half
un-
ob-
served

```
simulation.state_obs_variance = @(mean)(bsxfun(@times,[0.5^2,0.5^2],...
    ones(size(mean))));              % observation noise
simulation.ode_param = [2,1,4,1];    % true ODE parameters;
simulation.final_time = 2;           % end time for integration
simulation.int_interval = 0.01;      % integration interval
simulation.time_samp = 0:0.1:simulation.final_time; % sample times for observations
simulation.init_val = [5 3];         % initial state values

%symbols of observed states that appear in the 'ODEs.txt' file.
simulation.observed_states = {'[prey]','[predator]'};
odes_path = 'Lotka_Volterra_ODEs.txt';
```

$\mathbf{x}$

```
symbols.state = {'[prey]','[predator]'};    % symbols of states in 'ODEs.txt' file
```

θ

```
symbols.param = {'[\theta_1]','[\theta_2]','[\theta_3]','[\theta_4]'};
```

ϕ

```
kernel.param = [10,0.2];              % set values of rbf kernel parameters
```

γ

```
state.derivative_variance = 6*ones(1,length(symbols.state)); % gamma for gradient matching model
time.est = 0:0.1:4;                  % estimation times
opt_settings.pseudo_inv_type = 'Moore-Penrose';
opt_settings.coord_ascent_numb_iter = 40;  % number of coordinate ascent iterations

% The observed state trajectories are clamped to the trajectories
% determined by standard GP regression (Boolean)
opt_settings.clamp_obs_state_to_GP_fit = false;
plot_settings.size = [1200, 500]; plot_settings.layout = [1,3];
ode = import_odes(symbols,odes_path);
disp('ODEs:'); disp(ode.raw)
ODEs:
    '[\theta_1].*[prey] - [\theta_2].*[prey].*[predator]'
    '-[\theta_3].*[predator] + [\theta_4].*[prey].*[predator]'

[state,time,ode] = generate_ground_truth(time,state,ode,symbols,simulation,...
    odes_path);
if ~iscell(simulation.observed_states)
    ratio_observed = simulation.observed_states;
    state_obs_idx = zeros(1,simulation.numb_odes,'logical');
    idx = randperm(simulation.numb_odes);
    idx = idx(1:floor(simulation.numb_odes * ratio_observed));
    state_obs_idx(idx) = 1;
    simulation.observed_states = symbols.state(state_obs_idx);
end

[state,time,obs_to_state_relation] = generate_state_obs(state,time,simulation,...
    symbols);
state.sym.mean = sym('x%d%d',[length(time.est),length(ode.system)]);
state.sym.variance = sym('sigma%d%d',[length(time.est),length(ode.system)]);
ode_param.sym.mean = sym('param%d',[length(symbols.param),1]);
assume(ode_param.sym.mean,'real');
[h_states,h_param,p] = setup_plots(state,time,simulation,symbols,plot_settings);

tic; %start timer
[dC_times_invC,inv_C,A_plus_gamma_inv] = kernel_function(kernel,state,time.est);
coupling_idx = find_state_couplings_in_odes(ode,symbols);
```

$\mathbf{B}_{\theta k}\theta +$

$\mathbf{b}_{\theta k} \overset{!}{=}$

$\mathbf{f}_k(\mathbf{X},\theta)(5)$

$\mathbf{B}_{\theta k}$

$\mathbf{b}_{\theta k}$

$\mathbf{f}_k(\mathbf{X},\theta)$

θ

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

$\mathbf{f}(\mathbf{X},\theta)-$

$'\mathbf{C}_{\phi}\mathbf{C}_{\phi}^{-1}\mathbf{X}$

$\mathbf{x}_u$

$\mathbf{R}_{uk}\mathbf{x}_u +$

$\mathbf{r}_{uk} \overset{!}{=}$

$\mathbf{f}_k(\mathbf{X},\theta)$

$\mathbf{R}_{uk}$

$\mathbf{r}_{uk}$