Incremental nonlinear dynamic data reconciliation

Ralf Der[1], Olaf Lummer[2], Thomas List[3]

[1] *Universtität Leipzig, Institut für Informatik*

[2] *smartec GmbH, Leipzig*

[3] *Wacker Chemie, Fachstelle Simulation und Prozessüberwachung*

March 31, 1998

Abstract

Measurement noise reduction and parameter estimation is a topic of central importance in plant control. The complexity of real world plants and the working conditions in practice require robust real–time algorithms which are easy to implement, simple to use and economic in computer ressources. The state of the art is given by the novel approach of Liebman et al. called the NDDR (nonlinear dynamic data reconciliation) which is based on nonlinear dynamic programming. We present in the present paper a new algorithm based more traditionally on gradient descent methods supplemented with a self control of the parameters of the algorithm. It uses an iterative method for the rectification and correction of state variables and system parameters, what makes it a true on–line algorithm. Despite its simplicity, the perfomance of the new algorithm proved superior to that of the NDDR in the applications considered so far.

1 Introduction

Measurement noise reduction and parameter estimation is a topic of central importance in plant control. Many methods have been developed for solving this task. Recently a novel approach has been presented by Liebman et al [1]. Their NDDR (nonlinear dynamic data reconciliation) is based on dynamic programming.

The present paper describes an approach which has been designed for a real world application. The challenge was to develop a procedure for the control of a large chemical plant producing acetyl acetone (ACAC). The process taking place in the plant can be simulated by a compartment model including the chemical reactions, the heat transfer through the walls of the reaction tube and the flow inside the tube. The complexity of the plant and the working conditions in practice required a robust algorithm which is easy to implement and simple to use. Moreover the algorithm should be a true real–time one and was not to use too much computer ressources.

The NDDR algorithm by Liebman et al. was considered too complex so that we developed a new algorithm based more traditionally on gradient descent methods. Despite its simplicity, the perfomance of the new algorithm proved superior to that of the NDDR.

2 The problem

Let us consider by way of example a chemical plant with inputs $\mathbf{I}$, internal state $\mathbf{u}$ and outputs $\mathbf{o}$. The plant is modelled by the set of differential equations

$$\dot{u}_k = f_k\left(\mathbf{u}, \mathbf{I}, t\right) \tag{1}$$

where all variables depend on time, the ouput being a function of the inputs and the internal state of the plant

$$o_k = g_k\left(\mathbf{I}, \mathbf{u}, t\right) \tag{2}$$

and $\mathbf{u} = (u_1, \ldots, u_n)$ and so on. Typical problems arising in practice are

1. Not all state and/or onput variables are known at all times (the problem of incomplete information)
2. Measurement are corrupted by errors (the noise problem)
3. The process model given by the dynamical system eq. (1) is not correct. In most cases the parameters of the model are known only approximately (the system–identification or parameter–estimation problem)

As is well known these problems can be solved by using redundancies in the measurements of the process. The redundance must be sufficient in order to provide the necessary information to overcome the noisyness and partial lack of information in the problem.

In order to develop the method we need a mathematical formulation of the above problems. In particular point 2 is modeled as

$$x_k = x_k^{true} + \eta_k \tag{3}$$

where the measured value x_k deviates from x_k^{true} by the random number η_k. Usually white Gaussian noise is assumed.

3 The algorithm

Let us consider for the sake of simplicity a one dimensional system with state variable u^{true} governed by the differential equation

$$\dot{u}^{true} = f\left(u^{true}\right) \tag{4}$$

measured values u being corrupted by noise according to eq. (3). Measurements are taking place at discrete times

$$t_n = t_0 + n\Delta t \qquad n = 0, 1, \ldots$$

Δt being the time interval between measurements. We use a moving time window $W_t = [t - T, t)$ of length T

$$t' \in W_t \quad \mathit{if} \qquad t - T \leq t' < t \tag{5}$$

3.1 Error function

The basic idea of the algorithm is straightforward. We introduce estimates $\hat{u}_n$ of the true values u_n^{true} of the observable u. The estimates are initialized with the measured values and incrementally improved using the differential equation. By means of the latter we calculate the forecasts

$$\hat{u}_{n|n-1} = \begin{cases} \mathit{Solution\ at\ time}\ t_n \quad \mathit{of}\ \dot{u} = f\left(u\right) \\ \\ \mathit{starting\ at\ time}\ t_{n-1}\ \mathit{with}\ u = \hat{u}_{n-1} \end{cases} \tag{6}$$

Then we define the error function

$$E = \sum_n E_n \tag{7}$$

where

$$E_n = \frac{1}{2}\left(\hat{u}_{n|n-1} - \hat{u}_n\right)^2 \tag{8}$$

This error E is equal to zero whenever the $\hat{u}_n$ are points on a global solution $u\left(t'\right), \quad t' \in W_{t'}$ of the differential equation 4.

3.2 Gradient descent

By means of gradient descent

$$\Delta\hat{u}_n = -\epsilon\frac{\partial}{\partial\hat{u}_n}E \tag{9}$$

on this error function we may generate a solution of the differential equation 4. The error function is so to say degenerate with respect to the solutions of the differential equation since $E = 0$ for each one of the solutions. However the solution found in initializing the $\hat{u}_n$ with the measured values u_n is not necessarily the one which also does minimize the mean square deviation between u_n and the final values of $\hat{u}_n$. Correspondingly we add an extra penalty term so that the error now is

$$E = \frac{1}{2} \sum_n \left(\hat{u}_{n|n-1} - \hat{u}_n\right)^2 + \lambda \sum_n \left(\hat{u}_n - u_n\right)^2 \tag{10}$$

By gradient descent the first term moves the $\hat{u}_n$ so that the forecasts from time t_{n-1} agree with the estimates $\hat{u}_n$ themself so that a global soution of the differential equation is produced. The second term tries to keep the estimates to the measured values as close as possible. This will uphold gaps in the solution of the differential equation.

In principle one has to "cool" down the parameter λ gradually during the gradient descent procedure. Then in the limit $\lambda = 0$ one obtains the global solution which is the best one in the sense that its mean square deviation from the measured values is minimal. Reliable scenarios for the cooling are available in the theory of stochastic approximation. However we found in all our applications that the second term could be dropped ($\lambda = 0$) altogether without loss of quality of the solution. For the case of unbiased Gaussian noise and linear systems this can be understood in simple terms by the dynamics of the $\hat{u}_n$ generated by gradient descent.

Noting that the $\hat{u}_{n|n-1}$ are functions of the starting value $\hat{u}_{n-1}$

$$\hat{u}_{n|n-1} = \phi_n \left(\hat{u}_{n-1}\right) \tag{11}$$

equation (9) reads in more detail

$$\Delta \hat{u}_n = -\epsilon \left[\left(\hat{u}_{n+1|n} - \hat{u}_{n+1}\right) \frac{\partial}{\partial \hat{u}_n} \hat{u}_{n+1|n} - \left(\hat{u}_{n|n-1} - \hat{u}_n\right) \right] - \epsilon \lambda \left(\hat{u}_n - u_n\right) \tag{12}$$

The forecasts $\hat{u}_{n+1|n}$ and $\hat{u}_{n|n-1}$ can be obtained explicitly if the time interval Δt is not too large. However one must be careful to chose the correct approximation procedure for these forecasts. We have observed that a simple Taylor expansion with one or two terms is in general not sufficient. Instead we used a fourth order Runge–Kutta approximation which proved much superior to the Taylor expansion of even higher order.

The derivation $\frac{\partial}{\partial \hat{u}_n} \hat{u}_{n+1|n}$ is model specific. However one may use a simple approximation to avoid doing this calculation manually. According to eq. (11) one can approximate

$$\frac{\partial}{\partial \hat{u}_n} \hat{u}_{n|n-1} = \frac{\partial}{\partial \hat{u}_n} \phi_n \left(\hat{u}_{n-1}\right) \approx \frac{\phi_n \left(\hat{u}_{n-1} + h\right) - \phi_n \left(\hat{u}_{n-1}\right)}{h} \tag{13}$$

This approximation is very fast and is of sufficient precision.

3.3 The general case

In general both the internal state and the inputs are multidimensional, the dynamics of the system being governed by the set of differential equations

(1). The error function for this case is

$$E = \frac{1}{2}\sum_{n,\alpha}\left(\phi_{n,\alpha} - \hat{u}_{n,\alpha}\right)^2 + \lambda\sum_{n,\alpha}\left(\hat{u}_{n,\alpha} - u_{n,\alpha}\right)^2 \tag{14}$$

where the forecast

$$\phi_{n,\alpha} = \phi_{n,\alpha}\left(\hat{u}_{n-1,1},\ldots,\hat{u}_{n-1,M},\left\{\mathbf{x}\left(t'\right) \mid t' \in W_{t'}\right\}\right) \tag{15}$$

is just $u_\alpha(t_n)$ obtained from the solution of eq. 1 starting at t_{n-1} with $\{\hat{u}_{n-1,\beta} \mid \beta = 1, 2, ..., M\}$ as initial conditions. Note that the forecast is a functional of the inputs if the latter depend on time, i. e. the forecast depends on the set of values $\{\mathbf{x}(t') \mid t' \in W_{t'}\}$ of the input variables inside the time window.

The update of the estimates is obtained by gradient descent as

$$\begin{aligned} \Delta\hat{u}_{n,\alpha} &= -\epsilon_u \frac{\partial E}{\partial \hat{u}_{n,\alpha}} \qquad (16) \\ &= -\epsilon_u \sum_\beta \left(\phi_{n+1,\beta} - \hat{u}_{n+1,\beta}\right)^2 \frac{\partial \phi_{n+1,\beta}}{\partial \hat{u}_{n,\alpha}} \\ &\quad +\epsilon_u\left(\phi_{n,\alpha} - \hat{u}_{n,\alpha}\right) - \epsilon_u\lambda\left(\hat{u}_{n,\alpha} - u_{n,\alpha}\right) \end{aligned}$$

3.4 The on–line algorithm

So far we have developed the algorithm for the case of a fixed time window, the summation in eq. (7) running over the points inside the window. For on-line applications one has to move the window by one time step $W_t \Rightarrow W_{t+1}$ to account for the new measured value u_{n+1}. Between the time steps for all n inside the time window, the update (12) is repeated as often as possible. We call this one epoch of the algorithm. For each set of updates, the learning parameter ϵ is optimized. This increases both speed and quality of results.

Pseudocode for the multi-dimensional case:

1. Start an epoch:
   At time t_m, move the time window one step forward. Initialize the new estimates $\hat{u}_{mi}$ with the measured values u_{mi}.
2. Updating the estimates:
   - `repeat`
     Compute the update term p_{ni} for each $\hat{u}_{ni}$ in the time window (see (16)):

     $$p_{ni} := \frac{\partial E}{\partial u_{ni}} \tag{17}$$

     Store the current value of the energy function E.

Initialize the learning parameter as $\epsilon = 0.9$.
Add the updates, multiplied by the learning parameter to the estimates:

$$\Delta \hat{u}_{ni} = -\epsilon p_{ni} \tag{18}$$

- `repeat`
  Remove the last updates.
  Decrease the learning parameter as $\epsilon := \epsilon/2$
  Apply the update, store the value of the energy function E.
- `until` no further improvement can be achieved or new measurement data arrive.

* `until` no further improvement can be achieved or new measurement data arrive.

3. `goto` 1

Our practical experiences gathered with the algorithm justify the following notes:

- The inner loop can easily be formulated in a more efficient manner.
- The initial learning parameter does not necessarily have to be 0.9, it is only required to be a positive value less than 1.0 (but smaller initial values lead to decreased performance).
- The length T of the time window is dictated by the compromise between the need of redundancy and the computational costs. Adaptive regulation of this parameter is possible.

4 Parameter adaptation and drift corrections

In practice the paramaters of the model often are not known exactly. We use gradient descent on the above error function as well in order to find the exact parameters. We assume for this purpose that (i) the average deviation of the measured values u_n is zero and that (ii) the average curvature of the solutions of the differential equations (1) is a smooth function of parameter variations.[1] Then the mean square deviation of the errors will be minimal for the optimum solution (i. e. the one corresponding to the exact parameters). Consequently gradient descent on the error function (14) will produce the correct parameters after cooling λ down to zero.

[1] In pathological cases and for a fixed finite set of measured values $\{u_n\}$ there might be parameter values producing solutions which run through all the u_n. This might happen above all in the case of overfitting, i. e. if the number of paramters is much higher than the complexity of the system requires (number of u_n). We exclude these cases from the discussion.

4.1 Parameter adaptation by gradient descent

Assuming the differential equations are parametrized by a set $\mathbf{a} = (a_1, \ldots, a_P)$ of parameters, i. e. eq. (1) becomes

$$\dot{u}_k = f_k(\mathbf{u}, \mathbf{I}, \mathbf{a}, t) \tag{19}$$

and the forecasts are now written as

$$\phi_{n,\alpha} = \phi_{n,\alpha}\left(\hat{u}_{n-1,1}, \ldots, \hat{u}_{n-1,M}, a_1, \ldots, a_P, \left\{\mathbf{x}\left(t'\right) \mid t' \in W_{t'}\right\}\right)$$

the update rule for the parameters reading

$$\begin{aligned} \Delta a_\mu &= -\epsilon_a \frac{\partial E}{\partial a_\mu} \qquad (20) \\ &= -\epsilon_a \sum_\rho \left(\phi_{n+1,\rho} - \hat{u}_{n+1,\rho}\right)^2 \frac{\partial \phi_{n+1,\rho}}{\partial a_\mu} \end{aligned}$$

In the pseudocode for the algorithm given in Section 3.4 we have to insert after eq. (17) the step

...

Calculate the update

$$q_\mu := \frac{\partial E}{\partial a_\mu} \tag{21}$$

...

and after eq. (18) insert the step

...

Update

$$\Delta a_\mu = -\varepsilon_a q_\mu \tag{22}$$

...

Note that the measuring values do not occurr in the update rule at all. For $\lambda = 0$ the dynamics (16) would drive the $\hat{u}_n$ to some solution of the differential equation (19). If this is converged, the parameter dynamics (20) halts as well since $\phi_{n+1,\rho} = \hat{u}_{n+1,\rho}$. Before convergence is reached the parameters and the estimates $\hat{u}_n$ change simultaneously. Hence the algorithm converges to some exact solution of the model equations (19) with certain values of the paramters a_μ. The values a_μ and the solution reached depends on the initial values of the $\hat{u}_n$ and the relative values of the adaptation rates ϵ_u and ϵ_a.

This indeterminacy is removed by the λ term in the update rule (16). This one drives the $\hat{u}_n$ towards the measuring values u_n and thus prevents the differences $\phi_{n+1,\rho} - \hat{u}_{n+1,\rho}$ and hence the updates in the gradient rule (20) from becoming zero. Under a slow cooling of λ the combined gradient dynamics will drive both the estimates and the parameters to the correct values.

These considerations are appropriate for a fixed window with a fixed set of measuring values. Most interestingly we observed again, that in the

moving window scenario the parameter adaptation is feasible even with $\lambda = 0$. This can be understood by the following argument. Each new measuring value triggers the window to be shifted by one step starting a new epoch of gradient descent inside the window. This introduces the information over the measuring values into the update rule (20) since the new estimate $\hat{u}$ is initialized with the fresh measuring value. In the average over many such events the difference between the new measuring value and its forecast is minimal if the latter is evaluated with the best forecast model, i. e. using the exact parameters in the model equations (19). Consequently in the average over many window shifts the gradient dynamics (20) is observed to converge to the exact parameter values.

4.2 Systematic measurement errors (drifts)

We consider drift corrections as additional parameters so that the above update rule (20) can be used for the elimination of drifts. For instance assuming the measurement device of state variable U_s is expected to need a drift correction we replace in the r. h. s. of (19)

$$u_s \rightarrow u_s + b$$

where b is a new parameter which is incorporated into the parameter set A introduced above.

In the same way drifts in the input variables can be corrected.

5 Examples

Before applying our method to the complex task of the kethen reactor we have considered a few toy examples.

5.1 Logistic differential equation

In the first instance we studied data reconciliation and parameter estimation for the one–dimensional differential equation

$$\dot{u} = au - u^2 \tag{23}$$

where the parameter a was chosen $a = 1$ for most of the investigations. In this case the right hand side is the logistic function. We tested data reconciliation both with and without parameter adaptation. The results are given in Fig. 1.

The influence of the width of the time window on the performance of the algorithm is studied in 2.

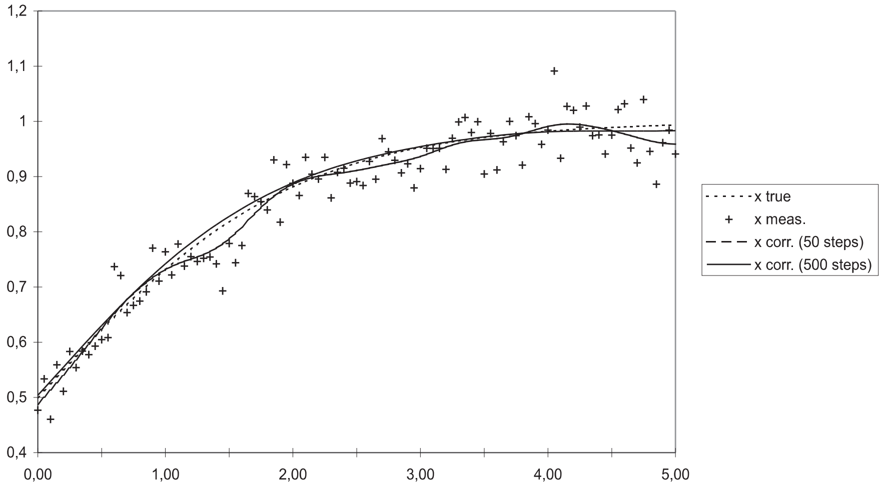

Figure 1: Iterative improvement of estimates with a simple form of the algorithm. (no time window, $\epsilon = 0.2$ fix, after 50 and 500 updates)

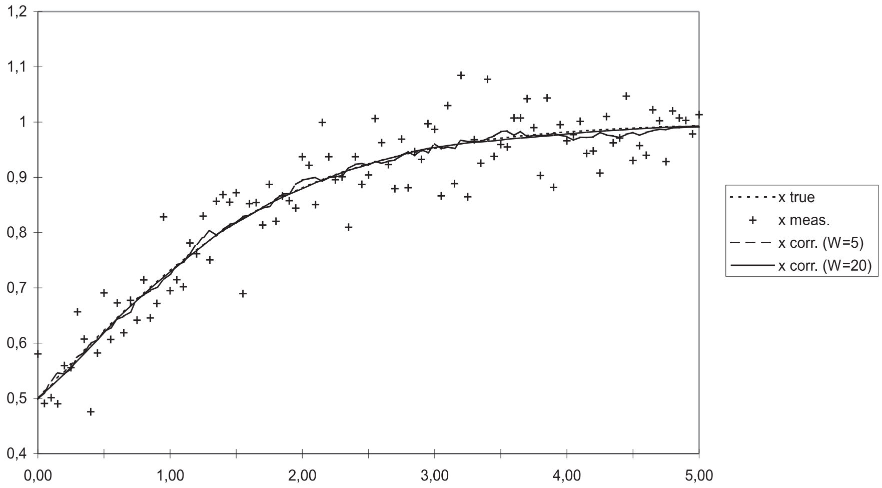

Figure 2: Influence of window width T on accuracy using $\epsilon = 0.2$ fixed, 100 update steps per time window of width $T = 5$ and $T = 20$, respectively. In practical applications, real-time requirements and limited CPU time restrict the size of the time window.

5.2 Predator–Prey System

The two–dimensional case was tested in terms of the Lotka–Volterra differential equations modeling a predator–prey system

$$\begin{aligned} \dot{u}_1 &= a_1\, u_1 - a_2\, u_1 u_2 \\ \dot{u}_2 &= a_3\, u_1 + a_4\, u_1 u_2 \end{aligned} \tag{24}$$

The dynamics is characterized by a nonlinear oscillatory behaviour. Fig. 2 clearly shows this behovior and also demonstrates nicely that our algorithm is capble of both parameter estimation and data reconciliation for this system.

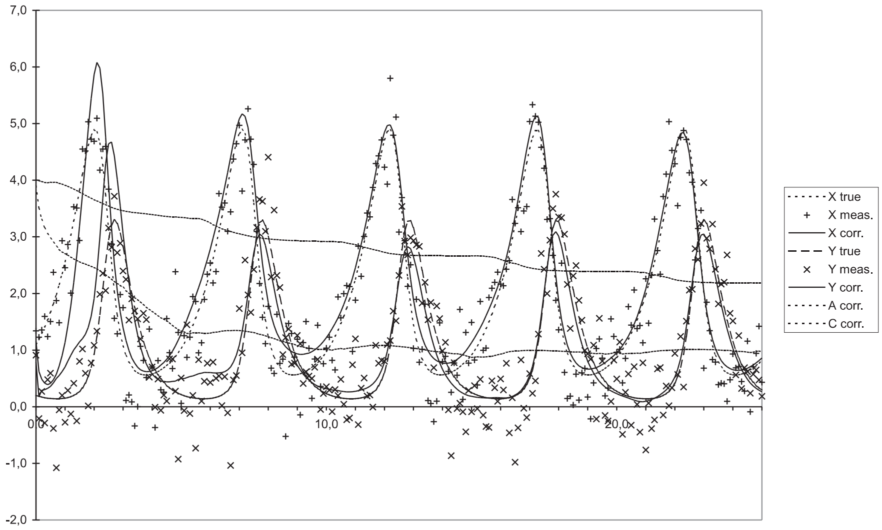

Figure 3: State estimation and parameter adaption in the preadator–prey–system. The initial estimates $a_1 = a_3 = 4$ are seen to converge towards the correct values $a_1 = 1$ and $a_3 = 2$. Convergence of the parameters is step like and is most pronounced if dynamics is high, i. e. during rapid changes in the state of the system.

5.3 A continuous–flow stirred tank reactor

In the standard model of the ontinuous–flow stirred tank reactor (CSTR) the differential equations of the system are

$$\begin{aligned} \dot{A} &= \frac{q}{V}\,(A_0 - A) - \alpha_d k A \\ \dot{T} &= \frac{q}{V}\,(T_0 - T) - \alpha_d \frac{\Delta H_r A_r}{p C_p T_r} k A - \frac{U A_r}{p C_p V}\,(T - T_c) \end{aligned} \tag{25}$$

where the reaction rate

$$k = K_0 \exp \frac{-E_A}{TT_r}$$

The parameters used in our simulations have been the ones given by Liebman et al. The following table gives a short comparison between the

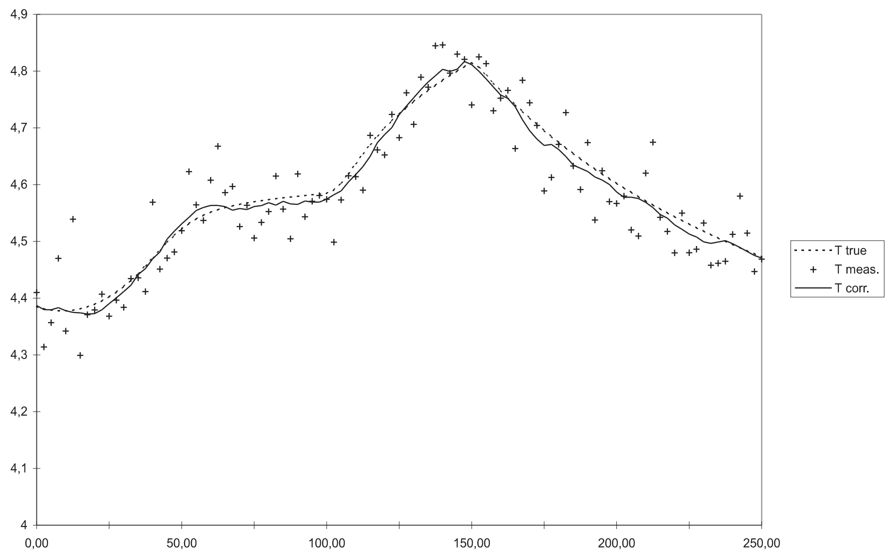

Figure 4: Bias correction with measurement data from the CSTR simulation: the estimate for the bias of A fluctuates near the correct value of 0.1.

results of our tests with the CSTR simulation and those given by Liebman et al. Unfortunately, no results for tests with bias estimation were available for NDDR. Note that in our simulations the values of the input variables A_0 and T_0 were not adapted by the algorithm, instead they were smoothed by a moving average.

	NDDR			our algorithm					
Bias of A	0			0			0.1		
T	5			5			5		
CPU time	1.14			0.085			0.085		
	SD	SD	SD	SD	SD	SD	SD	SD	SD
	meas.	est.	reduct.	meas.	est.	reduct.	meas.	est.	reduct.
A	0.0547	0.0192	64.9%	0.0443	0.0138	68.9%	0.1033	0.0197	80.9%
T	0.0545	0.0160	70.6%	0.0540	0.0148	72.6%	0.0495	0.0117	76.4%
A_0	0.0496	0.4550	-817.3%	0.0495	0.2924	-490.7%	0.0501	0.2825	-463.9%
T_0	0.0537	0.0252	53.1%	0.0551	0.0271	50.8%	0.0466	0.0244	47.6%

Table 1: Comparison between the NDDR algorithm run on a VAXStation 3200 and our new algorithm run on a 486/80 PC for the CSTR model.

6 Concluding remarks

The above examples clearly demonstrated the potentials of the algorithm. Presently we are working with a first implementation of the new algorithm for the full compartment model of the ACAC reactor. Our preliminary results demonstrate the good performance of the algorithm in this complex domain.

References

[1] M. J. Liebmann, T. F. Edgar, and L. S. Ladson. Efficient data reconciliation and estimation for dynamic processes using nonlinear dynamic programming techniques. *Computers in Chemical Engineering*, 16(10/11):961 – 985, 1992.