

CLPP

A User-Friendly Platform For Nonlinear Robust Observer Design



M. Alamir †, P. Bellemain †, L. Boillereaux ‡, I. Queinnec b, M. Titica ‡, N. Sheibat-Othman b, C. Cadet †, G. Besançon †

† Gipsa-lab, University of Grenoble / ‡ GEPEA, Saint Nazaire / b CNRS-LAAS, University de Toulouse / b LAGEP, University of Lyon

French ANR-Grant CLPP

Outline

- What is CLPP for?
- Theoretical Background
- Typical working session
- General comments

Problem Statement (1)

- **Reconstructing the State** of Biological/Chemical Process is mandatory for supervision and/or control task
- Analytical observers scarcely apply to real systems
- Optimization based observers need technicalities that are hard to master by process practitioners

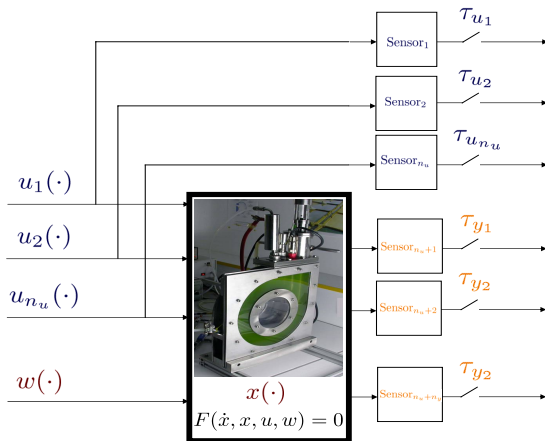
Problem Statement (1)

- **Reconstructing the State** of Biological/Chemical Process is mandatory for supervision and/or control task
- Analytical observers scarcely apply to real systems
- Optimization based observers need technicalities that are hard to master by process practitioners

CLPP Spec.

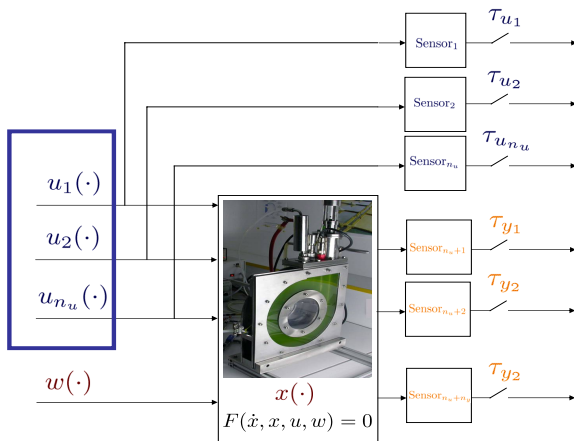
Help process researchers to concentrate on their main problems by **minimizing the Applied-Mathematics like skills** that are needed to design a dynamic state/parameters reconstruction algorithms.

Problem Statement (2)



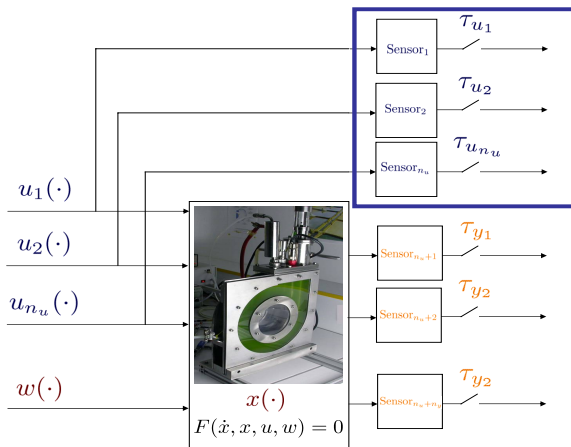
Consider a process with state x and dynamics that may be described by $F(\dot{x}, x, u, w) = 0$ where ...

Problem Statement (2)



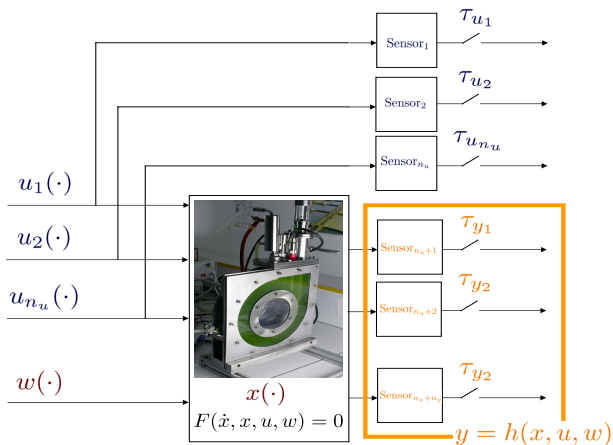
u is the vector of measured inputs :
ex. control, measured disturbances, ...

Problem Statement (2)



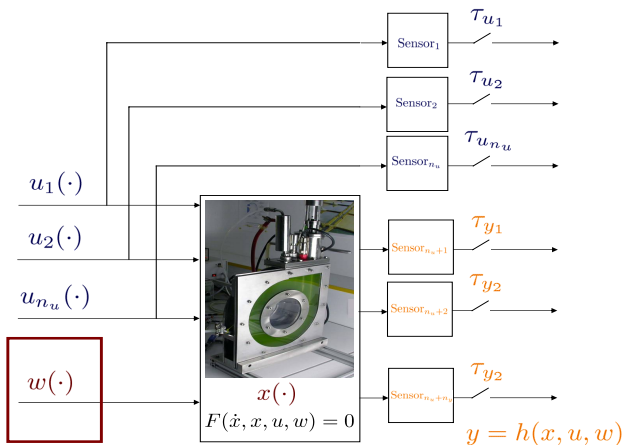
that are assumed to be acquired using dedicated sensors with different sampling rates that may not be uniform or a priori given.

Problem Statement (2)



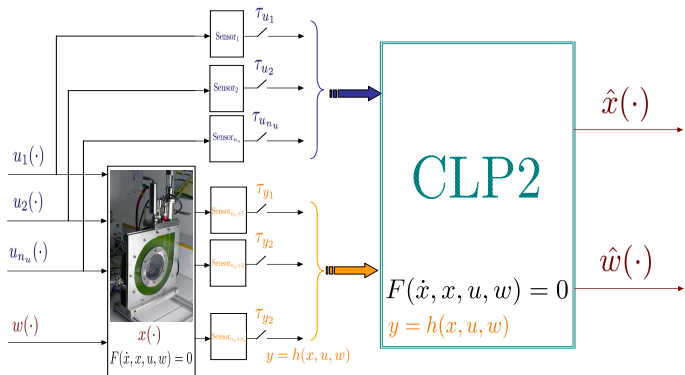
Additional measurements are assumed to be available which form the output vector $y = h(x, u, w)$.

Problem Statement (2)



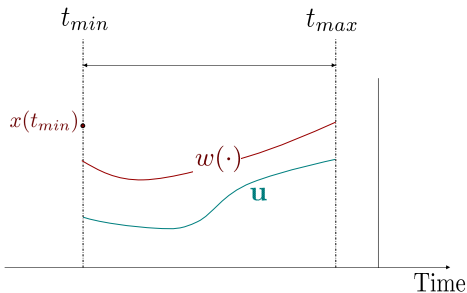
w is the vector of noise/unmeasured modeling error or uncertainty that may affect the system dynamics and the measurement.

Problem Statement (2)



The Software CLP2 is a **generic**, **user-friendly** and **automated** tool that uses the available information (model+measurement) in order to produce on-line dynamic estimation of the unknowns x and $w(\cdot)$





Theoretical Background (1) : reduced parametrization of unknowns



- Objective : Determine $x(t_{min})$ and $w(\cdot)$ over $[t_{min}, t_{max}]$
- $(x(t_{min}), w(\cdot))$ is infinite dimensional \rightarrow use a reduced dimensional parametrization :

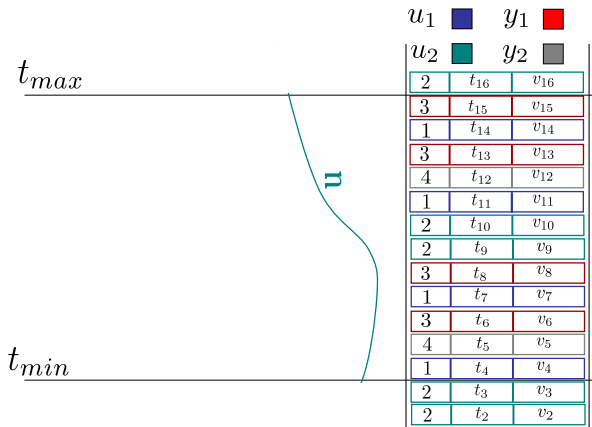
$$[x(t_{min}), w(\cdot)] \leftarrow \Pi(p) \quad ; \quad p \in \mathbb{P}$$

Theoretical Background (2) : Definition of the Cost Function

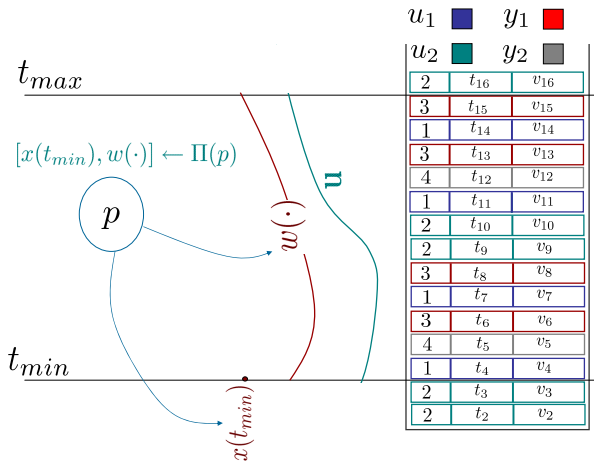
u_1  y_1 
 u_2  y_2 

t_{max}	2	t_{16}	v_{16}
	3	t_{15}	v_{15}
	1	t_{14}	v_{14}
	3	t_{13}	v_{13}
	4	t_{12}	v_{12}
	1	t_{11}	v_{11}
	2	t_{10}	v_{10}
	2	t_9	v_9
	3	t_8	v_8
	1	t_7	v_7
	3	t_6	v_6
	4	t_5	v_5
	1	t_4	v_4
t_{min}	2	t_3	v_3
	2	t_2	v_2

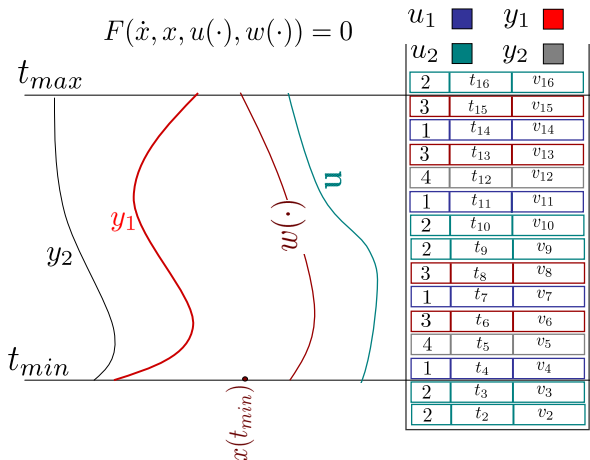
Theoretical Background (2) : Definition of the Cost Function



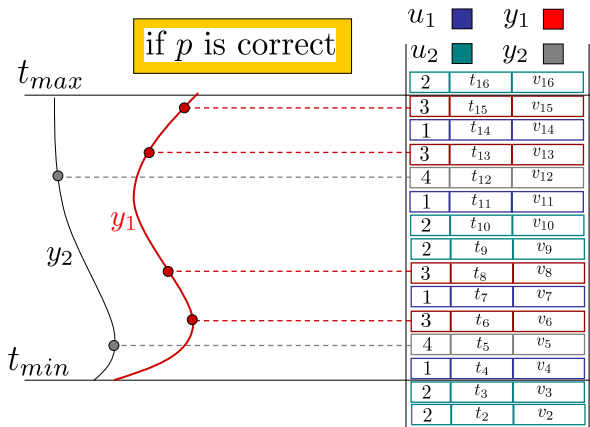
Theoretical Background (2) : Definition of the Cost Function



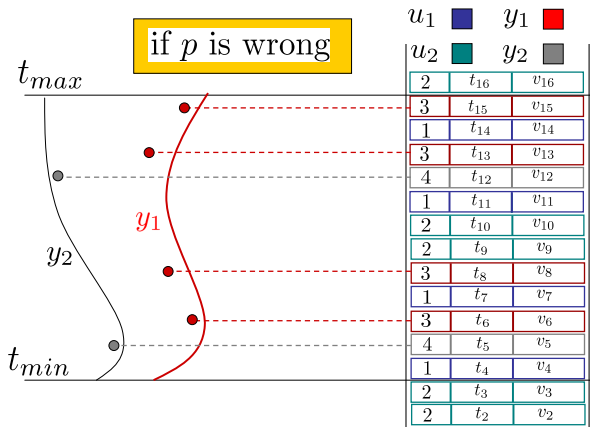
Theoretical Background (2) : Definition of the Cost Function



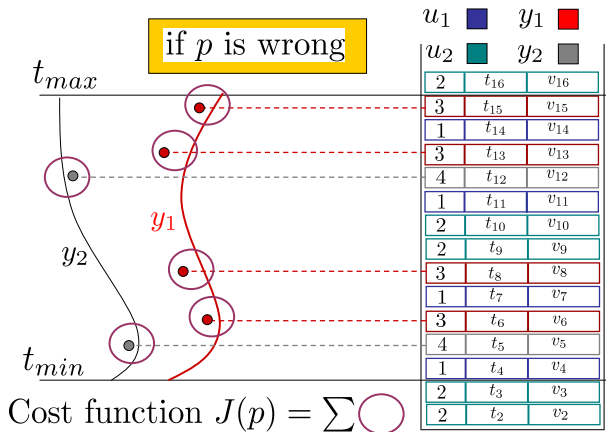
Theoretical Background (2) : Definition of the Cost Function



Theoretical Background (2) : Definition of the Cost Function



Theoretical Background (2) : Definition of the Cost Function



Theoretical Background (3) : Definition of the Cost Function

$$J^{(t)}(\rho) := \sum_{i=1}^{n_y} \left[\sum_{k \in \mathcal{K}_i(t)} |y_i(t_k) - y_i^p(t_k|\rho)|^2 \right]$$

- n_y is the number of sensors
 - $\mathcal{K}_i(t)$ is index of instants in $[t_{min}, t_{max}]$ at which y_i is measured.
 - $y_i^p(t_k|\rho)$ is the predicted output given the guess ρ
-

Theoretical Background (3) : Definition of the Cost Function

$$J^{(t)}(p) := \sum_{i=1}^{n_y} \left[\sum_{k \in \mathcal{K}_i(t)} |y_i(t_k) - y_i^p(t_k|p)|^2 \right]$$

- n_y is the number of sensors
- $\mathcal{K}_i(t)$ is index of instants in $[t_{min}, t_{max}]$ at which y_i is measured.
- $y_i^p(t_k|p)$ is the predicted output given the guess p

More precisely, the family of cost functions :

$$J_{\sigma}^{(t)}(p) := \sum_{i=1}^{n_y} \left[\sum_{k \in \mathcal{K}_i(t)} \phi_{\sigma}(t_k) \cdot |y_i(t_k) - y_i^p(t_k|p)|^2 \right]$$

is used in the software to cross potential singularities.

[M. Alamir, J. Welsh and G. C. Goodwin *Redundancy vs multiple starting points in nonlinear systems related inverse problems*. Automatica, 45 :1052-1057, 2009]

Theoretical Background (3) : Updating of the unknown vector p

Given the cost function :

$$J^{(t)}(p) := \sum_{i=1}^{n_y} \left[\sum_{k \in \mathcal{K}_i(t)} |y_i(t_k) - y_i^p(t_k|p)|^2 \right]$$

The updating rule for p is given by :

$$p(\tau_j) := \mathcal{S}^q(p^+(\tau_{j-1}))$$

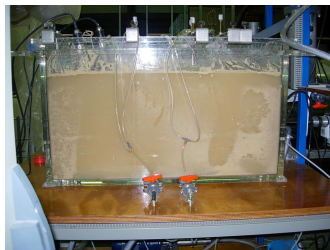
- \mathcal{S} is the map representing one iteration of the optimizer
- q is the **maximum number of iterations** performed at each updating period
- τ_j is the updating instant
- $p^+(\tau_{j-1})$ is the new initial guess inherited from the past estimation $p(\tau_{j-1})$

Case Study : Reduced Model of Activated Sludge Waste-water Process

Reduced Model from [Gomez-Quintero et al. 2000] :

$$\begin{aligned}\dot{x} &= f(x, u, w) \quad (x, u, w) \in \mathbb{R}^4 \times \mathbb{R}^6 \times \mathbb{R}^4 \\ y &= h(x, u, w) = (x_2, x_4)\end{aligned}$$

- $x = (S_S, S_{NO_3}, S_{NH_4}, S_{O_2})$
- S_S : Biodegradable substrate concentration
- S_{NO_3} : Nitrate concentration
- S_{NH_4} : Ammonia concentration
- S_{O_2} : Dissolved oxygen
- $w \in \mathbb{R}^4$: Unknown parameters vector
- $y := (S_{NO_3}, S_{O_2})$: measurement vector



$$\Rightarrow p \in [p_{min}, p_{max}] \in \mathbb{R}^8$$

A Typical Working Session

CLPP

CLPP is a software developed
in the control system department
of GIPSA-lab (SYSCO group)
in the context of the ANR CLPP
GRANT 2007-2010







New problem...

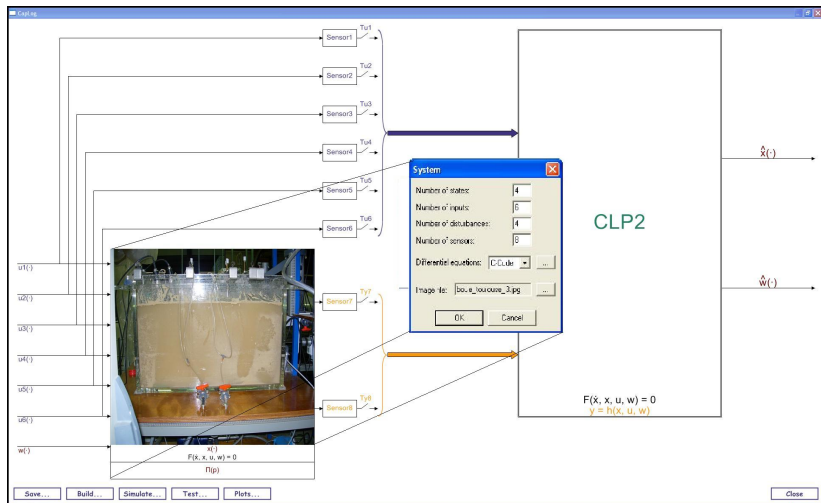
Existing solution...

Exit



Starts a new project

A Typical Working Session



Define the dimensions of the problem

A Typical Working Session

```

C.Code
void Ode(double t, double *x, double *u, double *w, double *xdot)
// t
// x
// u
// w
// xdot

double S=0;
double SN03=1;
double SNH4=2;
double S02=3;
double D=0;
double Dc=1;

double inter1=monod(S02,K_02H)+monod(SN03,K_NO3)*monod(K_02H,S02);
double inter2=monod(S02,K_02H)+eta_N03*monod(SN03,K_NO3)*monod(K_02H,S02);
double inter3=monod(SN03,K_NO3)*monod(K_02H,S02);
double inter4=monod(SNH4,K_NH4AUT)*monod(S02,K_02AUT);
double inter5=monod(S02,K_02H)+monod(SN03,K_NO3)*monod(K_02H,S02);
double inter6=monod(SNH4,K_NH4AUT)*monod(S02,K_02AUT);
double inter7=1-Y_H*Y_H*S*monod(S02,K_02H);
double inter8=monod(SNH4,K_NH4AUT)*monod(S02,K_02AUT);

double alpha[4];
for (int i=0;i<4;i++) alpha[i]=0;

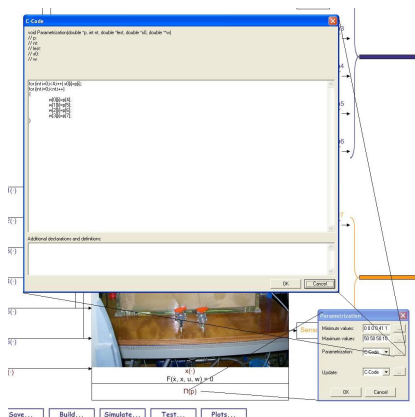
double dSdt=D*(V[2]+Dc*V[3]-D)+Dc*S*alpha[0]*Y_H*S*(inter1+alpha[3]*inter2);
double dSN03dt=(D+Dc)*SN03*alpha[0]*(1-Y_H)*(2.96*Y_H*S*(inter3+alpha[1]*inter4);
double dSNH4dt=D*V[4]-Dc*(SNH4-alpha[0]*NB*S*(inter5+alpha[1]*inter6+alpha[2];
double dS02dt=(D+Dc)*S02+u[5]*(S02-3*S02)-alpha[0]*inter7-4.57*alpha[1]*inter8;

xdot[0]=dSdt;
xdot[1]=dSN03dt;
xdot[2]=dSNH4dt;
xdot[3]=dS02dt;

Additional declarations and definitions:
const double V=0.03;
const double Y_H=0.64;
const double K_02H=0.2;
const double K_NO3=0.5;
const double eta_N03=0.4;
  
```

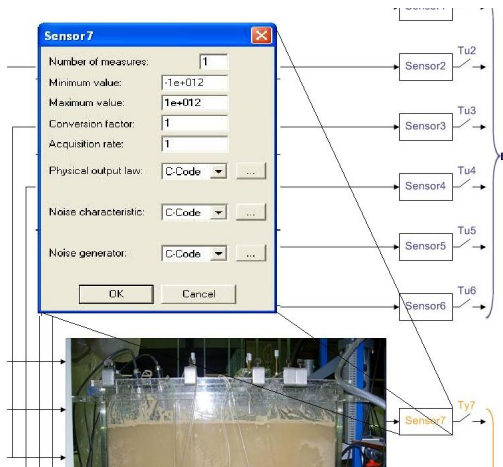
Define the system equations

A Typical Working Session



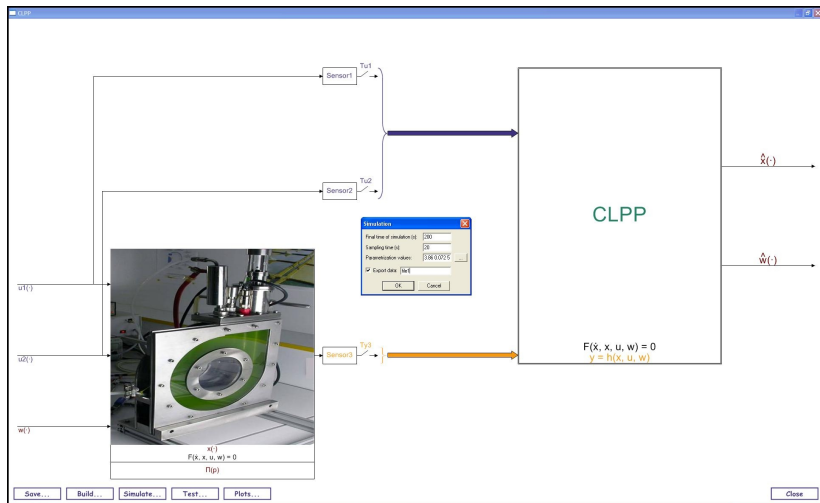
Define the parametrization map $[x(t_{min}), w(\cdot)] \leftarrow \Pi(p)$

A Typical Working Session



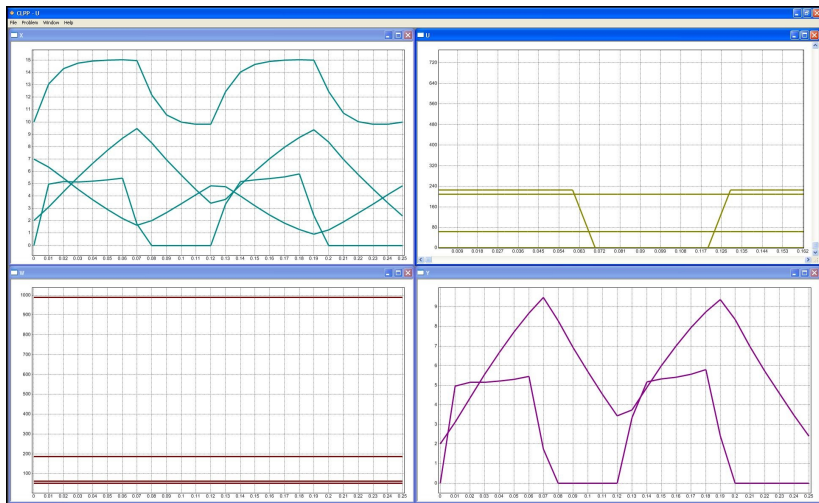
Define the sensors characteristic

A Typical Working Session



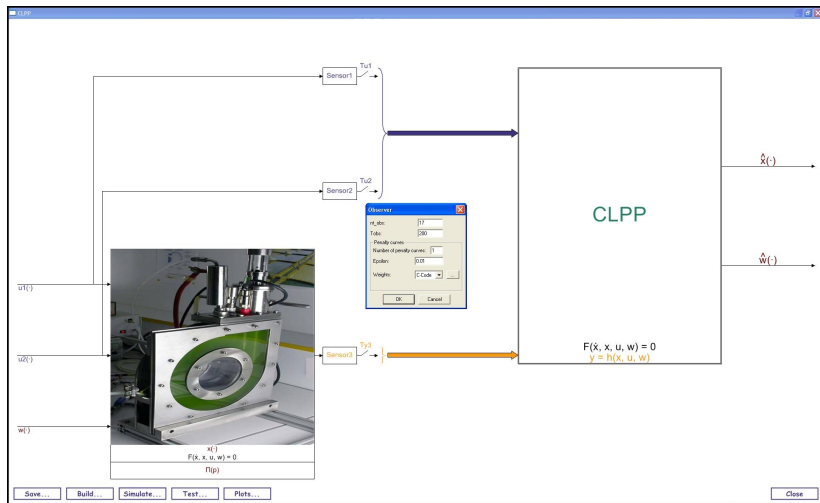
Simulate the system to check the set-up (1)

A Typical Working Session



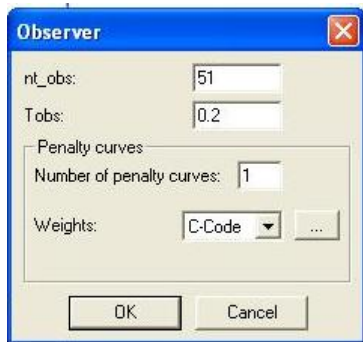
Simulate the system to check the set-up (2)

A Typical Working Session



Define the observer parameters

A Typical Working Session



Define the observer parameters

A Typical Working Session

Sensor7

$x(\cdot)$
 $F(\dot{x}, x, u, w) = 0$
 $\Pi(p)$

Test... Plots...

Solver settings

Solver algorithm: Mazen
 Mazen
 Nelder-Mead
 Torczon

dp:
 alpha: 1
 betaplus: 1.2
 betaminus: 0.8
 ddpmin: 0.001
 alphamin: 0.1

OK Cancel

Test

Number of iterations: 150
 Minimum points: 3
 Parametrization values: 121.5830 ...

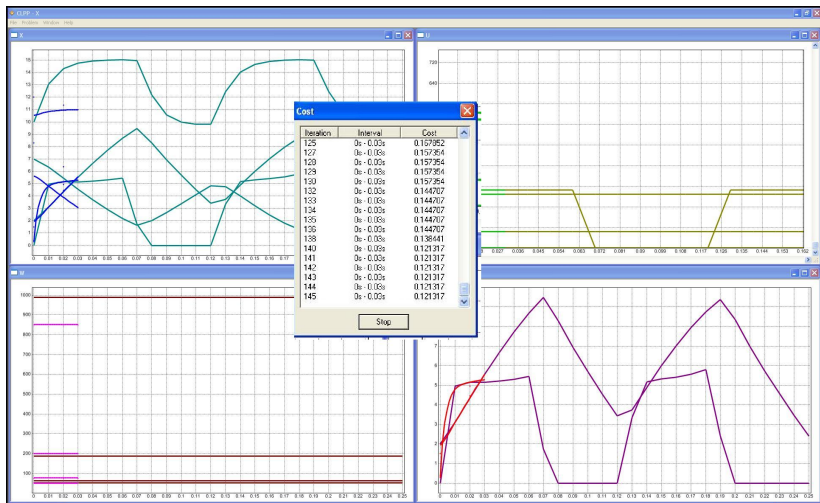
Import data:
 Export data:
 Display simulation

Solver settings...

OK Cancel

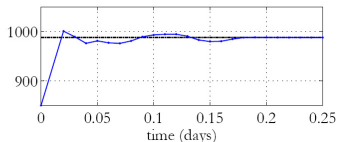
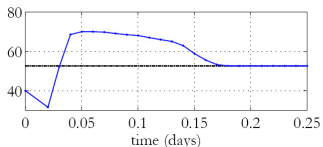
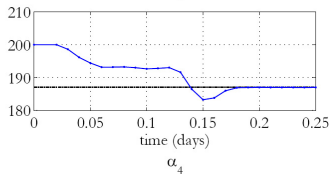
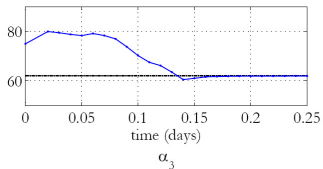
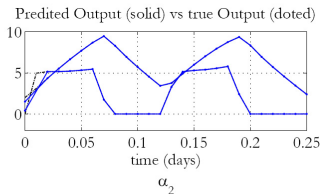
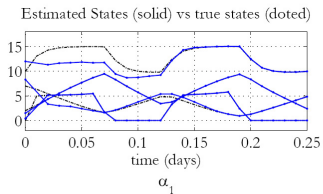
Test the observer performance : Choose the on-line iteration parameters

A Typical Working Session



Test the observer performance : check the convergence

A Typical Working Session



Convergence of the observer for the waste-water case study

Further Interesting Features of CLPP

- synchronous and non uniform measurement acquisition
- Easy use experimental data
- Can be used to detect badly posed inverse problems
- Optimize the process measurement choices
- Automatic creation of a MATLAB/SIMULINK bloc

Further Interesting Features of CLPP

- synchronous and non uniform measurement acquisition
- Easy use experimental data
- Can be used to detect badly posed inverse problems
- Optimize the process measurement choices
- Automatic creation of a MATLAB/SIMULINK bloc

```
n1  t1  v1
n2  t2  v2
.   .   .
ni  ti  vi
.   .   .
nf  tf  vf
..
```

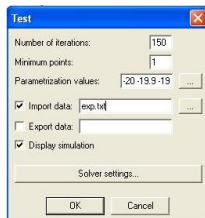
Further Interesting Features of CLPP

- synchronous and non uniform measurement acquisition
- Easy use experimental data
- Can be used to detect badly posed inverse problems
- Optimize the process measurement choices
- Automatic creation of a **MATLAB/SIMULINK bloc**

```

n1  t1  v1
n2  t2  v2
.   .   .
ni  ti  vi
.   .   .
nf  tf  vf
..

```



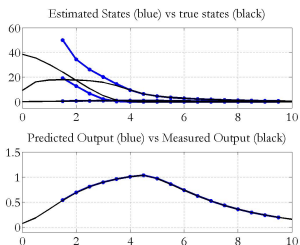
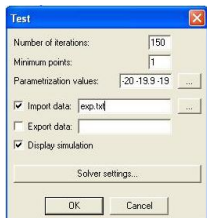
Further Interesting Features of CLPP

- synchronous and non uniform measurement acquisition
- Easy use experimental data
- Can be used to detect badly posed inverse problems
- Optimize the process measurement choices
- Automatic creation of a **MATLAB/SIMULINK bloc**

```

n1  t1  v1
n2  t2  v2
.   .   .
ni  ti  vi
.   .   .
nf  tf  vf
..

```



[Fouchard et al. *Biotechnology and Bioengineering*, 2009]

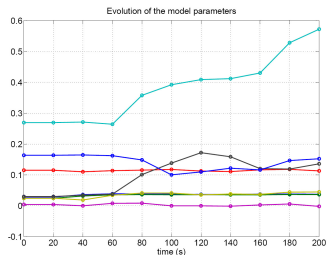
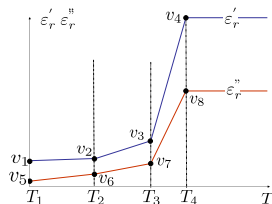
Micro-Wave Tempering

Physics \rightarrow

$$\frac{\partial T}{\partial t} = F(T, T_z, u, \underbrace{\varepsilon_r'(T), \varepsilon_r''(T)}_{??})$$

Discretization \rightarrow

$$\dot{x} = f(x, u, p) \quad (x, p) \in \mathbb{R}^{21} \times [0, 1]^8$$



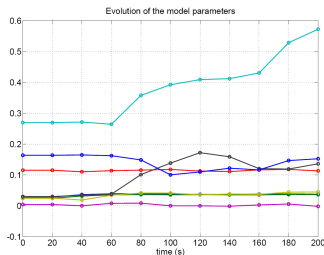
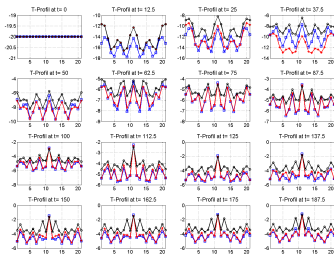
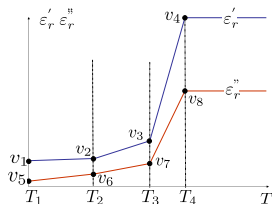
Micro-Wave Tempering

Physics \rightarrow

$$\frac{\partial T}{\partial t} = F(T, T_z, u, \underbrace{\varepsilon_r'(T), \varepsilon_r''(T)}_{??})$$

Discretization \rightarrow

$$\dot{x} = f(x, u, p) \quad (x, p) \in \mathbb{R}^{21} \times [0, 1]^8$$



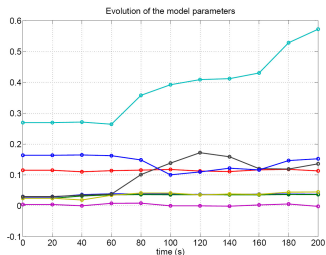
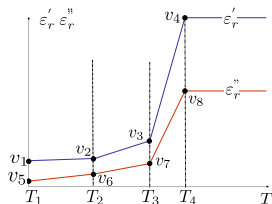
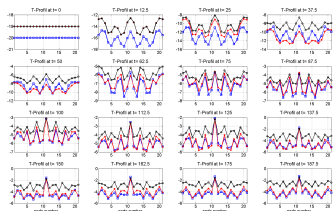
Micro-Wave Tempering

Physics \rightarrow

$$\frac{\partial T}{\partial t} = F(T, T_z, u, \underbrace{\varepsilon_r'(T), \varepsilon_r''(T)}_{??})$$

Discretization \rightarrow

$$\dot{x} = f(x, u, p) \quad (x, p) \in \mathbb{R}^{21} \times [0, 1]^8$$



Release

Freely available (for academic) at

www.gipsa-lab.fr/projects/ANR_CLPP

very soon (autumn 2010 !)

