Journal of Theoretical and Applied Information Technology
15th December 2012. Vol. 46 No.1
© 2005 - 2012 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

CHAOTIC PARTICLE SWARM OPTIMIZED KRIGING MODEL FOR CURVE FITTING

LI ZHANG

School of Civil Engineering, Hubei Polytechnic Univeristy, Huangshi 435003, China

ABSTRACT

Chaotic particle swarm optimization (CPSO) algorithm is proposed to optimize the Kriging model, which can improve the precision of curve fitting. A typical example is selected to demonstrate the advantage of the optimized Kriging model, compared with other curve fitting tools.

Keywords: *Kriging, Chaotic Particle Swarm Optimization, Curve Fitting*

1. INTRODUCTION

Curve fitting plays an important role in the numerical simulation, which is gaining attention by many researchers from different fields. Polynomials and artificial neural network (ANN) are used to simulate the curve by some researchers[1-11]. However the errors produced by using these methods to simulate the curve are hardly acceptable when the number of input sample points is small. It is necessary to find another model to simulate the curve, which only needs a small number of input sample points.

Kriging as an alternative model has been used in engineering design applications since it was first introduced by Sacks et al.(1989)[11] in a statistical literature. Lophaven published the open source code for the Kriging MATLAB toolbox. However the work done by Lophaven cannot get the optimal of the kriging parameter, which will affect the precision of kriging prediction in the curve fitting. In this paper, a chaos embedded particle swarm optimization algorithm is used here to search the global optimal of parameter θ for kriging model. And then, a typical example is selected to demonstrate the advantage of the optimized Kriging model, compared with other curve fitting tools.

2. KRIGING MODELING

2.1 Basic Model

Kriging treats the deterministic response of a system as a realization of a random function (stochastic process), y, that consists of a regression model[11] and a stochastic error.

$$y=\sum_{j=1}^{p}\beta_j f_j(x)+z(x)=f(x)^T\beta+z(x), \qquad (1)$$

with $\beta=[\beta_1,\beta_2,\dots,\beta_p]^T$ and $f(x)=[f_1(x),f_2(x),\dots,f_p(x)]^T$. β is a column vector of regression parameters and $f(x)$ is a column vector of basis functions. Here quadratic polynomials are used as the basis functions.And the error $z(x)$ is assumed to be a second-order stationary stochastic process with the following

statistical characteristics：

$$E[z(x)]=0,\ E[z(x+d)-z(x)]=0,\ E[z(\omega)z(x)]=\sigma^2R(\theta,\omega,x) \qquad (2)$$

where σ^2 is the variance of $z(x)$ and $R(\theta,\omega,x)$ is a correlation function with parameter θ to be determined.

We define S as a set of m design sites with $S=[s_1,\dots,s_m]^T$, $s_i\in\Re^n\left(i=1,\dots,m\right)$. Correspondingly, we have the expanded $m\times p$ design matrix F：

$$F=[f(s_1),\dots,f(s_m)]^T,\ F_{ij}=f_j(s_i) \qquad (3)$$

where $f(s_i)=[f_1(s_i),f_2(s_i),\dots,f_p(s_i)]^T$ and $f_j\left(j=1,2,\dots,p\right)$ are the same as the above.

Further, define R as the matrix R of stochastic-process correlations between $z's$ at design sites, with $R_{ij}=R(\theta,s_i,s_j),\ i,j=1,\dots,m$.

At an untried point x, let $r(x)=[R(\theta,s_1,x),\dots,R(\theta,s_m,x)]^T$ be the vector of correlations between $z's$ at design sites and x. The correlation model we use here is Gauss correlation function:

$$R(\theta,w,x)=\prod_{j=1}^{m}R_j(\theta,w_j-x_j)=\exp(-\theta_j d_j^2),\quad d_j=w_j-x_j \qquad (4)$$

2.2 Prediction Model

Kriging predicts the response of an untried point based on linear combination of $y_i \ (i=1,2,...,m)$,

$$\hat{y}(x)=c^T Y, \tag{5}$$

with $c=c(x)\in R^m$ and $Y=[y_1,y_2,...,y_m]^T$. The prediction error is

$$\hat{y}(x)-y(x)=c^T Y-y(x)=c^T Z-z+(F^T c-f(x))^T\beta \tag{6}$$

where $Z=[z_1,z_2,...,z_m]^T$ are the errors at the design sites. To keep the predictor unbiased, we demand

$$F^T c-f(x)=0 \ \text{ or } \ F^T c=f(x) \tag{7}$$

Under this condition the mean squared error (MSE) of the predictor is:

$$\varphi(x)=E[(\hat{y}(x)-y(x))^2]=\sigma^2(1+c^T Rc-2c^T r) \tag{8}$$

The Lagrangian function for the problem of minimizing φ with respect to c and subject to the constraint (7) is

$$L(c,\lambda)=\sigma^2(1+c^T Rc-2c^T r)-\lambda^T(F^T c-f). \tag{9}$$

The partial derivatives of $L(c,\lambda)$ with respect to c and λ are

$$\tilde{\lambda}=(F^T R^{-1}F)^{-1}(F^T R^{-1}r-f) \tag{10}$$

$$c=R^{-1}(r-F\tilde{\lambda}) \tag{11}$$

The matrix R and therefore R^{-1} is symmetric, and by means of (5) we find

$$\hat{y}(x)=r^T R^{-1}Y-(F^T R^{-1}r-f)^T(F^T R^{-1}F)^{-1}F^T R^{-1}Y \tag{12}$$

Also with the generalized least squares solution to the regression problem $Y \approx F\beta$, we have

$$\beta^*=(F^T R^{-1}F)^{-1}F^T R^{-1}Y \tag{13}$$

$$\sigma^2=\frac{1}{m}(Y-F\beta^*)^T(Y-F\beta^*) \tag{14}$$

and insert β^* in (12), we find the predictor

$$\hat{y}(x)=f(x)^T\beta^*+r(x)^T\gamma^* \tag{15}$$

with $\gamma^*=R^{-1}(Y-F\beta^*)$.

From (15), it follows that the gradient $\hat{y}'=\left[\frac{\partial\hat{y}}{\partial x_1}\cdots\frac{\partial\hat{y}}{\partial x_n}\right]^T$ can be expressed as

$$\hat{y}'(x)=J_f(x)^T\beta^*+J_r(x)^T\gamma^* \tag{16}$$

where J_f and J_r is the Jacobian of f and r, respectivel

$$(J_f(x))_{ij}=\frac{\partial f_i}{\partial x_j}(x), \quad (J_r(x))_{ij}=\frac{\partial R}{\partial x_j}(\theta,s_i,x),$$

(17)

From the above, we find that R, β^* and σ^2 are determined by θ ,where the optimum of θ (θ^*) can be determined by the maximum likelihood estimate, in other words, we have to determine the solution to the following constrained optimization problem.

$$\begin{cases} Min \quad \varphi(\theta)=\left|R(\theta)\right|^{\frac{1}{m}}\cdot\sigma(\theta)^2 \\ Subject\ to: \quad \theta\geq 0 \end{cases} \tag{18}$$

When θ^* is found, Kriging prediction model can be established. Pattern search method was used to find the θ^* in Lophaven[15]. However, this method would get trapped into % and the solution would be greatly affected by the given initial value θ. Therefore, an advanced and efficient global optimization algorithm, chaos embedded particle swarm optimization algorithm[1], is first used here to search θ^*.

3. CHAOS EMBEDDED PARTICLE SWARM OPTIMIZATION ALGORITHM

3.1 Particle Swarm Optimization

PSO is a population-based optimization technique, where the performance of each particle is measured according to a predefined fitness function. Each solution called a ‘‘particle’’, flies in the problem search space looking for the optimal position to land. A particle, as time passes through its quest, adjusts its position according to its own ‘‘experience’’ as well as the experience of neighboring particles.

In PSO, a swarm consists of m particles, which is defined by $X=\{x_1,x_2,...,x_i,...,x_m\}$, moving around in a D-dimensional search space. The position and velocity of the i th particle at the t th iteration is represented by $X_i(t)=(x_{i1},x_{i2},...,x_{iD})$ and $V_i(t)=(v_{i1},v_{i2},...,v_{iD})$ respectively. During the search process the particle successively adjusts its position toward the global optimum according to the two factors: the best position encountered by itself (i.e. local-best position or its experience) denoted as $P_i=(p_{i1},p_{i2},...,p_{iD})$ and the best position encountered by the whole swarm (i.e. global-best position) denoted as $P_g=(p_{g1},p_{g2},...,p_{gD})$.The velocity and position of the particle at next iteration are updated according to the following equations:

$$V_i\ (t+1)=wV_i\ (t)+c_1r_1(t)(P_i(t)-X_i(t))+c_2r_2(t)(P_g(t)-X_i(t)) \quad (19)$$

$$X_i(t+1)=X_i(t)+V_i(t+1) \quad (20)$$

Here, w is called inertia weight that controls the impact of previous velocity of particle on its current one; c_1 and c_2 are positive constant parameters called acceleration coefficients which control the maximum step size; r_1 and r_2 are two independently uniformly distributed random variables with range [0,1]

In PSO, proper control of global exploration and local exploitation is crucial in finding the optimum solution efficiently. And w plays an important role in balancing between exploration and exploitation in PSO [17-18], which can be determined by the linear descend inertia weight (LDIWF) model.

$$w=w_1-\frac{t(w_1-w_2)}{T} \quad (21)$$

Here, w_1 and w_2 are starting and final values of inertia weight respectively; t is the current iteration number, and T is maximum iteration number. Normally, w_1 is set to 0.9 and w_2 to 0.4.

3.2 Chaotic local search (CLS)

Step 1. Set $k=0$ and map the decision variables $x_j^{(k)}$ among the intervals $(x_{\min,j}, x_{\max,j})$, $j=1,2,...,D$ to the chaotic variables $cx_j^{(k)}$ located in the interval $[0,1]$ using(22);

$$cx_j^{(k)}=\frac{x_j^{(k)}-x_{\min,j}}{x_{\max,j}-x_{\min,j}} \quad (22)$$

Step 2. Determine the chaotic variables $cx_j^{(k+1)}$ for the next iteration using(23);

$$cx_j^{(k+1)}=\begin{cases}2cx_j^{(k)} & 0\le cx_j^{(k)}<\frac{1}{2}\\ 2(1-cx_j^{(k)}) & \frac{1}{2}\le cx_j^{(k)}\le 1\end{cases} \quad (23)$$

Step 3. Convert $cx_j^{(k+1)}$ to $x_j^{(k+1)}$ using (24);

$$cx_j^{(k+1)}=x_{\min,j}+cx_j^{(k+1)}(x_{\max,j}-x_{\min,j}) \quad (24)$$

Step 4. Evaluate the new solution with the decision variable $x_j^{(k+1)}$;

Step 5) If the new solution is better than $X^{(0)}=[x_1^{(0)},...,x_D^{(0)}]$ or the predefined maximum iteration is reached, output the new solution as the result of the CLS; otherwise, let $k=k+1$ and go back to Step 2.

3.3Chaotic PSO

Based on the proposed PSO with LIDWF and the chaotic local search, a two-phased iterative strategy named Chaotic PSO (CPSO) is proposed, in which PSO with LIDWF is applied to perform global exploration and CLS is employed to perform locally oriented search (exploitation) for the solutions resulted by PSO, and in which N denotes the size of population, f_i represents the function value of the i th particle, and $f_{best}[i]$ represents the local-best function value for the best position visited by the i th particle.

Step 1. Set $t=0$. For each particle i in the population: (1) Initialize V_i *and* X_i randomly. (2) Evaluate f_i . (3) Initialize P_g . (4) Initialize P_i .

Step 2. For each particle i, update V_i *and* X_i according to (19), (20) and (21).

Step 3. Evaluate f_i for all particles.

Step 4. Reserve the top $N/5$ particles.

Step 5. Implement the chaotic local search (CLS) for the best particle, and update the best particle using the result of CLS with variables $p_{g,j}^{(k)}$, $\mathrm{j}=1,2,...,D$.

Step 6. If iteration number k reaches its maximum value K , output the solution found best so far.

Step 7. Decrease the search space:

Step 8.
$$x_{\min,j}=\max(x_{\min,j},\ p_{g,j}-r*(x_{\max,j}-x_{\min,j})),\ 0<r<1$$
$$x_{\max,j}=\min(x_{\max,j},\ p_{g,j}-r*(x_{\max,j}-x_{\min,j})),\ 0<r<1$$

Step 9. Randomly generate 4 $4N/5$ new particles within the decreased search space and evaluate them.

Step 10. Construct the new population consisting of the $4N/5$ new particles in Step 9 and the $N/5$ particles in Step 6. (1) for each particle i , $P_i=X_i$ if $f_i<f_{best}[i],\ \forall i\le N$. (2) Find P_g such that $f[P_g]\le f_i\ ,\forall i\le N$.

Step 11. If the iteration number t reaches its maximum value T , output the result; otherwise, let $t=t+1$ and go back to Step 2.

4. FURTHER DEVELOPMENT OF KRIGING TOOL BOX

In Lophaven[15], a matlab-based kriging toolbox is provided to establish the model. The function [dmodel, perf] = dacefit(S, Y, regr, corr, theta0) is used to establish the basic model. The output argument dmodel provides the values of parameters needed in the prediction model such as θ and β^*, while perf provides the optimization information of the objective function $\varphi(\theta)$ such as the optimal and iteration numbers. Input arguments S and Y are the same as in the Section 2.1. The input argument regr is the basis function used in the basic model, and corr the correlation function, while theta0 is the arbitrary initial value of θ. And in the function dacefit, pattern search method is used by Lophaven to search the optimal of $\varphi(\theta)$, which would get trapped into the local minimal, as shown in Table 1.

After the further development of the toolbox in the present study, the function [dmodel, perf] = dacefitcpso(S, Y, regr, corr) can be used to establish the basic model. Note that the input arguments of the function dacefitcpso do not include theta0, which can be initialized in the CPSO that is embedded in dacefitcpso to search the optimal of $\varphi(\theta)$.

Table 1 compares the results of two methods using the data provided in Lophaven[15], which shows that dacefitcpso is better than dacefit. And Figure 1 shows that dacefitcpso finds out the optimal of $\varphi(\theta)$ after 40 iterations.

Table 1 : Results Of Two Methods

	Function	$\psi(\theta_1,\theta_2)$	θ_1	θ_2
Lophaven'work	dacefit	0.041968	3.53553	2.10224
Present sudy	dacefitcpso	0.036979	1.27689	5.07057

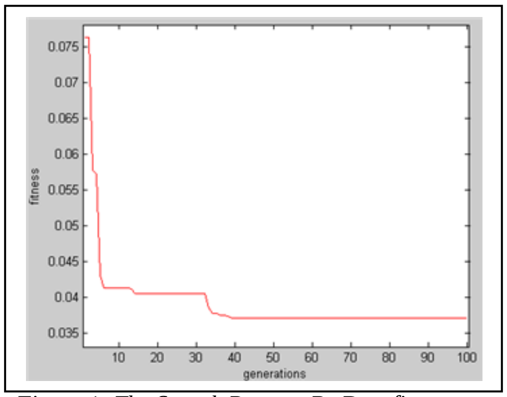

Figure 1: The Search Process By Dacefitcpso

5. COMPARISON WITH OTHER CURVE FITTING TOOLS

It is well known that performance of curve fitting can be evaluated from both its interpolating point and extrapolating points. Therefore, in the resent study, optimized Kriging model, polynomial function, back propagation artificial neural network(BP-ANN) and radial basis function artificial neural network (RBF-ANN) are used to simulate *y*=sinx with the sample points $(\frac{i*\pi}{10},\sin\frac{i*\pi}{10})$, $i=0,1,2,3,4,5,6$, as shown in Fig. 2 through Figure 1: 5. From the figures, it is found that the BP-ANN method simulates the function worst, even within the range of sample points; the RBF-ANN method and classical RSM will produce obvious error when extrapolating unobserved points; the kriging model by dacefitcpso is the best one to simulate the nonlinear function by interpolating and extrapolating the unobserved points.

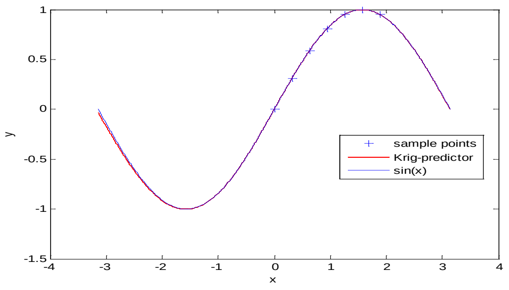

Figure 2: Simulation Of Y=Sinx By Dacefitcpso

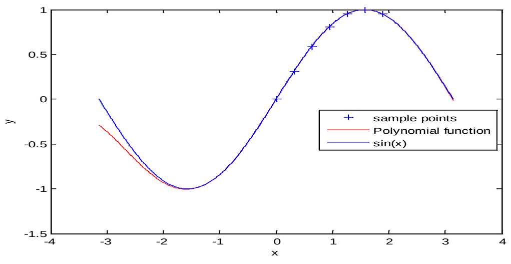

Figure 3: Simulation Of Y=Sinx By Classical Rsm

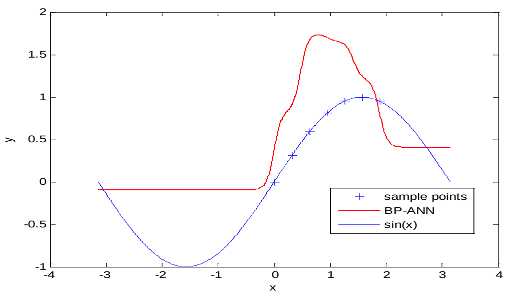

Figure 4: Simulation Of Y=Sinx By BP-ANN

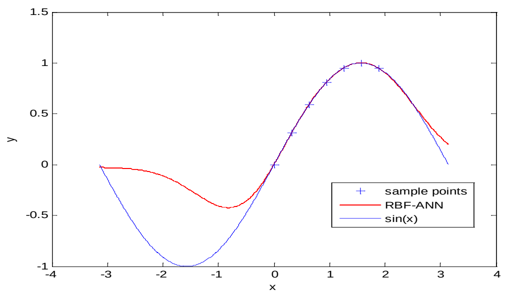

Figure5: Simulation Of Y=Sinx By RBF-ANN

6. CONCLUSIONS

The paper uses the chaotic particle swarm optimization algorithm to search the optimal value of parameters required in establishing kriging model, which is important for the precision of kriging model-based curve fitting. Then a typical example is selected to demonstrate the CPSO optimized Kriging model has a good curve fitting performance not only for the interpolating points but also for the extrapolating points, compared with other curve fitting tools like polynomials and artificial neural network.

REFERENCES:

[1] Guan XL, Melchers RE, "Multitangent-plane surface method for reliability calculation", *J Eng Mech*, Vol. 123, No. 4, 1997, pp. 996-1002.

[2] Faravelli L, "Response surface approach for reliability analysis", *J Eng Mech*, *Vol.* 115, No. 12, 1989, pp. 2763-81.

[3] Bucher CG, Bourgund U, "A fast and encient response surface approach for structural reliability problems", *Struct Saf*, Vol. 7, No. 1, 1990, pp. 57-66.

[4] Rajashekhar MR, Ellingwood BR, "A new look at the response surface approach for reliability analysis", *Struct Saf*, Vol. 12, No. 1, 1993, pp. 205-209.

[5] Deng J, Gu D, Li X, Yue Z, "Structural reliability analysis for implicit performance functions using artificial neural network", *Struct Safety*, Vol. 27, No. 1, 2005, pp. 25–48.

[6] Gomes HM, Awruch AM, "Comparison of response surface and neural network with other methods for structural reliability analysis", *Struct Safety*, Vol. 26, No. 1, 2004, pp. 49–67.

[7] Cho, S.E., "Probabilistic stability analyses of slopes using the ANN-based response surface", *Computers and Geotechnics*, Vol. 36, No. 5, 2009, pp. 787-797.

[8] Schueremans L, Van Gemert D, "Benefit of splines and neural networks in simulation based structural reliability analysis", *Struct Safety*, Vol. 27, No. 3, 2005, pp. 246–61.

[9] Chau KW, "Reliability and performance-based design by artificial neural network", *Adv Eng Software*, Vol. 38, No. 3, 2007, pp. 145–9.

[10] Hornik K, Stinchcombe M, White H, "Multi-layer feed-forward networks are universal approximators", *Neural Networks*, Vol.2, No. 5, 1989, pp. 359–68.

[11] Sacks J, Schiller SB, Welch WJ, "Design for computer experiment", *Technometrics*, Vol. 31, No. 1, 1989, pp. 41-47.

[12] Simpson TW, Mauery TM, Korte JJ, " Comparison of response surface and kriging models for multidisciplinary design optimization", *In: 7th AIAA/USAF/NASA/ ISSMO symposium on multidisciplinary analysis and optimization, St. Louis (MI)*, 2–4 September, AIAA-98-4755; 1998.

[13] Giunta A, Watson LT, Koehler J, "A Comparison of approximation modeling techniques: polynomial versus interpolating models", *In: 7th AIAA/USAF/NASA/ISSMO symposium on multidisciplinary analysis and optimization, St. Louis (MI)*, 2–4 September, AIAA-98-4758-CP; 1998.

[14] Irfan Kaymaz, "Application of kriging method to structural reliability problems", *Struct Saf*, Vol. 27, No. 2, 2005, pp. 133–6151.

[15] Lophaven SN, Nielsen HB, Søndergaard J, "DACE, *a Matlab kriging toolbox*, 2003.

[16] Liu Bo, Wang Ling, Jin Yi Hui, Tang Fang, Huang De Xian, "Improved particle swarm optimization combined with chaos", *Chaos, Solitons Fractals*, Vol. 25, No. 5, 2005, 1261–1271.