CB-001

標準技術 FMI/SSP を活用したシステムシミュレーション手法の提案
An Approach of System Simulation using FMI and SSP Standard

磯田 誠† 大森 康宏†
Makoto Isoda Yasuhiro Ohmori

1. はじめに

近年の産業界では，製品やサービスといったシステムの品質確保および開発効率化の両立を目指して，モデルベースシステムズエンジニアリング(Model-Based Systems Engineering，MBSE)の普及を推進している．また，システムを構成する個々の機器や装置についてはモデルベース開発(Model Based Development，MBD)が定着してきている．抽象度は違うが，ともに開発対象を記述するだけでなくシミュレーションして動作確認できるという特長がある．

このような特長を持つこともあって，定着してきている機器のシミュレーション技術をシステムの品質確保や開発効率化にも応用しようという状況にある．その一例として，Modelica 言語を用いてシステムをモデル化し，標準技術 FMI(Functional-Mockup Interface)[1] および SSP(System Structure and Parameterization)[2]に準拠した処理系(シミュレーションツール，シミュレーションエンジン，数値ソルバなどともいう)を用いてシステムシミュレーションする方法が，産業界でも熱心に取り組まれている[3][4].

しかし，システムシミュレーションに用いられる処理系は技術的に完全に成熟しているわけではなく，システムをモデル化できたとしても計算結果が異なったり計算に失敗するものが出てきたりすることがある．また，数多くの処理系が乱立しており，プロジェクトの費用対効果に悪影響を与えない処理系を採用できているか評価が難しい.

そこで本稿では，数多くの種類の処理系を実行した結果から全体として妥当な計算結果が得られ，かつプロジェクトのコスト増加も抑えるシステムシミュレーション手法を提案する．そして，提案手法の実現性を確認した結果を報告する.

2. 課題および解決のための施策

2.1 開発現場から捉えた課題

開発現場でシミュレーション技術を応用する時期には，最終成果物はまだなく中間成果物だけが存在するものと想定する．また，開発プロセスの観点では，要件定義・方式設計・詳細設計・製作・テストといった本流の活動を補強する活動でシミュレーション技術を用いると想定する.

以上より，以下の 2 点を開発現場の課題として捉えた.

(1) 正解とぴったり一致するというよりおおよそ妥当な結果が得られ，明らかな誤りを発見できること.
(2) シミュレーション技術の応用に余計な労力がかからず，開発本流のコスト増加を抑えること.

2.2 課題解決のための施策方針

2.1 に示した課題(1)と(2)を解決する施策方針をそれぞれ以下のように立て，これを具体的に実現するシステムシミュレーション手法を提案する.

(a) 産業界で使用実績のある標準技術を活用した上で，さらに計算結果の妥当性を保証するために工夫する.
(b) シミュレーション手順を定型化して，習得の手間や作業ミスなど新たなロスコストを抑制する.

課題(1)に対応する方針(a)では，産業界で熱心に取り組まれている標準技術 FMI/SSP に準拠することで計算結果の基本的な正しさを保証する．加えて，多くの種類の処理系を用いて出した計算結果を波形とみなして，波形類似度が高い計算結果が妥当な解であると判定する.

課題(2)に対応する方針(b)では，産業界でよく用いられている Modelica および Simulink を中心としたシミュレーション手順を規定する．技術的に成熟している言語やツールを採用し，社内外で蓄積されているノウハウを活用することで，作業定型化を推し進める.

2.3 施策(a) 波形類似度を用いた妥当性保証

制御器(コントローラ)および制御対象(プラント)の仕様記述をここではシステムモデルと呼ぶ．システムモデルを予め準備して，図 1 に示すように入力波形の時系列データを与えて各種処理系で計算を実行した結果から，出力波形の時系列データを得る.

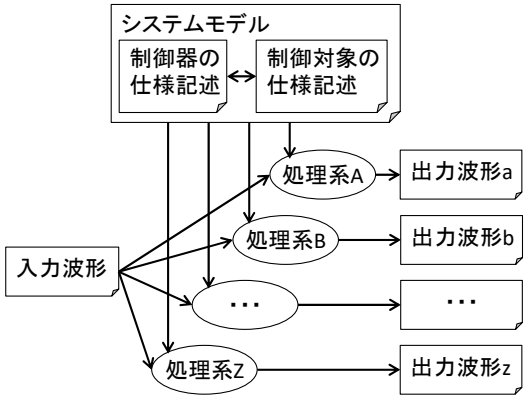

図 1 システムモデルと入力／出力波形の関係

各種処理系に対応して得られた複数の出力波形同士の値や形状など，特徴が類似する度合いを波形類似度と呼ぶ．類似度が高い波形は一致するものとみなし，類似度の点で一致した出力波形の中から，最終的な出力波形を選定する.

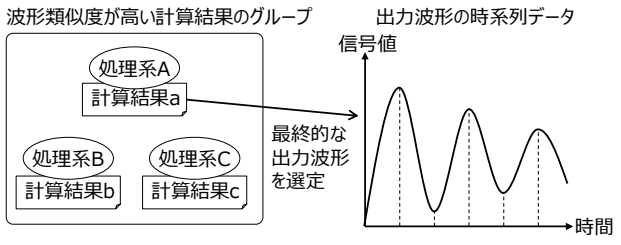

図 2 類似度が高い出力波形から最終的に選定

† 三菱電機株式会社 情報技術総合研究所 Mitsubishi Electric Corporation, Information Technology R & D Center

1 種類の処理系を用いるとするとただ 1 つの出力波形が得られるか全く得られないかである．さらに，ただ 1 つの出力波形が得られてもそれだけからは妥当性を判断できない．これに対して，互いに類似度が高い出力波形が複数存在すれば，すなわち計算結果が一致する処理系が複数種類存在すれば，計算結果の妥当性が高いと判断する．

2.4 施策(b) Modelica および Simulink を中心にしたシミュレーション手順

制御対象を表す非因果モデルを作成するところから始めて，システムシミュレータのソースコードを生成するまで通した手順を経験的に 5 ステップで規定する．

ここでは MSL(Modelica Standard Library)を用いることとして，作成したモデルを Modelica モデルと呼ぶ．また，制御器を Simulink で作成する場合は，制御器に対応する部分を一旦 Modelica で作成した後に Simulink に置き換える．

(1) **制御対象の非因果モデル作成**
制御対象を定式化した後に，MSL が提供するブロックを用いて Modelica モデルとして表現する．
このとき，仮の制御器を表す何らかの入力ブロックを付加しておく．

(2) **インタフェースを因果接続**
MSL が提供する物理ドメインの因果化アダプタを用いて，仮の制御器と制御対象のインタフェースを非因果接続から因果接続に変換する．

(3) **モデル分解と FMU 作成**
因果接続した箇所で Modelica モデルを分解して，個々のモデルに入力／出力信号を付加する．その後に，標準技術 FMI で規定する Co-Simulation タイプの FMU(Functional Mock-up Unit)にエクスポートする．
ここで制御器を表す Simulink モデルを因果接続すると，開発するアルゴリズムの良し悪しを確認できる．この時点で元々のモデルが Modelica なのか Simulink なのか意識する必要はなくなる．

(4) **システムシミュレータ組立て**
FMU とその間の結線に関する情報を，標準技術 SSP で規定される SSP ファイル(System Structure Package)にエクスポートする．
ZIP 形式の SSP ファイル中に自動生成される SSD ファイル(System Structure Description)から，シミュレーションに必要十分なシステムの要素・接続関係を抽出する．一緒にパラメータも抽出する．

(5) **シミュレータコード生成**
シミュレーションを実行する処理系を選択し，その上で動くシステムシミュレータのソースコードを自動生成する．

3. 標準技術 FMI/SSP 活用システムシミュレーション手法の実現性確認

2.3 および 2.4 に示した提案手法の実現性を確認するため，7 種の処理系に 8 個のサンプルモデルを解かせて，「計算結果の妥当性」および「作業定型化の度合い」を評価する．

3.1 処理系およびサンプルモデルの説明

本手法で候補に挙げる処理系を表 1 に示す(アルファベット順)．執筆時点で利用できる代表的な処理系(https://fmi-standard.org/tools/)からオープンソースライセンスを中心に選び，これに社内で使用実績が多い Simulink を加えた．

表 1　シミュレーションの処理系の候補

処理系	ベンダ	ライセンス	言語
Cosim	OSP	MPL2	コマンドライン
DACCOSIM	EDF	LGPL	専用
FMPy	Dassault Systèmes	2-clause BSD	Python
MasterSim	Bauklimatic Dresden Software	3-clause BSD	専用
OpenModelica	OSMC	GPL	Modelica
PyFMI	Modelon	LGPL	Python
Simulink	MathWorks	商用	専用

DACCOSIM: Distributed Architecture for Controlled CO-SIMulation, OSP: Open Simulation Platform, OSMC: Open Source Modelica Consortium

処理系に解かせるサンプルモデルと，取り扱う物理現象と，制御器に搭載する制御則を表 2 に示す．

表 2　サンプルモデルと物理現象と制御測

物理ドメイン	題材	物理現象	制御測
回転機械	回転台	物体の軸にトルクを加える	PD 制御
並進機械	台車	物体の端に移動力を加える	PD 制御
アナログ電気	RLC 回路	抵抗，インダクタ，キャパシタに電圧を印加	I-PD 制御
	増幅回路	オペアンプ，キャパシタに電圧を印加	I-PD 制御
伝熱	ヒーター	物体の端から加熱する	PI 制御
複合	回転型電動機	モータでトルクを発生させる	P 制御
	直動型電動機	モータで移動力を発生させる	P 制御
マルチボディ機械	垂直駆動アーム	鉛直下向きのアームにトルクを加える	PI-D 制御

本稿ではこの中から，回転機械ドメインの「回転台」とマルチボディ機械ドメインの「垂直駆動アーム」を解いた結果を紹介する．物体と軸からなる回転台の模式図，回転台の Modelica モデル，これに PD 制御をかける制御器の Simulink モデルをそれぞれ図 3，図 4，図 5 に示す．

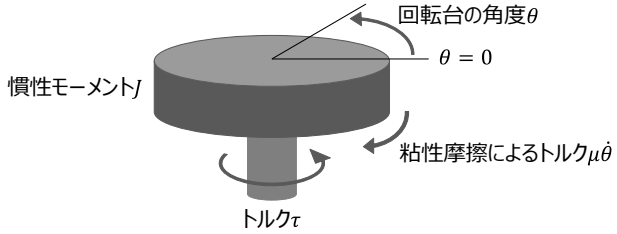

図 3　物体と軸からなる回転台の模式図

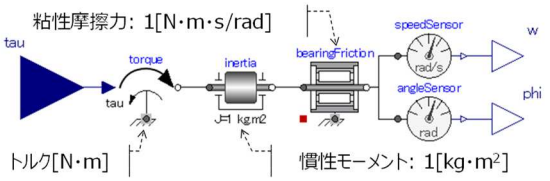

図 4　回転台の Modelica モデル

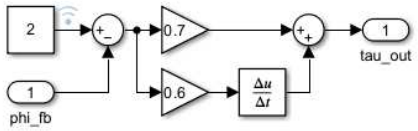

図 5　PD 制御をかける制御器の Simulink モデル

Simulink モデルと Modelica モデルの間で分解し，それぞれ FMU にエクスポートして FMU 間を結線した構造図を図 6 に示す．図 6 からシミュレータコードを生成すると，シミュレーションが実行できるようになる．

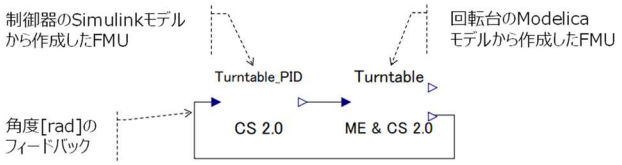

図 6　エクスポートした FMU 間を結線した構造図

同様に，「垂直駆動アーム」の模式図，Modelica モデル，Simulink モデルをそれぞれ図 7，図 8，図 9 に示す．

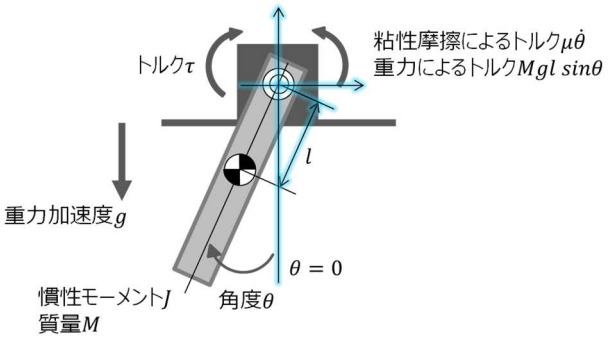

図 7　垂直駆動アームの模式図[5]

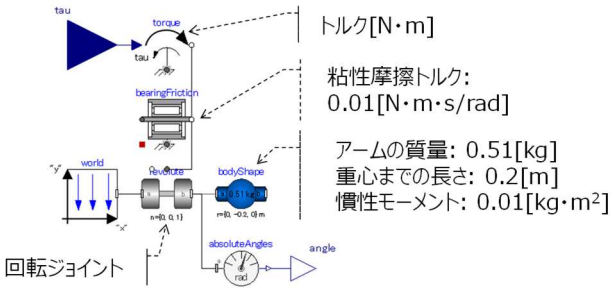

図 8　垂直駆動アームの Modelica モデル

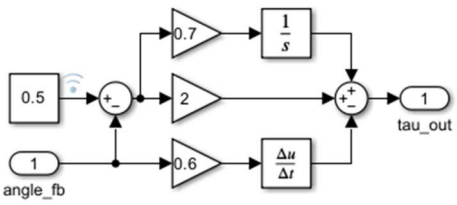

図 9　PI-D 制御をかける制御器の Simulink モデル

シミュレーションのパラメータは共通で以下のとおり．

- 開始時刻　0.0[s]
- 終了時刻　10.0[s]
- ステップサイズ　0.01[s]

3.2 計算結果の妥当性の評価

DTW(Dynamic Time Warping，動的時間伸縮法)[6]という時系列クラスタリング手法の一種を用いて，サンプルモデルごとに出力信号の総当たり組合せに対して波形類似度を求める．そして，

- 波形類似度が高い結果が出た処理系同士を同じグループに分類して，グループ番号を付ける．
- いくつかのグループに分かれた場合は，処理系の数が最も多いグループ番号を有力候補とする．
- すべての出力信号が有力候補に入ってくる処理系の中からいずれかを採用する(どれでもよい)．

「回転台」の波形類似度を求めて処理系にグループ番号を付けた結果を表 3 に示す．グループの総数は 3 つとしてグループ番号 0, 1, 2 で識別するようにした．

凡例: 0, 1, 2 グループ番号(太字は処理系の数が最も多い)
－ 波形記録なし(処理系によっては記録されない)

表 3　回転台の出力信号のグループ番号

題材	出力信号	Cosim	DACCOSIM	FMPy	MasterSim	OpenModelica	PyFMI	Simulink
回転台	角度	**0**	**0**	**0**	2	1	**0**	**0**
	トルク	0	**2**	**2**	－	1	**2**	－
	角速度	**0**	**0**	**0**	**0**	1	**0**	2

同様に，「垂直駆動アーム」の結果を表 4 に示す．

表 4　垂直駆動アームの出力信号のグループ番号

題材	出力信号	Cosim	DACCOSIM	FMPy	MasterSim	OpenModelica	PyFMI	Simulink
垂直駆動アーム	角度	**1**	**2**	**1**	**2**	**1**	**2**	0
	トルク	**1**	**1**	0	－	**1**	2	－

ここで，計算機環境の諸元は共通で以下のとおり．

- CPU:　Intel Core i5-8500, 3.00GHz
- OS:　Windows10 Enterprise
- メモリ:　32.0GB
- ストレージ:　930GB

まず表 3 について，太字で示したグループ番号を持つ処理系を有力候補とする．次に，出力信号ごとに解ける処理系が異なるので，それらの積集合の中から採用する．すると，赤太枠で囲んだ DACCOSIM，FMPy，PyFMI のいずれを採用してもよいという結果になる．同様に，表 4 では Cosim，DACCOSIM，OpenModelica のいずれでもよい．

この結果を含めて，サンプルモデルごとに最終的に採用してもよい処理系を表 5 に示す．

凡例: ○ 採用してもよい, ― 採用しない

表 5　サンプルモデルごとの処理系の採用可否

物理ドメイン	題材	Cosim	DACCOSIM	FMPy	MasterSim	OpenModelica	PyFMI	Simulink
回転機械	回転台	―	○	○	―	―	○	―
並進機械	台車	○	○	―	―	―	○	―
アナログ電気	RLC 回路	メモリ不足で波形類似度が出ない						
	増幅回路	―	○	○	―	―	―	―
伝熱	ヒーター	―	○	―	―	―	―	―
複合	回転型電動機	メモリ不足で波形類似度が出ない						
	直動型電動機	メモリ不足で波形類似度が出ない						
マルチボディ機械	垂直駆動アーム	○	○	―	―	○	―	―

RLC 回路，回転型電動機，直動型電動機の計算結果は出るが，メモリ不足で波形類似度が出ない結果になった．今回取り上げたサンプルモデルでは，メモリ不足になったものを除いて解が得られたと判断する．

しかし，出力信号ごとに解ける処理系の積集合を取った結果が空集合になる可能性は残る．この場合は，どの処理系を採用してよいか明確に判断できない．対策として，処理系の種類を増やすこと，DTW とは異なる波形類似度の求め方(Derivative DTW[6]や k-shape[7])も併用することを検討する．表 3 でもいずれかのグループに処理系が密集しやすい結果になったので，この傾向が強まることを期待する．

3.3 作業定型化の度合いの評価

3.2 に示した計算結果の妥当性の評価作業は，2.4 に示したシミュレーション手順に沿って実施した．この評価作業を通じて発見した作業定型化ポイントを以下に示す．なお，これらは経験的に得られたものであり，論理的・網羅的なものではないことに注意されたい．

制御対象の非因果モデル作成

- 制御対象のモデル化に工数をかけないようにするため，MSL のような非因果モデルライブラリを活用するのがよい．

インタフェースを因果接続

- インタフェースを正しく因果接続するには，理論的な根拠だけでなく経験やノウハウも必要である[8].
- 処理系によっては因果接続アダプタと呼ばれるモデルライブラリを提供しているので活用するのがよい．

モデル分解と FMU 作成

- FMU を人手で作成するのは難易度が高くて工数がかかる．処理系が FMU 自動生成機能を提供するなら活用するのがよい．

システムシミュレータ組立て

- FMU 間の結線は自動化が難しく人手作業が必要である．結線を容易化するための GUI やコマンドを提供する処理系を用いるのが望ましい．

シミュレータコード生成

- 本稿で候補に挙げた処理系は，シミュレーションを実行するためにシミュレータコードを作成しなければならないものが多い．そのため，図 6 に示した構造図から情報を抽出してシミュレータコードを自動生成するようにした．
- さらに自動化を進めて，処理系に図 6 の構造図自体を与えるだけで直接実行できるコマンドを提供することが望ましい．

以上より，既存ツールが提供するライブラリや自動化機能を活用したり，既存ツールのサポートが不十分な部分は内製ツールで補完したりすることで，新たなロスコストを抑制できると判断する．

執筆時点で処理系ごとの作業定型化ポイントのサポート有無を表 6 に示す．

表 6　作業定型化ポイントのサポート有無

作業定型化ポイント	Cosim	DACCOSIM	FMPy	MasterSim	OpenModelica	PyFMI	Simulink
非因果モデルライブラリ	―	―	―	―	○	―	○
因果化アダプタ	―	―	―	―	○	―	○
FMU を自動生成	―	―	―	―	○	―	○
FMU を結線	○	○	○	○	○	○	○
コマンドで構造図を実行	○	―	○	―	―	―	―

4. おわりに

本稿では標準技術 FMI/SSP を活用したシステムシミュレーション手法を提案し，実現性を確認した結果を報告した．

計算結果の妥当性の評価では，本稿で候補に挙げた処理系を用いて解が得られたと判断した．作業定型化の度合いの評価では，新たなロスコストを抑制できると判断した．

上記の評価結果から以下を今後の取り組みとして定める．

- 処理系の種類を今後増やしていくときに，手際よく実施できるように内製ツールで作業を自動化し，人は計算結果の確認に注力できるようにする．
- 波形類似度を求めるために用いる時系列クラスタリング手法として，今回取り上げなかった Derivative DTW や k-shape 他を評価して，解の妥当性を上げる．
- より現実的な題材として，4 種の倒立振子(台車型，回転型，アーム型，車輪型)を取り上げる．

参考文献

[1] “Functional Mock-up Interface,” https://fmi-standard.org/
[2] “System Structure and Parameterization,” https://ssp-standard.org/
[3] 自動車技術会, *FMI 活用ガイド Ver.1.0.1*(2018).
[4] Open Simulation Platform, https://open-simulation-platform.github.io
[5] 南 裕樹, *Python による制御工学入門*, オーム社, 2019.
[6] https://data-analysis-stats.jp/機械学習/dtwdynamic-time-warping 動的時間伸縮法/
[7] https://zenn.dev/shungo_a/articles/ffbdb3614867ca
[8] 自動車技術会, *非因果モデリングツールを用いた FMI モデル接続ガイドライン Ver.1.0*(2015).