A Globally Convergent Run-to-run Control Algorithm with Improved Rate of Convergence

G. François, B. Srinivasan and D. Bonvin
Laboratoire d'Automatique
École Polytechnique Fédérale de Lausanne
CH-1015 Lausanne, Switzerland.

Abstract— **Run-to-run control consists of using the measurements from previous runs to drive the outputs of the current run towards desired set points. From a run-to-run perspective, a dynamic system can be viewed as a *static* input-output map. For systems where this static map corresponds to a sector nonlinearity, a globally convergent fixed-gain run-to-run control algorithm has been recently proposed by the authors. This paper shows that it is possible to improve the rate of convergence of this algorithm by adapting its gain. Thus, the run-to-run controller switches from a Broyden update to a fixed low-gain update whenever it is necessary to jump over a local minimum. This algorithm is shown to be globally convergent for systems with sector nonlinearities.**

Keywords : Run-to-run control, Sector nonlinearity, Gauss-Newton, Broyden update, Convergence analysis.

I. Introduction

The class of systems where the operation is repeated over time has received increasing attention in recent years [1]. Many industrial processes, especially in the areas of batch chemical production, mechanical machining and semiconductor manufacturing, fall under the category of repetitive dynamic processes [2], [3]. Run-to-run control exploits the repetitive nature of a process to drive the outputs to desired set points. For this, the measurements obtained from *previous* runs are used to adapt the manipulated variables of the *current* run [4], [5]. These control schemes are very attractive in practice since they only require measurements that are available at the end of the run.

Though the system is truly dynamic, from a run-to-run perspective (i.e. upon integration of the within-run dynamics), the map between the manipulated and controlled variables is a *static* one [6], [7]. Upon completion, each run can be seen as mapping parameters that characterize the input trajectories to output values available at final time. An important feature of run-to-run control is the presence of an implicit one-run delay between the update of the manipulated variables at the beginning of the run and the availability of the outputs at the end of the run.

The standard run-to-run control techniques use linearization for controller design [8]. The static nonlinear map is linearized at some operating point, for which a linear controller is designed. The difficulty with this approach arises from the fact that the linearization, which is locally valid, may no longer be appropriate when the operating point changes. For systems with sector nonlinearities [9], [10], an algorithm that is globally convergent has been proposed recently [11]. Global convergence is ensured provided the controller gain is low, i.e. it does not exceed a limit that is inversely proportional to the slope of the sector. However, since the worst-case slope needs to be considered, the convergence of this low-gain controller may be rather slow.

The problem of computing the inputs that bring a static system towards desired set points is equivalent to that of solving a set of nonlinear equations. For this latter problem, several numerical methods that exhibit fast convergence are available in the literature [12]. However, since global convergence can rarely be proven for these methods, they are prone to converge to a local minimum.

This paper proposes to combine a Newton-Raphson type of update (for fast convergence) with a fixed low-gain update (to ensure global convergence). The run-to-run controller is typically updated using Broyden formula, except in the neighborhood of a local minimum where a fixed low gain is used. The main advantage of this algorithm is its relative high speed of convergence, while still guaranteeing global convergence.

The paper is organized as follows. The run-to-run control of linear and nonlinear systems is reviewed in Section II. The class of nonlinearities addressed in this paper and the earlier convergence results are also presented therein. Section III describes the standard approach for solving nonlinear equations, while the novel algorithm that is globally convergent and allows significant improvement in the rate of convergence is proposed in Section IV. This algorithm is then illustrated in Section V through the run-to-run control of a simple illustrative example, and Section VI concludes the paper.

II. Run-to-run Control

Consider the control of a repetitive dynamic process that is characterized by two independent time variables, the run time t, $t \in [0,t_f]$, and the run index k, $k = 1,2,\cdots$, where t_f is the final time in run k. The input profiles, $u_k[0,t_f]$, can be parameterized using a finite (typically low) number of input parameters, $p_k \in \Re^m$ [13]. Thus, knowledge of the parameters p_k allow the construction of the entire input profile $u_k(t), \forall t \in [0,t_f]$. This profile is applied to the dynamic system that evolves with the states $x_k(t), \forall t \in [0,t_f]$ and produces the run-end outputs $z_k \in \Re^q$ that are available at final time. Thus, from a run-to-run perspective, one does not consider the run-time dynamics between $u_k(t)$ and $x_k(t)$ explicitly, but rather the *static* input-output map $z_k = z(p_k)$. Figure 1 illustrates these concepts.

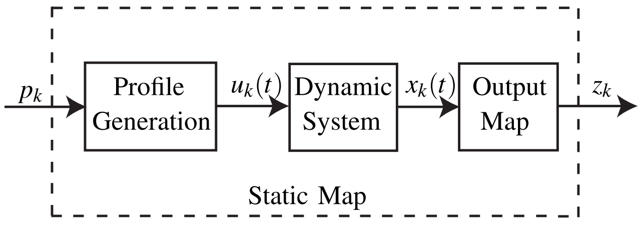

Fig. 1. Input-Output Static Map

In run-to-run control, the input parameters p_k are updated between consecutive runs using the run-end outputs of the previous runs to eventually meet the set points for the outputs, say $z_{ref} = 0$. A square system is assumed here for simplicity, i.e. $q = m$ Figure 2 represents schematically the run-to-run adaptation of the input parameters based on integral action:

$$p_{k+1} = p_k + K(z_{ref} - z_k) \tag{1}$$

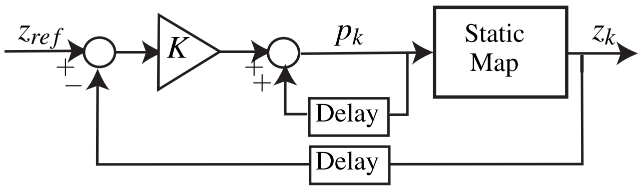

Fig. 2. Run-to-Run Control Based on Integral Action

where K is the $m \times m$ controller gain matrix. Note that the delays introduced in Figure 2 traduce the fact that the computation of the input parameters that will be used in the $k+1^{th}$ run uses the values of the input parameters and the outputs that are defined for the k^{th} run.

It is furthermore assumed that there exists a value p^* for which $z(p^*) = z_{ref} = 0$. The challenge in run-to-run control arises from the fact that p^* is typically unknown and needs to be determined.

A. Run-to-run Control of Linear Systems

If the static map is linear, i.e. $z = H(p - p^*)$ with H a $m \times m$ matrix, the following result provides an adaptation law that can be used to enforce $p_k = p^*$ and $z_k = 0$ as $k \to \infty$.

Theorem 1: Let $z = H(p - p^*)$, where H is full rank. Let the adaptation law be

$$p_{k+1} = p_k - \gamma H^{-1} z_k \tag{2}$$

where γ is a scalar gain, p_k the input parameters used in the k^{th} run, and z_k the corresponding run-end outputs. Then, for $0 < \gamma < 2$, $p_k \to p^*$ and $z_k \to 0$ as $k \to \infty$.
The proof is straightforward and can be found in [11].

B. Convergence Analysis for a Class of Nonlinear Systems

This section considers a class of nonlinearities where there is agreement between the local and global pictures. This class of nonlinear systems is a special case of systems with sector nonlinearities [10].

The following assumption is made:

There exists a full-rank $m \times m$ matrix $\bar{H}$ and a scalar $\alpha > 0$ such that

$$z^T z < \alpha z^T \bar{H}(p - p^*), \quad \forall p \neq p^* \tag{3}$$

Note that this assumption implies that there exists a p^* for which $z = 0$. Using the notation $\Delta p = p - p^*$, the classical definition of sector nonlinearity is $(z - a\bar{H}\Delta p)^T(b\bar{H}\Delta p - z) > 0$, with $0 \le a \le b$ [10]. This means that the nonlinearity lies between two linear functions, $a\bar{H}\Delta p$ and $b\bar{H}\Delta p$, as shown in Figure 3. The condition imposed here, $z^T z < \alpha z^T \bar{H}\Delta p$, is a special case of sector nonlinearity with $a = 0$ and $b = \alpha$. Note that, a lot of nonlinear systems, including real systems ([14]), can fall under this category provided a large value for b is choosen. Note that the

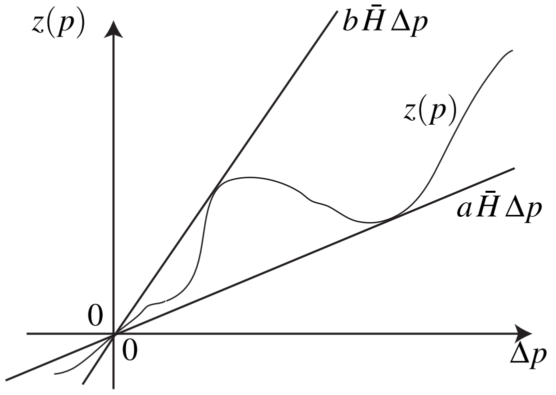

Fig. 3. One-dimensional example of sector nonlinearity

linearization is not around the (unknown) solution p^* but rather some arbitrary operating point. With this assumption, the following theorem can be stated.

Theorem 2: Consider the static map $z(p)$ with $z = 0$ for $p = p^*$. Let $\bar{H}$ be a full-rank $m \times m$ matrix and α a scalar such that (3) is satisfied. Also, consider the adaptation law

$$p_{k+1} = p_k - \gamma \bar{H}^{-1} z_k \tag{4}$$

where γ is a scalar gain. For $0 < \gamma < \frac{2}{\alpha}$, $p_k \to p^*$ and $z_k \to 0$ as $k \to \infty$.
The proof of this theorem uses a Lyapunov approach and can be found in [11]. It is interesting to note that the Lyapunov function used in the proof is $V_k = \Delta p_k^T \bar{H}^T \bar{H} \Delta p_k$ where $\Delta p_k = p_k - p^*$. Denoting by $\bar{z}_k$ the linear estimate of z_k, i.e. $\bar{z}_k = \bar{H}\Delta p_k$, it follows that $V_k = \bar{z}_k^T \bar{z}_k$. The interest of this remark will be showed in the next sections.

The main difficulty with this run-to-run control algorithm lies in its potentially slow rate of convergence: a larger sector, i.e. a larger α, calls for a smaller γ, and thus slower adaptation. Hence, α should be chosen as the smallest value that satisfies (3). Even with this choice, the rate of convergence may be slow and the algorithm extremely time-consuming when starting far form the solution.

III. Solving Nonlinear Equations Via Numerical Methods

Run-to-run control consists of iteratively finding the value p^* that satisfies $z(p^*) = z_{ref} = 0$ where $z(p)$ is a set of nonlinear algebraic equations. Hence, it is natural to want to design run-to-run controllers using the numerical methods that are available for solving nonlinear equations.

A. Newton-Raphson Method

The Newton-Raphson method is probably the most simple iterative method for solving the nonlinear equations $z(p) = 0$. With this method, the inverse of the Jacobian is used for adaptation:

$$p_{k+1} \quad = \quad p_k - J_k^{-1} z_k \tag{5}$$

where $J_k = \frac{\partial z}{\partial p}|_{p_k}$ is the Jacobian of $z(p)$ at $p = p_k$.

Under certain assumptions, it can be shown that, if the initialization point is sufficiently close to the extremum, Newton-Raphson method is well defined and converges quadratically [12]. However, this method has two main disadvantages:

1) It assumes the knowledge of the Jacobian J_k at all the points z_k. However, only the values of z_k are known in the context of run-to-run control.
2) There can be points p_k for which the Jacobian J_k loses rank, i.e. for the minima of $z(p)$. These points can make the algorithm diverge since the inverse of the Jacobian is not defined.

B. Jacobian Estimation using Broyden Formula

The adaptation law (5) uses the Jacobian of z. The problem now consists of estimating this Jacobian from values of z_k. The most straightforward idea is to use finite differences, which can be experimentally very expensive.

An alternative is to estimate the Jacobian matrix iteratively using, for example, quasi-Newton methods. Let's denote by $B_k(p_k)$ the estimate of the Jacobian matrix at p_k. Quasi-Newton methods build a linear model, $L_k(p) = z_k + B_k(p - p_k)$, which approximates z_k in the neighborhood of p_k.

Defining $\delta_k = z_{k+1} - z_k$ and $s_k = p_{k+1} - p_k$ leads to the classical secant equation:

$$B_{k+1} s_k = \delta_k \tag{6}$$

Except for the case $q = m = 1$, this equation has an infinite number of solutions for B_{k+1}. Thus, the challenge is to pick a solution that exhibits good properties. Broyden [15] proposed to select the Jacobian estimate that minimizes the variation between the successive linear models L_k and L_{k+1}, thus leading to the update formula:

$$B_{k+1} = B_k + \frac{(\delta_k - B_k s_k)\, s_k^T}{s_k^T s_k} \tag{7}$$

IV. Proposed Algorithm

To overcome the local nature of most of the numerical methods used for solving nonlinear equations, it is proposed here to switch from a *variable-gain update*, e.g. Newton-Raphson with J_k estimated using Broyden formula, to a *fixed low-gain update* whenever there is risk of converging towards a local minimum, for which the Jacobian may lose rank. This fixed low-gain update corresponds to the adaptation law in Theorem 2.

A. Algorithm Description

Let's introduce the function $f(p_k) = z_k^T z_k$. The algorithm proceeds as follows: a Newton-Raphson algorithm with adaptation of the Jacobian estimate is used as long as $f(p_k)$ decreases significantly. Hence, the adaptation proceeds with the variable-gain update as long as

$$f(p_{k+1}) \quad < \quad (1-\beta) f(p_k) \tag{8}$$

where β is a scalar, $0 < \beta < 1$, that is introduced to guarantee a certain reduction in f. If the descent rate is lower than β (and the solution has not been reached yet, as indicated by $f(p_k) < \varepsilon$, with ε a small positive scalar), a local minimum of $z(p)$ is being approached. In such a case, the algorithm switches to the fixed low-gain update until $f(p_k)$ has decreased sufficiently. Then, the algorithm switches back to the variable-gain update. The fact that f will decrease sufficiently with the fixed low-gain adaptation is guaranteed by Theorem 2. In fact, f can be seen as a Lyapunov function. The proposed algorithm is given in Figure 4.

B_0 is initialized as $\bar{H}$ unless a better guess is available. Furthermore, the runs done with fixed low-gain adaptation can be used to update the Jacobian estimate, using $\delta_k = z_{k+1}^0 - z_k^0$ and $s_k = p_{k+1}^0 - p_k^0$.

B. Convergence Analysis

Using the results of the previous sections, the following theorem can be stated.

Theorem 3: Consider the static map $z(p)$ with $z = 0$ for $p = p^*$. Let $\bar{H}$ be a full-rank matrix and α a scalar such that (3) is satisfied. Then, the algorithm in Figure 4 exhibits global convergence to $z = 0$ for $\gamma < \frac{2}{\alpha}$. Moreover, $f(p_k) = z_k^T z_k$ acts as a Lyapunov function.

Proof: Consider the function $f(p_k^0) = z_k^{0^T} z_k^0$. For it to be a Lyapunov function, it should be positive definite and decrease with the iteration number k. f is positive definite since it is a quadratic function. Also, $f(p^*) = 0$. To prove that $f(p_{k+1}^0) < f(p_k^0)$, two cases need to be distinguished.

1) If the variable-gain update generates a point for which (8) is satisfied, then $f(p_{k+1}^0) < (1-\beta) f(p_k^0) < f(p_k^0)$.
2) If the variable-gain update does not generate a point for which (8) is satisfied, then the algorithm switches to the fixed low-gain update. In this case also, it can be shown by contradiction that $f(p_{k+1}^0) < f(p_k^0)$:

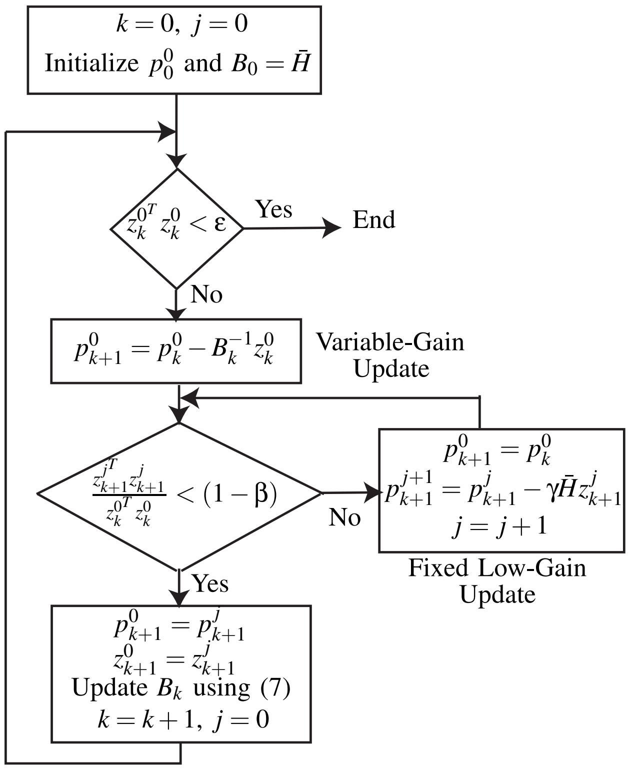
$k=0,\ j=0$
Initialize p_0^0 and $B_0=\bar{H}$
$z_k^{0^T} z_k^0 < \varepsilon$
Yes
End
No
$p_{k+1}^0 = p_k^0 - B_k^{-1} z_k^0$
Variable-Gain Update
$\frac{z_{k+1}^{j^T} z_{k+1}^j}{z_k^{0^T} z_k^0} < (1-\beta)$
No
$p_{k+1}^0 = p_k^0$
$p_{k+1}^{j+1} = p_{k+1}^j - \gamma\bar{H} z_{k+1}^j$
$j=j+1$
Fixed Low-Gain Update
Yes
$p_{k+1}^0 = p_{k+1}^j$
$z_{k+1}^0 = z_{k+1}^j$
Update B_k using (7)
$k=k+1,\ j=0$

Fig. 4. Proposed algorithm where k is the iteration number of the outer loop and j the iteration number of the inner loop

Suppose $f(p_{k+1}^0) \geq (1-\beta) f(p_k^0)$. At the inner-loop level, this implies that, for all j, $f(p_{k+1}^j) \geq (1-\beta) f(p_k^j) > (1-\beta)\varepsilon \neq 0$. On the other hand, since the fixed low-gain update is asymptotically convergent, there always exists a j for which $f(p_{k+1}^j) < \sigma$, for any non-zero positive σ. This leads to a contradiction and, hence, it can be stated that $f(p_{k+1}^0) < (1-\beta) f(p_k^0) < f(p_k^0)$.

Hence, Condition (8) is satisfied for all k by construction, and $f(p_k)$ is a Lyapunov function for the variable-gain update and converges exponentially in k towards 0. Note that this convergence is only exponential in k, i.e. for the outer loop, and not in the total number of iterations. ■

Five remarks are in order:

1) The algorithm uses the fast convergence property of the Newton-Raphson algorithm whenever possible, and the robust convergence property of the fixed-gain approach to cautiously avoid points for which the computation of the inverse of the Jacobian may lead to divergence.
2) Note the similarity of the Lyapunov functions for the inner loop $\bar{z}^T \bar{z}$ and for the outer loop $z_k^{0^T} z_k^0$. In fact, the function f is forced to be a Lyapunov function for the outer loop using the fact that its linear prediction $\bar{z}^T \bar{z}$ is a Lyapunov function for the inner loop.
3) A justification for the choice of the variable-gain adaptation lies in the fact that, if B_k is a good approximation of J_k, the variable-gain update is in the descent direction: $\partial f_k = \frac{\partial f}{\partial p}|_{p_k} \partial p = -z_k^T J_k B_k^{-1} z_k = -z_k^T z_k < 0$ if $B_k = J_k$.
4) Of course, if numerous calls to the inner loop are needed, it might even happen that the fixed low-gain adaptation converges faster than the proposed algorithm. This would mean that there are many points where a full Newton step does not decrease f. In this case, however, the problem can be considered asreally difficult to solve and the low-gain update can no longer be considered as slow.
5) Since $f(p_k^0) = z_k^{0^T} z_k^0$ is a Lyapunov function for the variable-gain update and not for the fixed low-gain update, $z_k^{j^T} z_k^j$ can grow from a run to the next upon switching to the fixed low-gain adaptation.

C. Proposed Algorithm vs. Numerical Optimization

Numerical optimization methods can also be used for solving nonlinear equations. The solution of $z(p) = 0$ corresponds to the global minimum of $f = z^T(p) z(p)$. Unfortunately, this optimization problem can have local minima.

The most commonly-used numerical optimization methods are the quasi-Newton methods, for which the search direction is a rotated version of the gradient. Along this direction, a line search procedure is performed, i.e. one searches for a step size that satisfies the Goldstein conditions [16], [17]. Then, the Hessian is updated using an appropriate update formula [18].

Though the function to be minimized in numerical optimization and the Lyapunov function for the proposed algorithm are similar, there are some fundamental differences between the two approaches:

1) The gradient-based numerical optimization methods may converge to a local minimum, in contrast to the proposed algorithm.
2) In the proposed algorithm, the fact that $f(p^*) = 0$ is used to choose the Lyapunov function. In gradient-based algorithms, this information is not exploited.
3) In the proposed algorithm, the fixed low-gain adaptation replaces standard line search procedures. In other words, instead of trying to satisfy the Goldstein conditions [16] that are based on the gradient of the objective function, "small" steps are done until the local minimum that is being reached is jumped over.

V. Illustrative example

The goal of this section is to show through a simple example the improved performances of the proposed algorithm compared to the fixed low-gain update of Theorem 2.

Consider the set of nonlinear equations $z(p)$:

$$\begin{aligned} z_1 &= p_1 + p_2 + 2\sin(p_1) \\ z_2 &= p_2 + 2\sin(p_1) + 2\sin(p_2) \end{aligned} \tag{9}$$

This square two-input-two-output system exhibits a sector nonlinearity characterized by $\bar{H} = I$ and $\alpha_{min} = 4.59$. To illustrate this, the function $T(z,p) = \frac{z^T z}{z^T \bar{H} p}$ is plotted as a function of p in Figure 5. This function is not defined for $p_1 = p_2 = 0$ and exhibits a maximum that explicits $\alpha_{min} = 4.59$. This is the smallest value of α for which System (9) satisfies Equation 3 with $\bar{H} = I$. Hence, the maximum gain for the fixed low-gain adaptation is $\gamma = \frac{2}{4.59} = 0.435$.

This system is such that $f = z^T z$ has at least three minima, two local ones and the global one $(0,0)\ \forall p_1, p_2 \in [-10;\ 10]$ (Figure 6). Note that the knowledge of the global solution $(0,0)$ is not mandatory and is only assumed for illustrating the convergence of the algorithm.

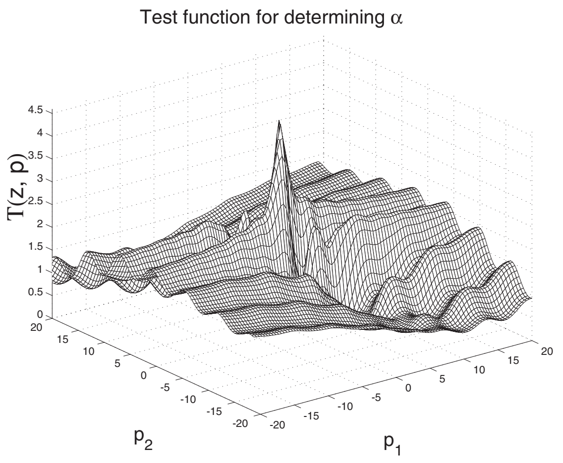

Fig. 5. Test function $T(z,p) = \frac{z^T z}{z^T \bar{H} p}$ for determining α

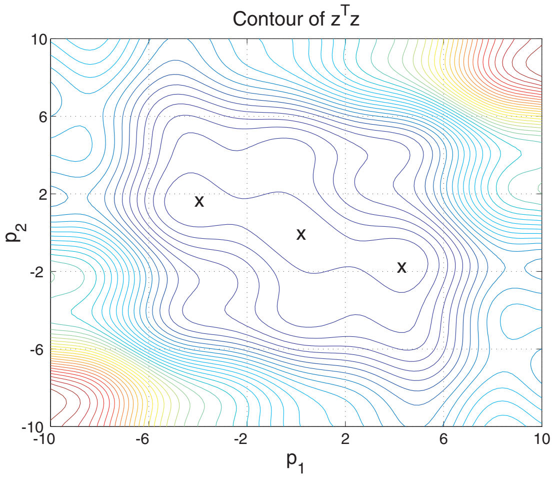

Fig. 6. Contour of the evaluation function with 40 levels

Figures 7 and 8 illustrate the convergence using both the fixed-gain (using (4) and the proposed algorithms (using (5), and using (4) whenever necessary). Both approaches converge to the global solution. Table I shows the evolution of f in function of the run number.

β was fixed to 0.01. In other words, the algorithm switched to the fixed low-gain adaptation whenever the outer loop did not succeed in reducing the function f by at least 1%. With this value of β, only one call to the inner loop of the algorithm was necessary. Also note that the termination tolerance on f, i.e. ε, was fixed to 0.005. As a result, the proposed algorithm was approximately 3 times faster than the low-gain one as can be seen in Table I.

TABLE I

ADAPTATION RESULTS

Algorithm	Run 3	Run 5	Run 7	Run 21
Fixed Gain	$f = 13.06$	$f = 10.29$	$f = 12.39$	$f = 0.0047$
Proposed	$f = 10.44$	$f = 3.01$	$f = 0.002$	-

Two remarks can be made:

1) p_1 and p_2 were initialized relatively far from the global minimum $\left(p_0^0 = \begin{bmatrix} 20 \\ 20 \end{bmatrix} \right)$. In this case, the fact that the adaptation gain is bounded penalizes the number of necessary iterations in the fixed low-gain algorithm.
2) The price to pay for converging faster is a stronger excitation of the inputs p, which leads to larger variations of the outputs, as can be seen in Figures 7 and 8.

Also, the proposed algorithm was compared to standard algorithms for solving nonlinear equations, using the same termination tolerance for f (0.005):

1) The standard Gauss-Newton approach that uses a Newton-Raphson update law and the Broyden formula to compute the Jacobian was tested. With the same initialization point, the *fsolve* procedure of the Matlab 6.5© software converged towards the solution of $z = 0$. However, when starting from $p_1^1 = \begin{bmatrix} -7 \\ 4 \end{bmatrix}$, the procedure stopped upon reaching a local minimum of z for which the Jacobian loses rank. This illustrates the possible failure of these algorithms.
2) The standard quasi-Newton approach that uses a mixed quadratic and cubic line search procedure and the BFGS formula for the Hessian update and the same initialization point was also tested. With these parameters, the quasi-Newton *fminunc* procedure of the Matlab 6.5© software converged towards a local minimum: after 43 evaluations of the function $f(p)$, the algorithm provided the values $p_{1,43} = -1.719$, $p_{2,43} = 3.923$, $z_1(p_{43}) = 0.226$, $z_2(p_{43}) = 0.537$ and $f(p_{43}) = 0.169$. This illustrates the local nature of numerical optimization algorithms.

VI. CONCLUSIONS

This work has demonstrated the effectiveness of run-to-run control applied to a class of nonlinear systems. To accelerate the speed of convergence of a fixed-gain run-to-run control algorithm, a new algorithm that uses tools from numerical optimization has been proposed. Despite the local nature of quasi-Newton methods, a proof of global

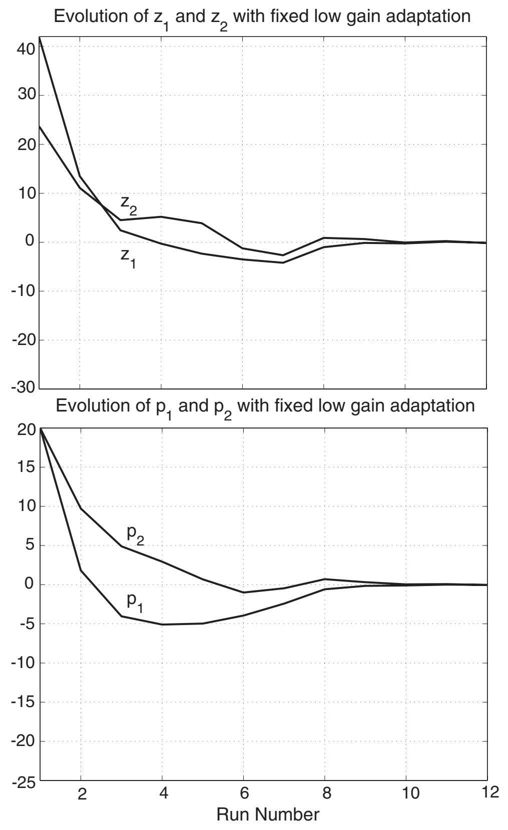

Fig. 7. Evolution of z and p with fixed low-gain adaptation

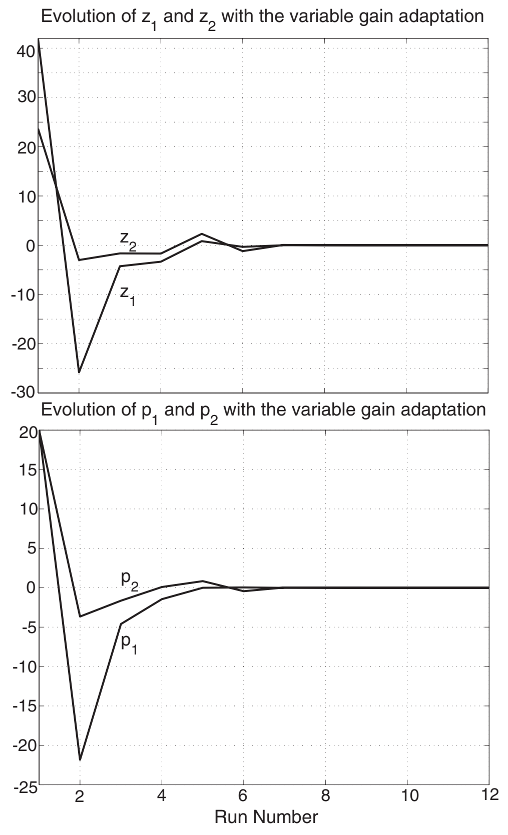

Fig. 8. Evolution of z and p using the proposed algorithm

convergence was obtained for a class of nonlinear systems. Significant improvement of the speed of convergence was observed and illustrated through the run-to-run control of a simple static example.

An interesting alternative would be to incorporate standard line search in the algorithm. If acceptable reduction of the Lyapunov function is not obtained with the variable-gain update, another point is sought in the same direction but with a smaller step size. The switch to fixed low-gain adaptation is done only when the latter fails. This way, the low-gain adaptation, which penalizes the rate of convergence, is performed only when absolutely necessary. Also, future research could extend the convergence analysis to other classes of nonlinear systems.

References

[1] W. Campbell, S. Firth, A. Toprac, and T. Edgar, "A comparison of run-to-run algorithms," in *American Control Conference*, Anchorage, 2002, pp. 4212–4217.

[2] E. Castillo and A. Hurwitz, "Run-to-run process control: Literature review and extensions," *Journal of Quality Technology*, vol. 29, pp. 184–196, 1997.

[3] E. Zafiriou, R. Adomaitis, and G. Gattu, "Approach to run-to-run control for rapid thermal processing," in *American Control Conference*, Seattle, 1995, pp. 1286–1288.

[4] S. Arimoto and T. Naniwa, "Learnability and adaptability from the viewpoint of passivity analysis," *Intelligent Automation and Soft Computing*, vol. 8, no. 2, pp. 71–94, 2002.

[5] A. Tayebi and M. Zaremba, "Robust iterative learning control design is straightforward for uncertain lti systems satisfying the robust performance condition," *IEEE Trans. Automat. Contr.*, vol. 48, no. 1, pp. 101–106, 2003.

[6] B. Srinivasan, C. J. Primus, D. Bonvin, and N. L. Ricker, "Run-to-run optimization via generalized constraint control," *Control Eng. Practice*, vol. 9, pp. 911–919, 2001.

[7] F. Doyle III, B. Srinivasan, and D. Bonvin, "Run-to-run control strategy for diabetes management," in *IEEE Engineering in Medicine and Biology Conference*, Istanbul, 2001.

[8] G. Francois, B. Srinivasan, and D. Bonvin, "Run-to-run optimization of an emulsion polymerization reactor," in *IFAC*, Barcelona, Spain, 2002, pp. 1258–1263.

[9] T. Chu, L. Huang, and L. Wang, "Guaranteed absolute stability of a class of delay systems with local sector nonlinearities via piece-wise linear Lyapunov function," in *American Control Conference*, Arlington, 2001, pp. 4212–4217.

[10] M. Vidyasagar, *Nonlinear Systems Analysis, Second Edition*. Prentice Hall, New Jersey, 1993.

[11] G. Francois, B. Srinivasan, and D. Bonvin, "Convergence analysis of run-to-run control for a class of nonlinear systems," in *American Control Conference*, Denver, Colorado, 2003, pp. 3032–3037.

[12] E. Kreysig, *Advanced Engineering Mathematics, Sixth Edition*. John Wiley And Sons, Inc, New York, 1988.

[13] B. Srinivasan, S. Palanki, and D. Bonvin, "Dynamic optimization of batch processes: I. Characterization of the nominal solution," *Comp. Chem. Eng.*, vol. 27, pp. 1–26, 2003.

[14] G. Francois, "Measurement-based run-to-run optimization of batch processes: Application to industrial acrylamide copolymerization," PhD thesis 3128, Ecole Polytechnique Federale de Lausanne (EPFL), Lausanne, Switzerland, 2004.

[15] C. Broyden, "A class of methods for solving nonlinear simultaneous equations," *Mathematics of Computation*, vol. 19, pp. 577–593, 1965.

[16] A. Goldstein, "On steepest descent," *SIAM J. Control*, vol. 3, pp. 147–151, 1965.

[17] W. Press, S. Teulosky, W. Vetterling, and B. Flannery, *Numerical Recipes in C, The Art of Scientific Computing*. Cambridge University Press, Cambridge, 1988.

[18] R. Fletcher, *Practical Methods of Optimization*. John Wiley And Sons, Inc, New York, 1991.