Implementing KPCA-based speaker adaptation methods with different optimization algorithms in a Persian ASR system

Zohreh Ansari\textsuperscript{a, *}, Farshad Almasgani\textsuperscript{a}

\textsuperscript{a}Department of Biomedical Engineering, Amirkabir University of Technology, Tehran, Iran

Abstract

In this paper, two kernel eigenspace-based speaker adaptation methods are implemented using FARSDAT database and their performances are compared with eigenspace-based ones. In the conducted experiments, short lengths of adaptation speech data (2-5 seconds) are used. Experimental results show that 4.5% improvement in phoneme recognition rate is achieved by supervised eigenspace-based methods. Implementing kernel eigenspace-based methods, 0.6% improves the results gained by utilizing eigenspace-based methods in 2 seconds of adaptation data. While, with this amount of data, traditional speaker adaptation methods cannot work efficiently. In addition, in this work, we employ another optimization algorithm instead of usual numerical methods, which is particle swarm optimization (PSO) and its performance in achieving rapid optimization is investigated.

Keywords: eigenspace-based speaker adaptation; kernel eigenspace-based speaker adaptation; kernel principal component analysis; particle swarm optimization; rapid speaker adaptation

1. Introduction

Acoustic mismatch reduction due to speaker variability between the training and testing conditions is a problem in automatic speech recognition (ASR) systems. In order to tackle this problem, speaker adaptation methods are introduced. Among speaker adaptation methods, eigenspace-based ones have shown to rapidly improve the performance of ASRs. It has been shown that eigenspace-based speaker adaptation methods such as eigenvoice (EV) (Kuhn, Junqua, Nguyen, & Niedzielski, 2000) and eigenspace-based MLLR (EMLLR) (Chen, Liau, Wang, & Lee, 2000) are more effective than traditional ones such as maximum a posteriori (MAP) (Gauvain & Lee, 1994) and maximum likelihood linear regression (MLLR) (Gales & Woodland, 1996) for rapid adaptation with a little amount of adaptation data. In these techniques, principal component analysis (PCA) (Jolliffe, 1986) is applied on the...
speaker space constructed from some parameters of training speaker models, in order to extract those dimensions carrying the most acoustic variations between training speaker parameters.

Although the extracted eigenvectors from the speaker space by employing PCA include some intra speaker variations, it is claimed that nonlinear generalizations of PCA such as kernel PCA (KPCA) (Schölkopf, Smole, & Müller, 1998) may extract more information by performing PCA on the data mapped from the input space to a high dimensional feature space.

In one of the KPCA-based speaker adaptation methods which is called KEMLLR, the MLLR transformation matrix for the new speaker is constructed in the feature space. Then, by using isotropic Gaussian kernels, the adapted model in speaker space can be calculated from the product of this matrix with the mean vector of speaker independent model in the kernel induced feature space (Mak, Hsiao, Ho, & Kwok, 2006). Another adaptation KPCA-based adaptation approach is eKEV. In this method, a pre-image in the input space is considered for the adapted new speaker model in the kernel induced feature space. Finding the optimal corresponded pre-image is done by making use of distance constrains of the approximated pre-image with a set of reference speaker models in the input space in the least square sense (Kwok & Tsang, 2003; Mak & Hsiao, 2007).

In the adaptation stage of both introduced KPCA-based speaker adaptation methods, the kernel eigenvoice/matrice weights are calculated by maximizing the likelihood of the adaptation data with respect to new speaker model through EM algorithm. Due to the nonlinearity characteristics of kernel functions, there are some algorithms like as generalized expectation maximization (GEM) (Dempster, Laird, & Rubin, 1977) which are used instead of EM. In GEM, some gradient-based numerical methods (Bazarr, Sherali, & Shetty, 2006) are used to improve the value of a target function in each maximization step.

There are some other optimization methods other than gradient-based ones which are proposed to maximize the continuous nonlinear target function. One of these algorithms is particle swarm optimization (PSO) (Clerc & Kennedy, 2002) which iteratively optimizes the function without making use of the gradient function.

The rest of this paper is organized as follows: first of all KPCA and the preimage problem is discussed in section 2. In section 3, the performance of KEMLLR and eKEV methods is explained in detail. The examined optimization algorithm as a replacement of gradient-based ones is introduced in section 4. By presenting experimental evaluations on FARDAT in section 5, the paper ends with discussion and conclusion in section 6.

2. Kernel Principal Component analysis and its Pre-image problem

In order to perform KPCA in a space constructed from a set of patterns (\(\{x_1, x_2, \ldots, x_N\} \in X\)), at first, these patterns must be mapped through a nonlinear mapping function \(\varphi\) to a higher dimensional space called feature space. Then, linear PCA is applied on this space, extracting those dimensions carrying the most variation between the patterns. To this end, at first, eigenvectors of the centered kernel matrix, defined as (1), are extracted through conducting eigen-decomposition as (2).

\[
\tilde{K}_{ij} = \varphi(x_i)^T \varphi(x_j) = HHK, H = I - \frac{1}{N} 11',1 = [1,\ldots,1]' (1)
\]

\[
\tilde{K} = \Psi U U', U = [a_1, \ldots, a_M], a_i = [\alpha_{i1}, \ldots, \alpha_{iN}]', \Psi = diag(\lambda_1, \ldots, \lambda_M) (2)
\]
Where, \( I \) is an identity matrix of dimension \( N \), and \( K \) is the non-centered kernel matrix. In the following, eigenvectors of the covariance matrix of the centered mapped patterns in the feature space are calculated as (3).

\[
v_m = \sum_{i=1}^{N} \frac{\alpha_{mi}}{\sqrt{\lambda_m}} \phi(x_i)
\]  

(3)

In some applications such as de-noising, it is required to map back the de-noised patterns existing in the subspace constructed from the kernel eigenvectors to input space. This problem involves finding the pre-image of these de-noised patterns in the input space. However, there is not an exact pre-image in this space; so, some algorithms are proposed to approximate that. One of these algorithms which is offered in (Gauvain & Lee, 1994), uses the relationship between feature space distances and input space ones, besides the idea of multidimensional scaling to embed the pre-image in the input space.

3. Kernel eigenspace-based speaker adaptation methods

3.1. KEMLLR adaptation

Performing KEMLLR adaptation involves the following steps:

1. Mapping the training speaker transformation supervectors to feature space and calculating the kernel matrix.

These transformation supervectors mapped to feature space are constructed by concatenating the rows of each MLLR matrix, calculated for each training speaker. During computations, it is revealed that in order to compute the mean vectors of the new speaker model, it is needed to access the information of each row. So, each row is mapped through a particular kernel function. In this way, the kernel matrix is defined by using composite kernels as it is shown in (4).

\[
k(x_i, x_j) = \sum_{r=1}^{R} k_r(x_{ir}, x_{jr})
\]  

(4)

2. Performing PCA in the feature space.

In this step, the eigenvectors, called eigenmatrices of the mapped patterns to feature space are extracted by applying PCA, as indicated in section 2.

3. Citing the new speaker adapted model in terms of eigenmatrix weights.

It is assumed that the new speaker transformation supervector mapped to the feature space, is a point located in the subspace expanded by the eigenvectors of the kernel feature space. With this in mind, the new speaker adapted model is computed as in (5) by estimating the similarity between each component of the mapped transformation supervector and the augmented vector of each Gaussian mean vector of the speaker independent model \( \xi_{g(si)} \) in the feature space as shown in (6)-(7).
\[ \mu_{kemlhr}^{(gr)} = \bar{y}_{hr} \xi_{s}(s) + \frac{1}{2} \left[ \xi_{s}(s) \right]^{2} + \frac{1}{\beta_{hr}} \log \left( k_{hr}^{(kemlhr)} \left( \hat{y}_{hr}, \xi_{s}(s) \right) \right) \]

\[ \xi_{s}(s) = [(\mu_{g}^{(s)})', 1]^T \]

Where,

\[ k_{hr}^{(kemlhr)}(\hat{y}_{hr}, \xi_{s}(s)) \equiv \varphi_{hr}^{(kemlhr)}(\hat{y}_{hr})\varphi_{hr}(\xi_{s}(s)) \]

\[ A_{hr}(g) + \sum_{m=1}^{M} \frac{w_{m}}{\sqrt{\lambda_{m}}} B_{hr}(m, g) \]

\[ A_{hr}(g) = \varphi_{hr}' \varphi_{hr}(\xi_{s}(s)) = \frac{1}{N} \sum_{i=1}^{N} k_{hr}(\hat{y}_{hr}^{(i)}, \xi_{s}(SI)) \]

\[ B_{hr}(m, g) = \sum_{i=1}^{N} \alpha_{mi} \left( k_{hr}(\hat{y}_{hr}^{(i)}, \xi_{s}(s)) - A_{hr}(g) \right) \]

Where, \( \mu_{gr} \) is the rth component of the gth Gaussian mean vector of the adapted model, \( \bar{y}_{hr} \) is the mean vector of the rth row vectors of the hth regression MLLR matrices, computed for all the training speakers, and \( \hat{y}_{hr} \) is the rth row vector of the hth regression MLLR matrix achieved by a weighted combination of eigenmatrices. The vector \( \theta \) in (5) is a vector of zeros with the same dimension of the \( \xi_{s}(s) \)’s vector.

4. Calculating the eigenmatrix weights.

In order to compute the speaker adapted model, it is necessary to calculate the eigenmatrix weights. To this end, these weights are estimated by maximizing the Q function defined as follows.

\[ Q(w) = -\frac{1}{2} \sum_{r=1}^{R} \sum_{t=1}^{T} \gamma_{t}(r) \left\| o_{t} - \mu_{g}(w) \right\|_{C_{r}}^{2} \]

By differentiating \( Q(w) \) with respect to each eigenmatrix weight, the gradient function is given by

\[ \frac{\partial Q(w)}{\partial w_{m}} = 2 \sum_{r=1}^{R} \sum_{t=1}^{T} \gamma_{t}(g)(o_{t} - \mu_{g}(w))C_{r}^{-1} \frac{\partial \mu_{g}(w)}{\partial w_{m}} \]

\[ \frac{\partial \mu_{kemlhr}^{(gr)}}{\partial w_{m}} = \frac{1}{2\beta_{hr}} \frac{B_{hr}(m, g)}{\sqrt{\lambda_{m}}} \left[ \frac{B_{hr}(m, g)}{k_{hr}^{(kemlhr)}(\hat{y}_{hr}, \xi_{s}(s))} - \frac{B_{hr}(m, g)}{k_{hr}^{(kemlhr)}(\hat{y}_{hr}, 0)} \right] \]

In (9), \( B_{hr}(m, -1) \) is computed when \( \xi_{s}(s) \) is substituted with \( \theta \). As it is shown in (9), the differentiated \( Q \) is a nonlinear function of \( w_{m} \). So, there is no closed form solution for computing the weights. The proposed algorithms in these situations are described in section 4.
3.2. eKEV adaptation

Performing eKEV adaptation involves the following steps:

1. Mapping each training speaker supervector to the kernel feature space and expanding the kernel eigenspace by the eigenvectors extracted through performing PCA in the feature space.

2. Expressing the new speaker adapted model in the constructed kernel eigenspace in terms of linear interpolation of kernel eigenvectors as follows.

\[
\tilde{\phi}_r^{(KEV)}(s_r) = \sum_{m=1}^{K} \sum_{i=1}^{N} w_m \alpha_{mi} \sqrt{\lambda_m} \tilde{\phi}(\mu_{ir})
\]

In the above equations, \( \mu_{ir} \) is the mean vector of the \( r \)th Gaussian corresponded to the \( i \)th training speaker model. And, \( \mathbf{w} \) is the vector of eigenmatrix weights. Moreover, \( \mathbf{s}_r \) is the mean vector of the \( r \)th Gaussian of the new speaker adapted model.

3. Estimating the similarity between the transformed adapted model and some reference speaker models in the feature space.

Reference speaker models can be chosen in different ways, one of them is selecting those speaker dependent models which are close to speaker independent one in terms of mahalanobis distance. By making use of this similarity, the distances between the expected pre-image and training speaker models in input space are given by (11).

\[
d_j = \left\| z_r^{(eKEV)} - z_j \right\|^2 = \sum_{r=1}^{R} d_{jr}
\]

\[
d_{jr} = -\frac{1}{\beta_r} \log(k_r^{(KEV)}(s_r^{(eKEV)}, z_{jr}))
\]

\[
k_r(s_{zr}^{(eKEV)}, z_{jr}) = \tilde{\phi}_r(s_{zr}^{(eKEV)}) \tilde{\phi}_r(z_{jr}) = A_r(j) + \sum_{m=1}^{M} \frac{w_m}{\sqrt{\lambda_m}} B_r(m, j)
\]

\[
A_r(j) = \frac{1}{N} \sum_{i=1}^{N} k_r(z_{ir}, z_{jr})
\]

\[
B_r(m, j) = \sum_{i=1}^{N} \alpha_{mi}(k_r(z_{ir}, z_{jr}) - A_r(j))
\]

In (12), \( z_{jr} \), is the \( r \)th Gaussian mean vector of the \( i \)th reference speaker model.

4. Assuming that the preimage of the new speaker adapted model in the feature space is in the subspace constructed by the eigenvectors of the centered matrix of reference speaker models (\( Z \)).
By making use of the distance constraints, calculated in (11), and the above assumption, the pre-image model is embedded into the speaker space as follows.

\[
\mathbf{s}_{x}^{(eKEV)} = C^{1/2} \mathbf{s}_{z}^{(eKEV)} = C^{1/2} (\mathbf{P} \times \mathbf{d}(w) + \mathbf{q})
\]  

(13)

Where, \( \mathbf{P} \) and \( \mathbf{q} \) are achieved by obtaining the singular value decomposition of \( \mathbf{Z} \) as shown in (14).

\[
\tilde{\mathbf{Z}} = \mathbf{U} \Psi \mathbf{V}' = \mathbf{UL}
\]

\[
\mathbf{P} = -\frac{1}{2} \mathbf{U} \Psi^{-1} \mathbf{V}'
\]

\[
\mathbf{q} = -\mathbf{P} \times \mathbf{d}_{0} + \tilde{\mathbf{Z}} , \mathbf{d}_{0} = [\| \mathbf{u}_{1} \|^2, \| \mathbf{u}_{2} \|^2, ..., \| \mathbf{u}_{n} \|^2]'
\]  

(14)

5. Calculating the eigenvector weights as it was indicated for achieving the values of eigenmatrix weights in KEMLLR adaptation.

By substituting (13) into the \( Q(w) \) function of (8) and differentiating it with respect to each eigenvector weight \( w \), we get

\[
\frac{\partial Q}{\partial w_{m}} = \sum_{r=1}^{R} \sum_{t=1}^{T} \gamma_{t}(r) \left( a_{t} - \mathbf{s}_{x}^{(eKEV)}(w) \right)^{T} \mathbf{C}_{r}^{-1} \times \mathbf{C}_{r}^{1/2} \mathbf{P}_{r} \frac{\partial d_{w}}{\partial w_{m}}
\]

\[
\frac{\partial d_{j}}{\partial w_{m}} = -\frac{1}{\sqrt{\lambda}_{m}} \sum_{r=1}^{R} \frac{B_{r}(m, j)}{\beta_{r} \times \tilde{k}_{r}^{(KEV)}(w_{j}, \mathbf{z}_{jr})} j = 1, ..., n
\]

(15)

From (14), it is revealed that the gradient of \( Q \) is nonlinear in \( w \) and there is no closed form solution for calculating the weights. Therefore, the following proposed are implemented.

4. Implemented optimization algorithms

As it was indicated in sections 3.1 and 3.2, there is no closed form solution for maximizing the likelihood function with respect to weight vectors due to the nonlinearity of kernel function. In these conditions, GEM algorithm is applied in which numerical methods like as gradient ascent and quasi-Newton method are used to improve the value of the weights during each maximization step.

Recently, some algorithms are proposed for optimizing the continuous nonlinear functions. One of these algorithms, called PSO, is originated from visualizing the social behavior of a group of birds or fishes. This algorithm optimizes a problem by iteratively improving the candidate solution due to the specified measure of quality. This method does not require neither defining the gradient of the target function nor an initial value for the parameters.

In PSO algorithm, each particle (the parameters to be calculated) flies in a hyperspace of the problem, searching the environment around itself to modify its location according to its experience as well as the best neighbor’s. This algorithm is computationally inexpensive in the required storage memory and its speed (Clerc & Kennedy, 2002). Moreover, as PSO optimizes the problem without implementing the information of the differentiated target function, it can be used for problems that are irregular, changing over time and noisy.
4.1. PSO algorithm

For formulating the PSO algorithm, two parameters must be defined: particle location and its speed. In the following equation the way of updating these parameters is shown.

\[ \mathbf{v}_{d+1} = \alpha \times (w \times \mathbf{v}_d + \varphi_1 \times \text{rand} \times (p\_\text{best} - \mathbf{x}_d) + \varphi_2 \times \text{rand} \times (g\_\text{best} - \mathbf{x}_d)) \]  
(16)

\[ \mathbf{x}_{d+1} = \mathbf{x}_d + \mathbf{v}_{d+1} \]  
(17)

Where, \( P\_\text{best} \) is the best position found by the particle (personal best) and \( g\_\text{best} \) is the best position found by swarm (best of personal bests) in each iteration. \( \mathbf{x}_d \) and \( \mathbf{v}_d \) are respectively the location and speed of each particle in \( d \)th iteration of the algorithm. \( \varphi_1 \) and \( \varphi_2 \) are acceleration constants, \( w \) is inertia weight and \( \text{rand} \) is a function generating a random number with uniform distribution in the interval \([0,1]\). Using the random function in this equation may be due to sudden turning of a bird flight direction. Finally, \( \alpha \) is a function of \( \varphi_1 \) and \( \varphi_2 \), implemented to improve the convergence and stability of PSO algorithm.

The inertia weight is commonly either taken as 1.4 or as a linear function of \( (d) \) decreasing from 0.9 to 0.4 and acceleration constants are usually set to 2.

5. Experimental evaluations

In order to evaluate the performance of the kernel based adaptation methods introduced in previous sections, some experiments are conducted on the FARSADAT database (Bijankhan & Sheikhzadegan, 1994) which is a Persian speech corpus. The available data is partitioned to two sections: training and test sections. The training section consists of the utterances from 250 speakers; while, the test section includes utterances of the remaining 54 speakers. The average length of the utterances is about 30 seconds. Two sessions are recorded for each speaker with the sampling rate of 22 kHz and about 30 db SNR. All training data are processed to extract 12 mel-frequency Cepstral coefficients (MFCC) and normalized frame energy as well as the first and the second order time derivations of these parameters. In total, a 39 dimensional acoustic vector from each frame of 25 ms at every 10 ms is used.

A speaker independent acoustic model is trained using training section by implementing HTK toolbox version 3.4. This model is a set of 29 phoneme models with strictly left to right 5-state hidden Markov models for vowels and 7-state HMMs for consonants. In addition, a 5-state HMM is considered to model silence. In the following experiments, only one Gaussian is considered for each state of HMMs; and the recognition performance measure used, is phoneme recognition rate.

In the kernel eigenspace-based adaptation methods, it is needed to train a set of speaker dependent models for 250 training speakers. Since there are not sufficient data from a single speaker in FARSADAT to train the related speaker dependent model, the adapted model is substituted the speaker dependent one. In this approach, a set of speaker adapted models were trained by borrowing the variances and transition matrices from the speaker independent model. So, only the Gaussian means were adapted by MLLR adaptation of HTK using the first session utterances of the training speakers. Subsequently, for each speaker, the adapted mean vectors are concatenated to form the supervector of that speaker. The training supervectors construct the speaker space.

In order to evaluate the performance of all mentioned approaches against each other and some other eigenspace-based speaker adaptation methods such as EV and EMLLR, short length (2-5 seconds) of first session utterances of test speakers is used for adaptation and those from second sessions are assigned for recognition. It must be denoted that the baseline speaker independent model yields 65.5% phoneme recognition rate.

5.1. Performing KEMLLR adaptation

For conducting KEMLLR adaptation, when gradient ascent method is implemented for optimizing the \( Q \) function, the following parameters are adjusted.
The constituent Gaussian kernel parameter $\beta_r$ is equal to 0.001 (this parameter is selected such that the similarity between any two points in the feature space remains between 0 and 1 and never approaches to 0 or 1).

The initial learning rate for performing gradient ascent is selected to be 0.001.

During computation the kernel eigenmatrix weights by using gradient ascent in GEM algorithm, it is seen that by selecting a suitable initial value for the weight vector, only one iteration of GEM is sufficient to achieve an acceptable adaptation result. Figure 1 shows the performance of EMLLR and KEMLLR techniques with respect to the number of extracted eigenmatrices, when only 2 seconds of the new speaker speech wave is implemented for adaptation.

Table 1 shows the results of performing MLLR, EMLLR and KEMLLR adaptations with 5 seconds of adaptation data and implementing 12 eigenmatrices/kernel eigenmatrices.

Table 1. The results of performing MLLR, EMLLR and KEMLLR adaptation methods within 5 seconds of adaptation data

<table>
<thead>
<tr>
<th>Phoneme recognition rate (%)</th>
<th>Model/adaptation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower than SI</td>
<td>MLLR</td>
</tr>
<tr>
<td>67.72</td>
<td>EMLLR</td>
</tr>
<tr>
<td>68.58</td>
<td>KEMLLR</td>
</tr>
</tbody>
</table>

5.2. performing eKEV adaptation

For conducting eKEV adaptation, the following parameters were adjusted.

- Constituent Gaussian kernel parameter $\beta_r$ is equal to 0.0005.
- Initial learning rate is 0.01.
- Considering one iteration of GEM.

Table 2 shows the results of applying eKEV adaptation when the number of reference speaker models is varied from 10 to 250.
Table 2. The effect of changing the number of neighbors in eKEV performance

<table>
<thead>
<tr>
<th>The number of neighbors</th>
<th>250</th>
<th>150</th>
<th>100</th>
<th>50</th>
<th>30</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phoneme Recognition Rate (%)</td>
<td>68.72</td>
<td>68.72</td>
<td>68.53</td>
<td>68.09</td>
<td>67.75</td>
<td>66.14</td>
</tr>
</tbody>
</table>

Figure 2 shows how eKEV performs in the condition that the number of neighbors as well as eigenvectors varies and 2 seconds of adaptation data is employed for adaptation.

Fig 2. The performance of eKEV with respect to the number of kernel eigenvoices as well as the number of neighbors

5.3. Evaluating the performance of PCA-based and KPCA-based methods

In order to compare the performance of PCA-based methods such as EV and EMLLR with KPCA-based ones such as eKEV and KEMLLR, 2 seconds of adaptation data is used with the best initializations for each method. The recognition results of implementing each of these adaptation methods are shown in Fig 3.

Fig 3. Comparing PCA-based adaptation methods with KPCA-based ones in 2 seconds of adaptation data
5.4. Implementing PSO

In order to apply PSO, the corresponded parameters are initialized as follows:

- The population of particles is selected to be 40 which are searching in a space with the dimension equals to the number of eigenvectors/matrices.
- The number of iterations (updating each particle position) is 100.
- The initial position of each particle in the searching space is a random vector, each component of which varies between -2 to 2.
- The initial speed vector for each particle is commonly in the range of \([-v_{\text{max}}, v_{\text{max}}]\) and \(v_{\text{max}}\) is usually chosen to be between 10% to 20% of the whole range in which each component of the position vector can vary.
- The value of \(\alpha\) parameter is fixed to 0.73.

Figure 4 shows the result of applying PSO in eKEV adaptation to one speaker. The vertical row indicates the best position found by the swarm at each iteration of the algorithm. As it can be seen, the swarm reaches to the best position at almost 30th iteration which takes far more time than it is needed for gradient ascent algorithm to find the best value for target function.

6. Discussion and conclusion

As it was shown in Figs. 1-2, the performance of both KEMLLR and eKEV improves when the number of kernel eigenvectors/matrices is increased provided that there is enough adaptation data to estimate the eigenvector/matrix weights. Whenever there is not sufficient amount of data to estimate the weights robustly, the recognition rate begins to decrease. Moreover, in the case of performing eKEV adaptation, as it is illustrated in Table 1, increasing the number of neighbor models to 150 models improves the recognition rate of the adapted model. This performance may be due to distribution of the training speaker models in the speaker space. These training speaker models in the input space are nearly close to each other causing many speaker models to have the same effect on determining the pre-image in the speaker space and only the effect of 100 models of them can be ignored.

Table 1 shows that kernel eigenspace based adaptation techniques perform better than eigenspace based ones; since, KPCA which is a nonlinear dimension reduction function, extracts more acoustic information from the speaker space than PCA.

In addition, by conducting related experiments, it is revealed that substituting numerical methods with PSO algorithm in GEM does not end to a faster optimization algorithm. This behavior dues to the PSO algorithm which is a derivative free method. This means that more function evaluations would normally be used by PSO than by a derivative based approach (Bazarra, Sherali, & Shetty, 2006). Thus, nicely behaved problems as well as our
problem, in which derivatives are available, would most likely be solved more efficiently using a derivative based method.

References