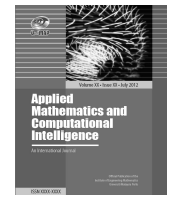
App. Math. and Comp. Intel., Vol. 1 (2012) 56–62
http://amci.unimap.edu.my


# A preliminary report on the utilization of Galerkin-PSO method for solving interpolation-like problem

**Ayu Fitri Yanti**[a,*], **Iwan Pranoto**[b]

[a] *Mathematics Department*
*Faculty of Mathematics and Natural Sciences*
*Institut Teknologi Bandung*
*40132, Bandung, Indonesia.*

[b] *Industrial and Financial Mathematics Research Group*
*Faculty of Mathematics and Natural Sciences*
*Institut Teknologi Bandung*
*40132, Bandung, Indonesia.*

Received: 29 February 2012; Revised: 19 April 2012; Accepted: 20 July 2012

**Abstract:** *The interpolation-like problem discussed in this paper is to search an optimal curve minimizing a functional cost and at the same time interpolating several given points. Instead of solving the optimization problem with constrain directly, we transform the problem into a pure optimization problem, without constrain. After that, the Galerkin Method is used to make the problem finite dimensional one. The problem becomes finding a minimal point and value of a finite dimensional function. The Particle Swarm Optimization (PSO) algorithm is used to minimize this function.*

**Keywords:** *Interpolation, Galerkin method, Particle swarm optimization.*

**PACS:** *02.70.-c, 02.60.-x*

## 1 Introduction

Interpolation is the process of defining a function if the values at specified points are given. In [1], they discuss an interpolation method to find a differentiable function that passes prescribed points and at the same time minimizes some energy integral analytically. This paper discusses an approach. The approach first transforms the original optimization problem with constrains into a pure minimization one without constrains. After that, using the Galerkin method, the minimization problem is transform into a discrete one. The method then approximates the infinite dimensional problem with a finite dimensional one. Thus, now the problem becomes a problem of minimizing some finite dimensional function. For minimiz-

*Corresponding Author: ayufitriyanti@hotmail.com (Ayu Fitri Yanti)

ing this finite dimensional function, it utilizes the Particle Swarm Optimization (PSO). The PSO is an evolutionary computation technique developed by Kennedy and Eberhart in 1995. This means that PSO have been around for just over fifteen years. However, it has been researched and utilized intensively in various areas. In this paper, we will observe the PSO performance in solving the interpolation-like problem as stated above.

## 2 Problem Formulation

### *2.1 Problem*

Suppose we are given $N+1$ points $(x_0, y_0), (x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$ with $0 = x_0 < x_1 < ... < x_N = 1$ and $y_0 = y_N = 0$. The problem here is to find a continuously differentiable function $f : [0, 1] \to R$ on $[0, 1]$ with $f(0) = f(1) = 0$ such that it minimizes the following functional

$$J(f) = \alpha \int_0^1 \left[f^{(\gamma)}(x)\right]^2 dx + (1-\alpha) \sum_{i=1}^{N-1} (f(x_i) - y_i)^2$$

where $0 < \alpha < 1, \gamma = 1$ and 2. The term $\int_0^1 \left[f^{(\gamma)}(x)\right]^2 dx$ is the energy integral, $\sum_{i=1}^{N-1}(f(x_i) - y_i)^2$ is the total error between the actual function value at $x_i$ with the expected value $y_i$. The constants $\alpha$ and $1 - \alpha$ are the weights. Here $f^{(\gamma)}(x)$ denotes the derivative of $f(x)$ of order $\gamma$. Even though the method proposed here should work with any rational value of $\gamma$, we restrict $\gamma = 1$ and 2. We consider Galerkin method [6] and Fourier series such as in [7].

The function is discretized by Galerkin method and it can be represented by the following Fourier series

$$f(x) = \sum_{j=1}^{\infty} a_j \sin j\pi x$$

for some $a_1, a_2, a_3, ...$ where $a_j$'s follow the following formula

$$a_j = 2 \int_0^1 f(x) \sin j\pi x, \ j = 1, 2, 3, ...$$

and satisfy the conditions $\sum_{j=1}^{\infty} j^{2\gamma} a_j^2 < \infty$.

Then, we approximate the function with the Fourier series $\hat{f}$, where

$$\hat{f}(x) = \sum_{j=1}^{\infty} a_j \sin j\pi x,$$

for some $M$. Now the problem becomes to find $(a_1, a_2, ..., a_M)$ which minimizes

$$\begin{aligned} J\left(\hat{f}(x)\right) &= \hat{J}(a_1, a_2, a_3, ..., a_M) \\ &= \alpha \int_0^1 \left[\hat{f}^{(\gamma)}(x)\right]^2 dx + (1-\alpha) \sum_{i=1}^{N-1} \left[\hat{f}(x_i) - y_i\right]^2. \end{aligned}$$

Since

$$\hat{f}^{(1)}(x) = \sum_{j-1}^{M} a_j j\pi \cos j\pi x,$$

then

$$\hat{J}(a_1, a_2, a_3, ..., a_M) = \alpha \int_0^1 \left[ \sum_{j-1}^{M} a_j j\pi \cos j\pi x \right]^2 dx + (1-\alpha) \sum_{i=1}^{N-1} \left[ \left( \sum_{j-1}^{M} a_j \sin j\pi x_i \right) - y_i \right]^2.$$

When the second derivative is considered, that is $\gamma = 2$, we have

$$\hat{f}^{(2)}(x) = -\sum_{j-1}^{M} a_j (j\pi)^2 \sin j\pi x.$$

Thus the function to be minimized will be

$$\hat{J}(a_1, a_2, a_3, ..., a_M) = \alpha \int_0^1 \left[ \sum_{j-1}^{M} a_j (j\pi)^2 \sin j\pi x \right]^2 dx + (1-\alpha) \sum_{i=1}^{N-1} \left[ \left( \sum_{j-1}^{M} a_j \sin j\pi x_i \right) - y_i \right]^2.$$

We will use the PSO algorithm to minimize the function $\hat{J} : R^M \to R$.

### *2.2 Particle Swarm Optimization*

Particle Swarm Optimization or PSO is introduced by Kennedy and Eberhart (1995)[2]. PSO is a population-based stochastic optimization technique inspired by the social behaviour of bird flocking or fish schooling. Each particle, or more precisely its coordinate, represents a candidate solution. Elements of the coordinate of a particle represent values to be searched. First these particles are "flown" into the search space. The original process for implementing the global version of PSO is as follows: let the $i^{th}$ particle position in a $M$-dimensional space be represented as $A_i = (a_{i1}, a_{i2}, ..., a_{iM})$. The best previous position (the position giving the best fitness value) of the $i^{th}$ particle is recorded and represented as $P_i = (p_{i1}, p_{i2}, ..., p_{iM})$. The index of the best particle among all the particles in the population is represented by the symbol $g$. The rate of the position change (velocity) for particle $i^{th}$ is represented by $V_i = (v_{i1}, v_{i2}, ..., v_{iM})$. The particles move on the search space following the following rules:

$$v_{im} = v_{im} + c_1 r_1 (p_{im} - a_{im}) + c_2 r_2 (p_{gm} - a_{im}) \tag{1}$$

$$v_{im} = V_{max}, \text{if } v_{im} > V_{max}$$

$$v_{im} = -V_{max}, \text{if } v_{im} < V_{max}$$

$$a_{im} = a_{im} + v_{im} \tag{2}$$

Here $v_{im}$ is momentum which represents inertia component and memory of previous flight direction. This term prevents the particle from drastically changing direction. The constants $c_1$ and $c_2$ are two positive constants. The symbols $r_1$ and $r_2$ are two random functions in the range $[0, 1]$. The term $c_1 r_1 (p_{im} - a_{im})$ is cognitive component which represents the private thinking of the particle itself, and $c_2 r_2 (p_{gm} - a_{im})$ is the social component which represents the collaboration among the particles. Finally, the symbol $V_{max}$ is a parameter controlling the global exploration of particles.

In equation 1, the right hand side consists of three parts: the first part is the previous velocity of the particle; the second and third parts are the ones contributing to the change of the velocity of a particle. It is used to calculate the particle's new velocity according to its previous velocity and the distances of its current position from its own best experience

(position) and the group's best experience. Then the particle moves toward a new position according to equation 2. The performance of each particle is measured according to a predefined fitness function which is related to the problem to be solved.

And then, as in [5], the term $V_{max}$ is modified to increase its convergence rate. Inertia weight concept is introduced to control the exploration and exploitation controlling the momentum $v_{im}$ [4]. Therefore, the original PSO turns into:

$$v_{im} = wv_{im} + c_1r_1(p_{im} - a_{im}) + c_2r_2(p_{gm} - a_{im})$$

According to [4], the recommended value of w should be on the range $[0.9, 1.2]$. In [3], it introduces the "lbest" concept. And then, constriction factor is introduced in [8], so that equation 1 becomes

$$v_{im} = \kappa\left[v_{im} + c_1r_1(p_{im} - x_{im}) + c_2r_2(p_{gm} - x_{im})\right]$$
$$\kappa = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}, \text{where } \varphi = c_1r_1 + c_2r_2, \varphi \geq 4$$

The optimal settings suggested in [8] correspond to $\varphi = 4.1, \kappa = 0.7298, c_1 = c_2 = 2.05$. So that, $\kappa c_1 = \kappa c_2 = 1.49618$.

### *2.3 Particle Swarm Optimization Algorithm for This Problem*

Suppose there are $K$ agents $z_1, z_2, z_3, ..., z_K$. Mathematically, each agent is a time-varying vector. The iteration process determines the vector value of the functions based on the previous values. The process initially starts at a randomly selected location in $R^M$. Suppose the coordinate of the $k^{th}$ agent is $z_k = [a_1^k, a_2^k, a_3^k, ..., a_M^k], k = 1, 2, ..., K$. Then, the position of all agents at a certain iteration can be written in the form of a matrix A.

$$\begin{bmatrix} a_1^1 & a_1^2 & ... & a_1^K \\ a_2^1 & a_2^2 & ... & a_2^K \\ : & : & : & : \\ a_M^1 & a_M^2 & ... & a_M^K \end{bmatrix}$$

Thus, the coordinate of the $k^{th}$ agent is the $k^{th}$ column vector of the matrix A.

a. Initialization
   Set the constants involved:

   1. The weights $\alpha$ and the dimension of the search space $M$
   2. The number of points $N$
   3. The number of agents $K$
   4. The cognitive parameter $c_1$
   5. The social parameter $c_2$
   6. The random number for individual influence $r_1,\ 0 \leq r_1 \leq 1$
   7. The random number for social influence $r_2,\ 0 \leq r_2 \leq 1$
   8. The maximum iteration *maxiter*
   9. The maximum velocity $V_{max}$
   10. The maximum position to prevent the particle out of the dynamic range $A_{max}$

11. The initial position of each particle is set randomly to create a matrix of size $M \times K$

$$A^0 = \begin{bmatrix} a_{11}^0 & a_{12}^0 & ... & a_{1K}^0 \\ a_{21}^0 & a_{22}^0 & ... & a_{2K}^0 \\ : & : & : & : \\ a_{M1}^0 & a_{M2}^0 & ... & a_{MK}^0 \end{bmatrix} = \begin{bmatrix} A_1^0 & A_2^0 & ... & A_K^0 \end{bmatrix}$$

12. The initial velocity of each particle is set randomly to create a matrix of size $M \times K$

$$V^0 = \begin{bmatrix} v_{11}^0 & v_{12}^0 & ... & v_{1K}^0 \\ v_{21}^0 & v_{22}^0 & ... & v_{2K}^0 \\ : & : & : & : \\ v_{M1}^0 & v_{M2}^0 & ... & v_{MK}^0 \end{bmatrix} = \begin{bmatrix} V_1^0 & V_2^0 & ... & V_K^0 \end{bmatrix}$$

13. Set $iter$=1 and maximum error
14. Set initial best previous position$P_k$ and initial best position of population $P_g$

$$P^0 = A^0 = \begin{bmatrix} A1^0 & A_2^0 & ... & A_K^0 \end{bmatrix} = \begin{bmatrix} P_1^0 & P_2^0 & ... & P_K^0 \end{bmatrix}$$

After calculating and comparing the value of $\hat{J}\left(P_k^{iter}\right), k = 1, 2, ..., K$, let $\hat{J}\left(P_g^0\right)$ be the smallest value of $\hat{J}\left(P_k^0\right)$, then $P_g = P_k^0$.

b. Optimization

- Calculate $\hat{J}\left(A_k^{iter}\right)$ value
- If $\hat{J}\left(A_k^{iter}\right) \leq \hat{J}\left(P_k^{iter}\right)$, then $\hat{J}\left(P_k^{iter}\right) = \hat{J}\left(A_k^{iter}\right), P_k^{iter} = A_k^{iter}$
- If $\hat{J}\left(A_k^{iter}\right) \leq \hat{J}\left(P_g^{iter}\right)$, then $\hat{J}\left(P_g^{iter}\right) = \hat{J}\left(A_k^{iter}\right), P_g^{iter} = A_k^{iter}$
- If stopping criteria is met then program is stopped
- Update particle's velocity following this equation

$$V_k^{iter+1} = \kappa[V_k^{iter} + c_1 r_1(P_k^{iter} - A_k^{iter}) + c_2 r_2(P_g^{iter} - A_k^{iter})],$$

$$-V_{max} \leq V_k^{iter} \leq V_{max}$$

- Update particle's position

$$A_k^{iter+1} = A_k^{iter} + V_k^{iter+1},$$

$$-A_{max} \leq A_k^{iter} \leq A_{max}$$

- Update iteration

$$iter = iter + 1$$

- Back to the initial step

c. Termination of the iteration
Termination criterion which used on this program is limiting the iteration and error between objective value at $k^{th}$ and $(k-1)^{th}$.

We will get one particle $P_g$ which makes the objective function has a minimum value. So, the function we are looking at the beginning is $f\left(P_g\right)$.

## 3 Results And Discussion

The following illustrations use dimension of search space ($M$) is set as 1, 2, 3, 10 and 20. The population size ($K$) is 10 and 20 particles. A constriction factor ($\kappa$) is set as 0.7298, $\kappa c_1 = \kappa c_2 = 1.49618$. $V_{max}$ and $A_{max}$ are set to be equal, 100 and 100 respectively. The weight ($\alpha$) is set as 0.9. The stopping criterion is used by measuring the error between objective value at $k^{th}$ and $(k-1)^{th}$ iterations set equal 0.001. The maximum of iterations ($maxiter$) is set equal 1, 10, and 100. A total of 15 runs for each combination are conducted. Tables 1 and 2 show some results for the first case ($\gamma = 1, M = 2$) and the second case ($\gamma = 2, M = 2$), respectively, both use $f(0) = 0, f(0.5) = 1, f(1) = 0$.

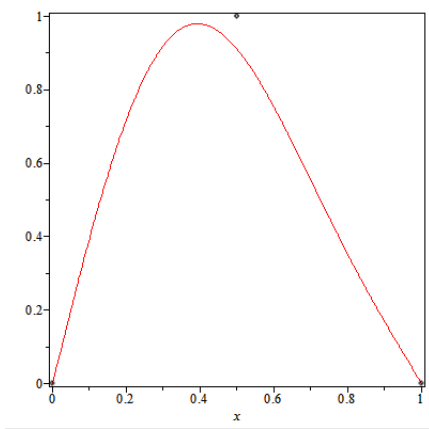



Figure 1: $\hat{f}(x) = 0.9102\sin(\pi x) + 0.1938\sin(2\pi x)$

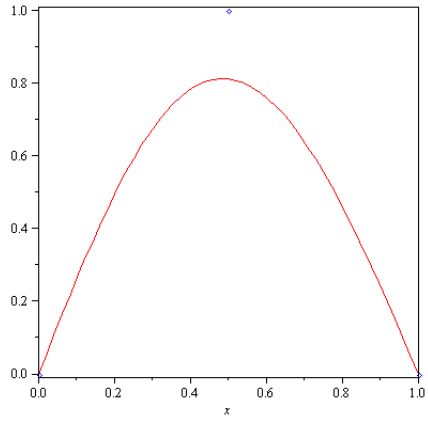



Figure 2: $\hat{f}(x) = 0.8112\sin(\pi x) + 0.0192\sin(2\pi x)$

## 4 Conclusion And Future Works

In this paper, we propose a method combining Galerkin method and Particle Swarm Optimization algorithm for solving an interpolation-like problem. It should be interesting if

some researchers apply the proposed method on different value of $\gamma$, various values of $\alpha$ and various combination of parameters.

**Acknowledgments**

The authors would like to thank the Community of Mathematics Department, Institut Teknologi Bandung, and the Faculty of Mathematics and Natural Sciences, Institut Teknologi Bandung.

**References**

[1] H. Gunawan, F. Pranolo, and E. Rusyaman. An interpolation method that minimizes on energy integral of fractional order. *In Proc. of ASCM*, pages 151–162, 2008.

[2] J. Kennedy and R. C. Eberhart. Particle swarm optimization. *In Proc. of IEEE International Conference on Neural Networks*, pages 1942–1948, 1996.

[3] R. C. Eberhart and J. Kennedy. New optimizer using particle swarm theory. *In Proc. of the Sixth International Symposium on Micromachine and Human Science*, pages 39–43, 1995.

[4] Y. Shi and R. Eberhart. A modified particle swarm optimizer. *In Proc. of IEEE World Congr. Computational Intelligence Evolutionary Computation*, pages 69–73, 1998.

[5] H. Fan. A modification to particle swarm optimization algorithm. *Engineering Computations*, 19:970–989, 2002.

[6] Y. Zhan and N. Ma. Galerkin method, *Computational Engineering*, Bochvm-RvhrVeniversitat.

[7] G. B. Folland, *Fourier analysis and its applications*, Wadsworth & Brooks/Cole, Pacific Grove,1992.

[8] M. Clerc and J. Kennedy. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transaction on Evolutionary Computation*, 6: 58–73, 2002.