Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Energy and Process Engineering

Eivind Hvistendahl

A generic lifetime estimation model for Francis turbines

Master's thesis in Energy and Environmental Engineering
Supervisor: Ole Gunnar Dahlhaug
Co-supervisor: Bjørn Winther Solemslie
June 2022

NTNU
Norwegian University of Science and Technology

Waterpower Laboratory
NTNU

Eivind Hvistendahl

A generic lifetime estimation model for Francis turbines

Master's thesis in Energy and Environmental Engineering
Supervisor: Ole Gunnar Dahlhaug
Co-supervisor: Bjørn Winther Solemslie
June 2022

Norwegian University of Science and Technology
Faculty of Engineering
Department of Energy and Process Engineering

NTNU
Norwegian University of
Science and Technology

Abstract

As more intermittent energy sources are connected to the grid, Francis turbines increasingly provide grid balancing services. This will cause the turbines to periodically operate in sub-optimal configurations. Since Francis turbines vary greatly in design, a generic turbine design is not clearly defined. The objective of this project was to develop a lifetime estimator for a generic Francis turbine. This project shows that the generalisation of some turbine aspects appears possible. It also showed that extrapolating specific experimental data between different turbines might be justified. A generic fatigue damage model for Francis turbine runners was constructed and compiled into a Functional Mock-up Unit (FMU). This was done both to ensure the continued model development and to enable model exchanges. By defining load intervals, with corresponding stress amplitudes, operating data was used to count stress cycles. Miner's summation was used together with material-specific S-N curves to locate the material fatigue life for different stress amplitudes. Operating data from the Bratsberg power plant in Norway was acquired, and the model was tested on this turbine. Based on the accumulated fatigue from the five years, a Monte Carlo simulation projected 5000 unique lifetime estimations. From the lifetime distribution analysis, the average turbine lifetime was 36.7 years with a standard deviation of 13.0. The major conclusion of this thesis is that a general fatigue damage model of the Francis turbine is possible. By utilising operating history, runner frequency, and S-N curve, the Palmgren-Miner was used to evaluate accumulated fatigue damage and project the expected turbine lifetime.

Keywords: Generic Life, Francis turbine, fatigue life modelling, Functional Mock-up Interface (FMI), Functional Mock-up Unit (FMU)

Sammendrag

Ettersom mer intermitterende energikilder kobles til strømnettet må Francisturbiner i økende grad tilby balanseringstjenester. Dette fører til at turbinene periodisk opererer i suboptimale konfigurasjoner. Siden Francisturbiner varierer mye i design, er et generisk turbindesign ikke tydelig definert. Med det sagt viser dette prosjektet at generalisering av noen turbinaspekter er mulig. Videre viser det at ekstrapolering av spesifikke eksperimentelle data mellom forskjellige turbiner rettferdiggjøres. En generisk utmattingsskademodell for Francis løpehjul ble derfor laget og videre omgjort til en Functional Mock-up Unit (FMU). Dette ble gjort både for å sikre den videre modellutviklingen og for å muliggjøre modellutveksling. Ved å definere lastintervaller, med tilhørende spenningsamplituder, kan driftsdata brukes til å telle lastsykluser. Miners summering brukte materialspesifikke S-N-kurver for å lokalisere materialtretthetslevetiden for forskjellige spenningsamplituder. Driftsdata fra Bratsberg kraftverk i Norge ble innhentet, og modellen ble testet på turbinen. Basert på den akkumulerte utmattingen fra de fem årene, projiserte en Monte Carlo-simulering 5000 unike levetidsestimater. Fra levetidsfordelingen var gjennomsnittlig turbinlevetid 36.7 år med et standardavvik på 13.0. Prosjektet ble sett på som en suksess både med tanke på modellen som ble laget, og FMU-byggingen. Prosjektet konkluderer med at driftshistorikk, løpehjulsfrekvens og en S-N-kurve, kan Palmgren-Miner-regelen brukes til å evaluere akkumulerte utmatting. Videre konkluderer oppgaven med at turbinlevetid kan projiseres fra dette.

Acknowledgments

This project was done with the help of my supervisors, co-supervisors, fellow students, and other staff members of the Waterpower Laboratory. The Waterpower Lab has provided a great working space and enabled the students here to mingle with experienced researchers and staff. It has also sponsored trips to conferences and meetings with companies. I am very grateful for the opportunity of getting to experience the hydropower industry from within.

A special thank to my main supervisor, Professor Ole Gunnar Dahlhaug, for inspiration and guidance. His enthusiasm for the students and the Waterpower Laboratory is praiseworthy. Some special gratitude is directed towards Bjørn Winther Solemslie from the Norwegian Institute for Nature Research. Without him the project would never have gotten as far as it did. Big thanks to my companions and supporters for making the time at NTNU worthwhile. Rock on.

Contents

List of Tables

List of Figures

Nomenclature

Abbreviation

CFD	Computational Fluid Dynamics
NTNU	Norwegian University of Science and Technology
TSO	Transmission system operator
ML	Minimal load
PL	Part load
BEP	BEP
FL	Full load
RSI	Rotor-stator interaction
LCF	Low cycle fatigue
HCF	High cycle fatigue
FEM	Finite Element Method
FSI	Fluid structure interaction
CAL	Constant amplitude loading
VAL	Variable amplitude loading
UTS	Ultimate Tensile Strength
SNL	Speed-no-load

SD	Standard deviation
FMU	Functional Mock-up Unit
FMI	Functional Mock-up Interface
Cr	Chromium
Ni	Nickel
C	Programming language
XML	File format
RSS	Root-Sum-Square
IIW	International Institute of Welding
rpm	Revolutions per minute
P-M	Palmgren-Miner

Latin symbols

A	Area	m^2
c	Absolute velocity	$\mathrm{m/s}$
D	Diameter	m
E	Elastic modulus	MPa
E_h	Specific hydraulic energy	$\mathrm{m}^2/\mathrm{s}^2$
F	Force	N
f_n	Runner frequency	Hz
f_s	Rheingan's frequency	Hz
g	Gravitationl constant	$\mathrm{m/s}^2$
H	Head	m
L	Life reduced	hours
n	Rotational speed of the runner	rpm
n_{ED}	Speed factor	–

N_f	Number of cycles	–
n_s	Specific speed	–
P	Power	W
Q	Volumetric flow rate	$\mathrm{m^3/s}$
Q_{ED}	Discharge factor	–
R	Stress ratio	–
T	Torque	Nm
u	Peripheral velocity	$\mathrm{m/s}$
w	Relative velocity	$\mathrm{m/s}$
z_{gv}	Number of guide vanes	–
z_r	Number of runner blades	–

Greek symbols

χ^2_k	Chi-squared	–
Δ	Difference	–
η_h	Hydraulic efficiency	–
ω	Rotational velocity	$\mathrm{rad/s}$
ρ	Density	$\mathrm{kg/m^3}$
ϵ	Strain	–
σ	Stress	Mpa

Superscripts and subscripts

abs	Absolute
a	Amplitude
eff	Effective
el	Elastic
GV	Guide vane

i,j	Index
k	Degrees of freedom
max	Maximum
min	Minimum
m	Mean
n	Net
nom	Nominal
pl	Plastic
u	Ultimate

Chapter 1

Introduction

Today, the planet is in a "green shift" to meet obligations of cutting carbon emissions set in the Paris Agreement of 2015 [1]. During the transition from fossil energy sources to renewables, energy engineering will face numerous new challenges. The main sources for renewable energy are predicted to be wind and solar [2]. In themselves, wind and solar, are weather dependent and therefore intermittent. To continue delivering reliable electricity to the end users, these energy sources need to be backed up and balanced by predictable energy sources. Hydro turbines makes a seemingly perfect fit for this task. Francis turbines are the most widespread hydro turbine and they have an instantaneous balancing capability, where power output is adjusted by megawatts per second.

While hydro turbines historically have provided steady, baseload power for predictable fluctuations in electricity consumption, new operating patterns will increasingly call for grid balancing services [3]. To balance intermittent energy, the hydro turbines are frequently forced outside their optimal operating range. As a result, the cyclic, mechanical stress amplitudes in the turbine runners reach damaging levels. In consequence, this means a higher rate of wear and shorter component lifetime. There are solid indications that more frequent start-stops and operation outside best efficiency range, causes more rapid fatigue in Francis turbine runners [3, 4, 5].

For hydropower operators, it is crucial to accurately predict the need for maintenance and replacements of components. Today, the planning of maintenance is most often calendar-based, and the turbine condition is not necessarily updated based on the current operating pattern [6]. With more dynamically operated turbines the maintenance predictions become increasingly demanding. From greater uncertainties, the power producers rely on expert inspectors and their experience to visually inspect, and evaluate the current condition of the hydro turbines [7]. The requirements for a turbine fatigue model able to replace this cumbersome and

expensive practice, are highly demanding to fulfill. It would need to be modular, easy to operate and able to interconnect with other models. As of now, there is no industry standard enabling the coupling of generic operating data with a fatigue model. The Functional Mock-up Interface (FMI) is a standard seeking to do just that. By creating an environment for individual models, known as Functional Mock-up Units (FMUs), to exchange outputs for inputs. With this model ecosystem, one can achieve large, complex models consisting of numerous precise submodels. By implementing the FMI standard in the hydropower industry, an FMU calculating the expected lifetime based on current operating data could be part of a larger ecosystem of FMUs co-operating to predict maintenance and refurbishment needs.

1.1 Objectives

This Master's thesis has the objective of developing a lifetime estimator for a generic Francis turbine runner. It is a part of Statkraft's "Generic Life" project initiated by Dr. Erik J. Wiborg, Dr. Bjørn W. Solemslie, and Dr. Erik Tengs. The author and three fellow students partook in the master's thesis contribution of Generic Life. Its goal is to take any operating data from any power plant, a few design parameters and calculate the expected lifetime based on this. It seeks to bridge the gap of applying fatigue physics to day-to-day operation. The thesis is first and foremost a proof of concept to demonstrate the feasibility of a generic lifetime estimator.

Chapter 2

Background theory

The theory presented is largely based on the author's project thesis from fall of 2021; "A generic lifetime reduction model for Francis turbines" [8].

2.1 Hydro turbines

Hydro turbines utilize water as a working medium to transform hydraulic energy to mechanical energy [9]. The mechanical energy drives the shaft connected to a generator transforming mechanical energy to electrical energy. Hydro turbines are divided into two groups; reaction and impulse turbines. In reaction turbines there is a pressure difference in the water from inlet to outlet. Impulse turbines convert internal pressure energy in the water to kinetic energy. The water then transfer kinetic energy to the hydro turbine, without a change in water pressure. The three most prevalent types are Francis, Pelton and Kaplan turbines [10]. Which one to use in different cases are suggested in 2.1, as a function of specific speed and head. Within the three turbine categories the designs also vary substantially. Specific speed is defined as a dimensionless quantity in Equation 2.1.

$$n_s = \frac{n \cdot \sqrt{Q}}{g \cdot H^{3/4}} \tag{2.1}$$

Where n is rotational speed, Q is volumetric flow rate, g is the gravitational constant and H is the total head. Head is a conversion of static pressure to meters, defined in Equation 2.2.

$$H = \frac{p}{\rho \cdot g} \tag{2.2}$$

Where p is static pressure and ρ is water density.

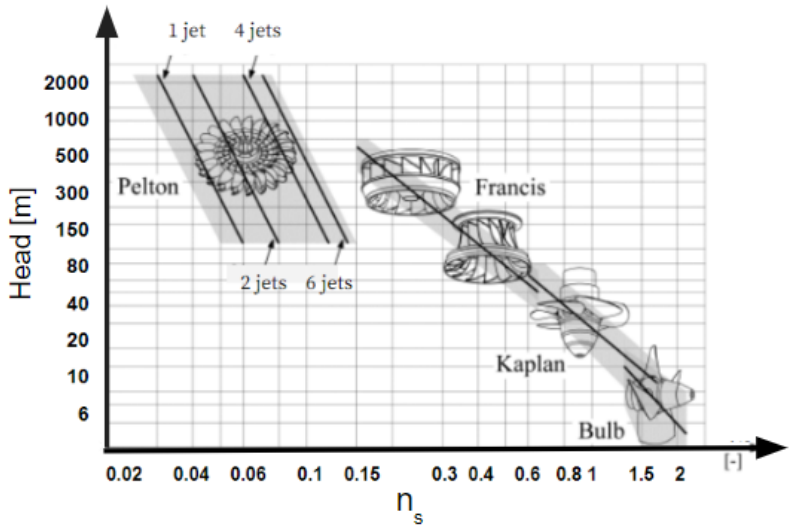


Figure 2.1: The typical use of Pelton, Francis and Kaplan turbines given specific speed and head. Bulb turbine is not covered in this project. [11].

Power produced in a hydropower plant is calculated as Equation 2.3.

$$P = \rho \cdot g \cdot Q \cdot H \cdot \eta_{\text{h}} \tag{2.3}$$

ρ, g, Q and H are the same as in Equation 2.1 and Equation 2.2. η_{h} is the hydraulic efficiency.

The preferred head ranges for different hydro turbines are visualised in Figure 2.1.

Kaplan turbine

The Kaplan turbine is a reaction type hydro turbine used for low head power generation and is reminiscent of a typical propeller in shape. Being preferred for lower heads, it often takes advantage of the relatively modest height differences in rivers.

Pelton turbine

The Pelton turbine is preferred at greater heads up to as much as 1886m in Switzerland [9]. It is an impulse type turbine where the pressure energy in the water is transformed to kinetic energy as a water jet with velocity given in Equation 2.4.

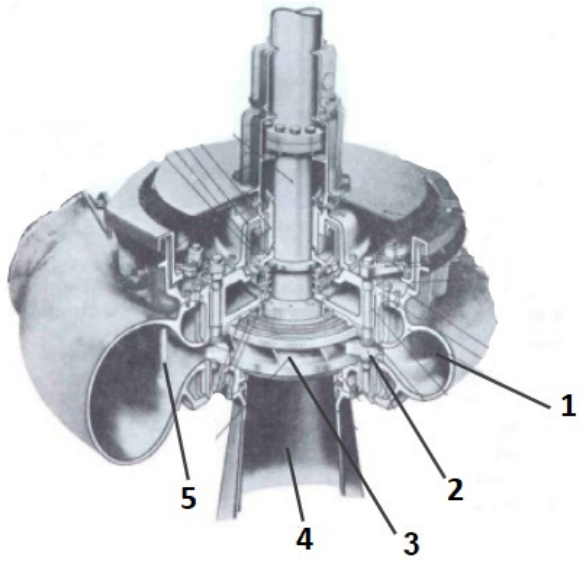


Figure 2.2: Overview of important components in a Francis turbine [14].

$$c = \sqrt{2 \cdot g \cdot H} \tag{2.4}$$

The kinetic energy of the water is further transformed to rotational energy as it is deflected by the buckets on a Pelton wheel. A higher number of nozzles in a Pelton turbine gives a flatter efficiency curve [10]. This enables it to operate at a wider load range with greater average efficiency.

2.1.1 Francis turbine

Francis turbines function as a combination of both impulse and reaction, meaning both hydraulic pressure and kinetic energy is transferred to the runner. Compared to the other turbine types addressed in this thesis, the geometry is rather complex. It is the most common of hydro turbines and the design we know today, dates back to 1920 [12]. Francis turbines provide both power grid stabilisation services and is then said to operate flexibly. These services make the Francis turbine a crucial part of maintaining stable power supply as more intermittent power is connected to the grid [13].

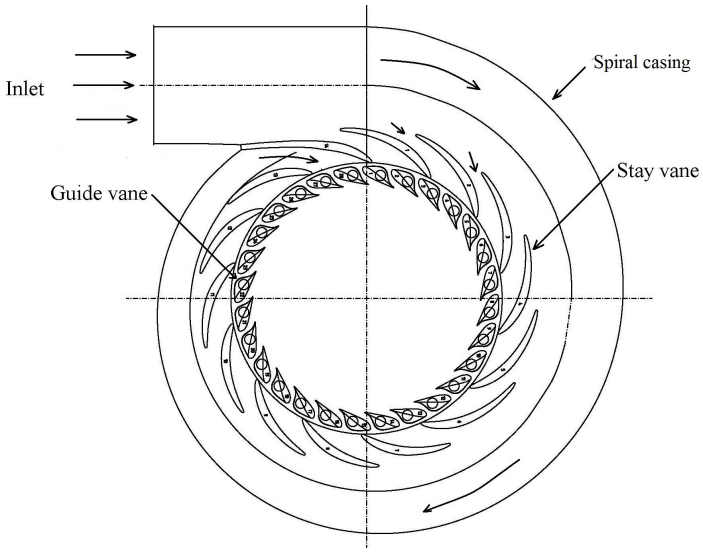


Figure 2.3: Top view of a spiral casing distributing water evenly around the runner inlet circumference [15].

Spiral casing

Component number 1 in Figure 2.2, is known as the spiral casing. Water enters it from the penstock and its purpose is to distribute water evenly around the circumference of the turbine runner inlet. The cross-sectional area of a spiral case decreases to maintain an even inflow rate as water enters the runner.

Guide vanes

Component number 2 is the guide vanes. They serve as the main governor of flow rate and consequently the power produced. After water is directed radially inwards by the stay vanes, the flow rate is controlled by the opening between two guide vanes. The control of flow is characterised by the angle of opening, α, illustrated in Figure 2.4.

Runner

Component number 3 is the Francis runner. The runner is the component where energy transformation occurs as it puts torque on a turbine shaft driving the generator. The runner blades are contained between the shroud and hub and is what propels

(a) Positive opening angle, α. **(b)** α equal zero.

Figure 2.4: Flow control in a Francis turbine [10].

the runner around. They are designed to produce a hydraulic lift force arising from pressure difference between the two sides of a blade. In addition, they also propel the runner by redirecting the water impulse. The runner blades are most susceptible to fatigue and maximum material stress can be found in welds between the blade and shroud close to the trailing edge [16].

Draft tube

As water leaves the runner it enters component 4, the draft tube. The draft tube will hold fluctuating flow phenomena, depending on what load the turbine is operating at.

Stay vanes

Component number 5 is the stay vanes. They are angled after turbine specific design to direct the water into the guide vanes.

2.2 Hydraulic efficiency

Professor H. Brekke describes hydraulic efficiency, η_h, as the amount of available energy transferred to mechanical energy in the runner divided by the net potential energy drop, $g\Delta H$ through the turbine [9]. Summed up in Equation 2.5.

$$\eta_h = \frac{u_1 \cdot c_{u1} - u_2 \cdot c_{u2}}{g \cdot \Delta H} \tag{2.5}$$

Hydraulic losses in Francis runners happen due to skin friction from fluid-structure interactions, viscous dissipation of turbulence and cavitation [17, 18].

2.2.1 Hill chart

Hill charts are used as visual representations of hydro turbine performance. It is comparable to a topographic map, where instead of constant height curves there are

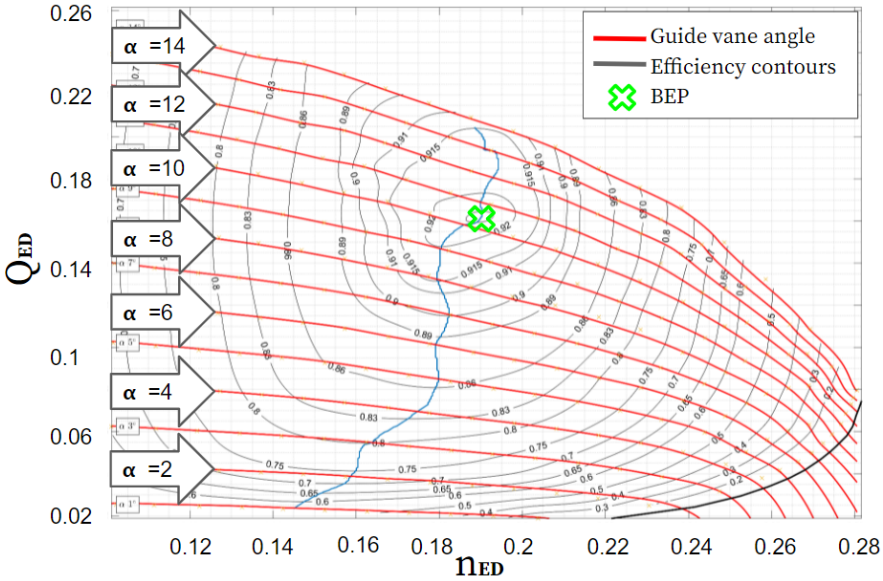


Figure 2.5: Hill chart of the Francis-101 runner from H. West. Red lines are constant degree of guide vane opening and the grey lines are constant efficiency contours. The BEP is marked on "top of the hill" as a green X, located at 92.14% efficiency in this case.

constant efficiency curves. The hill chart efficiency curves culminate to a "peak", known as the best efficiency point (BEP). In Francis turbines the efficiency contours are usually plotted as a function of speed factor, n_{ED}, and discharge factor Q_{ED}.

$$n_{\text{ED}} = \frac{(n/60) \cdot D}{\sqrt{E_h}} \tag{2.6}$$

$$Q_{\text{ED}} = \frac{Q}{D^2 \cdot \sqrt{E_h}} \tag{2.7}$$

Where n is rotational speed, D is runner outlet diameter, Q is volumetric flow rate and E_h is specific hydraulic energy.

It is also helpful for the reader to couple the discharge factor with the guide vane opening angle, α, as shown in Figure 2.4(b). A hill chart from the project work of Mr. H. West (2021) on the Francis-101 rig in the Waterpower Laboratory, is presented in Figure 2.5.

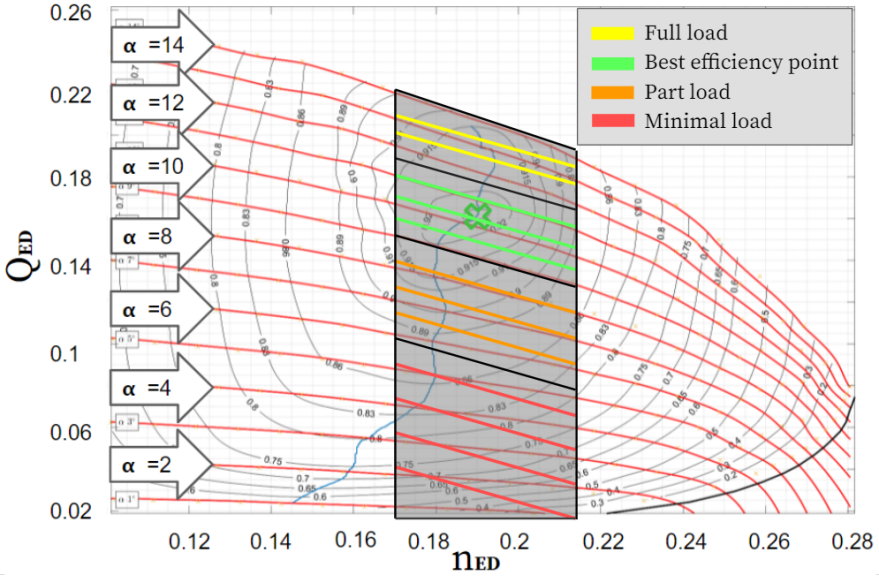


Figure 2.6: Illustration of load intervals placed on a hill chart.

2.2.2 Load intervals

Francis turbines are able to operate in a wide range of loads. In this thesis the working range is divided into load intervals. Load interval limits are set at certain percentages of the best efficiency point (BEP) load, shown in Equation 2.8.

$$\text{Load interval limit} = \frac{\text{Load at limit}}{\text{Load at BEP}} \cdot 100 \tag{2.8}$$

In ascending power output, the load intervals were as follows: minimal load (ML), part load (PL), best-efficiency point (BEP), and full load (FL). However, there is no definite point where one load interval stops and the next one starts. The load intervals are associated with different flow phenomena and pressure fluctuation characteristics that define them. A visualisation of load intervals on a hill chart is in Figure 2.6.

2.3 Pressure pulsation phenomena

In different load intervals, pressure will vary from different types of pulsation phenomena. Pulsations are dependent on factors such as head and load. The two most

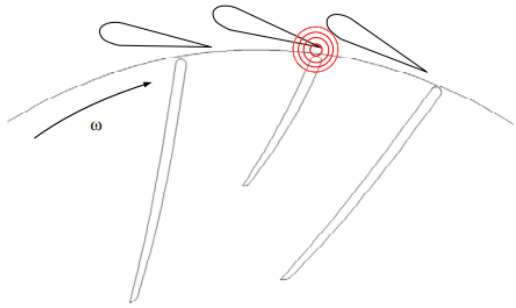


Figure 2.7: Rotating runner blades passing the stationary guide vanes, inducing RSI pressure fluctuations [19].

prevalent types of pressure fluctuations are rotor-stator interactions, RSI, and draft tube surges [16]. In addition to these, there are high frequency pressure oscillations at the trailing edges from von Kármán vortex shedding [19].

2.3.1 RSI

Pressure pulsations from rotor-stator interactions arise in the vaneless space as a runner blade closely passes a stationary guide vane trailing edge [19]. This is illustrated in Figure 2.7. The frequency of the RSI pulses is dependent on what reference system it is observed from. If the reference system is the rotating runner, the RSI has the guide vane passing frequency, f_{GV}, as formulated in Equation 2.9.

$$f_{GV} = z_{GV} \cdot f_n \tag{2.9}$$

z_{GV} is the number of guide vanes, and f_n is the rotational frequency of the runner. The RSI amplitude is dependent on the clearance between rotor and stator, where a smaller gap results in greater amplitudes [19].

2.3.2 Draft tube pressure pulsations

The draft tube pressure pulsations depend on which load interval the turbine operates in. This source of pressure pulsations at the turbine outlet is formed as the water pressure drops below local vapour pressure, causing cavitation [17]. Cavitation occurs either at the blades or in the form of a vortex rope connected to the runner hub. These draft tube pressure surges are detected in frequency spectra of pressure measurements inside the runner, meaning they also extend and propagate upstream. The vortex rope frequency is called Rheingan's frequency. Different draft tube flow

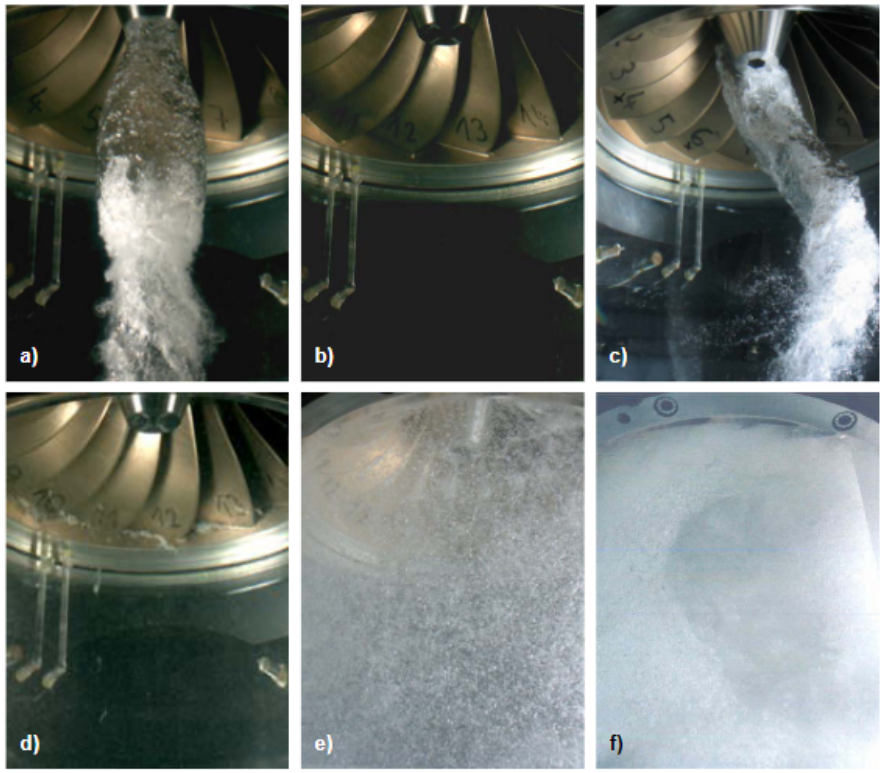


Figure 2.8: Different flow phenomena causing pressure fluctuations in the draft tube. a) full load, b) around BEP, c) part load, d) minimal load, e) speed-no-load, f) runaway. Taken from Seidel et al. [16].

phenomena and vortex rope configurations is depicted in Figure 2.8.

Full load

After exiting the runner outlet, the water is drawn radially outwards by centrifugal forces, leaving a low-pressure area in the draft tube center. This low-pressure area may drop below the vapour pressure in the water, causing the formation of a vortex rope stretching down from the hub. This vortex configuration is mostly stable but pressure fluctuations can be observed at certain conditions [16]. The flow has a swirl component moving opposite of the runner.

Around BEP

From Figure 2.8 no cavitation in the draft tube or at the trailing edge of runner blades is observed. BEP operation is the most stable load interval, and this is where the machine runs with the least amount of pressure fluctuations. It is also here the pressure amplitudes are the lowest.

Part load

Here, the fluid flow is again drawn radially outwards. In addition, the flow has a swirl component now moving in the same direction as the runner. Instability in the outlet flow combined with a low-pressure region at the center causes the formation of a vortex rope. Compared to the vortex rope at FL, this configuration is unstable and moves with the swirling flow. This causes pressure fluctuations with a given frequency called Rheingan's frequency. These pressure amplitudes affect the runner internally and can be observed in frequency spectra. Rheingan's frequency, f_s, is empirically approximated in Equation 2.10.

$$f_s = f_n \cdot 0.33 \pm 20\% \tag{2.10}$$

This is an estimate of Professor O.G. Dahlhaug.

Minimal load

The flow in the runner is characterised by turbulence. This causes local low-pressure regions and in effect cavitation. The bulk of the pulsations in this load interval has a stochastic frequency distribution.

Speed-no-load

At SNL the generator is not connected to the grid and the turbine rotates at synchronous speed without load.

Runaway

The runner rotates without the influence of opposing generator torque. The runner surpasses design speed. This causes the flow phenomena from SNL to further escalate, and cavitation is widespread.

Figure 2.9: Francis runner blades subjected to notable cavitation damage.

2.4 Damage types in a Francis turbine

Cavitation

Cavitation develops as the pressure in a fluid drops below the vapour pressure. Cavitation in Francis turbines can cause extensive damage and material wear as seen in Figure 2.9. Since it also produces noise, vibration and performance reduction, cavitation occurrences are kept at the lowest level possible [17].

Hammering

Hammering in Francis turbines may cause damage to the waterway from the guide vanes and further upstream. It occurs when there is an abrupt stop in flow causing a sharp pressure wave propagating upstream. If not properly handled components may be damaged. This is usually suppressed by surge shafts or air pillows in the penstock allowing water expansion.

Corrosion

Corrosion in Francis turbines and runners is handled by using stainless steels. The choice of steel is, among other considerations, a compromise between corrosion resistance and hardness to resist erosion.

Sand erosion

Sand erosion can cause massive wear on stay and guide vanes as well as the runner. In areas with sand rich water driving the turbine, components will regularly have to be replaced. This problem is reduced by sand traps and using replaceable parts, including interchangeable runner blades.

Fatigue/Cracks

Fatigue of a material occur when it is subjected to cyclic stress amplitudes over time. The number of load cycles a material experiences before fatigue failure, N_f, depends on the stress amplitude, σ_a, mean stress effects, and the material in question.

2.5 Fatigue

Fatigue is defined as the failure of a material, even below its strength limit, from cyclic loading.

2.5.1 Stress-Strain

Stress and strain are two fundamental parameters in mechanical engineering [20]. Stress is defined as a force, *F*, being applied to an area, *A*, it has unit *Pa* and is often measured in megapascals, *MPa*.

$$\sigma = \frac{F}{A} \tag{2.11}$$

Material under tensile stress responds by straining. Strain, ϵ, is the relation between elongation and original length of the specimen.

$$\epsilon = \frac{\Delta L}{L}. \tag{2.12}$$

Important properties of a material can be obtained from monotonic tensile strength tests [20], where the test specimen is subjected to increasing tensile stress until it breaks. Some properties of importance are the elastic modulus,

$$E = \frac{\Delta \sigma}{\Delta \epsilon}, \tag{2.13}$$

yield strength, σ_Y, and ultimate tensile strength, *UTS*, or σ_u. An example of such stress-strain test is shown in Figure 2.10. The slope of the elastic range equals the elastic modulus.

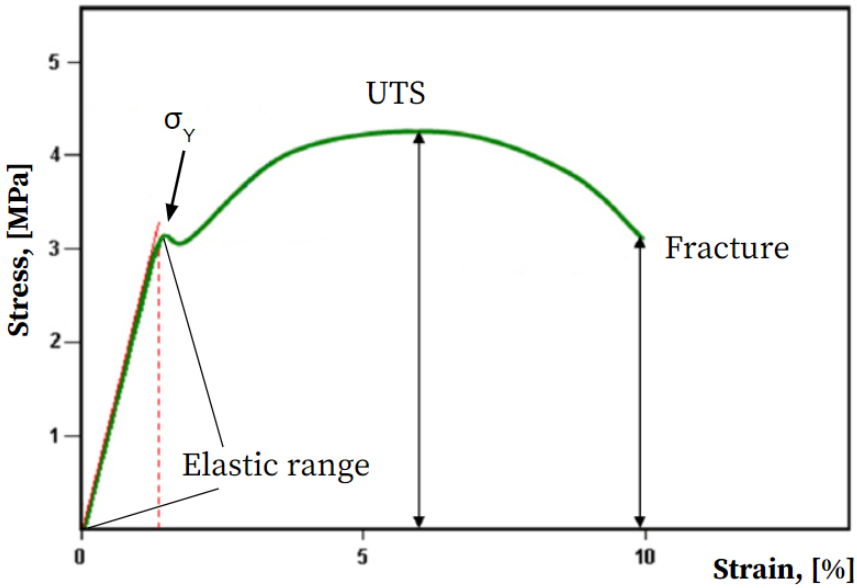


Figure 2.10: Example curve from a generic tensile strength test [21]. σ_Y is the yield strength and UTS is the Ultimate Tensile Strength.

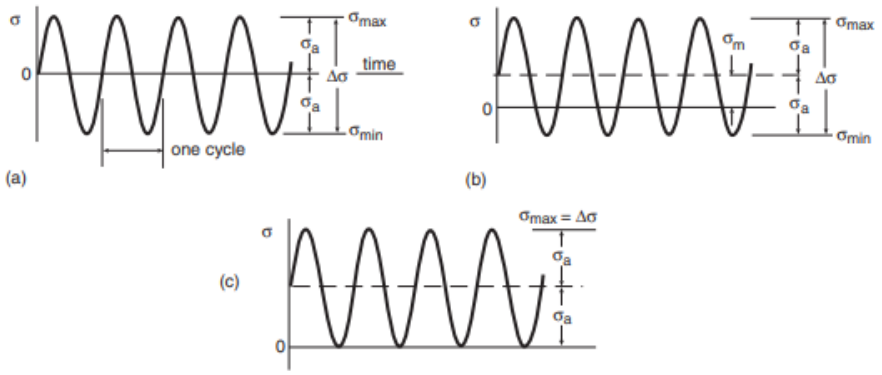


Figure 2.11: Illustrations of three modes of CAL. a) mean stress is zero, b) mean stress is above zero, c) mean stress is above zero and the specimen is never in compression [20].

The elastic range is where all deformation happens elastically, so specimen return to their original length if the stress returns to zero. All stress applied after the yield strength will cause plastic deformation, and permanent elongation equals the maximum elastic strain plus additional plastic strain. Total strain is shown in Equation 2.14.

$$\epsilon_t = \epsilon_{\text{el}} + \epsilon_{\text{pl}}. \tag{2.14}$$

2.5.2 Cyclic stress

Stress repeated over time is defined as cyclic stress where one period equals one cycle, shown in Figure 2.11. Cycles can be counted, and the variable, N is the number of cycles. Cyclic stress is categorised as either constant amplitude loading, CAL, or variable amplitude loading, VAL. A cyclic stress value below zero indicates compression. A generic CAL case with different mean stresses is shown in Figure 2.11.

Stress ratio, R, is the relation between minimum stress, and maximum stress in a cycle.

$$R = \frac{\sigma_{\text{min}}}{\sigma_{\text{max}}} \tag{2.15}$$

VAL is what many practical cases, including Francis turbines, will experience in

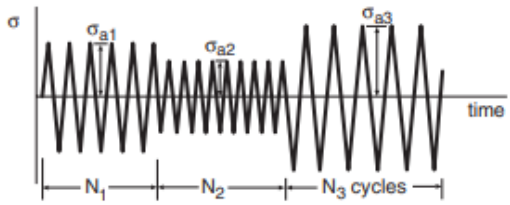


Figure 2.12: VAL with periods of CAL [20].

their lifetime. The pressure pulsation phenomena previously presented have different frequencies and amplitudes. They are due to this characterised as VAL. Stress amplitudes vary in time, but may still have prolonged periods of CAL. This is shown in Figure 2.12.

2.5.3 Start-Stop

Start-stop cycles are known as transient operating conditions. Together with ramp ups and downs, starts and stops are where the turbine experiences a shift in operating conditions. Start-stop cycles are known to cause substantial turbine runner damage. It is suggested to shorten runner life ranging from several hours to as much as days [22, 23, 24, 25].

2.5.4 S-N curve

Testing the fatigue life of a material is done by applying a constant amplitude stress, until failure. Common ways of applying stress are either by an axial push-pull test or a rotating-bending test. If this is conducted for a wide range of stresses, an S-N curve can be obtained by regression analysis of the failure points. S is for stress, and N is number of cycles. The regression curve might be plotted for a specific confidence level. For example, if a curve is plotted with a 95% confidence level, it means that only 5% of specimen will fail below the failure number, N_f. This is done to provide designers a safety margin for unexpected component failure. It is common to present stress amplitude and the cycles to failure on a logarithmic scale. The S-N curve will then assume a linear shape on log-log plots. Presented in Figure 2.13 is an example of the S-N curve for A517 steel. There exists a lower limit for stress amplitudes where a non-welded specimen seemingly has an infinite fatigue life. This limit, σ_e, is named the fatigue limit [20].

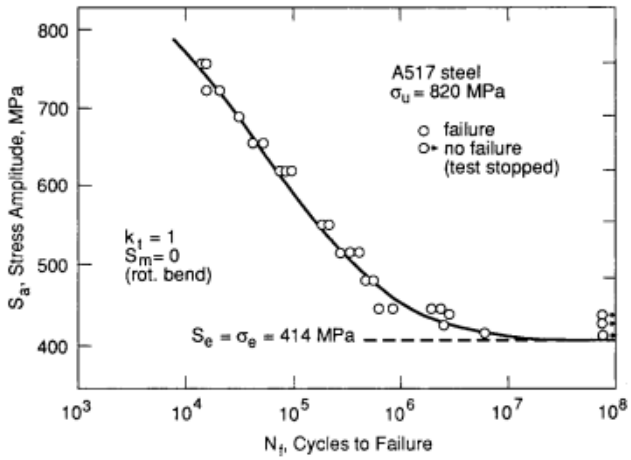


Figure 2.13: A rotating-bending test of a smooth A517 steel specimen. The number of cycles to failure, N_f, presented on a logarithmic scale. This test was done with fully reversible CAL, with mean stress, $\sigma_m = 0$. From Dowling [20].

2.5.5 Goodman's correction

In many practical cases, the periodic stress does not oscillate around zero mean stress. This is the case in Francis turbines for instance, where a constant mean stress is present in the runner blades. To account for the mean stress in an S-N approach, several correction methods exist. One of them is the Goodman method [20] where the effective stress, σ_{eff}, is calculated from Equation 2.16.

$$\sigma_{\text{eff}} = \sigma_a \left[\frac{\sigma_u}{\sigma_u - \sigma_m} \right]. \tag{2.16}$$

σ_a is stress amplitude, σ_m is the mean stress accounted for.

2.5.6 Cycle counting

For lifetime estimation of objects under cyclic stress, cycle counting is immensely valuable. By combining a S-N curve with the stress history of an object, one could count the number of cycles at different stress amplitudes. This is illustrated in Figure 2.14.

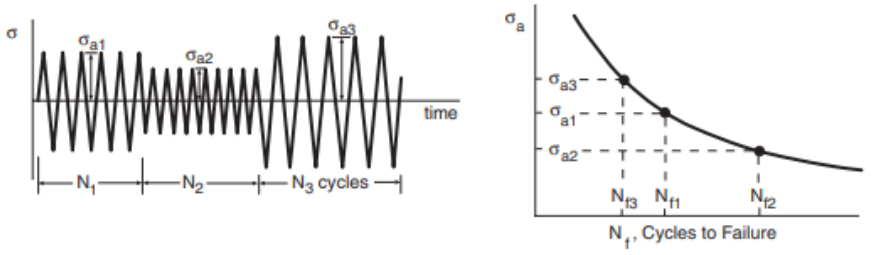


Figure 2.14: The variable loads, σ_{a1}, σ_{a2} and σ_{a3}, in a load history has a unique amount of cycles to failure, N_f, determined from an S-N curve [20].

Miner's rule

From Figure 2.14, a damage accumulation analysis can be conducted. One method for this is the Palmgren-Miner rule shown in Equation 2.17.

$$\frac{N_1}{N_{f1}} + \frac{N_2}{N_{f2}} + \frac{N_3}{N_{f3}} + \cdots = \sum \frac{N_j}{N_{fj}} = 1. \tag{2.17}$$

When the sum of different load cycles reaches 1, fatigue failure is expected to occur.

Miner's rule is not without limitations though. It assumes a linear accumulation of damage, which is not the case if a crack is present. The order of the stress sequences is not considered. This is important because a large amplitude stress could possibly initiate a crack and successive stress of small amplitudes would propagate the crack [20].

2.5.7 Impact of welds on fatigue life

In large members with long, welded seems there are micro cracks, inclusions and residual stress [20]. These are introduced in the welding process and is difficult to avoid [26]. For Francis turbines, the runner blades are often welded to the runner shroud and hub. The impact of welds on fatigue life is therefore highly relevant when examining fatigue in Francis runners. A crack will act as a stress raiser, such that stress around the crack is greater than if it was not there. Due to this, the S-N curves of welded members are shifted left relative to a non-welded specimen of the same material. In addition, there is no fatigue limit due to the imperfections introduced by the welding process. These adjusted S-N curves may be obtained from sources such as the Institute of Welding (IIW) [26].

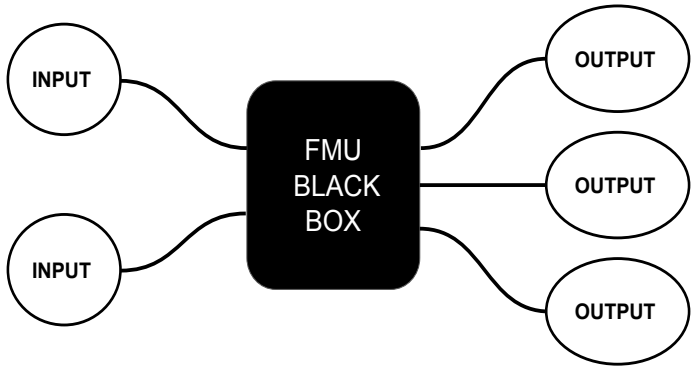


Figure 2.15: Visualisation of the FMU black box concept with two input variables and three output variables for this example.

2.6 Functional Mock-up Interface

The Functional Mock-up Interface (FMI) is a tool independent standard for easier exchange and co-simulation of dynamic models [27]. It can be viewed as an eco-system where models called Functional Mock-up Units (FMU) communicate. The two relevant aspects of the FMI functionality in this project are model exchange and co-simulation. For model exchange the implementation of the FMI generates code in the language, C. The models can be seen as "black boxes" taking inputs and passing outputs from and to other FMUs. The subsystems solve their given tasks and is governed by master algorithms. The master algorithms control the FMU exchanges within the FMI.

2.6.1 Functional Mock-up Unit

The implementations of the FMI are known as FMUs. Its these that convert the models from their respective modelling environments to interchangeable C-code. The core of the FMU is an XML-file called "modelDescription.xml" and every tool can read this in whichever preferred coding language.

2.7 Uncertainties

All measured values have an error associated with them. The error is the difference between a measured value and the actual value. They are divided into two groups; random and systematic. Random uncertainties are the fluctuations in measured values from randomness and follow a probability distribution. A systematic error

Error	Description
$\pm f_{X_a}$	Systematic error of the primary calibration method
$\pm f_{X_b}$	Random error of the primary calibration method
$\pm f_{X_c}$	Systematic error (repeatability) of the secondary instrument
$\pm f_{X_d}$	Random error of the secondary instrument
$\pm f_{X_c}$	Physical phenomena and external influences
$\pm f_{X_f}$	Error in physical properties

follows a pattern and can be accounted for to reduce its effect on the measurement result. A measured value is presented with an expected value $\pm$ its total error. The error is chosen from a confidence interval and is presented as a percentage. A confidence level of 95% would mean one would with 95% confidence conclude that an uncertain value will lie between the interval limits.

The uncertainties of a variable can be combined through the *Root-Sum-Square (RSS)* to give max uncertainty attributed to the measurement. Below are the encountered uncertainties when calibrating an instrument in the Waterpower Laboratory as an example, fetched from [28].

Below is the RSS presented for *n* number of uncertainties.

$$f_{\text{measurement}} = \pm\sqrt{f_1^2 + f_2^2 + ...f_n^2} \tag{2.18}$$

2.7.1 Probability distributions

Normal distribution

An uncertain parameter could possibly take on a number of so called distributions. The uncertain parameters of this model are expected to be normally distributed. This means the parameters have an expected value, known as the mean, and a standard deviation (SD), usually denoted σ.

Below is the equation for SD where x is a measured value, $\bar{x}$ is the mean of all measurements and N is number of measurements.

$$\text{SD} = \sqrt{\frac{\sum |x - \bar{x}|^2}{N}} \tag{2.19}$$

The probability of a parameter taking on a value outside the mean is decreasingly unlikely but 50% of all possible values of the parameter is above the mean and

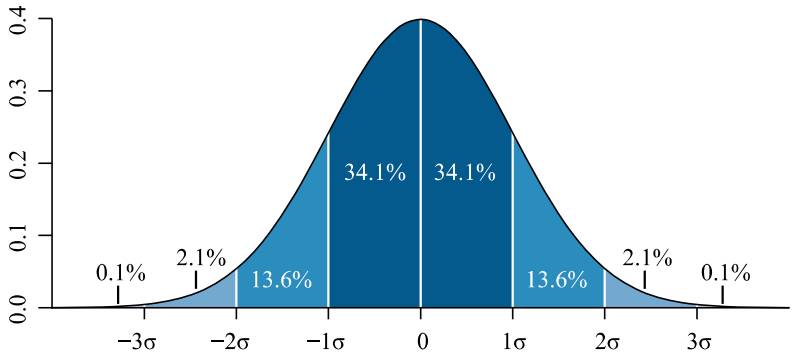


Figure 2.16: Normally distributed variable with mean equal zero, taking the shape of a Bell Curve. The y-axis gives the probability of a value on the x-axis occurring [29].

50% is below. If the mean and SD is known, a distribution of the probable values can consequently be plotted, and the graph is known as a "Bell Curve" shown in Figure 2.16. The SD decides how far the variable is likely to vary from the mean. A normal distribution can also be characterised by the "68-95-99.7-rule" where 68% of values lie within $\pm$ 1 SD, 95% within $\pm$ 2 SD and 99.7% within $\pm$ 3 SD.

Chi-squared distribution

Chi-squared distribution, χ^2_k, is a special case of the Gamma probability distribution. Compared to a normal distribution it has varying degree of positive skew. It is known by the number of degrees of freedom, k. The higher degree of freedom the more it appears as a normal distribution. A Chi-squared distribution is a result of underlying normally distributed variables.

2.7.2 Monte Carlo simulation

To evaluate the correlation between random input variables and their effect on the outcome, a Monte Carlo simulation is a useful tool. As one simulation will produce a different outcome than another, the results form a probability distribution. An array of N different results from N loops of the model is the end result of a Monte Carlo simulation. This outcome distribution can be treated as any probability distribution with an expected value, deviations and confidence intervals. Figure 2.18 is showing how a Monte Carlo simulation can be used in a Francis turbine lifetime estimator with probabilistic inputs.

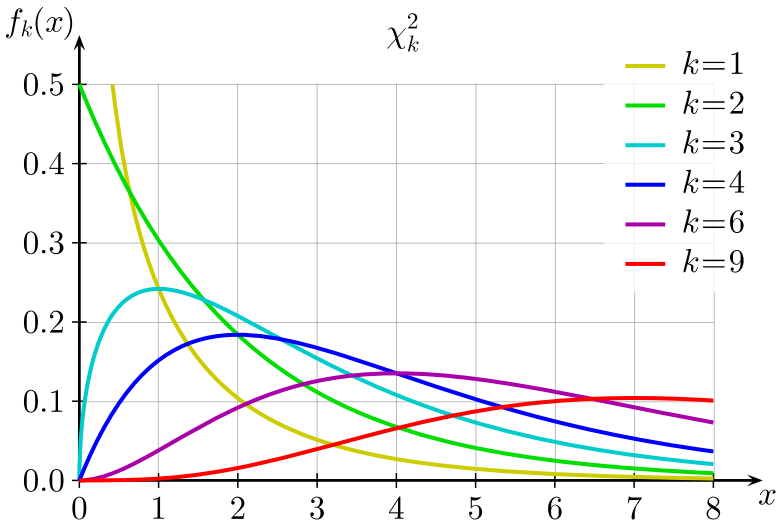


Figure 2.17: Chi-squared distribution with varying number of degrees of freedom, k. $f_k(x)$ is the probability of x occurring [30].

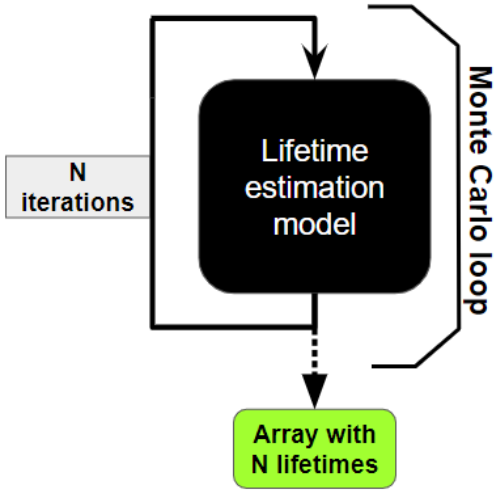


Figure 2.18: By looping over a lifetime estimation model with probabilistic input parameters, an array of results can be produced.

Chapter 3

Literature review

The search for relevant literature was executed with the same intentions as in the author's project work. This chapter is therefore close to verbatim to the project report: "A generic lifetime reduction model for Francis turbines" [8].

Fatigue in hydro turbines has been researched for nearly as long as they have been operated. Fluctuations in power output were recorded by W. J. Rheingans as early as 1940 [31]. It was shown that draft tube surges caused powerful, cyclic pressure amplitudes and affected the turbine runner performance. In the successive years, and especially these later decades, the interest of many hydropower researchers has been directed at Francis turbines operating at variable loads. This project will look at fatigue modelling of Francis turbines with aggressive operating patterns. Aggressive operating patterns are described by SINTEF researchers Welte & Solvang [32] in their study of new, possible operating patterns in Norwegian hydropower as:

- frequent starts and stops,
- frequent large load variations,
- frequent minimal load, part load, or full load operation.

The transmission system operators (TSOs), such as Statnett in Norway, project that balancing services will be frequently requested due to intermittent energy sources being increasingly admitted to the energy mix [33]. These balancing services include all types of operations presented by Welte & Solvang.

Turbines operating dynamically are argued to experience fatigue failure in a shorter time than those that do not [3, 4, 3]. On behalf of Voith Hydro, Seidel et al. [3]

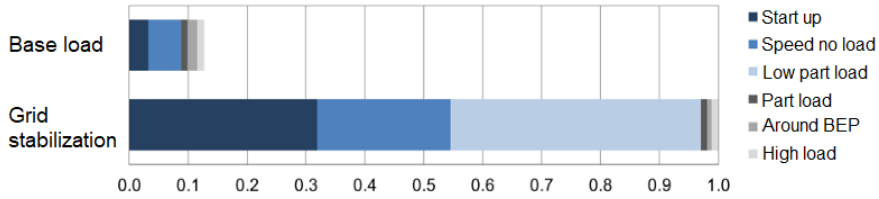


Figure 3.1: A comparison of relative damage contribution from base load and grid stabilisation operation on a prototype Francis runner from Voith Hydro [3].

tested a high head, prototype runner in two load regimes. The first load regime was a base load provider, and the second was a grid stabiliser. The stresses in the runner were found by strain gauge measurements and computational fluid dynamic (CFD) analyses. They concluded grid stabilising decreased fatigue life by as much as an order of magnitude, shown in Figure 3.1. Low part load was the most significant contributor, followed by start-ups. This result is considered an extreme and points to a considerable lifetime reduction in turbines that often provide grid balancing.

To have some context on why the case of Voith Hydro is an extreme, Welte & Solvang from SINTEF looked at the Norwegian yearly average of start-stop in both 1998 and 2007 [32]. The number of start-stop cycles in a year was on average 80 and 84, respectively. These numbers are considerably lower than Seidel et al., considering they used ten start-ups per day for the grid stabilisation case and one for the base load scheme. The SINTEF researchers point out that the average does not represent the individual turbines as some had doubled their number of start-stop cycles, and some halved their cycles. They continued by concluding it was the larger turbines (>100MW) that saw the most intermittent operation. In addition, Welte & Solvang did not investigate the fatigue impact of dynamical loads on the turbines, so the comparison stops at the number of start-stop cycles.

The work of Huang et al. in "Fatigue analyses of the prototype Francis runners based on site measurements and simulations" [34] from 2014 is of great value to this project. On behalf of Andritz Hydro, they did strain gauge measurements on five different turbines. The turbines ranged from Medium Power [<200 MW], High Power [200-400 MW], and Very High Power [>400 MW]. Huang and his team further backed observations that for operating points at 45% rated power, stress of stochastic character is replaced by the cyclic RSI and vortex rope stresses. The RSI stresses are of lower amplitudes but have a frequency 2-3 magnitudes greater than the stochastic ones. Seidel et al. (2012) [16] agrees with this observation as they conclude for higher head turbines, the RSI amplitudes dominate. Vortex rope frequency, known as Rheingans frequency, is in between the previous phenomena

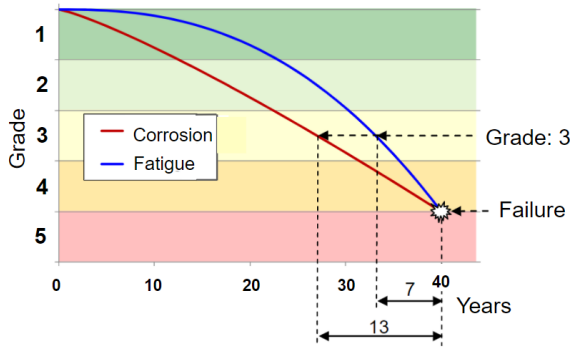


Figure 3.2: Development prediction of corrosion and fatigue damage types, showing for the same grade the remaining life is vastly different. Predictions of the maintenance interval is based on such models [7].

in terms of frequency and amplitude. One of their conclusions was that a single start-up does more damage to a turbine than a few hours of part load or speed-no-load, SNL. Their finite element method analysis supports the observation that the highest stresses are close to the runner blade trailing edge adjacent to the crown.

From the project "Value-adding maintenance in power production" of "Energy Norway", SINTEF set up an expert group on turbines to deliver the report "Failure model for hydropower plants: Failure mechanisms and condition criteria" [7]. This report elaborates on the methods for condition control and maintenance predictions. Today's practice is that either external inspectors or plant operators evaluate the condition and maintenance needs of a turbine. It goes into detail on how to assess the condition of the turbine and how the damage types are affected by the operating pattern (start-stop, different loads, etc.). Of all damage types, they give corrosion and fatigue precedence on a scale that prioritises damage types after their importance. When inspectors evaluate the condition of the turbine runner during maintenance, the next inspection is scheduled based on their damage assessment. Damage types are categorised from a grade 1-5, where 1 is pristine condition and 5 means that immediate replacement of the part is necessary. Based on the grade, the following inspection is scheduled after a given time. This method is called calendar-based maintenance planning. The projection of the next inspection follows different curves for the various damage types. Looking at corrosion and fatigue, their time evolution is somewhat different than the others. Corrosion has a weak exponential development, borderline linear, whereas fatigue develops exponentially. This difference is shown in Figure 3.2

Chapter 4

Methodology and method

The methodology of this project is to keep the model applicable to a generic Francis turbine runner and readily available to the public. To ensure this, a series of simplifications and assumptions were made throughout the project, and an FMU was constructed. The estimates and assumptions were made after consulting research employees at the Waterpower Laboratory, industry experts, and literature. Since the master's project is a direct continuation of the author's project work, the methodology is equivalent, and the method is predominantly the same as in the project thesis [8]. The model inputs are characterised as either constant or uncertain.

The first objective of the project was to acquire knowledge on the workings of Francis turbines. How they fatigue, established methods for lifetime estimations, and the FMI standard was examined. This happened through literature study and conversations with chosen staff members at the Waterpower Laboratory. The literature reviewed was provided mainly by Professor Dahlhaug, and the most influential literature is presented in Chapter 3. The acquisition of background knowledge was concentrated around the following points:

- dynamic loads on a Francis turbine,
- fluid structure interaction,
- fatigue design,
- condition assessment of Francis turbine,
- FMI and FMU.

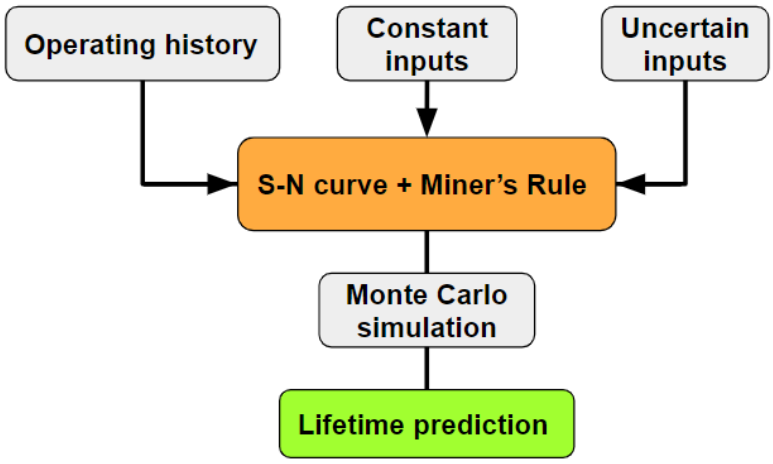


Figure 4.1: Model overview.

4.1 Example case and data processing

As necessary background theory was covered and stress values set, the focus shifted to the processing of raw data.

4.1.1 Bratsberg power plant

The decision was made to use an example power plant operated by Statkraft. This was done to access both operational and turbine design data more easily. Bratsberg power plant was chosen based on its geographical placement close to Trondheim. It has operated two 62 MW Francis turbines with 147 m head since 1977.

This enabled the Generic Life team to visit the plant and meet the power plant director. He confirmed the practice of manual inspection to decide the turbine condition and maintenance needs. Power outputs from Turbine 1, with the corresponding time stamp of measurements, were provided by Statkraft. A snippet of the operating data used is presented in Figure 4.3.

4.1.2 Operating data

The data supplied from Bratsberg stretched from 02.02.2016 to 13.10.2021. There were over 1.2 million data points of power output (MW) from this period, each with

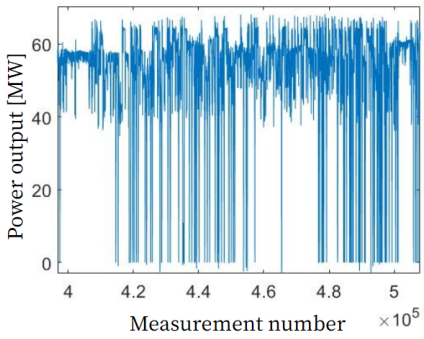
(a) Generator hall in Bratsberg hydropower plant.

(b) A replica of the Francis runner from Bratsberg hydropower plant.

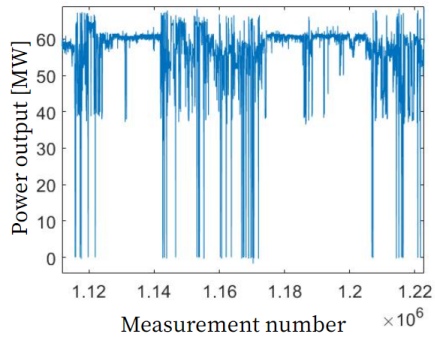
Figure 4.2: Generator hall of Bratsberg power plant, (a), and one of two Francis runners, (b).

	name	value	time_stamp	UTC_time
133598	Bratsberg\Unit_1|Power	56.52	1473775063	2016-09-13T13:57:43
133599	Bratsberg\Unit_1|Power	56.11	1473775126	2016-09-13T13:58:46
133600	Bratsberg\Unit_1|Power	56.44	1473775191	2016-09-13T13:59:51
133601	Bratsberg\Unit_1|Power	56.44	1473775200	2016-09-13T14:00:00
133602	Bratsberg\Unit_1|Power	57.34	1473775255	2016-09-13T14:00:55

Figure 4.3: Example of raw operating data provided by Statkraft. "Value" is power output from the turbine at time stamp in Universal Time Coordinated (UTC).



(a) A period of operating data showing frequent start-stop cycles.



(b) A period of operating data showing more stable operation around BEP.

Figure 4.4: Comparison of two arbitrary time intervals showing the operating patterns at Bratsberg power plant varying with time

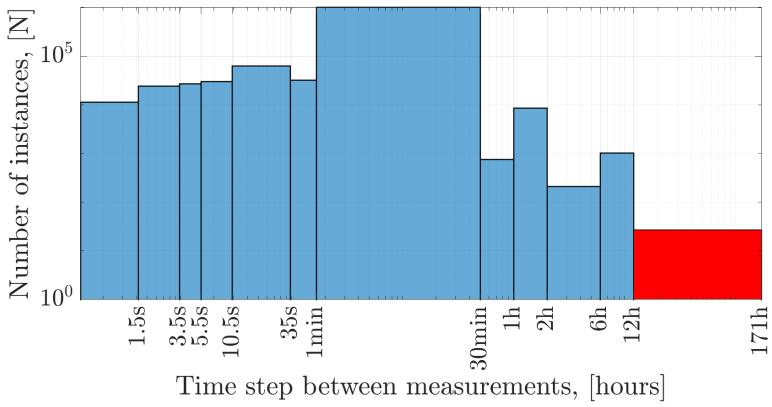


Figure 4.5: Histogram plot showcasing the distribution of the time steps between measurements. Width of the bins is the time step in hours, height of the bins is the number of instances with that time step. The red bin shows the excluded measurement interval.

a corresponding time stamp in UTC. The measurements were taken at irregular time steps ranging from one second to 170 hours. This meant the highest measurement resolution was in seconds, and the span in time steps was from 1 second to 170 hours. It was uncovered that for 27 time steps, the time period between measurements surpassed 12 hours. The fatigue contributions from these loads have not been considered. It is deemed too speculative to say anything about the turbine loads when time steps reach these lengths. The distribution of all the different time intervals is presented in Figure 4.5.

The power output measurements from Turbine 1 in the Bratsberg power plant occurred at irregular intervals. The bins in the histogram plot of Figure 4.5 encapsulate all measurement intervals within the bin edges. The purpose of the histogram plot is to showcase the wide range of time steps between measurements, so the bin limits do not affect the further analysis. The number of occurrences for each bin is presented in Table 4.1.

A simplification of the operating data from Figure 4.3 is shown in Table 4.2. In Equation 4.1, it is shown how the time between measurements equals the time spent at a single load for the turbine.

To explain how the time in each load interval was calculated, Table 4.2 is used as an example. For measurement number, i, the time spent at the corresponding power

Bin limits	Occurrences
0-1.5s	11336
1.5-3.5s	24312
3.5-5.5s	27227
5.5-10.5s	30035
10.5-35s	63094
35s-1min	32250
1-30min	1024940
0.5-1h	755
1-2h	8629
2-6h	208
6-12h	1032
12-24h	27

Table 4.1: Bin limits with number of occurrences in each bin.

Power	Time stamp
Output1	T1
Output2	T2
Output3	T3

Table 4.2: Simplified operating data.

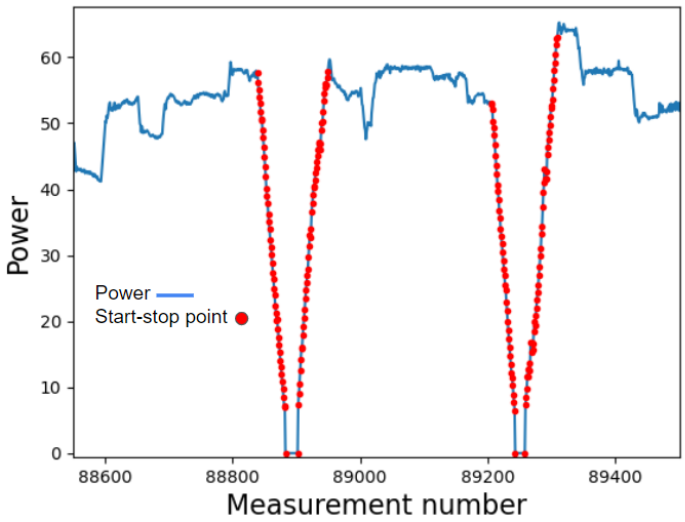


Figure 4.6: A snippet of the turbine power output showing two start-stop cycles and the registered start-stop points.

output is:

$$\text{Time(i)} = \text{T(i} + 1) - \text{T(i)} \tag{4.1}$$

As the start-stop damage contribution was treated as a separate event, they had to be counted. For a start-up of the turbine, it located the instances where power output equalled zero, and the subsequent measurement points had a positive gradient. For a turbine stop, the counter registered when the power again was zero and counted the previous power outputs when the power gradient was negative. All the points in either a start or a stop were removed from the fatigue calculation since they represent their own damage event. The start-stop handling and the points removed are shown in Figure 4.6. From the operating data, as shown in Figure 4.4, it was found that the Bratsberg runner experienced 710 start-stop cycles over 5.7 years, averaging 124 start-stops per year.

4.2 Model parameters

Model parameters are summed up in Table 4.3, clarifying which are treated as deterministic and which as probabilistic.

Deterministic	Probabilistic
Rotational speed	Start-stop damage
Number of guide vanes	Ramping damage
Load intervals	Rheingan's frequency
Operating data	Stress amplitudes
S-N curve constants	
Head	
BEP power	

Table 4.3: Table overview of all model inputs and whether they are treated deterministic or probabilistic.

4.2.1 Runner material and S-N curve

In order to analyse the fatigue contribution of the five-year measurement period, an S-N approach was chosen. The steel chosen as the runner material is 13Cr-4Ni. The highest stresses in the runner are found in the T-joint at the trailing edge adjacent to the runner shroud. A standard practice of runner production is to smooth the weld between blade and shroud. The S-N curve of a smooth butt-welded 13Cr-4Ni was found from the International Institute of Welding (IIW). The curve is presented in Figure 4.7 and curve constants are in Table 4.4, from IIW [26]. The curve in Figure 4.7 is plotted for a failure probability of 5%. IIW uses the following formula for S-N curves:

$$\Delta\sigma = \frac{C}{N}^{\frac{1}{m}}, \tag{4.2}$$

where $\Delta\sigma = 2\sigma_a$, C and m are material specific constants. The curve transitions to a more gentle slope at $N = 10^7$ as the fatigue regime change to high cycle fatigue (HCF). S-N curve used for this project is shown in Figure 4.7. The dotted line signifies the fatigue life curve of the 13Cr-4Ni weld at Variable Amplitude Loading (VAL) above 10^7 cycles.

Cycles	C	m
N $\leq 1e7$	$2.82e12$	3
N $> 10e7$, CAL	$9.063e46$	22
N $> 10e7$, VAL	$1.207e16$	5

Table 4.4: Curve constants for S-N curve of 13Cr-4Ni steel from IIW [26].

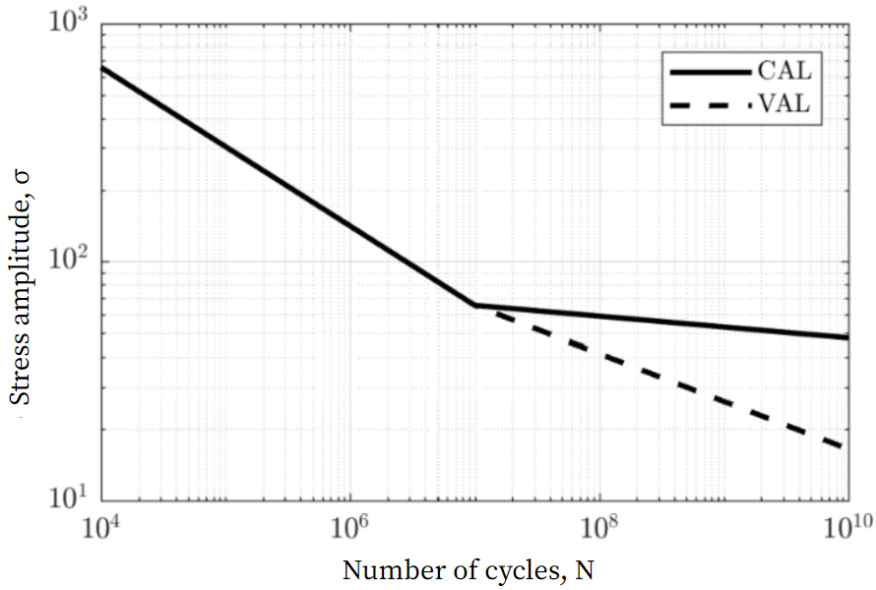


Figure 4.7: S-N curve for 13Cr-4Ni welded steel used in this project from Sannes [35] with curve constants from IIW [26]. Curves are created with a failure probability of $P_f \leq 5\%$.

4.2.2 Load intervals

An influential aspect of the model outcome is the definition of the load intervals. Load intervals have corresponding stress levels, and they must be well defined. The choice of load intervals is influenced by the work done by Mr. D. Sannes in his M.Sc. work [35] at the Waterpower Laboratory. Another influence in the finding of the intervals were pressure pulsation phenomena at the respective load ranges accounted for in section 2.3. In ascending order of power output from Minimal Load, Part Load, Best Efficiency Point, and Full Load. Presented in Table 4.5 as % of best efficiency load, $P = 57$ MW.

Load type	**Interval [% of BEP]**
ML	0 - 40
PL	40 - 90
BEP	90 - 110
FL	>110

Table 4.5: Load Intervals used in this project.

4.2.3 Stress amplitudes

The only damage type accounted for by the model is fatigue due to pressure pulsations in the water. This was done as fatigue is the single greatest damage contributor in Norwegian Francis runners. Values for stresses are obtained as a hybrid of a computational stress analysis from the Waterpower Laboratory [35] as well as field measurements of the current Bratsberg turbine runner [36]. These two sources combined the expert opinions of both industry experts and scientific staff at the Waterpower Laboratory. Although the analysis from the Waterpower Laboratory is of a small-scale model runner, the stress relations can be extrapolated to larger runners.

Goodman's correction (R=-1) was used to account for the mean stress. The accuracy of the results is evaluated as adequate for the use in this project. Emphasising that this project is foremost a proof of concept the exact stress values are not vital. They are as presented in Table 4.6.

Load interval	Stress amplitude [MPa]
ML	$15 \pm 20\%$
PL	$14 \pm 20\%$
BEP	$7.5 \pm 20\%$
FL	$11 \pm 20\%$

Table 4.6: Stress amplitudes used in this project [35].

4.2.4 Start-stop cycles and ramping

Transient operation such as start-stop cycles and ramping of the turbine causes significant lifetime reduction. In addition to reducing the refurbishment time of the generator by as much as 15 hours per start-stop [22], the turbine runner is argued to experience an equal amount of life reduction [25]. The start-stop damage contribution is therefore treated as following:

$$\Delta L_{\text{start-stop}} = \Delta D \cdot N_{\text{start-stop}}. \tag{4.3}$$

Where $\Delta L_{\text{start-stop}}$ is the lifetime reduction in hours, $\Delta D = 15 \pm 40\%$ hours and $N_{\text{start-stop}}$ is the number of start-stop cycles. A 15 hour lifetime reduction is treated in the model as 15 additional hours in BEP operation. Ramping points fatigue contributions were treated much the same way and was calculated as Eggen. A ramp up or ramp down is defined as a change of $0.25 \cdot P_{\text{nom}}$ from [25]:

$$\Delta L_{\text{ramping}} = k_{\text{ramp}} \cdot \Delta D \cdot N_{\text{ramping}}. \tag{4.4}$$

The factor, k (<1), is the loss of runner fatigue life based as a factor of ΔD. k is for this project set as $k = 0.2 \pm 40\%$. From correspondence with Mr. Eggen, the k-value is set to 0.2 by the author as there is no current certainty as to what this value is. The factor, k, is therefore assumed directly correlated to, but significantly less than, ΔD.

The 40% uncertainty related with both ramping and start-stop damage is evaluated as a credible uncertainty. This is mainly from [22] commenting that the 15 hours believed life reduction has a great uncertainty attributed to it. The ramping factor uncertainty is also set to 40% due to its proposed correlation to ΔD by [25].

4.2.5 Summary of design parameters

The Francis runner modelled in this simulation has design parameters listed in Table 4.7.

Parameter	Value
Rotational speed	300 rpm
Number of guide vanes	24
Material	13Cr4Ni steel
Head	130m
BEP power	57

Table 4.7: Overview of turbine design parameters.

4.3 Model construction

The lifetime estimation projected from the five-year time span was done using the previous listed parameters in Table 4.3. It consisted of six steps:

1. Multiply duration of each load interval by RSI and Rheingan's frequency to obtain number of cycles in each interval.
2. Count start-stop cycles and ramping points and add equivalent lifetime reduction in hours from these.
3. Apply correct stress value for each interval.
4. Use S-N, effective stress amplitude and number of cycles to evaluate life reduction.
5. Sum up each damage contribution using Miner's rule to obtain total life reduced.
6. Use Monte Carlo simulation to investigate the impact of the probabilistic parameters and project the lifetime of each case.

As specified, the FMU had a number of outputs. The outputs and their values are presented in Table 4.8.

Life reduced is the Miner's sum from the input operating time. It is this quantity that decides the projected lifetime as shown in Equation 4.5.

Output	Value
ML cycles	$3.97 \cdot 10^6$
PL cycles	$2.17 \cdot 10^9$
BEP cycles	$1.55 \cdot 10^10$
FL cycles	$2.22 \cdot 10^9$
Start-stops	710
Life reduced	Array
Lifetime projection	Array

Table 4.8: Outputs chosen in this project and their values.

$$\text{Projected lifetime} = \frac{1}{\text{Life reduced}} \cdot \text{Operating time} \tag{4.5}$$

From the Monte Carlo simulation an array of lifetime estimations by Equation 4.5 was constructed. This array was considered the result of the model.

Chapter 5

Results and discussions

The thesis results are divided into two separate parts. First is the proof of concept of constructing a Francis turbine fatigue model as an FMU, and second is the results from the analysis.

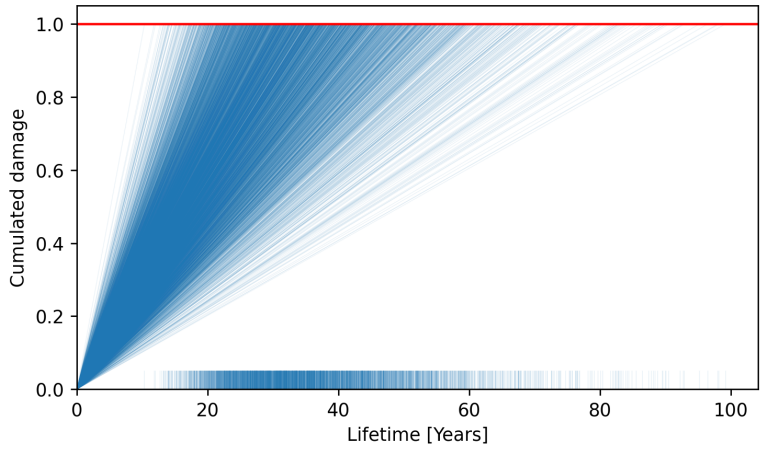
The importance of the FMI standard is further strengthened the more widespread usage it gets. Due to the implementation of this model as an FMU, future developers now have unrestricted access to a lifetime estimation model. In addition, parts of the model can be removed and coupled with other FMUs to increase the quality of the results incrementally. The hydropower industry, academic researchers, and other stakeholders can utilize this model and project report as a baseline for further work. This is precisely what the overall objective was. So, in light of this, the results are deemed successful. For the project development, the documentation in the FMU and scripts in the appendix will potentially be of great use.

The choice of stress values is the single most crucial factor for fatigue analysis results. The higher the stress values, the shorter the fatigue life [20]. It is not arbitrary what these values are, but it was not crucial that they were exact for this project. Emphasising that this project sought a generalised method, a goal of painstakingly accurate input parameters was not necessary. With that said, the values combined from both field measurements [36] and the Waterpower Laboratory [35] are deemed adequate for the sake of this project. Even though one achieves very low relative uncertainties in strain gauge measurements on a model runner under controlled circumstances, they are not directly applicable to a prototype runner. It would undoubtedly be best if the stress amplitudes used in a simulation would be of strain gauge measurements or precise FEM analyses. However, field measurements are expensive and laborious. To do them at the hundreds of Norwegian hydropower plants would not be economically viable. The other alternative of numerical analysis seems to be a more realistic option. Computing power is still

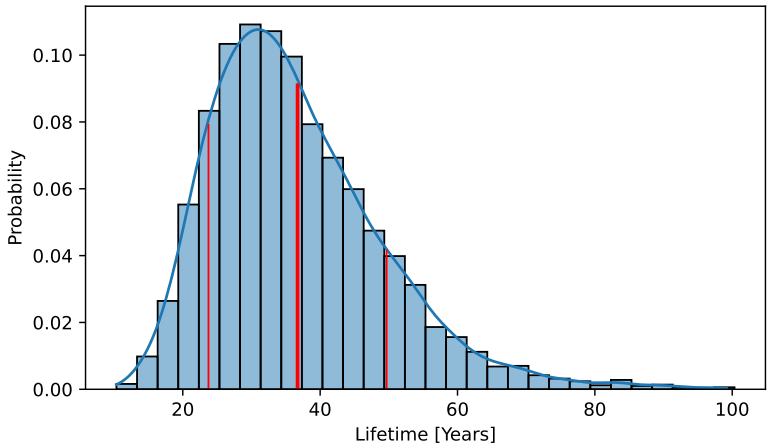
in solid development, and highly accurate modelling tools will become available in the future. This last point also supports the implementation of this project as an FMU. The optimal scenario is where a model user knows the stress amplitudes in the modelled runner with great certainty. This project will be available for anyone who wishes to develop it further and replace inaccurate values with direct inputs or other FMUs. The potential for this model is, in other words, limitless.

The final results of the fatigue analysis are shown in Figure 5.1(a) and 5.1(b). Figure 5.1(a) shows the lifetime projections of individual simulations within the Monte Carlo loop. The projections are based on the operating pattern of the Bratsberg power plant. On the x-axis, the x-coordinate of where each lifetime projection crosses the y = 1 line marked with blue ticks on the x-axis. 5000 Monte Carlo simulations appear to be a good amount as the distribution is close to continuous. The red line at y = 1 signifies a Miner's sum equal 1 from the Palmgren-Miner summation. From the S-N data used in this project, this is where the material will experience fatigue failure. In reality, there is only a 5% chance the material will fail at this point due to the nature of the S-N curve in use. A safety margin of 5% is the industry standard, and as previously commented, this means the chance of failure at this point is no more than 5%. The life of a specimen is nonetheless considered spent and needs replacement. Figure 5.1(b) shows the expected lifetime of L = 36.7 years. This is the weighted average of the distribution with a standard deviation of 13.0 years. The total distribution of lifetimes in histogram form is shown in Figure 5.1(b). Here it is seen how the different combination of probabilistic input parameters are expressed in varying lifetime estimates. It shows that with precise inputs, a highly accurate lifetime estimate can be calculated. By this, the results are uplifting and enhance the belief in further model development.

Another factor that plays a crucial role in fatigue life analysis is fatigue crack growth. From Richard & Sander [37] we know the fatigue life of a component is divided into crack initiation and propagation phases. Where the crack propagation phase is the majority of component fatigue life. One of the main assumptions and the criteria for using the Palmgren-Miner rule is constant fatigue development. In reality, a runner will live its fatigue life with fatigue cracks present. These cracks impact the maximum stress amplitudes due to the crack tip acting as a stress raiser [37, 20]. In reality, the lifetime cannot be projected as purely linear due to the fatigue cracks. In the author's opinion, this is a major limitation in the lifetime estimation. Subsequent work on either this model or a similar project will probably increase the accuracy of the estimates tremendously if fatigue crack growth is implemented. Fatigue crack development is a micro phenomenon compared to the more holistic S-N approach. This thesis therefore, did not include crack development from fear of losing model generality.



(a)



(b)

Figure 5.1: Figure 5.1(a) shows the Monte Carlo simulation with N = 5000 iterations of the linear lifetime projections. Figure 5.1(b) shows a histogram distribution of all lifetimes with bin width = 3.

For the purpose of this project the use of the Palmgren-Miner rule was deemed appropriate. There are several assumptions in the P-M rule that ignores certain factors impacting fatigue life. For instance, the order of variable amplitude loading cycles is not considered. This is important because cycles with higher stress amplitudes can initiate cracks, and subsequent cycles may develop these cracks, even though the amplitudes are relatively low. Further, linearity of damage accumulation is assumed as seen in the lines in Figure 5.1(a). There is also no consideration for environmental effects such as corrosion, erosion, or temperature.

Chapter 6

Conclusions

If intermittent energy sources are further implemented in the power grid, turbines will have to operate in sub-optimal configurations. This project therefore investigated a simple method for estimating the fatigue life based on the load history of any Francis turbine. The investigation revealed that it is possible to make a lifetime estimate based on a model using just a few inputs. To achieve this, the theoretical knowledge of relevant subjects was acquired, and were of great use. The project has hopefully started the work of filling a knowledge gap found in literature and previous projects.

In summary, the project concludes that there is valuable potential for a generic lifetime estimation model of Francis turbines. The value for the hydropower industry is potentially significant as the model can be applied broadly. If the model reaches sufficient accuracy, the cost of specific operating patterns would consequentially be precisely calculated.

The most important conclusions drawn from this project thesis are:

- Stress amplitudes from experimental measurements might be extrapolated to other Francis runners.
- Data from an existing hydropower plant can be used to evaluate the lifetime.
- A generic model was constructed that could process power plant data and output the expected lifetime.
- Based on the results of the model, the lifetime of the investigated turbine is expected to operate for 36.7 years with standard deviation 13.0 years.
- The limitations of the methods and the assumptions of the model has been identified, and addressed.

- Both python script and FMU is available and intended for further development.
- Concrete measures for model improvement, have been mapped for future work.

Chapter 7

Future work

As stated before, the possibilities and potential for this project are significant. This thesis was the continued work of the author's project thesis, and hopefully the Generic Life project will push on. A list of possibilities for future work is presented below.

- Find a common parameter of Francis turbines to achieve a higher accuracy in the generalisation.
- Consider fatigue crack propagation in fatigue life estimation.
- Continue work on the FMU to improve interchangeability and co-simulation capabilities.
- Test the FMU with other FMUs.

References

[1] "Paris Agreement," accessed 2021-10-28, https://treaties.un.org/pages/ViewDetails.aspx?src=TREATY&mtdsg_no=XXVII-7-d&chapter=27&clang=_en

[2] DNV, 2021, "Energy Transition Norway 2050," Tech. rep., Norsk Industri, P. 29.

[3] Seidel, U., Mende, C., Hübner, B., Weber, W., and Otto, A., 2014, "Dynamic loads in Francis runners and their impact on fatigue life," IOP Conference Series: Earth and Environmental Science, **22**.

[4] Gummer, J. and Etter, S., 2008, "Cracking of Francis runners during transient operation," **15**, pp. 81–85.

[5] Storli, P. T. and Nielsen, T. K., 2014, "Dynamic load on a Francis turbine runner from simulations based on measurements," Institute of Physics Publishing, doi:10.1088/1755-1315/22/3/032056.

[6] Eivind Solvang, M. I. B. B. T.-J. F., Dag Erik Nordgård, 2011, "Verdiskapende vedlikehold innen kraftproduksjon: Utfordringer og anbefalinger," EnergiNorge AS, (323-2011).

[7] T. Welte, E. S., J. Heggset, 2011, "Sviktmodell for Vannkraftverk: Skadetyper og tilstandskriterier," EnergiNorge AS - Energiakademiet, (323-2011).

[8] Hvistendahl, E., 2021, "A generic lifetime reduction model for Francis turbines," Tech. rep., Waterpower laboratory, NTNU, Trondheim.

[9] Brekke, H., 2001, *Hydraulic Turbines - Design, Erection and Operation*, Norwegian University of Science and Technology, Trondheim.

[10] Brekke, H., 2003, *Pumper Og Turbiner*, Vannkraftlaboratoriet NTNU, Trondheim.

[11] Hasmatuchi, V., 2012, "Hydrodynamics of a Pump-Turbine Operating at Off-Design Conditions in Generating Mode," Ph.D. thesis, École Polytechnique Fédérale de Lausanne.

[12] Lewis, B. J., Cimbala, J. M., and Wouden, A. M., 2014, "Major historical developments in the design of water wheels and Francis hydroturbines," IOP Conference Series: Earth and Environmental Science, **22**.

[13] Valkvæ, I., 2016, "Dynamic loads on Francis turbines," Master's thesis, NTNU Trondheim, Waterpower Laboratory, Gløshaugen.

[14] Kjølle, A., 2001, "Hydropower in Norway Mechanical Equipment A survey prepared," .

[15] Centre, N. H., 2014, "Test case, Numerical Study," 18.11.2021, https://www.ntnu.edu/nvks/f99-numerical-study

[16] Seidel, U., Hübner, B., Löfflad, J., and Faigle, P., 2012, "Evaluation of RSI-induced stresses in Francis runners," IOP Conference Series: Earth and Environmental Science, **15**.

[17] Cengel, Y. A. and Cimbala, J. M., 2014, *Fluid Mechanics: Fundamentals and applications*, McGraw-Hill Education, New York, NY.

[18] Pope, S. B., 2000, *Turbulent Flows*, Cambridge University Press, Cambridge, UK.

[19] Kobro, E., 2010, "Measurement of Pressure Pulsations in Francis Turbines," Ph.D. thesis, NTNU, Trondheim.

[20] Dowling, N. E., 2013, *Mechanical Behavior of Materials*, Pearson Education Limited, United Kingdom, Harlow.

[21] Alves, H., Otani, L., Segundinho, P., and Morales, E., 2015, "Elastic moduli characterization of wood and wood products using the Impulse Excitation Technique," , p. 23.

[22] Bakken, B. H. and Bjorkvoll, T., 2002, "Hydropower unit start-up costs," *IEEE Power Engineering Society Summer Meeting,* (Vol. 3), IEEE, pp. 1522–1527.

[23] Nilsson, O. and Sjelvgren, D., 1997, "Hydro unit start-up costs and their impact on the short term scheduling strategies of Swedish power producers," IEEE Transactions on power systems, **12**(1), pp. 38–44.

[24] Nilsson, O. and Sjelvgren, D., 1997, "Variable splitting applied to modelling of start-up costs in short term hydro generation scheduling," IEEE Transactions on Power Systems, **12**(2), pp. 770–775.

[25] Eggen, A. O., 2021, "Modell for kjøremønsterrelaterte kostnader," Tech. rep., HydroCen, Trondheim.

[26] Hobbacher, A. et al., 2016, *Recommendations for fatigue design of welded joints and components*, Vol. 47, Springer.

[27] Blochwitz, T., Otter, M., Arnold, M., Bausch, C., Clauß, C., Elmqvist, H., Junghanns, A., Mauss, J., Monteiro, M., Neidhold, T., et al., 2011, "The functional mockup interface for tool independent exchange of simulation models," *Proceedings of the 8th international Modelica conference*, Linköping University Press, pp. 105–114.

[28] Solemslie, B. W., 2001, "Compendium in Instrumentation, Calibration Uncertainty Analysis," .

[29] M. W. Toews, Wikimedia Commons, 2007, "Standard deviation diagram," [Online; accessed May 11, 2022], https://commons.wikimedia.org/wiki/File:Standard_deviation_diagram.svg

[30] Geek3, Wikimedia Commons, 2010, "Plot of the chi-square distribution," [Online; accessed May 13, 2022], https://commons.wikimedia.org/wiki/File:Chi-square_pdf.svg

[31] Rheingans, W. J., 1940, "Power swings in hydroelectric power plants," Trans. ASME, **62**(3), pp. 171–184.

[32] Welte, T. and Solvang, E., 2011, "Endring av kjøremønster i norske vannkraftverk," **Nr 320-2011**.

[33] Statnett 2017, "The Nordic Balancing Concept," .

[34] Huang, X., Chamberland-Lauzon, J., Oram, C., Klopfer, A., and Ruchonnet, N., 2014, "Fatigue analyses of the prototype Francis runners based on site measurements and simulations," IOP Conference Series: Earth and Environmental Science, **22**, p. 012014.

[35] Sannes, D. B., 2018, "Pressure Pulsation and Stresses in a Francis Turbine Operating at Variable Speed," Master's thesis, NTNU Trondheim, Waterpower Laboratory, Gløshaugen.

[36] Bjørndal, H., Moltubakk, T., and Aunemo, H., 2001, "Flow induced stresses in a medium head Francis runner – Strain gauge measurements in an operating plant and comparison with Finite Element Analysis," .

[37] Richard, H. A. and Sander, M., 2016, *Fatigue crack growth*, Springer.

Appendices

Appendix A

A.1 Lifetime estimator FMU

```
from pythonfmu import Fmi2Causality, Fmi2Slave,
↪  Boolean, Real, String, Fmi2Variability
import numpy as np
import pandas as pd




class Francis_lifetime_estimator(Fmi2Slave):

    author = "Student: Eivind Hvistendahl & Supervisor:
    ↪  Bjørn Winther Solemslie"
    description = "Generic lifetime estimator for
    ↪  Francis turbine runners"

    def __init__(self, **kwargs):
        super().__init__(**kwargs)


# ====================================
#DECLARATION OF INPUT VARIABLES
# ====================================



```

```
23
24          self.Hn = 130 #Nominal head
25          self.register_variable(Real("Hn",
            ↪  causality=Fmi2Causality.input))
26
27
28          self.BP = 57 #MW at best-point
29          self.register_variable(Real("BP",
            ↪  causality=Fmi2Causality.input))
30
31          self.Pnom = 62 #Nominal power
32          self.register_variable(Real("Pnom",
            ↪  causality=Fmi2Causality.input))
33
34          self.k_ramp_base = 0.2 #Ramping factor of
35          self.register_variable(Real("k_ramp_base",
            ↪  causality=Fmi2Causality.local))
36
37          self.delta_D_start_stop_base = 15 #Cost in
            ↪  reduced lifetime of one start-stop
38
            ↪  self.register_variable(Real("delta_D_start_stop_base",
            ↪  causality=Fmi2Causality.local))
39
40          self.rpm = 300 #Rotational speed
41          self.register_variable(Real("rpm",
            ↪  causality=Fmi2Causality.input))
42
43          self.f_rph = self.rpm*60 #/hour
44          self.register_variable(Real("f_rph",
            ↪  causality=Fmi2Causality.local))
45
46          self.z_GV = 24 #Guide vanes
47          self.register_variable(Real("z_GV",
            ↪  causality=Fmi2Causality.input))
48
49          self.filename = 'Unit1_bratsberg.csv'
50          self.register_variable(String("filename",
            ↪  causality=Fmi2Causality.input,
            ↪  variability=Fmi2Variability.discrete))
51
```

```
52
53          self.unit_name = 'Bratsberg\\Unit_1|Power'
54          self.register_variable(String("unit_name",
            ↪  causality=Fmi2Causality.local))
55
56
57          # Cycles below 1e7
58          self.C1 = 2.82e12
59          self.register_variable(Real("C1",
            ↪  causality=Fmi2Causality.input))
60          self.m1 = 3
61          self.register_variable(Real("m1",
            ↪  causality=Fmi2Causality.input))
62
63          # For cycles above 1e7 VAL (Sannes)
64          self.m2 = 5
65          self.register_variable(Real("m2",
            ↪  causality=Fmi2Causality.input))
66          self.C2 = 1.207e16
67          self.register_variable(Real("C2",
            ↪  causality=Fmi2Causality.input))
68
69
70          # Load intervals, MW
71          self.ML = 0.4*self.BP # Minimum load
72          self.register_variable(Real("ML",
            ↪  causality=Fmi2Causality.local))
73
74          self.PL = 0.9*self.BP # Part load
75          self.register_variable(Real("PL",
            ↪  causality=Fmi2Causality.local))
76
77          self.FL = 1.1*self.BP #Full load
78          self.register_variable(Real("FL",
            ↪  causality=Fmi2Causality.local))
79
80
81          # Pressure oscillation frequencies
82          self.f_RSI = self.f_rph * self.z_GV #RSI
            ↪  frequenecy per hour
```

```
83          self.register_variable(Real("f_RSI",
            ↪  causality=Fmi2Causality.local))
84
85          self.f_s_base = 0.33*self.f_rph #Rheingans
            ↪  freq
86          self.register_variable(Real("f_s_base",
            ↪  causality=Fmi2Causality.local))
87
88
89          # Stresses, MPa
90          self.ML_stress_base = 15
91          self.register_variable(Real("ML_stress_base",
            ↪  causality=Fmi2Causality.local))
92
93          self.PL_stress_base = 14
94          self.register_variable(Real("PL_stress_base",
            ↪  causality=Fmi2Causality.local))
95
96          self.BP_stress_base = 7.5
97          self.register_variable(Real("BP_stress_base",
            ↪  causality=Fmi2Causality.local))
98
99          self.FL_stress_base = 11
100         self.register_variable(Real("FL_stress_base",
            ↪  causality=Fmi2Causality.local))
101
102         # Vortex rope
103         self.VRTX_stress = 0.03*self.Hn
104         self.register_variable(Real("VRTX_stress",
            ↪  causality=Fmi2Causality.local))
105
106 # ====================================
107 # DECLARATION OF OUTPUT VARIBALES
108 # ====================================
109
110         self.BP_cycles = 10000000.
111         self.register_variable(Real("BP_cycles",
            ↪  causality=Fmi2Causality.output))
112
113         self.FL_cycles = 10000000.
```

```
114             self.register_variable(Real("FL_cycles",
                ↪  causality=Fmi2Causality.output))
115
116             self.PL_cycles = 10000000.
117             self.register_variable(Real("PL_cycles",
                ↪  causality=Fmi2Causality.output))
118
119             self.ML_cycles = 10000000.
120             self.register_variable(Real("ML_cycles",
                ↪  causality=Fmi2Causality.output))
121
122             self.BP_time = 10000.
123             self.register_variable(Real("BP_time",
                ↪  causality=Fmi2Causality.output))
124
125             self.ML_time = 10000.
126             self.register_variable(Real("ML_time",
                ↪  causality=Fmi2Causality.output))
127
128             self.PL_time = 10000.
129             self.register_variable(Real("PL_time",
                ↪  causality=Fmi2Causality.output))
130
131             self.FL_time = 10000.
132             self.register_variable(Real("FL_time",
                ↪  causality=Fmi2Causality.output))
133
134             self.total_time = 10000.
135             self.op_time = 10000.
136             self.down_time = 10000.
137
138             self.total_stars = 100
139             self.total_stops = 100
140
141             self.life_reduced = 0.0
142             self.ref_life_reduced = 0.0
143
144             self.start_stop_life_reduction = 100.
145             self.register_variable(
                ↪  Real("start_stop_life_reduction",
                ↪  causality=Fmi2Causality.output))
```

```
146
147         self.ramping_life_reduction = 100
148         self.register_variable(
            ↪  Real("ramping_life_reduction",
            ↪  causality=Fmi2Causality.output))
149
150
151
152         self.ML_time = 100
153         self.PL_time = 100
154         self.BP_time = 100
155         self.FL_time = 100
156
157         self.ML_cycles = 1000000
158         self.PL_cycles = 1000000
159         self.vortex_cycles = 1000000
160         self.register_variable(Real("vortex_cycles",
            ↪  causality=Fmi2Causality.output))
161
162         self.BP_cycles = 1000000
163         self.FL_cycles = 1000000
164         self.cycles_array = []
165 # ====================================
166 # DECLARATION OF LOCAL VARIBALES
167 # ====================================
168         self.df = pd.DataFrame()
169         #self.df = pd.DataFrame()
170
171         self.pwr = 100.
172         self.register_variable(Real("pwr",
            ↪  causality=Fmi2Causality.local))
173
174         self.time_stamp = 100.
175         self.register_variable(Real("time_stamp",
            ↪  causality=Fmi2Causality.local))
176
177         self.pwr_grad = []
178         self.time_grad = []
179
180         self.minPower_mask = [True]
```

```
181             self.register_variable(Boolean("minPower_mask",
                ↪  causality=Fmi2Causality.local))
182
183             self.total_stops = 0
184             self.register_variable(Real("total_stops",
                ↪  causality=Fmi2Causality.local))
185
186             self.total_starts = 0
187             self.register_variable(Real("total_starts",
                ↪  causality=Fmi2Causality.local))
188
189             self.in_startUp = False
190             self.register_variable(Boolean("in_startUp",
                ↪  causality=Fmi2Causality.local,
                ↪  variability=Fmi2Variability.discrete))
191
192             self.in_shutDown = False
193             self.register_variable(Boolean("in_shutDown",
                ↪  causality=Fmi2Causality.local,
                ↪  variability=Fmi2Variability.discrete))
194
195             self.start_stop_array = []
196             self.rampingarray = []
197
198             self.num_rampings = 0
199             self.register_variable(Real("num_rampings",
                ↪  causality=Fmi2Causality.local))
200
201             self.total_time = 420
202             self.register_variable(Real("total_time",
                ↪  causality=Fmi2Causality.local))
203
204             self.sigma_A = []
205             self.sigma_A_ref = []
206
207             self.N1 = []
208             self.Nref = []
209
210             self.counter = 1
211             self.register_variable(Real("counter",
                ↪  causality=Fmi2Causality.local))
```

```
212
213         self.life_reduced = 0.0
214         self.register_variable(Real("life_reduced",
            ↪  causality=Fmi2Causality.output))
215
216         self.reference_life_reduced = 0.0
217
            ↪  self.register_variable(Real("reference_life_reduced",
            ↪  causality=Fmi2Causality.output))
218
219         self.projected_lifetime = 0.0
220
            ↪  self.register_variable(Real("projected_lifetime",
            ↪  causality=Fmi2Causality.output))
221
222 # ====================================
223 # REGISTRATION OF LOCAL VARIBALES
224 # ====================================
225
226         self.df = pd.read_csv(self.filename, sep=',',
            ↪  index_col=[0])
227         self.UNIT1mask = self.df.name == self.unit_name
228         self.df = self.df.loc[self.UNIT1mask]
229         self.pwr = self.df.value
230         self.time_stamp = self.df.time_stamp
231         self.pwr_grad =
            ↪  np.gradient(np.array(self.df.value.tolist(),ndmin=1))
232         self.time_grad = np.gradient(np.array(
            ↪  self.df.time_stamp.tolist(),ndmin=1))/3600
233
234         #Ignores all power outputs below 10\% of FL
235         minPower_mask = self.minPower_mask
236         pwr = self.pwr
237         FL = self.FL
238         minPower_mask = pwr[:] < 0.1*FL
239         pwr.loc[minPower_mask]=0
240         self.pwr = pwr
241         self.minPower_mask = minPower_mask
242
243 # Start of Monte Carlo iteration
244
```

```
245     def do_step(self, current_time, step_size):
246
247 # ====================================
248 # Functions
249 # ====================================
250
251         def check_stop(pwr,i):
252             #In case first value of power output is 0,
                ↪  the funstion will go out of range
                ↪  unless first condition
253             if i > 7  and pwr[i-1] > 0  and pwr[i-2] >
                ↪  0  and pwr[i-3] > 0  and pwr[i-4] > 0
                ↪  and pwr[i-5] > 0  and pwr[i-6] > 0 :
254                 return True
255             else:
256                 return False
257
258         def check_start(pwr, i):
259
260             if  (pwr[i+1] > 0) and (pwr[i+2] > 0) and
                ↪  (pwr[i+3] > 0) and (pwr[i+4] > 0) and
                ↪  (pwr[i+5] > 0) and (pwr[i+6] > 0):
261
262                 return True
263             else:
264
265                 return False
266
267         def check_ramping(pwr, i):
268             if
                ↪  abs(self.pwr_grad[i]/(self.time_grad[i]*60))
                ↪  >= 0.25*self.Pnom:
269                 return True
270             else:
271                 return False
272 # ====================================
273 # Setting the uncertainties of wanted varibales
274 # ====================================
275 #The following uncertainties directly decide the
    ↪  uncertainty of the lifetime estimate
```

```
276 #if the estimated lifetime is desired to have a low
    ↪  varinace, the following variables
277 #has to be precisely decided.
278
279
280         delta_D_start_stop =
            ↪  np.random.normal(self.delta_D_start_stop_base,
            ↪  0.2*self.delta_D_start_stop_base,
            ↪  size=None)
281         k_ramp = abs(np.random.normal(self.k_ramp_base,
            ↪  0.2*self.k_ramp_base, size=None))
282         f_s = np.random.normal(self.f_s_base,
            ↪  0.1*self.f_s_base, size=None)
283
284
285         self.delta_D_start_stop = delta_D_start_stop
286         self.k_ramp = k_ramp
287         self.f_s = f_s
288
289         ML_stress =
            ↪  np.random.normal(self.ML_stress_base,
            ↪  0.1*self.ML_stress_base, size=None)
290         PL_stress =
            ↪  np.random.normal(self.PL_stress_base,
            ↪  0.1*self.PL_stress_base, size=None)
291         BP_stress =
            ↪  np.random.normal(self.BP_stress_base,
            ↪  0.1*self.BP_stress_base, size=None)
292         FL_stress =
            ↪  np.random.normal(self.FL_stress_base,
            ↪  0.1*self.FL_stress_base, size=None)
293
294 # ====================================
295 # Start-Stop and ramping
296 # ====================================
297         self.total_stops = 0
298         self.total_starts = 0
299         self.in_startUp = False
300         self.in_shutDown = False
301         self.start_stop_array = []
302         self.rampingarray = []
```

```
303         self.num_rampings = 0
304
305         pwr = self.pwr
306
307         for i, num in enumerate(pwr):
308             if num <= 0:
309                 if check_start(self.pwr,
                    ↪  self.pwr.index[i]):
310                     self.total_starts += 1
311
312                     self.rampiterator = pwr.index[i]
313                     self.in_startUp = True
314                     while (self.in_startUp):
315 #Includes points in startup config and removes them
    ↪  from operation
316                         if ( pwr.loc[self.rampiterator
                            ↪  + 5] -
                            ↪  pwr.loc[self.rampiterator]
                            ↪  > 0 ):
317
                                ↪  self.start_stop_array.append(
                                ↪  self.rampiterator)
318
                                ↪  self.pwr.loc[self.rampiterator]
                                ↪  = 0
319
320 #The line above makes sure start-stop is not counted as
    ↪  rampings since these points are treated in a
    ↪  separate damage phenomenon from delta_D_start_stop
321
322                             self.rampiterator += 1
323                         else:
324                             self.in_startUp = False
325                 if check_stop(pwr, pwr.index[i]):
326                     self.total_stops += 1
327
328                     self.rampiterator = pwr.index[i]
329                     self.in_shutDown = True
330
331                     while (self.in_shutDown):
```

```
332                             if (pwr.loc[self.rampiterator -
                                ↪  2] -
                                ↪  pwr.loc[self.rampiterator]
                                ↪  > 0):
333
                                   ↪  self.start_stop_array.append(
                                   ↪  self.rampiterator)
334
                                   ↪  self.pwr.loc[self.rampiterator]
                                   ↪  = 0
335 #The line above makes sure start-stop is not counted
336 #as rampings since these points are treated in a
    ↪  separate
337 #damage phenomenenon from delta_D_start_stop
338                                self.rampiterator -= 1
339                             else:
340                                self.in_shutDown = False
341             if check_ramping(self.pwr, i):
342                 self.rampingarray.append(pwr.index[i])
343                 self.num_rampings = self.num_rampings +
                    ↪  1
344         self.start_stop_array =
            ↪  np.array(self.start_stop_array)
345
346         self.time_stamp = np.array(self.df.time_stamp)
347         self.total_time =
            ↪  (self.time_stamp[len(self.time_stamp)-1]-self.time_stamp[
            ↪  #in hours
348
349         self.total_time_years =
            ↪  (self.total_time/24)/365
350
351         self.down_time =
            ↪  sum(self.time_grad[((self.pwr[:] == 0))])
352         self.op_time = sum(self.time_grad[((self.pwr[:]
            ↪  != 0))])
353
354         self.pwr.loc[self.start_stop_array] = 0 #Points
            ↪  in start-stop not counted as operating
            ↪  points
355
```

```
356
357         self.ML_time = sum(self.time_grad[((self.pwr[:]
            ↪  > 0) & (self.pwr[:] <= self.ML))])
358         self.PL_time = sum(self.time_grad[((self.pwr[:]
            ↪  > self.ML) & (self.pwr[:] <= self.PL))])
359         self.BP_time = sum(self.time_grad[((self.pwr[:]
            ↪  > self.PL) & (self.pwr[:] <= self.FL))])
360         self.FL_time = sum(self.time_grad[((self.pwr[:]
            ↪  > self.FL))])
361
362         #Contribution from start_stops and ramp:
363
364         self.start_stop_life_reduction =
            ↪  self.delta_D_start_stop*self.total_starts
365         self.ramping_life_reduction =
            ↪  self.k_ramp*self.num_rampings
366
367         #Below, the start-stop contribution in reduced
            ↪  life [hours] are added to BP operation.
368         self.BP_time = self.BP_time +
            ↪  self.start_stop_life_reduction +
            ↪  self.ramping_life_reduction
369 # ====================================
370 # Fatigue cycles
371 # ====================================
372
373         self.ML_cycles = self.ML_time * self.f_RSI
374         self.PL_cycles = self.PL_time * self.f_RSI
375         self.vortex_cycles = self.PL_time * self.f_s
376         self.BP_cycles = self.BP_time * self.f_RSI
377         self.FL_cycles = self.FL_time * self.f_RSI
378
379         self.cycles_array = [self.ML_cycles,
            ↪  self.PL_cycles, self.vortex_cycles,
            ↪  self.BP_cycles, self.FL_cycles]
380
381 # ====================================
382 # SN approach
383 # ====================================
384
385         # Cycles below 1e7
```

```
386         self.C1 = 2.82e12
387         self.m1 = 3
388         # For cycles above 1e7 VAL (Sannes)s
389         self.m2 = 5
390         self.C2 = 1.207e16
391
392         self.sigma_A = np.array([ML_stress, PL_stress,
            ↪  self.VRTX_stress, BP_stress, FL_stress])
393         self.sigma_A_ref = BP_stress
394
395 #Cycles complete for all contributions
396         self.N1 = np.divide(self.C1,
            ↪  self.sigma_A**self.m1)
397         self.counter = 0;
398
399         for val in self.N1:
400             if val > 1e7:
401                 self.N1[self.counter] =
                    ↪  np.divide(self.C2,
                    ↪  self.sigma_A[self.counter]**self.m2)
402                 self.counter = self.counter +1
403
404         self.Nref = np.divide(self.C1,
            ↪  self.sigma_A_ref**self.m1)
405
406         if self.Nref > 1e7:
407             self.Nref = np.divide(self.C2,
                ↪  self.sigma_A_ref**self.m2)
408
409
410         self.life_reduced =
            ↪  sum(np.divide(self.cycles_array,self.N1))
411         self.reference_life_reduced =
            ↪  self.BP_cycles/self.Nref
412
413
414         self.projected_lifetime =
            ↪  (1/self.life_reduced)*self.total_time_years
415 #Projected lifetime is a value in years and varies as
    ↪  the normally distributed
416 #input parameters varies.
```

```
417
418
419             return True
420
421 #\%\% End of Monte Carlo iteration
```

A.2 Script for running the FMU

```
import fmpy
from fmpy.fmi2 import FMU2Slave
from fmpy import read_model_description, platform
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

def print_prog_bar(iteration,
                   total,
                   prefix='',
                   suffix='',
                   decimals=1,
                   length=100,
                   fill='',
                   print_end="\r"):
    """
    Call in a loop to create terminal progress bar
    @params:
    iteration   - Required  : current iteration (Int)
    total       - Required  : total iterations (Int)
    prefix      - Optional  : prefix string (Str)
    suffix      - Optional  : suffix string (Str)
    decimals    - Optional  : positive number of
    ↪ decimals (Int)
    length      - Optional  : character length of bar
    ↪ (Int)
    fill        - Optional  : bar fill character (Str)
    print_end    - Optional  : end character (e.g.
    ↪ "\r", "\r\n") (Str)
    """
    percent = (
        "{0:." +
        str(decimals) +
        "f}").format(
            100 * (iteration / float(total)))
    filled_length = int(length * iteration // total)
    p_bar = fill * filled_length + '-' * (length -
    ↪ filled_length)
```

```
35      print(f'\r{prefix} |{p_bar }| {percent}% {suffix}',
        ↪  end=print_end)
36      # Print New Line on Complete
37      if iteration == total:
38          print('\n')
39
40
41  fmu_filename = "Francis_lifetime_estimator.fmu"
42
43  model_description =
    ↪  read_model_description(fmu_filename)
44
45  # collect the value references for the variables to
    ↪  read / write
46  vrs = {}
47  for variable in model_description.modelVariables:
48      vrs[variable.name] = variable.valueReference
49
50  # extract the FMU
51  unzipdir = fmpy.extract(fmu_filename)
52
53  fmu_args = {'guid': model_description.guid,
54              'modelIdentifier':
                ↪  model_description.coSimulation.modelIdentifier,
55              'unzipDirectory': unzipdir}
56
57  #Define all wanted output variables
58  result_vrs = [vrs['BP_cycles'],
59                vrs['PL_cycles'],
60                vrs['FL_cycles'],
61                vrs['ML_cycles'],
62                vrs['life_reduced'],
63                vrs['total_starts'],
64                vrs['total_stops'],
65                vrs['projected_lifetime']]
66
67  fmu = FMU2Slave(**fmu_args)
68
69  fmu.instantiate()
70
71  fmu.reset()
```

```
72
73  fmu.enterInitializationMode()
74  fmu.exitInitializationMode()
75
76  #Array of lifetimes
77  projected_lifetime_array = []
78
79  results = []
80  time = 0
81  step_size = 0.1
82  monte_carlo_number = 5000
83  for mc_cnt in range(monte_carlo_number):
84      print_prog_bar(mc_cnt, monte_carlo_number,
        ↪  length=50)
85      fmu.doStep(currentCommunicationPoint=time,
        ↪  communicationStepSize=step_size)
86      result = fmu.getReal(result_vrs)
87      projected_lifetime_array.append(result[7])
88      results.append(result)
89      time = time + step_size
90
91
92  #
    ↪  ================================================================
93  # Plotter
94  #
    ↪  ================================================================
95
96  pla = np.array(projected_lifetime_array)
97  std = np.std(pla)
98  avg = np.average(pla)
99  pla = pla[pla < avg + 5*std] #Removes lifetime over 5
    ↪  times the STD
100 pla_sort = np.sort(pla)
101 dx = np.zeros((len(pla),2))
102 dx[:,1] = pla
103 dy = np.zeros((len(pla),2))
104 dy[:,1] = 1
105
106
107 dx_rug = np.zeros((len(pla),2))
```

```
dx_rug[:,1] = pla
dx_rug[:,0] = pla
dy_rug = np.zeros((len(pla),2))
dy_rug[:,0] = 0.0
dy_rug[:,1] = 0.05

fig = plt.figure(figsize=(6, 3), dpi=250)

fig = plt.figure(figsize=(6, 3), dpi=250)

for i in range(len(pla)):
    plt.plot(dx[i],dy[i], linewidth = 0.3, color = 'tab:blue', alpha = 0.1)
for i in range(len(pla)):
    plt.plot(dx_rug[i], dy_rug[i], color = 'tab:blue', linewidth = 0.13, alpha = 0.3)
plt.plot([0,max(pla)+5],[1,1], color='r',linewidth = 1.3)
plt.xlabel("Lifetime [Years]")
plt.ylabel("Cumulated damage")
plt.xlim( (0, max(pla) +5) )
plt.ylim((0,1.05))
plt.show()


fig2 = plt.figure(figsize=(6, 3), dpi=250)
plt.xlabel("Lifetime [Years]")
sns.histplot(data=pla, binwidth=3, kde = True,stat ='probability')
plt.plot([avg,avg], [0,0.0992], color = 'r', linewidth = 2, alpha = 0.8)
plt.plot([avg+std,avg+std], [0,0.0414], color = 'r', linewidth = 1, alpha = 0.8)
plt.plot([avg-std,avg-std], [0,0.0831], color = 'r', linewidth = 1, alpha = 0.8)
plt.show()
```

NTNU
Norwegian University of
Science and Technology

Waterpower Laboratory
NTNU