Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Quality control of batch process using natural gradient based model-free optimization

N.Y. Lu * **F. Zhao** * **J.H. Lu** ** **R.Y. Qi** *

* *School of Automation Engineering, Nanjing Univ. of Aeronautics and Astronautics,China*
** *School of Computer Science and Engineering, Southeast Univ.,China*
(e-mail: luningyun@nuaa.edu.cn)

Abstract: Model-based optimization (MBO) has been widely applied for quality control of batch processes, however, it is not easy to obtain globally effective and accurate quality model with affordable effort. Instead of building a quality model, model-free optimization (MFO) uses process data directly, which is more efficient and economic for quality control of batch process. Considering the complex nonlinearity and dynamics in batch process, a quality control scheme using natural gradient based dynamic optimization is proposed in this paper. Optimization algorithm is developed from the aspect of manifold in non-Euclidean space. An approximation method is derived for the calculation of the natural gradient, and a multivariate iterative sensitivity matrix based on Riemannian geodesic distance is proposed to obtain a novel adaptive stepping strategy. The proposed quality control scheme has been verified in injection molding process. A set of comparison tests are presented to demonstrate the feasibility and effectiveness of the proposed method.

Keywords: Batch process; quality control; model-free optimization; natural gradient.

1. INTRODUCTION

Product quality control of batch processes is a challenging topic due to the high nonlinearity, the complex batch- and/or time-dependent dynamics, and the evolving relationship between high-dimensional process variables and various end-product qualities. In many existing quality control schemes of batch processes (Lu and Gao [2005],Chen and Turng [2005]), accurate quality model is preliminarily required, based on which, quality controller searches for the optimal process settings to meet the end-product quality requirements. The quality models can be data-driven models (Wan et al [2012]),or first-principle models (Lucyshyn et al [2012]). However, it is not easy to obtain globally effective and accurate quality model with affordable effort, especially for the batch processes that involve frequent process changeovers or serious process uncertainties. Compared with the traditional quality control via model-based optimization (MBO), quality control via model-free optimization (MFO) is arousing attentions recently.

MFO based quality control uses experimental measurements rather than a quality model to search for the optimal process settings. MFO can not only cope with process uncertainty(Gattu and Zafiriou [1999]), but also be conducted for quality control of batch process in an efficient and economic way. As an implicit batch-to-batch optimization, MFO requires a suitable data-driven approach to search for the optimal direction. The search algorithms can be generally categorized into two types: gradient-free and gradient-based. Gradient-free algorithms are usually called *direct search*. They can be performed without calculating the gradients, which therefore reduces the number of additional experiments in the MFO. Gradient-free algorithms are simple and effective, but they are suitable only for low-dimensional problems, and they often suffer from the slow convergence speed. Gradient-based algorithms use gradient to provide search direction, which can assure faster convergence than the gradient-free algorithms. But the calculation (or approximation) of the gradient is a difficult task. Considering that fast convergence of quality attributes is very important as it concerns the production cost and profit, gradient-based optimization is studied.

Concerning different implementations of gradient approximation, gradient-based MFO approaches can be further divided into two types: deterministic gradient based and stochastic gradient based. Deterministic gradient algorithms (Sun and Yuan [2006]) can compute gradient exactly using the derivative of the cost function. However, since these algorithms are based on the quadratic approximation, they work well only in a small neighborhood for the optimal point (Park et al [2000]). In addition, the insufficient understanding of process mechanism and uncertainty limit their applications to large-scale problems. On the other hand, stochastic gradient algorithms are conducted by stochastic approximation of the derivative. Variable perturbation (Stevenson [1981]) and finite difference (Randall [2005]) are the regular tools to obtain the gradient approximation. Two-point gradient algorithm

⋆ The work was supported by NSFC (61374141, 61073059, 61374116), Jiangsu Province NSF (BK2010409) and the Fundamental Research Funds for the Central Universities (NS2012039).

(Dai et al [2002]) and simultaneous perturbation stochastic approximation (SPSA) (Spall [1992]) are the most widely used. They have in common that the next operating profile is deduced from the current one and the previous ones, so they have super linear convergence speed(Chen et al [2010]). This makes them quite attractive for MFO based quality control of batch processes. However, stochastic gradient based methods have to endure high optimization cost as they need a lot of perturbation experiments for gradient approximation. In order to reduce the optimization cost, natural gradient (Amari [1998]) was then proposed and has become an attractive concept in the domain of stochastic search.

Natural gradient is based on the Riemannian geometry rather than the conventional geometry in the Euclidean space, and it employs the knowledge of Riemannian structure of the parameter space to adjust the gradient search direction. Using natural gradient is likely to obtain more reasonable directions of steepest ascent than the using of deterministic gradient or stochastic gradient. Furthermore, natural gradient algorithm does not require that the cost function is locally quadratic (Zhang et al [2013]).Natural gradient has been widely used in motion segmentation (Subbarao and Meer [2009]), medical imaging (Fletcher [2007]), etc. The application of natural gradient to industrial process optimization is, however, not reported yet. Considering the inherent complex nonlinearity and dynamics of batch processes, better quality control performance can be expected by using the natural gradient based dynamic optimization.

This paper attempts to analyze batch process data from the aspect of manifold in non-Euclidean space. A natural gradient based model-free optimization algorithm is used to solve the quality control problem for batch process. An approximation method is derived for the calculation of the natural gradient, and a multivariate iterative sensitivity matrix based on Riemannian geodesic distance is proposed to realize a novel adaptive stepping strategy, which can improve the convergence speed in MFO and therefore reduce the cost of MFO-based quality control scheme for batch process.

2. A NOVEL QUALITY CONTROL SCHEME FOR BATCH PROCESSES

2.1 Dynamic Optimization Based Quality Control

The objective of quality control for batch process is to search for the optimal process settings batch-to-batch by solving the following dynamic optimization problem (Kong et al [2011];Srinivasan et al [2002]):

$$\begin{aligned} \underset{\boldsymbol{x}}{Min} \quad & J = |f(\boldsymbol{x}) + e(\boldsymbol{x}) - Q_{tg}| \\ s.t. \quad & x_i^L \le x_i \le x_i^H, i = 1, ...p; \\ & \boldsymbol{x} = (x_1, ...x_p)^T \end{aligned} \tag{1}$$

where $\boldsymbol{x}$ is a p-dimensional vector of operating variables; Q_{tg} is the target product quality; $f(\cdot)$ is a function for describing the relationship (usually unknown) between operating variables and end product quality; $e(\cdot)$ is the total effect of the uncertainty (caused by model mismatch, disturbances, etc.); x_i^L and x_i^H define the feasible operating window.

In a gradient descent based batch optimization problem, the iterative optimization strategy(Schlegel [2011])is often adopted to determine the optimal process settings for the next batch as follows,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \mu_k \nabla J(\boldsymbol{x}_k), k = 1, 2, ... \tag{2}$$

where k is the index of iteration; $\nabla J(\boldsymbol{x}_k)$ is the gradient (i.e., the optimization direction), and μ_k is the step size along the optimization direction. The computations of $\nabla J(\boldsymbol{x}_k)$ and μ_k are two important issues which determine the optimization performance.

Considering that batch processes usually have strong nonlinearity and complex dynamics, the variable space should not be viewed simply as the Euclidean space. For an optimization problem in non-Euclidean space, differential geometry based gradient calculation method is more suitable for batch processes. Natural gradient (NG)(Amari [1998]) is a good choice, which was developed from a perspective of Riemannian geometry, aiming to obtain more accurate search directions than those conventional gradient methods. Using the natural gradient, the iterative optimization in Eq.(2) can be modified as,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \mu_k \tilde{\nabla} J(\boldsymbol{x}_k), k = 1, 2, ... \tag{3}$$

where $\tilde{\nabla} J(\boldsymbol{x}_k)$ is the natural gradient, defined by,

$$\tilde{\nabla} J(\boldsymbol{x}_k) = G^{-1}(\boldsymbol{x}_k) \nabla J(\boldsymbol{x}_k) \tag{4}$$

The natural gradient at the point $\boldsymbol{x}_k$ is the product of the inverse of Riemannian metric $G(\boldsymbol{x}_k)$ and the conventional gradient $\nabla J(\boldsymbol{x}_k)$. G is a positive definite matrix, reflecting the local data structure on a Riemannian manifold. In order to realize Eq.(3), the Riemannian metric, G, should be obtained firstly for the calculation of the natural gradient, $\tilde{\nabla} J(\boldsymbol{x}_k)$; and then, the step size, μ_k, need to be determined from the perspective of Riemannian manifold.

2.2 Calculation of Natural Gradient

2.2.1 Riemannian metric

Assuming that $\boldsymbol{x}_{k-1}$ and $\boldsymbol{x}_k$ are two adjacent iteration points in the operating space of batch process, they can be viewed as two points lying on a Riemannian manifold M, as shown in Fig.1.

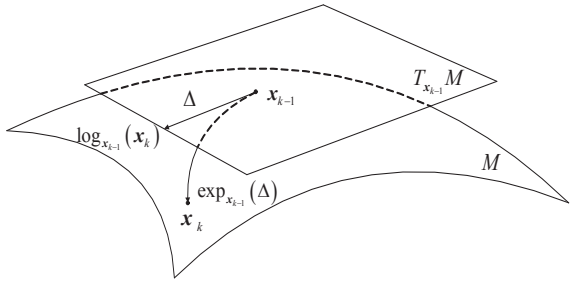

Fig. 1. Illustration of the Exponential and Logarithmic mapping between $\boldsymbol{x}_{k-1}$ and $\boldsymbol{x}_k$

$T_{\boldsymbol{x}_{k-1}}M$ is the tangent plane at point $\boldsymbol{x}_{k-1}$. The relationship between the two points $\boldsymbol{x}_{k-1}$ and $\boldsymbol{x}_k$ and the tangent Δ at point $\boldsymbol{x}_{k-1}$ can be described by the *exponential map* and the *logarithmic map* (Subbarao and Meer [2009]),

$$exp_{\boldsymbol{x}_{k-1}}(\Delta) = \boldsymbol{x}_k \tag{5}$$

$$log_{\boldsymbol{x}_{k-1}}(\boldsymbol{x}_k) = \Delta \tag{6}$$

Then, the Riemannian gradient (Castano-Moraga et al [2007]) at point $\boldsymbol{x}_k$ can be obtained by,

$$\nabla J_{Rg}(\boldsymbol{x}_k) = -log_{\boldsymbol{x}_{k-1}}(\boldsymbol{x}_k) \tag{7}$$

$$log_{\boldsymbol{x}_{k-1}}(\boldsymbol{x}_k) = \boldsymbol{x}_{k-1} log(\boldsymbol{x}_{k-1}^{-1}\boldsymbol{x}_k) \tag{8}$$

where

$$log(\boldsymbol{x}_{k-1}^{-1}\boldsymbol{x}_k) = \sum_{i=0}^{\infty} \frac{(-1)^{i-1}}{i}(\boldsymbol{x}_{k-1}^{-1}\boldsymbol{x}_k - \boldsymbol{I})^i \tag{9}$$

and $\boldsymbol{x}_{k-1}^{-1}$ is the Pseudo inverse of $\boldsymbol{x}_{k-1}$, $\boldsymbol{I}$ is a p-dimensional identical matrix.

With the Riemannian gradient $\nabla J_{Rg}(\boldsymbol{x}_k)$, the Riemannian metric $G(\boldsymbol{x}_k)$ can be estimated by (Yang and Laaksonen [2008]),

$$G(\boldsymbol{x}_k) = \nabla J_{Rg}(\boldsymbol{x}_k)(\nabla J_{Rg}(\boldsymbol{x}_k))^T \tag{10}$$

2.2.2 Calculation of the conventional gradient

The recursive procedure (Eq.(2) or Eq.(3)) is in the general stochastic approximation (SA) frame. Two-point gradient method (a variation of SPSA)(Pennec et al [2006]) is used to compute the conventional gradient. All elements of $\boldsymbol{x}_k$ are randomly perturbed to obtain two measurements, $Q(\boldsymbol{x}_k + c_k\Delta_k)$ and $Q(\boldsymbol{x}_k - c_k\Delta_k)$. Then, the gradient can be obtained approximately by,

$$\nabla J(\boldsymbol{x}_k) \approx \frac{Q(\boldsymbol{x}_k + c_k\Delta_k) - Q(\boldsymbol{x}_k - c_k\Delta_k)}{2c_k\Delta_k} \tag{11}$$

where Δ_k is a perturbation vector at the k^{th} iteration, c_k is the corresponding gain, $\boldsymbol{x}_k + c_k\Delta_k$ and $\boldsymbol{x}_k - c_k\Delta_k$ are two adjacent perturbation operating points. For convenience's sake, considering two adjacent iteration operating setting points $\frac{\boldsymbol{x}_k + \boldsymbol{x}_{k-1}}{2}$ and $\frac{\boldsymbol{x}_k - \boldsymbol{x}_{k-1}}{2}$, it is easy to get

$$\begin{aligned} Q(\boldsymbol{x}_k) - Q(\boldsymbol{x}_{k-1}) = Q(&\frac{\boldsymbol{x}_k + \boldsymbol{x}_{k-1}}{2} + \frac{\boldsymbol{x}_k - \boldsymbol{x}_{k-1}}{2}) \\ &- Q(\frac{\boldsymbol{x}_k + \boldsymbol{x}_{k-1}}{2} - \frac{\boldsymbol{x}_k - \boldsymbol{x}_{k-1}}{2}) \end{aligned} \tag{12}$$

Then, the gradient at $(\boldsymbol{x}_{k+1} + \boldsymbol{x}_k)/2$ is

$$\nabla J(\frac{\boldsymbol{x}_k + \boldsymbol{x}_{k-1}}{2}) = \frac{Q(\boldsymbol{x}_k) - Q(\boldsymbol{x}_{k-1})}{2 \cdot \frac{\boldsymbol{x}_k - \boldsymbol{x}_{k-1}}{2}} = \frac{Q(\boldsymbol{x}_k) - Q(\boldsymbol{x}_{k-1})}{\boldsymbol{x}_k - \boldsymbol{x}_{k-1}} \tag{13}$$

If the two points are very close, the gradient at $\boldsymbol{x}_k$ can be calculated approximately by

$$\nabla J(\boldsymbol{x}_k) \approx \nabla J(\frac{\boldsymbol{x}_k + \boldsymbol{x}_{k-1}}{2}) = \frac{Q_k - Q_{k-1}}{\boldsymbol{x}_k - \boldsymbol{x}_{k-1}} \tag{14}$$

With Eq.(10) and Eq.(14), it is easy to approximate the natural gradient at the k^{th} iteration using Eq.(4).

2.3 Determination of Step Size

Different step size rules lead to different optimization convergence performances. With respect to whether or not the step size is deterministic, step size rules can be classified into two branches: *Regular step size*(Sun and Yuan [2006]), such as Constant value, Given design formula, One-dimension search and Newton-type method; and *Adaptive step size* (Kalivas [1992];Ozen et al [2010]), such as Simulated annealing, Fuzzy control, etc.

For dynamic optimization problems, regular step size can not satisfy both convergence speed and tracking performance; while adaptive step size can accelerate the convergence generally. Thus, adaptive step size is focused in this paper, because batch processes usually have complex dynamics, and at the same time, have high requirement on the convergence speed. An adaptive step size algorithm based on Riemannian geodesic distance and iterative sensitivity matrix is proposed to determine the parameter μ_k in Eq.(3).

2.3.1 Riemannian geodesic distance

Let M be a Riemannian manifold. The tangent space at $x \in M$ to the manifold is denoted by T_x. Denote the metric of Riemannian manifold by $\langle\cdot,\cdot\rangle_x : T_x \times T_x \to R$. For a tangent vector $v \in T_x$ at the point $x \in M$,

$$\langle v, v\rangle_x = tr((x^{-1}v)^2) \tag{15}$$

Based on this metric, Simone [2010] has derived the geodesic curve function,

$$\alpha(t) = xexp(tx^{-1}v) \tag{16}$$

Then, the geodesic distance between two points x_1 and x_2 on the manifold is given by

$$d^2(x_1, x_2) = \int_0^1 tr(((\alpha(t))^{-1}\dot{\alpha}(t))^2)dt \tag{17}$$

In the problem of dynamic optimization based quality control of batch process, the Riemannian geodesic distance between two adjacent iterative operating points $\boldsymbol{x}_{k-1}$ and $\boldsymbol{x}_k$ can be obtained by

$$d_k = d(\boldsymbol{x}_{k-1}, \boldsymbol{x}_k) = \sqrt{tr(log^2(\boldsymbol{x}_{k-1}^{-1}\boldsymbol{x}_k))} \tag{18}$$

2.3.2 Iterative sensitivity matrix

From Eq.(18), the Riemannian geodesic distance d_k is a scalar. If multiplying the scalar distance by the obtained natural gradient $\tilde{\nabla}J(\boldsymbol{x}_k)$, each variable in $\boldsymbol{x}_k$ will get the same stepping speed. It is unreasonable because different variables may have different effects on the objective. In order to solve this problem, the following iterative sensitivity matrix is developed for determining the step size.

Multi-variable sensitivity can be computed by

$$\boldsymbol{s}_k = (Q_k - Q_{k-1}) * (\boldsymbol{x}_k - \boldsymbol{x}_{k-1})^{-1} = [s_k^1, s_k^2, ..., s_k^p]^T \tag{19}$$

where $(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})^{-1}$ is calculated by the pseudo inverse operation. In order to obtain the accumulative effect of each variable, it is necessary to calculate the sensitivity iteratively. The iterative sensitivity of the i^{th} variable is

$$S^i = \sum_k \frac{|s_k^i|}{\sum_p |s_k|} \tag{20}$$

The iterative sensitivity matrix for all process variables is given as

$$\boldsymbol{S}_k = \begin{bmatrix} S^1 & 0 & \cdots & 0 \\ 0 & S^2 & \cdots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & \cdots & 0 & S^p \end{bmatrix} \tag{21}$$

With Eq.(18) and Eq.(21), the proposed variable step size method, named Geodesic stepping matrix, can be computed by

$$\boldsymbol{M_k} = d_k \cdot \frac{\boldsymbol{S}_k}{tr(\boldsymbol{S}_k)} \tag{22}$$

which extends the step size from scalar to matrix, so that the variable with high sensitivity can obtain a relatively large stepping size to contribute more during process optimization.

Based on the above, Eq.(3) can be re-written as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{M_k}\tilde{\nabla}(\boldsymbol{x}_k), k = 1, 2, ... \tag{23}$$

2.4 Several practical aspects of Implementation

• Data normalization

All variables' measurements will be normalized into the range [0,1] to ensure a unified scale for different process variables. At the end of each iteration, the obtained operating settings need to be transformed back into their physical settings (i.e., the de-normalization) for manipulating the machine. The physical settings must be within the feasible region. If any of the operating settings exceeds the region, its nearest boundary value should be used to replace the calculated one.

• Start-up of the iterative optimization

Assume that the tolerance of quality deviation is ε, which is chosen according to production manager or customer's requirements. The quality control procedure is triggered when the measured quality exceeds the tolerance. By this time, the iteration index k is set to 1. Since the iterative optimization needs at least two initial points, the second point should be determined at the same time, which can be obtained by simultaneous perturbation method,

$$\boldsymbol{x}_2 = \boldsymbol{x}_1 + c\Delta \tag{24}$$

where Δ is a random perturbation vector, the elements in Δ are randomly assigned to be ± 1 with equal probability, c is a small positive scalar.

• Termination of the iterative optimization

Because of the uncertainty in both the processing and sensoring, it is better to feedback the average quality measurements in each iteration. Assume that the experiments conducted under the operating condition $\boldsymbol{x}_k$ are repeated N times, the feedbacked quality will be

$$\bar{Q}_k = \frac{1}{N}\sum_{i=1}^{N} Q_{k,i} \tag{25}$$

The iterative optimization process is repeated until the average quality measurements $\bar{Q}_k$ under the operating settings $\boldsymbol{x}_k$ in the k^{th} process adjustment satisfies the terminal criterion,

$$\left\|\bar{Q}_k - Q_{tg}\right\| \leq \varepsilon \tag{26}$$

The operating setting $\boldsymbol{x}_k$ is considered as the optimal setting, and the optimization process is terminated.

The major steps of implementation are listed as below.

(1) Compare the current product quality with the target value in real time.
(2) The quality control procedure is triggered when the observed quality measurement goes beyond the threshold. By this time, the iteration index k is set 1, the current operating condition and quality measurement are denoted as the first data pair $(\boldsymbol{x}_1, Q_1)$. Then, search the second iteration point $(\boldsymbol{x}_2, Q_2)$ using simultaneous perturbation method (*i.e.*, Eq.(24)).
(3) Calculate the Natural gradient $\tilde{\nabla}J(\boldsymbol{x}_k)(k \geq 3)$ using Eqs.(4), (10) and (14); meanwhile, calculate the stepping matrix $\boldsymbol{M_k}$ using Eqs.(19)∼(22); finally, the new iterative operation point $\boldsymbol{x}_{k+1}$ is yielded using Eq.(23).
(4) Conduct verification experiments under the operating condition $\boldsymbol{x}_{k+1}$, resulting in the average product quality $\bar{Q}_{k+1}$.
(5) If $\bar{Q}_{k+1}$ satisfies the termination criterion, terminate the quality control procedure; otherwise, repeat steps (3) and (4) iteratively.

3. CASE STUDY

Injection molding is an ideal batch process for the verification and application of the proposed quality control scheme.

3.1 Experimental Setup

The verification experiments are conducted using Moldflow Plastic Insight (MPI) , which is a commercial simulation software of injection molding process. The mold is a simple curving assembly part. Part weight is selected as quality variable, and its value can be obtained in the log file of each batch. The tolerance level of part weight variation is chosen less than 0.1%, Experimental material is High-density polyethylene (HDPE). Other parameters of injection molding machine and process controllers are set by default.

According to prior knowledge, part weight is manipulated by optimizing the set-points of mold temperature (x_1), melt temperature (x_2), injection time (x_3), packing time (x_4) and packing pressure (x_5). The descriptions and the feasible regions of operating variables are shown in Tab.1. Other experimental conditions are set by default.

Table 1. Operating variables for part weight control

Variable no.	description	bound	unit
x_1	mold temperature	[20-60]	°C
x_2	melt temperature	[120-255]	°C
x_3	injection time	[0.2-1]	s
x_4	packing time	[3-10]	s
x_5	packing pressure	[75-90]	Mpa

The machine disturbances are intentionally introduced to cause quality variation, which is realized by changing the grid length of the part mold in the MPI software. Because in Moldflow, different grid length corresponds to different match ratio, resulting in different part weights (Jay [2006]). In this paper, three kinds of grid length are used under the same condition, $\boldsymbol{x} = [50, 185, 0.4, 5, 85]^T$, and the corresponding match ratios and part weights are listed in Tab.2.

In the simulation, the first grid length in Table 2 is viewed as a baseline, so that the quality tolerance level is 0.02g because the weight variation is specified to be 0.1% (that is, $\varepsilon = 0.02$ in Eq(25)).

3.2 Results and Discussion

Three groups of testing experiments are arranged as following: optimization speed test, tracking performance test, and anti-disturbance test. In addition, the proposed natural gradient based MFO method (denoted as *Method A*) is compared with a SPSA based MFO method (Kong et al [2011])(denoted as *Method B*). The results and discussions are shown below.

Case I: Optimization speed test

In dynamic optimization of batch process, optimization cost is proportional to the number of batches needed during optimization. It can also be considered as the convergence speed in general optimization problems. Fig.2 shows the comparison results of optimization speed test for method A and Method B. The target value of part weight Q_{tg} is set to 19.32g, and the initial point Q_1 is 19.74g with grid length 2.99mm. The iteration process consists of a sequence of experimental runs, which contain the iteration runs and perturbation runs. For method A, there are no perturbation runs except the second iteration point (See Eq.(24)). But for Method B, there are 10 perturbation runs between the 6 iteration runs. From Fig.2, the two quality control trajectories start from the same initial point and converge into the same target value. However, the optimization costs are different. Method A reaches the target quality within 11 runs, whereas Method B needs 16 runs in total. For Method B, high optimization cost is caused by the perturbation runs (used for the gradient approximation of SPSA).

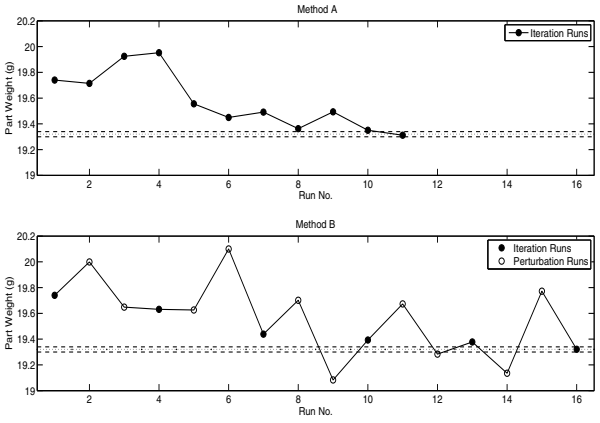

Fig. 2. Comparison results of optimization speed test

Case II: Tracking performance test

Batch processes are usually flexible for producing various products, thus the process is often operated over a range of conditions. Process engineers should find the new optimal operating condition to meet the new quality requirements for new process configuration as soon as possible. It is necessary to study the tracking performance of the proposed method.

Table 2. Grid lengths and weights

No.	grid length/mm	match ratio/%	weight/g
1	2.66	88.2	19.79
2	2.99	88.6	19.74
3	3.39	87.3	19.66

In this test, the initial point Q_1 is changed to 19.66g, which corresponds to No.3 in Tab.2. The initial quality target value is set to 19.79g and the tracking target is 19.39g. The comparison results of tracking performance test are plotted in Fig.3. The two methods can achieve similar performance when tracking the first target. After changing the target from 19.79g to 19.39g, Method A reaches the new quality target within 12 runs, whereas Method B needs 22 runs. The proposed method has quicker tracking speed.

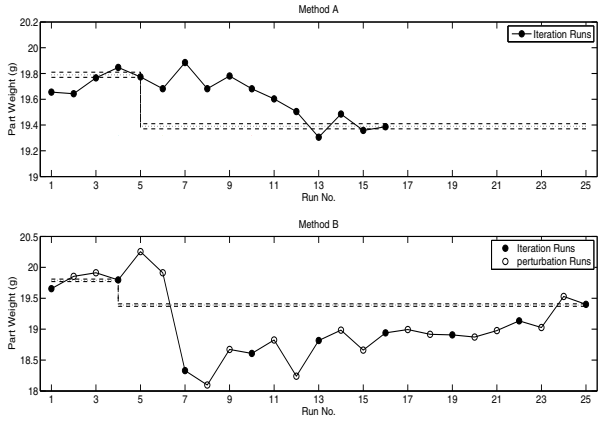

Fig. 3. Comparison results of tracking performance test

Case III: Anti-disturbance test

The product quality under the given optimal settings may vary due to process disturbances or uncertainties. In this anti-disturbance test, process disturbance is introduced by changing the mold grid length in Moldflow software, so that the products have fluctuating weights even under the same operating condition. The purpose of this test is to verify the robustness of the proposed method.

In this test, the initial point Q_1 is set to 19.74g, and the target quality is 19.5g. As shown in Fig.4, in the beginning, Method A needs 5 iteration runs to reach the target, while Method B needs 4 iteration runs and 6 additional perturbation runs. Then, the grid length of the mold is intentionally changed from 2.99mm to 3.39mm. By doing so, product weight is changed from 19.51g to 19.40g for Method A and from 19.51g to 19.39g for Method B. Weight fluctuations lead to a second optimization task. From the results, Method A has faster tracking speed than Method B, which also means the optimization cost of Method A is lower than Method B too.

In summary, the above experimental results show that the proposed quality control scheme can find the new optimal operating settings with faster optimization speed and less optimization cost.

4. CONCLUSIONS

A novel natural gradient based MFO has been presented for quality control of batch process from the aspect of Riemannian manifold and stochastic gradient optimization. The measurements rather than a quality model are directly

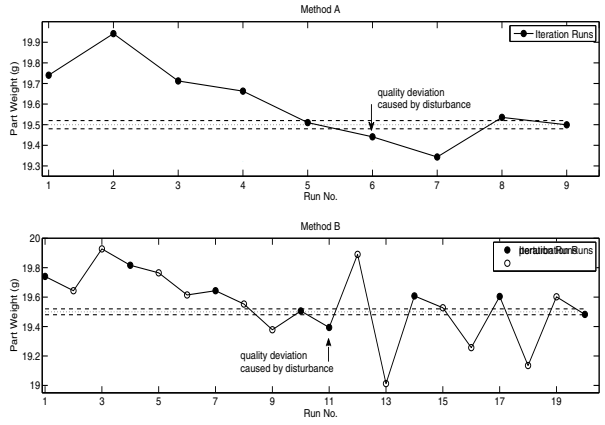

Fig. 4. Comparison results of anti-disturbance test

used to search for the optimal process setting. This iterative optimization method involves the determination of Natural gradient and adaptive step size. An approximation method for the calculation of the natural gradient has been derived in non-Euclidean space. A multivariate iterative sensitivity matrix based on Riemannian geodesic distance has been proposed to obtain a novel adaptive stepping strategy. The proposed method can improve the convergence speed in the MFO process and therefore reduce the cost of MFO-based quality control scheme for batch process. The proposed quality control scheme has been verified for product weight control in injection molding process using Moldflow software. The experimental results can illustrate the advantages of the proposed method.

REFERENCES

Amari,S.(1998). Natural gradient works efficiently in learning. *Neural Comput.*, 10(2):251–276.

Castano-Moraga,C.A., Lenglet,C., Deriche, R.,and Ruiz-Alzola, J.(2007). A Riemannian approach to anisotropic filtering of tensor fields. *Signal Processing*, 87(2):263–276.

Chen, X., Zhang, L.,and Kong, X.S.(2010). Automatic Velocity Profile Determination for Uniform Filling in Injection Molding. *Poly. Eng. Sci.*, 50(7):1358–1371.

Chen, Z.B. and Turng, L.S.(2005). A review of current developments in process and quality control for injection molding. *Adv. Polym. Tech.*, 24(3):165–182.

Dai, Y.H., Yuan, J.Y.,and Yuan, Y.X.(2002). Modified Two-Point Stepsize Gradient Methods for Unconstrained Optimization. *Comput. Optim. Appl.*, 22(1): 103–109.

Fletcher, P.T.(2007). Riemannian Geometry for the Statistical Analysis of Diffusion Tensor Data. *Signal Processing*, 87(2):250–262.

Gattu, G.,and Zafiriou, E.(1999). Methodology for on-line setpoint modification for batch reactor control in the presence of modeling error. *J Chem. Eng*, 75(1):21–29.

Jay S. *Moldflow Design Guide: A Resource for Plastics Engineers.* chapter 5. Moldflow Corporation, Massachusetts, 2006.

Kalivas, J.H.(1992). Optimization using variations of simulated annealing. *Chemomet. Intellig. Lab. Sys.*, 15(1):1–12.

Kong, X.S., Yang, Y., Chen, X., Shang, Z.J.,and Gao, F.R.(2011). Quality Control via Model-Free Optimization for a Type of Batch Process with a Short Cycle Time and Low Operational Cost. *Ind. Eng. Chem. Res.*, 50(5):2994–3003.

Lee, J.M. *Introduction to Topological Manifolds.* page 29, Springer, New York, 2000.

Liu, J.Q., Feng, D.Z.,and Zhang, W.W.(2009). Adaptive improved natural gradient algorithm for blind source separation. *Neural Comput.*, 21(3):872–889.

Lu, N.Y.,and Gao, F.R.(2005). Stage-Based online Quality Control for Batch Processes. *Ind. Eng. Chem. Res.*, 45(7):2272–2280.

Lucyshyn, T., Kipperer, M.,and Kukla, C.(2012). A Physical Model for a Quality Control Concept in Injection Molding. *J. Appl. Poly. Sci.*, 124(6):4926–4934.

Nelder, J.A.,and Mead, R.(1965). A Simplex-Method for Function Minimization. *Computer J.*, 7(4):308–303.

Ozen, A., Kaya, I.,and Soysal, B.(2010). Variable Step-Size Constant Modulus Algorithm Employing Fuzzy Logic Controller. *Wireless Pres. Commun.*, 54(2):237–250.

Park, H., Amari, S.,and Fukumizu, K.(2000). Adaptive natural gradient learning algorithms for various stochastic models. *Neural Netw.*, 13(7):755–764.

Pennec, X., Fillard, P.,and Ayache, N.(2006). A Riemannian framework for tensor computing. *Int. J. Comput. Vis.*, 66(1):41–66.

Randall, J.L. *Finite Difference Methods for Differential Equations.* page 6, University of Washington, Seattle, 2005.

Schlegel, H.B.(2011). Geometry optimization. *Wiley Interdiscip Rev Comput Mol Sci.*, 1(5):790–809.

Simone, F.(2010). Learning by Natural Gradient on Noncompact Matrix-Type Pseudo-Riemannian Manifolds. *IEEE Trans. Neural Netw.*, 21(5):841–852.

Spall, J.C.(1992). Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Trans. Automat. Contr.*, 37(2):332–341.

Srinivasan, B., Bonvin, D., Visser, E.,and Palanki, S.(2002). Dynamic optimization of batch processes: II. Role of measurements in handling uncertainty. *Comput. Chem. Eng.*, 27(1):27–44.

Stevenson, P.M.(1981). Optimized perturbation theory. *Phys. Rev.*, 23(12):2916–2944.

Subbarao, R.,and Meer, P.(2009). Nonlinear Mean Shift over Riemannian Manifolds. *Int J. Comput. Vis.*, 84(1): 1–20.

Sun, W.Y.,and Yuan, Y.X. *Optimization Theory and Methods.* chapter 3-4, Springer, New York, 2006.

Wan, J., Marjanovic, O.,and Lennox, B.(2012). Disturbance rejection for the control of batch end-product quality using latent variable models. *J Process Contr.*, 22(3):643–652.

Yang, Z.Y.,and Laaksonen, J.(2008). Principal whitened gradient for information geometry. *Neural Netw.*, 21(2): 232–240.

Zhang, Z.N., Sun, H.F.,and Peng L.Y.(2013). Natural gradient algorithm for stochastic distribution systems with output feedback. *Differ. Geom. Appl.*, 31(5):682–690.