Enhanced Particle Swarm Optimization Algorithms for Multiple-Input Multiple-Output System Modelling using Convolved Gaussian Process Models

Abstract: Convolved Gaussian Process (CGP) is able to capture the correlations not only between inputs and outputs but also among the outputs. This allows a superior performance of using CGP than standard Gaussian Process (GP) in the modelling of Multiple-Input Multiple-Output (MIMO) systems when observations are missing for some of outputs. Similar to standard GP, a key issue of CGP is the learning of hyperparameters from a set of input-output observations. It typically performed by maximizing the Log-Likelihood (LL) function which leads to an unconstrained nonlinear and non-convex optimization problem. Algorithms such as Conjugate Gradient (CG) or Broyden-Fletcher-Goldfarb-Shanno (BFGS) are commonly used but they often get stuck in local optima, especially for CGP where there are more hyperparameters. In addition, the LL value is not a reliable indicator for judging the quality intermediate models in the optimization process. In this paper, we propose to use enhanced Particle Swarm Optimization (PSO) algorithms to solve this problem by minimizing the model output error instead. This optimization criterion enables the quality of intermediate solutions to be directly observable during the optimization process. Two enhancements to the standard PSO algorithm which make use of gradient information and the multi-start technique are proposed. Simulation results on the modelling of both linear and nonlinear systems demonstrate the effectiveness of minimizing the model output error to learn hyperparameters and the performance of using enhanced algorithms.

Keywords: Enhanced PSO; Convolved Gaussian Process Models; Hyperparameters Learning

1 Introduction

Gaussian Process (GP) modelling is a non-parametric data-driven technique based on Bayesian theory. A major advantage of GP models, compared with parametric data-driven models such as Artificial Neural Network (ANN) and Fuzzy Models (FMs), is that the accuracy of the predicted outputs can be naturally measured through the variances that are computed as part of the modelling process. Another advantage is that GP models generally require fewer parameters (Kocijan 2011). These parameters, also known as hyperparameters, are estimated through a learning process using the measured input-output data of the system. GP models have found many applications in science and engineering (Bailer-Jones, Bhadeshia, and Mackay 1999; Ažman and Kocijan 2007; Wang, Fleet, and Hertzmann 2008; Gregorčič and Lightbody 2009; Yu 2012).

A standard GP model can be applied to a Multiple-Input Single-Output (MISO) system. For systems with multiple outputs, one can use a separate GP model for each output. This approach is referred to as Independent Gaussian Process (IGP) modelling. Its disadvantage is that since the GP models are independent of each other, any correlations between outputs will not be modelled (Boyle and Frean 2005; Alvarez and Lawrence 2009; Cao, Lai, and Alam 2014). An alternative way is to use Convolved Gaussian Process (CGP) models (Alvarez and Lawrence 2009), which are able to model not only the relationships between inputs and outputs but also correlations among all outputs. The importance of modelling this correlation becomes apparent when there are missing output data (Cao, Lai, and Alam 2014).

The hyperparameters of the CGP model can be estimated by maximizing a Log-Likelihood (LL) function. This maximization is typically performed by using gradient based solutions, such as Conjugate Gradient (CG) and Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithms. The algorithms are usually required to restart many times with different initial values to overcome the issue of getting stuck in local optima caused by the sensitiveness to initial values. Evolutionary algorithms, such as standard Particle Swarm Optimization (PSO), have been used as an alternative approach to learn the hyperparameters of GP (Zhu, Xu, and Dui 2010; Petelin and Kocijan 2011) and CGP model (Cao, Lai, and Alam 2014) due to they typically perform better than gradient based methods (Noel 2012). However, the issues of poor global search ability caused by poor initialization and slow convergence due to poor local search ability

remained in the existing works due to the use of standard PSO. In addition, a physically meaningful and reliable indicator of intermediate models' quality is preferred than the use of LL values.

In view of these shortcomings, we propose three enhanced PSO algorithms to solve the optimization problem of minimizing Mean Squared Error (MSE) values of model outputs. The first one is called multi-start PSO where the standard PSO is restarted several times to diversify the particles. The second one is the gradient-based PSO which makes use of gradient information to achieve faster convergence. The last one is a hybrid of these two methods that provides good particle diversity and faster convergence. These three algorithms are studied through the modelling of Multiple-Input Multiple-Output (MIMO) Linear Time-Varying (LTV) and Nonlinear Time-Varying (NLTV) systems. Furthermore, the use of MSE as fitness function provides us a direct and reliable indication of current solutions during the optimization process.

The rest of this article is organized as follows. Section 2 provides a brief overview of the CGP modelling technique. In Section 3, we reviewed the maximizing LL function problem for learning CGP model' hyperparameters, and defined the problem of minimizing MSE of model outputs. The standard PSO as well as three enhanced algorithms for the problems are introduced in Section 4. Simulation results comparing the proposed algorithms to standard PSO and CG approaches are presented and discussed in Section 5. Finally, Section 6 concludes the article.

2 Convolved Gaussian Process Models

Consider a system with n inputs $\mathbf{x} \in \mathbb{R}^n$ and m outputs $\mathbf{y}(\mathbf{x}) \in \mathbb{R}^m$ again. In the CGP, each output $\mathbf{y}_d(\mathbf{x})$ is modelled by,

$$\mathbf{y}_d(\mathbf{x}) = f_d(\mathbf{x}) + \epsilon_d(\mathbf{x}) \tag{1}$$

where $d = 1, 2, \ldots, m$ and $\epsilon_d(\mathbf{x})$ denotes an independent Gaussian white noise. The function $f_d(\mathbf{x})$ typically is defined by a linear convolution of a smoothing kernel $H_d(\mathbf{x})$ and a latent function $u(\mathbf{x})$,

$$f_d(\mathbf{x}) = \int H_d(\mathbf{x} - \tau) u(\tau) d\tau \tag{2}$$

The correlation between outputs is derived from the latent function $u(\mathbf{x})$ which has effects on all output functions. This latent function can be any appropriate random processes. If a Gaussian white noise is used, then resulting in a Dependent Gaussian Process (DGP) model. In the CGP, a wide range of latent functions are proposed to match the modelling requirements for different physical or dynamical systems (Alvarez 2011).

In addition, the CGP models allow using more than one type of latent function. Assuming Q groups of latent functions are considered, where for the q^{th} group, it has R_q smoothing kernels. Thus the d^{th} output function can be rewritten by,

$$f_d(\mathbf{x}) = \sum_{q=1}^{Q} \sum_{k=1}^{R_q} \int H_{d,q}^k(\mathbf{x} - \tau) u_q^k(\tau) d\tau \tag{3}$$

Then, the covariance between different outputs $\mathbf{y}_d(\mathbf{x})$ and $\mathbf{y}_{d'}(\mathbf{x}')$ can be obtained by,

$$\begin{aligned} \mathbf{K}_{\mathbf{y}_d, \mathbf{y}_{d'}}(\mathbf{x}, \mathbf{x}') =& Cov[\mathbf{y}_d(\mathbf{x}), \mathbf{y}_{d'}(\mathbf{x}')] \\ =& Cov[f_d(\mathbf{x}), f_{d'}(\mathbf{x}')] + Cov[\epsilon_d(\mathbf{x}), \epsilon_{d'}(\mathbf{x}')] \delta_{d,d'} \end{aligned} \tag{4}$$

where $\delta_{d,d'}$ is a Kronecker delta function thus $Cov[\epsilon_d(\mathbf{x}), \epsilon_{d'}(\mathbf{x}')] \delta_{d,d'}$ will lead to a diagonal matrix of noise variance $\left\{\sigma_d^2\right\}_{d=1}^m$ if it is assumed that $\epsilon_d(\mathbf{x}) \sim \mathcal{N}(0, \sigma_d^2)$, and the cross-covariance between $f_d(\mathbf{x})$ and $f_{d'}(\mathbf{x}')$ is given by,

$$\begin{aligned} &\mathbf{K}_{f_d, f_{d'}}(\mathbf{x}, \mathbf{x}') = Cov[f_d(\mathbf{x}), f_{d'}(\mathbf{x}')] \\ &= E\left[\sum_{q=1}^{Q} \sum_{k=1}^{R_q} \int H_{d,q}^k(\mathbf{x} - \tau) u_q^k(\tau) d\tau \sum_{q=1}^{Q} \sum_{k=1}^{R_q} \int H_{d',q}^k(\mathbf{x}' - \tau') u_q^k(\tau') d\tau'\right] \\ &= \sum_{q=1}^{Q} \sum_{k=1}^{R_q} k_q(\tau, \tau') \int H_{d,q}^k(\mathbf{x} - \tau) H_{d',q}^k(\mathbf{x}' - \tau) d\tau \end{aligned} \tag{5}$$

Data-driven modelling using CGP basically involves obtaining the appropriate smoothing kernels and latent functions that reflect the covariance between outputs.

As given in (3), the output function is a linear combination of independent random functions. Thus, if these functions are Gaussian processes, then $f_d(\mathbf{x})$ will also be a Gaussian process. In this case, the smoothing kernels can be expressed by,

$$H^k_{d,q}(\gamma) = \frac{\nu^k_{d,q}\left|\mathbf{P}^k_{d,q}\right|^{1/2}}{(2\pi)^{M/2}} \exp\left[-\frac{1}{2}(\gamma)^T\mathbf{P}^k_{d,q}(\gamma)\right] \tag{6}$$

where $\nu^k_{d,q}$ is a length-scale coefficient, $\mathbf{P}^k_{d,q}$ is an $n \times n$ precision matrix of the smoothing kernel. To simplify the model further, it is assumed that the covariances of latent functions $k_q(\eta)$ in every group are all same Gaussian,

$$k_q(\eta) = \frac{\upsilon_q\left|\mathbf{P}_q\right|^{1/2}}{(2\pi)^{M/2}} \exp\left[-\frac{1}{2}(\eta)^T\mathbf{P}_q(\eta)\right] \tag{7}$$

where υ_q is the length-scale coefficient and $\mathbf{P}_q$ is another $n \times n$ precision matrix.

To simplify the discussion again, it is assumed that $R_q = 1$ for all Q groups of latent functions. In addition, the precision matrices of the smoothing kernels are assumed to be the same for each group of latent functions. As a result, given the smoothing kernel (6) and latent function covariance (7), the covariance can be obtained by,

$$Cov[f_d(\mathbf{x}), f_{d'}(\mathbf{x}')] = \sum_{q=1}^{Q} \frac{\nu_{d,q}\nu_{d',q}\upsilon_q}{(2\pi)^{M/2}\left|\mathbf{P}\right|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{x}')^T\mathbf{P}^{-1}(\mathbf{x}-\mathbf{x}')\right] \tag{8}$$

where $\mathbf{P} = \mathbf{P}^{-1}_{\mathbf{d}} + \mathbf{P}^{-1}_{\mathbf{d}'} + \mathbf{P}^{-1}_{\mathbf{q}}$. Note that this multiple-output covariance function maintains a Gaussian form, i.e. $\mathbf{K}_{\mathbf{f_d},\mathbf{f_{d'}}}(\mathbf{x},\mathbf{x}') \sim \mathcal{N}(\mathbf{x}-\mathbf{x}'|0,\mathbf{P})$.

Then similar to standard GP models, given a set of observations $\left\{\mathbf{x}^j,\mathbf{y}_j\right\}_{j=1}^{J_d}$, where $\sum_{d=1}^{m} J_d = N$, a Gaussian distribution can be defined on the output functions by,

$$\mathbf{y}(\mathbf{x}) \sim \mathcal{N}\left(\mu(\mathbf{x}), \mathbf{K}_{\mathbf{y},\mathbf{y}}(\mathbf{x},\mathbf{x}')\right) \tag{9}$$

where the output vector $\mathbf{y}(\mathbf{x})$ is given by,

$$\mathbf{y}(\mathbf{x}) = [\mathbf{y}_1(\mathbf{x}), ..., \mathbf{y}_m(\mathbf{x})]^T \tag{10}$$

with the entries,

$$\mathbf{y}_d(\mathbf{x}) = \left[\mathbf{y}_d(\mathbf{x}^1), \mathbf{f_d}(\mathbf{x}^2), ..., \mathbf{f_d}(\mathbf{x}^{J_d})\right]^T \tag{11}$$

Without loss of generality, zero means are used. In addition, the covariance matrix $\mathbf{K}_{\mathbf{y},\mathbf{y}}(\mathbf{x},\mathbf{x}') \in \mathbb{R}^{N\times N}$ can be obtained by using (5) and (8). Usually, the computation of such a covariance matrix is computationally expensive. Thus, some sparse approximations have been proposed to reduce the complexities of CGP (Alvarez and Lawrence 2009). Then, the marginal likelihood can be defined by,

$$p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) \sim \mathcal{N}(\mathbf{y}|0,\mathbf{K}_{\mathbf{y},\mathbf{y}}) \tag{12}$$

The joint distribution of observed $\mathbf{y}$ and the predicted outputs $\mathbf{y}^* = \{y^*_1, \cdots, y^*_M\}$ at new input $\mathbf{x}^*$ is thus still a Gaussian and is given by,

$$\begin{bmatrix}\mathbf{y}\\ \mathbf{y}^*\end{bmatrix} \sim \mathcal{N}\left(0, \begin{matrix}\mathbf{K}_{\mathbf{y},\mathbf{y}} & \mathbf{K}_{\mathbf{f},\mathbf{f}^*}\\ \mathbf{K}_{\mathbf{f}^*,\mathbf{f}} & \mathbf{K}_{\mathbf{f}^*,\mathbf{f}^*}\end{matrix}\right) \tag{13}$$

Finally, similar to standard GP models again, the predictive distribution is a Gaussian,

$$\mathbf{y}^*|\mathbf{X},\mathbf{y},\boldsymbol{\theta},\mathbf{x}^* \sim \mathcal{N}(\mu(\mathbf{x}^*), \sigma^2(\mathbf{x}^*)) \tag{14}$$

where the mean $\mu(\mathbf{x}^*)$ and variance $\sigma^2(\mathbf{x}^*)$ functions are computed by,

$$\begin{aligned}\mu(\mathbf{x}^*) &= \mathbf{K}_{\mathbf{f}^*,\mathbf{f}}\mathbf{K}^{-1}_{\mathbf{y},\mathbf{y}}\mathbf{y}\\ \sigma^2(\mathbf{x}^*) &= \mathbf{K}_{\mathbf{f}^*,\mathbf{f}^*} - \mathbf{K}_{\mathbf{f}^*,\mathbf{f}}\mathbf{K}^{-1}_{\mathbf{y},\mathbf{y}}\mathbf{K}_{\mathbf{f},\mathbf{f}^*}\end{aligned} \tag{15}$$

Table 1 NLL and MSE values of two CGP models of system described by (20).

	Model 1	Model 2
NLL	≈ 51	≈ 242269
MSE	0.5313	0.0101

3 Hyperparameter Learning of CGP Models

3.1 Maximizing the Log-Likelihood Function

When doing predictions using (15), the covariance matrix $\mathbf{K}$ is required to be specified by a set of appropriate hyperparameters $\boldsymbol{\theta}$. They are usually obtained by maximizing the log of marginal likelihood function.

In CGP models, the marginal likelihood is equal to the integral over a product of the likelihood function and CGP prior over the latent functions, both of which are Gaussian. Thus, the marginal likelihood is also Gaussian and defined by,

$$\begin{aligned} p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) &= \int p(\mathbf{y}|\mathbf{f},\mathbf{X},\boldsymbol{\theta})p(\mathbf{f}|\boldsymbol{\theta})d\mathbf{f} \\ &= \frac{1}{(2\pi)^{\frac{N}{2}}|\mathbf{K}_{\mathbf{y},\mathbf{y}}|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\mathbf{y}^T\mathbf{K}_{\mathbf{y},\mathbf{y}}^{-\mathbf{1}}\mathbf{y}\right) \end{aligned} \tag{16}$$

This marginal likelihood can be viewed as the likelihood of hyperparameters corrupted by noise so that we simply call likelihood function. A good point estimate $\hat{\boldsymbol{\theta}}$ of hyperparameters can be subsequently obtained by maximizing this likelihood function. In practice, we usually estimate the hyperparameters by maximizing the log likelihood function due to its less computation complexities. The corresponding optimization problem can be subsequently defined as,

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \log p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) \tag{17}$$

where,

$$\log p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T\mathbf{K}_{\mathbf{y},\mathbf{y}}^{-\mathbf{1}}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}_{\mathbf{y},\mathbf{y}}| - \frac{N}{2}\log 2\pi \tag{18}$$

The unconstrained optimization problem (17) is not easy to solve due to it is typically nonlinear and non-convex. However, in CGP models, the derivatives of log likelihood function with respective to (w.r.t.) the hyperparameters $\boldsymbol{\theta}$ are mathematically analytical and can be obtained by,

$$\frac{\partial}{\partial\boldsymbol{\theta}_l}\log p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T\mathbf{K}_{\mathbf{y},\mathbf{y}}^{-\mathbf{1}}\frac{\partial\mathbf{K}}{\partial\boldsymbol{\theta}_l}\mathbf{K}_{\mathbf{y},\mathbf{y}}^{-\mathbf{1}}\mathbf{y} - \frac{1}{2}\mathrm{trace}(\mathbf{K}_{\mathbf{y},\mathbf{y}}^{-\mathbf{1}}\frac{\partial\mathbf{K}}{\partial\boldsymbol{\theta}_l}) \tag{19}$$

where $\boldsymbol{\theta}_l$ represents the l^{th} entry of hyperparameters $\boldsymbol{\theta}$.

3.2 Minimizing the MSE Function

Equation (18) is the natural choice as the objective function for the hyperparameter learning problem. However, there are some issues involved which we shall illustrate with the modelling of a single output nonlinear dynamic system. The system is described by the following difference equation:

$$y(k) = 0.893y(k-1) + 0.037y^2(k-1) - 0.05y(k-2) + 0.157u(k-1) - 0.05u(k-1)y(k-1) \tag{20}$$

where $u(k)$ is the input and $y(k)$ is the output at time instant k. 1000 uniformly distributed input values are randomly generated within the range $(-2, 4)$ and the corresponding outputs are computed. From these input-output data, 200 are randomly chosen for training the model. The hyperparameters of the CGP model are learned by minimizing the negative of the LL (NLL) function. The quality of the resulting CGP model is evaluated by computing the MSE of the outputs given by

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{y}_i - \hat{\mathbf{y}}_i(\boldsymbol{\theta})\right)^2 \tag{21}$$

using a different set of 50 values. Here, N is the number of test data, $\mathbf{y}_i$ are the i^{th} observed output values, and $\hat{\mathbf{y}}_i$ is corresponding mean value of the predictive distribution obtained by (15) given the hyperparameters θ.

Table 1 shows two different CGP models that results from limiting the search range of the hyperparameters to $[0, 100]$ for Model 1 and $[0, 1]$ for Model 2. From the MSE values, it is clear that Model 2 is able to predict the outputs

Initialization
PSO parameters: $N_p, c_1, c_2, \lambda_1, \lambda_2, \omega_{\text{start}}, \omega_{\text{end}}, k, T_{\max}$ and ξ
Randomly generated $\boldsymbol{\theta}$;
while $t < T_{max}$ **do**
  if $f(\mathbf{G}) \leq \xi$ **then**
    End;
  else
    for $i = 1$ *to* N_p **do**
      for $d = 1$ *to* D **do**
        Update $v_i^d(t)$ by using (25);
        Update $x_i^d(t)$ by using (27);
      end
      Update $\mathbf{P}_i$ and $V_i^{\text{pbest}}(t)$ by using (28);
      Update $\mathbf{G}$ and $V^{\text{gbest}}(t)$ by using (29);
    end
  end
  s $t = t + 1$;
end
Output: Optimized particle $\boldsymbol{\theta}_{\text{opt}}$.

Algorithm 1: Standard PSO based Hyperparameter Learning

more accurately compared with Model 1. However, the NLL value of Model 1 is much smaller than Model 2. If the NLL function is the objective function for minimization, one may conclude that Model 1 is the better model. Thus one cannot use the NLL (and hence the LL) values to accurately gauge the quality of the intermediate models obtained during the optimization process.

We therefore propose to minimize the MSE function (21) to learn CGP's hyperparameters by,

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{y}_i - \hat{\mathbf{y}}_i(\theta)\right)^2 \tag{22}$$

In addition, the following derivatives of MSE of outputs w.r.t. hyperparameters can be used to accelerate the optimization process,

$$\frac{\partial}{\partial \boldsymbol{\theta}_l} MSE = -\frac{2}{N} \sum_{i=1}^{N} \left\{ \left(\mathbf{y}_i - \hat{\mathbf{y}}_i(\boldsymbol{\theta})\right) \frac{\partial \hat{\mathbf{y}}_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right\} \tag{23}$$

with

$$\frac{\partial \hat{\mathbf{y}}_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{\partial \mathbf{K}_{\mathbf{f}^*,\mathbf{f}}}{\partial \boldsymbol{\theta}} \mathbf{K}_{\mathbf{y},\mathbf{y}}^{-\mathbf{1}} \mathbf{y} - \mathbf{K}_{\mathbf{f}^*,\mathbf{f}} \mathbf{K}_{\mathbf{y},\mathbf{y}}^{-\mathbf{1}} \frac{\partial \mathbf{K}_{\mathbf{y},\mathbf{y}}}{\partial \boldsymbol{\theta}} \mathbf{K}_{\mathbf{y},\mathbf{y}}^{-\mathbf{1}} \mathbf{y} \tag{24}$$

where the computation of $\frac{\partial \mathbf{K}_{\mathbf{f}^*,\mathbf{f}}}{\partial \boldsymbol{\theta}}$ and $\frac{\partial \mathbf{K}_{\mathbf{y},\mathbf{y}}}{\partial \boldsymbol{\theta}}$ can be found in (Rasmussen and Williams 2006; Alvarez and Lawrence 2011). This technique is in fact widely known as the least-square approach in the literature. In addition, from the viewpoint of non-Bayesian learning, minimizing the MSE is approximately equivalent to maximizing the LL. The proof of equivalence between these two learning strategies can be found in (Myung 2003).

4 Enhanced PSO Algorithms

In (Zhu, Xu, and Dui 2010; Petelin and Kocijan 2011; Cao, Lai, and Alam 2014), the standard PSO algorithm has been proven superior to gradient based CG and BFGS approaches in terms of accuracy and efficiency for the optimization problems (17) and (22). However, poor initializations can lead to poor global search ability, and they exhibit slow convergence due to poor local search ability. In this section, three enhancements are proposed to address these issues.

4.1 Standard PSO

We shall first outline the standard PSO algorithm for the hyperparameter learning of CGP models. Let there be a population of N_p particles, each of which, denoted by $\mathbf{x}_i = [x_i^1, \cdots, x_i^D]_{i=1,\cdots,N_p}^T \in \mathbb{R}^D$, represents a potential solution to the problem (17) or (22). Each particle also records its best position as $\mathbf{P}_i = [p_i^1, \cdots, p_i^D]^T$ and its best fitness value $V_i^{\text{pbest}} = f(\mathbf{P}_i)$, where $f(\cdot)$ denotes the fitness function and could be (18) or (21). In addition, the best position of all N_p particles is denoted by $\mathbf{G} = [g^1, \cdots, g^D]^T$ and the corresponding best fitness value is denoted by $V^{\text{gbest}} = f(\mathbf{G})$. In the iteration $t+1$, the velocity of i^{th} particle, given by $\mathbf{v}_i = [v_i^1, \cdots, v_i^D]^T$, along d^{th} dimension is updated according to the following rule,

$$v_i^d(t+1) = \omega(t) v_i^d(t) + c_1 \lambda_1 \left(p_i^d(t) - x_i^d(t)\right) + c_2 \lambda_1 \left(g^d(t) - x_i^d(t)\right) \tag{25}$$

where c_1 and c_2 are two acceleration factors, λ_1 and λ_2 are two random values between $[0, 1]$, $\omega(t)$ represents an inertia factor.

In general, a PSO algorithm consists of two search phases, known as "exploration" and "exploitation" respectively. They are governed by the inertia factor $\omega(t)$. The use of a larger value of $\omega(t)$ allows the particle to explore larger areas of the search space during the exploration phase. Meanwhile, a smaller value of $\omega(t)$ restricts the particle to a smaller region of the search space and allows the particle to converge to a local optimum in the exploitation phase. Thus, the inertia factor is usually reduced with time step. A commonly used $\omega(t)$ is defined by,

$$\omega(t) = \omega_{\text{end}} + (\omega_{\text{start}} - \omega_{\text{end}}) \exp(-k \times (\frac{t}{T_{\max}})) \tag{26}$$

where ω_{start} and ω_{end} are the pre-determined start and final values respectively, $T_{\max}$ denotes the maximum number of iterations. and the rate of decrease is governed by the constant k.

The new position of a particle can subsequently be obtained by,

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{27}$$

For the minimization problem (22), the $\mathbf{P}_i$ and V_i^{pbest} at $t+1$ iteration are updated according to the following rule,

$$\begin{aligned} \mathbf{P}_i(t+1) &= \begin{cases} \mathbf{x}_i(t+1) & f(\mathbf{x}_i(t+1)) \leq f(\mathbf{P}_i(t)) \\ \mathbf{P}_i(t) & f(\mathbf{x}_i(t+1)) > f(\mathbf{P}_i(t)) \end{cases} \\ V_i^{\text{pbest}}(t+1) &= f(\mathbf{P}_i(t+1)) \end{aligned} \tag{28}$$

In addition, the $\mathbf{G}$ and V^{gbest} at $t+1$ iteration are updated by,

$$\begin{aligned} \mathbf{G}(t+1) &= \text{argmin} \left\{ f(\mathbf{P}_1(t+1)), \cdots, f(\mathbf{P}_{N_p}(t+1)), f(\mathbf{G}(t)) \right\} \\ V^{\text{gbest}}(t+1) &= f(\mathbf{G}(t+1)) \end{aligned} \tag{29}$$

We can also use the rules (28) and (29) when the maximization problem (17) becomes the minimizing the negative of LL function (18).

For our hyperparameter learning problem, each particle is defined by

$$\boldsymbol{\theta} = \{\boldsymbol{\theta}_{K1}, ..., \boldsymbol{\theta}_{KM}, \boldsymbol{\theta}_{L1}, ..., \boldsymbol{\theta}_{LQ}\} \tag{30}$$

where $\boldsymbol{\theta}_{Kd} = \{\nu_{d1}, ..., \nu_{dQ}, \mathbf{P_d}\}_{d=1,\cdots,M}$ represents the hyperparameters of smoothing kernels (6), and $\boldsymbol{\theta}_{Lq} = \{\upsilon_q, \mathbf{P_q}\}_{q=1,...,Q}$ are the hyperparameters of latent functions (7). The algorithm of standard PSO based hyperparameter learning is presented in Algorithm 1.

4.2 Multi-Start PSO

In the "exploration" stage of optimization process, we want the particles to explore as much of the search space as possible. This can be achieved by setting the inertia factor $\omega(t)$ to a suitably large value which in turn is determined by ω_{start} and ω_{end} in (26). However, suitable values for these two constants are quite specific to each problem. Another way to achieve this objective is to diversify the swarm by introducing new particles. In this paper, all particles will be reinitialized if the global best position $\mathbf{G}$ remains unchanged or slightly changed for a given number of iterations N_G. This is referred as the multi-start PSO algorithm. One issue remained in the proposed algorithm is that the potentials of old particles may not be sufficiently exploited. This issue can be ignored due to we care the global search ability more than local one in the "exploration" stage. In addition, it has been proposed that only those particles that are trapped in a local optimum should be reinitialized (An et al. 2010). However, the rest of particles may still need to be reinitialized later. Besides, this approach requires checking the changes of multiple $f(\mathbf{P}_i)$. The proposed algorithm is therefore simpler due to only the change of $f(\mathbf{G})$ is checked. Algorithm 2 describes the approach of learning CGP models' hyperparameters through using the multi-start PSO.

4.3 Gradient-based PSO

Standard PSO also suffers from slow convergence during the "exploitation" phase. This issue can be solved through using the gradient/derivative information especially when approaching to the global or local optima. In this paper, a gradient-based PSO is proposed for the hyperparameters learning problem by combining the standard PSO and CG algorithm. In particular, the current global best position $\mathbf{G}$ will be exploited by solving the problem 17) or (22) by using the CG algorithm. The obtained solution is subsequently used to replace the current global position in the PSO

Initialization
PSO parameters: $N_p, c_1, c_2, \lambda_1, \lambda_2, \omega_{\text{start}}, \omega_{\text{end}}, k, T_{\max}$ and ξ
Randomly generated $\boldsymbol{\theta}$;
Multi-start PSO parameters: $\eta, N_G,\ N_\eta = 0$;
while $t < T_{max}$ **do**
  if $N_\eta = N_G$ **then**
    Randomly regenerated $\boldsymbol{\theta}$;
    $N_\eta = 0$;
  else
    if $f(\mathbf{G}) \leq \xi$ **then**
      End;
    else
      for $i = 1$ *to* N_p **do**
        for $d = 1$ *to* D **do**
          Update $v_i^d(t)$ by using (25);
          Update $x_i^d(t)$ by using (27);
        end
        Update $\mathbf{P}_i$ and $V_i^{\text{pbest}}(t)$ by using (28);
        Update $\mathbf{G}$ and $V^{\text{gbest}}(t)$ by using (29);
      end
      if $\|f(\mathbf{G}(t)) - f(\mathbf{G}(t-1))\| \leq \eta$ **then**
        $N_\eta = N_\eta + 1$;
      else
        $N_\eta = 0$;
      end
    end
  end
  $t = t + 1$;
end
Output: Optimized particle $\boldsymbol{\theta}_{\text{opt}}$.

Algorithm 2: Multi-Start PSO based Hyperparameter Learning

Initialization
PSO parameters: $N_p, c_1, c_2, \lambda_1, \lambda_2, \omega_{\text{start}}, \omega_{\text{end}}, k, T_{\max}$ and ξ
Randomly generated $\boldsymbol{\theta}$;
Gradient-based PSO parameters: $\eta, N_G,\ N_\eta = 0$;
while $t < T_{max}$ **do**
  if $N_\eta = N_G$ **then**
    Initializing CG parameters, $\boldsymbol{\theta}_0 = \mathbf{G}$;
    Solving the problem (17) or (22) to obtain $\boldsymbol{\theta}^*$;
    if $f(\boldsymbol{\theta}^*) \leq f(\mathbf{G}(t))$ **then**
      $\mathbf{G}(t+1) = \boldsymbol{\theta}^*$;
    else
      $\mathbf{G}(t+1) = \mathbf{G}(t)$;
    end
    $N_\eta = 0$;
  else
    if $f(\mathbf{G}) \leq \xi$ **then**
      End;
    else
      for $i = 1$ *to* N_p **do**
        for $d = 1$ *to* D **do**
          Update $v_i^d(t)$ by using (25);
          Update $x_i^d(t)$ by using (27);
        end
        Update $\mathbf{P}_i$ and $V_i^{\text{pbest}}(t)$ by using (28);
        Update $\mathbf{G}$ and $V^{\text{gbest}}(t)$ by using (29);
      end
      if $\|f(\mathbf{G}(t)) - f(\mathbf{G}(t-1))\| \leq \eta$ **then**
        $N_\eta = N_\eta + 1$;
      else
        $N_\eta = 0$;
      end
    end
  end
  $t = t + 1$;
end
Output: Optimized particle $\boldsymbol{\theta}_{\text{opt}}$.

Algorithm 3: Gradient-based PSO based Hyperparameter Learning

algorithm if it produces a better fitness value. Compared with the existing work in (Noel 2012) where all particles are exploited by using a gradient-based method, the proposed algorithm only conducts gradient-based search on the current global best position if its fitness value remains unchanged or slightly changed for a specified number of iterations N_G. The computational burden of using proposed algorithm is essentially reduced. The gradient based PSO for the hyperparameter learning of CGP models is given in Algorithm 3.

Initialization
PSO parameters: $N_p, c_1, c_2, \lambda_1, \lambda_2, \omega_{\text{start}}, \omega_{\text{end}}, k, T_{\max}$ and ξ
Randomly generated $\boldsymbol{\theta}$;
Hybrid PSO parameters: $\tau, \eta, N_G,\ N_\eta = 0$;
while $t < T_{max}$ **do**
  if $N_\eta = N_G$ **then**
    if $t \leq \tau \times T_{max}$ **then**
      Randomly regenerated $\boldsymbol{\theta}$;
    else
      Initializing CG parameters, $\boldsymbol{\theta}_0 = \mathbf{G}$;
      Solving the problem (17) or (22) to obtain $\boldsymbol{\theta}^*$;
      if $f(\boldsymbol{\theta}^*) \leq f(\mathbf{G}(t))$ **then**
        $\mathbf{G}(t+1) = \boldsymbol{\theta}^*$;
      else
        $\mathbf{G}(t+1) = \mathbf{G}(t)$;
      end
    end
    $N_\eta = 0$;
  else
    if $f(\mathbf{G}) \leq \xi$ **then**
      End;
    else
      for $i = 1$ *to* N_p **do**
        for $d = 1$ *to* D **do**
          Update $v_i^d(t)$ by using (25);
          Update $x_i^d(t)$ by using (27);
        end
        Update $\mathbf{P}_i$ and $V_i^{\text{pbest}}(t)$ by using (28);
        Update $\mathbf{G}$ and $V^{\text{gbest}}(t)$ by using (29);
      end
      if $\|f(\mathbf{G}(t)) - f(\mathbf{G}(t-1))\| \leq \eta$ **then**
        $N_\eta = N_\eta + 1$;
      else
        $N_\eta = 0$;
      end
    end
  end
  $t = t + 1$;
end
Output: Optimized particle $\boldsymbol{\theta}_{\text{opt}}$.

Algorithm 4: Hybrid PSO based Hyperparameter Learning

4.4 Hybrid PSO

The multi-start method in Section 4.2 and the gradient-based method in Section 4.3 can be combined in a single PSO algorithm so that both the "exploration" and the "exploitation" phases of the optimization process are enhanced. This leads to the hybrid PSO algorithm. In particular, the multi-start technique is first used such that the search space can be well covered. When the number of iterations N_G reaches a given proportion η of maximum iteration number, the optimization process is considered to have approached near global or local optima. The algorithm subsequently switches to the use of gradient-based technique. This allows a faster convergence rate due to the nature of using gradient-based solution compared to the use of rules (25) and (27). The proposed hybrid PSO is conceptually simple and allows to adjust the proportion η to suit the problem. The use of hybrid PSO in the problem of CGP models' hyperparameter learning is given in Algorithm 4.

5 Performance Evaluation

The optimization performances of proposed PSO based algorithms for CGP hyperparameters learning are demonstrated in the modelling of non-trivial MISO and MIMO systems. The proposed PSOs are compared with the CG algorithm. In addition, both the Negative value of Log-Likelihood (NLL) and MSE are used as the fitness function, respectively. The simulations are repeated 50 times with same training and test samples. The results in terms of MSE value, convergence rate and computer run-time are averaged values.

All simulations are performed on a computer with a 3.40GHz Intel® Core™ 2 Duo CPU with 16 GB RAM, using Matlab® version 8.1. In addition, the parameters related to the CGP and PSO models in the simulations are listed in the Table 2. Note that the number of restarts for CG is designed to give a fair comparison with PSO with the given population size and number of iterations.

Table 2 Parameters used in the simulations

Symbol	Description	Quantity
N_p	PSO population	20
$\mathbf{T}_{\max}$	Maximum Iterations	500
c_1,c_2	Acceleration Factors	1.5
ω_{start}	Start Inertial Factor	0.4
ω_{end}	End Inertial Factor	0.9
k	Shape Control Factor	0.8
CG Restarts	Restart Times	20×500
$\|\Delta\xi\|$	Minimum Fitness Variation	10^{-5}
$\nu_{d,i}, \upsilon_q$ α_i, β_j	Coefficients Search Range $\mathbf{P_d}$, $\mathbf{P_q}$ Elements Search Range	$[0, 100]$ for LTV
		$[0, 100]$ for NLTV with "Step"
		$[0, 1]$ for NLTV with "Curve"

Table 3 Comparison of two PSOs with different population sizes

N_p	MAE		Var	
	PSO/1	PSO/2	PSO/1	PSO/2
10	0.2297	0.2355	1.52e-02	2.44e-02
25	0.0054	0.0047	9.05e-03	3.64e-03
50	0.0022	0.0021	8.66e-03	4.37e-03
100	0.0011	0.0012	9.14e-03	9.74e-04

5.1 Standard PSO with MSE Fitness

We study the effectiveness and benefit of PSO based hyperparameters learning method. The proposed PSO with MSE fitness is compared to existing NLL fitness based PSO, as well as commonly used CG algorithm. The convergence of proposed PSO is not discussed due to it is not our concern in this simulation.

5.1.1 Single-Output Modelling

The system (20) is used as a numerical example in the simulation. Although this dynamical system has only 1 input and 1 output, the CGP modelling inputs will be $u(k-1)$, $y(k-1)$ and $y(k-2)$, making it a 3-input and 1-output model. Only a single output is used here for modelling to simplify the comparison. In addition, we randomly chose 1000 inputs in $u \sim \mathcal{U}(-2, 4)$ and apply them into the system. This allows us collect 1000 observations including inputs, states and outputs.

First we aim to verify the effectiveness of minimizing the model errors by using standard PSO in the hyperparameters learning problem. The 50 simulations are performed with the same 200 training and 50 test data randomly selected from the 1000 observations. To study the influence of PSO population size, the simulations are also independently performed for each population size of 10, 25, 50 and 100. The obtained results are given in Table 3 in terms of Mean Absolute Error (MAE) and average variance (Var) values, where PSO/1 represents the proposed approach and PSO/2 denotes the PSO with NLL fitness. Overall, the both two approaches produce equally good CGP model due to close MAE and Var values. In addition, the results also suggest that a value of 25 to 50 may be a good choice of PSO population size. This is because a bigger size normally requires much more runtime (exponentially increasing).

Next, we want to determine the effect of the hyperparameters search space. Two different cases are considered here. In the first case, it is assumed that a prior knowledge of value ranges for the parameters in (30) is available. Specifically, they are

$$\begin{aligned} \nu_{d,i}, \upsilon_q &\in \{1, 2\} \\ \alpha_i, \beta_j &\in \{0, 1\} \end{aligned} \tag{31}$$

where α_i and β_j are the elements of the diagonal precision matrices $\mathbf{P_d}$ and $\mathbf{P_q}$ respectively. In the second case, we do not assume any prior knowledge of the value ranges. The obtained MAE values using the proposed PSO are given in Figure 1 and are compared to those using CG. The results show that the learnt CGP models by using PSO and CG perform equally good when the search space is well defined. Figure 2a confirms that the predicted outputs are very close to the actual values. However, when the search space is not well constrained, the proposed PSO outperforms CG by a wide margin. Figure 2b shows that while the predicted CGP outputs by the PSO are still very close to the actual values, there are some clear deviations with by the CG.

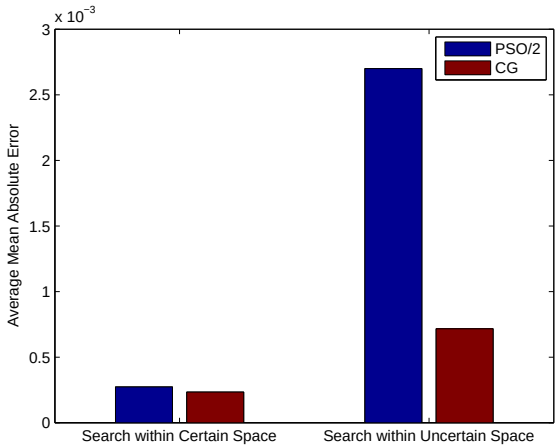
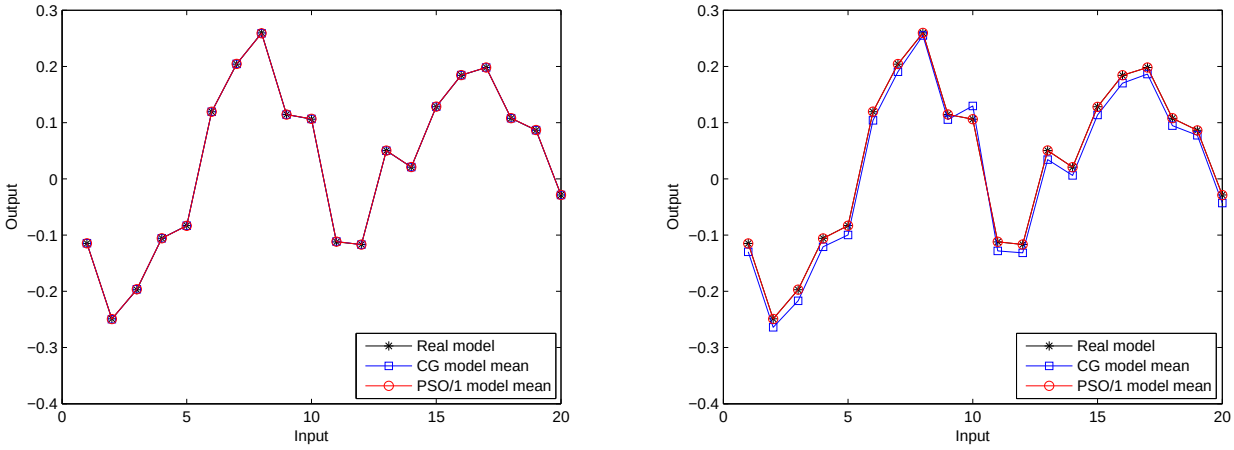


Figure 1 Obtained MAE in the single-output dynamical system modelling over 50 runs



(a) Well defined search range

(b) Not well defined search range

Figure 2 Predicted outputs in the single-output simulations

Table 4 Results of Linear Relationship

	MAE		SE	
	PSO	CG	PSO	CG
$\mathbf{y}_1$	1.41E-04	4.46E-04	0084	0.0327
$\mathbf{y}_2$	1.11E-04	2.18E-04	0.147	0.0446

5.1.2 Two-output Modelling

Systems with multiple-outputs can be modelled in two different ways. One is to use multiple single-output models and the other is to provide a single model for all outputs at the same time. While the first approach is often simpler, the latter approach is able to capture correlation between outputs. For example, a robot arm system with multiple degrees of freedom has multiple outputs that are strongly correlated. Another example is the prediction of steel mechanical properties in (Gaffour, Mahfouf, and Yang 2010), where the yield and tensile strength are predicted from the chemical compositions and grain size. These two "outputs" are highly correlated.

We shall continue to use the dynamical system in (20). Since it has only one output y (denoted y_1 here), a second output y_2 will be created which is a function of y_1. Two such functions are considered, one linear and the other nonlinear, given by $\mathbf{y}_2 = -\mathbf{y}_1$ and $\mathbf{y}_2 = \mathbf{exp}(\mathbf{y}_1)$ respectively. Two different sets of training data, each has 200 samples, are selected from the 1000 observations. The test data consists of 50 samples which are different from the training samples.

Tables 4 and 5 show the performances of using PSO and CG. For outputs $\mathbf{y}_1$ and $\mathbf{y}_2$ in both two MIMO systems, the CGP models learnt by using PSO exhibit smaller MAE and standard error (SE) values.

Table 5 Results of Nonlinear Relationship

	MAE		SE	
	PSO	CG	PSO	CG
$\mathbf{y}_1$	4.41E-04	5.36E-04	0.0065	0.043
$\mathbf{y}_2$	3.57E-04	8.11E-04	0.0064	0.0356

Table 6 CGP model accuracies over 50 runs for the LTV system.

	PSO				CG
	Standard	Gradient	Multi-Start	Hybrid	
MSE of y_1	0.4413	0.1159	0.3882	0.1975	0.2235
MSE of y_2	0.4934	0.1699	0.3806	0.1556	0.2473
Time	≈18s	≈23s	≈18s	≈19s	$\gg$ 5min

5.2 Enhanced PSO Algorithms

Next, we demonstrate the optimization performances of proposed enhanced PSO algorithms for CGP hyperparameters learning problem. Three enhanced PSOs are compared with the standard PSO and CG algorithms in the modelling two non-trivial MIMO systems, with respect to the model accuracy, runtime as well as convergence.

5.2.1 LTV System Modelling

Consider a 2-input-2-output LTV system (Majji 2009) defined by,

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{A}(t)\cdot\mathbf{x}(t) + \mathbf{B}(t)\cdot\mathbf{u}(t)\\ \mathbf{y}(t) &= \mathbf{C}(t)\cdot\mathbf{x}(t) + \mathbf{D}(t)\cdot\mathbf{u}(t)\end{aligned} \tag{32}$$

where $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{D}$ are defined as:

$$\begin{aligned}\mathbf{A}(t) &= \begin{bmatrix} 0.3-0.9\Gamma_{1t} & 0.1 & 0.7\Gamma_{2t} \\ 0.6\Gamma_{1t} & 0.3-0.8\Gamma_{2t} & 0.01 \\ 0.5 & 0.15 & 0.6-0.9\Gamma_{1t}\end{bmatrix}\\ \mathbf{B} &= \begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 0 & 1\end{bmatrix}\ \mathbf{C} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & -1 & 0\end{bmatrix}\ \mathbf{D} = 0.1\begin{bmatrix} 1 & 0 \\ 0 & 1\end{bmatrix}\end{aligned} \tag{33}$$

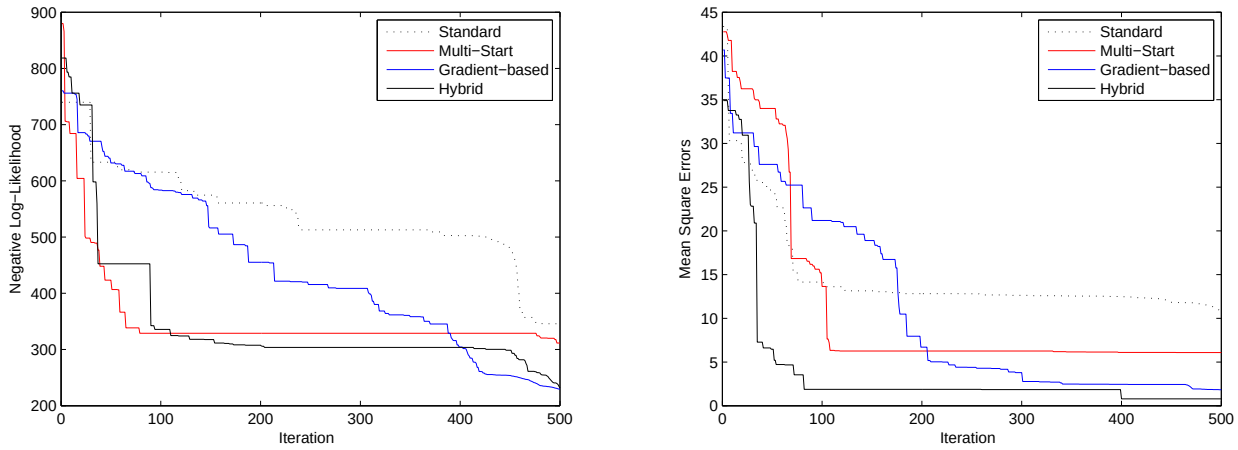
Matrix $\mathbf{A}$ has time-varying parameters $\Gamma_{1t} = \sin(10t)$ and $\Gamma_{2t} = \cos(10t)$. The two control inputs are given by $u_1(t) = 0.5\sin(12t)$ and $u_2(t) = \cos(7t)$. They have zero initial conditions.

Using a sampling interval of $0.05s$, 200 data records which include the inputs, states and outputs are generated. Out of these data, 60 samples are selected randomly for training the CGP models, and all 200 samples are employed for testing.

First, the results of using LL fitness function are discussed. Table 6 shows the MSE of the predicted outputs of the obtained CGP models. These results show that, the gradient and hybrid PSO algorithms are able to produce smaller MSE than the other two PSO algorithms. However, the runtime of the gradient PSO is longer than the other three PSO algorithms due to gradient computations. It is also interesting to note that the CG algorithm took significantly longer to converge to the solution than PSO algorithms.

The convergence behaviour of the PSO algorithms with LL fitness function is shown in Figure 3a, averaged over 50 runs. It shows that both the hybrid and multi-start algorithms converges to a near-optimal solution much faster than the other two alternatives. Thus it can be concluded that these two algorithms perform better global search ability at the early stages (approximately before 150 iterations) than standard and gradient PSO.

Next we discuss the results of using MSE fitness. Due to similar MSE values and runtime of 4 PSO algorithms are produced, therefore they are not shown here. The convergence behaviour of the PSO algorithms with MSE fitness function averaged over 50 runs are given in Figure 3b. The results show again that, overall the hybrid PSO has the best optimization performance than other 3 PSO algorithms due to the global search ability as well as multi-start PSO and local search ability as well as gradient PSO.



(a) Log-Likelihood Fitness

(b) MSE Fitness

Figure 3 Convergence behaviour of the four PSO algorithms in modelling the LTV system

5.2.2 NLTV System Modelling

The second simulation involves the CGP modelling of a NLTV system controlled by a Partial Form Dynamic Linearization (PFDL) based Model-Free Adaptive Control (MFAC) controller with the same parameters as in (Hou and Jin 2011). The 4-input and 2-output numerical system is described by,

$$\begin{aligned}
x_{11}(k+1) =& \frac{x_{11}(k)^2}{1+x_{11}(k)^2} + 0.3x_{12}(k) \\
x_{12}(k+1) =& \frac{x_{11}(k)^2}{1+x_{12}(k)^2+x_{21}(k)^2+x_{22}(k)^2} + a(k)u_1(k) \\
x_{21}(k+1) =& \frac{x_{21}(k)^2}{1+x_{21}(k)^2} + 0.2x_{22}(k) \\
x_{22}(k+1) =& \frac{x_{21}(k)^2}{1+x_{11}(k)^2+x_{12}(k)^2+x_{22}(k)^2} + b(k)u_2(k) \\
y_1(k+1) =& x_{11}(k+1) + 0.005 * \mathrm{rand}(1) \\
y_2(k+1) =& x_{21}(k+1) + 0.005 * \mathrm{rand}(1)
\end{aligned} \tag{34}$$

where the time-varying parameters are given by,

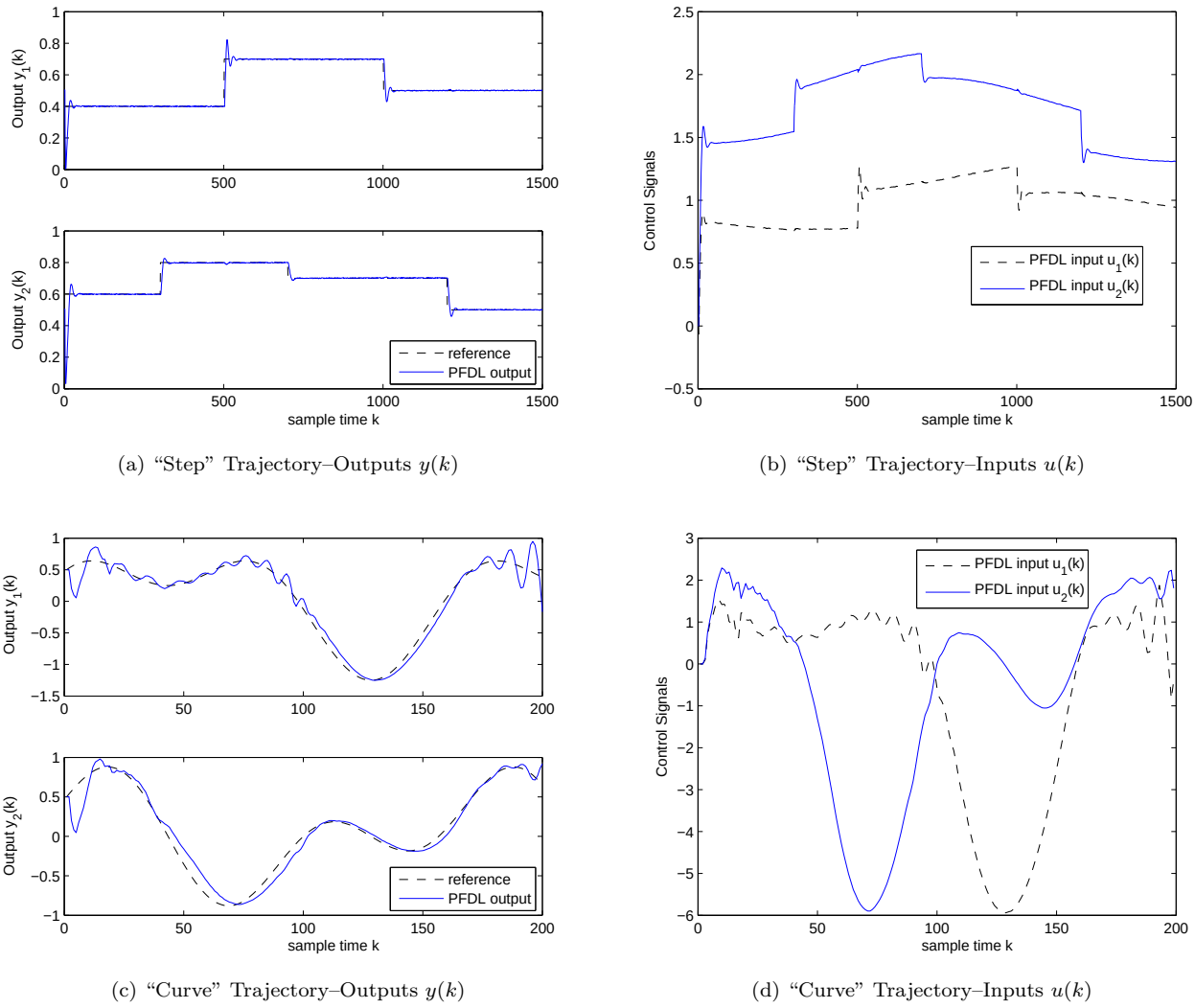
$$\begin{aligned}
a(k) &= 1 + 0.1\sin(2\pi k/1500) \\
b(k) &= 1 + 0.1\cos(2\pi k/1500)
\end{aligned} \tag{35}$$

This system is to track two trajectories. One involves a "Step" trajectory given by,

$$\begin{aligned}
y_1^*(k) &= \begin{cases} 0.4 & k \le 500 \\ 0.7 & 500 < k \le 1000 \\ 0.5 & 1000 < k \le 1500 \end{cases} \\
y_2^*(k) &= \begin{cases} 0.6 & k \le 300 \\ 0.8 & 300 < k \le 700 \\ 0.7 & 700 < k \le 1200 \\ 0.5 & 1200 < k \le 1500 \end{cases}
\end{aligned} \tag{36}$$

the other is "Curve" trajectory specified by,

$$\begin{aligned}
y_1^*(k) &= 0.75\sin(\frac{\pi k}{8}) + 0.5\cos(\frac{\pi k}{4}) \\
y_2^*(k) &= 0.5\cos(\frac{\pi k}{8}) + 0.5\sin(\frac{\pi k}{4})
\end{aligned} \tag{37}$$



(a) "Step" Trajectory–Outputs $y(k)$

(b) "Step" Trajectory–Inputs $u(k)$

(c) "Curve" Trajectory–Outputs $y(k)$

(d) "Curve" Trajectory–Inputs $u(k)$

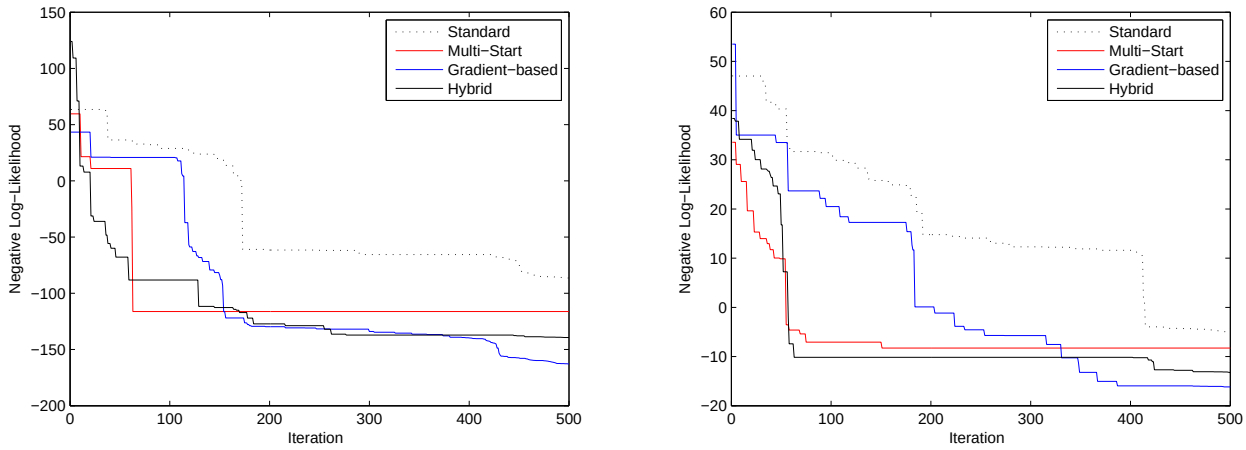
Figure 4 Reference PFDL inputs and outputs for the two trajectories

The initial values of the system are (Zhang, Ge, and Lee 2005): $x_{11}(1) = x_{11}(2) = x_{21}(1) = x_{21}(2) = 0.5$, $x_{12}(1) = x_{12}(2) = x_{22}(1) = x_{22}(2) = 0$, and $u_1(1) = u_1(2) = u_2(1) = u_2(2) = 0$. The reference outputs and inputs for the two trajectories are shown in Figures 4.

First, we discuss the results of using LL as fitness function. Table 7 shows the results of CGP modelling of the NLTV system tracking the both "Step" and "Curve" trajectories using the five different algorithms for hyperparameter learning. These results are average of 50 independent simulations. All three enhanced PSO algorithms outperform the standard PSO as well as CG. Specifically, the hybrid PSO produces the second best model accuracy (slightly bigger than the best one of gradient PSO), and while the runtime is comparable to the standard and multi-start PSO. Figure 5a shows the convergence behaviour of the various PSO algorithms for the "Step" trajectory. In this case, the hybrid and multi-start PSO converge faster than the other PSO algorithms initially and the hybrid algorithm eventually achieves a smaller model error. The similar convergence behaviours of the "Curve" trajectory can be found in Figure 5b .

Next, the convergence behaviours of PSO algorithms when using MSE fitness are discussed. For the "Step" trajectory simulation, three enhanced PSOs approach the near-optima faster than standard PSO algorithm while gradient and hybrid PSOs eventually achieve the smaller model errors. In addition, for the "Curve" trajectory simulation, three enhanced PSOs perform better optimization ability than standard PSO algorithms again. Specifically, the multi-start and hybrid PSOs converge faster while the gradient and hybrid PSOs are able to produce smaller MSE values.

The trade-off between the size of training data and the model accuracy is demonstrated using the hybrid PSO algorithm. The results, given in Table 8, are again averaged over 50 simulations. The training data are uniformly chosen from the control interval shown in Figures 4b and 4d. As expected, the model accuracy improves as the training data size increases. However, the algorithm runtime increases exponentially with data size. For a data size of 100, the

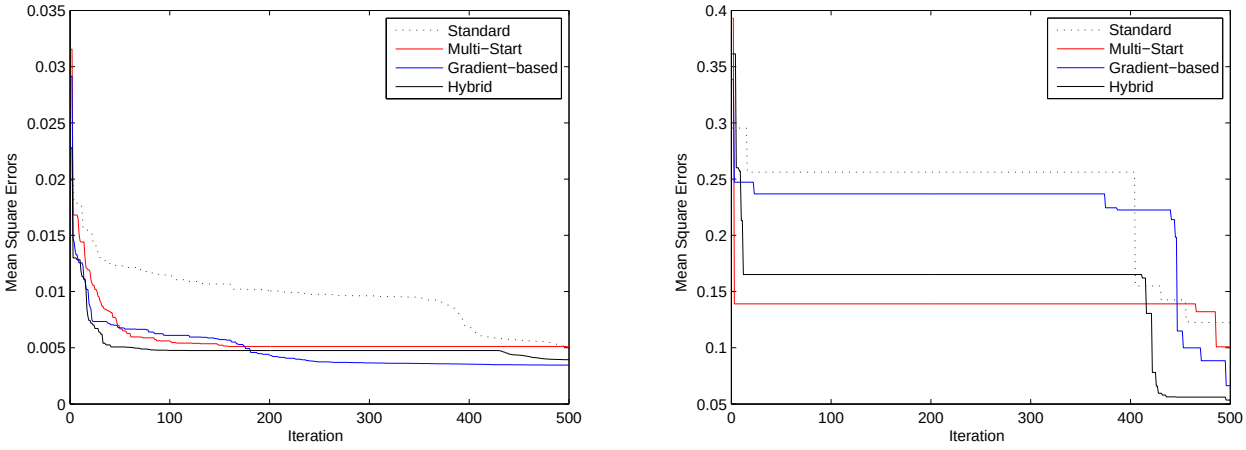


(a) "Step" Trajectory (b) "Curve" Trajectory

Figure 5 Convergence behaviour of the four PSO algorithms with LL fitness in modelling the NLTV system

Table 7 CGP model accuracies over 50 runs for the NLTV system.

	PSO				CG
	Standard	Gradient	Multi-Start	Hybrid	
"Step" Trajectory					
MSE of y_1	0.8378	0.0084	0.0179	0.0124	0.1221
MSE of y_2	0.2218	0.0062	0.0337	0.0127	0.1273
Time	≈28s	≈40s	≈28s	≈31s	$\gg$ 5min
"Curve" Trajectory					
MSE of y_1	0.3083	0.0417	0.1594	0.0633	0.1541
MSE of y_2	0.1627	0.0402	0.1098	0.0516	0.2333
Time	≈27s	≈39s	≈27s	≈29s	$\gg$ 5min



(a) "Step" Trajectory (b) "Curve" Trajectory

Figure 6 Convergence behaviour of the four PSO algorithms with MSE fitness in modelling the NLTV system

model error and PSO runtime for both trajectories are similar. Furthermore, the results also show that the system with piecewise constant outputs ("Step" trajectory) can be modelled with far fewer training data compared with the one with smooth outputs ("Curve" trajectory). This is despite the jump discontinuities in the outputs.

Table 8 Effects of training data size on model error and hybrid PSO runtime.

Training Data Size	Average MSE		Time
	y_1	y_2	
"Step" Trajectory			
20	0.0377	0.0511	≈12s
40	6.1475e-04	7.611e-04	≈17s
100	1.1292e-04	1.3543e-04	≈31s
200	1.3411e-05	1.8854e-05	≈110s
"Curve" Trajectory			
25	0.0562	0.0665	≈14s
50	0.0031	0.0032	≈18s
75	0.0012	0.0011	≈23s
100	1.1712e-04	1.9201e-04	≈29s

6 Conclusion

The hyperparameters of the GP models are conventionally learnt by minimizing the NLL function. This typically leads to an unconstrained nonlinear non-convex optimization problem that is usually solved by using the CG algorithm. Three enhanced PSO algorithms have been proposed in this chapter to improve the hyperparameter learning for CGP models of MIMO systems. They make use of gradient-based technique and also combine it with the multi-start technique. Using numerical LTV and NLTV systems, we have shown that these algorithms are more effective in avoiding getting stuck in local optima. Hence they are able to produce more accurate models of the systems. Results showed that the hybrid PSO algorithm allows the faster convergence and produces the more accurate models. These algorithms also use the MSE of model outputs rather than the LL function as the fitness function of optimization problems. This enables us to assess the quality of intermediate solutions more directly.

References

Alvarez, Mauricio and Neil D Lawrence (2009). "Sparse Convolved Gaussian Processes for Multi-output Regression". In: *Advances in Neural Information Processing Systems*, pp. 57–64.

Alvarez, Mauricio A (2011). "Convolved Gaussian process priors for multivariate regression with applications to dynamical systems". PhD thesis. University of Manchester.

Alvarez, Mauricio A and Neil D Lawrence (2011). "Computationally efficient convolved multiple output Gaussian processes". In: *Journal of Machine Learning Research* 12.May, pp. 1459–1500.

An, Ru et al. (2010). "A Modified PSO Algorithm for Remote Sensing Image Template Matching". In: *Photogrammetric engineering and remote sensing* 76.4, pp. 379–389.

Ažman, Kristjan and Juš Kocijan (2007). "Application of Gaussian processes for black-box modelling of biosystems". In: *ISA Transactions* 46.4, pp. 443–457.

Bailer-Jones, C.A.L., H.K.D.H. Bhadeshia, and D. J. C. Mackay (1999). "Gaussian process modelling of austenite formation in steel". In: *Materials Science and Technology* 15.3, pp. 287–294.

Boyle, Phillip and Marcus Frean (2005). "Dependent Gaussian Processes". In: *Advances in Neural Information Processing Systems*. MIT Press, pp. 217–224.

Cao, Gang, Edmund M-K Lai, and Fakhrul Alam (2014). "Particle Swarm Optimization for Convolved Gaussian Process Models". In: *International Joint Conference on Neural Networks (IJCNN)*. IEEE. Beijing, China, pp. 1573–1578.

Gaffour, Sidahmed, Mahdi Mahfouf, and Yong Yao Yang (2010). "'Symbiotic' data-driven modelling for the accurate prediction of mechanical properties of alloy steels". In: *IEEE Proceedings of International Conference of Intelligent Systems*. IEEE, pp. 31–36.

Gregorčič, Gregor and Gordon Lightbody (2009). "Gaussian process approach for modelling of nonlinear systems". In: *Engineering Applications of Artificial Intelligence* 22.4, pp. 522–533.

Hou, Zhongsheng and Shangtai Jin (2011). "Data-driven model-free adaptive control for a class of MIMO nonlinear discrete-time systems". In: 22.12, pp. 2173–2188.

Kocijan, Juš (2011). "Control Algorithms Based on Gaussian Process Models: A State-of-the-Art Survey". In: *Proceedings of the Special International Conference on Complex Systems: Synergy of Control, Communications and Computing - COSY 2011*. Ohrid, Republic of Macedonia, pp. 69–80.

Majji, Manoranjan (2009). "System identification: time varying and nonlinear methods". PhD thesis. Texas A&M University.

Myung, In Jae (2003). "Tutorial on maximum likelihood estimation". In: *Journal of mathematical Psychology* 47.1, pp. 90–100.

Noel, Mathew M (2012). "A new gradient based particle swarm optimization algorithm for accurate computation of global minimum". In: *Applied Soft Computing* 12.1, pp. 353–359.

Petelin, Dejan and Jußs Kocijan (2011). "Control system with evolving Gaussian process models". In: *IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS)*. IEEE, pp. 178–184.

Rasmussen, CE and CKI Williams (2006). *Gaussian Processes for Machine Learning*. en. Cambridge, MA, USA: MIT Press, p. 248.

Wang, Jack M, David J Fleet, and Aaron Hertzmann (2008). "Gaussian process dynamical models for human motion". In: 30.2, pp. 283–298.

Yu, Jie (2012). "A nonlinear kernel Gaussian mixture model based inferential monitoring approach for fault detection and diagnosis of chemical processes". In: *Chemical Engineering Science* 68.1, pp. 506–519.

Zhang, Jin, Shuzhi Sam Ge, and Tong Heng Lee (2005). "Output feedback control of a class of discrete MIMO nonlinear systems with triangular form inputs". In: 16.6, pp. 1491–1503.

Zhu, Fuwei, Chong Xu, and Guansuo Dui (2010). "Particle swarm hybridize with Gaussian Process Regression for displacement prediction". In: *IEEE Proceedings of International Conference on Bio-Inspired Computing: Theories and Applications*. IEEE, pp. 522–525.