Chair for electrical energy systems
Chairman: Univ.-Prof. Dr.-Ing. Matthias Luther

Student Research Paper

Critical clearing time of synchronous generators

Author: B. Eng. Maximilian Köhler
23176975

Supervisor: M. Sc. Ilya Burlakin

Submission date: December 20, 2023

Author’s declaration

I certify that I have prepared this thesis without outside help and without using sources other than those specified, and that the thesis has not been submitted in the same or a similar form to any other examination authority and has not been accepted by them as part of an examination. All statements that have been copied verbatim or in spirit are marked as such.

Erlangen, December 20, 2023

B. Eng. Maximilian Köhler

Note:

For reasons of readability, the generic masculine is used in this seminar paper. Female and other gender identities are explicitly included where this is necessary for the statement.

Assignment of the paper

Topic: Critical clearing time of synchronous generators

The critical clearing time (CCT) is an essential parameter in power system stability analysis. For example, in the case of synchronous generator (SG), the CCT determines the maximum fault-clearing time a generator can withstand without losing synchronism. This seminar will introduce the concept of CCT computing. We will discuss the factors influencing CCT, such as generator characteristics, system parameters, and fault type, and explore the methods used to calculate CCT in practical power system analysis.

The seminar research paper should contain:

- A literature research of governing equations describing the short-term dynamic behavior of SG, relevant fault types and their influence;
- an investigation of the influences from machine characteristics and system parameters on the CCT;
- a computed simulation model for numerical determination of the CCT with the equal area criterion (EAC);
- simulations of system faults and comparisons to analytical solutions.

Contents

1 Introduction

Dieses Kapitel dient zur Hinführung an das Thema. Hier ist herauszuarbeiten, warum das Thema von Interesse und Bedeutung ist. Des Weiteren ist kurz der Aufbau der Arbeit anzusprechen. Der Umfang des Kapitels sollte eine oder wenige Seiten betragen.

2 State of the art

General sources in terms of standard literature: [1]–[4]

Relevant basics:

- dynamic behavior synchronous generators
- determination of CCT (equal area criteria)
- relevant faults, their modeling and effects
- analytic ways to calculate the CCT
- numerical methods for solving differential equations

3 Methods

4 Model simulation results

5 Discussion and evaluation

6 Summary and outlook

In der Zusammenfassung werden die Ergebnisse der Arbeit kurz zusammengefasst. Der Umfang beträgt ca. eine Seite.

Acronyms

CCT	critical clearing time
EAC	equal area criterion
SG	synchronous generator

Bibliography

[1] D. Oeding and B. R. Oswald, *Elektrische Kraftwerke und Netze*, 8. Auflage. Berlin [Heidelberg]: Springer Vieweg, 2016, 1107 pp., ISBN: 978-3-662-52702-3. DOI: 10.1007/978-3-662-52703-0.

[2] J. D. Glover, T. J. Overbye, and M. S. Sarma, "Power system analysis & design," Boston, MA, 2017.

[3] P. S. Kundur and O. P. Malik, *Power System Stability and Control*, Second edition. New York Chicago San Francisco Athens London Madrid Mexico City Milan New Delhi Singapore Sydney Toronto: McGraw Hill, 2022, 948 pp., ISBN: 978-1-260-47354-4.

[4] J. Machowski, Z. Lubosny, J. W. Bialek, and J. R. Bumby, *Power System Dynamics: Stability and Control*, Third edition. Hoboken, NJ, USA: John Wiley, 2020, 1 p., ISBN: 978-1-119-52636-0 978-1-119-52638-4.

Appendix

A Code

A.1 Model functions

```python
if __name__ == "__main__":
    def showplot():
        from matplotlib import pyplot as plt
        x = [1,5,10,15]
        y = [12,59,100,155]

        plt.plot(x, y)
        plt.show()
```

Listing A.1: Module containing all relevant functions of the SMIB model in Python

A.2 Model of GK

```python
import matplotlib.pyplot as plt
import numpy as np

def mag_and_angle_to_cmplx(mag, angle):
    return mag * np.exp(1j * angle)

# Define the parameters of the system
fn = 60
H_gen = 3.5
X_gen = 0.2
X_ibb = 0.1
X_line = 0.65

# Values are initialized from loadflow
E_fd_gen = 1.075
E_fd_ibb = 1.033
P_m_gen = 1998/2200

omega_gen_init = 0
delta_gen_init = np.deg2rad(45.9)
delta_ibb_init = np.deg2rad(-5.0)

v_bb_gen_init = mag_and_angle_to_cmplx(1.0, np.deg2rad(36.172))

def differential():
    # Calculate the electrical power extracted from the generator at its busbar.
    E_gen_cmplx = mag_and_angle_to_cmplx(E_fd_gen, delta)
    P_e_gen = (v_bb_gen * np.conj((E_gen_cmplx - v_bb_gen) / (1j * X_gen))).real

    # transform the constant mechanical energy into torque
    T_m_gen = P_m_gen / (1 + omega)
```

```python
    # Differential equations of a generator according to Machowski
    domega_dt = 1 / (2 * H_gen) * (T_m_gen - P_e_gen)
    ddelta_dt = omega * 2 * np.pi * fn

    return domega_dt, ddelta_dt


def algebraic():
    # If the SC is on, the admittance matrix is different.
    # The SC on busbar 0 is expressed in the admittance matrix as a very large admittance (1000000) i.e. a very small impedance.
    if sc_on:
        y_adm = np.array([[(-1j / X_gen - 1j / X_line) + 1000000, 1j / X_line],
                          [1j / X_line, -1j / X_line - 1j / X_ibb]])
    else:
        y_adm = np.array([[-1j / X_gen - 1j / X_line, 1j / X_line],
                          [1j / X_line, -1j / X_line - 1j / X_ibb]])

    # Calculate the inverse of the admittance matrix (Y^-1)
    y_inv = np.linalg.inv(y_adm)

    # Calculate current injections of the generator and the infinite busbar
    i_inj_gen = mag_and_angle_to_cmplx(E_fd_gen, delta_gen) / (1j * X_gen)
    i_inj_ibb = mag_and_angle_to_cmplx(E_fd_ibb, delta_ibb_init) / (1j * X_ibb)

    # Calculate voltages at the bus by multiplying the inverse of the admittance matrix with the current injections
    v_bb_gen = y_inv[0, 0] * i_inj_gen + y_inv[0, 1] * i_inj_ibb
    v_bb_ibb = y_inv[1, 0] * i_inj_gen + y_inv[1, 1] * i_inj_ibb

    return v_bb_gen


def do_sim():

    # Initialize the variables
    omega_gen = omega_gen_init
    delta_gen = delta_gen_init
    v_bb_gen = v_bb_gen_init

    # Define time. Here, the time step is 0.005 s and the simulation is 5 s long
    t = np.arange(0, 5, 0.005)
    x_result = []

    for timestep in t:

        # Those lines cause a short circuit at t = 1 s until t = 1.05 s
        if 1 <= timestep < 1.05:
            sc_on = True
        else:
            sc_on = False

        # Calculate the differences to the next step by executing the differential equations at the current step
        domega_dt, ddelta_dt = differential(omega_gen, v_bb_gen, delta_gen, E_fd_gen)
        omega_gen = omega_gen + domega_dt * (t[1] - t[0])
        delta_gen = delta_gen + ddelta_dt * (t[1] - t[0])
```

```
        v_bb_gen = algebraic(delta_gen, sc_on)

        # Save the results, so they can be plotted later
        x_result.append(omega_gen)

    # Convert the results to a numpy array
    res = np.vstack(x_result)
    return t, res
```

Listing A.2: GK's SMIB model