A Framework for the integration of the test bench environment and the engine application software into Matlab/Simulink

Gerd Schlager, Valentin Kordesch, Heinz Waras

Abstract

The adoption of base engines for special applications in the environment of passenger cars, trucks, boats and special purpose vehicles is an area of expertise for the Engineering Center Steyr (ECS). In the context of the Electronic Control Unit (ECU) calibration a target oriented, analytic approach is necessary to fulfill the requirements under the constraints of the increasing cost pressure. To meet the increasing requirements in reference to the ECU calibration, at ECS the model based calibration toolbox (MBC) from Mathworks is used. The functionality of the MBC is expanded by a own toolbox, the ECS MV toolbox. This toolbox includes functions for the optimization of models with 2 stages and the interfaces to the test bench environment. With the toolbox Simulink models can be prepared, which the interface blocks to communicate with the test bench and the application software and additional measurement periphery. The Simulink interface block to the test bench environment includes an on- and offline modus and a free configurable internal and external limit monitoring.
In this paper an overview of the implementation of the interface to the test bench environment will be given. The different strategies for the calculation of the operating point trajectories will be explained and the details of the limit value monitoring are shown.
The functionality of the MBC and the MV toolbox in the context of calibration and optimization are shown on a 6 cylinder diesel engine.

1. Introduction

The calibration of the ECU has a great influence on the acceptance of an engine. Therefore the application engineer has to analyze the ECU maps with care, to achieve an optimal combustion process and therefore optimal performance, fuel consumption, emission behavior and so on.
Due to the availability of new technologies, for example common rail techniques with 4 pilot and 3 post injections, the complexity increase and therefore with the knowledge of expertise only sub optima can be achieved.
To meet the ascending requirements and therefore to handle the increasing number of degrees of freedom, at ECS statistical models in combination with mathematical optimization routines are used.

For the analysis of statistical models and the interpolation of maps different software packages are on the market. The focus of these software packages based on the mathematics and therefore this software packages are not equipped with interfaces to the test bench and application software. So the data for the analysis of the models are acquired by other software and has to be provided.

To avoid additional overhead the first step was to carry out a market analysis, to find a possible software package which can be integrated in the test bench infrastructure

of the ECS. With the software package, DoE test planes should be automatically executed on the test bench, measurement files for the different operation points created and the right actions carried out if limit violations occurs. The interface should not influence the infrastructure of the ECS, means it should be transparent to the test bench engineer and no modifications of the test bench and the application software should be necessary.

1.1 Market-Analysis

Up to the time of the investigations (spring 2004) there are two groups of software products available on the market.

- Software with an interface to the test bench and the application software: AVL Cameo
- Software without the required interfaces means statistical software: Matlab / Model Based Calibration Toolbox (MBC), Modde, Minitab, RS1/Discover, JMP, Statistica, I-Sight and so on.

The product AVL Cameo is a key turn solution, which includes a lot of the requirements defined at ECS. At ECS the test bench software Tornado from the company Krist, Seibt and CO is used. This software does not support a native interface to Cameo. At the time of the investigations Cameo based on Matlab and therefore a Matlab license is necessary for each Cameo PC which is an additionally drawback because it results in additionally costs and administration costs.

The investigations show, that no software product exists which has the necessary interfaces to Tornado and Inca and therefore to fulfill all requirements defined at ECS. Therefore the decision was to realize the interfaces as in-house code. For the model analysis and optimization an existing software product should be used, which is designed as an open framework to integrate the interfaces. It would also be fine if the chosen software can be used for code generation, to build a single executable which can be run on every host with a appropriate runtime.

Matlab is used at ECS a fairly long time. It is a de facto standard at ECS in the context of numerical simulation and optimization. With the Model Based Calibration Toolbox (MBC) from the Mathworks a Matlab toolbox exists, which includes a lot of methods, necessary for the effective planning of test plans and optimization of ECU maps.
Based on this facts the focus on the further investigations is concentrated on the MBC and Matlab/Simulink. The MBC has the following advantages:

- Open architecture (e.g. simple integration of own optimization algorithms)
- Models with different stages (local and global models)
- High model variety
- ECU structures can be modeled in Simulink and taken into consideration during the map calibration (feature calibration)
- Template system for different model structures

There is only one disadvantage: The missing interfaces to the test bench- and application software.

Due to the very high Matlab know-how at ECS, the excellent features of the MBC it was decided the add the additional functions and interfaces to the MBC, respectively create a new toolbox with the additional functions.

1.2 Requirements for the interfaces

For the optimization of the maps the following workflow is necessary (offline mode):

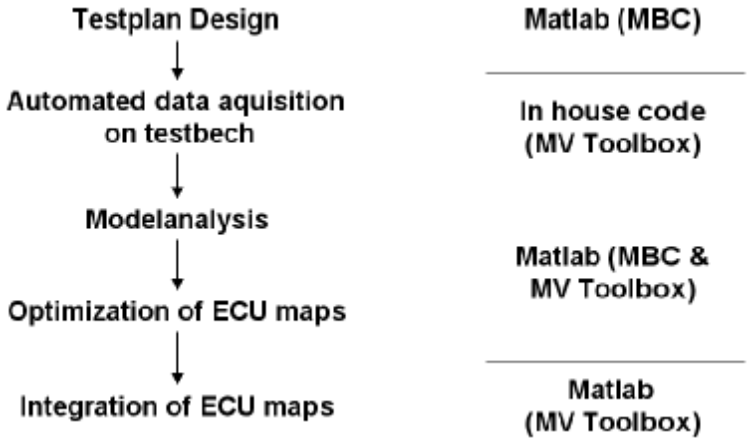

Figure 1: Offline optimization workflow

According to figure 1 it can be seen, that the primary task is the realization of the interfaces between the MBC and the test bench and application software in Matlab/Simulink.

The realization should be carried out, that the following facts are given:

- The communication should occur over the network. Also the application and the test bench software should be addressed independent from each other.
- There should be no additional configuration necessary in Inca.
- There should be no additional configuration necessary in Tornado.
- The integration of additional measurement software, for limit monitoring should be possible (e.g. detection of turbo charger surging).
- It should be possible that all values can be saved by Matlab.
- The configuration of the whole software should be in one file and human readable.

The connection should include an offline and online mode.
If the offline mode is used, each control variable value has to be defined explicitly. This means the values of the local and global control values has to be defined. This mode is used if there is only a basic population and therefore the maps are not well defined.
If a good data basis is available, means only a fine tuning is necessary then the online mode can be used. In the online mode, only the global control values has to be defined, the basic control values for the local models are measured during the execution of the test plan. This means after the operation point of a local model is reached, the control values of the local models are measured and due to the defined variation limits the test plan will be calculated.

The basic functions of the connection interface are defined due to the offline modus. To fulfill the requirements of the offline modus the following criteria are necessary:

- Read and write maps into Inca with the possibility to modify several values of a map.
- Read and manipulate the map axis from Inca.
- Read and write values from Tornado.
- Limit value monitoring (internal and external limit monitoring)
- Adaptive storage period between different operation points. A adaptive storage period is necessary to minimize the execution time under the aspect of valid measurements.
- Realization of the test bench interface due to a modular class based framework in C++ (see chapter 2.2.4).

For the realization of the on-line modus additional functions are necessary:

- On-line calculation of the local test planes based on the measurements in the global operation points.
- The possibility to switch between different local model structures. This is necessary to execute a global model with different regions in it. For example with and without pilot injection or in the case of a turbo charger with variable turbine geometry to switch between models for a open and closed loop control.

This requirements implicitly leads to some assumptions for the interfaces to Tornado and Inca. Therefore some investigations are carried out to analyze the available software interfaces which are implemented in Tornado and Inca.

1.3 Selection of appropriate software interfaces

For the communication between Tornado and Inca normally the interface ASAP3 is used. The interface ASAP3 is time consuming to configure in Tornado and also limited in reference to his capabilities.
Tornado has a Distributed Component Object Model (DCOM) [5] interface as universal interface. This interface allows the manipulation of each variable. Also different functions from Tornado can be activated remote via the DCOM interface. The capability of this interface is only limited due to the load of the PC and the load on the network, therefore this interface should be used in Simulink for the connection.

The application software Inca has the following interfaces:

- COM: The COM implementation of Inca 5.0 is not network-compatible, therefore no additional investigations are done.
- Matlab: The Matlab interface based on the COM interface and therefore this interface is also not of interest.
- ASAP3: This interface can not be used from multiple clients at the same time, means Matlab/Simulink and Tornado can not use the Interface parallel. Therefore the requirement to use Tornado and Inca independent from each other is not fulfilled. Another drawback of this interface is the limited capacity.

- ASAM-MCD: According to the specification of the ASAM consortium, this interface is able to handle multiple clients, means Tornado and Inca can use this interface parallel. After the implementation of this interface, based on the ASAM-MCD server from Inca 5.0, we have seen that the multi client capability is not implemented yet. Due to this fact the interface could not be used. If the multi client capability is given, it is possible to switch to this interface because the ASAM-MCD stack is included.

Because of parallel use of Inca, means the time parallel data exchange from Tornado and Simulink to Inca is necessary, acc. to the defined requirements an additional software package is taken into consideration. The IncaServer from the software company ASE, is a powerful network extension of the Inca COM Interface. This server is very stabile, supports multiple clients and is used at ECS since a long time. Based on this facts it can be seen, that the IncaServer is the right choice for realizing the test bench interface. Figure 2 shows the software architecture used in the engine test bench environment at ECS.

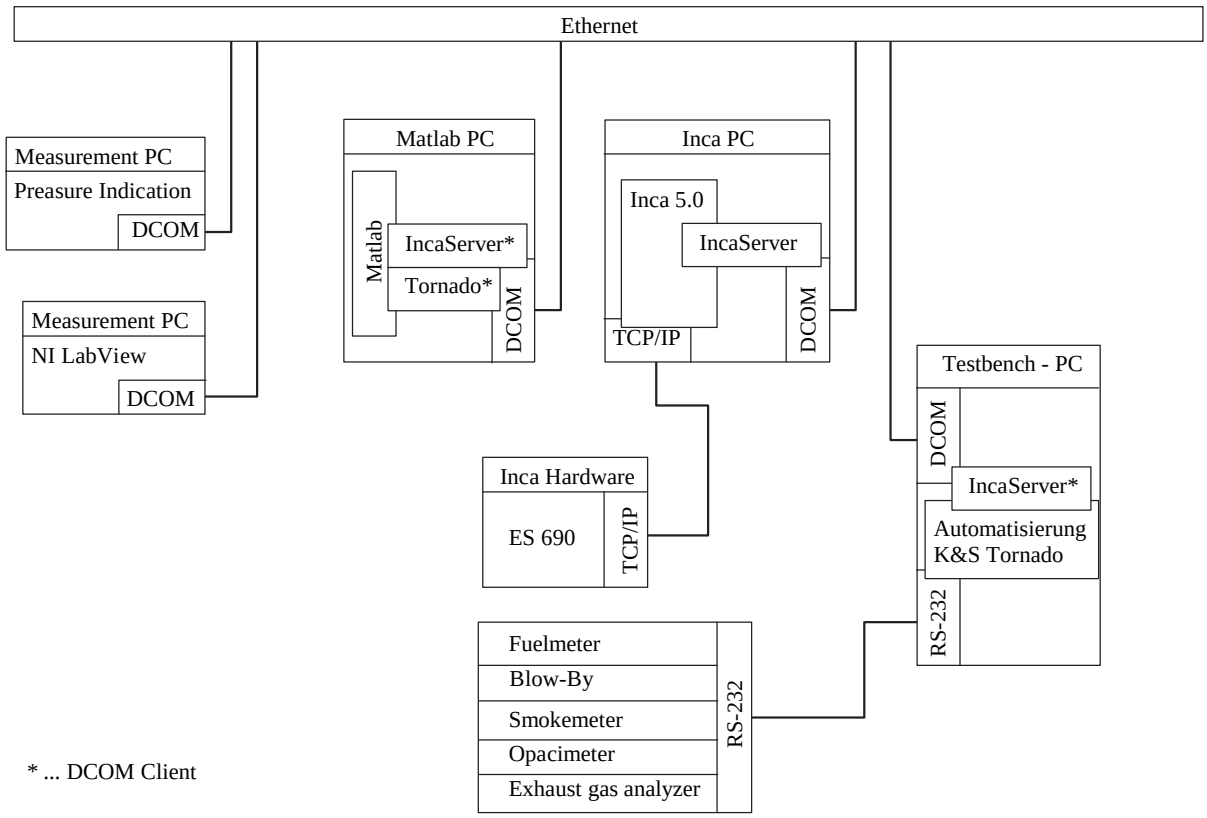

Figure 2: Test bench software environment

2. ECS MV Toolbox

For the integration of the test bench- and application software a Matlab toolbox, the MV toolbox was created. This toolbox has the following features:

- Simulink-block (C-Mex S-function) to communicate with Inca and Tornado over DCOM with the capabilities descript in chapter 1.2.

- Generic Simulink-block to interface with every program which is equipped with a DCOM interface. Therefore different programs can be integrated, for example compiled LabView programs to monitor different physical values (turbo surging)
- Excel Macro and GUI for the design of the xml configuration file.
- Calculation and optimization of local models over the whole operating range with the restriction to polynomial local models.
- Matlab command functions to write and read maps and there axis into Inca over the network
- Tornado measurement data import
- Inca dat File import acc. to the Bosch measurement data format (mdf)
- Inca dcm File import and export

2.1 Test bench integration

As described in the introduction, the connection to the test bench periphery is realized by a Simulink block. Hence a Simulink model is necessary to communicate with the programs used on the test bench and therefore all other available Simulink blocks can be used means the full functionality of Simulink can be used in the model. Furthermore it is possible to visualize the data in a scope and to save all measured data due to a simple write block or to carry out some calculations for example to activate the external limit monitoring or to switch between different model structures.
Figure 3 shows the model to execute an online test plan, means local and global models are used, whereas the initial values of the local models are measured during runtime and the test planes around this initial values are calculated online. This model includes also an initialization block for a external program over DCOM and the appropriate block to get values from the external program. The output of this program is used as input for the external limit violation monitoring. In this case a LabView program on a PXI hardware was used for external measurement acquisition (Figure 4). The LabView program is compiled with an ActiveX server and therefore the program can be controlled by every other Windows PC. The model also includes a stateflow block, which is used for the selection of the right model structure acc. to the global operation point.

Figure 5 shows the configuration GUI of the MV interface block. In this GUI the main configuration parameters has to be defined, means the configuration file and the dimension of the block output vectors. Also different modi or options can be activated. So in a first step the whole procedure can be simulated means the real communication is disabled and the program runs without any DCOM activity.
The configuration of the global and local models is stored in a single human readable configuration file acc. to the xml standard. To give an insight in the possibilities of the communication interface the main sections of the configuration file are described in more detail.

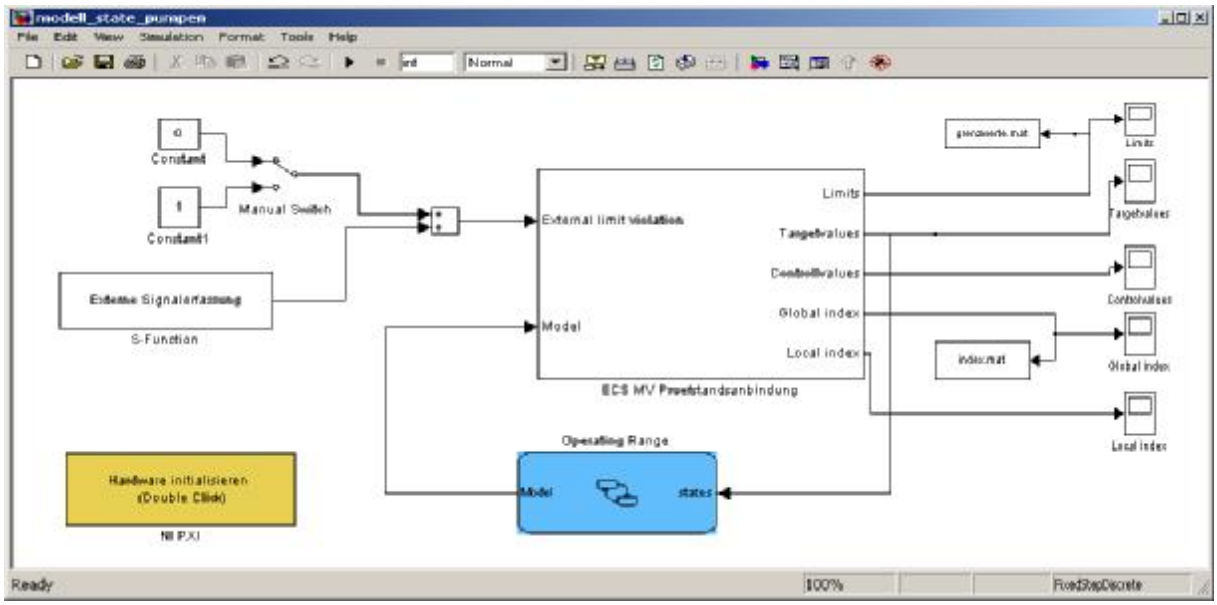

Figure 3: Simulink model for the execution of an on-line test plan

Figure 4: Additional measurement hardware

Figure 5: Parameter for the interface

The configuration file consists of four sections:

Tornado

- Hostname of the Tornado-PC: If a Tornado variable is defined in the configuration file the communication to the tornado PC will be established. Therefore the Tornado PC is controlled by the Simulink model.
- Measurement time: If a defined operation point is reached a measurement of all values of interest will be carried out. This measurement is activated by the Simulink block and executed in Tornado. All values are stored in a Tornado measurement file and can be used for off-line optimization. During to the measurement time no modifications of any control variables are allowed.

- Storage periode: To minimize the runtime for the execution of the hole test plan a rapid change between the different operation points is necessary. If this change is to fast, unwanted dynamic effects of the engine can occur. To avoid this dynamic effects a storage period is used in front of each measurement. The time for this storage period can be defined in relation to the necessary intermediate steps to change from one to another operation point.

Inca / IncaServer

- Hostname of the IncaServer-PC. Normally this is also the PC on which Inca runs.
- Inca Project settings. Normally the project is opened from the Tornado PC, if this is the case only the Hardware has to be defined in the configuration file.

Limits

- Variable identification: Name of the variable as mentioned in Tornado or Inca.
- Software: Software package where the interface can find the variable, means Tornado or Inca or another package if the appropriated class is included to the interface.
- Upper and lower limit: The limit for the activation of the internal limit violation. The limit monitoring is descript in more detail in chapter 2.2.2.

Measurement Points / Test plan settings

- Variable identification: Name of the variable or map as mentioned in Tornado or Inca.
- Software: Software package in which the interface can find the variable, means Tornado or Inca.
- Values: For the definition of the values, it has been taken into consideration that there are two possibilities.
    - Static map optimization–static values: <werte>1;1.1223;2.234</werte>; In this case the values are defined acc. to the calculations of the test plan. This is typically used for global test plans or in the off-line mode, means also the values for the local test plans are defined offline. In this case no modification of the values during runtime is possible.
    - Dynamic map optimization due to maps or values from the Matlab workspace. In this case the values has to be defined acc. to the following syntax: <werte>mat:variable:steps</werte>; If the keyword mat is used, the values of the variable has to be stored in the Matlab workspace and therefore the values can be changed during runtime. Also the shape of a map can be changed whereas in the static case the map is set to the defined constant value.
- Increment: The max. step size between two values. The increment can be defined absolute or relative and the executed increment is analyzed acc. to the chosen method means individual or uniform (see chapter 2.2.1).
- Control values: This value monitors the modification of the measurement values, means, if an Inca map is modified and the modifications lead to a violation of soft or hard internal limits this can be seen with this variable.

- Online section: In this section multiple local models can be defined. A local model consists of multiple variables, whereas each variable has the same settings as the values section except that there are no physical values defined in the <werte> tag but the appropriate dimension or vector of the local test plan which is scaled to the interval [-1,1]. If a global operation point is reached the actual value of the map is used as initial value for the analyis of the local test plan based on the defined limits and the scaled vector of the test plan.

If the configuration file is created, due to the configuration excel macro or due to the Matlab GUI the process of data acquisition is fully defined.

2.2 Details of the test bench interface

2.2.1 Step size control strategy

To change in the n dimensional input space from one state to the next state, two different approaches can be used. Due to the configuration parameter step size, which can be defined for each dimension, a different number of intermediate states between the source and the target state can occur. If the change from the state m to the state m+1 is carried out with an individual number of steps, for each dimension, the so called “Individuelle Schrittweitenregelung” modus is used.
If the modus “Gleichmäßige Schrittweitenregelung” is used, the same number of intermediate steps is used for all dimensions, means the n dimensional vector from the state m to the state m+1 is divided by the max. number of intermediate points. In this case only one configured step size is used, means the minimum step size in relation to the distance between the state m and m+1 in one dimension. Therefore with this method, the minimal geometrical vector between two states in an m dimensional space is used.
The time performance of both methods are identical, because in both cases the necessary time between both states depends on the dimension k, for which the step size / distance quotient will be the minima.
Normally the more intuitive method “Gleichmäßige Schrittweitenregelung” is used. The method “Individuelle Schrittweitenregelung” has shown some advantages in the range of the full load curve because in this region limit violations occur very often. With the method “Individuelle Schrittweitenregelung” and the right choice of the different step sizes, critical values can be influenced.

2.2.2 Limit violations

The possibility to monitor defined values for critical limits and set the appropriate actions is one of the necessary prerequisite to realize a full automatic test bench operation. Therefore the limit violation monitoring is one of the core component of the Simulink test bench interface.
To achieve a very flexible system, the limit violation monitoring is realized as external and internal limit monitoring.

Internal limit monitoring

The internal limit monitoring is integrated in the Simulink function block and therefore all variables and there appropriate limits are defined in the configuration file. The values are elected from Tornado or/and Inca and rated acc. to there limits.
If no limit is reached, the change from the state m to the state m+1 is carried out acc. to the calculated intermediate state vector (Figure 6, left). To minimize the time, necessary for the test bench run, the states are executed in the same order as defined in the configuration file, means no reference state is used. If the state order in the configuration file is defined with care, limit violations can be avoided, because the dynamic of the system can be influenced due to a good choice of the state vectors.
If a limit is reached, this limit violation is detected from the system in the next reading values sample time (typically 0.2 seconds) and the direction of the change of state is inverted instantaneous (Figure 6, middle). This means the values of the last intermediate state are written to Tornado or/and Inca. The number of intermediate states which are used in reverse order, depends on the time length of the limit violation. If the limit violation occurs only short with high dynamic (e.g. indicated peak pressure limit violation due to variation of main injection angle) normally the first intermediate state is used to achieve a state without limit violation. If the response dynamic of the monitored value is very slow (e.g. temperatures) it is possible, that no stabile state can be reached with the intermediate values from m+1 to m and so also the intermediate states between the state m to m-1 are taken into consideration. If the limit violation can not be handled in a defined time duration, the Simulink block activates the idle state for the engine. If the intermediate state vector is executed reverse and a stabile state is reached a measurement will be carried out. Therefore the values of this measurement describes a state without any limit violations. Based on this stabile measurement point, the vector of the intermediate points to the state m+2 is calculated and executed. To avoid to go to the point m, the time for the execution of the test plan will be minimized. This is important especially if global test plans are carried out, which has more then 500 measurement points.

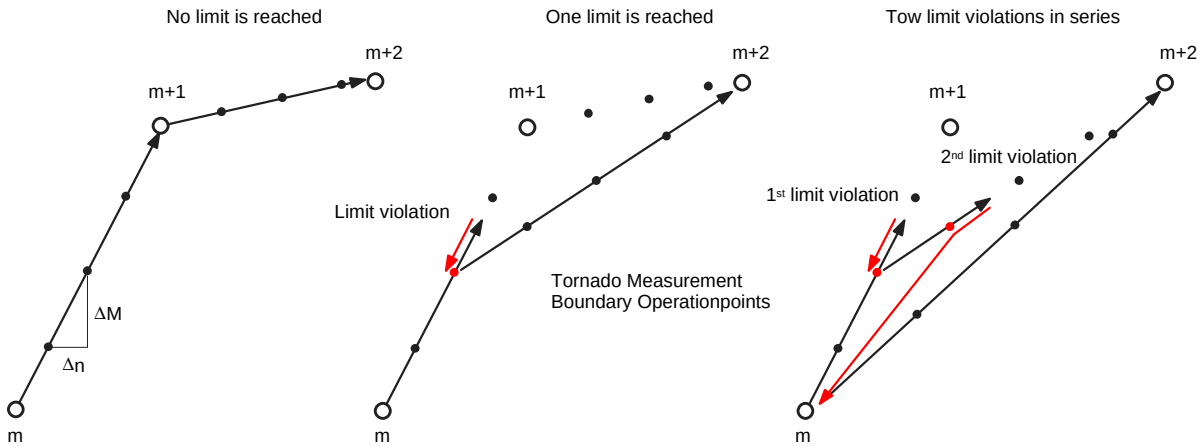

Figure 6: Trajectories of the input space

If limit violations occur on two successive trajectories, that means two target states could not be reached successive, the last known stabile state m will be used (Figure 6, right). If the state m is reached the intermediate state vector between m and m+2 are calculated and therefore a new trajectory will be used. If the state m+2 could not

be reached with the new trajectory, the limit violation is now handled as the first limit violation (Figure 6, middle) and therefore the same rules are used as described before.

External limit monitoring

Due to the input external limit violation (Figure 5) external values, that means values which are not acquired from the Simulink interface block can be monitored and therefore additional data acquisition software can be used. If the input of the external limit violation is set to one, the rules of the internal limit violation are executed, means the procedure described in the chapter before are carried out.
For example the external limit violation monitoring can be used to monitor turbo charger surging effects. Therefore a LabView program is created, which was compiled with a build in DCOM server and therefore the program can be remote controlled. The LabView program measures the pressure in front of the compressor and carries out an analysis. The result of this pressure analysis is a flag which represents the status of the turbo charger means surging or not. A Simulink m-function reads this flag from the LabView program over DCOM and writes to the external limit violation input. If the flag has the value 1, the internal limit violation is activated and therefore the last stabile state will be reached. So the optimization of the maps for a VTG turbo charger can be carried out automatically.

The mechanism described above are valid for the execution of local and global test plans.

2.2.3 Time schedule

The time scheduling of test bench interface is also a core component with a strong interaction with the methods for handling limit violations. To fulfill the requirements, means different sample times for reading and writing values, and to handle measurements and limit violations four parallel threads has to be used. Figure 7 shows a task with mixed threads, because time triggered and event based time slices are necessary.
Acc. to the common approach for simple scheduling algorithm for periodic tasks (rate monotonic scheduling), the thread with the highest frequency has the highest priority. Therefore the thread for reading values, which runs with the defined sample time, has the highest priority and can not be interrupted from another thread (non preemptive).

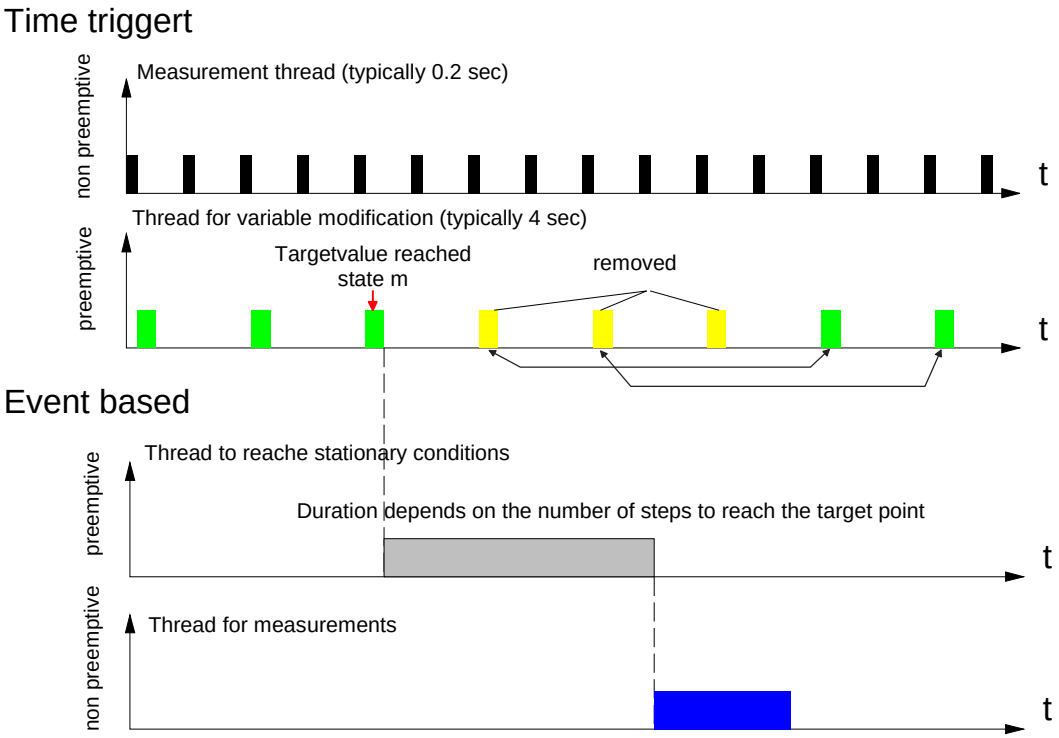

Figure 7: Time scheduling of the test bench interface

The value transfer from the Simulink block to Tornado or Inca is also managed in a periodic thread. This thread is preemptiv, because it can be interrupted and suspended from the measurement thread which is necessary to realize a storage period to achieve stabile conditions and to carry out a steady state measurement. This means the transition from the state m-1 to the state m, the intermediate states or there values are written periodically and if the last state of the intermediate vector is reached, which is also the target state m, the activation of the storage period is carried out. The storage period is only interrupted due to the measurement task, which is necessary to detect limit violations. After the duration of the storage period elapsed the measurement method of Tornado will be activated. After the measurement the new intermediate state vector to go from state m to state m+1 is calculated and executed. Figure 7 shows the described time scheduling.
Figure 8 shows the time behavior, in the case of a limit violation with very slow dynamic, because the limit violation occurs during the storage period. After the state m was reached, the flag for the storage period was activated and therefore the variable modification thread was suspended. If a limit violation occurs during the storage period, the storage period will be interrupted and the last values of the intermediate vector are written to Tornado or/and Inca instantaneously. As long as the limit violation is active the values of the intermediate vector from state m to state m-1 are executed with the sampling time of the write thread. If the limit violation is deactivated this is identified in the next read sampling time and therefore the storage period is activated and the write thread will be suspended. After the duration of the storage period is elapsed, the measurement is activated and the new intermediate state vector calculated.

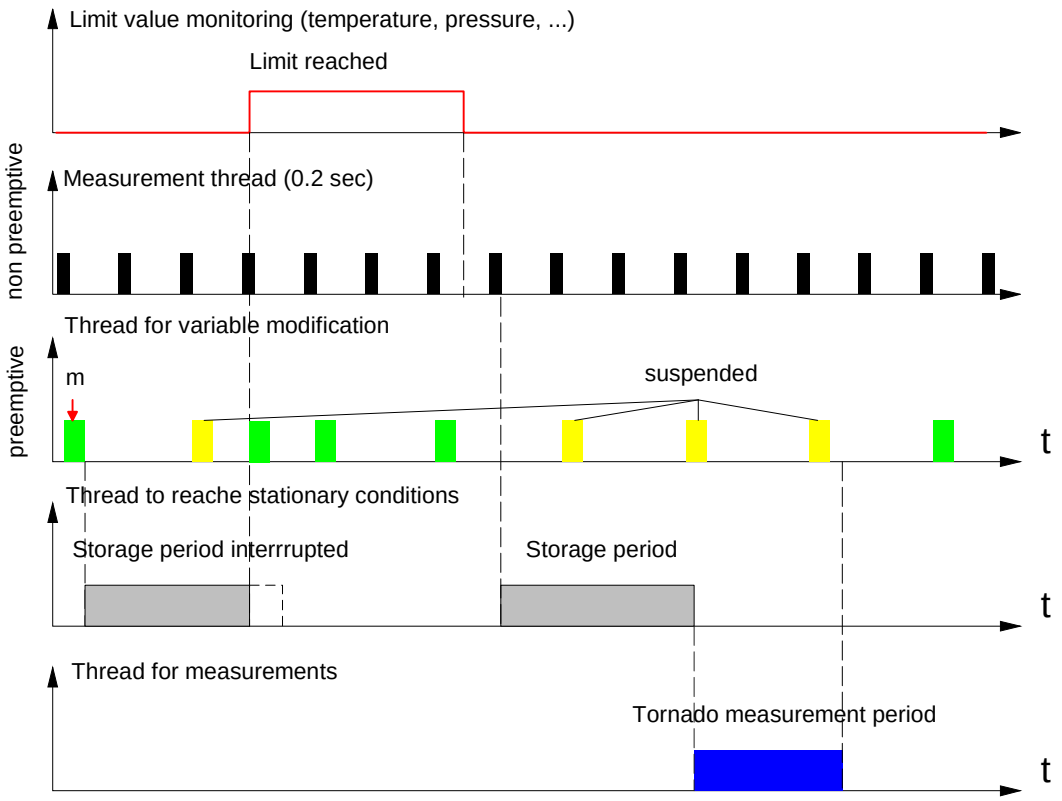

Figure 8: Time behavior in the case of a limit violation

The timing is handled by the system clock of the operation system, means no real time kernel will be used, so the application can run without any modifications on a normal Windows PC with Matlab/Simulink installed.
Therefore the Windows scheduler is used and only *soft real time* can be achieved. Because of the use of DCOM and no real time interface for the network communication also an additional time delay between the systems are accepted [8]. This means a time based analysis of the data acquired in Matlab/Simulink, for example a FFT, is not valid. If a dynamic data acquisition in Matlab is necessary, hard real time acc. to the specified sample time is necessary. In this case the model can be build with the Real Time Workspace or the XPC Target Toolbox for a appropriate hardware to fulfill the requirements in time.

2.2.4 Software - Implementation

The test bench interface was implemented as C++ Level 2 C-Mex S-function, to compile the dynamic link library for non real time and real time use. Figure 9 shows the software design of the implementation. There is one main class which handles the limit violation and the classes for the local and global models. The local model class are inherited from the global model class and extended at the possibility to calculate local test planes. The interaction due to the test bench and application software is managed by an abstraction layer, therefore additional software components can be included in future. Due to the class based approach maintenance and expandability exists.

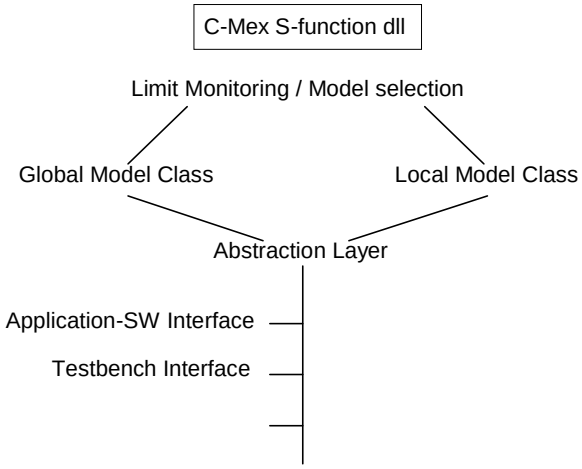

Figure 9: Software design

3. Applications

As example for the use of the MV Toolbox, the optimization of a 6 cylinder common-rail DI engine with a waste-gate turbo charger will be shown. The engine is equipped with a Bosch EDC 16 ECU. The basic maps of the ECU permits a stabile operation of the engine, but there is an optimization potential, especially in the context of fuel consumption and emission.

3.1 Calibration of the fuel quantity map

As an example for a global calibration, means a calibration over the hole range of operation points, the optimization of the torque-fuel quantity map will be shown.
This map is necessary for the right assignment of the brake torque and the fuel quantity. Because the most ECU maps are in relation to the engine speed and the fuel quantity the fuel quantity map is very important, because each error in this map leads to an offset in all other depending maps.

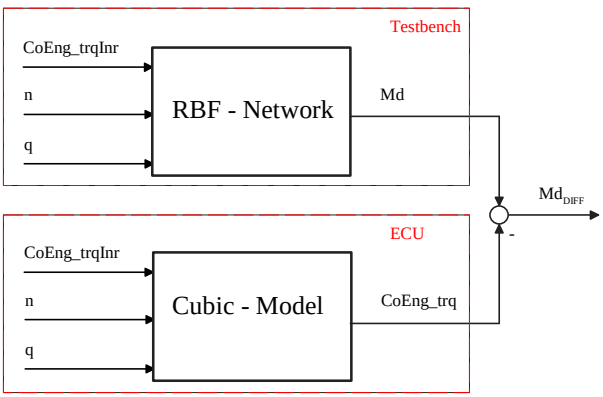

Figure 10: Model for the fuel quantity map calculation

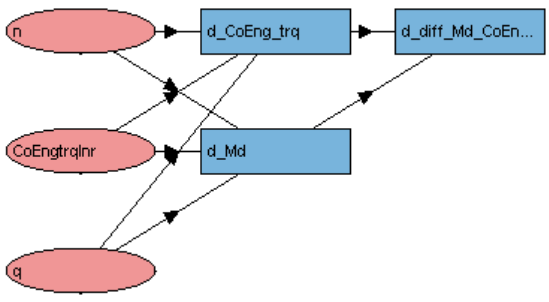

Figure 11: Model structure

The quality of the map can be judged due to the brake torque measured on the test bench and the calculated brake torque in the ECU. If the map is correct both torques are identically.
The input variables of the map are the inner torque and the engine speed. The dimension of the map is 16x16, means the inner torque range and also the engine speed is divided in 16 sections and therefore 256 supporting points are defined.
The inner torque and the brake torque (CoEng_trq) differs due to the friction torque. The shape of this interrelation is well known and therefore this interrelation is modeled with a cubic model.
Ideally the measured torque Md must have the same model structure as the calculated brake torque form the ECU (CoEng_trq). Depending on the fuel quantity, means to much or to less fuel is injected, the torque Md and CoEng_trq differs. Therefore the measured torque Md contains implicit the errors in the fuel quantity map and therefore the model for the torque Md has to describe local errors very well. Acc. to this requirement hybrid radial basis functions are used. Figure 10 shows the input and output variables for the model. The model structure used in CAGE can be seen in figure 11.
Due to the necessary accuracy for the mapping of local torque errors, typically 70 to 80 measurement points are used for this optimization analysis.
Figure 12 shows the torque differences before and after the optimization for 70 measurement points. The figure shows, that due to the optimization a dramatically reduction of the torque can be obtained. The result of the optimization, the fuel quantity map shows figure 13.
For this optimization no modifications in the application software Inca are necessary, because only the operating points are changed. Due to the functionality of the toolbox at this stage the knowledge of the full load curve is not necessary, because the toolbox can be used to find the full load curve acc. to the defined limits. Means if a operating point can not be reached, because a limit violation occurs (e.g. p_{max} – max. cylinder pressure, T_{30} turbine inlet gas temperature) a measurement on the last stabile point will be carried out and this point will be used as point of the full load curve.

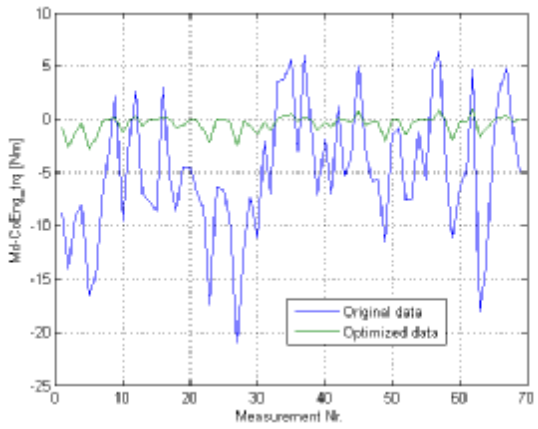

Figure 12: Torque differences

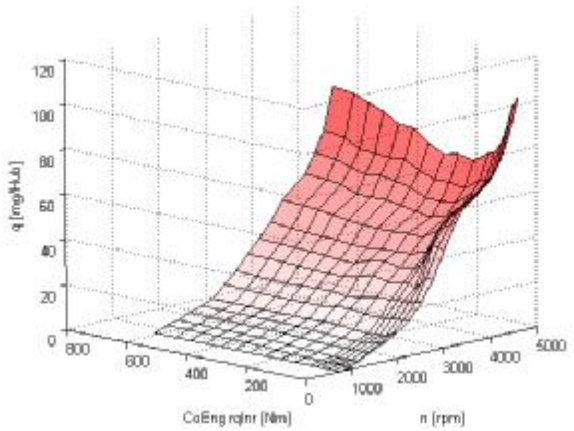

Figure 13: Optimized fuel quantity map

The expenditure of time for this optimization, under the assumption of the knowledge of the full load curve is about 3.5 hours for the calculation of the test plan, the execution of the test plan and the analysis of the new map. If the full load curve is not know, due to the iterative process to find the limits the expenditure of time can be expanded.

3.2 Emission and fuel consumption optimization – use of local and global models

The following example shows the functionality of the test bench interface in the online mode. In this case the emission and fuel consumption has to be reduced. Especially the NO_x emissions and also the fuel quantity has to be reduced under the premise that the particulates will increase, but also fulfill the limits. Also the limits for the max. pressure p_{max} and the limits for different temperatures, especially the turbine inlet temperature T_{30} should not be violated, to avoid component damage.
For the optimization local and global models are used. Normally global models are used to analyze the parameters of the local models for the hole operating range of an engine. This means global models for the values of interest, in this case NO_x, HC, CO, FSN, p_{max}, T_{30}, and so on are analyzed, on the basis of the local models. This means the local model parameters are interpolated due to a global model. Therefore the uncertainly of the local model will be expanded by the uncertainly of the global model. Based on the global model the ECU map will be analyzed. If this procedure is used there is also the problem that it can occur that the local model structure is not valid over the hole operating range because different functions which can be switched off or on, e.g. pilot injection.
Our goal are optimized maps to archive the target values, means we are not interested in the global models of the values. We use the results of the optimized local models as supporting points of the maps. This supporting points are used as inputs for the global models, which are used to make models for the ECU maps. Therefore we use one stage models from the MBC to calculate maps based on the outputs of local optimized models.

For this example the following variables are used as input parameters for the local models:

- Time between pilot and main injection
- Quantity for pilot injection
- Main injection angle
- Rail pressure

To analyze the maps over the full operating range, based on the optimization results of the local models, global models are used. Therefore the global variables describes the operation points and therefore the variables

- engine speed and
- engine brake torque

are used as input variable for the global models.

The maps for the rail pressure and the main injection angle are active over the hole operation range. Because the pilot injection is only active at lower speed, different models has to be taken into consideration. Acc. to figure 14 four model structures are necessary to describe the change between the operating points with and without pilot injection in the right manner.
In range 1 (model 1) all four parameters of the local models can be varied parallel. In range 2 the pilot injection is deactivated, means only the parameters main injection angle and rail pressure can be varied. Between range 1 and 2 a hysteresis appears and therefore the functionality depends on the direction means from lower to higher

speed or vice versa. If we change from range 2 into the hysteresis the pilot injection is inactive and therefore only the main injection angle and the rail pressure are factors and therefore model 3 is active. In the other case, means from range 1 in the hystersis theoretically all 4 factors are active. Theoretically, because it makes no sense to use all four factors, if there is no possibility to chose other maps for the rail pressure and main injection angle, if the pilot injection is active. Normally model 3, which includes the parameters pilot injection quantity and timing, is optimized based on the results of the optimization with model 4.

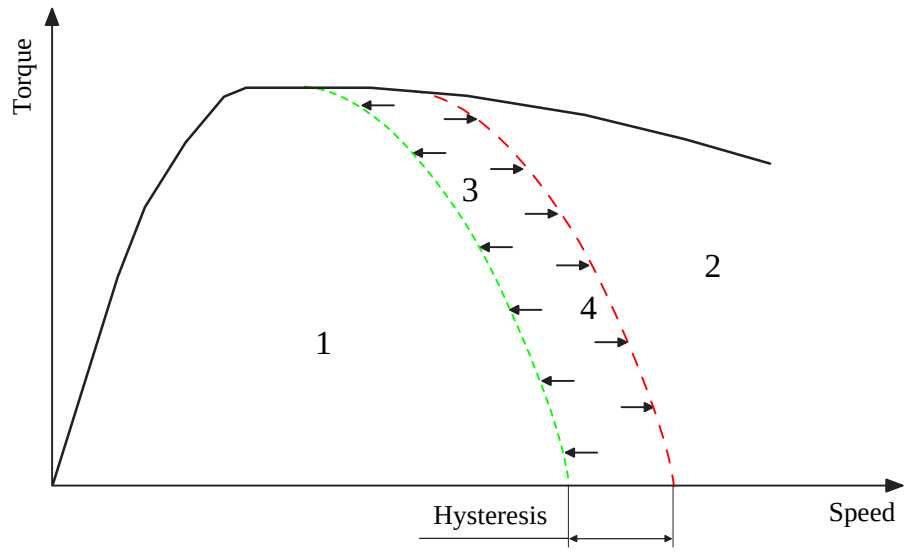

Figure 14: Pilot injection operation range

For the optimization the following model order was used:

2 **à** 3 **à** 1 **à** 4

During the data acquisition for model 1, the optimization of the local model 3 is carried out, because for the data acquisition of model 4 the optimized maps for rail pressure and main injection angle are used. This means the models 2, 3 and 1 can be run at once in online mode. For the data acquisition of model 4 a off-line optimization is necessary.

Data acquisition in online mode

The online mode can be used, if the available ECU maps leads to a stabile operation of the engine. To use the online mode a global test plan is necessary. This global test plan was calculated on the basis of the known full load curve with the MBC. Also two different local test plans, with two and four factors are calculated with the MBC scaled to [-1, 1] and exported. This means the variations of the values from the local test plans are carried out during run time. The values of the local test plans, based on the measurement value from the original map in the used operation point and the defined variation limits, which can be defined as a function of the operation point. In this case ±10% from the measured value for all operation points and variables are used. The data flow can be seen in figure 15.

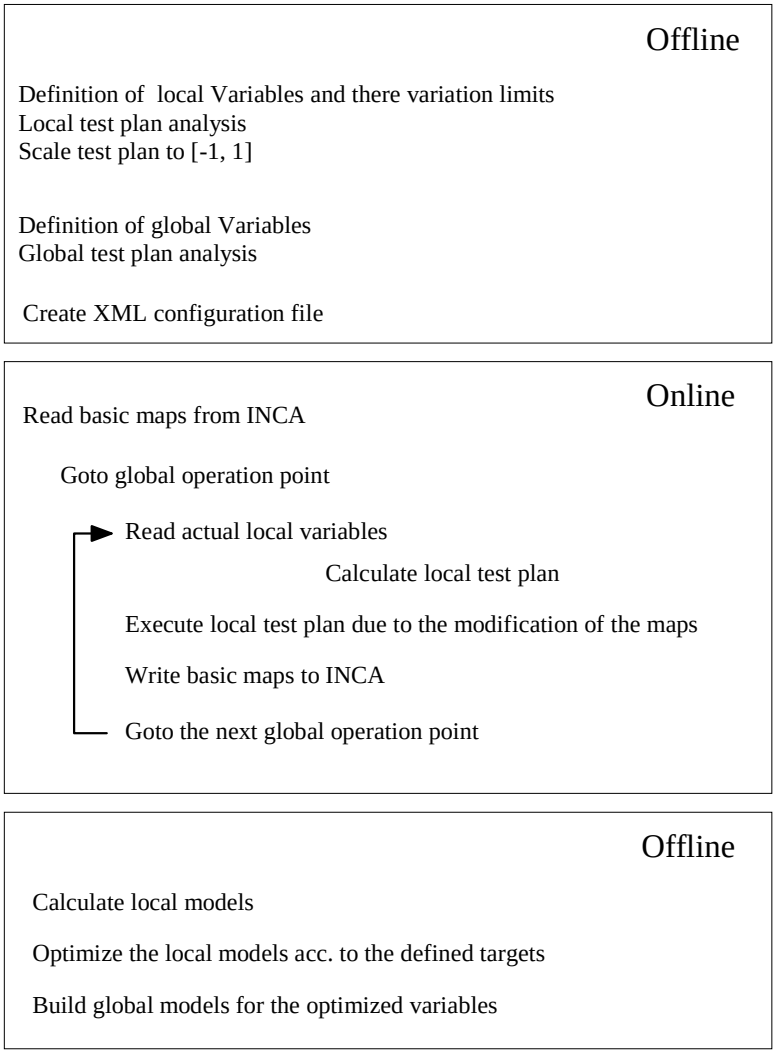

Figure 15: Online mode data flow

Figure 16 shows the output of the index scope of the data acquisition model (Figure 3). The measurement index describes the index of the state in the state vector, therefore this index also describes the number of the measurement point in the tornado measurement file. The measurement status flag, shows the actual status of the measurement. The following values are valid for this flag:

- 0: If the measurement status value is 0, there is a state change from the state m to m+1 acc. to the chosen step size control strategy and the defined step size. Between state m and m+1 n intermediate states are used. The actual position in the intermediate state vector is given by the intermediate condition number (Figure 17).
- 1: If the measurement status value is 1 a target state is reached and therefore no variables are changed and the storage period starts. The duration of the storage period depends on the number of necessary intermediate states. There is a change, if the intermediate state vector has more than 10 elements, in the duration of the storage period.
- 2: If the value is 2, a Tornado measurement is in progress and therefore the duration describes the necessary measurement time to acquire all values means smoke, fuel consumption and so on.

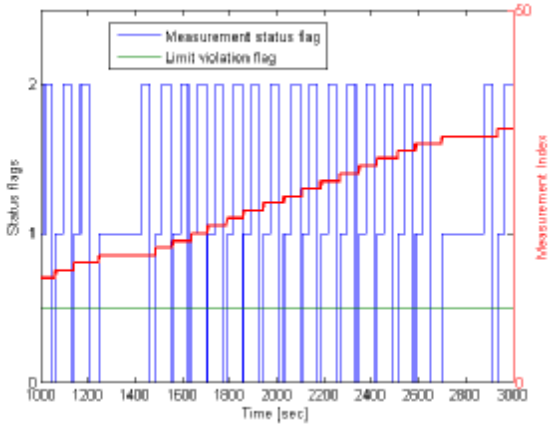

Figure 16: Status flags

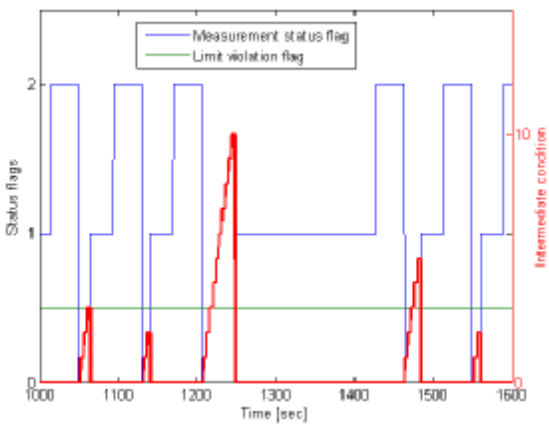

Figure 17: Intermediate states

The limit violation flag changed his state if a limit violation occurs. If there is no limit violation the value is 0.5, if a limit is reached this value is 1.5 (Figure 19).

Limit violation

For this optimization operating points on the full load curve are used. Due to the variation of the main injection angle the max. cylinder pressure violates his limit. Figure 18 shows the maximum cylinder pressure and the chosen limit. The indicated pressure is measured with a LabView program, with a build in DCOM server, so the max. pressure can be read over the network from the Simulink block and also from Tornado.

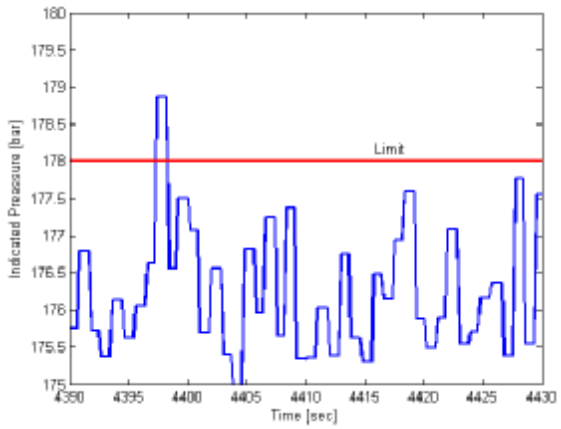

Figure 18: Peak pressure monitoring

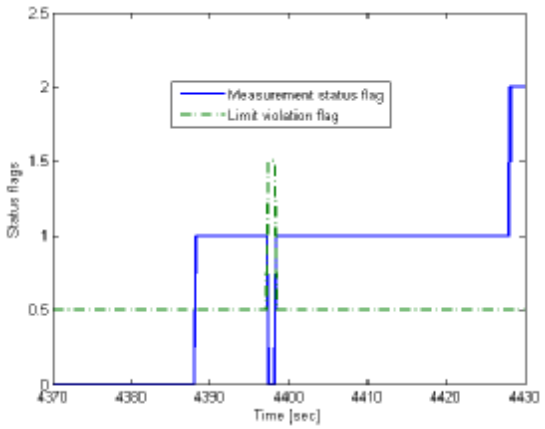

Figure 19: Limit violation

As it can be seen on figure 19 the limit violation occurs not in the transition between two defined states, means in the range of the intermediate state vector, but in the storage period because the measurement status flag has the value 1. If a limit violation occurs during the storage period, the storage period is interrupted and the last stabile point will be used. In this case the values of the last state from the intermediate vector are written to Inca and Tornado. After the state change the limit violation is disabled and the storage period starts again. Figure 19 shows that after the interrupt of the storage period the new storage period starts and has a duration acc. to the

defined values in the configuration file. After the storage period a Tornado measurement will be carried out and therefore the measurement status flag goes to 2.

After the data acquisition the local models are calculated. For the local models multiple, polynomial models of the following form are used:

$$y = a_0 + \sum_{i=1}^{k} a_i.x_i + \sum_{i=1}^{k}\sum_{\substack{j=1 \\ i<j}}^{k} a_{ij}.x_i.x_j + \sum_{i=1}^{k} a_{ii}.x_i^{\,2} + \ldots$$

The variables x_i are the measurement data and the factors a are the regression coefficient. Because the factors a are linear in this model this factors can be analyzed by a simple matrix calculation. For the analysis of models of this form a Matlab class was created in the MV Toolbox to analyze the local models and also to optimize this models acc. to the different target functions over the full operation range under assumption of the measurement files. For this class it is only necessary to define the measurement files, the model order and the weights for the target function.
The analysis for this example, means the calculation and optimization of 5 local models for 22 operating points with 4 input variables and 8 operating points with 2 input variables takes less than 1 minute (laptop with 1.6 GHz, 512 MB Ram)!
Figure 20 shows the graphical result of one local model.

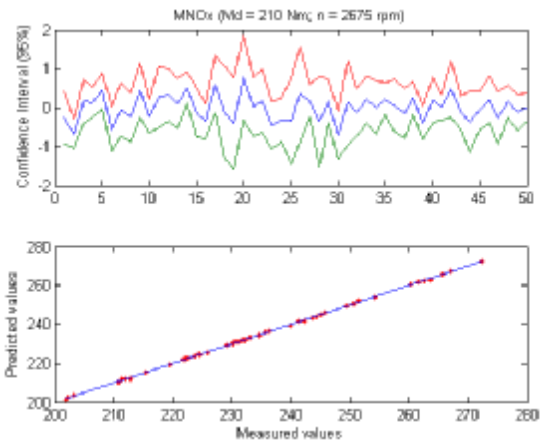

Figure 20: Model quality

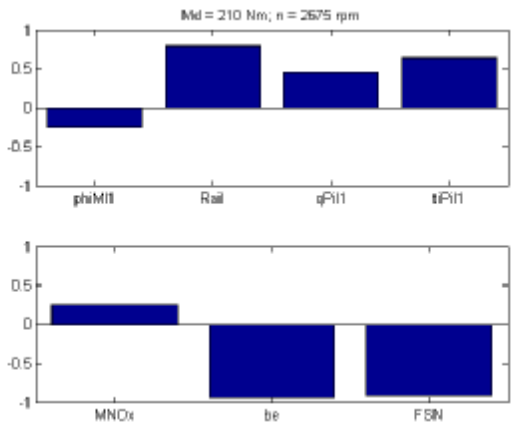

Figure 21: Optimization results

Optimization of local models with hybrid genetic algorithm

To achieve optimal maps for the ECU the local model has to be optimized for different target criteria. All variables are varied in an range of 10%. Because the models are polynomial, they are only valid in the variation range and therefore the optimization of the target functions should be carried out in this range. Means a optimization function is defined and with the chosen algorithm this function is optimized acc. to the defined criteria in the variation range.

The targets are:

Target	Weighting
NO_x **à** Min	1
be **à** Min	2
FSN **à** Min	0.5

This targets should be reached under the following constraints:

$$p_{max} < p_{Limit}$$
$$T_{30} < T_{Limit}$$

Based on the target criteria a function of goodness was defined. This function of goodness has to be minimized by an optimization algorithm. To minimize the function an algorithm with two stages is used. The first stage is an genetic algorithm (GA) [6] to find the global minimum in the high dimensional input space. In general the solution of the genetic algorithm is only in the vicinity of the real global minimum and therefore in a second stage a gradient based algorithm is used to find the real minima. So the output vector of the GA is used as input vector for the gradient based algorithm [7] and therefore it is granted that the values for the global minima are used. Figure 21 shows the results of the optimization. The range [-1,1] is scaled to the variation bandwidth of 10%. For the measured values the range [-1,1] represents the minimal and maximal value which results from the variations. In this case the fuel consumption and also the FSN can be decreased to the minima, whereas the NOx emissions are marginally over the average of all measurements.

Map analysis based on the results of the local model optimization

Based on the results of the local optimized models the maps for the ECU are analyzed with the MBC.

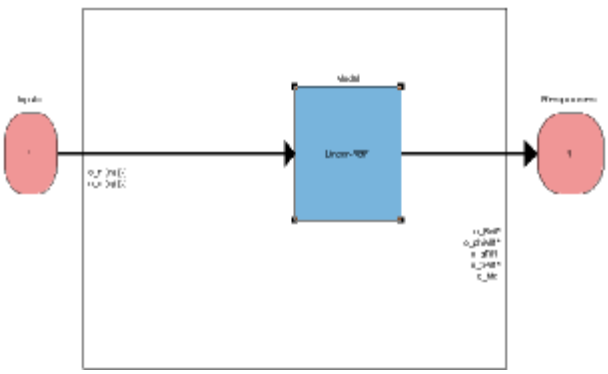

Figure 22: One stage model for map analysis

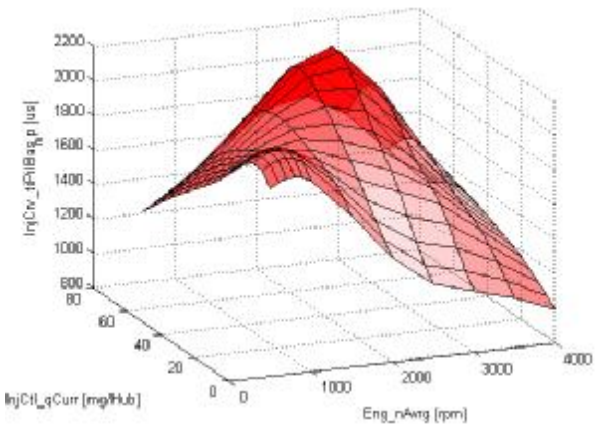

Figure 23: Pilot injection timing

The MBC is normally used to set up two stage models and perform the map analysis with this model. Because we are only interested in optimized maps, we use the MBC to analyze one stage models for the interpolation of the maps based on the supporting points of the local models. For the optimization we use hybrid RBF networks, because this kind of models has shown very good results. Figure 22 shows a one stage model for the model of the maps to analyze. Figure 23 shows the result of the one stage MBC model, a map for the timing of the pilot injection which can be written to Inca by the functions of the MV Toolbox.
With the methods described in this chapter all critical values are below there limits and all targets are reached. Also additional improvements in respect to the emissions are reached.

The expenditure of time for the data acquisition, for 22 operation points with 4 factors and therefore with 50 measurements per local model and 8 operating points with 2 factors and 16 measurements per local model was approximately 28 hours.

4. Summery

The MV toolbox, and especially his core component, the interface to the test bench- and to the application software is presented. Different possibilities to interface with the test bench software Tornado from the company Kristl, Seibt & CO and the application software INCA from ETAS are shown. The core components for the automated execution of test planes are discussed in detail. This means if limit values are under- or over-shot the right action has to be carried out. The implementation shows how this limit violations can be handled under the aspect to minimize the running time on the test bench.
The functionality of the toolbox is shown by the optimization of the specific fuel consumption under the aspect to fulfill the emission requirements. This optimization is done on a 6 cylinder diesel engine equipped with common rail technique and a waste-gate turbocharger.

5. Outlook

The MV Toolbox was up to now used for different projects with great success. The systematic approach leads to very good results in a fraction of time compared to a conventional optimization.
To improve the quality of the ECU calibration and to minimize the time for the data acquisition the following topics are in progress:

- Gradient based algorithm to predict limit violations during runtime faster, means the development of an estimator for limit violations to minimize the measurement runtime.
- Optimization of polynomial models during runtime to estimate the model quality and therefore to decide if additional measurements are necessary during runtime.
- Algorithm for the partition of the map axis to take the topology of the maps more into consideration.

5. References

[1] The Mathworks: The Matlab Programming Language
[2] The Mathworks: Model Based Calibration Toolbox
[3] B. Klein: Versuchsplanung – DoE, Oldenburg Verlag
[4] Douglas C. Montgomery: Design and Analysis of Experiments, Wiley
[5] Tanenbaum, Steen: Verteilte Systeme, Pearson Studium
[6] Chris Houck, Jeff Joines, Mike Kay: A Genetic Algorithm for Function Optimization: A Matlab Implementation; NCSU-IE TR 95-09, 1995
[7] Gill, P.E., W. Murray, and M.H. Wright, Practical Optimization, London, Academic Press, 1981
[8] Jeffry Richter: Windows: Programmierung für Experten, Microsoft Press