
ECU Software Abnormal Behavior Detection Based On Mahalanobis-Taguchi Technique

Yixin Chen and John Phillips
Delphi Powertrain Systems

ISBN 978-0-7680-1633-8



9 780768 016338

SAE *International*[™]

**2008 World Congress
Detroit, Michigan
April 14-17, 2008**

By mandate of the Engineering Meetings Board, this paper has been approved for SAE publication upon completion of a peer review process by a minimum of three (3) industry experts under the supervision of the session organizer.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of SAE.

For permission and licensing requests contact:

SAE Permissions
400 Commonwealth Drive
Warrendale, PA 15096-0001-USA
Email: permissions@sae.org
Tel: 724-772-4028
Fax: 724-776-3036



For multiple print copies contact:

SAE Customer Service
Tel: 877-606-7323 (inside USA and Canada)
Tel: 724-776-4970 (outside USA)
Fax: 724-776-0790
Email: CustomerService@sae.org

ISSN 0148-7191

Copyright © 2008 SAE International

Positions and opinions advanced in this paper are those of the author(s) and not necessarily those of SAE. The author is solely responsible for the content of the paper. A process is available by which discussions will be printed with the paper if it is published in SAE Transactions.

Persons wishing to submit papers to be considered for presentation or publication by SAE should send the manuscript or a 300 word abstract of a proposed manuscript to: Secretary, Engineering Meetings Board, SAE.

Printed in USA

ECU Software Abnormal Behavior Detection Based On Mahalanobis-Taguchi Technique

Yixin Chen and John Phillips
Delphi Powertrain Systems

Copyright © 2008 SAE International

ABSTRACT

To confirm the correct operation and detect the potential errors in the ever more complicated automotive ECU application software are very challenging. This paper presents a new approach to detect potential ECU application software abnormal behavior based on the Mahalanobis Distance, the Mahalanobis-Taguchi System, and vehicle driving data playback capability with a simulator. Vehicle driving data is recorded by instrumentation calibration tools and played back on the test bench to stimulate the ECU. In our study, the normal behavior is characterized by the Mahalanobis Distance (MD), which is calculated from the data set logged while playing back the recorded vehicle maneuvers while the ECU is flashed with "baseline software" that was believed to be error free. Then the MDs were calculated from a new data set logged while playing back the same vehicle maneuvers while the ECU was flashed with the new software. The large MDs resulting from the new software indicate times when the new software behaved abnormally. An MD from the data set that is above a threshold indicates a potential software error, which must be investigated in detail by an expert. Some encouraging experimental results using the method on Electronic Throttle Control (ETC) subsystem tests are presented and indicate the capability for this method to detect potential software errors.

INTRODUCTION

Automotive application software which operates in the Engine Control Module (ECU) is becoming more complicated, and it is challenging to confirm the correct operation and detect potential errors in the software. The current test method is to design many test cases based on the design specifications, which are sometimes called "positive tests" [1]. The other way to improve the software tests is so-called "negative tests" [1], which intentionally modify the inputs to non-normal conditions to induce software failures. This paper explores another novel way to detect ECU software abnormal behavior using the Mahalanobis Distance and the Mahalanobis-

Taguchi System. Basically, the Mahalanobis Distance (MD) [2] is a classic scalar measurement determined by matrix operations which are often used in designing pattern classifiers. The steps involved in the MTS are to first select a database of normal data which is normalized by subtracting the mean and dividing by the standard deviation of each column in the database. The normalized columns are then correlated, forming a correlation matrix. The correlation matrix is then inverted, which we will show is a numerically risky step. Next, a new database is collected from the software under test, which is normalized using the mean and standard deviation of the original data. The MD is calculated using the new normalized database and the original inverted correlation matrix. An MD which is above a threshold indicates that the new database is non-homogeneous with the original database and indicates a potential software error. The columns of the databases are manipulated to determine which columns are contributing to the high MD, and finally the data are displayed to an expert for final determination of the detection of a software error.

During the automotive ECU software development cycle, there are repeated changes to the software set due to many reasons, such as adding new functions, fixing software errors in previous releases, and making the code more compliant with industry standards. To thoroughly test each software build and release is very time consuming and sometimes not effective. The thorough functional tests are always conducted for major software releases, which give us the baseline software set for each specific subsystem. With the baseline software set, one can collect variable data in response to well defined vehicle drive cycles using instrumentation tools which are in common use within the automotive industry. The vehicle drive cycles can be duplicated in the test lab environment by using vehicle simulator playback capability. While playing back vehicle drive cycles with the ECU flashed with the baseline software set, the selected variables are recorded and used to calculate the normal space. Next the ECU is flashed with the software-under-test, and the same software variables

are collected while playing back the same vehicle drive cycles and are used to calculate the MD for comparison between the baseline software and the software-under-test. Post analysis focuses on finding the reason for the high MDs. In our experiment, the Electronic Throttle Control (ETC) subsystem was chosen as a case study.

This paper is organized as below. In Section 2, we will discuss more detail about the simulator playback capability, how this can be used to collect data for both baseline software and software-under-test. In Section 3, we will discuss the details of the Mahalanobis Distance and the Mahalanobis-Taguchi system (MTS) implementation and experimental results on ECU software abnormal behavior detections. Lastly, in Section 4, we will draw the conclusions and discuss the future work directions.

VEHICLE DRIVE CYCLE PLAYBACK MECHNIZATION

Fig. 1 shows the vehicle simulator which is used to duplicate vehicle drive cycles in our study. The vehicle simulator can provide the ECU with simulated vehicle

sensor input signals such as battery voltage, accelerator pedal position, engine crankshaft and camshaft position signals and so on. These signals can be generated by either writing scripts to drive the simulator output signals, or by playing back ECU input signals that have been previously recorded. The vehicle simulator is also able to capture the outputs from the ECU, such as fuel injector drive signal timing and duration, ignition coil drive signal timing and duration, cooling fan relay and air conditioner controls and so on. The simulator is also able to capture internal variables that are intermediate values used to interpret the system states, such as engine state, or used in preparation of the outputs such as desired engine speed or torque.

The selected ECU input, output, and internal variables are collected by recording them in engineering units during real vehicle driving using the instrumentation tool. Then the logged data is converted from engineering units into the raw input formats such as MAP sensor input voltage, coolant temperature sensor resistance and so on. Finally, the ECU raw inputs are played back from vehicle simulator. This input data conversion is shown in Fig. 2.

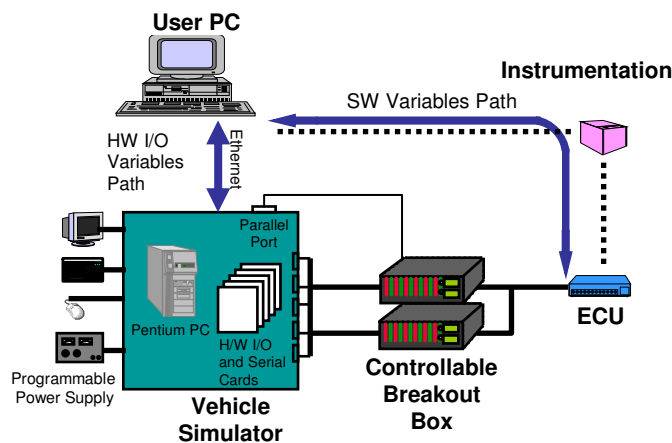


Figure 1: Vehicle simulator block diagram

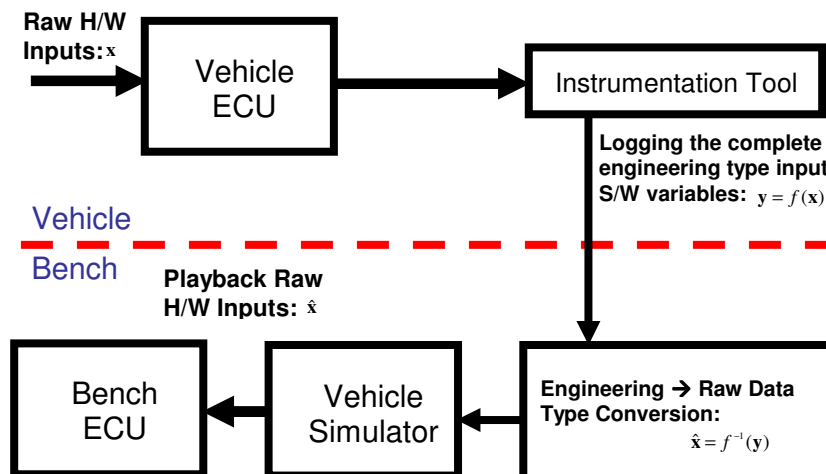


Figure 2: Vehicle data conversion for simulator playback

THE MAHALANOBIS-TAGUCHI SYSTEM IMPLEMENTATION FOR ECU SOFTWARE ABNORMAL BEHAVIOR DETECTION

MAHALANOBIS DISTANCE CALCULATION

FLOWCHART - Mahalanobis distance (MD) was introduced by P. C. Mahalanobis in 1936 [3]. It is based on correlations between variables by which patterns can be identified and analyzed. The Mahalanobis Distance has been applied in many situations, including the detection of medical conditions like liver disease, the identification of environmental driving conditions, the control of vehicle braking, active chassis control, and space flight. It is a useful way of determining similarity or difference of an unknown sample set to a known one.

Step 1 - Select the columns of the database

The first step in applying the Mahalanobis-Taguchi System is to select the measurements to be included in the database. The terms measurement and column are used interchangeably in this paper to reflect the structure of the database. The rows of the database represent samples collected at each time increment. The measurements should characterize the sub-system to be tested, but there are limitations that must be observed. Highly correlated measurements must be avoided, as the correlation matrix becomes singular in the presence of perfectly correlated measurements, and becomes numerically problematic as the correlation of two columns approaches unity. This is unfortunate because we are often interested in measurements that are highly correlated. For example, redundant inputs are often used in the electronic throttle control sub-system to insure that hardware failures can be detected and responded to in a safe manner, and a lack of correlation must be detected, but can not be included in the MTS. Variables that do not change during the course of the drive cycle, like brake status or ETC mode must also be avoided.

Step 2 - Normalize the columns

The next step in applying the MTS is to normalize the database. This is done by calculating the mean and standard deviation of each column in the database. This is the reason for the second restriction on the measurements that can be selected. Since the standard deviation appears in the denominator in the normalization step, it is required that every measurement selected have a non-zero standard deviation. The normalization is performed by subtracting the mean and dividing by the standard deviation of each column. The database is thus changed from engineering units to dimensionless units of standard deviation. The normalized data are referred to by the symbol Z .

Step 3 - Form the correlation matrix

The next step in applying the MTS is to form the correlation matrix. The correlation matrix is an m by m symmetric matrix, where m is the number of columns in

the database, and has ones along the diagonal. The values range from negative one to one and are arranged symmetrically in the matrix to represent the correlation between two columns of the normalized database. Notice that the correlation between two columns, say i and j , is the same as the correlation between columns j and i , which leads to the symmetry. The correlation between a column and itself is equal to one, which leads to the ones along the diagonal of the matrix. The correlation matrix is often represented by the symbol R .

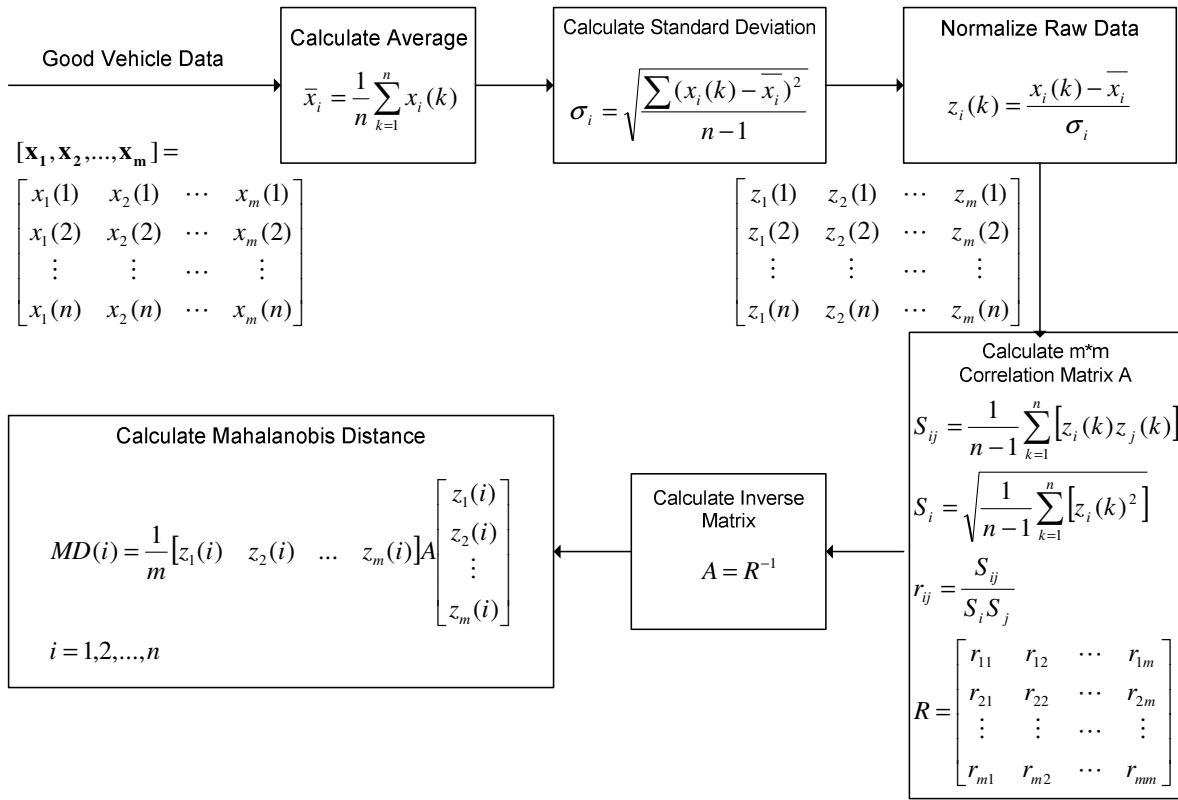
Step 4 - Invert the correlation Matrix

The next step in applying the MTS is to find the inverse of the correlation matrix. This can be a numerically difficult step. The problem is not in calculating the inverse, as many modern software tools are available that can handle the calculation, but is in possibility that the correlation may be singular or nearly singular such that the inverse either does not exist, or is numerically unstable. When applying MTS, one is advised to pay particular attention to this detail. The inverse correlation matrix is represented by the symbol A , and the elements of the matrix are denoted by a_{ij} .

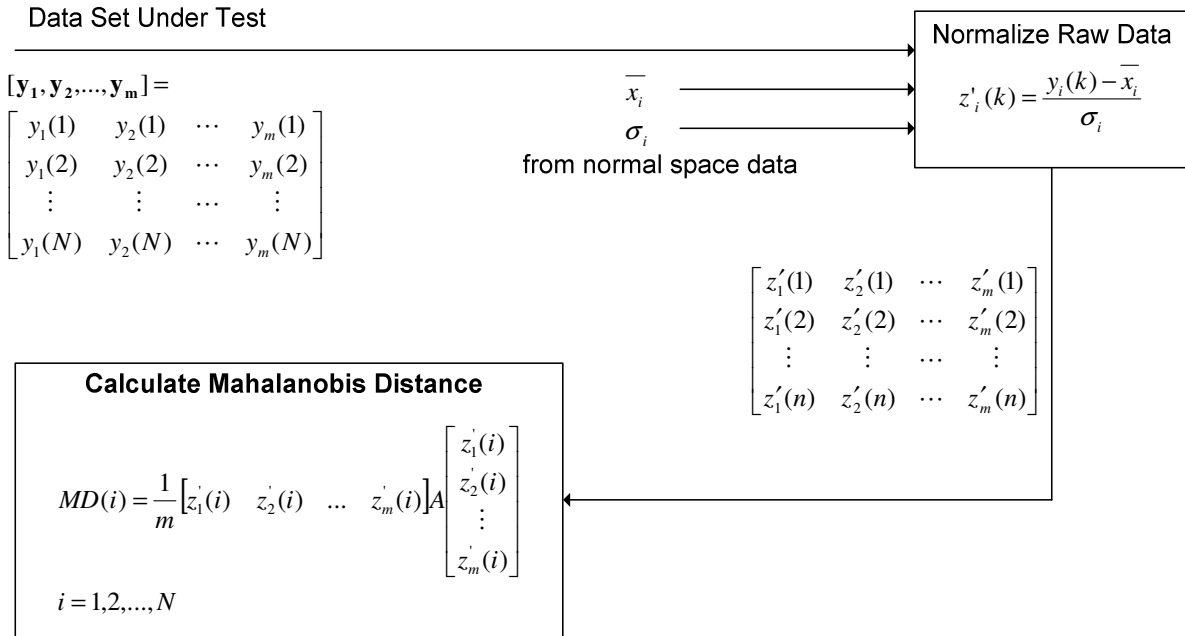
Step 5 - Calculate the Mahalanobis Distance

When calculating the Mahalanobis Distance, each row of the database is treated as a row vector. The row vector is right multiplied by the inverse correlation matrix. This multiplication results in a row vector which is then right multiplied by the transpose of the database row. This multiplication results in a scalar, which is divided by m , the number of columns of the database, to form the Mahalanobis Distance which is used to characterize the row of the database. This process is repeated for each row of the database. This scalar distance is approximately Chi squared distributed with m degrees of freedom. The reader is alerted that Woodall et. al. are critical of Taguchi for not treating the Mahalanobis Distance statistically [4], among other things. The authors follow Taguchi in treating the Mahalanobis Distance practically by selecting a threshold above which the software behavior is investigated by an expert without regard to the statistical significance of the threshold chosen.

These five steps are shown graphically being applied to the baseline database in Fig. 3(a), and being applied to the software-under-test in Fig. 3(b). Notice that once the values used for normalization, the column mean and standard deviation, and the correlation matrix and the inverse correlation matrix are not recalculated for the software-under-test. In this way the normal space is determined by the baseline software behavior, and is used to determine whether the software-under-test is within the normal space for every time step.



(a) Normal MD calculations based on baseline data set



(b) MD calculations for dataset-under-test

Figure 3

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	time	EAPS1RAW	EAPS2RAW	VBRKENG	EBRKLAMI	VMAFFILT	VMAFMETI	VKPH	ZANRAW	ETCMODE	VTHROF	EPEDLIND	EPEDPOS	ETPS1RAW	ETPS2RAW	ETPSIND	ETPSDES	
36359	283.598	12.69531	7.03125	1	1	43.40625	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36360	283.5938	12.69531	6.933594	1	1	43.53125	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36361	283.6016	12.69531	6.933594	1	1	43.77734	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36362	283.6094	12.69531	6.933594	1	1	43.90234	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36363	283.6172	12.69531	6.933594	1	1	44.02734	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36364	283.625	12.69531	6.933594	1	1	44.27344	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36365	283.6328	12.69531	6.933594	1	1	44.39453	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36366	283.6406	12.69531	6.933594	1	1	44.51953	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36367	283.6484	12.69531	6.933594	1	1	44.76563	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36368	283.6562	12.69531	6.933594	1	1	45.01173	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36369	283.664	12.69531	6.933594	1	1	45.13673	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36370	283.6718	12.69531	6.933594	1	1	45.38283	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36371	283.6796	12.69531	6.933594	1	1	45.50783	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36372	283.6874	12.69531	6.933594	1	1	45.75393	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36373	283.6952	12.69531	6.933594	1	1	45.87893	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36374	283.703	12.69531	6.933594	1	1	46.00393	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36375	283.7108	12.69531	6.933594	1	1	46.25	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36376	283.7186	12.69531	6.933594	1	1	46.375	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36377	283.7264	12.69531	6.933594	1	1	46.5	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36378	283.7342	12.69531	6.933594	1	1	46.62109	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36379	283.742	12.69531	6.933594	1	1	46.87109	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36380	283.7498	12.69531	6.933594	1	1	46.99219	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36381	283.7576	12.69531	6.933594	1	1	47.24219	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36382	283.7654	12.69531	6.933594	1	1	47.24219	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36383	283.7732	12.69531	6.933594	1	1	47.48828	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36384	283.781	12.69531	6.933594	1	1	47.73438	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36385	283.7888	12.69531	6.933594	1	1	47.98048	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36386	283.7966	12.69531	6.933594	1	1	48.10548	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36387	283.8044	12.69531	6.933594	1	1	48.23047	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36388	283.8122	12.69531	6.933594	1	1	48.47657	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36389	283.82	12.69531	6.933594	1	1	48.60157	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36390	283.8278	12.69531	6.933594	1	1	48.72657	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36391	283.8356	12.69531	6.933594	1	1	48.97267	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36392	283.8434	12.69531	7.03125	1	1	49.09767	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36393	283.8512	12.69531	7.03125	1	1	49.21875	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36394	283.859	12.69531	6.933594	1	1	49.46875	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36395	283.8668	12.69531	6.933594	1	1	49.59375	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36396	283.8746	12.69531	7.03125	1	1	49.71483	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36397	283.8824	12.69531	7.03125	1	1	49.96093	1.757813	0	17.3828	5	0	0.558461	0	10.22949	89.81934	1	1	
36398																		

Figure 4: A typical ECU software variable data set logged during vehicle driving

MEASUREMENT DATA SELECTIONS FOR MTS BASED ECU SOFTWARE TEST - Fig. 4 shows a typical ECU software variable data logged using vehicle calibration tool. The first column is "time stamp" and other columns are measurements for different software variables. There are totally 36397 sets of data in Fig. 4 which was recorded during about 284 second vehicle driving. Typically, engineers will plot data and then analyze manually if everything seems OK. The analysis is time consuming and the analyzing conclusion will depend on the engineers' experience a lot. Obviously, MTS technique should be useful to play a role to analyze the data homogeneity based on MDs in every time steps.

Another issue is what measurements to select to calculate MDs. The basic rules to select the measurement variables are: 1) to select the variables which can really evaluate the behavior of the subsystem-under-test; 2) to select variables which are not highly correlated. For the first rule, it usually needs a good high level understanding to the subsystem and may need to consult the subsystem expert. For the second rule, people should avoid to use the redundant software variables or two variables which have a sole linear relation. It's important to avoid choosing the variables which are highly correlated, or otherwise it will result in an ill-conditioned (or singular in worst case) matrix $\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_m]$, which can cause problem while getting correlation matrix \mathbf{A} by finding the inverse matrix of \mathbf{R} .

VEHICLE DRIVE CYCLE CONSIDERATIONS FOR NORMAL MD TRAININGS - As the first step, the normal MDs must be trained based on the data logged during the well designed vehicle drive cycles. In our experiments, ETC subsystem was chosen as the case

study. It should be pointed out that ETC subsystem behaves very different for highway constant speed cruising and highly acceleration maneuvers. The normal behavior in highway cruising will appear abnormal in highly acceleration maneuvers, and vice versa. If a data set includes both data logged during cruise and highly acceleration maneuvers, we found out that there will be many high MD points which are actually normal ECU behavior. For example, Fig. 5 shows an MD calculation results based on a data sequence logged during mixed vehicle maneuvers. There are so many high MD points and therefore it can not be used as normal MD space for classification purpose.

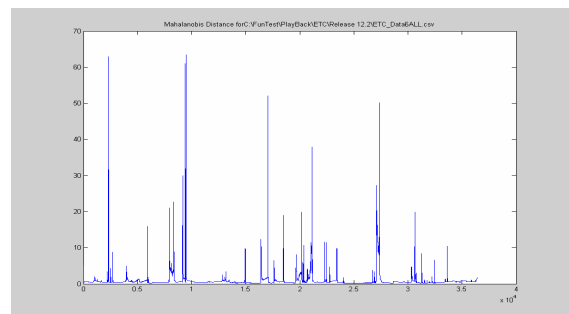


Figure 5: MD calculation results based on ETC data logged during mixed vehicle drive maneuvers

With the curiosity, we looked the detail on the data around the high MD points in Fig. 5. The detail analysis shows most high MDs are caused by ETC subsystem response to highly acceleration command from driver. But we did find one high MD shown as Fig. 6 which was caused by data error created by internal data conversion software. In Fig. 6, we expect EPEDPOSN drops after

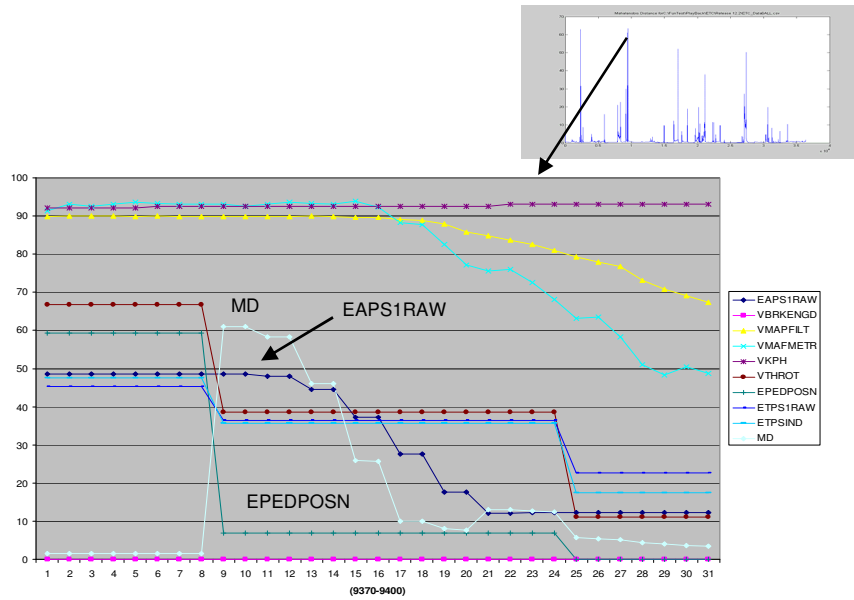


Figure 6: A data error caused by data conversion tool shows high MDs

EAPS1RAW drops because EAPS1RAW is the input for calculating EPEDPOSN. Due to the error caused by an internal data conversion software, EPEDPOSN drops before EAPS1RAW starts to drop, and therefore results in a much higher MD region. This does convince us the capability for MTS technique to detect potential software errors.

To avoid the high MDs like ones in Fig. 5, we have to carefully design our drive cycles. In general, different data set should be logged for different drive cycles and don't log the data set with mixed drive cycles. Figure 7 shows an MD sequence plot while data was logged in highway cruise.

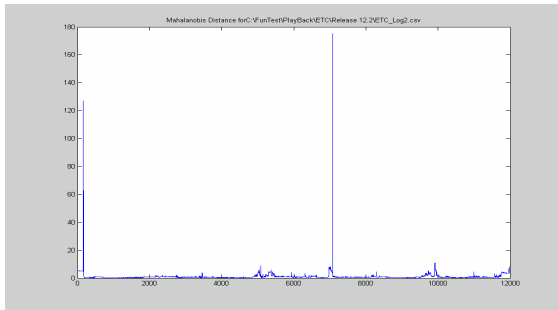


Figure 7: Normal space MDs for vehicle cruise driving

There are still two regions showing high MDs in Fig. 7. The analysis shows the 1st high MD was caused by an improper initialization on VMAFMETR (mass air flow measurement variable), which was indeed an unknown software error to us. This error shows up in the assumed "good" software set and has caused the high MDs. We can either manually trim the data to correct the VMAFMETR initialization or we can simply leave it and expect a high MD region will occur during initialization phase. The 2nd high MD region was caused by the very

minor difference in two highly correlated variables and this convinces us we should avoid choosing the highly correlated variables while doing MTS based data analysis.

SOME MORE EXPERIMENTAL RESULTS - To further test the capability of the proposed MTS based software test technique, we have manually induced 20 different software errors and generated 20 different software builds. The average MD for "good" software set is 0.9993 and 1.5 is chosen as a threshold for classifications. Table 1 shows the software classification results.

In Table 1, the software errors induced for the first three software sets don't affect the ETC sub-system behavior, and therefore those three software builds are still classified as "Normal". The errors induced for other 17 software builds do impair ETC sub-system behavior and all those software builds are classified as "Abnormal" by using a fixed MD threshold 1.5.

Table 1: Classification Results for 20 Software Builds with Artificially Induced Errors

S/W Build	Average MD	Classification Results
"good" S/W	0.9993	n.a.
1	1.47	Normal
2	1.06	Normal
3	1.07	Normal
4	10362	Abnormal
5	10029	Abnormal
6	10039	Abnormal
7	10013	Abnormal
8	2.9286	Abnormal
9	3	Abnormal
10	165	Abnormal
11	20.4	Abnormal
12	26.3	Abnormal

13	24.12	Abnormal
14	29.41	Abnormal
15	25.6	Abnormal
16	57224	Abnormal
17	23.72	Abnormal
18	34.24	Abnormal
19	19.29	Abnormal
20	30.42	Abnormal

CONTACT

Yixin Chen, Senior Electronics Systems Engineer
 Engine Management Systems Group
 Delphi Powertrain Systems
 yixin.chen@delphi.com

John Phillips, Senior Staff Research Engineer
 Engine Management Systems Group
 Delphi Powertrain Systems
 john.d.phillips@delphi.com

CONCLUSION AND FUTURE WORK

We have proposed a novel scheme to detect potential ECU software abnormal behaviors based on Mahalanobis-Taguchi technique. With the vehicle simulator playback capability, we have been able to implement a MTS based software testing system for ECU software testing purpose. The preliminary experimental results indicate the capability for MTS based technique to detect the potential ECU software errors. Specifically, ETC sub-system in automotive application was chosen as case study. Nevertheless, people should be able to apply the idea to other applications too.

As future effort, more work needs to be done to train the normal MD space for different vehicle drive cycles, so a normal MD space library can be built for different testing purposes. In addition, after software is classified as "Abnormal", the further algorithms need to be designed to find the specific high MD points and analyze which software variable(s) has caused the high MDs.

ACKNOWLEDGMENTS

We would like to thank Alan Wu from ASI, Gary Nichols, John Waller, and Rob Carpenter from Delphi Powertrain for their helpful discussions during this project.

REFERENCES

1. Magdy Hanna, Principles of Software Testing, The 2006 International Conference on Practical Software Quality and Testing, Sept. 11-15, 2006, Minneapolis, MN
2. Genichi Taguchi, Subir Chowdhry, Yui Wu 2001, The Mahalanobis-Taguchi System, McGraw-Hill
3. P.C. Mahalanobis, On the generalized distance in statistics, Proceedings of the National Institute of Science of India 12 (1936) 49-55
4. W. H. Woodall et al, "A Review and Analysis of the Mahalanobis-Taguchi System," Technometrics, Volume 45, Number 1, February 2003.
5. Richard O. Duda, Peter E. Hart, David G. Stork, Pattern Classification, 2nd edition, 2001, John Wiley & Sons, Inc.
6. Kenji Kurihara, Establishment of Fault Diagnosis System for Racing Car by Mahalanobis-Taguchi System (MTS)