Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Formulation of Fatigue Dynamics as Hybrid Dynamical System for Model Predictive Control

Stefan Loew * **Dragan Obradovic** **

* *Wind Energy Institute of Technical University of Munich, Boltzmannstrasse 15, 85748 Garching, Germany, and Siemens AG, Corporate Technology, Otto-Hahn-Ring 6, 81739 Munich, Germany(e-mail: loew.stefan@siemens.com).*
** *Siemens AG, Corporate Technology, Otto-Hahn-Ring 6, 81739 Munich, Germany.*

Abstract: The standard fatigue estimation procedure is formulated as a hybrid dynamical system, which subsequently is utilized to calculate an economic terminal cost in MPC. This formulation is enabled by the development of a novel algorithm for continuous stress cycle identification. A second hybrid dynamical system is designed to provide fatigue cost gradients. The formulation turns out to be a powerful generalization of previous fatigue cost formulations, and additionally introduces consideration of past stress into the cost function. Presented closed-loop simulations using a wind turbine model provide insight into the subsystems of the hybrid dynamical system, and show the benefit of memorizing the past.

Keywords: Optimal Control, Hybrid Systems, Mechatronic Systems, Energy Systems

1. INTRODUCTION

Fatigue is damage of a material caused by cyclic application of mechanical stress. Fatigue has large impact on the operating costs of devices and thus control of fatigue is used to increase the total economic profit. Model Predictive Controllers (MPC) enable optimal control of many devices by using predictions of future system excitation (Findeisen and Allgöwer (2002)). Based on these input predictions, stress time-series at crucial spots in the device structure can be predicted. In *Direct Online Rainflow-counting* (DORFC, Loew et al. (2019)), these stress predictions are used for direct incorporation of the standard fatigue damage estimation procedure in the cost function of Model Predictive Control.

Questions:
Question 1) Since within the formulation of DORFC, fatigue cost is a discontinuous function of all time-samples of stress within the prediction horizon, neither the concept of stage cost nor of terminal cost applies (Grüne and Pannek (2017)). However, formal proofs of stability and recursive feasibility usually are commonly researched for those standard concepts (Grüne and Pannek (2017), Findeisen and Allgöwer (2002), Rawlings and Mayne (2009) p.112 ff.).
In Loew et al. (2020), stage cost formulations are achieved by some approximations, externalization of the fatigue cost evaluation from the MPC algorithm, and insertion of its results into the MPC via time-varying parameters. However, the question arises, if *direct stage or terminal cost formulations* without those additions are possible as well.

Question 2) In preceding methods (Loew et al. (2019), Loew et al. (2020)), fatigue cost is evaluated only based on the states in the prediction horizon. However, fatigue is a long-term effect, and correct evaluation requires knowledge about the entire stress history. As an extreme example, the very first stress sample after commissioning of the machine can form a stress cycle with the current stress sample several years later. Thus, the question arises, if *information about the stress history* can be included consistently into the predictive cost function.

Question 3) Taking up the previous question, storing the entire stress history requires a high amount of computational memory in the order of Gigabytes. Analyzing this high amount of data at every controller step requires excessive computing power. Thus, a smart method is required which enables *pertaining a reduced dataset* in the controller without losing information.

Question 4) Other approaches for fatigue control in literature use Markovian and causal surrogate models for fatigue dynamics (Luna et al. (2020), Gros and Schild (2017), Barradas-Berglind et al. (2015)). However, analysis in Loew et al. (2020) shows that cost functions are closer to the nature of the fatigue estimation process if stress state values are penalized based on their past *and* future evolution. In Loew et al. (2020), this property is implemented by above mentioned externalization of fatigue evaluation. However, the question arises, if this property also can be achieved by a standard cost formulation with *Markovian and causal properties*.

Contribution & Outline: In Sec. 2, *Questions 2)* and *3)* are solved by the development of a value-continuous fatigue cost calculation with residue handling. In Sec. 3,

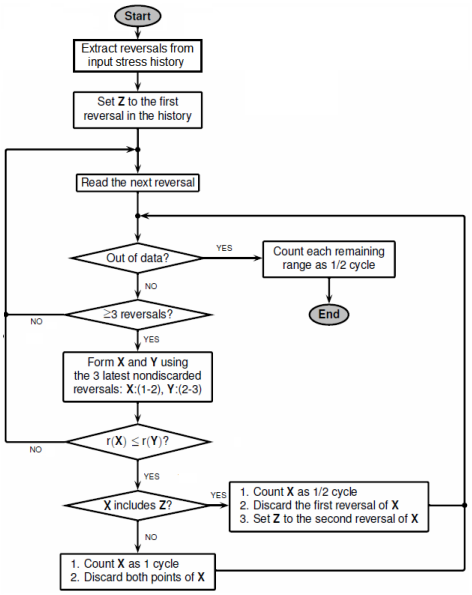

Fig. 1. Flowchart of the MATLAB-implementation `rainflow()` of the *Three-point Rainflow algorithm* (simplified from The MathWorks Inc. (2018)). Stress extrema are called "reversals". The range $r(\boldsymbol{X}) = |X(2) - X(1)|$ of a stress value pair $\boldsymbol{X}$ is the absolute value of the difference between both stresses.

fundamentals on delayed and hybrid systems are presented as a basis for further modeling. In Sec. 4, fatigue cost calculation is cast into a hybrid dynamical system which solves *Question 4)*. In Sec. 5, a link to previous formulations and an incorporation of the hybrid dynamical system into the terminal cost of an MPC is provided, which solves *Question 1)*. Sec. 6 provides first insight into this novel MPC implementation. Sec. 7 provides conclusion and outlook.

One comment on notation: The variable notation with bar $\bar{a}$ means sampled on the control intervals of the prediction horizon and with tilde $\tilde{a}$ means estimated from measurements.

2. FATIGUE ESTIMATION

2.1 Cycle identification and fatigue estimation

Rainflow-counting (RFC) (ASTM (1985), The MathWorks Inc. (2018)) is the standard method for decomposition of stress time-series to stress cycles and therefore is part of the standard fatigue estimation process. A flowchart of the Rainflow algorithm is displayed in Fig. 1. The Rainflow algorithm is based on reversals (extrema) of the stress trajectory, and contains algorithmic branches and loops. Thus, a crucial property of the Rainflow algorithm is its discontinuous output-behavior. Furthermore, the number N_c of identified cycles is variable, but bounded by the number of extrema. The outputs of RFC can be converted to the variables in Table 1 where the value of weight equals $w_c = 1$ if a full stress cycle is detected and equals $w_c = 0.5$ if a half cycle (half period) is detected.

2.2 Moving-window cycle identification & Residue

Originally, Rainflow analysis is performed on the entire stress history. However, in Heinrich et al. (2019) it is shown

Table 1. Converted outputs of RFC for stress cycles c

Quantity	Variable	Unit
Stress amplitude	$\sigma_{a,c}$	$[Pa]$
Stress mean	$\sigma_{m,c}$	$[Pa]$
Sample index of cycle maximum	$k_{max,c}$	$[-]$
Sample index of cycle minimum	$k_{min,c}$	$[-]$
Weight	w_c	$[-]$

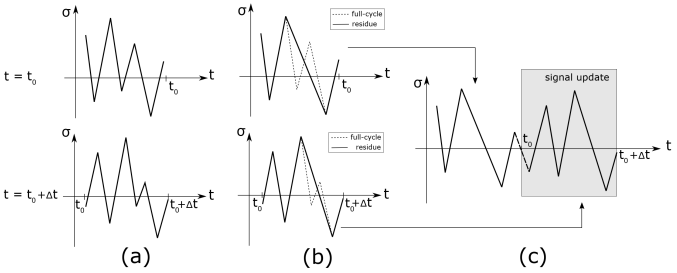

Fig. 2. Rainflow analysis on a moving window (Heinrich et al. (2019)).

that Rainflow analysis also can be performed on a moving window, like stated in Alg. 1.

In *Question 2)* of Sec. 1, the importance of considering the stress history in fatigue evaluation is pointed out. This is due to so called *transition cycles* which grow over a long period of time and can reach high stress amplitudes with dominating fatigue impact (Marsh et al. (2016)). Since transition cycles per definition have not been closed yet, they appear as half cycles in Rainflow analysis. Thus, to account for transition cycles in the moving-window algorithm, the corresponding half-cycle stress samples are carried along in the so-called *residue* (Köhler et al. (2012)).

Algorithm 1 Rainflow-analysis on a moving window; according to Heinrich et al. (2019)

Input: Existing stress string $\boldsymbol{\sigma}_{exist}$, Periodic update of new stress string $\boldsymbol{\sigma}_{new}$
Output: Full cycles

1: Extract full cycles from existing string ($t \leq t_0$) using Rainflow algorithm, Store residue in $\boldsymbol{x}_{res,1}$
while true **do**
2: Receive string $\boldsymbol{\sigma}_{new}$ of new stress samples obtained on $(t_0, t_0 + \Delta t]$
3: Extract full cycles from $\boldsymbol{\sigma}_{new}$ using Rainflow algorithm, Store residue in $\boldsymbol{x}_{res,2}$
4: Merge residues $\boldsymbol{x}_{res,1}$ and $\boldsymbol{x}_{res,2}$
5: Extract full cycles from $\{\boldsymbol{x}_{res,1}, \boldsymbol{x}_{res,2}\}$ using Rainflow algorithm, Store residue in $\boldsymbol{x}_{res,1}$
end

As an illustration, the Rainflow analysis on a moving window is shown in Fig. 2. At step *a)*, full cycles are extracted from an existing stress string which occurs at $t \leq t_0$. Additionally, a string of new stress samples ($t_0 < t \leq t_0 + \Delta t$) is obtained and full cycles are extracted from it as well. Step *b)* shows extracted full cycles and residues of both. Step *c)* shows the string which results from concatenation of both residues. Consequently, this string as well is analyzed for full cycles.

Depending on the stress signal, a high number of samples can be accumulated in the residue. Highest possible dimensions of the residue vector result from diverging and converging stress time series because they result in a very high number of half cycles (Köhler et al. (2012)). However, long-term diverging series are unrealistic because unstable machine behavior typically is counteracted by the controller or an emergency shutdown. Long-term converging series are irrelevant, since very low-amplitude cycles can be neglected without significant error in fatigue estimation. Concluding, the dimension N_{res} of the residue vector is finite and in practical tests remained well below 100.

2.3 One-step value-continuous cycle identification

The moving-window approach of Sec. 2.2 results in periodic updates of identified full cycles. This corresponds to a periodic value-discrete update of fatigue cost. However, gradient-based optimization in MPC requires value-*continuous* evolution of cost functions. Therefore, also the dynamics of fatigue cost needs to be expresses in a continuous fashion. Additionally, just like the plant states, fatigue cost should evolve on arbitrarily small timesteps unlike time periods Δt of Alg. 1. Both goals are achieved by the following two novel steps of enhancement:

As the first step, Alg. 1 is adapted to a one-step cycle identification which is shown in Alg. 2. The advantage is that this algorithm does not pause until a new string of stress samples is available, but directly provides an update of full cycles with each new stress sample.

Algorithm 2 One-step Rainflow-analysis

Input: Existing stress string $\boldsymbol{\sigma}_{exist}$, Periodic update of a scalar new stress sample σ
Output: Full cycles
Initialization: Stress residue $\boldsymbol{x}_{res} = \boldsymbol{\sigma}_{exist}$
while true **do**

1: Merge residue $\boldsymbol{x}_{res}$ and new stress sample σ
2: Extract full cycles from $\{\boldsymbol{x}_{res}, \sigma\}$ using Rainflow algorithm, Store residue in $\boldsymbol{x}_{res}$

end

As the second step, Alg. 2 is enhanced by additional consideration of half cycles which start "growing" already due to infitesimally small variations in stress. This consideration results in continuous output of the computation of fatigue cost which additionally is added to the algorithm, like stated in Alg. 3.

These enhancements do not introduce further assumptions. Thus, like shown in Fig. 3, this continuous fatigue cost calculation provides the same output like a batch evaluation over the entire stress trajectory. As expected, the fatigue cost rate never is negative, and thus fatigue cost is monotonously increasing. Concluding, Alg. 3 also solves *Question 3)* of Sec. 1 by condensing information of past full cycles in the scalar $x_{fatigue,FC}$, and by only memorizing stress samples which are extrema and have not contributed to full cycles yet.

3. SYSTEM CLASSES

In order to integrate the continuous fatigue cost estimation of Alg. 3 in an MPC, it has to be cast into a class of

Algorithm 3 One-step value-continuous fatigue cost estimation

Input: Existing stress string $\boldsymbol{\sigma}_{exist}$, Periodic update of a scalar new stress sample $\sigma(k+1)$ at each step k
Output: Periodic update of fatigue cost $J_{fatigue}(k)$ at each step k
Initialization: Zero total fatigue cost $J_{fatigue}(0) = 0$ and fatigue cost $x_{fatigue,FC}(0) = 0$ of full cycles, Stress residue $\boldsymbol{x}_{res}(0) = \boldsymbol{\sigma}_{exist}$, $k = 0$
while true **do**

1: Merge residue $\boldsymbol{x}_{res}(k)$ and new stress sample $\sigma(k+1)$
2: Extract full and half cycles from $\{\boldsymbol{x}_{res}(k), \sigma(k+1)\}$ using Rainflow algorithm, Store residue in $\boldsymbol{x}_{res}(k+1)$
3: Calculate fatigue cost based on full cycles, Add result to $x_{fatigue,FC}(k)$ to obtain $x_{fatigue,FC}(k+1)$
4: Calculate fatigue cost based on full and half cycles, Add result to $x_{fatigue,FC}(k)$ to obtain $J_{fatigue}(k+1)$
5: $k = k+1$

end

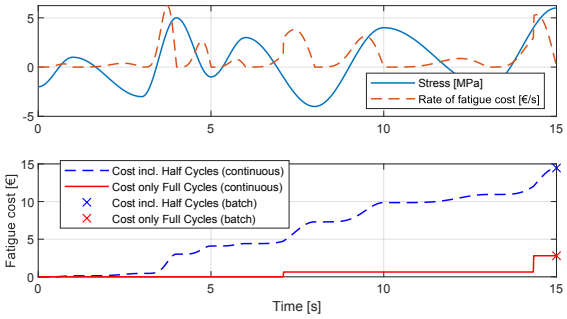

Fig. 3. Upper: Exemplary input stress trajectory, and fatigue cost rate obtained by Finite Differences from continuous fatigue cost including half cycles. Lower: continuous (one-step) and batch estimation using only full cycles or full+half cycles.

dynamical systems. A system class is required which allows for incorporation of past states, for dynamic variation of state dimensions, and for discontinuous updates of states. In the following, system classes from literature are assessed which seem promising with regard to these properties.

3.1 Delay differential equations

Delay differential equations (DDEs) are differential equations in which the state differential at a certain time depends on past state values (see Shampine and Thompson (2009)). However, discontinuous updates of states and their dimensions do not seem to be possible. Additionally, delays would have to increase with simulation time since the instances of residue-sampling are fixed in *absolute* time. Therefore, DDEs are *not utilized* in the present work.

3.2 Hybrid dynamical systems

Definition: A hybrid dynamical system is characterized by a coexistence of value-continuous and value-discrete dynamics (Aihara and Suzuki (2010)). They are able to describe physical mechanisms like impact and friction, and cyber-physical mechanisms like switching. Therefore for the present work, hybrid systems are highly suitable

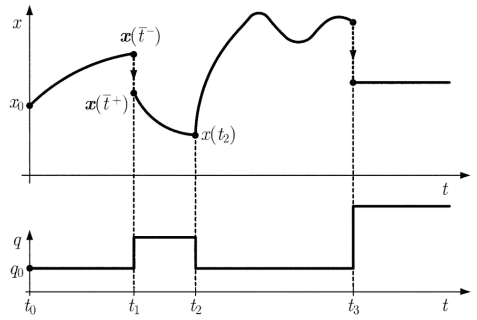

Fig. 4. Switching of dynamics at t_2 and resets of the value-continuous state x at t_1 and t_3, both triggered by the value-discrete state q (Lunze and Lamnabhi-Lagarrigue (2009)).

for modeling the interplay of physical value-continuous dynamics and algorithmic discontinuous mappings.

According to Aihara and Suzuki (2010) and Lunze and Lamnabhi-Lagarrigue (2009), hybrid dynamical systems can be described by the mappings

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{F}_{\boldsymbol{q}(t)}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{1a}$$
$$\boldsymbol{q}(t) = \boldsymbol{G}(\boldsymbol{q}(t^-), \boldsymbol{x}(t^-), \boldsymbol{u}(t)) \tag{1b}$$
$$\boldsymbol{x}(t) = \boldsymbol{R}(\boldsymbol{q}(t^-), \boldsymbol{x}(t^-), \boldsymbol{u}(t)) \tag{1c}$$
$$\boldsymbol{y}(t) = \boldsymbol{H}(\boldsymbol{q}(t), \boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{1d}$$

where

- $\boldsymbol{x}(t) \in \mathbb{R}^{N_x}$ are value-continuous states.
- $\boldsymbol{u}(t) \in \mathbb{R}^{N_{in}}$ are control variables.
- $\boldsymbol{q}(t) \in \mathbb{R}^{N_q}$ are value-discrete states.
- $t^- = \lim_{\epsilon \to 0} \tau - \epsilon$ denotes the time instant right before a discontinuous transition/reset event τ of states. t equivalently denotes the time instant after such an event.
- $\boldsymbol{y}(t) \in \mathbb{R}^{N_y}$ is the output of the system.
- $\boldsymbol{F}_{\boldsymbol{q}(t)}$ is a switched set of continuous differential equations. Which subsystem is active, is determined by the value-discrete states $\boldsymbol{q}(t)$.
- $\boldsymbol{G}$ is a discontinuous transition-map for the value-discrete states from $\boldsymbol{q}(t^-)$ to $\boldsymbol{q}(t)$.
- $\boldsymbol{R}$ is a discontinuous reset-map for jumps of the value-continuous states from $\boldsymbol{x}(t^-)$ to $\boldsymbol{x}(t)$.
- $\boldsymbol{H}$ is an output function.

An exemplary time-behavior of a simple hybrid dynamical system is shown in Fig. 4.

4. FATIGUE DYNAMICS AS HYBRID DYNAMICAL SYSTEM

4.1 Formulation of fatigue dynamics

In the following, plant dynamics and fatigue evolution of Alg. 3 will be cast into the framework of an hybrid dynamical system which was introduced in Sec. 3.2. Since the ultimate goal is utlization of this hybrid dynamical system in an MPC cost function, there will be a distinction between time instances in the controllable prediction horizon (*prediction*) and time instances before execution of the current MPC-step (*past*). The resulting subsystems stem from plant dynamics, extrema & cycle identification, update of residue, update of fatigue cost of past full cycles, and output of total accumulated fatigue cost. This is visualized in Fig. 5. Interestingly, all types of mappings of hybrid dynamical systems are utilized; namely several continuous differential equations, a transition-map, three reset maps, and one output function. Additionally, there are value-discrete and value-continuous state sets of variable dimensions.

Plant: The time-continuous evolution of value-continuous plant states $\boldsymbol{x}(t)$ is obtained by the set

$$\dot{\boldsymbol{x}} = \boldsymbol{F}_{sys}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t)) \tag{2}$$

of differential equations with control variables $\boldsymbol{u}(t)$, disturbances $\boldsymbol{d}(t)$ and initial states $\boldsymbol{x}(t_0) = \tilde{\boldsymbol{x}}(t_0)$. Fatigue is caused by stress $\sigma(t) \in \boldsymbol{x}(t)$ which is considered as one of the states of the system.

Cycle identification: According to Alg. 3, cycle identification by the Rainflow algorithm is based on a stress string which is a concatenation of the residual stress samples and the new stress sample. This string is input to the transition-map

$$\boldsymbol{q}_{struct}(t) = \boldsymbol{G}_{RFC}(\{\boldsymbol{x}_{res}(t), \sigma(t)\}) \tag{3}$$

which instantaneously updates the value-discrete structural states. The structural states are defined by

$$\boldsymbol{q}_{struct} = \begin{pmatrix} \boldsymbol{w} \\ \boldsymbol{k}_{max} \\ \boldsymbol{k}_{min} \end{pmatrix} = \begin{pmatrix} (w_1, w_2, ..., w_{N_c}) \\ (k_{max,1}, k_{max,2}, ..., k_{max,N_c}) \\ (k_{min,1}, k_{min,2}, ..., k_{min,N_c}) \end{pmatrix} \tag{4}$$

where the number N_c of identified cycles - and thus the dimensions - are variable. The structural states are cycle weights $\boldsymbol{w}$, sample indices $\boldsymbol{k}_{max}$ of cycle maxima and sample indices $\boldsymbol{k}_{min}$ of minima (compare to Tab. 1).

Residue: The update of the value-continuous residue states is obtained by the reset-map

$$\boldsymbol{x}_{res}(t) = \boldsymbol{R}_{res}(\boldsymbol{q}_{struct}(t^-), \boldsymbol{x}_{res}(t^-), \sigma(t^-), t) \tag{5}$$

with the residue states

$$\boldsymbol{x}_{res} = (\{\boldsymbol{x}_{res,past}, \boldsymbol{x}_{res,pred}\}) = \begin{pmatrix} (\sigma_{res,past,1}, \sigma_{res,past,2}, ..., \sigma_{res,past,N_{past}}) \\ (\sigma_{res,pred,1}, \sigma_{res,pred,2}, ..., \sigma_{res,pred,N_{pred}}) \end{pmatrix}^T \tag{6}$$

where the numbers of past N_{past} and predicted N_{pred} stress samples - and thus the dimensions - are variable. Since the reset-map depends on the structural states $\boldsymbol{q}_{struct}$, it is event-driven. Additionally, the reset-map is time-driven since the reset-behavior differs for the beginning $t = t_0$ and within $t_0 < t \leq t_{end}$ the horizon.

The reset-map comprises two sub-maps $\boldsymbol{R}_{res,past}$ and $\boldsymbol{R}_{res,pred}$ for treatment of past and predicted residue, respectively. Fig. 6 shows the sampling of past and prediction trajectory via reset-maps to obtain the residue states.

The update of the past-residue is obtained by the reset-map

$$\boldsymbol{x}_{res,past} = \boldsymbol{R}_{res,past} = \begin{cases} \{\sigma_{res,past,l}(t^-), \sigma(t^-)\} & \forall l \in \{k_{max,j}, k_{min,j}\} \\ \quad \forall j | \neg(w_j = 1 \wedge k_{max,j} \leq N_{past} \wedge k_{min,j} \leq N_{past}) \\ \quad \textbf{if } t = t_0 \\ \text{not active} \quad \textbf{if } t_0 < t \leq t_{end} \end{cases} \tag{7}$$

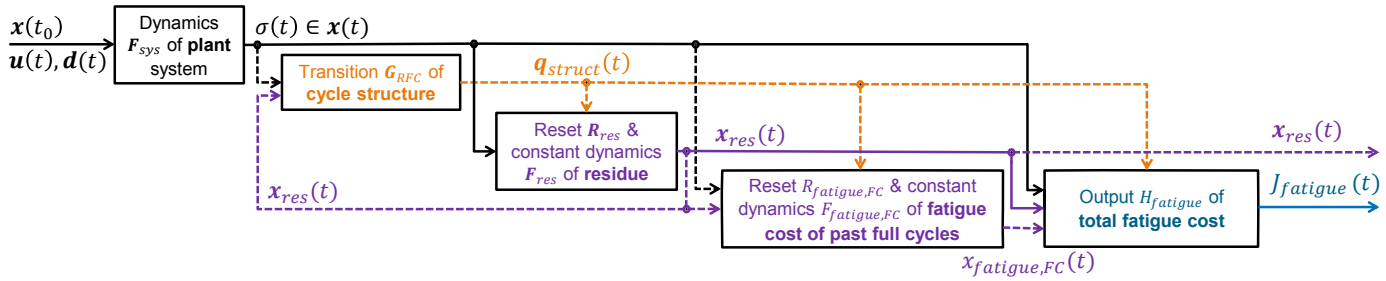

Fig. 5. Structure of the hybrid dynamical system for plant dynamics and fatigue cost evaluation. Black = value-continuous dynamics, Orange = transition-map, Purple = reset-map, Blue = output function. Dashed lines = paths which are neglected at fatigue cost gradient calculation.

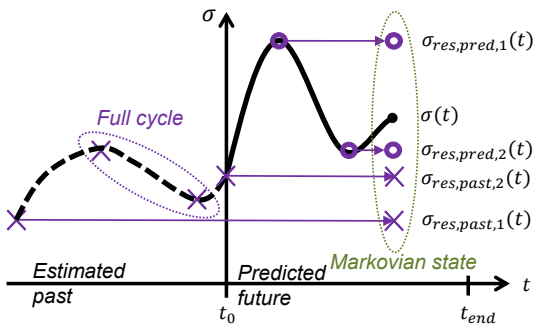

Fig. 6. Generation of past-residue $\boldsymbol{x}_{res,past}$ and prediction-residue $\boldsymbol{x}_{res,pred}$ from estimated past and predicted future stress evolution. Both past extrema which form a full cycle are excluded from the residue and their fatigue cost is added to the fatigue cost $x_{fatigue,FC}$ of past full cycles.

where the new estimated stress sample $\sigma(t^-)$ is appended to the previous past-residue $\sigma_{res,past,l}(t^-)$ at the beginning $t = t_0$ of the prediction horizon. From the previous past-residue, only samples are maintained which are extrema and thus contribute to stress cycles (index $l \in \{k_{max,j}, k_{min,j}\}$). This is further limited to extrema which do *not* contribute to full cycles ($w_j = 1$) which entirely occur in the past-residue ($k_{max,j} \leq N_{past} \wedge k_{min,j} \leq N_{past}$).

The update of the prediction-residue is obtained by the reset-map

$$\boldsymbol{x}_{res,pred} = \boldsymbol{R}_{res,pred} = \begin{cases} \emptyset & \textbf{if } t = t_0 \\ \{\sigma_{res,pred,l}(t^-), \sigma(t^-)\} & \forall l + N_{past} \in \{\boldsymbol{k}_{max}, \boldsymbol{k}_{min}\} \\ & \textbf{if } t_0 < t \leq t_{end} \end{cases} \tag{8}$$

where the new estimated stress sample is appended to the previous prediction-residue within the horizon $t_0 < t \leq t_{end}$. From the previous prediction-residue, all samples are maintained which contribute to stress cycles. At the beginning $t = t_0$ of a new MPC-step, the prediction-residue is initialized to an empty set.

In between of those resets, the residue states are held constant; which is represented by the continuous dynamics

$$\dot{\boldsymbol{x}}_{res}(t) = \boldsymbol{F}_{res} = \boldsymbol{0}. \tag{9}$$

Fatigue cost of past full cycles: The update of fatigue cost of full cycles which occurred at $t \leq t_0$ is obtained by the reset-map

$$x_{fatigue,FC}(t) = R_{fatigue,FC}(\boldsymbol{q}_{struct}(t^-), \boldsymbol{x}_{res}(t^-), \sigma(t^-), x_{fatigue,FC}(t^-)) = \begin{cases} \sum_j (w_j \, f_{fatigue,j}(\{\boldsymbol{x}_{res}(t^-), \sigma(t^-)\}, k_{max,j}, k_{min,j})) + \\ \quad + x_{fatigue,FC}(t^-) \quad \forall l \in \{k_{max,j}, k_{min,j}\} \\ \quad \forall j | (w_j = 1 \wedge k_{max,j} \leq N_{past} \wedge k_{min,j} \leq N_{past}) \\ \quad \textbf{if } t = t_0 \\ \text{not active} \quad \textbf{if } t_0 < t \leq t_{end} \end{cases} \tag{10}$$

which calculates a sum of convex fatigue cost functions $f_{fatigue}$ over all full cycles in the past residue (see Loew et al. (2019) for more details). Each fatigue cost function is composed by two stress samples out of the string $\{\boldsymbol{x}_{res}(t^-), \sigma(t^-)\}$; determined by the indices $k_{max,j}, k_{min,j}$. The summed fatigue cost is added to the previous version of the state $x_{fatigue,FC}(t^-)$. This state as well is accompanied by a constant continuous dynamics $\dot{x}_{fatigue,FC}(t) = F_{fatigue,FC} = 0$.

Fatigue cost: The continuous calculation of total accumulated fatigue cost is obtained by the output function

$$J_{fatigue}(t) = y_{fatigue}(t) = H_{fatigue}(\boldsymbol{q}_{struct}(t), \{\boldsymbol{x}_{res}(t), \sigma(t)\}, x_{fatigue,FC}(t)) = \sum_c (w_c \, f_{fatigue,c}(\{\boldsymbol{x}_{res}(t), \sigma(t)\}, k_{max,c}, k_{min,c})) + \\ + x_{fatigue,FC} \tag{11}$$

which calculates a sum of fatigue cost over all identified full and half cycles c, and adds this to the fatigue cost of past full cycles $x_{fatigue,FC}$.

Summary: The total hybrid system Φ, its states χ and output γ are defined by

$$\boldsymbol{\Phi} \begin{cases} \boldsymbol{F}_{sys} \\ \boldsymbol{G}_{RFC} \\ \boldsymbol{R}_{res} \\ \boldsymbol{F}_{res} \\ R_{fatigue,FC} \\ F_{fatigue,FC} \\ H_{fatigue} \end{cases} \quad \boldsymbol{\chi} \begin{cases} \boldsymbol{x} \\ \boldsymbol{q}_{struct} \\ \boldsymbol{x}_{res} \\ \\ x_{fatigue,FC} \\ \\ \end{cases} \quad \gamma \begin{cases} \\ \\ \\ \\ \\ \\ J_{fatigue}. \end{cases} \tag{12}$$

Since at any point of time the states $\boldsymbol{\chi}(t)$ contain all relevant information about the entire history, they form a *Markovian state*.

4.2 Derivation of fatigue cost gradient

For gradient-based MPC, gradients of states and cost functions w.r.t. the control variables $\bar{\boldsymbol{u}}$ have to be simulated in parallel to the states and cost functions (12). This gradient dynamics as well is cast into the framework

$$\begin{aligned}
\boldsymbol{\Phi}_{\bar{u}} &\left\{ \begin{matrix} \boldsymbol{F}_{sys,\bar{u}}, \boldsymbol{G}_{RFC,\bar{u}}, \boldsymbol{R}_{res,\bar{u}}, \boldsymbol{F}_{res,\bar{u}}, \\ R_{fatigue,FC,\bar{u}}, F_{fatigue,FC,\bar{u}}, H_{fatigue,\bar{u}} \end{matrix} \right. \\
\boldsymbol{\chi}_{\bar{u}} &\left\{ \frac{d\boldsymbol{x}}{d\bar{\boldsymbol{u}}}, \frac{d\boldsymbol{q}_{struct}}{d\bar{\boldsymbol{u}}}, \frac{d\boldsymbol{x}_{res}}{d\bar{\boldsymbol{u}}}, \frac{dx_{fatigue,FC}}{d\bar{\boldsymbol{u}}} \right. \qquad (13) \\
\gamma_{\bar{u}} &\left\{ \frac{dJ_{fatigue}}{d\bar{\boldsymbol{u}}} \right.
\end{aligned}$$

of an hybrid dynamical system. Since fatigue cost is the output of system (12), all gradient calculations culminate in the gradient

$$\begin{aligned}
\gamma_{\bar{u}_{i,j}}(t) = \frac{dJ_{fatigue}(t)}{d\bar{u}_{i,j}} = \frac{\partial H_{fatigue}(t)}{\partial \boldsymbol{\chi}} \frac{d\boldsymbol{\chi}(t)}{d\bar{u}_{i,j}} = \\
\frac{\partial H_{fatigue}(t)}{\partial \sigma} \frac{d\sigma(t)}{d\bar{u}_{i,j}} + \qquad (14a) \\
+ \frac{\partial H_{fatigue}(t)}{\partial \boldsymbol{q}_{struct}} \frac{d\boldsymbol{q}_{struct}(t)}{d\bar{u}_{i,j}} + \qquad (14b) \\
+ \sum_{l=1}^{N_{past}} \frac{\partial H_{fatigue}(t)}{\partial \sigma_{res,past,l}} \frac{d\sigma_{res,past,l}(t)}{d\bar{u}_{i,j}} + \qquad (14c) \\
+ \sum_{l=1}^{N_{pred}} \frac{\partial H_{fatigue}(t)}{\partial \sigma_{res,pred,l}} \frac{d\sigma_{res,pred,l}(t)}{d\bar{u}_{i,j}} + \qquad (14d) \\
+ \frac{\partial H_{fatigue}(t)}{\partial x_{fatigue,FC}} \frac{dx_{fatigue,FC}(t)}{d\bar{u}_{i,j}} \qquad (14e)
\end{aligned}$$

of fatigue cost output w.r.t. individual zero-order-hold-samples $\bar{u}_{i,j}$ of the control variables. The individual terms of (14) are derived in the following.

Plant: For term (14a), $\frac{\partial H_{fatigue}(t)}{\partial \sigma}$ is obtained by Automatic Differentiation (AD), and $\frac{d\sigma(t)}{d\bar{u}_{i,j}}$ from the respective Variational Differential Equation $\boldsymbol{F}_{sys,\bar{u}_{i,j}}$.

Cycle identification: For term (14b), $\frac{d\boldsymbol{q}_{struct}(t)}{d\bar{u}_{i,j}}$ cannot be obtained because the transition-map $\boldsymbol{G}_{RFC}$ cannot be differentiated. Therefore, the assumption stated in Loew et al. (2019) is applied, that the update of control variables $\bar{u}_{i,j}$ by the optimization algorithm only leads to mild variations in the state trajectories within one MPC-step. Therefore, the structure $\boldsymbol{q}_{struct}$ of identified cycles can be assumed to be invariant w.r.t. the controls within one MPC-step. This corresponds to

$$\frac{d\boldsymbol{q}_{struct}(t)}{d\bar{u}_{i,j}} = \boldsymbol{G}_{RFC,\bar{u}_{i,j}} = \boldsymbol{0} \qquad (15)$$

and a vanishing term (14b). Consequently, in Fig. 5, all paths are marked as neglected for gradient calculation which are connected to the transition map $\boldsymbol{G}_{RFC}$. Validation and further assessment of above mentioned assumption are subject of current research.

Residue: All summation terms of (14c) are equal to zero since the past residue samples from $t \leq t_0$ are independent from the control variables in $t > t_0$. Therefore, $\frac{d\sigma_{res,past,l}(t)}{d\bar{u}_{i,j}} = R_{res,past,l,\bar{u}_{i,j}} = 0 \quad \forall l \quad \forall t \geq t_0$.

For term (14d), gradient $\frac{\partial H_{fatigue}(t)}{\partial \sigma_{res,pred,l}}$ is obtained by AD. The gradients

$$\begin{aligned}
&\frac{d\sigma_{res,pred,l}(t)}{d\bar{u}_{i,j}} = \boldsymbol{R}_{res,pred,l,\bar{u}_{i,j}} = \\
&\begin{cases} \emptyset & \text{if } t = t_0 \\ \left\{ \frac{d\sigma_{res,pred,l}(t)}{d\bar{u}_{i,j}}(t^-), \frac{d\sigma(t)}{d\bar{u}_{i,j}}(t^-) \right\} & \\ \quad \forall l + N_{past} \in \{\boldsymbol{k}_{max}, \boldsymbol{k}_{min}\} & \text{if } t_0 < t \leq t_{end} \end{cases} \qquad (16)
\end{aligned}$$

of prediction-residue are updated by a reset-map of identical design to the reset-map of the prediction-residue itself (8). This gradient update is a simplification since the reset-map also depends on the structural states $\boldsymbol{q}_{struct}$, and formally gradients w.r.t. this variable would have to be derived as well. However, these terms are neglected due to assumption (15). Analogously to the residue states (9), also their gradients are held constant by $\frac{d\dot{\boldsymbol{x}}_{res}}{d\bar{\boldsymbol{u}}} = \boldsymbol{F}_{res,\bar{u}} = \boldsymbol{0}$.

Fatigue cost of past full cycles: Term (14e) is equal to zero since the past fatigue cost $x_{fatigue,FC}$ of full cycles is independent from the control variables in $t \geq t_0$. Therefore $\frac{dx_{fatigue,FC}(t)}{d\bar{u}_{i,j}} = 0$, $R_{fatigue,FC,\bar{u}} = 0$ and $F_{fatigue,FC,\bar{u}} = 0 \ \forall t \geq t_0$.

Fatigue cost: Concluding, fatigue cost gradient calculation reduces from (14) to the approximation

$$\begin{aligned}
\frac{dJ_{fatigue}(t)}{d\bar{u}_{i,j}} = \frac{\partial H_{fatigue}(t)}{\partial \sigma} \frac{d\sigma(t)}{d\bar{u}_{i,j}} + \\
+ \sum_{l=1}^{N_{pred}} \frac{\partial H_{fatigue}(t)}{\partial \sigma_{res,pred,l}} \frac{d\sigma_{res,pred,l}(t)}{d\bar{u}_{i,j}}. \qquad (17)
\end{aligned}$$

5. LINK TO EXISTING METHODS AND ENHANCEMENT

The present formulation of fatigue cost in a hybrid dynamical system is very general. In the following, this is demonstrated briefly by deriving from it two existing formulations of direct fatigue cost implementation in MPC. Additionally, both approaches are revolutionized in the sense that stress history from the past now can be considered in the cost function in a natural and consistent way.

5.1 Parametric Online Rainflow-counting

Derivation: In PORFC (Loew et al. (2020)), fatigue cost calculation within the MPC is continuous since the discontinuous subsystem is externalized and its state is fixed for one MPC-step. Implementation-wise, the continuous state prediction of the plant additionally is simulated before each evaluation of the MPC in order to perform Rainflow analysis on the output stress trajectory. The result of the Rainflow analysis, which corresponds to the discrete states $\boldsymbol{q}_{struct}$, is converted to *time-varying parameters* which are fed to the MPC. The remaining problem in the MPC then is continuous. A fairly similar approach can be seen in A. W. Winkler et al. (2018), were the sequence of discrete states is fixed a-priori and their event-times are optimized

in a continuous and smooth MPC problem.
In comparison to the hybrid dynamical system (12), subsystems in PORFC are:

- Externalized from MPC: $\boldsymbol{G}_{RFC}$.
- Neglected: $\boldsymbol{R}_{res}, \boldsymbol{F}_{res}, R_{fatigue,FC}, F_{fatigue,FC}$ since Rainflow analysis is performed batch-wise on the entire stress prediction, and no past residue is taken into account.
- Remaining in MPC: $\boldsymbol{F}_{sys}, H_{fatigue}$

Enhancement: Following the hybrid dynamical system formulation, past stress is introduced to the fatigue calculation of PORFC by adding all neglected subsystems to above mentioned external pre-preparation phase of the MPC.

5.2 Direct Online Rainflow-counting

Derivation: In DORFC Loew et al. (2019), the entire discontinuous fatigue cost calculation is performed within the MPC. Therefore, calculation of fatigue cost and its gradients already are similar to the hybrid dynamical systems (12) and (13), respectively. The two main differences are:

- Fatigue cost is calculated in a batch fashion over the entire prediction horizon. Thus, compared to the hybrid dynamical system, the stress update $\sigma(t^-)$ is not a scalar but is the entire stress trajectory ($\sigma(t)\ \forall t \in [t_0, t_{end}]$) over the prediction horizon.
- A past residue is not taken into account.

Enhancement: The fatigue cost calculation of DORFC can be replaced by a terminal cost $\Delta J_{fatigue}$ utilizing the present hybrid dynamical system. This terminal cost represents the *added* fatigue cost within the prediction horizon. This formulation results in the following time-continuous Nonlinear Programming problem:

$$\min_{\bar{u}} - \int_{t_0}^{t_{end}} J_{revenue}(\boldsymbol{x}(t))dt + \Delta J_{fatigue}(\boldsymbol{\chi}(t_{end}), \boldsymbol{\chi}(t_0)) \quad (18a)$$

where:

$$\begin{aligned}&\Delta J_{fatigue}(\boldsymbol{\chi}(t_{end}), \boldsymbol{\chi}(t_0)) = \\ &H_{fatigue}(\boldsymbol{\chi}(t_{end})) - H_{fatigue}(\boldsymbol{\chi}(t_0)) = \\ &H_{fatigue}(\boldsymbol{q}_{struct}(t_{end}), \{\boldsymbol{x}_{res}(t_{end}), \sigma(t_{end})\}, x_{fatigue,FC}(t_{end})) \\ &- H_{fatigue}(\boldsymbol{q}_{struct,0}, \{\boldsymbol{x}_{res,0}, \tilde{\sigma}\}, x_{fatigue,FC,0})\end{aligned} \quad (18b)$$

subject to:

$$\boldsymbol{\chi}(t) \mapsto \boldsymbol{\Phi}(\boldsymbol{\chi}(t)) \quad (18c)$$

$$\boldsymbol{\chi}(0) = (\tilde{\boldsymbol{x}}, \boldsymbol{q}_{struct,0}, \boldsymbol{x}_{res,0}, x_{fatigue,FC,0}) \quad (18d)$$

$$\begin{pmatrix} \boldsymbol{x} - \boldsymbol{x}_{max} \\ -\boldsymbol{x} + \boldsymbol{x}_{min} \end{pmatrix} \leq \boldsymbol{0} \quad (18e)$$

$$\boldsymbol{u}_{min} \leq \bar{\boldsymbol{u}} \leq \boldsymbol{u}_{max} \quad (18f)$$

If subsystems regarding past stress $\boldsymbol{R}_{res,past} = \boldsymbol{0}$ and past fatigue cost $R_{fatigue,FC} = 0$ are neglected, (18) returns similar results like the DORFC-formulation of Loew et al. (2019). If those subsystems are considered, DORFC is enhanced significantly since the MPC now can minimize stress cycles which start in the past and terminate in the prediction.

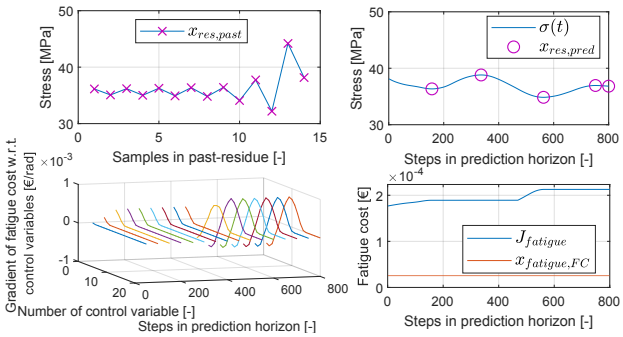

Fig. 7. *Upper left:* past-residue. *Upper right:* prediction trajectory $\sigma(t)$ and prediction-residue. *Lower right:* total fatigue cost and fatigue cost of full cycles over the prediction horizon. *Lower left:* gradients of fatigue cost w.r.t. to all samples $j = 1...N_u$ of one $i = 1$ control variable $\bar{u}_{i=1,j}$ in the prediction horizon. The gradient trajectory is given at every 50 out of 800 integrator steps in the prediction horizon.

6. DEMONSTRATION & VERIFICATION

The hybrid dynamical system formulation (18) is implemented in the Economic Nonlinear Model Predictive Controller of Loew et al. (2019) and tested with a wind turbine model. Goals for the MPC are maximization of revenue by harvested energy and minimization of structural fatigue at the root of the turbine tower. Important states are rotor speed and stress at the tower root. Control variables are generator torque and collective blade pitch angle. See Loew et al. (2019) for more details. Further examples of wind turbine control via MPC can be found in Luna et al. (2020), Gros and Schild (2017) and Barradas-Berglind et al. (2015). In the following, a brief insight and initial simulation results are provided.

Fig. 7 shows some of the hybrid states at a specific MPC-step. The controls are discretized zero-order-hold by $N_u = 20$ samples over the prediction horizon. The evolution of states is displayed on the integrator grid which has 800 steps per horizon. Fatigue cost $J_{fatigue}$ evolves continuously, like expected, but as well exhibits non-smoothness e.g. at step 480. This is due to the closing of a full cycle which is formed by $x_{res,pred,1}$ and $x_{res,pred,2}$, and the consequent continuation of a large half cycle which starts in the past-residue at $x_{res,past,13}$. This continuation as well results in a jump of the fatigue cost gradients. However, since (18b) is a terminal cost, only the fatigue cost gradients at the final integrator step 800 are used in the optimization. At the beginning of the horizon, the offset of $J_{fatigue}$ w.r.t. the fatigue cost of past full cycles $x_{fatigue,FC}$ equals the fatigue cost of all half cycles formed by the past-residue samples $\boldsymbol{x}_{res,past}$.

Fig. 8 shows first results from wind turbine simulations using the *DORFC* formulation from Loew et al. (2019) and the new *DORFC-R* formulation (18) including utilization of past-residue. Very generally, *DORFC-R* beneficially seems to result in smoother stress trajectories with less excursions. Especially in $82s < t < 100s$ a long-term flat behavior is observed, originating from the long-term stress memory due to past-residue $\boldsymbol{x}_{res,past}$. For the turbine rotor speed at $65s < t < 80s$ both formulations exhibit entirely

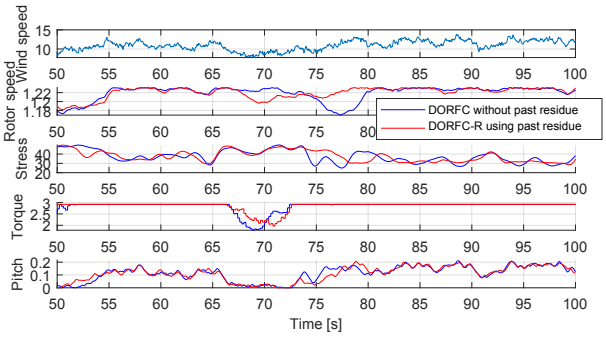

Fig. 8. Wind turbine simulation using the previous *DORFC* formulation without residue, and using the new hybrid dynamical system formulation *DORFC-R* with residue.

different strategies. Comprehensive analysis has to be done to judge which behavior is superior. In general, these very first results need to be extended by simulations which are planned for the future.

7. CONCLUSION & OUTLOOK

Within the present work, a novel algorithm for continuous fatigue cost calculation has been presented. Using this algorithm, it has been possible to cast the complete standard fatigue estimation procedure into the general framework of a hybrid dynamical system. For implementation in gradient-based MPC, a second hybrid dynamical system has been developed which provides fatigue cost gradients w.r.t. control variables. The hybrid dynamical system has been implemented in the cost function of an Economic Nonlinear MPC where it provides accurate fatigue estimation by efficiently memorizing stress samples which even might date back for several years. By deriving previous fatigue cost formulations from the present hybrid dynamical system, the impressing generality of this approach has been proven.

In the future, this new MPC implementation will be tested thoroughly by applying it to a high-fidelity wind turbine simulator and to a LiIon battery energy storage system. Furthermore, the generalization of fatigue cost functions to hybrid dynamical systems will allow for systematic theoretical analysis of stability (Müller and Allgöwer (2012)).

ACKNOWLEDGEMENTS

The authors would like to thank Prof. Carlo L. Bottasso for the fruitful discussions and his academic guidance. Further thanks goes to the Siemens-colleagues Mohamed Khalil, Armin Nurkanovic and Sebastian Albrecht.

REFERENCES

A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli (2018). Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters*, 3(3), 1560–1567. doi:10.1109/LRA.2018.2798285.

Aihara, K. and Suzuki, H. (2010). Theory of hybrid dynamical systems and its applications to biological and medical systems. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 368(1930), 4893–4914. doi:10.1098/rsta.2010.0237.

ASTM (1985). Standard practices for cycle counting in fatigue analysis.

Barradas-Berglind, J.d.J., Wisniewski, R., and Soltani, M. (2015). Fatigue damage estimation and data-based control for wind turbines. *IET Control Theory & Applications*, 9(7), 1042–1050. doi:10.1049/iet-cta.2014.0730.

Findeisen, R. and Allgöwer, F. (2002). An introduction to nonlinear model predictive control. In *21st Benelux Meeting on Systems and Control*, 1–23.

Gros, S. and Schild, A. (2017). Real-time economic nonlinear model predictive control for wind turbine control: International journal of control. *International Journal of Control*, 1–14. doi:10.1080/00207179.2016.1266514.

Grüne, L. and Pannek, J. (2017). *Nonlinear Model Predictive Control: Theory and Algorithms*. Communications and Control Engineering. Springer International Publishing, 2nd ed. 2017 edition.

Heinrich, C., Khalil, M., Martynov, K., and WEVER, U. (2019). Online remaining lifetime estimation for structures. *Mechanical Systems and Signal Processing*, 119, 312–327. doi:10.1016/j.ymssp.2018.09.028.

Köhler, M., Jenne, S., Pötter, K., and Zenner, H. (2012). *Zählverfahren und Lastannahme in der Betriebsfestigkeit*. Springer, Dordrecht.

Loew, S., Obradovic, D., Anand, A., and Szabo, A. (2020). Stage cost formulations of online rainflow-counting for model predictive control of fatigue. In *Accepted to European Control Conference 2020*.

Loew, S., Obradovic, D., and Bottasso, C.L. (2019). Direct online rainflow-counting and indirect fatigue penalization methods for model predictive control. In *2019 18th European Control Conference (ECC)*, 3371–3376. IEEE. doi:10.23919/ECC.2019.8795911.

Luna, J., Falkenberg, O., Gros, S., and Schild, A. (2020). Wind turbine fatigue reduction based on economic-tracking nmpc with direct ann fatigue estimation. *Renewable Energy*, 147, 1632–1641. doi:10.1016/j.renene.2019.09.092.

Lunze, J. and Lamnabhi-Lagarrigue, F. (2009). *Handbook of Hybrid Systems Control: Theory, Tools, Applications*. Cambridge University Press, Cambridge. doi:10.1017/CBO9780511807930.

Marsh, G., Wignall, C., Thies, P.R., Barltrop, N., Incecik, A., Venugopal, V., and Johanning, L. (2016). Review and application of rainflow residue processing techniques for accurate fatigue damage estimation. *International Journal of Fatigue*, 82, 757–765. doi:10.1016/j.ijfatigue.2015.10.007.

Müller, M.A. and Allgöwer, F. (2012). Improving performance in model predictive control: Switching cost functionals under average dwell-time. *Automatica*, 48(2), 402–409. doi:10.1016/j.automatica.2011.11.005.

Rawlings, J. and Mayne, D.Q. (2009). *Model Predictive Control: Theory and Design*. Nob Hill Pub.

Shampine, L. and Thompson, S. (2009). Delay differential equations. *Delay Differential Equations: Recent Advances and New Directions*. doi:10.1007/978-0-387-85595-0_9.

The MathWorks Inc. (2018). Rainflow counts for fatigue analysis.