2009 Third UKSim European Symposium on Computer Modeling and Simulation

Particle Swarm Modelling of a Flexible Beam Structure

M. Mohamad, *Member, IEEE,* M. O. Tokhi, *Senior Member, IEEE*, S.F Toha, *Member, IEEE,* I. Abd. Latiff
Department of Automatic Control and Systems Engineering
The University of Sheffield
Mappin Street, Sheffield, S1 3JD, United Kingdom
m.mohamad@sheffield.ac.uk

Abstract— **This paper presents a particle swarm optimization (PSO) algorithm with dynamic spread factor inertia weight and its application to dynamic modeling of a flexible beam structure. In this study, system identification scheme based on PSO is formulated to obtain a dynamic model of the beam in parametric form. A PSO algorithm with dynamic spread factor inertia weight is proposed and its performance is assessed in comparison to a standard PSO in modelling the flexible beam structure.**

Keywords- Particle swarm optimization, dynamic spread factor, flexible beam structure, dynamic modeling

I. INTRODUCTION

Particle swarm optimisation (PSO) was developed by Keneddy and Eberhart in 1995 as a new method in evolutionary computation. Keneddy, a social psychologist and Eberhart, an electrical engineer were trying to develop an optimiser based on social behaviour model and they were influenced by Heppner and Grenander's work on simulating the behaviour of bird flocking in finding cornfield. The fundamentals of PSO come from theory of social sharing presented by researchers in fish schooling. The theory states*: information sharing among each member of the group as, discoveries and previous experiences, will give benefit more than competition among them* [1]. Based on the theory, PSO adjusts the particle's position based on its own experience and other member's discovery in each generation in finding the optimum solution to the problem. This means, the particles communicate with each other in every search generation toward the optimum point.

As a method categorized under evolutionary computation (EC), PSO shares many similarities with other EC methods such as using population of random solution (stochastic method), updating generation for optima searches, applying fitness concept based on an objective function, and adjustment done in position update is conceptually similar to mutation applied in evolutionary programming. The only difference is that PSO does not apply the survival of fittest concept as others. This means that there is no competition among particles, no elimination of the weakest particle, indeed all particles share their information/discoveries and will eventually converge to the same optimum point [2].

II. PARTICLE SWARM OPTIMISATION

PSO starts with initialising a random population of possible solutions called particles. A particle is a moving point in the problem search space composed of three dimensional vectors that store its current position, $\vec{x}_i$ current velocity, $\vec{v}_i$ and the best position, $\vec{p}_i$ achieved so far. Each particle then evaluates the fitness of their current position against an objective function, and the fitness value will provide the best position achieved so far at the point of the search. If the current fitness value is better than the best fitness value in memory, then the current position will be set as the best position. All the particles then will move to a new position by updating their velocity using (1);

$$\begin{aligned} v_{i(t)} &= wv_{i(t-1)} + c_1 \times r1 \times (p_i - x_i) \\ &\quad + c_2 \times r2 \times (p_g - x_i) \end{aligned} \tag{1}$$

where $v_{i(t)}$ is the updated velocity of particle i, $v_{i(t-1)}$ is the current velocity of particle i, c_1 and c_2 are constants, r1 and r2 are uniformly distributed random numbers between 0 and 1, p_i is personal best (pbest); the best position of particle i so far, and pg is global best (gbest); the best position of the particle in the entire population so far and xi is the current position of particle i. The first modification to the original algorithm was done by Shi and Eberhart in 1998, to give the ability of exploration and exploitation to the particles by introduced inertiaweight,w into the original equation. Since then, equation (1) always known as original pso equation.

Once $\vec{v}_i$ has been calculated, particles will update their position according to

$$x_{i(t)} = x_{i(t-1)} + v_{i(t)} \tag{2}$$

Equation (1) describes the flying trajectory of particles. There are 3 parts in (1) namely momentum part, cognitive part and social part. Fig. 1 shows the effect of each part on direction of the new velocity. It can be seen that the particles are always been pulled toward pbest and gbest at every generation.

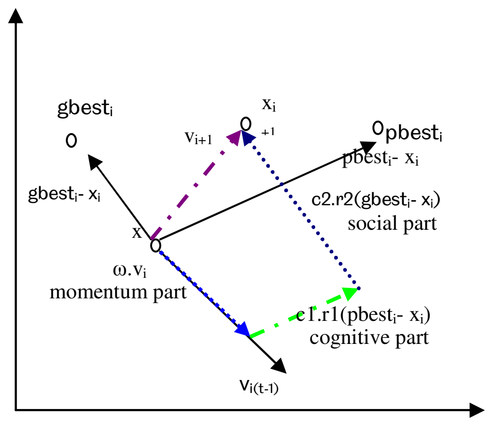

Figure 1. Velocity update of particles in PSO

$v_{i(t-1)}$ is a momentum part where it provides the previous velocity as a momentum to particles. Without this part, the previous velocity is memory less and the new velocity will only depend on the current position and global best position (the cognitive part is unlikely in this case to exist). The global best particle will not move until a new global best is found and all particles tend to move toward the global best. Therefore, the solution will solely depend on the initial population, where the global best position of initial population will be the solution unless there is another acceptable solution presents on moving trajectory of particles toward global best. Other than that, the search space is also getting smaller in every generation and it can be concluded that without the momentum part, equation (1) will likely behave as a local search [3] . With the presence of the momentum part, equation (1) will favour the global search where the particle will have an ability to explore new areas of the search space. The momentum part will pull the particles toward previous moving trajectory, prevent the particles from only moving toward the personal best position and global best position and eventually update the velocity to a new direction. The inertia weight provides a balance between global search and local search. Higher value of *w* results an overshoot or overflies particles, providing an exploration behaviour (global search), while lower value of *w* will shrink the search space providing exploitation ability (local search). The mathematical equation representing this concept is given as

$$\omega = (\omega_1 - \omega_2) \times \frac{(MAXITER - iter)}{MAXITER} + \omega_2 \qquad (4)$$

Linearly decreasing inertia weight from 0.9 to 0.4 over time has shown significant improvement in performance compared to constant inertia weight [3].

The cognitive part in the velocity update equation represents personal discoveries of the particles. It pulls the particles toward the personal best position achieved and contributes to magnitude and direction of the new velocity. It works like a memory, where one always tends to return to a place with history of success. The random number will make the particles to wander stochastically around the personal best position and the cognition parameter controls the impact of the personal best position on the new velocity. Like in the second part, the third part also pulls particles toward some best position stochastically, but this time toward the best position achieved by the entire population. This social part symbolizes the communication between each particle where they share their personal discovery and experience, and contribute to global best position. The social parameter, c2 same as c1, provides an impact of global best on the new velocity direction and magnitude.Kennedy and Eberhart proposed in the original PSO, that the cognitive and social scaling parameters c1 and c2 are selected such that c1=c2=2, in order to allow a mean of 1 (when multiplied by the random numbers r1 and r2). [4] have proposed a time varying acceleration coefficient (TVAC) along with time varying inertia weight (TVIW) in order to avoid premature convergence and to ensure convergence. TVAC is set with large value of cognitive parameter, c1 and small value for social parameter, c2 at the beginning in order to let the particles explore around search space. As time varying, value c1 will decrease and c2 will increase so that particles eventually forget their own interest/history and keep attracted towards global best position so that they will converge at global optimum. The modification can be mathematically represented as follows:

$$c_1 = (c_{1f} - c_{1i}) \frac{iter}{MAXITER} + c_{1i} \qquad (5)$$

$$c_2 = (c_{2f} - c_{2i}) \frac{iter}{MAXITER} + c_{2i} \qquad (6)$$

With this new modification, the PSO algorithm has become more effective in avoiding premature convergence for static problems.

The PSO algorithm thus described can be formulated as below.

PSO Algorithm

1. *Initialize particles randomly in the search space.*
2. *Assign random initial velocities for each particle.*
3. *Loop*
4. *Evaluate the fitness of each particle according to a user defined objective function.*
5. *Compare particle's fitness evaluation with current pbest. Update pbest if current value is better than $pbest_{i.}$*
6. *Identify gbest*
7. *Calculate the new velocities for each particle using equation 1, and limit it by Vmax.*
8. *Move the particles, equation (2).*
9. *If stopping criteria met, exit loop*
10. *End loop*

III. Dynamic Spread Factor

Dynamic spread factor was first introduced in 2009 and found highly effective in improving major issues in basic PSO like lack of diversity and premature convergence [5].As discussed in section 2, suitable selection of the inertia weight provides a balance between global and local exploration and exploitation. In time varying inertia weight, the value of inertia weight will decrease linearly as the iteration increases, but this is slightly different in the PSO with spread factor (SFPSO), where it continuously modifies the inertia weight value based on the instantaneous spreading of the particles from the global best, not solely based on the iteration time. The instantaneous spreading of the particles or spread factor depends on two factors, the spread of the particles and the distance of the average particle with respect to global best position. This used to calculate the inertia weight

$$\begin{aligned} spread &= max(position) - min(position) \\ deviation &= sum\,(abs(position) - gbest)/particle \end{aligned} \quad (7)$$

A momentum part is introduced to provide momentum to the particles when the initial inertia weight has reached zero. In this way local optima can be avoided. The spread factor and inertia weight thus are obtained as;

$$SF = 0.5(spread + deviation)/(x\max - x\min) \quad (8)$$

$$w = \exp(-iter/(SF \times \max\_iteration)) \quad (9)$$

$$w = \exp(-iter/(SF \times \max\_iteration)) + mom \quad (10)$$

Where equation (8) and (9) comprise the basic SFPSO and equations (8) and (10) comprise the SFPSO with momentum factor.

IV. Parametric modelling of flexible beam structure

The PSO algorithms were employed to gain a parametric model for flexible beam structure using the ARMA model structure

$$\begin{aligned} \hat{y}(k) &= -[a_1, a_2, a_3, a_4][y(k-1),..., y(k-4)]^T \\ &+ [b_1, b_2, b_3, b_4][u(k-1),..., u(k-4)] \end{aligned} \quad (11)$$

where a_i and b_i are the parameters to be identified. 20001 input/output data values were obtained from a fixed-free flexible beam structure simulated within Matlab/Simulink using finite difference technique. The PSO algorihm was used to obtain the parameter a_i and bi that give the smallest mean square error (MSE) between the measured system output and the ARMA model predicted output.

The particle swarm optimisation begins with initialize a population of swarm containing 20 particles with 8 variables. The particles was distributed randomly in search space within the range [-2,+2]. The first 4 rows in all particles were assigned to b_1,..,b_4 and the next 4 rows to a_1,…,a_4. The predicted output based on equation (8) is calculated using actual input and output data and parameters from particles. The objective function of the optimisation process is the mean square error between actual output and predicted output. This is given as:

$$f(x) = \frac{\sum_{k=1}^{n} \left| y(k) - \hat{y}(k) \right|^2}{n} \quad (12)$$

Where n=20001. The process continues for each particle and follows the steps in PSO algorithm.

Beside the objective function, [6] has proposed stability constraint to be satisfied in PSO algorithm so that the algorithm will give a solution of a stable model. By adding this constraint along objective function, it help particles to avoid a solution that lead to unstable model rather than get a set of parameters with best minimum error, but produce unstable model after long iteration. A model is a stable model when all the poles of its discrete transfer function are within the unity circle. If any pole is outside unity circle, then it is unstable model. In order to avoid unstable solution leading the optimisation process, a penalty value was added into the objective value (fitness value) of a particle, so it will favour a stable solution to be selected as a leading particle in search space. The pse-do code for this process is shown below:

Step 1: Assign elements of a particles as b_1,….,b_4 and a_1,….,a_4

Step 2: Form transfer function, H(z) taking all element such as

$$H(z) = \frac{b_1 z^3 + b_2 z^2 + b_3 z + b_4}{z^4 + a_1 z^3 + a_2 z^2 + a_3 z + a_4}$$

Step 3: Calculate poles of the transfer function (roots of the denominator)

Step 4: Modify objective value,

```
    If [poles] > 1, then f(x) = f(x) + penalty value
        Else      f(x) = f(x)
```

V. Results and discussion

The performance of SFPSO in modelling the flexible beam was studied in comparison to the PSO with linear variant inertia weight (LVIW) with 3 different acceleration coefficient settings. Each algorithm was run 10 times and results of run with the smallest MSE were recorded. Table 1 shows the acceleration coefficient parameter for different setting.

TABLE I. Acceleration Coefficient setting

Setting	C1	C2
Original	2	2
TVC1	(2.5 – 0)	2
TVAC	(2.5 – 0)	(0 – 2.5)

TABLE II. RESULTS

Parameter Selection	Min MSE	Iteration	Dead Time
PSO TVAC	6.86E-11	585	400
Original PSO	9.30E-11	607	
SFPSO MOM 0.1 TVC1	1.03E-10	253	112
SFPSO MOM 0.4	1.16E-10	706	299
SFPSO MOM 0.2	1.17E-10	396	197
SFPSO TVC1	1.29E-10	218	
SFPSO MOM 0.3 TVC1	1.34E-10	492	244
PSO TVC1	1.34E-10	436	
SFPSO MOM 0.1	1.36E-10	243	140
SFPSO MOM 0.3	1.47E-10	537	211
SFPSO TVAC	1.62E-10	443	
SFPSO MOM 0.2 TVC1	1.76E-10	313	244
SFPSO MOM 0.4 TVAC	1.77E-10	899	810
SFPSO MOM 0.2 TVAC	2.27E-10	564	221
Original SFPSO	2.30E-10	145	
SFPSO MOM 0.4 TVC1	2.79E-10	601	373
SFPSO MOM 0.1 TVAC	2.94E-10	646	146
SFPSO MOM 0.3 TVAC	4.01E-10	553	510

Table 2 shows the results of the PSO algorithms with different parameter settings. The two main results which will be evaluated for the performance analysis are the MSE and the numbers of iterations convergence achieved by each algorithm.

It is noted that the smallest MSE is achieved by PSO TVAC with 6.86×10^{-11}, the original PSO achieved 9.3×10^{-11} and the largest MSE was 4.01×10^{-10} as achieved by SFPSO TVAC with a momentum factor of 0.3 while other algorithms achieved solutions with MSEs between 1.03×10^{-10} and 2.94×10^{-10}. The analysis found that the differences between these MSE values are not very significant and it can be deduced that all the algorithms give acceptable solution in parametric modelling of the flexible beam system.

Since all the MSE values were within acceptable region, it is further shown that the SFIW gave a much faster solution (low iteration number) when compared to LVIW. The fastest algorithm convergence was achieved with original SFPSO, which gave a solution only after 145 iterations, while original PSO converged after 607 iterations. For TVC1 setting; SFPSO converged at 218 iterations whereas to PSO TVC1converged at 436 iterations, and the trend was the same with TVAC setting; SFPSO converged at 443 iterations and PSO at 585 iterations.

The effect of the spread factor on the inertia weight can be seen in Fig 2. Wide spreading particles from the best fitness position will result in higher inertia weight (global search) and when the particles start moving toward the optimum solution, the inertia weight value will decrease to allow a local search. It is interesting to see that the inertia value can also increase and decrease again, because it is not solely depended on the iteration but on the two spread factors. Once the particles are within the locality of the best position, the inertia weight will fall down to zero value, forcing the algorithm to converge at less iterations compared to linearly varying inertia weight method (LVIW).

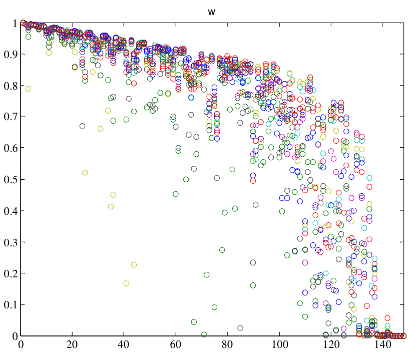

Figure 2. Dynamic spread factor inertia weight

By adding a momentum part into inertia weight calculation, the iteration taken is increasing with the momentum. The higher the momentum, the longer the algorithm will take to converge. This effect can be seen from fig 3 for the original and TVC1 acceleration coefficient setting.

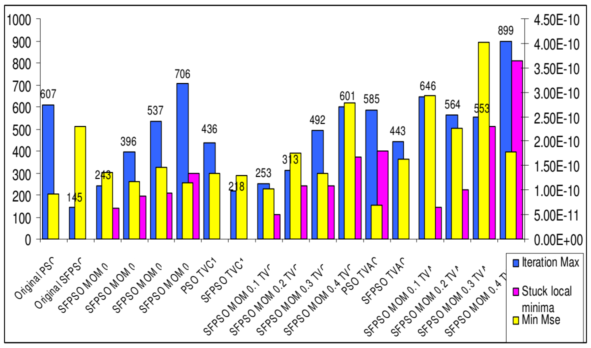

Figure 3. Comparison chart

Momentum part also creates a dead time phenomena were the particles get stuck at local optima on early iterations for long times before the algorithms eventually converge to an optimum solution. Fig 4 shows the dead time problem for TVAC setting where the SFIW has not worked very well with this setting, as all the algorithms took longer to stop (above 500 iterations). This problem occurs due to the longer global search caused by the TVAC setting where the particles take longer time to forget their own interest and it is shown by dead time problem that result in higher iterations to converge. It is thus shown that SFIW will provide a good solution with original or TVC1 acceleration coefficient setting while SFIW will take longer time with TVAC setting.

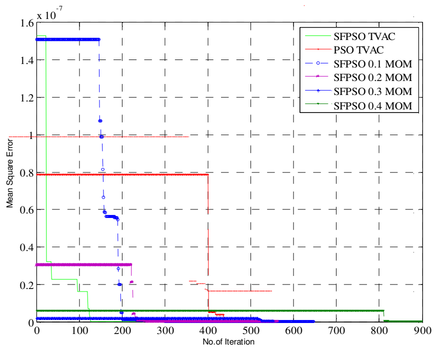

Figure 4. Convergence profile for SFPSO with TVAC setting

The eight variables for the ARMA model were obtained and the transfer function was formed,

$$\frac{0.5701\ s^4 - 1.496\ s^3 + 1.798\ s^2 - 1.177\ s + 0.3044}{s^5 - 0.6887\ s^4 - 0.2939\ s^3 - 0.2048\ s^2 + 0.3855\ s - 0.1977} \quad (13)$$

Figure 5 shows that the predicted output of PSO model follows the plant output very well in time domain. . The frequency domain plot (Figure 6) of the predicted PSO model and plant outputs indicates that the model has successfully captured the system dynamics of the first 5 dominant mode.The pole-zero diagram (Figure 7) shows that all the poles lie inside the unit circle whereas some zeros are outside. This indicates that the model is stable and non-minimum phase model.The model reached an MSE level of 0.00005337 (Figure 8). Correlation validation of model is shown in Figure 9,10,11,12 and 13. It is noted that all the five correlation functions are within the 95% confidence bands indicating that the model behaviour is unbiased and close to that of the real system.

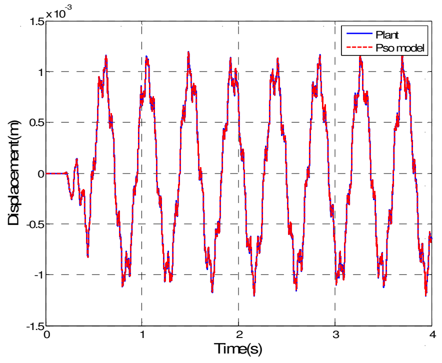

Figure 5. Actual and predicted output

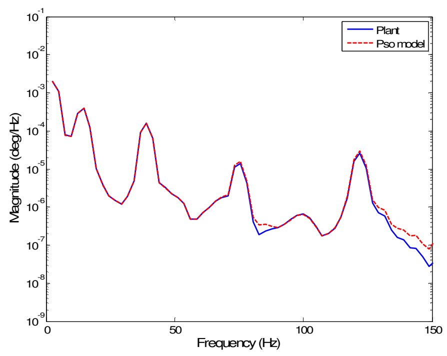

Figure 6. Power spectral density of actual and predicted PSO

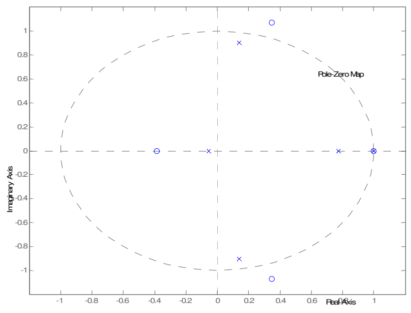

Figure 7. Pole-zero diagram

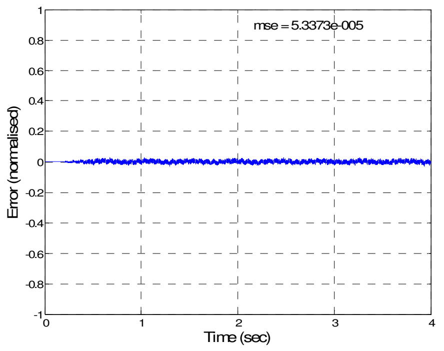

Figure 8. Mean square error between plant and predicted model

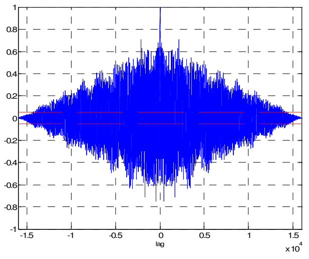

Figure 9. Auto-correlation validation test

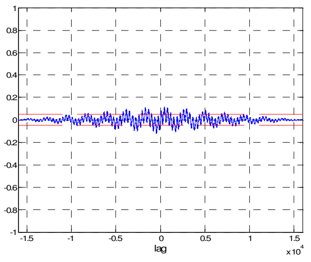

Figure 10. Cross-correlation of input-residuals

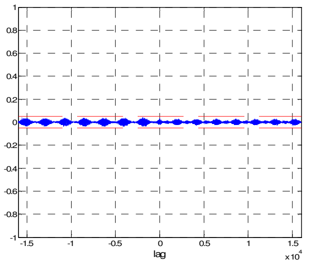

Figure 11. Cross-correlation of input square-residuals

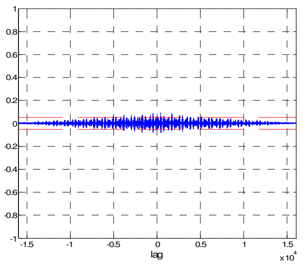

Figure 12. Cross-correlation of input square-residual square

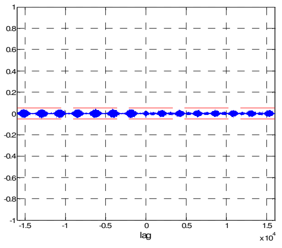

Figure 13. Cross-correlation of residual and (input*residual)

VI. Conclusion

The PSO with SFIW and momentum factor has been introduced and its performance with 3 different acceleration coefficients setting namely, original, TVC1 and TVAC setting in parametric modelling of a flexible beam system.It has been founded that the dynamic spread factor PSO method achieved optimum solution with faster convergence.

Acknowledgment

M.Mohamad acknowledges the support of a financial research fellowship of the Universiti Teknologi Malaysia (UTM) and Ministry of Higher Education Malaysia.

References

[1] Kennedy, J. and R. Eberhart. *Particle swarm optimization*. in *Neural Networks, 1995. Proceedings., IEEE International Conference on*. 1995.

[2] Shi, Y. and R.C. Eberhart. *Empirical study of particle swarm optimization*. in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*. 1999.

[3] Yuhui, S. and C.E. Russell, *Parameter Selection in Particle Swarm Optimization*, in *Proceedings of the 7th International Conference on Evolutionary Programming VII*. 1998, Springer-Verlag.

[4] Ratnaweera, A., S.K. Halgamuge, and H.C. Watson, *Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients.* Evolutionary Computation, IEEE Transactions on, 2004. **8**(3): p. 240-255.

[5] I. Abd Latiff, and M. O. Tokhi, “Fast convergence strategy for particle swarm optimization using spreading factor,” Proceedings of the IEEE Congress on Evolutionary Computation, Trondheim, Norway, 18-21 May 2009

[6] M. S. Alam, and M. O. Tokhi, “Modelling of a twin rotor system: a particle swarm optimisation approach,” Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering, vol. 221, 2007, pp. 353-375.