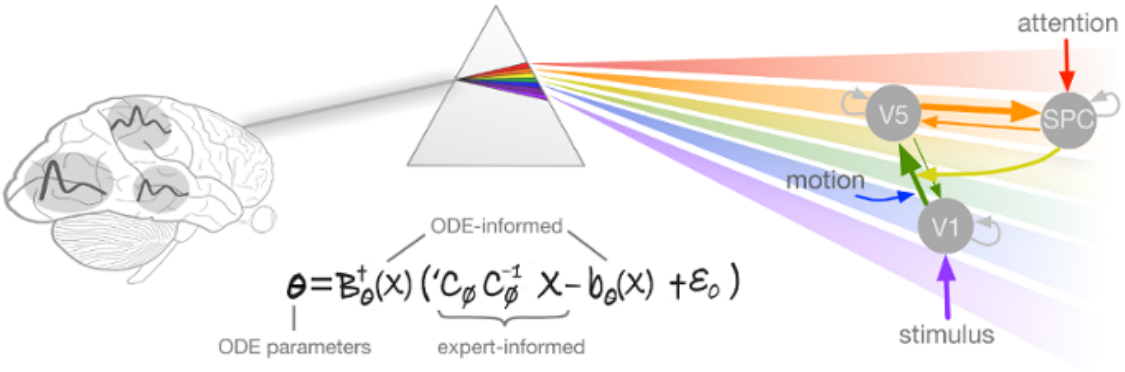
.

Code Documentation of Variational Gradient Matching for Dynamical Systems



Authors:

Nico Stephan Gorbach and **Stefan Bauer**, email: nico.gorbach@gmail.com

Contents:

Code documentation for Variational Gradient Matching (VGM) described in the NIPS (2018) paper Scalable Variational Inference for Dynamical Systems by Nico S. Gorbach, Stefan Bauer and Joachim M. Buhmann (arxiv paper: https://arxiv.org/pdf/1705.07079.pdf). Please cite our paper if you use our program for a further publication. The derivations in this document are also given in the doctoral thesis https://www.research-collection.ethz.ch/handle/20.500.11850/261734 as well as in parts of Wenk et al. (2018). Matlab code is illustrated in purple and the console output is illustrated in blue.

The code is available at `https://github.com/ngorbach/Variational_Gradient_Matching_for_Dynamical_Systems`. Online documentation of this code is available at `https://ngorbach.github.io/Variational_Gradient_Matching_for_Dynamical_Systems/#17`. In the repository, run one of the Matlab scripts *VGM_for_Lotka_Volterra.m*, *VGM_for_Lorenz96.m* and *VGM_for_Lorenz_attractor.m*. Alternatively, one can also run the live scripts *VGM_for_Lotka_Volterra_live_script.mlx*, *VGM_for_Lorenz96_live_script.mlx* or *VGM_for_Lorenz_Attractor_live_script.mlx*. The Matlab symbolic toolbox is required for this code.

Contents

VGM for Lotka-Volterra

Example dynamical system used in this code: **Lotka-Volterra** system with **half of the time points unobserved**. The ODE parameters are also unobserved.

1.1 User Input

1.1.1 Simulation Settings

- **true ODE parameter**: Input a row vector of real numbers of size 1 x 4:

```
simulation.ode_param = [2,1,4,1];
```

- **final time for simulation**: Input a positive real number:

```
simulation.final_time = 2;
```

- **observation noise**: Input a function handle:

```
simulation.state_obs_variance = @(mean)(bsxfun(@times,[0.5^2,0.5^2],...
ones(size(mean))));
```

- **time interval between observations**: Input a positive real number:

```
simulation.interval_between_observations = 0.1;
```

1.1.2 Estimation Settings

- **Kernel Parameters ϕ**: Input a row vector of positive real numbers of size 1 x 2:

```
kernel.param = [10,0.2];
```

- **Error variance on state derivatives (i.e. γ)**: Input a row vector of positive real numbers of size 1 x 2:

```
state.derivative_variance = [6,6];
```

- **Estimation times**: Input a row vector of positive real numbers in ascending order:

```
time.est = 0:0.1:4;
```

Preliminary operations

```
close all; clc; addpath('VGM_functions')
```

1.2 Preprocessing

```
[symbols,simulation,ode,odes_path,coupling_idx,opt_settings,plot_settings] = ...
preprocessing_Lotka_Volterra (simulation);
```

```
ODEs:

/ d x_1                                  \
| ----- == theta_1 x_1 - theta_2 x_1 x_2 |
|   dt                                   |
|                                        |
| d x_2                                  |
| ----- == theta_4 x_1 x_2 - theta_3 x_2 |
\   dt                                   /
```

1.3 Mass Action Dynamical Systems

A deterministic dynamical system is represented by a set of K ordinary differential equations (ODEs) with model parameters $\boldsymbol{\theta} \in \mathbb{R}^d$ that describe the evolution of K states $\mathbf{x}(t) = [x_1(t), \ldots, x_K(t)]^T$ such that:

$\dot{\mathbf{x}}(t) = \frac{d\mathbf{x}(t)}{dt} = \mathbf{f}(\mathbf{x}(t), \boldsymbol{\theta}) \qquad (1),$

A sequence of observations, $\mathbf{y}(t)$, is usually contaminated by measurement error which we assume to be normally distributed with zero mean and variance for each of the K states, i.e. $\mathbf{E} \sim \mathcal{N}(\mathbf{E}; \mathbf{0}, \mathbf{D})$, with $\mathbf{D}_{ik} = \sigma_k^2 \delta_{ik}$. For N distinct time points the overall system may therefore be summarized as

$\mathbf{Y} = \mathbf{X} + \mathbf{E},$

where

$\mathbf{X} = [\mathbf{x}(t_1), \ldots, \mathbf{x}(t_N)] = [\mathbf{x}_1, \ldots, \mathbf{x}_K]^T,$

$\mathbf{Y} = [\mathbf{y}(t_1), \ldots, \mathbf{y}(t_N)] = [\mathbf{y}_1, \ldots, \mathbf{y}_K]^T,$

and $\mathbf{x}_k = [x_k(t_1), \ldots, x_k(t_N)]^T$ is the k'th state sequence and $\mathbf{y}_k = [y_k(t_1), \ldots, y_k(t_N)]^T$ are the observations. Given the observations $\mathbf{Y}$ and the description of the dynamical system (1), the aim is to estimate both state variables $\mathbf{X}$ and parameters $\boldsymbol{\theta}$.

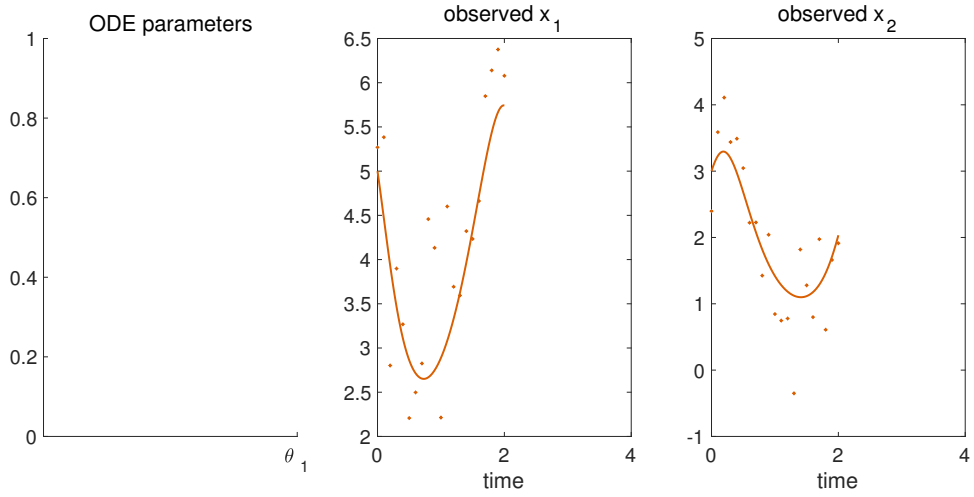
We consider only dynamical systems that are *locally linear* w.r.t. ODE parameters $\boldsymbol{\theta}$ and individual states $\mathbf{x}$. Such ODEs include mass-action kinetics and are given by:

$f_k(\mathbf{x}(t), \theta) = \sum_{i=1} \theta_{k_i} \prod_{j \in \mathcal{M}_{k_i}} x_j \qquad (2),$

with $\mathcal{M}_{k_i} \subseteq \{1, \ldots, K\}$ describing the state variables in each factor of the equation (i.e. the functions are linear in parameters and contain arbitrary large products of monomials of the states).

1.4 Simulate Trajectories

```
[simulation,obs_to_state_relation,fig_handle,plot_handle] = ...
simulate_state_dynamics(simulation,state,symbols,ode,odes_path,time,plot_settings);
```



start timer

```
tic;
```

1.5 Prior on States and State Derivatives

Gradient matching with Gaussian processes assumes a joint Gaussian process prior on states and their derivatives:

$$\begin{pmatrix} \mathbf{X} \\ \dot{\mathbf{X}} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{X} \\ \dot{\mathbf{X}} \end{pmatrix} ; \begin{matrix} \mathbf{0} \\ \mathbf{0} \end{matrix} , \begin{matrix} \mathbf{C}_\phi & \mathbf{C}'_\phi \\ {}'\mathbf{C}_\phi & \mathbf{C}''_\phi \end{matrix} \right) \qquad (3),$$

with

$$\text{cov}(x_k(t), x_k(t)) = C_{\phi_k}(t, t'),$$

$$\text{cov}(\dot{x}_k(t), x_k(t)) = \frac{\partial C_{\phi_k}(t,t')}{\partial t} =: C'_{\phi_k}(t, t'),$$

$$\text{cov}(x_k(t), \dot{x}_k(t)) = \frac{\partial C_{\phi_k}(t,t')}{\partial t'} =: {}'C_{\phi_k}(t, t'),$$

$$\text{cov}(\dot{x}_k(t), \dot{x}_k(t)) = \frac{\partial C_{\phi_k}(t,t')}{\partial t \partial t'} =: C''_{\phi_k}(t, t').$$

1.6 Matching Gradients

Given the joint distribution over states and their derivatives (3) as well as the ODEs (2), we therefore have two expressions for the state derivatives:

$$\dot{\mathbf{X}} = \mathbf{F} + \boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_1 \sim \mathcal{N}(\boldsymbol{\epsilon}_1; \mathbf{0}, \mathbf{I}\gamma),$$

$$\dot{\mathbf{X}} = {}'\mathbf{C}_\phi \mathbf{C}_\phi^{-1} \mathbf{X} + \boldsymbol{\epsilon}_2, \boldsymbol{\epsilon}_2 \sim \mathcal{N}(\boldsymbol{\epsilon}_2; \mathbf{0}, \mathbf{A}),$$

where $\mathbf{F} := \mathbf{f}(\mathbf{X}, \boldsymbol{\theta})$ and $\mathbf{A} := \mathbf{C}''_\phi - {}'\mathbf{C}_\phi \mathbf{C}_\phi^{-1} \mathbf{C}'_\phi$ and γ is the error variance in the ODEs. Note that, in a deterministic system, the output of the ODEs $\mathbf{F}$ should equal the state derivatives $\dot{\mathbf{X}}$. However, in the first equation above we relax this constraint by adding stochasticity to the state derivatives $\dot{\mathbf{X}}$ in order to compensate for a potential model mismatch. The second equation above is obtained by deriving the conditional distribution for $\dot{\mathbf{X}}$ from the joint distribution in equation (3). Equating the two expressions in the equations above we can eliminate the unknown state derivatives

1.7 Rewrite ODEs as Linear Combination in Parameters

Since, according to the mass action dynamics (equation 2), the ODEs are *linear in the parameters* $\boldsymbol{\theta}$ we can rewrite the ODEs in equation (2) as a linear combination in the parameters:

$$\mathbf{B}_{\boldsymbol{\theta}k}\boldsymbol{\theta} + \mathbf{b}_{\boldsymbol{\theta}k} \stackrel{!}{=} \mathbf{f}_k(\mathbf{X}, \boldsymbol{\theta}) \qquad (5),$$

where matrices $\mathbf{B}_{\boldsymbol{\theta}k}$ and $\mathbf{b}_{\boldsymbol{\theta}k}$ are defined such that the ODEs $\mathbf{f}_k(\mathbf{X}, \boldsymbol{\theta})$ are expressed as a linear combination in $\boldsymbol{\theta}$.

```
[ode_param.lin_comb.B,ode_param.lin_comb.b] = ...
rewrite_odes_as_linear_combination_in_parameters(ode,symbols);
```

1.8 Posterior over ODE Parameters

Inserting (5) into (4) and solving for $\boldsymbol{\theta}$ yields:

$$\boldsymbol{\theta} = \mathbf{B}_{\boldsymbol{\theta}}^{+}\left('\mathbf{C}_{\boldsymbol{\phi}}\mathbf{C}_{\boldsymbol{\phi}}^{-1}\mathbf{X} - \mathbf{b}_{\boldsymbol{\theta}} + \boldsymbol{\epsilon}_0\right),$$

where $\mathbf{B}_{\boldsymbol{\theta}}^{+}$ denotes the pseudo-inverse of $\mathbf{B}_{\boldsymbol{\theta}}$. Since $\mathbf{C}_{\boldsymbol{\phi}}$ is block diagonal we can rewrite the expression above as:

$$\begin{aligned}\boldsymbol{\theta} &= \left(\mathbf{B}_{\boldsymbol{\theta}}^T\mathbf{B}_{\boldsymbol{\theta}}\right)^{-1}\ \mathbf{B}_{\boldsymbol{\theta}}^T\left(\textstyle\sum_k {}'\mathbf{C}_{\boldsymbol{\phi}_k}\mathbf{C}_{\boldsymbol{\phi}_k}^{-1}\mathbf{X}_k - \mathbf{b}_{\boldsymbol{\theta}k} + \boldsymbol{\epsilon}_0^{(k)}\right)\\ &= \left(\mathbf{B}_{\boldsymbol{\theta}}^T\mathbf{B}_{\boldsymbol{\theta}}\right)^{-1}\left(\textstyle\sum_k \mathbf{B}_{\boldsymbol{\theta}k}^T\left('\mathbf{C}_{\boldsymbol{\phi}_k}\mathbf{C}_{\boldsymbol{\phi}_k}^{-1}\mathbf{X}_k - \mathbf{b}_{\boldsymbol{\theta}k} + \boldsymbol{\epsilon}_0^{(k)}\right)\right),\end{aligned}$$

where we subsitute the Moore-Penrose inverse for the pseudo-inverse (i.e. $\mathbf{B}_{\boldsymbol{\theta}}^{+} := \left(\mathbf{B}_{\boldsymbol{\theta}}^T\mathbf{B}_{\boldsymbol{\theta}}\right)^{-1}\mathbf{B}_{\boldsymbol{\theta}}^T$). We can therefore derive the posterior distribution over ODE parameters:

$$p(\boldsymbol{\theta} \mid \mathbf{X}, \boldsymbol{\phi}, \gamma) = \mathcal{N}\left(\boldsymbol{\theta}; \left(\mathbf{B}_{\boldsymbol{\theta}}^T\mathbf{B}_{\boldsymbol{\theta}}\right)^{-1}\left(\textstyle\sum_k \mathbf{B}_{\boldsymbol{\theta}k}^T\ \left('\mathbf{C}_{\boldsymbol{\phi}k}\mathbf{C}_{\boldsymbol{\phi}k}^{-1}\mathbf{X}_k - \mathbf{b}_{\boldsymbol{\theta}k}\right)\right),\ \mathbf{B}_{\boldsymbol{\theta}}^{+}\ (\mathbf{A} + \mathbf{I}\gamma)\ \mathbf{B}_{\boldsymbol{\theta}}^{+T}\right) \qquad (6).$$

1.9 Rewrite ODEs as Linear Combination in Individual States

Since, according to the mass action dynamics (equation 2), the ODEs are *linear in the individual state* $\mathbf{x}_u$ we can rewrite the ODE $\mathbf{f}_k(\mathbf{X}, \boldsymbol{\theta})$ as a linear combination in the individual state $\mathbf{x}_u$:

$$\mathbf{R}_{uk}\mathbf{x}_u + \mathbf{r}_{uk} \stackrel{!}{=} \mathbf{f}_k(\mathbf{X}, \boldsymbol{\theta}),$$

where matrices $\mathbf{R}_{uk}$ and $\mathbf{r}_{uk}$ are defined such that the ODE $\mathbf{f}_k(\mathbf{X}, \boldsymbol{\theta})$ is expressed as a linear combination in the individual state $\mathbf{x}_u$.

```
[state.lin_comb.R,state.lin_comb.r] = ...
rewrite_odes_as_linear_combination_in_ind_states(ode,symbols,coupling_idx.states);
```

1.10 Posterior over Individual States

Given the linear combination of the ODEs w.r.t. an individual state, we define the matrices $\mathbf{B}_u$ and $\mathbf{b}_u$ such that the expression $\mathbf{f}(\mathbf{X}, \boldsymbol{\theta}) - {}'\mathbf{C}_{\boldsymbol{\phi}}\mathbf{C}_{\boldsymbol{\phi}}^{-1}\mathbf{X}$ is rewritten as a linear combination in an individual state $\mathbf{x}_u$:

$$\mathbf{B}_u\mathbf{x}_u + \mathbf{b}_u \stackrel{!}{=} \mathbf{f}(\mathbf{X}, \boldsymbol{\theta}) - {}'\mathbf{C}_{\boldsymbol{\phi}}\mathbf{C}_{\boldsymbol{\phi}}^{-1}\mathbf{X} \qquad (7).$$

Inserting (7) into (4) and solving for $\mathbf{x}_u$ yields:

$$\mathbf{x}_u = \mathbf{B}_u^{+}\left(\boldsymbol{\epsilon}_0 - \mathbf{b}_u\right),$$

where $\mathbf{B}_u^{+}$ denotes the pseudo-inverse of $\mathbf{B}_u$. Since $\mathbf{C}_{\boldsymbol{\phi}}$ is block diagonal we can rewrite the expression above as:

$$\begin{aligned}\mathbf{x}_u &= \left(\mathbf{B}_u\mathbf{B}_u^T\right)^{-1}\mathbf{B}_u^T\textstyle\sum_k\left(\boldsymbol{\epsilon}_0^{(k)} - \mathbf{b}_{uk}\right)\\ &= \left(\mathbf{B}_u\mathbf{B}_u^T\right)^{-1}\textstyle\sum_k\mathbf{B}_{uk}^T\left(\boldsymbol{\epsilon}_0^{(k)} - \mathbf{b}_{uk}\right),\end{aligned}$$

where we substitute the Moore-Penrose inverse for the pseudo-inverse (i.e. $\mathbf{B}_{\boldsymbol{\theta}}^{+} := \left(\mathbf{B}_{\boldsymbol{\theta}}^T\mathbf{B}_{\boldsymbol{\theta}}\right)^{-1}\mathbf{B}_{\boldsymbol{\theta}}^T$). We can therefore derive the posterior distribution over an individual state $\mathbf{x}_u$:

$$p(\mathbf{x}_u \mid \mathbf{X}_{-u}, \boldsymbol{\phi}, \gamma) = \mathcal{N}\left(\mathbf{x}_u; \left(\mathbf{B}_u\mathbf{B}_u^T\right)^{-1}\left(-\textstyle\sum_k\mathbf{B}_{uk}^T\mathbf{b}_{uk}\right),\ \mathbf{B}_u^{+}\left(\mathbf{A} + \mathbf{I}\gamma\right)\mathbf{B}_u^{+T}\right) \qquad (8),$$

with $\mathbf{X}_{-u}$ denoting the set of all states except state $\mathbf{x}_u$.

1.11 Mean-field Variational Inference

To infer the parameters $\boldsymbol{\theta}$, we want to find the maximum a posteriori estimate (MAP):

$$\begin{aligned}&\boldsymbol{\theta}^* := \arg\max_{\boldsymbol{\theta}}\ \ln p(\boldsymbol{\theta} \mid \mathbf{Y}, \boldsymbol{\phi}, \gamma, \sigma)\\ &= \arg\max_{\boldsymbol{\theta}}\ \ln\textstyle\int p(\boldsymbol{\theta}, \mathbf{X} \mid \mathbf{Y}, \boldsymbol{\phi}, \gamma, \boldsymbol{\sigma})d\mathbf{X}\\ &= \arg\max_{\boldsymbol{\theta}}\ \ln\textstyle\int p(\boldsymbol{\theta} \mid \mathbf{X}, \boldsymbol{\phi}, \gamma)p(\mathbf{X} \mid \mathbf{Y}, \boldsymbol{\phi}, \boldsymbol{\sigma})d\mathbf{X} \qquad (9).\end{aligned}$$

However, the integral above is intractable due to the strong couplings induced by the nonlinear ODEs $\mathbf{f}$ which appear in the term $p(\boldsymbol{\theta} \mid \mathbf{X}, \boldsymbol{\phi}, \gamma)$.

We use mean-field variational inference to establish variational lower bounds that are analytically tractable by decoupling state variables from the ODE parameters as well as decoupling the state variables from each other. Note that, since the ODEs described by equation (2) are *locally linear*, both conditional distributions $p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{Y}, \boldsymbol{\phi}, \gamma, \boldsymbol{\sigma})$ (equation (6)) and $p(\mathbf{x}_u \mid \boldsymbol{\theta}, \mathbf{X}_{-u}, \mathbf{Y}, \boldsymbol{\phi}, \gamma, \boldsymbol{\sigma})$ (equation (8)) are analytically tractable and Gaussian distributed as mentioned previously. The decoupling is induced by designing a variational distribution $Q(\boldsymbol{\theta}, \mathbf{X})$ which is restricted to the family of factorial distributions:

$$\mathcal{Q} := \left\{Q : Q(\boldsymbol{\theta}, \mathbf{X}) = q(\boldsymbol{\theta})\textstyle\prod_u q(\mathbf{x}_u)\right\}.$$

The particular form of $q(\boldsymbol{\theta})$ and $q(\mathbf{x}_u)$ are designed to be Gaussian distributed which places them in the same family as the true full conditional distributions. To find the optimal factorial distribution we minimize the Kullback-Leibler divergence between the variational and the true posterior distribution:

$$\hat{Q} := \arg\min_{Q(\boldsymbol{\theta},\mathbf{X})\in\mathcal{Q}}\mathrm{KL}\left[Q(\boldsymbol{\theta}, \mathbf{X})\ ||\ p(\boldsymbol{\theta}, \mathbf{X} \mid \mathbf{Y}, \boldsymbol{\phi}, \gamma, \sigma)\right] \qquad (10),$$

where $\hat{Q}$ is the proxy distribution. The proxy distribution that minimizes the KL-divergence (10) depends on the true full conditionals and is given by:

$$\begin{aligned}&\hat{q}(\boldsymbol{\theta}) \propto \exp\left(\ E_{Q_{-\boldsymbol{\theta}}}\ln p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{Y}, \boldsymbol{\phi}, \gamma, \sigma)\ \right) \qquad (11)\\ &\hat{q}(\mathbf{x}_u) \propto \exp\left(\ E_{Q_{-u}}\ln p(\mathbf{x}_u \mid \boldsymbol{\theta}, \mathbf{X}_{-u}, \mathbf{Y}, \boldsymbol{\phi}, \gamma, \boldsymbol{\sigma})\ \right) \qquad (12).\end{aligned}$$

1.12 Fitting Observations of State Trajectories

We fit the observations of state trajectories by standard GP regression. The data-informed distribution $p(\mathbf{X} \mid \mathbf{Y}, \boldsymbol{\phi}, \sigma)$ in euqation (9) can be determined analytically using Gaussian process regression with the GP prior $p(\mathbf{X} \mid \boldsymbol{\phi}) = \prod_k \mathcal{N}(\mathbf{x}_k; \mathbf{0}, \mathbf{C}_{\boldsymbol{\phi}_k})$:

$p(\mathbf{X} \mid \mathbf{Y}, \boldsymbol{\phi}, \gamma) = \prod_k \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_k(\mathbf{y}_k), \sigma_\mathbf{k})$,

where $\boldsymbol{\mu}_k(\mathbf{y}_k) := \boldsymbol{\sigma}_k^{-2} \left(\boldsymbol{\sigma}_k^{-2}\mathbf{I} + \mathbf{C}_{\boldsymbol{\phi}_k}^{-1} \right)^{-1} \mathbf{y}_k$ and $\boldsymbol{\sigma}_k^{-1} := \boldsymbol{\sigma}_k^{-2}\mathbf{I} + \mathbf{C}_{\phi_\mathbf{k}{}^{-1}}$.

```
[mu,inv_sigma] = fitting_state_observations(inv_C,obs_to_state_relation,...
simulation,symbols);
```

1.13 Coordinate Ascent Variational Gradient Matching

We minimize the KL-divergence in equation (10) by coordinate descent (where each step is analytically tractable) by iterating between determining the proxy for the distribution over ODE parameters $\hat{q}(\boldsymbol{\theta})$ and the proxies for the distribution over individual states $\hat{q}(\mathbf{x}_u)$.

Initialize the state estimation by the GP regression posterior

```
state.proxy.mean = array2table([time.est',mu],...
'VariableNames',['time',symbols.state_string]);
```

Coordinate ascent

```
for i = 1:opt_settings.coord_ascent_numb_iter
```

1.13.1 Proxy for ODE parameters

Expanding the proxy distribution in equation (11) for $\boldsymbol{\theta}$ yields:

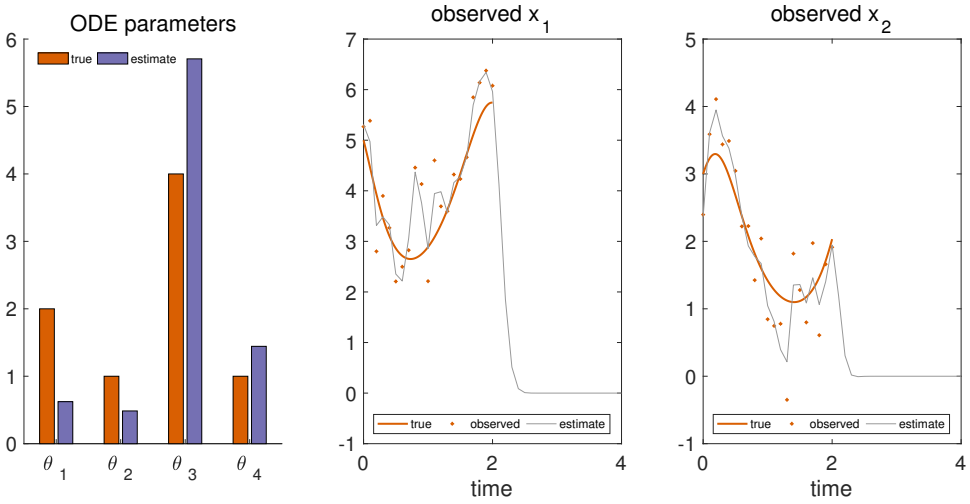
$\hat{q}(\boldsymbol{\theta}) \propto \exp\left(E_{Q_{-\boldsymbol{\theta}}} \ln p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{Y}, \boldsymbol{\phi}, \gamma, \boldsymbol{\sigma}) \right)$
$= \exp\left(E_{Q_{-\boldsymbol{\theta}}} \ln \mathcal{N}\left(\boldsymbol{\theta}; \left(\mathbf{B}_{\boldsymbol{\theta}}^T \mathbf{B}_{\boldsymbol{\theta}}\right)^{-1} \left(\sum_k \mathbf{B}_{\boldsymbol{\theta}k}^T \left({}'\mathbf{C}_{\boldsymbol{\phi}k} \mathbf{C}_{\boldsymbol{\phi}k}^{-1} \mathbf{X}_k - \mathbf{b}_{\boldsymbol{\theta}k} \right) \right), \mathbf{B}_{\boldsymbol{\theta}}^{+} \left(\mathbf{A} + \mathbf{I}\gamma \right) \mathbf{B}_{\boldsymbol{\theta}}^{+T} \right) \right)$,

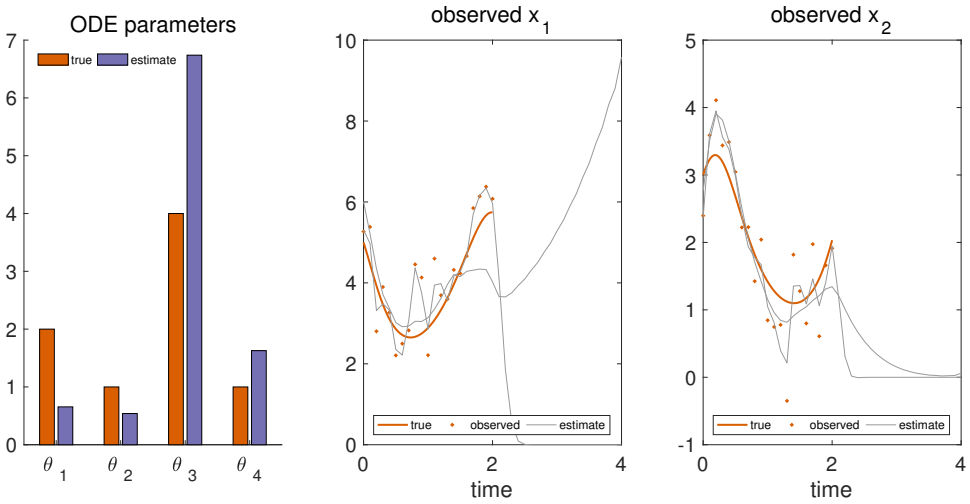
where we substitute $p(\boldsymbol{\theta} \mid \mathbf{X}, \boldsymbol{\phi}, \gamma)$ with its density given in equation (6).

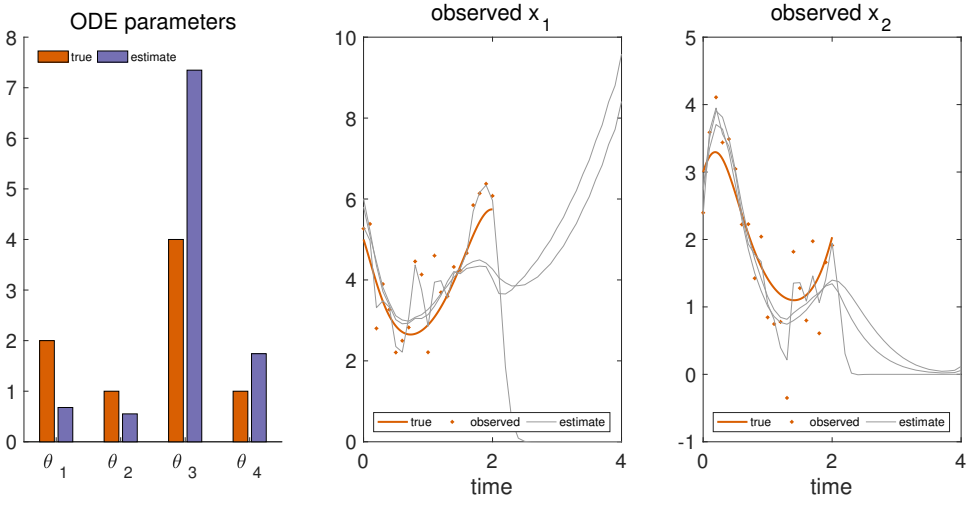
```
[param_proxy_mean,param_proxy_inv_cov] = ...
proxy_for_ode_parameters(state.proxy.mean{:,symbols.state_string},...
dC_times_invC,ode_param.lin_comb,symbols,A_plus_gamma_inv,opt_settings);
```

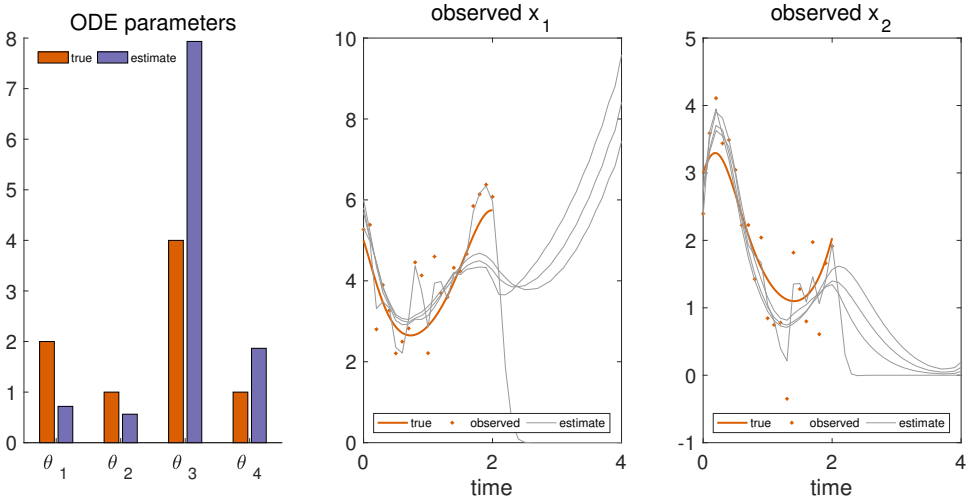
Intermediate results

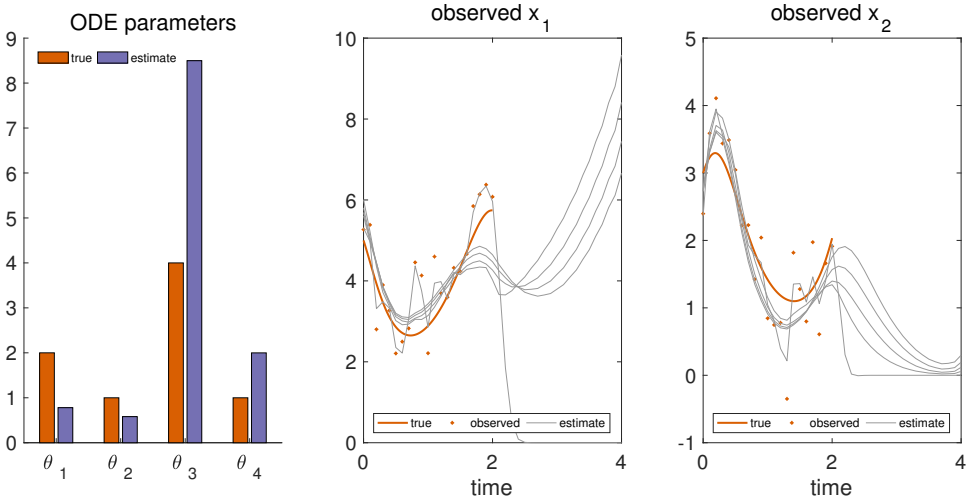
```
if i==1 || ~mod(i,1)
    plot_results(fig_handle,state.proxy,simulation,param_proxy_mean,...
    plot_handle,symbols,plot_settings,'not_final');
end
```

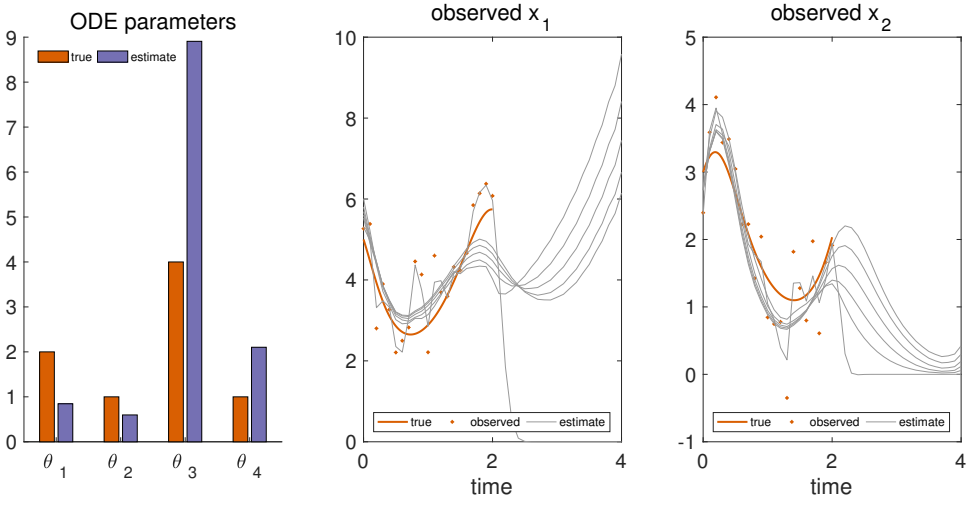

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

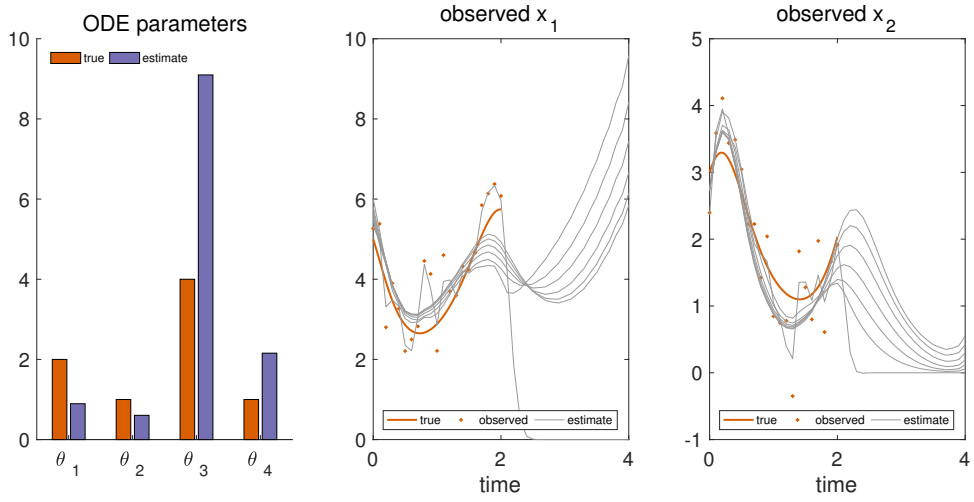

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

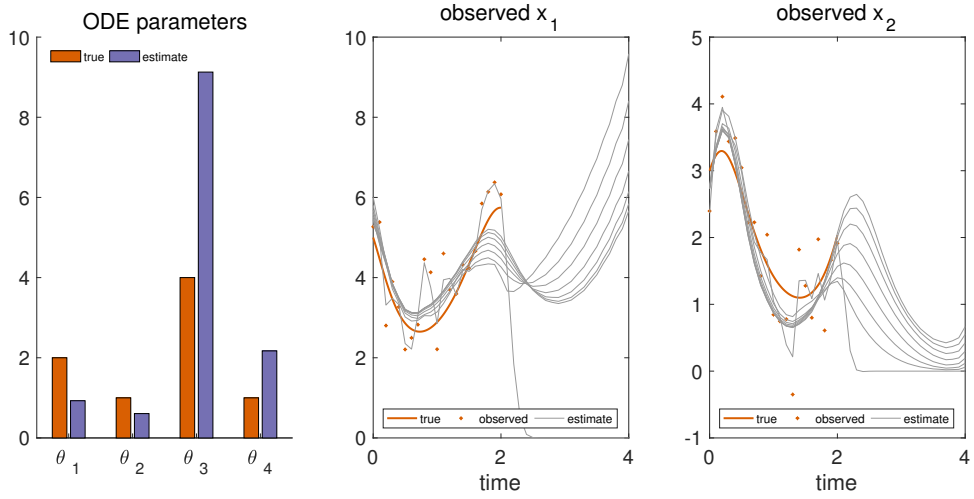

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

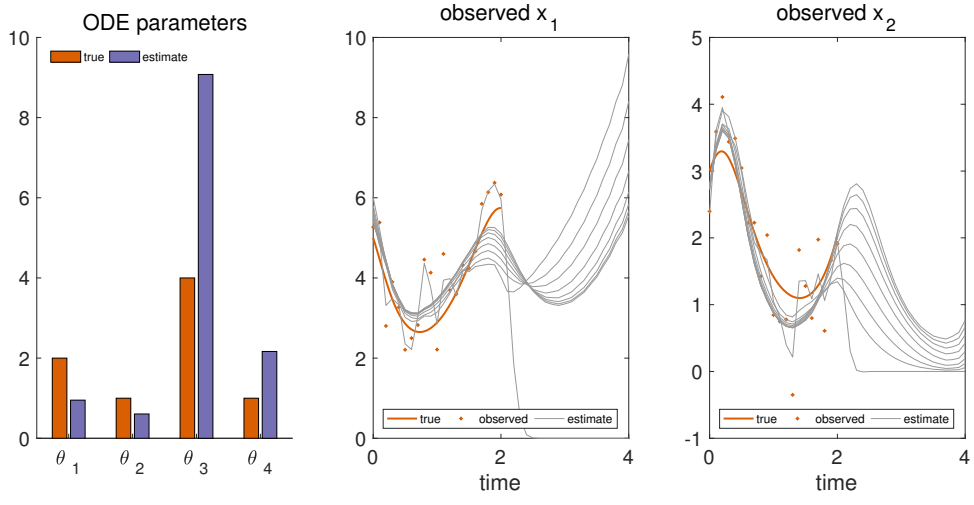

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

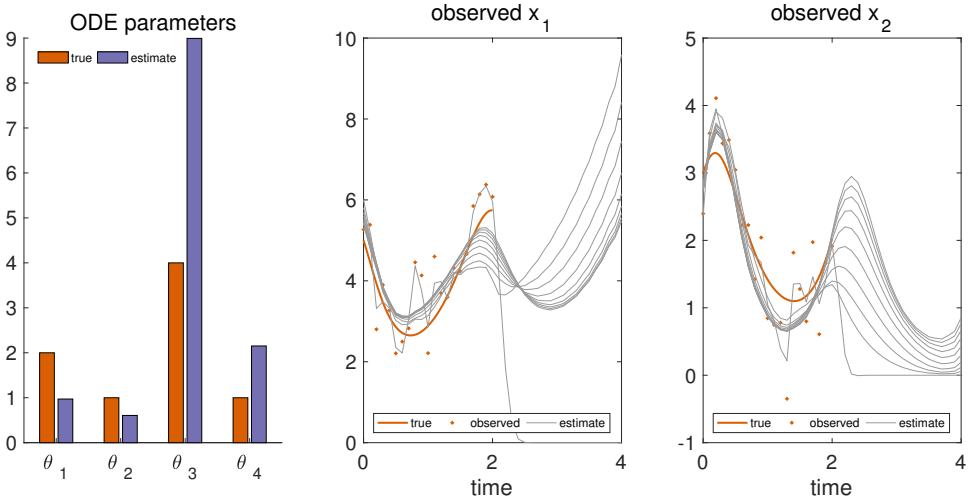

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

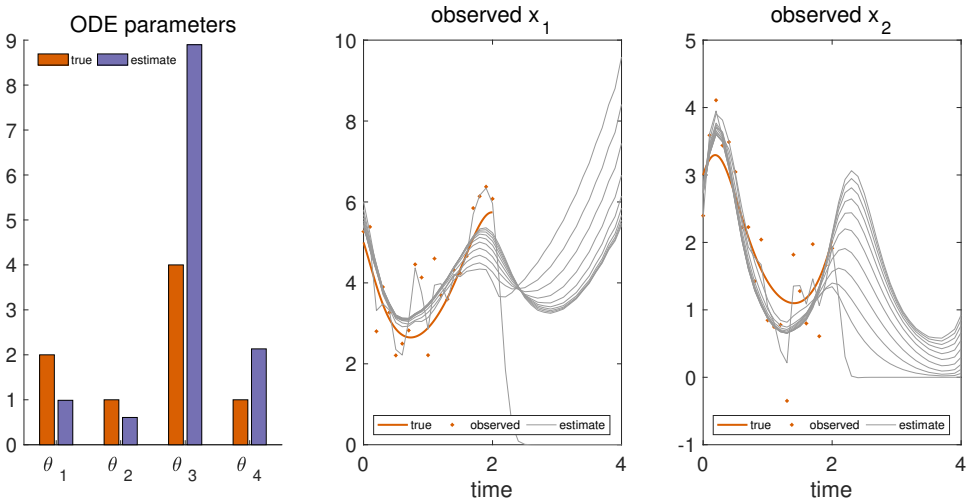

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

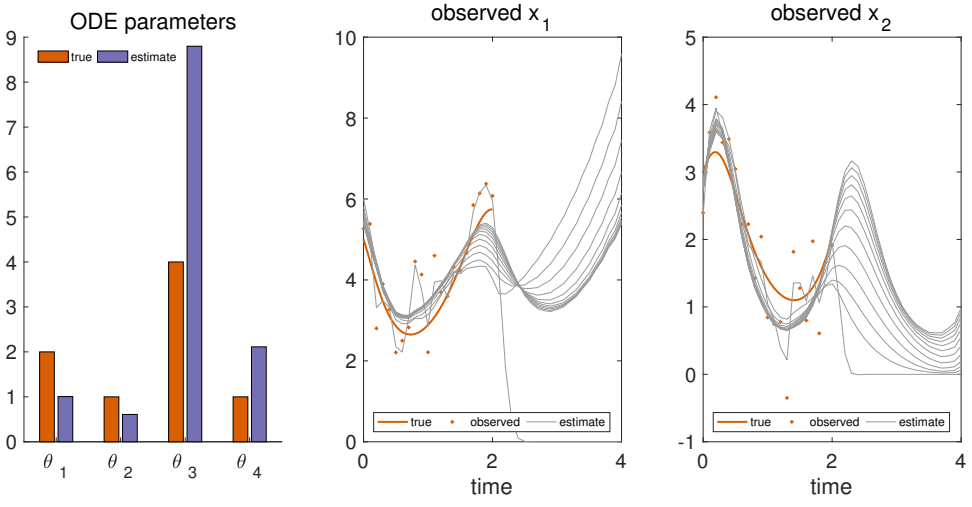

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

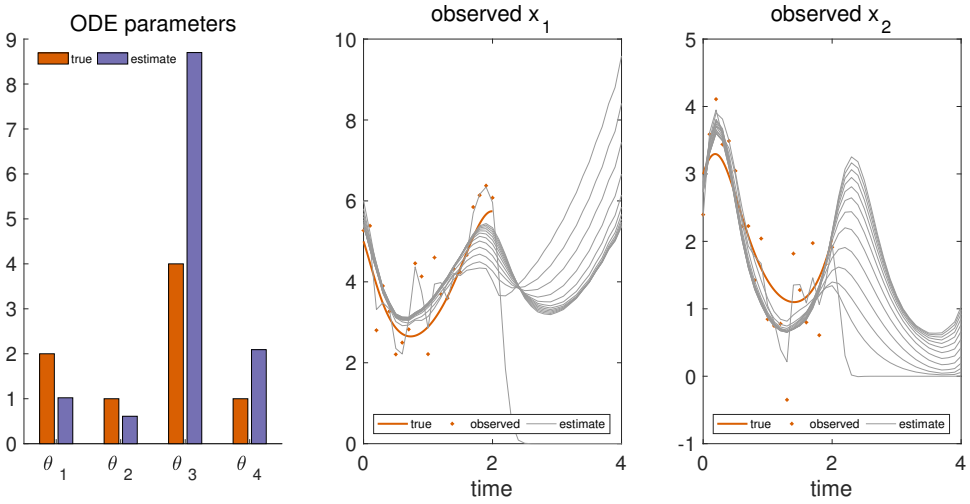

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

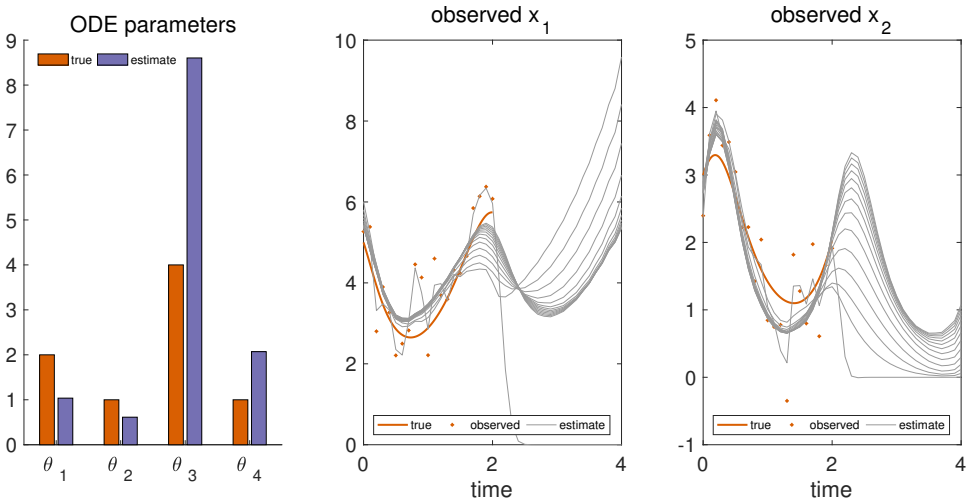

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

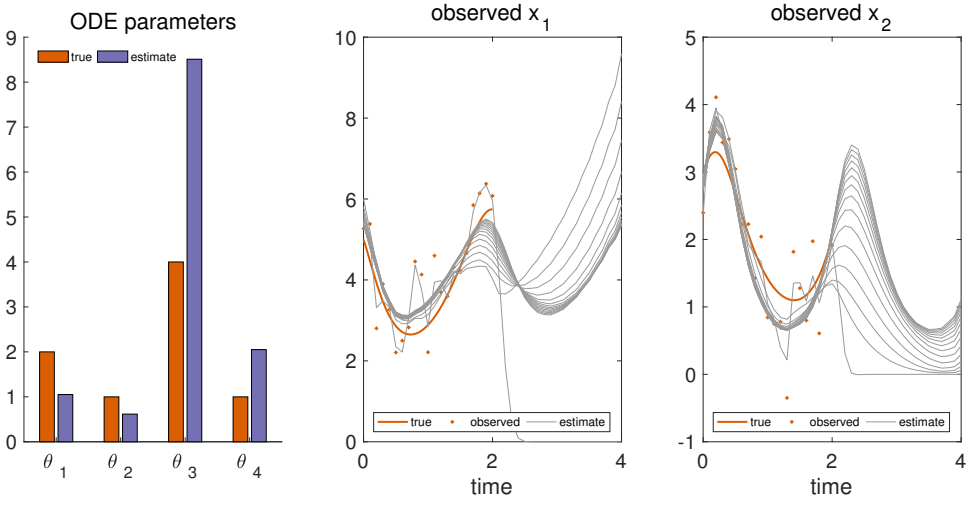

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

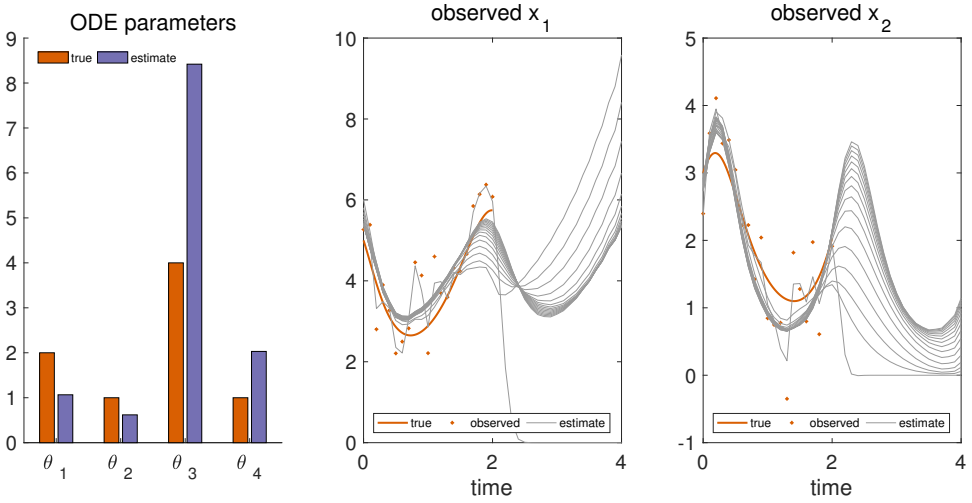

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

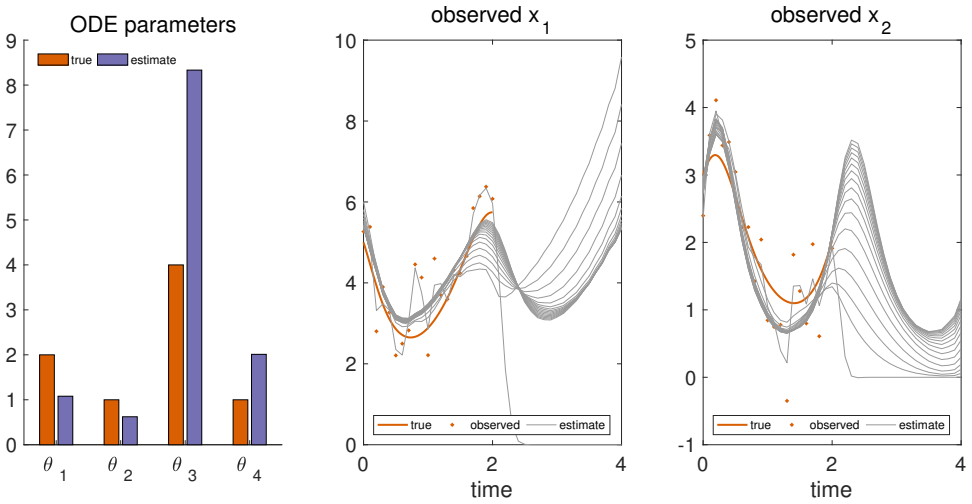

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

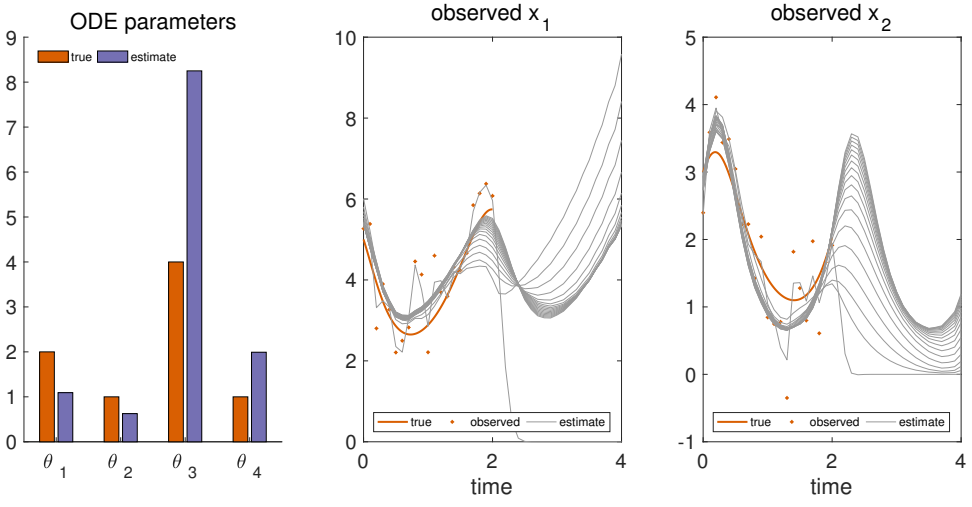

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

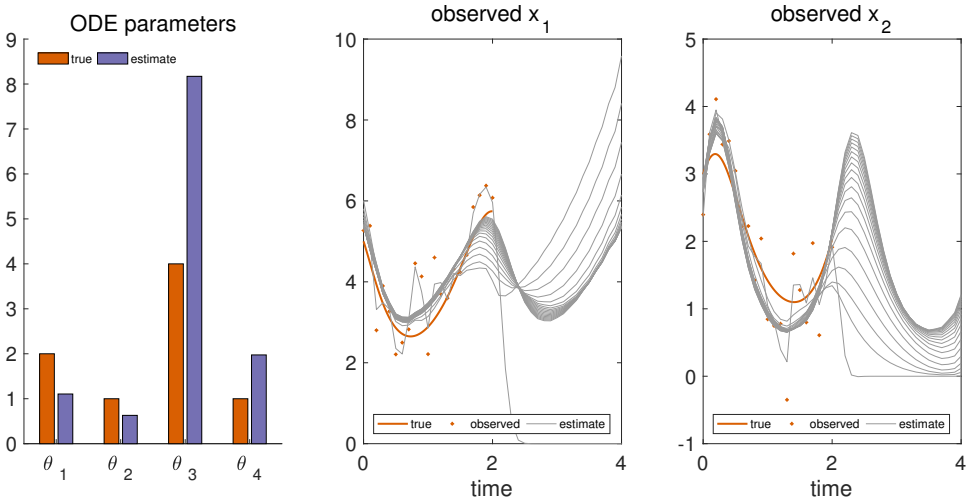

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

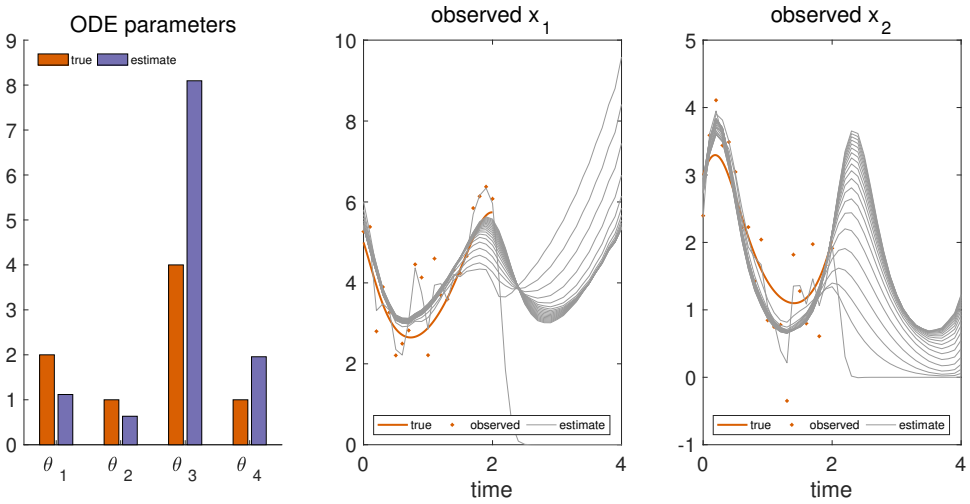

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

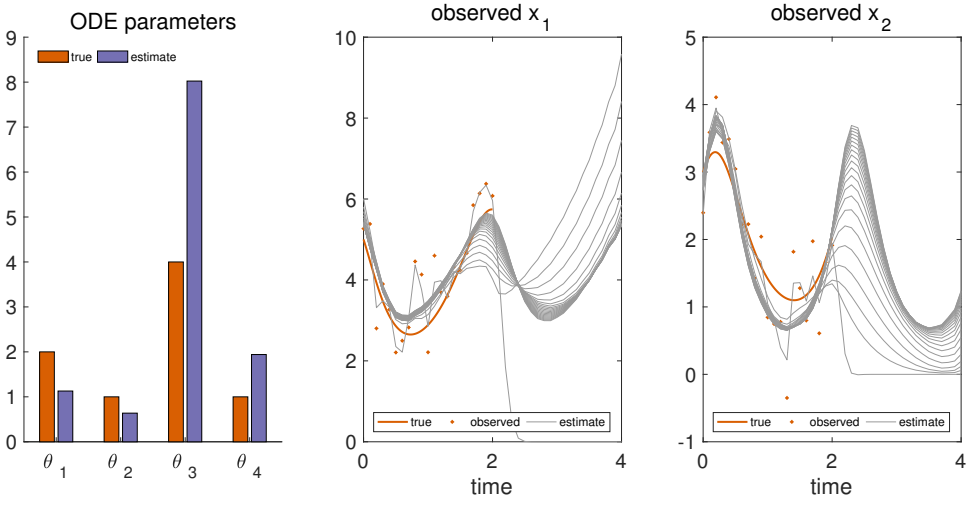

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

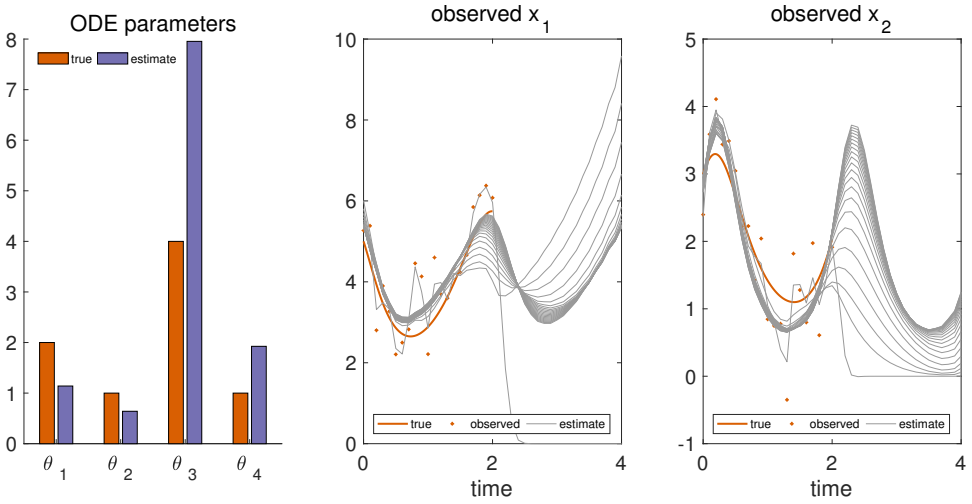

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

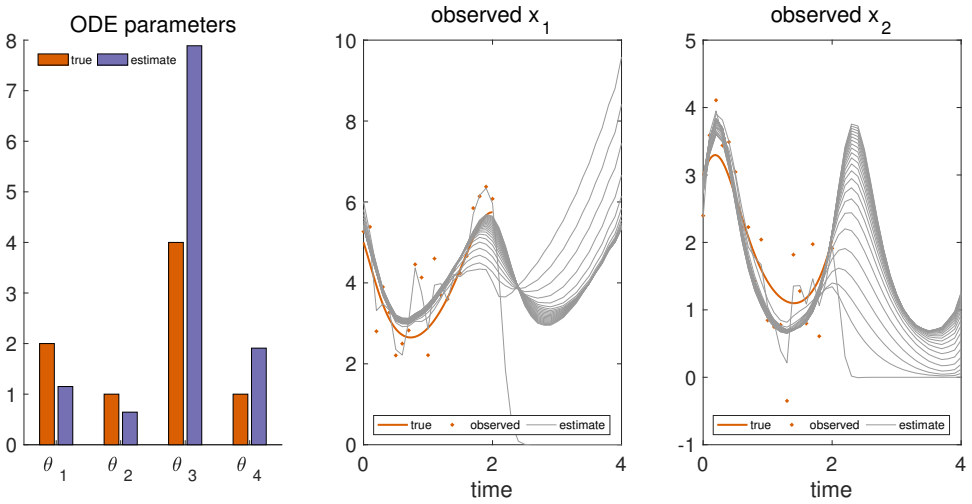

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

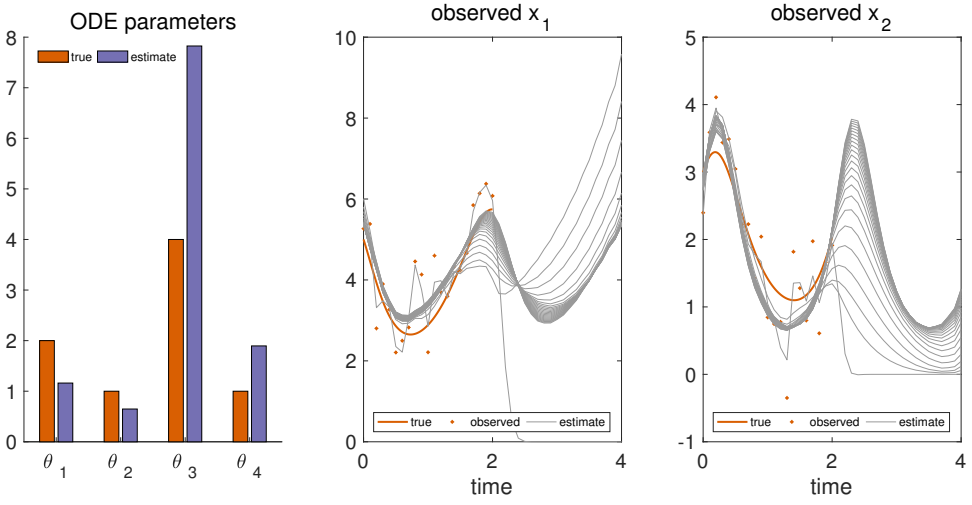

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

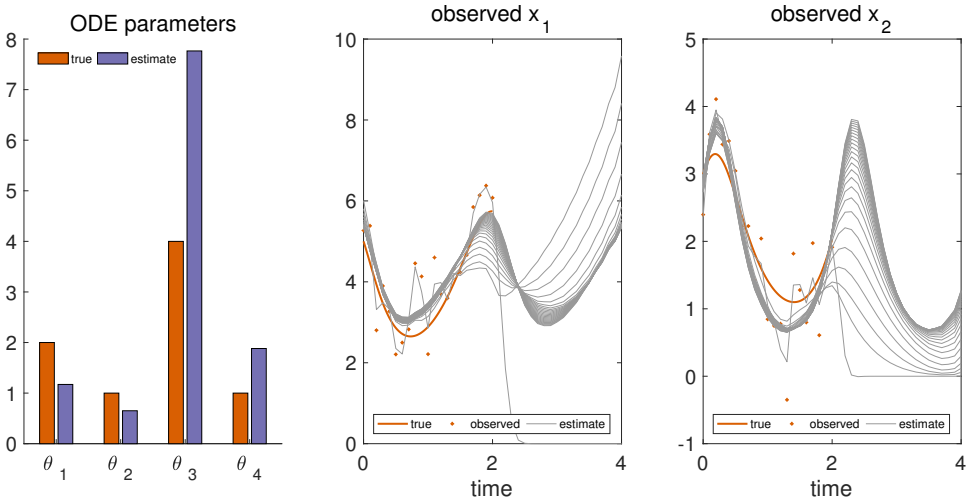

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

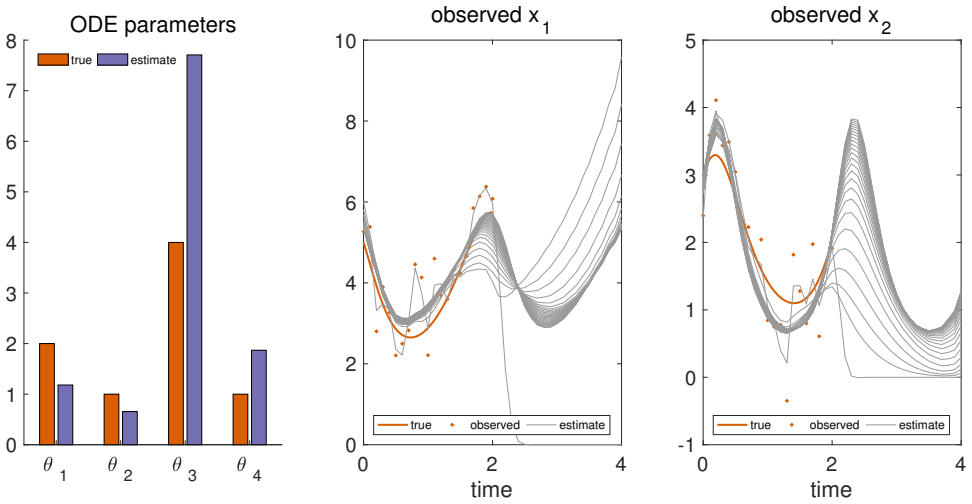

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

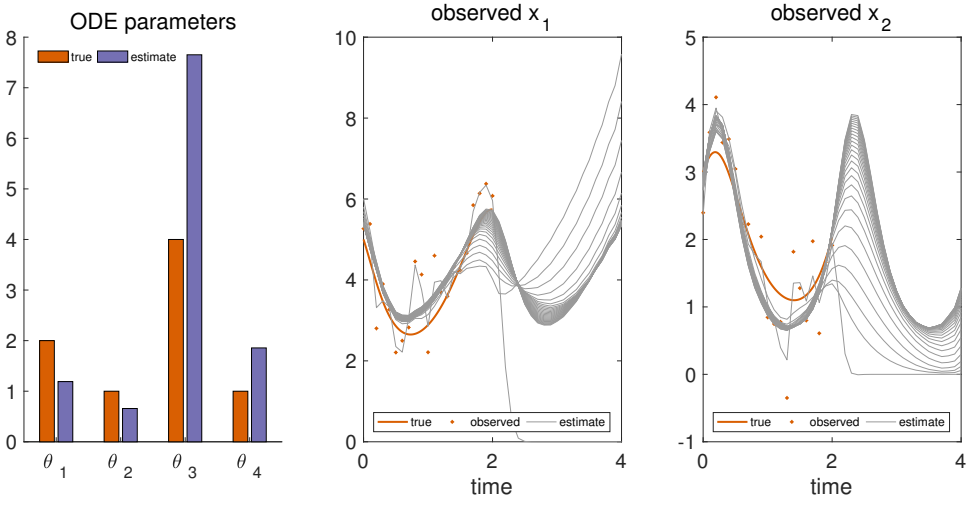

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

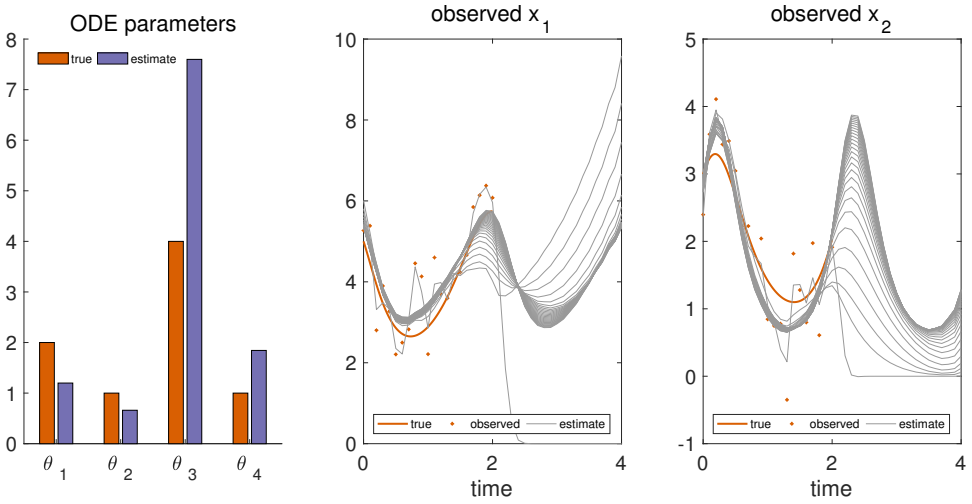

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

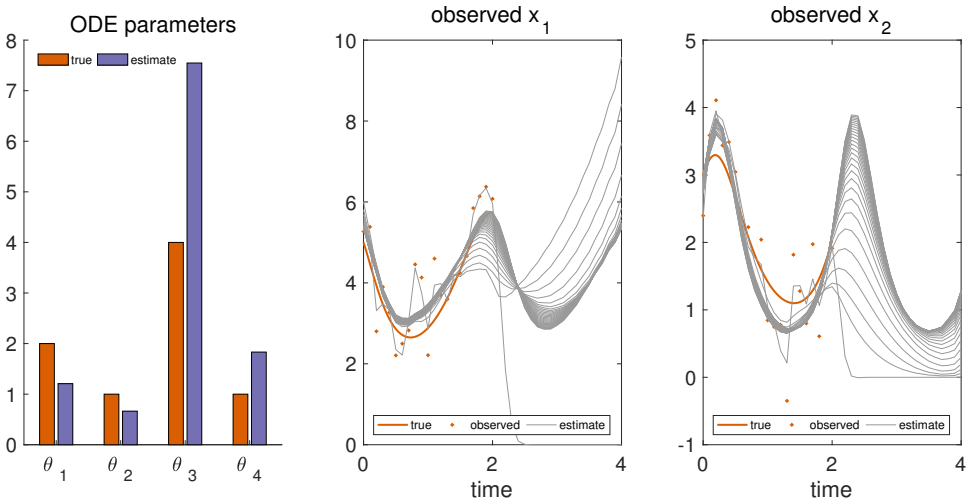

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

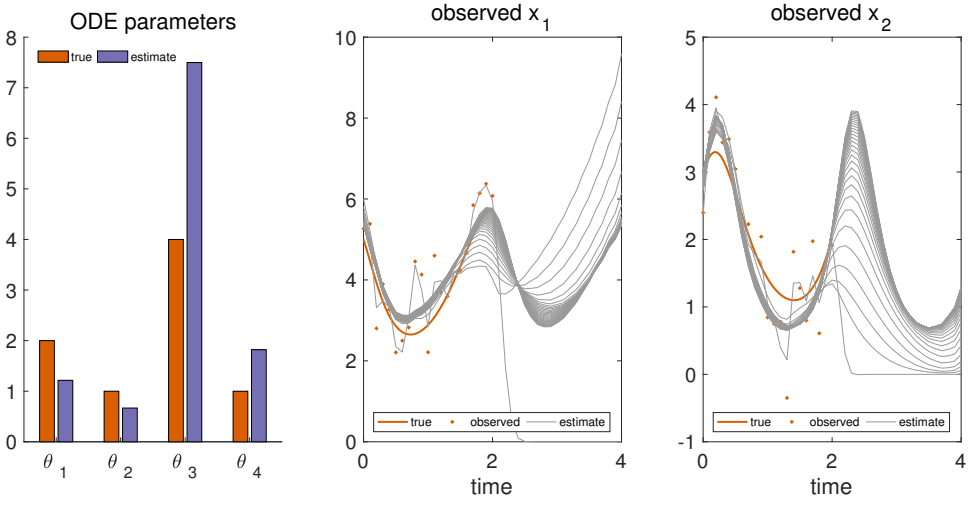

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

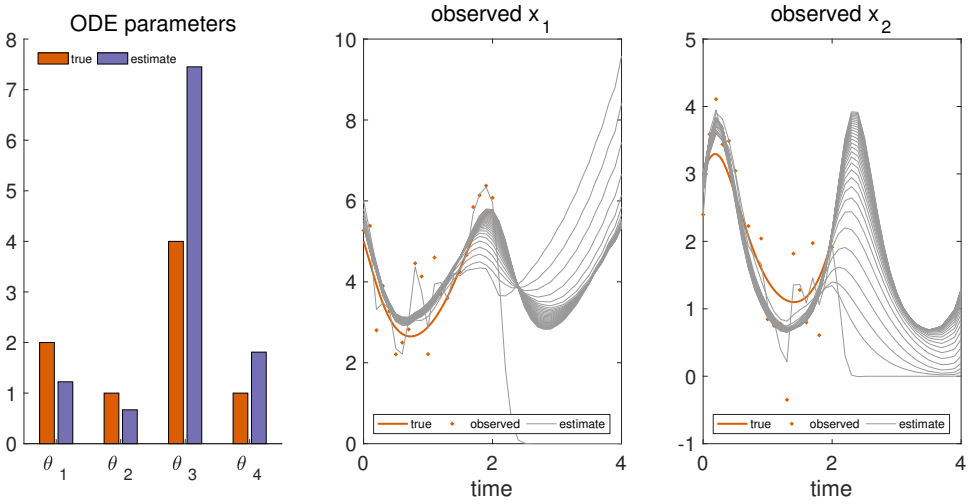

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

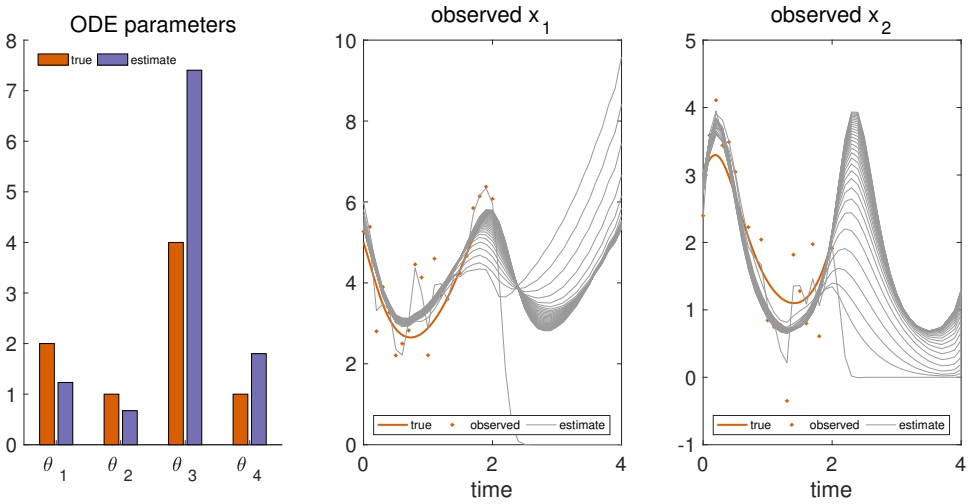

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

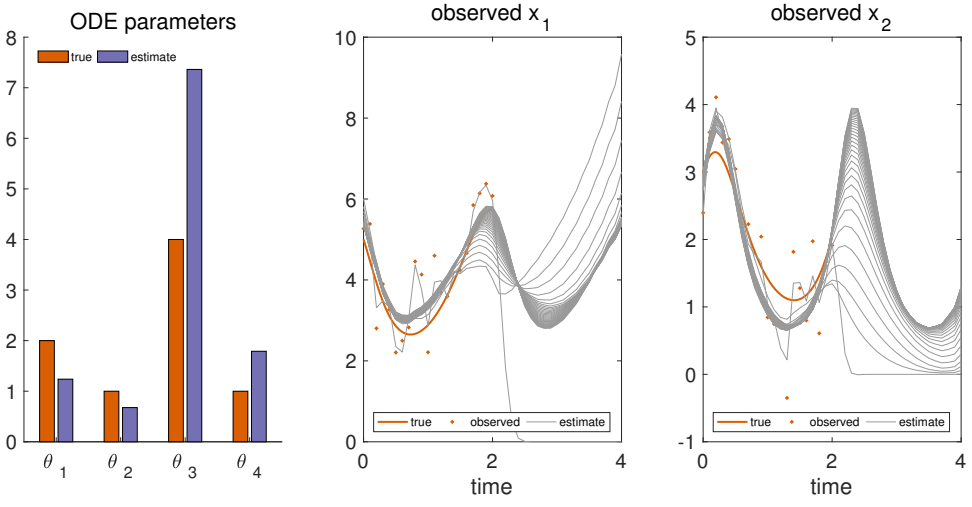

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

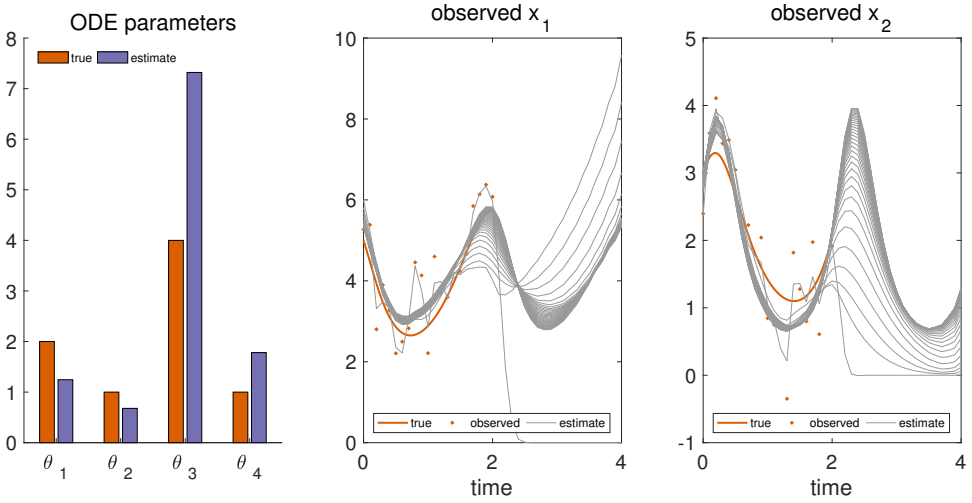

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

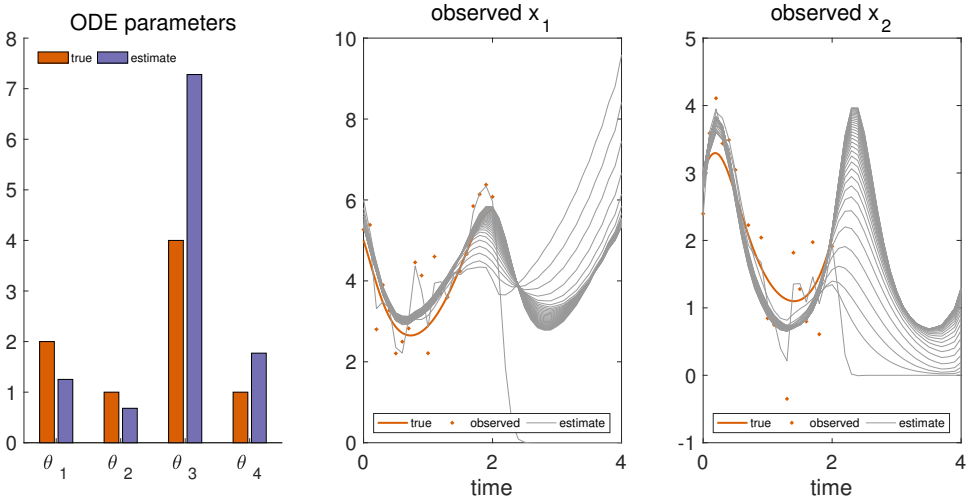

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

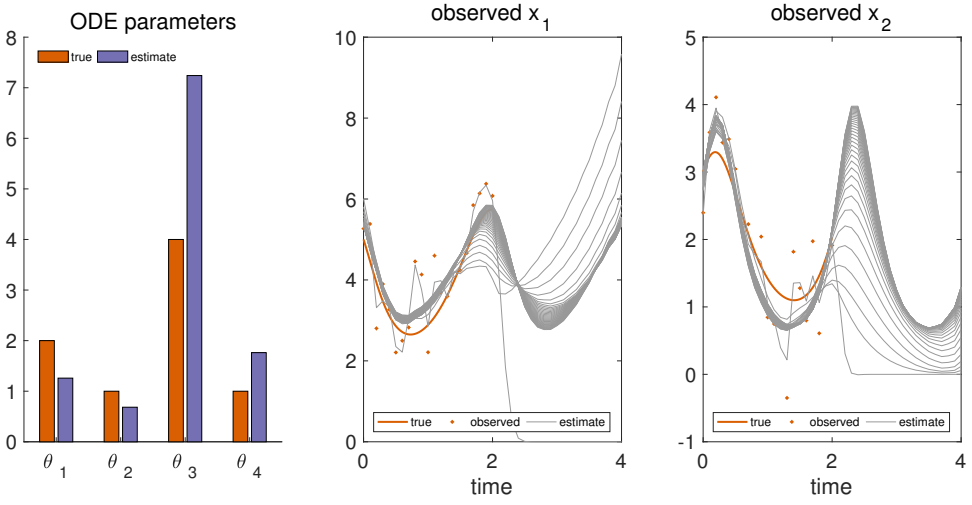

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

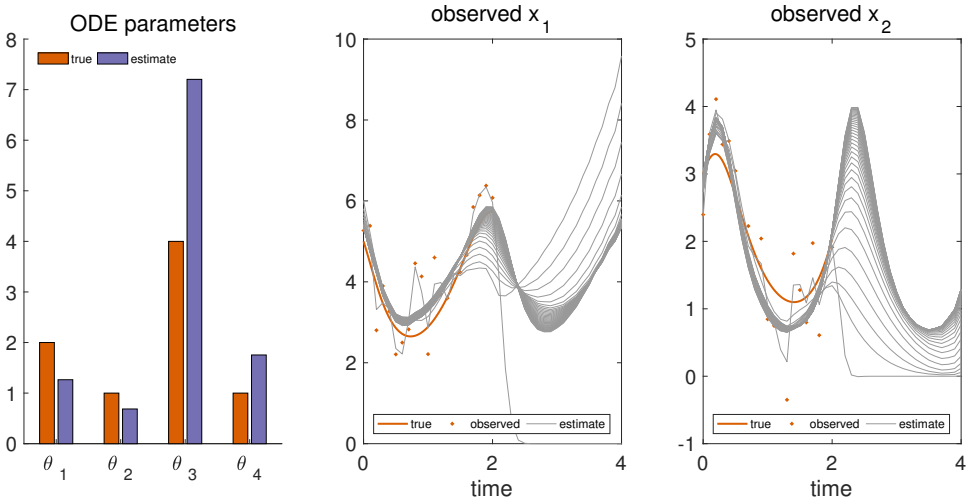

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

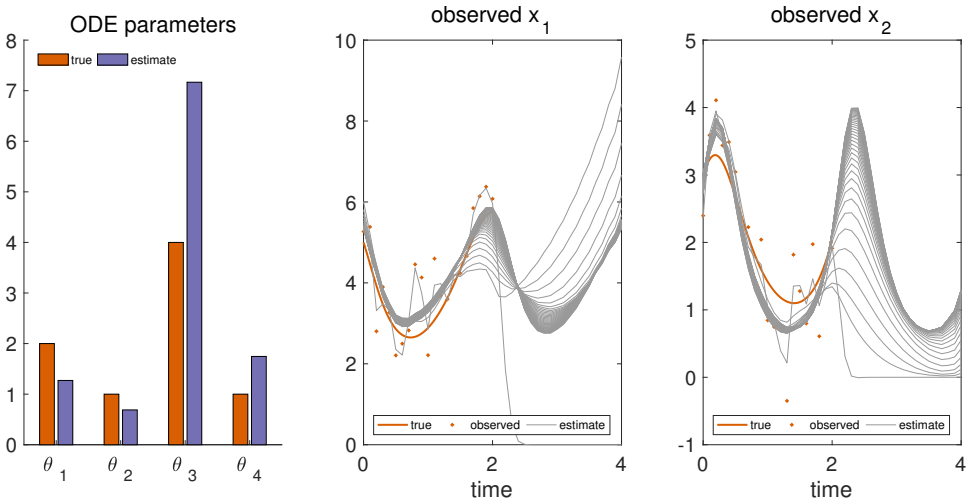

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

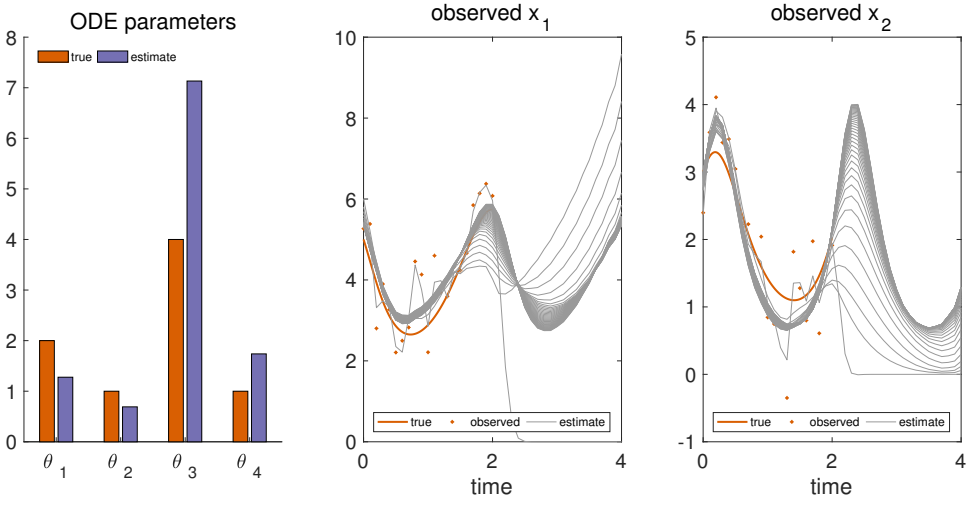

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time

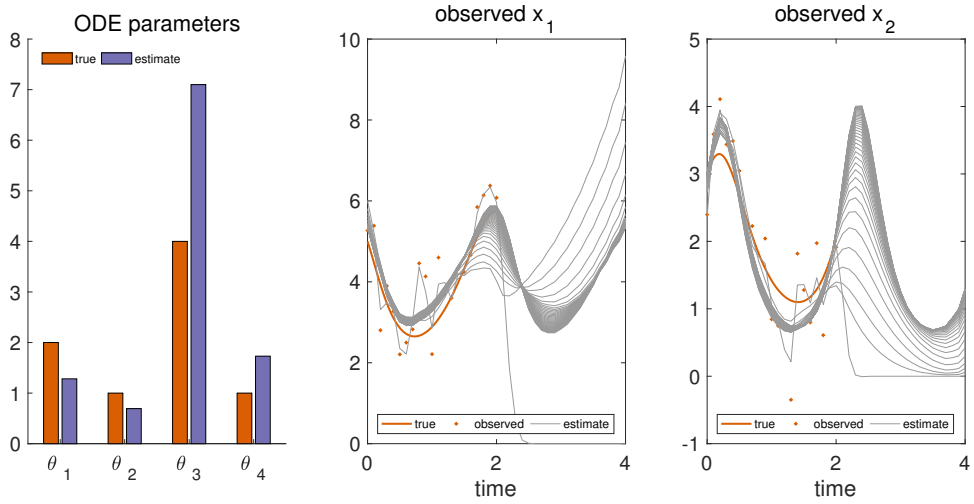

ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time


ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time


ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time


ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time


ODE parameters
true
estimate
θ 1
θ 2
θ 3
θ 4
observed x 1
observed x 2
true
observed
estimate
time



1.13.2 Proxy for individual states

Expanding the proxy distribution in equation (12) over the individual state $\mathbf{x}_u$:

$$\hat{q}(\mathbf{x}_u) \overset{(a)}{\propto} \exp\left(E_{Q_{-u}} \ln(p(\mathbf{x}_u \mid \boldsymbol{\theta}, \mathbf{X}_{-u}, \boldsymbol{\phi}, \gamma) p(\mathbf{x}_u \mid \mathbf{Y}, \boldsymbol{\phi}, \boldsymbol{\sigma})) \right)$$
$$\overset{(b)}{=} \exp\left(E_{Q_{-u}} \ln \mathcal{N}\left(\mathbf{x}_u; -\mathbf{B}_u^{+}\mathbf{b}_u,\ \mathbf{B}_u^{+}\left(\mathbf{A} + \mathbf{I}\gamma\right)\mathbf{B}_u^{+T}\right) + E_{Q_{-u}} \ln \mathcal{N}\left(\mathbf{x}_u; \boldsymbol{\mu}_u(\mathbf{Y}), \boldsymbol{\Sigma}_u\right) \right)$$
$$= \exp\left(E_{Q_{-u}} \ln \mathcal{N}\left(\mathbf{x}_u; -\mathbf{B}_u^{+}\mathbf{b}_u,\ \mathbf{B}_u^{+}\left(\mathbf{A} + \mathbf{I}\gamma\right)\mathbf{B}_u^{+T}\right) + E_{Q_{-u}} \ln \mathcal{N}\left(\mathbf{x}_u; \boldsymbol{\mu}_u(\mathbf{Y}), \boldsymbol{\sigma}_u\right) \right).$$

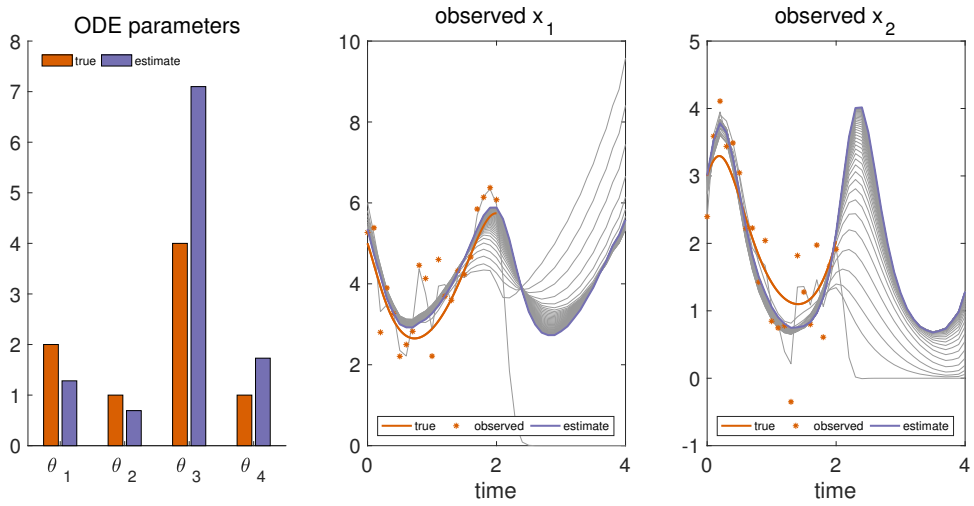
In (a) we decompose the full conditional into an ODE-informed distribution and a data-informed distribution and in (b) we substitute the ODE-informed distribution $p(\mathbf{x}_u \mid \boldsymbol{\theta}, \mathbf{X}_{-u}, \boldsymbol{\phi}, \gamma)$ with its density given by equation (8).

```
[state.proxy.mean{:,symbols.state_string},state.proxy.inv_cov] = ...
proxy_for_ind_states(state.lin_comb,...
state.proxy.mean{:,symbols.state_string},...
param_proxy_mean',dC_times_invC,coupling_idx.states,...
symbols,mu,inv_sigma,simulation.observed_states,...
A_plus_gamma_inv,opt_settings);
```

```
end
```

Final results

```
plot_results(fig_handle,state.proxy,simulation,param_proxy_mean,...
   plot_handle,symbols,plot_settings,'final');
```



1.14 Time Taken

```
disp(['time taken: ' num2str(toc) ' seconds'])
```

```
time taken: 44.1589 seconds
```

VGM for Lorenz 96

Example dynamical system used in this code: **Lorenz 96** system with a **100 ODEs, with half** of the states **unobserved**. The ODE parameter is also unobserved.

2.1 User Input

2.1.1 Simulation Settings

- **true ODE parameter**: Input a real number:

```
simulation.ode_param = 8;
```

- **number of ODEs**: Input a positive integer:

```
simulation.numb_odes = 100;
```

- **ratio of number of observed states over number of unobserved states**: Input a real number in the interval $(0, 1]$:

```
simulation.observed_states = 0.5;    % only 50% of the states are observed
```

- **Observation noise**: Input a function handle:

```
simulation.state_obs_variance = @(mean)(bsxfun(@times,0.1,...
ones(size(mean))));
```

- **Time interval between observations**: Input a positive real number:

```
simulation.interval_between_observations = 0.1;
```

2.1.2 Estimation Settings

- **Kernel parameters ϕ**: Input a row vector of positive real numbers of size 1 x 2:

```
kernel.param = [10,0.2];
```

- **Error variance on state derivatives (i.e. γ)**: Input a row vector of positive real numbers of size 1 x number of ODEs:

```
state.derivative_variance = 6*ones(1,simulation.numb_odes);
```

- **Estimation times**: Input a row vector of positive real numbers in ascending order:

```
time.est = 0:0.1:4;
```

Preliminary operations

```
close all; clc; addpath('VGM_functions');
```

2.2 Preprocessing

```
[symbols,simulation,ode,odes_path,coupling_idx,opt_settings,plot_settings] = ...
proprocessing_Lorenz96(simulation);
```

```
ODEs:

/   d x_1                                    \
|   ----- == alpha - x_1 + x_100 (x_2 - x_99) |
|     dt                                     |
|                                            |
|   d x_2                                    |
|   ----- == alpha - x_2 + x_1 (x_3 - x_100) |
|     dt                                     |
|                                            |
|    d x_3                                   |
|    ----- == alpha - x_3 - x_2 (x_1 - x_4)  |
|      dt                                    |
|                                            |
|    d x_4                                   |
|    ----- == alpha - x_4 - x_3 (x_2 - x_5)  |
|      dt                                    |
|                                            |
|    d x_5                                   |
|    ----- == alpha - x_5 - x_4 (x_3 - x_6)  |
|      dt                                    |
|                                            |
|    d x_6                                   |
|    ----- == alpha - x_6 - x_5 (x_4 - x_7)  |
|      dt                                    |
|                                            |
|    d x_7                                   |
|    ----- == alpha - x_7 - x_6 (x_5 - x_8)  |
|      dt                                    |
|                                            |
|    d x_8                                   |
|    ----- == alpha - x_8 - x_7 (x_6 - x_9)  |
|      dt                                    |
|                                            |
|    d x_9                                   |
|    ----- == alpha - x_9 - x_8 (x_7 - x_10) |
|      dt                                    |
```

```
|                                                |
|   d x_10                                       |
|   ------ == alpha - x_10 - x_9 (x_8 - x_11)    |
|     dt                                         |
|                                                |
|  d x_11                                        |
|  ------ == alpha - x_11 - x_10 (x_9 - x_12)    |
|    dt                                          |
|                                                |
|  d x_12                                        |
|  ------ == alpha - x_12 - x_11 (x_10 - x_13)   |
|    dt                                          |
|                                                |
|  d x_13                                        |
|  ------ == alpha - x_13 - x_12 (x_11 - x_14)   |
|    dt                                          |
|                                                |
|  d x_14                                        |
|  ------ == alpha - x_14 - x_13 (x_12 - x_15)   |
|    dt                                          |
|                                                |
|  d x_15                                        |
|  ------ == alpha - x_15 - x_14 (x_13 - x_16)   |
|    dt                                          |
|                                                |
|  d x_16                                        |
|  ------ == alpha - x_16 - x_15 (x_14 - x_17)   |
|    dt                                          |
|                                                |
|  d x_17                                        |
|  ------ == alpha - x_17 - x_16 (x_15 - x_18)   |
|    dt                                          |
|                                                |
|  d x_18                                        |
|  ------ == alpha - x_18 - x_17 (x_16 - x_19)   |
|    dt                                          |
|                                                |
|  d x_19                                        |
|  ------ == alpha - x_19 - x_18 (x_17 - x_20)   |
|    dt                                          |
|                                                |
|  d x_20                                        |
|  ------ == alpha - x_20 - x_19 (x_18 - x_21)   |
|    dt                                          |
|                                                |
|  d x_21                                        |
|  ------ == alpha - x_21 - x_20 (x_19 - x_22)   |
|    dt                                          |
|                                                |
|  d x_22                                        |
|  ------ == alpha - x_22 - x_21 (x_20 - x_23)   |
|    dt                                          |
|                                                |
|  d x_23                                        |
|  ------ == alpha - x_23 - x_22 (x_21 - x_24)   |
|    dt                                          |
|                                                |
|  d x_24                                        |
```

```
|  ------ == alpha - x_24 - x_23 (x_22 - x_25) |
|    dt                                        |
|                                              |
|  d x_25                                      |
|  ------ == alpha - x_25 - x_24 (x_23 - x_26) |
|    dt                                        |
|                                              |
|  d x_26                                      |
|  ------ == alpha - x_26 - x_25 (x_24 - x_27) |
|    dt                                        |
|                                              |
|  d x_27                                      |
|  ------ == alpha - x_27 - x_26 (x_25 - x_28) |
|    dt                                        |
|                                              |
|  d x_28                                      |
|  ------ == alpha - x_28 - x_27 (x_26 - x_29) |
|    dt                                        |
|                                              |
|  d x_29                                      |
|  ------ == alpha - x_29 - x_28 (x_27 - x_30) |
|    dt                                        |
|                                              |
|  d x_30                                      |
|  ------ == alpha - x_30 - x_29 (x_28 - x_31) |
|    dt                                        |
|                                              |
|  d x_31                                      |
|  ------ == alpha - x_31 - x_30 (x_29 - x_32) |
|    dt                                        |
|                                              |
|  d x_32                                      |
|  ------ == alpha - x_32 - x_31 (x_30 - x_33) |
|    dt                                        |
|                                              |
|  d x_33                                      |
|  ------ == alpha - x_33 - x_32 (x_31 - x_34) |
|    dt                                        |
|                                              |
|  d x_34                                      |
|  ------ == alpha - x_34 - x_33 (x_32 - x_35) |
|    dt                                        |
|                                              |
|  d x_35                                      |
|  ------ == alpha - x_35 - x_34 (x_33 - x_36) |
|    dt                                        |
|                                              |
|  d x_36                                      |
|  ------ == alpha - x_36 - x_35 (x_34 - x_37) |
|    dt                                        |
|                                              |
|  d x_37                                      |
|  ------ == alpha - x_37 - x_36 (x_35 - x_38) |
|    dt                                        |
|                                              |
|  d x_38                                      |
|  ------ == alpha - x_38 - x_37 (x_36 - x_39) |
|    dt                                        |
```

```
|                                                |
|  d x_39                                        |
|  ------ == alpha - x_39 - x_38 (x_37 - x_40)   |
|    dt                                          |
|                                                |
|  d x_40                                        |
|  ------ == alpha - x_40 - x_39 (x_38 - x_41)   |
|    dt                                          |
|                                                |
|  d x_41                                        |
|  ------ == alpha - x_41 - x_40 (x_39 - x_42)   |
|    dt                                          |
|                                                |
|  d x_42                                        |
|  ------ == alpha - x_42 - x_41 (x_40 - x_43)   |
|    dt                                          |
|                                                |
|  d x_43                                        |
|  ------ == alpha - x_43 - x_42 (x_41 - x_44)   |
|    dt                                          |
|                                                |
|  d x_44                                        |
|  ------ == alpha - x_44 - x_43 (x_42 - x_45)   |
|    dt                                          |
|                                                |
|  d x_45                                        |
|  ------ == alpha - x_45 - x_44 (x_43 - x_46)   |
|    dt                                          |
|                                                |
|  d x_46                                        |
|  ------ == alpha - x_46 - x_45 (x_44 - x_47)   |
|    dt                                          |
|                                                |
|  d x_47                                        |
|  ------ == alpha - x_47 - x_46 (x_45 - x_48)   |
|    dt                                          |
|                                                |
|  d x_48                                        |
|  ------ == alpha - x_48 - x_47 (x_46 - x_49)   |
|    dt                                          |
|                                                |
|  d x_49                                        |
|  ------ == alpha - x_49 - x_48 (x_47 - x_50)   |
|    dt                                          |
|                                                |
|  d x_50                                        |
|  ------ == alpha - x_50 - x_49 (x_48 - x_51)   |
|    dt                                          |
|                                                |
|  d x_51                                        |
|  ------ == alpha - x_51 - x_50 (x_49 - x_52)   |
|    dt                                          |
|                                                |
|  d x_52                                        |
|  ------ == alpha - x_52 - x_51 (x_50 - x_53)   |
|    dt                                          |
|                                                |
|  d x_53                                        |
```

```
|  ------ == alpha - x_53 - x_52 (x_51 - x_54) |
|    dt                                        |
|                                              |
|  d x_54                                      |
|  ------ == alpha - x_54 - x_53 (x_52 - x_55) |
|    dt                                        |
|                                              |
|  d x_55                                      |
|  ------ == alpha - x_55 - x_54 (x_53 - x_56) |
|    dt                                        |
|                                              |
|  d x_56                                      |
|  ------ == alpha - x_56 - x_55 (x_54 - x_57) |
|    dt                                        |
|                                              |
|  d x_57                                      |
|  ------ == alpha - x_57 - x_56 (x_55 - x_58) |
|    dt                                        |
|                                              |
|  d x_58                                      |
|  ------ == alpha - x_58 - x_57 (x_56 - x_59) |
|    dt                                        |
|                                              |
|  d x_59                                      |
|  ------ == alpha - x_59 - x_58 (x_57 - x_60) |
|    dt                                        |
|                                              |
|  d x_60                                      |
|  ------ == alpha - x_60 - x_59 (x_58 - x_61) |
|    dt                                        |
|                                              |
|  d x_61                                      |
|  ------ == alpha - x_61 - x_60 (x_59 - x_62) |
|    dt                                        |
|                                              |
|  d x_62                                      |
|  ------ == alpha - x_62 - x_61 (x_60 - x_63) |
|    dt                                        |
|                                              |
|  d x_63                                      |
|  ------ == alpha - x_63 - x_62 (x_61 - x_64) |
|    dt                                        |
|                                              |
|  d x_64                                      |
|  ------ == alpha - x_64 - x_63 (x_62 - x_65) |
|    dt                                        |
|                                              |
|  d x_65                                      |
|  ------ == alpha - x_65 - x_64 (x_63 - x_66) |
|    dt                                        |
|                                              |
|  d x_66                                      |
|  ------ == alpha - x_66 - x_65 (x_64 - x_67) |
|    dt                                        |
|                                              |
|  d x_67                                      |
|  ------ == alpha - x_67 - x_66 (x_65 - x_68) |
|    dt                                        |
```

```
|                                                |
|  d x_68                                        |
|  ------ == alpha - x_68 - x_67 (x_66 - x_69)   |
|    dt                                          |
|                                                |
|  d x_69                                        |
|  ------ == alpha - x_69 - x_68 (x_67 - x_70)   |
|    dt                                          |
|                                                |
|  d x_70                                        |
|  ------ == alpha - x_70 - x_69 (x_68 - x_71)   |
|    dt                                          |
|                                                |
|  d x_71                                        |
|  ------ == alpha - x_71 - x_70 (x_69 - x_72)   |
|    dt                                          |
|                                                |
|  d x_72                                        |
|  ------ == alpha - x_72 - x_71 (x_70 - x_73)   |
|    dt                                          |
|                                                |
|  d x_73                                        |
|  ------ == alpha - x_73 - x_72 (x_71 - x_74)   |
|    dt                                          |
|                                                |
|  d x_74                                        |
|  ------ == alpha - x_74 - x_73 (x_72 - x_75)   |
|    dt                                          |
|                                                |
|  d x_75                                        |
|  ------ == alpha - x_75 - x_74 (x_73 - x_76)   |
|    dt                                          |
|                                                |
|  d x_76                                        |
|  ------ == alpha - x_76 - x_75 (x_74 - x_77)   |
|    dt                                          |
|                                                |
|  d x_77                                        |
|  ------ == alpha - x_77 - x_76 (x_75 - x_78)   |
|    dt                                          |
|                                                |
|  d x_78                                        |
|  ------ == alpha - x_78 - x_77 (x_76 - x_79)   |
|    dt                                          |
|                                                |
|  d x_79                                        |
|  ------ == alpha - x_79 - x_78 (x_77 - x_80)   |
|    dt                                          |
|                                                |
|  d x_80                                        |
|  ------ == alpha - x_80 - x_79 (x_78 - x_81)   |
|    dt                                          |
|                                                |
|  d x_81                                        |
|  ------ == alpha - x_81 - x_80 (x_79 - x_82)   |
|    dt                                          |
|                                                |
|  d x_82                                        |
```

```
|  ------ == alpha - x_82 - x_81 (x_80 - x_83) |
|    dt                                        |
|                                              |
|  d x_83                                      |
|  ------ == alpha - x_83 - x_82 (x_81 - x_84) |
|    dt                                        |
|                                              |
|  d x_84                                      |
|  ------ == alpha - x_84 - x_83 (x_82 - x_85) |
|    dt                                        |
|                                              |
|  d x_85                                      |
|  ------ == alpha - x_85 - x_84 (x_83 - x_86) |
|    dt                                        |
|                                              |
|  d x_86                                      |
|  ------ == alpha - x_86 - x_85 (x_84 - x_87) |
|    dt                                        |
|                                              |
|  d x_87                                      |
|  ------ == alpha - x_87 - x_86 (x_85 - x_88) |
|    dt                                        |
|                                              |
|  d x_88                                      |
|  ------ == alpha - x_88 - x_87 (x_86 - x_89) |
|    dt                                        |
|                                              |
|  d x_89                                      |
|  ------ == alpha - x_89 - x_88 (x_87 - x_90) |
|    dt                                        |
|                                              |
|  d x_90                                      |
|  ------ == alpha - x_90 - x_89 (x_88 - x_91) |
|    dt                                        |
|                                              |
|  d x_91                                      |
|  ------ == alpha - x_91 - x_90 (x_89 - x_92) |
|    dt                                        |
|                                              |
|  d x_92                                      |
|  ------ == alpha - x_92 - x_91 (x_90 - x_93) |
|    dt                                        |
|                                              |
|  d x_93                                      |
|  ------ == alpha - x_93 - x_92 (x_91 - x_94) |
|    dt                                        |
|                                              |
|  d x_94                                      |
|  ------ == alpha - x_94 - x_93 (x_92 - x_95) |
|    dt                                        |
|                                              |
|  d x_95                                      |
|  ------ == alpha - x_95 - x_94 (x_93 - x_96) |
|    dt                                        |
|                                              |
|  d x_96                                      |
|  ------ == alpha - x_96 - x_95 (x_94 - x_97) |
|    dt                                        |
```

```
|                                                |
|  d x_97                                        |
|  ------ == alpha - x_97 - x_96 (x_95 - x_98)   |
|    dt                                          |
|                                                |
|  d x_98                                        |
|  ------ == alpha - x_98 - x_97 (x_96 - x_99)   |
|    dt                                          |
|                                                |
| d x_99                                         |
| ------ == alpha - x_99 - x_98 (x_97 - x_100)   |
|   dt                                           |
|                                                |
| d x_100                                        |
| ------- == alpha - x_100 + x_99 (x_1 - x_98)   |
\    dt                                         /
```

2.3 Mass Action Dynamical Systems

A deterministic dynamical system is represented by a set of K ordinary differential equations (ODEs) with model parameters $\boldsymbol{\theta} \in \mathbb{R}^d$ that describe the evolution of K states $\mathbf{x}(t) = [x_1(t), \ldots, x_K(t)]^T$ such that:

$\dot{\mathbf{x}}(t) = \frac{d\mathbf{x}(t)}{dt} = \mathbf{f}(\mathbf{x}(t), \boldsymbol{\theta}) \qquad (1),$

A sequence of observations, $\mathbf{y}(t)$, is usually contaminated by measurement error which we assume to be normally distributed with zero mean and variance for each of the K states, i.e. $\mathbf{E} \sim \mathcal{N}(\mathbf{E}; \mathbf{0}, \mathbf{D})$, with $\mathbf{D}_{ik} = \sigma_k^2 \delta_{ik}$. For N distinct time points the overall system may therefore be summarized as

$\mathbf{Y} = \mathbf{X} + \mathbf{E},$

where

$\mathbf{X} = [\mathbf{x}(t_1), \ldots, \mathbf{x}(t_N)] = [\mathbf{x}_1, \ldots, \mathbf{x}_K]^T,$

$\mathbf{Y} = [\mathbf{y}(t_1), \ldots, \mathbf{y}(t_N)] = [\mathbf{y}_1, \ldots, \mathbf{y}_K]^T,$

and $\mathbf{x}_k = [x_k(t_1), \ldots, x_k(t_N)]^T$ is the k'th state sequence and $\mathbf{y}_k = [y_k(t_1), \ldots, y_k(t_N)]^T$ are the observations. Given the observations $\mathbf{Y}$ and the description of the dynamical system (1), the aim is to estimate both state variables $\mathbf{X}$ and parameters $\boldsymbol{\theta}$.

We consider only dynamical systems that are *locally linear* w.r.t. ODE parameters $\boldsymbol{\theta}$ and individual states $\mathbf{x}$. Such ODEs include mass-action kinetics and are given by:

$f_k(\mathbf{x}(t), \theta) = \sum_{i=1} \theta_{ki} \prod_{j \in \mathcal{M}_{ki}} x_j \qquad (2),$

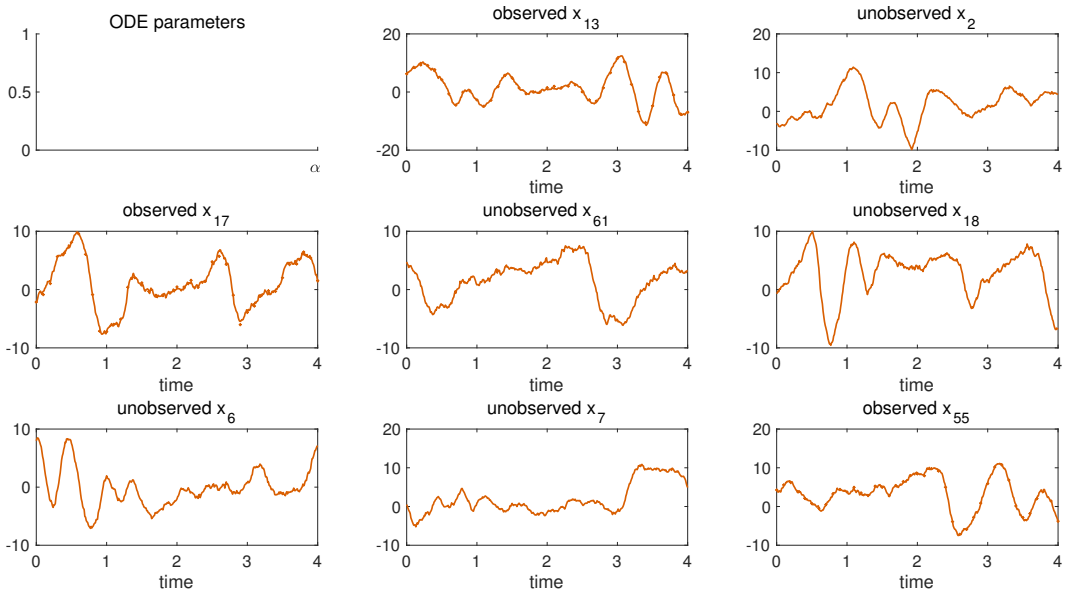
with $\mathcal{M}_{k_i} \subseteq \{1, \ldots, K\}$ describing the state variables in each factor of the equation (i.e. the functions are linear in parameters and contain arbitrary large products of monomials of the states).

2.4 Simulate Trajectories

Displaying only 8 of 100 randomly chosen states:

```
[simulation,obs_to_state_relation,fig_handle,plot_handle] = ...
```

```
simulate_state_dynamics(simulation,state,symbols,ode,odes_path,...
time,plot_settings);
```



start timer

```
tic;
```

2.5 Prior on States and State Derivatives

Gradient matching with Gaussian processes assumes a joint Gaussian process prior on states and their derivatives:

$$\begin{pmatrix} \mathbf{X} \\ \dot{\mathbf{X}} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{X} \\ \dot{\mathbf{X}} \end{pmatrix} ; \begin{matrix} \mathbf{0} \\ \mathbf{0} \end{matrix} , \begin{matrix} \mathbf{C}_{\phi} & \mathbf{C}_{\phi'} \\ {}'\mathbf{C}_{\phi} & \mathbf{C}''_{\phi} \end{matrix} \right) \qquad (3),$$

with

$$\text{cov}(x_k(t), x_k(t)) = C_{\phi_k}(t, t'),$$

$$\text{cov}(\dot{x}_k(t), x_k(t)) = \frac{\partial C_{\phi_k}(t,t')}{\partial t} =: C'_{\phi_k}(t, t'),$$

$$\text{cov}(x_k(t), \dot{x}_k(t)) = \frac{\partial C_{\phi_k}(t,t')}{\partial t'} =: {}'C_{\phi_k}(t, t'),$$

$$\text{cov}(\dot{x}_k(t), \dot{x}_k(t)) = \frac{\partial C_{\phi_k}(t,t')}{\partial t \partial t'} =: C''_{\phi_k}(t, t').$$

2.6 Matching Gradients

Given the joint distribution over states and their derivatives (3) as well as the ODEs (2), we therefore have two expressions for the state derivatives:

$$\dot{\mathbf{X}} = \mathbf{F} + \boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_1 \sim \mathcal{N}(\boldsymbol{\epsilon}_1; \mathbf{0}, \mathbf{I}\gamma),$$

$$\dot{\mathbf{X}} = {}'\mathbf{C}_{\phi}\mathbf{C}_{\phi}^{-1}\,\mathbf{X} + \boldsymbol{\epsilon}_2, \boldsymbol{\epsilon}_2 \sim \mathcal{N}(\boldsymbol{\epsilon}_2; \mathbf{0}, \mathbf{A}),$$

where $\mathbf{F} := \mathbf{f}(\mathbf{X}, \boldsymbol{\theta})$ and $\mathbf{A} := \mathbf{C}''_{\boldsymbol{\phi}} - {}'\mathbf{C}_{\boldsymbol{\phi}}\mathbf{C}_{\boldsymbol{\phi}}^{-1}\mathbf{C}'_{\boldsymbol{\phi}}$ and γ is the error variance in the ODEs. Note that, in a deterministic system, the output of the ODEs $\mathbf{F}$ should equal the state derivatives $\dot{\mathbf{X}}$. However, in the first equation above we relax this contraint by adding stochasticity to the state derivatives $\dot{\mathbf{X}}$ in order to compensate for a potential model mismatch. The second equation above is obtained by deriving the conditional distribution for $\dot{\mathbf{X}}$ from the joint distribution in equation (3). Equating the two expressions in the equations above we can eliminate the unknown state derivatives

2.7 Rewrite ODEs as Linear Combination in Parameters

Since, according to the mass action dynamics (equation 2), the ODEs are *linear in the parameters* $\boldsymbol{\theta}$ we can rewrite the ODEs in equation (2) as a linear combination in the parameters:

$$\mathbf{B}_{\boldsymbol{\theta}k}\boldsymbol{\theta} + \mathbf{b}_{\boldsymbol{\theta}k} \stackrel{!}{=} \mathbf{f}_k(\mathbf{X}, \boldsymbol{\theta}) \qquad (5),$$

where matrices $\mathbf{B}_{\boldsymbol{\theta}k}$ and $\mathbf{b}_{\boldsymbol{\theta}k}$ and $\mathbf{b}_{\boldsymbol{\theta}k}$ are defined such that the ODEs $\mathbf{f}_k(\mathbf{X}, \boldsymbol{\theta})$ are expressed as a linear combination in $\boldsymbol{\theta}$.

```
[ode_param.lin_comb.B,ode_param.lin_comb.b] = ...
rewrite_odes_as_linear_combination_in_parameters(ode,symbols);
```

2.8 Posterior over ODE Parameters

Inserting (5) into (4) and solving for $\boldsymbol{\theta}$ yields:

$$\boldsymbol{\theta} = \mathbf{B}_{\boldsymbol{\theta}}^{+}\left({}'\mathbf{C}_{\boldsymbol{\phi}}\mathbf{C}_{\boldsymbol{\phi}}^{-1}\mathbf{X} - \mathbf{b}_{\boldsymbol{\theta}} + \boldsymbol{\epsilon}_0\right),$$

where $\mathbf{B}_{\boldsymbol{\theta}}^{+}$ denotes the pseudo-inverse of $\mathbf{B}_{\boldsymbol{\theta}}$. Since $\mathbf{C}_{\boldsymbol{\phi}}$ is block diagonal we can rewrite the expression above as:

$$\begin{aligned}\boldsymbol{\theta} &= \left(\mathbf{B}_{\boldsymbol{\theta}}^{T}\mathbf{B}_{\boldsymbol{\theta}}\right)^{-1}\mathbf{B}_{\boldsymbol{\theta}}^{T}\left(\textstyle\sum_k {}'\mathbf{C}_{\boldsymbol{\phi}_k}\mathbf{C}_{\boldsymbol{\phi}_k}^{-1}\mathbf{X}_k - \mathbf{b}_{\boldsymbol{\theta}k} + \boldsymbol{\epsilon}_0^{(k)}\right)\\ &= \left(\mathbf{B}_{\boldsymbol{\theta}}^{T}\mathbf{B}_{\boldsymbol{\theta}}\right)^{-1}\left(\textstyle\sum_k \mathbf{B}_{\boldsymbol{\theta}k}^{T}\left({}'\mathbf{C}_{\boldsymbol{\phi}_k}\mathbf{C}_{\boldsymbol{\phi}_k}^{-1}\mathbf{X}_k - \mathbf{b}_{\boldsymbol{\theta}k} + \boldsymbol{\epsilon}_0^{(k)}\right)\right),\end{aligned}$$

where we subsitute the Moore-Penrose inverse for the pseudo-inverse (i.e. $\mathbf{B}_{\boldsymbol{\theta}}^{+} := \left(\mathbf{B}_{\boldsymbol{\theta}}^{T}\mathbf{B}_{\boldsymbol{\theta}}\right)^{-1}\mathbf{B}_{\boldsymbol{\theta}}^{T}$). We can therefore derive the posterior distribution over ODE parameters:

$$p(\boldsymbol{\theta} \mid \mathbf{X}, \boldsymbol{\phi}, \gamma) = \mathcal{N}\left(\boldsymbol{\theta}; \left(\mathbf{B}_{\boldsymbol{\theta}}^{T}\mathbf{B}_{\boldsymbol{\theta}}\right)^{-1}\left(\textstyle\sum_k \mathbf{B}_{\boldsymbol{\theta}k}^{T}\left({}'\mathbf{C}_{\boldsymbol{\phi}k}\mathbf{C}_{\boldsymbol{\phi}k}^{-1}\mathbf{X}_k - \mathbf{b}_{\boldsymbol{\theta}k}\right)\right),\ \mathbf{B}_{\boldsymbol{\theta}}^{+}\left(\mathbf{A} + \mathbf{I}\gamma\right)\mathbf{B}_{\boldsymbol{\theta}}^{+T}\right) \qquad (6).$$

2.9 Rewrite ODEs as Linear Combination in Individual States

Since, according to the mass action dynamics (equation 2), the ODEs are *linear in the individual state* $\mathbf{x}_u$ we can rewrite the ODE $\mathbf{f}_k(\mathbf{X}, \boldsymbol{\theta})$ as a linear combination in the individual state $\mathbf{x}_u$:

$$\mathbf{R}_{uk}\mathbf{x}_u + \mathbf{r}_{uk} \stackrel{!}{=} \mathbf{f}_k(\mathbf{X}, \boldsymbol{\theta}),$$

where matrices \mathbf{R}_{uk} and $\mathbf{r}_{uk}$ are defined such that the ODE $\mathbf{f}_k(\mathbf{X}, \boldsymbol{\theta})$ is expressed as a linear combination in the individual state $\mathbf{x}_u$.

```
[state.lin_comb.R,state.lin_comb.r] = ...
rewrite_odes_as_linear_combination_in_ind_states(ode,symbols,coupling_idx.states);
```

2.10 Posterior over Individual States

Given the linear combination of the ODEs w.r.t. an individual state, we define the matrices $\mathbf{B}_u$ and $\mathbf{b}_u$ such that the expression $\mathbf{f}(\mathbf{X}, \boldsymbol{\theta}) - {}'\mathbf{C}_{\boldsymbol{\phi}}\mathbf{C}_{\boldsymbol{\phi}}^{-1}\mathbf{X}$ is rewritten as a linear combination in an individual state $\mathbf{x}_u$:

$$\mathbf{B}_u\mathbf{x}_u + \mathbf{b}_u \stackrel{!}{=} \mathbf{f}(\mathbf{X}, \boldsymbol{\theta}) - {}'\mathbf{C}_{\boldsymbol{\phi}}\mathbf{C}_{\boldsymbol{\phi}}^{-1}\mathbf{X} \qquad (7).$$

Inserting (7) into (4) and solving for $\mathbf{x}_u$ yields:

$$\mathbf{x}_u = \mathbf{B}_u^{+}\left(\boldsymbol{\epsilon}_0 - \mathbf{b}_u\right),$$

where $\mathbf{B}_u^{+}$ denotes the pseudo-inverse of $\mathbf{B}_u$. Since $\mathbf{C}_{\boldsymbol{\phi}}$ is block diagonal we can rewrite the expression above as:

$$\begin{aligned}\mathbf{x}_u &= \left(\mathbf{B}_u\mathbf{B}_u^T\right)^{-1}\mathbf{B}_u^T\textstyle\sum_k\left(\boldsymbol{\epsilon}_0^{(k)} - \mathbf{b}_{uk}\right)\\ &= \left(\mathbf{B}_u\mathbf{B}_u^T\right)^{-1}\textstyle\sum_k\mathbf{B}_{uk}^T\left(\boldsymbol{\epsilon}_0^{(k)} - \mathbf{b}_{uk}\right),\end{aligned}$$

where we substitute the Moore-Penrose inverse for the pseudo-inverse (i.e. $\mathbf{B}_{\boldsymbol{\theta}}^{+} := \left(\mathbf{B}_{\boldsymbol{\theta}}^T\mathbf{B}_{\boldsymbol{\theta}}\right)^{-1}\mathbf{B}_{\boldsymbol{\theta}}^T$). We can therefore derive the posterior distribution over an individual state $\mathbf{x}_u$:

$$p(\mathbf{x}_u \mid \mathbf{X}_{-u}, \boldsymbol{\phi}, \gamma) = \mathcal{N}\left(\mathbf{x}_u; \left(\mathbf{B}_u\mathbf{B}_u^T\right)^{-1}\left(-\textstyle\sum_k\mathbf{B}_{uk}^T\mathbf{b}_{uk}\right),\ \mathbf{B}_u^{+}\left(\mathbf{A} + \mathbf{I}\gamma\right)\mathbf{B}_u^{+T}\right) \qquad (8),$$

with $\mathbf{X}_{-u}$ denoting the set of all states except state $\mathbf{x}_u$.

2.11 Mean-field Variational Inference

To infer the parameters $\boldsymbol{\theta}$, we want to find the maximum a posteriori estimate (MAP):

$$\begin{aligned}\boldsymbol{\theta}^* &:= \arg\max_{\boldsymbol{\theta}}\ \ln p(\boldsymbol{\theta} \mid \mathbf{Y}, \boldsymbol{\phi}, \gamma, \sigma)\\ &= \arg\max_{\boldsymbol{\theta}}\ \ln\textstyle\int p(\boldsymbol{\theta}, \mathbf{X} \mid \mathbf{Y}, \boldsymbol{\phi}, \gamma, \boldsymbol{\sigma})d\mathbf{X}\\ &= \arg\max_{\boldsymbol{\theta}}\ \ln\textstyle\int p(\boldsymbol{\theta} \mid \mathbf{X}, \boldsymbol{\phi}, \gamma)p(\mathbf{X} \mid \mathbf{Y}, \boldsymbol{\phi}, \boldsymbol{\sigma})d\mathbf{X} \qquad (9).\end{aligned}$$

However, the integral above is intractable due to the strong couplings induced by the nonlinear ODEs $\mathbf{f}$ which appear in the term $p(\boldsymbol{\theta} \mid \mathbf{X}, \boldsymbol{\phi}, \gamma)$.

We use mean-field variational inference to establish variational lower bounds that are analytically tractable by decoupling state variables from the ODE parameters as well as decoupling the state variables from each other. Note that, since the ODEs described by equation (2) are *locally linear*, both conditional distributions $p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{Y}, \boldsymbol{\phi}, \gamma, \boldsymbol{\sigma})$ (equation (6)) and $p(\mathbf{x}_u \mid \boldsymbol{\theta}, \mathbf{X}_{-u}, \mathbf{Y}, \boldsymbol{\phi}, \gamma, \boldsymbol{\sigma})$ (equation (8)) are analytically tractable and Gaussian distributed as mentioned previously. The decoupling is induced by designing a variational distribution $Q(\boldsymbol{\theta}, \mathbf{X})$ which is restricted to the family of factorial distributions:

$$\mathcal{Q} := \left\{Q : Q(\boldsymbol{\theta}, \mathbf{X}) = q(\boldsymbol{\theta})\textstyle\prod_u q(\mathbf{x}_u)\right\}.$$

The particular form of $q(\boldsymbol{\theta})$ and $q(\mathbf{x}_u)$ are designed to be Gaussian distributed which places them in the same family as the true full conditional distributions. To find the optimal factorial distribution we minimize the Kullback-Leibler divergence between the variational and the true posterior distribution:

$$\hat{Q} := \arg\min_{Q(\boldsymbol{\theta},\mathbf{X})\in\mathcal{Q}}\mathrm{KL}\left[Q(\boldsymbol{\theta}, \mathbf{X})\ ||\ p(\boldsymbol{\theta}, \mathbf{X} \mid \mathbf{Y}, \boldsymbol{\phi}, \gamma, \sigma)\right] \qquad (10),$$

where $\hat{Q}$ is the proxy distribution. The proxy distribution that minimizes the KL-divergence (10) depends on the true full conditionals and is given by:

$$\hat{q}(\boldsymbol{\theta}) \propto \exp\left(E_{Q_{-\boldsymbol{\theta}}} \ln p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{Y}, \boldsymbol{\phi}, \gamma, \sigma) \right) \qquad (11)$$
$$\hat{q}(\mathbf{x}_u) \propto \exp\left(E_{Q_{-u}} \ln p(\mathbf{x}_u \mid \boldsymbol{\theta}, \mathbf{X}_{-u}, \mathbf{Y}, \boldsymbol{\phi}, \gamma, \boldsymbol{\sigma}) \right) \qquad (12).$$

2.12 Fitting Observations of State Trajectories

We fit the observations of state trajectories by standard GP regression. The data-informed distribution $p(\mathbf{X} \mid \mathbf{Y}, \boldsymbol{\phi}, \sigma)$ in euqation (9) can be determined analytically using Gaussian process regression with the GP prior $p(\mathbf{X} \mid \phi) = \prod_k \mathcal{N}(\mathbf{x}_k; \mathbf{0}, \mathbf{C}_{\boldsymbol{\phi}_k})$:

$$p(\mathbf{X} \mid \mathbf{Y}, \boldsymbol{\phi}, \gamma) = \prod_k \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_k(\mathbf{y}_k), \sigma_k),$$

where $\boldsymbol{\mu}_k(\mathbf{y}_k) := \boldsymbol{\sigma}_k^{-2} \left(\boldsymbol{\sigma}_k^{-2}\mathbf{I} + \mathbf{C}_{\boldsymbol{\phi}_k}^{-1} \right)^{-1} \mathbf{y}_k$ and $\boldsymbol{\sigma}_k^{-1} := \boldsymbol{\sigma}_k^{-2}\mathbf{I} + \mathbf{C}_{\phi_k}^{-1}$.

```
[mu,inv_sigma] = fitting_state_observations(inv_C,obs_to_state_relation,...
simulation,symbols);
```

2.13 Coordinate Ascent Variational Gradient Matching

We minimize the KL-divergence in equation (10) by coordinate descent (where each step is analytically tractable) by iterating between determining the proxy for the distribution over ODE parameters $\hat{q}(\boldsymbol{\theta})$ and the proxies for the distribution over individual states $\hat{q}(\mathbf{x}_u)$.

Initialize the state estimation by the GP regression posterior

```
state.proxy.mean = array2table([time.est',mu],...
'VariableNames',['time',symbols.state_string]);
```

Coordinate ascent

```
for i = 1:opt_settings.coord_ascent_numb_iter
```

2.13.1 Proxy for ODE Parameters

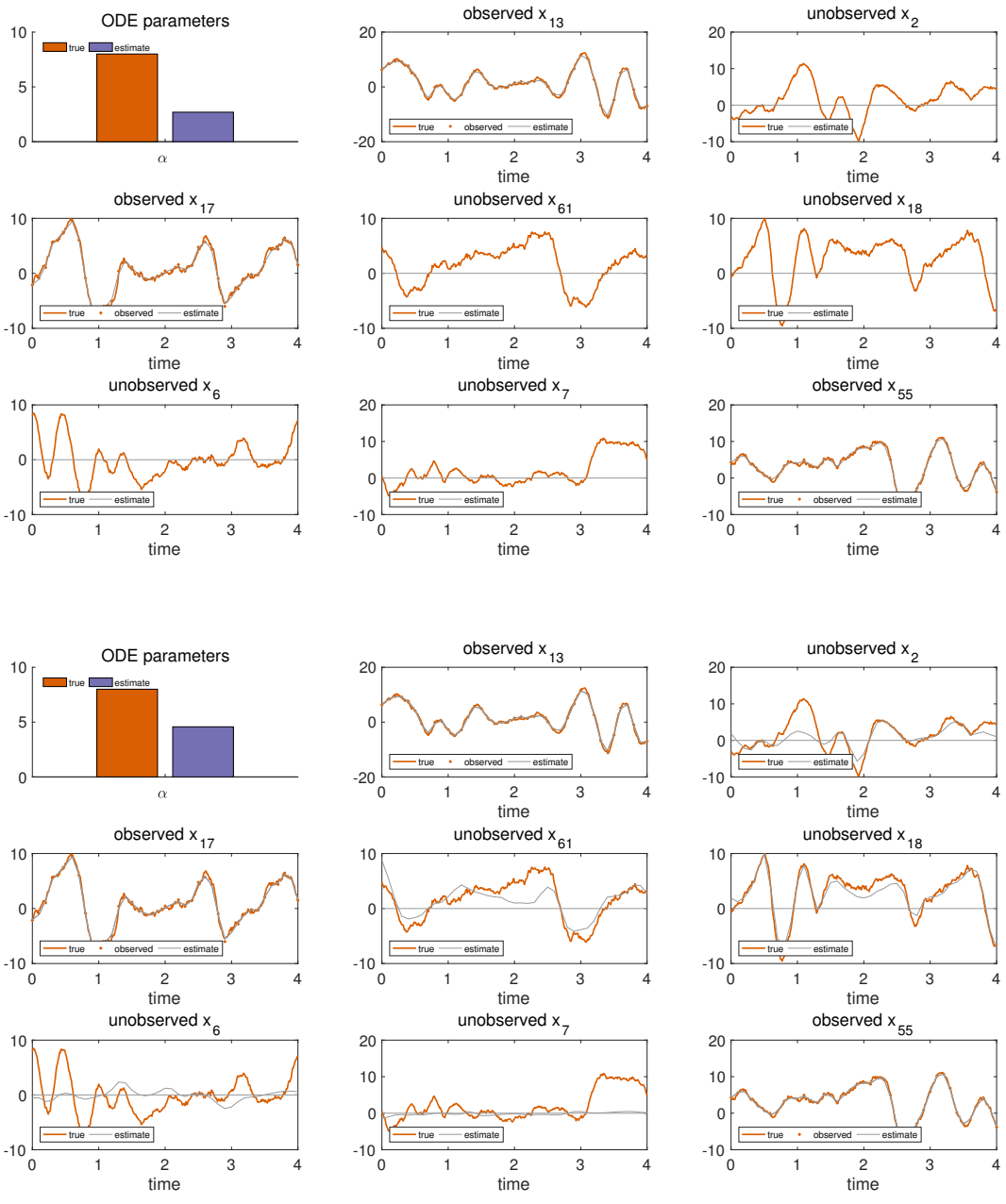
Expanding the proxy distribution in equation (11) for $\boldsymbol{\theta}$ yields:

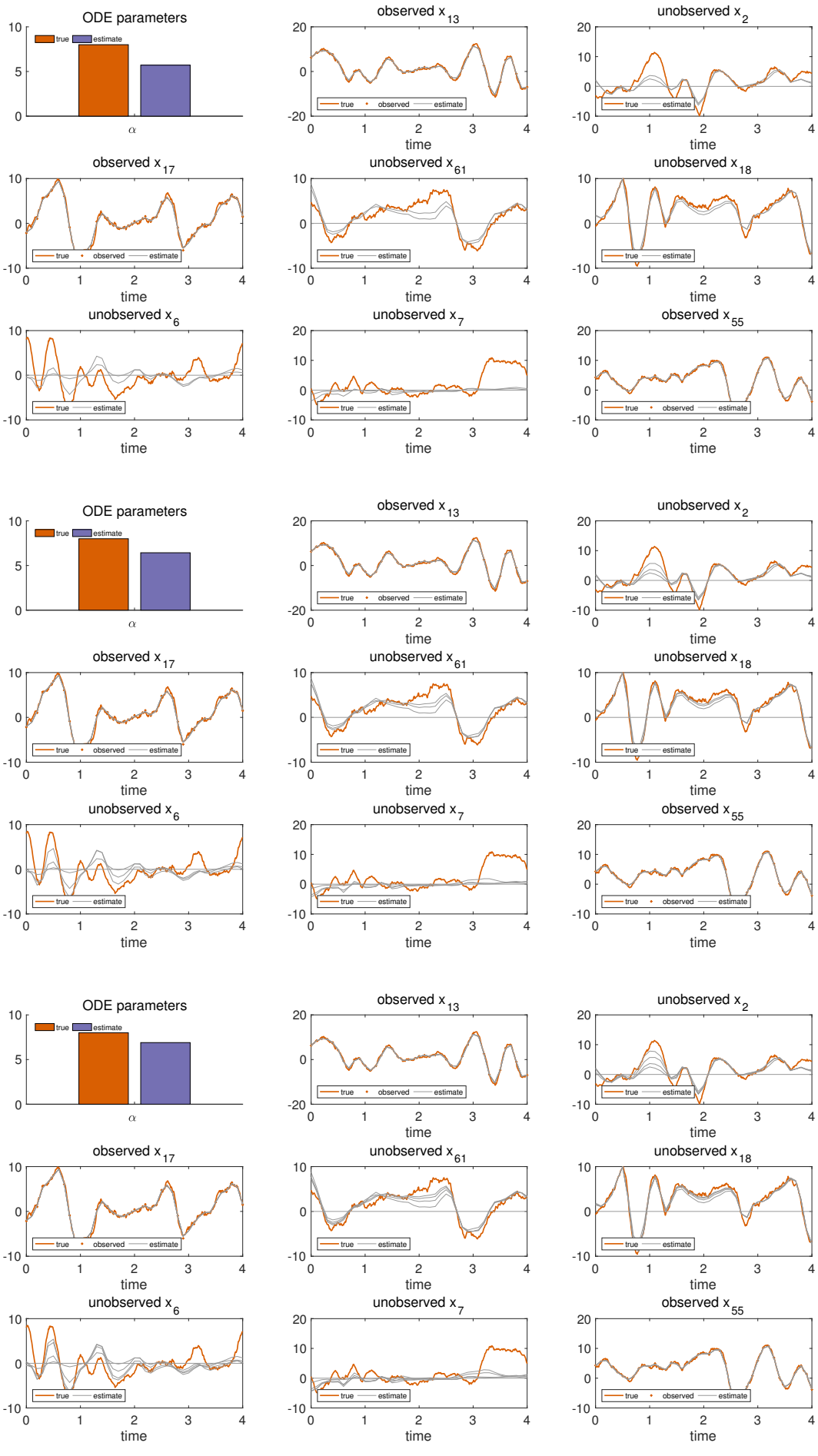
$$\begin{aligned}\hat{q}(\boldsymbol{\theta}) &\propto \exp\left(E_{Q_{-\boldsymbol{\theta}}} \ln p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{Y}, \boldsymbol{\phi}, \gamma, \boldsymbol{\sigma}) \right) \\ &= \exp\left(E_{Q_{-\boldsymbol{\theta}}} \ln \mathcal{N}\left(\boldsymbol{\theta}; \left(\mathbf{B}_{\boldsymbol{\theta}}^T \mathbf{B}_{\boldsymbol{\theta}}\right)^{-1} \left(\sum_k \mathbf{B}_{\boldsymbol{\theta}k}^T \left({}'\mathbf{C}_{\boldsymbol{\phi}_k} \mathbf{C}_{\boldsymbol{\phi}k}^{-1} \mathbf{X}_k - \mathbf{b}_{\boldsymbol{\theta}_k} \right) \right), \mathbf{B}_{\boldsymbol{\theta}}^{+} \left(\mathbf{A} + \mathbf{I}\gamma \right) \mathbf{B}_{\boldsymbol{\theta}}^{+T} \right) \right)\end{aligned}$$

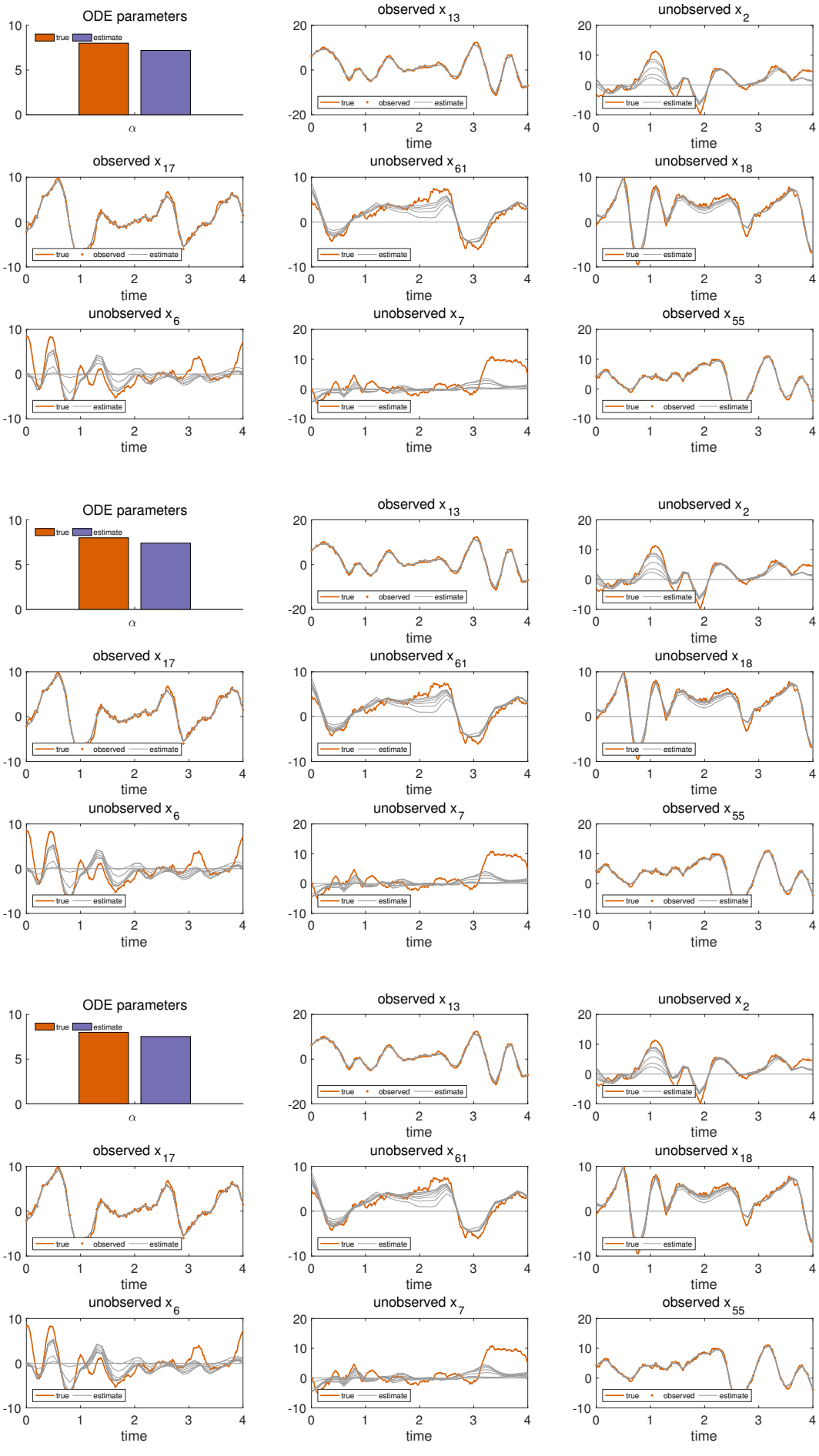
where we substitute $p(\boldsymbol{\theta} \mid \mathbf{X}, \boldsymbol{\phi}, \gamma)$ with its density given in equation (6).

```
[param_proxy_mean,param_proxy_inv_cov] = ...
proxy_for_ode_parameters(state.proxy.mean{:,symbols.state_string},...
dC_times_invC,ode_param.lin_comb,symbols,A_plus_gamma_inv,opt_settings);
```

```
if i==1 || ~mod(i,1)
    plot_results(fig_handle,state.proxy,simulation,param_proxy_mean,...
    plot_handle,symbols,plot_settings,'not_final');
end
```

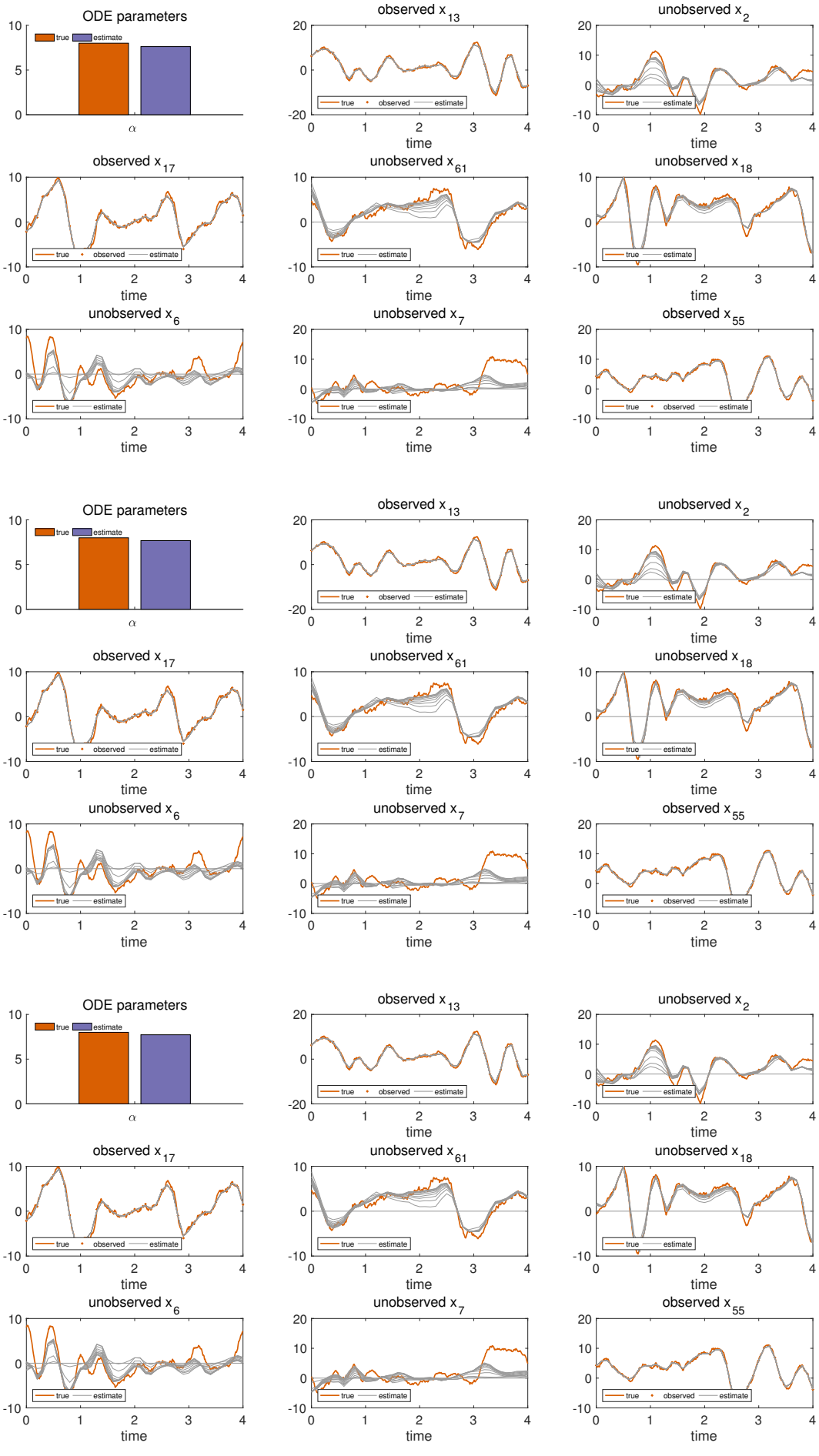



ODE parameters
true
estimate
α
observed x 13
unobserved x 2
observed x 17
unobserved x 61
unobserved x 18
unobserved x 6
unobserved x 7
observed x 55
time
observed
ODE parameters
true
estimate
α
observed x 13
unobserved x 2
observed x 17
unobserved x 61
unobserved x 18
unobserved x 6
unobserved x 7
observed x 55
time
observed
ODE parameters
true
estimate
α
observed x 13
unobserved x 2
observed x 17
unobserved x 61
unobserved x 18
unobserved x 6
unobserved x 7
observed x 55
time
observed



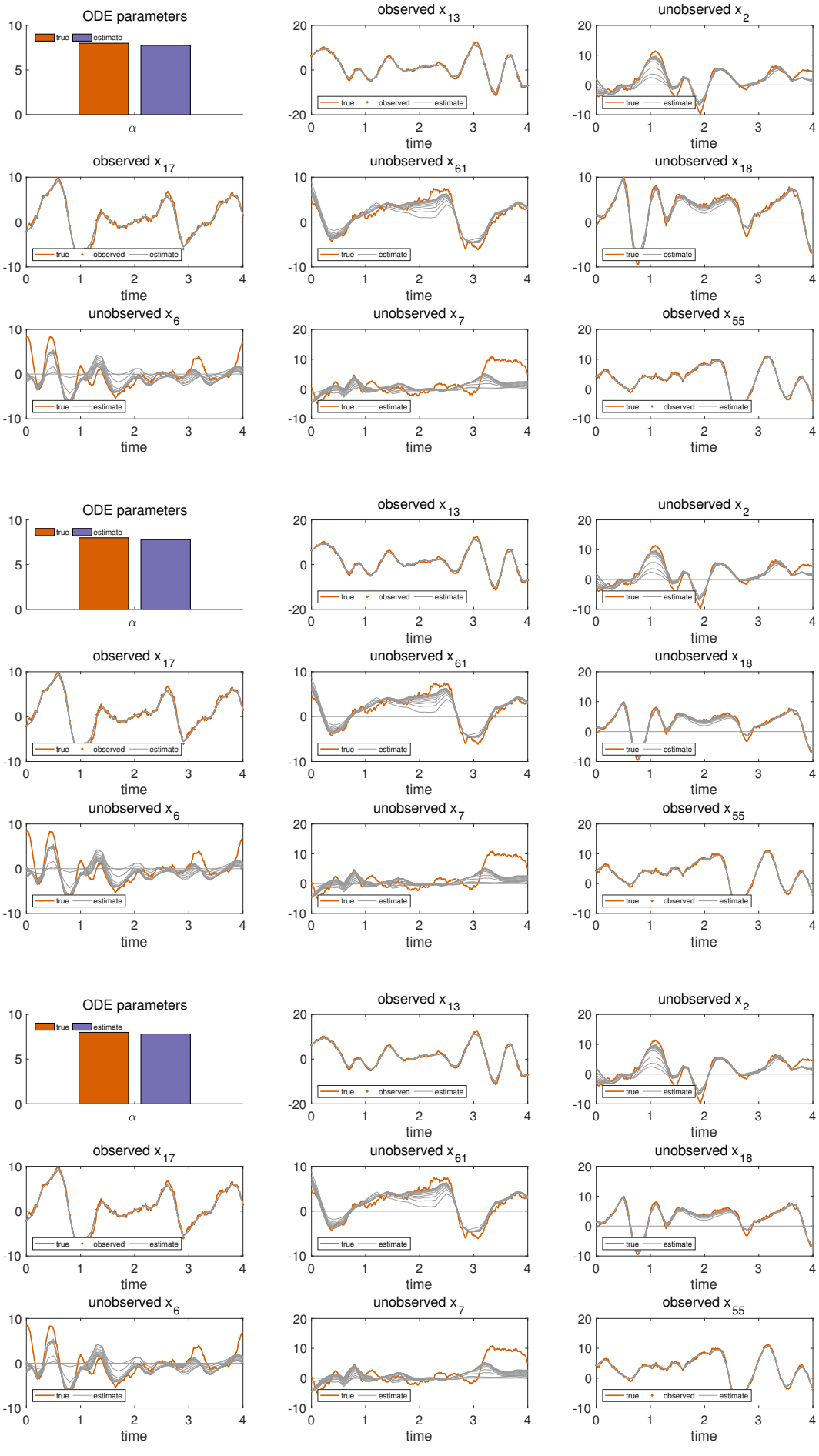
ODE parameters
observed x 13
unobserved x 2
observed x 17
unobserved x 61
unobserved x 18
unobserved x 6
unobserved x 7
observed x 55

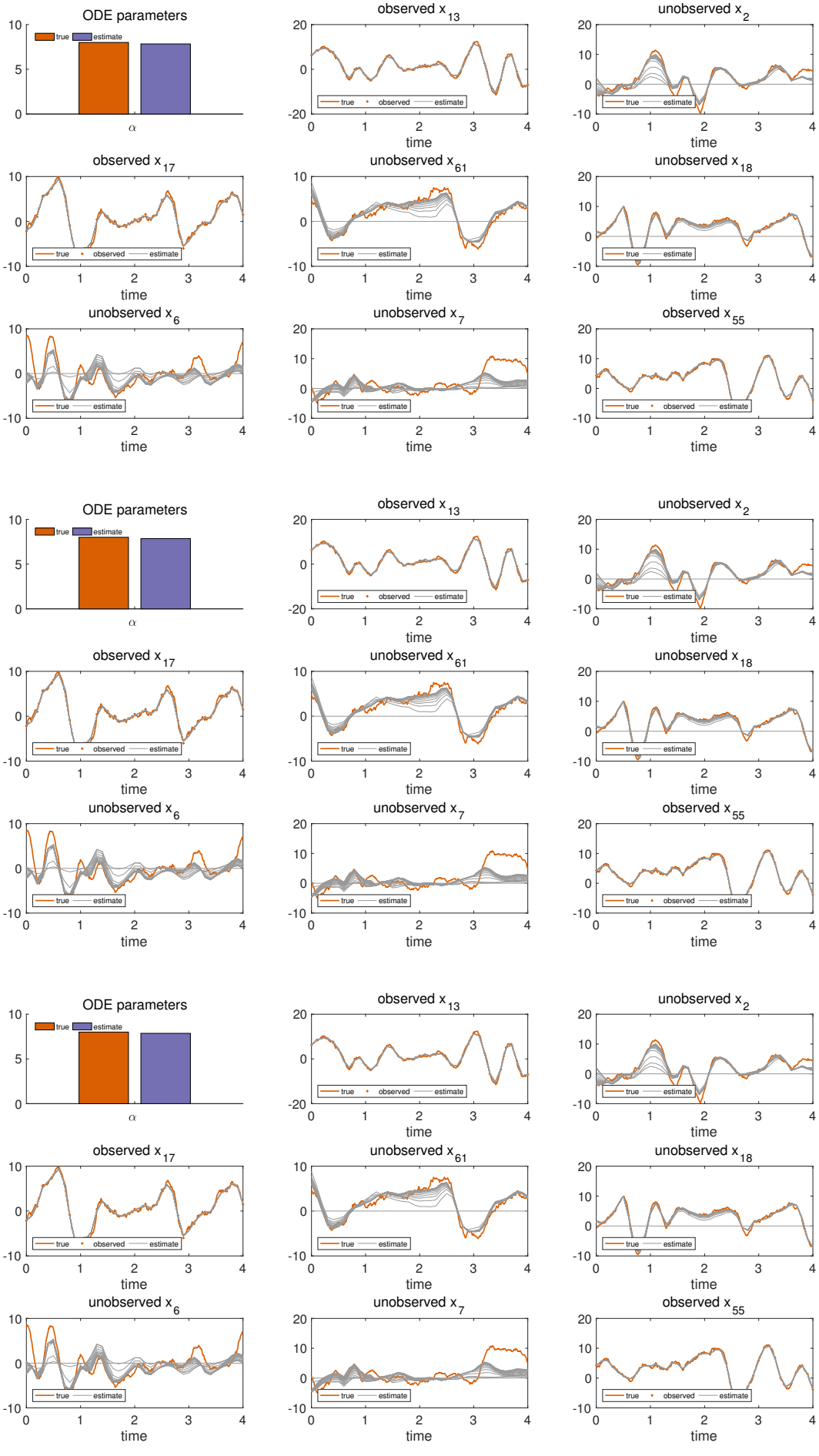
ODE parameters
observed x 13
unobserved x 2
observed x 17
unobserved x 61
unobserved x 18
unobserved x 6
unobserved x 7
observed x 55

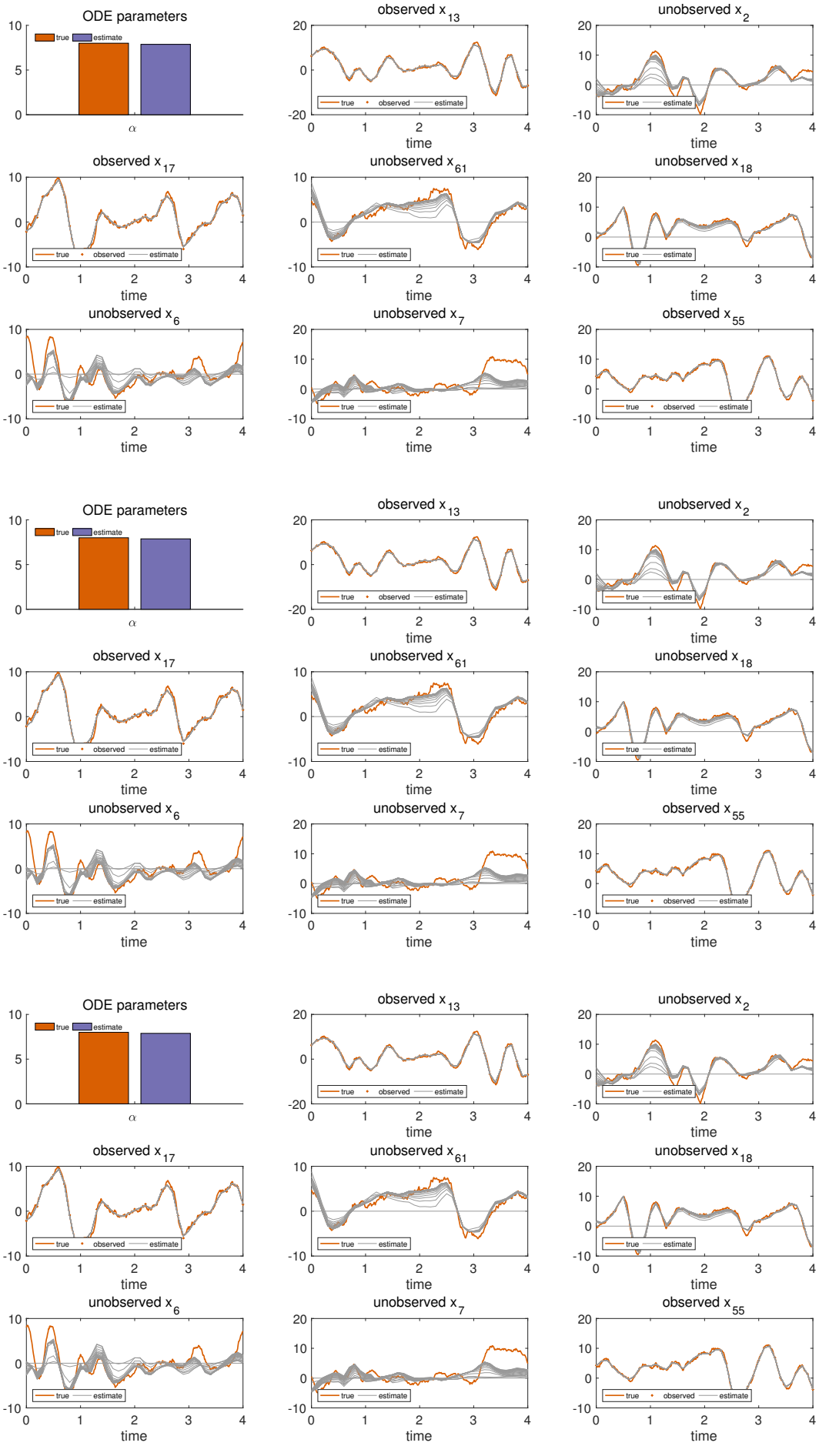


ODE parameters
true
estimate
α
observed x 13
true
observed
estimate
time
unobserved x 2
true
estimate
time
observed x 17
true
observed
estimate
time
unobserved x 61
true
estimate
time
unobserved x 18
true
estimate
time
unobserved x 6
true
estimate
time
unobserved x 7
true
estimate
time
observed x 55
true
observed
estimate
time

ODE parameters
true
estimate
α
observed x 13
true
observed
estimate
time
unobserved x 2
true
estimate
time
observed x 17
true
observed
estimate
time
unobserved x 61
true
estimate
time
unobserved x 18
true
estimate
time
unobserved x 6
true
estimate
time
unobserved x 7
true
estimate
time
observed x 55
true
observed
estimate
time


ODE parameters
true
estimate
α
observed x 13
unobserved x 2
observed x 17
unobserved x 61
unobserved x 18
unobserved x 6
unobserved x 7
observed x 55
time
ODE parameters
true
estimate
α
observed x 13
unobserved x 2
observed x 17
unobserved x 61
unobserved x 18
unobserved x 6
unobserved x 7
observed x 55
time
ODE parameters
true
estimate
α
observed x 13
unobserved x 2
observed x 17
unobserved x 61
unobserved x 18
unobserved x 6
unobserved x 7
observed x 55
time


ODE parameters
true
estimate
α
observed x 13
true
observed
estimate
time
unobserved x 2
true
estimate
time
observed x 17
true
observed
estimate
time
unobserved x 61
true
estimate
time
unobserved x 18
true
estimate
time
unobserved x 6
true
estimate
time
unobserved x 7
true
estimate
time
observed x 55
true
observed
estimate
time
ODE parameters
true
estimate
α
observed x 13
true
observed
estimate
time
unobserved x 2
true
estimate
time
observed x 17
true
observed
estimate
time
unobserved x 61
true
estimate
time
unobserved x 18
true
estimate
time
unobserved x 6
true
estimate
time
unobserved x 7
true
estimate
time
observed x 55
true
observed
estimate
time
ODE parameters
true
estimate
α
observed x 13
true
observed
estimate
time
unobserved x 2
true
estimate
time
observed x 17
true
observed
estimate
time
unobserved x 61
true
estimate
time
unobserved x 18
true
estimate
time
unobserved x 6
true
estimate
time
unobserved x 7
true
estimate
time
observed x 55
true
observed
estimate
time


ODE parameters
true
estimate
α
observed x13
unobserved x2
observed x17
unobserved x61
unobserved x18
unobserved x6
unobserved x7
observed x55
time

2.13.2 Proxy for Individual States

Expanding the proxy distribution in equation (12) over the individual state $\mathbf{x}_u$:

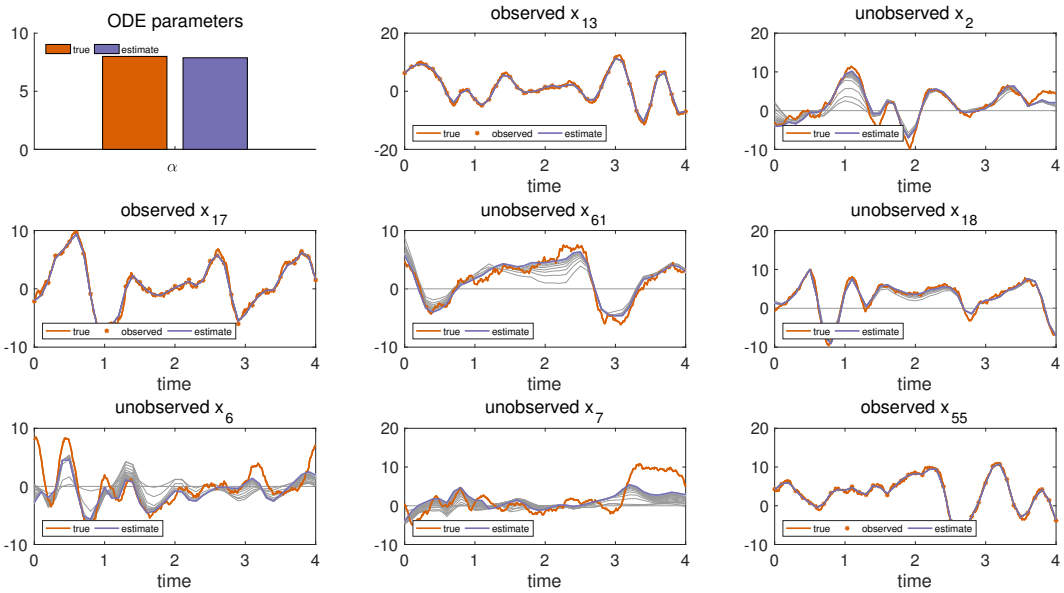
$$\begin{aligned}\hat{q}(\mathbf{x}_u) &\overset{(a)}{\propto} \exp\left(E_{Q_{-u}} \ln(p(\mathbf{x}_u \mid \boldsymbol{\theta}, \mathbf{X}_{-u}, \boldsymbol{\phi}, \gamma) p(\mathbf{x}_u \mid \mathbf{Y}, \boldsymbol{\phi}, \boldsymbol{\sigma})) \right) \\ &\overset{(b)}{=} \exp\left(E_{Q_{-u}} \ln \mathcal{N}\left(\mathbf{x}_u; -\mathbf{B}_u^{+}\mathbf{b}_u,\ \mathbf{B}_u^{+}\left(\mathbf{A} + \mathbf{I}\gamma\right)\mathbf{B}_u^{+T}\right) + E_{Q_{-u}} \ln \mathcal{N}\left(\mathbf{x}_u; \boldsymbol{\mu}_u(\mathbf{Y}), \boldsymbol{\Sigma}_u\right) \right) \\ &= \exp\left(E_{Q_{-u}} \ln \mathcal{N}\left(\mathbf{x}_u; -\mathbf{B}_u^{+}\mathbf{b}_u,\ \mathbf{B}_u^{+}\left(\mathbf{A} + \mathbf{I}\gamma\right)\mathbf{B}_u^{+T}\right) + E_{Q_{-u}} \ln \mathcal{N}\left(\mathbf{x}_u; \boldsymbol{\mu}_u(\mathbf{Y}), \boldsymbol{\sigma}_u\right) \right)\end{aligned}$$

In (a) we decompose the full conditional into an ODE-informed distribution and a data-informed distribution and in (b) we substitute the ODE-informed distribution $p(\mathbf{x}_u \mid \boldsymbol{\theta}, \mathbf{X}_{-u}, \boldsymbol{\phi}, \gamma)$ with its density given by equation (8).

```
[state.proxy.mean{:,symbols.state_string},state.proxy.inv_cov] = ...
 proxy_for_ind_states(state.lin_comb,...
 state.proxy.mean{:,symbols.state_string},...
 param_proxy_mean',dC_times_invC,coupling_idx.states,...
 symbols,mu,inv_sigma,simulation.observed_states,...
     A_plus_gamma_inv,opt_settings);
```

Final result

```
plot_results(fig_handle,state.proxy,simulation,param_proxy_mean,...
plot_handle,symbols,plot_settings,'final');
```



2.14 Time Taken

```
disp(['time taken: ' num2str(toc) ' seconds'])
```

```
time taken: 348.5171 seconds
```

VGM for Lorenz Attractor

Example dynamical system used in this code: **Lorenz attractor** system with the **y-dimension unobserved**. The ODE parameters are also unobserved.

3.1 User Input

3.1.1 Simulation Settings

- **True ODE parameters:** Input a row vector of real numbers of size 1 x 3:

```
simulation.ode_param = [10,28,8/3];
```

- **Final time for simulation:** Input a positive real number:

```
simulation.final_time = 20;
```

- **Observation noise:** Input a function handle:

```
simulation.state_obs_variance = @(mean)(bsxfun(@times,[2,2],...
ones(size(mean))));
```

- **Time interval between observations**: Input a positive real number:

```
simulation.interval_between_observations = 0.1;
```

3.1.2 Estimation Settings

- **Kernel parameters** ϕ: Input a row vector of positive real numbers of size 1 x 2:

```
kernel.param = [10,0.2];
```

- **Error variance on state derivatives (i.e.** γ): Input a row vector of positive real numbers of size 1 x 3:

```
state.derivative_variance = [6,6,6];
```

- **Estimation times**: Input a row vector of positive real numbers in ascending order:

```
time.est = 0:0.1:20;
```

Preliminary operations

```
close all; clc; addpath('VGM_functions')
```

3.2 Preprocessing

```
[symbols,simulation,ode,odes_path,coupling_idx,opt_settings,plot_settings] = ...
preprocessing_Lorenz_Attractor (simulation);
```

```
ODEs:

/  d x                    \
|  --- == -sigma (x - y)  |
|   dt                   |
|                        |
| d y                    |
| --- == rho x - y - x z  |
|  dt                    |
|                        |
|  d z                   |
|  --- == x y - lambda z |
\   dt                   /
```

3.3 Mass Action Dynamical Systems

A deterministic dynamical system is represented by a set of K ordinary differential equations (ODEs) with model parameters $\boldsymbol{\theta} \in \mathbb{R}^d$ that describe the evolution of K states $\mathbf{x}(t) = [x_1(t), \ldots, x_K(t)]^T$ such that:

$\dot{\mathbf{x}}(t) = \frac{d\mathbf{x}(t)}{dt} = \mathbf{f}(\mathbf{x}(t), \boldsymbol{\theta}) \qquad (1),$

A sequence of observations, $\mathbf{y}(t)$, is usually contaminated by measurement error which we assume to be normally distributed with zero mean and variance for each of the K states, i.e. $\mathbf{E} \sim \mathcal{N}(\mathbf{E}; \mathbf{0}, \mathbf{D})$, with $\mathbf{D}_{ik} = \sigma_k^2 \delta_{ik}$. For N distinct time points the overall system may therefore be summarized as

$\mathbf{Y} = \mathbf{X} + \mathbf{E},$

where

$\mathbf{X} = [\mathbf{x}(t_1), \ldots, \mathbf{x}(t_N)] = [\mathbf{x}_1, \ldots, \mathbf{x}_K]^T,$

$\mathbf{Y} = [\mathbf{y}(t_1), \ldots, \mathbf{y}(t_N)] = [\mathbf{y}_1, \ldots, \mathbf{y}_K]^T,$

and $\mathbf{x}_k = [x_k(t_1), \ldots, x_k(t_N)]^T$ is the k'th state sequence and $\mathbf{y}_k = [y_k(t_1), \ldots, y_k(t_N)]^T$ are the observations. Given the observations $\mathbf{Y}$ and the description of the dynamical system (1), the aim is to estimate both state variables $\mathbf{X}$ and parameters $\boldsymbol{\theta}$.

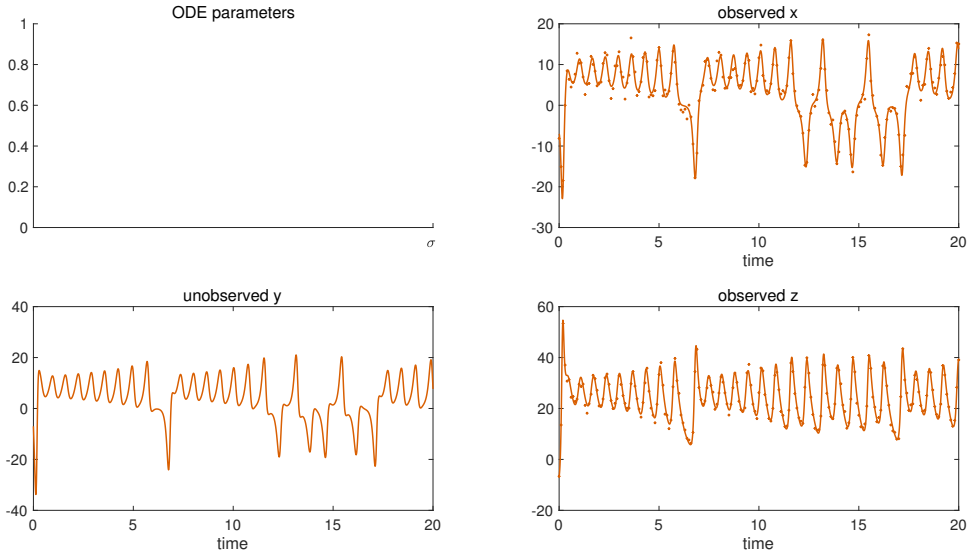
We consider only dynamical systems that are *locally linear* w.r.t. ODE parameters $\boldsymbol{\theta}$ and individual states $\mathbf{x}$. Such ODEs include mass-action kinetics and are given by:

$f_k(\mathbf{x}(t), \theta) = \sum_{i=1} \theta_{ki} \prod_{j \in \mathcal{M}_{ki}} x_j \qquad (2),$

with $\mathcal{M}_{k_i} \subseteq \{1, \ldots, K\}$ describing the state variables in each factor of the equation (i.e. the functions are linear in parameters and contain arbitrary large products of monomials of the states).

3.4 Simulate Trajectories

```
[simulation,obs_to_state_relation,fig_handle,plot_handle] = ...
simulate_state_dynamics(simulation,state,symbols,ode,odes_path,time,plot_settings);
```



start timer

```
tic;
```

3.5 Prior on States and State Derivatives

Gradient matching with Gaussian processes assumes a joint Gaussian process prior on states and their derivatives:

$$\begin{pmatrix} \mathbf{X} \\ \dot{\mathbf{X}} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{X} \\ \dot{\mathbf{X}} \end{pmatrix} ; \begin{matrix} \mathbf{0} \\ \mathbf{0} \end{matrix} , \begin{matrix} \mathbf{C}_\phi & \mathbf{C}'_\phi \\ {}'\mathbf{C}_\phi & \mathbf{C}''_\phi \end{matrix} \right) \qquad (3),$$

with

$\mathrm{cov}(x_k(t), x_k(t)) = C_{\phi_k}(t, t'),$

$\mathrm{cov}(\dot{x}_k(t), x_k(t)) = \frac{\partial C_{\phi_k}(t,t')}{\partial t} =: C'_{\phi_k}(t, t'),$

$\mathrm{cov}(x_k(t), \dot{x}_k(t)) = \frac{\partial C_{\phi_k}(t,t')}{\partial t'} =: {}'C_{\phi_k}(t, t'),$

$\mathrm{cov}(\dot{x}_k(t), \dot{x}_k(t)) = \frac{\partial C_{\phi_k}(t,t')}{\partial t \partial t'} =: C''_{\phi_k}(t, t').$

3.6 Matching Gradients

Given the joint distribution over states and their derivatives (3) as well as the ODEs (2), we therefore have two expressions for the state derivatives:

$$\dot{\mathbf{X}} = \mathbf{F} + \boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_1 \sim \mathcal{N}\left(\boldsymbol{\epsilon}_1; \mathbf{0}, \mathbf{I}\gamma\right),$$

$$\dot{\mathbf{X}} = {}'\mathbf{C}_{\boldsymbol{\phi}}\mathbf{C}_{\boldsymbol{\phi}}^{-1}\,\mathbf{X} + \boldsymbol{\epsilon}_2, \boldsymbol{\epsilon}_2 \sim \mathcal{N}\left(\boldsymbol{\epsilon}_2; \mathbf{0}, \mathbf{A}\right),$$

where $\mathbf{F} := \mathbf{f}(\mathbf{X}, \boldsymbol{\theta})$ and $\mathbf{A} := \mathbf{C}_{\boldsymbol{\phi}}'' - {}'\mathbf{C}_{\boldsymbol{\phi}}\mathbf{C}_{\boldsymbol{\phi}}^{-1}\mathbf{C}_{\boldsymbol{\phi}}'$ and γ is the error variance in the ODEs. Note that, in a deterministic system, the output of the ODEs $\mathbf{F}$ should equal the state derivatives $\dot{\mathbf{X}}$. However, in the first equation above we relax this contraint by adding stochasticity to the state derivatives $\dot{\mathbf{X}}$ in order to compensate for a potential model mismatch. The second equation above is obtained by deriving the conditional distribution for $\dot{\mathbf{X}}$ from the joint distribution in equation (3). Equating the two expressions in the equations above we can eliminate the unknown state derivatives

3.7 Rewrite ODEs as Linear Combination in Parameters

Since, according to the mass action dynamics (equation 2), the ODEs are *linear in the parameters* $\boldsymbol{\theta}$ we can rewrite the ODEs in equation (2) as a linear combination in the parameters:

$$\mathbf{B}_{\boldsymbol{\theta}k}\boldsymbol{\theta} + \mathbf{b}_{\boldsymbol{\theta}k} \overset{!}{=} \mathbf{f}_k(\mathbf{X}, \boldsymbol{\theta}) \qquad (5),$$

where matrices $\mathbf{B}_{\boldsymbol{\theta}k}$ and $\mathbf{b}_{\boldsymbol{\theta}k}$ are defined such that the ODEs $\mathbf{f}_k(\mathbf{X}, \boldsymbol{\theta})$ are expressed as a linear combination in $\boldsymbol{\theta}$.

```
[ode_param.lin_comb.B,ode_param.lin_comb.b] = ...
rewrite_odes_as_linear_combination_in_parameters(ode,symbols);
```

3.8 Posterior over ODE Parameters

Inserting (5) into (4) and solving for $\boldsymbol{\theta}$ yields:

$$\boldsymbol{\theta} = \mathbf{B}_{\boldsymbol{\theta}}^{+}\left({}'\mathbf{C}_{\boldsymbol{\phi}}\mathbf{C}_{\boldsymbol{\phi}}^{-1}\mathbf{X} - \mathbf{b}_{\boldsymbol{\theta}} + \boldsymbol{\epsilon}_0\right),$$

where $\mathbf{B}_{\boldsymbol{\theta}}^{+}$ denotes the pseudo-inverse of $\mathbf{B}_{\boldsymbol{\theta}}$. Since $\mathbf{C}_{\boldsymbol{\phi}}$ is block diagonal we can rewrite the expression above as:

$$\begin{aligned}\boldsymbol{\theta} &= \left(\mathbf{B}_{\boldsymbol{\theta}}^T\mathbf{B}_{\boldsymbol{\theta}}\right)^{-1}\,\mathbf{B}_{\boldsymbol{\theta}}^T\left(\textstyle\sum_k {}'\mathbf{C}_{\boldsymbol{\phi}_k}\mathbf{C}_{\boldsymbol{\phi}_k}^{-1}\mathbf{X}_k - \mathbf{b}_{\boldsymbol{\theta}k} + \boldsymbol{\epsilon}_0^{(k)}\right)\\ &= \left(\mathbf{B}_{\boldsymbol{\theta}}^T\mathbf{B}_{\boldsymbol{\theta}}\right)^{-1}\left(\textstyle\sum_k \mathbf{B}_{\boldsymbol{\theta}k}^T\left({}'\mathbf{C}_{\boldsymbol{\phi}_k}\mathbf{C}_{\boldsymbol{\phi}_k}^{-1}\mathbf{X}_k - \mathbf{b}_{\boldsymbol{\theta}k} + \boldsymbol{\epsilon}_0^{(k)}\right)\right),\end{aligned}$$

where we subsitute the Moore-Penrose inverse for the pseudo-inverse (i.e. $\mathbf{B}_{\boldsymbol{\theta}}^{+} := \left(\mathbf{B}_{\boldsymbol{\theta}}^T\mathbf{B}_{\boldsymbol{\theta}}\right)^{-1}\mathbf{B}_{\boldsymbol{\theta}}^T$). We can therefore derive the posterior distribution over ODE parameters:

$$p(\boldsymbol{\theta} \mid \mathbf{X}, \boldsymbol{\phi}, \gamma) = \mathcal{N}\left(\boldsymbol{\theta}; \left(\mathbf{B}_{\boldsymbol{\theta}}^T\mathbf{B}_{\boldsymbol{\theta}}\right)^{-1}\left(\textstyle\sum_k \mathbf{B}_{\boldsymbol{\theta}k}^T\,\left({}'\mathbf{C}_{\boldsymbol{\phi}k}\mathbf{C}_{\boldsymbol{\phi}k}^{-1}\mathbf{X}_k - \mathbf{b}_{\boldsymbol{\theta}k}\right)\right),\ \mathbf{B}_{\boldsymbol{\theta}}^{+}\left(\mathbf{A} + \mathbf{I}\gamma\right)\mathbf{B}_{\boldsymbol{\theta}}^{+T}\right) \qquad (6).$$

3.9 Rewrite ODEs as Linear Combination in Individual States

Since, according to the mass action dynamics (equation 2), the ODEs are *linear in the individual state* $\mathbf{x}_u$ we can rewrite the ODE $\mathbf{f}_k(\mathbf{X}, \boldsymbol{\theta})$ as a linear combination in the individual state $\mathbf{x}_u$:

$$\mathbf{R}_{uk}\mathbf{x}_u + \mathbf{r}_{uk} \stackrel{!}{=} \mathbf{f}_k(\mathbf{X}, \boldsymbol{\theta}),$$

where matrices $\mathbf{R_{uk}}$ and $\mathbf{r}_{uk}$ are defined such that the ODE $\mathbf{f}_k(\mathbf{X}, \boldsymbol{\theta})$ is expressed as a linear combination in the individual state $\mathbf{x}_u$.

```
[state.lin_comb.R,state.lin_comb.r] = ...
rewrite_odes_as_linear_combination_in_ind_states(ode,symbols,coupling_idx.states);
```

3.10 Posterior over Individual States

Given the linear combination of the ODEs w.r.t. an individual state, we define the matrices $\mathbf{B}_u$ and $\mathbf{b}_u$ such that the expression $\mathbf{f}(\mathbf{X}, \boldsymbol{\theta}) - {}'\mathbf{C}_{\boldsymbol{\phi}}\mathbf{C}_{\boldsymbol{\phi}}^{-1}\mathbf{X}$ is rewritten as a linear combination in an individual state $\mathbf{x}_u$:

$$\mathbf{B}_u\mathbf{x}_u + \mathbf{b}_u \stackrel{!}{=} \mathbf{f}(\mathbf{X}, \boldsymbol{\theta}) - {}'\mathbf{C}_{\boldsymbol{\phi}}\mathbf{C}_{\boldsymbol{\phi}}^{-1}\mathbf{X} \qquad (7).$$

Inserting (7) into (4) and solving for $\mathbf{x}_u$ yields:

$$\mathbf{x}_u = \mathbf{B}_u^{+}\left(\boldsymbol{\epsilon}_0 - \mathbf{b}_u\right),$$

where $\mathbf{B}_u^{+}$ denotes the pseudo-inverse of $\mathbf{B}_u$. Since $\mathbf{C}_{\boldsymbol{\phi}}$ is block diagonal we can rewrite the expression above as:

$$\begin{aligned}\mathbf{x}_u &= \left(\mathbf{B}_u\mathbf{B}_u^T\right)^{-1}\mathbf{B}_u^T\textstyle\sum_k\left(\boldsymbol{\epsilon}_0^{(k)} - \mathbf{b}_{uk}\right)\\ &= \left(\mathbf{B}_u\mathbf{B}_u^T\right)^{-1}\textstyle\sum_k\mathbf{B}_{uk}^T\left(\boldsymbol{\epsilon}_0^{(k)} - \mathbf{b}_{uk}\right),\end{aligned}$$

where we subsitute the Moore-Penrose inverse for the pseudo-inverse (i.e. $\mathbf{B}_{\boldsymbol{\theta}}^{+} := \left(\mathbf{B}_{\boldsymbol{\theta}}^T\mathbf{B}_{\boldsymbol{\theta}}\right)^{-1}\mathbf{B}_{\boldsymbol{\theta}}^T$). We can therefore derive the posterior distribution over an individual state $\mathbf{x}_u$:

$$p(\mathbf{x}_u \mid \mathbf{X}_{-u}, \boldsymbol{\phi}, \gamma) = \mathcal{N}\left(\mathbf{x}_u; \left(\mathbf{B}_u\mathbf{B}_u^T\right)^{-1}\left(-\textstyle\sum_k\mathbf{B}_{uk}^T\mathbf{b}_{uk}\right),\ \mathbf{B}_u^{+}\left(\mathbf{A} + \mathbf{I}\gamma\right)\mathbf{B}_u^{+T}\right) \qquad (8),$$

with $\mathbf{X}_{-u}$ denoting the set of all states except state $\mathbf{x}_u$.

3.11 Mean-field Variational Inference

To infer the parameters $\boldsymbol{\theta}$, we want to find the maximum a posteriori estimate (MAP):

$$\begin{aligned}&\boldsymbol{\theta}^* := \arg\max_{\boldsymbol{\theta}}\ \ln p(\boldsymbol{\theta} \mid \mathbf{Y}, \boldsymbol{\phi}, \gamma, \sigma)\\ &= \arg\max_{\boldsymbol{\theta}}\ \ln\textstyle\int p(\boldsymbol{\theta}, \mathbf{X} \mid \mathbf{Y}, \boldsymbol{\phi}, \gamma, \sigma)d\mathbf{X}\\ &= \arg\max_{\boldsymbol{\theta}}\ \ln\textstyle\int p(\boldsymbol{\theta} \mid \mathbf{X}, \boldsymbol{\phi}, \gamma)p(\mathbf{X} \mid \mathbf{Y}, \boldsymbol{\phi}, \sigma)d\mathbf{X} \qquad (9).\end{aligned}$$

However, the integral above is intractable due to the strong couplings induced by the nonlinear ODEs $\mathbf{f}$ which appear in the term $p(\boldsymbol{\theta} \mid \mathbf{X}, \boldsymbol{\phi}, \gamma)$.

We use mean-field variational inference to establish variational lower bounds that are analytically tractable by decoupling state variables from the ODE parameters as well as decoupling the state variables from each

other. Note that, since the ODEs described by equation (2) are *locally linear*, both conditional distributions $p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{Y}, \boldsymbol{\phi}, \gamma, \sigma)$ (equation (6)) and $p(\mathbf{x}_u \mid \boldsymbol{\theta}, \mathbf{X}_{-u}, \mathbf{Y}, \boldsymbol{\phi}, \gamma, \sigma)$ (equation (8)) are analytically tractable and Gaussian distributed as mentioned previously. The decoupling is induced by designing a variational distribution $Q(\boldsymbol{\theta}, \mathbf{X})$ which is restricted to the family of factorial distributions:

$$\mathcal{Q} := \left\{ Q : Q(\boldsymbol{\theta}, \mathbf{X}) = q(\boldsymbol{\theta}) \prod_u q(\mathbf{x}_u) \right\}.$$

The particular form of $q(\boldsymbol{\theta})$ and $q(\mathbf{x}_u)$ are designed to be Gaussian distributed which places them in the same family as the true full conditional distributions. To find the optimal factorial distribution we minimize the Kullback-Leibler divergence between the variational and the true posterior distribution:

$$\hat{Q} := \arg\min_{Q(\boldsymbol{\theta},\mathbf{X}) \in \mathcal{Q}} \mathrm{KL}\left[Q(\boldsymbol{\theta}, \mathbf{X}) \mid\mid p(\boldsymbol{\theta}, \mathbf{X} \mid \mathbf{Y}, \boldsymbol{\phi}, \gamma, \sigma) \right] \qquad (10),$$

where $\hat{Q}$ is the proxy distribution. The proxy distribution that minimizes the KL-divergence (10) depends on the true full conditionals and is given by:

$$\hat{q}(\boldsymbol{\theta}) \propto \exp\left(E_{Q_{-\boldsymbol{\theta}}} \ln p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{Y}, \boldsymbol{\phi}, \gamma, \sigma) \right) \qquad (11)$$
$$\hat{q}(\mathbf{x}_u) \propto \exp\left(E_{Q_{-u}} \ln p(\mathbf{x}_u \mid \boldsymbol{\theta}, \mathbf{X}_{-u}, \mathbf{Y}, \boldsymbol{\phi}, \gamma, \sigma) \right) \qquad (12).$$

3.12 Fitting Observations of State Trajectories

We fit the observations of state trajectories by standard GP regression. The data-informed distribution$ $p(\mathbf{X} \mid \mathbf{Y}, \boldsymbol{\phi}, \sigma)$ in euqation (9) can be determined analytically using Gaussian process regression with the GP prior

$$p(\mathbf{X} \mid \phi) = \prod_k \mathcal{N}(\mathbf{x}_k; \mathbf{0}, \mathbf{C}_{\boldsymbol{\phi}_k})$$

:

$p(\mathbf{X} \mid \mathbf{Y}, \boldsymbol{\phi}, \gamma) = \prod_k \mathcal{N}(\mathbf{x}_k; \mu_k(\mathbf{y}_k), \sigma_k)$,

where $\mu_k(\mathbf{y}_k) := \sigma_k^{-2} \left(\sigma_k^{-2}\mathbf{I} + \mathbf{C}_{\boldsymbol{\phi}_k}^{-1} \right)^{-1} \mathbf{y}_k$ and $\sigma_k^{-1} := \sigma_k^{-2}\mathbf{I} + \mathbf{C}_{\phi_k}^{-1}$.

```
[mu,inv_sigma] = fitting_state_observations(inv_C,obs_to_state_relation,...
simulation,symbols);
```

3.13 Coordinate Ascent Variational Gradient Matching

We minimize the KL-divergence in equation (10) by coordinate descent (where each step is analytically tractable) by iterating between determining the proxy for the distribution over ODE parameters $\hat{q}(\boldsymbol{\theta})$ and the proxies for the distribution over individual states $\hat{q}(\mathbf{x}_u)$.

Initialize the state estimation by the GP regression posterior

```
state.proxy.mean = array2table([time.est',mu],...
'VariableNames',['time',symbols.state_string]);
```

Coordinate ascent

```
for i = 1:opt_settings.coord_ascent_numb_iter
```

3.13.1 Proxy for ODE Parameters

Expanding the proxy distribution in equation (11) for $\boldsymbol{\theta}$ yields:

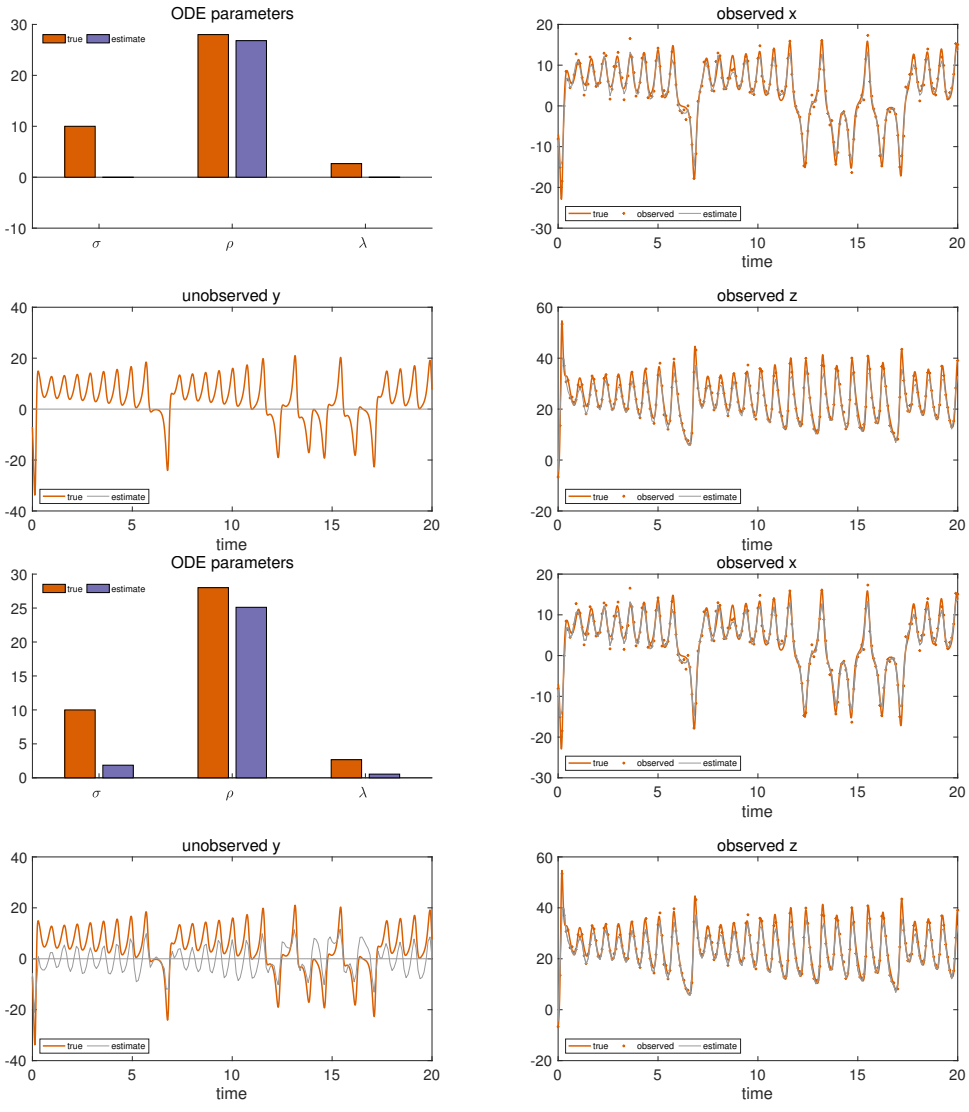
$$\hat{q}(\boldsymbol{\theta}) \propto \exp\left(E_{Q_{-\boldsymbol{\theta}}} \ln p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{Y}, \boldsymbol{\phi}, \gamma, \sigma) \right)$$
$$= \exp\left(E_{Q_{-\boldsymbol{\theta}}} \ln \mathcal{N}\left(\boldsymbol{\theta}; \left(\mathbf{B}_{\boldsymbol{\theta}}^T \mathbf{B}_{\boldsymbol{\theta}}\right)^{-1} \left(\sum_k \mathbf{B}_{\boldsymbol{\theta}k}^T \left('\mathbf{C}_{\boldsymbol{\phi}k} \mathbf{C}_{\boldsymbol{\phi}k}^{-1} \mathbf{X}_k - \mathbf{b}_{\boldsymbol{\theta}k}\right)\right), \mathbf{B}_{\boldsymbol{\theta}}^{+} \left(\mathbf{A} + \mathbf{I}\gamma\right) \mathbf{B}_{\boldsymbol{\theta}}^{+T}\right) \right),$$

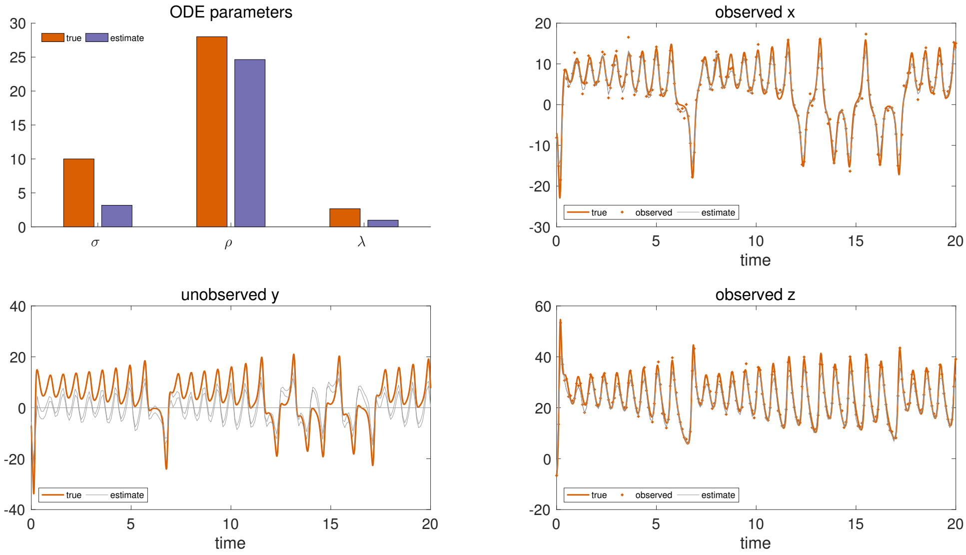
where we substitute $p(\boldsymbol{\theta} \mid \mathbf{X}, \boldsymbol{\phi}, \gamma)$ with its density given in equation (6).

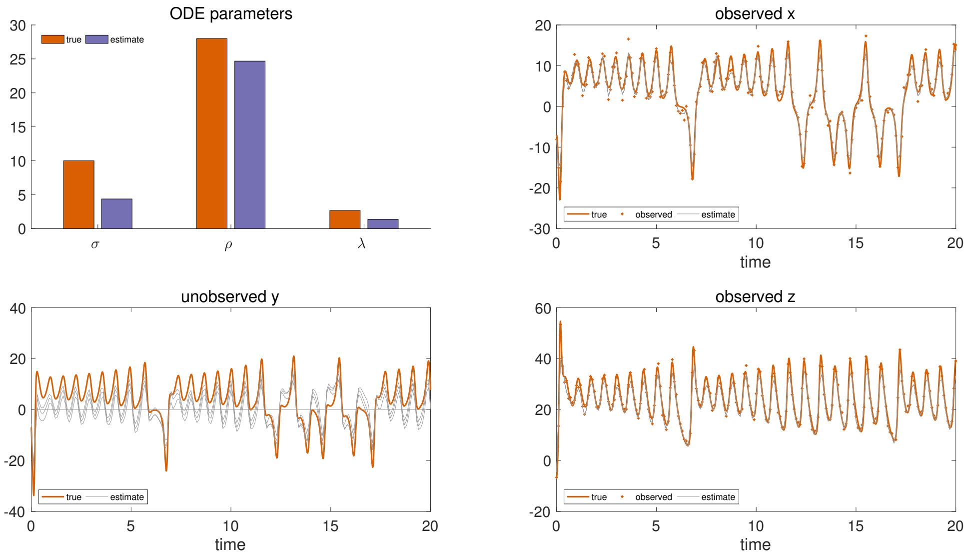
```
[param_proxy_mean,param_proxy_inv_cov] = ...
proxy_for_ode_parameters(state.proxy.mean{:,symbols.state_string},...
dC_times_invC,ode_param.lin_comb,symbols,A_plus_gamma_inv,opt_settings);
```

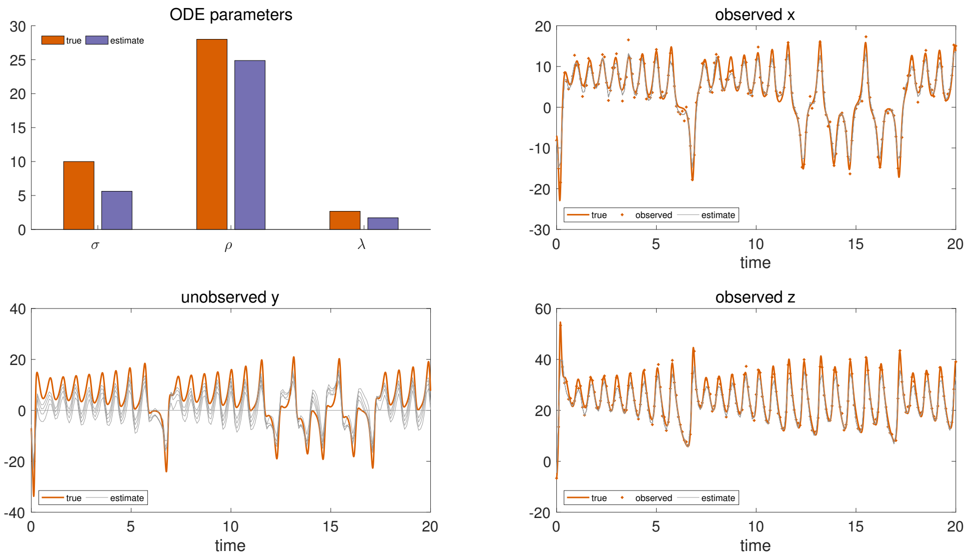
Intermediate results

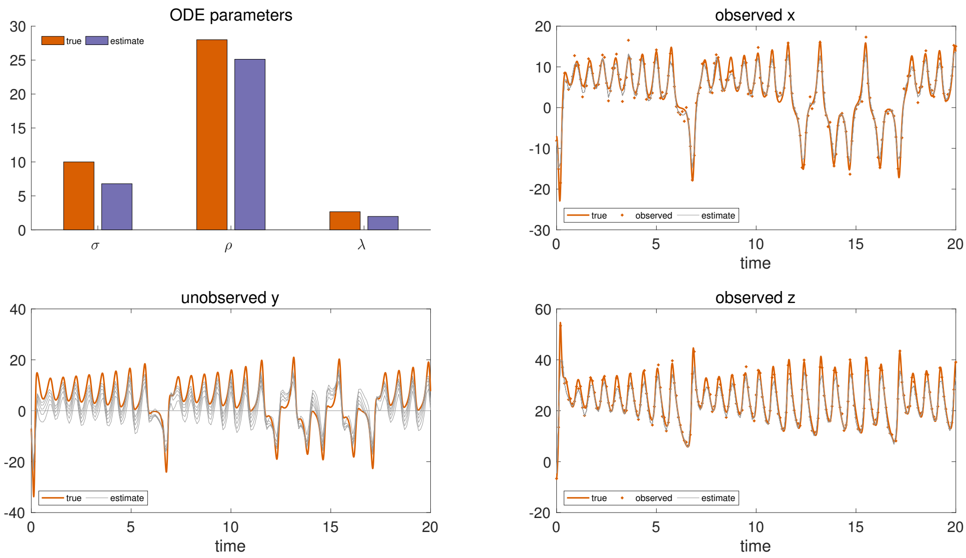
```
if i==1 || ~mod(i,1)
    plot_results(fig_handle,state.proxy,simulation,param_proxy_mean,...
    plot_handle,symbols,plot_settings,'not_final');
end
```

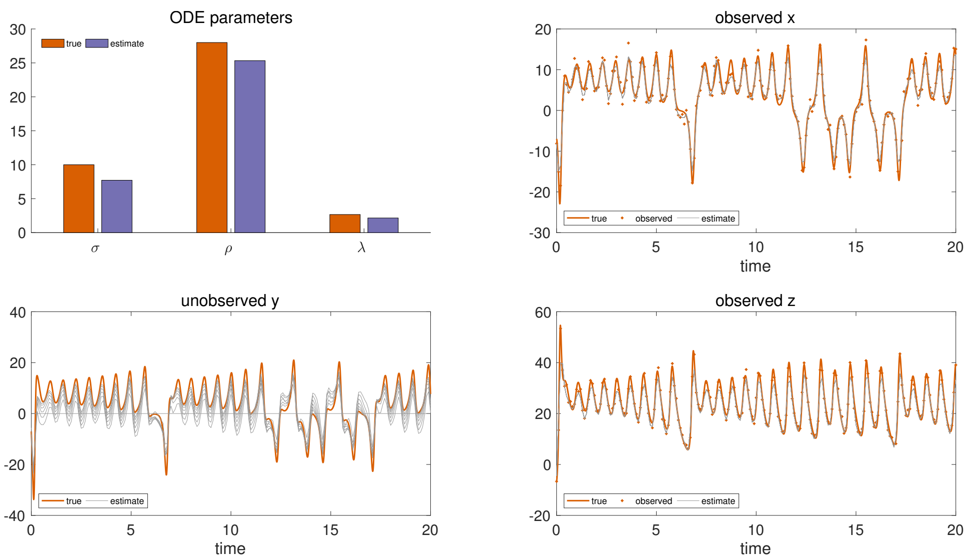


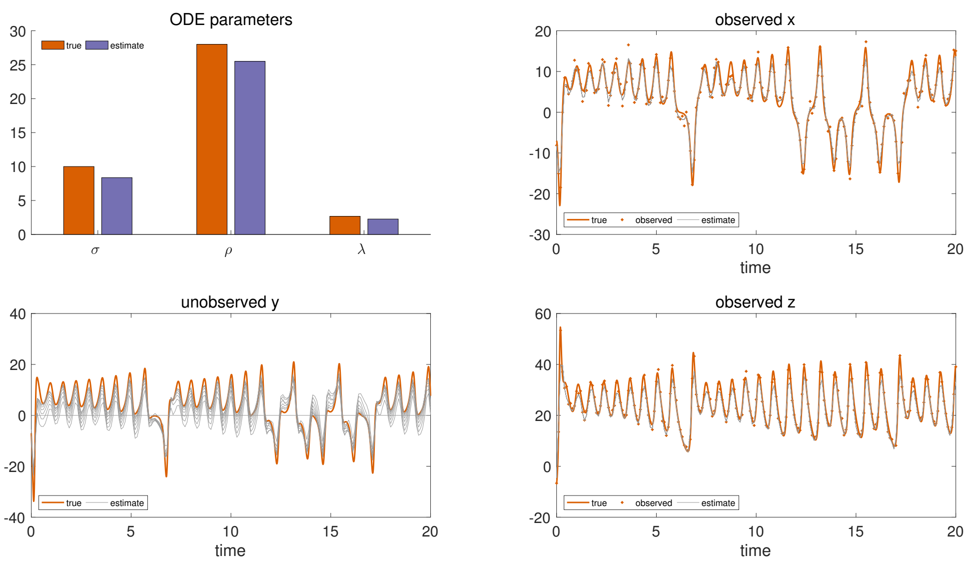

ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time

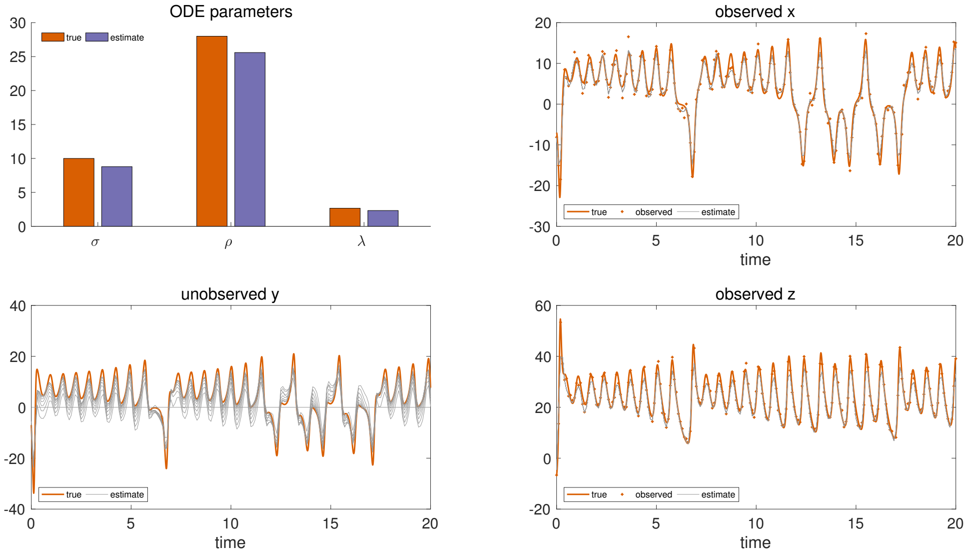

ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time

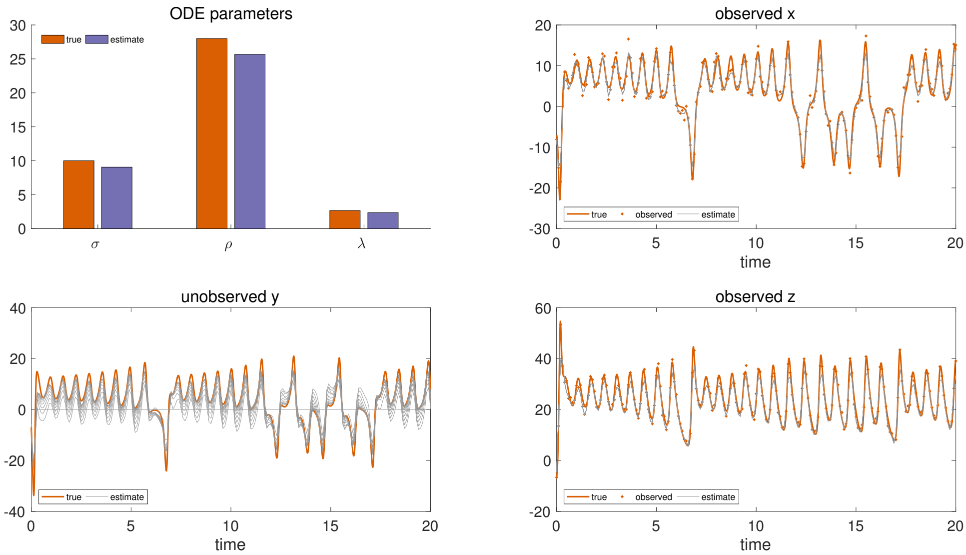

ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time

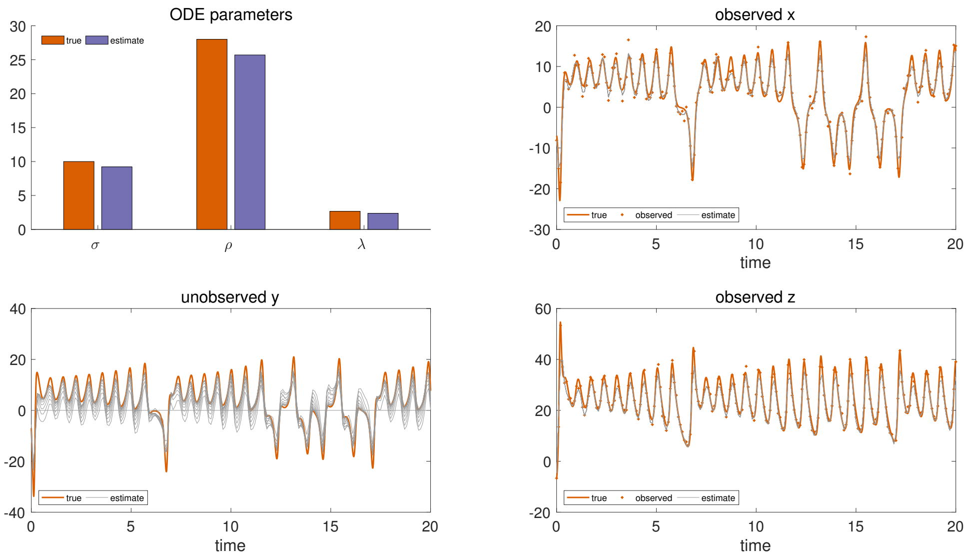

ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time

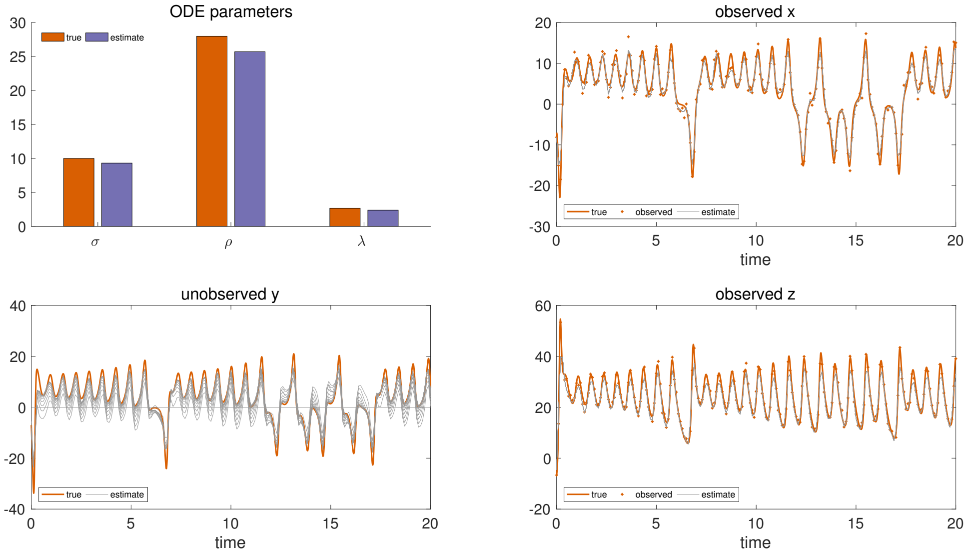

ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time

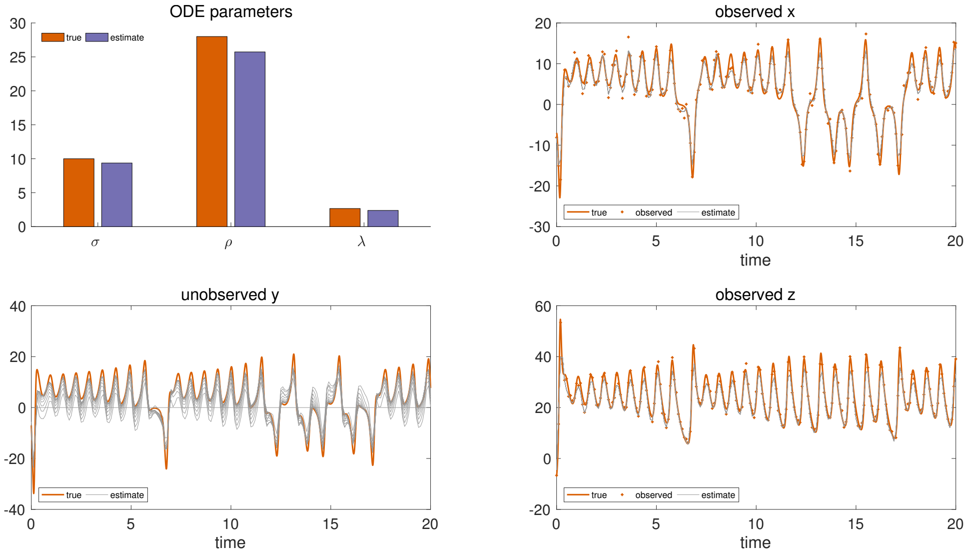

ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time

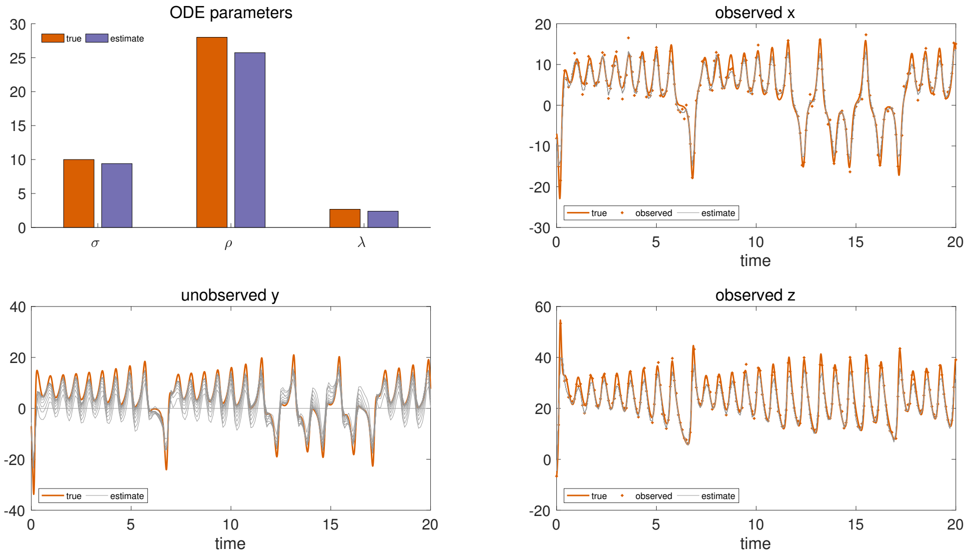

ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time

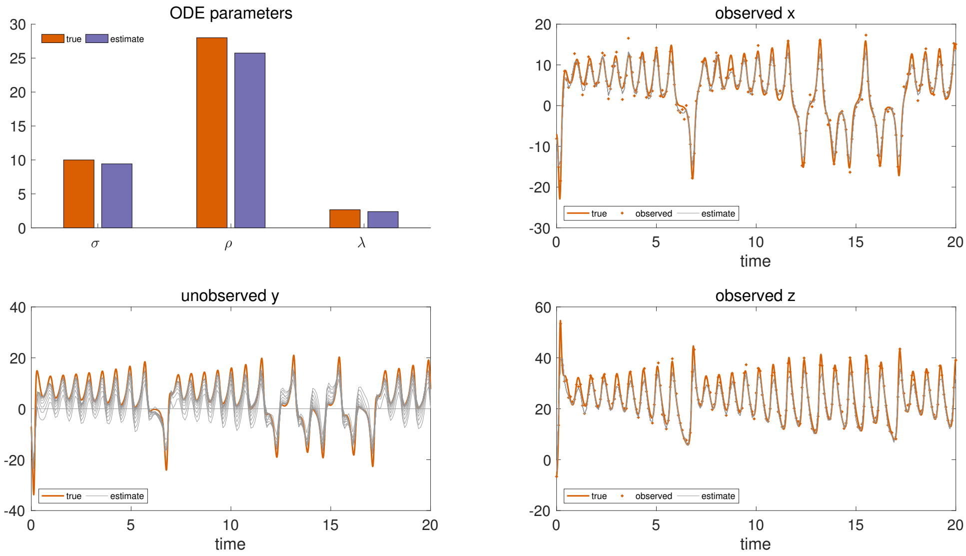

ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time

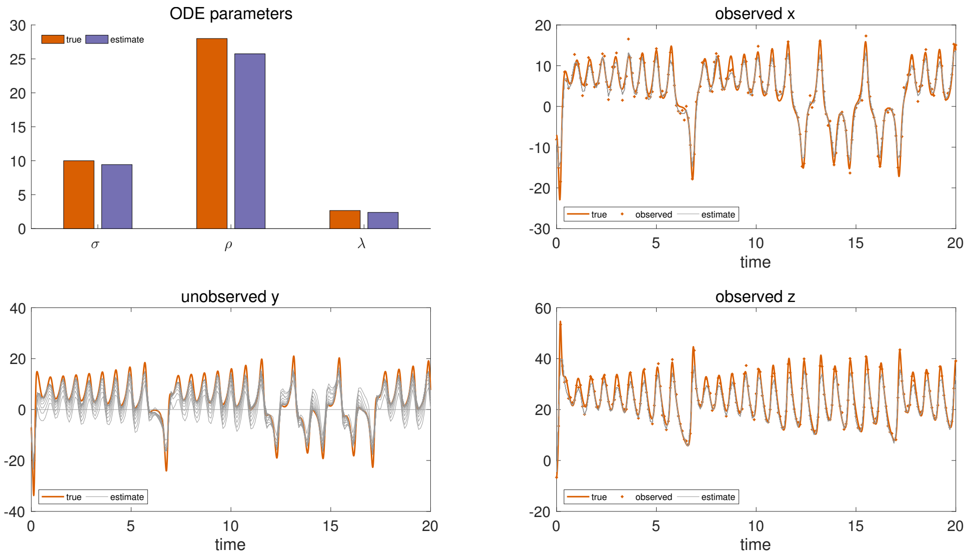

ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time

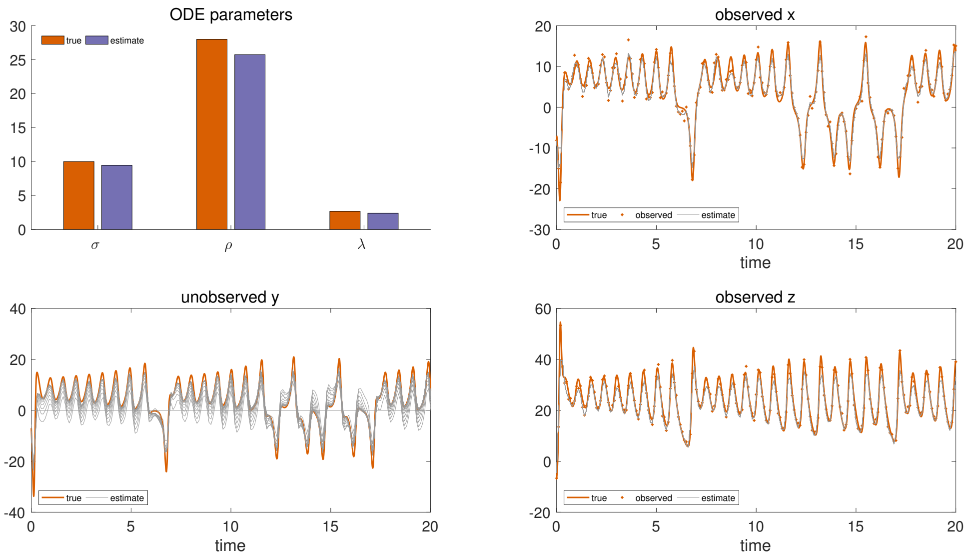

ODE parameters
true
estimate
σ
ρ
λ
observed x
observed
time
unobserved y
observed z

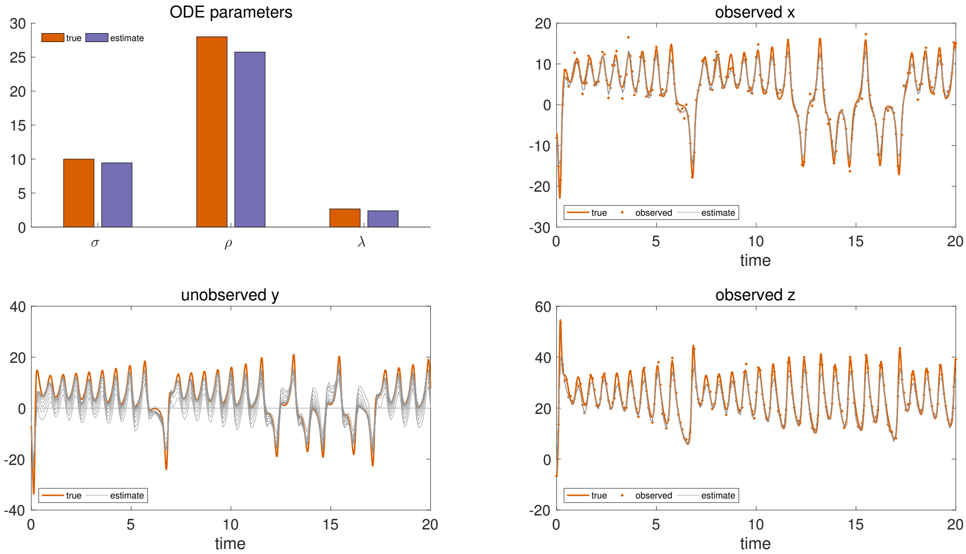

ODE parameters
true
estimate
σ
ρ
λ
observed x
observed
time
unobserved y
observed z

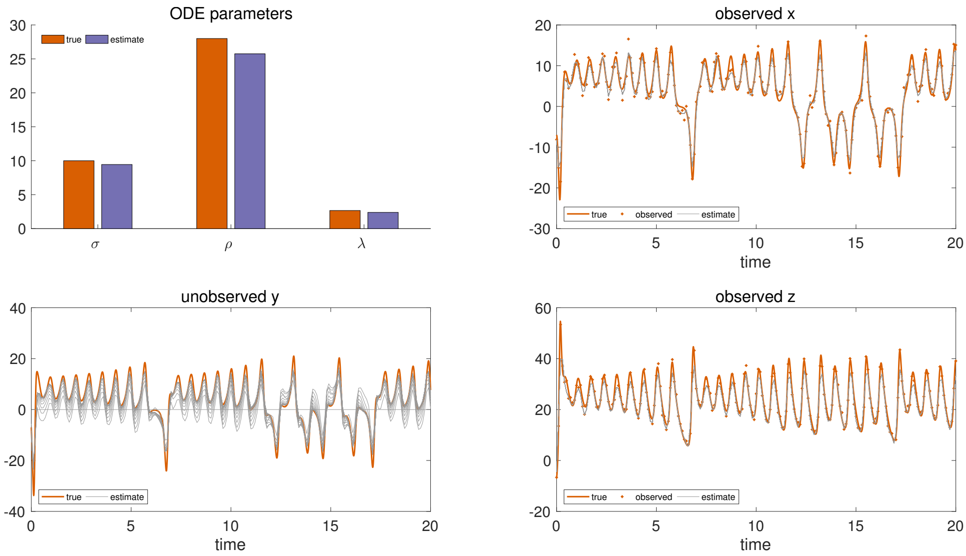

ODE parameters
true
estimate
σ
ρ
λ
observed x
observed
time
unobserved y
observed z

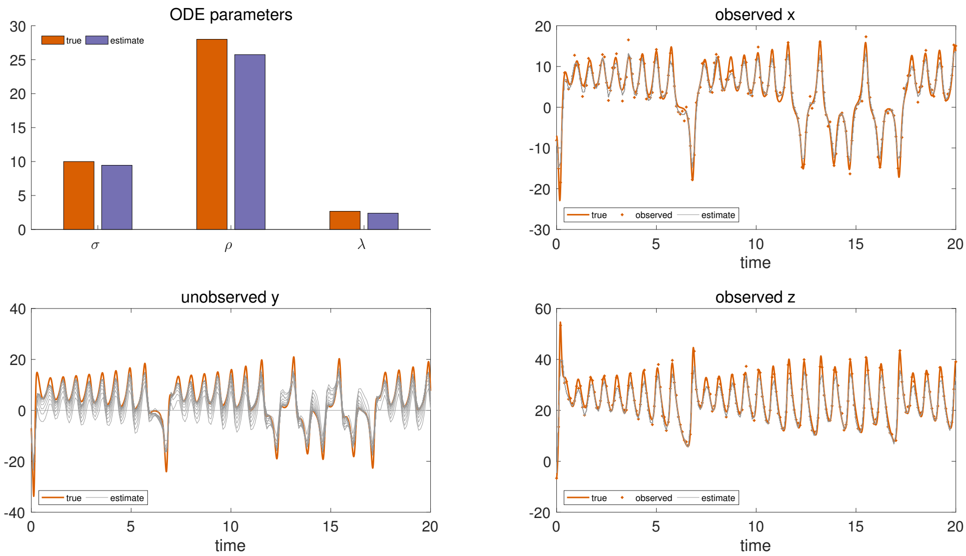

ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time

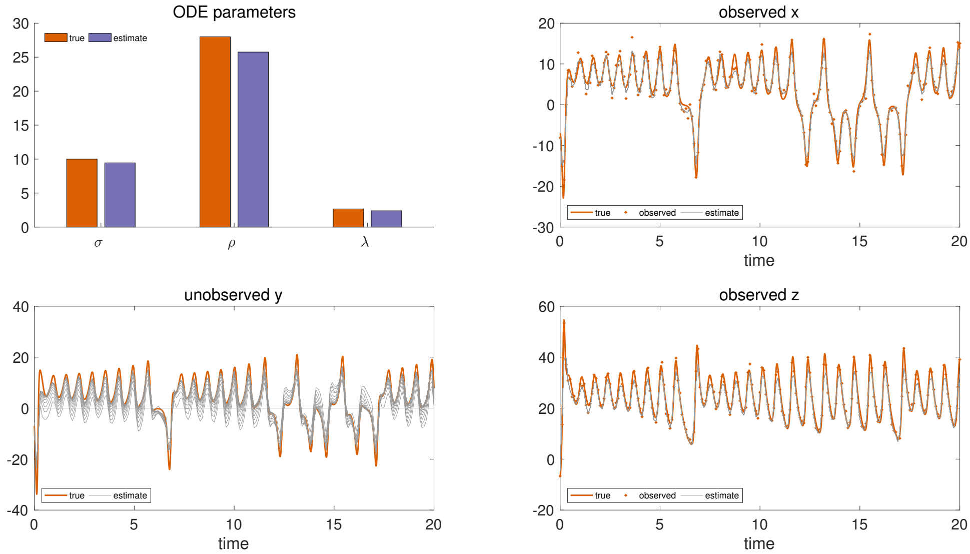

ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time


ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time

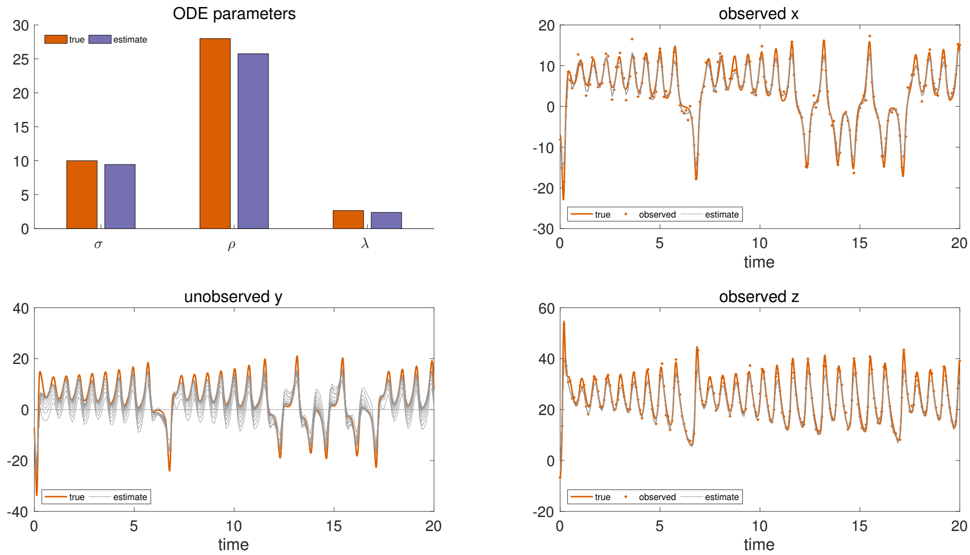

ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time


ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time

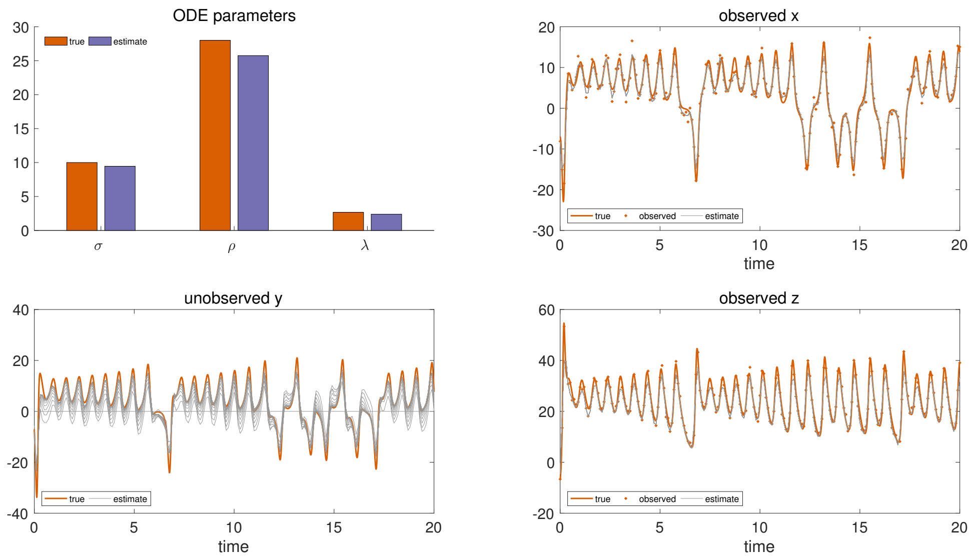

ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time



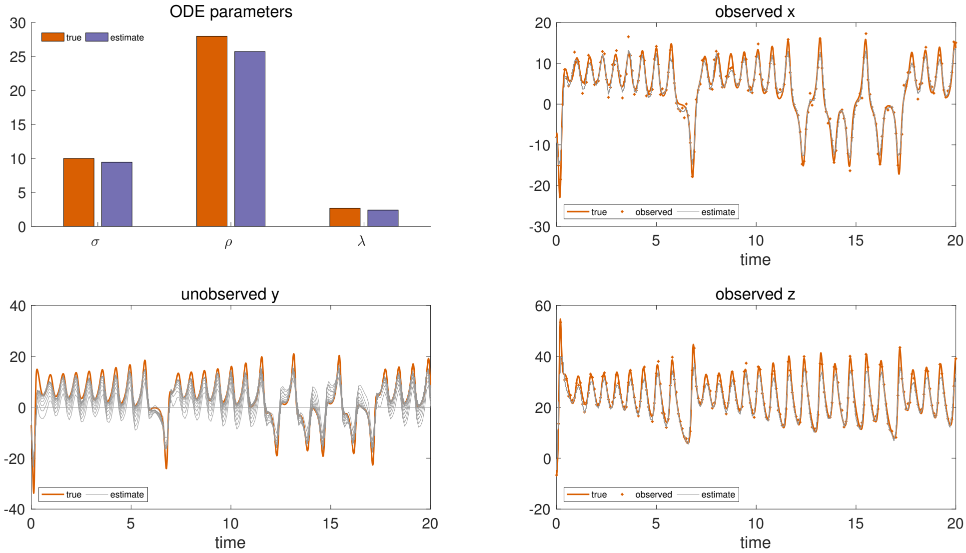
ODE parameters
true
estimate
σ
ρ
λ
observed x
unobserved y
observed z
true
observed
estimate
time

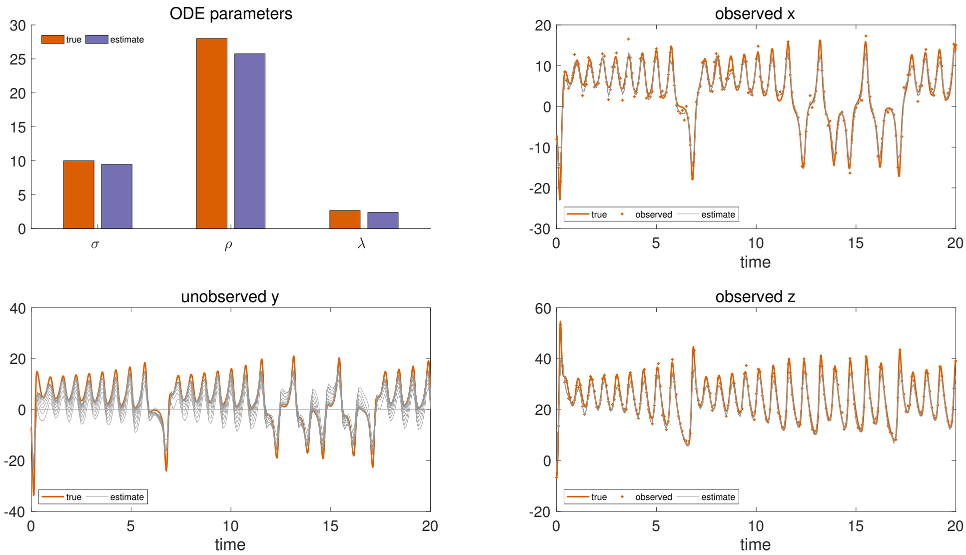


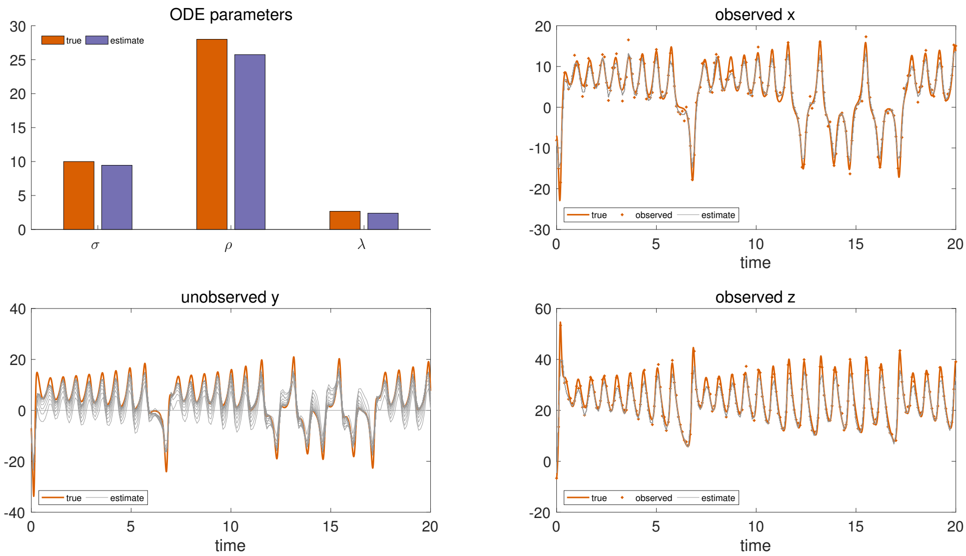
ODE parameters
true
estimate
σ
ρ
λ
observed x
unobserved y
observed z
true
observed
estimate
time

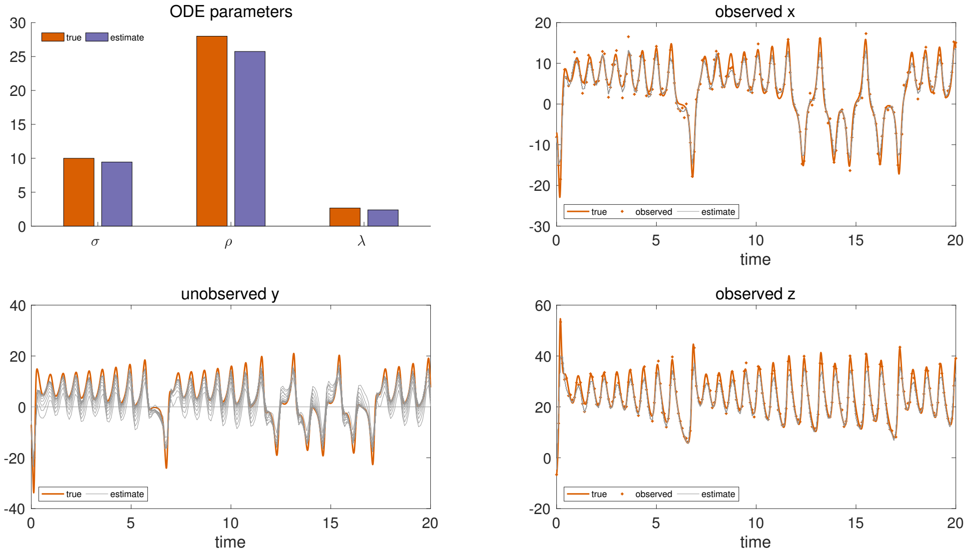


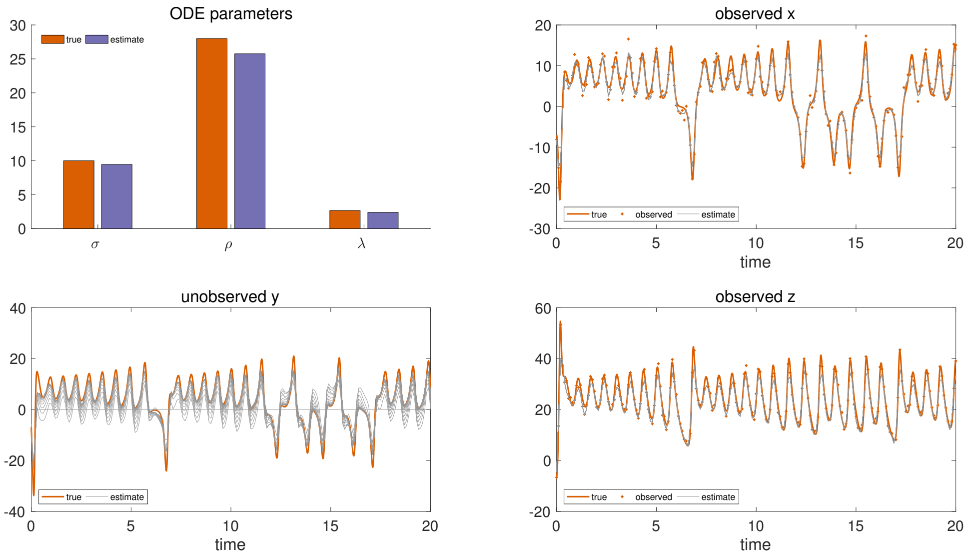
ODE parameters
true
estimate
σ
ρ
λ
observed x
unobserved y
observed z
true
observed
estimate
time

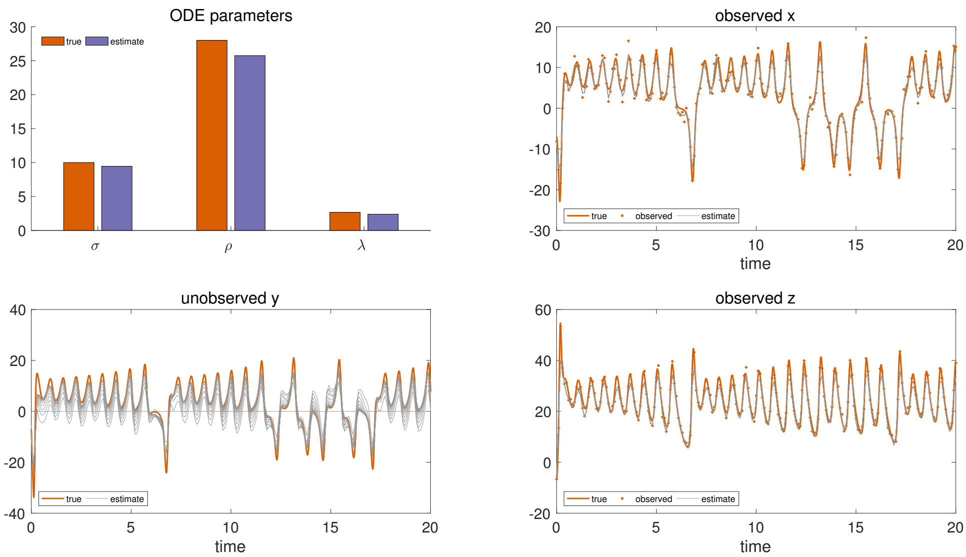

ODE parameters
true
estimate
σ
ρ
λ
observed x
unobserved y
observed z
true
observed
estimate
time

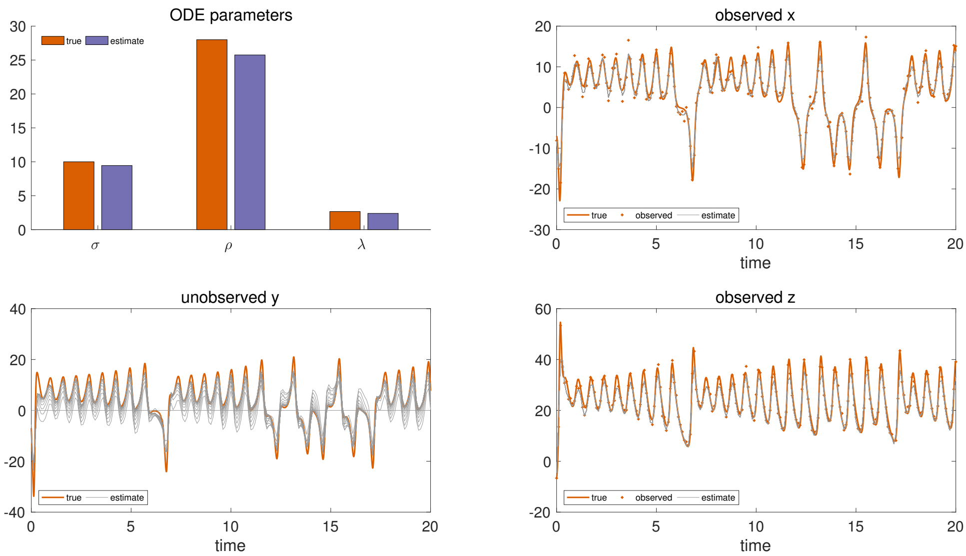

ODE parameters
true
estimate
σ
ρ
λ
observed x
unobserved y
observed z
true
observed
estimate
time

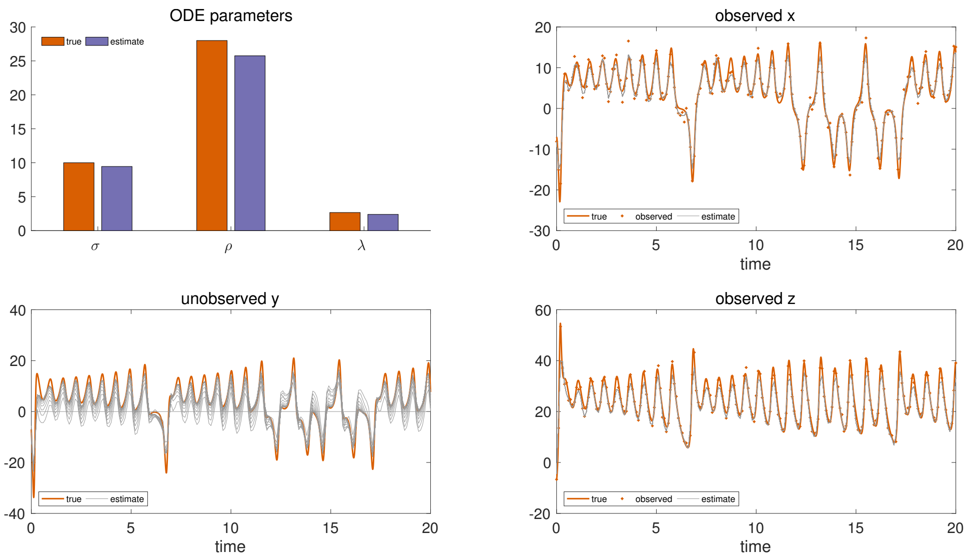

ODE parameters
true
estimate
σ
ρ
λ
observed x
unobserved y
observed z
true
observed
estimate
time

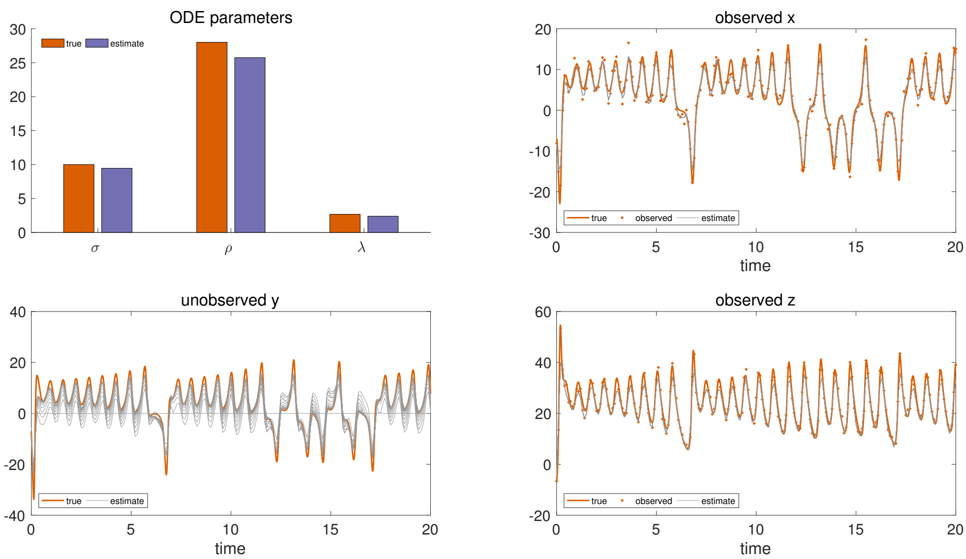

ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time

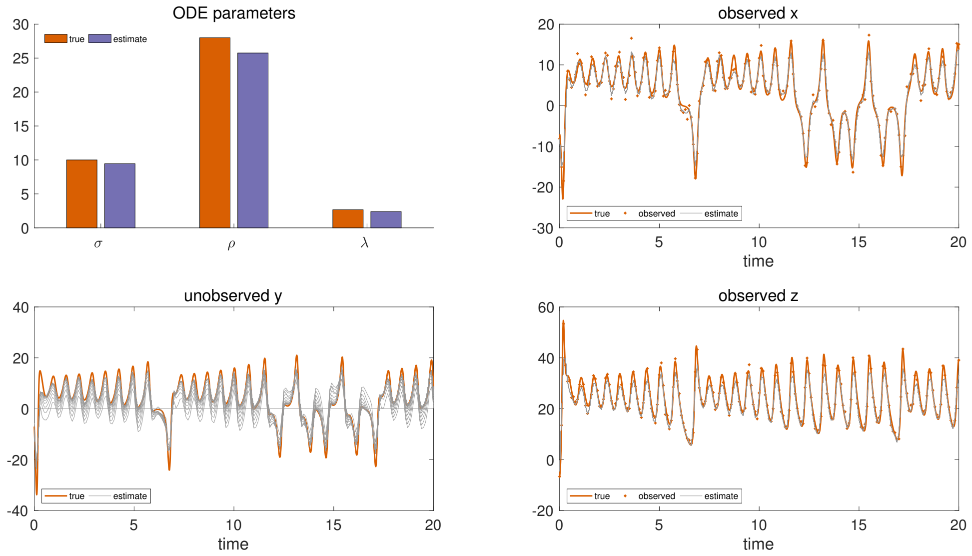

ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time

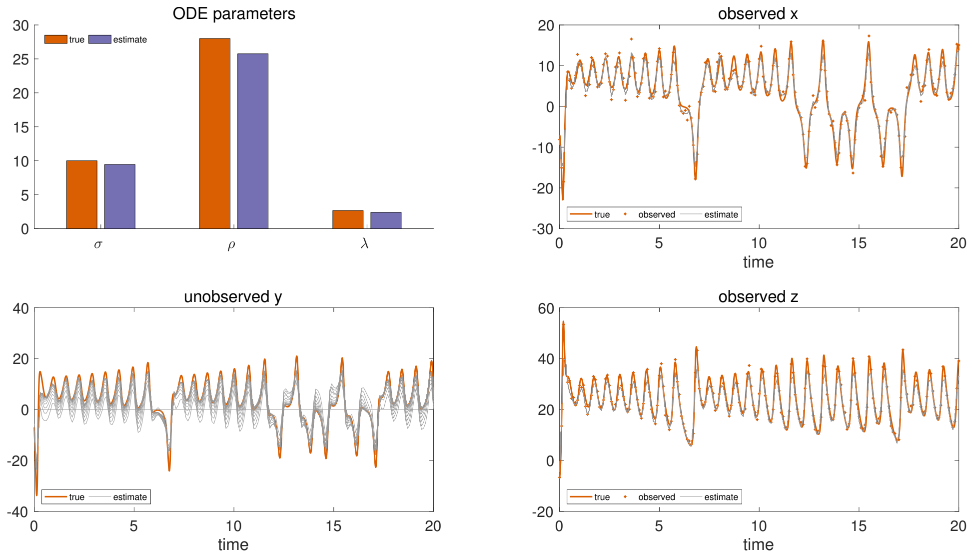

ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time

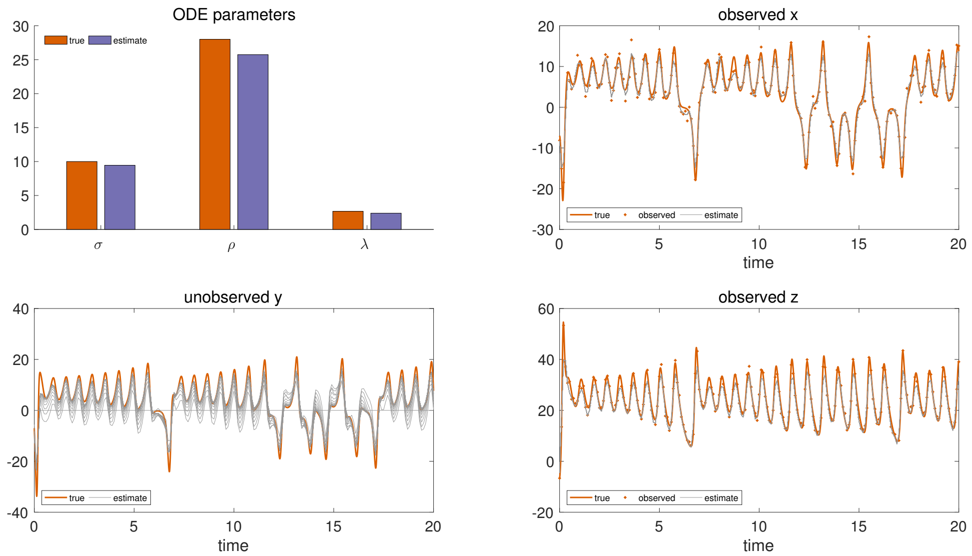

ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time

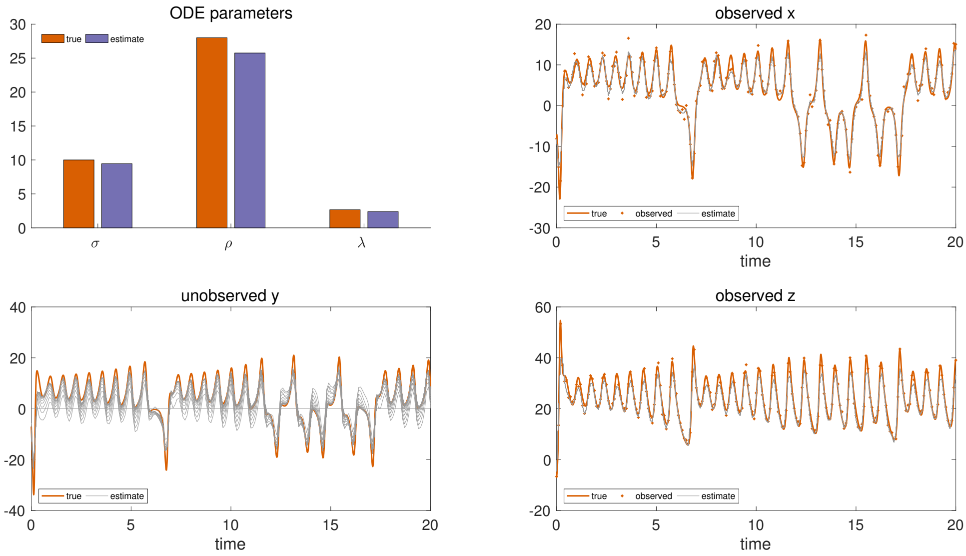

ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time

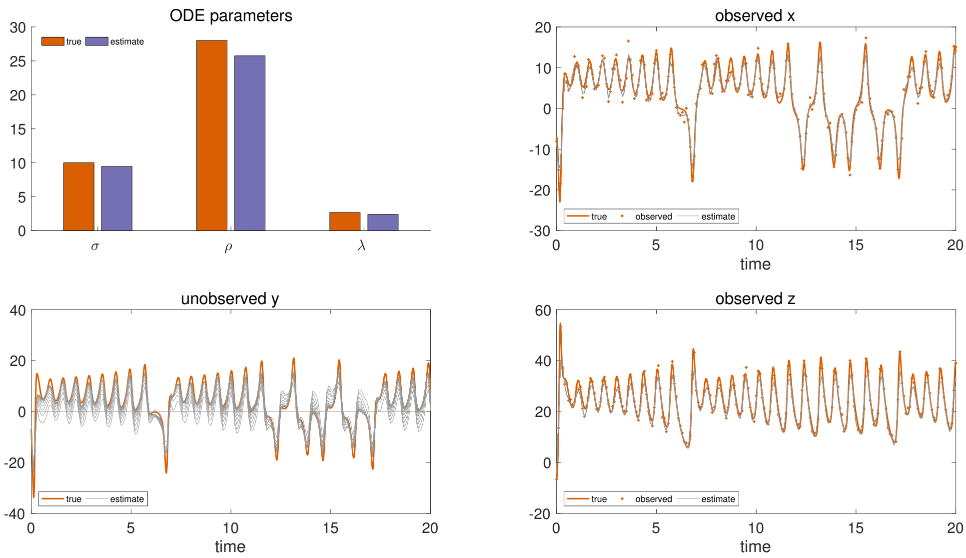

ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time

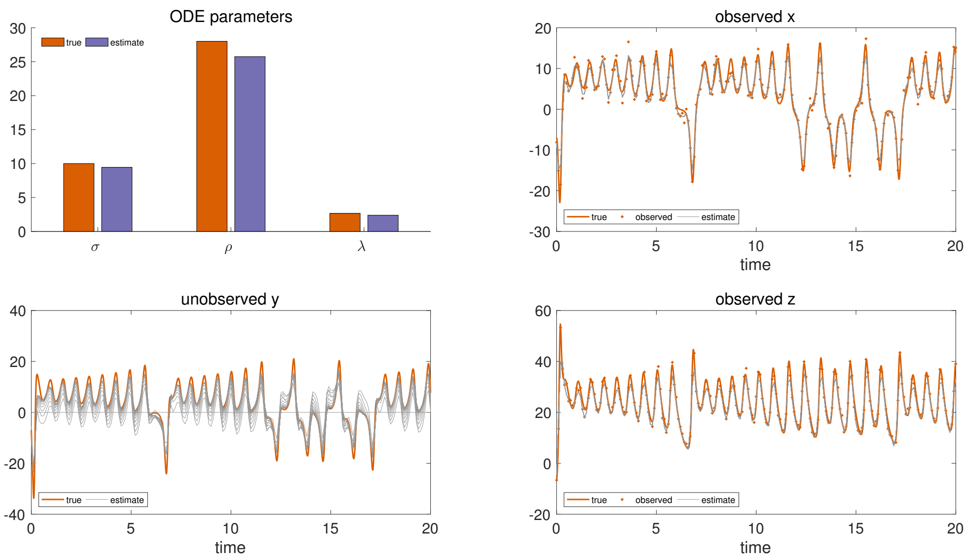

ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time

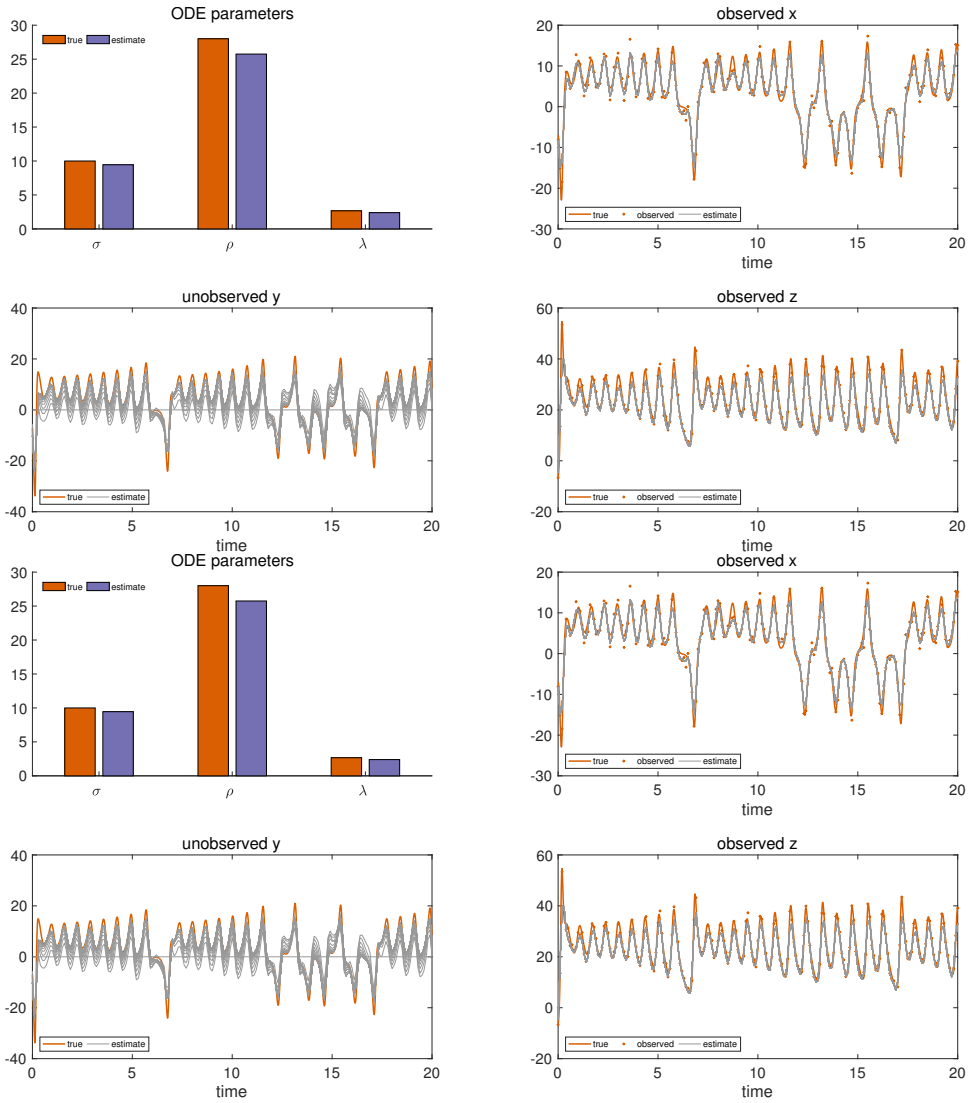

ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time


ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time


ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time


ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time


ODE parameters
true
estimate
σ
ρ
λ
observed x
true
observed
estimate
time
unobserved y
true
estimate
time
observed z
true
observed
estimate
time



3.13.2 Proxy for Individual States

Expanding the proxy distribution in equation (12) over the individual state $\mathbf{x}_u$:

$$\hat{q}(\mathbf{x}_u) \overset{(a)}{\propto} \exp\left(E_{Q_{-u}} \ln(p(\mathbf{x}_u \mid \boldsymbol{\theta}, \mathbf{X}_{-u}, \boldsymbol{\phi}, \gamma) p(\mathbf{x}_u \mid \mathbf{Y}, \boldsymbol{\phi}, \sigma)) \right)$$
$$\overset{(b)}{=} \exp\left(E_{Q_{-u}} \ln \mathcal{N}\left(\mathbf{x}_u; -\mathbf{B}_u^{+}\mathbf{b}_u,\ \mathbf{B}_u^{+}\left(\mathbf{A} + \mathbf{I}\gamma\right)\mathbf{B}_u^{+T}\right) + E_{Q_{-u}} \ln \mathcal{N}\left(\mathbf{x}_u; \mu_u(\mathbf{Y}), \boxtimes_u\right) \right)$$
$$= \exp\left(E_{Q_{-u}} \ln \mathcal{N}\left(\mathbf{x}_u; -\mathbf{B}_u^{+}\mathbf{b}_u,\ \mathbf{B}_u^{+}\left(\mathbf{A} + \mathbf{I}\gamma\right)\mathbf{B}_u^{+T}\right) + E_{Q_{-u}} \ln \mathcal{N}\left(\mathbf{x}_u; \mu_u(\mathbf{Y}), \sigma_u\right) \right).$$

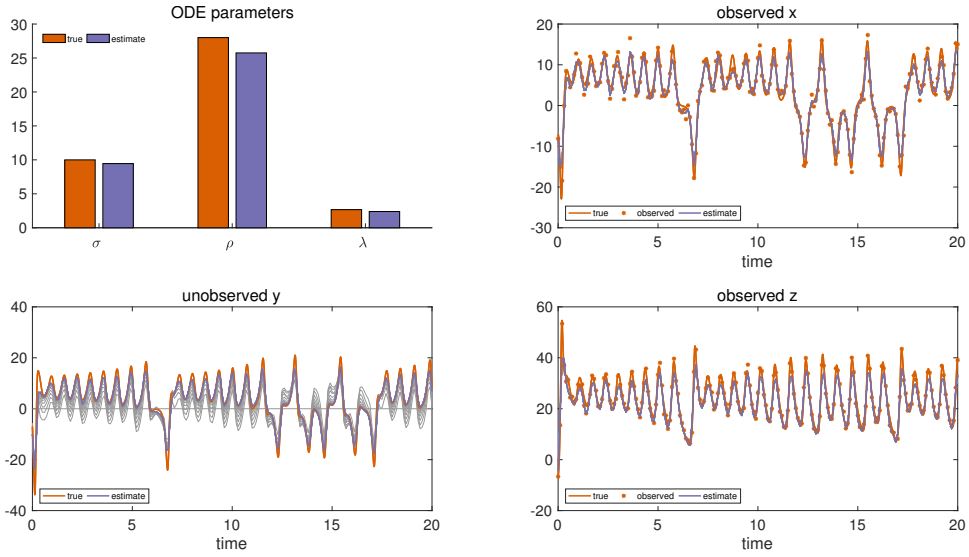
In (a) we decompose the full conditional nto an ODE-informed distribution and a data-informed distribution and in (b) we substitute the ODE-informed distribution $p(\mathbf{x}_u \mid \boldsymbol{\theta}, \mathbf{X}_{-u}, \boldsymbol{\phi}, \gamma)$ with its density given by equation (8).

```
[state.proxy.mean{:,symbols.state_string},state.proxy.inv_cov] = ...
proxy_for_ind_states(state.lin_comb,...
state.proxy.mean{:,symbols.state_string},...
param_proxy_mean',dC_times_invC,coupling_idx.states,...
symbols,mu,inv_sigma,simulation.observed_states,...
A_plus_gamma_inv,opt_settings);


end
```

Final results

```
plot_results(fig_handle,state.proxy,simulation,param_proxy_mean,...
    plot_handle,symbols,plot_settings,'final');
```



3.14 Time Taken

```
disp(['time taken: ' num2str(toc) ' seconds'])
```

```
time taken: 49.3925 seconds
```

VGM for Dynamic Causal Modeling

Example dynamical system used in this code: Dynamic Causal Modeling (**visual attention system**) with **three hidden neuronal- and 12 hidden hemodynamic states**. The system is affected by **given external inputs** and the states are only **indirectly observed** through the BOLD signal change equation.

4.1 User Input

4.1.1 Simulation Settings

- **Simulation ODEs**: Input the ODEs "type" used to generate the data as a string. Options:

```
 'nonlinear_forward_modulation_by_attention',
'forward_modulation_and_driven_by_attention',
'forward_modulation_by_attention', 'backward_modulation_by_attention',
'backward_modulation_and_driven_by_attention', 'absent_modulation',
'absent_attention_input', 'absent_photic_input', 'driven_by_attention',
'photic_input'.
```

```
simulation.odes = 'nonlinear_forward_modulation_by_attention';
```

- **Observed States**: Input a cell vector containing the symbols (characters) in the '*ODEs.txt' file. Eg: to observe deoxyhemoglobin content, blood volume and blood flow set

```
simulation.observed_states =   {'q_1','q_3','q_2','v_1','v_3','v_2',...
  'f_1','f_3','f_2'}).
```

```
simulation.observed_states = {};
```

- **Final time for simulation**: Input a positive real number:

```
simulation.final_time = 359*3.22;
```

- **Observation noise**: Input a function handle:

```
simulation.state_obs_variance = @(x)(repmat(bsxfun(@rdivide,var(x),5),...
size(x,1),1));
```

- **Time interval between observations**: Input a positive real number:

```
simulation.interval_between_observations = 0.1;
```

4.1.2 Estimation Settings

- **Candidate ODEs**: Input the ODEs "type" used to generate the data as a string. Options:

```
 'nonlinear_forward_modulation_by_attention',
'forward_modulation_and_driven_by_attention',
'forward_modulation_by_attention', 'backward_modulation_by_attention',
'backward_modulation_and_driven_by_attention', 'absent_modulation',
'absent_attention_input', 'absent_photic_input', 'driven_by_attention',
'photic_input'.
```

```
candidate_odes = 'nonlinear_forward_modulation_by_attention';
```

- **Kernel parameters ϕ**: Input a row vector of positive real numbers of size 1 x 2:

```
kernel.param = [10,0.2];
```

- **Error variance on state derivatives (i.e. γ)**: Input a row vector of positive real numbers of size 1 x number of ODEs:

```
state.derivative_variance = 6.*ones(11-3,1);
```

- **Estimation times**: Input a row vector of positive real numbers in ascending order:

```
time.est = 0:3.22:359*3.22;
```

Preliminary operations

```
close all; clc; addpath('VGM_functions')
```

4.2 Preprocessing for candidate ODEs

```
[symbols,ode,plot_settings,state,simulation,odes_path,coupling_idx,opt_settings] = ...
    preprocessing_dynamic_causal_modeling (simulation,candidate_odes,state);
```

```
True ODEs:
```

$$
\begin{vmatrix}
 & & \left/ \left/ 3\ \exp(-f\_1) \right. \right. & \left. \right\backslash & \left. \right\backslash \\
 & 25\ \exp(-q\_1)\ \exp(f\_1) & \left| \left| - \right| \right. & -\ 1 & \left| \right. \\
d\ q\_1 & & \left\backslash \left\backslash\ 5 \right. \right. & \left. \right/ & \left| \right. \\
----- == - \#3 - & ---------- & ---------- & ---------- & \left| \right. \\
dt & & 16 & & \left| \right. \\
 & & & & \left| \right. \\
 & & \left/ \left/ 3\ \exp(-f\_3) \right. \right. & \left. \right\backslash & \left| \right. \\
 & 25\ \exp(-q\_3)\ \exp(f\_3) & \left| \left| - \right| \right. & -\ 1 & \left| \right. \\
d\ q\_3 & & \left\backslash \left\backslash\ 5 \right. \right. & \left. \right/ & \left| \right. \\
----- == - \#1 - & ---------- & ---------- & ---------- & \left| \right. \\
dt & & 16 & & \left| \right.
\end{vmatrix}
$$

```
|                                                                                  |
|                                          / / 3 \exp(-f_2)     \                  |
|                  25 exp(-q_2) exp(f_2) | | - |          - 1 |                  |
|   d q_2                                \ \ 5 /              /                  |
|   ----- == - #2 - ------------------------------------------                   |
|     dt                                16                                       |
|                                                                                  |
|               d v_1    5 exp(-v_1) exp(f_1)                                     |
|               ----- == -------------------- - #3                                |
|                 dt              8                                                |
|                                                                                  |
|               d v_3    5 exp(-v_3) exp(f_3)                                     |
|               ----- == -------------------- - #1                                |
|                 dt              8                                                |
|                                                                                  |
|               d v_2    5 exp(-v_2) exp(f_2)                                     |
|               ----- == -------------------- - #2                                |
|                 dt              8                                                |
|                                                                                  |
|                     d f_1                                                        |
|                     ----- == s_1 exp(-f_1)                                       |
|                       dt                                                         |
|                                                                                  |
|                     d f_3                                                        |
|                     ----- == s_3 exp(-f_3)                                       |
|                       dt                                                         |
|                                                                                  |
|                     d f_2                                                        |
|                     ----- == s_2 exp(-f_2)                                       |
|                       dt                                                         |
|                                                                                  |
|              d s_1          3 s_1   8 exp(f_1)    8                             |
|              ----- == n_1 - ----- - ---------- + --                              |
|                dt             5         25       25                            |
|                                                                                  |
|              d s_3          3 s_3   8 exp(f_3)    8                             |
|              ----- == n_3 - ----- - ---------- + --                              |
|                dt             5         25       25                            |
|                                                                                  |
|              d s_2          3 s_2   8 exp(f_2)    8                             |
|              ----- == n_2 - ----- - ---------- + --                              |
|                dt             5         25       25                            |
|                                                                                  |
|             d n_1                                                               |
|             ----- == a_11 n_1 + a_12 n_2 + c_11 u_1                             |
|               dt                                                                |
|                                                                                  |
|             d n_3                                                               |
|             ----- == a_32 n_2 + a_33 n_3 + c_33 u_3                             |
|               dt                                                                |
|                                                                                  |
| d n_2                                                                            |
| ----- == a_22 n_2 + a_23 n_3 + n_1 (a_21 + d_213 n_3 + b_212 u_2)               |
\   dt                                                                            /
```

where

```
           / 17 v_3 \
```

```
      exp| ------ | 5
         \    8   /
#1 == ---------------
              8

         / 17 v_2 \
      exp| ------ | 5
         \    8   /
#2 == ---------------
              8

         / 17 v_1 \
      exp| ------ | 5
         \    8   /
#3 == ---------------
              8
```

4.3 Simulate Trajectories

- **Preprocessing for true ODEs**

```
[symbols_true,ode_true] = ...
preprocessing_dynamic_causal_modeling (simulation,simulation.odes,state);
```

```
Candidate ODEs:

/                                        / / 3 \exp(-f_1)     \   \
|                  25 exp(-q_1) exp(f_1) | | - |          - 1 |   |
|    d q_1                               \ \ 5 /              /   |
|    ----- == - #3 - ---------------------------------------    |
|      dt                              16                        |
|                                                                |
|                                        / / 3 \exp(-f_3)     \   |
|                  25 exp(-q_3) exp(f_3) | | - |          - 1 |   |
|    d q_3                               \ \ 5 /              /   |
|    ----- == - #1 - ---------------------------------------    |
|      dt                              16                        |
|                                                                |
|                                        / / 3 \exp(-f_2)     \   |
|                  25 exp(-q_2) exp(f_2) | | - |          - 1 |   |
|    d q_2                               \ \ 5 /              /   |
|    ----- == - #2 - ---------------------------------------    |
|      dt                              16                        |
|                                                                |
|                d v_1    5 exp(-v_1) exp(f_1)                   |
|                ----- == -------------------- - #3              |
|                  dt              8                             |
|                                                                |
|                d v_3    5 exp(-v_3) exp(f_3)                   |
|                ----- == -------------------- - #1              |
|                  dt              8                             |
|                                                                |
```

```
|                d v_2    5 exp(-v_2) exp(f_2)                    |
|                ----- == -------------------- - #2               |
|                  dt              8                              |
|                                                                 |
|                     d f_1                                       |
|                     ----- == s_1 exp(-f_1)                      |
|                       dt                                        |
|                                                                 |
|                     d f_3                                       |
|                     ----- == s_3 exp(-f_3)                      |
|                       dt                                        |
|                                                                 |
|                     d f_2                                       |
|                     ----- == s_2 exp(-f_2)                      |
|                       dt                                        |
|                                                                 |
|              d s_1          3 s_1   8 exp(f_1)    8            |
|              ----- == n_1 - ----- - ---------- + --             |
|                dt             5         25       25            |
|                                                                 |
|              d s_3          3 s_3   8 exp(f_3)    8            |
|              ----- == n_3 - ----- - ---------- + --             |
|                dt             5         25       25            |
|                                                                 |
|              d s_2          3 s_2   8 exp(f_2)    8            |
|              ----- == n_2 - ----- - ---------- + --             |
|                dt             5         25       25            |
|                                                                 |
|             d n_1                                               |
|             ----- == a_11 n_1 + a_12 n_2 + c_11 u_1             |
|               dt                                                |
|                                                                 |
|             d n_3                                               |
|             ----- == a_32 n_2 + a_33 n_3 + c_33 u_3             |
|               dt                                                |
|                                                                 |
| d n_2                                                           |
| ----- == a_22 n_2 + a_23 n_3 + n_1 (a_21 + d_213 n_3 + b_212 u_2) |
\   dt                                                            /
```

where

```
          / 17 v_3 \
        exp| ------ | 5
          \    8   /
   #1 == ---------------
                8

          / 17 v_2 \
        exp| ------ | 5
          \    8   /
   #2 == ---------------
                8

          / 17 v_1 \
        exp| ------ | 5
          \    8   /
   #3 == ---------------
```

8

- **Sample ODE parameters that lead to non-diverging trajectories:**

```
non_diverging_trajectories = false; i = 0;
while ~non_diverging_trajectories
```

- **Sample ODE parameters:** Non-selfinhibitory neuronal couplings are sampled uniformly in the interval $[-0.8, 0.8]$:

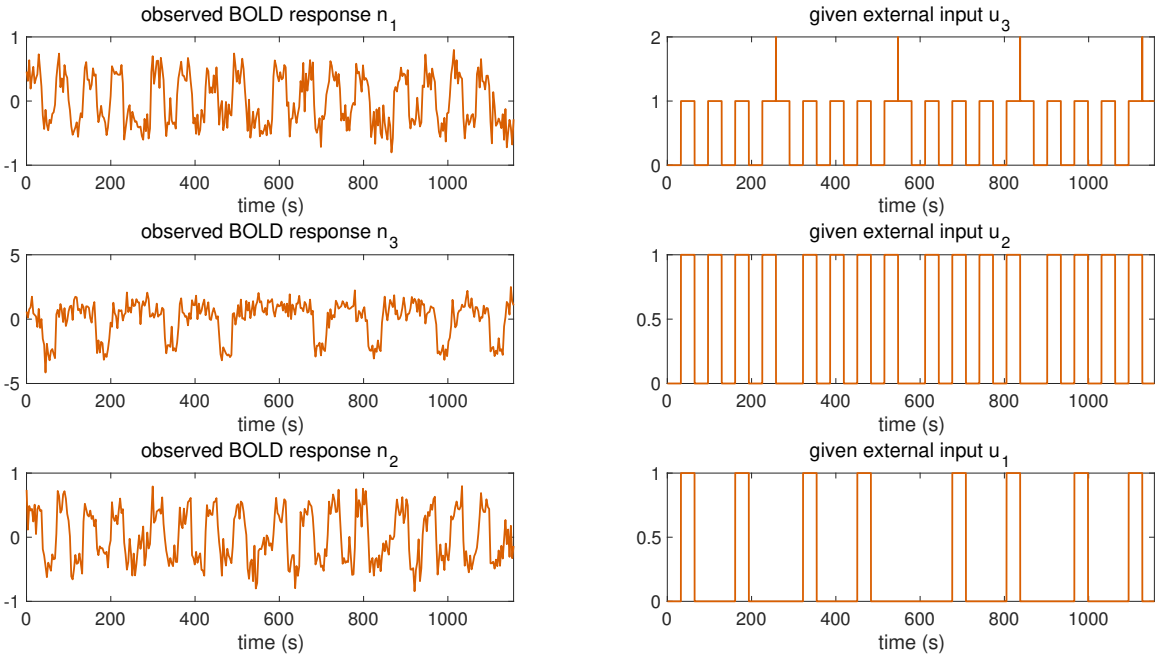
```
simulation.ode_param = -0.8 + (0.8-(-0.8)) * rand(1,length(symbols_true.param));
 % simulation.ode_param = [0.46,0.13,0.39,0.26,0.5,0.26,0.1,1.25,-1,-1,-1];
 % published ODE parameters (slightly modified from Stephan et al., 2008)
```

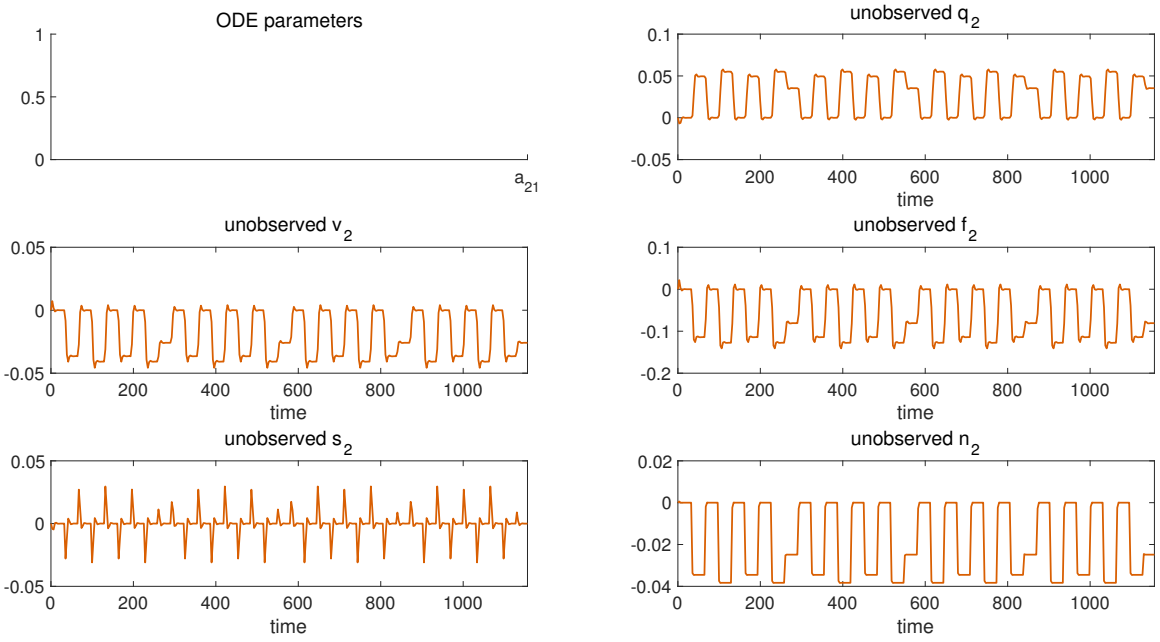
- **Self-inhibitory neuronal couplings set to -1:**

```
simulation.ode_param(end-2:end) = -1;
```

- **Numerical integration**

```
try
    simulation_old = simulation;
    [simulation,obs_to_state_relation,fig_handle,plot_handle] = ...
    simulate_state_dynamics_dcm(simulation,symbols_true,ode_true,...
    time,plot_settings,state.ext_input,'plot');
    non_diverging_trajectories = 1;
end
```






```
end
```

4.4 Mass Action Dynamical Systems

A deterministic dynamical system is represented by a set of K ordinary differential equations (ODEs) with model parameters $\boldsymbol{\theta} \in \mathbb{R}^d$ that describe the evolution of K states $\mathbf{x}(t) = [x_1(t), \ldots, x_K(t)]^T$ such that:

$$\dot{\mathbf{x}}(t) = \frac{d\mathbf{x}(t)}{dt} = \mathbf{f}(\mathbf{x}(t), \boldsymbol{\theta}), \qquad (1)$$

A sequence of observations, $\mathbf{y}(t)$, is usually contaminated by measurement error which we assume to be normally distributed with zero mean and variance for each of the K states, i.e. $\mathbf{E} \sim \mathcal{N}(\mathbf{E}; \mathbf{0}, \mathbf{D})$, with $\mathbf{D}_{ik} = \sigma_k^2 \delta_{ik}$. For N distinct time points the overall system may therefore be summarized as

$$\mathbf{Y} = \mathbf{X} + \mathbf{E},$$

where

$$\mathbf{X} = [\mathbf{x}(t_1), \ldots, \mathbf{x}(t_N)] = [\mathbf{x}_1, \ldots, \mathbf{x}_K]^T,$$

$$\mathbf{Y} = [\mathbf{y}(t_1), \ldots, \mathbf{y}(t_N)] = [\mathbf{y}_1, \ldots, \mathbf{y}_K]^T,$$

and $\mathbf{x}_k = [x_k(t_1), \ldots, x_k(t_N)]^T$ is the k'th state sequence and $\mathbf{y}_k = [y_k(t_1), \ldots, y_k(t_N)]^T$ are the observations. Given the observations $\mathbf{Y}$ and the description of the dynamical system (1), the aim is to estimate both state variables $\mathbf{X}$ and parameters $\boldsymbol{\theta}$.

We consider only dynamical systems that are *locally linear* w.r.t ODE parameters $\boldsymbol{\theta}$ and individual states $\mathbf{x}$. Such ODEs include mass-action kinetics and are given by:

$$f_k(\mathbf{x}(t), \theta) = \sum_{i=1} \theta_{ki} \prod_{j \in \mathcal{M}_{ki}} x_j, \qquad (2)$$

with $\mathcal{M}_{k_i} \subseteq \{1, \ldots, K\}$ describing the state variables in each factor of the equation (i.e. the functions are linear in parameters and contain arbitrary large products of monomials of the states).
start timer

```
tic;
```

4.5 Prior on States and State Derivatives

Gradient matching with Gaussian processes assumes a joint Gaussian process prior on states and their derivatives:

$$\begin{pmatrix} \mathbf{X} \\ \dot{\mathbf{X}} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{X} \\ \dot{\mathbf{X}} \end{pmatrix} ; \begin{matrix} \mathbf{0} \\ \mathbf{0} \end{matrix} , \begin{matrix} \mathbf{C}_{\boldsymbol{\phi}} & \mathbf{C}'_{\boldsymbol{\phi}} \\ {}'\mathbf{C}_{\boldsymbol{\phi}} & \mathbf{C}''_{\boldsymbol{\phi}} \end{matrix} \right) \qquad (3),$$

with

$$\text{cov}(x_k(t), x_k(t)) = C_{\boldsymbol{\phi}_k}(t, t'),$$

$$\text{cov}(\dot{x}_k(t), x_k(t)) = \frac{\partial C_{\boldsymbol{\phi}_k}(t,t')}{\partial t} =: C'_{\boldsymbol{\phi}_k}(t, t'),$$

$$\text{cov}(x_k(t), \dot{x}_k(t)) = \frac{\partial C_{\boldsymbol{\phi}_k}(t,t')}{\partial t'} =: {}'C_{\boldsymbol{\phi}_k}(t, t'),$$

$$\text{cov}(\dot{x}_k(t), \dot{x}_k(t)) = \frac{\phi_k(t,t')}{\partial t \partial t'} =: C''_{\boldsymbol{\phi}_k}(t, t').$$

4.6 Matching Gradients

Given the joint distribution over states and their derivatives (3) as well as the ODEs (2), we therefore have two expressions for the state derivatives:

$$\dot{\mathbf{X}} = \mathbf{F} + \boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_1 \sim \mathcal{N}\left(\boldsymbol{\epsilon}_1; \mathbf{0}, \mathbf{I}\gamma\right),$$

$$\dot{\mathbf{X}} = {}'\mathbf{C}_{\boldsymbol{\phi}}\mathbf{C}_{\boldsymbol{\phi}}^{-1}\,\mathbf{X} + \boldsymbol{\epsilon}_2, \boldsymbol{\epsilon}_2 \sim \mathcal{N}\left(\boldsymbol{\epsilon}_2; \mathbf{0}, \mathbf{A}\right),$$

where $\mathbf{F} := \mathbf{f}(\mathbf{X}, \boldsymbol{\theta})$ and $\mathbf{A} := \mathbf{C}''_{\boldsymbol{\phi}} - {}'\mathbf{C}_{\boldsymbol{\phi}}\mathbf{C}_{\boldsymbol{\phi}}^{-1}\mathbf{C}'_{\boldsymbol{\phi}}$ and γ is the error variance in the ODEs. Note that, in a deterministic system, the output of the ODEs $\mathbf{F}$ should equal the state derivatives $\dot{\mathbf{X}}$. However, in the first equation above we relax this contraint by adding stochasticity to the state derivatives $\dot{\mathbf{X}}$ in order to compensate for a potential model mismatch. The second equation above is obtained by deriving the conditional distribution for $\dot{\mathbf{X}}$ from the joint distribution in equation (3). Equating the two expressions in the equations above we can eliminate the unknown state derivatives

4.7 Rewrite ODEs as Linear Combination in Parameters

Since, according to the mass action dynamics (equation 2), the ODEs are *linear in the parameters* $\boldsymbol{\theta}$ we can rewrite the ODEs in equation (2) as a linear combination in the parameters:

$$\mathbf{B}_{\boldsymbol{\theta}k}\boldsymbol{\theta} + \mathbf{b}_{\boldsymbol{\theta}k} \stackrel{!}{=} \mathbf{f}_k(\mathbf{X}, \boldsymbol{\theta}), \qquad (5)$$

where matrices $\mathbf{B}_{\boldsymbol{\theta}k}$ and $\mathbf{b}_{\boldsymbol{\theta}k}$ are defined such that the ODEs $\mathbf{f}_k(\mathbf{X}, \boldsymbol{\theta})$ are expressed as a linear combination in $\boldsymbol{\theta}$.

```
[ode_param.lin_comb.B,ode_param.lin_comb.b] = ...
rewrite_odes_as_linear_combination_in_parameters(ode,symbols);
```

4.8 Posterior over ODE Parameters

Inserting (5) into (4) and solving for $\boldsymbol{\theta}$ yields:

$$\boldsymbol{\theta} = \mathbf{B}_{\boldsymbol{\theta}}^{+} \left({}'\mathbf{C}_{\boldsymbol{\phi}} \mathbf{C}_{\boldsymbol{\phi}}^{-1} \mathbf{X} - \mathbf{b}_{\boldsymbol{\theta}} + \boldsymbol{\epsilon}_0 \right),$$

where $\mathbf{B}_{\boldsymbol{\theta}}^{+}$ denotes the pseudo-inverse of $\mathbf{B}_{\boldsymbol{\theta}}$. Since $\mathbf{C}_{\boldsymbol{\phi}}$ is block diagonal we can rewrite the expression above as:

$$\begin{aligned}
\boldsymbol{\theta} &= \left(\mathbf{B}_{\boldsymbol{\theta}}^{T} \mathbf{B}_{\boldsymbol{\theta}}\right)^{-1} \mathbf{B}_{\boldsymbol{\theta}}^{T} \left(\textstyle\sum_k {}'\mathbf{C}_{\boldsymbol{\phi}_k} \mathbf{C}_{\boldsymbol{\phi}_k}^{-1} \mathbf{X}_k - \mathbf{b}_{\boldsymbol{\theta} k} + \boldsymbol{\epsilon}_0^{(k)} \right) \\
&= \left(\mathbf{B}_{\boldsymbol{\theta}}^{T} \mathbf{B}_{\boldsymbol{\theta}}\right)^{-1} \left(\textstyle\sum_k \mathbf{B}_{\boldsymbol{\theta} k}^{T} \left({}'\mathbf{C}_{\boldsymbol{\phi}_k} \mathbf{C}_{\boldsymbol{\phi}_k}^{-1} \mathbf{X}_k - \mathbf{b}_{\boldsymbol{\theta} k} + \boldsymbol{\epsilon}_0^{(k)} \right) \right),
\end{aligned}$$

where we subsitute the Moore-Penrose inverse for the pseudo-inverse (i.e. $\mathbf{B}_{\boldsymbol{\theta}}^{+} := \left(\mathbf{B}_{\boldsymbol{\theta}}^{T} \mathbf{B}_{\boldsymbol{\theta}}\right)^{-1} \mathbf{B}_{\boldsymbol{\theta}}^{T}$). We can therefore derive the posterior distribution over ODE parameters:

$$p(\boldsymbol{\theta} \mid \mathbf{X}, \boldsymbol{\phi}, \gamma) = \mathcal{N}\left(\boldsymbol{\theta}; \left(\mathbf{B}_{\boldsymbol{\theta}}^{T} \mathbf{B}_{\boldsymbol{\theta}}\right)^{-1} \left(\textstyle\sum_k \mathbf{B}_{\boldsymbol{\theta} k}^{T} \left({}'\mathbf{C}_{\boldsymbol{\phi} k} \mathbf{C}_{\boldsymbol{\phi} k}^{-1} \mathbf{X}_k - \mathbf{b}_{\boldsymbol{\theta} k} \right) \right), \mathbf{B}_{\boldsymbol{\theta}}^{+} \left(\mathbf{A} + \mathbf{I}\gamma\right) \mathbf{B}_{\boldsymbol{\theta}}^{+T} \right) \qquad (6).$$

4.9 Rewrite Hemodynamic ODEs as Linear Combination in (monotonic functions of) Individual Hemodynamic States

4.9.1 Rewrite the BOLD Signal Change Equation as a Linear Combination in a Monotonic Function of the Deoxyhemoglobin Content $\exp(\mathbf{q})$

$$\mathbf{R}_{q\,\lambda}\ \exp(\mathbf{q})\ +\ \mathbf{r}_{v\,\lambda} \overset{!}{=} \lambda(\mathbf{q}, \mathbf{v}).$$

```
[state.deoxyhemo.R,state.deoxyhemo.r] = ...
rewrite_bold_signal_eqn_as_linear_combination_in_deoxyhemo(symbols);
```

4.9.2 Rewrite the Deoxyhemoglobin Content ODE as a Linear Combination in a Monotonic Function of the Blood Volume $\exp\left(17/8\,\mathbf{v}\right)$

$$\mathbf{R}_{v\dot{q}}\ \exp\left(17/8\,\mathbf{v}\right)\ +\ \mathbf{r}_{v\dot{q}} \overset{!}{=} \mathbf{f}_{\dot{q}}(\mathbf{X}, \boldsymbol{\theta}).$$

```
[state.vol.R,state.vol.r] = rewrite_deoxyhemo_ODE_as_linear_combination_in_vol(...
ode,symbols);
```

4.9.3 Rewrite the Blood Volume ODE as a Linear Combination in a Monotonic Function of the Blood Flow $\exp(\mathbf{f})$

$$\mathbf{R}_{f\,\dot{v}}\ \exp(\mathbf{f}) + \mathbf{r}_{f\,\dot{v}} \overset{!}{=} \mathbf{f}_{\dot{v}}(\mathbf{X}, \boldsymbol{\theta}).$$

```
[state.flow.R,state.flow.r] = rewrite_vol_ODE_as_linear_combination_in_flow(...
ode,symbols);
```

4.9.4 Rewrite the Blood Flow and Vasosignalling ODEs as a linear combination in Vasosignalling s

$$\mathbf{R}_{s\dot{f}}\,\mathbf{s} + \mathbf{r}_{s\dot{f}} \overset{!}{=} \mathbf{f}_{\dot{f}}(\mathbf{X}, \boldsymbol{\theta}), \qquad \mathbf{R}_{s\dot{s}}\,\mathbf{s} + \mathbf{r}_{\dot{s}} \overset{!}{=} \mathbf{f}_{\dot{s}}(\mathbf{X}, \boldsymbol{\theta}).$$

```
[state.vaso.R,state.vaso.r] = ...
rewrite_vaso_and_flow_odes_as_linear_combination_in_vaso(ode,symbols);
```

4.10 Rewrite Neuronal ODEs as Linear Combination in Individual Neuronal States

We rewrite the ODE(s) $\mathbf{f}_k(\mathbf{X}, \boldsymbol{\theta})$ as a linear combination in the individual state $\mathbf{n}_u$:

$$\mathbf{R}_{uk}\mathbf{n}_u + \mathbf{r}_{uk} \overset{!}{=} \mathbf{f}_k(\mathbf{X}, \boldsymbol{\theta}),$$

where matrices $\mathbf{R}_{uk}$ and $\mathbf{r}_{uk}$ are defined such that the ODE $\mathbf{f}_k(\mathbf{X}, \boldsymbol{\theta})$ is expressed as a linear combination in the individual state $\mathbf{n}_u$.

```
[state.neuronal.R,state.neuronal.r] = ...
rewrite_odes_as_linear_combination_in_ind_neuronal_states(ode,symbols,...
coupling_idx.states);
```

4.11 Posterior over Individual States

Given the linear combination of the ODEs w.r.t. an individual state, we define the matrices $\mathbf{B}_u$ and $\mathbf{b}_u$ such that the expression $\mathbf{f}(\mathbf{X}, \boldsymbol{\theta}) - {}'\mathbf{C}_{\boldsymbol{\phi}}\mathbf{C}_{\boldsymbol{\phi}}^{-1}\mathbf{X}$ is rewritten as a linear combination in an individual state $\mathbf{x}_u$:

$$\mathbf{B}_u\mathbf{x}_u + \mathbf{b}_u \overset{!}{=} \mathbf{f}(\mathbf{X}, \boldsymbol{\theta}) - {}'\mathbf{C}_{\boldsymbol{\phi}}\mathbf{C}_{\boldsymbol{\phi}}^{-1}\mathbf{X} \qquad (7).$$

Inserting (7) into (4) and solving for $\mathbf{x}_u$ yields:

$$\mathbf{x}_u = \mathbf{B}_u^{+}\left(\boldsymbol{\epsilon}_0 - \mathbf{b}_u\right),$$

where $\mathbf{B}_u^{+}$ denotes the pseudo-inverse of $\mathbf{B}_u$. Since $\mathbf{C}_{\boldsymbol{\phi}}$ is block diagonal we can rewrite the expression above as:

$$\begin{aligned}\mathbf{x}_u &= \left(\mathbf{B}_u\mathbf{B}_u^T\right)^{-1}\mathbf{B}_u^T\textstyle\sum_k\left(\boldsymbol{\epsilon}_0^{(k)} - \mathbf{b}_{uk}\right)\\ &= \left(\mathbf{B}_u\mathbf{B}_u^T\right)^{-1}\textstyle\sum_k\mathbf{B}_{uk}^T\left(\boldsymbol{\epsilon}_0^{(k)} - \mathbf{b}_{uk}\right),\end{aligned}$$

where we substitute the Moore-Penrose inverse for the pseudo-inverse (i.e. $\mathbf{B}_{\boldsymbol{\theta}}^{+} := \left(\mathbf{B}_{\boldsymbol{\theta}}^T\mathbf{B}_{\boldsymbol{\theta}}\right)^{-1}\mathbf{B}_{\boldsymbol{\theta}}^T$). We can therefore derive the posterior distribution over an individual state $\mathbf{x}_u$:

$$p(\mathbf{x}_u \mid \mathbf{X}_{-u}, \boldsymbol{\phi}, \gamma) = \mathcal{N}\left(\mathbf{x}_u; \left(\mathbf{B}_u\mathbf{B}_u^T\right)^{-1}\left(-\textstyle\sum_k\mathbf{B}_{uk}^T\mathbf{b}_{uk}\right),\ \mathbf{B}_u^{+}\left(\mathbf{A} + \mathbf{I}\gamma\right)\mathbf{B}_u^{+T}\right) \qquad (8),$$

with $\mathbf{X}_{-u}$ denoting the set of all states except state $\mathbf{x}_u$.

4.12 Mean-field Variational Inference

To infer the parameters $\boldsymbol{\theta}$, we want to find the maximum a posteriori estimate (MAP):

$$\begin{aligned}\boldsymbol{\theta}^* &:= \arg\max_{\boldsymbol{\theta}} \ \ln p(\boldsymbol{\theta} \mid \mathbf{Y}, \boldsymbol{\phi}, \gamma, \boldsymbol{\sigma}) \\ &= \arg\max_{\boldsymbol{\theta}} \ \ln \int p(\boldsymbol{\theta}, \mathbf{X} \mid \mathbf{Y}, \boldsymbol{\phi}, \gamma, \boldsymbol{\sigma}) d\mathbf{X} \\ &= \arg\max_{\boldsymbol{\theta}} \ \ln \int p(\boldsymbol{\theta} \mid \mathbf{X}, \boldsymbol{\phi}, \gamma) p(\mathbf{X} \mid \mathbf{Y}, \boldsymbol{\phi}, \boldsymbol{\sigma}) d\mathbf{X} \qquad (9).\end{aligned}$$

However, the integral above is intractable due to the strong couplings induced by the nonlinear ODEs $\mathbf{f}$ which appear in the term $p(\boldsymbol{\theta} \mid \mathbf{X}, \boldsymbol{\phi}, \gamma)$.

We use mean-field variational inference to establish variational lower bounds that are analytically tractable by decoupling state variables from the ODE parameters as well as decoupling the state variables from each other. Note that, since the ODEs described by equation (2) are *locally linear*, both conditional distributions $p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{Y}, \boldsymbol{\phi}, \gamma, \boldsymbol{\sigma})$ (equation (6)) and $p(\mathbf{x}_u \mid \boldsymbol{\theta}, \mathbf{X}_{-u}, \mathbf{Y}, \boldsymbol{\phi}, \gamma, \boldsymbol{\sigma})$ (equation (8)) are analytically tractable and Gaussian distributed as mentioned previously. The decoupling is induced by designing a variational distribution $Q(\boldsymbol{\theta}, \mathbf{X})$ which is restricted to the family of factorial distributions:

$$\mathcal{Q} := \left\{ Q : Q(\boldsymbol{\theta}, \mathbf{X}) = q(\boldsymbol{\theta}) \textstyle\prod_u q(\mathbf{x}_u) \right\}.$$

The particular form of $q(\boldsymbol{\theta})$ and $q(\mathbf{x}_u)$ are designed to be Gaussian distributed which places them in the same family as the true full conditional distributions. To find the optimal factorial distribution we minimize the Kullback-Leibler divergence between the variational and the true posterior distribution:

$$\hat{Q} := \arg\min_{Q(\boldsymbol{\theta},\mathbf{X}) \in \mathcal{Q}} \mathrm{KL}\left[Q(\boldsymbol{\theta}, \mathbf{X}) \mid\mid p(\boldsymbol{\theta}, \mathbf{X} \mid \mathbf{Y}, \boldsymbol{\phi}, \gamma, \boldsymbol{\sigma}) \right] \qquad (10),$$

where $\hat{Q}$ is the proxy distribution. The proxy distribution that minimizes the KL-divergence (10) depends on the true full conditionals and is given by:

$$\begin{aligned}\hat{q}(\boldsymbol{\theta}) &\propto \exp\left(E_{Q_{-\boldsymbol{\theta}}} \ln p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{Y}, \boldsymbol{\phi}, \gamma, \boldsymbol{\sigma}) \right) \qquad (11) \\ \hat{q}(\mathbf{x}_u) &\propto \exp\left(E_{Q_{-u}} \ln p(\mathbf{x}_u \mid \boldsymbol{\theta}, \mathbf{X}_{-u}, \mathbf{Y}, \boldsymbol{\phi}, \gamma, \boldsymbol{\sigma}) \right) \qquad (12).\end{aligned}$$

4.13 Denoising BOLD Observations

We denoise the BOLD observation by standard GP regression.

```
bold_response.denoised_obs = ...
denoising_BOLD_observations(simulation.bold_response{:,{'n_1','n_3','n_2'}},...
inv_C,symbols,simulation);
```

4.14 Fitting Observations of State Trajectories

We fit the observations of state trajectories by standard GP regression. The data-informed distribution$ $p(\mathbf{X} \mid \mathbf{Y}, \boldsymbol{\phi}, \boldsymbol{\sigma})$ in euqation (9) can be determined analytically using Gaussian process regression with the GP prior $p(\mathbf{X} \mid \boldsymbol{\phi}) = \prod_k \mathcal{N}(\mathbf{x}_k; \mathbf{0}, \mathbf{C}_{\boldsymbol{\phi}_k})$:

$$p(\mathbf{X} \mid \mathbf{Y}, \boldsymbol{\phi}, \gamma) = \prod_k \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_k(\mathbf{y}_k), \boldsymbol{\sigma}_k), \text{ where } \boldsymbol{\mu}_k(\mathbf{y}_k) := \boldsymbol{\sigma}_k^{-2} \left(\boldsymbol{\sigma}_k^{-2}\mathbf{I} + \mathbf{C}_{\boldsymbol{\phi}_k}^{-1} \right)^{-1} \mathbf{y}_k \text{ and } \boldsymbol{\sigma}_k^{-1} := \boldsymbol{\sigma}_k^{-2}\mathbf{I} + \mathbf{C}_{\boldsymbol{\phi}_k}^{-1}.$$

```
[mu,inv_sigma] = fitting_state_observations(inv_C,obs_to_state_relation,...
simulation,symbols);
```

4.15 Coordinate Ascent Variational Gradient Matching

We minimize the KL-divergence in equation (10) by coordinate descent (where each step is analytically tractable) by iterating between determining the proxy for the distribution over ODE parameters $\hat{q}(\boldsymbol{\theta})$ and the proxies for the distribution over individual states $\hat{q}(\mathbf{x}_u)$.

Initialize the state estimation by the GP regression posterior

```
state.proxy.mean = array2table([time.est',mu],...
'VariableNames',['time',symbols.state_string]);
bold_response.obs_old = bold_response.denoised_obs;
ode_param.proxy.mean = zeros(length(symbols.param),1);
```

Coordinate ascent

```
for i = 1:opt_settings.coord_ascent_numb_iter
```

4.16 Proxy for Hemodynamic States

Determine the proxies for the states, starting with deoxyhemoglobin followed by blood volume, blood flow and finally vasosignalling. The information flow in the hemodynamic system is shown in its factor graph below:

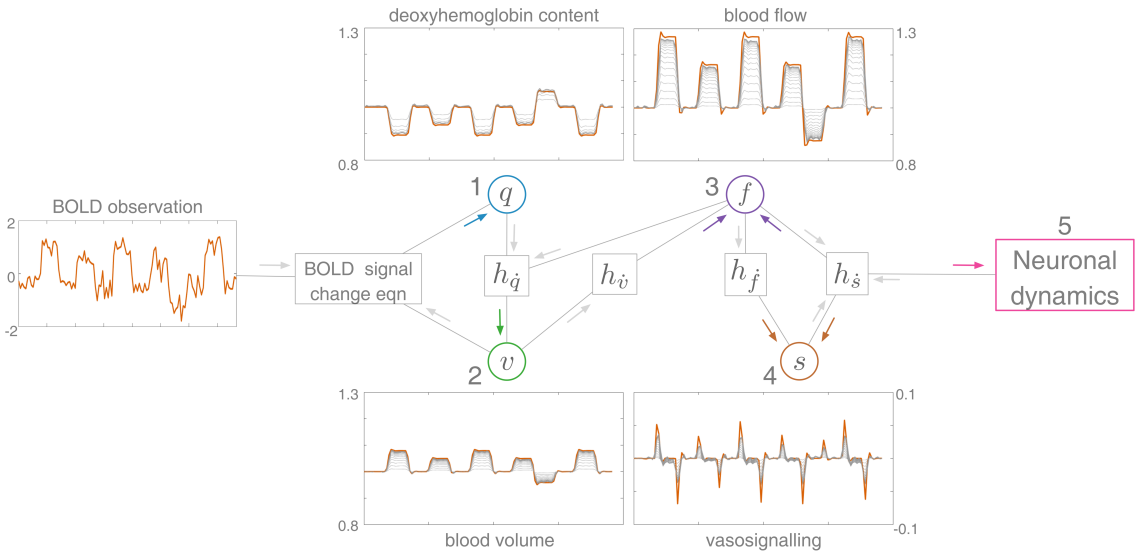


Figure 4.1: The model inversion for the hemodynamic factor graph above occurs locally w.r.t. individual states. Given the expression for the BOLD signal change equation, we invert the BOLD signal change equation analytically to determine the deoxyhemoglobin content $\mathbf{q}$ (1). The newly inferred deoxyhemoglobin content $\mathbf{q}$ influences the expression for the factor associated with the change in deoxyhemoglobin content $\mathbf{h}_{\dot{\mathbf{q}}}$, which we subsequently invert analytically to infer the blood volume $\mathbf{v}$ (2). Thereafter, we infer the blood flow $\mathbf{f}$ (3) by inverting the factors associated with the change in blood volume $\mathbf{h}_{\dot{\mathbf{v}}}$ as well as vasosignalling $\mathbf{h}_{\dot{\mathbf{s}}}$, followed by inferring vasosignalling $\mathbf{s}$ (4) by inverting the factors associated with blood flow induction $\mathbf{h}_{\dot{\mathbf{f}}}$ and vasosignalling $\mathbf{h}_{\dot{\mathbf{s}}}$. Finally, the neuronal dynamics (5) are learned, in part, by inverting the factor associated with vasosignalling $\mathbf{h}_{\dot{\mathbf{s}}}$. The typical trajectories of each of the states are shown (red) together with their iterative approximation (grey lines) obtained by graphical DCM.

4.16.1 Proxy for Deoxyhemolgobin Content

Damping is required since we invert only the factor for the BOLD signal change equation w.r.t. a monotonic function of deoxyhemoglobin content $\exp(\mathbf{q})$.

- **Undamped proxy:**

```
state_proxy_undamped = proxy_for_deoxyhemoglobin_content(state.deoxyhemo,...
state.proxy.mean{:,symbols.state_string},bold_response.denoised_obs,...
symbols,A_plus_gamma_inv,opt_settings);
```

- **Damped proxy:**

```
state.proxy.mean{:,{'q_1','q_3','q_2'}} = (1-opt_settings.damping) * ...
state.proxy.mean{:,{'q_1','q_3','q_2'}} + ...
opt_settings.damping * state_proxy_undamped;
```

4.16.2 Proxy for Blood Volume

Damping is required since we invert only the a subset of ODEs w.r.t. a monotonic function of blood volume $\exp(17/8\,\mathbf{v})$.

- **Undamped proxy:**

```
state_proxy_undamped = proxy_for_blood_volume(state.vol,...
dC_times_invC,state.proxy.mean{:,symbols.state_string},...
ode_param.proxy.mean,symbols,A_plus_gamma_inv,opt_settings);
```

- **Damped proxy:**

```
state.proxy.mean{:,{'v_1','v_3','v_2'}} = (1-opt_settings.damping) * ...
state.proxy.mean{:,{'v_1','v_3','v_2'}} + ...
opt_settings.damping * state_proxy_undamped;
```

4.16.3 Proxy for Blood Flow

Damping is required since we invert only the a subset of ODEs w.r.t. a mononic function of blood flow $\exp(\mathbf{f})$.

- **Undamped proxy:**

```
state_proxy_undamped = proxy_for_blood_flow(state.flow,...
dC_times_invC,state.proxy.mean{:,symbols.state_string},...
ode_param.proxy.mean,symbols,A_plus_gamma_inv,opt_settings);
```

- **Damped proxy:**

```
state.proxy.mean{:,{'f_1','f_3','f_2'}} = (1-opt_settings.damping) * ...
state.proxy.mean{:,{'f_1','f_3','f_2'}} + ...
opt_settings.damping * state_proxy_undamped;
```

4.16.4 Proxy for Vasosignalling

No damping is required because we invert all ODEs w.r.t. vasosingalling **s**.

```
state.proxy.mean{:,{'s_1','s_3','s_2'}} = proxy_for_vasosignalling(...
state.vaso,dC_times_invC,state.proxy.mean{:,symbols.state_string},...
ode_param.proxy.mean,symbols,A_plus_gamma_inv,opt_settings);
```

4.17 Proxy for Neuronal States

Determine the proxies for the neuronal states. An example of the information flow in the neuronal part of the nonlinear forward modulating (nonlinear forward modulation by attention) is shown in its factor graph below:

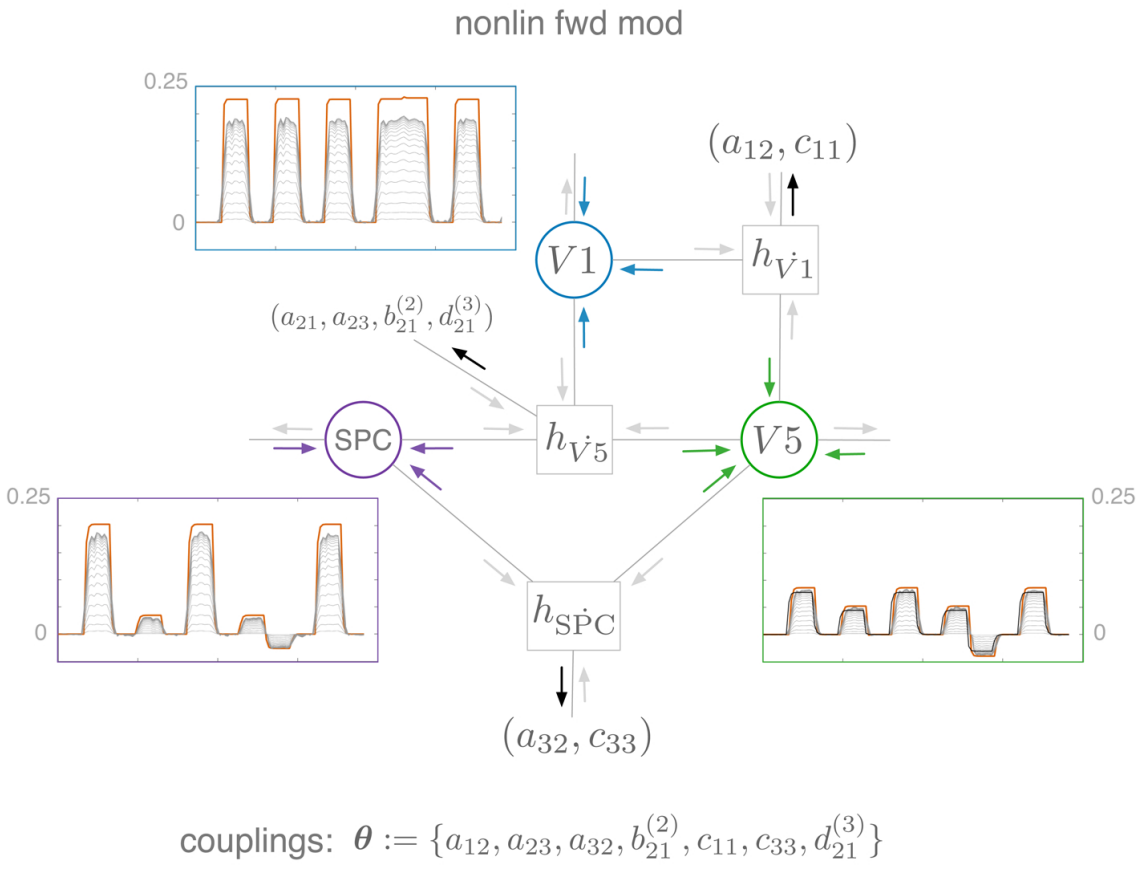


Figure 4.2: In the neuronal factor graph (for the nonlinear forward modulation) above each individual state appears linear in every factor in the neuronal model. We can therefore analytically invert every factor to determine the neuronal state. The typical trajectories of each of the states are shown (red) together with their iterative approximation (grey lines) obtained by variational gradient matching.

No damping is required because we invert all ODEs w.r.t. neuronal populations **n**.

```
state.proxy.mean{:,{'n_1','n_3','n_2'}} = ...
proxy_for_neuronal_populations(state.neuronal,...
  state.proxy.mean{:,symbols.state_string},ode_param.proxy.mean',...
    dC_times_invC,coupling_idx.states,symbols,A_plus_gamma_inv,...
    opt_settings);
```

Keep initial value at zero:

```
state_idx = cellfun(@(x) ~strcmp(x(1),'u'),symbols.state_string);
state.proxy.mean{:,symbols.state_string(state_idx)} = ...
bsxfun(@minus,state.proxy.mean{:,symbols.state_string(state_idx)},...
    state.proxy.mean{1,symbols.state_string(state_idx)});
```

4.18 Proxy for ODE parameters

Expanding the proxy distribution in equation (11) for $\boldsymbol{\theta}$ yields:

$$\begin{aligned}\hat{q}(\boldsymbol{\theta}) &\propto \exp\left(E_{Q_{-\boldsymbol{\theta}}} \ln p(\theta \mid \mathbf{X}, \mathbf{Y}, \boldsymbol{\phi}, \gamma, \boldsymbol{\sigma}) \right) \\ &= \exp\left(E_{Q_{-\boldsymbol{\theta}}} \ln \mathcal{N}\left(\boldsymbol{\theta}; \left(\mathbf{B}_{\boldsymbol{\theta}}^T \mathbf{B}_{\boldsymbol{\theta}}\right)^{-1} \left(\sum_k \mathbf{B}_{\boldsymbol{\theta}k}^T \left({}'\mathbf{C}_{\phi_k} \mathbf{C}_{\phi k}^{-1} \mathbf{X}_k - \mathbf{b}_{\boldsymbol{\theta}_k} \right) \right), \mathbf{B}_{\boldsymbol{\theta}}^{+} \left(\mathbf{A} + \mathbf{I}\gamma\right) \mathbf{B}_{\boldsymbol{\theta}}^{+T} \right) \right)\end{aligned}$$

where we substitute $p(\boldsymbol{\theta} \mid \mathbf{X}, \boldsymbol{\phi}, \gamma)$ with its density given in equation (6). No damping is required because we invert all ODEs w.r.t. neuronal couplings $\boldsymbol{\theta}$.

```
if i>200 || i==opt_settings.coord_ascent_numb_iter
[ode_param.proxy.mean,ode_param.proxy.inv_cov] = proxy_for_ode_parameters(...
        state.proxy.mean{:,symbols.state_string},dC_times_invC,...
        ode_param.lin_comb,symbols,A_plus_gamma_inv,opt_settings);
end
```

4.19 Intercept due to Confounding Effects

The BOLD response is given by:

$$\mathbf{y} = \boldsymbol{\lambda}(\mathbf{q}, \mathbf{v}) + \boldsymbol{\mathcal{X}}\, \boldsymbol{\beta}$$

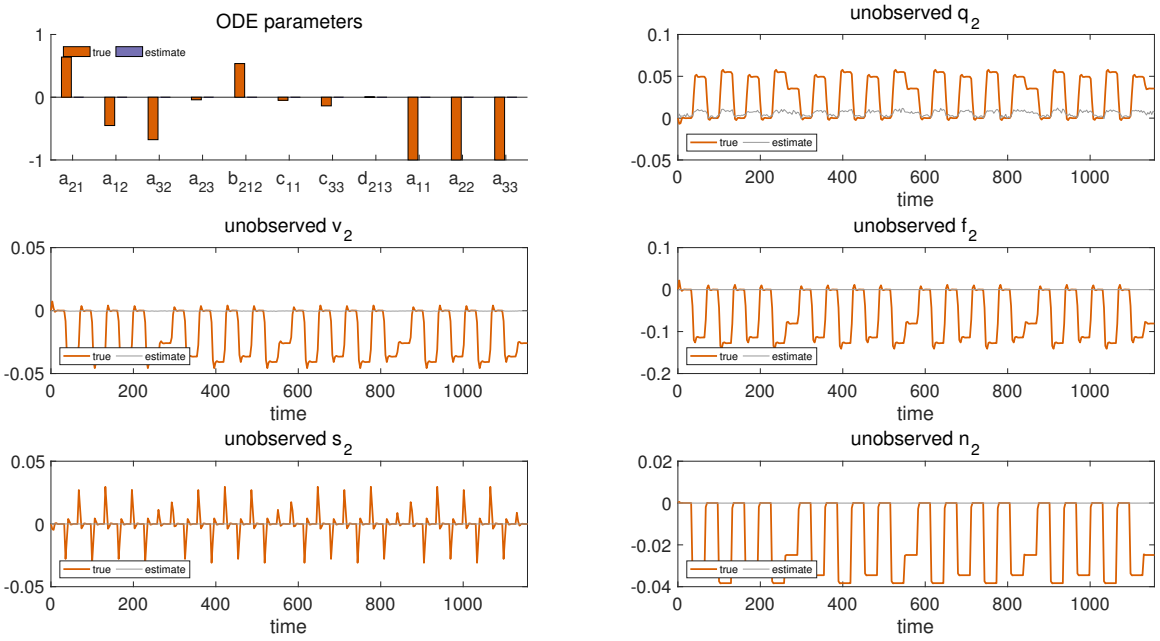
,
where $\mathbf{y}$ are the BOLD observations, $\boldsymbol{\lambda}(\mathbf{q}, \mathbf{v})$ is the BOLD signal change equation and the matrix $\mathcal{X}$ is given. The intercept is determined by a minimum least squares estimator:

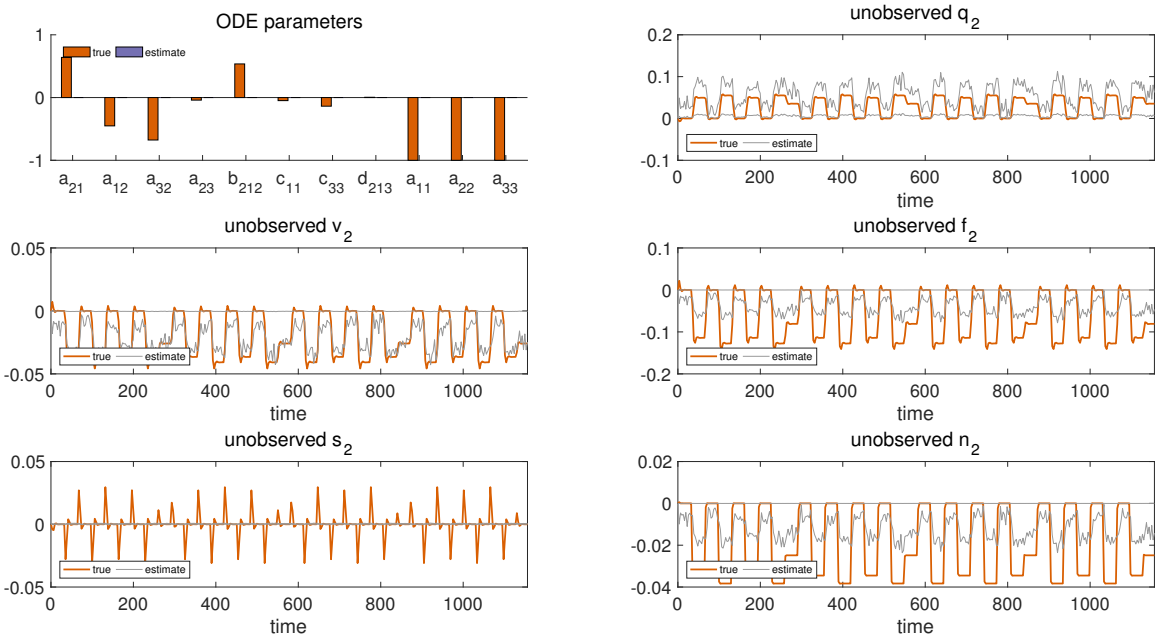
$$\boldsymbol{\mathcal{X}}\, \hat{\boldsymbol{\beta}} := \boldsymbol{\mathcal{X}} (\boldsymbol{\mathcal{X}}^T \boldsymbol{\mathcal{X}})^{-1} \boldsymbol{\mathcal{X}}^T (\mathbf{y} - \boldsymbol{\lambda}(\mathbf{q}, \mathbf{v}))$$

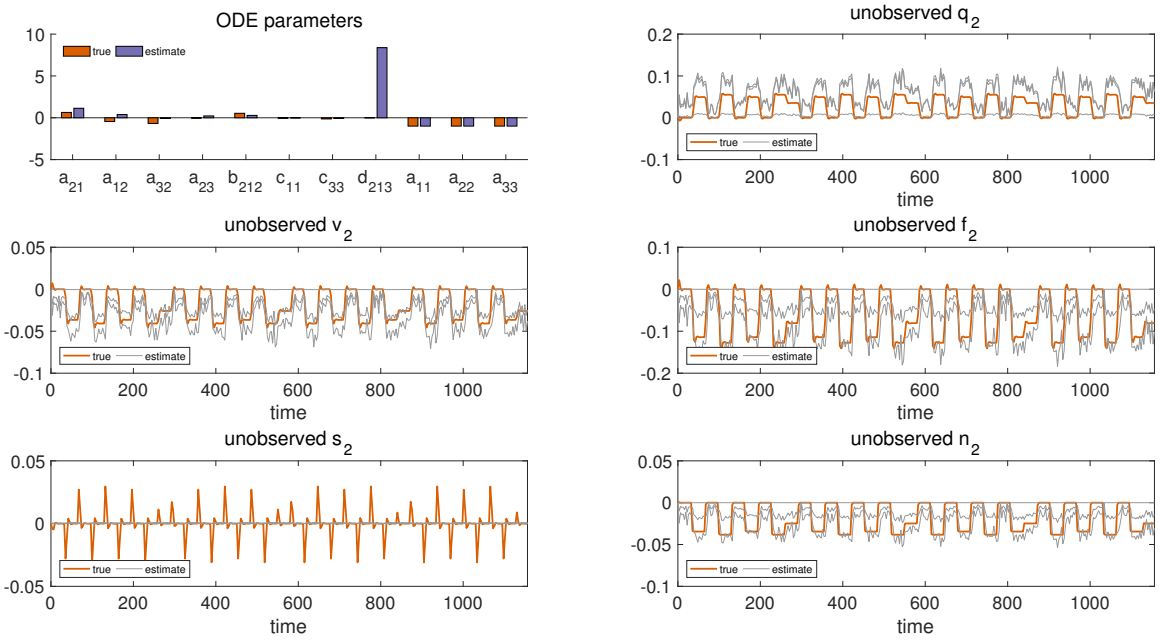
```
bold_signal_change = bold_signal_change_eqn(...
state.proxy.mean{:,{'v_1','v_3','v_2'}},...
state.proxy.mean{:,{'q_1','q_3','q_2'}});
intercept = simulation.X0 * (simulation.X0' * simulation.X0)^(-1) * ...
simulation.X0' * (bold_response.obs_old-bold_signal_change);
bold_response.denoised_obs = bold_response.obs_old + intercept;
```

Intermediate Results:

```
if i==1 || ~mod(i,20)
 plot_results(fig_handle,state.proxy,simulation,ode_param.proxy.mean,...
    plot_handle,symbols,plot_settings,'not_final');
end
```








```
end
```

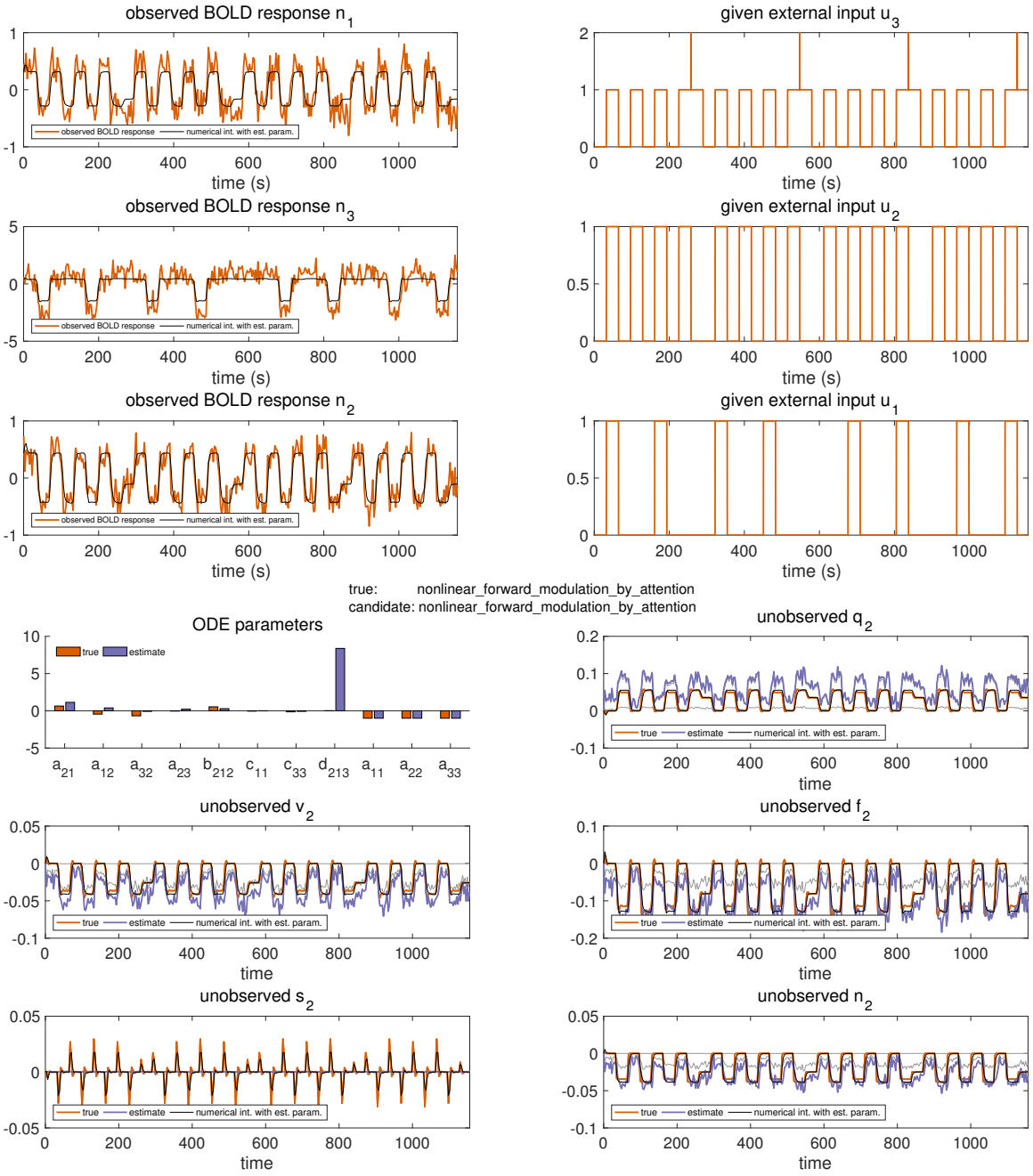
4.20 Numerical Integration with Estimated ODE Parameters

See whether we actually fit the BOLD responses well. Curves are shown in black.

```
simulation2 = simulation_old; simulation2.ode_param = ode_param.proxy.mean';
[simulation2,obs_to_state_relation] = simulate_state_dynamics_dcm(...
simulation2,symbols,ode,time,plot_settings,state.ext_input,'no plot');
state.proxy.num_int = simulation2.state;
```

Final Results

```
plot_results(fig_handle,state.proxy,simulation,ode_param.proxy.mean,...
plot_handle,symbols,plot_settings,'final',...
simulation2.bold_response_true,simulation.odes,candidate_odes);
```



4.21 Time Taken

```
disp(['time taken: ' num2str(toc) ' seconds'])
```

```
time taken: 25.8028 seconds
```

References

Gorbach, N.S. Validation and Inference of Structural Connectivity and Neural Dynamics with MRI data. 2018. ETH Zürich Doctoral Thesis. https://www.research-collection.ethz.ch/handle/20.500.11850/261734.

Gorbach*, N.S. , Bauer*, S. and Buhmann, J.M., Scalable Variational Inference for Dynamical Systems. 2017a. Neural Information Processing Systems (NIPS). https://papers.nips.cc/paper/7066-scalable-variational-inference-for-dynamical-systems.pdf. arXiv paper: https://arxiv.org/abs/1705.07079.

Bauer*, S. , Gorbach*, N.S. and Buhmann, J.M., Efficient and Flexible Inference for Stochastic Differential Equations. 2017b. Neural Information Processing Systems (NIPS). https://papers.nips.cc/paper/7274-efficient-and-flexible-inference-for-stochastic-systems.pdf.

Wenk, P., Gotovos, A., Bauer, S., Gorbach, N.S., Krause, A. and Buhmann, J.M., Fast Gaussian Process Based Gradient Matching for Parameters Identification in Systems of Nonlinear ODEs. 2018. In submission to Conference on Uncertainty in Artificial Intelligence (UAI). arxiv paper: https://arxiv.org/abs/1804.04378.

Calderhead, B., Girolami, M. and Lawrence. N.D., 2002. Accelerating Bayesian inference over nonlinear differential equation models. In Advances in Neural Information Processing Systems (NIPS) . 22.

*The authors have contributed equally to their respective papers.