特集：モデルベース開発

10

SKYACTIV の MBD 検証環境について

MBD Verification Environment for SKYACTIV

臼田 浩平*1 Kohei Usuda
小森 賢*2 Satoshi Komori
三吉 拓郎*3 Takuro Miyoshi
寺岡 陽一*4 Yoichi Teraoka
本城 創*5 So Honjo
久禮 晋一*6 Shinichi Kure

要約

環境に対する意識が高まる中，マツダでは技術開発の長期ビジョンである「サステイナブル“Zoom-Zoom”宣言」を策定し「走る歓び」と「優れた環境・安全性能」の両立を目指しパワートレインを進化させてきたが，これに伴ってエンジン制御は，大規模かつ複雑化してきた。

このような環境の中でも，走行，環境性能向上の要求をタイムリーに実現していくためには，制御の品質を確保しながら短期間で開発を完了しなければならない。そこで，SKYACTIV-G の制御開発では，「モデルベース開発」（MBD：Model Based Development）を全面的に適用し，各開発段階で必要な机上検証システムを構築し効率化を進めてきた。

更に，SKYACTIV-G の商品化以降も搭載車種をすばやく拡大するために，机上検証領域の拡大や，机上検証システムの自動化と検証速度の改善を継続的に進めて，制御の品質確保と短期開発の両立化を図った。

Summary

In response to the growing environmental awareness, MAZDA declared "Sustainable Zoom-Zoom", a long-term corporate vision for technology development, and has evolved a powertrain aiming to achieve both "driving pleasure" and "excellent environment and safety performance." Along with it, engine control has become larger and more complicated.

To improve drivability and environmental performance in time for the requirement under such conditions, it is necessary to develop the engine control of the highest-possible quality in a short time. To heighten the development efficiency of the SKYACTIV-G control, Model-based Development (MBD) was applied overall and a theoretical verification system was established for each development stage.

In order to quickly deploy the system to other models after the SKYACTIV-G launch, expansion of the theoretical verification scope, automated theoretical verification system, and shorter verification speed have been addressed to achieve both high quality and faster development speed.

1. はじめに

環境や燃費に対する意識が高まる中，マツダでは技術開発の長期ビジョンとして「サステイナブル“Zoom-Zoom”宣言」を策定し，「走る歓び」と「優れた環境・安全性能」の両立を目指して SKYACTIV-G を開発してきた。そして，2011 年にデミオへ初めて搭載し(1)，その後も各車種へ搭載を拡大してきた。

SKYACTIV-G の開発では，パワートレインの進化に伴って大規模，複雑化する制御に対して，MBD を全面的に適用することで，制御の品質確保と短期開発の

*1〜6　パワートレインシステム開発部
Powertrain System Development Dept.

両立を目指してきた。

本稿では，制御の品質確保と短期開発を実現するために開発し，活用してきた制御の品質確認手段（机上検証システム）について紹介する。

2. MBD と机上検証の実現

2.1　開発／検証プロセスと MBD の役割

自動車の制御開発で一般的に用いられている，V 字で表現した開発プロセスを Fig. 1 に示す。V 字の右側は，実機や実際の車両を使った開発段階（検証段階）であり，ここで不具合が発覚した場合には大きな手戻りにつながってしまう。手戻りを最少化するには，各開発段階で細かく検証サイクルを回して，不具合の後工程流出を防止する必要がある。SKYACTIV-G の制御開発では，MBD を適用することにより各段階での検証を可能とした。

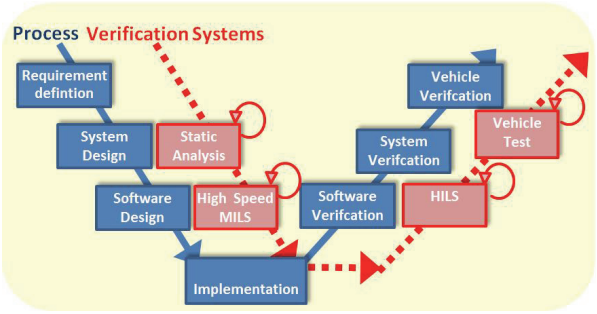


Fig. 1　MBD and Verification System in V-Process

2.2　MBD と机上検証システム

MBD 適用に際しては，各制御モデルと制御対象モデル，そして，机上検証システムを 2 段階に分けて開発した。

各々の定義は以下の通り。

① 制御モデル：エンジン制御の仕様を MATLAB/Simulink などのモデリングツールで表現したもの。

② 制御対象モデル：エンジンや車両などの動きをモデリングツールやプログラミング言語で表現したもの。

③ 机上検証システム：制御の品質を確認するシステムであり，制御単体で検証するシステムや制御モデルと制御対象モデルを結合して動作検証するシステム。

第 1 段階では，制御モデルの構成を定め，制御対象モデルを開発した（3 章）。机上検証システムは，左バンク（Fig. 1）の制御モデル開発（ソフトウェア設計）段階で，静的解析システム及び，制御モデルと制御対象モデルをモデリングツール上で接続した MILS（Model In the Loop Simulation）環境を整備した。

また，右バンク（Fig. 1）では制御ソフトウェアの検証段階で HILS（Hardware In the Loop Simulation）環境を構築した。更に，検証結果の管理については構成管理システムを活用して，モデル開発履歴と検証結果の可視化を実現した。

第 2 段階では，机上検証システムの自動化と検証速度の改善を進めてきた。具体的には，左バンクの静的解析システムの検証時間を改善し（4.1 節），MILS 環境においては，新たに開発した「高速 MILS 環境」へ置き換えることで，短時間で網羅的な検証ができる仕組みを構築した（4.2 節）。また，高速 MILS 環境については，大きく分けて下記の 3 つの機能を備えた検証環境として整備した。

① C 言語化及び並列演算による検証速度の向上機能。

② 自動車の操作系統を再現した GUI（Graphical User Interface）及び自動走行機能。

③ 車両系制御，TCM（Transmission Control Module），DSC（Dynamic Stability Control），MRCC（Mazda Radar Cruise Control）などと制御モデル間の多重通信や協調制御を確認できる機能。

以降では，制御と制御対象モデルの概要及び，机上検証システムの自動化と検証速度の改善について示す。

3. 制御モデルと制御対象モデル

まず MBD の中核を成すモデルについて簡単に説明する。

3.1　制御モデル[1]

SKYACTIV-G の制御は，複数のセンサ，アクチュエータ信号を入出力に持ち，また TCM や DSC などの複数の車両系制御と多重通信処理を行っている。この多くの外部信号と整合性を取りながら，複数の制御機能を効率的に開発するために SKYACTIV-G の制御モデルは，エンジンや車両の挙動を制御するアプリケーション層と外部信号を物理変換するプラットフォーム層に分ける構造を選択した（Fig. 2）。

また，複数の機能を効率的に開発するため，機能ごと，更に時間や角度同期単位にモジュールを分ける構造を採用して，複数の技術者による並行開発を実現した。

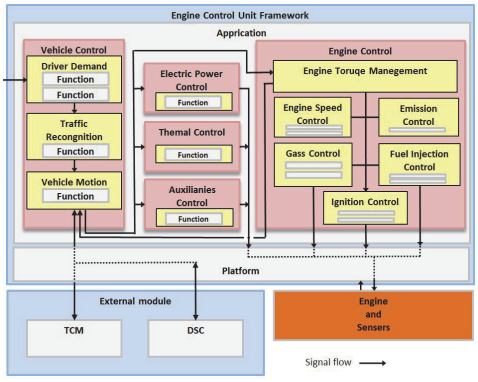


Fig. 2　Framework of Engine Control

3.2　制御対象モデル

制御モデルの動作を確認するために，制御対象となるモデルを机上検証システム（高速 MILS）上で制御モデルと結合している。具体的には，エンジンや各種センサやアクチュエータ，補機とトランスミッション，空気と路面など車両抵抗をモデル化したものである（Fig. 3）。このプラントモデル群は HILS と共通化することで，V 字の左右両バンクにおける検証の等価性を確保している。その中で，エンジンモデルは精度と速度が両立できる統計モデルを採用し[2]，オートマチックトランスミッションについては，プラネタリ構造から油圧システムまでを詳細に解いた物理モデルと演算速度の速い簡易モデルを採用することで，検証要求に応じてモデルを切り替え，検証時間を改善した。

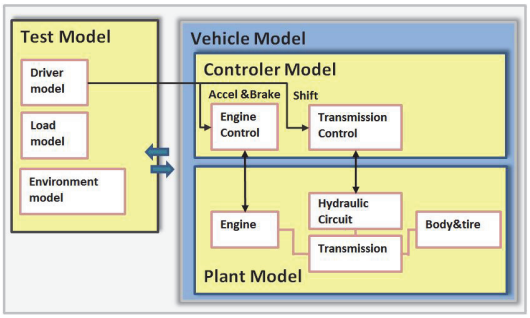


Fig. 3　Framework of Vehicle Model

4. 机上検証システムの改善

4.1　制御モデルの静的解析システム

静的解析とは，制御モデル（ソースコード）を実行せずに，モデリングルールの違反か所や欠陥を検出する検証である。具体的には，モデルの記述ミスによって生じる，ゼロを取り得る変数による除算やオーバフローといった，欠陥を検出する。欠陥を含んだまま,制御が車両に搭載されれば不意に制御プログラムは動作を止めてしまう恐れがある。これが車両走行中に起これば重大な結果を招きかねない。そのため,この検証は確実に実施する必要がある。

また，この検証システムでは，欠陥以外にも検証対象の入力値（組み合わせ）に対して検証対象から出力される信号が，仕様の範囲内に収まることを確認している。

更に，この検証には以下に示すメリットがある。

① 手戻りが小さい。V 字左バンクのモデル開発段階（設計初期）で検証可能なため，欠陥を発見した場合でも，即座にモデルや仕様に修正を反映できる。

② 網羅性が高い。動的検証では動作した部分しか検証できないが，実行可能なパス（分岐）は全て検証可能。

③ テストケースの準備が不要。

④ 実行可能なパスの可視化。

しかし，SKYACTIV-G の制御モデルは，その要求と仕様の情報量から，モデルの大規模化と複雑化が生じるため，制御モデルを全て検証することに，数千という時間が必要となる。このままでは，現実的に部分的な検証しかできない。また，一般的にモデルの大規模化と複雑化は，検証の網羅性も低下させてしまう。

そこで，品質確保と短期開発を両立させるために，制御モデルの構造を活用しながら，モデルの検証単位を最適化し，関数（時間や角度同期）単位まで細分化して，1 回の検証における複雑さを低減させた。更に，変更点を中心に影響範囲を自動抽出するシステムを開発することで，検証範囲を限定し検証時間を改善した（Fig. 4）。

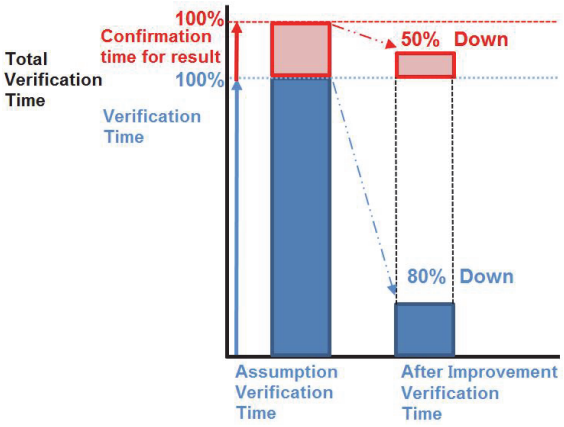


Fig. 4　Verification Time Reduction Effect by Automated System

また，膨大な検証結果の解析については，自動レポート機能（Table 1）を開発することで，PASS/FAIL を選別し，詳細を確認すべき項目を絞り込み，確認工数を削減している（Fig. 4）。

Table 1　Static Verification Report

No.	Spec Name	Function Name	Engineer Name	Date and time	Comprehensive Judgment	Coverage(%)	Coverage Judgment	Description Defect	Output Defect
1	Application1	sample1	–	–	PASS	100	S	PASS	PASS
2		sample2	–	–	PASS	100	S	PASS	PASS
3		sample3	–	–	FAIL	76.56	B	PASS	FAIL
4	Application2	sample4	–	–	PASS	100	S	PASS	PASS
5		sample5	–	–	FAIL	100	S	FAIL	PASS
6	Application3	sample6	–	–	FAIL	—	D	FAIL	PASS
7		sample7	–	–	PASS	100	S	PASS	PASS
8	Platform(Input)1	sample8	–	–	PASS	100	S	PASS	PASS
9		sample9	–	–	PASS	100	S	PASS	PASS
10	Platform(Output)1	sample10	–	–	PASS	100	S	PASS	PASS
11		sample11	–	–	PASS	100	S	PASS	PASS
12		sample12	–	–	FAIL	–	–	FAIL	FAIL

上述，2 点の改良を加えた静的解析システムの全体像を Fig. 5 に示す。設計者が，制御モデルの変更を構成管理システムへ登録すると自動的に変更部分を検証し，レポートを出力する仕組みを構築した。これにより，静的解析は，検証作業を完全に自動化し，レポートから限定された範囲を確認する仕組みとすることで短時間でも品質の確保が可能となった。

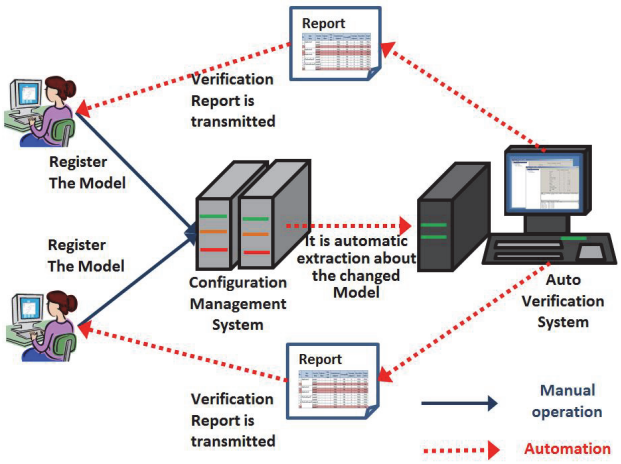


Fig. 5　Automatic Static Verification System

4.2　高速 MILS 環境

(1) 高速 MILS 環境の概要

高速 MILS 環境とは，制御モデルと MATLAB/Simulink を含む他のモデリングツールで作成した制御対象モデル全てを C 言語化し，これを実行体となる DLL（Dynamic Link Library）へ変換した，「制御モデルの動作を確認する」ための，机上検証システムである。

MATLAB/Simulink や他のモデリングツールは，モデルを開発しやすい反面，C 言語などのプログラミング言語に対して実行速度が遅くなる。これはモデリングツール上で作成，完結された「MILS 環境」も同様であり，大規模で複雑なモデルになれば，より顕著に現れてくる。そこで，この問題を解決するために，計算速度の改善が期待できる C 言語化を選択した（Fig. 6）。

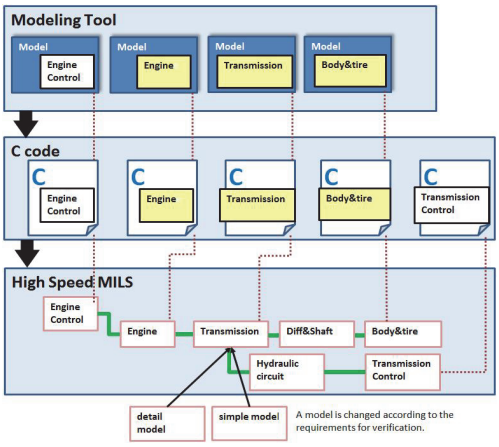


Fig. 6　Framework of High Speed MILS

高速 MILS 環境の構成は，Fig. 6 に示すように，エンジンやトランスミッションなどの制御対象モデルをユニット単位ごとに C 言語化後，それぞれ別個体の DLL ファイルとしている。そして，この DLL ファイルを切り替えることで，検証の要求や詳細度に応じて簡単に制御対象モデルを入れ替え可能とし，効率的な検証を実現した。

なお，C 言語から高速 MILS に組み込む作業も全て自動化を図っている。

(2) 高速 MILS 環境の GUI

高速 MILS-GUI にはリアルタイムのグラフ表示と数値表示機能を備え（Fig. 7），手動操作用のステアリングスイッチ各種とシフレバー,ブレーキ,アクセルなどを備えている（Fig. 8）。

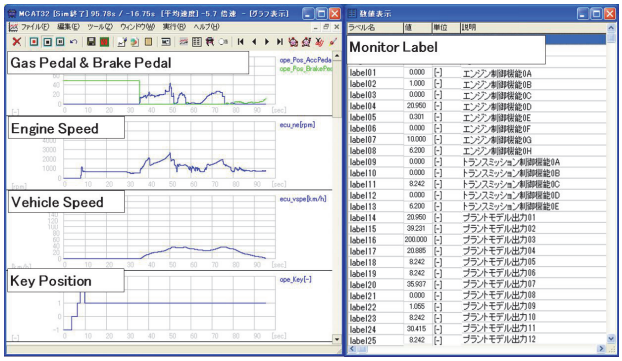


Fig. 7　Graph and Numeric Display of High Speed MILS

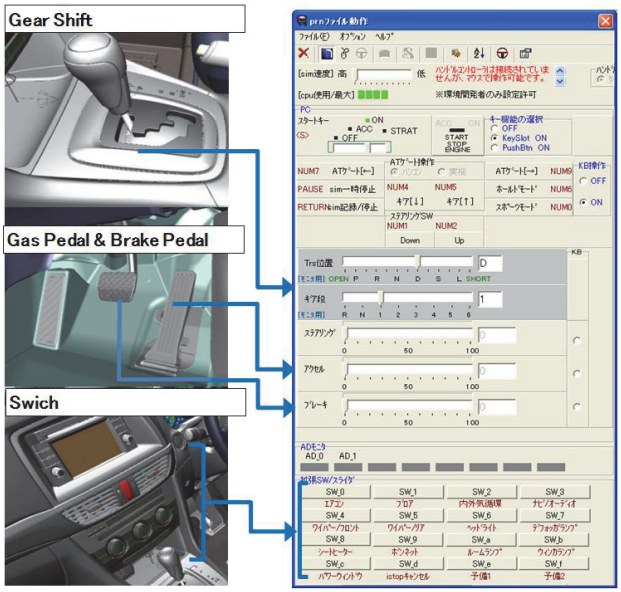


Fig. 8　Driver Operation Panel of High Speed MILS

(3) 制御モデルの組み込み

制御モデルの C 言語化は，MATLAB/ Simulink のオートコード機能を活用している。ただし，制御モデルは全ての機能を含めると規模が非常に大きく，C 言語化するだけで膨大な時間が必要となる。そのため，複数の技術者によって効率良く並行開発を進めるには，モデル編集から高速 MILS に組み込むまでの作業時間を改善する必要がある。

そこで制御モデルに限っては，制御モデルの構造（Fig. 2）を活用することで，一定単位のユニット（機

能）を切り出し，一部のみ C 言語化する技術を開発して C 言語生成時間を改善した。また，先に記した自動組み込み機能を組み合わせることで，簡単かつ高速にモデリングツールから高速 MILS 環境へのモデル反映が可能となった（Fig 9）。

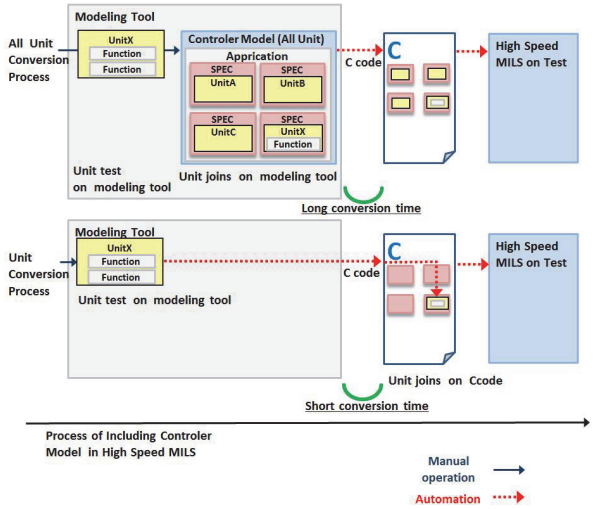


Fig. 9　Controller Model C-Conversion Time

(4) 並列計算による高速化

高速 MILS 環境には，SKYACTIV-G 制御モデルとこれに接続される他の制御や制御対象モデルが複数搭載されている。そのため，C 言語化による高速化を実現しても，接続するモデル数の増加やそれぞれの詳細度が増せば，必然と検証速度は低下していく。そこで，これら問題の影響を低減し，更に検証速度を改善した効率的なシステムとするため，新たに並列計算技術を開発した。並列化による検証速度の改善効果を Fig. 10 に示す。

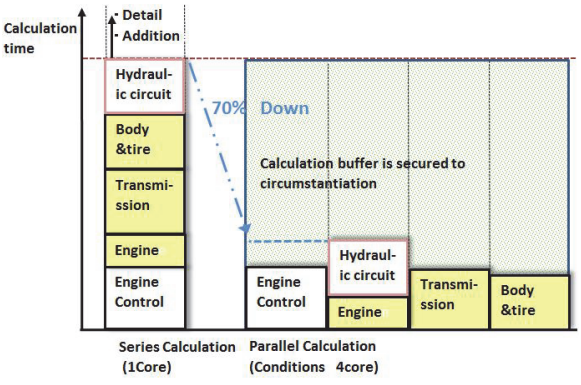


Fig. 10　Improvement Effect by Parallel Calculation System

並列演算の構成は，高速 MILS 環境の構成（Fig. 6）を活用し，各モデル（DLL）単位で計算コアを分散する構成としている。そして，実行スケジュールは，入力→実行→出力の 3 タスクに分けることで，全てのモデルが前回の計算結果を使って計算する仕組みとした。入力タスクで信号バスから各モデルの計算値を取得し，次に実行タスクでこの入力値の結果を使って演算を行う。そして，出力部分で計算結果を信号バスに返す仕様としている（Fig. 11）。入力→実行→出力の実行スケジュールは，単一の計算コアしか持たない計算機器においても有効となるため，直列実行と並列実行の間で，計算結果の等価性を保つことができる。言いかえれば，ユーザの利用環境によって検証結果が異なることを防止している。

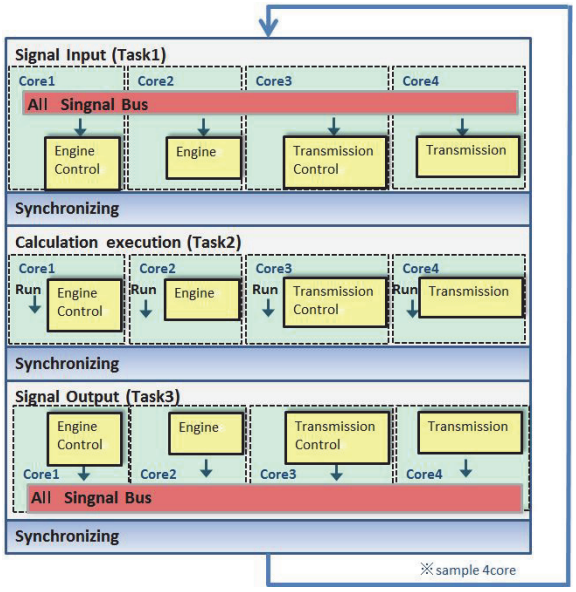


Fig. 11　Framework of Parallel Computing

(5) 自動レポートシステム

制御モデルは複数の技術者によって並行開発されているため，各技術者が全体の影響を全て把握することは困難である。そこで，基本的な走行条件については，高速 MILS 環境での実行結果に対する，自動レポート機能を構築することで PASS/FAIL を自動判定して品質確認をサポートしている。

(6) 高速 MILS 環境の機能項目

高速 MILS 環境では，机上での検証項目拡大を図るために以下に示す検証機能を開発した。

a.全パラメータのキャリブレーション機能

b.自動走行機能

c.各種センサやスイッチをフェールさせる機能　など

5. 机上検証システムの適用事例

5.1　高速 MILS による検証

高速 MILS は，SKYACTIV-G の制御モデルを中心に，幅広い制御開発領域に適用を図ってきた。具体的には，「駆動力制御」，「DSC とのブレーキ協調制御」，「i-ELOOP（減速エネルギー回生システム）制御」，「クルーズコントロール制御」等である。実際の適用事例を 2 つ紹介する。

（事例 1）　駆動力制御

走行性能と燃費のために常に最適な駆動力を発生させるには，パワートレイン全体を制御する必要がある。そのため，エンジン制御と TCM 間で通信を行い，発生トルクとギヤ比を常に最適な状態に制御しなければならない。そこで，高速 MILS に搭載されたエンジン制御と TCM の通信機能を活用し，お互いの基本動作を検証した。更に実車よりも高速に検証できる機能を活用することで，机上でさまざまな走行パターンを再現し，駆動力制御の初期品質向上を図った。

（事例 2）　クルーズコントロール制御

高速 MILS 環境の GUI（Fig. 12）を操作することで，実車相当のクルーズコントロールスイッチ操作が再現可能となった。

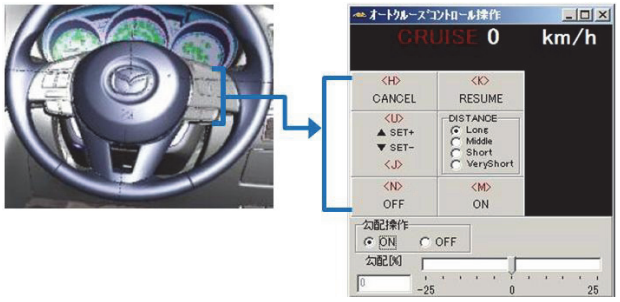


Fig. 12　High Speed MILS GUI in Auto Cruse Switch

実際にドライバ指示（GUI 入力）によって設定車速が操作できる。オートクルーズ ON 状態で，「SET＋」スイッチを押しこむとオートクルーズの設定速度が次第に上がっていく。そして，道路の勾配変化が発生しても，制御が働き，目標車速を維持していることが確認できる（Fig. 13）。また，さまざまなテストケースを検証することによって，机上でオートクルーズ制御の基本品質向上を図ることができた。

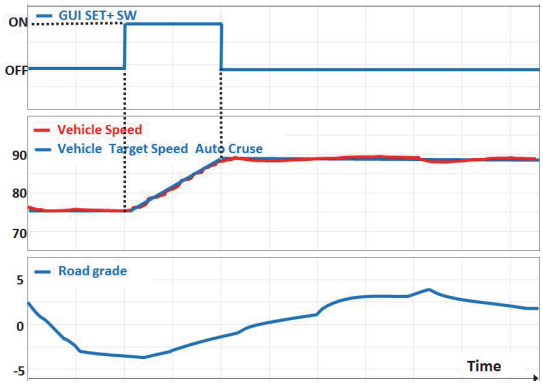


Fig. 13　Test Results of Auto Cruse Control

6. 机上検証システムの改善効果

V 字左バンクにおいて，静的解析システムの自動化を実現し，更に，高速 MILS 環境の実現によって短時間で制御モデルの動作検証が可能となった。また，検証事例のように机上検証領域を拡大させることで，制御開発において，品質確保と短期開発を両立することができた（Fig. 14）。

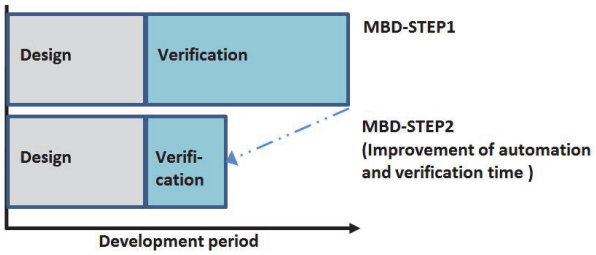


Fig. 14　Improvement of Verification Time

7. おわりに

以上のように，制御開発の V 字プロセスを手戻りなく進めるために，検証システムを充実させ，机上開発の領域を拡大してきた。

また，この机上検証システムを活用することで，制御開発 V 字プロセスの右バンクに移る前段階で，高品質の制御開発を実現できた。今後も「サステイナブル“Zoom-Zoom”」を実現していくパワートレインの進化と技術開発において，MBD を更に発展させ，制御の品質確保と短期開発を両立しながら更なる効率化を図っていく。

参考文献

(1)　江角ほか:SKYACTIV-G 制御技術の紹介，マツダ技報，No.29　pp.36-40（2011）

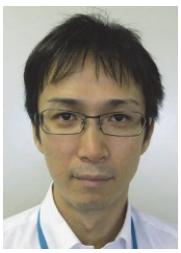
(2)　寺岡ほか:エンジンの制御系仮想開発環境と新型エンジン開発への適用，自技会論文集 No.20124464 Vol.43（2012）

■著　者■


臼田 浩平


小森 賢


三吉 拓郎


寺岡 陽一


本城 創


久禮 晋一