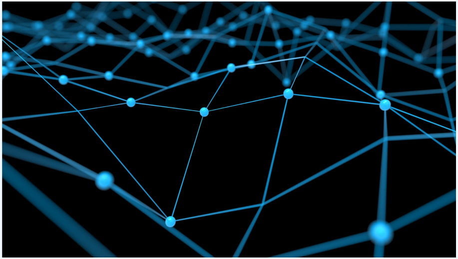
Optimisation based observer and system identifier

Federico Oliva
Domenico Cascone
Enrico Varriale
Corrado Possieri
Mario Sassano
Daniele Carnevale

Università degli studi di Roma "Tor Vergata"



Tor Vergata
Università degli studi di Roma
Dicii
Dipartimento di Ingegneria Civile e Ingegneria Informatica

Adaptive observer design to estimate state and parameters of a nonlinear dynamical system. The main characteristics are the following:

- optimisation based approach
- offline analysis

Inspired by [1], this approach could be used to set initial estimates on local observers (KF) or to identify models.

General model:

- State vector: x
- Measures: y
- Input: u
- parameters: θ

$$\delta x = f(x, \theta, u)$$
$$y = h(x, \theta)$$

Starting from measurements y, the observer aims to both estimate the state x and a set of parameters θ either of the model $f(\cdot)$ or the output mapping $h(\cdot)$.

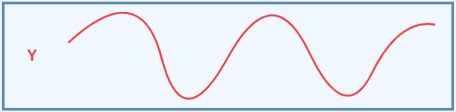


Figure: measured signal Y

Algorithm rationale:

1. Down sampling: N_w
2. Window size: N_{T_s}

e.g. $(N_w = 6,\ N_{T_s} = 4)$

The measurements Y are assumed to be collected every N_{Ts} sampling times and stored in a buffer of dimension N_w.

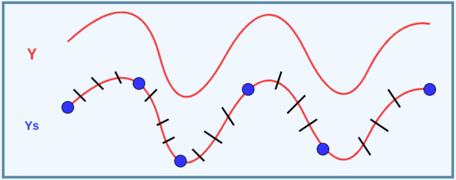


Figure: down-sampled signal Y_s

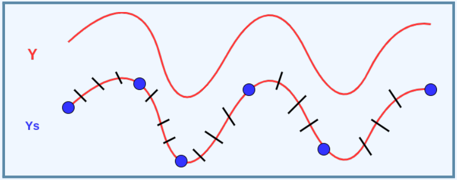


Figure: examples of estimated real signals

Algorithm rationale:

1. Down sampling: N_w
2. Window size: N_{T_s}
3. Optimisation: find best $\xi = [\hat{x}, \hat{\theta}]$ minimising a cost function J evaluated on the current data window

$$e_i = Y_i - H_i(\hat{\xi}_1) = \begin{bmatrix} y_i^1 \\ \vdots \\ y_i^p \end{bmatrix} - \begin{bmatrix} h_1(\hat{\xi}_i) \\ \vdots \\ h_p(\hat{\xi}_i) \end{bmatrix} \in \mathbb{R}^p$$

$$Y_1 = Y(t_k - N_w \cdot N_{T_s})$$

$$\mathbf{E} = [e_1, e_2, \dots, e_{N_w}] \in \mathbb{R}^{p \times N_w}$$

$$J = J(\mathbf{E})$$

Algorithm rationale:

1. Down sampling: N_w
2. Window size: N_{T_s}
3. Optimisation: find best $\xi = [\hat{x}, \hat{\theta}]$ minimising a cost function J evaluated on the current data window

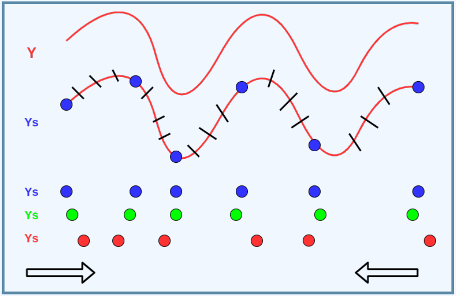


Figure: this approach works both forward and backward in time

Augmented state:

$$\hat{\xi} = \begin{bmatrix} \hat{x}_1 \\ \vdots \\ \hat{x}_n \\ \hat{\theta}_1 \\ \vdots \\ \hat{\theta}_m \end{bmatrix} \implies \min_{\hat{\xi}} J(E)$$

Optimisation run on $\hat{\xi} \implies$ state and parameters estimation. The minimisation algorithm considers the augmented state as the argument.

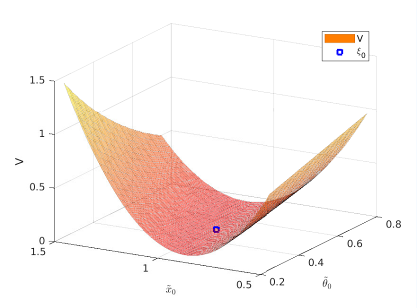


Figure: This figure shows the cost function V($\hat{\xi}$)

Mockup system:

$$\dot{\xi}_1(t) = -\xi_2(t)\xi_1(t), \tag{1a}$$
$$\dot{\xi}_2(t) = 0, \tag{1b}$$
$$y(t) = \xi_1(t). \tag{1c}$$

Evaluation set:

$$\Omega \subset \mathbb{R}^n \text{ and } \xi_0 = [1, 0.5]. \tag{2}$$

Region of Attraction of an equilibrium point ξ_0:

$$\text{RoA}(\xi) \subset \Omega \text{ s.t. } \lim_{t\to\infty} \xi(t) = \xi_0. \tag{3}$$

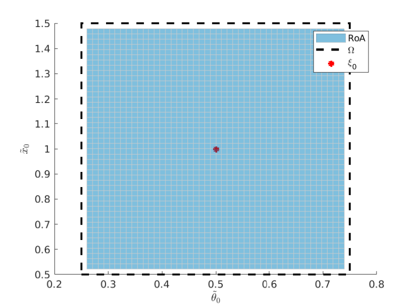


Figure: This figure shows the RoA(ξ) for 6a on Ω. The point ξ_0 is highlighted in red.

Mockup system:

$$\dot{\xi}_1(t) = -|\xi_2(t)|\xi_1(t), \quad (1a)$$
$$\dot{\xi}_2(t) = 0, \quad (1b)$$
$$y(t) = \xi_1(t). \quad (1c)$$

Evaluation set:

$$\Omega \subset \mathbb{R}^n \text{ and } \xi_0 = [1, 0.5]. \quad (2)$$

Region of Attraction of an equilibrium point ξ_0:

$$\text{RoA}(\xi) \subset \Omega \text{ s.t. } \lim_{t\to\infty} \xi(t) = \xi_0. \quad (3)$$

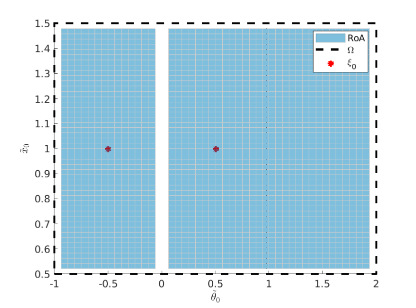


Figure: This figure shows the RoA(ξ) for 6a on Ω. The point ξ_0 is highlighted in red.

Mockup system:

$$\dot{\xi}_1(t) = -|\xi_2(t)|\xi_1(t), \qquad (1a)$$
$$\dot{\xi}_2(t) = 0, \qquad (1b)$$
$$y(t) = \xi_1(t). \qquad (1c)$$

Evaluation set:

$$\Omega \subset \mathbb{R}^n \text{ and } \xi_0 = [1, 0.5]. \qquad (2)$$

Region of Attraction of an equilibrium point ξ_0:

$$\mathrm{RoA}(\xi) \subset \Omega \text{ s.t. } \lim_{t\to\infty} \xi(t) = \xi_0. \qquad (3)$$

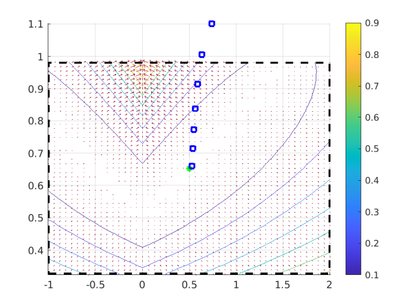


Figure: Contour plot estimation starting from $\hat{\xi}_0 = (1.1, 0.7)$. Optimisations are the blue squares while the green one is ξ_0.

Mockup system:

$$\dot{\xi}_1(t) = -|\xi_2(t)|\xi_1(t), \tag{1a}$$
$$\dot{\xi}_2(t) = 0, \tag{1b}$$
$$y(t) = \xi_1(t). \tag{1c}$$

Evaluation set:

$$\Omega \subset \mathbb{R}^n \text{ and } \xi_0 = [1, 0.5]. \tag{2}$$

Region of Attraction of an equilibrium point ξ_0:

$$\mathrm{RoA}(\xi) \subset \Omega \text{ s.t. } \lim_{t\to\infty} \xi(t) = \xi_0. \tag{3}$$

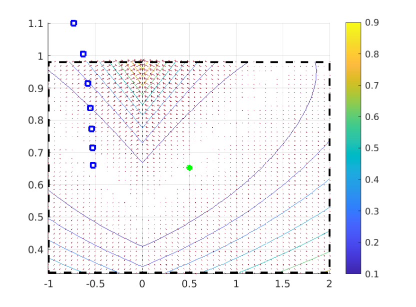


Figure: Same as before but with $\hat{\xi}_0 = (1.1, -0.7)$.

This section goes through some preliminary results on the state and parameters estimation capabilities of the proposed algorithm. The paragraph unfolds as follows:

- Cost function discussion
- Down sampling

Each optimisation step is assumed to have a maximum number of iteration allowed of $Max_{iter} = 100$ and a down-sampling of $(N_w = 5, N_{T_s} = 3)$.

Cost function:

$$J(\mathbf{E}) = \sum_{i=1}^{N_w} \left[e_i^T W_1(i) e_i \right] + (\theta_0 - \theta)^T W_2 (\theta_0 - \theta), \tag{4}$$

where $W_1, W_2 \in \mathbb{R}^{n+m}$ are weight matrices and θ_0 is the initial estimate before the optimisation process.

Down sampling selection:

1. Model: double pendulum
2. Estimation: state + friction coefficients
3. $N_w = 5$ and $T_s = 0.05$s

Examples:

```
simulations/lecture/pendulum_friction_correct_sampling_0503
simulations/lecture/pendulum_friction_wrong_sampling_0503
```

W3 term in J:

1. Model: double pendulum
2. Estimation: state + friction coefficients
3. $N_w = 5$ and $T_s = 0.05$s

Examples:

```
simulations/lecture/pendulum_friction_correct_sampling_0503
simulations/lecture/pendulum_friction_spring_term_0503
```

This section describes some improvements of the algorithm, with the main goal to speed up the estimation process.

Next covered topics:

- Filters on cost function: how act on J in order to both speed up the estimation and increase the system observability.
- Adaptive sampling: how to speed up the process by implementing self tuning capabilities

How to speed up the computational time?

- error definition: $e_i = Y_i - H_i(\hat{\xi}_1)$
- N_w tuning: from [1] a good choice is $N_w \geq 2n + 1$ with $n = \dim \xi$
- proposed improvement: add filtered errors in the cost function J. By adding information, N_w decreases, i.e.

$$J = J(\mathbf{E}, \dot{\mathbf{E}}, \int \mathbf{E}). \tag{5}$$

By reducing N_w state propagation will take lesser time, speeding up the process.

How to speed up the computational time?

- error definition: $e_i = Y_i - H_i(\hat{\xi}_1)$
- N_w tuning: from [1] a good choice is $N_w \geq 2n + 1$ with $n = \dim \xi$
- proposed improvement: add filtered errors in the cost function J. By adding information, N_w decreases, i.e.

$$J(\mathbf{E}) = \sum_{i=1}^{N_w} \left[e_i^T W_1(i) e_i + \dot{e}_i^T W_2(i) \dot{e}_i + \int e_i^T W_3(i) e_i \right] + (\theta_0 - \theta)^T W_5 (\theta_0 - \theta), \quad (5)$$

Mockup system:

$$\dot{\xi}_1(t) = -\xi_2(t)\xi_1(t), \tag{6a}$$
$$\dot{\xi}_2(t) = 0, \tag{6b}$$
$$y(t) = \xi_1(t). \tag{6c}$$

Cost function :

$$\mathcal{V}(\tilde{\xi}) : \mathbb{R}^n \implies \mathbb{R} \tag{7}$$

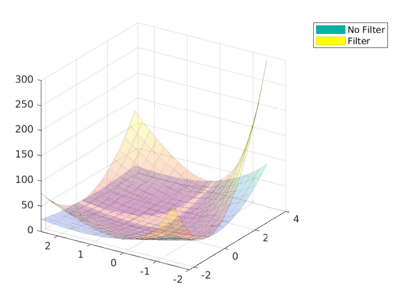


Figure: Comparison between cost function with different down-sampling, namely with $N = 3$ and $N = 3$ with filter.

Mockup system:

$$\dot{\xi}_1(t) = -\xi_2(t)\xi_1(t), \tag{6a}$$
$$\dot{\xi}_2(t) = 0, \tag{6b}$$
$$y(t) = \xi_1(t). \tag{6c}$$

Convex region:

$$\mathcal{H} \subset \mathrm{RoA}(\xi_0) \tag{7}$$

in which $\triangle f(\xi)|_{\xi^*} > 0$

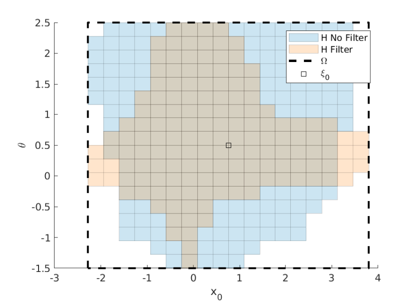


Figure: Comparison between $\mathcal{H}$ with different down-sampling, namely with $N = 3$ with the derivative filter and $N = 3$. The point ξ_0 is highlighted as a black square.

Refine - Filters action on J

- $N_w = 5$,no filtering: 29s
- $N_w = 3$,yes filtering: 65s

Examples:

```
simulations/lecture/pendulum_friction_correct_sampling_0503
simulations/lecture/pendulum_friction_derivative_term_0303
```

The sampling rate should be chosen depending on the signal richness. Consider the time evolution of the estimation error:

$$e_i = Y_i - H_i(\hat{\xi}_1) \in \mathbb{R}^p, i \in \{1, \ldots, N_w\} \tag{6}$$

The goodness of the estimation is gauged through the index ν, namely:

$$\nu = \sum_{i=1}^{N_w} ||e_i|| \tag{7}$$

Refining - adaptive sampling

The richer the signal, the higher ν gets, as smaller model differences are enhanced during the dynamics integration. Therefore the estimation process can be triggered at runtime thresholding ν:

Policy (Adaptive sampling)

```
if (nu < nu_bar)
    measure();
else
    measure();
    estimate();
end
```

- $N_w = 5$, yes filtering, no adaptive: 29s
- $N_w = 3$, yes filtering, yes adaptive: 24.6s

Examples:

```
simulations/lecture/pendulum_friction_derivative_term_0303
simulations/lecture/pendulum_friction_derivative_term_0303_ad
```

Refining - recap parameters

Convergence:

- RoA
- $\mathcal{H}$

Optimisation parameters:

- J filters
- $\mathtt{Max}_{\mathtt{iter}}$
- Adaptive ON/OFF

Sampling parameters:

- N_w
- N_{T_s}

Thanks for the attention