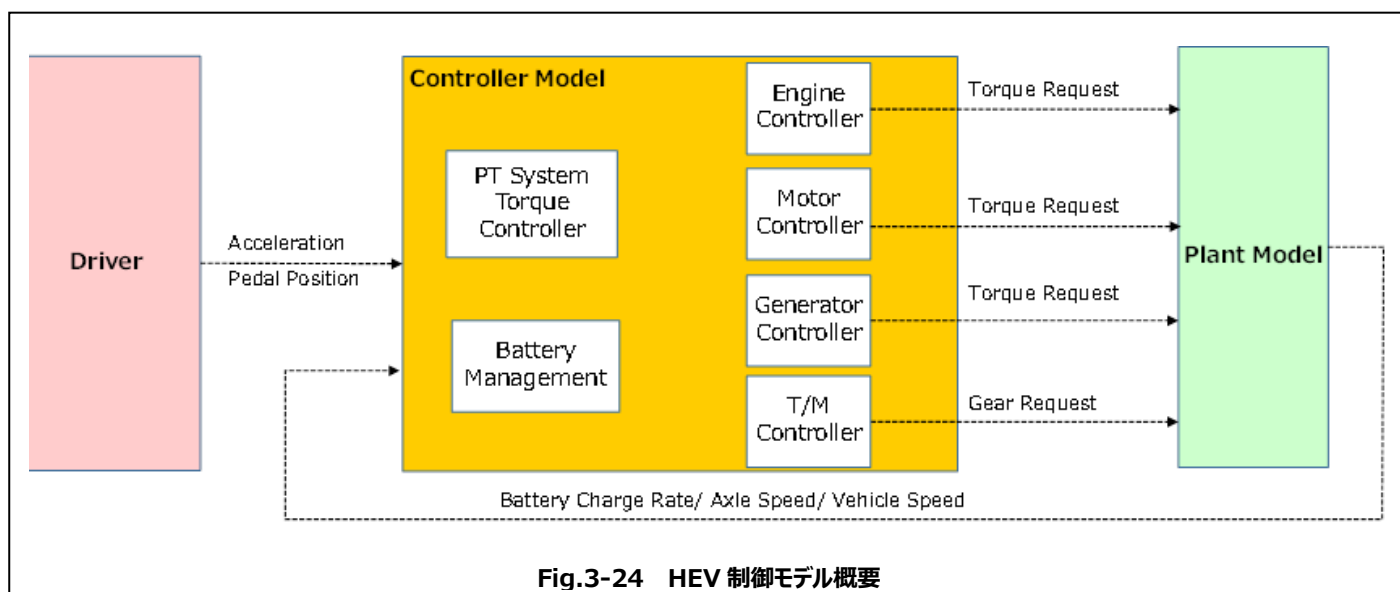
今回作成した HEV パワートレインのプラントモデルの動作検証にはコントローラモデルが必要となる。もちろん、各コンポーネントを接続して大規模モデル化する前に、コンポーネント単体での動作確認は済んでいることが望ましい。

動作検証に使用するソフトウェアモデルは、実際の開発対象のものを接続しても良いが、仮に初期動作の確認時に問題があった場合、プラント側に問題があるのかソフトウェアモデル側に問題があるのか原因特定が困難になる場合があるため、プラントモデルの初期動作確認時は、出来るだけシンプルなコントローラモデルが望ましい。

また、今回、Simulink で準備したコントローラモデルは、あらかじめ設定した目標車速に車両モデルが追従するようにドライバーモデルからアクセルペダル開度が指示され、そのアクセル開度に基づき簡易的に HEV パワートレインのトルクおよびバッテリー充電量を制御するため、プラントモデルに以下の要求を行う。尚、あくまでサンプルの簡易的なコントローラモデルであり、現実に沿っていない点や十分な動作検証がされていないことをご理解いただきたい。

- ・ギヤ変速指示
- ・走行時エンジントルク、モータトルク分配、要求
- ・減速時モータ回生トルク要求
- ・ジェネレータトルク要求

(1) モデル概要



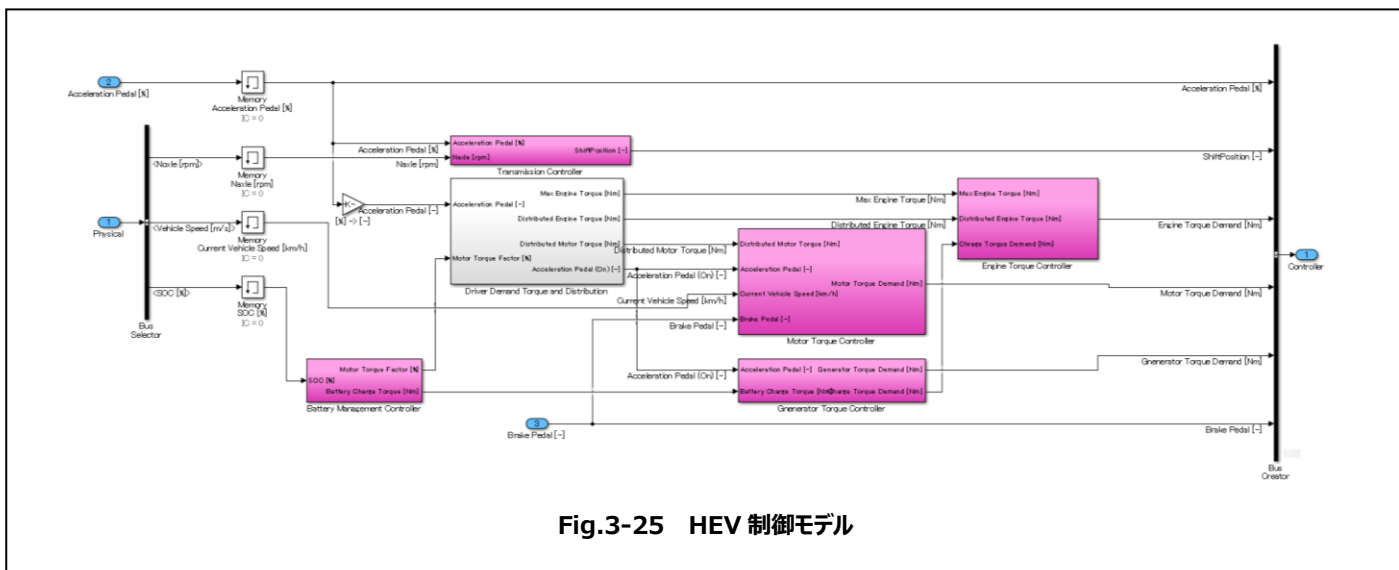


Fig.3-25 HEV 制御モデル

(2) 接続例

JMAAB プラントモデリングガイドラインでは、コントローラモデルは各コンポーネントの階層に配置することが推奨されているが、階層が深くなるデメリットがあり、現在のプラントモデル WS で議論した結果、目的に応じて使い分けることも必要との結論に達しているため、今回は階層が深くならず可読性が良い以下の方式を選択した。

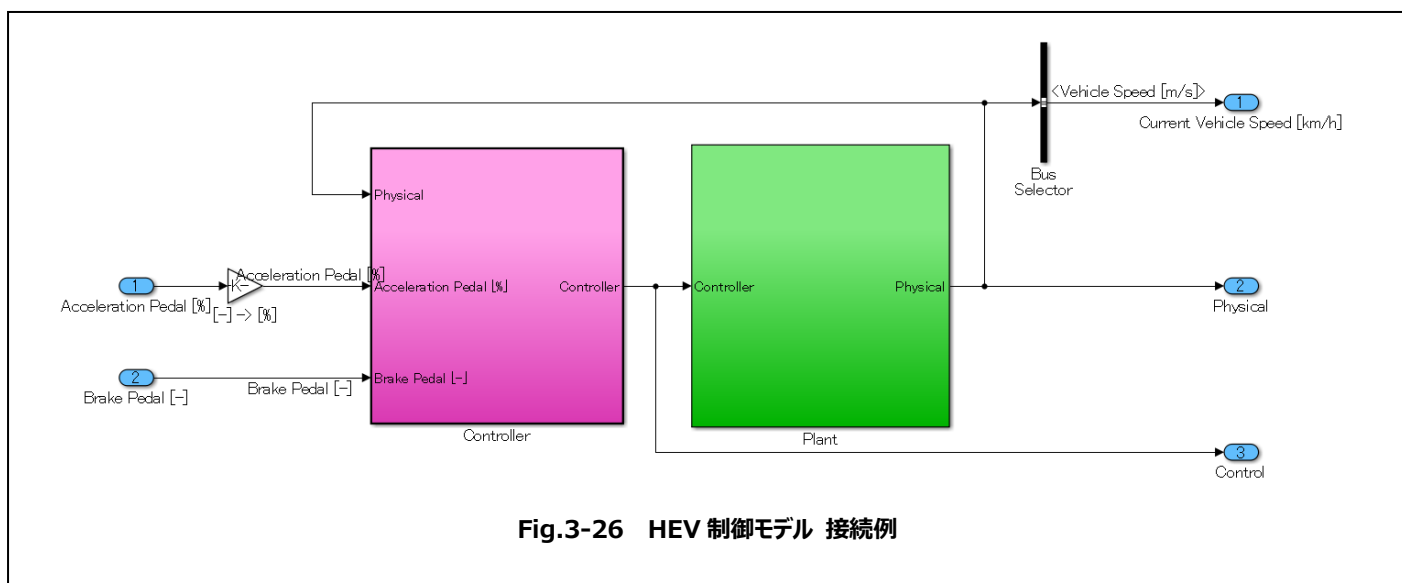
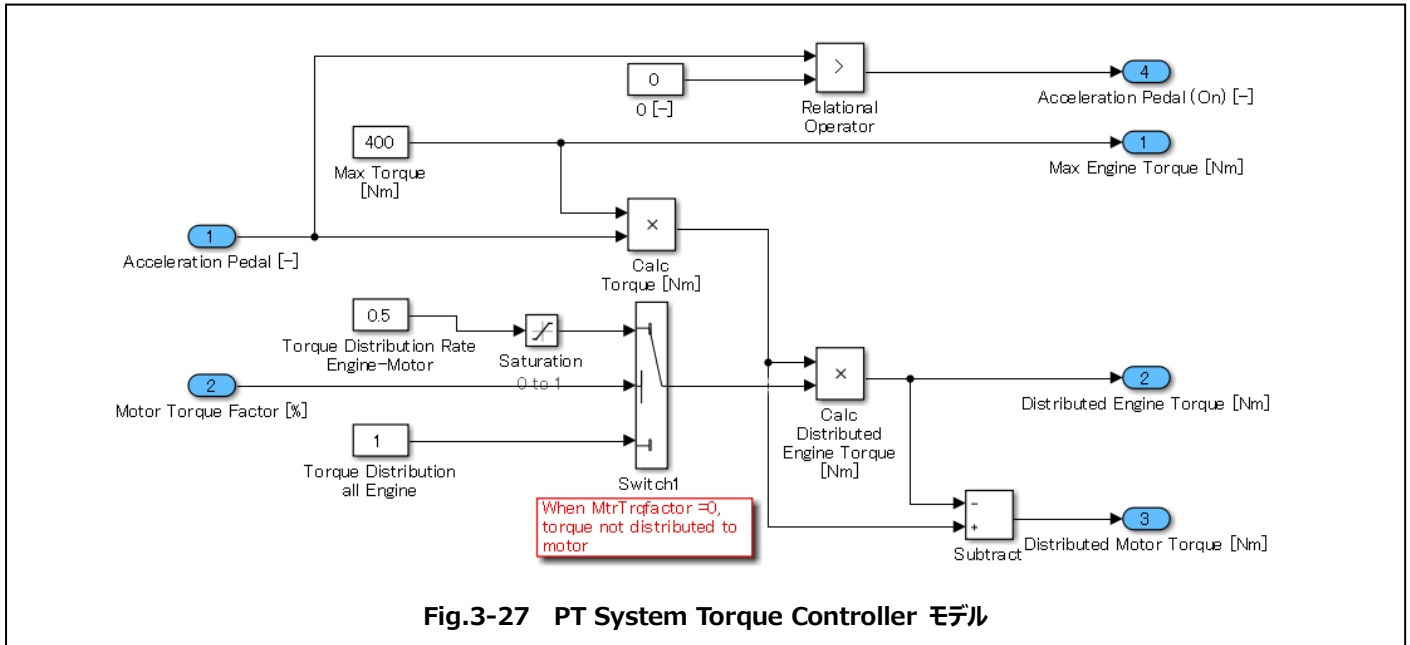


Fig.3-26 HEV 制御モデル 接続例

(3) モデル詳細

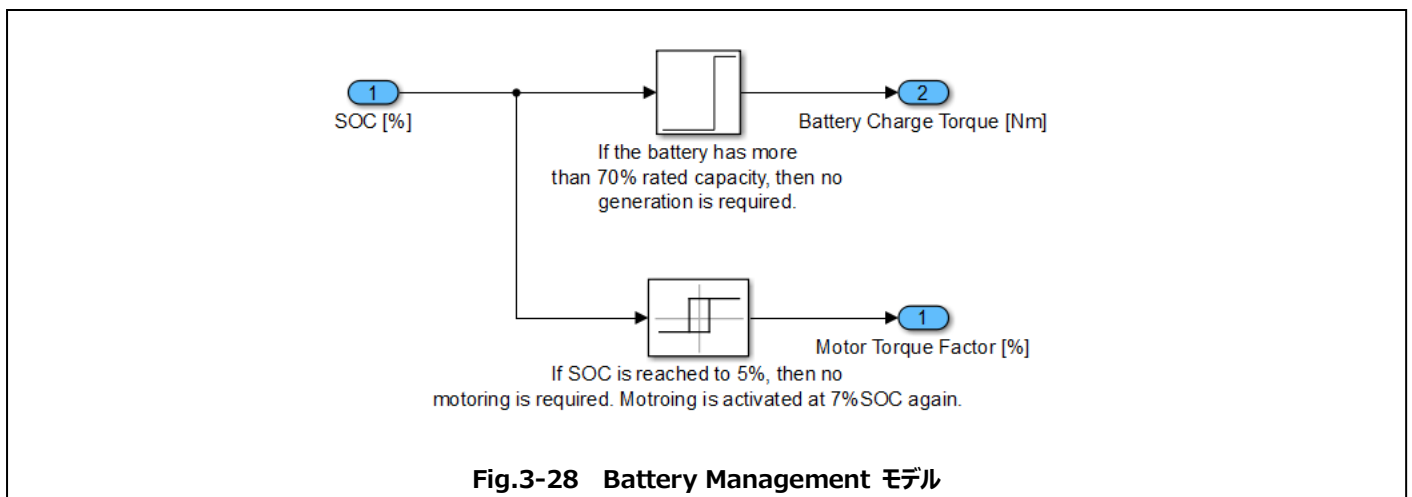
(3-1) PT System Torque Controller

設定した最大トルクにドライバアクセル開度の比率に乗じて、ドライバ要求トルクを算出し、エンジンとモータに走行に必要なトルクを配分する。配分比率は1 : 1としているが、バッテリー充電率が低くなった場合は、モーターアシストを停止し、エンジンのみで走行する。尚、EV 走行は考慮していない。



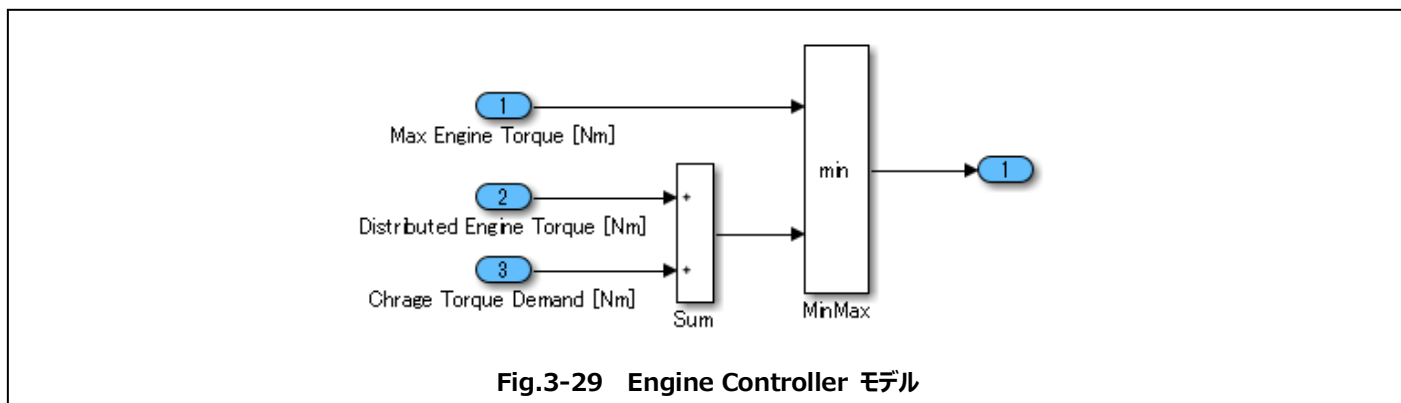
(3-2) Battery Management

バッテリー充電率が70%未満の場合、発電モータが一定トルクで発電し、70%以上の場合は過充電防止のため、発電モータトルクを0にする。また、バッテリー充電率が低い場合（5%未満）走行モータへの要求トルクを0にする。



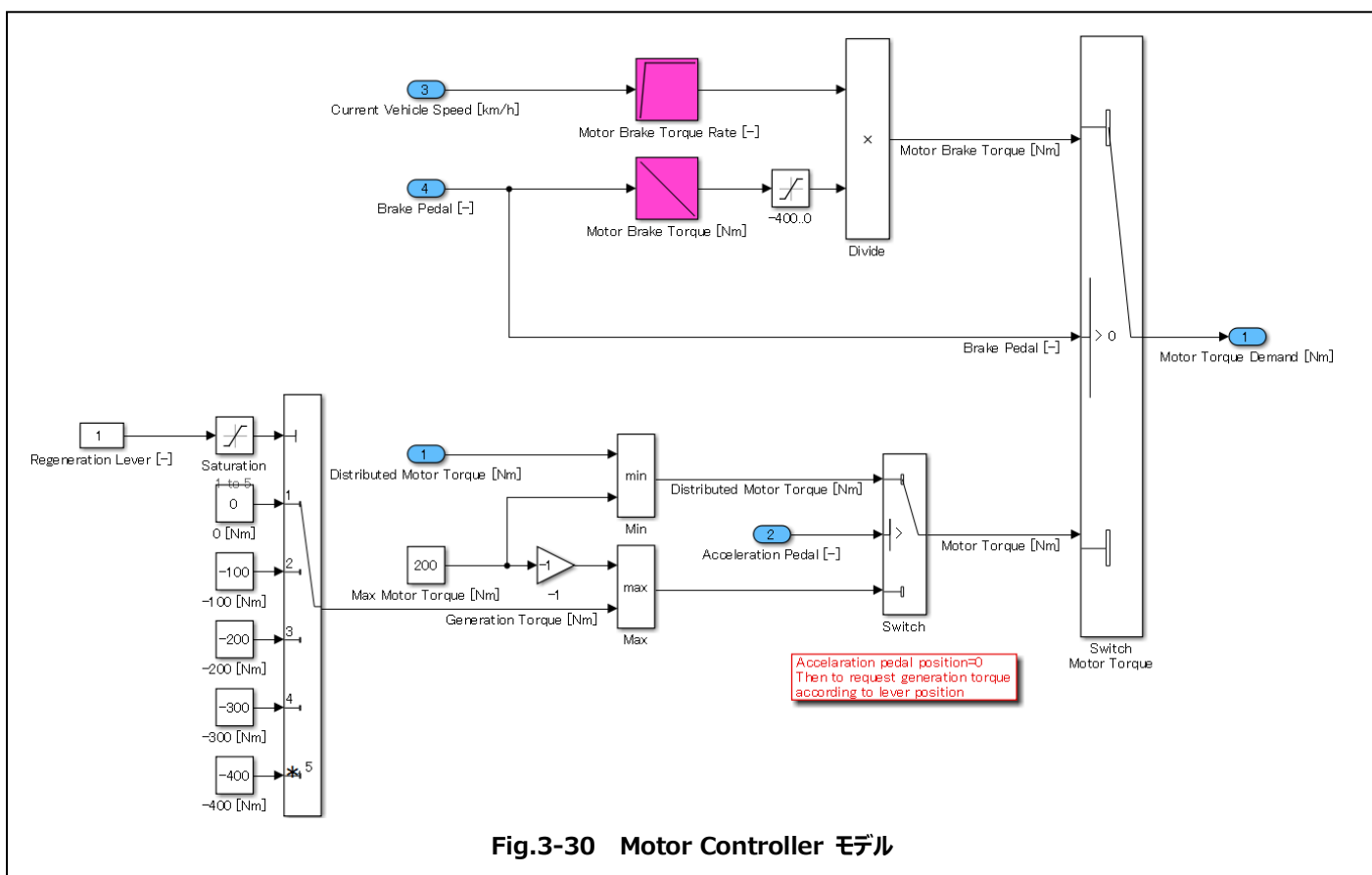
(3-3) Engine Controller

走行に必要なトルクとバッテリー充電に必要なトルクを合算し、あらかじめ設定した最大エンジントルクの範囲内でエンジン要求トルクを出力する。



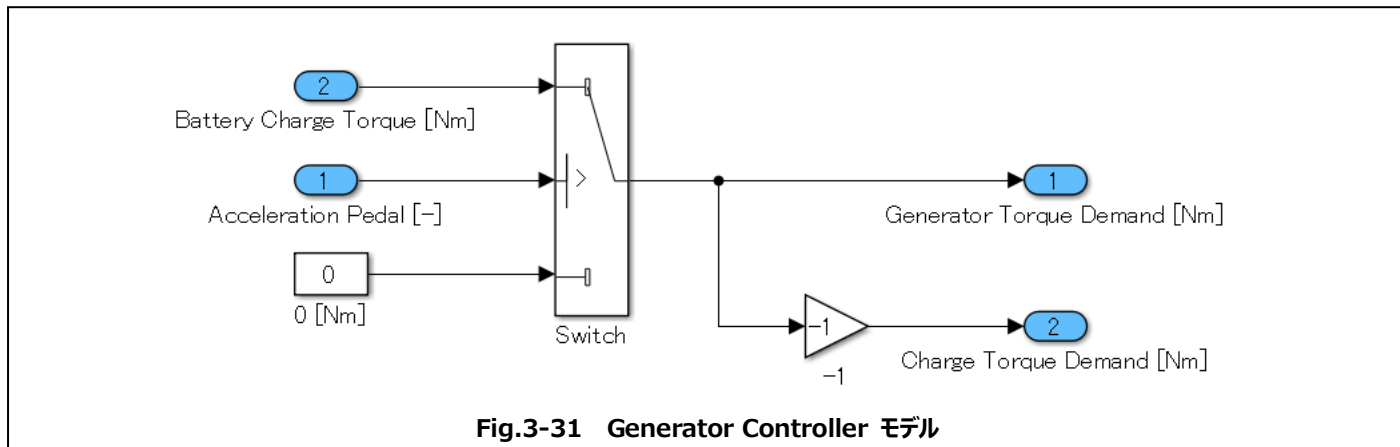
(3-4) Motor Controller

ブレーキペダルが踏まれていない場合かつアクセルペダルが踏まれている場合は、走行トルクとして配分されたトルクを要求する。ブレーキペダルが踏まれた場合は、ペダル開度と現在の車速からモータの減速トルクを算出し要求する。



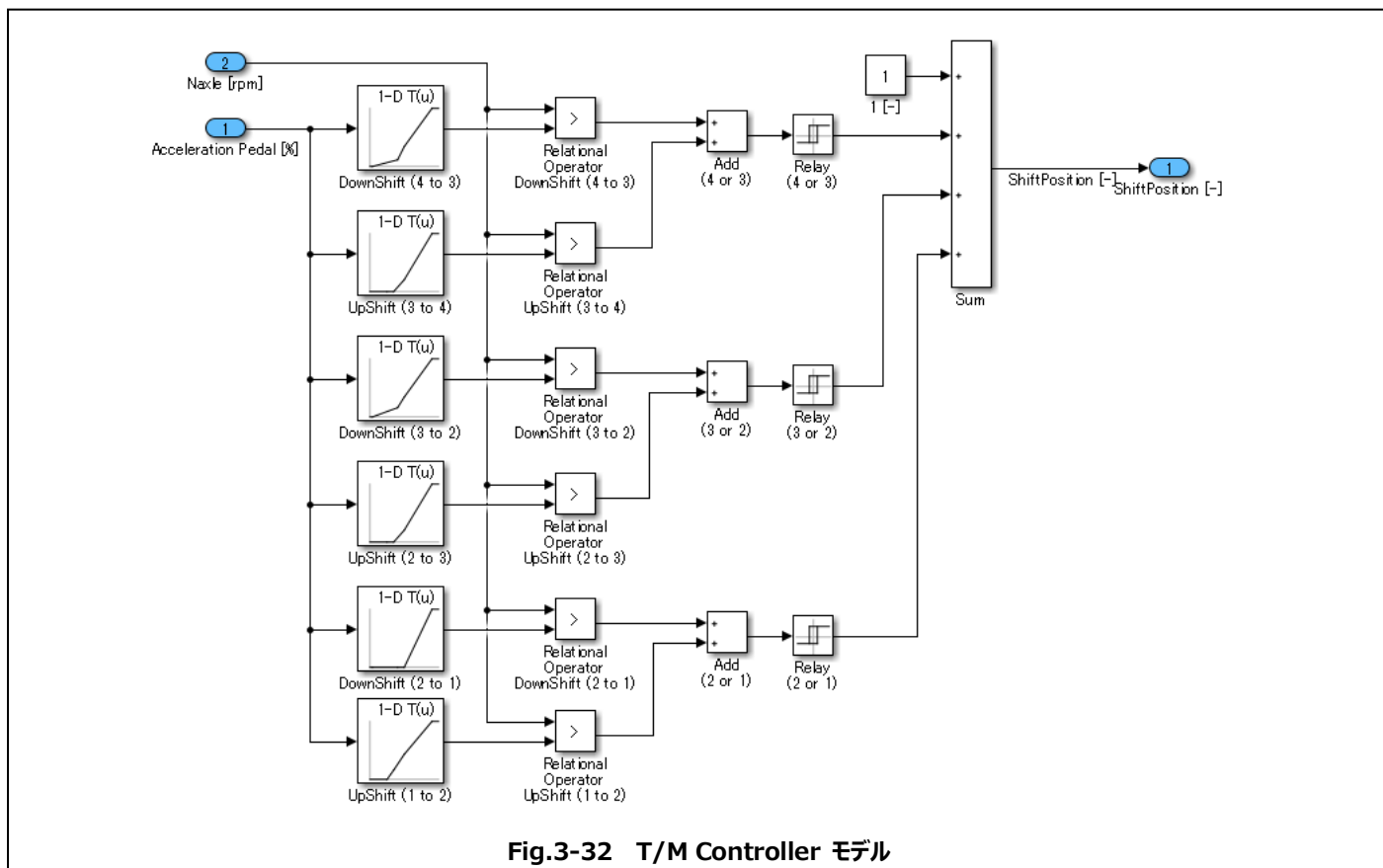
(3-5) Generator Controller

アクセルペダルが踏まれている場合のみ、発電トルクを要求する。



(3-6) T/M Controller

アクセルペダルとトランスミッション出口のアクスル回転数から、ギアポジションを選択し、要求ギヤを決定する。



3.4 評価

ここでは、パワートレーンモデルにおいてトルク、ギヤ段などが正常に動作し、代表的な台上試験モード（10 – 15モード）の目標車速に車両が追従しているかどうか、またバッテリー充電率をシミュレーションにより検証した。尚、車体のモデルは、次の章で説明される車両運動モデルと接続して車速をシミュレーションしたが、ドライバーモデルは、アクセルペダルとブレーキペダルの操作のみ実行し、ステアリング操作は行っていない。

(1) 車速追従性

目標車速に対して、シミュレーションされた車速は問題なく追従している。



Fig.3-33 シミュレーション結果 その1

(2) トルク制御とバッテリー充電率の推移

バッテリーSOCは、発電用モータが動作することにより上昇、かつ70%を超えると発電モータトルクは使用を制限する。

一方、減速時は走行用モータが回生しており、これによってもバッテリーSOCは上昇していることがわかる。エンジントルクは、車両走行分と発電モータ分の合算分が出力されており、プラントモデルは、当初の目論見どおり意図通り動作していることがわかる。

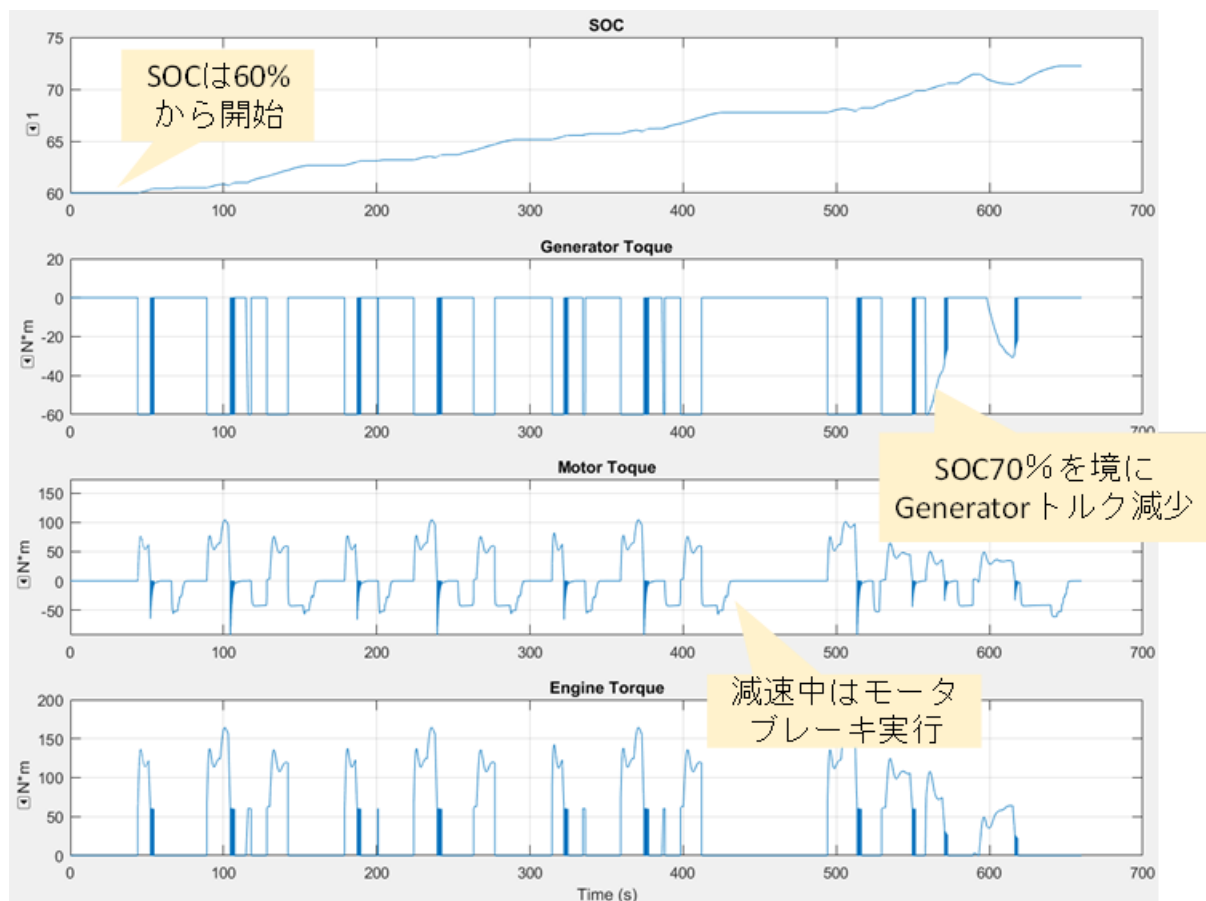


Fig.3-34 シミュレーション結果 その2

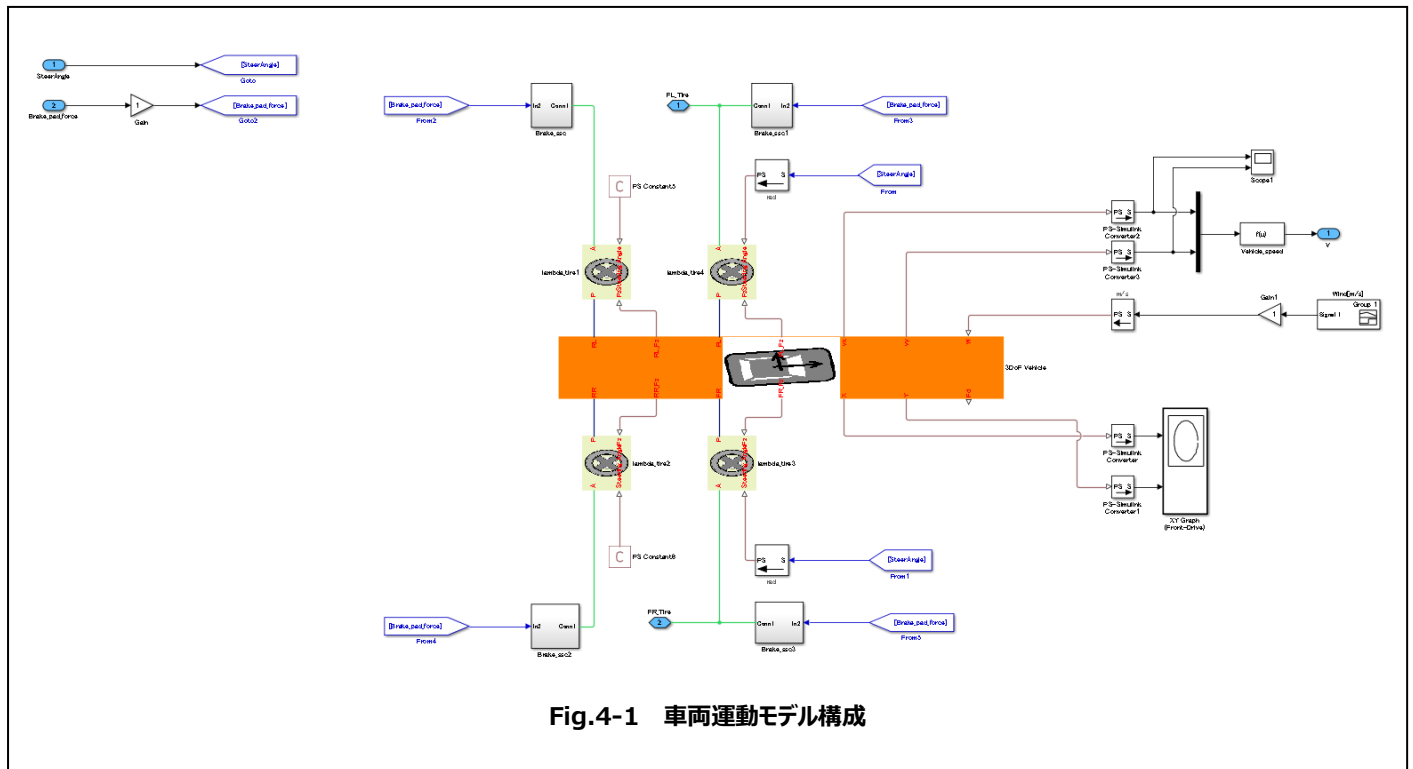
4 車両運動モデル

4.1 目的

本章では、車両平面運動モデルを題材に、Simscape カスタムドメインの作成、および、車両としての挙動を視覚的に確認するためのモデルの作成方法について紹介する。

4.2 構成

ここで取り扱う車両運動モデルは、タイヤ、車体、ステアリング、ブレーキ から構成される。



【タイヤ】

四輪独立に車軸周りに回転するものとし、前輪はディファレンシャルギアとつながっており、トルクと回転数の授受を行う。舵角は前輪の左右に等しい値をステアリングから入力する。その他は車体とやりとりする。

【車体】

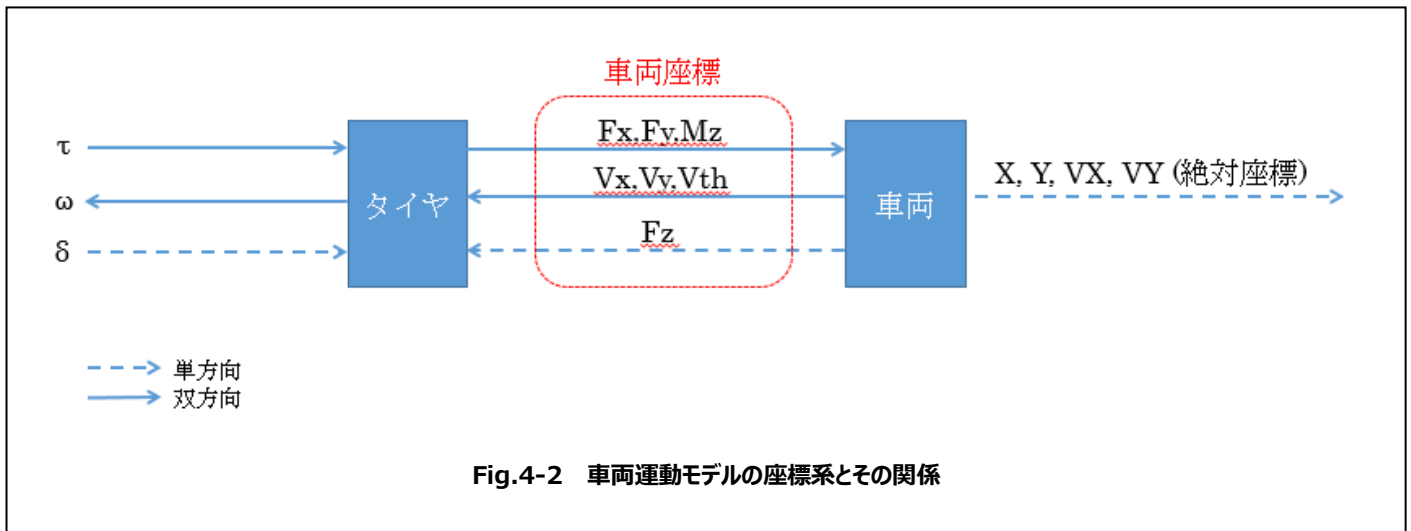
タイヤとのみ力・トルクと速度・角速度をやり取りする。

【ステアリング】

ステアリングの舵角に応じ、ギア比からタイヤ実舵角を決める。

【ブレーキ】

摩擦を模擬して回転数が 0 となるよう一定トルクを与え、0 となったあとは回転しないよう必要なトルクを与える。



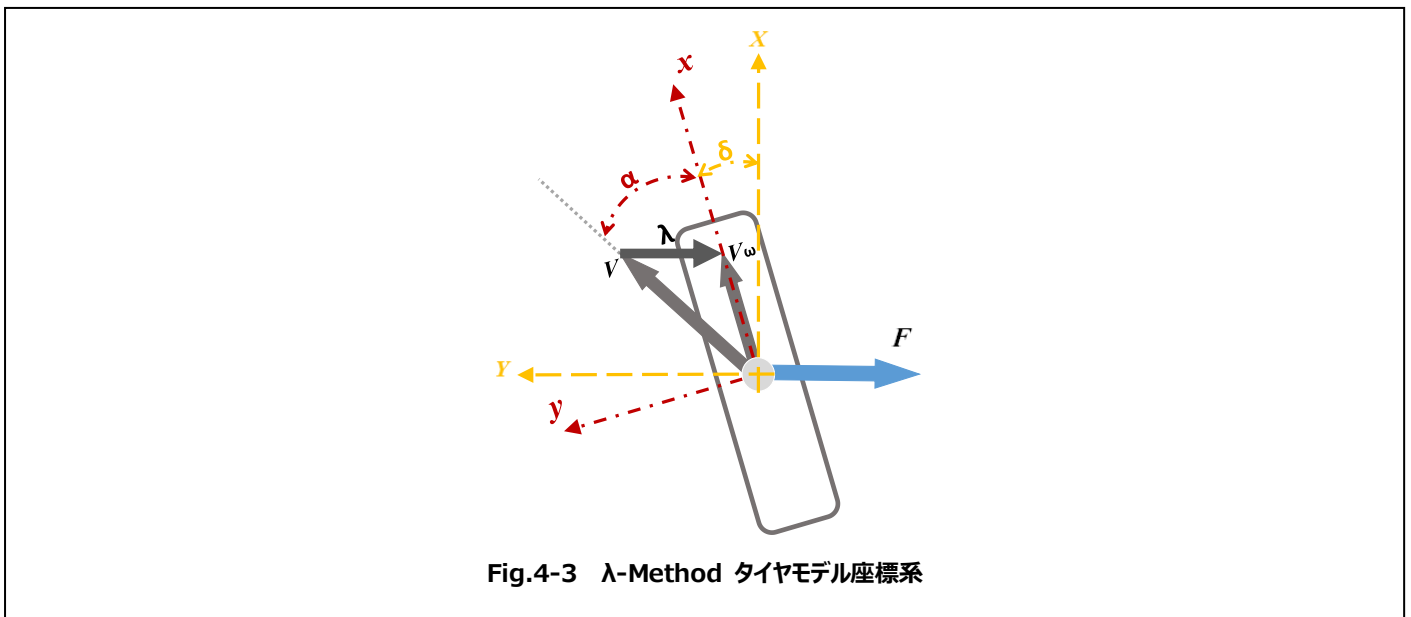
4.3 部品モデル

4.3.1 タイヤ

本書ではタイヤは車軸周りの回転と舵角に応じた回転を行い、それ以外の自由度はなく車体に固定されているものとする。タイヤはデフと車軸周りのトルクと角速度をやり取りする。またタイヤと車体は平面並進方向の力・接地点周りの力のモーメントをやりとりする。入力としてタイヤへの実舵角と接地反力を受け入れる。ここでは演算が簡便でありかつ実用性のある λ -Method タイヤを用いた。

(1) λ -Method タイヤ^[4-1]

スリップ率をベクトルとして扱い、Fig4-3 のように、ベクトル λ の方向にタイヤ力が発生すると考える手法となる。



スリップ率ベクトル $\vec{\lambda}$

タイヤ回転各速度を速度に換算したベクトル \vec{V}_ω 、車両速度ベクトル \vec{V} から、スリップ率ベクトル $\vec{\lambda}$ が求まる。

$$\vec{\lambda} = \frac{\vec{V}_\omega - \vec{V}}{\max(|\vec{V}|, |\vec{V}_\omega|)}$$

ここで、 \vec{V}_ω はタイヤ回転各速度を速度に換算したベクトル、 \vec{V} は車両速度ベクトルとなる。

タイヤの発生する力ベクトル \vec{f}

$$\vec{f} = \mu(|\lambda|)Mg \frac{\vec{\lambda}}{|\lambda|}$$

ここで、 $\mu(|\lambda|)$ はスリップ率から換算可能なタイヤと路面摩擦係数関数、 Mg は動的な垂直荷重となる。ここで、 $|\lambda|$ は、以下となる。

$$|\lambda| = \sqrt{\lambda_x^2 + \lambda_y^2}$$

また、 λ_x 、 λ_y は、上記スリップ率ベクトル $\vec{\lambda}$ より、以下と表せる。

$$\lambda_x = \frac{V_\omega - V \cos(\delta)}{\max(|V|, |V_\omega|)}$$

$$\lambda_y = \frac{-V \sin(\delta)}{\max(|V|, |V_\omega|)}$$

次に、路面摩擦係数関数 $\mu(|\lambda|)$ から、前後方向の摩擦係数 μ_x 、横方向の摩擦係数 μ_y は、 $\angle \vec{\lambda}$ を用いて次のよう表される。

$$\mu_x = \mu(|\lambda|) \sin(\angle \vec{\lambda})$$

$$\mu_y = \mu(|\lambda|) \cos(\angle \vec{\lambda})$$

尚 $\angle \vec{\lambda}$ は、以下と表される。

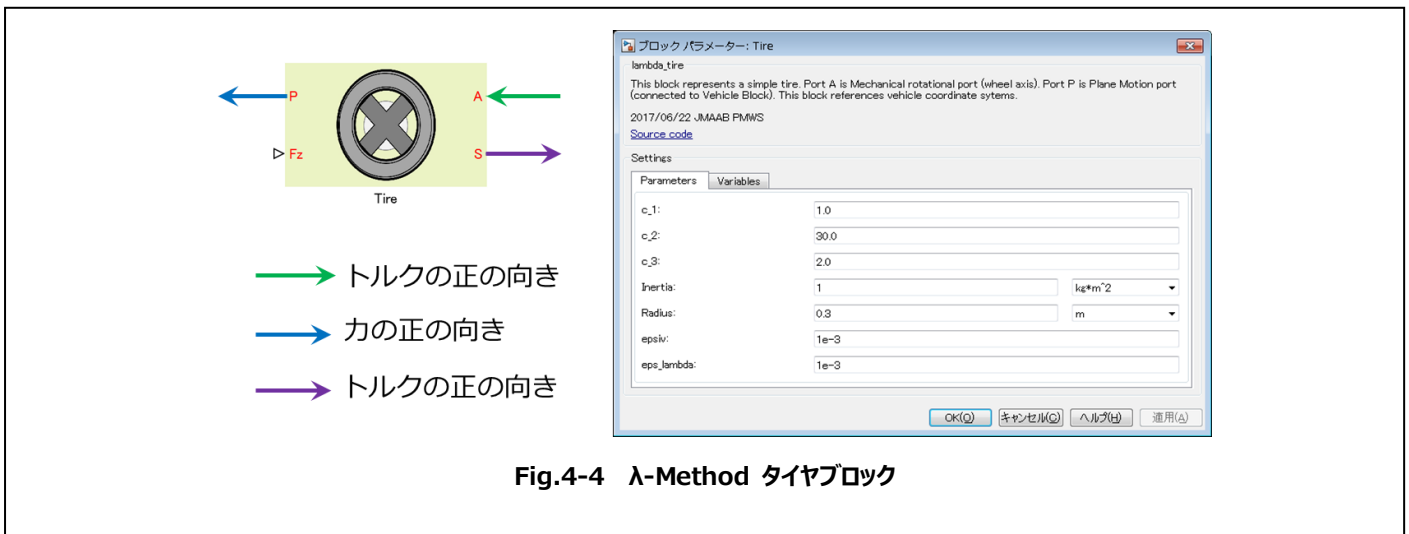
$$\angle \vec{\lambda} = \arctan\left(\frac{\lambda_x}{\lambda_y}\right)$$

上記の各式からタイヤ力を演算する。求めたタイヤ力は、タイヤ固定座標系 (x, y) を基準としているため、車両固定座標系 (X, Y) に変換する。

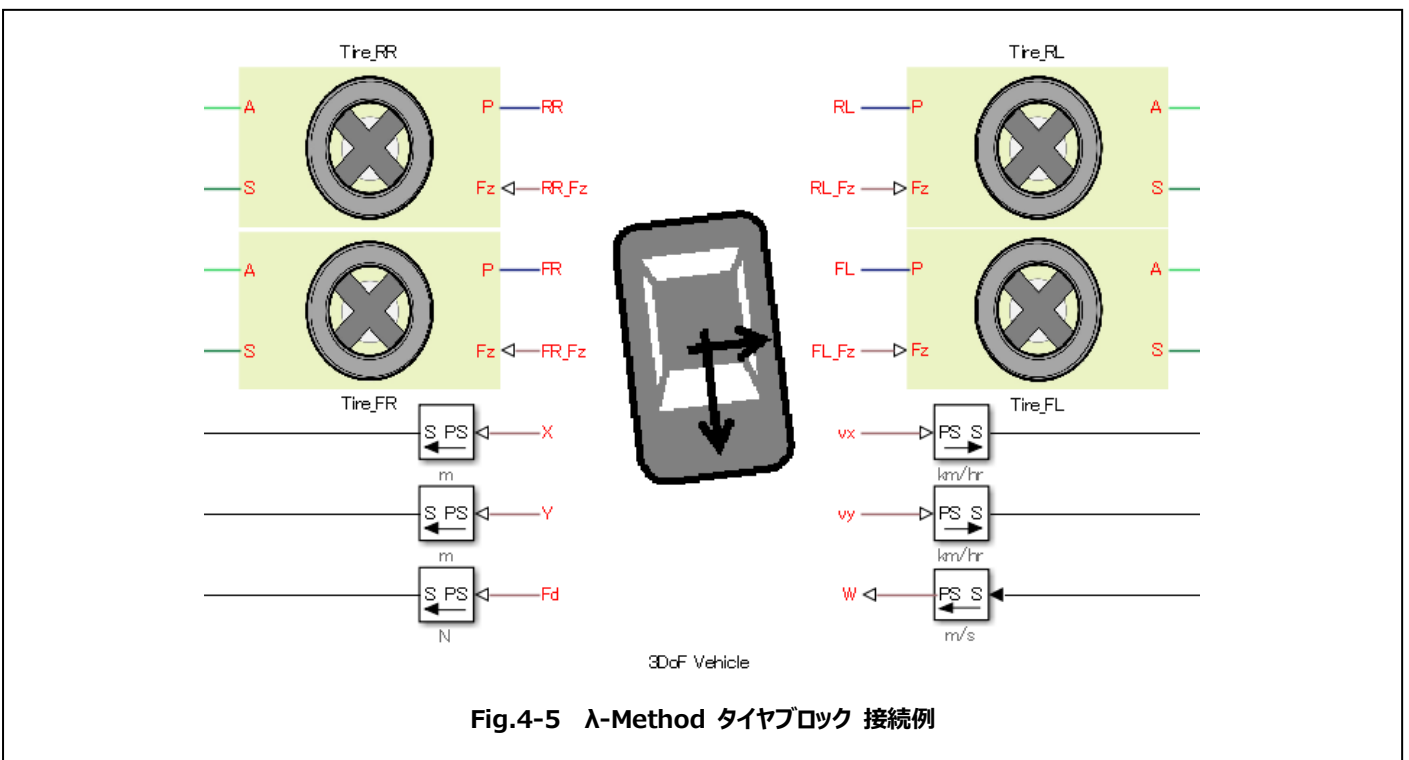
$$\begin{bmatrix} F_x \\ F_y \end{bmatrix} = \begin{bmatrix} \cos(\delta) & -\sin(\delta) \\ \sin(\delta) & \cos(\delta) \end{bmatrix} \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

尚、Fig4-3 は前輪におけるタイヤ回転方向を x 軸とした場合の、車両座標系の概念図である。 δ は、タイヤの実舵角となる。

(1) モデルブロック



(2) 接続例



(3) モデル詳細

名称	Lambda-Method Tire						
概要	トルク/力の向き：ディファレンシャルギアからのトルク入力が正、タイヤから車両に対する出力トルク・力方向が正						
	計算内容：						
	① タイヤ角速度、車両速度 及び タイヤ舵角から、タイヤ固定座標系に対する x,y 方向のスリップ率を計算						
	② 上記 ① の計算結果の絶対値、及び、そのなす角を計算						
	③ 上記 ② の演算結果より、タイヤ-路面間摩擦を計算						
	④ タイヤ固定座標系に対する、x, y 軸報告のタイヤが発生する力を計算						
⑤ 車両固定座標系に変換							
ノード		ポート名	物理ドメイン	Through /Across	内容	表示位置	
	1	A	機械(回転)	回転数[rad/s]、トルク [N*m]	駆動トルクのやり取り	left	
	2	P	カスタム	力[N]、速度[m/s]/回転数[rad/s]、	車両との物理量のやり取り	right	
	3	S	カスタム	トルク(Nm)、回転数 (rad/s)	ステアリングとの物理量のやり取り	left	
入力		ポート名	内容	範囲	初期値	単位	表示位置
	1	Fz	車両動的垂直荷重	-	0	N	right
パラメータ		パラメータ名	内容	範囲	初期設定値	単位	
	1	c1	タイヤ-路面摩擦決定係数	-	1	-	
	2	c2	タイヤ-路面摩擦決定係数	-	3.00E+01	-	
	3	c3	タイヤ-路面摩擦決定係数	-	2	-	
	4	Inertia	イナーシャ	-	1	Kg*m^2	
	5	Radius	タイヤ半径	-	0.3	m	
	6	epsiv	スリップ率計算用最小許容値 (ゼロ割防止用)	-	1.00E-06	m/s	
変数		変数名	内容	範囲	初期値	単位	

1	V_X	車両固定座標系に対する X 軸方向の車両速度	-	0	m/s	
2	V_Y	車両固定座標系に対する Y 軸方向の車両速度	-	0	m/s	
3	V	車両速度ベクトル	-	0	m/s	
4	V_omega	タイヤ角速度に対するタイヤみかけ速度	-	0	m/s	
5	maxV	ゼロ割防止用変数	-	0.001	m/s	
6	delta	実舵角	-	0	rad	
7	lambda	スリップ率ベクトル	-	0	-	
8	lambda_x	タイヤ固定座標系に対する x 軸方向のスリップ率	-	0	-	
9	abs_lambda_x	タイヤ固定座標系の x 軸方向のタイヤスリップ率の大きさ	-	0	-	
10	lambda_y	タイヤ固定座標系に対する y 軸方向のスリップ率	-	0	-	
11	abs_lambda_x	タイヤ固定座標系の	-	0	-	

			y 軸方向のタイヤスリップ率の大きさ				
12	lambda_angle		スリップ率のなす角	-	0	rad	
13	mu		タイヤ-路面間の摩擦係数の総和	-	0	-	
14	mu_x		タイヤ固定座標系の x 軸方向のタイヤ-路面間摩擦係数	-	0	-	
15	mu_y		タイヤ固定座標系の y 軸方向のタイヤ-路面間摩擦係数	-	0	-	
16	f_x		タイヤ固定座標系に対する x 軸方向のタイヤが発生する力	-	0	N	
17	f_y		タイヤ固定座標系に対する y 軸方向のタイヤが発生する力	-	0	N	
18	Fx		タイヤ発生力を車両固定座標 X 軸方向に座標変換した値	-	0	N	

	19	Fy	タイヤ発生力を車両固定座標 Y 軸方向に座標変換した値	-	0	N	
	20	Mz	セルフアライニングトルク	-	0	N*m	
	21	Tin	タイヤへのエンジン側からのトルク入力	-	0	N*m	

方程式、その他

タイヤ固定座標系に対するタイヤ発生力

$$f_x = \frac{M g \mu_x \lambda_x}{\max(|\lambda_x|, \epsilon)}$$

$$f_y = \frac{M g \mu_y \lambda_y}{\max(|\lambda_y|, \epsilon)}$$

車両固定座標系に対するタイヤ力

$$\begin{bmatrix} F_x \\ F_y \end{bmatrix} = \begin{bmatrix} \cos(\delta) & -\sin(\delta) \\ \sin(\delta) & \cos(\delta) \end{bmatrix} \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

(4) Simscape コード

```
component lambda_method_tire
% Lambda-Method Tire
% Ver. 1.0 Jun/22, 2017 JMAAB Plant Modeling Workshop

nodes
  A = foundation.mechanical.rotational.rotational; % A:left
  P = PMWS_Simscape_Library.Chassis.PlaneMotion; % P:right
  S = PMWS_Simscape_Library.Chassis.custom_rotation % S:left
end

inputs
  Fz = { 1.0, 'N' }; % Fz :right
end

variables

  V_X = { 0.0, 'm/s' };
  V_Y = { 0.0, 'm/s' };
  V_omega = { 0.0, 'm/s' };
  maxV = { 0.001, 'm/s' };

  delta = { 0, 'rad' };
  lambda = { 0.0, '1' };
  lambda_x = { 0.0, '1' };
  abs_lambda_x = { 0.0, '1' };
  lambda_y = { 0.0, '1' };
  abs_lambda_y = { 0.0, '1' };
  lambda_angle = { 0.0, 'rad' };

  mu = { 0.0, '1' };
  mu_x = { 0.0, '1' };
  mu_y = { 0.0, '1' };

  f_x = { 0.0, 'N' };
  f_y = { 0.0, 'N' };

  Fx = { 0.0, 'N' };
  Fy = { 0.0, 'N' };
  Mz = { 0.0, 'N*m' };

  Tin = { 0.0, 'N*m' };

end

parameters
  c_1 = { 1.0, '1' };
  c_2 = { 30.0, '1' };
  c_3 = { 2.0, '1' };

  I = { 1, 'kg*m^2' }; % Inertia
  R = { 0.3, 'm' }; % Radius

  epsiv = { 1e-6, '1' };
  eps_lambda = { 0.000001, '1' };

end
```

```

branches
    Tin : A.t -> *;
    Fx  : * -> P.Fx;
    Fy  : * -> P.Fy;
    Mz  : * -> S.Mz;
end

equations

%% V_X & V_Y :Vehicle fixed coordinate
V_X == P.Vx;
V_Y == P.Vy;
V == sqrt(V_X^2 + V_Y^2);

delta == S.delta;
V_omega == R * A.w;

maxV == max([abs(V), abs(V_omega), epsiv*{1,'m/s'} ]);
lambda_x == (V_omega - (V_X*cos(delta) + V_Y*sin(delta)))/maxV;
abs_lambda_x == max([abs(lambda_x), epsiv]);

lambda_y == -(V_X*sin(delta) - V_Y*cos(delta))/maxV;
abs_lambda_y == max([abs(lambda_y), epsiv]);

lambda == sqrt(lambda_x^2 + lambda_y^2);
lambda_angle == atan(abs_lambda_x / abs_lambda_y);
mu == -1.1 * c_1*(exp(-c_2* lambda ) - exp(-c_3* lambda ));

mu_x == mu * sin(lambda_angle);
mu_y == mu * cos(lambda_angle);

%% f_x & f_y :Tire fixed coordinate
%%
f_x == (Fz * lambda_x * mu_x) / (abs_lambda_x);
f_y == (Fz * lambda_y * mu_y) / (abs_lambda_y);

I * A.w.der == Tin - R*f_x;

%% F_X & F_Y :Vehicle fixed coordinate
%%
Fx == f_x*cos(delta) - f_y*sin(delta);
Fy == f_x*sin(delta) + f_y*cos(delta);
Mz == { 0.0, 'N*m' };

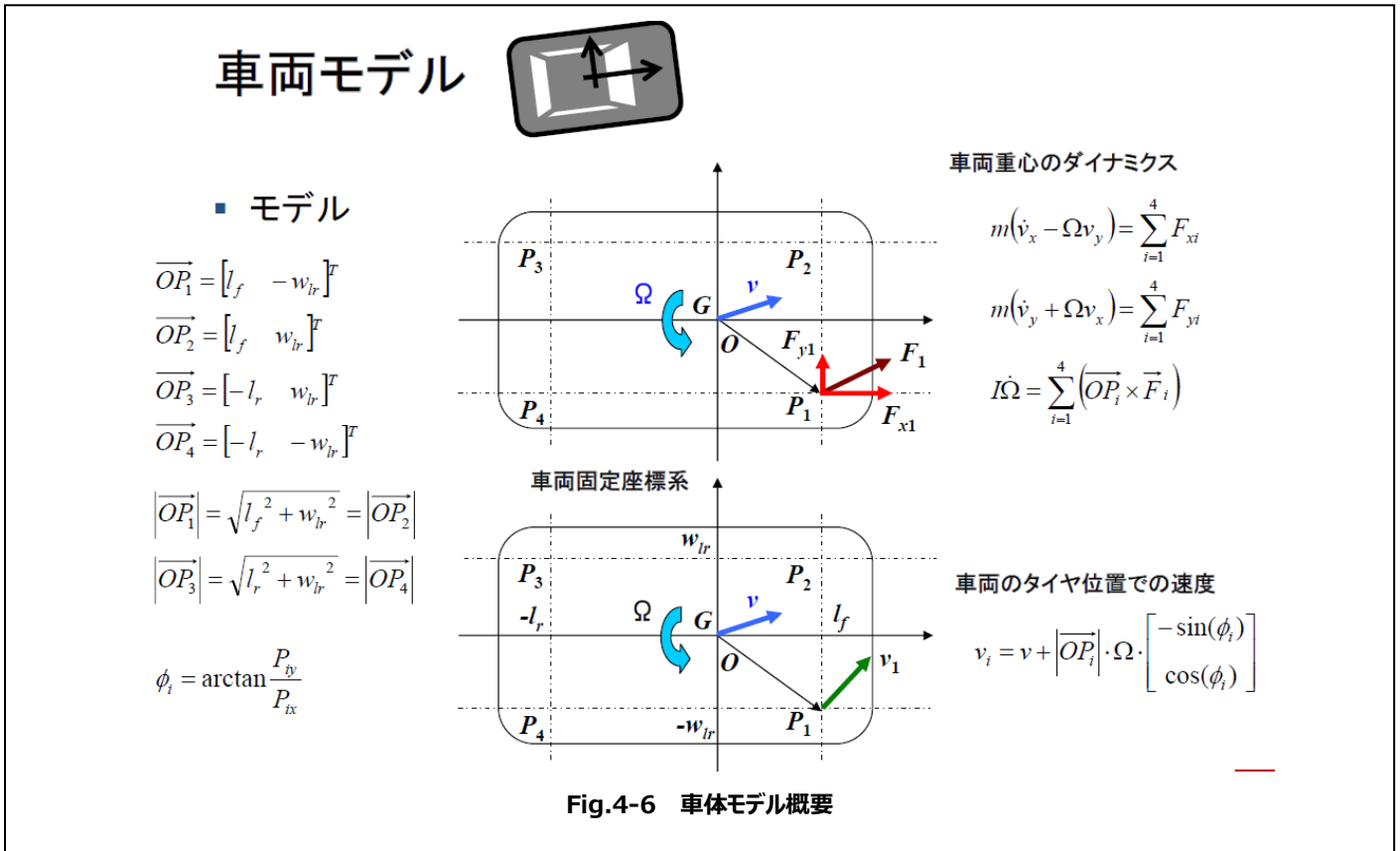
end
end

```

4.3.2 車体

車体についてはサスペンションのない最も簡単なモデルとする。そのため回転の自由度はヨー（Z 軸周りの回転）のみとし、ピッチ、ロールは無視する。並進は上下方向の運動は無視し、平面内の運動とする。

そのため運動方程式は並進 2、回転 1 の合計 3 つとなる。他の部品とは前後左右のタイヤとのみ結合しているものとする。以上の設定による**車体モデル（便宜上、車両モデルと呼ぶこともある）**について以下に示す。（2009 年 PMA-WS 資料より）。



P_i がタイヤの位置を示す。G は車両の重心で、車両座標系の原点と一致する。

x 方向、y 方向の運動方程式はそれぞれ以下の通り。

$$m \left(\frac{d}{dt} v_x - v_y \omega \right) = F_{x1} + F_{x2} + F_{x3} + F_{x4} + F_d,$$

$$m \left(\frac{d}{dt} v_y - v_x \omega \right) = F_{y1} + F_{y2} + F_{y3} + F_{y4} + D_2 v_y,$$

$$F_d = \frac{1}{2} C_d \rho A_f (v_x - W)^2 \times \text{sign}(v_x - W).$$

ここで m は車重、 D_2 は y 方向の抵抗係数、 F_d は空気抵抗、 C_d は空気の抵抗係数、 ρ は空気の密度、 A_f は車両前面の面積、 W は x 方向の風速。

回転系の運動方程式

$$I \frac{d}{dt} \omega = M_z,$$

$$M_z = P_{1x} F_{y1} - P_{1y} F_{x1} + P_{2x} F_{y2} - P_{2y} F_{x2} + P_{3x} F_{y3} - P_{3y} F_{x3} + P_{4x} F_{y4} - P_{4y} F_{x4} - D_3 \omega.$$

車両座標系の速度を絶対座標系に変換

$$v_{x \text{ wcs}} = v_x \cos \theta_{yaw} - v_y \sin \theta_{yaw},$$

$$v_{y \text{ wcs}} = v_y \sin \theta_{yaw} + v_x \cos \theta_{yaw},$$

$$\frac{d}{dt} \theta_{yaw} = \omega.$$

ここで、 $v_{x \text{ wcs}}$ 、 $v_{y \text{ wcs}}$ はそれぞれ絶対座標系での x 、 y 方向の速度、 θ_{yaw} はヨー角。

絶対座標での車両位置 x_{wcs} 、 y_{wcs} の計算

$$\frac{d}{dt} x_{wcs} = v_{x \text{ wcs}},$$

$$\frac{d}{dt} y_{wcs} = v_{y \text{ wcs}}.$$

(1) モデルブロック

The figure shows a 3DoF Vehicle model block and its parameter settings window. The model block is a grey rectangle with a car-like shape, with a coordinate system (X, Y, Z) and velocity vectors (vx, vy, W) shown. The parameter settings window is titled 'ブロックパラメーター: 3DoF Vehicle' and contains the following parameters:

Parameter	Value	Unit
Mass:	1500	kg
ϵ :	9.81	m/s ²
Inertia:	2e3	kg*m ²
From G to Front:	1.8	m
From G to Rear:	2.4	m
From G to Right/Left:	1	m
Frontal area:	3	m ²
Drag coefficient:	0.4	
Atmospheric air density:	1.2	kg/m ³
D_1:	0.01	kg/s
D_2:	0.01	kg/s
D_3:	0.01	m*s*N

Fig.4-7 車体モデルブロック

(2) 接続例

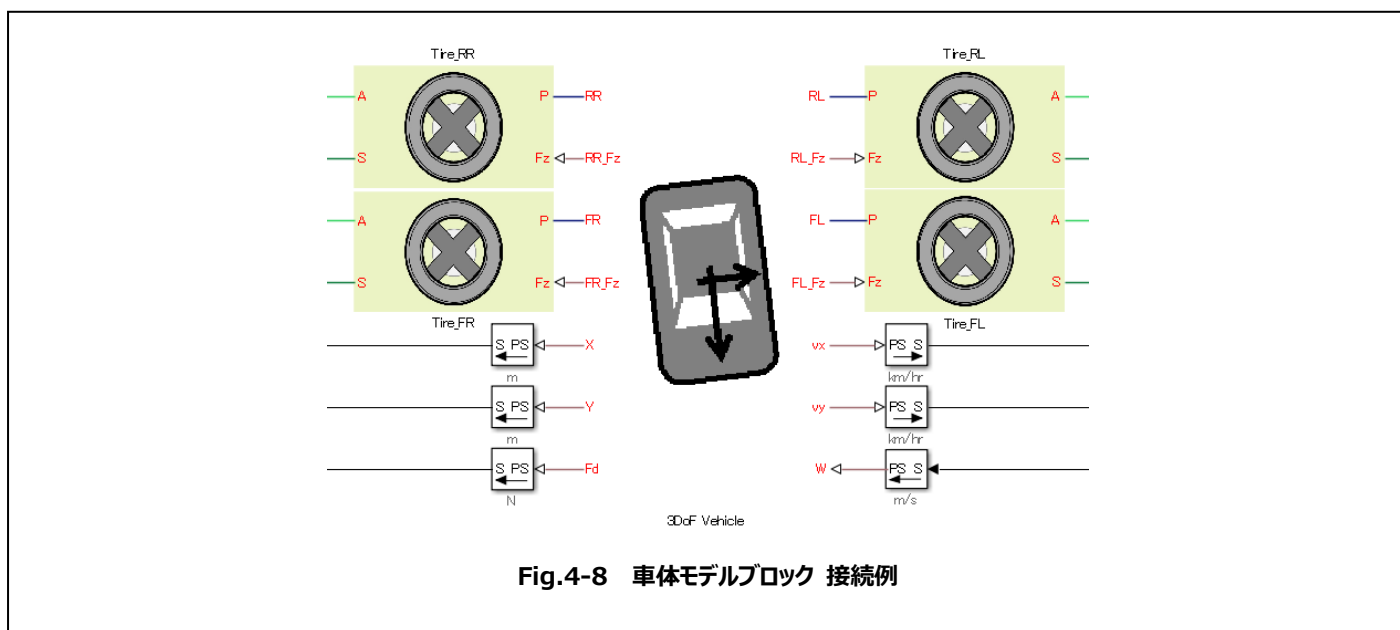


Fig.4-8 車体モデルブロック 接続例

(3) モデル詳細

名称	3DoF Vehicle					
概要	3自由度の簡易車両モデル					
	トルク、力の向き：すべてのポートで外から内が正					
	計算内容：					
	① 入力された風速と車速から空気抵抗を計算					
	② 各タイヤから入力される力と空気抵抗から車両の加速度を計算					
	③ 各タイヤから入力される力と重心位置からz軸周り(ヨー方向)の角速度を計算					
ノード	④ 車速とz軸周りの角速度から各タイヤの速度を計算					
	⑤ 車速(車両座標系)やヨー角を使って絶対座標系の車速と位置を計算					
		ポート名	物理ドメイン	Through/Across	内容	表示位置
	1	RR	2自由度並進 カスタム	X軸並進力[N]、Y軸並進力[N] X軸速度[m/s]、Y軸速度[m/s]	タイヤとの物理量のやり取り	bottom
	2	FR	2自由度並進 カスタム	X軸並進力[N]、Y軸並進力[N] X軸速度[m/s]、Y軸速度[m/s]	タイヤとの物理量のやり取り	bottom
	3	RL	2自由度並進 カスタム	X軸並進力[N]、Y軸並進力[N] X軸速度[m/s]、Y軸速度[m/s]	タイヤとの物理量のやり取り	top

	4	FL	2自由度並進 カスタム	X軸並進力[N]、Y軸並進力[N] X軸速度[m/s]、Y軸速度[m/s]	タイヤとの物理量のやり取り	top	
入力		ポート名	内容	範囲	初期値	単位	表示位置
	1	W	風速	-	0	m/s	top
出力		ポート名	内容	範囲	初期値	単位	表示位置
	1	RR_Fz	右リアタイヤの垂直抗力	-	0	N	bottom
	2	FR_Fz	右リアタイヤの垂直抗力	-	0	N	bottom
	3	RL_Fz	右リアタイヤの垂直抗力	-	0	N	top
	4	FL_Fz	右リアタイヤの垂直抗力	-	0	N	top
	5	X	絶対座標系での車両のX座標	-	0	m	bottom
	6	Y	絶対座標系での車両のY座標	-	0	m	bottom
	7	Fd	空気抵抗	-	0	N	bottom
	8	vx	車両座標系での車速のX成分	-	0	m/s	top
	9	vy	車両座標系での車速のY成分	-	0	m/s	top
パラメータ		パラメータ名	内容	範囲	初期設定値	単位	
	1	m	車重	-	1500	kg/s	
	2	g	重力加速度	-	9.81	m/s ²	
	3	I	Z軸周りの慣性モーメント	-	2.00E+03	kg*m ²	
	4	lf	重心からフロント車軸までの距離	-	1.8	m	

	5	lr	重心からリア車軸までの距離	-	2.4	m	
	6	wrl	重心から左右の車両端までの距離	-	1	m	
	7	Af	車両前面の面積	-	3	m ²	
	8	Cd	空気の抵抗係数	-	0.4	-	
	9	rho	空気の密度	-	1.2	kg/m ³	
	10	D_2	Y方向の車速に比例した抵抗の係数	-	0.01	kg/s	
	12	D_3	Z軸周りの回転数に比例した抵抗の係数	-	0.01	N*m*s	
変数		変数名	内容	範囲	初期値	単位	
	1	Fx1	右フロントタイヤから受けるX方向の力	-	0	N	
	2	Fx2	左フロントタイヤから受けるX方向の力	-	0	N	
	3	Fx3	リアタイヤから受けるX方向の力	-	0	N	
	4	Fx4	右リアタイヤから受けるX方向の力	-	0	N	
	5	Fy1	右フロントタイヤから受けるY方向の力	-	0	N	
	6	Fy2	左フロントタイヤから受けるY方向の力	-	0	N	

7	Fy3	左リアタイヤから受けるY方向の力	-	0	N	
8	Fy4	右リアタイヤから受けるY方向の力	-	0	N	
9	Mz	ヨー方向に回転させようとするトルク	-	0	N*m	
10	Omega	ヨー方向の角速度	-	0	rad/s	
11	Px1	車両座標系での右フロントタイヤのX座標	-	1	m	
12	Px2	車両座標系での左フロントタイヤのX座標	-	1	m	
13	Px3	車両座標系での左リアタイヤのX座標	-	1	m	
14	Px4	車両座標系での右リアタイヤのX座標	-	1	m	
15	Py1	車両座標系での右フロントタイヤのY座標	-	1	m	
16	Py2	車両座標系での左フロントタイヤのY座標	-	1	m	
17	Py3	車両座標系での左リアタイヤのY座標	-	1	m	
18	Py4	車両座標系での右リアタイヤのY座標	-	1	m	

19	OP1	右フロントタイヤから重心までの距離	-	1	m	
20	OP2	左フロントタイヤから重心までの距離	-	1	m	
21	OP3	左リアタイヤから重心までの距離	-	1	m	
22	OP4	右リアタイヤから重心までの距離	-	1	m	
23	phi1	X 軸から右フロントタイヤの角度	-	0	rad	
24	phi2	X 軸から左フロントタイヤの角度	-	0	rad	
25	phi3	X 軸から左リアタイヤから重心の角度	-	0	rad	
26	phi4	X 軸から右リアタイヤから重心の角度	-	0	rad	
27	vx_wcs	絶対座標系での車速の X 成分	-	0	m/s	
28	vy_wcs	絶対座標系での車速の Y 成分	-	0	m/s	
29	yaw	ヨー角	-	0	rad	

方程式、その他

車両座標系でのタイヤの X 座標

$$Px1 = lf$$

$$Px2 = lf$$

$$Px3 = -lr$$

$$Px4 = -lr$$

車両座標系でのタイヤの Y 座標

$$Py1 = -wrl$$

$$Py2 = wrl$$

$$Py3 = wrl$$

$$Py4 = -wrl$$

車両重心からタイヤまでの距離

$$OP1 = \text{sqrt}(Px1^2 + Py1^2)$$

$$OP2 = \text{sqrt}(Px2^2 + Py2^2)$$

$$OP3 = \text{sqrt}(Px3^2 + Py3^2)$$

$$OP4 = \text{sqrt}(Px4^2 + Py4^2)$$

車両座標系での各タイヤの位置を示す角度

$$phi1 = \text{atan}(Py1/Px1)$$

$$phi2 = \text{atan}(Py2/Px2)$$

$$phi3 = \text{atan}(Py3/Px3)$$

$$phi4 = \text{atan}(Py4/Px4)$$

運動方程式

$$m * (dvx(t)/dt - \Omega(t) * vy(t)) = Fx1(t) + Fx2(t) + Fx3(t) + Fx4(t) + Fd(t)$$

$$Fd(t) = (1/2) * Cd * \rho * Af * (vx(t) - W(t))^2 * \text{sign}(vx(t) - W(t))$$

$$m * (dvy(t)/dt + \Omega(t) * vx(t)) = Fy1(t) + Fy2(t) + Fy3(t) + Fy4(t) + D_2 * vy(t)$$

$$I * d\Omega(t)/dt = Mz(t)$$

$$Mz(t) = Px1 * Fy1(t) - Py1 * Fx1(t) + Px2 * Fy2(t) - Py2 * Fx2(t) + Px3 * Fy3(t) - Py3 * Fx3(t) + Px4 * Fy4(t) - Py4 * Fx4(t) + D_3 * \Omega(t)$$

車速とヨー角の角速度から各タイヤの速度を算出

$$T1.Vx(t) = vx(t) - OP1 * \Omega(t) * \sin(phi1)$$

$$T1.Vy(t) = vy(t) + OP1 * \Omega(t) * \cos(phi1)$$

$$T2.Vx(t) = vx(t) - OP2 * \Omega(t) * \sin(phi2)$$

$$T2.Vy(t) = vy(t) + OP2 * \Omega(t) * \cos(phi2)$$

$$T3.Vx(t) = vx(t) - OP3 * \Omega(t) * \sin(phi3)$$

$$T3.Vy(t) = vy(t) + OP3 * \Omega(t) * \cos(phi3)$$

$$T4.Vx(t) = vx(t) - OP4 * \Omega(t) * \sin(phi4)$$

$$T4.Vy(t) = vy(t) + OP4 * \Omega(t) * \cos(phi4)$$

垂直荷重(ピッチやロールを考慮していないため、各タイヤに車重が 4 等分されるとしている)

$$T1_Fz = m * g / 4.0$$

$$T2_Fz = m * g / 4.0$$

$$T3_Fz = m * g / 4.0$$

$$T4_Fz = m * g / 4.0$$

車両座標系の速度を絶対座標系に変換

$$\begin{aligned}v_{x_wcs}(t) &= v_x(t) * \cos(\text{yaw}(t)) - v_y(t) * \sin(\text{yaw}(t)) \\v_{y_wcs}(t) &= v_x(t) * \sin(\text{yaw}(t)) + v_y(t) * \cos(\text{yaw}(t)) \\d\text{yaw}(t)/dt &= \text{Omega}(t)\end{aligned}$$

絶対座標系での車両位置を計算

$$\begin{aligned}dx_wcs(t)/dt &= v_{x_wcs}(t) \\dy_wcs(t)/dt &= v_{y_wcs}(t)\end{aligned}$$

(4) Simscape コード

```
component Vehicle_3DoF
% 3DoF Vehicle
% Ver. 1.0 Jun/22, 2017 JMAAB Plant Modeling Workshop

nodes
    T4 = PMWS_Simscape_Library.Chassis.PlaneMotion;    % RR:bottom
end
outputs
    T4_Fz = { 1.0 , 'N' };                            % RR_Fz:bottom
end
nodes
    T3 = PMWS_Simscape_Library.Chassis.PlaneMotion;    % RL:top
end
outputs
    T3_Fz = { 1.0 , 'N' };                            % RL_Fz:top
end
nodes
    T2 = PMWS_Simscape_Library.Chassis.PlaneMotion;    % FL:top
end
outputs
    T2_Fz = { 1.0 , 'N' };                            % FL_Fz:top
end
nodes
    T1 = PMWS_Simscape_Library.Chassis.PlaneMotion;    % FR:bottom
end
outputs
    T1_Fz = { 1.0 , 'N' };                            % FR_Fz:bottom
end
```



```

parameters
    m  = { 1500, 'kg' }; % Mass
    g  = { 9.81, 'm/s^2' };
    I  = { 2e3, 'kg*m^2' }; % Inertia
    lf = { 1.8, 'm' }; % From G to Front
    lr = { 2.4, 'm' }; % From G to Rear
    wrl = { 1, 'm' }; % From G to Right/Left
    Af = { 3, 'm^2' }; % Frontal area
    Cd = { 0.4, '1' }; % Drag coefficient
    rho = { 1.2, 'kg/m^3' }; % Atmospheric air density

    D_1 = {0.01, 'kg/s'};
    D_2 = {0.01, 'kg/s'};
    D_3 = {0.01, 'N*m*s'};
end

outputs
    vx = { 0.0, 'm/s' }; % vx:top
    vy = { 0.0, 'm/s' }; % vy:top
    x_wcs = { 0.0, 'm' }; % X:bottom
    y_wcs = { 0.0, 'm' }; % Y:bottom
end

inputs
    W = { 0, 'm/s' }; % W : top
end

outputs %%Drag force
    Fd = { 0, 'N' }; % Fd:bottom
end

branches
    Fx1 : T1.Fx -> *;
    Fy1 : T1.Fy -> *;
    Fx2 : T2.Fx -> *;
    Fy2 : T2.Fy -> *;
    Fx3 : T3.Fx -> *;
    Fy3 : T3.Fy -> *;
    Fx4 : T4.Fx -> *;
    Fy4 : T4.Fy -> *;
end

```

equations

$$Px1 == lf;$$

$$Px2 == lf;$$

$$Px3 == -lr;$$

$$Px4 == -lr;$$

$$Py1 == -wrl;$$

$$Py2 == wrl;$$

$$Py3 == wrl;$$

$$Py4 == -wrl;$$

$$OP1 == \text{sqrt}(Px1^2+Py1^2);$$

$$OP2 == \text{sqrt}(Px2^2+Py2^2);$$

$$OP3 == \text{sqrt}(Px3^2+Py3^2);$$

$$OP4 == \text{sqrt}(Px4^2+Py4^2);$$

$$\text{phi1} == \text{atan}(Py1/Px1);$$

$$\text{phi2} == \text{atan}(Py2/Px2);$$

$$\text{phi3} == \text{atan}(Py3/Px3)+\text{pi};$$

$$\text{phi4} == \text{atan}(Py4/Px4)+\text{pi};$$

$$Fd == (1/2)*Cd*\rho*Af*(vx-W)^2*\text{sign}(vx-W);$$

$$m*(vx.\text{der}-\Omega*vy) == Fx1+Fx2+Fx3+Fx4 \dots \\ - Fd ;$$

$$m*(vy.\text{der}+\Omega*vx) == Fy1+Fy2+Fy3+Fy4 - D_2 * vy;$$

$$I*\Omega.\text{der} == Mz;$$

$$Mz == Px1*Fy1-Py1*Fx1 + Px2*Fy2-Py2*Fx2 \dots \\ + Px3*Fy3-Py3*Fx3 + Px4*Fy4-Py4*Fx4 - D_3 * \Omega \quad ;$$

$$T1.Vx == vx - OP1*\Omega*\sin(\text{phi1});$$

$$T1.Vy == vy + OP1*\Omega*\cos(\text{phi1});$$

$$T1_Fz == m*g/4.0;$$

$$T2.Vx == vx - OP2*\Omega*\sin(\text{phi2});$$

$$T2.Vy == vy + OP2*\Omega*\cos(\text{phi2});$$

$$T2_Fz == m*g/4.0;$$

```
T3.Vx == vx - OP3*Omega*sin(phi3);
T3.Vy == vy + OP3*Omega*cos(phi3);
T3_Fz == m*g/4.0;
```

```
T4.Vx == vx - OP4*Omega*sin(phi4);
T4.Vy == vy + OP4*Omega*cos(phi4);
T4_Fz == m*g/4.0;
```

```
%速度を絶対座標系の速度に変換。
```

```
yaw.der == Omega;
vx_wcs == vx * cos(yaw) - vy * sin(yaw);
vy_wcs == vx * sin(yaw) + vy * cos(yaw);
```

```
%位置の微分が速度
```

```
x_wcs.der == vx_wcs;
y_wcs.der == vy_wcs;
```

```
end
```

```
end
```

4.3.4 ステアリング

ステアリングモデルは、シンプルなモデルとして表現した。

タイヤ実舵角は、次式に示すように、ハンドル舵角に対しトータルのギア比を乗算することで一意に求まるようなモデルとなっている。したがって、ステアリング-タイヤ間のダイナミクスは考慮していない。

$$\delta_{Tire\ Angle} = G_{Total\ Gear\ Ratio} \cdot \theta_{Steering\ Angle}$$

(1) モデルブロック

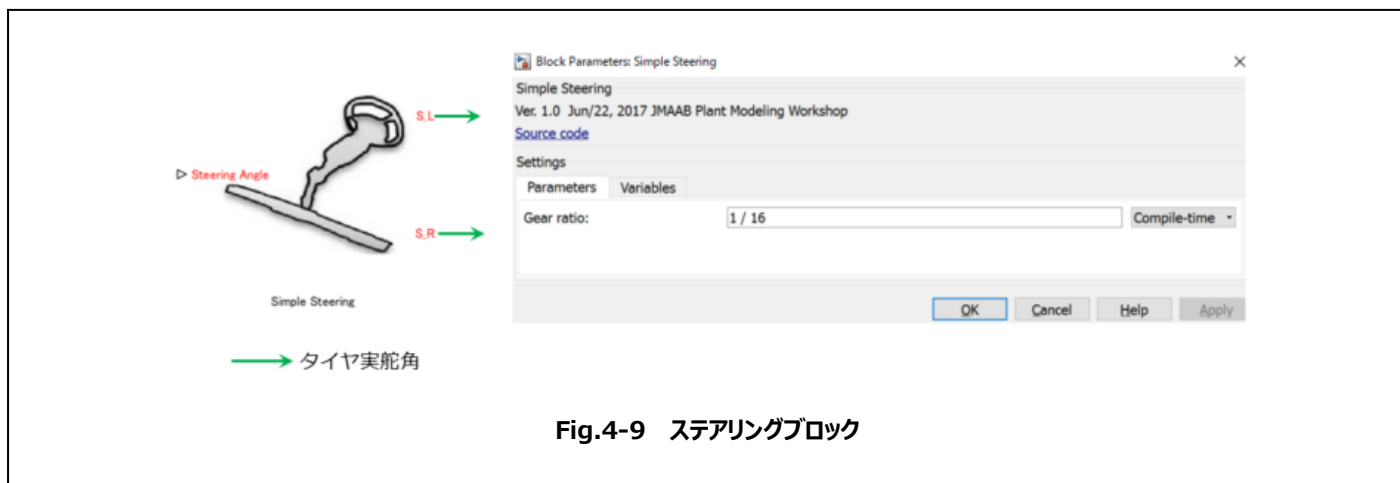


Fig.4-9 ステアリングブロック

(2) 接続例

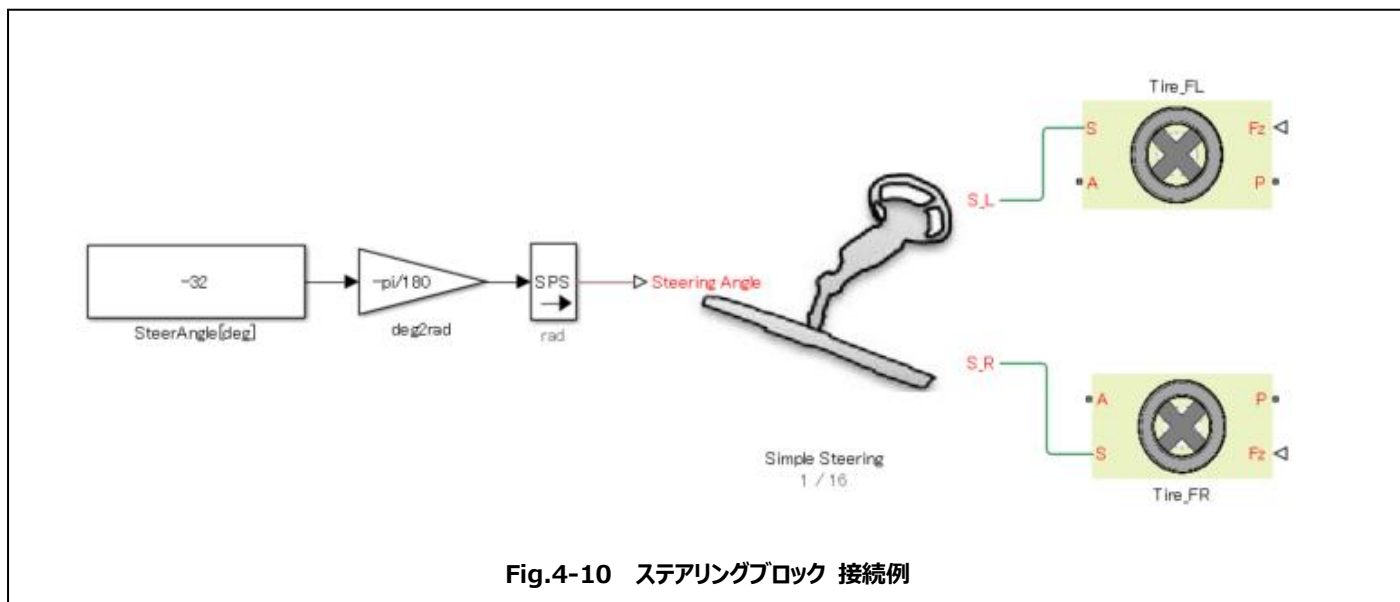


Fig.4-10 ステアリングブロック 接続例

(3) モデル詳細

名称	Steering						
概要	シンプルなステアリング ステアリングの舵角に応じ、ギア比からタイヤ実舵角を求める。 (ダイナミクスは考慮していない)						
ノード		ポート名	物理ドメイン	Through / Across	内容	表示位置	
	1	S_L	機械回転	トルク(Nm), 回転(rad)	回転角度	right	
	2	S_R	機械回転		回転角度	right	
入力		ポート名	内容	範囲	初期値	単位	表示位置
	1	Steering Angle	ステアリング 舵角	—	0	rad	left
パラメータ		パラメータ名	内容	範囲	初期設定値	単位	
	1	Gear Ratio	ステアリング 機構内のトータルギア比	—	1/16	—	
変数		変数名	内容	範囲	初期値	単位	
	1	Mz_L	—	—	0	N*m	
	2	Mz_R	—	—	0	N*m	
方程式、その他							
ステアリングの舵角とタイヤの舵角関係式 $TireAngle = TotalGearRatio \cdot SteeringAngle$							

(4) Simscape コード

```
component Steering
% Simple Steering
% Ver. 1.0 Jun/22, 2017 JMAAB Plant Modeling Workshop

inputs
    delta_H = { 0, 'rad' }; % Steering Angle : left
end

nodes
    S_L = PMWS_Simscape_Library.Chassis.custom_rotation % S_L:right
    S_R = PMWS_Simscape_Library.Chassis.custom_rotation % S_R:right
end

parameters
    ratio = { 1/16, '1' }; % Gear ratio
end

variables
    Mz_L = { 0.0, 'N*m' };
    Mz_R = { 0.0, 'N*m' };
end

branches
    Mz_L : S_L.Mz -> *;
    Mz_R : S_R.Mz -> *;
end

equations
    S_L.delta == ratio * delta_H;
    S_R.delta == ratio * delta_H;
end
end
```

4.3.5 ブレーキ

本節では、車両モデル内で使用するブレーキモデルについて説明する。

まず、ブレーキ状態の判定条件は下図とする。入力 ω と F_B に対し、閾値 ω_{thr} と F_{B_thr} によって、Locked state, Positive unlock state, Negative unlock state に分かれる。

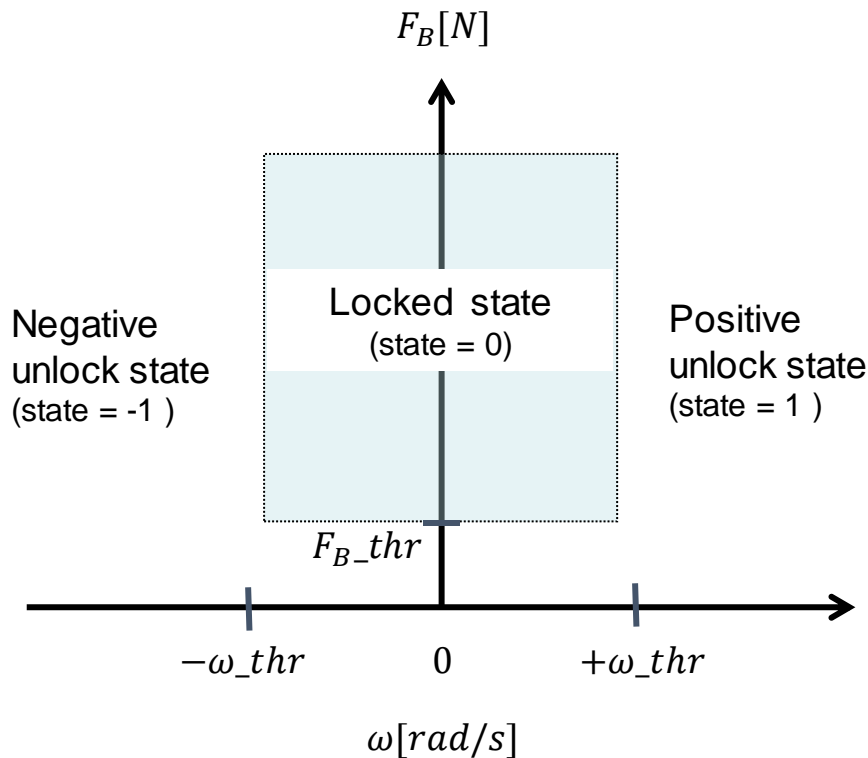


Fig.4-11 ブレーキ状態判定

Positive unlock state の場合は、下記式でブレーキトルクを計算する。

$$\tau_k = -\mu \times r_{eff} \times F_B$$

一方、Negative unlock state の場合は、下記式となる。

$$\tau_k = \mu \times r_{eff} \times F_B$$

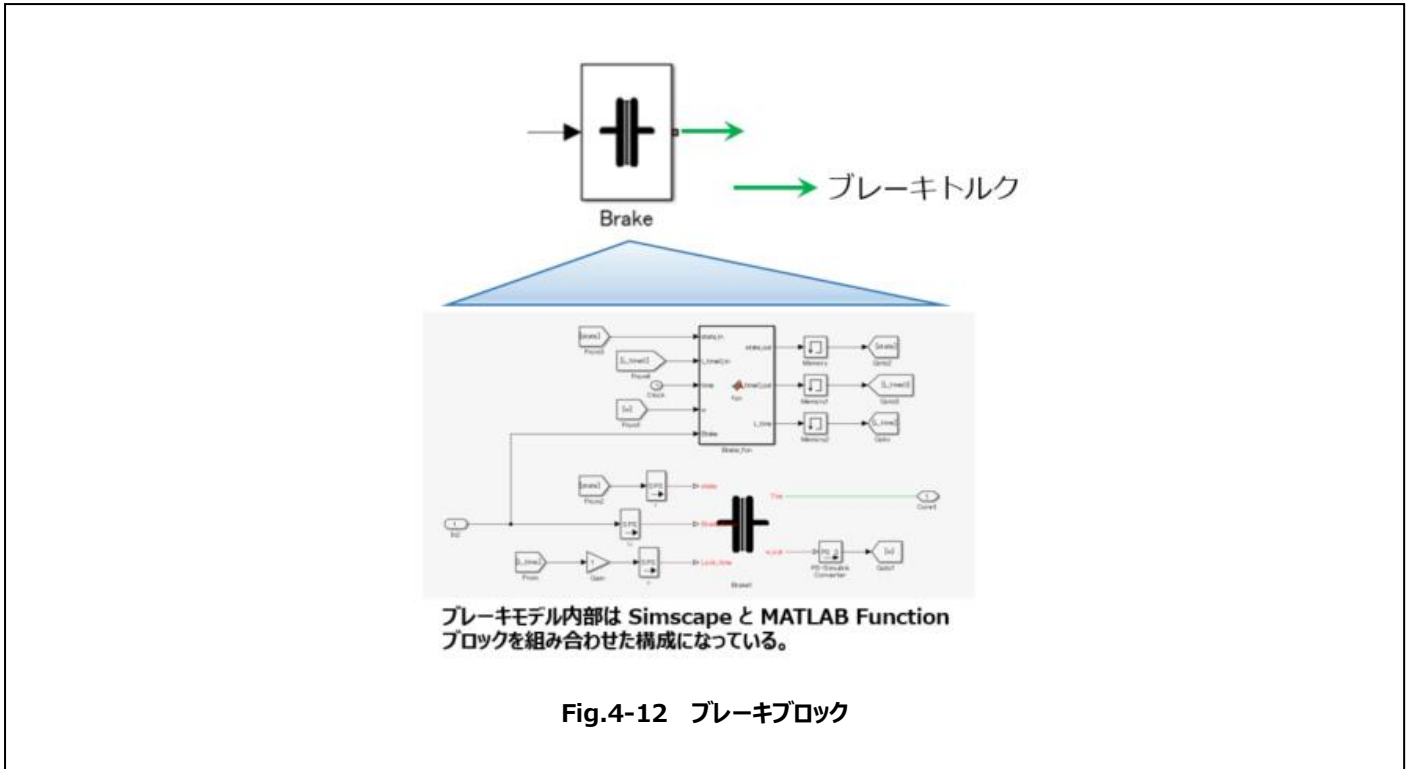
ここで、 $\tau_k [Nm]$ は、ブレーキトルクを表し、ブレーキパッドにかかる力 $F_B [N]$ に比例する。 $\mu [-]$ は動摩擦係数、 $r_{eff} [m]$ は有効半径である。

Unlock state の場合、回転方向と逆に摩擦力によるブレーキトルクがかかる。しかし、 F_B が十分大きくかつ相対回転 ω が 0 になると、2 つの剛体は結合し、1 つの剛体として振る舞う。このとき、物理方程式は、2 つの運動方程式が 1 つの運動方程式となるが、Simscape ではシミュレーション中に方程式の数を変えることはできないため、何らかの近似が必要となる。本入門書では、近似の方法として、以下の方法を行った。

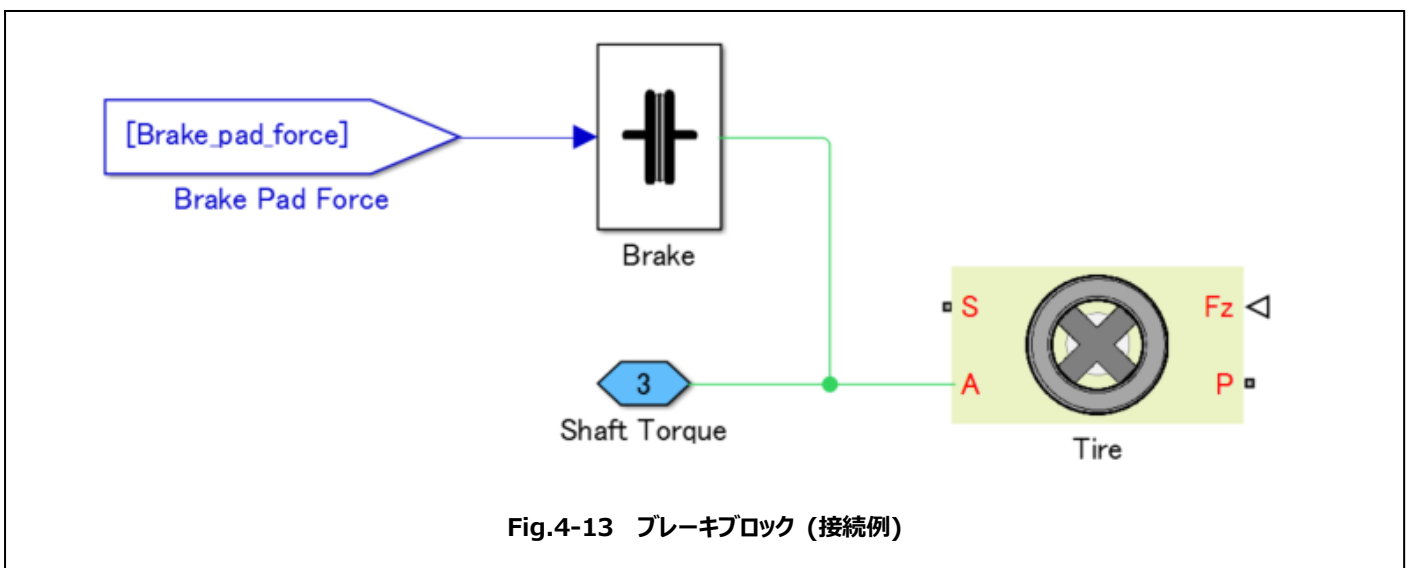
$$\tau_k = -\mu \times r_{eff} \times F_B \times \frac{\omega}{\omega_{thr}} \times \exp\left(\frac{T_L}{\Delta T}\right)$$

ここで、 T_L は Locked state の継続時間、 ΔT は半減期である。

(1) モデルブロック



(2) 接続例



(3) モデル詳細

名称	Brake					
概要	3自由度の簡易車両モデル					
	トルクの向き：外から内が正					
	計算内容：					
	① 入力された圧着力から制動トルクを計算 ② ブレーキの状態を判定し、制動力の値を決定(開放状態：0[N*m]、ロック状態は指数関数的に減衰)					
ノード	ポート名	物理ドメイン	Through/Across	内容	表示位置	
1	Tire	機械(回転)	回転数[rad/s]、 トルク[N*m]	タイヤと制動トルクのやり取り	right	
入力	ポート名	内容	範囲	初期値	単位	表示位置
1	state	状態(ロック、アンロックなど)	-	0	-	left
2	Brake_force	圧着力	-	0	N	left
3	Lock_time	ロック状態の継続時間	-	0	s	left
出力	ポート名	内容	範囲	初期値	単位	表示位置
1	w_out	タイヤの回転数	-	0	rad/s	right
パラメータ	パラメータ名	内容	範囲	初期設定値	単位	
1	w_thr	ロックとアンロック状態の回転数の閾値	-	3	rad/s	
2	T_L	ロック状態時のトルク減衰の半減期	-	1	s	
3	uk	動摩擦係数	-	1	-	
4	Er	有効半径	-	1	m	
5	eps	条件判定用の微小値	-	0.1	-	
変数	変数名	内容	範囲	初期値	単位	
1	w_T	タイヤの回転数	-	0	rad/s	
2	t_F	制動トルク(状態考慮前)	-	0	N*m	
3	t_T	制動トルク	-	0	N*m	
4	state	状態	-	0	-	

方程式、その他

圧着力から制動トルクを計算

$$t_F = -uk * Er * Brake_force$$

状態を考慮し、制動トルクを決定する

if アンロック状態で順回転

$$t_T = T_F$$

else if アンロック状態で逆回転

$$t_T = -t_F$$

else if ロック状態

$$t_T = t_F / w*thr * w_T * \exp(L_time / T_L)$$

else (開放状態)

$$t_T = 0$$

(4) Simscape コード

```
component Brake
% Brake
% Ver. 1.0 Jun/22, 2017 JMAAB Plant Modeling Workshop

nodes
    Tire = foundation.mechanical.rotational.rotational; % Tire:right
end

inputs
    state_in = { 0, '1' }; % state:left
    Brake_force = { 0, 'N' }; % Brake_force:left
    L_time = {0, 's'}; % Lock_time:left
end

outputs
    w_out = { 0, 'rad/s' }; % w_out:right
end

variables
    w_T = { 0, 'rad/s' }; % Rotational velocity
    t_F = { 0, 'N*m' }; % Brake Torque(dummy)
    t_T = { 0, 'N*m' }; % Brake Torque(output)
end

parameters
    w_thr = {3.0, 'rad/s'} ;
    Brake_force_thr = {0.1, 'N'} ;
    T_L = {1.0, 's'} ;

    uk = { 1.0, '1' }; % Kinetic friction coefficient
    Er = { 1.0, 'm' }; % Effective radius
end

parameters (Access=private)
    eps = {0.1, '1'} ;
end
```

```

branches
    t_T : * -> Tire.t;
end

equations
    w_T == Tire.w;

    state==state_in;

    t_F == -uk * Er * Brake_force ;

    %% Positive Unlock state
    if( lt(abs(state-1),eps ))
        t_T == t_F;

    %% Negative Unlock state
    elseif( lt(abs(state + 1),eps ))
        t_T == -t_F;

    %% Lock state
    elseif( lt(abs(state),eps ))
        t_T == t_F/w_thr * w_T * exp(L_time /T_L );

    else
        t_T == 0.0;
    end
    w_out == w_T ;
end
end

```

4.4 評価

ここでは、一定の車速と一定の舵角で旋回させた時(定常円旋回)の軌跡を理論値と比較することで、モデルの評価を行った。検証を簡単に行うために、過渡状態ではなく定常状態のシミュレーションで評価とした。

定常円旋回の半径を求める理論式は、自動車の運動と制御^[4-2]を参考に導出した。

定常円旋回時の車両の速度 V と角速度 ω 、旋回半径 r の関係は、

$$r = \frac{V}{\omega}$$

車両横方向の力のつりあいは、自動車の運動と制御によると、各タイヤの横滑り角と実舵角が微小な場合、次のようにあらわせる。

$$mV\omega = Y_1 + Y_2 + Y_3 + Y_4$$

ここで、 m は車両の慣性質量、 Y_i は各タイヤに働くコーナリングフォースである。

上記の 2 式より、定常円旋回時の旋回半径は、

$$r = \frac{mV^2}{Y_1 + Y_2 + Y_3 + Y_4}$$

旋回半径の理論値は、上記の式を用いて求めた。

下記の手順で定常円旋回のシミュレーションを実施した。

- ① 一定のトルク(前輪にそれぞれ 100[N*m])を入力し加速
- ② 車速が一定になり定常円旋回
- ③ 一定のトルク(前輪にそれぞれ 200[N*m])を入力し加速
- ④ 車速が一定になり定常円旋回

シミュレーション時の実舵角(タイヤの切れ角)は、2[deg]である。

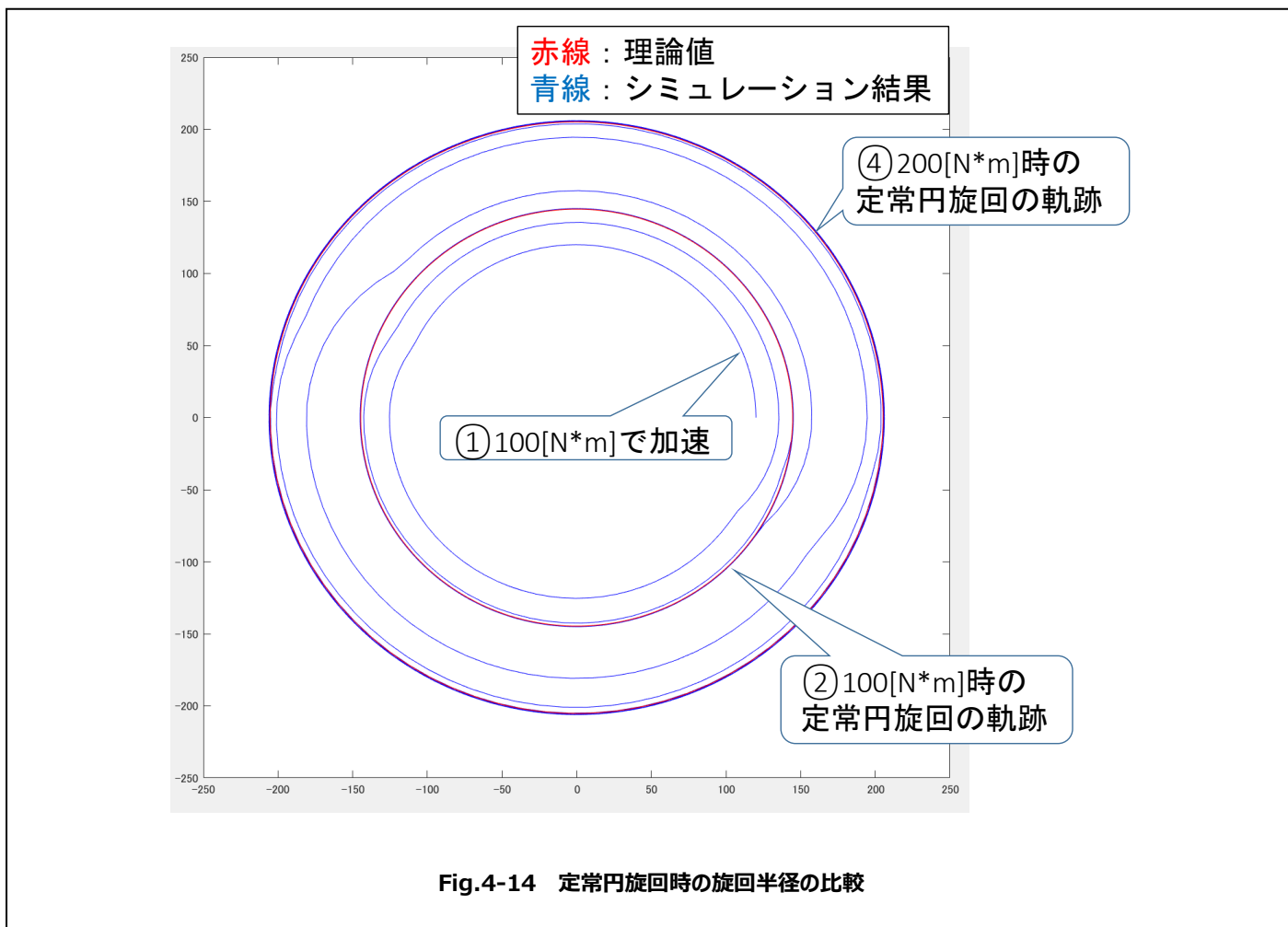


Fig.4-14 定常円旋回時の旋回半径の比較

表 4-1 定常円旋回時の旋回半径の比較

	入力トルク 100[N*m]	入力トルク 200[N*m]
理論値	144.7[m]	205.3[m]
シミュレーション	144.7[m]	205.5[m]

定常円旋回時の軌跡を理論値とシミュレーション結果で比較を行ったものが Fig.4-14 と 表 4-1 である。理論値とシミュレーション結果の旋回半径がほぼ一致することが確認できた。

【参考文献】

- [4-1] 堀内 泰 : 車両挙動安定化制御システムにおけるタイヤモデルの提案, 自動車技術会
- [4-2] 安部 正人 : 自動車の運動と制御 (車両運動力学の理論形成と応用),

5. Simscape Tips

本章では、Simscape のモデリングからシミュレーション結果の確認までのひと通りの流れに関する情報を紹介する。

5.1 Simscape モデルの作成

ここでは、Simscape モデル作成のための便利機能、端子接続に関する注意事項について説明する。なお、Simscape の新規モデル作成から結果表示までの流れが、下記 URL で確認できる。

<https://jp.mathworks.com/help/physmod/simscape/gs/essential-steps-for-constructing-a-physical-model.html>

5.1.1 モデル作成時の便利機能

Simscape モデル作成時に、便利な機能を 2 つ紹介する。

(1) ssc_new コマンド

ssc_new コマンドをコマンドウィンドウで実行すると、Simscape モデルをシミュレーションするのに必要な Solver Configuration ブロックを含む新規モデルが起動する(Fig. 5-1 参照)。

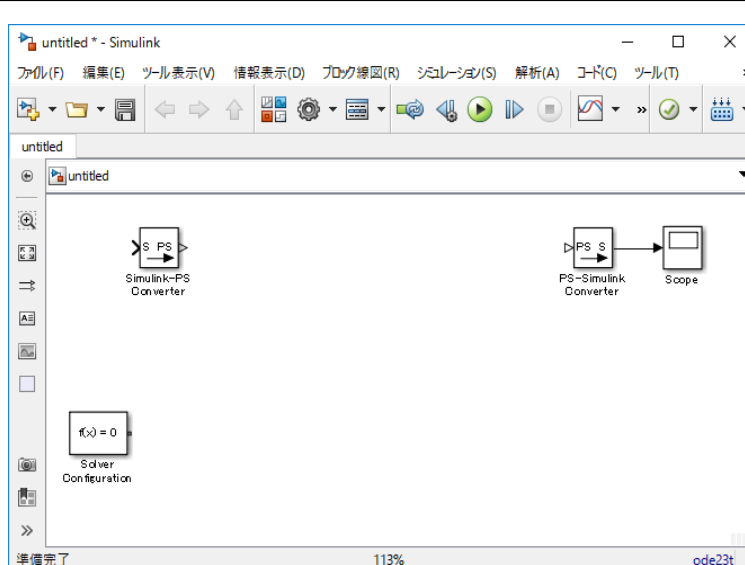


Fig. 5-1 ssc_new コマンドを実行して開くモデル

(2) テンプレート機能

R2015b 以降のバージョンでは、電気、機械（回転・並進）などのドメイン毎にテンプレートがある。Fig. 5-2 の Simulink アイコンをクリックして開く Simulink スタートページにて、主に使用するドメインのテンプレートが用意されている。使用するドメインのテンプレートを選択することで、リファレンスブロックを含む、Simscape モデルに必要なブロックを含むモデルが起動する。

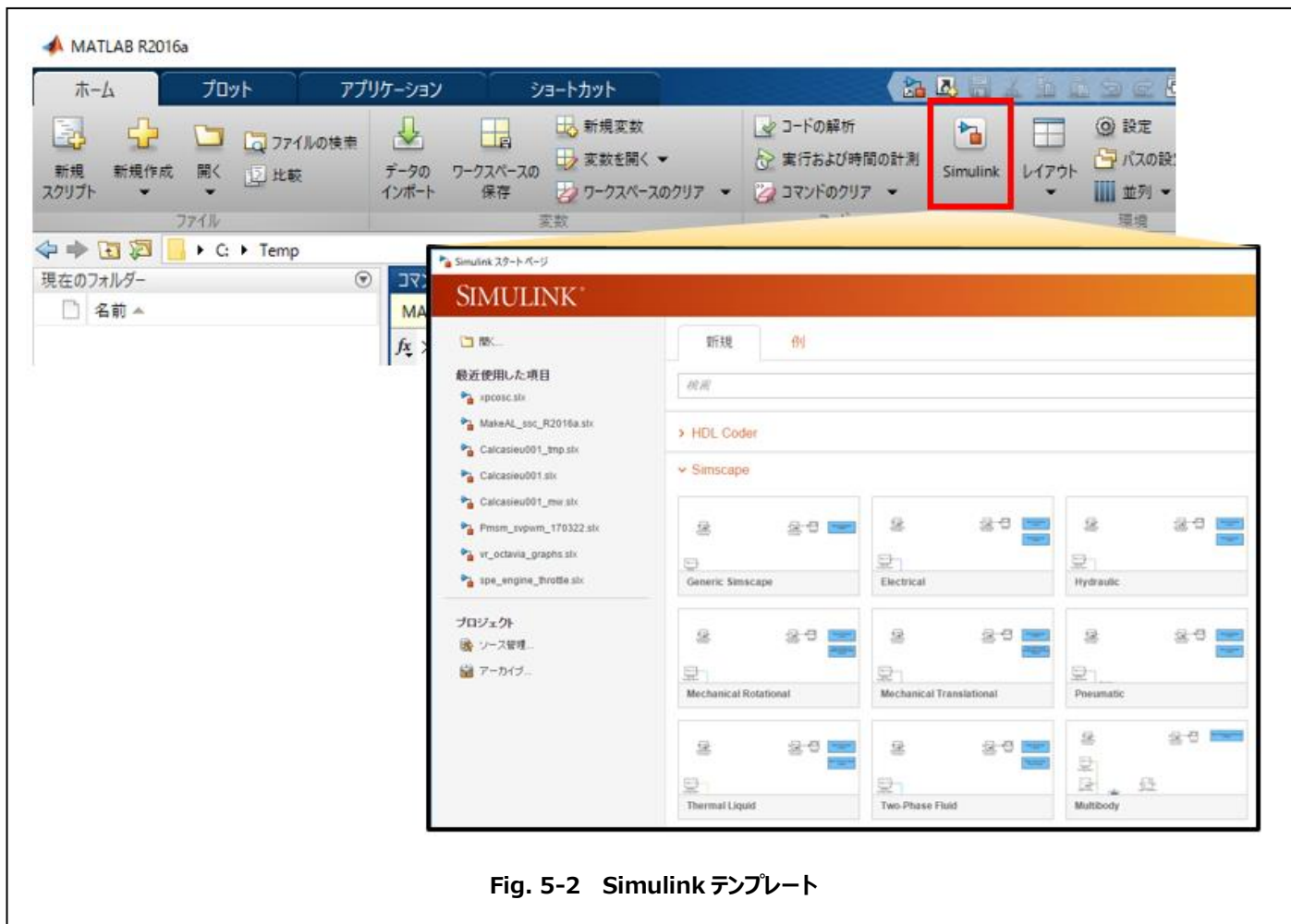


Fig. 5-2 Simulink テンプレート

5.1.2 Simulink の端子と Simscape の端子の接続方法

Simulink の端子と Simscape の端子は直接接続できないため、変換が必要となる。Simulink の端子を Simscape に端子に接続するには Simulink-PS Converter ブロックを、Simscape 端子を Simulink に端子に接続するには PS-Simulink Converter ブロックを使用する。下記ドキュメントページで確認できる。

<https://jp.mathworks.com/help/physmod/simscape/ug/connecting-simscape-diagrams-to-simulink-sources-and-scopes.html>

5.1.3 カスタムコンポーネントの作成

Simscape では、カスタムコンポーネントを作成することが可能である。下記に、本書で公開している情報を補足するための簡単な説明を行い、ドキュメントリンクを紹介する。

カスタムコンポーネントを作成するための構文、概要、コンポーネントの一例など、下記 URL から参照可能となっている。

<http://www.mathworks.com/help/physmod/simscape/creating-custom-components.html>

次に、Simscape を使用するうえで重要な変数について紹介する。Simscape は、一般に言われている、非因果系モデリングツールの 1 つである。コンポーネント間の接続を考える際に重要となるのが、下記の 2 つのタイプの変数である。

- ・スルー変数
- ・アクロス変数

なお、機械（回転、並進）・電気などのドメインに関する、スルー変数、アクロス変数は、下記ドキュメントページにて確認できる。

<https://jp.mathworks.com/help/physmod/simscape/ug/basic-principles-of-modeling-physical-networks.html#bq89sba-3>

コンポーネント同士は、端子を信号線で接続することで関連づけられる。Fig 5-3 にあるような、コンポーネントの端子（ノード）A, B に対し、「流れ」が A から B に向かえばスルー変数（TV）は符号が正、アクロス変数は $AV = AV_A - AV_B$ で決定される。Fig. 5-3 のコンポーネントを抵抗に見立てると、スルー変数（TV）の向きが電流の流れる方向を表し、アクロス変数（AV）が抵抗にかかる電圧を表していることが理解できる。

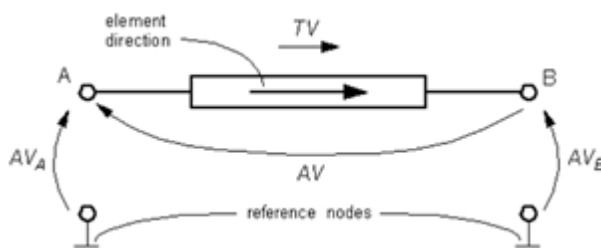


Fig. 5-3 変数の向き

変数の向きの説明は、下記ドキュメントにて確認することが可能である。

<https://jp.mathworks.com/help/physmod/simscape/ug/basic-principles-of-modeling-physical-networks.html#bq89sba-5>

5.2 シミュレーションのための推奨設定

Simscape モデルをシミュレーションする際の設定として、モデルのコンフィギュレーションパラメータ、Simscape 専用の設定がある。ここでは、それぞれの推奨設定を紹介する。

5.2.1 モデルのコンフィギュレーションパラメータ（ソルバー・ゼロクロッシング）

Simscape モデルのシミュレーションに影響するモデル設定について紹介する。

(1) ソルバー設定

モデルのコンフィギュレーションには、ソルバを設定する箇所がある。Simscape を使ってシミュレーションする場合、ソルバーの選択は、可変ステップソルバーを選ぶ場合は ode15s または ode23t を、固定ステップソルバーを選ぶ場合は ode14x を推奨している。

<https://jp.mathworks.com/help/physmod/simscape/ug/making-optimal-solver-choices-for-physical-simulation.html>

ここで、可変ステップソルバーは、Simulink ソルバーが収束状況を判定し、ステップサイズを自動調整してシミュレーションを行う。これに対し、固定ステップは、指定したステップサイズでシミュレーションを行う。

Simscape でシミュレーションを行う場合、まずは可変ステップソルバーでシミュレーションし、ステップ幅を確認する。次に、計算速度の高速化を行うのであれば、ソルバーを固定ステップソルバーに変えて、シミュレーション結果が妥当か確認しながらサンプル時間を決定するアプローチが一般的である。下記 URL に、高速化のワークフローに関して記述されている。

<https://jp.mathworks.com/help/physmod/simscape/ug/real-time-model-preparation-workflow.html>

(2) ゼロクロッシング

ソルバーコンフィギュレーションの設定について、ゼロクロッシングの設定項目がある。ゼロクロッシングは、時間の経過に伴うシステムの状態やダイナミクスの不連続な変化を表す。ゼロクロッシングを正確に特定することが、期待するシミュレーション結果を得るためには重要である。まずは、ゼロクロッシングの設定を有効と設定することを推奨する。

5.2.2 Solver Configuration ブロック

Solver Configuration ブロックでは、Simscape 回路をシミュレーションするための設定を行う。設定の 1 つである、ローカルソルバーは、Simscape 回路を離散化するか否かを決定する。まずは、OFF に設定し、連続として扱うことを推奨する。

5.3 シミュレーション結果の確認

シミュレーション終了後、結果を確認する際の便利機能を紹介する。

5.3.1 シミュレーション データのプロット

Simscape 回路にある各ブロックのスルー変数、アクロス変数について、Fig. 5-4 のように、時刻歴応答が確認できる。

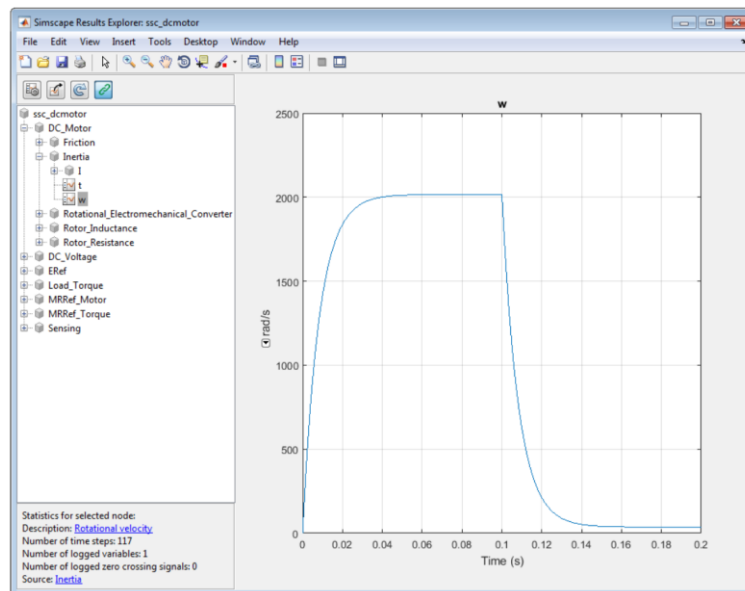


Fig. 5-4 Simscape Result Explorer

確認するには、モデルのコンフィギュレーションパラメータの [Simscape] ペーンにある 'Log simulation data' を All に設定し、'Open viewer after simulation' にチェックを入れてシミュレーションを行う。コンフィギュレーションパラメータの設定を図解しているものが、下記のドキュメントページで確認できる。

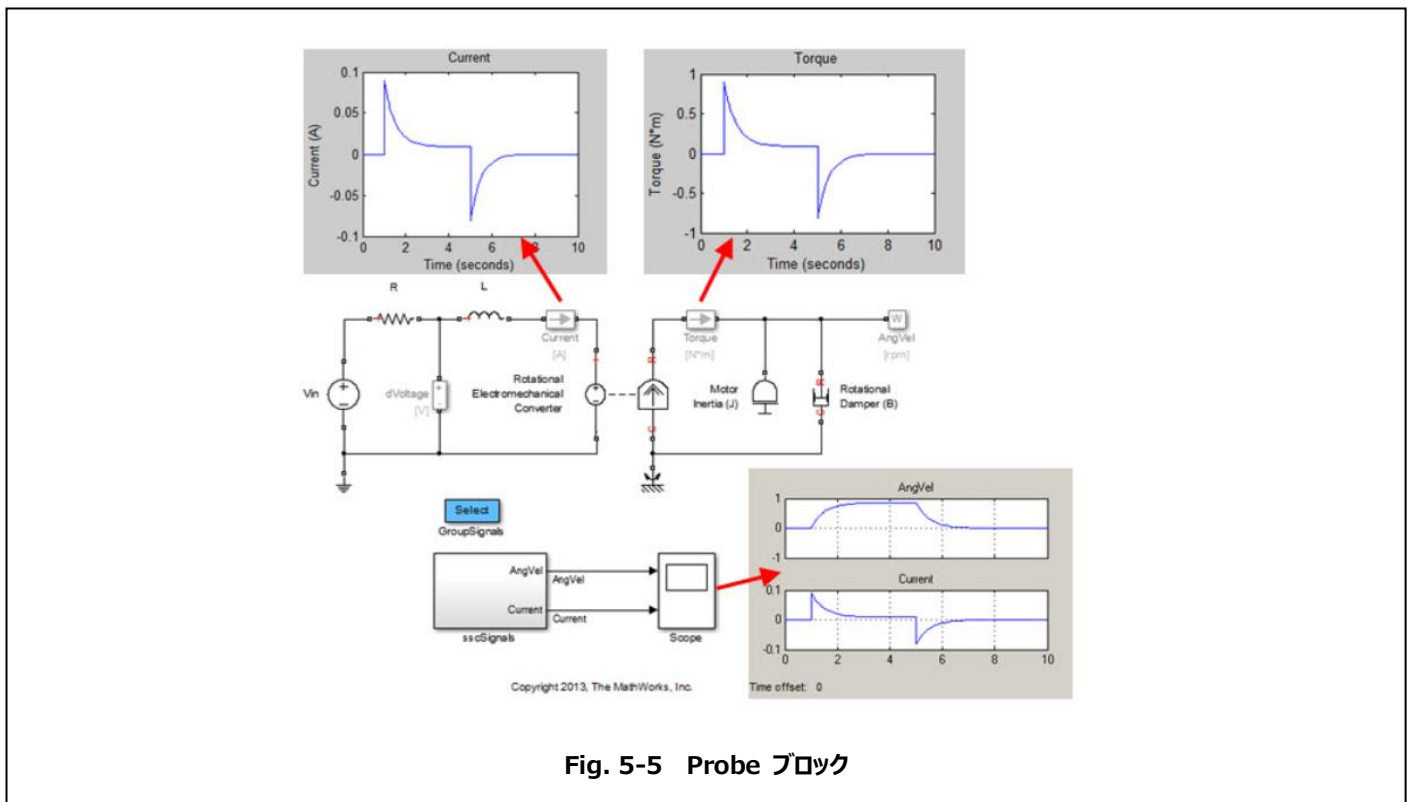
<https://jp.mathworks.com/help/physmod/simscape/ug/log-and-plot-simulation-data.html>

また、シミュレーション結果を確認する Simscape Result Explorer の使い方に関する説明は、下記のドキュメントページで確認できる。

<https://jp.mathworks.com/help/physmod/simscape/ug/using-the-simscape-results-explorer.html>

5.3.2 Probe ブロック

MATLAB Central (ユーザーコミュニティサイト) にて、ブロック単体を接続して、手軽に応答を確認できる、Probe ブロックライブラリが公開されている。



本ブロックは、下記 URL からダウンロードできる。

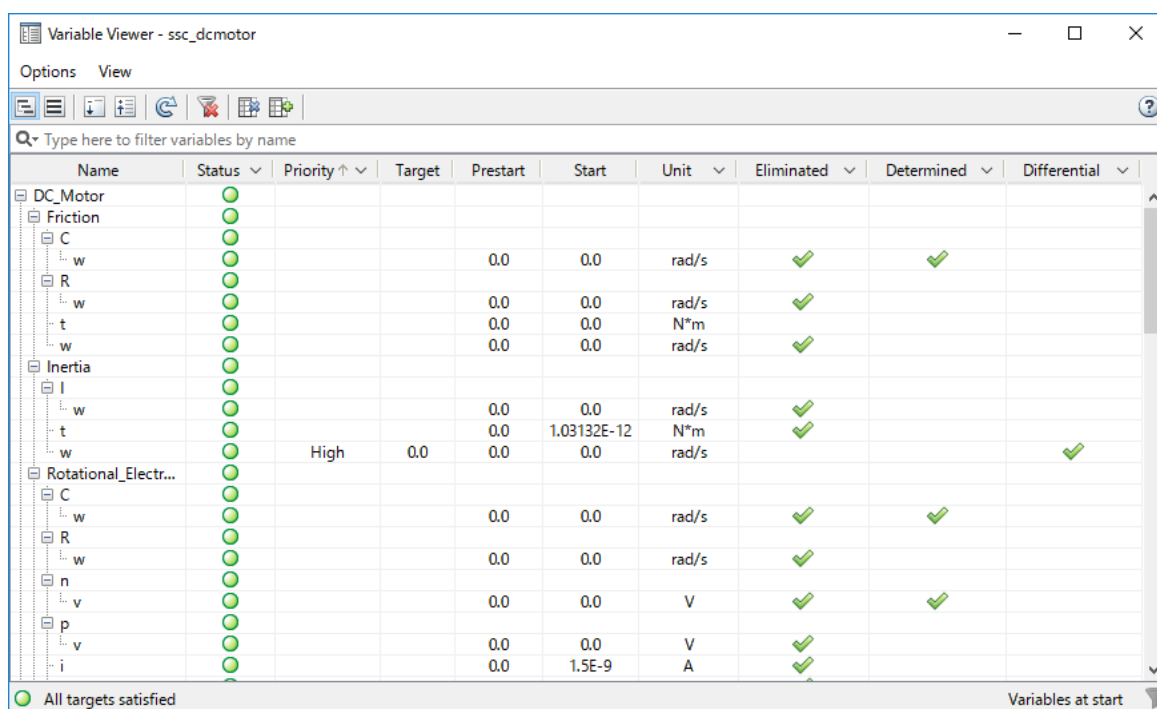
<http://jp.mathworks.com/matlabcentral/fileexchange/17969-simscape-probes>

5.4 変数の初期値の確認

シミュレーション開始時に、初期化に関するエラーが出て、シミュレーションできないことがある。ここでは、Simscape モデルの初期値やステータスの確認する機能、注目する設定について説明する。

5.4.1 Variable Viewer

Variable Viewer では、Simscape モデルの初期化ステータス、設定した初期値などが確認できる。



Name	Status	Priority	Target	Prestart	Start	Unit	Eliminated	Determined	Differential
DC_Motor	●								
Friction	●								
C	●								
w	●			0.0	0.0	rad/s	✓	✓	
R	●								
w	●			0.0	0.0	rad/s	✓		
t	●			0.0	0.0	N*m			
w	●			0.0	0.0	rad/s	✓		
Inertia	●								
I	●								
w	●			0.0	0.0	rad/s	✓		
t	●			0.0	1.03132E-12	N*m	✓		
w	●	High	0.0	0.0	0.0	rad/s			✓
Rotational_Electr...	●								
C	●								
w	●			0.0	0.0	rad/s	✓	✓	
R	●								
w	●			0.0	0.0	rad/s	✓		
n	●								
v	●			0.0	0.0	V	✓	✓	
p	●								
v	●			0.0	0.0	V	✓		
i	●			0.0	1.5E-9	A	✓		

Fig. 5-6 Variable Viewer

Variable Viewer について、下記ドキュメントページにて確認できる。

<https://jp.mathworks.com/help/physmod/simscape/ug/variable-viewer.html>

5.4.2 初期値と優先度の設定

特定の変数について初期化に失敗する場合、初期値や優先度の設定を変更することで、エラーを回避できることがある。設定画面に関して、下記ドキュメントページに記載されている。

<https://jp.mathworks.com/help/physmod/simscape/ug/set-priority-and-initial-target-for-block-variables.html>

