Time reduction of the Dynamic Programming computation in the case of hybrid vehicle

Emmanuel Vinot

VINOT, Emmanuel, 2016, Time reduction of the Dynamic Programming computation in the case of hybrid vehicle , International Journal of Applied Electromagnetics and Mechanics, pp. 1-15, DOI: 10.3233/JAE-140163

TIME REDUCTION OF THE DYNAMIC PROGRAMMING COMPUTATION IN THE CASE OF HYBRID VEHICLE

Emmanuel VINOT*

*IFSTTAR, 25 Avenue François Mitterrand, 69500 Bron, France
E-mail: emmanuel.vinot@ifsttar.fr

Abstract. Deterministic Dynamic Programming is frequently used to solve the management problem of hybrid vehicles (choice of mode and power sharing between thermal and electric sources). However, it is time consuming and thus difficult to use in global sizing optimization or in parametric studies. This paper presents a comparison between three methods to compute the DDP problems. These methods are applied on the well known case of the Toyota PRIUS. It proves that a dense matrix method can reduce the computation time by up to 10 compared to classical solving methods.

Keywords: Deterministic Dynamic Programming, Solving Algorithms, Hybrid Vehicle

1. Introduction

Hybrid vehicles are an effective solution to reduce CO_2 and pollutant emissions [1][2][3]. In such architecture, the electrical driveline provides additional functionality to the drive train. Among these functionalities, regenerative braking and electrical propulsion mode are the most efficient to improve systems efficiency. Combined electrical and thermal propulsion and battery recharge by the engine also allow substantial gain. Finally, as in series parallel hybrid, engine and wheel speed decoupling is another possibility to reduce fuel consumption.

These different capabilities have to be used in the best manner to reduce fuel consumption as much as possible. The choice of the operating mode (i.e. the power energy management) is thus a key point of the efficiency of hybrid architecture [4][5][6]. It has to fix the electric or hybrid mode, the battery current and the engine operating points.

Moreover, the fuel consumption of an hybrid vehicle can only be announced taking into account the battery charge or discharge over a driving cycle long enough to be representative of real uses conditions. Thus, the power management law cannot be simply developed to minimize fuel consumption on each step of time but has to respect a global amount of charge of the battery.

An optimal method is then very useful to determine the best energy management law over a known in advance representative driving cycle [4][5][6]. These methods are obviously applied offline as the drive cycle needs to be known. Optimal off-line management remains nevertheless very useful to developed efficient online management laws [6], to perform objective comparisons between hybrid solutions [7], to size components [8][9], and in optimal global design process [10][11] to avoid the unknown and non-monotonic impact of the management on the fuel economy [9][12].

Two off-line methods are currently used in order to calculate the optimal management; the Pontryagin's minimum principle [13] and the Discrete Dynamic Programming (DDP) [14][15]. These two methods can take a long time to achieve the optimal management over a drive cycle of several hundreds of seconds. This can be a critical point especially in comparative or parametric studies and even more critical in a design process [10][11] when fuel consumption of thousands of solutions can be calculated. It is then a serious issue to reduce the computer effort as much as possible during such calculation.

In the scope of this paper, the authors focus on DDP [14][15][16][17][18] and investigate the different possibilities to reduce the calculation time. The DDP principle (section 2) allows to find the optimal

State of Charge (SOC) trajectory of the battery to minimize the fuel consumption on a drive cycle known in advance. This is performed in a restrictive SOC versus time meshed area limited by battery maximum charge and discharge possibilities. Another specification, when applying DDP to hybrid vehicle is the necessary consideration of the electric mode (discrete state). When sampling the SOC you have to be sure that one of the samples corresponds to the electric mode with no fuel consumption associate (section 3).

Several manners to improve time calculation can then be investigated:

- Reducing the size of the graph as much as possible (section 3)
- Building the graph to ensure that the electrical modes correspond to existing edges (section 3).
- Using vectorial cost calculation. So that all edge costs of the graph are calculated all at once before solving the graph (section 5).
- Improving the manner to solve the DDP problem avoiding loops and big matrix (section 4),

In this paper, the authors present the method they used to reduce the DDP calculation time to a handful of seconds (3 to 30). The four precedent points are treated. Then an example on the well known Toyota Hybrid PRIUS [19][20] is presented.

2. Dynamic Programming applied to SP_HEV

Regarding hybrid vehicle management, an optimization problem can be formulated to minimize the global fuel consumption over a global driving cycle respecting a certain amount of battery discharge [14][15][16][17][18]. The state variable is commonly the state of charge (SOC) defined as:

$$SOC(T) = SOC(t_o) - \frac{100}{3600\, C_{batt}} \int_{t_o}^{T} I(t)\eta_F \, dt \tag{1}$$

Where t_0 and T are respectively the initial and final time of a drive cycle, C_{batt} is the battery capacity in Ah.

The global fuel consumption is:

$$J = \int_{t_o}^{T} \dot{m}_f(t, \delta_{SOC})dt \tag{2}$$

Where δ_{SOC} is the SOC variation at time t and $\dot{m}_f$ the fuel rate.

The global constraint on SOC variation has to be respected:

$$\Delta SOC = SOC_{initial} - SOC_{final} = K \tag{3}$$

Using discrete dynamic programming principle, the problem formulation is the following:

The time is sample in N steps. The SOC is sample in M step between SOC$_{min}$ and SOC$_{max}$ (Fig. 1).

Solving this problem using DDP means to find the best SOC trajectory in a SOC versus time area. The SOC constraint is guaranteed if the limits of the space are built to cross on a specified final state of charge (Fig. 1, section 3).

To solve this problem an iterative method using Bellman principle [14] is currently used. A cost is associated to each edge between two consecutive points at time k and k+1 (Fig. 10). The fuel consumption on a trajectory between two points is simply the sum of the edge cost (fuel rate) on this trajectory.

Bellman principle stipulates that every sub-trajectory of an optimal trajectory is optimal [14]. It is therefore easy to find the best trajectory over the cycle using iterative methods.

The edge cost from point (k,i_1) to point (k+1,i_2) is the instantaneous fuel rate $\dot{m}_f((\mathrm{k,i1})\ (\mathrm{k+1,i2}))$ which depends on the soc variation between the beginning and the end of the edge δsoc. For instance we assume that $\dot{m}_f((\mathrm{k,i1})\ (\mathrm{k+1,i2}))$ depends only on the soc variation and vehicle speed, an example is given in section 5.

Starting at point (1, i_0) with a fuel consumption $J^*_{(1,i_0)} = 0$, the general principle is the following:

At time k for a state SOC(i_1), i.e in point (k,i_1), the minimum fuel consumption from the beginning to this point is noted $J^*_{(k,i_1)}$, and is called “cost to go” for the following of this paper. It is known for each point in the column k (i.e for i1 between $i_{\mathrm{lim\_min}}$(k) and $i_{\mathrm{lim\_max}}$(k) (Fig. 2)

Using the Bellman principle the cost to go from beginning to a point (k+1,i_2) of time k+1 may easily be determined. The principle is to add the costs of each edge J((k,i_1) $\rightarrow$(k+1,i_2)) reaching the point (k+1,i_2) to the cost to go of the point (k,i_1) and take the minimum. Repeating this operation for each point (k+1,i_2) of the column k+1 gives the best trajectories from beginning to time k+1.

The optimal cost to go at time k+1 is thus given by:

$$\begin{aligned}&[J^*_{(k+1,i_2)}, i_1] = \min_{i_1}[\dot{m}_{f\,(k,i_i)\rightarrow(k+1,i_2)} + J^*_{(k,i_1)}] \\ &for \quad i_1 = i_{1\min}(i_2) + i_{1\max}(i_2)\end{aligned} \tag{4}$$

At each step of time, for each point of the column, the index i_1 which minimizes the fuel consumption is known (4). Thus it is easy, at the end of the process to determine the best trajectory.

Note that point (k+1,i_2) can only be reached by a limited number of edges due to system limitation. The battery has a limited rate of charge or discharge depending on the battery itself and on the system limitation. Thus the spread of possible edges reaching point i_2 is limited between $i_{1\min}(i_2)$ and $i_{1\max}(i_2)$.

Section 4 of this paper focuses on different manners to solve the DDP problem and on the most efficient one in the cases of hybrid vehicle problems.

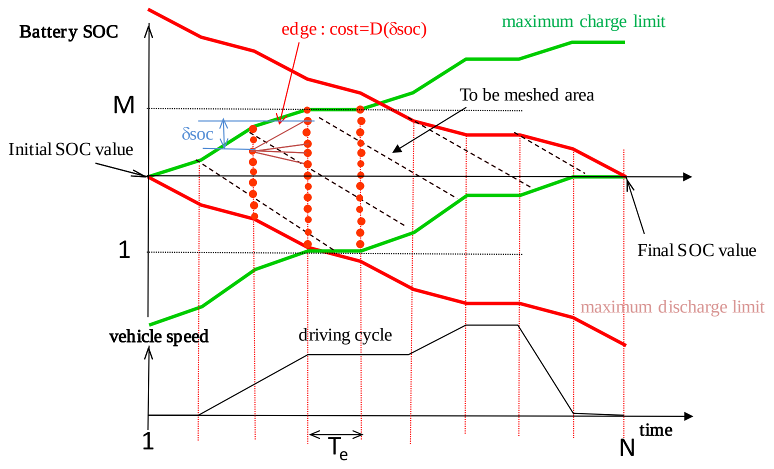

Fig. 1 DDP applied to Hybrid vehicle

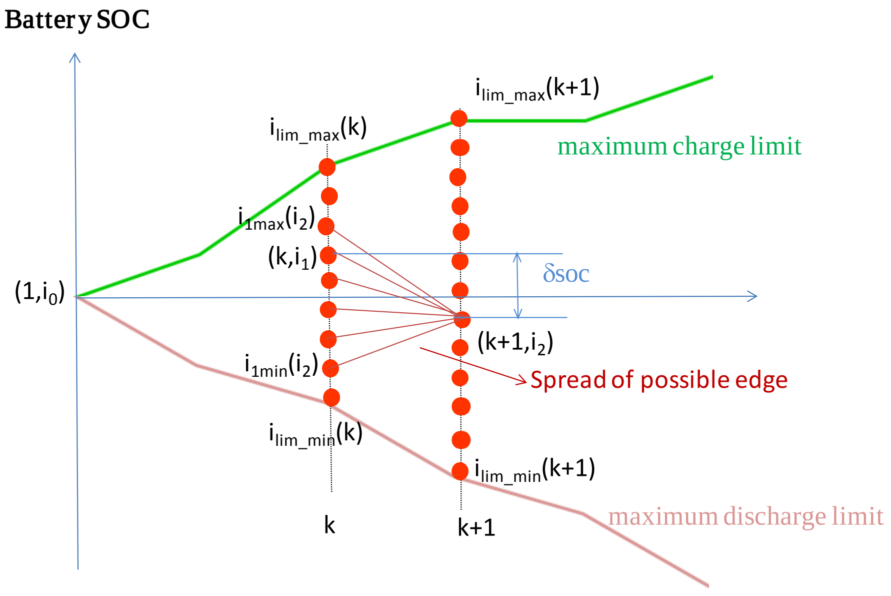

Fig. 2 DDP recurrent principle

3. Graph Boundaries

An important point to reduce the computation effort is obviously to restrain the area of study. In our case, as the state variable is the battery state of charge the area is limited by a maximum discharge limit and a maximum charge limit Fig. 3.

The maximum discharge limit at each time corresponds to the zero emission mode (electric mode). That means that the vehicle is propelled only by electricity using the energy stocked in the battery. In the case of regenerative braking, this limit corresponds to recharging the battery only by the wheel (no energy is added by the engine).

The maximum charge limit at each time is calculated considering that the engine provides all its possible power, a part is going to the wheel to propel the vehicle, the rest recharges the battery. Note that this remains true in the limits of systems and components capabilities.

The last limits to be applied are those of the battery SOC itself Fig. 3. On a hybrid vehicle the battery remains relatively small, thus it may be rapidly completely charged (SOC=100%) or discharged

(SOC=0%). Moreover, for battery ageing considerations, a battery management system usually limits the minimum and maximum value. Common values are (40% and 80 %,[20]).

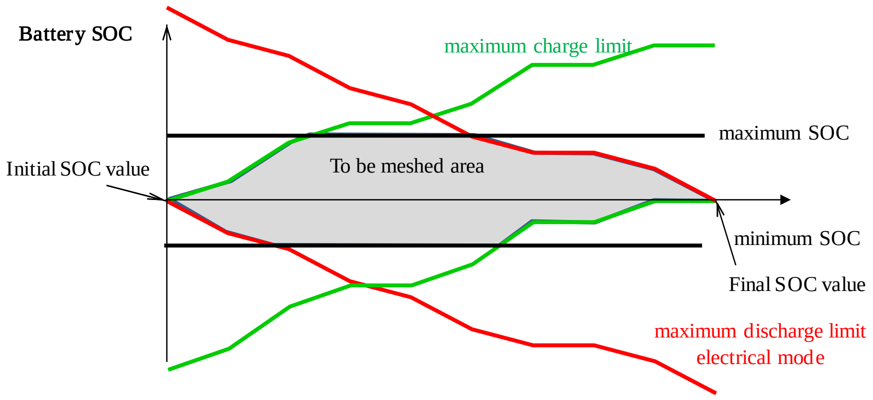

Fig. 3 Battery SOC limitation

Once the limits of the graph are defined, the graph has to be meshed in the two axes. The easiest way is to sample the time in N step of T_e starting at zero, and the SOC in M step between SOC_{min} and SOC_{max} starting at initial SOC value SOC_0, then you obtain a regular mesh (Fig. 5). The drawback of such a grid is that the electrical mode (corresponding to maximum discharge limits) has virtually no chance to correspond to an effective edge. In fact, as the soc variation δsoc_{elec} of this mode is perfectly known at each time by the drive cycle, it has few chances to match the SOC variation of an existing edge. This is of great importance; not considering electrical mode in hybrid vehicles implies great error on fuel consumption and does not allow making the strategy choice. A solution retained in [16] is to then consider that the nearest edge is the one corresponding to the electrical mode and to affect a cost of zero to this edge. Doing this creates errors on the SOC variations (Fig. 4) which are then cumulated along the whole cycle. Thus the final SOC constraint is not respected.

A solution is to mesh the area in such manner that the electrical mode corresponds to the edges of the graph (Fig. 5). For that, in each column, the grid starts from the maximum discharge limit (which is the electrical mode) and goes up. The SOC sample remains the same for each column.

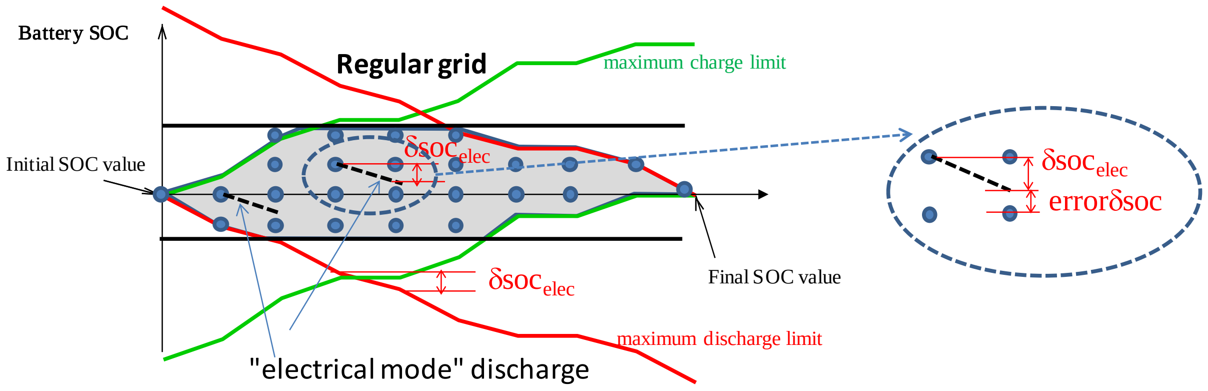

Fig. 4 Regular Grid

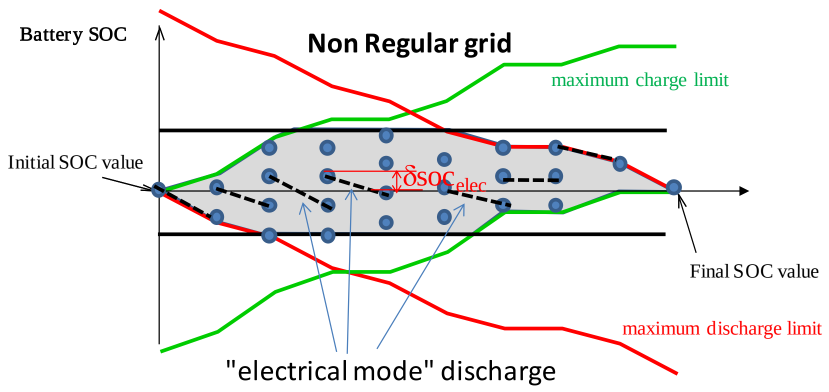

Fig. 5 Non regular grid

Once the graph boundaries have been constructed to minimize the meshed area (taking into account systems limitation) and to include all electrical mode, the cost of each edge of the graph has to be calculated.

4. Solving algorithms

Once the edge cost calculation is performed, the problem is to solve the DDP for a graph with a weighted oriented edge. Different manners can be imagined to solve this problem, from intuitive manner to complex matrix manner. The fact is that avoiding loops in the algorithms and too big matrix may be a good way to improve the algorithm efficiency especially using Matlab software.

This section deals with three different manners to solve the DDP problem:

(i) A first intuitive method (section 4.A) which consists in applying directly the principle described in section 2, using a loop on each point of a column inside the loop at each time k.

(ii) A matrix method (section 4.B) which treated each point of a column in a vectorial manner but used great sparse cost matrix with a lot of non defined elements.

(iii) A third method (section 4.C) based on dense matrix of index of cost and weight.

A. « Point to point » method

This method is one of the more intuitive. The idea, in the iterative principle, is to deal separately with each point of a column at time k+1. A loop is then used for each point of the column (algorithm Fig. 7). On a point i_2 of column k+1, the cost of each edge (fuel rate) which can reach this point is added to the cost to go of these points. The trajectory and the corresponding fuel consumption corresponding to the minimum fuel consumption $J^*_{(k+1,i2)}$ is selected.

The advantage of this method is that only the values of “valid” edges are considered. The main drawback is the loop on i_2 included in the time main loop on k (algorithm Fig. 7), and the presences of a call to a minimum function inside the loop. The computer effort used by this algorithm is detailed Table 3 and Table 4. An idea to reduce the computer effort consists in avoiding calling minimum function in the loop on i_2. For that, a matrix containing all the possible weights of each point of the column is built (section 4.B).

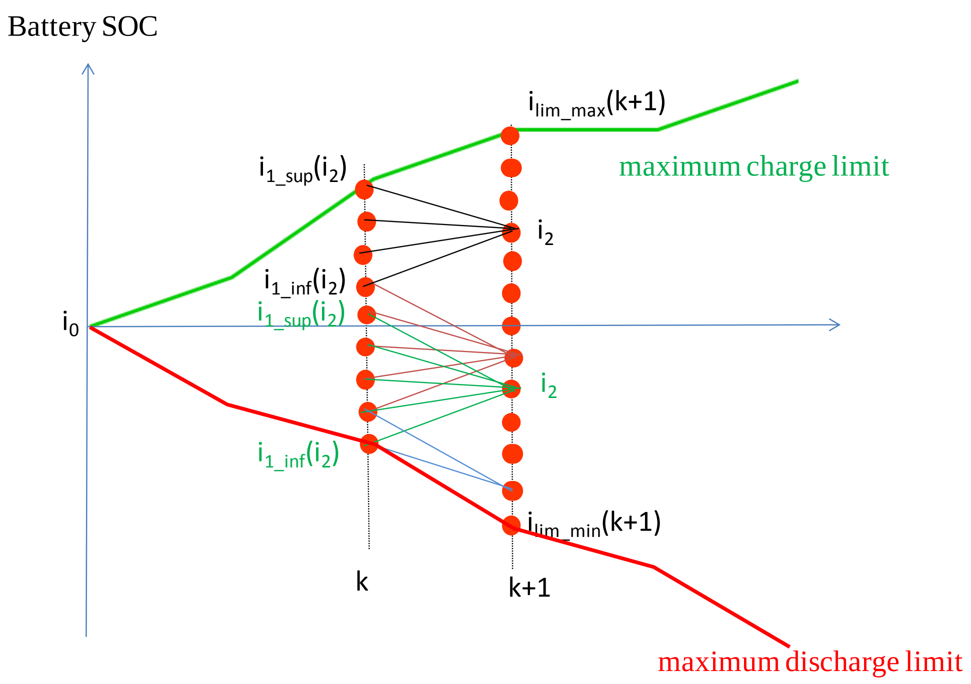

Fig. 6 Point to point method

```
$J^*_{(k+1,1:i_{2\_max})} = \infty$
for $k = 2 \to k_{max}$    (time step)
   for $i_2 = i_{lim\_min(k+1)} \to i_{lim\_max(k+1)}$    (gridde state value)
      (calculate the cost to go and the corresponding index)
      $[J^*_{(k+1,i_2)}, index_{(k+1,i_2)}] = \min[\dot{m}_{f\,(k,i_{1\_inf(i_2)}:i_{1\_sup(i_2)})\to(k+1,i_2)} + J^*_{(k,i_{1\_inf(i_2)}:i_{1\_sup(i_2)})}]$
   end
end
```

Fig. 7 Algorithm use in a point to point method

B. Matrix method

This method Fig. 8 is based on an M*M matrix calculation [21] with M the maximum point numbers in a column of the graph. The principle is at each time (k+1) to add the cost to go at time k to the cost of each edge from time k to time k+1. If the edge between two points is not possible (due to the systems operation) an infinite weight is applied. The advantages of this method are its simplicity and you obtain in one operation the best cost and the index of the best trajectory (Fig. 9). For that you only have to add the cost matrix C to the cost to go matrix J^*_k (composed of the best previous fuel consumption at time k, and take the minimum of each lines of this matrix in one call to a minimum function with a matrix on argument (Fig. 9).

The drawbacks are the size of the matrix, typically 1000^2 to 5000^2 elements. This implies great computer efforts to create the matrix and performed operation on it (Fig. 9). Moreover you still need a loop on i_2 inside the main loop to build the cost matrix C which is extracted of the global cost matrix C_m.

Using such a method leads to a highly sparse matrix (20 defined values on the diagonal on a 1000*1000 matrix for example). Thus another method using as much as possible dense smaller matrix is presented section 4.C.

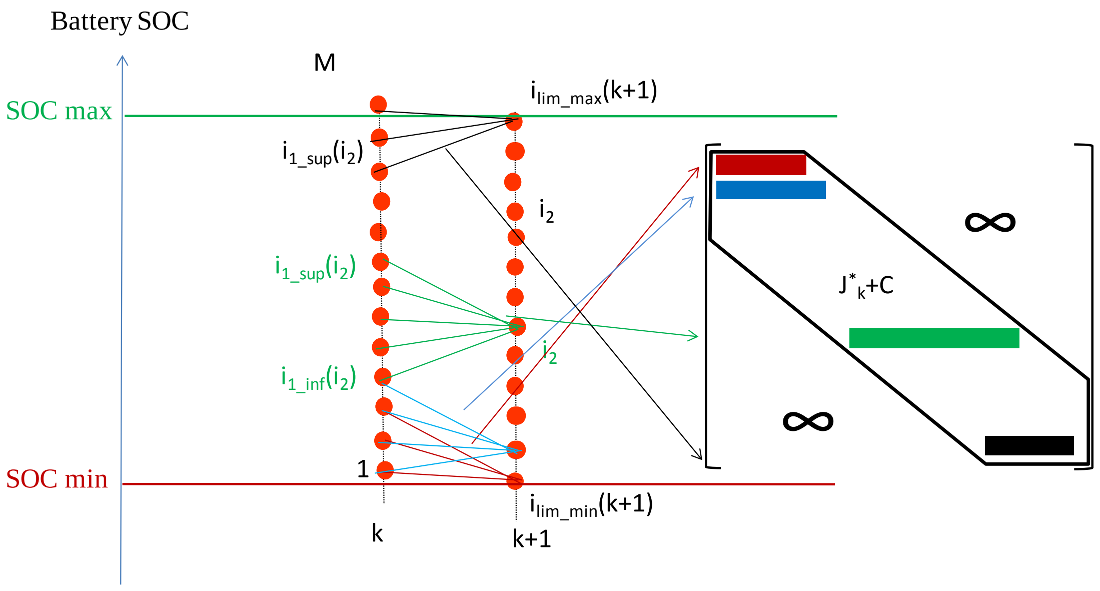

Fig. 8 Matrix method

```
for $k = 2 \rightarrow k_{\max}$    (time step)
    for $i_2 = i_{\lim\_\min(k)} \rightarrow i_{\lim\_\max(k)}$    (gridded state value)
        (building the cost matrix C)
        $C(i_{1\_\inf(i_2)} : i_{1\_\sup(i_2)}, i_2) = C_m(i_{e\_\inf(i_2)} : i_{e\_\sup(i_2)}, k)$
    end
    (calculate the cost to go for all point of the column, and the corresponding index)
    $[J^*_{(k+1,1:M)}, index_{(k+1,1:M)}] = \min[J^*_k + C]$
end
```

Fig. 9 Algorithm use in a matrix method

C. Index matrix method

The idea is instead of using sparse M*M matrix (section 4.B), using a smaller dense cost matrix. This matrix contains quite only defined value (Fig. 10) and has a size max(S_{max}(k)* M. In fact, the lines are only composed of the defined values of the spread of edge to reach points (k+1,i_2). I.e only the edges with effective fuel rate are considered. "Impossible" edges due to system capabilities and limitations are ignored. The key point is then to build the cost plus cost to go matrix (C+J^*_{k+1}) in an efficient manner (algo Fig. 11) and then take the minimums of each lines. Two index matrix $index_W$ and $index_C$ are built.

$Index_W$ (Fig. 10) contains on each line the indexes of the previous points at column k reaching a point i_2 of column k+1. Each line starts at the first defined value and is completed by non defined values if needed.

The matrix index$_C$ contains on each line the indexes in the spread of cost which reach point i_2. Note that the spread of cost is in fact the column k of a cost matrix C_m containing all costs of the graph.

Once these two indexes matrix are built (using a loop) the corresponding cost C_m(indexC) and previous best fuel consumption J^*_k(index$_W$) are added. The minimum of the lines of the resulting matrix gives the best fuel consumption at instant k $J^*_{(k,1:M)}$ and the best trajectories index (Fig. 11).

Such a method allows working with small dense matrix which leads to a computer effort gain in building matrix and matrix operations.

Note that an additional gain in computer effort can be performed building the indexes matrix column by column and not line by line. Thus the loop to construct it is smaller (S_{max} iteration instead of M), typically 20 operations instead of 1000. For simplicity reasons the algorithm is presented line by line but the computer effort is indicated using the column by column method.

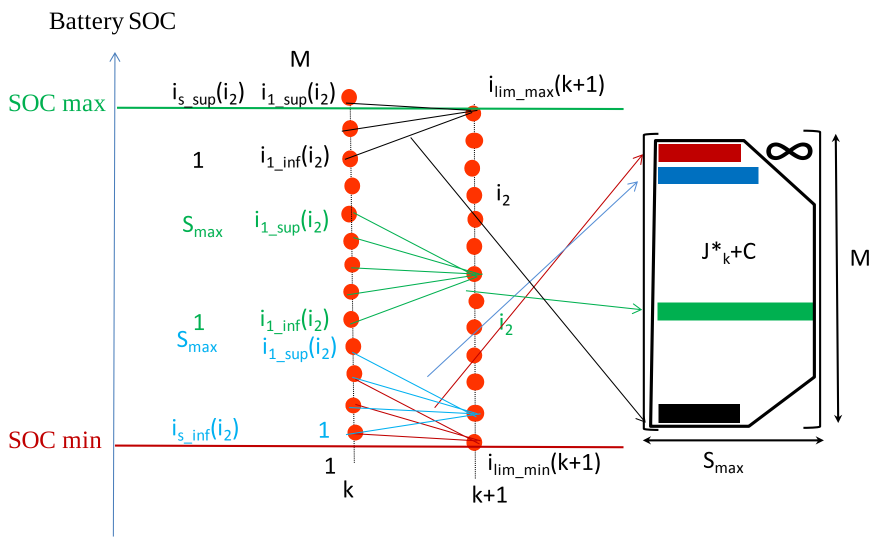

Fig. 10 index matrix method principle

```
for $k = 2 \rightarrow k_{max}$    (time step)
  $C = \infty$
  for $i_2 = i_{lim\_min(k)} \rightarrow i_{lim\_max(k)}$    (building index matrix)
    $index_C(1:i_{s\_sup(i_2)}, i_2) = i_{s\_inf(i_2)} : i_{s\_sup(i_2)}$
    $index_W(1:i_{s\_sup(i_2)}, i_2) = i_{1\_inf(i_2)} : i_{1\_sup(i_2)}$
  end
  (calculate the cost to go for each point of the column)
  $[J^*_{(k+1,1:M)}, index_{min}] = \min[J^*_k(index_W) + C_m(index_C)]$
  (determine the corresponding index)
  $index_{(k+1,1:M)} = index_W(index_{min})$
end
```

Fig. 11 index matrix method algorithm

Three methods to solve the DDP problems in the case of HEV have been developed using Matlab Software: a first intuitive one, an intuitive matrix method and a method using dense matrix. To provide effective comparison of the computer effort, the example of the well known Toyota PRIUS is presented in the following of the paper.

5. Application on the Toyota PRIUS-II

The DDP have been applied to different types of hybrid architectures [6][16][17] including different SP_HEV architectures [18]. In the scope of this paper, to provide data on computer effort during the process of optimisation, the well known Toyota PRIUS is chosen as an example.

The Toyota PRIUS II on sale since 2004 has been tested in our labs. Then the car and component characteristic have been inserted in our library of hybrid vehicle simulation [22][23] and presented a good set of parameters in the scope of this paper.

In the following, the Toyota Hybrid system architecture used in the PRIUS and the manner to calculate the cost is briefly remind. Thus the computer effort is detailed for each previous method and for different graph sampling.

A. THS architecture

The considered topology is that presented in Fig. 12. It is composed of one engine (ICE) and two electrical machines, one called EM1 and the other called EM2 fed by two electrical inverters (INV1 and INV 2) linked to a battery. A final axle and gear (FG) are linked to the wheels. A planetary gear (PG) enables to split the ICE energy mechanically to the electrical machine EM1 and to the wheels. The ICE is linked to the Planet carrier, the rotor of the electrical machine EM2 to the sun gear. The shaft of the FG ix connected both to the ring of the PG and to the rotor of EM1.

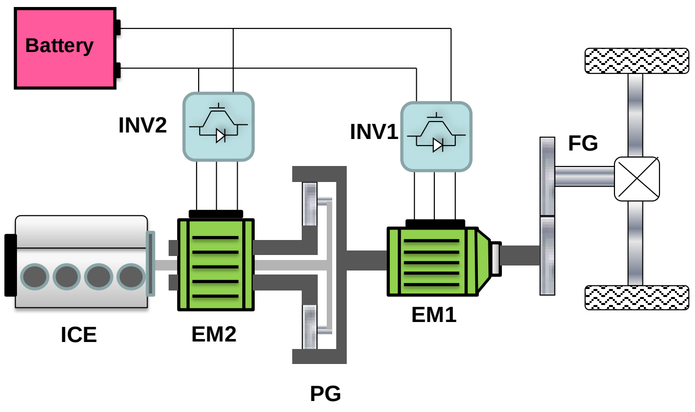

Fig. 12 THS architecture

B. Edges costs calculation

In DDP formulation, the cost of the edges, i.e. the fuel rate is performed using a backward approach Fig. 13, [18]. This is based on the knowledge of the driving cycle and the vehicle characteristics (inertia and resistant forces). Then one can easily calculate the wheel torque T_r with the vehicle speed (wheel speed, ω_r) and acceleration, and go upstream from the wheels to battery and engine.

The vehicle model is an energetic model based on the VEHLIB library [22],[23].

The electrical machines are modelled with efficiency maps and the engine with fuel consumption maps. The battery model is a resistive model with parameters dependant on SOC and current.

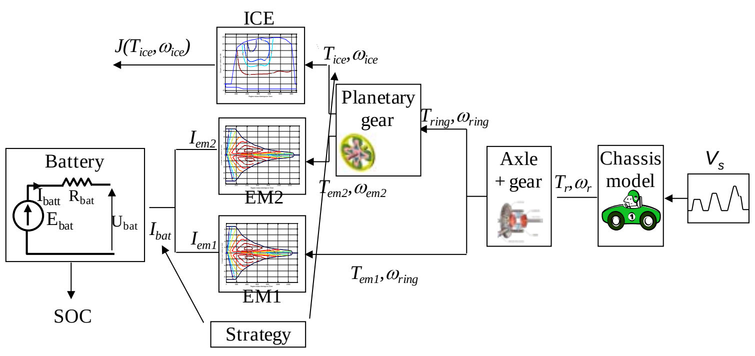

Fig. 13: Backward approach of THS architecture

On each edge of the graph, I_{bat}, T_r and ω_r are known. The equation of each component of the system has then to be used to calculate the engine operating point and thus its fuel consumption.

From the wheel torque and speed constraints (T_r,ω_r), one can easily deduce:

$$T_r = J_{veh}\frac{d\omega_r}{dt} + T_f \tag{5}$$

Where T_r is the wheel torque, T_f is the load torque calculated from the resistant forces, J_{veh} is the overall inertia of the vehicle brought back to the wheels and ω_r is the wheel speed.

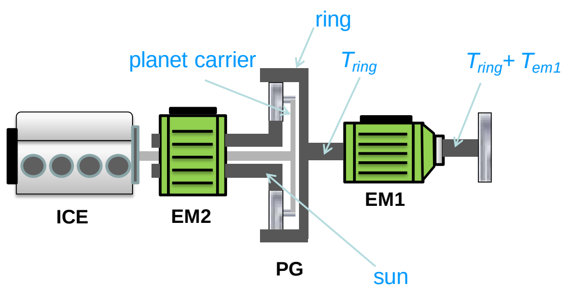

Fig. 14 Planetary gear connexions

Considering Fig. 13 and Fig. 14 the torque relation on the ring and EM1 shaft is:

$$T_{ring} + T_{em1} = \frac{T_r}{\eta^{sign(T_r \cdot \omega_r)} R_{trans}} \tag{6}$$

Where T_{ring} is the torque on the planetary gear ring, T_{em1} is the torque on the EM1 shaft, η is the efficiency of the transmission (axle plus gear), R_{trans} is the transmission ratio, *Sign*(T_r,ω_r) is the sign of the wheel power. Thus the wheel torque is divided or multiplied by the gear efficiency depending on the direction of the wheel power. Note that the electrical machine and planetary ring revolution is the same.

The Willis relation in the planetary gear imposes:

$$\omega_{em2} = k\omega_{ring} + (1-k)\omega_{ice} \tag{7}$$

Where ω_{em2}, ω_{ring} and ω_{ice} are respectively the speeds of the electrical machine EM2 (sun gear), the planetary ring (electrical machine EM1) and the engine (planet carrier). k is the planetary gear ratio (k=-$N_{teeth\_ring}/N_{teeth\_sun}$).

The torque relations on the planetary gear give:

$$T_{em2} = -T_{ice} / (1-k) \tag{8}$$

$$T_{em2} = -T_{ring} / (\eta_{plan} k) \tag{9}$$

Where T_{ice} is the ICE torque and η_{plan} is the efficiency of the planetary gear.

O the electrical node between the electrical machines and the battery:

$$I_{em1} + I_{em2} - I_{bat} = 0 \tag{10}$$

Where I_{em1}, I_{em2} and I_{bat} are respectively the DC current in electrical inverters of the EM1 and EM2 electrical machines (considered as receptors) and in battery (considered as a generator).

And finally the relations on the electrical machines impose:

$$(E - RI_{bat})I_{em1} = T_{em1}\omega_{em1} + Pt_{em1}(T_{em1}, \omega_{em1}) \tag{11}$$

$$(E - RI_{bat})I_{em2} = T_{em1}\omega_{em2} + Pt_{em2}(T_{em2}, \omega_{em2}) \tag{12}$$

Where Pt_{em1} and Pt_{em2} are the losses in the electrical machines and inverters, E is the open circuit voltage of the battey and R is the internal battery resistance which depends only on SOC. Note that ω_{em1} is the speed of the common shaft to EM1 and planetary ring and is known using final gear relation:

$$\omega_{em1} = \omega_{ring} = R_{trans}\omega_r \tag{ 13}$$

Considering (4) to (10), and I_{bat} fixed (on an edge) we obtain 7 equations and 8 unknown variables (ω_{em2}, ω_{ice}, T_{em2}, T_{ice}, T_{ring}, T_{em1}, I_{em2}, I_{em1}). That means one degree of freedom to be fixed. This degree of freedom allows choosing, for example, the engine speed to minimize the fuel consumption for a given edge (I_{bat}, T_r and ω_r imposed), [18].

The process of cost calculation is totally independent on the solving of the graph. And is performed once and for all edge of the graph before the DDP process. The computer effort for the calculation of the edges costs and graph solving is then easy to determine separately.

C. Computer effort

This part presents the computer effort to solve DDP for the three previously presented methods. The computer effort is indicated in second and is calculated using the Matlab instruction “cputime”. The simulations are performed on cpu with the characteristic presented Table 1. The bogomips calculated by Linux give a good indication of the potential million of instructions per second using this processor. This can allow a comparison with other types of processors.

The computer effort is presented Table 3 and Table 4 for different driving cycles representing urban, road and highway driving condition. The NEDC normalized European driving cycle is also presented.

It is noted that the memory used with the different methods remains small (less than 2Go) and well under the capacity of the cpu.

Two different graph precisions are used. It represents the SOC difference in % between two consecutive points in a column. The time sample is fixed to 1s to each cycle.

Table 3 and Table 4 shows that the third method is the most efficient to solve DDP applied to the hybrid vehicle. In fact the time to solve the DDP can be divided by a factor two to six depending on the cycle and the precision. A method using "full" matrix (method B), can improve the time calculation compared to point to point method (method A) if the graph precision is not too high but became definitely too expensive in computer effort as the size of the matrix (graph precision) increases. It is also noted that an increase of the graph precision below 0.01% does not change significantly the fuel consumption. Using such precision, the total time calculation of DDP (edge cost calculation and DDP solving) does not exceed 10 s and can be reduced to 2 s in small cycle. It is then possible to use DDP as a tool to determine fuel consumption in more a complex process [11].

Table 1 Cpu characteristic

Model name	Intel Xeon X5650
Cpu MHz	2660
Cache size	12.3 KB
Memory	24 GB
bogomips	5320

Table 2 Graph information

SOC max	80 %
SOC min	20 %
Speed engine sample	50.π/30 rad/s
Torque engine sample	5 Nm

Table 3 Computer effort for different driving cycle with a graph precision of 0.05%

	drive cycle			
	URBAN	ROAD	HIGHWAY	NEDC
graph precision	**-0.05**			
time of cycle in s	560	843	1805	1181
million of edge	3.5	6.3	18	7.9
time of edge cost calculation in s	1	1.9	5.1	2.5
fuel consumption in l/100 km	3.772	4.124	5.309	3.720
	time to solve DP in s			
A: point to point method	3.8	7.2	18.7	10.4
B: Matrix method	3.0	4.9	13.5	6.9
C: Index matrix method	0.6	1.2	4.6	1.9

Table 4 Computer effort for different driving cycle with a graph precision of 0.01%

	drive cycle			
	URBAN	ROAD	HIGHWAY	NEDC
graph precision	**-0.01**			
time of cycle in s	560	843	1805	1181
million of edge	86	151	435	194
time of edge cost calculation in s	4.5	9.3	25	11.6
fuel consumption in l/100 km	3.768	4.118	5.299	3.714
	time to solve DP in s			
A: point to point method	20.3	38.9	104.0	54.0
B: Matrix method	68.0	106.1	248.0	149.0
C: Index matrix method	7.9	19.8	62.6	23.6

6. Conclusion

Different methods to solve dynamic programming applied to hybrid vehicle have been studied; an intuitive method using a “point to point” approach, a matrix based method easy to implement but using full sparse matrix and a method based on dense matrix.

The last method takes advantage of one particularity of DDP applied to the hybrid vehicle: the spread of possible edge from one point of a column to the next column is highly limited by the system capacities. The number of edges from one point is thus small compared to the number of points in a column.

The last method is then the most efficient in terms of computer effort. In fact it allows to use small index matrix (compared to method 2), and avoid to call a minimum function in loops (column loops inserted in time loops).

The tool developed to calculate the fuel consumption by means of DDP has been used and is currently used in more complex studies. With a global time calculation less than 30s (usually less than 5s) it can be used in comparative hybrid architecture studies and even in global optimal design architecture process.

8. References

[1] I. Husain, Electric and hybrid vehicles: design fundamentals, CRC press, 2011.

[2] C. Chan, «The state of the art of electric and hybrid vehicles,» *Proceedings of the IEEE,* vol. 90, n° 12, pp. 247-275, 2002.

[3] J. M. Miller, Propulsion systems for hybrid vehicles, vol. 45, Iet, 2004.

[4] K. Chau et Y. Wong, «Overview of power management in hybrid electric vehicles,» *Energy conversion and management,* vol. 43, n° 115, pp. 1953-1968, 2002.

[5] A. Sciarretta et L. Guzzella, «Control of hybrid electric vehicles,» *Control systems, IEEE,* vol. 27, n° 12, pp. 60-70, 2007.

[6] M. Koot, J. Kessels, B. de Jager, W. Heemels, P. Van den Bosch et M. Steinbuch, «Energy management strategies for vehicular electric power systems,» *Vehicular Technology, IEEE Transactions on,* vol. 54, n° 13, pp. 771-782, 2005.

[7] E. Vinot, R. Trigui, B. Jeanneret, J. Scordia et F. Badin, «HEVs comparison and components sizing using dynamic programming,» chez *Vehicle Power and Propulsion Conference, 2007. VPPC 2007. IEEE*, 2007.

[8] C. R. Akli, B. Sareni, X. Roboam et A. Jeunesse, «Integrated optimal design of a hybrid locomotive with

multiobjective genetic algorithms,» *International Journal of Applied Electromagnetics and Mechanics*, vol. 30, n° 13, pp. 151-162, 2009.

[9] M.-J. Kim et H. Peng, «Power management and design optimization of fuel cell/battery hybrid vehicles,» *Journal of Power Sources,* vol. 165, n° 12, pp. 819-832, 2007.

[10] C. Romaus, K. Gathmann et J. Bocker, «Optimal energy management for a hybrid energy storage system for electric vehicles based on stochastic dynamic programming,» *Vehicle Power and Propulsion Conference (VPPC), 2010 IEEE*, 2010.

[11] V. Reinbold, E. Vinot et L. Gerbaud, «Global optimization of a parallel hybrid vehicle using optimal energy management,» *International Journal of Applied Electromagnetics and Mechanics,* vol. 43, n° 11, pp. 115-126, 2013.

[12] C. Desai et S. S. Williamson, «Optimal design of a parallel Hybrid Electric Vehicle using multi-objective genetic algorithms,» chez *Vehicle Power and Propulsion Conference, 2009. VPPC'09. IEEE*, 2009.

[13] S. Delprat, J. Lauber, T.-M. Guerra et J. Rimaux, «Control of a parallel hybrid powertrain: optimal control,» *Vehicular Technology, IEEE Transactions on,* vol. 53, n° 13, pp. 872-881, 2004.

[14] R. Bellman, Dynamic Programming, Princeton University Press, 1957.

[15] D. E. Kirk, Optimal control theory, Dover publications Inc, 2012.

[16] J. Scordia, M. Desbois-Renaudin, R. Trigui, B. Jeanneret, F. Badin et C. Plasse, «Global optimisation of energy management laws in hybrid vehicles using dynamic programming,» *International journal of vehicle design,* vol. 39, n° 14, pp. 349-367, 2005.

[17] L. V. Perez, G. R. Bossio, D. Moitre et G. O. Garcia, «Optimization of power management in an hybrid electric vehicle using dynamic programming,» *Mathematics and Computers in Simulation,* vol. 73, n° 11, pp. 244-254, 2006.

[18] E. Vinot, R. Trigui, Y. Cheng, C. Espanet, A. Bouscayrol et V. Reinbold, «Improvement of an EVT-based HEV using dynamic programming,» *IEEE Transactions on Vehicular Technology,* vol. 63, n° 11, pp. 40-50, 2014.

[19] M. Kamiya, «Development of traction drive motors for the Toyota hybrid system,» *IEEJ TRANSACTIONS ON INDUSTRY APPLICATIONS D,* vol. 126, n° 14, p. 473, 2006.

[20] E. Vinot, J. Scordia, R. Trigui, B. Jeanneret et F. Badin, «Model simulation, validation and case study of the 2004 THS of Toyota Prius,» *International Journal of Vehicle Systems Modelling and Testing,* vol. 3, n° 13, pp. 139-167, 2008.

[21] F. Mensing, R. Trigui et E. Bideaux, «Vehicle trajectory optimization for application in ECO-driving,» chez *Vehicle Power and Propulsion Conference (VPPC), 2011 IEEE,* 2011.

[22] B. Jeanneret, R. Trigui, F. Badin et F. Harel, «New Hybrid concept simulation tools, evaluation on the Toyota Prius car,» chez *16th International electric vehicle symposium, Beijing, China, 1999*, China, 1999.

[23] R. Trigui, M. Desbois-Renaudin et B. Jeanneret, «Global Forward-Bacward Approach for a Systematic Analysis and Implementation,» *EET2004, march 18-20 th, Estoril, Protugal*, 2004.