Bi-level optimal control method and application on hybrid electric vehicles torque split problem

Rémy Dutto [1,2,3]

[1]IRIT: Toulouse IT Institute

[2]IMT: Toulouse Mathematics Institute

[3]Vitesco Technologies

Journées annuelles 2023 du GdR MOA, Université Perpignan Via Domitia

Introduction

In collaboration with:

- Olivier Cots, IRIT, Toulouse
- Olivier Flebus, Vitesco Technologies, Toulouse
- Sophie Jan, IMT, Toulouse
- Serge Laporte, IMT, Toulouse
- Mariano Sans, Vitesco Technologies, Toulouse

Outline

Cycle

We consider an Hybrid Electric Vehicle (HEV) on a predefined cycle, i.e. speed and slope trajectories are prescribed.

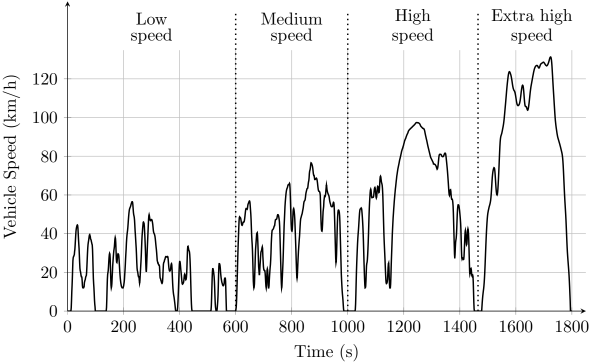


Figure: Worldwide harmonized Light vehicles Test Cycle (WLTC).

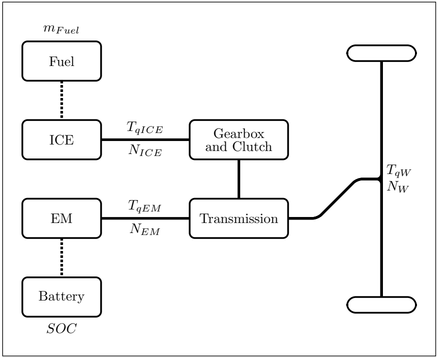
Requested wheels torque $T_{qW}(t)$ and rotation speed $N_W(t)$ are obtained with the information of our vehicle (mass, wheel diameter, aerodynamic coefficient. . .).

Static model

Inputs of our static model:

Name	Description	Unit
Cost		
m_{Fuel}	Fuel consumption	g
State		
SOC	Battery state of charge	
Commands		
$Gear$	Gearbox selector	
T_{qICE}	ICE torque	N.m
External inputs		
T_{qW}	Wheels torque	N.m
N_W	Wheels rotation speed	RPM

Figure: Schema of the selected HEV.



Outputs: $\dot{m}_{Fuel}$ and $\dot{SOC}$, where $\dot{}$ stands for $\frac{\mathrm{d}}{\mathrm{d}t}$.

Optimal control problem formulation

Objective: Minimize fuel consumption

The following Lagrange optimal control problem is considered:

$$(\text{OCP}) : \begin{cases} \displaystyle\min_{x,u} & \displaystyle\int_{t_0}^{t_f} f^0\big(t, x(t), u(t)\big)\, dt, \\ \text{s.t.} & \dot{x}(t) = f\big(t, x(t), u(t)\big) \qquad t \in [t_0, t_f] \text{ a.e.}, \\ & u(t) \in U(t) \qquad \forall t \in [t_0, t_f], \\ & x(t_0) = x_0, \quad x(t_f) = x_f, \end{cases}$$

where for all $t \in [t_0, t_f]$:

- $x(t) = SOC \in \mathbb{R}^n, n = 1$
- $u(t) = \big(T_{qICE}, Gear\big) \in \mathbb{R}^m, m = 2$
- f^0 is the instantaneous fuel consumption function
- f describes the instantaneous evolution of the state of charge

Remark: f^0 and f are C^1 with respect to x and u.

Model properties

This problem and its implementation are complex:

Model properties

This problem and its implementation are complex:

- Non autonomous (N_w and T_{qW})

Model properties

This problem and its implementation are complex:

- Non autonomous (N_w and T_{qW})
- Discrete (*Gear*) and continuous commands (T_{qICE})

Model properties

This problem and its implementation are complex:

- Non autonomous (N_w and T_{qW})
- Discrete ($Gear$) and continuous commands (T_{qICE})
- Command bounds $U(t)$ (ICE and EM rotation speeds, battery current ...)

Model properties

This problem and its implementation are complex:

- Non autonomous (N_w and T_{qW})
- Discrete ($Gear$) and continuous commands (T_{qICE})
- Command bounds $U(t)$ (ICE and EM rotation speeds, battery current . . .)
- Tabulated data (torque losses, fuel consumption . . .)

Model properties

This problem and its implementation are complex:

- Non autonomous (N_w and T_{qW})
- Discrete ($Gear$) and continuous commands (T_{qICE})
- Command bounds $U(t)$ (ICE and EM rotation speeds, battery current ...)
- Tabulated data (torque losses, fuel consumption ...)
- Time horizon much larger than integration time step size Δt

Model properties

This problem and its implementation are complex:

- Non autonomous (N_w and T_{qW})
- Discrete ($Gear$) and continuous commands (T_{qICE})
- Command bounds $U(t)$ (ICE and EM rotation speeds, battery current ...)
- Tabulated data (torque losses, fuel consumption ...)
- Time horizon much larger than integration time step size Δt
- Coded in Matlab Simulink

Pontryagin's Maximum Principle

If (x, u) is solution of (OCP), it exists $p \in \mathrm{AC}([t_0, t_f], \mathbb{R}^n)$ and $p^0 \in \{-1, 0\}$ such that $(p, p^0) \neq 0$,

$$\begin{aligned} \dot{x}(t) &= \nabla_p h\big(t, x(t), p(t), u(t)\big) \qquad t \in [t_0, t_f] \text{ a.e.}, \\ \dot{p}(t) &= -\nabla_x h\big(t, x(t), p(t), u(t)\big) \quad t \in [t_0, t_f] \text{ a.e.}, \end{aligned}$$

and such that the maximisation condition is satisfied

$$h\,(t, x(t), p(t), u(t)) = \max_{u \in U(t)} h\,(t, x(t), p(t), u) \qquad t \in [t_0, t_f] \text{ a.e.},$$

where $h(t, x, p, u) = p^0 \cdot f^0(t, x, u) + p \cdot f(t, x, u)$ is the *pseudo-Hamiltonian*.

Hypothesis 1

If (x, u) is a solution of (OCP) then the associated extremal (x, p) is normal, i.e. $p^0 = -1$.

Pseudo-Hamiltonian system

With the notation $z = (x, p)$, assuming the *Hamiltonian*

$$H(t, z) = \max_{u \in U(t)} h(t, z, u)$$

is defined and smooth, the *Hamiltonian vector field* is computed as follows:

$$\vec{H}(t, z) = \big(\nabla_p H(t, z), -\nabla_x H(t, z)\big)$$

The *exponential map* $\exp_{\vec{H}}(t_1, t_0, z_0)$ is the solution at time t_1 of the Cauchy problem

$$\begin{cases} \dot{z}(t) = \vec{H}\left(t, z(t)\right), \\ \text{s.t.} \quad z(t_0) = z_0, \end{cases}$$

Indirect simple shooting

The Pontryagin's Maximum Principle gives necessary conditions leading to the resolution of the following Two Points Boundary Value Problem

$$(\text{TPBVP}) : \begin{cases} z_f = \exp_{\vec{H}}(t_f, t_0, z_0) \\ \text{s.t.} \;\; \pi_x(z_0) = x_0, \\ \qquad\;\; \pi_x(z_f) = x_f, \end{cases}$$

where $\pi_x(x, p) = x$.

The indirect simple shooting method aims to solve the (TPBVP) and is defined as finding a zero of the *shooting function*

$$\begin{array}{rccc} S_s & : & \mathbb{R}^{2n} & \longrightarrow & \mathbb{R}^{2n} \\ & & z_0 & \longmapsto & \begin{pmatrix} \pi_x(z_0) - x_0 \\ \pi_x\left(\exp_{\vec{H}}(t_f, t_0, z_0)\right) - x_f \end{pmatrix}. \end{array}$$

Motivations

The HEVs torque split and gear shift problem was solved by indirect simple shooting method.

We aim to:

- Speed up the computation
- Decrease the number of computations
- Reduce the sensitivity of the shooting function

Indirect multiple shooting

The time interval $[t_0, t_f]$ is decomposed into $t_0 < t_1 < \cdots < t_N < t_{N+1} = t_f$.

[1] H.G. Bock and K.J. Plitt. A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems. *IFAC Proceedings Volumes*, 17(2):1603–1608, 1984

Indirect multiple shooting

The time interval $[t_0, t_f]$ is decomposed into $t_0 < t_1 < \cdots < t_N < t_{N+1} = t_f$. (TPBVP) is transformed to

$$\text{(MPBVP)} : \begin{cases} \forall i = 0, \ldots, N, \quad z_{i+1} = \exp_{\vec{H}}(t_{i+1}, t_i, z_i), \\ \text{s.t.} \quad \pi_x(z_0) = x_0, \quad \pi_x(z_{N+1}) = x_f. \end{cases} \tag{1}$$

[1] H.G. Bock and K.J. Plitt. A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems. *IFAC Proceedings Volumes*, 17(2):1603–1608, 1984

Indirect multiple shooting

The time interval $[t_0, t_f]$ is decomposed into $t_0 < t_1 < \cdots < t_N < t_{N+1} = t_f$. (TPBVP) is transformed to

$$\text{(MPBVP)} : \begin{cases} \forall i = 0, \ldots, N, \quad z_{i+1} = \exp_{\vec{H}}(t_{i+1}, t_i, z_i), \\ \text{s.t.} \quad \pi_x(z_0) = x_0, \quad \pi_x(z_{N+1}) = x_f. \end{cases} \tag{1}$$

The corresponding shooting function is therefore

$$\begin{array}{rccc} S_m \quad : & \mathbb{R}^{2n(N+1)} & \longrightarrow & \mathbb{R}^{2n(N+1)} \\ & \begin{pmatrix} z_0 \\ z_1 \\ \vdots \\ z_{N-1} \\ z_N \end{pmatrix} & \longmapsto & \begin{pmatrix} \pi_x(z_0) - x_0 \\ \exp_{\vec{H}}(t_1, t_0, z_0) - z_1 \\ \vdots \\ \exp_{\vec{H}}(t_N, t_{N-1}, z_{N-1}) - z_N \\ \pi_x\left(\exp_{\vec{H}}(t_{N+1}, t_N, z_N)\right) - x_f \end{pmatrix}. \end{array} \tag{2}$$

S_m is known to be less sensitive to the initial guess than S_s.[1]

[1] H.G. Bock and K.J. Plitt. A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems. *IFAC Proceedings Volumes*, 17(2):1603–1608, 1984

Bi-level formulation

(OCP) is transformed into the equivalent *Bi-level Optimal Control Problem*:

$$
\text{(BOCP)} : \begin{cases} \displaystyle\min_{X \in \mathcal{X}} \; V(X) = \sum_{i=0}^{N} V_i\left(X_i, X_{i+1}\right) \\ \text{s.t.} \quad X_0 = x_0, \quad X_{N+1} = x_f \end{cases}
$$

where $X = (X_0, \ldots, X_{N+1})$, $\mathcal{X}$ is the domain of admissible intermediate states

Bi-level formulation

(OCP) is transformed into the equivalent *Bi-level Optimal Control Problem*:

$$(\text{BOCP}) : \begin{cases} \displaystyle\min_{X \in \mathcal{X}} \; V(X) = \sum_{i=0}^{N} V_i\left(X_i, X_{i+1}\right) \\ \text{s.t.} \quad X_0 = x_0, \quad X_{N+1} = x_f \end{cases}$$

where $X = (X_0, \dots, X_{N+1})$, $\mathcal{X}$ is the domain of admissible intermediate states and V_i is the optimal value of $(\mathrm{OCP}_{\mathrm{i,a,b}})$, where

$$(\mathrm{OCP}_{\mathrm{i,a,b}}) : \begin{cases} V_i\left(a, b\right) = \displaystyle\min_{x,u} \int_{t_i}^{t_{i+1}} f^0\left(t, x(t), u(t)\right) dt & \\ \text{s.t.} \qquad \dot{x}(t) = f\left(t, x(t), u(t)\right) & t \in [t_i, t_{i+1}] \text{ a.e.}, \\ \qquad\qquad u(t) \in U(t) & \forall t \in [t_i, t_{i+1}], \\ \qquad\qquad x(t_i) = a, \quad x(t_{i+1}) = b. & \end{cases}$$

Link with other methods

$N+1$: number of intervals and value functions $\,/\,\Delta t$: integration time step size

Condition	Problem	Methods
$N=0$	TPBVP	Simple shooting

Link with other methods

$N+1$: number of intervals and value functions / Δt : integration time step size

Condition	Problem	Methods
$N = 0$	TPBVP	Simple shooting
$N = \dfrac{t_f - t_0}{\Delta t}$	Optimization	DP or Direct

Link with other methods

$N+1$: number of intervals and value functions $\big/$ Δt : integration time step size

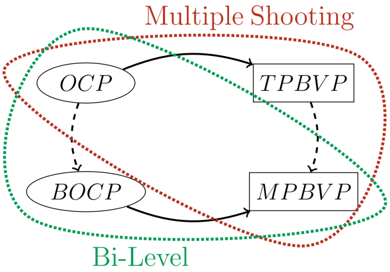
Condition	Problem	Methods
$N=0$	TPBVP	Simple shooting
$N=\dfrac{t_f-t_0}{\Delta t}$	Optimization	DP or Direct
Else	MPBVP	"Multiple shooting"

Link with other methods

$N+1$: number of intervals and value functions / Δt : integration time step size

Condition	Problem	Methods
$N = 0$	TPBVP	Simple shooting
$N = \dfrac{t_f - t_0}{\Delta t}$	Optimization	DP or Direct
Else	MPBVP	"Multiple shooting"

"Multiple shooting": another way to get the same problem:



Commutative diagram: Theorem

Theorem 1

Under suitable regularity assumption, the Pontryagin's co-states and the value function satisfy the following relations:[1]

$$\forall i \in [\![0, N]\!], \quad \nabla_a V_i\big(x(t_i), x(t_{i+1})\big) = -p(t_i)$$

$$\forall i \in [\![0, N]\!], \quad \nabla_b V_i\big(x(t_i), x(t_{i+1}\big) = p(t_{i+1})$$

where (x, u) *is a solution of* $(OCP_{i,a,b})$ *and* (x, p) *an associated extremal.*

[1] Frank H. Clarke and Richard B. Vinter. The Relationship between the Maximum Principle and Dynamic Programming. *SIAM Journal on Control and Optimization*, 25(5):1291–1311, 1987

Commutative diagram: Necessary conditions

Denoting $\lambda = (\lambda_0, \lambda_f)$, the Lagrangian of (BOCP) is

$$L(X, \lambda) = \sum_{i=0}^{N} V_i\left(X_i, X_{i+1}\right) - \lambda_0(X_0 - x_0) - \lambda_f(X_{N+1} - x_f).$$

Commutative diagram: Necessary conditions

Denoting $\lambda = (\lambda_0, \lambda_f)$, the Lagrangian of (BOCP) is

$$L(X, \lambda) = \sum_{i=0}^{N} V_i\left(X_i, X_{i+1}\right) - \lambda_0(X_0 - x_0) - \lambda_f(X_{N+1} - x_f).$$

If X is solution of (BOCP), we have $\forall i \in \{1, ..., N\}$

$$\left(\begin{array}{c} \text{KKT} \\ \text{Conditions} \end{array} \right) \implies \left\{ \begin{array}{r} \nabla_a V_0(X_0, X_1) - \lambda_0 = 0 \\ \nabla_b V_{i-1}(X_{i-1}, X_i) + \nabla_a V_i(X_i, X_{i+1}) = 0 \\ \nabla_b V_N(X_N, X_{N+1}) - \lambda_f = 0 \end{array} \right.$$

Commutative diagram: Necessary conditions

Denoting $\lambda = (\lambda_0, \lambda_f)$, the Lagrangian of (BOCP) is

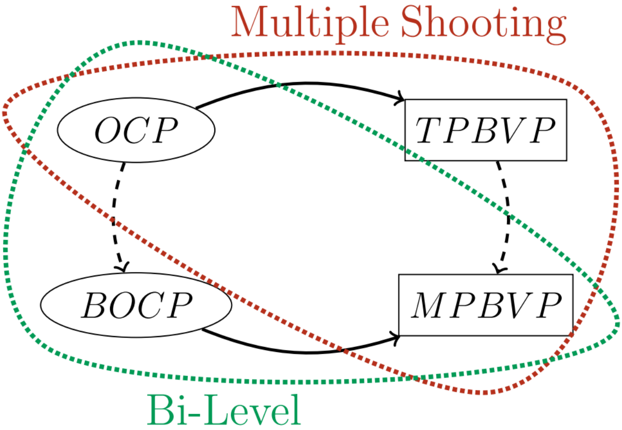
$$L(X, \lambda) = \sum_{i=0}^{N} V_i\left(X_i, X_{i+1}\right) - \lambda_0(X_0 - x_0) - \lambda_f(X_{N+1} - x_f).$$

If X is solution of (BOCP), we have $\forall i \in \{1, ..., N\}$

$$\begin{pmatrix} \text{KKT} \\ \text{Conditions} \end{pmatrix} \implies \begin{cases} \nabla_a V_0(X_0, X_1) - \lambda_0 = 0 \\ \nabla_b V_{i-1}(X_{i-1}, X_i) + \nabla_a V_i(X_i, X_{i+1}) = 0 \\ \nabla_b V_N(X_N, X_{N+1}) - \lambda_f = 0 \end{cases}$$

$$\left(+ \text{ Theorem 1} \right) \implies \begin{cases} p_0(t_0) + \lambda_0 = 0 \\ -p_{i-1}(t_i) + p_i(t_i) = 0 \\ -p_N(t_{N+1}) + \lambda_f = 0 \end{cases}$$

Commutative diagram



Main idea

Let's assume that the value functions V_i are known a priori.
(BOCP) becomes an optimization problem

$$\text{(Macro)} : \begin{cases} \min\limits_{X \in \mathcal{X}} \; V(X) = \sum\limits_{i=0}^{N} V_i\left(X_i, X_{i+1}\right) \\ \text{s.t.} \quad X_0 = x_0, \quad X_{N+1} = x_f, \end{cases}$$

to get the optimal intermediate states $X^\star = (X_1^\star, \ldots, X_N^\star)$

Main idea

Let's assume that the value functions V_i are known a priori.
(BOCP) becomes an optimization problem

$$\text{(Macro)} : \begin{cases} \displaystyle\min_{X \in \mathcal{X}} \; V(X) = \sum_{i=0}^{N} V_i\left(X_i, X_{i+1}\right) \\ \text{s.t.} \quad X_0 = x_0, \quad X_{N+1} = x_f, \end{cases}$$

to get the optimal intermediate states $X^\star = (X_1^\star, \ldots, X_N^\star)$ and $N+1$ independent optimal control problems

$$\text{(Micro)} : \begin{cases} \displaystyle\min_{x,u} \; \int_{t_i}^{t_{i+1}} f^0\left(t, x(t), u(t)\right) dt \\ \text{s.t.} \quad \dot{x}(t) = f\left(t, x(t), u(t)\right), \qquad t \in [t_i, t_{i+1}] \text{ a.e.,} \\ \qquad\; u(t) \in U(t), \qquad \forall t \in [t_i, t_{i+1}], \\ \qquad\; x(t_i) = X_i^\star, \quad x(t_{i+1}) = X_{i+1}^\star. \end{cases}$$

where $\left(X_i^\star, -\nabla_a V_i(X_i^\star, X_{i+1}^\star)\right)$ is a solution of the associated TPBVP

Proposed approach

The proposed approach is based on an approximation C_i of the value function V_i.

Proposed approach

The proposed approach is based on an approximation C_i of the value function V_i.

(BOCP) becomes an optimization problem

$$(\text{Macro}) : \begin{cases} C(X) = \min_{X \in \mathcal{X}} \displaystyle\sum_{i=0}^{N} C_i\left(X_i, X_{i+1}\right) \\ \text{s.t.} \qquad\qquad X_0 = x_0, \quad X_{N+1} = x_f, \end{cases}$$

to get the (optimal ?) intermediate states $\hat{X} = \left(\hat{X}_1, \ldots, \hat{X}_N\right)$ and $N+1$ independent optimal control problems

$$(\text{Micro}) : \begin{cases} \min_{x,u} \displaystyle\int_{t_i}^{t_{i+1}} f^0\left(t, x(t), u(t)\right) dt \\ \text{s.t.} \quad \dot{x}(t) = f\left(t, x(t), u(t)\right), & t \in [t_i, t_{i+1}] \text{ a.e.}, \\ \qquad u(t) \in U(t), & \forall t \in [t_i, t_{i+1}], \\ \qquad x(t_i) = X_i, \quad x(t_{i+1}) = X_{i+1}. \end{cases}$$

$\left(\hat{X}_i, -\nabla_a C_i(\hat{X}_i, \hat{X}_{i+1})\right)$ is not necessary a solution of the associated TPBVP

Error analysis

How to control the cost error due to the approximation ?

We denote $\tilde{\mathcal{X}} = \{X \in \mathcal{X} \mid X_0 = x_0, \quad X_{N+1} = x_f\}$

Proposition 1

Let $X^\star \in \underset{\tilde{\mathcal{X}}}{\arg\min}\, V$ *and* $\hat{X} \in \underset{\tilde{\mathcal{X}}}{\arg\min}\, C$*. If*

$$\forall X \in \tilde{\mathcal{X}}, \quad \forall i \in [\![0, N]\!], \quad \left|V_i(X) - C_i(X)\right| \leq \frac{\epsilon}{2(N+1)}$$

then

$$\left|V(X^\star) - V(\hat{X})\right| \leq \epsilon$$

Pseudo-Hamiltonian system

Due to the numerical implementation, the maximized Hamiltonian cannot be easily computed.

Pseudo-Hamiltonian system

Due to the numerical implementation, the maximized Hamiltonian cannot be easily computed.

The *maximizing control* is computed (assuming the arg max unique)

$$u^*(t,z) = \arg\max \left\{ h\left(t,z,u\right), u \in \tilde{U}(t) \right\}$$

where $\tilde{U}(t)$ is a discretization of $U(t)$.

Pseudo-Hamiltonian system

Due to the numerical implementation, the maximized Hamiltonian cannot be easily computed.

The *maximizing control* is computed (assuming the arg max unique)

$$u^*(t,z) = \arg\max \left\{ h\,(t,z,u)\,,\, u \in \tilde{U}(t) \right\}$$

where $\tilde{U}(t)$ is a discretization of $U(t)$. The pseudo-Hamiltonian vector field is

computed as follows:

$$\vec{H}(t,z) = \big(\nabla_p h\big(t,z,u^*(t,z)\big), -\nabla_x h\big(t,z,u^*(t,z)\big)\big)$$

where $\nabla_x h$ is calculated by finite differences.

How to compute C_i ?

Pseudo-Hamiltonian flow database

A database of extremals is created by computing the flow of $\vec{H}$ over $[t_i, t_{i+1}]$, $\forall i \in [\![0, N]\!]$ and for all z_0 in a discretization of the phase space.

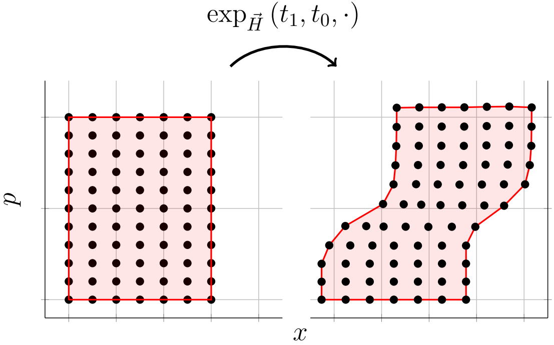


Figure: Example of Hamiltonian flow.

For each time interval $[t_i, t_{i+1}]$, we create a database of 1275 extremals.

Cost transition functions C_i

Each transition cost C_i is modeled by a simple smooth neural network

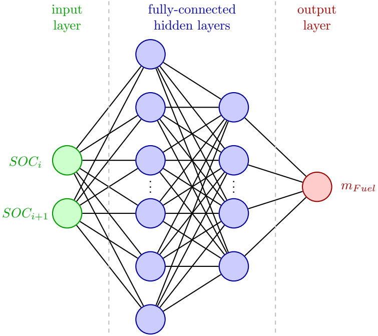


Figure: Schema of the network

Architecture: 2 hidden layers (16/8 neurons), tanh and sigmoïd activations

(*Macro*) problem resolution

The intermediate admissible state $\mathcal{X}$ can be approximated by:

$$\mathcal{X} = \left\{ X \mid X_{i+1} \in \left[X_i - \Delta_i^-, X_i + \Delta_i^+ \right], \forall i = 0, \dots, N \right\}$$

where Δ_i^- and Δ_i^+ are two scalars depending on the interval $[t_i, t_{i+1}]$.

Thanks to neural networks, ∇C_i can be computed by backward propagation.

(*Macro*) is solved by the Newton conjugate gradient from Scipy on Python.
The constraints in $X \in \mathcal{X}$ is taken into account through penalization.

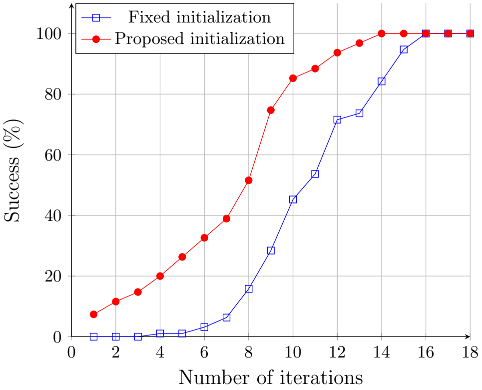
(*Micro*) problems resolution

(Micro) problems, that is $(\mathrm{OCP}_{\mathrm{i},\mathrm{X_i},\mathrm{X_{i+1}}})$, are solved by simple shooting method, with the trust region dogleg algorithm from fsolve on Matlab.

Thanks to Theorem 1, the couple

$$\left(X_i, -\nabla_a C_i(X_i, X_{i+1})\right)$$

is a natural initial guess to find a zero of the shooting function.



Results

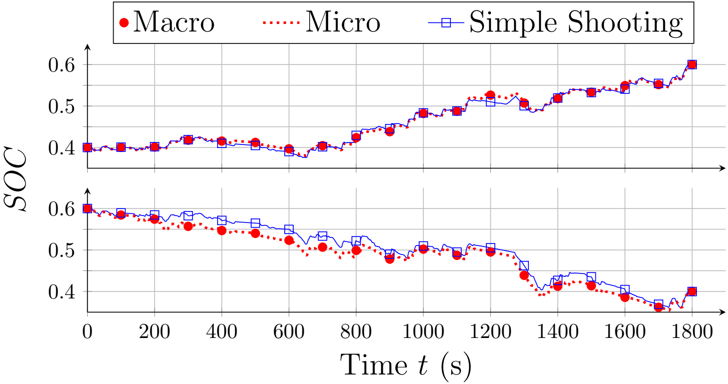


Figure: State trajectories of the simple shooting and the bi-level methods.

Associated cost error: 0.34g (0.039%) and 1.71g (0.244%).

Conclusion

Done:

- New sub-optimal method based on bi-level decomposition
- Link with other optimal control methods
- Applied to industrial complex problem

Conclusion

Done:

- New sub-optimal method based on bi-level decomposition
- Link with other optimal control methods
- Applied to industrial complex problem

Results (Proposed VS simple shooting method):

- Small cost difference
- Faster convergence with proposed initialization
- Speed up computation for online part

Conclusion

Done:

- New sub-optimal method based on bi-level decomposition
- Link with other optimal control methods
- Applied to industrial complex problem

Results (Proposed VS simple shooting method):

- Small cost difference
- Faster convergence with proposed initialization
- Speed up computation for online part

Next steps:

- Generalization: multiple cycles
- More complex model: thermal transient and steady state