EPFL

Laboratoire d'Automatique

Riccati Recursion and Trust Region for NMPC

Semester Project Final Presentation

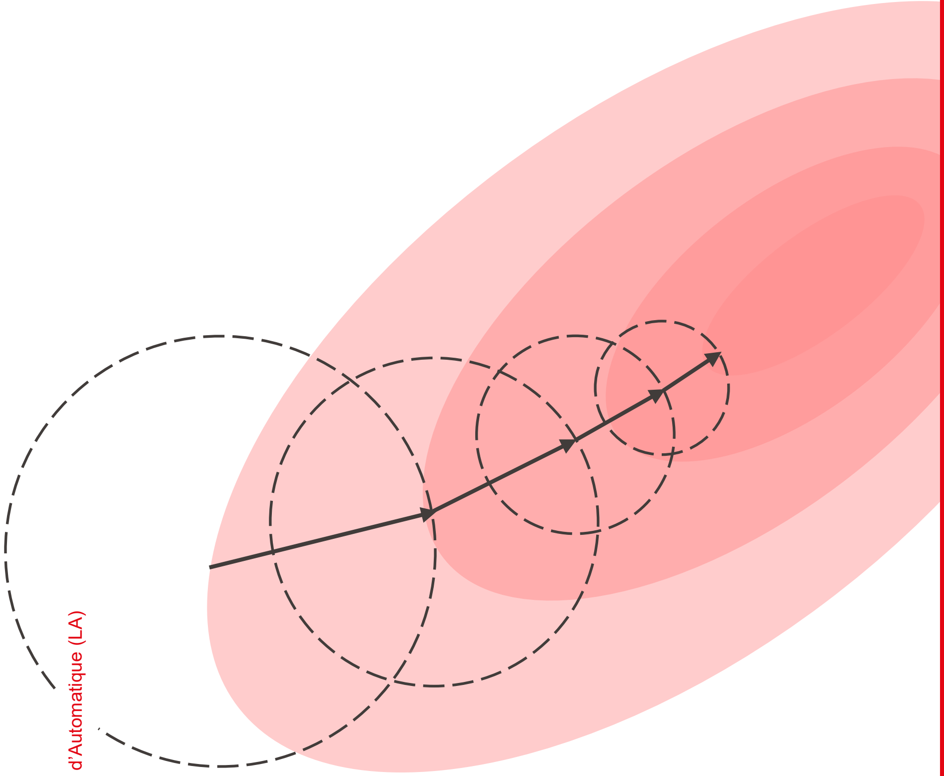
Stephen Monnet

Supervisor : Shaohui Yang

Professor : Colin Jones

18th January 2022

École polytechnique fédérale de Lausanne



OUTLINE

Theoretical Background

Formulation

Matlab Implementation

Test of the Algorithm

Conclusion

EPFL

Theoretical Background

KKT

$$\min_x \quad f(x)$$

$$\text{s.t.} \quad h(x) = 0$$
$$g(x) \leq 0$$

$$\nabla_x \mathcal{L}(x^*, \lambda^*, \nu^*) = 0 \quad (1a)$$

$$h(x^*) = 0, \quad g(x^*) \leq 0 \quad (1b)$$

$$\lambda^* \geq 0 \quad (1c)$$

$$\lambda^{*T} g(x^*) = 0 \quad (1d)$$

$$\mathcal{L}(x, \lambda, \nu) \coloneqq f(x) + \lambda^T g(x) + \nu^T h(x)$$

$i = 0$ $\longleftarrow$ **Backward** $\longleftarrow$ $i = N-1$

$$u_i = u_i(x_i)$$

$$K_i$$

$$x_{i+1} = Ax_i + Bu_i$$

$i = 0$ $\longrightarrow$ **Forward** $\longrightarrow$ $i = N-1$

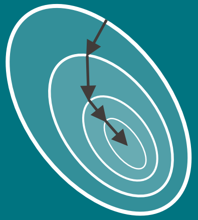
Riccati Recursion

SQP

$$k \leftarrow k + 1$$

$$\nabla^2_{xx}\mathcal{L}^k \quad \nabla_x \mathcal{L}^k \quad \nabla^2_{x\nu}\mathcal{L}^k \quad \nabla_\nu h^k \quad \nabla_x h^k$$

$$x^{k+1}, \nu^{k+1}$$



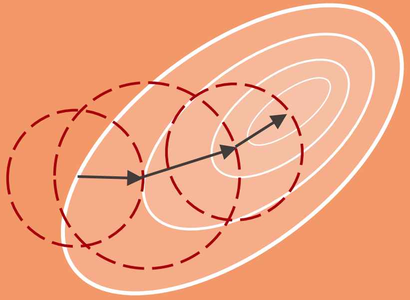
$$\Delta x^k, \Delta \nu^k$$

$$\|\nabla \mathcal{L}^k\| > \epsilon$$

$$\begin{aligned} \nabla_x \mathcal{L}^k + \nabla^2_{xx}\mathcal{L}^k \Delta x^k + \nabla^2_{x\nu}\mathcal{L}^k \Delta \nu^k &= 0 \\ h^k + \nabla_x h^k \Delta x^k + \nabla_\nu h^k \Delta \nu^k &= 0 \end{aligned}$$

$$\rho^k = \frac{\mathcal{L}^k - \mathcal{L}^{k+1}}{m^k - m^{k+1}}$$

$$\Delta^{k+1}$$



Trust Region

Stephen Monnet

Laboratoire d'Automatique (LA)

Trust Region in Riccati Recursion

$$\min \sum_{i=0}^{N-1} \frac{1}{2} \begin{bmatrix} \Delta u_i \\ \Delta x_i \end{bmatrix}^T \begin{bmatrix} R_i & S_i^T \\ S_i & Q_i \end{bmatrix} \begin{bmatrix} \Delta u_i \\ \Delta x_i \end{bmatrix} + \begin{bmatrix} \Delta u_i \\ \Delta x_i \end{bmatrix}^T \begin{bmatrix} r_i \\ q_i \end{bmatrix} + \frac{1}{2} \Delta x_N^T Q_N \Delta x_N + \Delta x_N^T q_N$$

$$s.t.\ x_0 - \bar{x} + \Delta x_0 = 0$$

$$\bar{x}_i + A_i \Delta x_i + B_i \Delta u_i - \Delta x_{i+1} = 0 \qquad i = 0, \dots, N-1$$

$$\min_{\Delta u} \quad \frac{1}{2} \boldsymbol{\Delta u}^T \boldsymbol{H} \boldsymbol{\Delta u} + \boldsymbol{g}^T \boldsymbol{\Delta u} + \boldsymbol{c}$$

$$\text{s.t.} \quad \|\boldsymbol{\Delta u}\|_2 \leq \Delta$$

$$\boxed{\lambda \leftarrow \lambda + \frac{\|\boldsymbol{\Delta u}\|_2^2}{\|q\|_2} \cdot \frac{\|\boldsymbol{\Delta u}\|_2 - \Delta}{\Delta}}$$

KKT

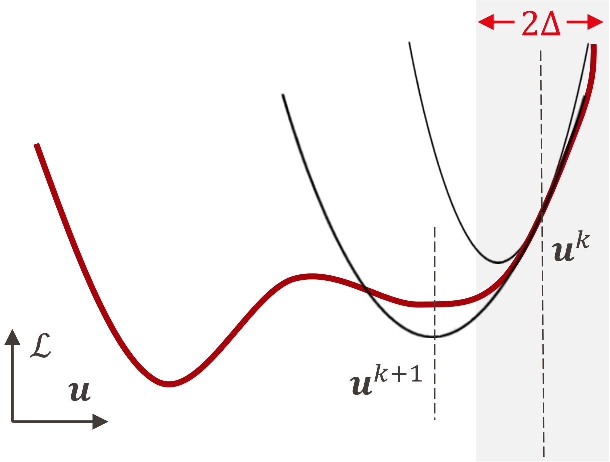
$$(H + \lambda^* I)\boldsymbol{\Delta u}^* = -g$$

$$\lambda^* (\Delta - \|\boldsymbol{\Delta u}^*\|_2) = 0$$

$$(H + \lambda^* I) \succcurlyeq 0$$

$$\lambda^* \geq 0$$

$$\frac{1}{\|\boldsymbol{\Delta u}^*\|_2} - \frac{1}{\Delta} = 0$$



Backward

start with $P_N = Q_N, p_N = q_N$

for $i = N-1, N-2, \dots, 1, 0$ do :

$$\bar{R}_i = R_i + \lambda I + B_i^T P_{i+1} B_i$$

Increase eigenvalues s.t. $\bar{R}_i \succ 0$

$$\Lambda_i = \text{chol}(\bar{R}_i) \text{ s.t. } \Lambda_i^T \Lambda_i = \bar{R}_i$$

Reduce computational cost

$$L_i = \Lambda_i^{-T}(S_i + B_i^T P_{i+1} A_i)$$

$$P_i = Q_i + A_i^T P_{i+1} A_i - L_i^T L_i$$

$$l_i = \Lambda_i^{-T}(r_i + B_i^T(P_{i+1} b_i + p_{i+1}))$$

$$p_i = q_i + A_i^T(P_{i+1} b_i + p_{i+1}) - L_i^T l_i$$

end loop

Forward

start with x_0

for $i = 0, 1, \dots, N-1$ do

$$\Delta u_i = -\Lambda_i^{-1}(L_i \Delta x_i + l_i)$$

$$\Delta x_{i+1} = A_i \Delta x_i + B_i \Delta u_i + b_i)$$

end loop

Stephen Monnet

Formulation : Lagrangian & KKT

Non-linear Optimal Control Problem

$$\begin{aligned}\min_{\mathbf{X},\mathbf{U}} \quad & \sum_{i=0}^{N-1} l(x_i, u_i) + V_f(x_N) \\ \text{s.t.} \quad & x_0 - \bar{x} = 0, \\ & x_i + f(x_i, u_i)\delta t - x_{i+1} = 0, \qquad i = 0, ..., N-1\end{aligned}$$

Lagrangian

$$\mathcal{L}(\mathbf{X}, \mathbf{U}, \mathbf{\Lambda}) = \sum_{i=0}^{N-1} l(x_i, u_i) + V_f(x_N) - \lambda_0^T(x_0 - \bar{x}) + \sum_{i=0}^{N-1} \lambda_{i+1}^T(x_i + f(x_i, u_i)\delta t - x_{i+1})$$

KKT

Hamiltonian : $\mathcal{H}(x_i, u_i, \lambda_{i+1}) := l(x_i, u_i) + \lambda_{i+1}^T f(x_i, u_i)\delta t$

$$\begin{aligned} r_{x,N} &:= \nabla_x V_f(x_N) - \lambda_N = 0 \\ r_{x,i} &:= \nabla_x \mathcal{H}(x_i, u_i, \lambda_{i+1}) + \lambda_{i+1} - \lambda_i = 0 \qquad i = 0, ..., N-1 \\ r_{u,i} &:= \nabla_u \mathcal{H}(x_i, u_i, \lambda_{i+1}) = 0 \qquad i = 0, ..., N-1 \end{aligned}$$

Formulation : Newton Step

Primal Feasibility

$$\underbrace{x_0^{k+1} - \bar{x}}_{\text{Approx at k+1}} \approx \underbrace{x_0^k - \bar{x}}_{\text{Value at k}} + \underbrace{(x_0^{k+1} - x_0^k)}_{\text{Newton step}} = x_0^k - \bar{x} + \Delta x_0 = 0$$

$$\bar{x}_i + A_i \Delta x_i + B_i \Delta u_i - \Delta x_{i+1} = 0, \qquad i = 0, ..., N-1$$

$$\bar{x}_i = x_i + f(x_i, u_i)\delta t - x_{i+1}$$
$$A_i = \mathbf{I} + \nabla_x f(x_i, u_i)\delta t$$
$$B_i = \nabla_u f(x_i, u_i)\delta t$$

Stationarity

$$r_{x,N}^{k+1} \approx r_{x,N} + Q_{xx,N} \Delta x_N - \Delta\lambda_N = 0$$

$$r_{x,i}^{k+1} \approx r_{x,i} + Q_{xx,i}\Delta x_i + Q_{xu,i}\Delta u_i + A_i \Delta\lambda_{i+1} - \Delta\lambda_i = 0$$

$$r_{u,i}^{k+1} \approx r_{u,i} + Q_{xu,i}\Delta x_i + Q_{uu,i}\Delta u_i + B_i^T \Delta\lambda_{i+1} = 0$$

Formulation : Equivalent QP

Consider $\{\Delta\lambda_i\}_{i=0}^{N}$ as Lagrange Multipliers

$$\min_{\mathbf{X},\mathbf{U}} \quad \underbrace{\sum_{i=0}^{N-1} \frac{1}{2}\left\{\begin{bmatrix}\Delta x_i\\ \Delta u_i\end{bmatrix}^T \begin{bmatrix}Q_{xx,i} & Q_{xu,i}\\ Q_{ux,i} & Q_{uu,i}\end{bmatrix}\begin{bmatrix}\Delta x_i\\ \Delta u_i\end{bmatrix} + \begin{bmatrix}r_{x,i}\\ r_{u,i}\end{bmatrix}^T\begin{bmatrix}\Delta x_i\\ \Delta u_i\end{bmatrix}\right\} + \Delta x_N^T Q_{xx,N}\Delta x_N + r_{x,N}^T\Delta x_N}_{2^{nd}\text{ order approximation of the Lagrangian}}$$

$$\text{s.t.} \quad x_0 - \bar{x} + \Delta x_0 = 0$$

$$\bar{x}_i + A_i\Delta x_i + B_i\Delta u_i - \Delta x_{i+1} = 0, \qquad i = 0, \ldots, N-1$$

Solving this QP $\longleftrightarrow$ Solving KKT approximation

Formulation : Trust Region

Reformulate QP in dense form

$$\begin{bmatrix}\Delta u_0\\ \Delta u_1\\ \vdots\\ \Delta u_{N-1}\end{bmatrix}^T \begin{bmatrix}Q_{uu,0} & \mathbf{0} & \dots & \mathbf{0}\\ \mathbf{0} & Q_{uu,1} & \dots & \mathbf{0}\\ \vdots & & \ddots & \vdots\\ \mathbf{0} & \dots & \dots & Q_{uu,N-1}\end{bmatrix}\begin{bmatrix}\Delta u_0\\ \Delta u_1\\ \vdots\\ \Delta u_{N-1}\end{bmatrix}$$

$$\mathbf{\Delta x}^T Q_{xx}\mathbf{\Delta x} + \mathbf{\Delta u}^T Q_{ux}\mathbf{\Delta x_{N-1}} + \mathbf{\Delta x_{N-1}}^T Q_{xu}\mathbf{\Delta u} + \mathbf{\Delta u}^T Q_{uu}\mathbf{\Delta u} + R_x^T\mathbf{\Delta x} + R_u^T\mathbf{\Delta u}$$

Use equality constraints to express :

$$\boldsymbol{\Delta x} = \boldsymbol{f}(\boldsymbol{\Delta u})$$

$$x_0 - \bar{x} + \Delta x_0 = 0$$
$$\bar{x}_i + A_i\Delta x_i + B_i\Delta u_i - \Delta x_{i+1} = 0$$

$$\longrightarrow \begin{bmatrix}\Delta x_0\\ \Delta x_1\\ \Delta x_2\\ \vdots\\ \Delta x_{N-1}\\ \Delta x_N\end{bmatrix} = \mathbf{M}\begin{bmatrix}\Delta u_0\\ \Delta u_1\\ \Delta u_2\\ \vdots\\ \Delta u_{N-1}\end{bmatrix} + \mathbf{h}$$

$$\begin{aligned}\min \quad & \mathbf{\Delta u}^T\mathbf{H}\mathbf{\Delta u} + \mathbf{g}^T\mathbf{\Delta u} + \mathbf{c}\\ \text{s.t.} \quad & \|\mathbf{\Delta u}\|_2 \le \Delta\end{aligned}$$

Trust region radius

Implementation : NOCP Class

MATLAB®

Stephen Monnet

```
function obj = NOCP(primal, dual, dynamic, cost, params)
```

```
function [] = build_SQP(obj)
```

$Q_{xx,i}$ $Q_{xu,i}$ $Q_{uu,i}$ $\boldsymbol{H}$ …

```
function [x, u, lambda] = updateSol(obj)
```

$x^{k+1} \leftarrow x^k + \Delta x^k \quad u^{k+1} \leftarrow u^k + \Delta u^k \quad \lambda^{k+1} \leftarrow \lambda^k + \Delta\lambda^k$

```
function [L] = evalLagrangian(obj)
```

$\mathcal{L}(x^k, u^k, \lambda^k) = \cdots$

```
function [DELTA] = updateRadius(obj)
```

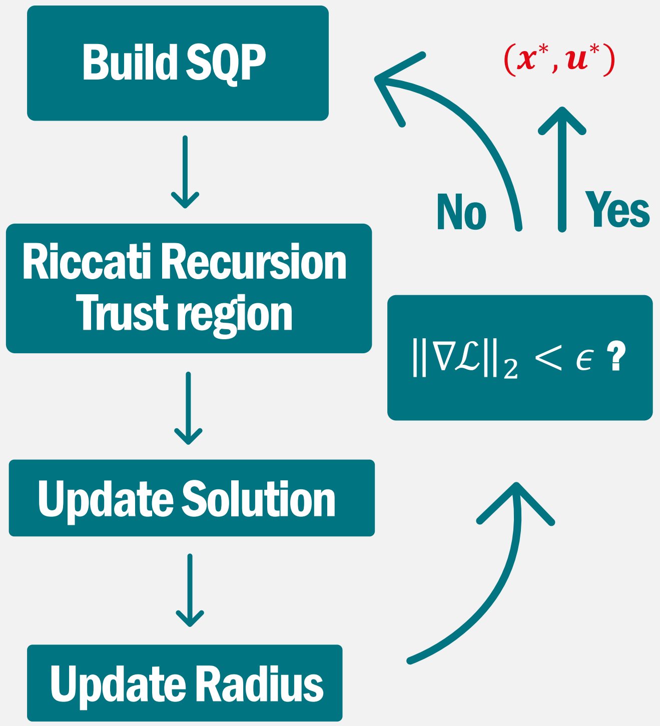
$\rho^k = \cdots \quad \rightarrow \quad \Delta^{k+1} = \cdots$

```
function [dlambda] = update_dlambda(obj)
```

$i = N-1, N-2, \dots, 0$

$$\Delta\lambda_i^k = A_i \Delta\lambda_{i+1}^k + Q_{xu,i}^k \Delta u_i^k + Q_{xx,i}^k \Delta x_i^k + r_{x,i}^k$$

```
function [x, u] = solve(obj)
```



Implementation : riccati_TR Class

```
function obj = riccati_TR(N, LAMBDA, DELTA, A, B, Q, R, S, q, r, b)
```

```
function [] = backward(obj)
```

$$
\begin{aligned}
&i = N-1, N-2, \dots, 1, 0 \\
&\quad \bar{R}_i = R_i + \lambda I + B_i^T P_{i+1} B_i \\
&\quad \Lambda_i = chol(\bar{R}_i) \\
&\quad L_i = \Lambda_i^{-T}(S_i + B_i^T P_{i+1} A_i) \\
&\quad \quad \vdots
\end{aligned}
$$

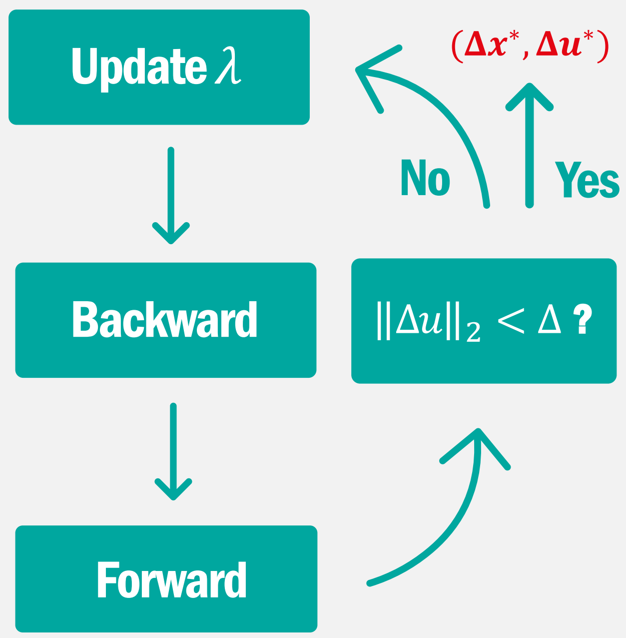
```
function [dx, du] = forward(obj)
```

$$
\begin{aligned}
&i = 0, 1, \dots, N-1 \\
&\quad \Delta u_i = -\Lambda_i^{-1}(L_i \Delta x_i + l_i) \\
&\quad \Delta x_{i+1} = A_i \Delta x_i + B_i \Delta u_i + b_i
\end{aligned}
$$

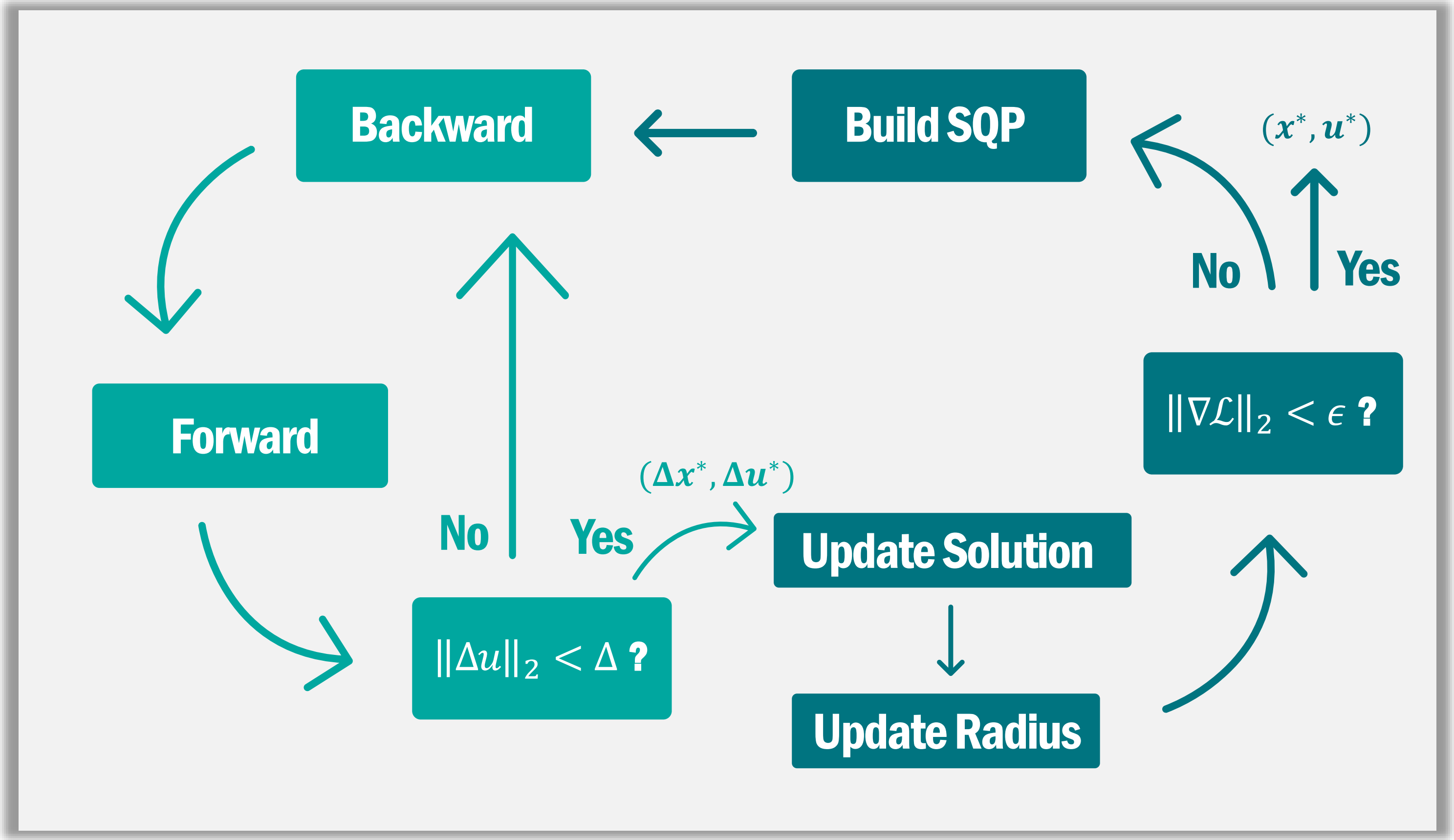
```
function [LAMBDA] = updateLambda(obj)
```

$$
\lambda \leftarrow \lambda + \frac{\|\Delta u\|_2^2}{\|q\|_2^2} \cdot \frac{\|\Delta u\|_2 - \Delta}{\Delta}
$$

```
function [dx, du] = solve(obj)
```



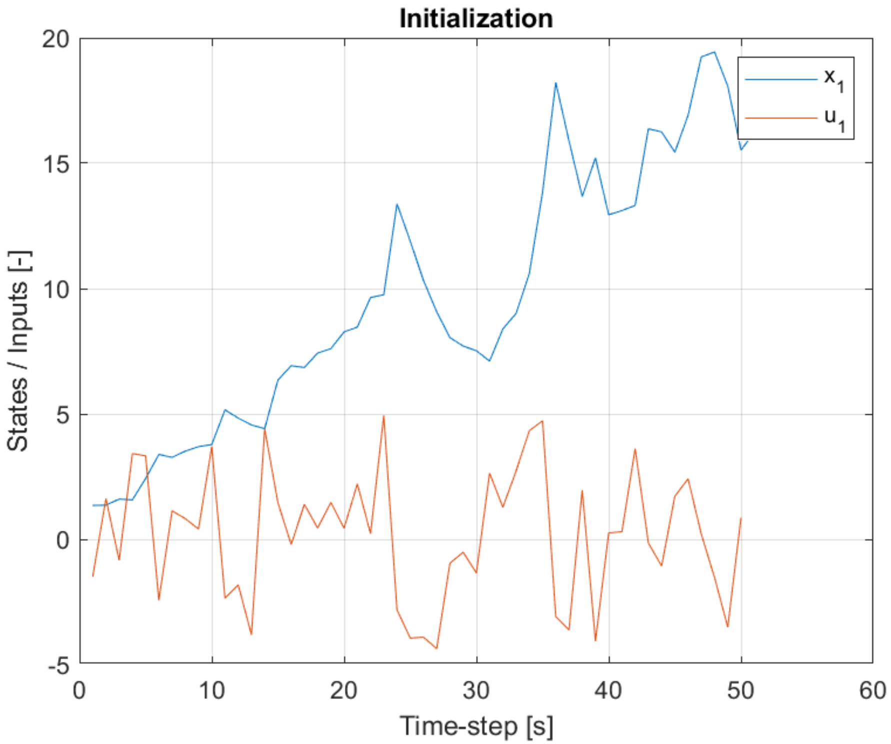
Implementation : Full Algorithm



Stephen Monnet

Results : 1 state & 1 input

$$\dot{x} = x \cdot u + u^2$$

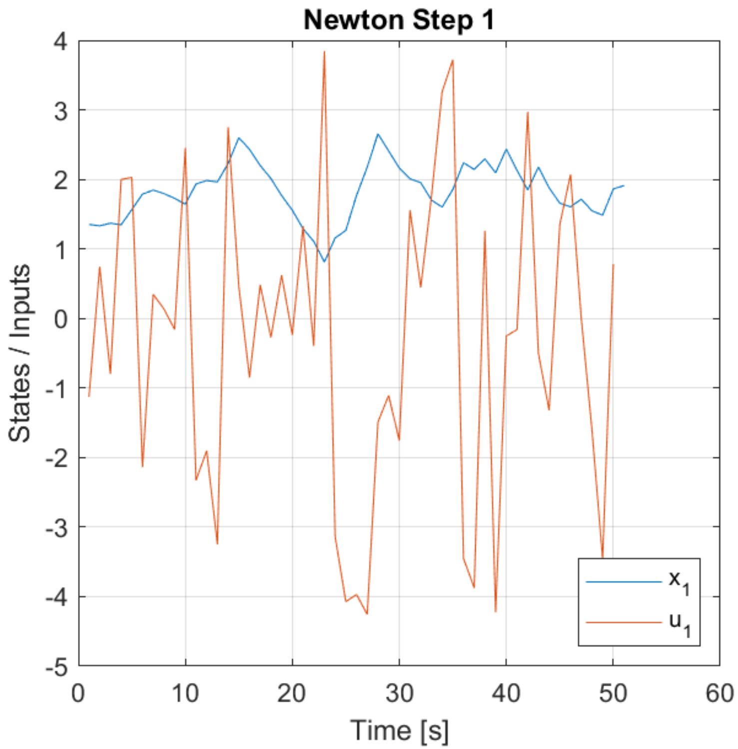


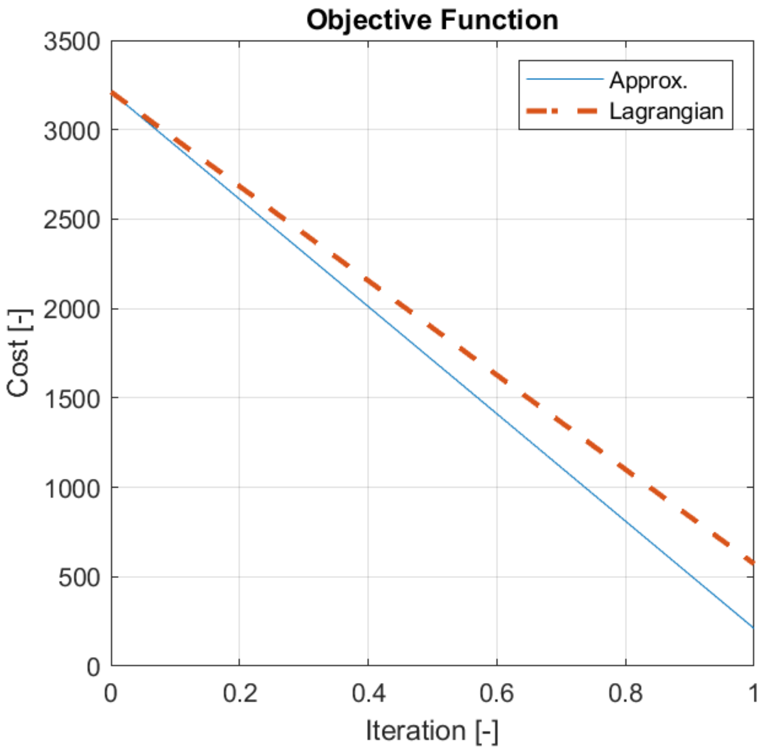
Random initialization of $u_i \in [-5; 5]$

Initialization of the x_i with forward propagation

Results : 1 state & 1 input

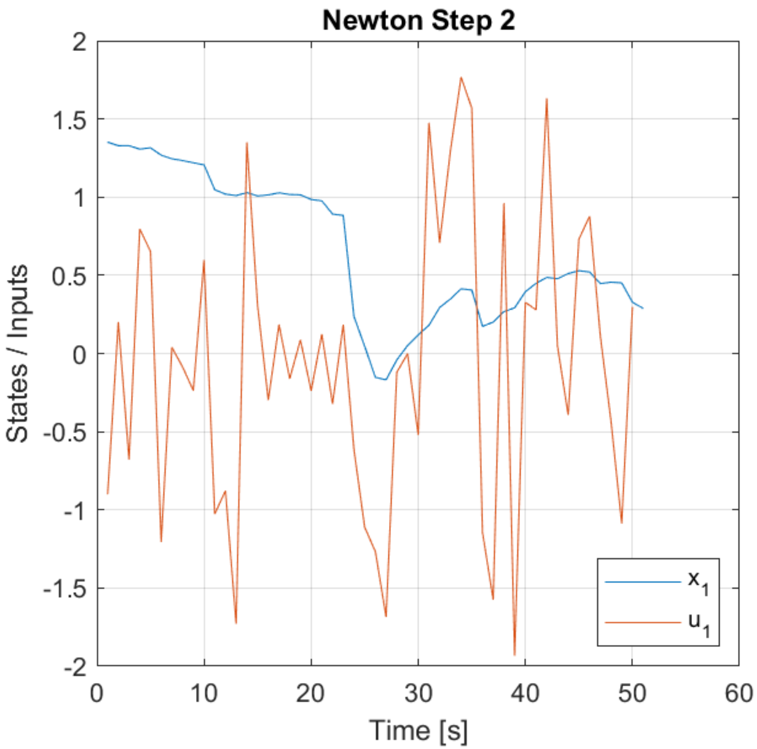
$$\dot{x} = x \cdot u + u^2$$

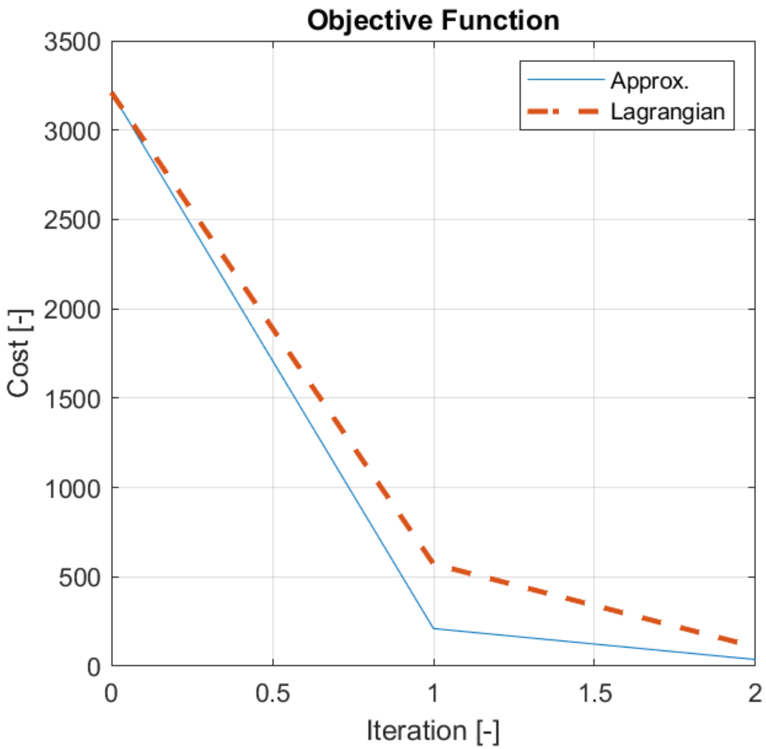




Results : 1 state & 1 input

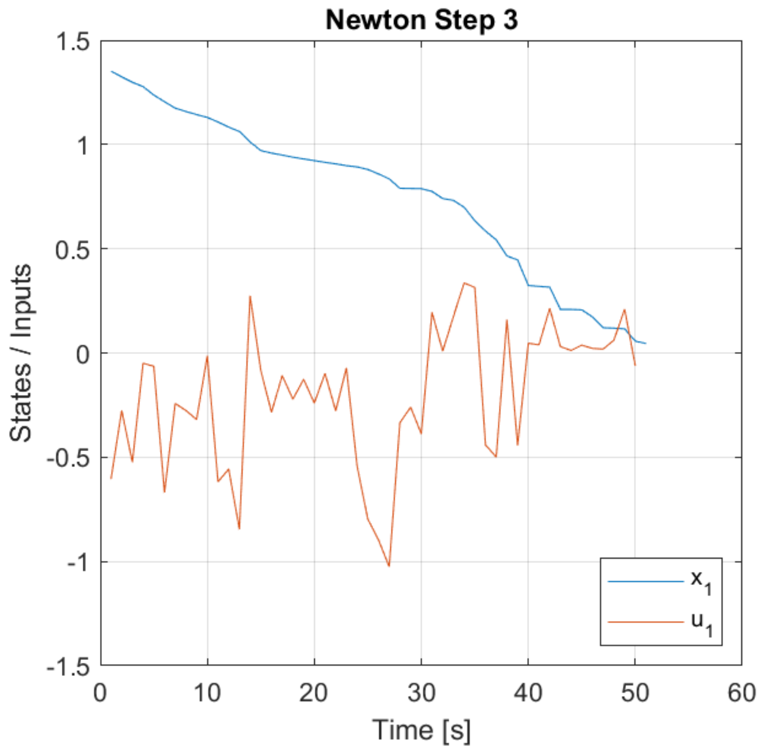
$$\dot{x} = x \cdot u + u^2$$

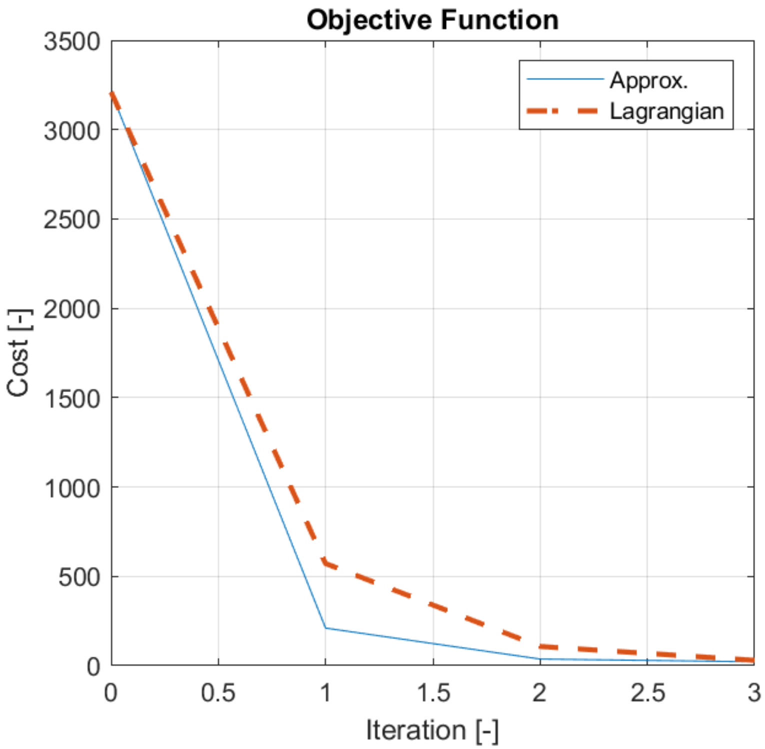




Results : 1 state & 1 input

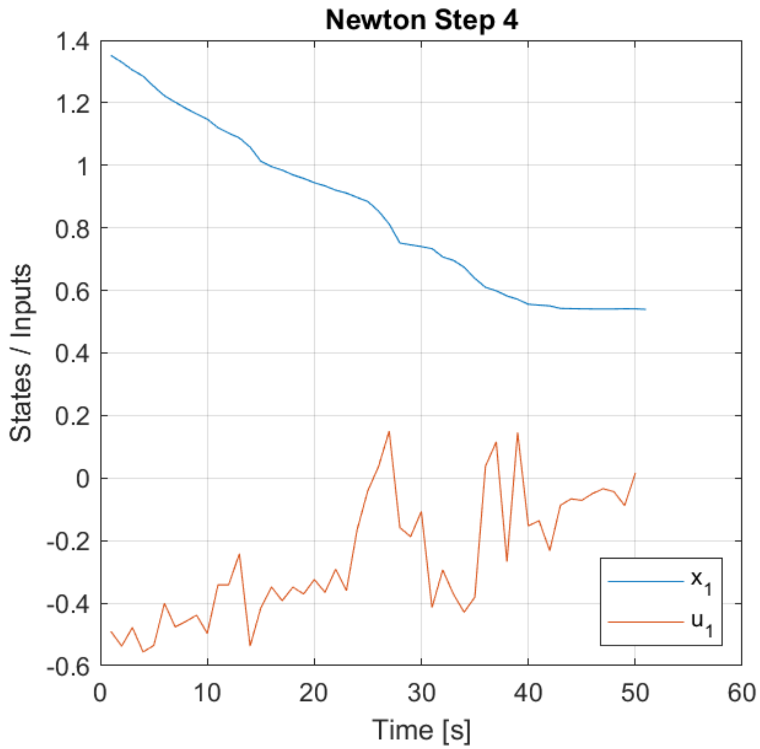
$$\dot{x} = x \cdot u + u^2$$

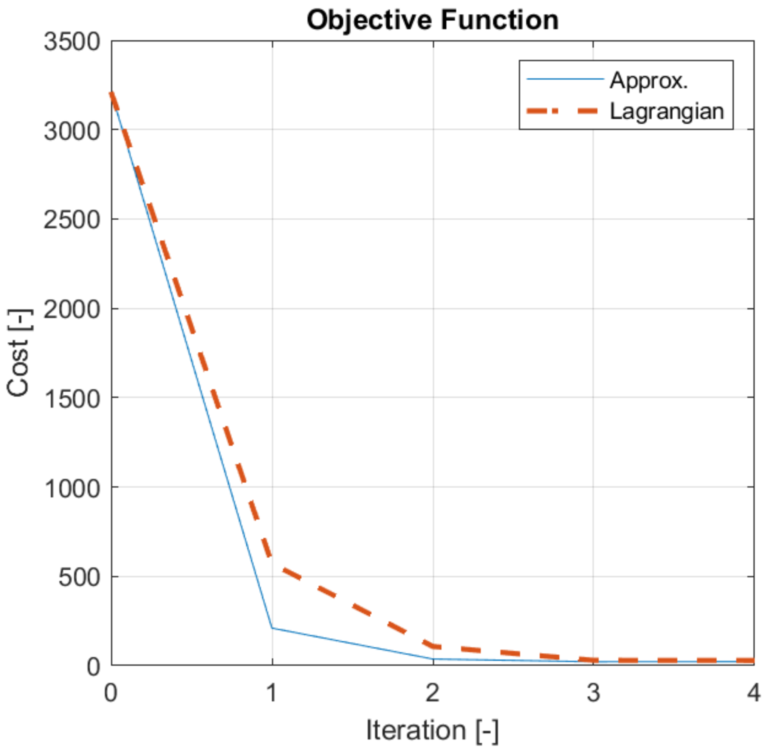




Results : 1 state & 1 input

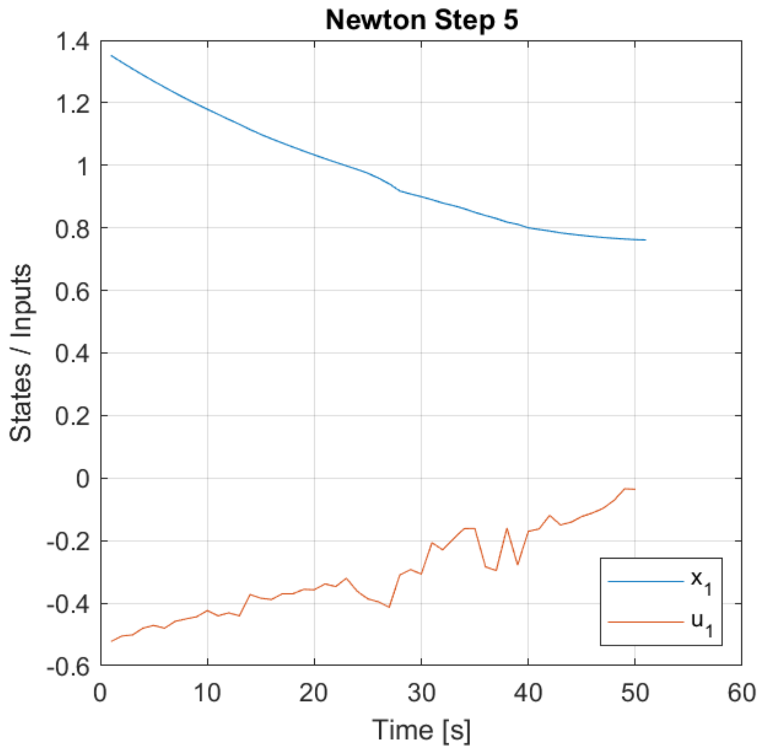
$$\dot{x} = x \cdot u + u^2$$

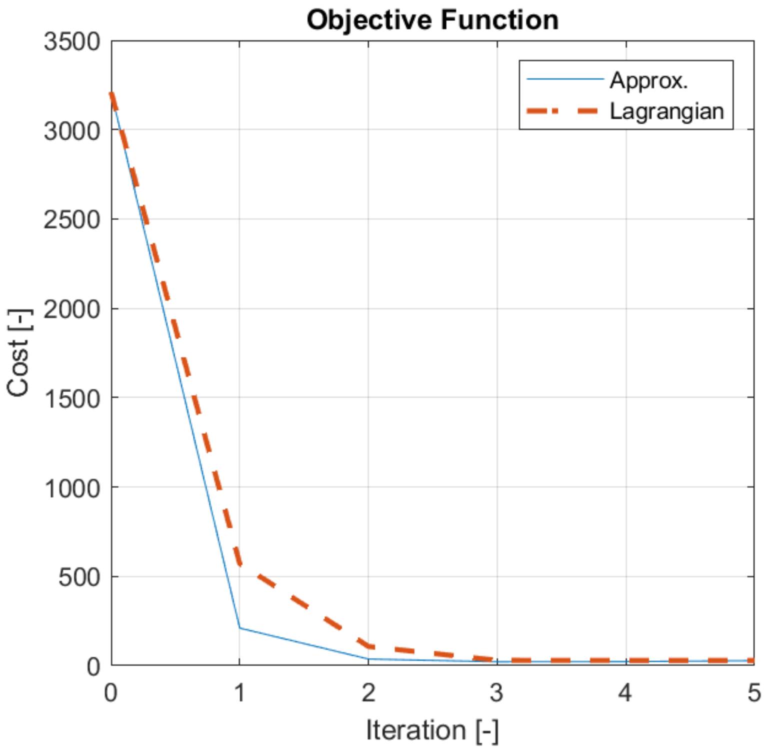




Results : 1 state & 1 input

$$\dot{x} = x \cdot u + u^2$$



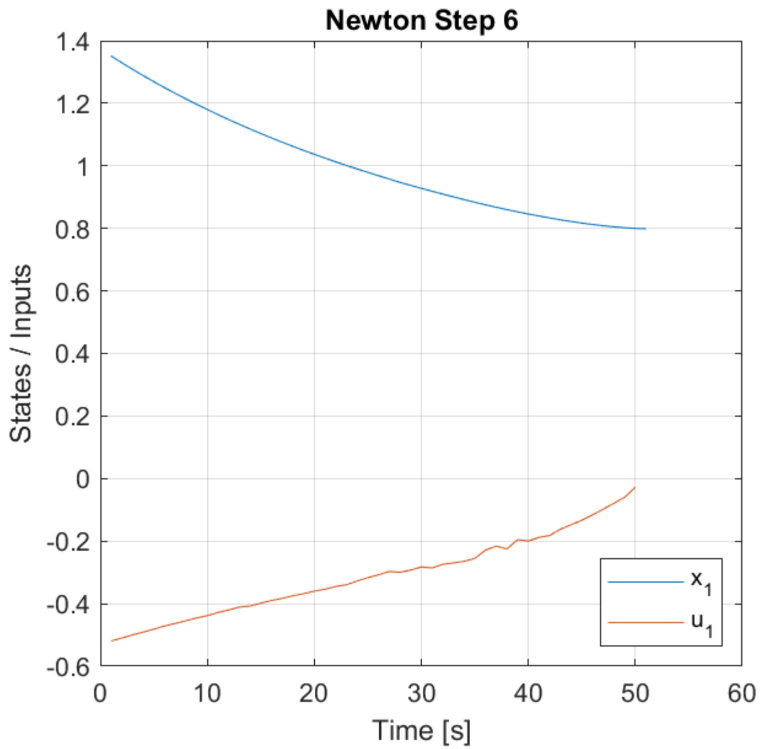


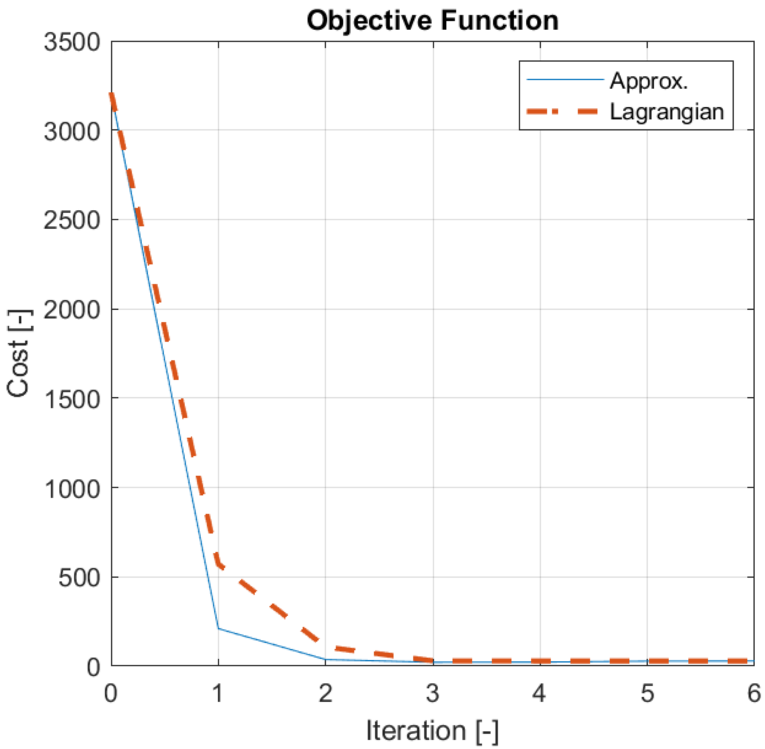
EPFL

Results : 1 state & 1 input

$$\dot{x} = x \cdot u + u^2$$

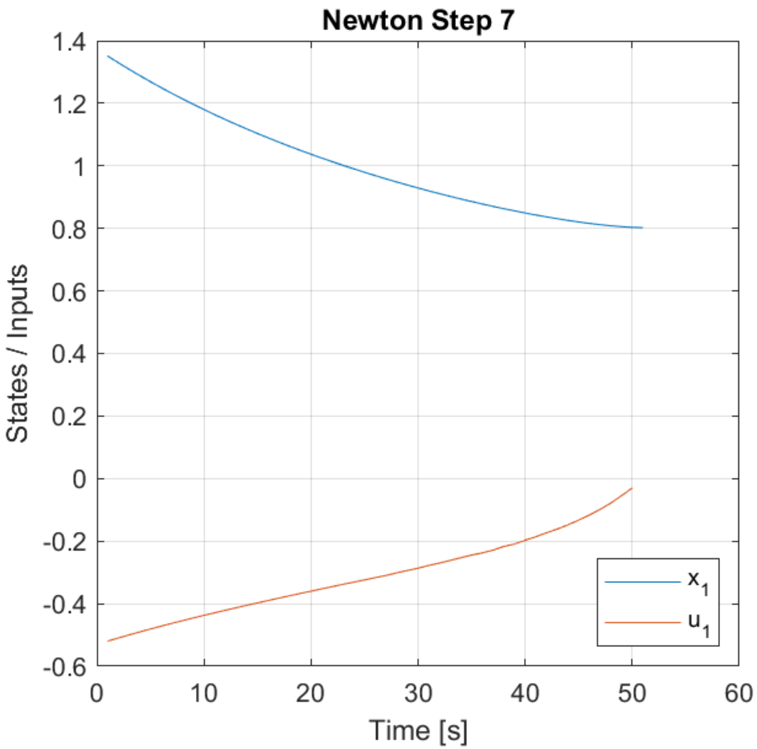
Stephen Monnet

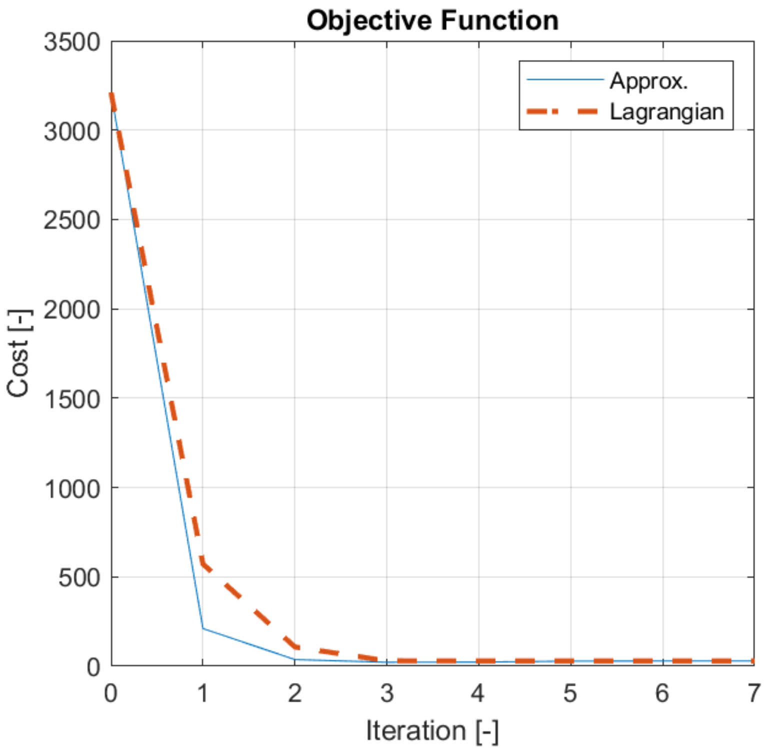




Results : 1 state & 1 input

$$\dot{x} = x \cdot u + u^2$$

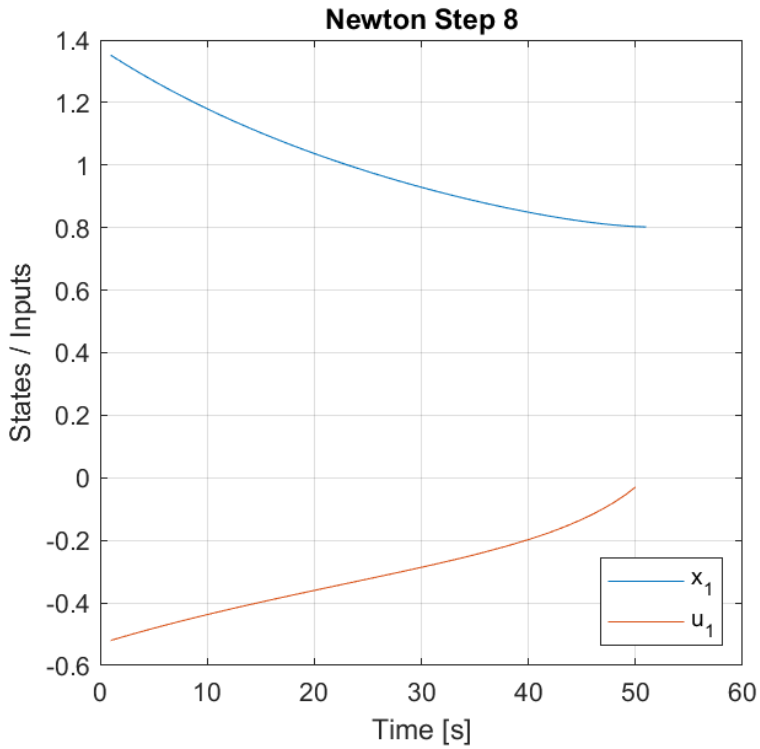


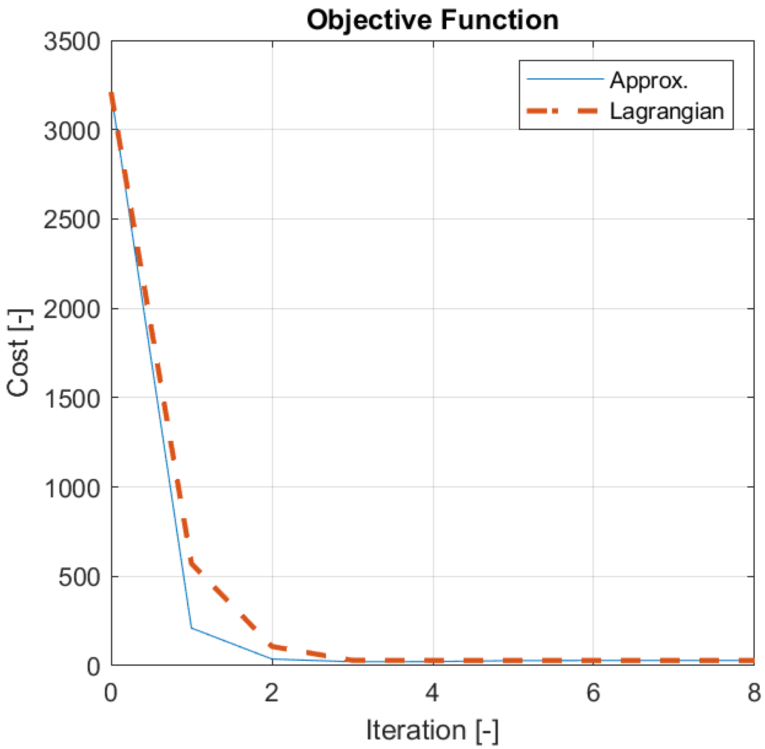


Results : 1 state & 1 input

$$\dot{x} = x \cdot u + u^2$$

Stephen Monnet

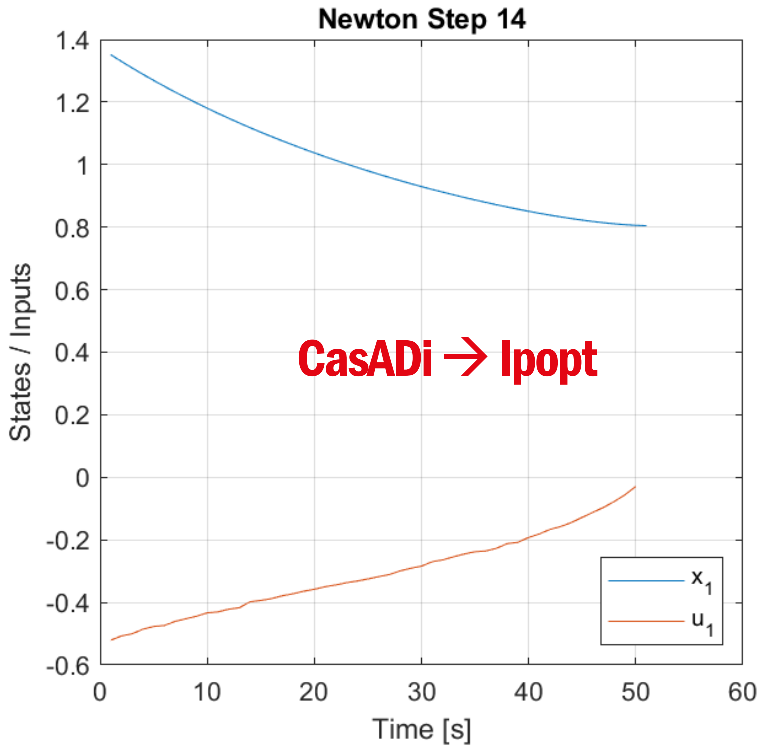


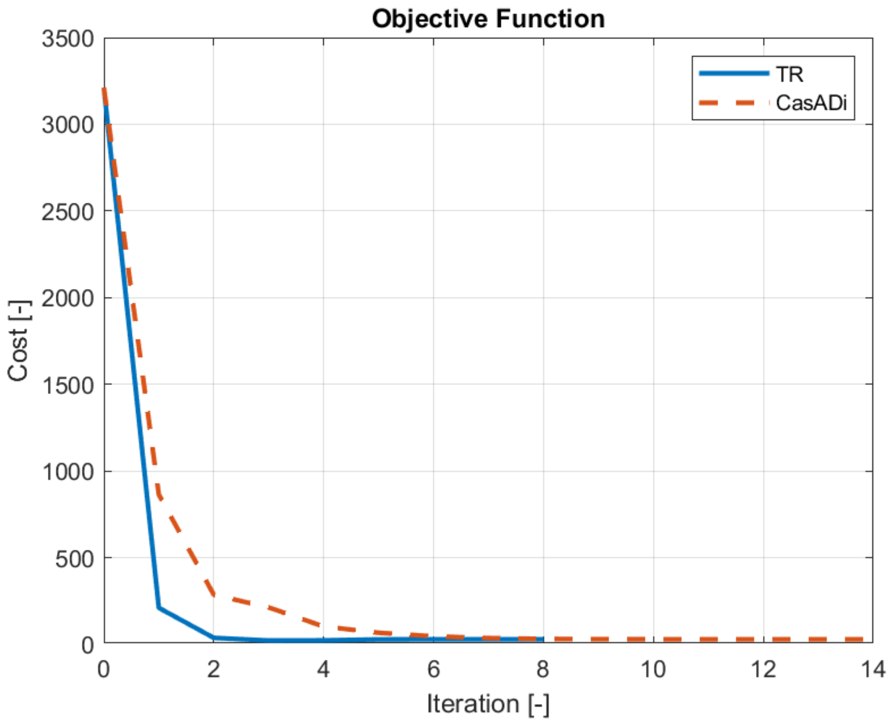


Laboratoire d'Automatique (LA)

Results : 1 state & 1 input

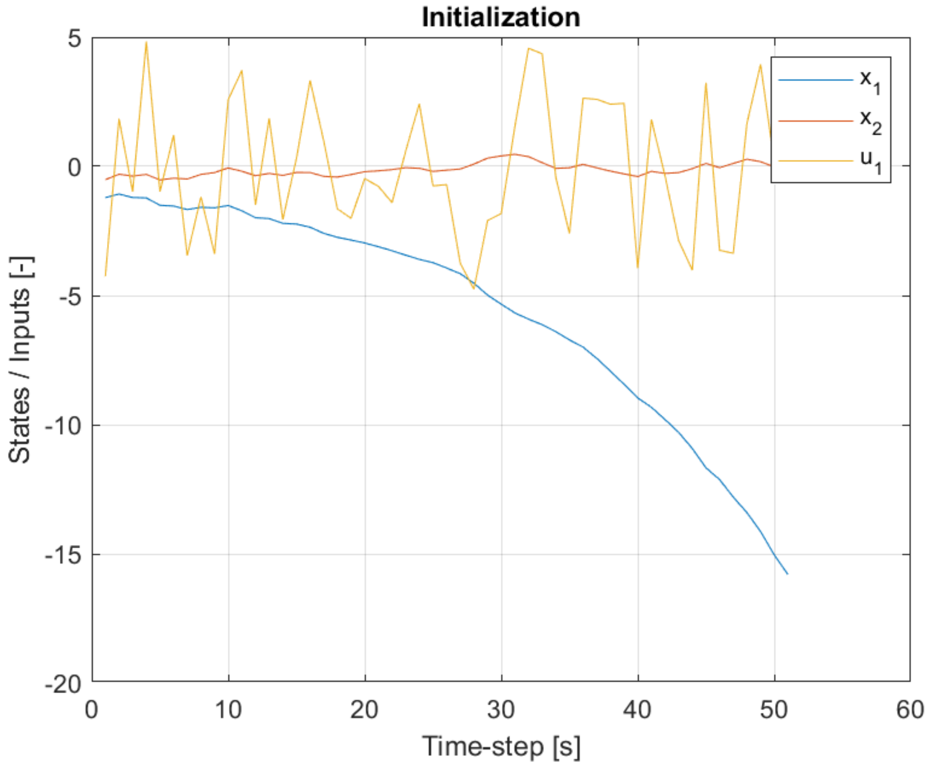
$$\dot{x} = x \cdot u + u^2$$





Results : 2 states & 1 input

$$\dot{x} = \begin{bmatrix} x_1 + u \cdot \sin(x_1) \\ -x_2 - u \cdot \cos(x_2) \end{bmatrix}$$



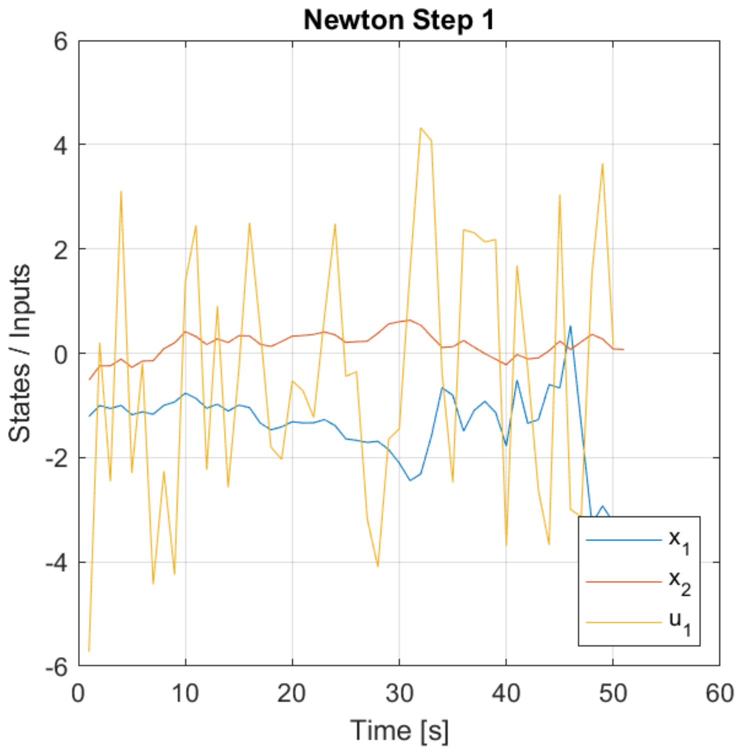
Random initialization of $u_i \in [-5; 5]$

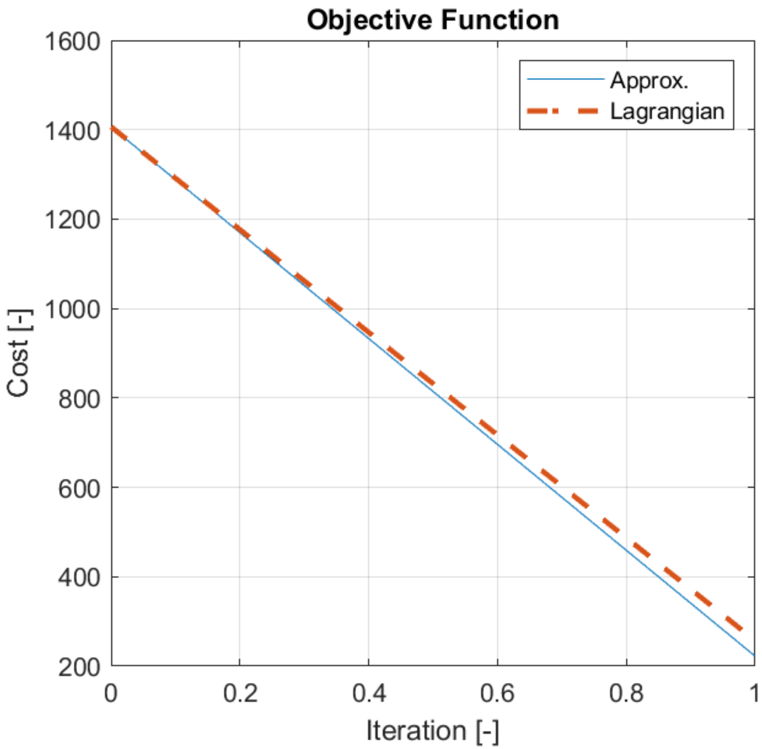
Initialization of the x_i with forward propagation

Results : 2 states & 1 input

$$\dot{x} = \begin{bmatrix} x_1 + u \cdot \sin(x_1) \\ -x_2 - u \cdot \cos(x_2) \end{bmatrix}$$

Stephen Monnet



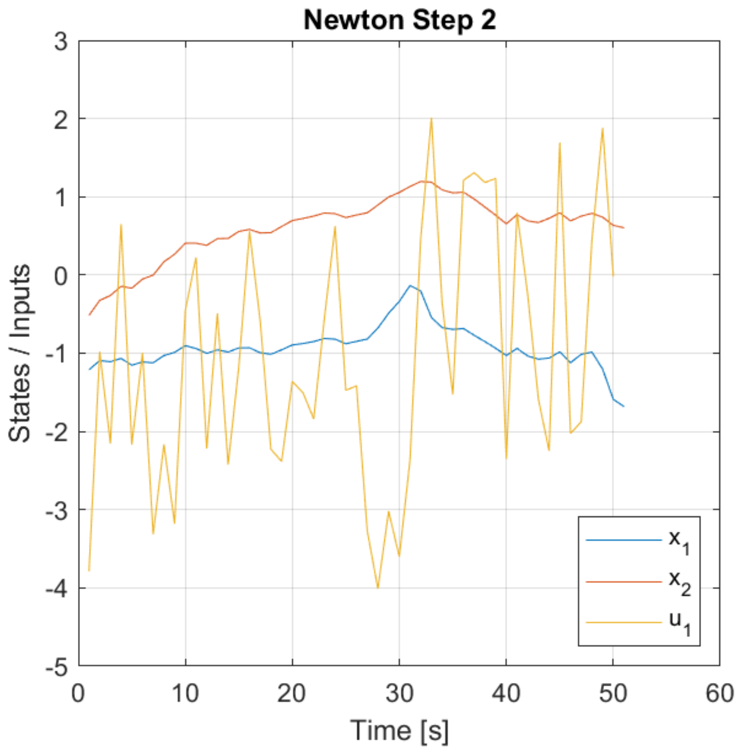


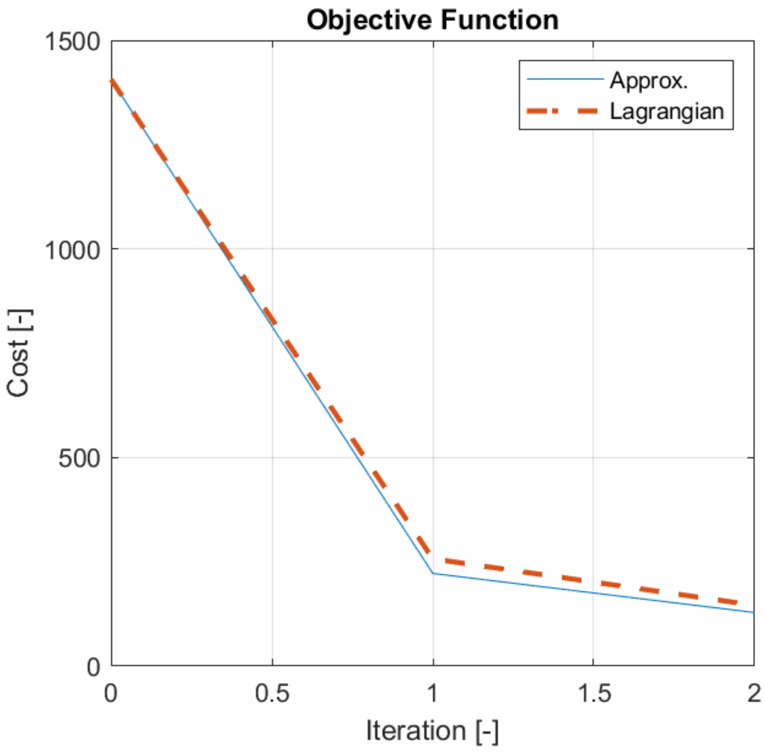
EPFL

Results : 2 states & 1 input

$$\dot{x} = \begin{bmatrix} x_1 + u \cdot \sin(x_1) \\ -x_2 - u \cdot \cos(x_2) \end{bmatrix}$$

Stephen Monnet

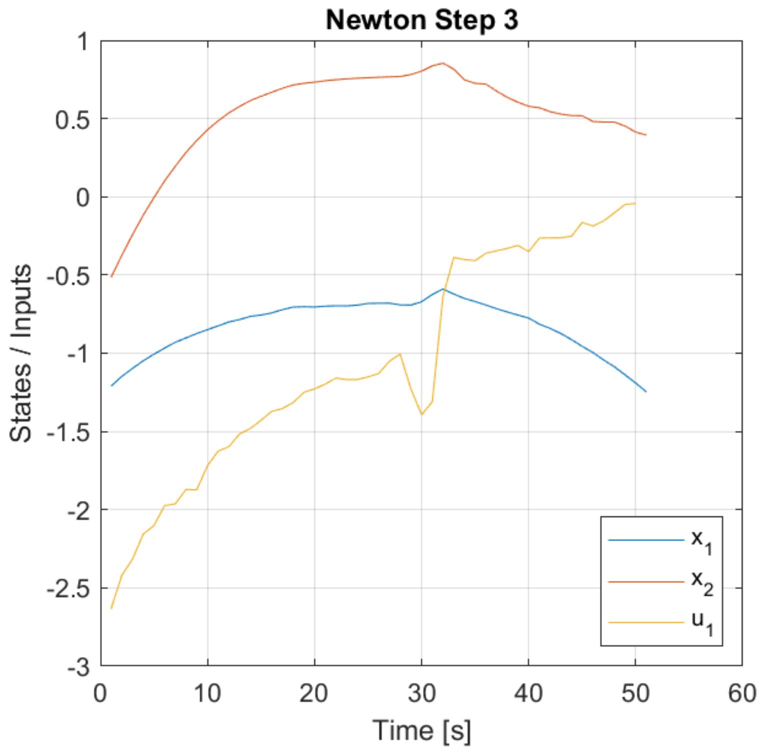


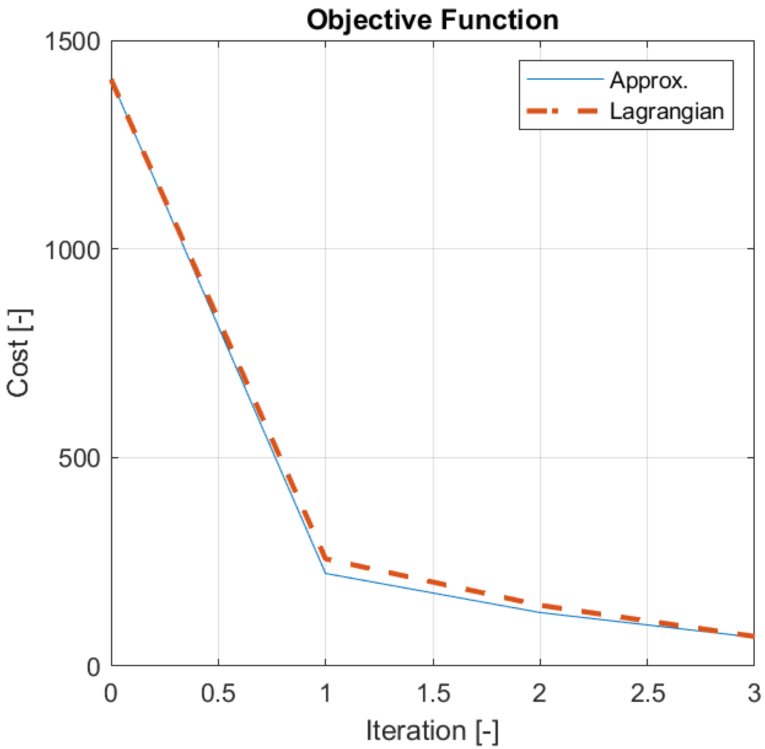


Results : 2 states & 1 input

$$\dot{x} = \begin{bmatrix} x_1 + u \cdot \sin(x_1) \\ -x_2 - u \cdot \cos(x_2) \end{bmatrix}$$

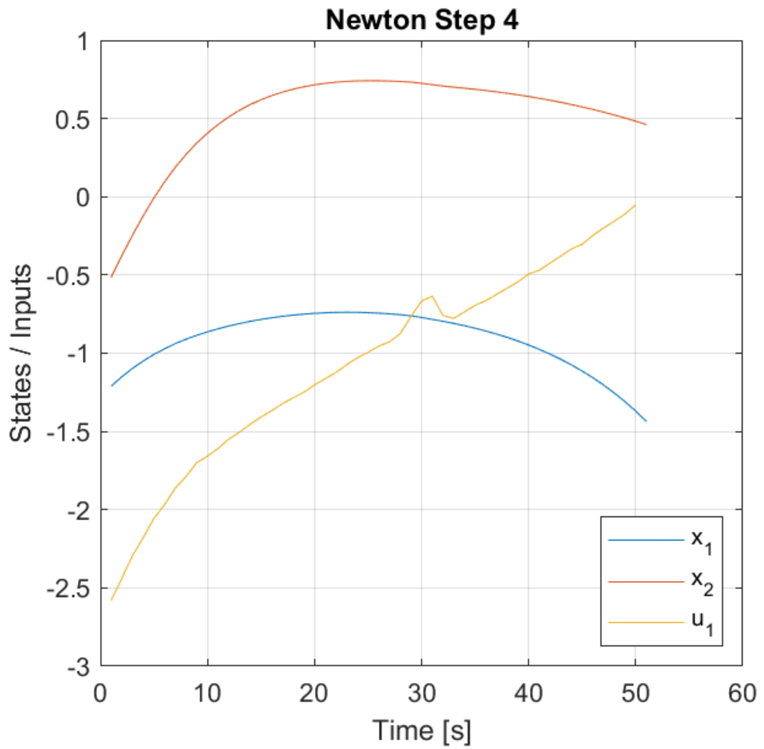
Stephen Monnet

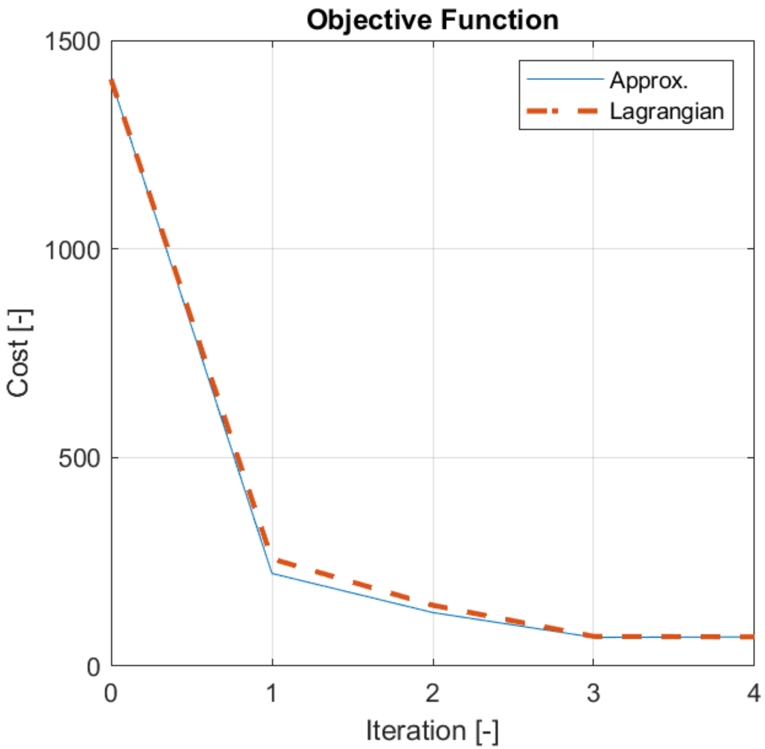




Results : 2 states & 1 input

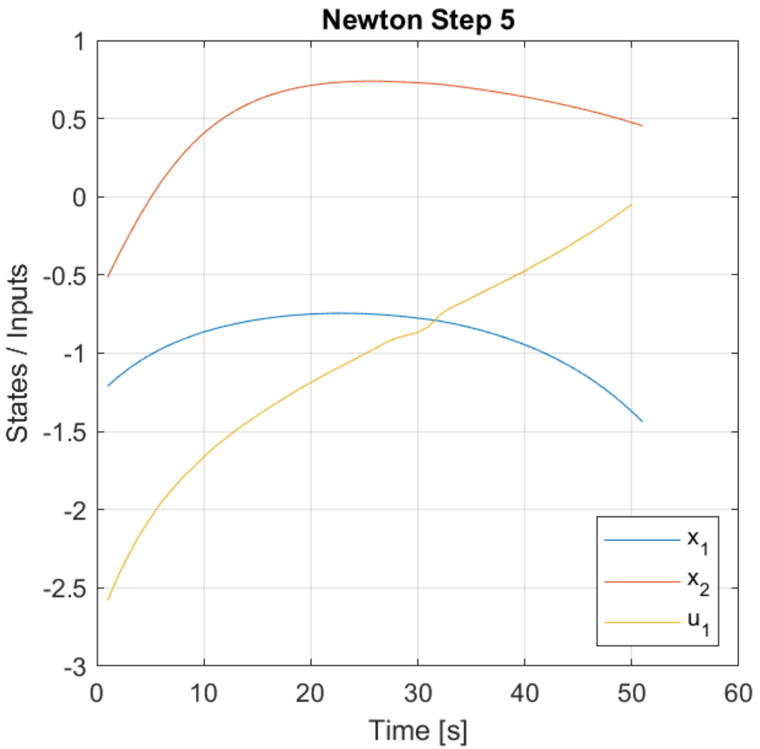
$$\dot{x} = \begin{bmatrix} x_1 + u \cdot \sin(x_1) \\ -x_2 - u \cdot \cos(x_2) \end{bmatrix}$$

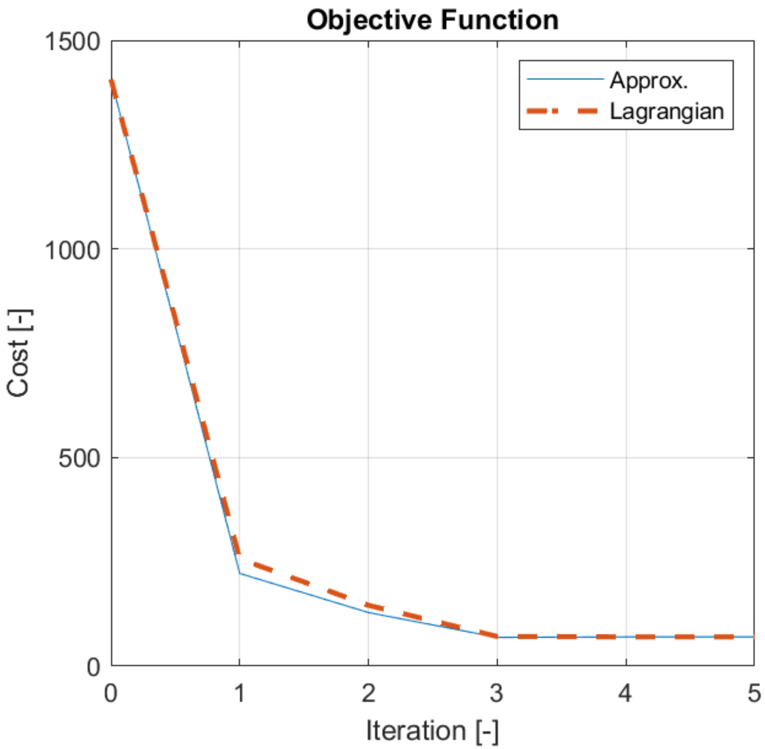




Results : 2 states & 1 input

$$\dot{x} = \begin{bmatrix} x_1 + u \cdot \sin(x_1) \\ -x_2 - u \cdot \cos(x_2) \end{bmatrix}$$

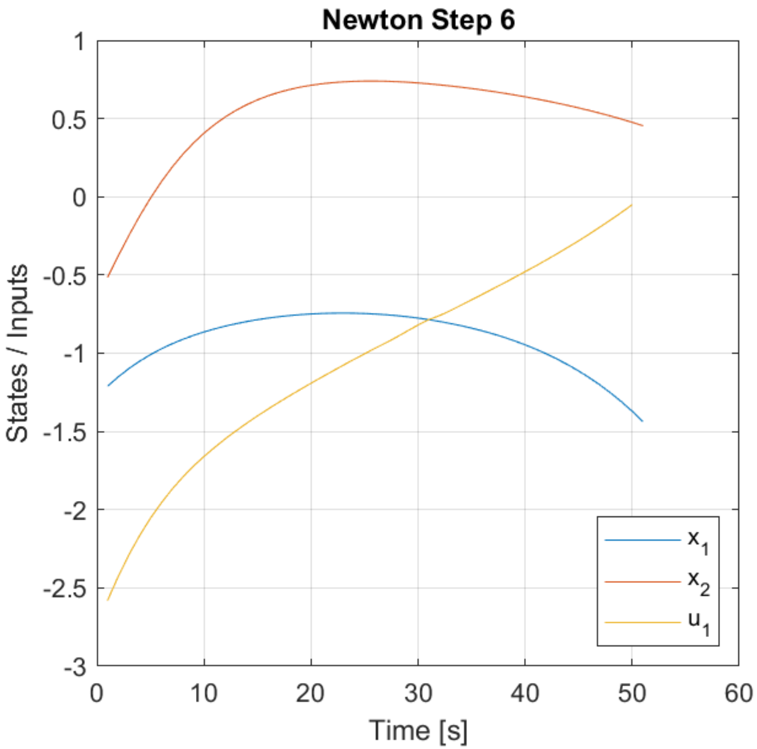


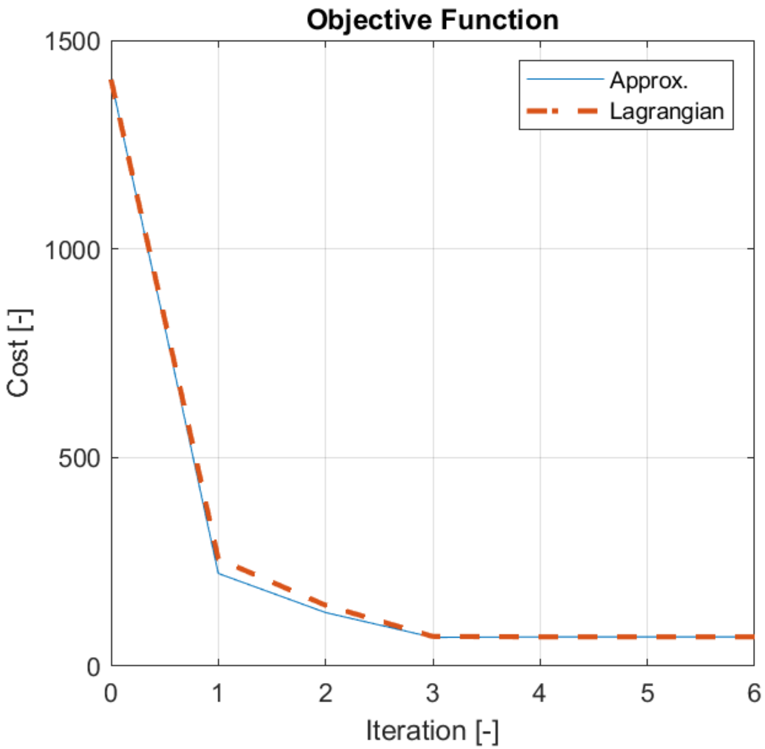


EPFL

Results : 2 states & 1 input

$$\dot{x} = \begin{bmatrix} x_1 + u \cdot \sin(x_1) \\ -x_2 - u \cdot \cos(x_2) \end{bmatrix}$$



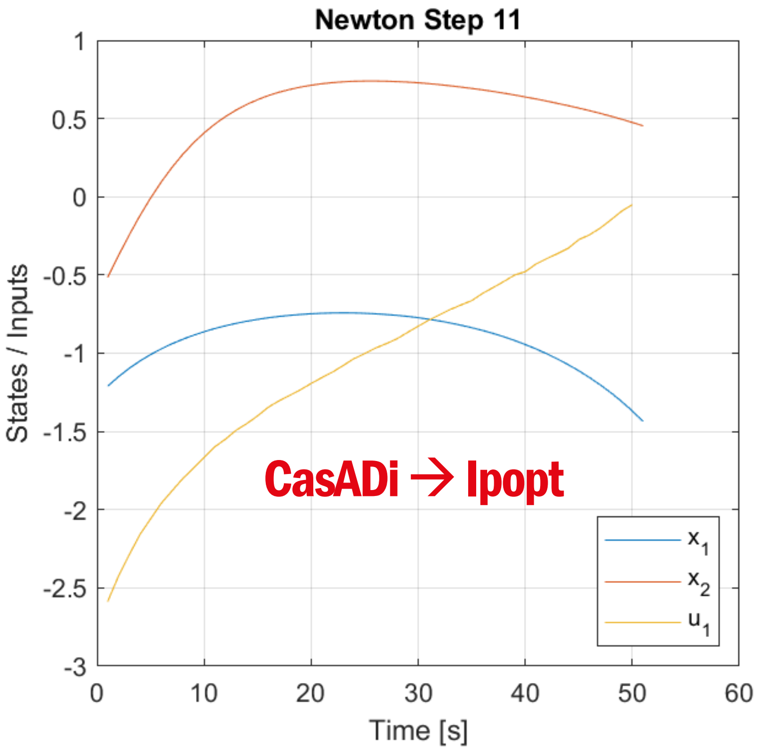


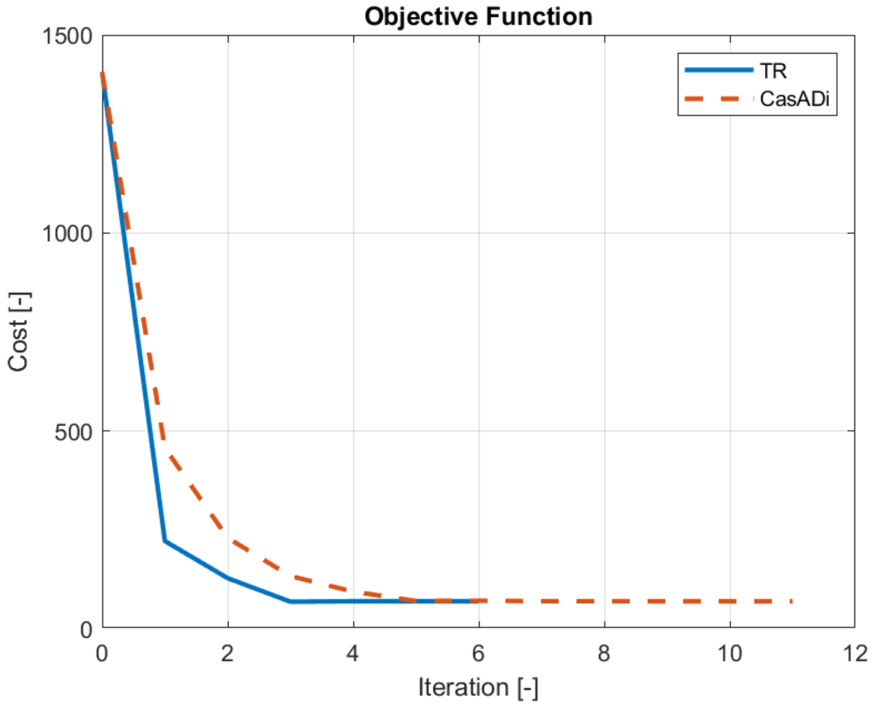
Stephen Monnet

Laboratoire d'Automatique (LA)

Results : 2 states & 1 input

$$\dot{x} = \begin{bmatrix} x_1 + u \cdot \sin(x_1) \\ -x_2 - u \cdot \cos(x_2) \end{bmatrix}$$



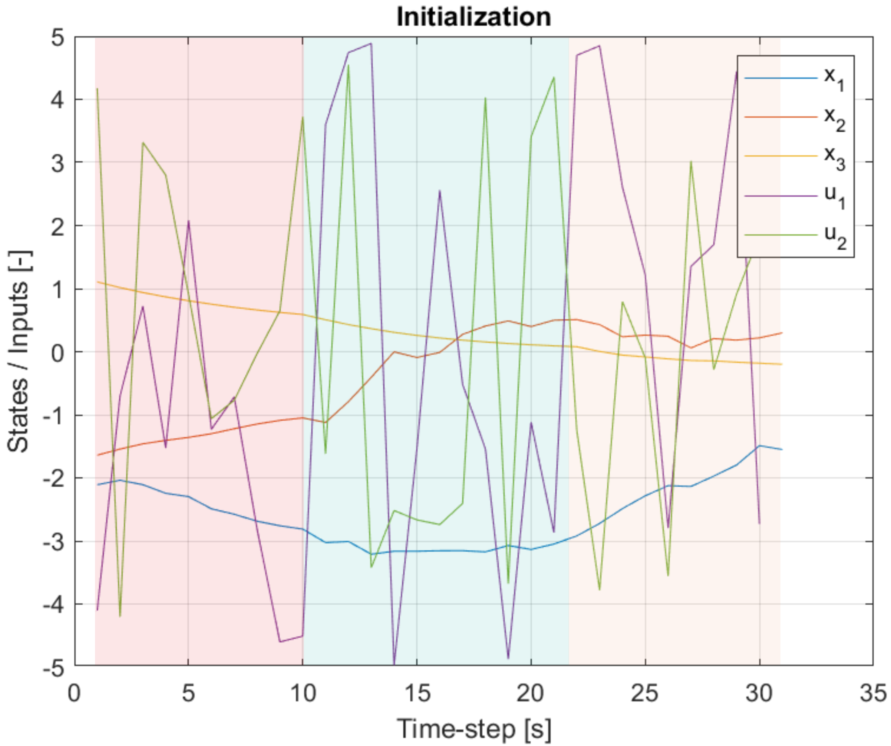


Results : 3 states & 2 inputs

Switching Time System

$$x_{i+1} = x_i + f_1(x_i, u_i) \cdot \delta t \quad \forall i \in [0; 10]$$
$$x_{i+1} = x_i + f_2(x_i, u_i) \cdot \delta t \quad \forall i \in [11; 22]$$
$$x_{i+1} = x_i + f_3(x_i, u_i) \cdot \delta t \quad \forall i \in [23; 30]$$

Stephen Monnet



$$f_1(x) = \begin{bmatrix} x_1 + u_1 \cdot \sin(x_1) \\ -x_2 - u_2 \cdot \cos(x_2) \\ x_2 \cdot x_3 \end{bmatrix}$$

$$f_2(x) = \begin{bmatrix} x_2 + u_2 \cdot \sin(x_2) \\ -x_1 - u_1 \cdot \cos(x_1) \\ x_1 \cdot x_3 \end{bmatrix}$$

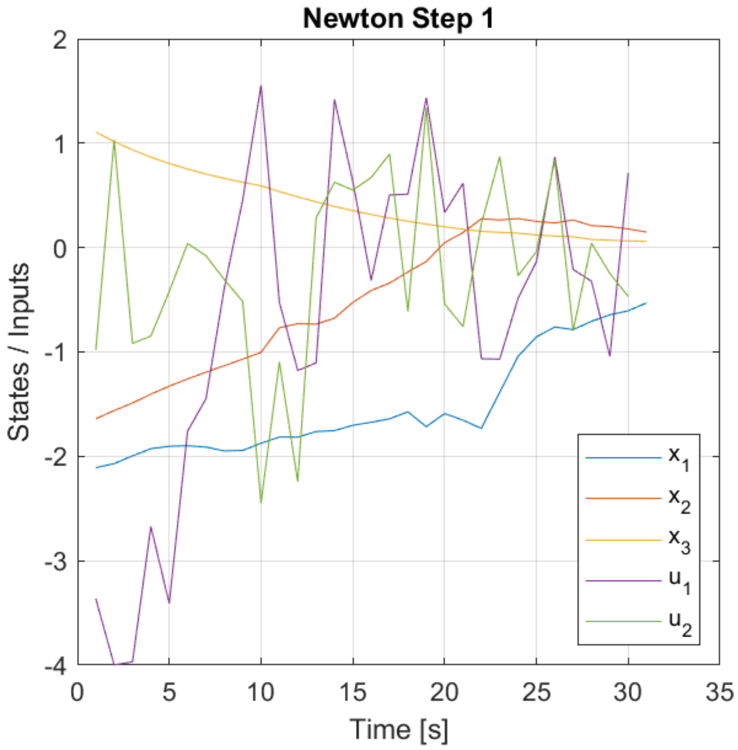
$$f_3(x) = \begin{bmatrix} -x_1 - u_1 \cdot \sin(x_1) \\ -x_2 + u_2 \cdot \cos(x_2) \\ x_1 \cdot x_2 \end{bmatrix}$$

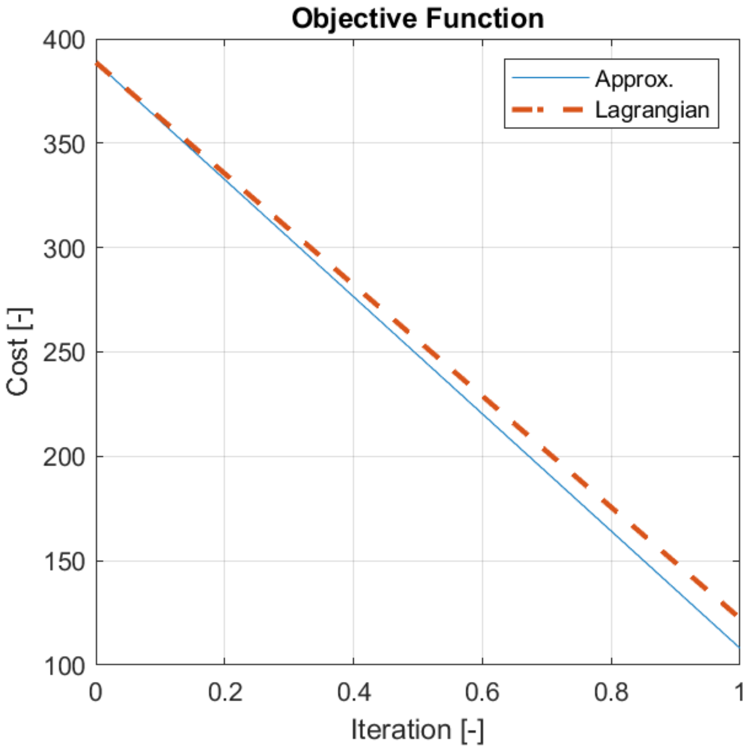
EPFL

Results : 3 states & 2 inputs

Switching Time System

$$x_{i+1} = x_i + f_1(x_i, u_i) \cdot \delta t \quad \forall i \in [0;10]$$
$$x_{i+1} = x_i + f_2(x_i, u_i) \cdot \delta t \quad \forall i \in [11;22]$$
$$x_{i+1} = x_i + f_3(x_i, u_i) \cdot \delta t \quad \forall i \in [23;30]$$

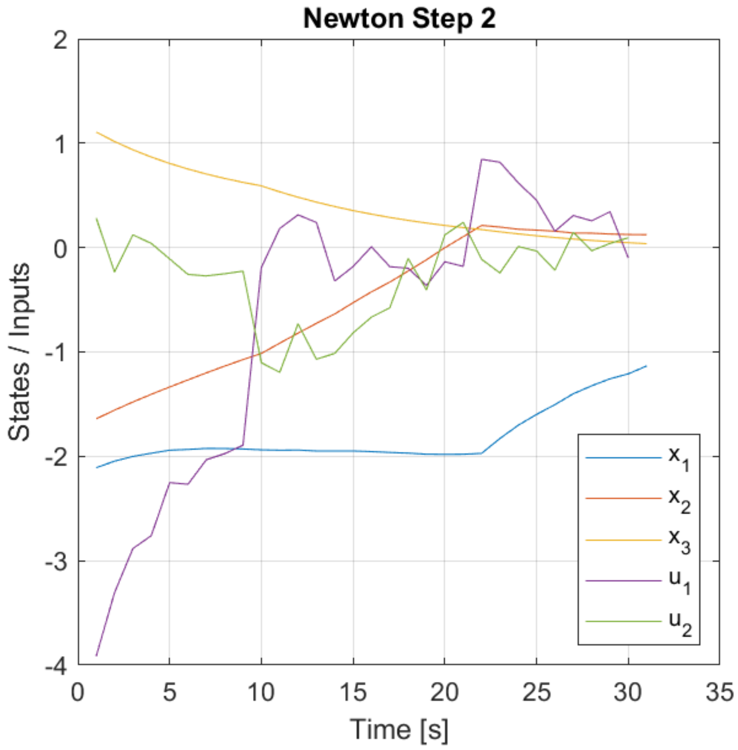


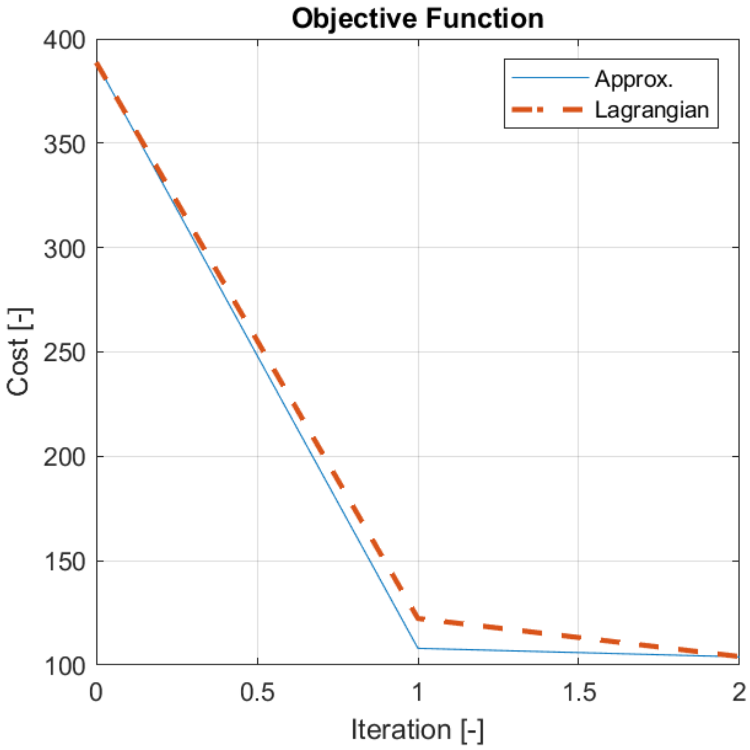


Results : 3 states & 2 inputs

Switching Time System

$$x_{i+1} = x_i + f_1(x_i, u_i) \cdot \delta t \quad \forall i \in [0;10]$$
$$x_{i+1} = x_i + f_2(x_i, u_i) \cdot \delta t \quad \forall i \in [11;22]$$
$$x_{i+1} = x_i + f_3(x_i, u_i) \cdot \delta t \quad \forall i \in [23;30]$$

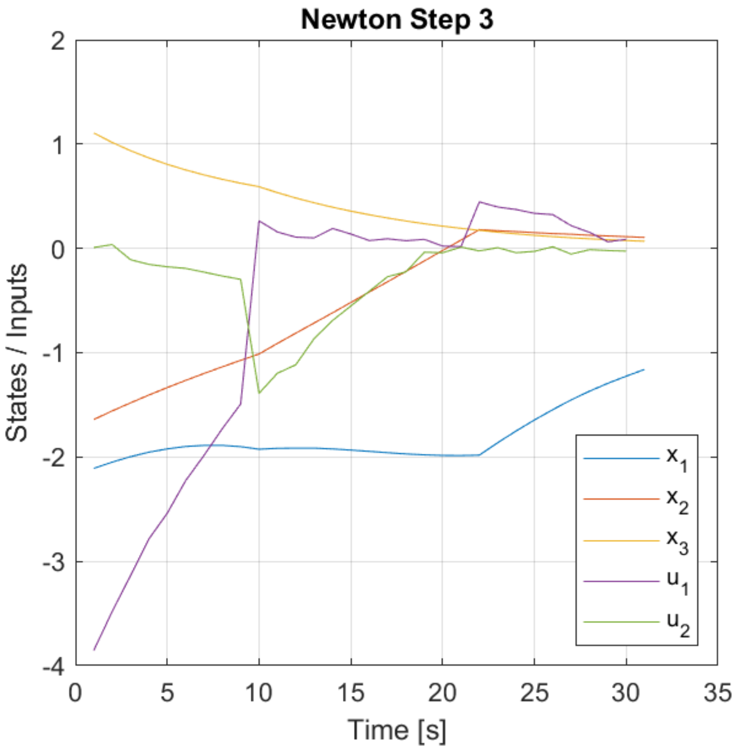


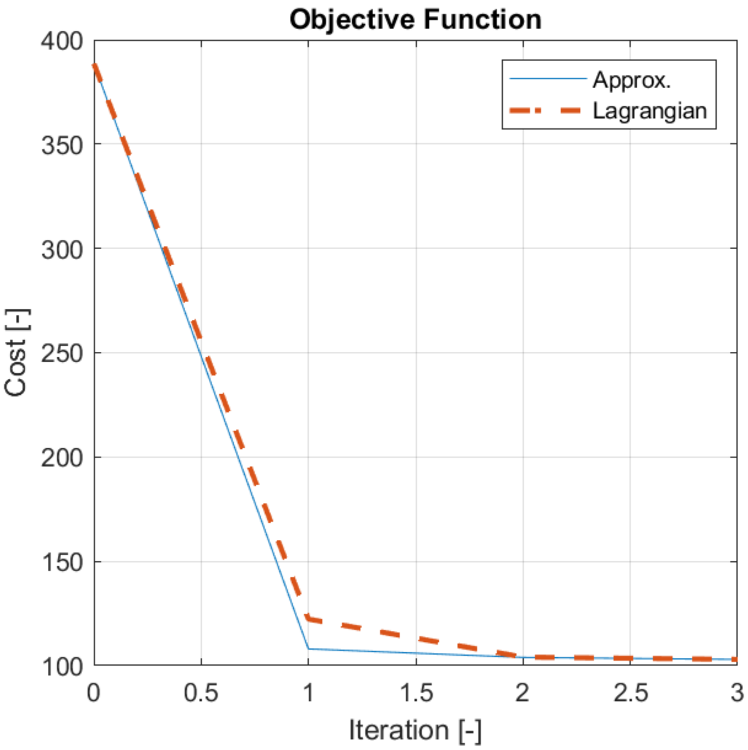


Results : 3 states & 2 inputs

Switching Time System

$$x_{i+1} = x_i + f_1(x_i, u_i) \cdot \delta t \quad \forall i \in [0;10]$$
$$x_{i+1} = x_i + f_2(x_i, u_i) \cdot \delta t \quad \forall i \in [11;22]$$
$$x_{i+1} = x_i + f_3(x_i, u_i) \cdot \delta t \quad \forall i \in [23;30]$$



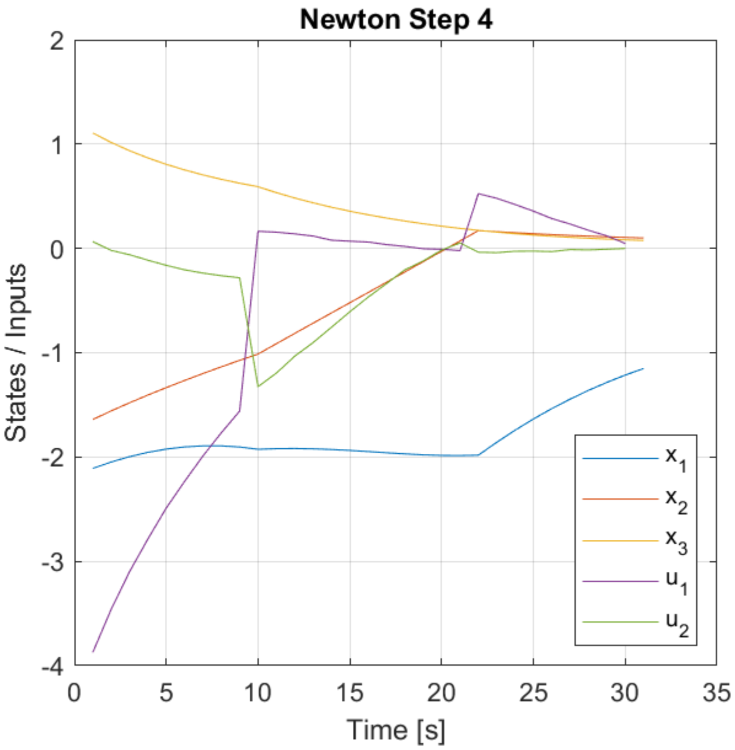


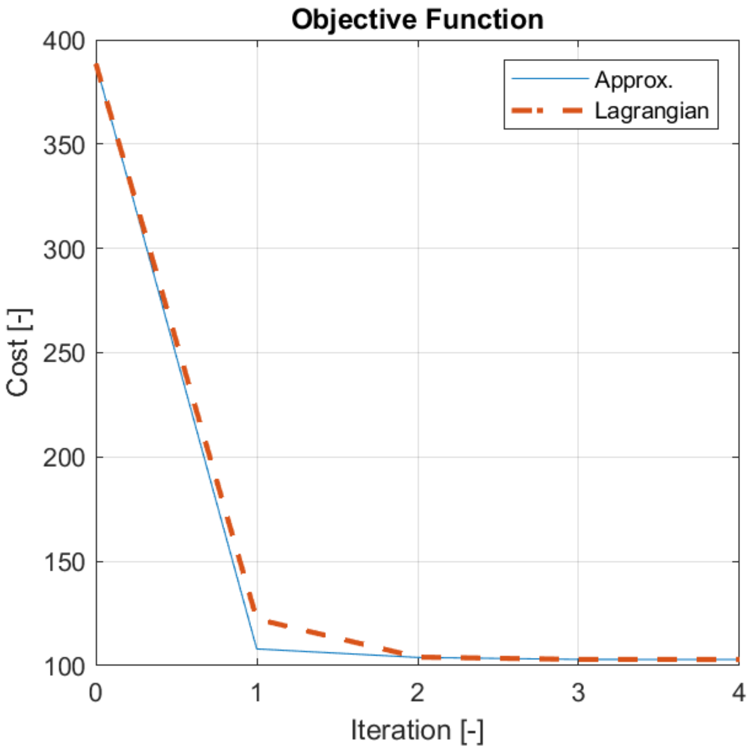
Results : 3 states & 2 inputs

Switching Time System

$$x_{i+1} = x_i + f_1(x_i, u_i) \cdot \delta t \quad \forall i \in [0;10]$$
$$x_{i+1} = x_i + f_2(x_i, u_i) \cdot \delta t \quad \forall i \in [11;22]$$
$$x_{i+1} = x_i + f_3(x_i, u_i) \cdot \delta t \quad \forall i \in [23;30]$$

Stephen Monnet



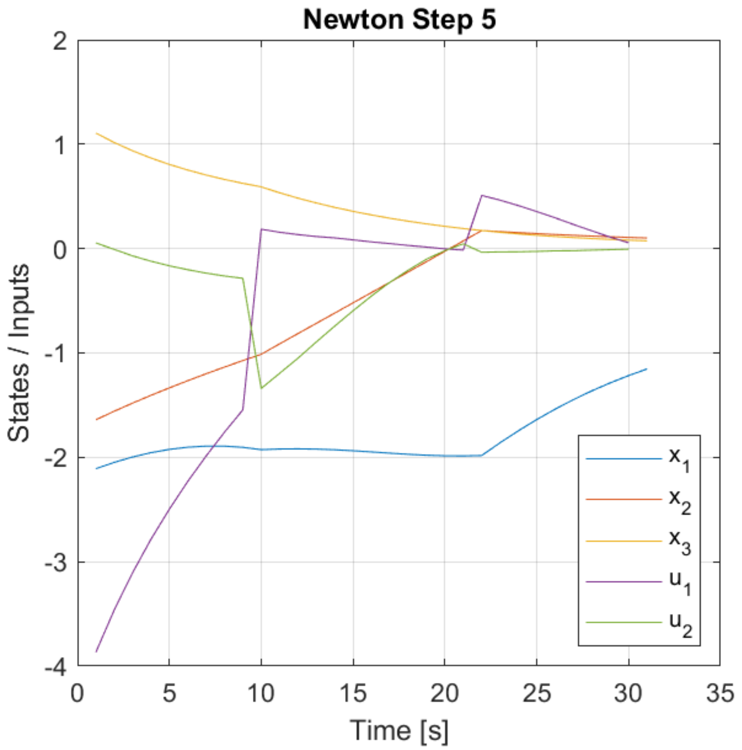


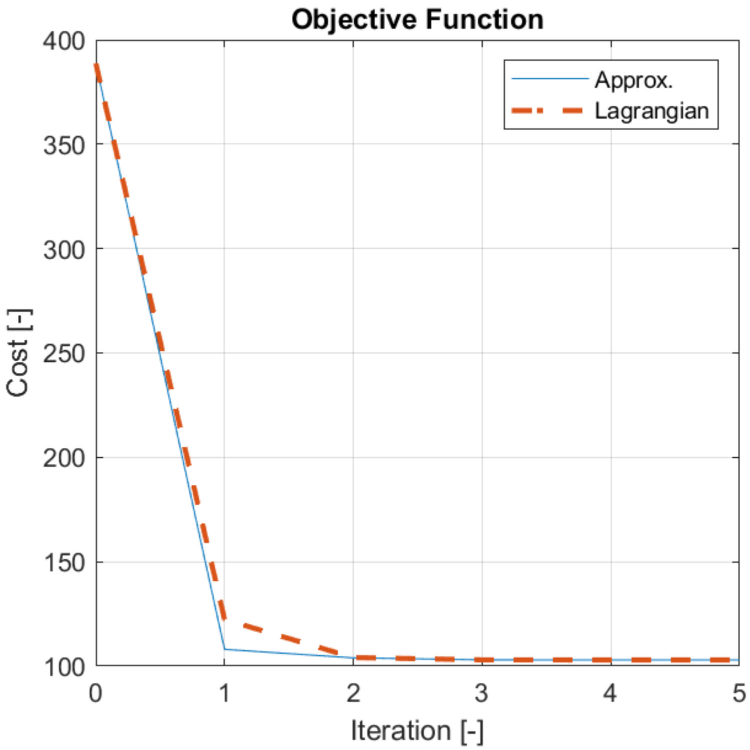
Results : 3 states & 2 inputs

Switching Time System

$$x_{i+1} = x_i + f_1(x_i, u_i) \cdot \delta t \quad \forall i \in [0;10]$$
$$x_{i+1} = x_i + f_2(x_i, u_i) \cdot \delta t \quad \forall i \in [11;22]$$
$$x_{i+1} = x_i + f_3(x_i, u_i) \cdot \delta t \quad \forall i \in [23;30]$$

Stephen Monnet

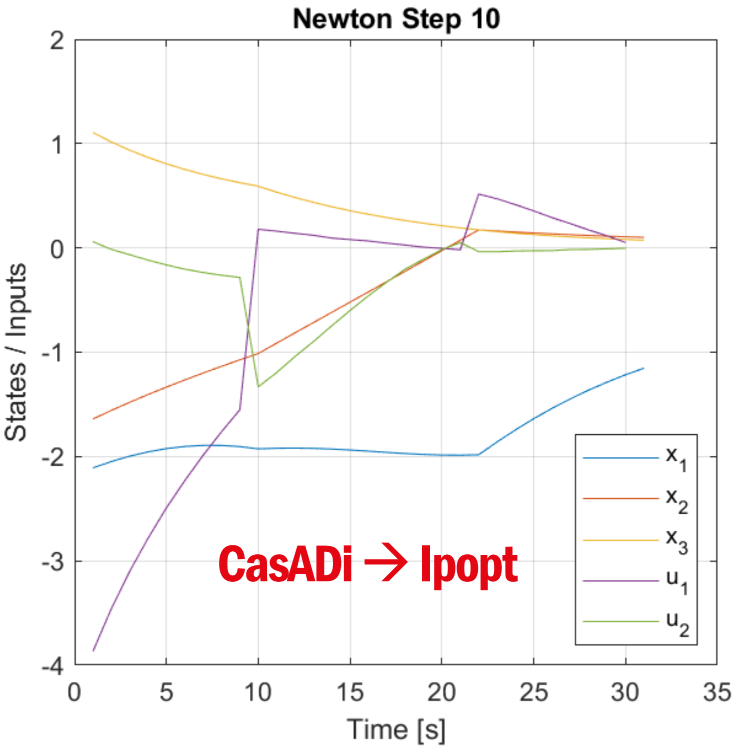


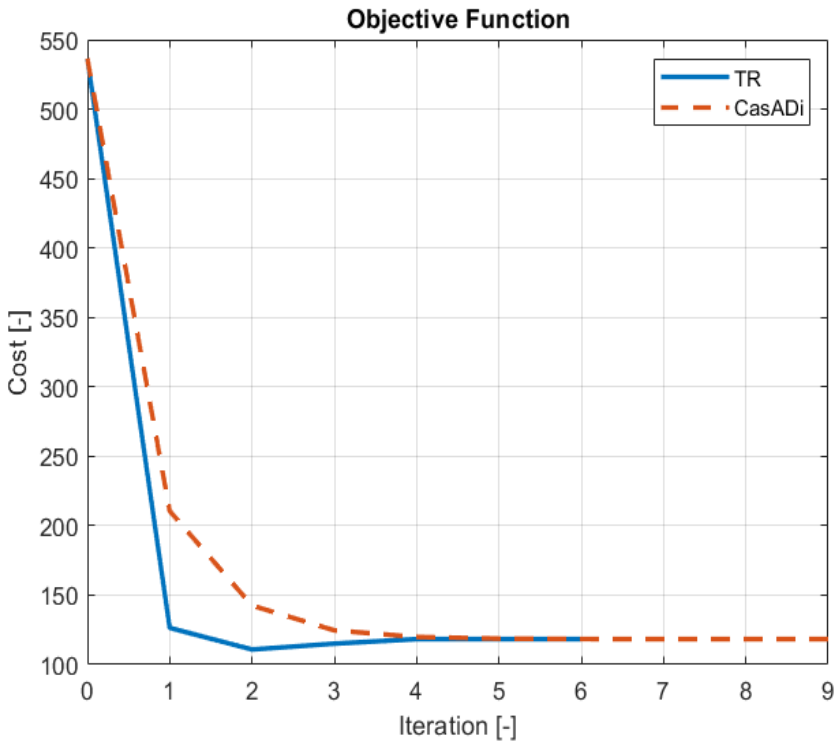


Results : 3 states & 2 inputs

Switching Time System

$$x_{i+1} = x_i + f_1(x_i, u_i) \cdot \delta t \quad \forall i \in [0; 10]$$
$$x_{i+1} = x_i + f_2(x_i, u_i) \cdot \delta t \quad \forall i \in [11; 22]$$
$$x_{i+1} = x_i + f_3(x_i, u_i) \cdot \delta t \quad \forall i \in [23; 30]$$





Stephen Monnet

Conclusion

Work Done

Theoretical Development
- SQP
- Trust-Region
- Riccati Recursion

Implementation
- NOCP & TR_riccati
- Gradient & Hessian Computation

Results
- SISO, MIMO
- Switching-Time systems

Future work

Switching-Time Optimization
- Must adapt algorithm to inequality constrained NOCP

Computational Time
- $f(n_x, n_u, N)$
- Comparison with state-of-the-art solvers

EPFL

Questions ?

Stephen Monnet

École
polytechnique
fédérale
de Lausanne