Implementation of a *Modelica* Online Optimization for an Operating Strategy of a Hybrid Powertrain

Henrik Wigermo, BMW Group, Energymanagement 88077 München
Johannes von Grundherr, BMW Group,Energymanagement 88077 München
Thomas Christ BMW Hybrid Cooperation, Vehicle Architecture,
1960 Technology Dr., Troy, Michigan USA

January 21, 2008

Abstract

The paper presents a method of implementing an optimization based control algorithm within the Modelica framework. To find the optimal point within a given objective function the golden section search is employed. Its implementation in Modelica is presented. The optimizer based control strategy is applied to control a simplified electrical circuit and to a hybrid electric vehicle.

Keywords: Modelica; Optimization; Hybrid Vehicle; Simulation; Fuel Consumption

1 Introduction

Online optimization is increasingly being implemented for better results in controlling complex systems. It is especially helpful if the control objective depends on several input parameters which influence the outcome in a non intuitive way. One example is the operational strategy of a powersplit hybrid electric vehicle.
Compared to conventional transmissions, hybrid transmissions allow for several additional degrees of freedom: The combustion engine speed can be controlled independently from vehicle speed and battery power can be used for propulsion or the storage of braking energy. Although the main control objective is the fuel economy of the vehicle, other goals like dynamic response, driveability, acoustic impression and tailpipe emissions have to be achieved. In many cases the definition of the control objective is given by a calibration table or multidimensional mappings. Since a mapping normally cannot be expressed analytically, the solution to the optimization problem has to be computed online for each control step.

In the development process of hybrid vehicles, simulation is a key issue. It is used to study aspects like fuel consumption and performance and to understand complex system interactions. Since the hybrid vehicle powertrain is composed of mechanical, electrical, chemical and thermodynamical components, *Modelica* is a very useful tool for this. The control software of the hybrid vehicle is normally implemented using tools like Simulink or ASCET. The actual powertrain control is only a small part of the entire controls software. A great deal of code which is interconnected to the actual powertrain control concerns system diagnosis or remedial actions, and does not need to be simulated. To study the powertrain behavior only the relevant parts of the control code are transferred to *Modelica*.
In this paper, we shall present a simple optimization algorithm and give an example on how it can be implemented in *Modelica*. We will also take a look on a possible employment of such an algorithm; the powertrain control of a hybrid electric vehicle. In addition, the following points have been investigated: How will an algorithm that requires fixed time-steps work together with an complex vehicle model? How does the optimization influence the simulation time? How can standard *Modelica* elements like tables be integrated in the optimization algorithm, since it doesn't allow graphical programming?

2 Problem statement

A Plant P is controlled by its input u and disturbed by d. y is the observed measurement. In an early control development stage the plant can be represented by a simulation model. The control task is to follow a given reference y_{ref} so that an objective function $J(y, y_{ref})$ is minimized. For linear systems and quadratic objective

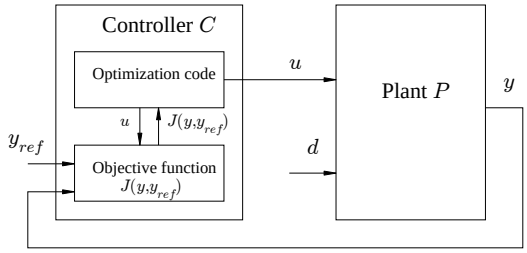

Figure 1: Control optimization problem

functions the choice of controller is well understood. A linear state feedback control can be directly derived from the linear plant given by the system matrices A, B, C, D and by the coefficients of the quadratic objective function.

For nonlinear objective functions the optimization can be carried out by an optimization algorithm. In each optimization step the algorithm calls the objective function, iterating the control signal u to generate the optimal solution u^*.

In our case the plant is a *Modelica* model. The control using the optimization algorithm is also integrated in *Modelica*. A tutorial example of such an optimization is shown in section 3.2.

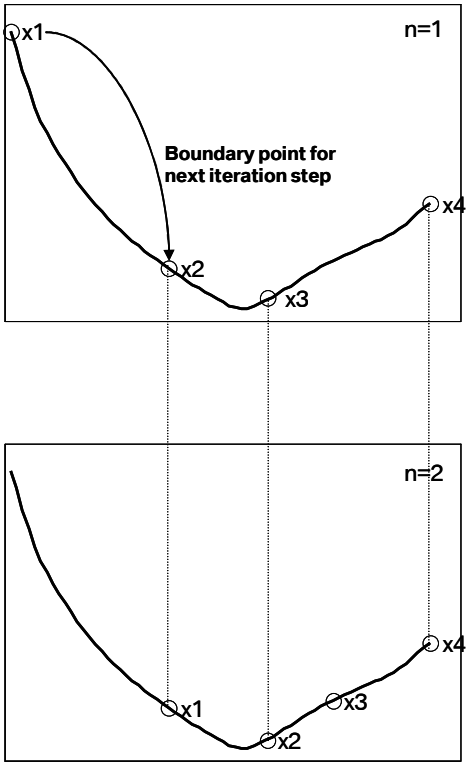

Figure 2: Principle of Golden Section Search Algorithm

3 Online optimization

An optimization algorithm used for the given problem has to be robust, i.e. it needs to come up with a solution after a finite number of iterations. Such an algorithm is golden section search. In this paper its integration into the *Modelica* framework is shown.

3.1 Optimization algorithm - Golden section search

The golden section search derives its name from the fact that it narrows its search interval with the golden ratio $\frac{1}{2}(1+\sqrt{(5)})$ in each step. The technique is effective only for unimodal functions, where a maximum or minimum is known to exist within a given interval. As starting points the lower and upper limit of the search interval are chosen. Using the golden section, two new points within the interval are evaluated and compared. The point with the highest functional value is chosen as a new boundary point, and points outside of this are no longer considered. The algorithm continues to search until the maximum number of iterations is reached or the termination condition suggested in [4]: $\mid x4 - x1 \mid > \tau(\mid x2 \mid + \mid x3 \mid)$ is satisfied. τ is a tolerance parameter. *Modelica* code 1 describes the golden section search algorithm:

```
function goldenSectionSearch
  extends Modelica.Icons.Function;
  parameter Real tau=0.001;
  ...
  constant Real C=0.5*(3 - sqrt(5));
  constant Real R=1-C;
  ...
algorithm
    x1:= xLowerLimit;
    x4:= xUpperLimit;
    x2:= R*x1 + C*x4;
    x3:= C*x1 + R*x4;
    fx2:=optFunction(x2,alpha,IbatDes,
    Ri,Iload,gammaI);
    fx3 :=optFunction(x3,alpha,IbatDes,
    Ri,Iload,gammaI);

while abs(x4-x1)>
    tau*(abs(x2)+abs(x3)) loop
  if (fx3<fx2) then
        x1:=x2;
        x2:=x3;
        x3:=R*x3 + C*x4;
        fx2:=fx3;
        fx3:=optFunction(x3,alpha,
          IbatDes,Ri,Iload,gammaI);
```

```
  else
        x4:=x3;
        x3:=x2;
        x2:=R*x2 + C*x1;
        fx3:=fx2;
        fx2:=optFunction(x2,alpha,
          IbatDes,Ri,Iload,gammaI);
    end if;
end while; if
  (fx2<fx3) then
    xmin:=x2;
    fxmin:=fx2;
else
    xmin:=x3;
    fxmin:=fx3;
end if;

end goldenSectionSearch;
```

Modelica Code 1: Golden Section Search Algorithm

3.2 Optimization example

The following example (see fig. 3) illustrates the control problem: A time varying electric load $I_{load}(t)$ is to be supplied with power from an energy storage device (e.g. a battery) in such a way that the power losses are minimal and the State-of-Charge (SOC) is kept at a fairly constant level (to optimize the lifetime of the energy storage device). The system can be influenced from an external current source I_{opt}, which can deliver power at all times but with losses that are time-dependent. This means at times it can be efficient to charge the battery and to use the stored energy at a later time when the losses of the current source are high. α is a control variable which we choose in order to weigh the importance of the SOC-control.

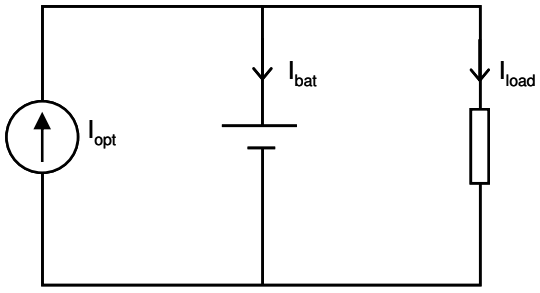

Figure 3: Example Electric Circuit

From these control objectives we define the objective function to be minimized as:

$$Cost = \underbrace{\alpha \mid I_{bat,des}(SOC) - I_{opt} \mid}_{SOCControl} + \underbrace{R_i(I_{opt} - I_{load})^2}_{BatteryLoss} + \underbrace{\gamma_I(t) I_{opt}}_{CurrentCost} \quad (1)$$

I_{opt} is our control variable; the current of the external current source. The battery losses are assumed to be a quadratic function of the current through the battery internal resistance. The SOC-optimal battery current $I_{bat,des}$ is a function of the battery SOC and is chosen to the following curve:

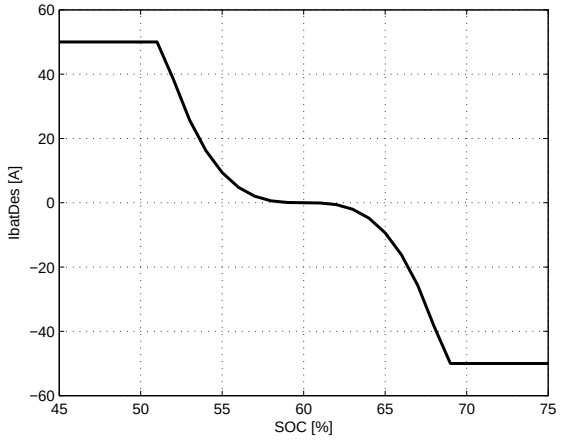

Figure 4: $I_{bat,des}$ as a function of battery state of charge

$\gamma_I(t)$ is a time-varying function that decides the loss power of the external current source. In this example, we have chosen it to be sinodial (see figure 6).

3.2.1 Results

We let the optimization algorithm defined in chapter 3.1 find the optimal solution to the objective function (1). The variable *Iopt* is computed through a function call of `goldenSectionSearch`.
Figure 5 shows the calculated optimal current, as well as the load current and the resulting battery current. We can see that high (battery discharging) peaks in the load current have been compensated for with the current source in order to minimize the battery losses.
In figure 6, the optimized current has been compared to a control strategy that only considers the battery SOC (as described in figure 4). We can conclude that the optimization chooses to charge the battery at times when the current is inexpensive, but at the same time manages to keep the SOC at levels similar to the SOC-controlled strategy, not very far from the target value of 60%.

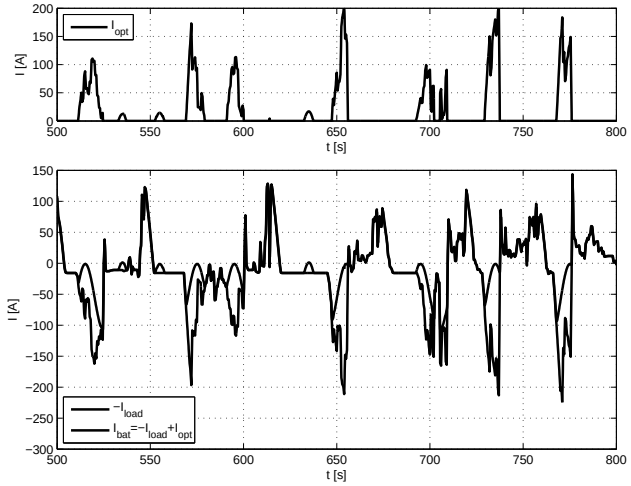

Figure 5: Optimization result: Controlled current I_{opt}, load current I_{load} and resulting battery current I_{bat}

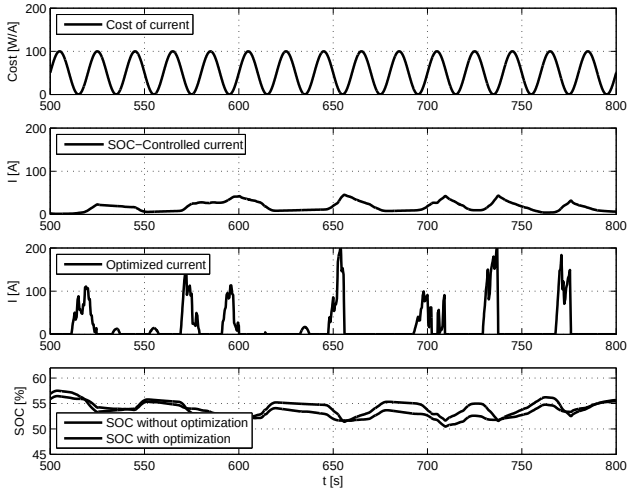

Figure 6: Optimization result: Cost of current, SOC-Controlled current, Optimized current and SOC

As a measurement on how good the optimization has worked, we compute the total system losses (battery losses and losses of the external current source). By integration of the loss power, as shown in figure 7, we see that the energy lost in the optimized system is only about half of the SOC-controlled strategy. The heat developed in the battery is proportional to the loss power, and the operating temperature of the battery rises over time. However, with the optimal control the battery losses are kept down, and the temperature remains at a lower level than the SOC-controlled strategy.

3.2.2 Implementation of tables in *Modelica* text

A difficulty in the implementation of the online optimization is the use of table look-ups for the objective function within *Modelica* text algorithm sections. In order to do this, one must initialize the table using *dymTableInit*. The table/mapping can then be called from a function using the function *dymTableIpo1* or *dymTableIpo2*.

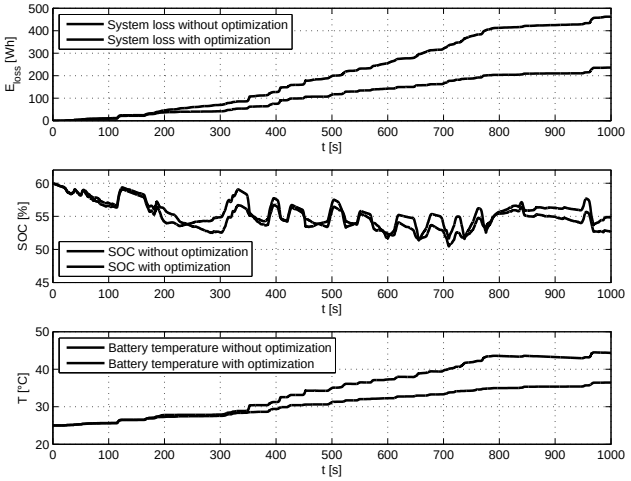

Figure 7: Comparison optimized system with SOC-controlled system: System losses, battery SOC and temperature

```
...
equation
  when initial() then
    Data.EngineFuelFlow=dymTableInit
    (2.0, smoothness, "FuelFlowAllCyl",
    engineFuelFlowTable, table, 0.0);
  end when;
...
```

Modelica Code 2: Table Interpolation in Modelica Text

3.2.3 Comments on simulation time

In a simple example like the one given above, the simulation time of a model containing an optimization algorithm is good, only somewhat slower than an equal model using a traditional control approach. However when combined with a complex vehicle model, generating a lot of events due to system state changes, a fixed-step optimization algorithm can slow the simulation time down considerably. In these cases, it has been shown that time-discrete sampling of the optimization algorithm increases the computation speed. A well considered sampled optimization algorithm delivers virtually the same result as the non-sampled, but without recomputing the optimal solution for each event triggered by the plant. Using this method, we

have achieved simulation performance comparable to our traditional control concepts.

4 Hybrid vehicle application

This section will present a simulation model of a hybrid electric vehicle using a control strategy based on online optimization. In this case, the optimization only governs the choice of engine torque, but it could also be employed for the choice of gear, or in EVT-mode (Electrically Variable Transmission) the speed of the internal combustion engine. The advantage of such an implementation would be that the vehicle would adapt its gear strategy depending on the current conditions. However such a strategy also has the disadvantage that the gear choice is not always comprehensible to the driver.
The following control objectives are considered in our objective function [5]:

- Combustion engine losses
- Battery losses
- Electric machine losses
- Battery SOC control

Below simulation results from an FTP72[1] simulation of a hybrid electric vehicle are shown. In figure 8 the vehicle speed is plotted with our control signal, the optimal combustion engine torque. T_{ICE} is available for us to choose at all times except the phases where the vehicle is powered electrically. It has been chosen to minimize the listed control objectives.
Figure 9 shows the resulting power and SOC of the battery. At a given engine speed the battery power is proportional to the combustion engine torque, and therefore also directly connected to our control signal. We can conclude that even albeit a high portion of pure electrical driving in this cycle, the SOC remains around the target SOC of 60%.

5 Discussion and conclusion

This paper shows that it is possible to implement optimization algorithms for the control of a plant, e.g. a hybrid electric vehicle, in *Modelica*. Using online optimization, a fixed-step optimization algorithm can find a solution to a number of complex and interconnected control objectives. Although the optimization

[1]The Federal Test Procedure legislation fuel cycle

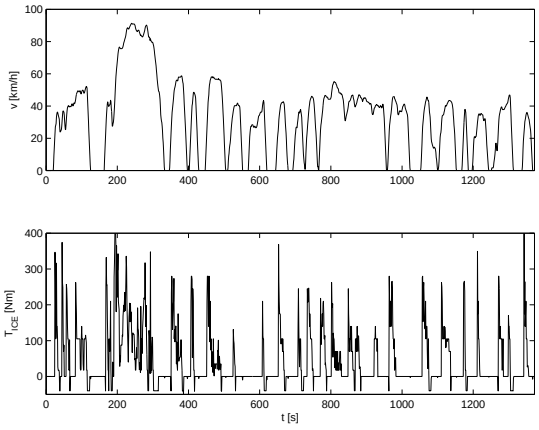

Figure 8: Vehicle speed (above) and combustion engine torque (below) as a function of time

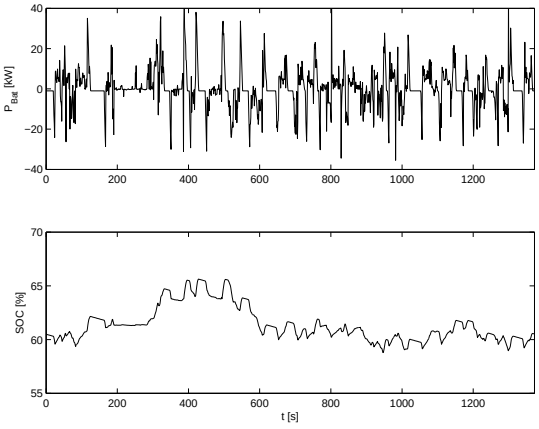

Figure 9: Battery power (above) and state of charge (below) as a function of time

algorithm has to be called at each step of the simulation, the simulation time was comparable to models using traditional control strategies.

References

[1] Eborn J. On Model Libraries for Thermo-hydraulic Applications. Lund, Sweden: PhD thesis, Department of Automatic control, Lund Institute of Technology, 2001.

[2] Tummescheit H. Design and Implementation of Object-Oriented Model Libraries using Modelica. Lund, Sweden: PhD thesis, Department of Automatic control, Lund Institute of Technology, 2002.

[3] Tummescheit H, Eborn J. Chemical Reaction Modeling with ThermoFluid/MF and MultiFlash. In: Proceedings of the 2th Modelica Conference 2002, Oberpfaffenhofen, Germany, Modelica Association, 18-19 March 2002.

[4] Press, W. H.; Teukolsky, S. A. & Vetterling, W. T. et al. (1999), Numerical Recipes in C, The Art of Scientific Computing (second ed.), Cambridge University Press, Cambridge, ISBN 0-521-43108-5.

[5] US 2007/0032926 A1 FORD GLOBAL TECHNOLOGIES: Optimal Engine Operating Power Management Strategy for a Hybrid Electric Vehicle Powertrain. 8.2.2007.