Technical Note

Optimization of a fed-batch fermentation process control competition problem using the NEOS server

J Liang and **Y-Q Chen***
Center for Self-Organizing and Intelligent Systems, Department of Electrical and Computer Engineering, Utah State University, Logan, Utah, USA

Abstract: An optimal control solution to a fed-batch fermentation process, responding to a competition call, was developed using the NEOS (network enabled optimization solution) server (http://www-neos.mcs.anl.gov/neos/). Substantial improvement to the nominal performance achieved in the paper demonstrates the ability of the NEOS server and the asynchronous parallel pattern search algorithm.

Keywords: optimal control, fed-batch process, network enabled optimization

1 INTRODUCTION

The optimal control problem considered in this paper is from a competition raised by the Biotechnological Control Forum [**1**]. The goal was to optimize the performance of a fed-batch fermentation process, using any technique to model the batch process and to design the controller.

Several features make this optimization problem difficult. Firstly, the model of the plant is given as a black-box model (in scrambled C source code) that simulates a fed-batch fermentation process; i.e. no mathematical model is available. The users can only apply an input signal to the black box and obtain the corresponding output signal from the black box. Secondly, both the initial states of the process and the parameters of the model vary randomly from batch to batch. For the same input, the output for each batch would not be the same. The presence of randomness makes the normal gradient-based optimization methods very hard and inefficient to implement, if not impossible. Thirdly, this is a multiple-input multiple-output (MIMO) non-linear (according to its input–output properties) system.

Various methods have been attempted. In reference [**2**] a dynamic integrated system optimization and parameter estimation (DISOPE) technique was used to achieve optimal control. The results obtained improved the optimal value of the objective function by over 21 per cent compared with the response to the nominal input profiles provided as the benchmark for competitors. In reference [**3**], an identification technique involving a hybrid of mechanistic modelling and genetic programming techniques was used to identify the black-box plant model. The performance of the model as a 25-time-step-ahead predictor shows that the r.m.s. (root mean square) error between the actual and the predicted output is less than 5 per cent in the range of interest. In this paper, an optimal controller was designed using the NEOS (network enabled optimization solution) server [**4**–**6**] and asynchronous parallel pattern search (APPS) algorithm [**7**]. An improvement of 105 per cent compared with the nominal output has been achieved. If the improved batch period is considered, the improvement is about 292 per cent.

The remainder of the paper is organized as follows. In section 2 the fed-batch fermentation optimal control problem is given in detail. Section 3 introduces the NEOS server. Since there are many different solvers in the NEOS server, in section 4 the selection of the solver specific to the fed-batch optimization problem and the implementation of the controller design are described. In section 5 the results obtained are presented and analysed. Finally, section 6 concludes this paper.

2 FED-BATCH FERMENTATION OPTIMAL CONTROL PROBLEM

Most of this section, for the purpose of completeness, is from the README file [**1**] provided to the competitors with slight modifications.

The MS was received on 6 January 2003 and was accepted after revision for publication on 12 May 2003.
* *Corresponding author: CSOIS, Department of Electrical and Computer Engineering, Utah State University, 4160 Old Main Hill, Logan, UT 84322-4160, USA.*

The fed-batch fermentation process produces a metabolite as the product. Two substrates 1 and 2 are needed as the inputs. The operators know from the experience the following:

1. Nominal feed profiles (as given) will give a reasonable production. Feeds of substrates at too high or too low levels seem to reduce the production.
2. The feed rate of substrate 2 seem to be complementary to the feed rate of substrate 1. Only feeding of substrate 1 does not yield any production while only feeding of substrate 2 yields a small product.

The goal was to investigate the optimal feed pattern in order to improve the process productivity. To study the effect of these two feeds, other process environment variables, such as temperature and pH, are assumed to be kept constant at their optimum.

To make the exercise comparable with working on a practical plant, the model representing the plant will not be given to the investigators explicitly. The investigators will be given the following information:

(a) a black-box plant simulator model in scrambled source code or compiled code;
(b) a set of typical input–output data;
(c) some explicitly specified constraints;
(d) some control objectives.

2.1 The black-box model

The process to be modelled and controlled is supplied as a black box with a set of specified inputs. Apart from the basic differential equations describing the process, the following features are inside the black box:

(a) the initial state which varies randomly within a subspace for each batch;
(b) the parameters of the model which vary within specified bounds;
(c) a set of unmeasurable disturbances;
(d) a set of constraints.

The black-box model is described as shown in Fig. 1.

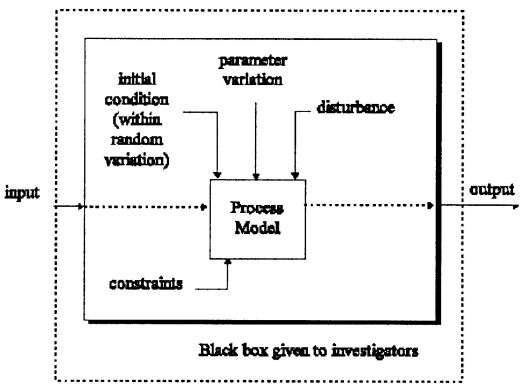

Fig. 1 Block diagram of the black-box model

This black box will be supplied to the investigators in a scrambled C source code. The users can apply input signals to the black box and obtain output signals from the black box. The users may use this information for modelling purposes as well as for controller design. Note that, as the initial state vector of the process varies randomly within a subspace for each batch, and the parameters of the model vary from batch to batch and also there are unmeasurable disturbances, the users are reminded that, for the same input signal time series, they would not obtain the same output time series for each batch.

The process has two inputs U_1 and U_2 in terms of substrate feed rate. There are five measurements which are as follows: X, biomass; S_1, concentration of substrate 1; S_2, concentration of substrate 2; P, product concentration; V, volume.

2.2 Constraints

There are two inputs in terms of substrate feed rate. There are limits on the maximum feed rate. The volume of the fermenter is also limited.

2.3 The requirements for the investigation

Investigators are required to carry out the following tasks:

(a) to analyse and model the black-box process;
(b) to design a controller to control the process for a certain period of time (one batch) and for a certain number of batches.

The performance criteria may be deliberately given in a loose form, but maximization of the productivity will be a prime aim. A suggested criterion is

$$J = \frac{\text{product concentration} \times \text{volume}}{\text{time length of a batch}} \tag{1}$$

The investigators may use any techniques to model the batch process, such as system identification techniques, neural networks or fuzzy systems. Also they can use any techniques to design the controller, such as adaptive control, robust control, neural controller or fuzzy controller. The performance of the control system designed by the investigators will be reflected in a number of simulation runs using the black-box model as the process.

2.4 Simulation results format

As the batch process starts with different initial values for each batch and because of the process parameter variations from batch to batch, investigators are requested to produce simulation results for a number of

mi00000503 07-07-03 10:25:46 Rev 14.05
The Charlesworth Group, Huddersfield 01484 517077

batches. Only in this way then can they adequately verify the performance of their approaches.

3 INTRODUCTION TO THE NEOS SERVER

The NEOS server is an environment for solving optimization problems remotely over the Internet, originally launched in 1994 by the Optimization Technology Center with the support from the US Department of Energy and Northwestern University [**4**–**6**]. Compared with the conventional application-based optimization methods, the NEOS server has the following advantages:

1. It is simple. Users need only to define the optimization problem in some popular formats, be it a programming language (Fotran, C or Matlab) or a modelling language (AMPL [**8**] or GAMS [**9**]). The details of the solution process are handled by the NEOS server and are transparent to users.
2. It is powerful. With more than 50 solvers, the NEOS server covers almost every branch of optimization problems. The solvers are based on the algorithms developed and tested by academic and commercial researchers for years and proved to be robust and efficient.
3. It is fast. The NEOS machines usually have powerful processors and large memories. Some algorithms make use of parallel computing techniques. Thus a low-end personal computer is sufficient for users to solve large optimization problems.
4. It is cheap. Users do not need to purchase any optimization software, which is usually very expensive. The NEOS server is free to anyone.
5. It is flexible. The algorithmic parameters of the solvers are adjustable for users, although the default is sufficient for most optimization problems. Users can communicate with the NEOS server via email, the Web, or the submission tools running on their own computers. Intermediate results can be viewed for time-consuming problems. Jobs submitted to the NEOS server can be killed at any time.

Currently, there are more than 50 solvers available, including linear and non-linear, unconstrained and constrained optimization, mixed integer programming, linear network programming, non-differentiable optimization, stochastic linear programming, complementary problems, semi-infinite optimization and global optimization.

Using the NEOS server is straightforward. A solver is first chosen based on the type of the problem and input file format available for both the solvers and the user. For example, since the file format that is used in this paper is in C, a server that accepts the user's routine in C rather than in AMPL or GAMS format is chosen. However, if the solvers only accepts AMPL, the C code has to be rewritten in AMPL language format. The optimization problem is then defined according to the required input format. The users submit the problem via the interface provided by the NEOS server. Finally, after the problem has been solved, the users will interpret the results sent back from the NEOS server.

The NEOS server offers four interfaces for submitting optimization problems. They are email, Web browser, the NEOS submission tool (NST) [**10**] and Kestrel [**11**].

Email is the present authors' favourite. Users need only edit one submission file following the template and submit it via email. The NEOS server then send back the solution information via email.

The Web browser is the most popular interface for problem submission. It is easy to understand and learn. However, multiple files need to be edited and submitted via the Web browser.

NST is capable of monitoring the compiling and executing on line. However, it requires users to download and install a piece of software, which is slightly more difficult than the previous two methods.

Kestrel is the most recent interface. It is a modelling language environment capable of choosing a solver, receiving the results, interpreting and manipulating the results directly. The disadvantage is that problems must be submitted in AMPL or GAMS format.

The use of the email interface is straightforward. Users first open their favourite email client (outlook express, Eudora, etc.), fill in the email address with `neos@mcs.anl.gov`, paste the content of the submission file to the body and submit the email. A sampled part of the final result sent back from the NEOS server is listed below:

```
00→ Search 28 New Min f=-1.29e+05 x=[ 1.03e+01 1.04e+01
5.00e+01 1.12e+01 1.19e+01 0.00e+00 0.00e+00 1.67e+01 1.94e+01
5.00e+01 5.00e+01 2.83e+01 3.04e+01 7.02e+00 3.31e+01 3.38e+01
5.00e+01 3.46e+01 3.47e+01 3.48e+01 3.49e+01 3.49e+01 3.50e+01
3.50e+00 5.00e+01 3.50e+00 5.00e+01 5.00e+01 2.85e+01 3.47e+00
5.00e+01 5.00e+01 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00
0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00
5.00e+01 0.00e+00 ] step=5.00e-01 tag=000_508 conv=[ 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]
```


mi00000503 07-07-03 10:25:46 Rev 14.05
The Charlesworth Group, Huddersfield 01484 517077

4 NEOS SOLVER SELECTION AND IMPLEMENTATION DETAILS

The fed-batch optimal control problem can be transcribed as a non-linear constrained mathematical programming problem. The plant to be controlled is given as scrambled C code. Thus, only solvers accepting C/C++ code as the input format can be selected. There are two candidate solvers: the limited memory variable metric method (BLMVM [**12**] and APPS [**7**].

BLMVM is an algorithm for solving non-linear optimization problems with lower and upper bounds that constrain the variables. The algorithm uses projected gradients to construct a BFGS matrix [**13**] and determines a step direction. BLMVM is suited to large optimization problems because of its limited storage requirement and its ability to solve a problem in parallel, making use of multiple processors. Since the algorithm needs gradient information of the objective function, which is not given, this has to be calculated. The automatic gradient computation via the automatic differentiation technique [**14**] is one of the most impressive features of the NEOS server. However, the randomness in the fed-batch fermentation process makes the gradient computation highly unreliable. The present authors tried to calculate the objective function 20 000 times to obtain the averaged value, aiming to remove the uncertainty; the BLMVM still failed. Finally, the APPS solver, which does not rely on the gradient information, was chosen.

APPS is a non-linear optimization algorithm based on the traditional parallel pattern search algorithm, which does not require the derivative information, with two important improvements. Firstly, the objective function evaluations on different processors are not assumed to be in approximately the same amount of time; thus, parallel computing can be made more efficient. Secondly, the APPS is able to handle the situation that either processes or processors fail during the computation, making the algorithm more robust. Although it converges more slowly than the gradient-based algorithm, because APPS takes advantage of parallel computing, the overall computational time cost can be reduced substantially. Since no gradient information is used, APPS has another advantage that it does not get stuck to the local maximum as easily as the gradient-based algorithm, as demonstrated in section 5. In the unconstrained case, APPS is even globally convergent [**15**].

To solve an optimization problem, firstly, it is necessary to define the objective function. An additional tuning knob c is added to the original performance function (1) as the objective function:

$$J' = J - c \quad \text{Std(batch output)}, \; c \geqslant 0 \tag{2}$$

where the constant c is used to penalize the standard deviation (Std) of the batch output such that the variance of the batch output can be kept from being too large.

Affected by noise, the objective function varies even for the same input profile in different batches. To overcome this problem, for the same input, the averaged value of the objective function is simply calculated on the basis of 5000 calculations of objective function. The number of calculations, 5000, was obtained by trial and error. Small numbers, such as 3000, make APPS fail because of too inaccurate evaluation of objective function.

Next the design variables are chosen. The given nominal model uses the sampling time of 1 s, which corresponds to 240 discrete control inputs. Since it is necessary to compute the objective function 5000 times to obtain the average, it was found to be impractical to use the sampling time of 1 s, given the limit on the number of iterations on the NEOS server. The sampling time was chosen to be 5 s and the results obtained are still satisfactory.

It is required that, when the volume of fermenter reaches 4000, the inputs should be set to zero. This requirement could not be specified on a format accepted by the APPS server, which accepts only hard constraints on the input. By observing the output plots corresponding to the nominal input, after the volume of fermenter reaches 4000, it can be seen that the objective function no longer improves. So our work-around is, during the computation of objective function, if the volume of fermenter reaches 4000, the maximum value of the objective function obtained so far is chosen as the final objective function value and the calculation is ended. The final results show that this work-around is successful.

5 RESULTS AND DISCUSSION

Comparisons between the nominal and the optimal profile are now given.

If the main concern is only the maximum averaged J (rate of output) and it does not matter how large the variance of J is ($c = 0$), comparisons are shown in Figs 2 and 3 for 20 simulation runs. It can be seen that not

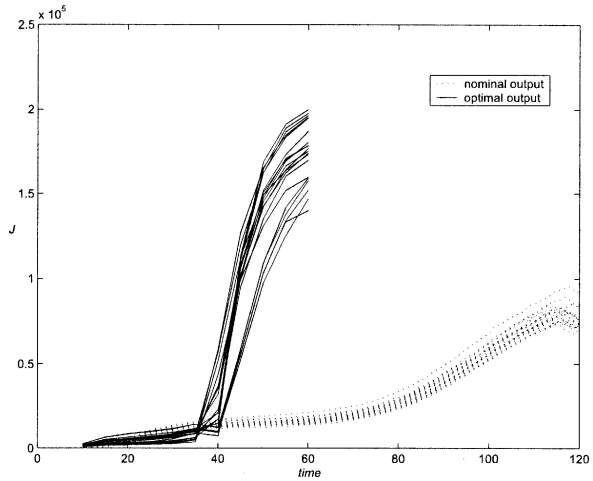

Fig. 2 Nominal output versus optimal output without variance optimization ($T = 120$)

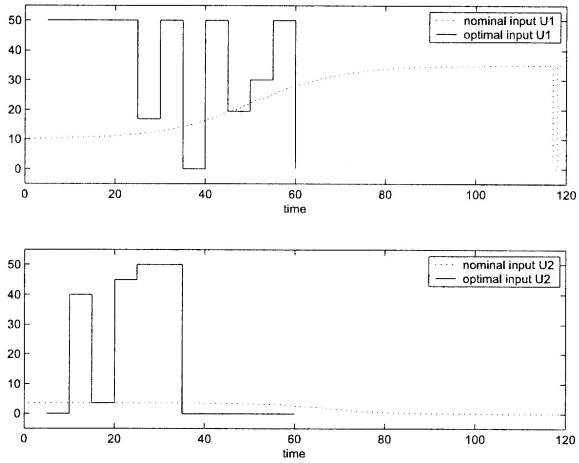

Fig. 3 Nominal input versus optimal input without variance optimization ($T = 120$)

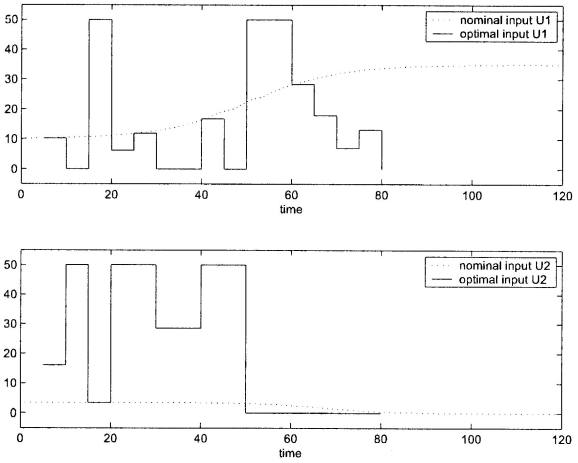

Fig. 5 Nominal input versus optimal output with variance optimization ($T = 120$)

only is the maximum average response improved by 105 per cent (from 8.3×10^4 to 1.7×10^5), but also the optimal batch period, which is the time corresponding to the maximal rate of output, is reduced from 115 to 60. It is known that a reduction in batch period is highly desirable. So, the overall productivity improvement is about 292 per cent.

The control of output variance is also very important in fed-batch fermentation process. Smaller output variance means better quality consistence. This is why an additional tuning knob c is added to the original performance function. For $c = 0$, the proportion of the standard deviation to the averaged J is 12 per cent, which is considered to be large. Therefore $c = 2$ is chosen, which makes penalization of the standard deviation account for 24 per cent of the value of objective function in the $c = 0$ case. The results shown in Figs 4 and 5 prove the effect of this additional tuning knob. Although the maximum averaged response is smaller (1.5×10^5), the standard deviation is greatly reduced from 2.0×10^4 to 7.2×10^3, accounting for only 6 per cent of the averaged response. The results also show one of the features of the APPS algorithm, namely that the local maximum does not easily make APPS stick.

The present authors studied the scrambled C code and removed carefully the random noise from the code. Thus, the fed-batch optimization became a much simpler non-linear constrained optimization problem without any randomness. This time the BLMVM solver, which is a gradient-based solver, can be chosen, with a sampling time of 5 s. The obtained optimal input is fed into the original black box (with random noise); Fig. 6 shows the result. Clearly, the maximal averaged J is much smaller than in Fig. 2. Figure 7 is the corresponding inputs. The only explanation is that the optimal solution achieved from BLMVM was a local maximum, although the optimal solution achieved from APPS may not be

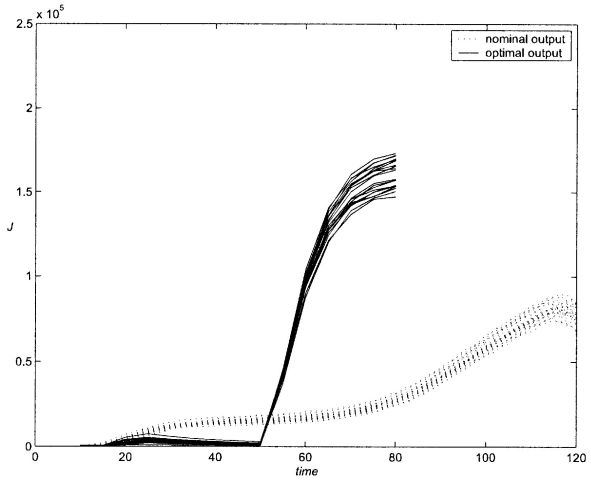

Fig. 4 Nominal output versus optimal output with variance optimization ($T = 120$)

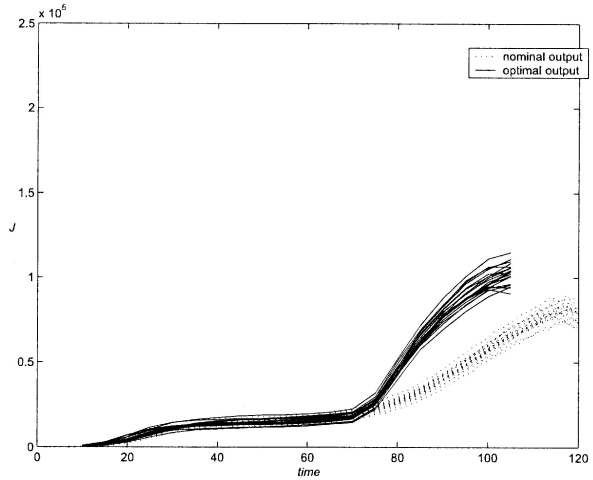

Fig. 6 Nominal output versus optimal output obtained from BLMVM ($T = 120$)

mi00000503 07-07-03 10:25:46 Rev 14.05
The Charlesworth Group, Huddersfield 01484 517077

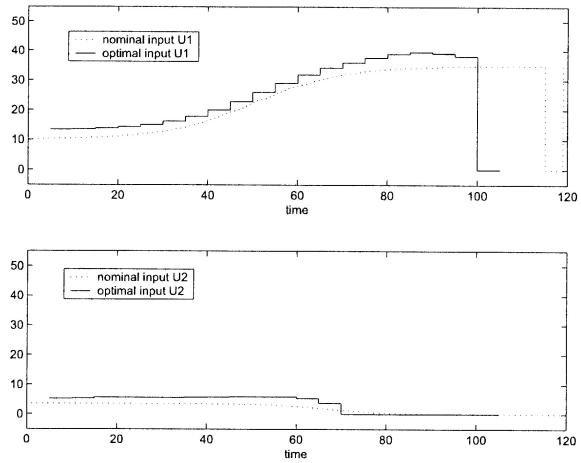

Fig. 7 Nominal input versus optimal input obtained from BLMVM ($T = 120$)

global. This is an interesting example to show the advantages of APPS.

Finally, to verify that using a sampling time of 5 s instead of the nominal 1 s does not generate too large an accumulated error which might contribute to the final optimal response, the input in Fig. 3 was resampled using a sampling time of 1 s and this input was fed to the plant. Figure 8 shows the output, which is very close to that in Fig. 2.

The comparisons are summarized in Table 1.

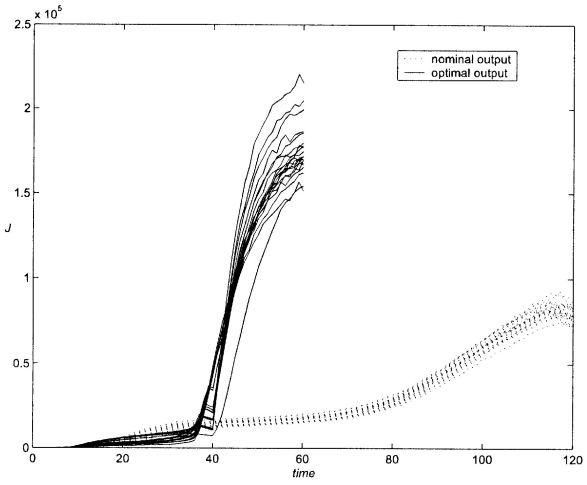

Fig. 8 Nominal output versus output from the resampled input in Fig. 3

Table 1 Comparison between nominal control and optimal control

Input	Maximum J	Optimal batch time	Standard deviation
Nominal	8.3×10^4	115	4.6×10^3
Optimal ($c = 2$)	1.5×10^5	80	7.2×10^3
Optimal ($c = 0$)	1.7×10^5	60	2.0×10^4
Optimal ($c = 0$) resampled	1.7×10^5	57	1.8×10^4

6 CONCLUDING REMARKS

An optimal controller was designed by making use of the NEOS server for the fed-batch fermentation process, which produced substantial improvement in the nominal control. Compared with gradient-based algorithms, the APPS algorithm has been proved to be powerful for stochastic optimization problems, in which case the gradient calculation is not reliable. The NEOS server is powerful, easy to use and freely available. The NEOS server could be the first choice for tough optimization problems, especially in the Internet age.

REFERENCES

1 **Leigh, R.** Fed-batch fermentation process modeling and control competition—editorial. *Trans. Inst. Measmt Control*, 1998, **20**(1), 1–3.
2 **Becerra, V. M.** and **Roberts, P. D.** Application of a novel optimal control algorithm to a benchmark fed-batch fermentation process. *Trans. Inst. Measmt Control*, 1998, **20**(1), 11–18.
3 **McKay, B., Sanderson, C. S., Willis, M. J., Barford, J. P.** and **Barton, G. W.** Evolving a hybrid model of a fed-batch fermentation process. *Trans. Inst. Measmt Control*, 1998, **20**(1), 4–10.
4 **Ferris, M. C., Mesnier, M. P.** and **Moré, J. J.** NEOS and condor: solving optimization problems over the Internet. *ACM Trans. Math. Software*, 2000, 26(1), 1–18.
5 **Dolan, E. D., Fourer, R., Moré, J. J.** and **Munson, T. S.** The NEOS server for optimization: version 4 and beyond. Preprint ANL/MCS-P947-0202, Argonne National Laboratory, 2002.
6 **Dolan, E. D.** NEOS server 4.0 administrative guide. Technical Memorandum ANL/MCS-TM-250, Argonne National Laboratory, 2002.
7 **Hough, P. D., Kolda, T. G.** and **Torczon, V. J.** Asynchronous parallel pattern search for nonlinear optimization. *SIAM J. Scient. Computing*, 2001, **23**(1), 134–156.
8 **Fourer, R., Gay, D. M.** and **Kernighan, B. W.** *AMPL: A Modeling Language for Mathematical Programming*, 2002 (Duxbury Press, Belmont, California).
9 **Brooke, A., Kendrick, D., Meeraus, A.** and **Raman, R.** GAMS: a user's guide. http//www.gams.com, 2002.
10 **Czyzyk, J., Mesnier, M.** and **Moré, J. J.** The NEOS server. *IEEE J. Comput. Sci. Engng*, 1998, **5**(3), 68–75.
11 **Dolan, E. D.** and **Munson, T. S.** The kestrel interface to the NEOS server. Technical Memorandum ANL/MCS-TM-248, Argonne National Laboratory, 2001.
12 **Benson, S. J.** and **Moré, J. J.** A limited memory variable metric method for bound constrained minimization. Preprint ANL/MCS-P909-0901, Argonne National Laboratory, 2001.
13 **Gill, P. E., Murray, W.** and **Wright, M. H.** *Practical Optimization*, 1981 (Academic Press, New York).
14 **Griewank, A., Juedes, D.** and **Utke, J.** Algorithm 755: ADOL-C: a package for the automatic differentiation of algorithms written in C/C++. *ACM Trans. Math. Software*, 1996, **22**(2), 131–167.
15 **Torczon, V.** On the convergence of pattern search algorithms. *SIAM J. Optimization*, 1997, (7), 1–25.