Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Departement für
Umwelt, Verkehr, Energie und Kommunikation UVEK

Bundesamt für Energie BFE

Final report 15.03.2022

Design Methodology for Supervisory Control of Vehicle Propulsion Systems, Simulation and Test Bench Studies (MERFAS)

Auftraggeber:
Bundesamt für Energie BFE
CH-3003 Bern
www.bfe.admin.ch

Auftragnehmer:
Institut for Dynamic Systems and Control (IDSC), ETHZ
Sonneggstrasse 3
8092 Zürich
www.idsc.ethz.ch

Autoren:
Prof. Dr. Christopher Onder, ETHZ, IDSC onder@idsc.mavt.ethz.ch
Johannes Ritzmann, ETHZ, IDSC jritzman@idsc.mavt.ethz.ch

BFE-Bereichsleiter:	Luca Castiglioni
BFE-Programmleiter:	Luca Castiglioni
BFE-Vertragsnummer:	SI/501643-01

Table of Content

1 Introduction and Project Goal

Todays' Diesel-based propulsion systems power a wide range of vehicles, giving rise to a highly diversified product portfolio. To limit the development costs for such a vast portfolio, these systems are designed in a modular fashion and consist of various sophisticated and well-understood components such as engines, gearboxes, aftertreatment systems, etc. Ensuring compliance with legislative limitations while reducing the fuel consumption is no longer possible by optimizing the single system components alone, given their high degree of technical maturity. Further improvement of system performance therefore requires the exploitation of the interactions between the components by means of a holistic control approach.

This project aims at developing a transferable methodology for the layout and a framework for the implementation of predictive supervisory controllers targeting the optimal system-wide coordination of the major energy flows and components. These supervisory controllers will continuously define the set points for all relevant components at every point in time, relying on well-designed component controllers to perform their tasks. An example of such a system is a Diesel-electric hybrid vehicle, where the operation of the electric machine has an impact on the possible system response and the fuel consumption of the Diesel engine. These systems are over-actuated and, in order to obtain the best result regarding fuel consumption, require the use of specialized models and controllers.

In the scope of this project, we design a predictive supervisory controller for a delivery van. We intend to show that, by hybridizing such a vehicle and applying a predictive supervisory controller, which considers the vehicle's exhaust gas aftertreatment system, the fuel consumption can be reduced by 20% compared to the same van using a state-of-the-art conventional propulsion system, while continuing to meet the given emission legislation.

To facilitate the development of the modelling and optimization framework required for designing predictive supervisory controllers, three systems will be considered. This allows for incremental development of the functionality of the modelling and optimization framework, as well as the utilization of parts of the final framework for other projects, as soon as they become available. The three systems are:
1) Diesel engine with exhaust gas aftertreatment system (ATS)
2) Diesel-electric hybrid vehicle with ATS
3) Fuel-cell-electric hybrid vehicle

The research partners FPT Motorenforschung AG and ETH Zurich are currently working on the development of a predictive supervisory controller for vehicle propulsion systems as described in the scope of this project. The funds granted by the BFE-MERFAS subproject SI/501643-01 are used to cover the wage costs of research assistants at ETH who develop software tools and perform pre-studies in the scope of the overall project.

In this report, we present the results of the work carried out by the research assistants employed using the funds from BFE. First, we present the dynamic programming (DP) Matlab function updated within the scope of this project. This function was used to calculate optimal solutions, which are used as benchmarks for future online optimization methods and controllers. The DP function was used in various subprojects within the scope of MERFAS as well as other research projects at ETH and FPT. Second, we present two pre-studies for System 2 of the overall project.

2 Dynamic Programming Function

The success of the entire project heavily relies on the availability of fast solvers to solve the optimal control problem (OCP). For online real-time control, direct methods are typically used [1]. While these solvers tend to give results within reasonable time, they are only guaranteed to find the global optimum for convex problems. However, most of the OCPs considered in this work are not convex. Therefore, a different kind of optimization algorithm, namely DP, is required to obtain benchmarks, which are used to evaluate the quality of the solution obtained by direct methods.

In the late 2000s a generic DP function for Matlab was developed at IDSC. The corresponding documentation can be found in [2], [3], and [4]. In the literature on powertrain optimization, this code is generally the go-to method for global optimization, e.g., [5], or even as a benchmark for the development of tailored DP optimization method, e.g., [6]. While this function is proven and accepted in the field, the optimization routine is not very flexible and the required calculation time quickly becomes excessive for larger problems.

We therefore decided to adapt and improve this function by optimizing its structure, with the goal of reducing the computational time and the memory required for larger problems. To this end, several research assistants were hired.

The remainder of this section is structured as follows. First, we give a brief introduction to the DP theory. Second, we give a summary of work carried out at IDSC prior to the MERFAS project. Third, we outline the development. Finally, we present two application examples.

2.1 DP Theory

DP is based on Bellman's principle of optimality [7], which states "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision". A more intuitive explanation can be given based on Figure 1, where the temporal evolution of a dynamic state $x(t)$ is illustrated. Bellman's principle of optimality states that if $[(\mathrm{a}), (\mathrm{b})]$ is the optimal trajectory from x_0 to x_f, then (b) must be the optimal trajectory from $x^\star(t_1)$ to x_f. As a result, the overall OCP can be split up into and solved as a series of incremental subproblems, where by the "optimal cost to go", denoted $J^\star(x, t)$, is introduced to describe the minimal cost associated with reaching x_f from $x(t)$.
In the following we will outline the DP algorithm to solve the generic OCP stated below with inputs $u(t)$

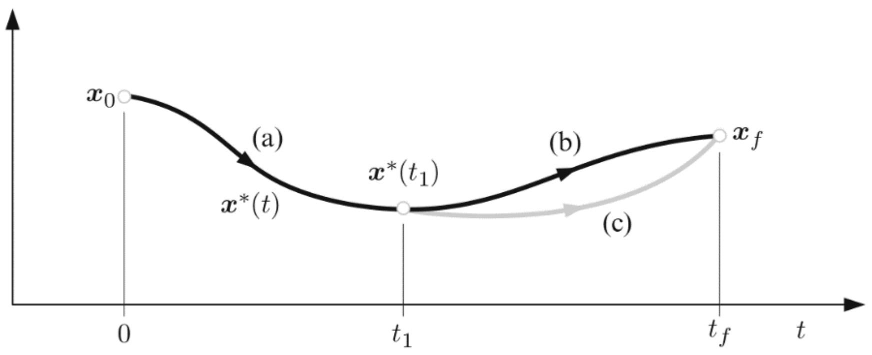

Figure 1 - Visualization of Bellman's principle of optimality. Adopted from [19].

and dynamic states $x(t)$.

$$\min_u \int_0^{t_\mathrm{f}} g\big(x(t), u(t)\big)\ \mathrm{d}t + h\big(x(t_\mathrm{f})\big)$$

$$\begin{aligned}\text{subject to }\ &\dot{x}(t) = f\big(x(t), u(t)\big)\\ &x(0) = x_0\end{aligned}$$

By discretizing both the state and the time of the OCP, a finite (but very large) number of possible trajectories result, among which the optimal one is to be identified. Form the final time we get $N = t_\mathrm{f}/\Delta t$, where Δt is the time step. The time is denoted $t_k = k\ \Delta t$ and the state at t_k is denoted $x[k]$. The main steps of the DP algorithm are briefly outlined in the following, while a more detailed theory is given in [8].

1) Initialize the final optimal cost to go based on the terminal cost function $h(x[N])$ of the OCP, i.e., $J_N(x[N]) = h(x[N])$.
2) Build the optimal cost to go map and the optimal input map by moving backwards in time, i.e.,
$J_k(x[k]) = \min\limits_u\big(g(x[k], u)\Delta t + J_{k+1}(x[k] + f(x[k], u)\Delta t)\big)$
$u^\star[k] = \operatorname*{argmin}\limits_u\big(g(x[k], u)\Delta t + J_{k+1}(x[k] + f(x[k], u)\Delta t)\big)$
3) Move forwards in time, starting at $x[0] = x_0$ and applying the previously calculated input $u^\star[k]$

In the specific DP algorithm discussed in the following, linear interpolation is used during Steps 2 and 3 to facilitate the use of a coarser state and input discretization with a limited loss of optimality. Further details can be found in [2].

2.2 Previous Development

In [2], the framework of the original generic DP function for Matlab, called dpm, is presented. The user needs to define the structure of the OCP and provide a model function, while the dpm function performs the required steps of the DP algorithm to solve the problem. In [3], the framework was extended by introducing the boundary line method to precalcuate the boundary of the feasible region for single-state problems. This method alleviates interpolation issues at the boundary of the feasible region and allowes the user to use a much coarser discretization with limited loss of optimality, thereby speeding up the optimization. In [4], a similar method, called the level set method, was introduced that is applicable to multi-state problems. Even though an exact characterisation of the feasibility bound is no longer achieved, the method still allows for a coarser grid at a limited loss of optimality.

The original dpm function was coded as a single function, making it hard for a user/developer to fix bugs, improve performance, or adapt the code to a specific problem. Furthermore, the most general problem structure was assumed, meaning that while the code was very versatile, it would needlessly repeat certain calculations for problems of certain types.

Roughly two years before the start of the MERFAS project, a PhD student at IDSC started implementing a DP framework, called dpa, similar to that of the previous publications, but based on object-oriented programming. By defining a DP optimization class that contains predefined methods and properties, the optimization framework becomes much more organized, providing a user/developer with a clearer overview of the code structure and facilitating adaptions to individual subroutines of the optimization. Furthermore, steps were taken to avoid redundant calculations by exploiting problem structure. While the new framework showed promising results, the code was never finalized, as the PhD student working on it shifted his research in another direction.

2.3 Function Development

In the scope of the MERFAS project, we continued the development of the dpa function. The bulk of the work thereon was carried out by research assistants whose wages were paid using BFE funds. The goal of this work was to develop a generic DP function for Matlab that is at least as versatile as the original dpm function, but which is easier to extend, exploits problem structure where possible, and results in shorter calculation times and a reduced memory requirement.

The work on the dpa function was split into four steps. First, we extended the existing dpa framework to regain the full functionality of the dpm function. Second, we developed a testing framework with which future developments are tested to avoid unintended adverse effects on the software. Third, we accelerated the optimization, by exploiting problem structure wherever possible, by allowing the user to formulate a less memory intensive model evaluation, and by allowing the user to reduce the data. Finally, we tidied up the code and prepared it for distribution. These steps are outlined in the following.

2.3.1 Regain Functionality of dpm

The version of the dpa function from which we started our work could solve simple academic examples such as the fishery problem presented in [2], or the two-tank problem presented in [4], but it was not yet applicable to generic problems. Furthermore, neither the boundary line method for single-state problems nor the level set method for multi-state problems were implemented. The first step therefore was to adapt the code in such a way as to regain the full functionality of the dpm code.

To check the implementation of the boundary line method, we compared the obtained results of the two methods, i.e., dpa and dpm, for the fishery problem stated in [2] and [3]. The optimization task thereby is to determine the optimal fishing rate in order to maximize the number of fish extracted from a lake for a given initial population and a lower bound on the final population. The dynamic state of the problem is the population of fish in the lake. The rate of increase in the population rises with the fish population and falls with the fishing rate. Figure 2 shows the comparison of the analytic solution to the solutions found using dpm and dpa with the boundary line method. We can conclude that both the dpm and the dpa algorithm find the correct solution.

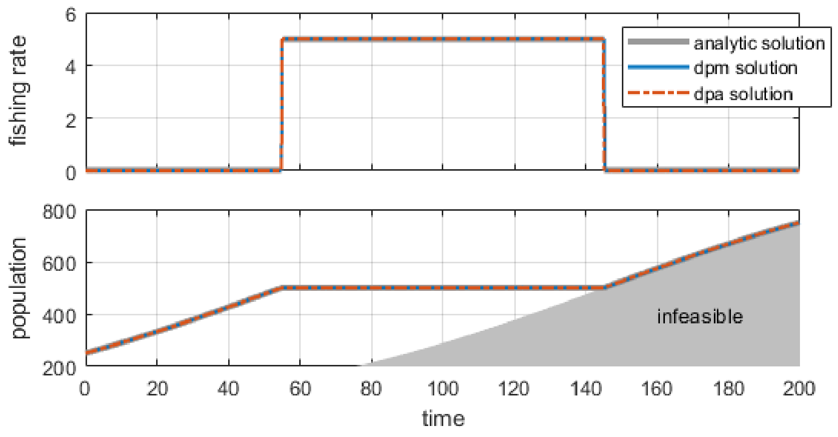

Figure 2 - Result comparison for the fishery problem.

To check the implementation of the level set method, we compared the obtained results of the two methods, i.e., dpa and dpm, for the two-tank system stated in [4]. The optimization task thereby is to fill two tanks to a minimum level within a given time by controlling two valves in order to minimize the total water consumption. Figure 3 shows the comparison of the analytic solution to the solution found using dpm and dpa with the level set method. We can conclude that both the dpm and the dpa algorithm find the correct solution. The difference between the obtained solutions and the analytical solution can be attributed to the discretization used by DP and is reduced when a finer grid is used. The minor differences between the dpm and the dpa solution can be attributed to slight adaptations in the later stages of development of the dpa.

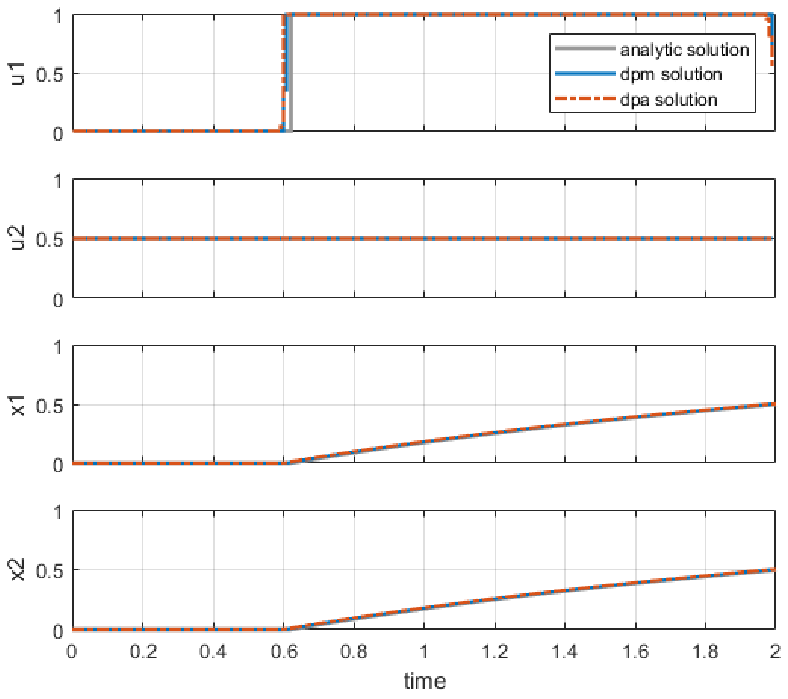

Figure 3 - Result Comparison for the two tank problem.

From the two optimizations shown, we can deduce that the functionality of the dpm function was fully regained by the dpa function.

2.3.2 Implement Testing Framework

During software development, it can happen that a change in the code that improves one feature has adverse effects on other features. As the developer is likely to focus only on the feature he wishes to improve, these adverse effects can easily be overlooked and included in the code without anyone noticing. To avoid this, we implemented a testing framework that checks certain subfunctions of the code (unit tests) and solves an array of optimization problems (functional tests) to check that no unintended changes to the code functionality have been introduced. This testing framework is run each time a change to the code has been implemented.

2.3.3 Code Acceleration

In this section, we outline the three main steps through which the computational time and the memory usage of the dpa function are reduced.

Structure Exploitation

During the second step of the DP optimization, as outlined in Section 2.1, the optimal cost to go is calculated according to

$$J_k(x[k]) = \min_u\big(g(x[k],u)\Delta t + J_{k+1}(x[k] + f(x[k],u)\Delta t)\big)\,.$$

While the cost to go at the next time step, i.e., $J_{k+1}(x[k+1])$, has already been computed in the preceding step, the system model must be evaluated to obtain the values of the stage cost $g(x[k],u)$ and the state update $x[k+1] = x[k] + f(x[k],u)\Delta t$. In general, the system model has to be evaluated at each time step. However, if the time step, the state grid, and the input grid are the same at all evaluation steps and no disturbance is present, the evaluation of the system model is equivalent at each evaluation step. As a result, the model evaluation can be performed once at the beginning of the optimization and the result used throughout, which reduces the computational effort.

To illustrate the benefit of this method, we consider the fishery problem of [2]. As was done in [3], the problem is solved using the boundary line method and considering a fixed grid, i.e., the state and input grids are not changed during the optimization. As the problem features a fixed time step and does not have a disturbance, this problem is an ideal candidate to illustrate the benefit of structure exploitation. The results of the dpa algorithm with and without structure exploitation are stated in Table 1. While the obtained solutions are equivalent, the structure exploitation reduces the computation time.

Table 1 - Comparison of solutions to the fishery problem with and without structure exploitation. The results were obtained on a PC with a quad-core 3.4 GHz processor and 64 GB of RAM.

Method	**Objective**	**Computation time [s]**
without structure exploitation	450.69	14.8
with structure exploitation	450.69	12.3

Vector Mode

During the second step of the DP optimization, as outlined in Section 2.1, the system model has to be evaluated for all possible combinations of the state and input values. As Matlab can process array computations faster than loops of scalar computations, we build up arrays that contain the variation in the states and inputs and then call the system model once with these arrays as input. This has the benefit that all computations within the model function can be performed element-wise.

For illustration purposes, we consider a system with the two states x_1 and x_2 and a single input u. Let us assume further that the stage cost is given by $g = x_1\, x_2 + u\, x_1$. The resulting data structure is shown in Figure 4 and will be explained in the following. First, consider the variables x_1, x_2, and u. When considering the actual variation in the respective variables, all three variables can be viewed as grid vectors containing the grid points in their respective dimensions. Figure 4 shows these vectors in the column “vector mode”.

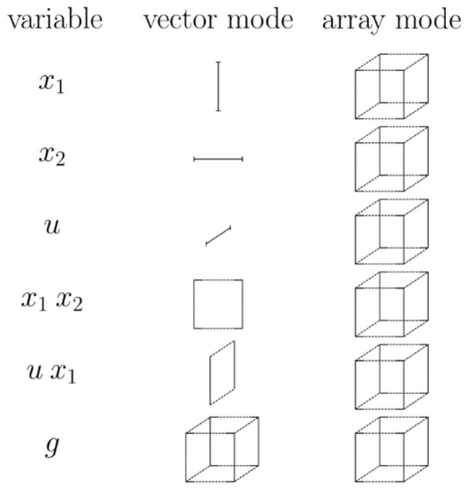

Figure 4 - Data structure of variables during system model evaluation for vector and array mode.

In the standard dpm implementation, which uses array mode, Matlab’s *ndgrid*-command is used to build arrays from these vectors as shown in the right column of Figure 4. This has the advantage that the user can use arbitrary computations in the system model, as long as they are carried out elements-wise. The correct dimensions of the outputs result by design. As a consequence, the optimization is easier to set up. Unfortunately, each variable needs a lot of memory and often contains repeated data. If, for example, a gridding of 1000 grid points for each variable is chosen, the vector x_1 contains 1000 values and requires 8 kB of memory, while the corresponding array x_1, which contains no additional information, contains 1000^3=1 billion values and requires 8GB of memory. Clearly this method is unsuited for problems that require a fine discretization or feature many states or inputs.

To facilitate solving such problems, we have introduced vector mode, which requires the user to design the system model function such that it can deal with the vector-based data structure. Here, the states and inputs are passed to the system model in their actual size, i.e., as vectors, as shown in the middle column of Figure 4. The system model must then be designed in such a way that the variable size is increased only if required. For Matlab’s element-wise operations, such as addition or multiplication, this is done automatically. Two examples for multiplications, namely $x_1\, x_2$ and $u\, x_1$ are given in Figure 4. The resulting variable structure is extended to contain all relevant data while avoiding repetitions. Finally, the user has to guarantee that the stage cost g returned by the system model has the same size in vector mode as in array mode. For the example stated here, this results by design.

Vector mode affects the data handling, but not the information contained in the variables. Therefore, it does not influence the quality of the solution, but can reduce the computation time and the memory usage, allowing larger problems to be solved efficiently. The actual reduction in memory strongly depends on the specific system model. The dpa function was implemented to use array mode as standard, as this allows new users to solve small optimization problems quicker. By activating vector mode, advanced users can solve larger problems but need to be more involved when setting up the system model.

Table 2 shows the objective value, the memory usage, and the computation time for the hybrid electric vehicle (HEV) problem presented in [2]. It shows that, for this example, switching from array mode to vector mode results in a reduction of almost 70% for both the memory usage and the computation time.

Data Precision

Even with vector mode activated, we found that solving large problems often requires a lot of memory. By default, Matlab uses double precision, resulting in a memory usage of 8 bytes per numeric value. To simultaneously reduce the memory usage and the computational time of dpa, the data precision can be reduced from double to single. This is achieved by defining the lower and upper bounds of all states and input grids, the disturbance values, and all model parameters using single precision. The subsequent dpa optimization will then be carried out using single precision. As with vector mode, such an implementation can only be expected from an advanced user, but will greatly increase the size of the problem that can be solved.

The bottom two rows of Table 2 show that the memory usage can be further reduced by 46% and the computation time can be reduced by 27.5% compared to the case where vector mode is employed with double precision. Furthermore, no increase in the objective function was observed, as the precision loss from using single precision instead of double precision is marginal.

Table 2 - Comparison of DP solutions to the HEV problem. The results were obtained on a PC with a quad-core 3.4GHz processor and 64GB of RAM.

Method	**Objective**	**Memory [MB]**	**Computation time [min]**
dpm	0.12028	2’000	24.4
dpa (array mode, double precision	0.12006	1’900	12.3
dpa (vector mode, double precision)	0.12006	600	4.2
dpa (vector mode, single precision)	0.12006	330	3.1

Compared to dpm, the optimization with dpa using vector mode and single precision uses only 16.5% of the memory and takes only 12.7% of the computation time for the considered problem. This shows that the steps taken to allow larger problems to be solved more effectively were indeed successful.

2.3.4 Code Finalization

During the development of the code, the focus was on functional development. As a result, the optimization result returned to the user was given in the form used by the optimization, which was not very convenient for the user. Furthermore, multiple methods and properties became redundant during the function development, but were not fully removed from the code.

In this final development step, the user-friendliness of the dpa function was improved, especially by improving the structure of the optimization results, and any development dead ends of the code were removed.

2.4 Application within MERFAS Project

In this section, an example for the application of the dpa optimization routine within the MERFAS project is given. Specifically, the problem of optimizing the operation of a hybrid electric vehicle (HEV) with a variable engine calibration that can be adapted to the current mission, the given NO_x limits, and the aftertreatment system (ATS) operation is solved. In this report, only the DP optimization itself is outlined and a single result is shown. Further details of the approach and additional optimization results can be found in [9].

The objective of the OCP is to minimize the fuel consumed by the vehicle on its mission. Thereby, a charge-sustaining operation is enforced and an upper bound on the emitted tailpipe NO_x mass must be respected. Dynamic states for the battery state of charge and the ATS temperature are introduced, while the emitted tailpipe NO_x mass is included in the objective using an optimization weight.

Table 3 lists the states and inputs considered in the optimization, as well as the chosen gridding. The two engine strategy inputs define the operation of the variable engine calibration according to a preoptimization outlined in [9]. Due to the number of states and inputs considered during the optimization, the application of dpm would have been challenging. With dpa, the problem could be solved efficiently and the solutions for different vehicle configurations, driving cycles, and NO_x limits were compared.

Table 3 - States and Inputs considered in the DP optimization of [9].

Type	Variable	Grid points	Step size
State	Battery state of charge	51	0.006
State	ATS temperature	46	10 K
Input	Engine strategy input – ATS heating	21	0.05
Input	Engine strategy input – NO_x emission	21	0.071
Input	Selected gear (incl. engine on/off)	7	1
Input	Torque split	41	0.1

Figure 5 shows the achievable operation w.r.t. fuel consumption and tailpipe NO_x emissions for different vehicle configurations. Conv FEC corresponds to a conventional vehicle (no hybridization) using a fixed engine calibration and is considered as the baseline. HEV FEC corresponds to an HEV with a fixed engine calibration, where only the torque split can be optimized. HEV FEC w. gears corresponds to an HEV with a fixed engine calibration, where the torque split and the gear selection can be optimized. HEV VEC corresponds to an HEV with a variable engine calibration, where the torque split can be optimized. HEV VEC w. gears corresponds to an HEV with a variable engine calibration, where the torque split and the gear selection can be optimized. The results show that hybridization and optimal gear selection can significantly reduce the fuel consumption, but the achievable reduction in the tailpipe NO_x emissions is limited. With a variable engine calibration, on the other hand, the further fuel consumption reduction is limited, but the achievable reduction in tailpipe NO_x emissions is significant

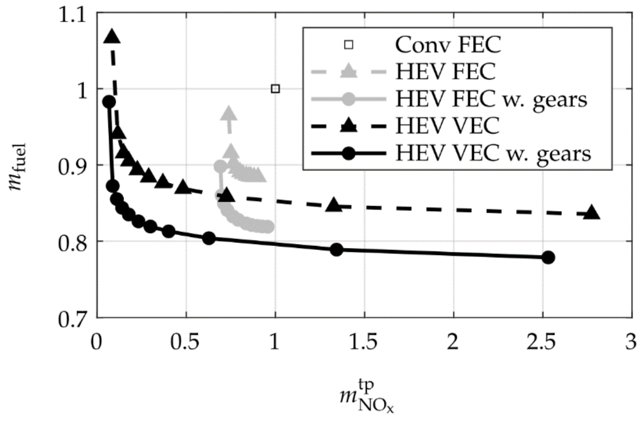

Figure 5 - Optimal operation of considered vehicle configurations for the WLTC. All values were normalized.

This result shows how the stated project goal of achieving a 20% reduction in fuel consumption at equivalent tailpipe NO_x emission could be achieved for the considered HEV compared to the corresponding conventional vehicle.

2.5 Application within other Research Projects

The dpa optimization routine was also applied within research projects outside the scope of the MERFAS project. An example thereof is presented in [10]. The DP optimization used therein is outlined briefly in this section.

An electric motor heats up when used and its temperature is constrained by an upper bound to prevent damage to its components. To avoid exceeding this temperature limit, the use of the electric motor is limited if the motor temperature exceeds a certain value. In the literature this is referred to as thermal derating. The temperature evolution of the electric motor(s) of an HEV should therefore be considered in the energy management of HEVs.

In [10], a predictive supervisory controller is developed to tackle this challenge. The controller's performance is assessed by comparing it to a benchmark obtained using dpa. The considered vehicle features two electric motors, whose operation is set by two torque splits. The objective of the controller is to minimize the fuel consumption over a mission, while enforcing a charge-sustaining operation and respecting the temperature limits on both motor temperatures. The states and inputs considered in the OCP are listed in Table 4, together with the chosen gridding. The number of states considered is even higher than for the example stated in Section 2.4, meaning that an efficient DP optimization is even more important. This is evident from the resulting computation time of 30 days, achieved on a PC with a 24-core 4.3GHz processor and 256GB of RAM, when exploiting all the features of the dpa function outlined in Section 2.3. Solving the problem using dpm would simply have been impossible.

Table 4 - States and Inputs considered in the DP optimization of [10].

Type	Variable	Grid points	Step size
State	Battery state of charge	101	0.004
State	Temperature of motor 1	101	0.3 K
State	Temperature of motor 2	101	0.45 K
Input	Torque split 1	71	0.028
Input	Torque split 2	71	0.028

2.6 Summary

The dpa code developed in the scope of the MERFAS project has two main benefits compared to the original dpm code. First, as shown in Table 2, it is significantly faster. Furthermore, the features outlined in Section 2.3 can be employed, to drastically reduce the memory consumption and the computation time further, making the method applicable to significantly larger problems. Second, the new code structure simplifies further development of the code, and the testing framework is a useful tool to assess the effect of changes on the general functionality, reducing the chance of introducing bugs to the code during development.

3 Prestudies for Diesel-Electric Hybrid Vehicles

Early in the project phase, we found that the main challenge of the work on System 1 is the development of suitable models for the engine, while the main challenge of System 2 was the development of optimization routines capable of dealing with the mixed-integer problem resulting from the introduction of the discrete gear selection and engine on/off input. To facilitate tackling these two challenges in parallel, rather than one after the other, we introduced two subproblems for System 2 that allowed us to develop controllers in an incremental fashion. In this section, we present the resulting optimization routines and online control structures developed for these two subprobelms.

The powertrain layout of the considered HEV is shown in Figure 6. Throughout this section, an engine with a fixed engine calibration is considered, i.e., the engine operation is defined solely by the speed and the torque at which it is operated.

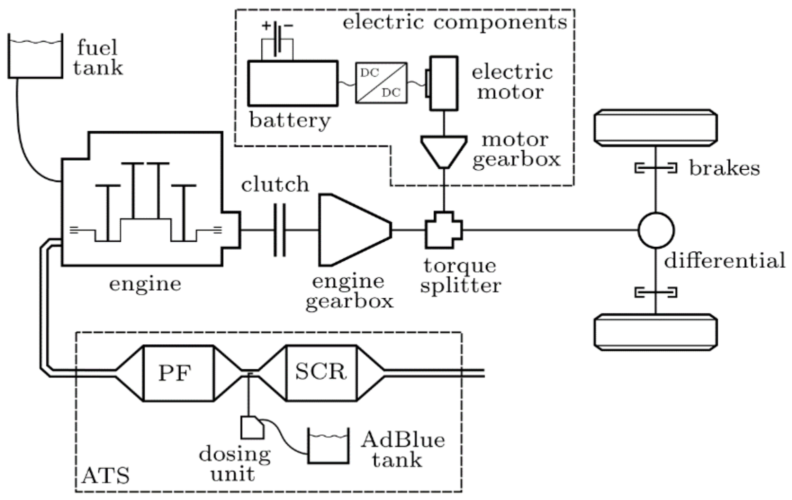

Figure 6 - Powertrain of the considered Diesel-electric hybrid vehicle. The ATS is only considered in Section 3.2.

3.1 Energy Management

The first subproblem does not consider pollutant emissions. As a result, the ATS need not be considered and the task of the resulting control problem is to perform an energy management of the hybrid powertrain. In the following, an overview of the work originally developed in the master’s thesis [11] and resulting in the publication [12] and the patent [13] is given. Further details can be found in the aforementioned works.

We consider the parallel-hybrid powertrain architecture shown in Figure 6. The electric machine is connected to the torque splitter via a fixed-transmission-ratio gearbox and draws power from the battery via a DC-DC converter. The Diesel engine is connected to the torque splitter via a clutch and a six-speed automatic gearbox. The degrees of freedom that have to be set by the energy management system are the clutch position (engaged/disengaged), the selected gear of the engine gearbox, and the torque split. The goal of the supervisory controller is to minimize the amount of fuel consumed, while achieving a charge-sustaining operation.

In [12], based on simplified models of the powertrain components, a convex optimization problem was formulated, for which an optimal control law was derived based on Pontryagin's minimum principle [14]. Next, bisection was used to obtain the constant costate value of the resulting two-point boundary value problem for inactive path constraints. For the case of active path constraints, a multi-point boundary value problem was solved. The developed optimization method achieved quasi-optimal performance and was able to find the optimal solution for driving missions with a duration of 30-60 minutes in less than one second.

Based on this efficient optimization method, a multi-layer predictive energy management controller was developed in [11]. Its structure is shown in Figure 7. Based on the predicted vehicle velocity and road gradient, the Reference Trajectory Generator (RTG) performs the optimization introduced above. It is evaluated only once at the start of the mission. The resulting state-of-charge trajectory over the driving mission is passed to the Model Predictive Control (MPC) block. Using feedback of the current battery

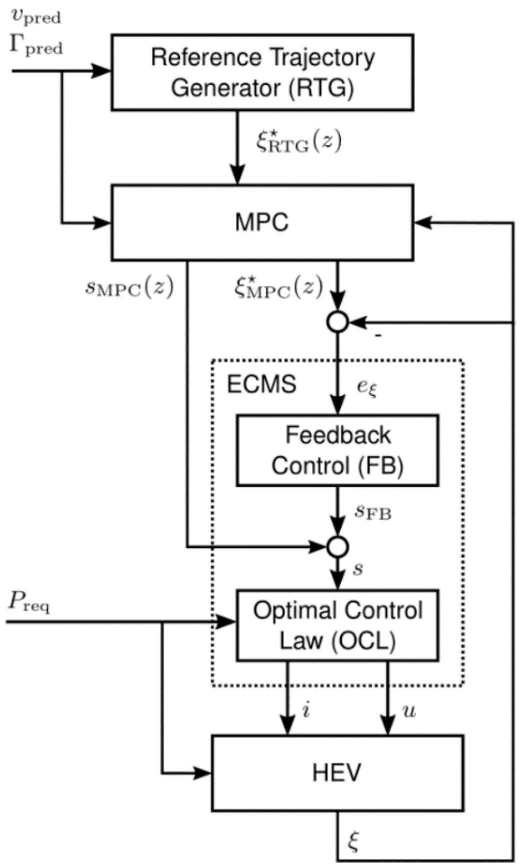

Figure 7 - Energy management control architecture.

state of charge, and predictive data, this block tries to bring the battery state of charge to the reference value over a certain prediction horizon. The MPC is updated periodically during the vehicle operation. The resulting equivalence factor s, obtained from the costate resulting from the optimization, and the updated state-of-charge trajectory are passed to the Equivalent Consumption Minimization Strategy (ECMS) block. Here a further feedback controller corrects the equivalence factor and the final equivalence factor is passed to the Optimal Control Law (OCL) block, where the optimal input for the gearbox and the clutch, i.e., the gearbox variable i, and the optimal torque split u are calculated.

To evaluate the performance of the proposed energy management controller, we considered a realistic driving mission. For the predictive data, we assume that the elevation profile over the mission is known and that the velocity is not affected by traffic. For the simulation cycle, we introduce a velocity disturbance, by including traffic in the simulation, but assume that the elevation is not affected by a disturbance.

We compare the performance of the proposed method, denoted causal, to that of the non-causal benchmark obtained by optimization on the driving mission with traffic using DP, and to that of a state-of-the-art online energy management controller presented in [15], denoted pRSG-ECMS.

Figure 8 shows the results of the performance analysis. The plot on the left shows the resulting state-of-charge trajectories. All controllers manage to achieve a charge-sustaining operation. The causal method results in a state-of-charge trajectory very similar to the one obtained using DP, while the pRSG-ECMS results in a much larger offset. The same result is observed in the fuel consumption, where the causal method results in an increase in the fuel consumption by 1.6% compared to DP, while pRSG-ECMS results in an increase in the fuel consumption by 10.4%. In conclusion, the developed method constitutes a promising supervisory controller for the energy management problem of hybrid vehicles. The optimization method and the controller architecture were patented by FPT Motorenforschung AG in [13].

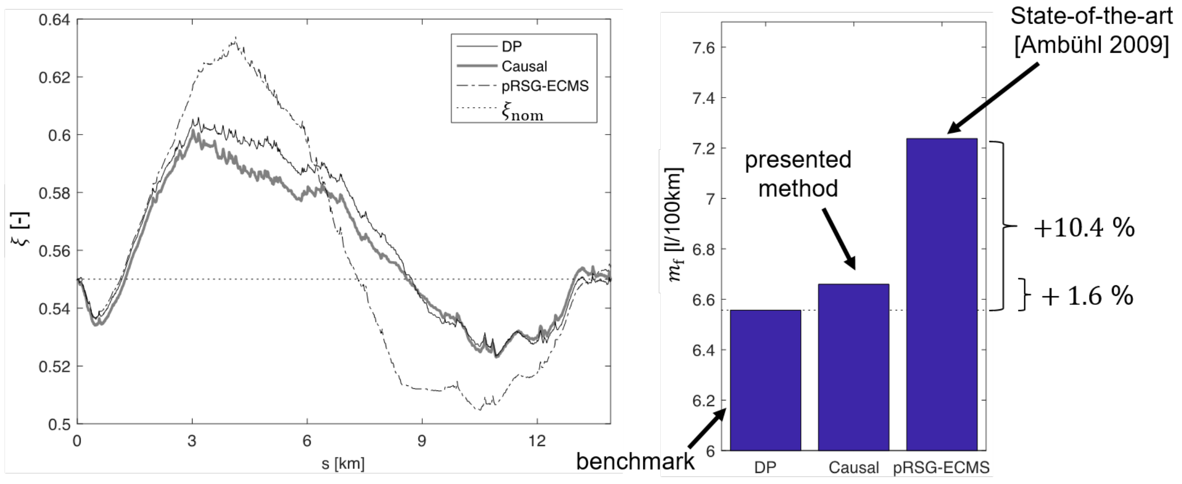

Figure 8 - Performance analysis of the online energy management controller. Left: State-of-charge trajectory over the driving mission. Right: Resulting fuel consumption. The non-causal benchmark is denoted DP, the method developed in this work is denoted causal, and the state-of-the-art method is denoted pRSG-ECMS.

3.2 Energy and Emissions Management

The second subproblem considers pollutant emissions. As a result, the ATS needs to be considered and the task of the resulting control problem is to perform an energy and emissions management of the hybrid powertrain. In the following, an overview of the work originally developed in the master's thesis [16] and resulting in the publication [17], and the patent [18], is given. Further details can be found in the aforementioned works.

While the inputs of the extended problem are the same as before, i.e., the gearbox variable i and the torque split u, two further states must be considered, i.e., the ATS temperature and the accumulated tailpipe NO_x mass. We use a single thermal state to model the ATS. The DeNOx efficiency of the ATS is related to the ATS temperature and the exhaust mass flow by a static look-up map. The engine-out NO_x mass flow, the exhaust temperature, and the exhaust mass flow are related to the engine speed and torque by static look-up maps. We solve the resulting non-convex mixed-integer optimization problem using DP to obtain a benchmark.

As a result of the two additional states, two additional costates appear in the Hamiltonian. Using PMP, we found that the costate corresponding to the battery state of charge can be treated as before and is piecewise constant if the dependency of the battery losses on the battery state of charge is neglected. Furthermore, the costate corresponding to the accumulated tailpipe NO_x mass is also constant, as the current tailpipe NO_x mass flow does not depend on the accumulated tailpipe NO_x mass. This costate can be interpreted as an equivalence factor describing the fuel-equivalent cost of the tailpipe NO_x mass flow. The final costate corresponds to the ATS temperature and is not constant, as the derivative of the ATS temperature depends on the current ATS temperature. A single-shooting method, as applied in the previous section, is no longer sufficient to solve the resulting optimization problem.

To guarantee that the proposed optimization method finds the global optimum a simplified model was introduced in [17], in order to facilitate the formulation of a mixed-integer convex problem. The resulting OCP is solved using the proposed Iterative Convex Optimization (ICO) algorithm, which splits the mixed-integer optimization into an integer optimization dealing with the integer variables and a convex optimization dealing with the continuous variables. By iterating over the two methods, the solution that minimizes the cost of both optimizations is sought. Figure 9 shows an overview of the ICO algorithm. The Convex Optimization block solves the convex optimization problem for given integers. The Integer Optimization block minimizes an approximation of the Hamiltonian function for the damped costates previously calculated by the convex optimization. We initialize the algorithm using the solution of the energy management problem, which neglects the ATS and emissions. Once the algorithm has converged, it is stopped.

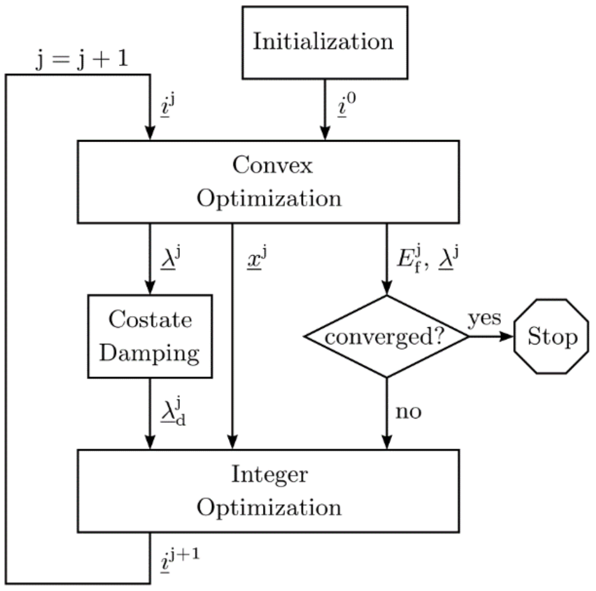

Figure 9 - Overview of ICO algorithm.

To evaluate the performance of the proposed algorithm in the absence of model mismatch, we first compare it to a DP benchmark obtained on the simplified model. As the ICO algorithm does not suffer from a quantization error, it outperforms the DP benchmark and reduces the amount of fuel consumed by 0.24% for an equivalent accumulated tailpipe NO_x mass and an equivalent final battery state of charge. We conclude that the ICO algorithm converges to the global optimum.

Unfortunately, further analysis and testing of the algorithm has shown that cases exist where no robust convergence can be achieved. While some adaptations of the algorithm could alleviate the issue, no robust tuning of the costate damping was found that results in reliable convergence for different missions and the optimization method was not employed further.

4 Conclusion and Outlook

The dynamic programming algorithm dpa developed in the scope of the MERFAS project was shown to retain all the functionalities of its predecessor dpm. It further allows optimization problems to be solved using less memory and resulted in shorter computation times. The algorithm was successfully applied to solve larger problems with multiple states and inputs, both in the scope of the MERFAS project as well as in other projects.

The prestudies performed for the Diesel-electric hybrid vehicle resulted in an efficient optimization method for the energy management problem when neglecting pollutants. Based on this method, a close-to-optimal predictive supervisory controller was developed and evaluated in simulation. When considering pollutant emissions and the ATS dynamics, the OCP becomes harder to solve. While the developed iterative convex optimization method could be successfully applied to individual problems, its strong sensitivity to the mission, currently disqualifies the method from an application in a predictive supervisory controller.

From the results achieved in this work, there are two avenues for future research. First, the energy management controller developed in Section 3.1 could be implemented on an actual vehicle for which pollutant considerations are of less importance or are dealt with using a different control method. Second, a more in depth investigation into the convergence issue of the iterative convex optimization method presented in Section 3.2 could be performed to better understand and overcome its shortcomings.

References

[1] J. Asprion, "Optimal control of Diesel engines," PhD Thesis, ETH Zurich, 2013.

[2] O. Sundström and L. Guzzella, "A generic dynamic programming Matlab function," *Control Applications,(CCA) & Intelligent Control,(ISIC), 2009 IEEE,* pp. 1625-1630, 2009.

[3] O. Sundström, D. Ambühl and L. Guzzella, "On implementation of dynamic programming for optimal control problems with final state constraints," *Oil & Gas Science and Technology–Revue de l'Institut Français du Pétrole,* vol. 65(1), pp. 91-102, 2010.

[4] P. Elbert, S. Ebbesen and L. Guzzella, "Implementation of dynamic programming for n-dimensional optimal control problems with final state constraints," *IEEE Transactions on Control Systems Technology,* vol. 21(3), pp. 924-931, 2013.

[5] L. Engbroks, D. Görke, S. Stefan, J. Strenkert and B. Geringer, "Analytic Solution to the Dynamic Programming sub-problem in Hybrid," in *5th IFAC Conference on Engine and Powertrain Control, Simulation and Modeling*, Changchun, China, 2018.

[6] J. van Schijndel, M. C. F. Donkers, F. P. T. Willems and W. P. M. H. Heemels, "Dynamic programming for integrated emission management in diesel engines," in *IFAC Proceedings Volumes*, Cape Town, South Africa, 2014.

[7] R. Belman, Dynamic Programming, Rand Corp. Santa Monica CA, 1956.

[8] D. P. Bertsekas, Dynamic programming and optimal control, Athena Scientific, Belmont, MA, 1995.

[9] J. Ritzmann, O. Chinellato, R. Hutter and C. Onder, "Optimal Integrated Emission Management through Variable Engine Calibration," *Energies,* vol. 14, no. 22, 2021.

[10] D. Machacek, K. Barhoumi, J. Ritzmann, T. Huber and C. Onder, "Multi-level model predictive control for the energy management of hybrid electric vehicles including thermal derating," *IEEE Transaction on vehicular technologies,* (submitted).

[11] A. Christon, "Model predictive control of hybrid electric vehicles," Master's Thesis ETH Zurich, Zurich, 2019.

[12] J. Ritzmann, A. Christon, M. Salazar and C. Onder, "Fuel-optimal power split and gear selection strategies for a hybrid electric vehicle," *SAE Technical Paper,* 2019.

[13] J. Ritzmann, O. Chinellato, A. Christon, S. Mauro and C. Onder, "A method and a system for controlling a vehicle on a mission". Patent EP3756963, 30 12 2020.

[14] V. Boltyanski, R. Gamkrelidze, E. Mishchenko and L. Pontryagin, "The maximum principle in the theory of optimal processes of control," *IFAC Proceedings,* vol. 1, p. 464–469.

[15] D. Ambühl, "Energy management strategies for hybrid electric vehicles," PhD Thesis, ETH Zurich, 2009.

[16] G. Lins, "Integrated energy and emissions management of HEVs," Master's Thesis ETH Zurich, Zurich, 2019.

[17] J. Ritzmann, G. Lins and C. Onder, "Optimization method for the energy and emissions management of a hybrid electric vehicle wih an exhaust aftertreatment system," *IFAC-PapersOnLine,* vol. 53.2, pp. 13797-13804, 2020.

[18] J. Ritzmann, O. Chinellato, G. Lins, S. Mauro and C. Onder, "A method and a system for controlling a vehicle on a mission". Patent WO2021229526, 18 11 2021.

[19] M. Werling, Integrated Trajectory Optimization. In: Winner H., Hakuli S., Lotz F., Singer C. (eds) Handbook of Driver Assistance Systems., Springer, Cham. , 2016.