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

Rémy Dutto [1,2,3]

[1]Institut de Recherche en Informatique de Toulouse

[2]Institut de Mathématiques de Toulouse

[3]Vitesco Technologies

11ième Biennale Française des Mathématiques Appliquées et Industrielles du 22 mai au 26 mai 2023 — Le Gosier, Guadeloupe

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

1. Torque split optimal control problem
   - System modelling
   - Optimal control problem formulation
2. Optimal control method
   - Classical indirect methods
   - Bi-level formulation
3. Numerical methods and results
   - Numerical methods
   - Results

Cycle

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

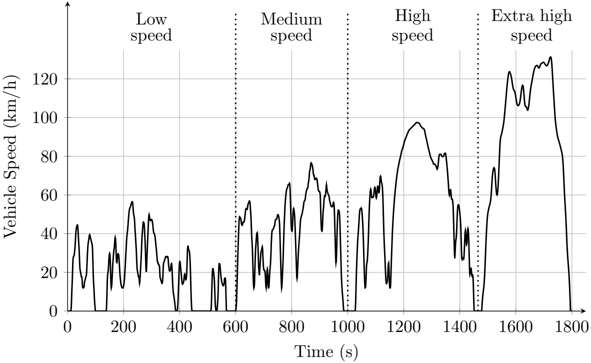


Figure: Worldwide harmonized Light vehicles Test Cycle (WLTC).

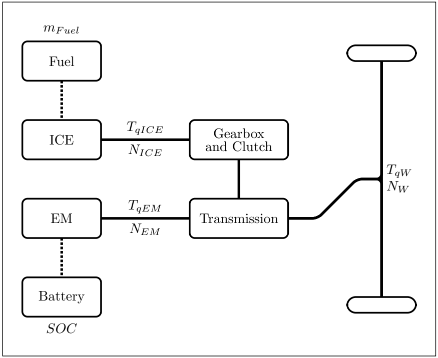
Requested wheels torque $T_{qW}(t)$ and rotation speed $N_W(t)$ are obtained with the information of our vehicle (mass, wheel diameter, aerodynamic coefficient. . .).

Static model

Inputs of our static model:

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

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

where:

- $x = SOC$ (State Of Charge)
- $u = \big(T_{qICE}, Gear\big)$
- 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 Maximum Principle

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

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

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

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

Pseudo-Hamiltonian system

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

$$u^*(t,x,p) = \arg\max \left\{ H\left(t,x,p,u\right) \;\middle|\; u \in U(t) \right\}.$$

Pseudo-Hamiltonian system

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

$$u^*(t, x, p) = \arg\max \left\{ H\left(t, x, p, u\right) \mid u \in U(t) \right\}.$$

The *pseudo-Hamiltonian vector field* is computed as follows:

$$\vec{H}(t, x, p) = \left(f\left(t, x, u^*(t, x, p)\right), -\frac{\partial H}{\partial x}\left(t, x, p, u^*(t, x, p)\right) \right)$$

Pseudo-Hamiltonian system

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

$$u^*(t, x, p) = \arg\max \left\{ H\left(t, x, p, u\right) \mid u \in U(t) \right\}.$$

The *pseudo-Hamiltonian vector field* is computed as follows:

$$\vec{H}(t, x, p) = \left(f\left(t, x, u^*(t, x, p)\right), -\frac{\partial H}{\partial x}\left(t, x, p, u^*(t, x, p)\right) \right)$$

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

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

where $z = (x, p)$.

Indirect simple shooting

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

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

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*

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

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

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

[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

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

The corresponding shooting function is therefore

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

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*:

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

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*:

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

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

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

Link with other methods

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

Condition	Problem	Methods
$N = 0$	TPBVP	Simple shooting

Link with other methods

$N+1$: number of intervals and value functions $\big/$ 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

Link with other methods

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

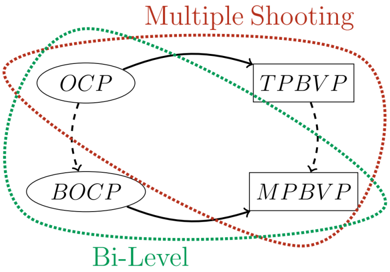
Condition	Problem	Methods
$N=0$	TPBVP	Simple shooting
$N=\dfrac{t_f-t_0}{\Delta t}$	Optimization	DP or Direct
Else	MPBVP	"Multiple shooting"

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]

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

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

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

[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 $\forall i \in \{1, ..., N\}$

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

Commutative diagram: Necessary conditions

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

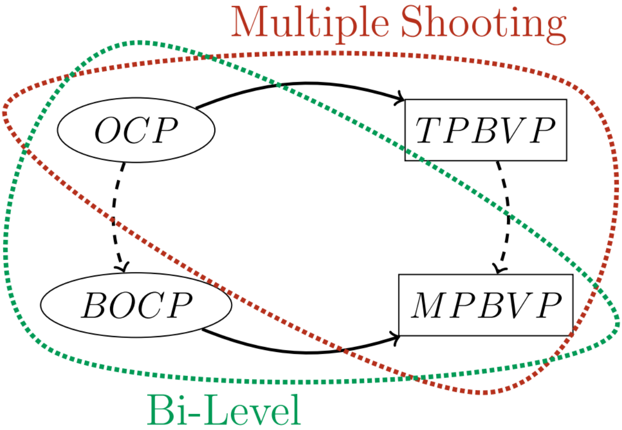
$$L(X, \lambda) = \sum_{i=0}^{N} V_i\left(X_i, X_{i+1}\right) - \lambda_0(X_0 - x_0) - \lambda_f(X_{N+1} - x_f).$$

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

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

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

Commutative diagram



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

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

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

Proposed approach

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

$(BOCP)$ becomes an optimization problem

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

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

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

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 according to

$$u^*(t, x, p) \in \arg\max \left\{ H\left(t, x, p, u\right), u \in \tilde{U}(t) \right\}$$

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 according to

$$u^*(t, x, p) \in \arg\max \left\{ H(t, x, p, u), u \in \tilde{U}(t) \right\}$$

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

The pseudo-Hamiltonian vector field is computed as follows:

$$\vec{H}(t, x, p) = \left(f\Big(t, x, u^*(t, x, p)\Big), -\frac{\partial H}{\partial x}\Big(t, x, p, u^*(t, x, p)\Big) \right)$$

where $\dfrac{\partial H}{\partial x}$ is calculated by finite differences.

Pseudo-Hamiltonian flow and approximated value functions

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 initial state and co-state space.

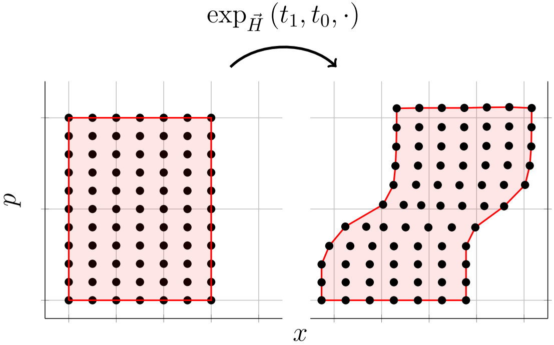


Figure: Example of Hamiltonian flow.

Pseudo-Hamiltonian flow and approximated value functions

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 initial state and co-state space.

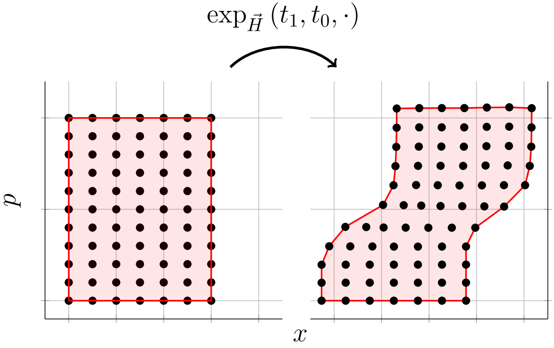


Figure: Example of Hamiltonian flow.

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

(*Macro*) problem resolution

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

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

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 ($OCP_{i,x_i,x_{i+1}}$), are solved by simple shooting method, with the trust region dogleg algorithm from fsolve on Matlab.

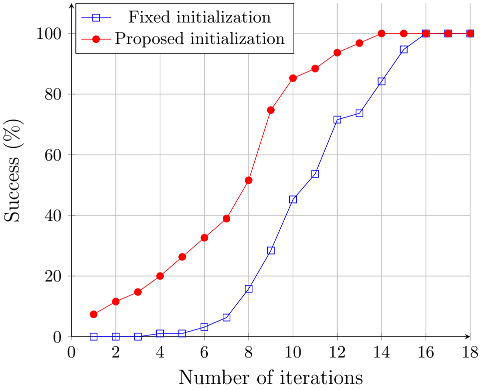
Thanks to Theorem 1,

$$\hat{z}_i = (X_i, \hat{p}_i)$$

with

$$\hat{p}_i = -\frac{\partial C_i}{\partial a}(X_i, X_{i+1})$$

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



Results

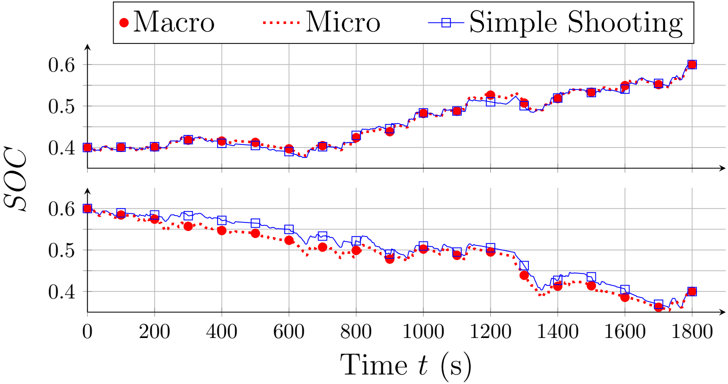


Figure: State trajectories of the simple shooting and the bi-level methods.

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
- More robust 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
- More robust with proposed initialization
- Speed up computation for online part

Next step:

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