Evaluation of mixed integer nonlinear programming routines

Evaluation of Mixed Integer Nonlinear Programming routines

J. De Wit
s436710

DCT 2005.42

Traineeship report

Coaches: ir. M.W.T. Koot
dr. ir. A.G. de Jager

Supervisor: prof. dr. ir. M. Steinbuch

Technische Universiteit Eindhoven
Department of Mechanical Engineering
Dynamics and Control Technology Group

Eindhoven, September, 2005

Abstract

In future road vehicles an increase in electric power consumption will occur. To prevent an similar increase in fuel consumption, smart strategies for the generation, storage/retrieval, distribution and consumption of electric power are needed. This report considers a dual storage power net, where a genarator is connected to a discrete switching device. This switch divides the electric power to the battery or to a super capacitor. These two storage devices are connected through a DC/DC converter. The optimization problem, which finds the minimal fuel consumption and the switch position for a given trajectory, result in a Mixed Integer NonLinear Programming problem (MINLP).
Mixed Integer NonLinear Programming problems are difficult to solve for large numbers of variables, in particular for large numbers of discrete variables. This report deals with three MINLP techniques, Branch and Bound (BNB), Outer Approximation (OA) and Generalized Benders Decomposition (GBD). These techniques are tested on a vehicle model, which uses a switch, a battery and a supercap to store electrical power.
The BNB technique is able to solve the MINLP problem for a certain problem size, whereas the other techniques are not able to solve the problem within limited iteration and time. But even the BNB technique requires a lot of iterations and time to solve larger problems.
Therefore other techniques, other implementations or solvers should be considered to solve large MINLP problems.

Contents

Chapter 1

Introduction

Present day consumers demand more performance, comfort and safety from a road vehicle. To satisfy the consumers, future vehicles will be increasingly equipped with electronic devices. Such devices are air conditioning or climate control, electronic windows, sound systems, electronic steering, drive by wire, cruise control, driving assistance, stability programs et cetera. These electronics consume electric power. Over the past twenty years the electric power consumption in standard road vehicles has increased approximately four percent every year. In the near future even higher power demands are expected [1], [2].

To keep up with future power demands, the automobile industry has suggested new 42 V power net topologies, which should replace the traditional 14 V, which are used currently [1], [2]. These power nets will be able to meet the future demands, but this will coincide with an increase in fuel consumption. Future vehicles have to meet strict requirements about exhaust emissions. With the rising prices of fuel, costumers will demand fuel economic vehicles. Therefore smart strategies for the generation, storage/retrieval, distribution and consumption of electric power are needed. But the driver should not experience different vehicle behavior as a result of such strategy. This implies that the available torque and power should be present when the driver wants it.
In this report a vehicle power net is considered, where a generator is connected to a switching device that divides the electric power between a super capacitor (supercap) and a battery. This switch is discrete. The battery and the supercap are connected to a DC/DC converter.
When the speed trajectory of the car is known, the minimal fuel consumption can be calculated using optimization. To solve this problem, a mixed integer nonlinear optimization routine is needed.
The aim of this internship is to find available techniques and select the most suitable for the given problem. This is done by implementation and testing these techniques in a Matlab environment. Literature will provide existing techniques, these techniques will be adapted to be able to solve the problem.

The remainder of this report is build up as follows: Chapter 2 describes the model of the vehicle and the loads a vehicle is subjected to. Chapter 3 gives an overview of the mixed integer nonlinear programming techniques. In Chapter 4 these techniques will be implemented and compared. Chapter 5 will summarize the results of this research.

Chapter 2

The model

The strategy, discussed in this report, is focussed on the generation, storage/retrieval and distribution of the electronic power. It considers an advanced dual storage net. In the next section the vehicle model and the advanced dual storage net will be described.

2.1 The vehicle model

The advanced dual storage net consists of a generator, a battery, a supercap, a DC/DC converter and an internal combustion engine (ICE). The power flow starts with the combustion of fuel. The mechanical power is split in two directions: one part goes to the drive train (DT) and the other part is used by a generator (GEN), which generates electrical power. The electrical power is connected to a discrete switch (S). The power can be sent to the supercap (C) or to the battery (B). The load (Load), the electric power that is needed, is connected to the battery. The supercap and the battery are connected to a DC/DC converter (DC). A schematic drawing can be seen in Figure 2.1.

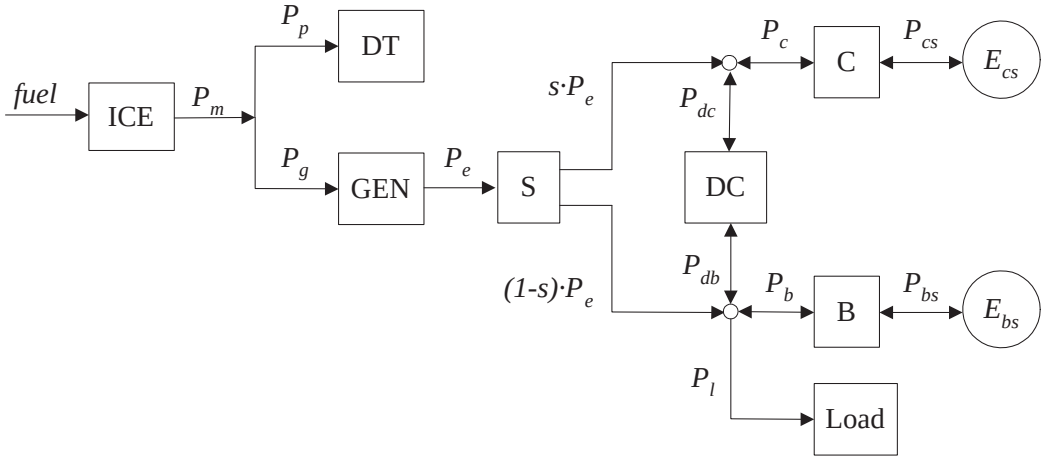

Figure 2.1: Power flow in a dual storage power net

A supercap can be considered as a very large capacitor. A battery can operate between 20% and 100% SOC (= State Of Charge, the relative energy level), while maintaining an acceptable board net voltage, whereas a supercap can only be operated between 80% and 100% SOC. The open cell voltage is linear with the energy level for the supercap. The open cell voltage of the battery is assumed to be linear between 20% and 100% SOC. The battery has a much bigger storage capacity, but the losses of the supercap are much smaller, than the losses of the battery. These losses will be assumed to be quadratic with the power. Therefore the losses will be less, when low power is delivered for a long period of time, then a high power for a shorter period.
When the generator delivers high powers (mainly during deceleration phases), the energy will be stored in the supercap. From there the energy will be transferred at a lower power through the DC/DC converter to the battery or the load. With this strategy the energy losses will be reduced.

The model structure and the component models can be found in [4]. The report describes the mathematical formulation of the model and its constraints. This report is summarized in Appendix A.

2.2 The optimization problem

The mechanical power needed for the drive train and the electric load will be prescribed. Therefore the model can structured such that there will be three design variables: P_{cs}, the power stored in the supercap, P_{bs}, the power stored in the battery and s, the position of the switch.
These three variables will be the input of the cost function. The output of the cost function will reflect the fuel consumption, which has to be minimized.
The cost function becomes a high order polynomial. This polynomial is a smooth, nonlinear function, which is most likely to be nonconvex. Because the cost function contains both continuous and discrete variables, the optimization problem becomes a NonLinear Mixed Integer Programming problem:

$$\begin{aligned} &\min_{x,y} \; f(P_{bs}, P_{cs}, s) \\ &s.t. \; \mathbf{A}\underline{x} \;\; \leqslant \;\; \underline{b} \\ &\quad \mathbf{A}_{eq}\underline{x} \;\; = \;\; \underline{b_{eq}} \\ &(P_{bs}, P_{cs}) \; \epsilon \; \Re^{2n}, s \; \epsilon \; \{0,1\}^n \end{aligned} \tag{2.1}$$

Chapter 3

Mixed integer nonlinear programming routines

This chapter describes three mixed integer nonlinear programming (MINLP) techniques: Branch and Bound (BNB), Outer Approximation (OA) and Generalized Benders Decomposition (GBD). The overview is restricted to solvers for convex MINLP. These techniques are used to solve the nonlinear discrete optimization problem described in the previous Chapter. A unified overview and derivation of the MINLP techniques can be found in [3], [6] and [7].

3.1 MINLP

The basic mathematical description of a MINLP problem is as follows:

$$\begin{aligned} &\min_{x,y} \ f(x,y) \\ s.t.\ &g_j(x,y) \ \leqslant \ 0, \ j \in J \\ &x \in \Re^n, \ y \in \mathbf{Z}^m \end{aligned} \tag{3.1}$$

where $f(\cdot),\ g(\cdot)$ are convex, differentiable functions, J is the index set of inequalities and x and y are the continuous and discrete variables, respectively.

3.2 Branch and Bound

This method first solves the continuous nonlinear programming (NLP) relaxation, the discrete variables are considered continuous (the NLP subproblem). After this the program generated subproblems, where the domain of the variable (still continuous) is being restricted. This is called branching. Then it solves these subproblems. This process continues until the variable is fixed to a (integer) value.
The advantage of this approach (when compared with explicit enumeration) lies in the fact that not all subproblems have to be solved.
The BNB method is generally only attractive, if the NLP subproblems are relative inexpensive to solve or when only a few of them need to be solved. This could be either because of the low dimensionality of the discrete variables or because the integrality gap of the continuous NLP relaxation of the NLP subproblem is small. If possible it is recommended to approximate the

nonlinear cost function by a quadratic cost function, this will reduce the calculation effort of finding the local minimum [6] [8].

3.3 Outer Approximation

The OA method solves in a cycle of iterations a mixed integer linear programming (MILP) problem and a relaxed NLP, with fixed integer variables. The local minimum of the NLP gives a solution (x^k,y^k) for the continuous variables. The corresponding cost function and the nonlinear constraints are then linearized around this solution. α Is the value of the linearized cost function. This will result in a linear inequality constraint for the MILP.

$$\begin{aligned} &\min_{x,y} \ \alpha \\ &s.t.\ \alpha \geq f(x^k,y^k) + \bigtriangledown f(x^k,y^k)^T \left[\begin{array}{c} x-x^k \\ y-y^k \end{array} \right] \\ &\qquad g_j(x^k,y^k) + \bigtriangledown g_j(x^k,y^k)^T \left[\begin{array}{c} x-x^k \\ y-y^k \end{array} \right] \leq 0,\ j \,\epsilon\, J \Bigg\} k = 1,2,\ldots \\ &\qquad x^k \,\epsilon\, \Re^n,\ y^k \,\epsilon\, \mathbf{Z}^m \end{aligned} \tag{3.2}$$

Rearranging this constraint, results in an linear inequality constraint, that can be used to solve the MILP subproblem:

$$\left[\begin{array}{cc} \bigtriangledown f(x^k,y^k)^T & -1 \\ \bigtriangledown g_j(x^k,y^k)^T & 0 \end{array} \right] \left[\begin{array}{c} \left[\begin{array}{c} x \\ y \end{array} \right] \\ \alpha \end{array} \right] \leq \left[\begin{array}{c} \bigtriangledown f(x^k,y^k)^T \left[\begin{array}{c} x^k \\ y^k \end{array} \right] - f(x^k,y^k) \\ \bigtriangledown g_j(x^k,y^k)^T \left[\begin{array}{c} x^k \\ y^k \end{array} \right] - g_j(x^k,y^k) \end{array} \right] \tag{3.3}$$

This can be written as:

$$\mathbf{A}\underline{x} \leq \underline{b} \tag{3.4}$$

An updated version of this constraint is added at every cycle to the MILP subproblem. Therefore at every cycle one linear constraint from the linearized cost function and j linearized nonlinear constraints are addded. The cost function and the nonlinear constraints, used in the MILP subproblem, are linearized functions in the minimum of the NLP subproblem [6].
The cycle of iterations will start with the MILP subproblem. The cost function is linearized around the initial conditions. The output of this optimization is the intersection between the linear bounds and the linearized function. After the MILP subproblem the relaxed NLP subproblem is solved. At this subproblem feasible solutions are found, whereas the MILP subproblem only results in feasible solutions with regard to the discrete variables. The resulting variables are used as the starting points for the next iteration.
The OA method should reduce the number of NLP problems, that have to be solved.

3.4 Generalized Benders Decomposition

The GBD method is similar to the OA method. The difference arises in the definition of the MILP. The GBD method only considers active constraints of the NLP problem. By using the $Karush-Kuhn-Tuckerconditions$ and eliminating the continuous variables, the inequalities can be reduced to a single linear constraint [6].

$$\min_{x,y} \ \alpha$$

$$\alpha \geq f(x^k, y^k) + \bigtriangledown_y f(x^k, y^k)^T (y - y^k) + (\mu^k)^T [g(x^k, y^k) + \bigtriangledown_y g(x^k, y^k)^T (y - y^k)] \quad (3.5)$$

$$x^k \, \epsilon \, \Re^n, \ y^k \, \epsilon \, \mathbf{Z}^m$$

In equation 3.5 $g(x, y)$ represents the constraint function and μ represents the Lagrange Multiplier. Rearranging the equation and inserting the continuous variables results in the following equation.

$$\left[\begin{array}{ccc} 1 & \bigtriangledown_y f(x^k, y^k)^T & -1 \end{array} \right] \left[\begin{array}{c} x \\ y \\ \alpha \end{array} \right] \leq (\sum x^k) - f(x^k, y^k) + \bigtriangledown_y f(x^k, y^k)^T y^k - (\mu^k)^T g(x^k, y^k) \quad (3.6)$$

This inequality constraint is added at every iteration to the other inequality constraints, which results in.

$$\mathbf{A}\underline{x} \leq \underline{b} \quad (3.7)$$

Therefore the computational cost for solving the MILP subproblem of the GBD method is less then the MILP subproblem of the OA method since only one constraint is added per iteration [6]. So the MILP subproblem has less constraints the the OA method, but usually more iterations are needed. The MILP problem is then solved by using a linearized function of the cost function and the new equality constraints. The cycle is then repeated with resulting variables as input for NLP problem.

3.5 Conclusion

BNB is useful, when there are not many integer variables or when the the integrality gap between the discrete and continuous subproblem is small, otherwise a lot of NLP's have to be solved.
The OA method is advantageous compared to Branch and Bound, when the NLP takes much computation time, because less iterations are needed
In the GBD method only active constraints are considered this will reduce the complexity of the NLP subproblem and therefore the computation time. The disadvantage is that more cycles of iteration are needed to solve the subproblem.
When the computational costs of the NLP subproblem are high, the OA and GBD method are recommended. The trade off between the OA and GBD method is determined by the number of times the NLP has to be solved.

Chapter 4

Comparison of the optimization routines

In this Chapter the routines will be compared. Therefore a part of the drive cycle will used. The resulting discrete variables represent the usage of the switch. This is needed to be able to validate the vehicle model. The fuel consumption will then be compared with a model with a continuous switch and a model without a switch, DC/DC converter and a supercap. (x^k,y^k) The different routines will be evaluated on the time, that is needed to find a solution for the problem and the output, which contains the fuel consumption and the corresponding variables.

First the drive cycle and the loads, that are used for the tests are discussed. The next section deals with the implementation of the optimization routines, which are discussed in Chapter 4. Then the results will be summarized. In the fourth section of this Chapter the results will be compared with the results of the model with a continuous switch and the model without the supercap. This is done to evaluate the model with respect to the discrete switch. (x^k,y^k) The Matlab scripts, that are used for the implementation, are shown in Appendix B.

4.1 The drive cycle and loads

This section will deal with the mechanical power for the drive train and the electrical power, that are needed to operate the vehicle. The artificial data, that is used to describe the mechanical and electrical power, is chosen such that the switch will switch several times. The mechanical and electrical power is represented by simple sinusoidal waves. So there will be sections, where it is expensive to generate electrical power and section, where it will be cheaper to generate electrical power. This data is used to test the performance of the model and more important the performance of the optimization routines.

First two periods of a sinusoid are distributed over 20 seconds, which results in 60 variables for the BNB method and 61 for OA and GBD method (n integer plus 2·n+1 continuous variables). Next the same waves are distributed over 30 seconds, which results in 90, 91 and 91 variables respectively. As last two periods are distributed over 40 seconds, therefore there are 120, 121 and 121 variables respectively. (x^k,y^k) Figure 4.1 shows the mechanical and electrical power over a period of 40 seconds. Appendix B.1 shows all the loads, which are used for the tests.

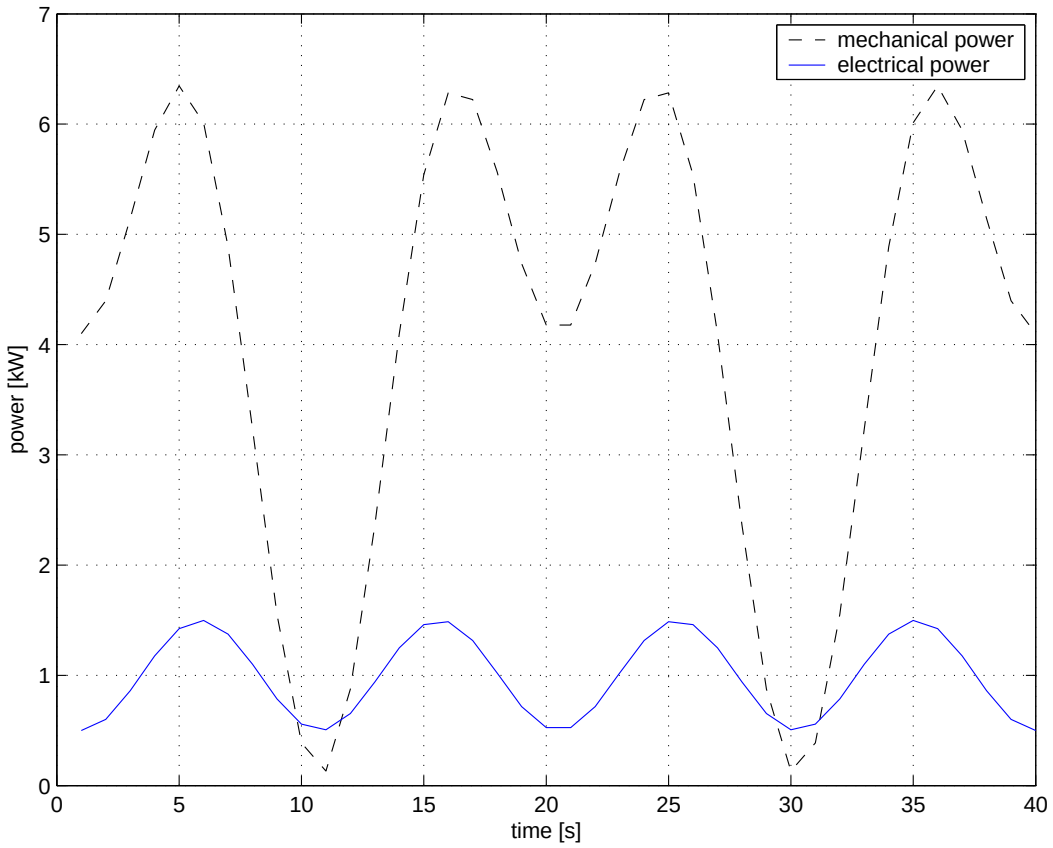

Figure 4.1: Mechanical and electrical power

4.2 Implementation of the routines

The computer, that is used to solve the optimization problems, is an DELL *Dimension 2400* desktop with an Intel Pentium 4 processor 2.8 GHz and 512 MB of RAM.

4.2.1 Branch and Bound

The BNB method used for this research, is $bnb20.m$ (Revision: 1.20), which is an algorithm implemented in MATLAB and is designed by E.C. Kuipers, Applied Physics, Rijksuniversiteit Groningen. The nonlinear subproblems are solved with $fmincon.m$ from the MATLAB Optimization Toolbox [9].

options for $fmincon.m$

- TolX = 10^{-6}
  Termination tolerance on X, where X are the variables corresponding to the minimum function value. The lower the value, the more accurate the solution will be, but it will also increase the computational costs.
- TolFun = 10^{-6}

Termination tolerance on the function value. The value is a trade off between accuracy and computational costs.

4.2.2 Outer Approximation

There are no nonlinear constraints in this specific optimization problem, therefore equation 3.3 can be reduced to:

$$\min_{x,y} \ \alpha$$

$$\left[\ \bigtriangledown f(x^k, y^k)^T \quad -1 \ \right] \left[\begin{bmatrix} x \\ y \end{bmatrix} \right] \leq \bigtriangledown f(x^k, y^k)^T \begin{bmatrix} x^k \\ y^k \end{bmatrix} - f(x^k, y^k) \tag{4.1}$$

This can be written as:

$$\mathbf{A}\underline{x} \leq \underline{b} \tag{4.2}$$

Therefore *one* constraint will adeded at every iteration.

Solvers

For the MILP subproblem the MATLAB [9] version of lp_solve 5.1.1.3 [10] is used, a free solver originally developed by $Michel\ Berkelaar$ at Eindhoven University of Technology. This solver is based on the revised simplex method and the Branch-and-bound method for the integers. The relaxed NLP subproblem is solved with $fmincon.m$ from the Matlab Optimization Toolbox.

Optimization options

- TolX = 10^{-6}
  Termination tolerance on X, where X are the variables corresponding to the minimum function value.
- TolFun = 10^{-6}
  Termination tolerance on the function value.
- MaxSQPIter = 50
  Maximum number of Sequential Quadratic Programming iterations. This gives a maximum of to the number of computation to solve the subproblem.

Stopping criterium

The resulting cost function value of NLP subproblem has to converge to the resulting value of the MILP subproblem. The differences between the values has to be less then some tolerance value. For this case the difference should be less then $5 \cdot 10^{-3}$.
The corresponding discrete variables are the same for both subproblems, because the output of the MILP subproblem is used for the NLP subproblem, at which the discrete variables are fixed. Therefore only the continuous variables will change within a cycle, but the solution of the MILP subproblem is the solution of the linearized problem, which does not have to be feasible. Therefore only the cost function value of both subproblems have to correspond and then the iteration is terminated. The maximum number of cycles is limited to 400 as back-up.

4.2.3 Generalized Benders Decomposition

This specific problem does not change the formulation of the constraints for the MILP subproblem. Therefore equation 3.5 is implemented.

Solvers

For the GBD routine the same solvers as for the OA routine are used. Also the same optimization options, as in at the BNB and OA routines, are used. Therefore the routines can be compared.

Stopping criterium

Contrary to the OA method, the GBD method will always give a feasible solution for the NLP subproblem, due to the used Karush-Kuhn-Tucker conditions. Therefore the resulting continuous design variables can be used as an additional stopping criterium. Together with the cost function values of the subproblems the stopping criteria will be able to stop, when the subproblems have found the same optimum (the same cost function value within some margin). As back-up the maximum number of cycles is limited to 400.

4.3 Results of the routines

The results of the tests are shown in the next tables. In Appendix B the course of the iterations during the tests of the OA and GBD methods are shown.

4.3.1 Branch and Bound

no. of variables	no. of discrete variables	starting point	value	no. of iterations	no. of possible nodes	duration
60	20	zeros	555.2706931	29	10^6	12.9 s
60	20	ones	555.2706931	29	10^6	12.8 s
90	30	zeros	846.6874144	55	10^9	55.8 s
90	30	ones	846.6874141	55	10^9	55.4 s
120	40	zeros	1123.1755111	221	10^{12}	5.8 min
120	40	ones	1123.1755111	221	10^{12}	5.9 min

Table 4.1: Results of Branch and Bound

Table 4.1 shows that the BNB method is able to solve the MINLP problem. Regardless of the used starting points, the method needs the same amount of time and cycles to find the minimum. The calculation time increases rapidly with the number of variables.

4.3.2 Outer Approximation

no. of variables	no. of discrete variables	starting point	value	no. of nodes or iterations	duration
61	20	zeros	555.2705934	40	54.1 s
61	20	ones	555.2705934	41	54.5 s
91	30	zeros	846.6874142	115	11.8 min
91	30	ones	846.6874144	119	10.8 min
121	40	zeros	1124.4130679	> 186	> 4.5 hours
121	40	ones	1123.5838414	> 259	> 8.7 hours

Table 4.2: Results of Outer Approximation

The OA method solves the first two problems successfully, but the starting points influence the required time and number of iterations. The third test, with 121 variables, is too large for the used NLP and MILP toolboxes. The method is unable to solve the problem within a reasonable time. The optimizations was terminated. Table 4.2 makes it also clear, that calculation time is increased considerable with the number of variables.

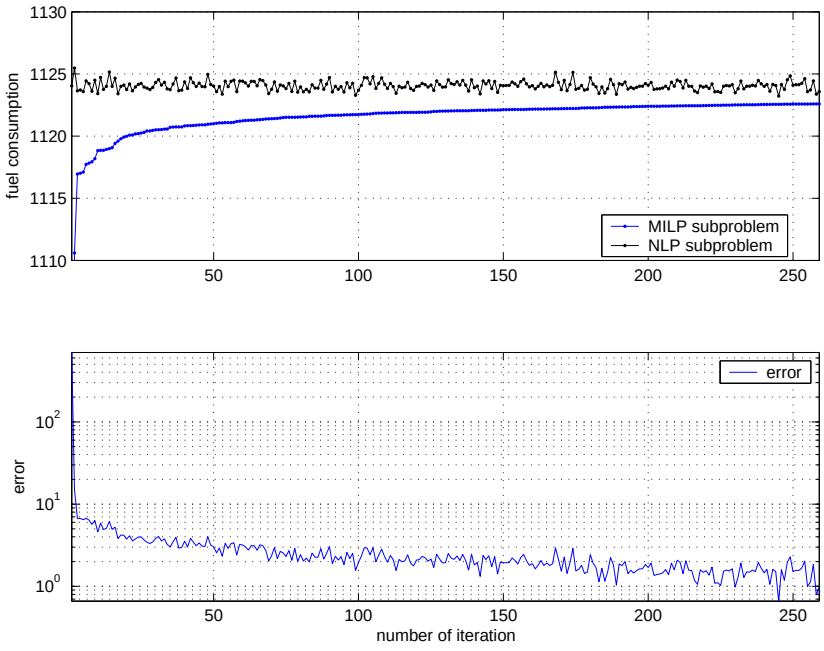

Figure 4.2: Data Outer Approximation, period of 40 seconds, ones as starting points

Figures 4.2 shows the course of the optimization and the errors for the test with 121 variables. The course is represented from the bottom up by the value of the linearized problem (MILP) and the value of the NLP subproblem. It shows clearly, that this routine is still converging to a solution, but as discussed above, it will take a lot of time before a solution with a desired accuracy is found.
Figures B.4 to B.9, in Appendix B.2, show the courses of the optimization and the errors at every cycle.

4.3.3 Generalized Benders Decomposition

no. of variables	no. of discrete variables	starting point	value	no. of nodes or iterations	duration
61	20	zeros	555.2705932	40	40.9 s
61	20	ones	555.2705931	40	39.5 s
91	30	zeros	846.6874143	105	3.03 min
91	30	ones	846.6874143	102	2.95 min
121	40	zeros	1123.7816152	400 (=max)	46.30 min
121	40	ones	1123.5956219	400 (=max)	45.40 min

Table 4.3: Results of Generalized Benders Decomposition

This method is also able to solve the first two problems, but it was not able to solve the third problem within the maximum number of iterations (400). Besides that, this method, with the used solvers, needs a large amount of time to solve this test. Figure 4.3 shows the test with 121 variables. It can be seen that the routine is still converging to the solution, but it will take a considerable number of iterations and time to find an accurate solution.
Table 4.3 shows that the time, that is required to solve the problem, increases rapidly with the increasing number of variables.

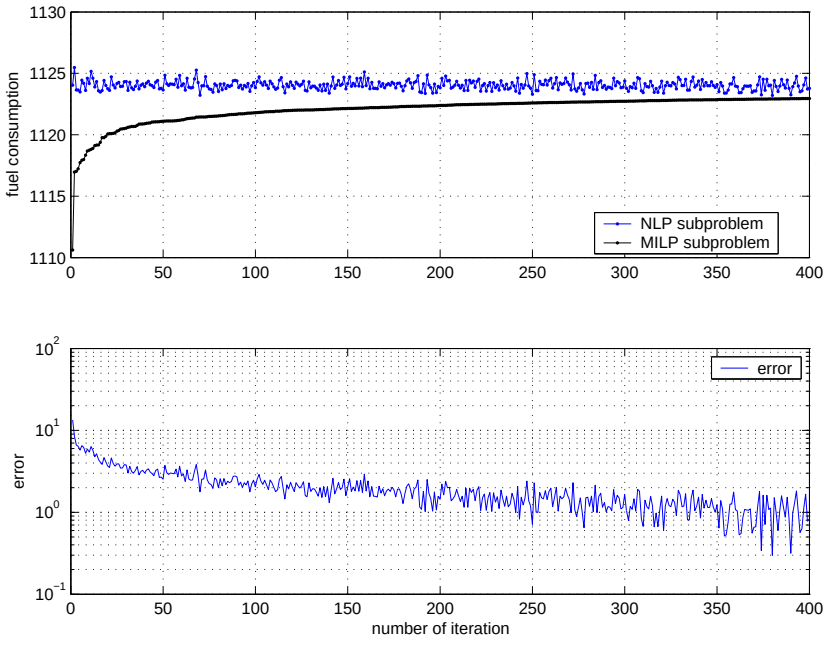

Figure 4.3: Data Generalized Benders Decomposition, period of 40 seconds, zeros as starting points

Figures B.14 and B.15, in Appendix B.3, show the errors between the values of the subproblems. These figures also show that a considerable number of iterations is needed to solve the third problem.

4.3.4 Conclusion

The Tables, 4.1 to 4.3, show clearly that the BNB method performs better then the other methods. The BNB method is able to solve all of the given problems and solves them faster. But even the BNB method is limited by the number of variables.
In this specific problem the integrality gap of the continuous NLP relaxation is small, this will reduce the number of NLP subproblems that have to be solved. Therefore the BNB method is an attractive method for this problem.
The large amount of time that is needed to solve the MINLP problem with the OA and GBD techniques can be explained by the slow converging rate of these techniques. Therefore a large amount of iterations are needed to solve the problem. To solve larger problems, faster techniques and/or solvers are needed.
Appendix B.4.1 to B.4.3 show the variables of solutions of every test. It can be seen for the tests, where all routines were able to find a solution (periods of 20 and 30 seconds), the solutions are similar

4.4 Different models

In this section the model with the discrete switch is evaluated, by comparing it with two other models: a model with a continuous switch and a model without the supercap.

4.4.1 Model with a continuous switch

Table 4.4, shows that use of a continuous switch does decrease the fuel consumption. This can be explained by the fact that the supercap is used optimal with a continuous switch. The figure in Appendix B.4.3 show the positions of the discrete switch and continuous switch for every test.

no. of variables	starting point	value
60	zeros	553.9269335
60	ones	553.9269179
90	zeros	845.5139630
90	ones	845.5139630
120	zeros	1120.7407429
120	ones	1120.7407425

Table 4.4: Results of the model with a continuous switch

4.4.2 Model without the supercap

Table 4.5 shows the results of the tests, with the model without the supercap, the DC/DC converter and the switch. These results show that usage of a supercap will reduce the fuel consumption.

no. of variables	starting point	value
60	zeros	581.7406253
60	ones	581.7406253
90	zeros	876.2941362
90	ones	876.2941362
120	zeros	1170.8537338
120	ones	1170.8537339

Table 4.5: Results of the model without a supercap

4.4.3 Conclusion

The first two tests show, that the models with a switch and a supercap will reduce the fuel consumption. This can also be seen in Table 4.6. This table shows the results of the BNB routine and the results of the tests with the model without a supercap. The reduction of the fuel consumption is about 4 %.

no. of variables	with supercap	without supercap	reduction by supercap
60	555.2706931	581.7406253	4.55 %
90	846.6874144	876.2941362	3.38 %
120	1123.1755111	1170.8537338	4.07 %

Table 4.6: Fuel consumption supercap vs. no supercap

4.5 Other Solvers

The NLP solver requires a lot of calculation time to solve each subproblem. All methods used the $fmincon.m$ from the Matlab Optimization Toolbox, therefore the solver itself does not cause the differences between the methods. The differences arise by number of times the solver is required. This effect is enhanced, when the number of variables is increased.
To obtain a shorter computation time, other solvers can be explored. The following subsections briefly describe a few sources, where other solvers can be found.

4.5.1 TOMLAB

TOMLAB [11] is a general purpose development environment in Matlab for research, teaching and practical solution of optimization problems. It is developed to fulfill the need for advanced, robust and reliable tools to be used in the development of algorithms and software for the solution of applied optimization problems. It claims to be flexible, easy-to-use, robust and reliable for the solution of all types of applied optimization problems.
TOMLAB is compatible with the Matlab Optimization Toolbox 2.1. TOMLAB supplies a large collection of Matlab solvers as well as software packages. TOMLAB provides algorithms, which are able to solve Mixed Integer Linear, Quadratic and Nonlinear Programming problems, $TOMLAB/MINLP_{bb}$ $v1.2$. This solver uses a BNB algorithm. Furthermore it provides Non-

linear Programming routines. So TOMLAB can be used to solve the MINLP problem with the routines, which are described in this report.

4.5.2 LINDO API

LINDO Systems has developed an Application Programming Interface, called LINDO API. This interface can be used with Matlab. LINDO Systems assures that it can solve large scale NLP problems, whereas the Matlab solver, $fmincon.m$, is not a large scale solver [5]. The report of B. Zijlstra [5] describes the usage and performance of Matlab as an interface for LINDO API. The free trail version was too limited for the implementation of the problem, therefore no tests are conducted with this solver.

4.5.3 AMPL

AMPL is a comprehensive and powerful algebraic modelling language for linear and nonlinear optimization problems, in discrete or continuous variables [13]. AMPL, developed by BELL Laboratories, uses common and familiar concepts to formulate the optimization problem. The algorithm is able to manage the communication with an appropriate solver.
The modelling language is readable for offline and online solvers. Some of those online solvers can be found at the neos server.
`http://www-neos.mcs.anl.gov/neos/solvers/NCO:IPOPT/solver-www.html`
The access to this server is free and their are no limits on the size of the problem. But due to limited time, the problem was not implemented in this environment.

Chapter 5

Conclusions and recommendations

In this report several MINLP routines, namely Branch and Bound, Outer Approximation and Generalized Benders Decomposition, were evaluated for an energy management problem of a vehicle with a dual power storage net. The results show that, with these implementation, the Branch and Bound method is the best method for this specific problem. It is able to solve all the problems within a short time in comparison to the other methods. The other methods are not able to solve all the problems in limited time and they needed a lot more computational time to find the minimum of the given problems.
The NLP solver that is used for the tests, requires a lot of calculation time to solve the subproblems. The calculation time increases rapidly with an increase of the number of variables. So for larger problems, other solvers are needed.
These results are determined by three issues of this optimization problem. The problem description determines the type of the optimization problem, other ways of specifying the problem can result in other optimization types and in different types of problems occurring during an optimization. The second issue is the optimization methods that is used. Each techniques uses specific characteristics of a problem, therefore every method will have its specific advantages and disadvantages. The third issue is the implementation of the techniques for instance the type of solvers and settings, that are used and the environment in which the techniques are programmed. In this report only the techniques are discussed for a given problem description.
Therefore other techniques, other implementations or solvers should be considered to solve large MINLP problems.

Bibliography

[1] M. Koot, J. Kessels, and B. de Jager. Energy management in a vehicle with a dual storage power net. In *Proc. of the 16th IFAC World Congress*, Prague, Czech Republic, July 2005.

[2] M. Koot, J. Kessels, B. de Jager, M. Heemels, and P. van den Bosch. Energy management strategies for vehicle power nets. In *Proc. of the American Control Conf.*, pages 4072–4077, Boston, USA, June 2004.

[3] M.W.T. Koot. *Internal report: Literature study on mixed integer programming*. Department of Mechanical Engineering, Eindhoven University of Technology, 2004.

[4] M.W.T. Koot. *Internal report: Optimization of the energy flow in a dual storage power net*. Department of Mechanical Engineering, Eindhoven University of Technology, 2004.

[5] B. Zijlstra. *LINDO API as a Matlab Interface, SE 420409*. Department of Mechanical Engineering, Eindhoven University of Technology, 2004.

[6] I. E. Grossmann. Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and Engineering*, 3(3):227–252, 2002.

[7] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, New York, 2^{nd} edition, 2000.

[8] I. E. Grossmann I. Quesada. *An LP/NLP based Branch And Bound algorithm for convex MINLP optimization problems*. Computers and Chemical Engineering 16, 937-947, 1992.

[9] Matlab. `http://www.mathworks.com/`.

[10] lpsolve. `http://www.geocities.com/lpsolve/`.

[11] The tomlab optimization environment. `http://tomlab.biz/`.

[12] Lindo. `http://www.lindo.com/`.

[13] Ampl. `http://www.ampl.com/`.

Appendix A

MINLP model of the dual storage power net

This appendix describes the MINLP model of the dual storage power net as derived by M.W.T. Koot. It is an excerpt from [4].

The components can be modeled as quadratic relations between incoming and outgoing power. The model can be structured such that the design variables of the optimization problem are the power stored in the battery P_{bs}, the power stored in the supercap P_{cs}, and the position of the switch s. This has some benefits in writing the optimization problem as a (mixed integer) nonlinear or quadratic programming problem with linear constraints.

A.1 Components

Battery:

$$P_b = P_{bs} + b\,P_{bs}^2 \tag{A.1}$$

Supercap:

$$P_c = P_{cs} + c\,P_{cs}^2 \tag{A.2}$$

DC/DC converter:

$$P_{db} = P_{dc} + P_d \tag{A.3}$$

$P_{db} > 0$ means power is going from the battery to the supercap. P_d represents the losses in the DC/DC converter. P_d is always positive and increases with the power going trough the converter. It can for instance be modeled as:

$$P_d = d\,P_{dc}^2 \quad \text{or as:} \quad P_d = d\,P_{db}^2 \tag{A.4}$$

Generator:

$$P_g = g_0 + g_1\,P_e + g_2\,P_e^2 \tag{A.5}$$

Mechanical power:

$$P_m = P_p + P_g \tag{A.6}$$

Engine:

$$f = f_0 + f_1\,P_m + f_2\,P_m^2 \tag{A.7}$$

A.2 Power flow

Electrical power flow:

$$P_b = (1-s)\,P_e - P_{db} - P_l \quad \Rightarrow \quad (1-s)\,P_e = P_b + P_l + P_{db} \tag{A.8}$$

$$P_c = s\,P_e + P_{dc} \quad \Rightarrow \quad s\,P_e = P_c - P_{dc} \tag{A.9}$$

Combining these yields:

$$P_e = s\,P_e + (1-s)\,P_e = P_l + P_b + P_c + P_{db} - P_{dc} \tag{A.10}$$

Using the relation for the DC/DC converter this becomes:

$$P_e = P_l + P_b + P_c + P_d \tag{A.11}$$

Independently from the position of the switch, the electric energy provided by the generator is equal to the sum of the powers to the load, the battery, and the supercap, and the power losses of the DC/DC converter.

The losses of the DC/DC converter need to be expressed as a polynomial of the design variables P_{bs}, P_{cs}, and s. Two possible ways are presented.

First possibility:

$$P_d = d\,P_{dc}^2 \tag{A.12}$$

where:

$$P_{dc} = P_c - s\,P_e \tag{A.13}$$

where:

$$P_e = P_l + P_b + P_c + P_d = P_l + P_b + P_c + d\,P_{dc}^2 \tag{A.14}$$

So this becomes:

$$P_{dc} = P_c - s\,(P_l + P_b + P_c + d\,P_{dc}^2) \tag{A.15}$$

or:

$$s\,d\,P_{dc}^2 + P_{dc} - P_c + s\,(P_l + P_b + P_c) = 0 \tag{A.16}$$

This can be solved to P_{dc}, but does not yield a polynomial relation in the design variables.

The difficulty is that P_e depends on P_d, and P_d depends on P_e.

By neglecting the term P_d in the electric power flow equation for a moment:

$$P_e \approx P_l + P_b + P_c \tag{A.17}$$

this becomes:

$$P_{dc} \approx P_c - s\,(P_l + P_b + P_c) \tag{A.18}$$

which is a polynomial relation in the design variables.

Using this relation yields the following electric power flow equation:

$$P_e = P_l + P_b + P_c + P_d = P_l + P_b + P_c + d\,(P_c - s\,(P_l + P_b + P_c))^2 \tag{A.19}$$

which is a polynomial relation in the design variables.

A.3 Cost function

The relations described above can be combined such that the fuel use is expressed as a high order polynomial of P_{bs}, P_{cs}, and s :

$$f = fuel(P_{bs}, P_{cs}, s) \tag{A.20}$$

The design variables are:

$$x = [\ P_{bs}\ P_{cs}\ s\ ]^T \tag{A.21}$$

A.4 Constraints

Switch:

$$0 \leq s \leq 1 \quad \text{or} \quad s = \{0, 1\} \tag{A.22}$$

Storage power:

$$P_{bs\,min} \leq P_{bs} \leq P_{bs\,max} \qquad P_{cs\,min} \leq P_{cs} \leq P_{cs\,max} \tag{A.23}$$

Storage energy:

$$E_{bs\,min} \leq E_{bs} \leq E_{bs\,max} \quad \text{where} \quad E_{bs}(k) = E_{bs}(0) + \sum_{i=1}^{k} P_{bs}(i)\,\Delta t \tag{A.24}$$

$$E_{cs\,min} \leq E_{cs} \leq E_{cs\,max} \quad \text{where} \quad E_{cs}(k) = E_{cs}(0) + \sum_{i=1}^{k} P_{cs}(i)\,\Delta t \tag{A.25}$$

Endpoint constraints:

$$E_{bs}(n) = E_{bs}(0) \quad \Rightarrow \quad \sum_{k=1}^{n} P_{bs}(k) = 0 \tag{A.26}$$

$$E_{cs}(n) = E_{cs}(0) \quad \Rightarrow \quad \sum_{i=1}^{k} P_{cs}(i) = 0 \tag{A.27}$$

Electric power:

$$P_{e\,min} \leq P_e \leq P_{e\,max} \tag{A.28}$$

where:

$$P_e = P_l + P_b + P_c + P_d = P_l + P_{bs} + P_{cs} + b\,P_{bs}^2 + c\,P_{cs}^2 + P_d \tag{A.29}$$

Since the quadratic terms and P_d are always positive, P_e can be conservatively bounded on the lower side by:

$$P_{bs} + P_{cs} \geq P_{e\,min} - P_l \tag{A.30}$$

The upper bound on P_e and P_m and the bounds on P_{dc} are neglected for now. They will already be limited because of the bounds on P_{bs} and P_{cs}.

Appendix B

Figures

B.1 Loads

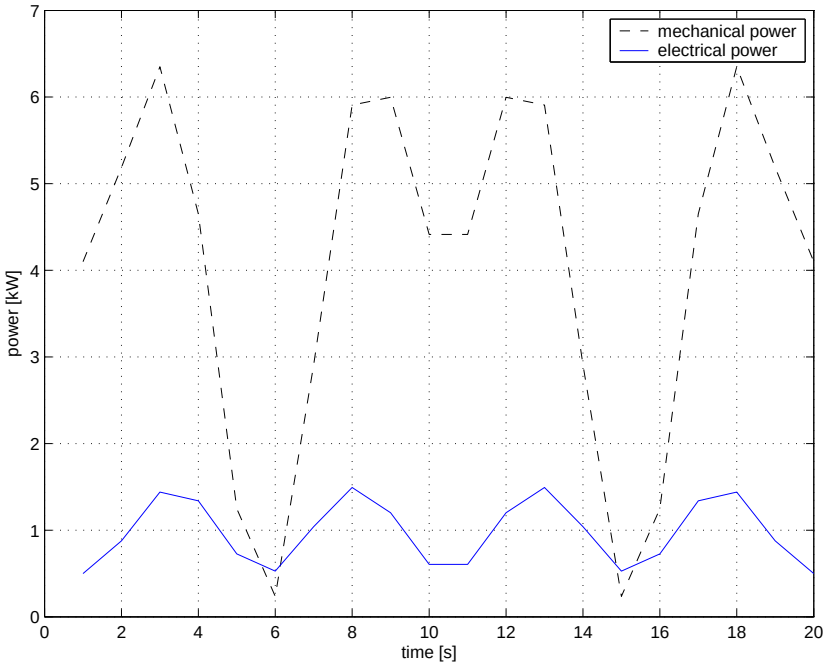

Figure B.1: Mechanical and electrical power for $n = 20$

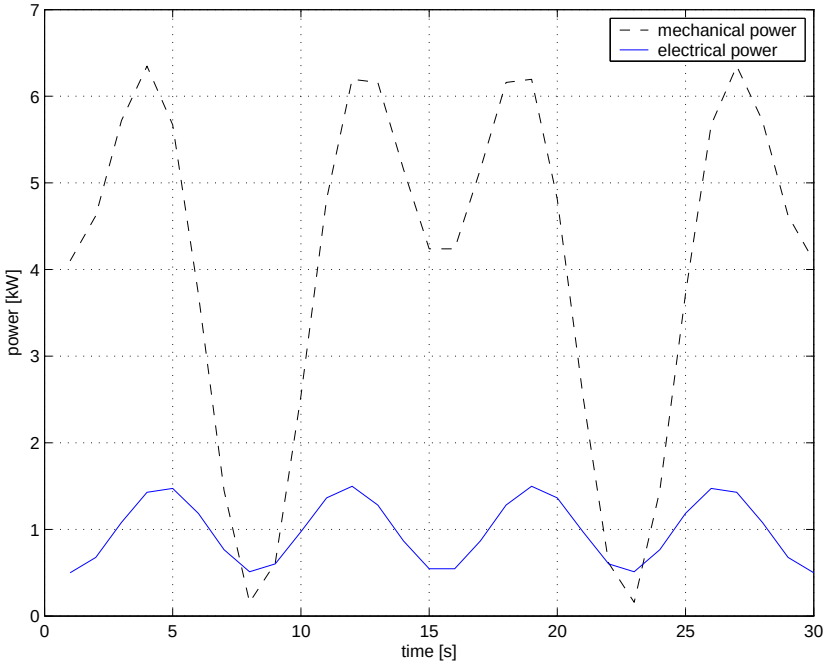

Figure B.2: Mechanical and electrical power for $n = 30$

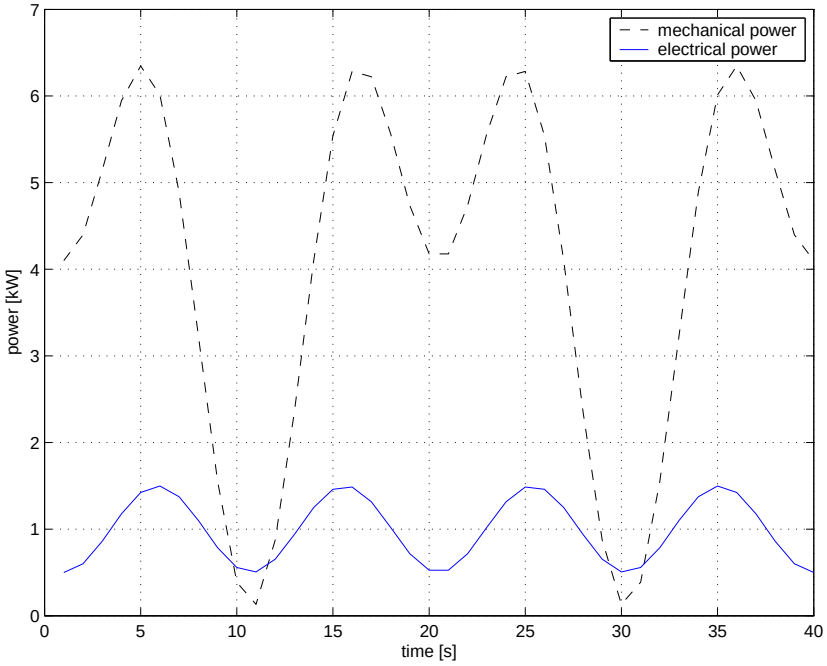

Figure B.3: Mechanical and electrical power for $n = 40$

B.2 Outer Approximation

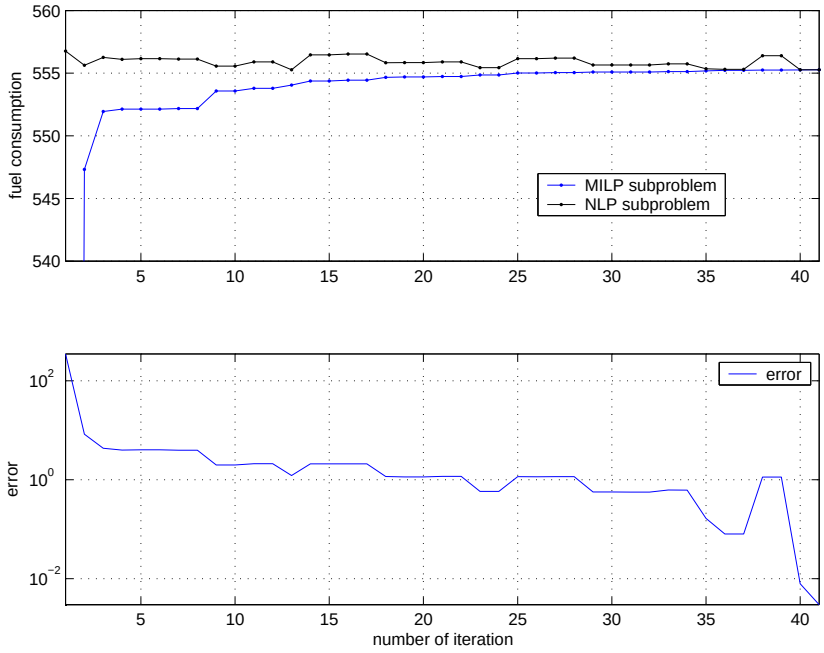

Figure B.4: Data Outer Approximation, period of 20 seconds, zeros as starting points

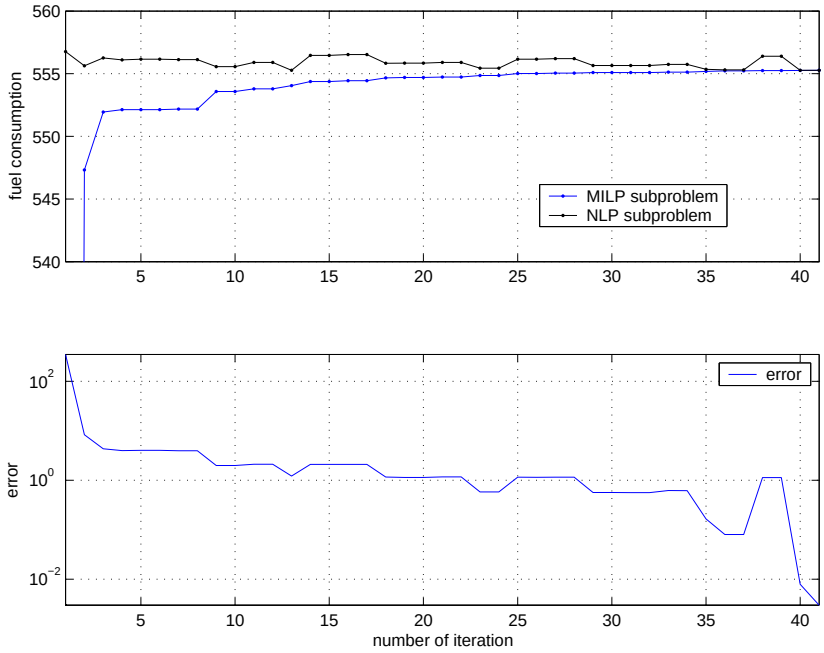

Figure B.5: Data Outer Approximation, period of 20 seconds, ones as starting points

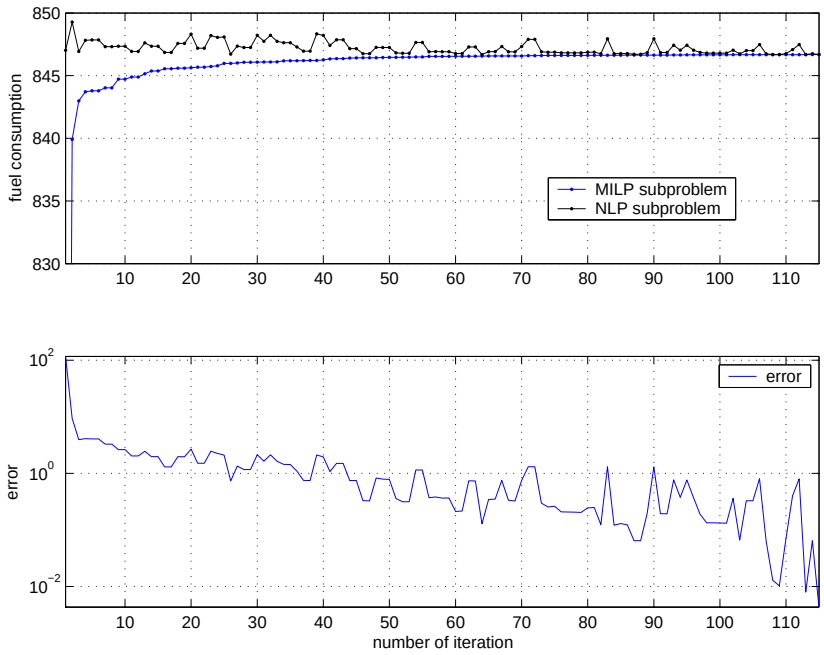

Figure B.6: Data Outer Approximation, period of 30 seconds, zeros as starting points

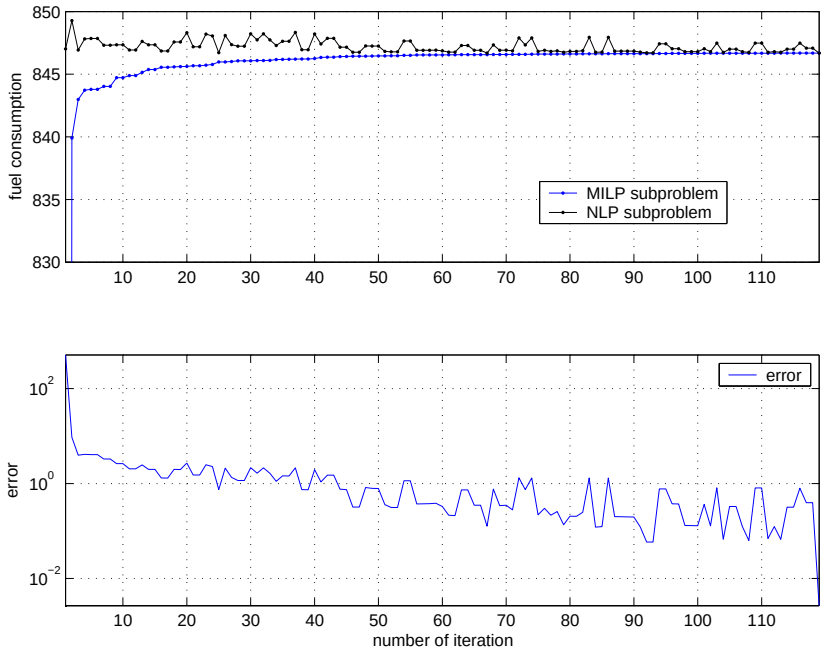

Figure B.7: Data Outer Approximation, period of 30 seconds, ones as starting points

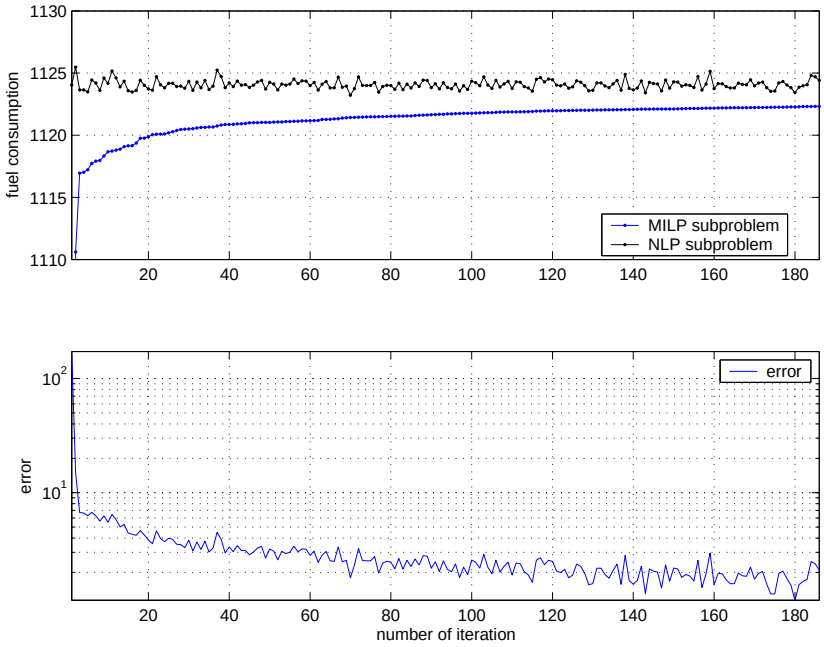

Figure B.8: Data Outer Approximation, period of 40 seconds, zeros as starting points

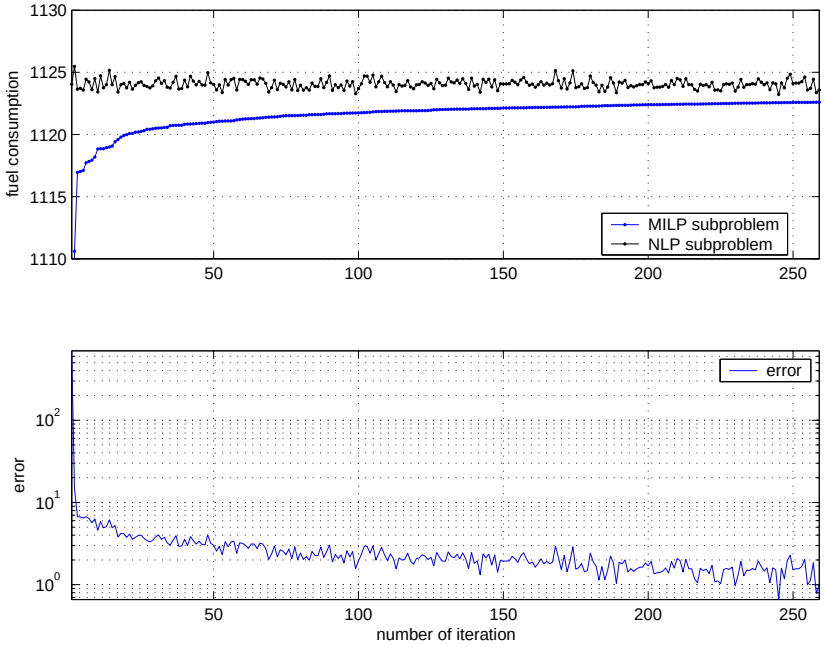

Figure B.9: Data Outer Approximation, period of 40 seconds, ones as starting points

B.3 Generalized Benders Decomposition

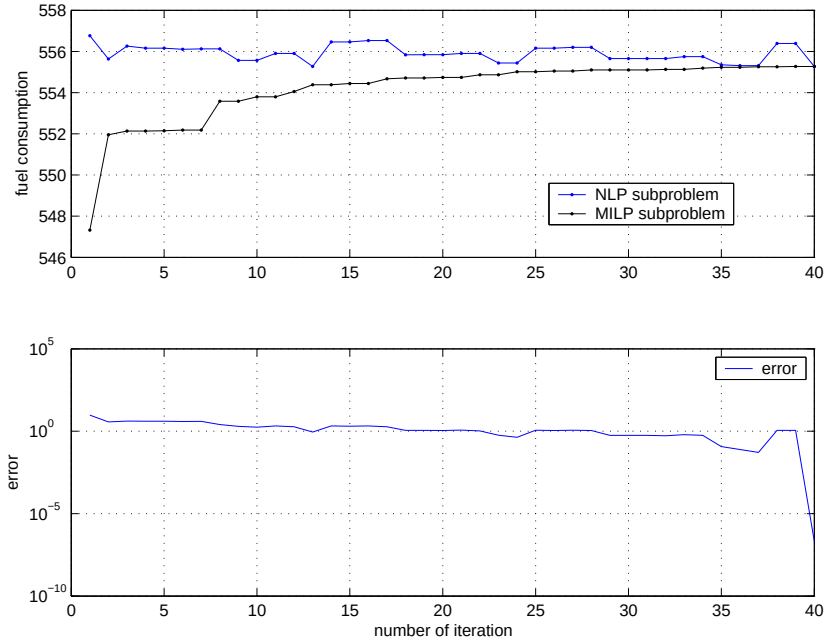

Figure B.10: Data Generalized Benders Decomposition, period of 20 seconds, zeros as starting points

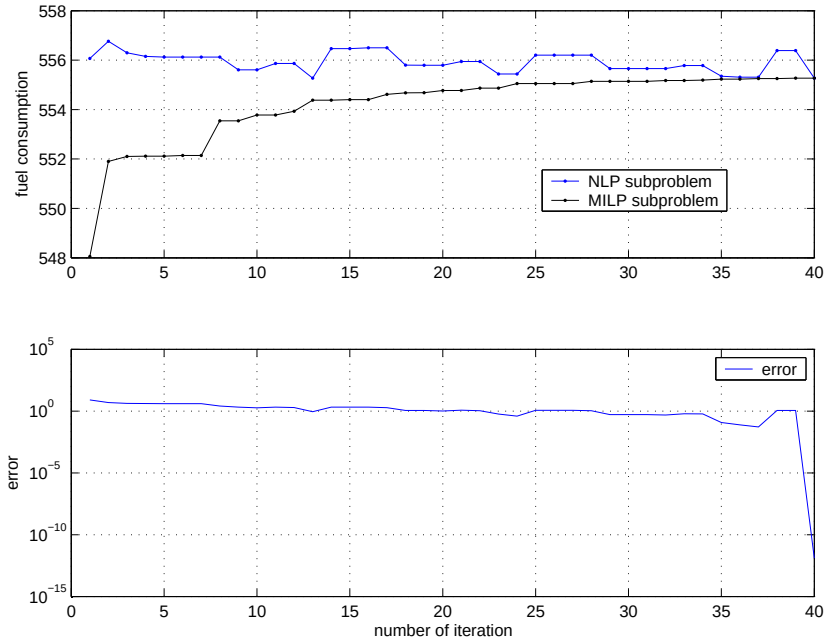

Figure B.11: Data Generalized Benders Decomposition, period of 20 seconds, ones as starting points

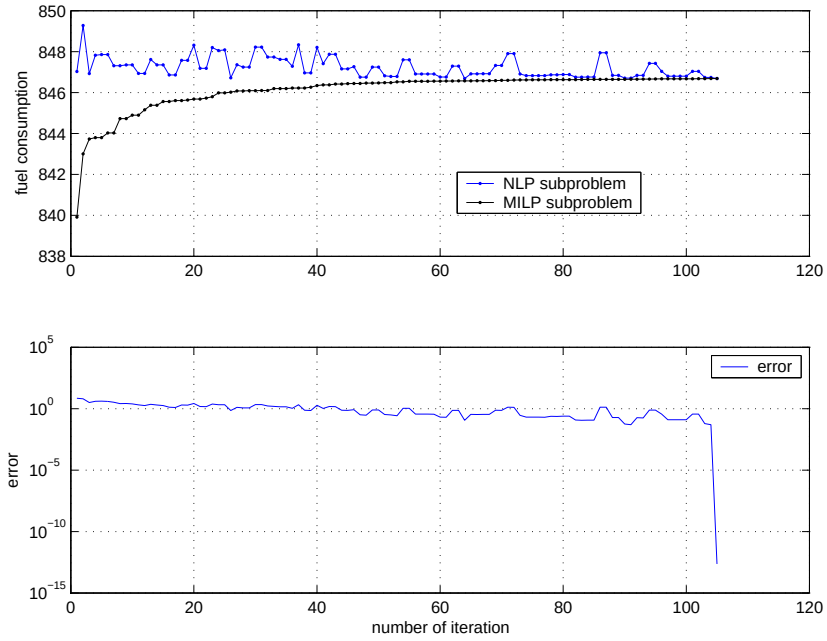

Figure B.12: Data Generalized Benders Decomposition, period of 30 seconds, zeros as starting points

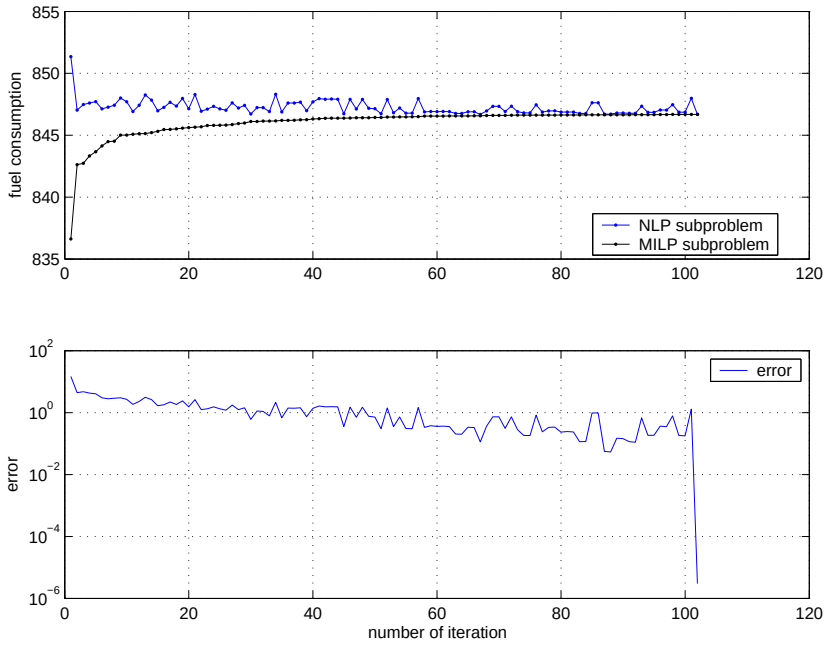

Figure B.13: Data Generalized Benders Decomposition, period of 30 seconds, ones as starting points

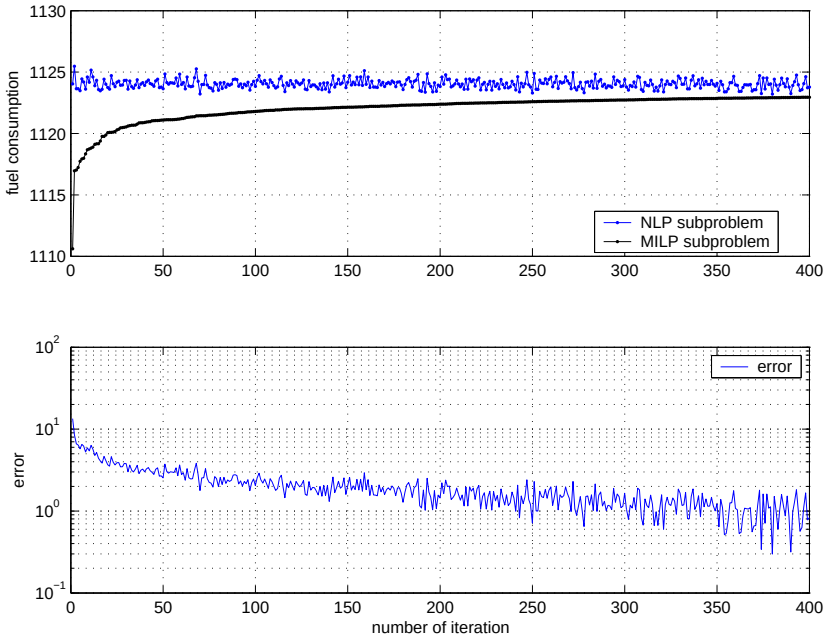

Figure B.14: Data Generalized Benders Decomposition, period of 40 seconds, zeros as starting points

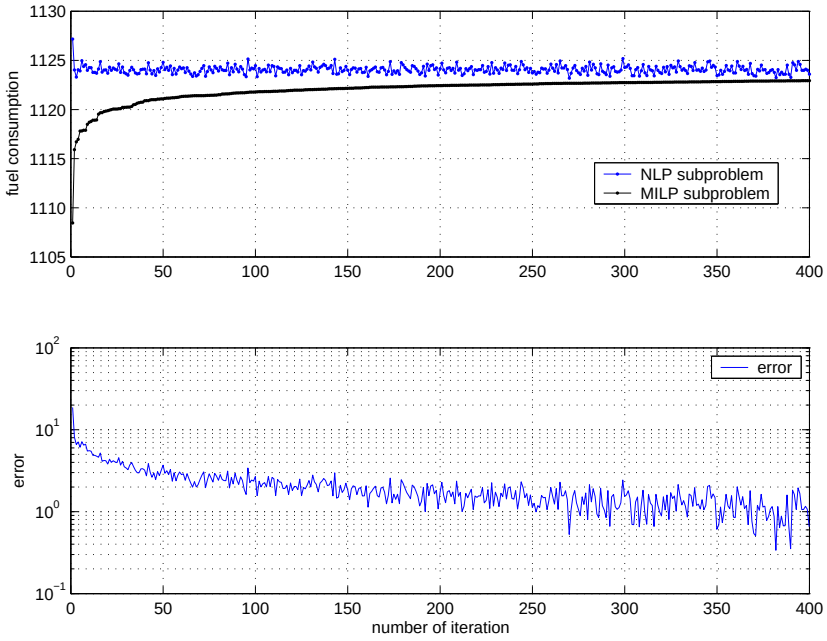

Figure B.15: Data Generalized Benders Decomposition, period of 40 seconds, ones as starting points

B.4 The variables

B.4.1 Power stored in the battery

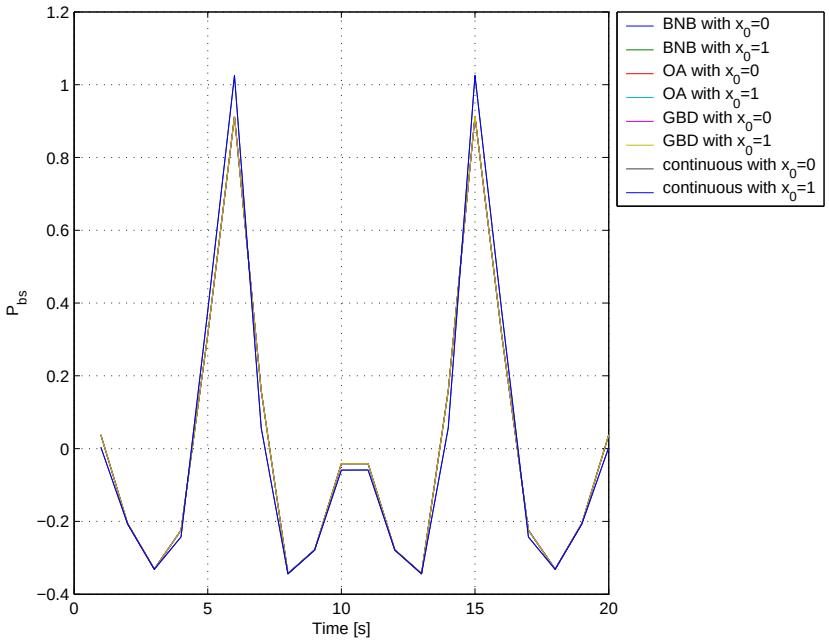

Figure B.16: Power stored in the battery, period of 20 second

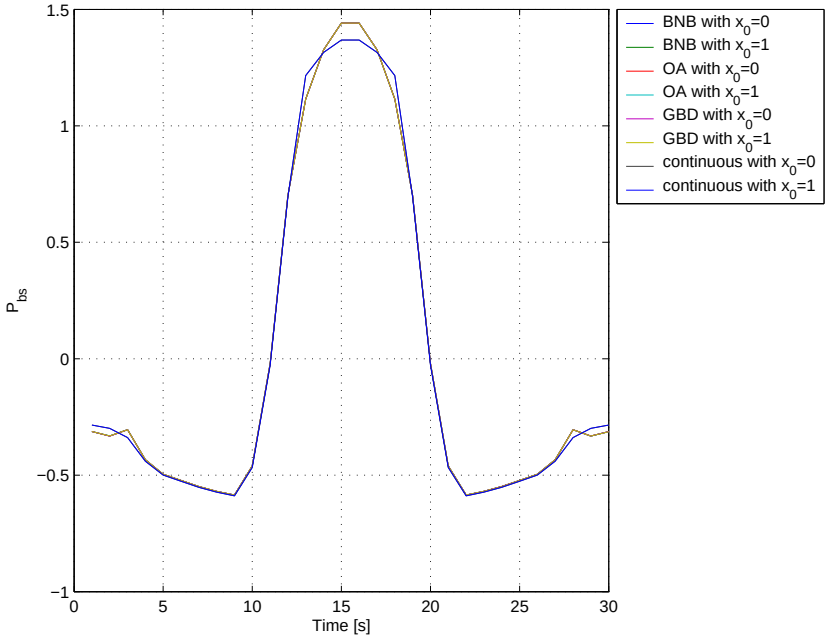

Figure B.17: Power stored in the battery, period of 30 second

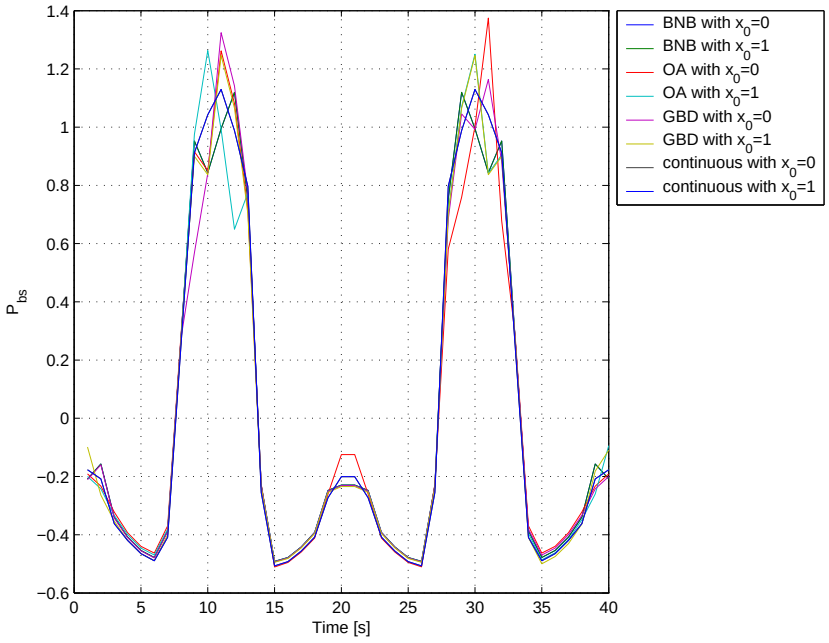

Figure B.18: Power stored in the battery, period of 40 second

B.4.2 Power stored in the supercap

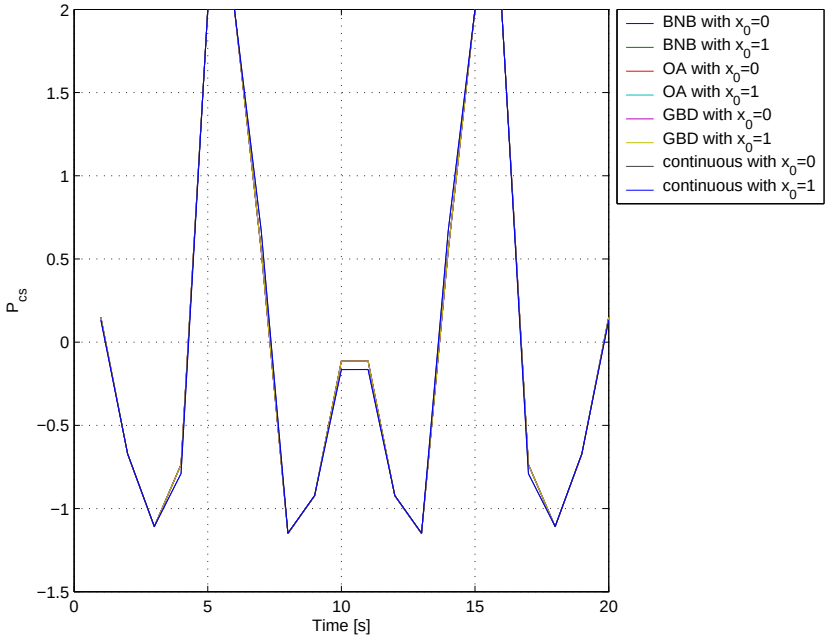

Figure B.19: Power stored in the supercap, period of 20 second

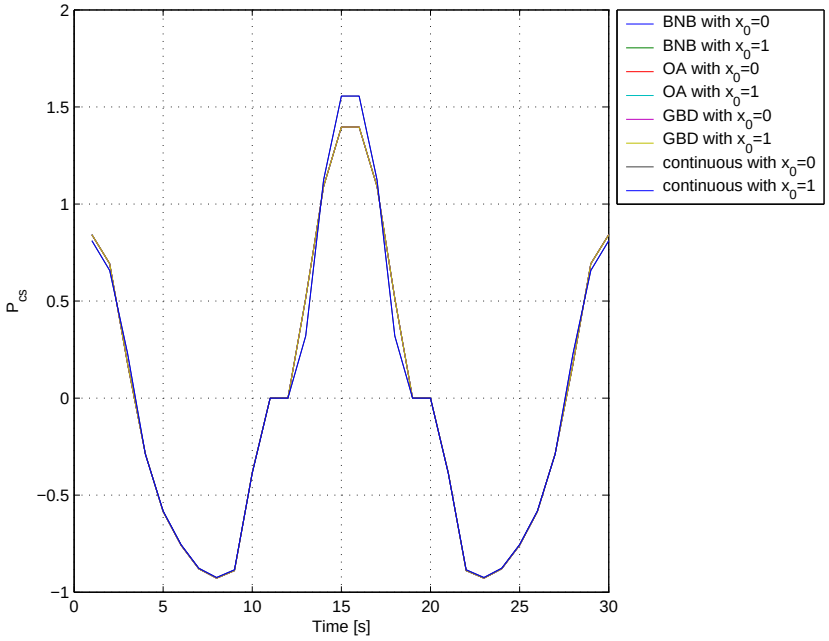

Figure B.20: Power stored in the supercap, period of 30 second

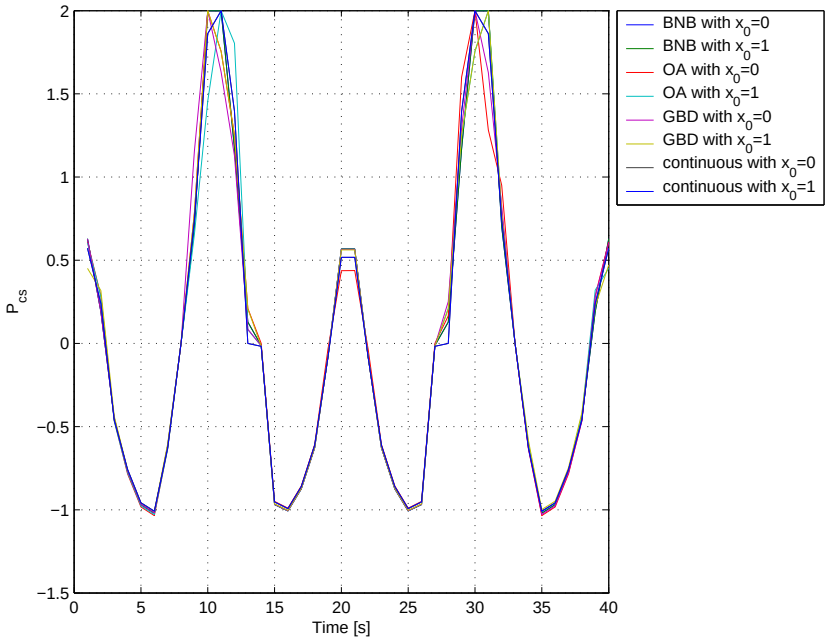

Figure B.21: Power stored in the supercap, period of 40 second

B.4.3 The position of the switch

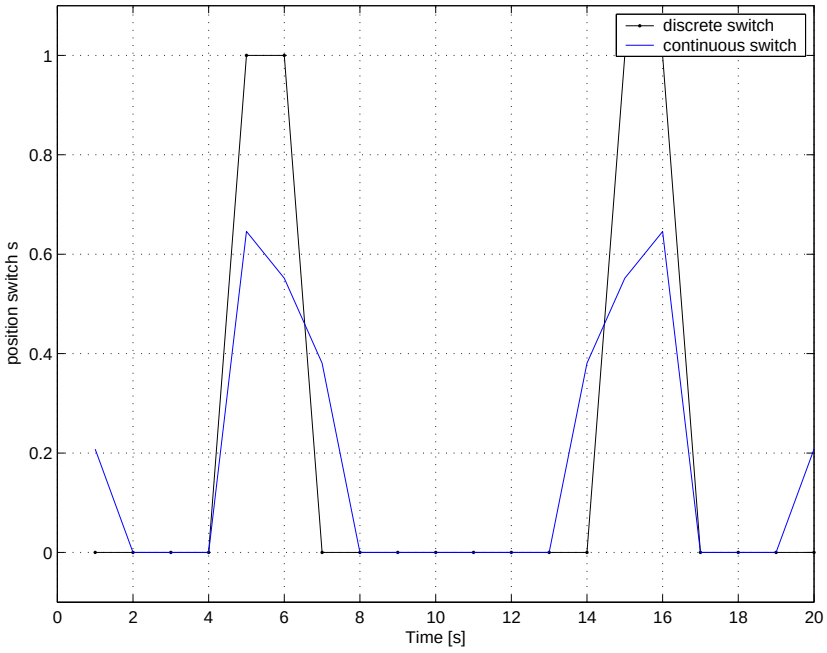

Figure B.22: The position of the switch, period of 20 second

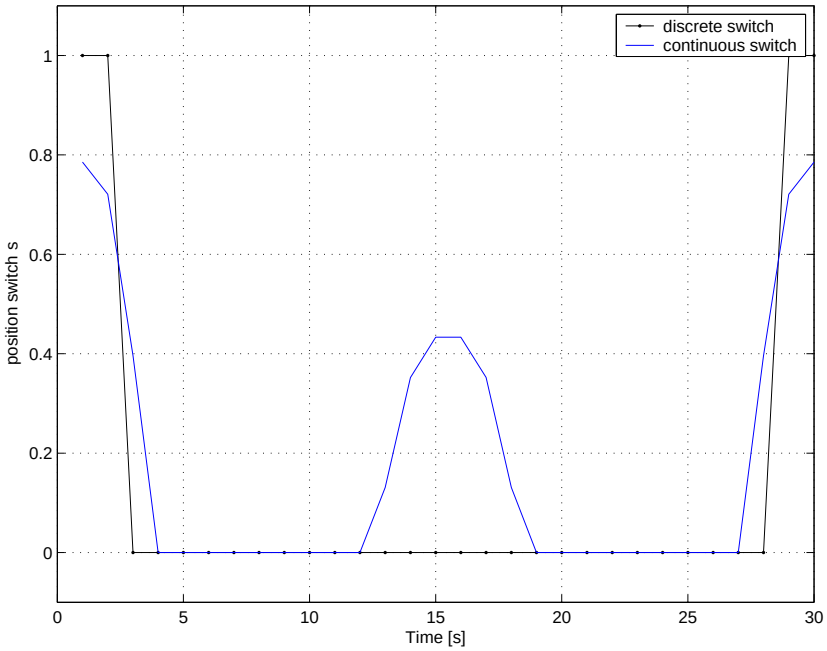

Figure B.23: The position of the switch, period of 30 second

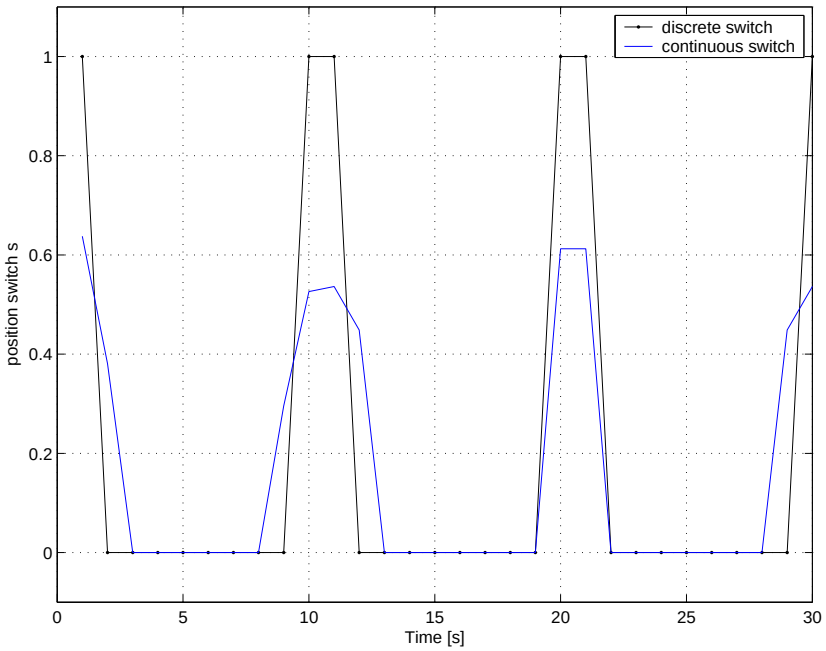

Figure B.24: The position of the switch, period of 40 second