8th IFAC Symposium on Advanced Control of Chemical Processes
The International Federation of Automatic Control
Singapore, July 10-13, 2012

Advanced-multi-step Nonlinear Model Predictive Control

Xue Yang * **Lorenz T. Biegler** **

* *Chemical Engineering Department, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: xuey@andrew.cmu.edu)*
** *Chemical Engineering Department, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: lb01@andrew.cmu.edu)*

Abstract: Nonlinear Model Predictive Control (NMPC) has gained wide attention through the application of dynamic optimization. However, this approach is susceptible to computational delay, especially if the optimization problem cannot be solved within one sampling time. In this paper we propose an *advanced-multi-step* NMPC (amsNMPC) method based on nonlinear programming (NLP) and NLP sensitivity. This method includes two approaches: the serial approach and the parallel approach. These two approaches solve the background nonlinear programming (NLP) problem at different frequencies and update manipulated variables within each sampling time using NLP sensitivity. We present a continuous stirred tank reactor (CSTR) example to demonstrate the performance of amsNMPC and analyze the results.

Keywords: model-based control; optimization; predictive control; nonlinear programming; sensitivity.

1. INTRODUCTION

Model Predictive Control (MPC) is a widely used feedback control strategy. It has the advantage of handling variable bounds and dealing directly with multi-input-multi-output systems. Often used to track setpoints, it solves an optimization problem derived from dynamics of a real system. If the system is nonlinear, then a nonlinear optimization problem must be solved, leading to Nonlinear Model Predictive Control (NMPC) considered here.

The input to the NMPC controller is the current plant state, and its output is the manipulated variable to be injected into the plant, and obtained from the solution of an NLP problem. However, solution for the manipulated variable begins only after current state is available. Moreover, if the NLP solution requires non-negligible CPU time, a computational delay occurs between obtaining the state and injecting the control. This delay could lead to deterioration of controller performance and system stability. To prevent computational delay, an *advanced-step NMPC (asNMPC)* method was proposed in Zavala and Biegler (2009). The idea is to use the prediction of the future state to solve the NLP problem within the previous sampling time. Once the real state is obtained (or estimated), NLP sensitivity is used to update the manipulated variable online. Since the update only requires a single backsolve, it requires negligible computation time. Hence, the manipulated variable is available right after the real state is obtained and computational delay is avoided. Nominal stability and robust stability of NMPC and asNMPC have also been proved.

However, asNMPC requires the NLP problem to be solved within one sampling time. If NLP solution takes longer, neither asNMPC nor NMPC can be applied. On the other hand, there are several fast MPC or NMPC methods that deal with this case. For MPC, which uses quadratic programming, a partial enumeration was proposed in Pannocchia et al. (2007), where the most frequent active sets are stored in a list. For every given state, the list is searched and if the corresponding active set is in the list, the optimal solution is determined directly. Otherwise, a suboptimal MPC problem is solved to get a quick solution, and the original problem is also solved to get a new active set, which updates the list. For NMPC a neighboring-extremal updates (NEU) was proposed in Würth et al. (2009). The NLP problem is solved only once over an infinite horizon and then a quadratic programming (QP) problem is solved at every sampling time to update the manipulated variables. Additional QP iterations may also be executed in order to get convergent results. An extended NEU approach in Wolf et al. (2011) includes an error estimate to the updated manipulated variable to reduce computational delay. Finally, a real-time iteration scheme was proposed in Diehl et al. (2005) where only one Newton or QP iteration of the NLP is performed at every sampling time.

For the case where the NLP solution requires more than one sampling time, we propose the amsNMPC method. This method appears in two variants: a serial approach and a parallel approach. The serial approach updates the manipulated variable every sampling time but solves the NLP problem at a lower frequency. The parallel approach applies multiple processors to solve a new NLP problem at every sampling time. When the controller receives a state measurement, the solution of a previous NLP problem is updated to obtain the corresponding manipulated variable, and a free processor is applied to a

new NLP problem. Each time an NLP problem is solved, the processor is then freed to solve new NLP problems.

This study is organized as follows. A brief introduction of numerical methods that form the basis of amsNMPC appears in the next section. Section 3 then presents serial and parallel approaches, and a CSTR example demonstrates the performance of amsNMPC in Section 4. Section 5 concludes the study and discusses future work.

2. METHODOLOGY

Basic concepts of NMPC can be found in Rawlings and Mayne (2009), where the current plant state and the plant model are used to predict future plant states. Based on these states, an NLP problem is solved to get the corresponding manipulated variables, and the manipulated variable at the first time step is injected into the plant. Here we assume that the dynamics of the plant can be described by

$$\hat{z}(k+1) = f(\hat{z}(k), u(k)) \tag{1}$$

where $\hat{z}(k)$ is the plant state and $u(k)$ is the manipulated variable. In a realistic scenario there are also errors such as measurement noise and plant-model mismatch, so (1) provides just a model prediction without errors. Here z_l and v_l denote predicted values of $\hat{z}(k+l)$ and $u(k+l)$, and y_l denotes the predicted output. Given the current state $\hat{z}(k)$, the NLP problem for NMPC at time t_k is formulated as

$$\begin{aligned} \min_{v_l, z_l} \quad & J := F(z_N) + \sum_{l=0}^{N-1} \psi(z_l, v_l) \\ s.t.\ & z_{l+1} = f(z_l, v_l),\ z_0 = \hat{z}(k),\ l = 0, ..., N-1 \\ & y_l = C z_l, y_l \in Y, v_l \in V \end{aligned} \tag{2}$$

where N is the horizon length, F and ψ are quadratic tracking terms, and f and ψ are Lipschitz continuous. y_l and v_l are bounded by constraints $y_l \in Y, v_l \in V$. The current plant state $\hat{z}(k)$ is the initial value of the NLP problem, and it is considered a fixed parameter. The optimal solution (z_l^*, v_l^*) leads to the optimal objective function value $J^*(\hat{z}(k))$, and v_0^* is injected into the plant as $u(k)$. Once $\hat{z}(k+1)$ is known, the horizon moves one step forward and the problem is solved for $u(k+1)$. For suitable choices of F and ψ, smoothness of f, and bounds on the manipulated variables, the system can be ensured to be stable, as proved in in Rawlings and Mayne (2009). Note that (2) has the same structure for all t_k and the only change in the data is $\hat{z}(k)$. $\hat{z}(k)$ is treated as a parameter p in the NLP solution and sensitivity algorithms discussed next.

2.1 NLP Sensitivity Concept and Calculation

If there is a small perturbation in the parameters of the NLP problem, sensitivity analysis can be used to find an approximation of the perturbed NLP solution. For notational convenience, we simplify the form of problem (2) as

$$\min_x \quad \phi(x;p), \quad s.t.\ c(x;p) = 0,\ x \geq 0 \tag{3}$$

where x are the optimization variables and p are fixed parameters.

The NLP (3) is solved by an interior point method described in Biegler (2010) with bound constraints $x \geq 0$ replaced by logarithmic barrier terms added to the objective function. Thus problem (3) is reformulated as:

$$\min_x \quad \phi(x;p) - \mu_l \sum_{i=1}^{n_x} \ln(x_i), \quad s.t.\ c(x;p) = 0 \tag{4}$$

We define the primal-dual Lagrange function $L(x, \lambda, \nu; p) = \phi(x;p) + c(x;p)^T\lambda - x^T\nu$ and the solution vector $s = [x^T, \lambda^T, \nu^T]^T$, where $\nu = \lim_{\mu \to 0} \mu X^{-1} e$. Having the optimal solution $s^*(p_0)$ with $p = p_0$ (subscript '*' indicates optimal solution) then expanding the Karush-Kuhn-Tucker (KKT) conditions at the optimal solution with parameter p, leads to the following:

$$\begin{aligned} 0 = \nabla_x L(s^*(p)) = \nabla_x L(s^*(p_0)) + \frac{d}{dp}(\nabla_x L(s^*(p_0)))^T \Delta p \\ + O(\|\Delta p\|^2) \end{aligned} \tag{5}$$

where $\nabla_x L(s^*(p_0)) = 0$ and $\Delta p = p - p_0$. Neglecting $O(\|\Delta p\|^2)$, we have

$$\frac{d}{dp}(\nabla_x L(s^*(p_0)))^T \Delta p = (M \frac{ds^T}{dp} + N_p)\Delta p \approx 0 \tag{6}$$

where $M = \begin{bmatrix} \nabla_{xx} L(s(p_0)) & \nabla_x c(s(p_0)) & -I \\ \nabla_x c(s(p_0))^T & 0 & 0 \\ V(p_0) & 0 & X(p_0) \end{bmatrix}$ is the KKT matrix, $N_p = \begin{bmatrix} \nabla_{xp} L(s(p_0)) \\ \nabla_p c(s(p_0))^T \\ 0 \end{bmatrix}$, $V = diag(\nu)$ and $X = diag(x)$. Expanding the right hand side of (6), we have

$$M \frac{ds^T}{dp} \Delta p + N_p \Delta p \approx 0 \tag{7}$$

and from $\frac{ds^T}{dp}\Delta p = \Delta s$, $N_p \Delta p = N$, (7) becomes

$$M \Delta s(p) \approx -N. \tag{8}$$

Thus, for every perturbation Δp, the solution of the neighboring problem can be approximated by (8).

2.2 Updating NLP Sensitivity with Additional Restrictions

Additional restrictions or conditions based on changes in s or p at the NLP solution are treated through unit vectors added to the rows and columns of the KKT matrix, and additional conditions added to the right hand side. The system (8) is reformulated as:

$$\left[\begin{array}{c|c} M & E_1 \\ \hline E_2 & 0 \end{array}\right] \left[\begin{array}{c} \Delta s \\ \hline \Delta p \end{array}\right] = - \left[\begin{array}{c} 0 \\ \hline r_1 \end{array}\right] \tag{9}$$

where $r_1 = s_i - s_i^*$ or $p_i - p_{0i}$, and E_1, E_2 are matrices containing unit vectors whose elements corresponding to the affected variables have a value of 1. If another variable or parameter is affected, E_1 and E_2 need to be updated by moving the element with value 1 to the location corresponding to the newly affected variable. System (9) is solved using the Schur complement method developed in Pirnay et al. (2011):

$$M \Delta s = -E_1 (E_2 M^{-1} E_1)^{-1} r_1 \tag{10}$$

Thus we can calculate the approximation of the perturbed solution using

$$s(p) = s^*(p_0) + \Delta s(p) \tag{11}$$

and the updated manipulated variable is contained in the perturbed solution vector $s(p)$.

Note that the KKT matrix M is directly available and already factorized after the NLP problem is solved. It is also used for the update of the following manipulated variables until the next NLP problem is solved. Since only backsolves of (9) are done, the update takes much less time than solving the NLP problem to get new solutions.

2.3 Active Set Changes

The direct sensitivity update of the manipulated variable $u(k+1)$ may lead some variables to violate their bounds. We apply two methods to deal with this case.

Fix-relax and Quadratic Programming The fix-relax strategy is described in Zavala (2008). If variable bounds are violated or multipliers become negative, additional conditions can be introduced in (9) to *fix* the perturbed variables to their bounds or *relax* negative multipliers to zero. If additional variables still violate their bounds after this step, or some other multipliers become negative, even more conditions are added so that eventually all variables remain within bounds with nonnegative multipliers. Repeated application of this fix-relax strategy leads to the solution of the following QP problem (Pirnay et al. (2011), Bartlett and Biegler (2006)):

$$\begin{aligned} \min_{\Delta x} \ \Phi = \Delta x^T \nabla_{xp} L(s^*(p_0); p_0)\Delta p \\ + \frac{1}{2}\Delta x^T \nabla_{xx} L(s^*(p_0); p_0)\Delta x \\ s.t. \ \ \nabla_p c(x^*; p_0)^T \Delta p + \nabla_x c(x^*; p)^T \Delta x = 0, \ x^* + \Delta x \geq 0 \end{aligned} \tag{12}$$

Clipping in First Interval If the scale of the problem is large, solving the QP problem (12) may require too many fix-relax iterations. Moreover, with large noise or plant-model mismatch, we might not be able to find a feasible solution for the QP problem. More QP problems could be conducted to get convergent results, but at additional cost. Instead we apply variable *Clipping in the First Interval*. Here we care only about the current manipulated variable and the next predicted state. If bounds are not violated for v_0 or z_1 after perturbation, nothing more needs to be done; otherwise, we find the largest steplength $\alpha \in [0, 1]$ such that

$$v^L \leq v_0 + \alpha \Delta v_0 \leq v^U, z^L \leq z_1 + \alpha \Delta z_1 \leq z^U$$

For notational convenience, in the following sections, we use the QP label to indicate a QP solution with fix-relax, and CFI to indicate clipping in the first interval.

3. amsNMPC FORMULATION

3.1 Serial Approach

When N_{samp} sampling times are required to solve the NLP problem, we solve the NLP N_{samp} sampling times in advance to get the manipulated variable for the current state. Fig. 1 shows the locations of $\hat{z}$, u, z and v. Suppose the optimal solution of the last NLP solution is known at t_k. Knowing $\hat{z}(k)$, v_0 is updated using (8) and injected into the plant as $u(k)$, while $z_{N_{samp}}$ is also updated from Δs. Then between t_k and $t_{k+N_{samp}}$, an NLP problem is solved in background using the updated $z_{N_{samp}}$ as the initial

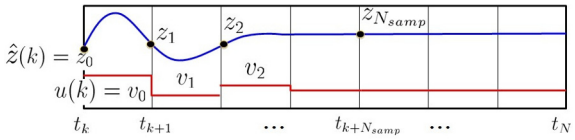


Fig. 1. Distribution of $\hat{z}$, u, z and v

value. In the meantime, the current manipulated variables $v_1, v_2, ..., v_{N_{samp}-1}$ are updated online using the sensitivity $(s^*(p_0) + \Delta s(p))$ based on solution of the previous NLP problem and (9). The serial approach works as follows:
Online: having $\hat{z}(k)$, update v_0 and $z_{N_{samp}}$ using Δs from (8).
Background: solve problem (2) between t_k and $t_{k+N_{samp}}$ with an updated $z_{N_{samp}}$ as the initial value.
Online: for $i = 1 : N_{samp} - 1$,

> At t_{k+i}, having $\hat{z}(k+i)$, update E_1 and E_2 in (9). Update v_i using Δs from (9). Inject the updated v_i as $u(k+i)$ to the plant.

Set $k = k + N_{samp}$ and repeat the cycle.

3.2 Parallel Approach

Again, suppose it takes N_{samp} sampling times to solve the NLP problem and the solution of the last NLP problem is known at t_k. Here, $v_{0|k}$ is updated from (8) once $\hat{z}(k)$ is obtained. Also, we define $z_{N_{samp}|k}$ as the N_{samp}th predicted state from the NLP solution, given the initial condition $\hat{z}(k)$. The updated $z_{N_{samp}|k}$ is the prediction of $\hat{z}(k + N_{samp})$, which becomes the initial value of the next NLP problem. Note that the manipulated variables are updated using (8) only and N_{samp} processors are applied. Once the manipulated variable is updated, a new linearized KKT system is obtained at each sampling time and at t_{k+1}, a new processor is applied to deal with the next background NLP. The parallel approach is implemented as follows:
For $i = 0 : N_{samp} - 1$,

> *Online*: at t_{k+i}, having $\hat{z}(k+i)$, update $v_{0|k+i}$ and $z_{N_{samp}|k+i}$ using $(s^*(p_0)+\Delta s(p))$ from (8). Inject the updated $v_{0|k+i}$ as $u(k+i)$ to the plant.
> *Background*: take the updated $z_{N_{samp}|k+i}$ as the initial value and solve the NLP problem (2) using a new processor.

Set $k = k + N_{samp}$ and repeat the cycle.

The serial approach and the parallel approach have strong similarities. They both solve the NLP problems N_{samp} steps in advance using the predicted states at $k + N_{samp}$ from the current optimal solution. The difference is that the serial approach uses only one processor, so the NLP problem is solved every N_{samp} sampling times, but the first N_{samp} manipulated variables in the horizon are updated by (9). The parallel approach uses multiple processors, so the NLP problem is solved every sampling time, but the first manipulated variable is updated using (8), and different KKT matrices are used at every sampling time. With either approach, amsNMPC is expected to reduce the on-line computational cost by two to three orders of

magnitude because backsolves take much less time than solving the NLP problem.

4. SIMULATION EXAMPLE

A dynamic CSTR example is used to demonstrate both amsNMPC strategies. The reaction $A \rightarrow B$ is exothermic and dimensionless model proposed by Hicks and Ray (1971) takes the form

$$\frac{dz_1}{dt} = \frac{F}{V}(1 - z_1) - k' e^{-E'/z_2} z_1^{\beta} \quad (13)$$

$$\frac{dz_2}{dt} = \frac{F}{V}(z_f - z_2) + k' e^{-E'/z_2} z_1^{\beta} - \alpha\nu(z_2 - z_c) \quad (14)$$

where z_1 and z_2 are the dimensionless concentration of A and temperature in the CSTR respectively, F is the flow rate of feed, $z_f = 0.395$ and $z_c = 0.382$ are the dimensionless concentration and temperature of feed respectively, $k' = 300 * 7.6^{\beta-1}$ is the dimensionless rate constant, $E' = 5$ is the ratio of dimensionless activation energy and gas constant, V is the CSTR volume which is assumed to be a constant, β is the order of the reaction, $\alpha = 1.95 * 10^{-4}$ is dimensionless heat transfer area and ν is the reactor jacket heat-transfer coefficient which is influenced by the coolant flow rate. We use ν and $\frac{V}{F}$ as two manipulated variables and take $\beta = 3$ in this example. The objective function takes the form

$$\min \quad J := \sum_{l=0}^{N} 10^6[(z_{1l} - z_{1des})^2 + (z_{2l} - z_{2des})^2] + \sum_{l=0}^{N-1} 2 * 10^{-3}[(u_{1l} - u_{1des})^2 + (u_{2l} - u_{2des})^2] \quad (15)$$

where $N = 50$, $z_{1des} = 0.1768$, $z_{2des} = 0.7083$, $u_{1des} = 800$, $u_{2des} = 10$, and z and u are bounded by $0 \le z_1 \le 1$, $0 \le z_2 \le 1$, $0 \le u_1 \le 2500$, $1 \le u_2 \le 40$.

We now compare the performance of ideal NMPC (iNMPC) and amsNMPC. For iNMPC, we assume that there is no computational delay and that the NLP solution takes a negligible amount of time; we use $N_{samp} = 0$ to indicate iNMPC. For amsNMPC we include the cases $N_{samp} = 1$, 2 and 3. When $N_{samp} = 1$, the parallel approach and the serial approach are the same and only one processor is applied.

First, we consider the case where there is no measurement noise nor model mismatch, and the setpoint changes during the process. In this case, iNMPC and amsNMPC have virtually identical performance with any differences due to numerical precision and truncation errors. This also indicates the nominal stability of iNMPC as well as amsNMPC.

Next, we introduce different amounts of measurement noise with a standard deviation given by a certain percentage of the current setpoint value. We also introduce plant-model mismatch by changing the parameter k' in (13) and (14) by a certain percentage. We also require active set changes and show the necessity of QP or CFI.

4.1 Performance with Setpoint Change

Assuming that there is no plant-model mismatch nor measurement noise, we first increase the setpoint by 10% and then decrease it by 10%. In this case the serial approach and the parallel approach behave exactly the same, and amsNMPC has identical performance as iNMPC. All controllers are able to catch the setpoint change immediately with only small overshoots, as seen in Fig. 2.

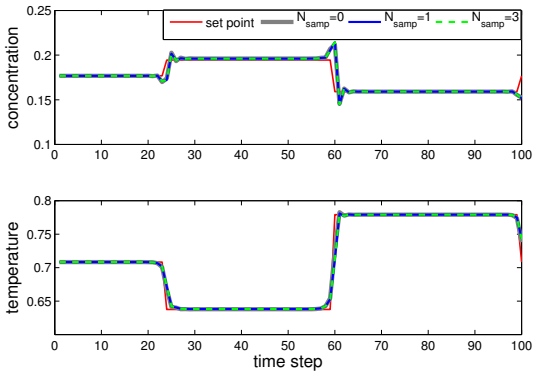
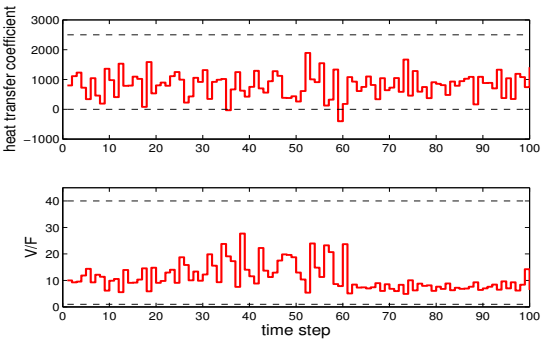


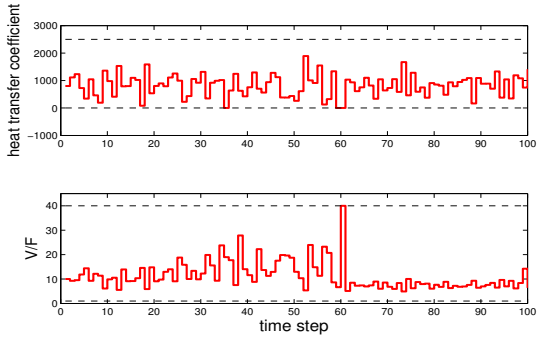
Fig. 2. Performance with 10% setpoint change

4.2 Performance with Setpoint Change and Measurement Noise

Besides 10% setpoint change, we also add measurement noise of 3%. Fig. 3 shows the control profile of amsNMPC with $N_{samp} = 1$ where dashed lines indicate variable bounds. From Fig. 3(a) we can see that bounds of manipulated variables are violated and the controller will encounter difficulties in control actions. As the noise level increases, these violations are more likely to happen, and either QP or CFI steps are needed. Using QP updates we see from Fig. 3(b) that manipulated variables now remain within their bounds. Using the QP strategy, we compare the performance of iNMPC and amsNMPC. For a 10% setpoint change and 3% measurement noise, the serial approach and the parallel approach still behave identically so we only plot the performance of the serial approach. From Fig. 4 we can see that amsNMPC behaves identically as iNMPC. It could also be noticed that when the noise is small, N_{samp} does not make any difference. We then apply CFI instead of QP and get the same results as shown in Fig. 4. Next we increase the noise level to 5%. Different performance of the serial approach and the parallel approach starts to occur. We also apply CFI to the serial approach, and compare the performance of the two approaches, with QP and CFI. From Fig. 5 we can see that iNMPC is able to tolerate the noise but amsNMPC does not behave as well as iNMPC. When $N_{samp} = 1$,the controller makes some effort to track the setpoint, but with larger N_{samp}, amsNMPC totally fails between $time = 40$ and $time = 60$. This is due to the use of previous, inaccurate KKT linearizations. As Fig. 5(b) shows, when $N_{samp} = 3$, the parallel approach is slightly better, but generally its performance is the same as the serial approach. Finally, from the comparison of Fig. 5(a) and Fig. 5(c), we can see that the effect of CFI is not as good as that of QP.



(a) amsNMPC without QP ($N_{samp} = 1$)



(b) amsNMPC with QP ($N_{samp} = 1$)

Fig. 3. Control profile of amsNMPC without/with QP ($N_{samp} = 1$)

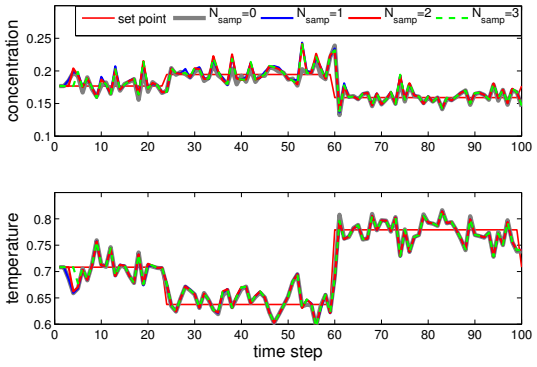
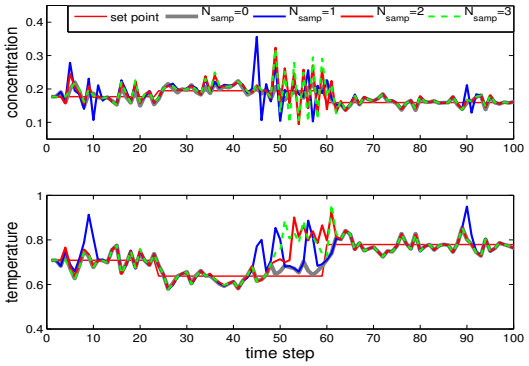


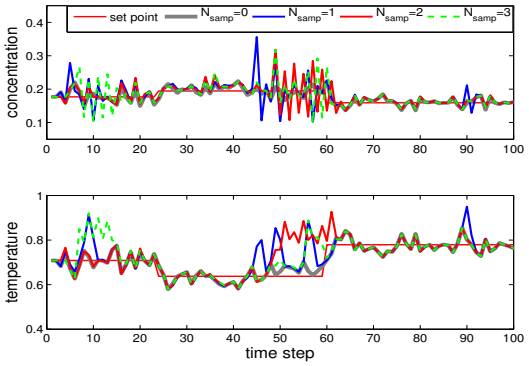
Fig. 4. Performance with 10% setpoint change and 3% measurement noise

4.3 Performance with Plant-model Mismatch

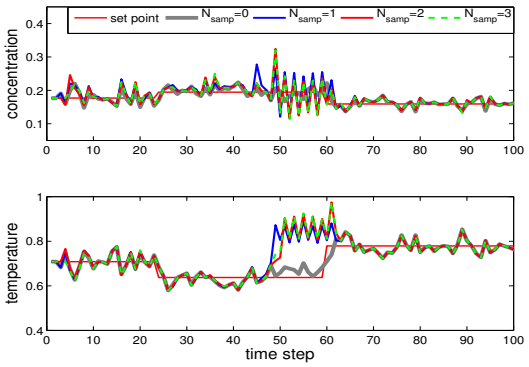
After looking into the performance of amsNMPC with QP under different noise levels, we introduce plant-model mismatch. Since the performance of the serial approach and the parallel approach is very similar, we only show the result of the serial approach. We first introduce plant-model mismatch of 10% and then increase it to 30%. The comparisons between different mismatch levels are shown in Fig. 6. From Fig. 6 we can see that like iNMPC, amsNMPC is able to handle a relatively large level of plant-model mismatch with offset, and as mismatch increases, the offset of amsNMPC becomes larger than that of iNMPC.



(a) 10% setpoint change, 5% measurement noise, serial approach with QP



(b) 10% setpoint change, 5% measurement noise, parallel approach with QP

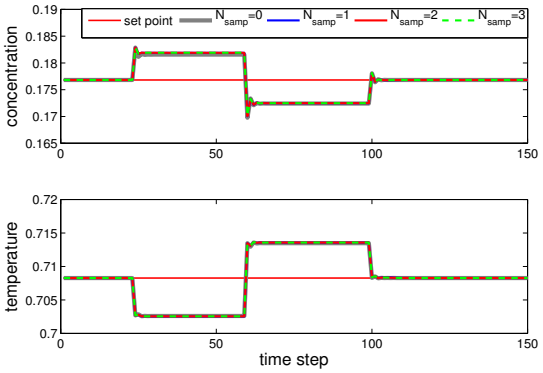


(c) 10% setpoint change, 5% measurement noise, serial approach with CFI

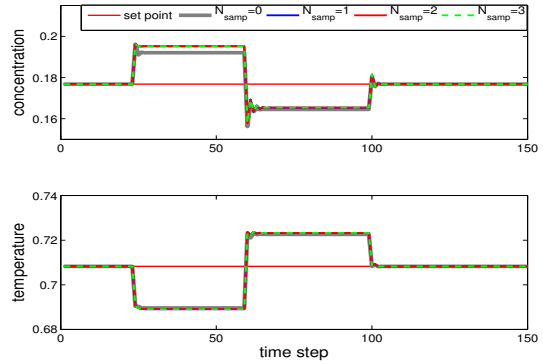
Fig. 5. Performance with 10% setpoint change and 5% measurement noise

5. CONCLUSIONS AND FUTURE WORK

In this study, we propose the amsNMPC strategy with negligible computational delay. Two variants are developed, the serial and the parallel approach. amsNMPC behaves identically as iNMPC in case of setpoint change, small noise levels and small plant-model mismatch, and it has the advantage of being able to solve NLP problems whose solutions take more than one sampling time. CFI also works as well as QP with the above cases. However, nominal stability and robust stability of amsNMPC and



(a) 10% plant-model mismatch



(b) 30% plant-model mismatch

Fig. 6. Performance of the serial approach with plant-model mismatch

CFI need to be analyzed. For the serial approach, as N_{samp} increases, an offset from setpoint starts to occur. Also its tolerance of measurement noise drops. Here, the performance becomes worse due to inaccurate (i.e., old) linearizations. For the parallel approach, with large N_{samp}, less offset in the states is observed compared with the serial approach, but generally it is not obvious that the parallel approach is much better than the serial approach. Large plant-model mismatch and strong nonlinearity will lead both approaches to fail, especially as N_{samp} increases. CFI also behaves worse as noise level increases.

The current example is much smaller than many industrial problems. We plan to generalize the extend the sensitivity approach and incorporate it within sIPOPT, a sensitivity extension for IPOPT developed by Pirnay et al. (2011). We also plan to test amsNMPC method for the case $N_{samp} > 1$ with real problems which have thousands of variables, such as distillation columns.

Finally, with ideal NMPC it is assumed that the plant states are immediately available at the beginning of a new sampling time and all states could be achieved. This is rarely true and a state estimation step is required as well. Moreover, in order to make use of the nonlinear model more efficiently, moving horizon estimation(MHE) from Michalska and Mayne (1995) could be used. To get offset-free plant states, the coupling of MHE and asNMPC, done by Huang (2010) along with the coupling of MHE and amsNMPC, will also be considered in further research.

REFERENCES

Bartlett, R.A. and Biegler, L.T. (2006). QPSchur: A dual, active-set, Schur-complement method for large-scale and structured convex quadratic programming. *Optimization and Engineering*, 7, 5–32.

Biegler, L.T. (2010). *Nonlinear Programming: Concepts, Algorithms, and Application to Chemical Processes*. SIAM, Philadelphia.

Diehl, M., Bock, H.D., and Schlöder, J.P. (2005). A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM J. Control and Optimization*, 43, 1714–1736.

Hicks, G.A. and Ray, W.H. (1971). Approximation methods for optimal control synthesis. *The Canadian Journal of Chemical Engineering*, 49, 522–528.

Huang, R. (2010). *Nonlinear Model Predictive Control and Dynamic Real Time Optimization for Large-scale Processes*. Ph.D. thesis, Carnegie Mellon University.

Michalska, H. and Mayne, D.Q. (1995). Moving horizon observers and observer-based control. *IEEE Transactions on Automatic Control*, 40, 995–1006.

Pannocchia, G., Rawlings, J.B., and Wright, S.J. (2007). Fast, large-scale model predictive control by partial enumeration. *Automatica*, 43, 852–860.

Pirnay, H., López-Negrete, R., and Biegler, L.T. (2011). Optimal sensitivity based on IPOPT. Submitted for publication. Url: http://www.optimization-online.org/DB_HTML/2011/04/3008.html.

Rawlings, J.B. and Mayne, D.Q. (2009). *Model predictive control: theory and design*. Nob Hill Publishing, Madison.

Wolf, I.J., Würth, L., and Marquarde, W. (2011). Rigorous solution vs. fast update: Acceptable computational delay in NMPC. In *Conference on Decision and Control*.

Würth, L., Hannemann, R., and Marquarde, W. (2009). Neighboring-extremal updates for nonlinear model-predictive control and dynamic real-time optimization. *Journal of Process Control*, 19, 1277–1288.

Zavala, V.M. (2008). *Computational Strategies for the Optimal Operation of Large-Scale Chemical Processes*. Ph.D. thesis, Carnegie Mellon University.

Zavala, V.M. and Biegler, L.T. (2009). The advanced-step NMPC controller: Optimality, stability and robustness. *Automatica*, 45, 86–93.