Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Dynamic Safety Constraints by Scenario Based Economic Model Predictive Control [⋆]

Torstein I. Bø [*] **Tor Arne Johansen** [*]

[*] *Center for Autonomous Marine Operations and Systems,*
Department of Engineering Cybernetics,
Norwegian University of Science and Technology, 7491 Trondheim,
Norway (e-mail: torstein.bo@itk.ntnu.no,
tor.arne.johansen@itk.ntnu.no).

Abstract: This paper studies use of scenario based model predictive control (MPC) to control a plant into a fault-tolerant state set. For some systems it is not always sufficient that the controller is reconfigured after a fault, it is also necessary that the system is in a given state set when the fault occurs in order to guarantee that the constraints are not violated. This is often achieved by using static safety constraints. However, it may be hard to find such constraints, and they may be conservative. This article presents a method for implementing dynamic safety constraints based on fault scenarios with an economic MPC. The controller is tested using closed-loop simulations.

Keywords: Fault-tolerant control, predictive control, scenario based control.

1. INTRODUCTION

There is a need for automatic handling of faults in many industrial control systems. These faults can be due to actuator faults, sensor faults, external faults, or internal faults. There are also often constraints on the system to make sure that the system is safe. However, it can take some time before the controller is fully reconfigured and the system recovers after a fault. Conservative safety limits are therefore sometimes used to make sure that the system is safe also during the transients after the fault. However, instead of using static constraints for such problems, we suggest to include the scenario and safety requirements in the controller explicitly through dynamic constraints.

To establish a fault-tolerant controller, model predictive control (MPC) will be used. A significant effort has already been made on fault-tolerant control using MPC. Maciejowski (1999) studies some of the properties inherent in standard linear model predictive control. Pranatyasto and Qin (2001) studies fault detection and identification for a system controlled with MPC.

In this paper, multiple scenarios are used internally in the controller to make sure that the controller is able to recover the system after any faults characterized by the scenarios. In robust model predictive control, scenarios are commonly used in a combination with MPC.

Scenarios have earlier been used to incorporate disturbances and model uncertainties (Bernardini and Bemporad, 2009; Calafiore and Fagiano, 2013; Schildbach et al., 2013). It has been proposed to use feedback to avoid conservative estimates when multiple scenarios are used, both for linear models (Scokaert and Mayne, 1998) and nonlinear Limon et al. (2009)

Scenario-based model predictive control has also been suggested to be used in for optimization of hedge options (Bemporad et al., 2012), for scheduling of batch processes (Bonfill et al., 2008), and scheduling of emergency vehicles (Goodwin and Medioli, 2013).

There has been some studies on transients of the plant when reconfiguring controllers due to faults. Kovâcshâzy et al. (2001) investigates responses due to reconfiguration of the controller. For faults which can be predicted, Lao et al. (2013) have suggested to use MPC to make a smooth accommodation of the fault. Blanke et al. (2006) suggest to use back calculation and a progressive accommodation scheme to achieve new LQR-gains during reconfiguration.

Another approach to control a plant to a safe set is to use backward reachable set to calculate the fault-tolerant set (Gillula et al., 2011). Torrisi and Bemporad (2001) suggest a method for validating that a controller can avoid unsafe sets for linear hybrid systems using reachability analysis. A similar study is done for nonlinear hybrid system using barrier certificates (Prajna et al., 2007). Coogan and Arcak (2012) presents a method for selecting switching rules for a hybrid system, such that the state variables avoid an unsafe set.

The method presented in the present paper is a variant of the method presented in Bernardini and Bemporad (2009). However, in this paper the models are nonlinear, fault-tolerance and economic objective emphasized, and a deterministic framework is used.

[⋆] The authors of this paper are funded by Design to verification of control systems for safe and energy efficient vessels with hybrid power plants (D2V), where the Research Council of Norway is the main sponsor. The second author is also funded by the Research Council of Norway, Statoil, and DNV through the Center of Excellence AMOS. NFR: 210670/070, 223254/F50.

The present paper presents a method for establishing dynamic safety constraints based on fault scenario and is an extension of Bø and Johansen (2013). This is done by using multiple prediction for each fault scenario in addition to the nominal scenario internally in the MPC.

2. PROBLEM STATEMENT

Consider a plant where we have safety limits and constraints on the control input and the states. In addition there are some fault scenarios which we must be able to handle. The set of scenarios is assumed to be fixed.

We would like to design a controller which makes sure that the plant can be recovered after any of these fault scenarios. This consists of three requirements:

(1) Safe State Set: The controller should drive the system to a state set where it is possible to recover the system if one of the fault scenarios occurs.
(2) Robust Control: The controller should use a control input which is appropriate for the nominal system and the fault system until the fault is detected and the fault is accommodated by the controller.
(3) Detection of fault-preparedness: It should be detected if the system is not fault-prepared.

The fault scenarios of interest in this paper are modeled as change of the dynamics or constraints of the system.

3. MODEL DESCRIPTION

The plant is described with one model for each scenario. The difference equation for the states is of the form:

$$\mathbf{x}(t_{k+1}) = \mathbf{f}(\mathbf{x}(t_k), \mathbf{u}(t_k)) \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^m$ are the state and input vectors. The state and control input vector for the control horizon are given by:

$$\mathbf{X}(t_i) = \left[\mathbf{x}^\top(t_i) \ \ldots \ \mathbf{x}^\top(t_{i+N-1})\right]^\top \tag{2}$$

$$\mathbf{U}(t_i) = \left[\mathbf{u}^\top(t_i) \ \ldots \ \mathbf{u}^\top(t_{i+N-1})\right]^\top \tag{3}$$

where t_i is the time instant at the beginning of the control horizon and N denotes the length of the control horizon. Note that we assume no unknown disturbances, model uncertainties, or measurement noise; however, these effects can be included as scenarios along with faults. The state and control input are constrained by:

$$\mathbf{G}_{non-relaxable}(\mathbf{x}, \mathbf{u}) \leq \mathbf{0} \tag{4a}$$

$$\mathbf{G}_{relaxable}(\mathbf{x}, \mathbf{u}) \leq \mathbf{s} \tag{4b}$$

$$\mathbf{0} \leq \mathbf{s} \tag{4c}$$

where $\mathbf{G}_{non-relaxable}(\mathbf{x}, \mathbf{u}) : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^{n_{cn}}$ and $\mathbf{G}_{relaxable}(\mathbf{x}, \mathbf{u}) : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^{n_{cr}}$. It is assumed that all non-relaxable constraints (hard constraints) are stacked in the first constraints, and relaxable constraints (soft constraints) are stacked in the second. The vector $\mathbf{s}$ contains slack variables, such that $\mathbf{s} = \mathbf{0}$ when the relaxable constraints are satisfied and it has positive elements when the relaxable constraints are relaxed. The cost function is given by:

$$J'\left(\mathbf{x}(t_k), \mathbf{U}(t_k)\right) = \sum_{i=0}^{N-1} l(\mathbf{x}(t_{k+i}), \mathbf{u}(t_{k+i})) \tag{5}$$

where $l(\mathbf{x}, \mathbf{u})$ is the smooth stage cost.

We will use superscript $^{(n)}$ and $^{(fj)}$ to distinguish between the different models and constraints for each scenario, where n denotes the nominal scenario and fj fault scenario number j. The controller may use multiple predictions for the same scenario, where the fault occurs at different times in the prediction horizon. We will use the following notation to distinguish the different predictions at different times for the fault : $\mathbf{x}^{(fj)}(t|t_f = t_j)$ where t_j is the time when the fault occurs. Further, we will use the word *event* for predictions starting from different times, and *scenarios* for predictions with different faults. This means that $\mathbf{x}^{(f1)}(t_3|t_f = t_2)$ is the predicted state at time instant t_3, for the *fault* scenario number 1, and corresponding to the event starting at t_2.

It is assumed that the state variables for the fault scenario dynamics correspond to a subset of the state variables for the nominal scenario dynamics. This means that:

$$n^{(fj)} \leq n^{(n)} \tag{6a}$$

$$m^{(fj)} \leq m^{(n)} \tag{6b}$$

$$\mathbf{x}^{(fj)}(t_f) = E^{(fj)}\mathbf{x}^{(n)}(t_f) \tag{6c}$$

$$\mathbf{u}^{(fj)}(t_f) = F^{(fj)}\mathbf{u}^{(n)}(t_f) \tag{6d}$$

4. FAULT-TOLERANT MPC

Multiple events are used per scenario to achieve the control statement:

(1) Safe State Set: We would like to find a trajectory such that the plant can be recovered from any fault event. In addition predicted trajectories are added to make the trajectory safe. For each scenario, one trajectory is added per time step in the nominal trajectory and each trajectory is starting from the nominal trajectory. In addition, terminal equality constraints are added to achieve stability of the average closed loop cost (which is economical MPCs counterpart to stability of the state variables, (Angeli et al., 2012)) and recursive feasibility. The optimization problem is to find optimal control sequences, such that all trajectories are satisfying all constraints. If this problem is feasible, the controller has found a nominal trajectory which is such that the plant can be recovered if any of the fault scenarios occurs along the trajectory or after the end of the finite horizon. If it is not possible to find such trajectories, the controller will remove the relaxable constraints in the beginning of the trajectories. This may give some trajectories that does not satisfy all the constraints; however, after some time the controller may be able to satisfy all constraints for all scenarios.
(2) Robust Control Input: Fault detection and fault accommodation will often take some time. It is therefore important that the control input is appropriate in the time between the fault occurs and the fault is accommodated in the controller. The fault trajectories mentioned above will therefore be restricted to use the same control input as the nominal trajectory during the first time steps of the fault trajectory.

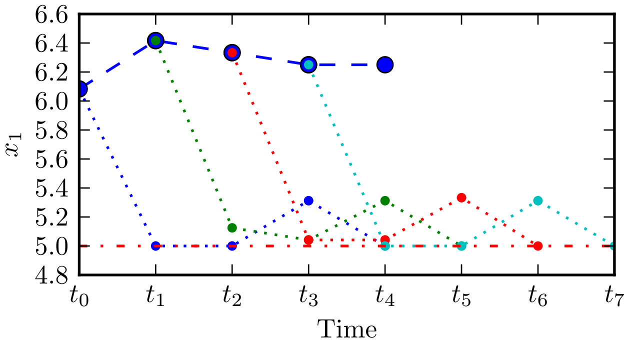

Fig. 1. Possible trajectories for the case which will be presented in Section 7. The dashed blue line is the predicted trajectory of the nominal scenario. The dotted lines are the predicted fault scenarios, starting at $t_f = t_0$, t_1, t_2, t_3. The dotted red line represents the lower limits which all trajectories should be above.

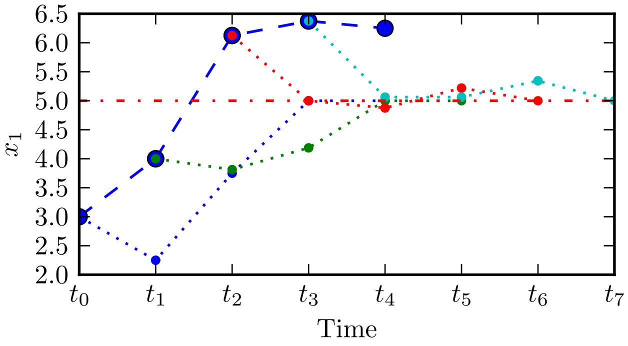

Fig. 2. Nominal and fault trajectory when $N_{relaxed} = 3$. Note that the three first fault predictions can violate the state constraint $x_1 \geq 5$ for the entire horizon and terminal constraint.

(3) Detection of fault-preparedness: The controller will check if some of the constraints was relaxed, if not the plant is prepared for the fault scenarios.

Some possible trajectories are presented in Fig. 1.

We will now present the constraints of the optimization problem. The fault scenario is subject to the constraints. However, the constraints are relaxed only at the $N_{relaxed}$ first steps of the prediction, to make the plant safe as early as possible. In Fig. 2 a set of predictions are shown with $N_{relaxed} = 3$. The constraints for the nominal scenario during the relaxed period, $t_0 \leq t_k \leq t_{N_{relaxed}}$, are:

$$\begin{aligned}&\mathbf{G}^{(n)}_{non-relaxable}(\mathbf{x}^{(n)}(t_k), \mathbf{u}^{(n)}(t_k)) \leq \mathbf{0}\\ &\mathbf{G}^{(n)}_{relaxable}(\mathbf{x}^{(n)}(t_k), \mathbf{u}^{(n)}(t_k)) \leq \mathbf{s}^{(n)}(t_k)\\ &\mathbf{0} \leq \mathbf{s}^{(n)}(t_k)\end{aligned} \tag{7}$$

Since the fault scenarios act as dynamic constraints for the state at the time the simulated fault occurs, the slack variables are introduced for the scenarios starting in the beginning of the horizon, $t_0 \leq t_f \leq t_{N_{relaxed}}$ and $t_f \leq t_k < t_f + t_N$:

$$\begin{aligned}&\mathbf{G}^{(fj)}_{non-relaxable}(\mathbf{x}^{(fj)}(t_k|t_f), \mathbf{u}^{(fj)}(t_k|t_f)) \leq \mathbf{0}\\ &\mathbf{G}^{(fj)}_{relaxable}(\mathbf{x}^{(fj)}(t_k|t_f), \mathbf{u}^{(fj)}(t_k)) \leq \mathbf{s}^{(fj)}(t_k|t_f)\\ &\mathbf{0} \leq \mathbf{s}^{(fj)}(t_k|t_f)\end{aligned} \tag{8}$$

while for the rest of the prediction, all of the constraints should be satisfied: For the nominal scenario, $t_{N_{relaxed}} < t_k < t_N$:

$$\begin{aligned}&\mathbf{G}^{(n)}_{non-relaxable}(\mathbf{x}^{(n)}(t_k), \mathbf{u}^{(n)}(t_k)) \leq \mathbf{0}\\ &\mathbf{G}^{(n)}_{relaxable}(\mathbf{x}^{(n)}(t_k), \mathbf{u}^{(n)}(t_k)) \leq \mathbf{0},\end{aligned} \tag{9}$$

and the fault events, $t_{N_{relaxed}} < t_f < t_N$ and $t_f \leq t_k < t_f + t_N$:

$$\begin{aligned}&\mathbf{G}^{(fj)}_{non-relaxable}(\mathbf{x}^{(fj)}(t_k|t_f), \mathbf{u}^{(fj)}(t_k|t_f)) \leq \mathbf{0}\\ &\mathbf{G}^{(fj)}_{relaxable}(\mathbf{x}^{(fj)}(t_k|t_f), \mathbf{u}^{(fj)}(t_k)) \leq \mathbf{0}\end{aligned} \tag{10}$$

The predicted states are constrained to the dynamics of the system:

$$\begin{aligned}&\mathbf{x}^{(n)}(t_{k+1}) = \mathbf{f}^{(n)}\left(\mathbf{x}^{(n)}(t_k), \mathbf{u}^{(n)}(t_k)\right)\\ &\mathbf{x}^{(fj)}(t_{k+1}|t_f) = \mathbf{f}^{(fj)}\left(\mathbf{x}^{(fj)}(t_k|t_f), \mathbf{u}^{(fj)}(t_k|t_f)\right)\end{aligned} \tag{11}$$

The initial condition of the nominal scenario is $\mathbf{x}(t_0)$ and the initial condition of the fault scenario is the value of the nominal scenario at the initial time of the event:

$$\begin{aligned}&\mathbf{x}^{(n)}(t_0) = \mathbf{x}(t_0)\\ &\mathbf{x}^{(fj)}(t_k|t_f = t_k) = E_j\mathbf{x}^{(n)}(t_k)\end{aligned} \tag{12}$$

Before we continue with terminal constraints, we introduce the definition of an optimal equilibrium.

Definition 1. An optimal equilibrium for fault scenario j, $(\mathbf{x}^{(fj)o}, \mathbf{u}^{(fj)o})$ is any solution to:

$$\begin{aligned}&(\mathbf{x}^{(fj)o}, \mathbf{u}^{(fj)o}) = \underset{\mathbf{x}^{(fj)},\mathbf{u}^{(fj)}}{\operatorname{argmin}}\ l^{(fj)}(\mathbf{x}^{(fj)}, \mathbf{u}^{(fj)})\\ &\text{such that}\\ &\mathbf{x}^{(fj)} = \mathbf{f}^{(fj)}(\mathbf{x}^{(fj)}, \mathbf{u}^{(fj)})\\ &\mathbf{G}^{(fj)}_{relaxable}(\mathbf{x}^{(fj)}, \mathbf{u}^{(fj)}) \leq \mathbf{0}\\ &\mathbf{G}^{(fj)}_{non-relaxable}(\mathbf{x}^{(fj)}, \mathbf{u}^{(fj)}) \leq \mathbf{0}\end{aligned} \tag{13}$$

The optimal steady state cost is $l^{(fj)o} = l^{(fj)}(\mathbf{x}^{(fj)o}, \mathbf{u}^{(fj)o})$. We assume an optimal equilibrium exists and is unique.

To make sure that there exist an infinite safe trajectory prolonging the fault events trajectories, the terminal state is constrained to the optimal equilibrium:

$$\begin{aligned}&\mathbf{x}^{(fj)}(t_N|t_f) = \mathbf{x}^{(fj)o} - \mathbf{s}_N^{-(fj)}(t_f) + \mathbf{s}_N^{+(fj)}(t_f)\\ &\mathbf{0} \leq \mathbf{s}_N^{-(fj)}(t_f)\\ &\mathbf{0} \leq \mathbf{s}_N^{+(fj)}(t_f)\end{aligned} \tag{14}$$

where $\mathbf{s}_N^-$ and $\mathbf{s}_N^+$ are slack variables and $\mathbf{x}^{(fj)o}$ is found by solving (13). Note that the slack variable must be zero to guarantee that the fault event trajectory can be prolonged.

A stability constraint is used for the nominal scenario, both to make sure that the closed loop average cost is stable and to make sure that the control problem is recursively feasible. We start by defining an optimal safe equilibrium.

Definition 2. An optimal safe equilibrium for the nominal scenario, $(\mathbf{x}^{(n)s}, \mathbf{u}^{(n)s})$ is any solution to:

$$\begin{aligned}
&(\mathbf{x}^{(n)s}, \mathbf{u}^{(n)s}) = \underset{\mathbf{x}^{(n)},\mathbf{u}^{(n)}}{\text{argmin}}\ l^{(n)}(\mathbf{x}^{(n)}, \mathbf{u}^{(n)}) \\
&\text{such that} \\
&(\mathbf{x}^{(n)}, \mathbf{u}^{(n)}) \in \mathbb{Z}_{safe} \\
&\mathbf{G}^{(n)}_{relaxable}(\mathbf{x}^{(n)}, \mathbf{u}^{(n)}) \leq \mathbf{0} \\
&\mathbf{G}^{(n)}_{non-relaxable}(\mathbf{x}^{(n)}, \mathbf{u}^{(n)}) \leq \mathbf{0}
\end{aligned} \tag{15}$$

where $\mathbb{Z}_{safe}$ is the set of equilibrium points satisfying $\mathbf{x}^{(n)} = \mathbf{f}^{(n)}(\mathbf{x}^{(n)}, \mathbf{u}^{(n)})$ such that for each fault scenario j there exist a feasible trajectory with the length N:

- starting at $E_j\mathbf{x}^{(n)}$,
- the control input of the fault scenario is $F_j\mathbf{u}^{(n)}$ during the first τ_{robust} samples of the trajectory,
- the constraints (4) are satisfied with $\mathbf{s} = \mathbf{0}$ at every point on the trajectory.

In other words, this is the set of all equilibria the plant can be recovered from. The optimal safe steady state cost is $l^{(n)s} = l^{(n)}(\mathbf{x}^{(n)s}, \mathbf{u}^{(n)s})$.

We assume that there exist a unique optimal safe equilibrium. The terminal constraint for the nominal trajectory is:

$$\mathbf{x}^{(n)}(t_N) = \mathbf{x}^{(n)s} \tag{16}$$

where $\mathbf{x}^{(n)s}$ is found by solving (15).

Remark 3. In the MPC literature it is known that equilibrium terminal constraints often lead to numerical problems and may suffer from small region of attraction. However, the authors are not aware of any result on stability for systems with optimal steady state solution with active constraints unless equality terminal constraint is used. One exception is Grüne (2013), but this approach requires strong assumptions.

In order to allow time for fault detection, identification, and reconfiguration, we require that the control input of the fault scenarios should be the same as the nominal scenario from the occurrence of the fault till τ_{robust} time units after the fault. If $t_f + \tau_{\text{robust}} \geq t_N$ the inputs are constrained to be equal to $\mathbf{u}^{(n)s}$ since we can prolong the nominal trajectory with the pairs $(\mathbf{x}^{(n)s}, \mathbf{u}^{(n)s})$.

$$\begin{aligned}
\mathbf{u}^{(fj)}(t_i|t_f = t_k) &= F_j\mathbf{u}^{(n)}(t_i) \quad && k \leq i < \min(k + \tau_{\text{robust}}, N) \\
\mathbf{u}^{(fj)}(t_i|t_f = t_k) &= F_j\mathbf{u}^{(n)s} && N \leq i < k + \tau_{\text{robust}}
\end{aligned} \tag{17}$$

Penalty functions $g_i(\mathbf{s})$ and $g_N(\mathbf{s}^+, \mathbf{s}^-)$ are added to the cost function to minimize the violation of the constraints.

$$\begin{aligned}
J(\mathbf{x}(t_k), \mathbf{U}(t_k), \mathbf{S}(t_k)) &= J'(\mathbf{x}(t_k), \mathbf{U}(t_k)) \\
&+ \sum_{i=0}^{N-1} g_i(s(t_{k+i})) + g_N(s^-_N(t_{k+N}), s^+_N(t_{k+N}))
\end{aligned} \tag{18}$$

$$\mathbf{S}(t_k) = \left[\mathbf{s}^\top(t_k)\ \ldots\ \mathbf{s}^\top(t_{k+N-1})\ \mathbf{s}^{-\top}_N(t_{k+N})\ \mathbf{s}^{+\top}_N(t_{k+N}))\right]^\top \tag{19}$$

Finally, all optimization variables can be stacked in a vector:

$$\mathcal{U} = \begin{bmatrix}
\mathbf{U}^{(n)} \\
\mathbf{S}^{(n)} \\
\mathbf{U}^{(f1)}(t_f = t_0) \\
\mathbf{S}^{(f1)}(t_f = t_0) \\
\mathbf{U}^{(f1)}(t_f = t_1) \\
\mathbf{S}^{(f1)}(t_f = t_1) \\
\vdots \\
\mathbf{U}^{(f1)}(t_f = t_{N-1}) \\
\mathbf{S}^{(f1)}(t_f = t_{N-1}) \\
\vdots \\
\mathbf{U}^{(fM)}(t_f = t_0) \\
\mathbf{S}^{(fM)}(t_f = t_0) \\
\vdots \\
\mathbf{U}^{(fM)}(t_f = t_{N-1}) \\
\mathbf{S}^{(fM)}(t_f = t_{N-1})
\end{bmatrix} \tag{20}$$

The objective function is:

$$\begin{aligned}
\phi(\mathbf{x}(t_0), \mathcal{U}) &= w^{(n)} J^{(n)}\left(\mathbf{x}^{(n)}(t_0), \mathbf{U}^{(n)}(t_0), \mathbf{S}^{(n)}(t_0)\right) \\
&+ \sum_{j=1}^{M} \sum_{i=0}^{N-1} w_i^{(fj)} \left[\sum_{k=0}^{N} g_k^{(fj)}(\mathbf{s}^{(fj)}(t_k|t_f = t_i))\right. \\
&\left. + g_N^{(fj)}(\mathbf{s}^{-(fj)}(t_f = t_i), \mathbf{s}^{+(fj)}(t_f = t_i))\right]
\end{aligned} \tag{21}$$

where the first term is the cost of the nominal scenario and the second term is the penalty of the fault scenarios. M is the number of fault scenarios. $w^{(n)}$ and $w_i^{(fj)}$ are positive weights for the nominal scenario and fault scenario j. Note that the stage cost of the fault scenarios are not included, since we only want the fault scenarios to be feasible and it is not important that their trajectories are optimal.

To make sure that the plant gets safe as fast as possible, a minimal time approach will be used (Rawlings and Muske, 1993). This means that before solving for the optimal trajectory we solve this problem:

$$N^*_{relaxed} = \arg\min N_{relaxed} \quad \text{s.t. } (7) - (17) \tag{22}$$

This means that we find the shortest time where we need to relax the relaxable constraints to make the problem feasible. However, as noted by Scokaert and Rawlings (1999), this may mean that the controller is not robust to small perturbations.

Remark 4. Vada et al. (2001) has shown that for linear systems where the cost is quadratic, a linear penalty function can be designed to ensure that the problem is feasible as early as possible in the prediction horizon. This means that the optimization problem (22) is always fulfilled without being explicitly solved if such a penalty function is used.

An optimal control sequence is then found, by using $N^*_{relaxed}$:

$$\mathcal{U}^* = \arg\min_{\mathcal{U}} \phi(\mathbf{x}(t_0), \mathcal{U}) \quad \text{s.t. } (7) - (17) \tag{23}$$

The controller will then apply $\kappa(\mathbf{x}(t_0)) = \mathbf{u}^{(n)*}(t_0) \in \mathcal{U}^*$, and the closed loop system is:

$$\mathbf{x}^{(n)}(t_{k+1}) = \mathbf{f}^{(n)}(\mathbf{x}^{(n)}(t_k), \kappa^{(n)}(\mathbf{x}^{(n)}(t_k)) \tag{24}$$

The algorithm can be summarized in three steps:

(1) Find the optimal equilibrium for the fault scenarios and optimal safe equilibrium for the nominal scenario by solving (13) and (15).
(2) Find $N_{relaxed}$ by solving (22).
(3) Find the optimal control sequence by solving (23).

Note that step (1) can be solved in advance off-line.

5. FEASIBILITY AND STABILITY

To investigate the stability of the fault tolerant MPC we first present an equivalent optimization problem. We will utilize the fact that there exist a time-invariant set which can be used as a fixed constraint set, since the scenarios and system are time-invariant.

Definition 5. The safe state set is the largest forward invariant set, $\mathbb{X}_{safe}$, containing all $\mathbf{x}(t_0)$ such the optimization problem (23) is feasible with zero slack variables (i.e., $\mathbf{S}(t_k) = \mathbf{0}$). Let $\mathbb{X}_N$ denote the set where there exist a solution to (23) with horizon length N.

Lemma 6. Assuming the states have converged to $\mathbb{X}_{safe}$ (i.e., $N_{relaxed} = 0$) the problem can be simplified to the following:

$$\begin{aligned}
&\mathbf{U}^* = \arg\min_{\mathbf{U}} \sum_{i=0}^{N-1} l^{(n)}(\mathbf{x}^{(n)}(t_i), \mathbf{u}^{(n)}(t_i)) \\
&\text{such that} \\
&\mathbf{x}^{(n)}(t_i) \in \mathbb{X}_{safe} \\
&\left.\begin{aligned}
&\mathbf{x}^{(n)}(t_{i+1}) = \mathbf{f}^{(n)}(\mathbf{x}^{(n)}(t_i), \mathbf{u}^{(n)}(t_i)) \\
&\mathbf{G}^{(n)}_{relaxable}(\mathbf{x}^{(n)}(t_i), \mathbf{u}^{(n)}(t_i)) \le \mathbf{0} \\
&\mathbf{G}^{(n)}_{non-relaxable}(\mathbf{x}^{(n)}(t_i), \mathbf{u}^{(n)}(t_i)) \le \mathbf{0}
\end{aligned}\right\} t_0 \le t_i < t_N \\
&\mathbf{x}^{(n)}(t_N) = \mathbf{x}^{(n)s}
\end{aligned} \tag{25}$$

This optimization problem gives the controller and the nominal closed loop system:

$$\kappa_l^{(n)}(\mathbf{x}^{(n)}(t_k)) = \mathbf{u}^{(n)*}(t_k) \tag{26}$$

$$\mathbf{x}(t_{k+1})^{(n)} = \mathbf{f}^{(n)}(\mathbf{x}^{(n)}(t_k), \kappa_l^{(n)}(\mathbf{x}(t_k))) \tag{27}$$

The optimal nominal trajectory of optimization problem (23) is also the optimal trajectory of (25).

Proof The optimization problem (23) can be reformulated to (25) by the steps bellow. In $\mathbb{X}_{safe}$ all constraints are satisfied, including the relaxable constraints, so the slack variables and penalty functions can therefore be removed. The fault scenarios does only make sure that $\mathbf{x}^{(n)} \in \mathbb{X}_{safe}$ and does not alter the cost, it can therefore be removed from the optimization problem when $\mathbf{x}^{(n)} \in \mathbb{X}_{safe}$ and $\mathbf{x}^{(n)}$ is constrained to be within $\mathbb{X}_{safe}$. ■

Lemma 7. For all closed loop trajectories starting from $\mathbf{x}^{(n)}(t_0) \in \mathbb{X}_{safe}$ the trajectory of the nominal system system (24) will not leave $\mathbb{X}_{safe}$ and the average cost,

$$\bar{l} = \limsup_{N\to\infty} \sum_{k=0}^{N} \frac{l(x(t_k), u(t_k)}{N+1}, \tag{28}$$

will be lower or equal to the optimal safe steady state cost.

Proof This results follows directly from Theorem 1 in Angeli et al. (2012) by using the equivalent optimization problem (25). ■

Theorem 8. The following holds for the nominal scenario:

(1) The optimization problem (23) will stay feasible if it is initially feasible.
(2) It will take a maximum of N steps to reach $\mathbb{X}_{safe}$ from the time (23) is feasible, and from that time it will stay in $\mathbb{X}_{safe}$.
(3) The closed loop system has an average cost which is less than or equal to the optimal safe steady state cost for all initial conditions in $\mathbb{X}_N$.

Proof The recursive feasibility can be assured by a proof similar to Mayne et al. (2000). If we have a feasible input sequence at the previous step, we can always shift this sequence one step and extended the tail with $\mathbf{u}^{(n)}(t_{N-1}) = \mathbf{u}^{(n)s}$, denote this trajectory $\mathbf{U}'$ and the corresponding state trajectory $\mathbf{X}'$. This will make $\mathbf{x}^{(n)}(t_N) = \mathbf{x}^{(n)s} \in \mathbb{X}_{safe}$ and the nominal scenario feasible. We know that the shifted part of this trajectory will make all fault scenarios feasible, since they were feasible at the previous step. For the prolonged part $(\mathbf{x}^{(n)}(t_{N-1}), \mathbf{u}^{(n)}(t_{N-1}) = (\mathbf{x}^{(n)s}, \mathbf{u}^{(n)s})$, we know that this is not only feasible, but it does also satisfy the relaxable and non-relaxable constraints. Hence this input sequence is feasible and therefore the problem is recursive feasible, and part (1) is proven.

Next, we prove that the closed loop system will enter $\mathbb{X}_{safe}$ within N steps by induction. The trajectory $\mathbf{U}'$ will make sure that the relaxable constraints are feasible one time step earlier in the horizon than at the previous iteration. We know that the solution will honor the constraints as early as possible due to the minimization of $N_{relaxed}$. If the problem is feasible at t_0, the prediction of $\mathbf{x}(t_N)$ ends up in $\mathbb{X}_{safe}$. At each following step the length of the tail which honor the relaxable constraints is increased with at least one step. Therefore it will not take longer than the length of the prediction horizon N to reach $\mathbb{X}_{safe}$. This proves result (2).

The third result follows from the fact that the states will reach $\mathbb{X}_{safe}$ in finite time. Further, from Lemma 7, we know that inside $\mathbb{X}_{safe}$ the cost is lower than or equal to $l^{(n)s}$. Let t_{k_e} be the time the closed loop system enters $\mathbb{X}_{safe}$. The cost from t_0 to t_{k_e} is bounded, since $l(\cdot)$ is smooth on $\mathbb{X}_N$. The average cost will then be:

$$\begin{aligned}
&\limsup_{N\to\infty} \sum_{i=0}^{N} \frac{l(\mathbf{x}(t_i), \mathbf{u}(t_i))}{N+1} \\
&= \limsup_{N\to\infty} \sum_{i=0}^{k_e} \frac{l(\mathbf{x}(t_i), \mathbf{u}(t_i))}{N+1} + \sum_{i=k_e}^{N} \frac{l(\mathbf{x}(t_i), \mathbf{u}(t_i))}{N+1} \\
&= \limsup_{N\to\infty} \frac{C_1}{N+1} + \sum_{i=k_e}^{N} \frac{l(\mathbf{x}(t_i), \mathbf{u}(t_i))}{N+1} \\
&= \limsup_{N\to\infty} \sum_{i=k_e}^{N} \frac{l(\mathbf{x}(t_i), \mathbf{u}(t_i))}{N+1} \le l^s
\end{aligned} \tag{29}$$

■

Corollary 9. From the time $\mathbf{x}$ enters $\mathbb{X}_{safe}$ the plant can be recovered from any of the fault scenarios, without violating the constraints.

Proof This result follows directly from the definition of $\mathbb{X}_{safe}$ and the fact that the state will stay in $\mathbb{X}_{safe}$ after it has entered it. ■

6. RECONFIGURABLE CONTROL

The controller may be reconfigured in the event that one of the fault scenarios occur. If fault scenario j occurs, we switch to solving the following optimization problem:

$$\begin{aligned}
&\mathbf{U}^{(fj)*} = \underset{\mathbf{U}^{(fj)}}{\arg\min} \sum_{i=0}^{N-1} l^{(fj)}(\mathbf{x}^{(fj)}(t_i), \mathbf{u}^{(fj)}(t_i)) \\
&\text{such that} \\
&\mathbf{x}^{(fj)}(t_{i+1}) = \mathbf{f}^{(fj)}(\mathbf{x}^{(fj)}(t_i), \mathbf{u}^{(fj)}(t_i)) \qquad (30)\\
&\mathbf{G}^{(fj)}_{relaxable}(\mathbf{x}^{(fj)}(t_i), \mathbf{u}^{(fj)}(t_i)) \le \mathbf{0} \\
&\mathbf{G}^{(fj)}_{non-relaxable}(\mathbf{x}^{(fj)}(t_i), \mathbf{u}^{(fj)}(t_i)) \le \mathbf{0} \\
&\mathbf{x}^{(fj)}(t_N) = \mathbf{x}^{(fj)o},
\end{aligned}$$

where $\mathbf{x}^{(fj)o}$ is found by solving (13). The control law and closed loop system are then:

$$\kappa^{(fj)}(\mathbf{x}^{(fj)}(t_0)) = \mathbf{u}^{(fj)*}(t_0) \qquad (31)$$

$$\mathbf{x}^{(fj)}(t_{k+1}) = \mathbf{f}^{(fj)}(\mathbf{x}(t_k), \kappa^{(fj)}(\mathbf{x}(t_k))) \qquad (32)$$

Theorem 10. Assume fault scenario j occurs and the controller is switched from $\kappa^{(n)}$ to $\kappa^{(fj)}$ within τ_{robust} time steps. In addition, in the time between the fault occurs and the controller is reconfigured the predicted control input is used, without re-optimizing (23). Assume that (23) was feasible with slack variables equal to zero at the time step when the fault occurs. Then the following holds for the fault scenario:

(1) The optimization problem (30) is feasible, and will stay feasible.
(2) The average cost will be less than or equal to $l^{(fj)o}$.

Proof Feasibility can be proven by shifting the control sequence $\mathbf{U}^{(fj)}(t_f = 0)$ given from (23) and extending it with $\mathbf{u}^{(fj)o}$. Since it is assumed that it is solved with $\mathbf{S}^{(fj)}(t_f = t_0) = \mathbf{0}$ this control sequence will satisfy the relaxable constraints. It will also terminate at $\mathbf{x}^{(fj)o}$ since $\mathbf{x}^{(fj)}(t_N|t_f = t_0)$ from solving (23) terminates at $\mathbf{x}^{(fj)o}$ and prolonging the control trajectory with $\mathbf{u}^{(fj)o}$ will then make the state variables stay at $\mathbf{x}^{(fj)o}$. It will stay feasible since we can always use the previous trajectory, shift it and prolong it with $\mathbf{u}^{(fj)o}$, this will be a feasible solution (but may not be optimal) to (30). This proves the first result.

The second results follows directly from Theorem 1 in Angeli et al. (2012). ■

7. CASE STUDY

In this section we will present simulation results from a closed loop simulation of a linear plant with nonlinear cost. The differential equations are:

$$\begin{aligned}
\mathbf{x}^{(n)}(t_{k+1}) = &\begin{bmatrix} 1 & 5 & 5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}^{(n)}(t_k) \\
&+ \begin{bmatrix} 12.5 & 12.5 \\ 5 & 0 \\ 0 & 5 \end{bmatrix} \mathbf{u}^{(n)}(t_k) + \begin{bmatrix} -5b \\ 0 \\ 0 \end{bmatrix}
\end{aligned} \qquad (33)$$

where u_1 and u_2 is the control input and b is a known constant. The non-relaxable constraints (superscript is omitted as the constraints are valid for all scenarios):

$$\begin{aligned}
0 \le x_1 \le 1 \quad -0.1 \le u_1 \le 0.1 \\
0 \le x_2 \le 1 \quad -0.1 \le u_2 \le 0.1
\end{aligned} \qquad (34)$$

The relaxable constraint is:

$$5 \le x_1 \le 20 \qquad (35)$$

This can model a buffer tank with two slow pumps, where the pumps have both saturation limits and rate constraints.

The fault scenario are that one of the motors suddenly stops. This gives:

$$\mathbf{x}^{(f1)}(t_{k+1}) = \begin{bmatrix} 1 & 5 \\ 0 & 1 \end{bmatrix} \mathbf{x}^{(f1)}(t_k) + \begin{bmatrix} 12.5 \\ 5 \end{bmatrix} u_2^{(f1)}(t_k) + \begin{bmatrix} -5b \\ 0 \end{bmatrix} \qquad (36)$$

$$\mathbf{x}^{(f2)}(t_{k+1}) = \begin{bmatrix} 1 & 5 \\ 0 & 1 \end{bmatrix} \mathbf{x}^{(f2)}(t_k) + \begin{bmatrix} 12.5 \\ 5 \end{bmatrix} u_1^{(f2)}(t_k) + \begin{bmatrix} -5b \\ 0 \end{bmatrix} \qquad (37)$$

where $\mathbf{x}^{(f1)} = \left[x_1^{(f1)} \; x_3^{(f1)}\right]^\top$ and $\mathbf{x}^{(f2)} = \left[x_1^{(f2)} \; x_2^{(f2)}\right]^\top$. The stage costs are:

$$l^{(n)}(\mathbf{x}, \mathbf{u}) = \left(x_1^{(n)}\right)^4 + \left(x_2^{(n)}\right)^2 + \left(x_3^{(n)}\right)^2 \qquad (38)$$

$$l^{(f1)}(\mathbf{x}, \mathbf{u}) = \left(x_1^{(f1)}\right)^4 + \left(x_3^{(f1)}\right)^2 \qquad (39)$$

$$l^{(f2)}(\mathbf{x}, \mathbf{u}) = \left(x_1^{(f2)}\right)^4 + \left(x_2^{(f2)}\right)^2 \qquad (40)$$

The violation cost functions are (sub- and superscript is omitted as the function is used for all scenarios and slack variables):

$$g(\mathbf{s}) = (\mathbf{s} - 100 \times \mathbf{1})^2, \qquad (41)$$

for appropriate size of $\mathbf{1}$.

The controller is implemented in ACADO (Houska et al., 2011). The constant b is set to 0.5, and the initial values are $x_1 = 3$ and $x_2 = x_3 = 0.1$. In Fig. 3 are results from closed-loop simulation shown with and without fault. In Fig. 1 and 2, the predicted trajectories are plotted starting at t_2 and t_0.

8. CONCLUSION

This paper proposes a method to introduce safety constraints based on fault scenarios. The controller uses the fault scenarios internally in the model predictive controller to make sure that it controls the states to a state set where the plant is recoverable if faults occur. In addition a method for detecting that the system is fault-prepared is presented.

The performance of the controller was tested by closed-loop simulation of a linear plant. The simulations show that the controller fulfills the control objectives.

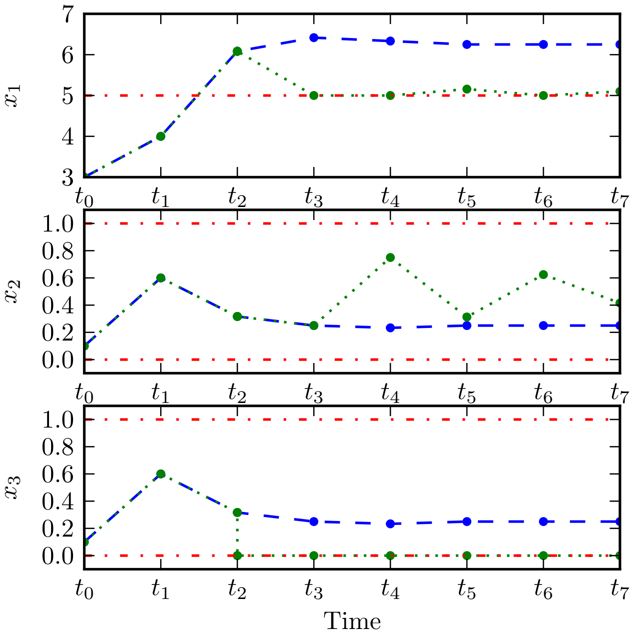

Fig. 3. Closed loop simulation of the linear plant. Results from a fault free simulation are plotted with dashed blue lines. The dotted green lines present a simulation where the fault occurs at $t = t_2$ (first time when $N_{relaxable} = 0$), the controller is reconfigured at $t = t_3$. The dotted red lines are the lower constraints.

A simplified version of this controller is presented in Bø and Johansen (2013), which did show that this method also can be implement on complex systems.

REFERENCES

Angeli, D., Amrit, R., and Rawlings, J.B. (2012). On average performance and stability of economic model predictive control. *IEEE Trans. Automat. Contr.*, 57(7), 1615–1626.

Bø, T.I. and Johansen, T.A. (2013). Scenario-based fault-tolerant model predictive control for diesel-electric marine power plant. In *MTS/IEEE Ocean. Bergen*, 1–5.

Bemporad, A., Bellucci, L., and Gabbriellini, T. (2012). Dynamic option hedging via stochastic model predictive control based on scenario simulation. *Quant. Financ.*, 1–13.

Bernardini, D. and Bemporad, A. (2009). Scenario-based model predictive control of stochastic constrained linear systems. In *Proc. 48h IEEE Conf. Decis. Control*, 6333–6338.

Blanke, M., Kinnaert, M., Lunze, J., and Staroswiecki, M. (2006). Diagnosis and Fault-Tolerant Control. In *Diagnosis Fault-Tolerant Control*, 10–23. Springer Berlin Heidelberg, 2nd edition.

Bonfill, A., Espuña, A., and Puigjaner, L. (2008). Proactive approach to address the uncertainty in short-term scheduling. *Comput. Chem. Eng.*, 32(8), 1689–1706.

Calafiore, G.C. and Fagiano, L. (2013). Robust Model Predictive Control via Scenario Optimization. *IEEE Trans. Automat. Contr.*, 58(1), 219–224.

Coogan, S. and Arcak, M. (2012). Guard synthesis for safety of hybrid systems using sum of squares programming. In *51st IEEE Conf. Decis. Control*, 6138–6143.

Gillula, J.H., Hoffmann, G.M., Vitus, M.P., and Tomlin, C.J. (2011). Applications of hybrid reachability analysis to robotic aerial vehicles. *Int. J. Rob. Res.*, 30(3), 335–354.

Goodwin, G.C. and Medioli, A.M. (2013). Scenario-based, closed-loop model predictive control with application to emergency vehicle scheduling. *Int. J. Control*, 86(8), 1338–1348.

Grüne, L. (2013). Economic receding horizon control without terminal constraints. *Automatica*, 49(3), 725–734.

Houska, B., Ferreau, H.J., and Diehl, M. (2011). ACADO toolkit — An open-source framework for automatic control and dynamic optimization. *Optim. Control Appl. Methods*, 32(3), 298–312.

Kovâcshâzy, T., Péceli, G., and Simon, G. (2001). Transient reduction in reconfigurable control systems utilizing structure dependence. In *Instrum. Meas. Technol. Conf. 2001. IMTC 2001. Proc. 18th IEEE*, 1143–1147.

Lao, L., Ellis, M., and Christofides, P.D. (2013). Proactive Fault-Tolerant Model Predictive Control. *AIChE J.*, 59(8), 2810–2820.

Limon, D., Alamo, T., Raimondo, D.M., Peña, D.M.n., Bravo, J.M., Ferramosca, A., and Camacho, E.F. (2009). Input-to-state stability: a unifying framework for robust model predictive control. In L. Magni, D.M. Raimondo, and F. Allgöwer (eds.), *Nonlinear Model Predict. Control*, volume 384, 1–26. Springer Berlin Heidelberg.

Maciejowski, J.M. (1999). Modelling and predictive control: enabling technologies for reconfiguration. *Annu. Rev. Control*, 23, 13–23.

Mayne, D.Q., Rawlings, J.B., Rao, C.V., and Scokaert, P.O.M. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(6), 789–814.

Prajna, S., Jadbabaie, A., and Pappas, G.J. (2007). A Framework for Worst-Case and Stochastic Safety Verification Using Barrier Certificates. *IEEE Trans. Automat. Contr.*, 52(8), 1415–1428.

Pranatyasto, T.N. and Qin, S. (2001). Sensor validation and process fault diagnosis for FCC units under MPC feedback. *Control Eng. Pract.*, 9(8), 877–888.

Rawlings, J.B. and Muske, K.R. (1993). The stability of constrained receding horizon control. *IEEE Trans. Automat. Contr.*, 38(10), 1512–1516.

Schildbach, G., Fagiano, L., Frei, C., and Morari, M. (2013). The Scenario Approach for Stochastic Model Predictive Control with Bounds on Closed-Loop Constraint Violations. *Automatica*.

Scokaert, P.O.M. and Mayne, D.Q. (1998). Min-max feedback model predictive control for constrained linear systems. *IEEE Trans. Automat. Contr.*, 43(8), 1136–1142.

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

Torrisi, F.D. and Bemporad, A. (2001). Discrete-time hybrid modeling and verification. In *IEEE Conf. Decis. Control*, 2899–2904.

Vada, J., Slupphaug, O., and Johansen, T.A. (2001). Optimal prioritized infeasibility handling in model predictive control: parametric preemptive multiobjective linear programming approach. *J. Optim. Theory Appl.*, 109(2), 385–413.