Control Engineering Practice 20 (2012) 869–881

Contents lists available at SciVerse ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

Nested partitions for global optimization in nonlinear model predictive control

Majid H.M. Chauhdry [a,b,*], Peter B. Luh [a]

[a] Department of Electrical & Computer Engineering, University of Connecticut, Storrs, CT 06269-2157, USA
[b] Department of Electrical Engineering, University of Engineering & Technology, Lahore, 54890, Pakistan

ARTICLE INFO

Article history:
Received 30 September 2010
Accepted 9 May 2012
Available online 22 June 2012

Keywords:
Nonlinear model predictive control
Nonconvex optimization
Stochastic optimization
Nested partitions
Partitioning scheme
Solution quality

ABSTRACT

In Nonlinear Model Predictive Control (NMPC), the optimization problem may be nonconvex. It is important to find a global solution since a local solution may not be able to operate the process at desired setpoints. Also the solution must be available before the control input has to be applied to the process. In this paper, a stochastic algorithm called the Nested Partitions Algorithm (NPA) is used for global optimization. The NPA divides the search space into smaller regions and either concentrates search in one of these regions called the most promising region or backtracks to a larger region in the search space based on a performance index. To adapt the NPA to solve dynamic NMPC with continuous variables, a new partitioning scheme is developed that focuses on the first few control moves in the control horizon. The expected number of iterations taken by the NPA is presented. Convergence speed is improved by reducing the size of the starting most promising region based on a good starting point. The discrete sampling nature of the NPA may cause difficulty in finding the global solution in a continuous space. A gradient-based search is used with the NPA to overcome this difficulty. The solution quality is assessed in terms of the error from the actual global minimum. The algorithm is shown to give a feasible solution that provides asymptotic stability. Case studies are used to show the algorithm performance in terms of tracking setpoints, cost, solution quality and convergence time.

1. Introduction

Model Predictive Control (MPC) is an important process control technique based on constrained optimization (e.g., Piché, Sayyar-Rodsari, Johnson, & Gerules, 2000; Norquay, Palazoglu, & Romagnoli, 1999; Henson, 1998; Camacho & Bordons, 2004; Qin & Badgwell, 2003; Zanin, Tvrzská de Gouvêa, & Odloak, 2002; Kawathekar & Riggs, 2007; Havlena & Findejs, 2005; Zhao, Guiver, Neelakantan, & Biegler, 2001; Seki, Ogawa, Ooyama, Akamatsu, Ohshima, & Yang, 2001). It produces control inputs to operate the process outputs at the desired setpoints by solving an optimization problem for each time step. Optimization is done using a process model to represent the actual process. Either a first principle or an empirical model can be used. In Nonlinear MPC (NMPC), the nonlinear system dynamics may cause the optimization problem to be nonconvex. In this case it is important to find the global solution since a local solution may not be able to drive the process outputs at the setpoints. Timing is also crucial as the solution has to be available before the next time step.

Usually gradient based deterministic methods such as Sequential Quadratic Programming (SQP) and Branch-and-Bound are used to solve the nonlinear optimization problem posed by NMPC (e.g., Martinsen, Biegler, & Foss, 2004; Kittisupakorn, Thitiyasook, Hussain, & Daosud, 2009; Diehl, Bock, Schlöder, Findeisen, Nagy, & Allgöwer, 2002; Sriniwas & Arkun, 1997; Cannon, 2004; Mu, Rees, & Liu, 2005; Sousa, Babuška, & Verbruggen, 1997; Baptista, Sousa, & Sá da Costa, 2001). In SQP, at each iteration, the system model is linearized to form a convex optimization problem, the solution to which is then used in the next iteration. Due to a gradient-based strategy, SQP may converge to a local minimum. Branch-and-Bound uses convex underestimates (relaxations) to solve the nonconvex problem. Good convex underestimates for some nonconvex functions may not be available. Sparse SQP methods (e.g., Betts & Huffman, 1992; Yokoyama, Suzuki, & Tsuchiya, 2008) use the sparsity of the hessian to reduce the computation time of the algorithm. Although the computation time has been reduced, the algorithm due to its gradient based strategy may still converge to a local minimum.

To avoid the problems related with gradient based methods, stochastic algorithms are being used. Stochastic optimization algorithms such as Genetic Algorithms (GA) (e.g., Onnen, Babuška, Kaymak, Sousa, Verbruggen, & Isermann, 1997; Sarimveis & Bafas, 2003; Al-Duwaish & Naeem, 2001; Martínez, Senent, & Blasco, 1998;

* Corresponding author. Postal address: 371 Fairfield Way Unit 2157, Storrs, Connecticut 06269-2157, USA. Tel.: +1 860 486 1801; fax: +1 860 486 1273.
E-mail addresses: majid.chauhdry@engr.uconn.edu (M.H.M. Chauhdry), luh@engr.uconn.edu (P.B. Luh).

0967-0661/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.conengprac.2012.05.003

Potočnik & Grabec, 2002; Fleming & Purshouse, 2002), Simulated Annealing (SA) (e.g., Ingber, 1993; Aggelogiannaki & Sarimveis, 2007), Integrated Controlled Random Search for Dynamic Systems (ICRS/DS) by Banga and Seider (1996) and Banga, Irizarry-Rivera, and Seider (1998), and the Nested Partitions Algorithm (NPA) by Shi and Ólafsson (2000, 2009) are gradient independent and have the ability to escape local minima. The Genetic Algorithms based on the process of survival of the fittest, generate a population of solutions called generation at each iteration and evaluate the fitness for each solution. For the next iteration, a new population is generated from the previous one. The Simulated Annealing based on the physical annealing process of solids, generates a solution set at each iteration and accepts solutions based on a probability distribution. The temperature and the number of solutions accepted decrease after each iteration until the algorithm freezes to a single solution. The ICRS/DS algorithm reduces the variance and moves the mean of the Gaussian distribution based on samples taken from the search space during each iteration. The algorithm stops after the change in either cost or solution values is small enough. The Nested Partitions Algorithm partitions the search space into smaller regions and takes random samples from each partition. Based on a performance index of the samples, a "most promising region" is selected. Either the algorithm further partitions the most promising region or "backtracks" to a bigger region if the sample with the best performance index is not from the most promising region. The search process is like a tree having the smallest most promising region containing the global minimum at the maximum depth of the tree. As shown by Shi and Ólafsson (2000, 2009), the algorithm converges to the global minimum with probability one if allowed to reach maximum depth. Other strategies that partition the search space and utilize search trees are in e.g., Bemporad & Filippi (2006), Johansen (2004) and Johansen & Grancharova (2003). These strategies compute the control law as an approximate explicit MPC in the parametric sense and the partitioning is done in the state space rather than the input.

In this paper, the Nested Partitions Algorithm is used to solve continuous nonconvex optimization problems in NMPC by finding the optimal control input directly from the input search space rather than in the parametric sense. The Nested Partitions Algorithm was originally designed to solve discrete nonconvex optimization problems and has been applied to problems such as the traveling salesman problem. The NPA has been chosen as it can be specialized to NMPC by the development of a partitioning scheme based on the moving horizon nature of NMPC. The partitioning scheme developed here, concentrates search more on the first few control moves in the control horizon than the later ones. The expected number of iterations taken by the NPA is also calculated. Solution time is improved by developing a method to reduce the number of iterations taken by the NPA to reach maximum depth. This is done by using a good starting point to start the algorithm with a smaller most promising region rather than the entire search space. As the algorithm is based on taking random samples from the search space, it will converge to a point in the vicinity of the global solution. In such a situation, the process output may not exactly follow the setpoint. To overcome this, a gradient based strategy is used with the NPA. The solution quality is assessed using the mean absolute error between the solution given by the algorithm and the true global minimum. The solution is shown to be feasible and asymptotically stable.

2. Nested partitions algorithm for nonlinear model predictive control

In this section, a brief review of the NMPC problem formulation is given in Section 2.1 with system dynamics, a continuous search space, and receding control and prediction horizons. In Section 2.2, a brief review of the stability of the NMPC problem is given. An overview of the Nested Partitions Algorithm is given in Section 2.3. Several schemes are developed to adapt the NPA to solve the dynamic NMPC. Using information from the receding horizon and that only the first control move of the control horizon is implemented at each time step, a new partitioning scheme that focuses on the first few control moves in the control horizon is presented in Section 2.4. The expected number of iterations taken by the NPA is calculated in Section 2.5. To obtain the solution before the next time step, a method is presented to improve the expected time taken by the NPA to reach maximum depth by using a good starting point to reduce the initial most promising region in Section 2.6. While solving the optimization problem, the NPA gets into the region containing the global minimum but due to its discrete sampling nature the NPA may converge to a point in the vicinity of the global minimum. A gradient-based strategy is used with the NPA to overcome this difficulty in Section 2.7. If the optimization algorithm cannot converge before the next time step, it is terminated with a feasible solution whose quality is assessed based on the mean absolute error between the solution and the true global minimum in Section 2.8. The algorithm gives a feasible solution for NMPC and provides closed loop stability as discussed in Section 2.9.

2.1. Problem formulation for NMPC

In the NMPC problem, the nonlinear discrete-time process model used is

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

where $\hat{y}$ is the predicted output vector and $\hat{u}$ is the predicted input vector. Starting with initial conditions $y(0)$ and $u(-1)$, the objective of NMPC is to reduce the error between the process outputs and the desired trajectory y_{sp} called output setpoints. This is done by applying control inputs $u(k)$ to the process. The control inputs are calculated by solving an optimization problem (e.g., Long, Polisetty, & Gatzke, 2006; Norquay et al. 1999; Henson, 1998; Oliveira, Amaral, Favier, & Dumont, 2000; Clarke, Mohtad, & Tuffs, 1987) for each time step k over a control horizon M for the inputs and a prediction horizon P for the outputs. The optimization problem $\mathcal{P}_k$ is given as:

$$\min_{\substack{\Delta\hat{u}(k+i),\hat{\varepsilon}_L(k),\hat{\varepsilon}_U(k) \\ 0\le i\le M-1}} J_k, \tag{2}$$

where

$$\begin{aligned} J_k = &\sum_{i=1}^{P-1} (\hat{y}(k+i)-y_{sp}(k+i))^T Q(\hat{y}(k+i)-y_{sp}(k+i)) \\ &+(\hat{y}(k+P)-y_{sp}(k+P))^T Q(\hat{y}(k+P)-y_{sp}(k+P)) \\ &+\sum_{i=0}^{M-1} \Delta\hat{u}(k+i)^T S\Delta\hat{u}(k+i) \\ &+\sum_{i=1}^{P-1} R^T\hat{\varepsilon}_L(k+i)+R^T\hat{\varepsilon}_U(k+i), \end{aligned} \tag{3}$$

s.t.

$$\Delta u_L \le \Delta\hat{u}(k+i) \le \Delta u_U \quad \text{for } 0\le i\le M-1, \tag{4}$$

$$u_L \le \hat{u}(k+i) \le u_U \quad \text{for } 0\le i\le M-1, \tag{5}$$

$$\hat{y}(k+i)-\hat{\varepsilon}_U(k+i) \le y_U \quad \text{for } 1\le i\le P-1, \tag{6}$$

$$-\hat{y}(k+i)-\hat{\varepsilon}_L(k+i) \le -y_L \quad \text{for } 1\le i\le P-1, \tag{7}$$

$$\hat{\varepsilon}_U(k+i), \hat{\varepsilon}_L(k+i) \ge 0 \quad \text{for } 1\le i\le P-1, \tag{8}$$

where $\Delta\hat{u}(k+i)=\hat{u}(k+i)-\hat{u}(k+i-1)$ and Q and S are positive semi-definite weight matrices on outputs and control moves, respectively. Subscripts L and U on Δu, u and y represent lower and upper bounds, respectively. The input constraints are handled by calculating the feasible region between the upper and lower bounds of the inputs. The constraints on y are treated as soft constraints as by Ricker, Subrahmanian, and Sim (1989), Mayne, Rawlings, Rao, and Scokaert (2000) and Zheng and Morari (1995). The penalty term for the soft constraints is formulated as an exact penalty by choosing a reasonably high value for the positive weight vector R on the constraint violation ε_U and ε_L (e.g., Santos, Biegler, & Castro, 2008; Santos, Afonso, Castro, Oliveira, & Biegler, 2001; Scokaert & Rawlings, 1999; Kerrigan & Maciejowski, 2000). The exact penalty formulation produces the same feasible solution as the original hard constrained problem when the latter is feasible. The exact penalty formulation violates the output constraints to produce a solution if the original hard constrained problem is infeasible. Output setpoints for which the hard constrained problem has infeasible output trajectories are not considered here. The second term in (3) is the terminal error penalty term. If $M<P$, move blocking is used as by Cagienard, Grieder, Kerrigan, and Morari (2007) and Gondhalekar, Imura, and Kashima (2009) in which the input moves from M to $P-1$ are kept zero or in other words the input is kept constant. It is the nonlinear system model f that may cause the overall objective function to be nonconvex. The NMPC algorithm is given step by step as:

For any time step k

Step 1: Get output $y(k)$ from the process.
Step 2: Use a global optimization algorithm to solve the optimization problem $\mathcal{P}_k$ given by (2)–(8).
Step 3: Use the first control move to implement the input $u(k)=\hat{u}(k)=\Delta\hat{u}(k)+u(k-1)$, where $u(k-1)$ is the input applied to the process in the previous time step.
Step 4: Set $k=k+1$ and go to *Step* 1.

A review of the stability of the NMPC problem (2)–(8) is given next.

2.2. Stability

The NMPC formulation in Section 2.1 has been shown to be asymptotically stable for example in Santos et al. (2008) where a perfect model f with no process disturbances is considered. At steady state values $(y(k)-y_{sp}(k), \Delta u(k))=(0,0)$, the value of $f=0$. A function belonging to class $\mathcal{H}_\infty$ is defined. A function $W(r)$: $\Re_+ \to \Re_+$, $r \in \Re_+$, belongs to class $\mathcal{H}_\infty$ if: (a) $W(r)$ is continuous; (b) $W(r)=0 \Leftrightarrow r=0$; $W(r)$ is nondecreasing; (d) $W(r)\to\infty$ when $r\to\infty$. The cost function J_k in (3) belongs to $\mathcal{H}_\infty$. Also J_k is bounded from below by zero. For stability, the terminal constraint $\hat{y}(k+P)-y_{sp}(k+P)=0$ is added to the problem. This constraint is usually satisfied by choosing a relatively high penalty on the terminal error $\hat{y}(k+P)-y_{sp}(k+P)$ in (3) or by choosing a sufficiently long prediction horizon P that ensures an admissible trajectory to satisfy the terminal constraint. Due to this constraint, the solution to problem $\mathcal{P}_k$ in (2)–(8) satisfies $(\hat{y}(k+i)-y_{sp}(k+i), \Delta\hat{u}(k+i))=(0,0)$ for $i\geq P$, assuming that (0,0) is within the input and output constraints. The globally optimal solution for $\mathcal{P}_k$ gives a cost value J_k^*. As a perfect model with no process disturbances is considered, the optimal solution to the problem $\mathcal{P}_k$ at time step k can be used as a starting feasible solution to the problem $\mathcal{P}_{k+1}$ at time step $k+1$ giving cost J_{k+1}. Clearly $J_{k+1}^* \leq J_{k+1}$. Furthermore, the value J_{k+1}^* can be no greater than J_k^* as the terminal constraint is only to be satisfied after one more time step. Substituting zero for the terminal error in the NMPC problem and taking the difference between J_k^* and J_{k+1}^* gives:

$$J_k^*-J_{k+1}^* \geq (y(k+1)-y_{sp}(k+1))^T Q(y(k+1)-y_{sp}(k+1)) + \Delta u(k)^T S\Delta u(k)+R^T\varepsilon_L(k+1)+R^T\varepsilon_U(k+1), \quad (9)$$

where the right hand side of the inequality belongs to class $\mathcal{H}_\infty$. This means that the sequence $\{J_k^*\}$ is nonincreasing over N time steps. Taking the sum over N time steps:

$$J_0^*-J_N^* = \sum_{k=0}^{N-1}(J_k^*-J_{k+1}^*) \geq \sum_{k=0}^{N-1}(y(k+1)-y_{sp}(k+1))^T Q(y(k+1)-y_{sp}(k+1)) + \Delta u(k)^T S\Delta u(k)+R^T\varepsilon_L(k+1)+R^T\varepsilon_U(k+1)), \quad (10)$$

and letting $N\to\infty$, the term inside the summation after the inequality approaches zero meaning $\hat{y}(k)-y_{sp}(k)\to 0$. Thus the NMPC problem is asymptotically stable.

2.3. Nested partitions algorithm

The Nested Partitions Algorithm by Shi and Ólafsson (2000, 2009) starts with a full dimensional search space Θ bounded by Θ_L and Θ_U and symmetric and polyhedral, more specifically a hyper-rectangle or hyper-cube in shape. At each iteration there is a region σ, where $\sigma\subseteq\Theta$, called the most promising region that is partitioned into M_P equal sized subregions along each dimension. The partitioning can be viewed as dividing each dimension of σ with M_P equally spaced hyper-planes that are orthogonal to that dimension. If the number of dimensions is M, then there are $M\cdot M_P$ subregions $\sigma_1, \sigma_2, \ldots, \sigma_{M\cdot Mp}$. The partitions also form symmetric and bounded polyhedra (hyper-rectangles or hyper-cubes). The region $\Theta\backslash\sigma$ is called the surrounding region. Each of the M_P+1 regions is sampled. Different distributions can be used for sampling. The uniform distribution is used here. A performance index is calculated for each sample. If the sample with the best performance index is from one of the subregions $\sigma_1, \sigma_2, \ldots, \sigma_{M\cdot Mp}$, say σ_i, then σ_i becomes the most promising region for the next iteration and the current most promising region σ becomes the superregion $s(\sigma_i)$ of the subregion σ_i. If the sample with the best performance index is from the surrounding region $\Theta\backslash\sigma$, then the algorithm backtracks and the superregion $s(\sigma)$ of the current most promising region σ becomes the most promising region for the next iteration. The new most promising region is then partitioned in a similar fashion.

From a graph theory perspective, the Nested Partitions Algorithm forms a tree with the algorithm starting with the entire search space Θ as the most promising region at depth zero and the smallest possible most promising region containing the global minimum at maximum depth. Considering each dimension i with maximum depth $d_{i,max}$ for $0\leq i\leq M-1$, if R_i is the size the most promising region at $d_{i,max}$ and $|\Theta_{U,i}-\Theta_{L,i}|$ is the size of the entire search space, then there will be $|\Theta_{U,i}-\Theta_{L,i}|/R_i$ regions of size R_i along dimension i at $d_{i,max}$. Also, there will be $M_P^{d_{max}}$ possible regions of size R_i at $d_{i,max}$. This means that $M_P^{di,max}=|\Theta_{U,i}-\Theta_{L,i}|/R_i$. After taking *log* on both sides and rearranging, the maximum depth $d_{i,max}$ is determined by:

$$d_{i,\max} = \left\lceil \frac{\log(|\Theta_{U,i}-\Theta_{L,i}|/R_i)}{\log(M_P)} \right\rceil. \quad (11)$$

For simplicity, assume all dimensions have the same maximum depth d_{max} and the same size $|\Theta_U-\Theta_L|$ i.e., Θ is a hyper-cube. As shown by Shi and Ólafsson (2000, 2009), the NPA converges to the most promising region containing the global minimum at maximum depth d_{max} with probability one. The condition on the most promising region at d_{max} is that it should be

small enough that any sample taken from the most promising region would have a better performance index than any sample from the surrounding region. The Nested Partitions Algorithm introduced above is summarized in the following steps:

Step 1: Initialize the entire search space Θ as the most promising region σ. Set initial depth $d=0$. During initialization, the surrounding region $\Theta\backslash\sigma$ is empty.

Step 2: Partition the most promising region σ along all M dimensions into M_P equal sized subregions $\sigma_1, \sigma_2, \ldots, \sigma_{M \cdot Mp}$. Set $d=d+1$.

Step 3: Sample all $M \cdot M_P$ partitions and the surrounding region $\Theta\backslash\sigma$.

Step 4: Calculate the performance index for each sample.

Step 5: Further Partitioning:
If the sample with the best performance index is from one of the subregions σ_i where $\sigma_i \in \{\sigma_1, \sigma_2, \ldots, \sigma_{M \cdot Mp}\}$, then:
(i) Set the current most promising region σ as the superregion of σ_i, i.e., $s(\sigma_i)=\sigma$.
(ii) Set σ_i as the most promising region for the next iteration, i.e., $\sigma=\sigma_i$.
(iii) If $d=d_{max}$, then stop. Else go to *Step* 2.
Backtracking:
If the sample with the best performance index is from the surrounding region $\Theta\backslash\sigma$ then:
(i) Set the superregion $s(\sigma)$ of σ as the next most promising region, i.e., $\sigma=s(\sigma)$.
(ii) Set $d=d-1$. Go to *Step 2*.

Fig. 1 shows the Nested Partitions Algorithm for a 2-dimensional static problem where both dimensions are partitioned simultaneously with $M_P=2$. Backtracking (shown by the red arrow) takes place at depth $d=3$ for the situation when the best performance index after sampling is from the surrounding region.

2.4. Partitioning scheme

To adapt the NPA to NMPC, a new partitioning scheme is developed. For each time step of NMPC, the NPA is used to solve the optimization problem (2)–(8). The cost function in (3) is used as the performance index. The region Θ is defined as the region bounded by Δu_U, Δu_L, u_U and u_L to satisfy (4) and (5). Inside this region, the output constraints (6) and (7) are handled through the exact penalty function. The region bounded by Δu_U and Δu_L satisfying (4) forms a hyper-rectangle. The feasible region Θ

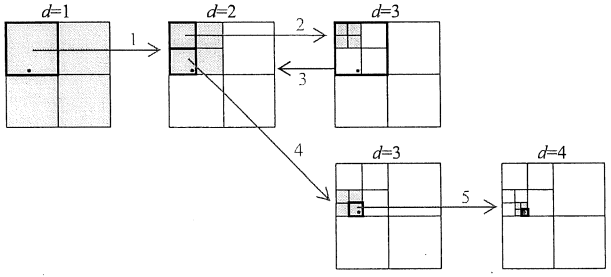
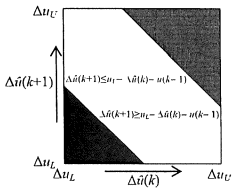


Fig. 1. Nested Partitions Algorithm with $M_P=2$ showing partitions at depths $d=1$, 2, 3 and 4 for a 2-dimensional search space. The gray area shows the most promising region being partitioned, the white region shows the surrounding region and the highlighted region shows the most promising region selected to be partitioned in the next iteration as indicated by the arrows. The numbers on the arrows show the sequence of the iterations. The red dot shows the global minimum point. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



Step by step calculation of the feasible region:
1. $\hat{u}(k+i)=\hat{u}(k+i-1)+\Delta\hat{u}(k+i)$ for $i=0$, 1 and $\hat{u}(k-1)=u(k-1)$
2. $u_L \le \hat{u}(k+1) \le u_U$
3. $u_L \le \hat{u}(k)+\Delta\hat{u}(k+1) \le u_U$
4. $u_L \le u(k-1)+\Delta\hat{u}(k)+\Delta\hat{u}(k+1) \le u_U$
5. $u_L-u(k-1)-\Delta\hat{u}(k) \le \Delta\hat{u}(k+1) \le u_U-u(k-1)-\Delta\hat{u}(k)$

Samples are taken between: $\begin{pmatrix} \max(\Delta u_L, u_L-\hat{u}(k)) \\ \max(\Delta u_L, u_L-\Delta\hat{u}(k)-u(k-1)) \end{pmatrix}$ and $\begin{pmatrix} \min(\Delta u_L, u_L-\hat{u}(k)) \\ \min(\Delta u_L, u_L-\Delta\hat{u}(k)-u(k-1)) \end{pmatrix}$

Fig. 2. Calculation of the region Θ satisfying (4) and (5). The rectangle represents the region bounded by Δu_L and Δu_U, where the horizontal and vertical axes represent $\Delta\hat{u}(k)$ and $\Delta\hat{u}(k+1)$, respectively. The white area is the feasible region Θ and the shaded area in blue and red is the infeasble region. The region used for sampling is also given. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

satisfying both (4) and (5) is calculated by drawing hyper-planes representing u_L and u_U in this hyper-rectangle. The hyper-planes are defined by recursively using the equation $\hat{u}(k+i)=\hat{u}(k+i-1)+\Delta\hat{u}(k+i)$ for $0 \le i \le M-1$ and $\hat{u}(k-1)=u(k-1)$. After drawing the hyper-planes, the feasible region Θ is a full-dimensional polyhedron bounded from all sides by the hyper-planes and Δu_U and Δu_L. Fig. 2 shows the feasible region Θ (shown in white) for an NMPC controller with control horizon $M=2$ starting with an initial value of $\hat{u}(k-1)=u(k-1)$. This way of calculating the feasible region Θ is different from the ray-shooting method as in Bemporad and Filipi (2006) and Gilbert and Tan (1991) where the feasible region is calculated as an approximation. Fig. 2 also shows a step-by-step derivation of the equations of the hyper-planes representing u_L and u_U and the region within which the samples are taken.

For problem $\mathcal{P}_k$ at time step k, only the first predicted control move $\Delta\hat{u}(k)$ of the optimal solution is actually implemented out of the M predicted control moves. As mentioned in Section 2.2, the optimal solution to problem $\mathcal{P}_k$ at time step k is used as a starting feasible solution for problem $\mathcal{P}_{k+1}$ at time step $k+1$. The first predicted control move $\Delta\hat{u}(k+1)$ out of the M control moves of the optimal solution to problem $\mathcal{P}_{k+1}$ is implemented at time step $k+1$ and so on. This property of the receding horizon in NMPC forms the basis of the partitioning scheme for the NPA in which more focus is put when partitioning the first few control moves in the control horizon than the later ones.

Let the M control moves $\Delta\hat{u}(k)$, $\Delta\hat{u}(k+1)$, …, $\Delta\hat{u}(k+M-1)$ in the control horizon represent an M-dimensional search space. The feasible region Θ is a subset of this search space. As partitioning will be done along the M dimensions of the control moves, symmetry of Θ is of no concern. Samples are taken only from the region Θ in the partitions as calculated in Fig. 2. Any sample not satisfying (6) and (7) are penalized by the exact penalty term in (3). Let $d_0, d_1, \ldots, d_{M-1}$ represent the partitioning depths of the M control moves, respectively with maximum depths $d_{0,max}, d_{1,max}, \ldots, d_{M-1,max}$, respectively. For simplicity assume $d_{0,max}=d_{1,max}=\ldots=d_{M-1,max}=d_{max}$. The maximum depth d_{max} is calculated using $\Delta u_U-\Delta u_L$ in (11). The hyper-rectangle bounded by Δu_U and Δu_L is the initial most promising region. The dimension along the first control move $\Delta\hat{u}(k)$ is divided into M_P partitions by drawing hyper-planes orthogonal to $\Delta\hat{u}(k)$ resulting in M_P bounded M-dimensional hyper-rectangles. Samples are taken from the subset of the feasible region inside each hyper-rectangle. The samples are M-dimensional vectors. The hyper-rectangle with the sample having the best value of the performance index becomes the next most promising region and the union of the remaining

hyper-rectangles becomes the surrounding region. The depths at this point are $d_0=1$ and $d_1=0$, $d_2=0$, …, $d_{M-1}=0$. Partitioning is continued until $\Delta\hat{u}(k)$ has been partitioned to a certain depth, say $d_0=n_0$. At this point, the depths of all M control moves are $d_0=n_0$ and $d_1=0$, $d_2=0$, …, $d_{M-1}=0$. In a similar way using the hyper-rectangle representing the most promising region at $d_0=n_0$, the dimension along the second control move $\Delta\hat{u}(k+1)$ is partitioned to a certain depth, say $d_1=n_1$. At this point, the depths are $d_0=n_0$, $d_1=n_1$ and $d_2=0$, $d_3=0$, …, $d_{M-1}=0$. Partitioning is continued similarly for the remaining control moves until all M control moves have been partitioned to certain depths, $d_0=n_0$, $d_1=n_1$, …, $d_{M-1}=n_{M-1}$. To put more focus, the first few control moves in the control horizon are partitioned to a larger depth than the later ones i.e., $n_0 \geq n_1 \geq \ldots \geq n_{M-1}$. Partitioning is continued by further partitioning $\Delta\hat{u}(k)$ from $d_0=n_0$ to $d_0=2n_0$, then partitioning $\Delta\hat{u}(k+1)$ from $d_1=n_1$ to $d_1=2n_1$ and so on until all M control moves have been partitioned to depths $d_0=2n_0$, $d_1=2n_1$, …, $d_{M-1}=2n_{M-1}$. Using this scheme, the control moves in the beginning of the control horizon reach maximum depth before the later ones. A control move is not partitioned beyond maximum depth even if the later ones still have not reached maximum depth. The solution is the sample giving the best performance index throughout the partitioning. The algorithm stops after maximum depth has been reached for all control moves or is terminated to get a solution before the next time step. The NPA with this partitioning scheme is summarized in the following steps:

Step 1: Initialize σ as the region between Δu_U and Δu_L. Set initial depths $d_0=0$, $d_1=0$, …, $d_{M-1}=0$. During initialization, the surrounding region is empty.

while $0 \leq i \leq M-1$ go from *Step2* through *Step 6*.

Step 2: If $d_i=d_{max}$ then $i=i+1$. Start *Step 2* again. Else $temp=d_i$.

Step 3: Partition the most promising region σ along dimension $\Delta\hat{u}(k+i)$ into M_P equal sized subregions $\sigma_1, \sigma_2, \ldots, \sigma_{Mp}$. Set $d_i=d_i+1$.

Step 4: Sample all M_P partitions and the surrounding region.

Step 5: Calculate the performance index for each sample.

Step 6: Further Partitioning:
If the sample with the best performance index is from one of the subregions σ_i where $\sigma_i \in \{\sigma_1, \sigma_2, \ldots, \sigma_{Mp}\}$, then:
(i) Set the current most promising region σ as the superregion of σ_i, i.e., $s(\sigma_i)=\sigma$.
(ii) Set σ_i as the most promising region for the next iteration, i.e., $\sigma=\sigma_i$.
(iii) If $d_i=d_{max}$ or $d_i=temp+n_i$ then $i=i+1$and go to *Step* 2. Else go to *Step* 3.
Backtracking:
If the sample with the best performance index is from the surrounding region $\Theta\backslash\sigma$ then:
(i) Set the superregion $s(\sigma)$ of the current most promising region σ as the next most promising region, i.e., $\sigma=s(\sigma)$.
(ii) Set $d_i=d_i-1$. Go to *Step 3*.

End of while loop

Step 7: If $d_j=d_{max} \forall 0 \leq j \leq M-1$ then stop. Else $i=0$ and restart while loop.

The partitioning scheme for an NMPC controller with control horizon $M=2$ and maximum depth $d_{max}=4$ is shown in Fig. 3. In contrast to the static problem in Fig. 1 where both dimensions are partitioned simultaneously and equally ($n_0=n_1$), Fig. 3 shows the partitioning scheme where both dimensions are partitioned sequentially and unequally with $n_0=2$ and $n_1=1$ based on the dynamics of NMPC. Control move $\Delta\hat{u}(k)$ is first partitioned to depth $d_0=2$ and then $\Delta\hat{u}(k+1)$ is partitioned to depth $d_1=1$. Then $\Delta\hat{u}(k)$ is further partitioned progressively to depth $d_0=4$. At this point a situation is presented in which the best performance index after sampling is from the surrounding region and not from any of the partitions of the current most promising region. This causes the algorithm to backtrack from depth $d_0=4$ to the superregion at depth $d_0=3$. After this $\Delta\hat{u}(k)$ is partitioned again to depth $d_0=4$ and this time the sampling has resulted in selecting the correct most promising region. Next $\Delta\hat{u}(k+1)$ is further partitioned to depth $d_1=4$. It can be seen from Fig. 3 that n_0 and n_1 are less than the maximum depth. In general $n_0, n_1, \ldots, n_{M-1} \ll d_{max}$ or in other words, the number of depths through which any control move is partitioned each time should be significantly less than the maximum depth. The reason behind this is that the optimization problem (2)–(8) is based on all M control moves and partitioning any control move alone through a very large number of depths may cause that control move to move towards a value that is away from the global minimum causing excessive backtracking.

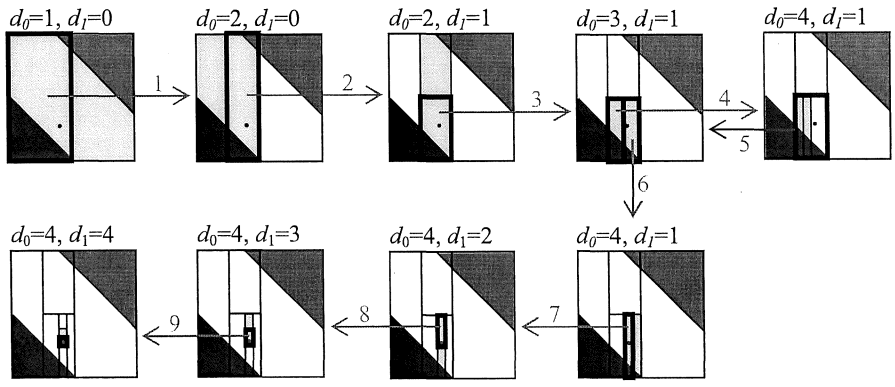


Fig. 3. Partitioning scheme for $M_p=2$, $d_{max}=4$, $n_0=2$ and $n_1=1$. Horizontal and vertical sides of the rectangles represent $\Delta\hat{u}(k)$ and $\Delta\hat{u}(k+1)$, respectively. The blue and red shaded areas show the infeasible region and the red dot indicates the global minimum point. Sampling is only done in the gray and white areas shown as the feasible region inside the current most promising and surrounding regions, respectively. The algorithm is terminated with $d_0=d_1=4$. As $\Delta\hat{u}(k)$ reaches d_{max} before $\Delta\hat{u}(k+1)$, the global minimum is trapped along the $\Delta\hat{u}(k)$ dimension before the $\Delta\hat{u}(k+1)$ dimension. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2.5. Expected number of iterations

In NMPC, the solution has to be computed within the sampling time. Due to this time constraint, the NPA may need to be stopped before reaching the maximum depth. This will affect the quality of the solution. To reach the maximum depth, the NPA is expected to go through a certain number of iterations I, where an iteration is when the NPA either further partitions a subregion of the most promising region or it backtracks. To get a rough idea of how many iterations I the NPA will take to reach maximum depth, the expected value of I is calculated. In the following, the expected number of iterations to reach maximum depth is first calculated for a problem with control horizon one and then generalized for problems with control horizon $M>1$.

For a problem with control horizon one starting at depth zero, if the algorithm always selects the correct region, i.e., the region containing the optimal solution, then there would be no backtracks and the NPA will reach maximum depth in d_{max} iterations. This may not always be the case and the NPA may go through backtracking in which case the expected number of iterations to reach maximum depth depends on the probability p of selecting the correct region. Shi and Ólafsson (2009) using results from Weesakul (1961) and considering the NPA as a random walk, the most promising region at d_{max} as the absorbing barrier, and assuming a probability $p>0.5$ of moving towards the absorbing barrier, give the expected number of iterations to reach d_{max} as:

$$E[I] \leq \frac{d_{max}}{(2p-1)}. \tag{12}$$

Similarly, for problems with control horizon $M>1$ and maximum depth d_{max} of all M control moves, the NPA will require Md_{max} iterations to reach the maximum depths when there is no backtracking. When there is backtracking, (12) can be extended to obtain the expected number of iterations to reach the maximum depths as:

$$E[I] \leq \frac{Md_{max}}{(2p-1)}. \tag{13}$$

A complete derivation of the expected number of iterations is given by Shi and Ólafsson (2009). Based on (13), the number of iterations the NPA is expected to take will be between Md_{max} and $Md_{max}/(2p-1)$. If fewer iterations are allowed, the NPA might only reach a depth lower than the maximum depth. The following subsection provides a technique on how to reduce the number of iterations required to reach maximum depth.

2.6. Reducing the number of iterations in the NPA

In MPC, the solution to problem $\mathcal{P}_k$ at time step k is used as a starting feasible solution for problem $\mathcal{P}_{k+1}$ at time step $k+1$. From the NPA point of view, this information can be used as a good heuristic to reduce the size of the initial most promising region hence having a starting depth greater than zero and in turn improving the algorithm's expected time to reach maximum depth. Using the solution to problem $\mathcal{P}_k$ at time step k as a good starting point, instead of using $d_{start}=0$ as the starting depth and the region between Δu_U and Δu_L as the starting most promising region, respectively the NPA can begin with a starting depth greater than zero. The starting most promising region depth $d_{start}>0$ contains the neighborhood of the starting point and is smaller than the region between Δu_U and Δu_L. The feasible region is calculated as in Section 2.4. The starting depth can be decided heuristically based on the depth at which the NPA was terminated in the previous time step or based on experience. From the starting depth, the size of the initial most promising region can be decided. The number of depths the algorithm has to go through by starting at depth $d_{start}>0$ is $d_{max}-d_{start}$. This is less than d_{max} and in turn will improve the expected number of iterations required to reach maximum depth.

2.7. Getting the global solution

When the Nested Partitions Algorithm is applied to NMPC, it searches for the global solution in a continuous space using discrete random samples. Due to the discrete sampling nature of the NPA, instead of converging to the global solution $\Delta\hat{u}^*$ (dropping $(k+i)$ for $0\leq i\leq M-1$ for ease of notation), it will converge to a point $\underline{\Delta\hat{u}}$ in its vicinity inside the region of attraction $A(\Delta\hat{u}^*)$, where $A(\Delta\hat{u}^*)\subseteq\Theta$ such that a local search heuristic starting with any $\underline{\Delta\hat{u}}\in A(\Delta\hat{u}^*)$, would yield $\Delta\hat{u}^*$. Shi and Ólafsson (2000, 2009) use a local search heuristic within each partition and run it for each sample using the sample as the starting point of the search. For NMPC, this may not be feasible as it is not known how much time will be required especially as the local search heuristic is to be run for each sample. To avoid this, the solution obtained from the NPA is used as a starting point for a gradient-based approach to solve (2)–(8) to form the NPA plus Gradient Search (NPA-GS) to find the global minimum $\Delta\hat{u}^*$. If $\underline{\sigma}$ with depth $\underline{d}\leq d_{max}$ is the largest possible most promising region such that $\underline{\sigma}\subseteq A(\Delta\hat{u}^*)$, then to calculate maximum depth d_{max} in (11), R and M_P should be selected such that the NPA can easily reach $\underline{\sigma}$ at depth $\underline{d}$ well before the next sampling time. Clearly, the most promising region at maximum depth containing the global minimum is a subset of $\underline{\sigma}$. Furthermore, as the solution from the NPA is the sample with the best cost value throughout the partitioning, the solution is most likely already in the vicinity of the global minimum. If time is available after reaching $\underline{d}$ then the NPA should be allowed to reach maximum depth. The previous subsection provides a method to reduce the number of iterations required to reach maximum depth. This gives the gradient-based search a better starting point closer to the global minimum. As the starting point lies in the vicinity of the global solution, the gradient-based algorithm will not require excessive computational effort.

2.8. Solution quality

As shown in the previous section, sometimes the NPA may only be able to reach a depth between $\underline{d}$ and d_{max} inside the region of attraction of the optimal solution. To get the optimal solution, the NPA-GS goes through a gradient-based search starting from the solution given by the NPA. In deterministic methods, solution quality can be assessed by a lower bound, e.g., dual value using Lagrange relaxation provides a lower bound to the optimal primal value and the percentage difference between the primal and dual values is used as a measure of solution quality which is calculated along with the solution online. For the NPA-GS, multiple runs (Monte Carlo simulations) of the algorithm are performed offline in which it is only allowed to reach the depth it is able to reach online. The simulations are performed offline as the solution has to be applied before the next sampling time and running the algorithm multiple times online would not be possible. The solution quality is then given as the error between the average cost of these runs and the actual global minimum. The actual global minimum can also be found offline by running the algorithm for a long enough time e.g., allow ten times more time than the time taken by the algorithm online. If the algorithm is run N times with solution costs J_n for $1\leq n\leq N$, the average cost J_{ave} is given by:

$$J_{ave}=\frac{1}{N}\sum_{n=1}^{N}J_n. \tag{14}$$

The mean absolute error (MAE) is given as the absolute of the difference between the average cost J_{ave} and the actual global minimum J^*:

$$MAE = |J_{ave} - J^*|. \tag{15}$$

Furthermore, the standard error (SE) calculated as the root mean square of the deviation from the actual value J^* is used as a measure of dispersion of the solution from the actual value:

$$SE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(J_n - J^*)^2}. \tag{16}$$

The solution quality is affected by depth at which the algorithm stops. It is expected that the greater the depth the algorithm is allowed to reach, the better the solution quality can be.

2.9. Feasibility and stability using NPA-GS

The solution given by the NPA-GS for the NMPC problem in Section 2.1 is both feasible and provides asymptotic closed loop stability of Section 2.2. The input feasibility is satisfied by sampling only in the region within the bounds on the inputs and input moves as calculated in Section 2.4. The exact penalty approach used in the NMPC formulation provides the same output feasible solution as the solution from the original feasible hard constrained problem. As discussed in Section 2.6, the NPA-GS uses the solution to problem $\mathcal{P}_k$ with cost J_k^* at time step k as a starting feasible solution to problem $\mathcal{P}_{k+1}$ with cost J_{k+1} at time step $k+1$. The NPA-GS improves on the starting point to give a cost value $J_{k+1}^* \le J_{k+1}$. Using the same arguments as in Section 2.2, $J_{k+1}^* \le J_k^*$. This results in a nonincreasing sequence J_k^* which is required for asymptotic stability. Furthermore, as the NMPC optimization problem is solved directly, the stability analysis in Section 2.2 directly applies to NMPC solved using the NPA-GS unlike approximate explicit NMPC algorithms (e.g., Bemporad & Filippi, 2001; Johansen, 2004; Johansen & Grancharova, 2003) where the stability is shown within a tolerance limit due to the approximation.

3. Case studies

In this section, NMPC is used to control different processes. The NPA-GS was implemented in MATLAB® where the `fmincon` function (which uses Sequential Quadratic Programming) of the Optimization Toolbox™ was used for the gradient-search at the end of the NPA. For comparison with the NPA-GS, the `fmincon` function is also used as a separate gradient-based solver. Three examples are presented here, i.e., a single-input-single-output model, a continuously stirred tank reactor (CSTR) with Vanne de Vusse reactions and a bioreactor. In all the examples, the NPA-GS is compared with the gradient-based solver in terms of tracking the setpoints and cost. In addition, the first example demonstrates solution quality along with the number of iterations taken by the NPA, the second example shows the time taken by the solvers to solve the problem and the effect of allowing a smaller depth than d_{max} on solution quality, the third example demonstrates the control of a more complex system with multiple inputs and outputs with constraints.

Example 1. Single-Input-Single-Output Model

The nonlinear single-input-single-output model used by Sriniwas and Arkun (1997) is used here. The model equation is:

$$y(k+1) = 1.0 + y(k)u(k-2) - 2u(k-1)u(k). \tag{17}$$

The cost function to be minimized is:

$$J = \sum_{i=1}^{P-1}(\hat{y}(k+i) - y_{sp})^2 + 1.5(\hat{y}(k+P) - y_{sp})^2 + \sum_{i=0}^{M-1}\Delta\hat{u}(k+i)^2, \tag{18}$$

s.t.

$$-0.5 \le \hat{u}(k+i) \le 1.0 \quad \text{for } 0 \le i \le M-1, \tag{19}$$

$$-0.5 \le \Delta\hat{u}(k+i) \le 1.0 \quad \text{for } 0 \le i \le M-1. \tag{20}$$

There are no constraints on the output y. The setpoint y_{sp} is 0.0 and the initial conditions are: $y(k)=0.0$, $u(k-1)=0.0$ and $u(k-2)=0.0$. The cost function with $M=1$and $P=2$ for the first time step is shown in Fig. 4. The NMPC is applied to the process for a simulation time of 20 samples. For the NPA, the starting depth $d_{init}=0$, maximum depth $d_{max}=8$ and the number of partitions $M_p=2$.

The cost function in Fig. 4 shows that with any starting point less than zero, the gradient-based solver will converge to -0.5 (local minimum) and with any starting point greater than zero it will converge to 0.6124 (global minimum). Suppose the starting point for the gradient-based solver is -0.1. Fig. 5 shows the NMPC results. It can be seen that the controller using the gradient-based solver is unable to follow the setpoint as it converges to the local minimum. The controller using the NPA-GS is able to track the setpoint as it is able to find the global minimum. For comparison with results by Sriniwas and Arkun (1997), the model is also controlled with $M=2$ and $P=2$ as shown in Fig. 5. Sriniwas and Arkun (1997) used a conventional QP solver and a convex relaxation based global optimization strategy to solve this problem. The conventional QP solver with an initial condition of -0.1 has been shown to converge to a local minimum and hence the controller was unable to follow the setpoint. The output y using the NPA-GS is seen to reach the setpoint before sampling time 5 compared to the optimization approach by Sriniwas and Arkun (1997) which reaches the setpoint at sampling time 6.

The total simulation cost for $M=1$ and $P=2$ using (18) is 6.9722 using the gradient-based solver and is 1.4691 using the NPA-GS. For $M=2$ and $P=2$, the costs using the gradient-based solver and NPA-GS are 6.7472 and 1.4561, respectively. This total

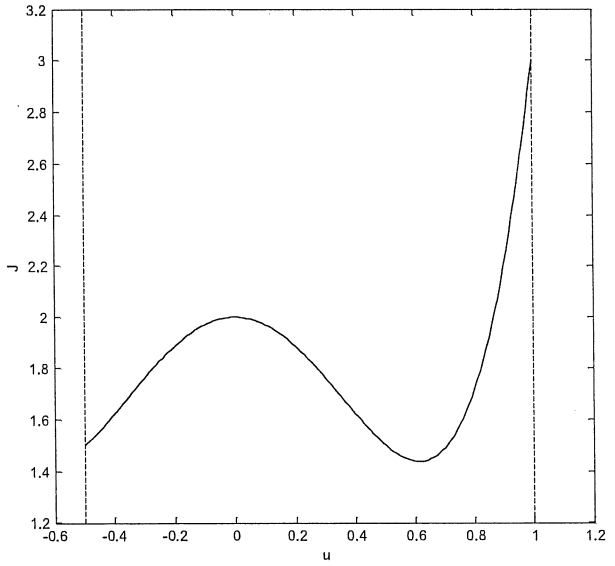


Fig. 4. The nonconvex cost function for the first time step for the single-input-single-output model. Vertical lines at –0.5 and 1.0 show constraints on the input.

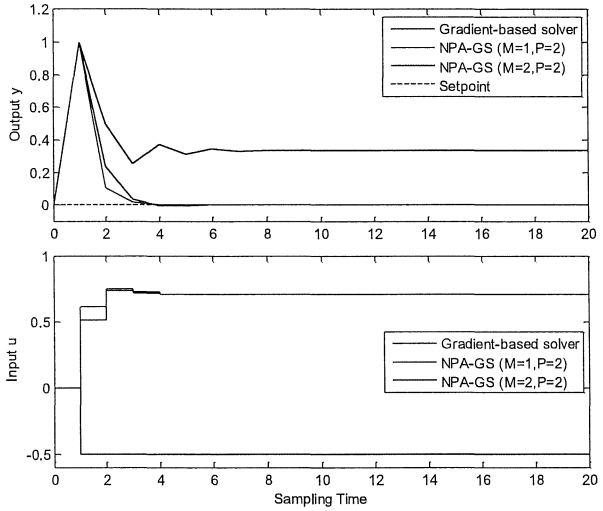


Fig. 5. Closed-loop results for the single-input-single-output model using the gradient-based solver and the NPA-GS. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

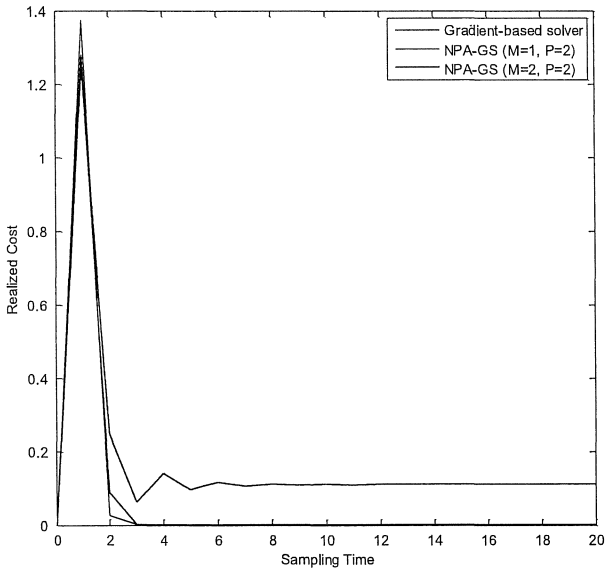


Fig. 6. Realized cost for the single-input-single-output example using gradient-based solver and NPA-GS. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

cost using the NPA-GS for $M=2$ and $P=2$ is lower than that using the solver in Sriniwas and Arkun (1997) where the cost is shown pictorially and just for the first and third time steps, the cost is approximately 1.5 and 0.4, respectively. The realized cost which is the actual cost incurred using only the input used and its corresponding output is shown for each sampling time (time step) in Fig. 6. The cost using the NPA-GS eventually goes down to zero whereas the cost using the gradient-based solver is higher as the controller using the gradient-based solver is unable to follow the setpoint.

The expected number of iterations calculated using (12) and the number of iterations taken by the NPA to reach $d_{max}=8$ to solve the NMPC problem ($M=1$ and $P=2$) for each time step is shown in Fig. 7. For (12), $p=2/3$ (heuristically calculated using $p=M_p/(M_p+1)$) is used and the expected number of iterations

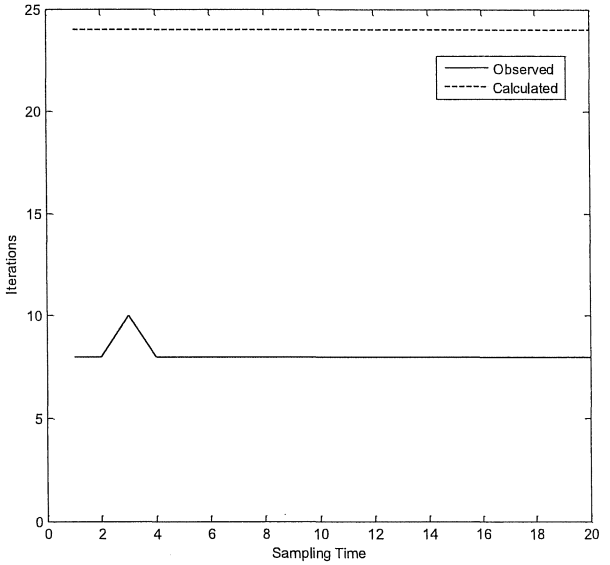


Fig. 7. Number of iterations taken by the NPA to solve the NMPC problem with $M=1$ and $P=2$ for each time step for the single-input-single-output model.

Table 1
Solution quality for NPA-GS for the single-input-single-output example.

	$M=1, P=2$	$M=2, P=2$
MAE	4.802×10^{-9}	2.037×10^{-5}
SE	2.470×10^{-8}	6.443×10^{-5}

comes out to be 24. As mentioned in Section 2.4, the number of iterations taken by the NPA to reach maximum depth $d_{max}=8$ is between 8 and 24.

The solution quality given in Section 2.8 is calculated by running the NPA-GS 10 times. The mean absolute errors and the standard errors for the first sampling time are given in Table 1. It can be seen from Table 1 that the mean absolute errors are very low (virtually zero) for the NPA-GS indicating the high quality of the solution.

Example 2. CSTR with Vanne de Vusse Reactions

The continuously stirred tank reactor (CSTR) with Vanne de Vusse reactions (e.g., Lee, Kaisare, & Lee, 2006; Long et al., 2006; Sistu & Bequette, 1995) is used as the second example. The Vanne de Vusse reactions inside the system are:

$$A \xrightarrow{k_1} B \xrightarrow{k_2} C$$

$$2A \xrightarrow{k_3} D$$

The model for the CSTR based on material balances is given as:

$$\frac{dC_a}{dt} = (F/V)(C_{a0}-C_a)-k_1C_a-k_3C_a^2, \tag{21}$$

$$\frac{dC_b}{dt} = k_1C_a-k_2C_b-(F/V)C_b, \tag{22}$$

where F is the feed flow rate, V is the volume of the CSTR, and C_a and C_b are the concentrations of reactants A and B, respectively. Constants k_1, k_2 and k_3 are the reaction rate constants. C_{a0} is the concentration of A in the feed. The nominal values for k_1, k_2, k_3 and C_{a0} are given in Table 2.

Table 2
Nominal parameter values for CSTR.

Parameter	Description	Value
K_1	Reaction rate (h^{-1})	50
K_2	Reaction rate (h^{-1})	100
K_3	Reaction rate ($L\,gmol^{-1}\,h^{-1}$)	10
C_{a0}	Feed concentration of A ($gmol\,L^{-1}$)	10

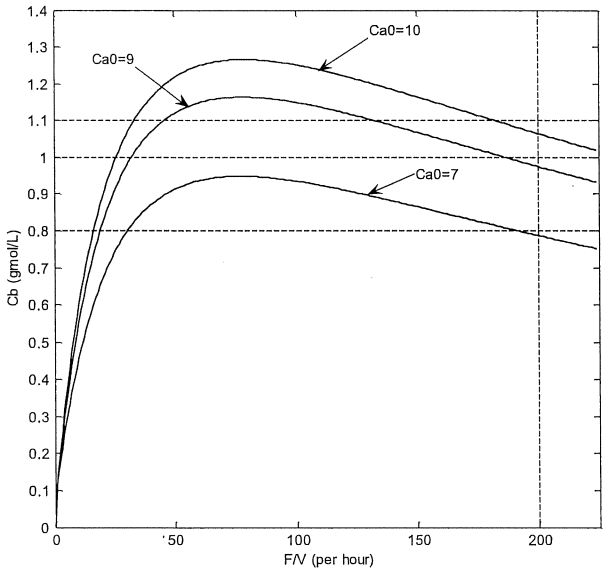


Fig. 8. Steady state output concentration C_b versus steady state input F/V of the CSTR for different values of input feed concentrations C_{a0}. The vertical line at $200\ h^{-1}$ is an upper bound on F/V. The dashed horizontal lines represent setpoints to be used by the controller later on.

The model is discretized using a sampling time of 0.002 h or 7.2 s. The steady state output concentration C_b versus steady state input F/V of the CSTR for different values of input feed concentrations C_{a0} is shown in Fig. 8. The dashed horizontal lines represent setpoints to be used in NMPC later. The points of intersection between the steady state input/output curves and the setpoints give the corresponding steady state input values required to reach the setpoint.

The cost function used in the controller is:

$$J=\sum_{i=1}^{P-1} Q(i)(\hat{y}(k+i)-y_{sp}(k+i))^2+Q(P)(\hat{y}(k+P)-y_{sp}(k+P))^2 + \sum_{i=0}^{M-1} S(k)\Delta\hat{u}(k+i)^2, \quad (23)$$

s.t.

$$0\le\Delta\hat{u}(k+i)\le 200 \quad \text{for } 0\le i\le M-1, \quad (24)$$

$$0\le\hat{u}(k+i)\le 200 \quad \text{for } 0\le i\le M-1. \quad (25)$$

There are no output constraints. The input u is F/V and the output y is the concentration C_b. The input weight S is 0.0008 for all i, output weight $Q=10$ for $1\le i\le P-1$ and the terminal error weight is $Q(P)=10{,}000$. The control and prediction horizons are $M=15$ and $P=30$, respectively. The initial conditions are: $u=181\ h^{-1}$, $C_b=1.1\ gmol\,L^{-1}$ and $C_a=6.18\ gmol\,L^{-1}$. The initial input feed concentration C_{a0} is $10\ gmol\,L^{-1}$ which is changed to $9\ gmol\,L^{-1}$ and $7\ gmol\,L^{-1}$ at 0.2 h and 0.35 h, respectively. The initial setpoint y_{sp} is $1.1\ gmol\,L^{-1}$ which is changed to $1.0\ gmol\,L^{-1}$ and $0.8\ gmol\,L^{-1}$ at 0.1 h and 0.5 h, respectively.

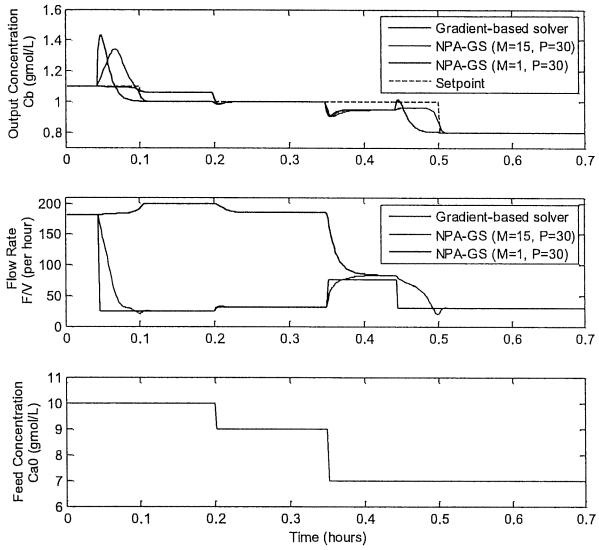


Fig. 9. Closed-loop results for the CSTR using the gradient-based solver and the NPA-GS. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The NMPC is applied to the process for a simulation time of 350 samples or 0.7 h. For NPA, $d_{init}=0$ and $d_{max}=8$ for all input moves, $M_p=4$, $n_0=2$, $n_1=2$ and $n_2, n_3, \ldots, n_{14}=1$.

The closed loop performance using gradient-based solver and the NPA-GS are shown in Fig. 9. Initially both controllers follow the setpoint of $1.1\ gmol\,L^{-1}$ with $u=181\ h^{-1}$. But when the setpoint is changed to $1.0\ gmol\,L^{-1}$, the gradient-based solver with a starting point of $181\ h^{-1}$ starts moving the input to $200\ h^{-1}$, which from Fig. 8 can only take the output to $1.062\ gmol\,L^{-1}$ and is unable to follow the setpoint. The controller using the NPA-GS is able to move the input from $181\ h^{-1}$ towards $25\ h^{-1}$ which causes the output concentration to temporarily transition away before coming back to the setpoint. Both controllers are able to track the setpoint when C_{a0} is disturbed from 10 to $9\ gmol\,L^{-1}$. When C_{a0} is disturbed from 9 to $7\ gmol\,L^{-1}$, both controllers are unable to track the setpoint of $1.0\ gmol\,L^{-1}$. This is because a value of $C_b=1.0\ gmol\,L^{-1}$ cannot be achieved for $C_{a0}=7\ gmol\,L^{-1}$ as shown in Fig. 8. Both controllers are able to track when the setpoint is changed to $0.8\ gmol\,L^{-1}$. Long et al. (2006) used a convex enveloping technique along with branch-and-bound for global optimization to control the CSTR. For comparison with results by Long et al. (2006), the NMPC problem (23)–(25) is also solved using the NPA-GS with control and prediction horizons $M=1$ and $P=30$, respectively and the results are shown in Fig. 9. The NPA-GS is seen to track all setpoints except for $1.0\ gmol\,L^{-1}$ when C_{a0} is $10\ gmol\,L^{-1}$. The NPA-GS is seen to respond to change in setpoints earlier compared the global optimization algorithm used by Long et al. (2006). Setpoint tracking from the local solution technique (MINOS version 5.51) used by Long et al. (2006) is similar to the results using the gradient-based solver in this paper.

The total cost computed for $M=15$ and $P=30$ using (23) is 4.4154×10^3 for the gradient-based solver and 1.4427×10^3 for the NPA-GS. For $M=1$ and $P=30$, the total cost using (23) for the NPA-GS is 1.5195×10^3. For comparison of numerical values of cost, the results in Long et al. (2006) could not be reproduced as the linearized convex equivalent of the control problem is not given. For a sampling time of 7.2 s and a total simulation time of 0.7 h, a total of 350 problems are solved. Fig. 10 shows the time taken by the gradient-based solver and the NPA-GS to compute

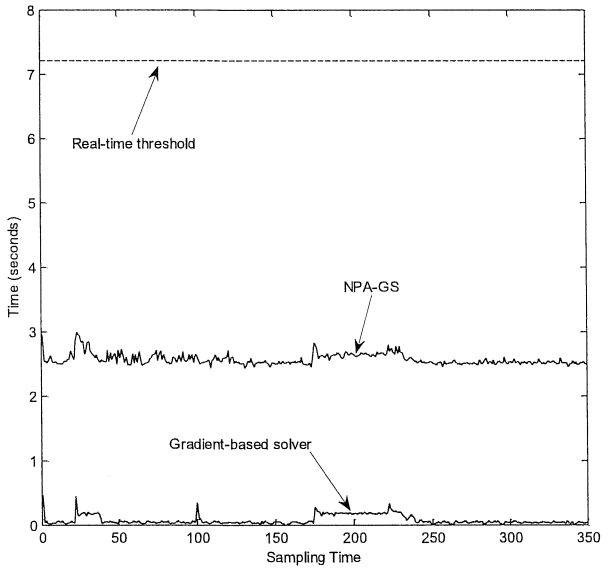


Fig. 10. Time taken by the gradient-based solver and the NPA-GS to solve the optimization problem at each sampling time $M=15$ and $P=30$.

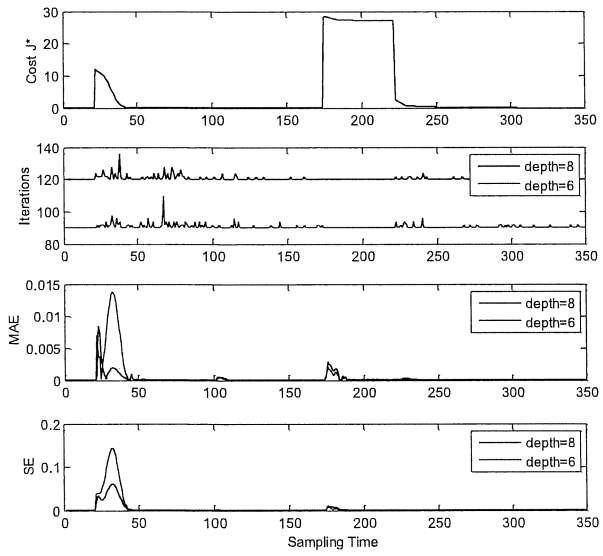


Fig. 11. Solution quality for the CSTR problem with $M=15$ and $P=30$. The cost J^*, number of iterations, mean absolute error and standard error are shown for the NPA-GS to solve the optimization problem at each sampling time. The effect of allowing the algorithm to reach different depths on solution quality is shown. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the solution for each sampling time for $M=15$ and $P=30$. The real-time threshold for this process is 7.2 s.

Fig. 11 shows the solution quality of the NPA-GS for the controller with $M=15$ and $P=30$. A total of 10 Monte Carlo simulations were performed to calculate the solution quality. To give a better insight of solution quality, the number of iterations taken by the algorithm is also shown. To see the effect on solution quality by only allowing the algorithm to reach a depth (for example due to a slower computer) smaller than the maximum depth within the sampling time, the algorithm was allowed to reach depths 8 and 6 on separate runs. To reach the maximum depth 8, the number of iterations in each sampling time is between 120 and 200. This is in agreement with Section 2.4 as $d_{0,max}+\cdots+d_{14,max}=15\times 8=120$ and $(d_{0,max}+\cdots+d_{14,max})/(2p-1)=200$ (from (13) with a heuristic $p=M_p/(M_p+1)=4/5$). It can be seen that the mean absolute error with depth 8 is negligible compared to the actual cost. Also the standard error of the costs from the Monte Carlo simulations is very small indicating a very small dispersion around the true value. To reach depth 6, which is less than the maximum depth, the number of iterations is between $15\times 6=90$ and $15\times 6/(2\times 4/5-1)=150$. The effect of only allowing the algorithm to reach a depth smaller than the maximum depth is seen as the mean absolute error and the standard error are now higher. But the overall solution quality is still quite high as the MAE and the SE are still negligible compared to the actual cost.

Example 3. Bioreactor

The bioreactor (e.g., Saha, Krishnan, Rao, & Patwardhan, 2004; Patwardhan & Madhavan, 1993; Amrit & Saha, 2007), also known as the continuous fermenter, is used as the third example. The first principle model for this process is given as:

$$\frac{dX}{dt}=-DX+\mu X, \tag{26}$$

$$\frac{dS}{dt}=D(S_f-S)-\frac{1}{Y_{X/S}}\mu X, \tag{27}$$

$$\frac{dP}{dt}=-DP+(\alpha\mu+\beta)X, \tag{28}$$

where P represents the product concentration, X is the biomass concentration, S is the substrate concentration, D is the dilution rate and S_f is the input feed substrate concentration. Constants α and β are yield parameters for the product, $Y_{X/S}$ is the cell-mass yield and parameter μ represents the specific growth rate and is given by:

$$\mu=\frac{\mu_m(1-\frac{P}{P_m})S}{K_m+S+\frac{S^2}{K_i}}, \tag{29}$$

where P_m and K_m are the product and substrate saturation constants, respectively, μ_m is the maximum specific growth rate and K_i is the substrate inhibition constant. The nominal values of the model parameters are given in Table 3. The model is discretized with a sampling time of 1 h. The steady state output concentration P versus steady state input S_f, and the steady state biomass concentration X versus steady state input D of the bioreactor is shown in Fig. 12.

The control objective is to control the outputs P and X using the inputs S_f and D while enforcing boundary constraints on P, X and state S. The cost function to be minimized is:

$$\begin{aligned}J=&\sum_{i=1}^{19}Q(i)(\hat{y}(k+i)-y_{sp}(k+i))^2+Q(20)(\hat{y}(k+20)-y_{sp}(k+20))^2\\&+\sum_{i=0}^{19}10\Delta\hat{S}_f(k+i)^2+\sum_{i=0}^{19}10^4\Delta\hat{D}(k+i)^2+R_P\hat{\varepsilon}_{P,L}(k)+R_P\hat{\varepsilon}_{P,U}(k)\\&+R_X\hat{\varepsilon}_{X,L}(k)+R_X\hat{\varepsilon}_{X,U}(k)+R_S\hat{\varepsilon}_{S,L}(k)+R_S\hat{\varepsilon}_{S,U}(k),\end{aligned} \tag{30}$$

Table 3
Nominal parameter values for the bioreactor.

Parameter	Description	Value
$Y_{X/S}$	Cell-mass yield (g/g)	0.4
A	Product yield parameter (g/g)	2.2
B	Product yield parameter (h^{-1})	0.2
μ_m	Maximum specific growth rate (h^{-1})	0.48
P_m	Product saturation constant (g/L)	50
K_m	Substrate saturation constant (g/L)	1.2
K_i	Substrate inhibition constant (g/L)	22

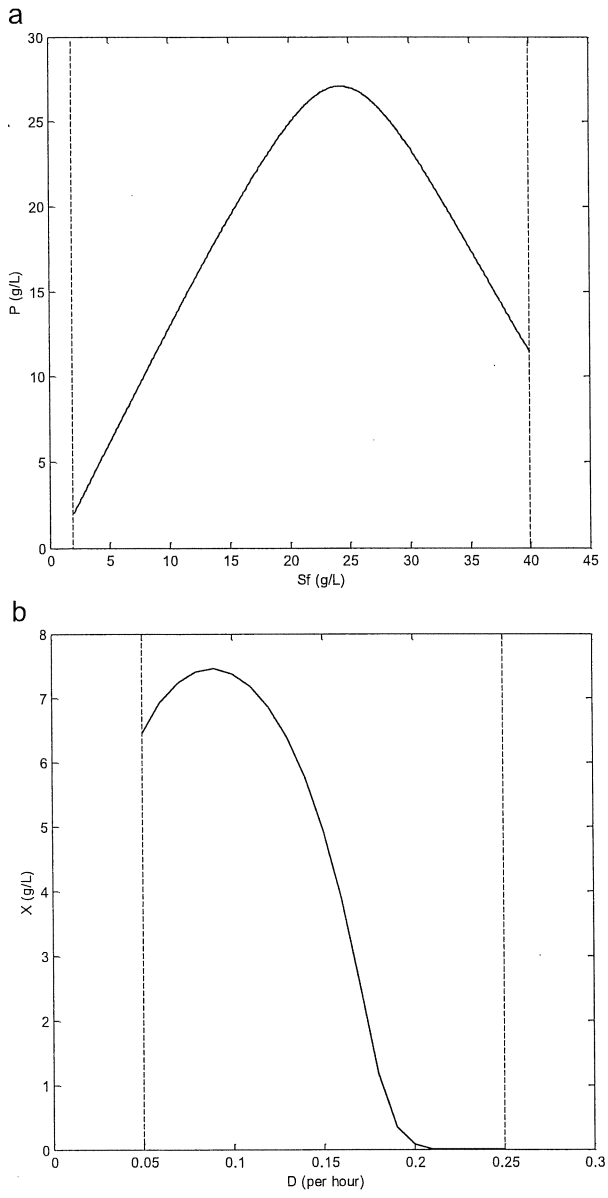


Fig. 12. (a) Steady state output concentration P versus steady state input S_f of the bioreactor for a fixed dilution rate $D=0.15\ \mathrm{h}^{-1}$ with a lower and upper bound of 2 g/L and 40 g/L on S_f, respectively. (b) Steady state biomass concentration X versus steady state input D of the bioreactor for a fixed substrate feed concentration $S_f=25$ g/L with a lower and upper bound of $0.05\ \mathrm{h}^{-1}$ and $0.25\ \mathrm{h}^{-1}$ on D, respectively.

s.t.

$$2 \le \Delta\hat{S}_f(k+i) \le 40 \quad \text{for } 0 \le i \le 19, \tag{31}$$

$$2 \le \hat{S}_f(k+i) \le 40 \quad \text{for } 0 \le i \le 19, \tag{32}$$

$$0.05 \le \Delta\hat{D}(k+i) \le 0.25 \quad \text{for } 0 \le i \le 19, \tag{33}$$

$$0.05 \le \hat{D}(k+i) \le 0.25 \quad \text{for } 0 \le i \le 19, \tag{34}$$

$$\hat{P}(k+i) - \hat{\varepsilon}_{P,U} \le 30 \quad \text{for } 0 \le i \le 20, \tag{35}$$

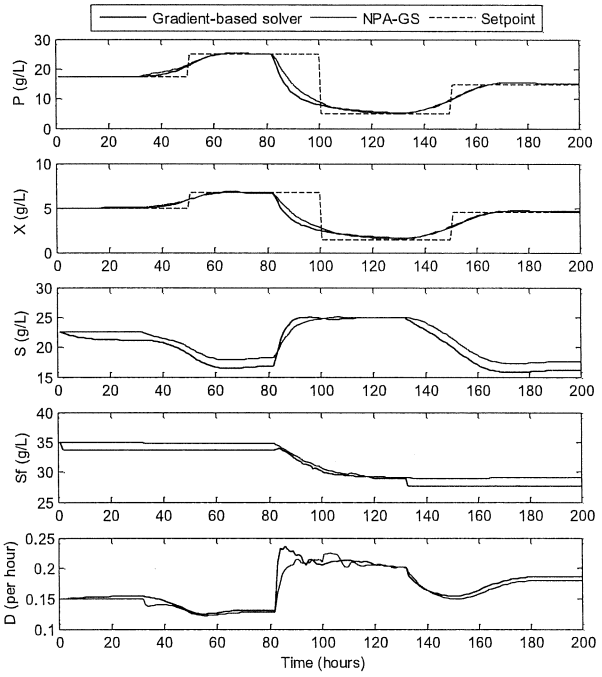


Fig. 13. Closed-loop results for the bioreactor using the gradient-based solver and the NPA-GS. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$-\hat{P}(k+i) - \hat{\varepsilon}_{P,L} \le 0 \quad \text{for } 0 \le i \le 20, \tag{36}$$

$$\hat{X}(k+i) - \hat{\varepsilon}_{X,U} \le 8 \quad \text{for } 0 \le i \le 20, \tag{37}$$

$$-\hat{X}(k+i) - \hat{\varepsilon}_{X,L} \le 0 \quad \text{for } 0 \le i \le 20, \tag{38}$$

$$\hat{S}(k+i) - \hat{\varepsilon}_{S,U} \le 25 \quad \text{for } 0 \le i \le 20, \tag{39}$$

$$-\hat{S}(k+i) - \hat{\varepsilon}_{S,L} \le 0 \quad \text{for } 0 \le i \le 20, \tag{40}$$

$$\hat{\varepsilon}_{P,L}, \hat{\varepsilon}_{P,U}, \hat{\varepsilon}_{X,L}, \hat{\varepsilon}_{X,U}, \hat{\varepsilon}_{S,L}, \hat{\varepsilon}_{S,U} \ge 0. \tag{41}$$

The input vector u is (S_f, D) and the output vector y is (P, X). The input weights are (10, 10^4) for all i, the output weights are $Q=(0.012, 0.0012)$ for $1 \le i \le 19$ and the terminal error weights are $Q(20)=(4200, 0.024)$ and the weights for the exact penalty term are $R_P=R_X=R_S=5\times10^6$. The initial conditions are: $X=4.949$ g/L, $S=22.63$ g/L, $P=17.49$ g/L, $D=0.15\ \mathrm{h}^{-1}$ and $S_f=35$ g/L. The NMPC is applied to the process for a simulation time of 200 samples or 200 h. The initial setpoint y_{sp} is (17.49, 4.95) which is changed to (25, 6.73), (5, 1.48) and (15, 4.55) at 50 h, 100 h and 150 h, respectively. For the NPA, $d_{init}=5$, $d_{max}=10$ and $M_p=4$ for all input moves, $n_0=2$, $n_1=2$ and n_2, n_3, …, $n_{19}=1$. Fig. 13 shows the closed loop performance of the gradient-based solver and the NPA-GS. Both controllers follow the setpoints (17.49, 4.95), (25, 6.73) and (15, 4.55). But when the setpoint is changed from (25, 6.73) to (5, 1.48), both controllers cannot bring the biomass concentration X to its setpoint value. This is because the substrate concentration S has reached its upper bound value. The Optimization Toolbox™ for MATLAB® has been used by Saha et al. (2004) and Patwardhan and Madhavan (1993) to control the process without output or state constraints to only one setpoint (25, 7.3) compared to a variety of setpoints in Fig. 13.

The total simulation cost using (30) is 1721.2 using the gradient-based solver and is 961.8 using the NPA-GS. For comparison of numerical values of cost, the results in Saha et al.

(2004) and Patwardhan and Madhavan (1993) could not be reproduced as the respective models for the bioreactor used in their techniques are not completely given.

4. Conclusions

The stochastic Nested Partitions Algorithm for global optimization has been adapted to solve the nonconvex optimization problems in NMPC. To adapt, a new partitioning scheme has been developed for the NPA. The time the NPA takes to reach maximum depth has been improved by using a good starting point to reduce the size of the initial most promising region and hence increasing the starting depth. Due to its discrete sampling nature, the NPA converges to a point in the vicinity of the global minimum, a gradient-based algorithm has been used with the NPA to overcome this. The mean absolute error along with standard error has been used as a measure of solution quality. The NPA-GS gives a solution that is feasible and provides closed loop stability. The NPA-GS has been implemented and used in NMPC to control different processes. The NPA-GS has been shown to better track the setpoints compared to solutions from a gradient-based solver such as SQP that tends to converge to a local minimum. The NPA is shown to work fast enough for real-time control of the examples considered in this paper. The solution quality presented in Section 2.7 is calculated offline. For future work new techniques can be developed to compute the solution quality online.

Acknowledgments

The authors are grateful to Carl H. Neuschaefer at Alstom Power Inc. in Windsor, CT, USA for the support for this research. The authors would also like to acknowledge Dr. Leyuan Shi at University of Wisconsin-Madison, WI, USA for the useful insights on the Nested Partitions Algorithm.

References

Aggelogiannaki, E., & Sarimveis, H. (2007). A simulated annealing algorithm for prioritized multiobjective optimization—Implementation in an adaptive model predictive control configuration. *IEEE Transactions on Systems, Man, and Cybernetics*, *37*(4), 902–915.

Al-Duwaish, H., & Naeem, W. (2001). Nonlinear model predictive control of Hammerstein and Wiener models using genetic algorithms. In *Proceedings of the IEEE international conference on control applications*. Mexico City, Mexico.

Amrit, R., & Saha, P. (2007). Stochastic identification of bioreactor process exhibiting input multiplicity. *Bioprocess and Biosystems Engineering*, *30*, 165–172.

Banga, J. R., Irizarry-Rivera, R., & Seider, W. D. (1998). Stochastic optimization for optimal and model-predictive control. *Computers & Chemical Engineering*, *22*(4/5), 603–612.

Banga, J. R., & Seider, W. D. (1996). Global optimization of chemical processes using stochastic algorithms. In: C. Floudas, & P. Pardalos (Eds.), *State of the art in global optimization* (pp. 563–584). Dordrecht: Kluwer Academic Publishers.

Baptista, L. F., Sousa, J. M., & Sá da Costa, J. M. G. (2001). Fuzzy predictive algorithms applied to real-time force control. *Control Engineering Practice*, *9*, 411–423.

Bemporad, A., & Filippi, C. (2001). Suboptimal explicit MPC via approximate multiparametric quadratic programming. In *Proceedings of the IEEE conference on decision and control*. Orlando FL, USA.

Bemporad, A., & Filippi, C. (2006). An algorithm for approximate multiparametric convex programming. *Computational Optimization and Applications*, *35*, 87–108.

Betts, J. T., & Huffman, W. P. (1992). Application of sparse nonlinear programming to trajectory optimization. *Journal of Guidance, Control, and Dynamics*, *15*(1), 198–206.

Cagienard, R., Grieder, P., Kerrigan, E. C., & Morari, M. (2007). Move blocking strategies in receding horizon control. *Journal of Process Control*, *17*, 563–570.

Camacho, E. F., & Bordons, C. (2004). *Model predictive control* (2nd ed.). London: Springer-Verlag.

Cannon, M. (2004). Efficient nonlinear model predictive control algorithms. *Annual Reviews in Control*, *28*, 229–237.

Clarke, D. W., Mohtad, C., & Tuffs, P. S. (1987). Generalized predictive control—Part I. The basic algorithm. *Automatica*, *23*(2), 137–148.

Diehl, M., Bock, H. G., Schlöder, J. P., Findeisen, R., Nagy, Z., & Allgöwer, F. (2002). Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, *12*, 577–585.

Fleming, P. J., & Purshouse, R. C. (2002). Evolutionary algorithms in control systems engineering: A survey. *Control Engineering Practice*, *10*, 1223–1241.

Gilbert, E. G., & Tan, K. T. (1991). Linear systems with state and control constraints: The theory and application of maximal output admissible sets. *IEEE Transactions on Automatic Control*, *36*(9), 1008–1020.

Gondhalekar, R., Imura, J., & Kashima, K. (2009). Controlled invariant feasibility —A general approach to enforcing strong feasibility in MPC applied to move-blocking. *Automatica*, *45*, 2869–2875.

Havlena, V., & Findejs, J. (2005). Application of model predictive control to advanced combustion control. *Control Engineering Practice*, *13*, 671–680.

Henson, M. A. (1998). Nonlinear model predictive control: current status and future directions. *Computers & Chemical Engineering*, *23*, 187–202.

Ingber, L. (1993). Simulated annealing: Practice versus theory. *Mathematical and Computer Modelling*, *18*(11), 29–57.

Johansen, T. A. (2004). Approximate explicit receding horizon control of constrained nonlinear systems. *Automatica*, *40*(2), 293–300.

Johansen, T. A., & Grancharova, A. (2003). Approximate explicit constrained linear model predictive control via orthogonal search tree. *IEEE Transactions on Automatic Control*, *48*(5), 810–815.

Kawathekar, R., & Riggs, J. B. (2007). Nonlinear model predictive control of a reactive distillation column. *Control Engineering Practice*, *15*, 231–239.

Kerrigan, E.C., & Maciejowski, J.M. (2000). Soft constraints and exact penalty functions in model predictive control. In *Proceedings of the UKACC international conference*. Cambridge, UK.

Kittisupakorn, P., Thitiyasook, P., Hussain, M. A., & Daosud, W. (2009). Neural network based model predictive control for a steel pickling process. *Journal of Process Control*, *19*, 579–590.

Lee, J. M., Kaisare, N. S., & Lee, J. H. (2006). Choice of approximator and design of penalty function for an approximate dynamic programming based control approach. *Journal of Process Control*, *16*, 135–156.

Long, C. E., Polisetty, P. K., & Gatzke, E. P. (2006). Nonlinear model predictive control using deterministic global optimization. *Journal of Process Control*, *16*, 635–643.

Martínez, M., Senent, J. S., & Blasco, X. (1998).). Generalized predictive control using genetic algorithms (GAGPC). *Engineering Applications of Artificial Intelligence*, *11*, 355–367.

Martinsen, F., Biegler, L. T., & Foss, B. A. (2004). A new optimization algorithm with application to nonlinear MPC. *Journal of Process Control*, *14*, 853–865.

Mayne, D. Q., Rawlings, J. B., Rao, C. V., & Scokaert, P. O. M. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, *36*, 789–814.

Mu, J., Rees, D., & Liu, G. P. (2005). Advanced controller design for aircraft gas turbine engines. *Control Engineering Practice*, *13*, 1001–1015.

Norquay, S. J., Palazoglu, A., & Romagnoli, J. A. (1999). Application of Wiener model predictive control (WMPC) to an industrial C2-splitter. *Journal of Process Control*, *9*, 461–473.

Oliveira, G. H. C., Amaral, W. C., Favier, G., & Dumont, G. A. (2000). Constrained robust predictive controller for uncertain processes modeled by orthonormal series functions. *Automatica*, *36*, 563–571.

Onnen, C., Babuška, R., Kaymak, U., Sousa, J. M., Verbruggen, H. B., & Isermann, R. (1997). Genetic algorithms for optimization in predictive control. *Control Engineering Practice*, *5*(10), 1363–1372.

Patwardhan, S. C., & Madhavan, K. P. (1993). Nonlinear model predictive control using second-order model approximation. *Industrial & Engineering Chemistry Research*, *32*(2), 334–344.

Piché, S., Sayyar-Rodsari, B., Johnson, D., & Gerules, M. (2000). Nonlinear model predictive control using neural networks. *IEEE Control Systems Magazine*, *20*(3), 53–62.

Potočnik, P., & Grabec, I. (2002). Nonlinear model predictive control of a cutting process. *Neurocomputing*, *43*, 107–126.

Qin, S. J., & Badgwell, T. A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, *11*, 733–764.

Ricker, N.L., Subrahmanian, T., & Sim, T. (1989). Case studies of model predictive control in pulp and paper production. In *Proceedings of the 1988 IFAC workshop—model based process control*. Oxford: Pergamon Press.

Saha, P., Krishnan, S. H., Rao, V. S. R., & Patwardhan, S. C. (2004). Modeling and predictive control of MIMO nonlinear systems using Wiener–Laguerre models. *Chemical Engineering Communications*, *191*, 1083–1119.

Santos, L. O., Afonso, P. A. F. N.A., Castro, J. A. A. M., Oliveira, N. M. C., & Biegler, L. T. (2001). On-line implementation of nonlinear MPC: an experimental case study. *Control Engineering Practice*, *9*, 847–857.

Santos, L. O., Biegler, L. T., & Castro, J. A. A. M. (2008). A tool to analyze robust stability for constrained nonlinear MPC. *Journal of Process Control*, *18*, 383–390.

Sarimveis, H., & Bafas, G. (2003). Fuzzy model predictive control of non-linear processes using genetic algorithms. *Fuzzy Sets and Systems*, *139*, 59–80.

Scokaert, P. O. M., & Rawlings, J. B. (1999). Feasibility issues in linear model predictive control. *AIChE Journal*, *45*(8), 1649–1659.

Seki, H., Ogawa, M., Ooyama, S., Akamatsu, K., Ohshima, M., & Yang, W. (2001). Industrial application of a nonlinear model predictive control to polymerization reactors. *Control Engineering Practice*, *9*, 819–828.

Shi, L., & Ólafsson, S. (2000). Nested partitions method for global optimization. *Operations Research*, *48*(3), 390–407.

Shi, L., & Ólafsson, S. (2009). *Nested partitions method, theory and applications*. New York: Springer.

Sistu, P. B., & Bequette, B. W. (1995). Model predictive control of processes with input multiplicities. *Chemical Engineering Science*, *50*(6), 921–936.

Sousa, J. M., Babuška, R., & Verbruggen, H. B. (1997). Fuzzy predictive control applied to an air-conditioning system. *Control Engineering Practice*, *5*(10), 1395–1406.

Sriniwas, G. R., & Arkun, Y. (1997). A global solution to the nonlinear model predictive control algorithms using polynomial ARX models. *Computers & Chemical Engineering*, *21*(4), 431–439.

Weesakul, B. (1961). The random walk between a reflecting and an absorbing barrier. *The Annals of Mathematical Statistics*, *32*(3), 765–769.

Yokoyama, N., Suzuki, S., & Tsuchiya, T. (2008). Convergence acceleration of direct trajectory optimization using novel hessian calculation methods. *Journal of Optimization Theory and Applications*, *136*(3), 297–319.

Zanin, A. C., Tvrzská de Gouvêa, M., & Odloak, D. (2002). Integrating real-time optimization into the model predictive controller of the FCC system. *Control Engineering Practice*, *10*, 819–831.

Zhao, H., Guiver, J., Neelakantan, R., & Biegler, L. T. (2001). A nonlinear industrial model predictive controller using integrated PLS and neural net state-space model. *Control Engineering Practice*, *9*, 125–133.

Zheng, A., & Morari, M. (1995). Stability of model predictive control with mixed constraints. *IEEE Transactions on Automatic Control*, *40*(10), 1818–1823.