DAFTAR PUSTAKA

Åström, K. J, dan Hägglund, Tore. 1995. *PID Controllers: Theory, Design, and Tuning -2nd Ed.* Amerika Serikat: Instrument Society of America.

Afandy, M., Samman, F. A., and Salam, A. E. U. 2019. *Performance Comparative study on DC-DC Boost Converters Non-Isolated Configurations*. International Conference on Information and Communications Technology (ICOIACT), Yogyakarta, Indonesia, 2019, pp. 728-732, doi: 10.1109/ICOIACT46704.2019.8938481.

Afandy, M., Rahman, F. R., dkk. 2022. *Pemodelan dan Analisis Kendali PI Static dan PI Adaptive DC-DC Boost Converter*. Jambura Journal of Electrical and Electronics Engineering Vol. 4, No. 2.

Akbar, M., Efendi, Z., and Nugraha, S. D. 2022. *SEPIC Converter for Lead Acid Battery Charger Using Fuzzy Logic type-2 Controller*. Journal on Advanced Research in Electrical Engineering, Vol. 6, No. 1.

Ali, M. O., dan Ahmad, A. H. 2020. *Design, Modeling, and Simulation of Controlled SEPIC DC-DC Converter-Based Genetic Algorithm.* International Journal of Power Electronics and Drive System (IJPEDS) Vol. 11, No. 4.

Arkeman, Y., dkk. 2014. *Algoritma Genetika Tujuan Jamak*. Bogor: PT Penerbit IPB Press.

Chen, Guanrong., dan Pham, T. T. 2001. *Introduction to Fuzzy Sets, Fuzzy Logic, and Fuzzy Control System.* New York: CRC Press LLC.

Dwyer, A. O. 2009. *Handbook of PI and PID Controller Tuning Rules -3rd Edition.* London: Imperial College Press.

Gofar, Abd., dan Anshory, Izza. 2016. *Simulasi dan Implementasi Sistem SEPIC (Single Ended Primary Inductor Converter) untuk Mengoptimalkan Keluaran Daya Photovoltaic (Panel Surya) Menggunakan Arduino Uno.* Jurnal Elektro UMSIDA.

Mahmudy, W. F. 2015. *Modul Kuliah: Dasar-Dsar Algoritma Evolusi.* Malang: Universitas Brawijaya.

Ogata, Katsuhiko. 2010. *Modern Control Engineering Fifth Edition.* Amerika Serikat: Pearson Education, Inc.

Prasetiyo, R., dkk. 2016. *Pemodelan dan Simulasi Topologi Single-Ended Primary-Inductor Converter (SEPIC) untuk Mini 3 Plus Wind Turbine.* Prosiding Seminar Nasional Fisika (E-Journal) SNF2016.

Putri, B. R., Sudiharto, I., dkk. 2020. *Design SEPIC Converter for Battery Charging Using Solar Panel*. 2nd International Conference on Science & Technology.

Rajendra, R., and Pratihar, D. K. 2011. *Particle Swarm Optimization Algorithm vs Genetic Algorithm to Develop Integrated Scheme for Obtaining Optimal Mechanical Structure and Adaptive Controller of a Robot*. Intelligent Control and Automation, 2011, 2, 430-449.

Robert, S. 2016. *DC/DC Book of Knowledge Practical tips for the User.* Austria: RECOM Engineering GmbH & Co KG.

Setiawan, Agung., Yanto, Budi., dan Yasdomi, Kiki. 2018. *Logika Fuzzy dengan MATLAB (Contoh Kasus Penelitian Penyakit Bayi dengan Fuzzy Tsukamoto).* Bali: Jayapangus Press.

Setiawan, Iwan. 2008. *Kontrol PID untuk Proses Industri.* Surabaya: PT. Elex Media Komputindo.

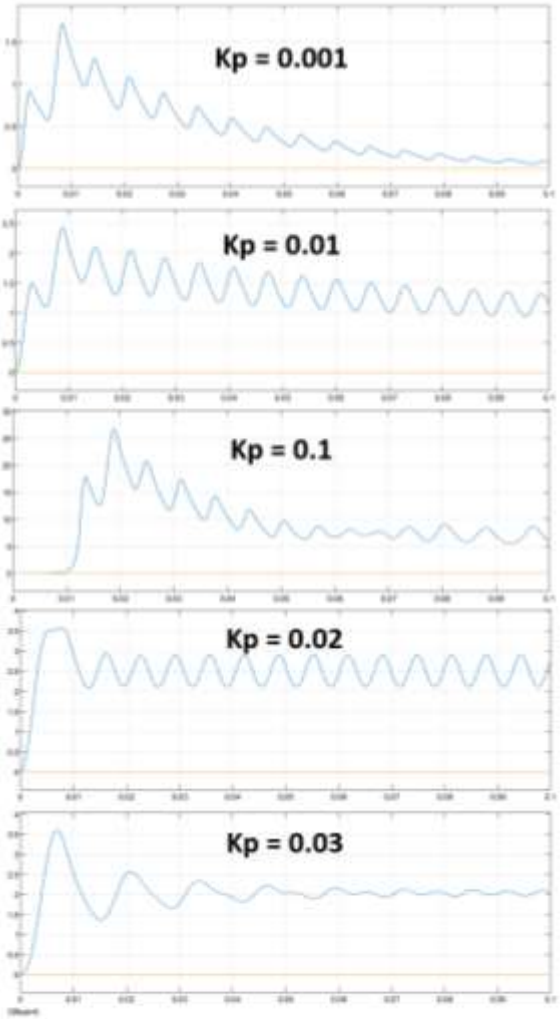
Srun, C., Samman, F. A., and Sadjad, R. S. 2018. *A High Voltage Gain DC-DC Converter Design based on Charge Pump Circuit Configuration with a Voltage Controller.* 2nd International Conference on Applied Electromagnetic Technology (AEMT), Lombok, Indonesia, 2018, pp. 79-84, doi: 10.1109/AEMT.2018.8572403.

Wahyuni, Indah. 2021. *Logika Fuzzy Tahani (Teori dan Implementasi).* Yogyakarta: Komojoyo Press.

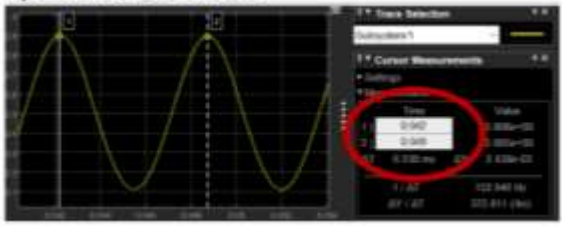
Zukhruf, Febi., and Farazila, R. B. 2021. *Pengantar Optimasi dalam Rekayasa Transportasi*. Bandung: ITB Press.

LAMPIRAN

Lampiran A. Proses Pencarian Nilai Ku dan Tu pada Metode ZN



Kp = 0.02, Tu = 0.007



Lampiran B. Program Algoritma Metode GA

```
clc

% Inisialisasi nilai awal
populationSize = 50;
nGen = 2;
lb = [0 0];
ub = [1 1];
population = lb + (ub - lb) .* rand(populationSize, nGen);
maxGenerations = 8;
bestError = zeros(maxGenerations, 1);
error = zeros(populationSize, 1);

generation = 1;
while generation <= maxGenerations
    % Evaluasi error/performa masing-masing kromosom/individu
    for i = 1:populationSize
        error(i) = calculateError(population(i, :));
    end

    % Seleksi parents menggunakan seleksi turnamen
    parentIndex = selectionTournament(error, 18, populationSize);
    parents = population(parentIndex, :);

    % Operasi crossover titik potong tunggal
    offspring = crossover(parents, parentIndex, nGen);

    % Operasi mutasi
    mutationProbability = 0.1;
    mutatedOffspring = mutation(offspring, mutationProbability);

    % Evaluasi error/performa individu/kromosom hasil reproduksi
    offspringError = zeros(size(offspring, 1), 1);
    for i = 1:size(offspring, 1)
        offspringError(i) = calculateError(mutatedOffspring(i, :));
    end

    % Penggantian populasi
    [population, error] = replace(population, error, parentIndex,
offspring, offspringError, populationSize);

    % Pencatatan nilai error terbaik setiap generasi
    bestError(generation) = min(error);

    % Tampilkan hasil setiap generasi
    disp(['Generasi: ', num2str(generation), ' | Error Terbaik
Sementara: ', num2str(bestError(generation))]);

    generation = generation + 1;
end

% Hasil optimasi
[minError, minIndex] = min(error);
```

```
bestParameters = population(minIndex, :);
disp('Hasil Optimisasi:');
disp(['Parameter Terbaik: ', num2str(bestParameters)]);
disp(['Error Terbaik: ', num2str(minError)]);

% Fungsi untuk menghitung performa sistem kontrol PID pada SEPIC
converter
function error = calculateError(k)
    assignin('base', 'k', k);
    % Kode simulasi SEPIC converter
    sim("PlantforGAtesting.slx");
    % Fungsi Objektif
    error = ISE(length(ISE));
end

% Fungsi-fungsi bantu
function parentIndex = selectionTournament(error, m, populationSize)
    parentIndex = zeros(m, 1);
    for i = 1:m
        candidates = randperm(populationSize, 2);
        if error(candidates(1)) <= error(candidates(2))
            parentIndex(i) = candidates(1);
        else
            parentIndex(i) = candidates(2);
        end
    end
end

function offspring = crossover(parents, parentIndex, nGen)
    n = numel(parentIndex);
    offspringNum = n/2;
    crossoverPoint = randi([1, nGen-1]);
    offspring = zeros(offspringNum, nGen);
    for i = 1:offspringNum
        candidates = randperm(n, 2);
        offspring(i, :) = [parents(candidates(1), 1:crossoverPoint),
parents(candidates(2), crossoverPoint+1:end)];
    end
end

function mutatedOffspring = mutation(offspring, mutationProbability)
    mutatedOffspring = offspring;
    n = numel(mutatedOffspring);
    ranNum = 0.01*rand(n,1);
    for i = 1:n
        if ranNum < mutationProbability
            mutatedOffspring(i) = ranNum(i);
        end
    end
end

function [newPopulation, newError] = replace(population, error,
parentIndex, offspring, offspringError, populationSize)
    newPopulation = population;
```

```
    newError = error;
    m = numel(parentIndex)/2;
    sorting = sort(newError, 'descend');
    nMaxError = sorting(m);
    maxErrorIndices = zeros(m,1);
    for i = 1:populationSize
        if newError(i) >= nMaxError
            maxErrorIndices(i) = i;
        end
    end
    maxErrorIndices = maxErrorIndices(maxErrorIndices ~= 0);
    for a = 1:m
        if newError(maxErrorIndices(a)) >= offspringError(a)
            newPopulation(maxErrorIndices(a), :) = offspring(a, :);
            newError(maxErrorIndices(a)) = offspringError(a);
        end
    end
end
```

Lampiran C. Program Algoritma Metode PSO

```
clc

% Inisialisasi nilai awal
nParticles = 50;
nVariables = 2;
maxIterations = 8;
minValues = [0, 0];
maxValues = [1, 1];

% Inisialisasi posisi dan kecepatan partikel awal secara acak
positions = rand(nParticles, nVariables) .* (maxValues - minValues) +
minValues;
velocities = zeros(nParticles, nVariables) .* (maxValues - minValues) +
minValues;

% Inisialisasi posisi dan nilai pBest (nilai terbaik partikel) awal
pBestPositions = positions;
pBestValues = 100*ones(nParticles, 1);

% Inisialisasi gBest (nilai terbaik global) awal
[~, gBestIndex] = min(objectiveFunction(positions));
gBestPosition = positions(gBestIndex, :);
gBestValue = pBestValues(gBestIndex);

% Iterasi PSO
iteration = 1;
while iteration <= maxIterations

    % Update pBest
    for i = 1:nParticles
        particlePosition = positions(i, :);
        particleValue = objectiveFunction(particlePosition);

        if particleValue < pBestValues(i)
            pBestPositions(i, :) = particlePosition;
            pBestValues(i) = particleValue;
        end
    end

    % Update gBest
    [~, newGBestIndex] = min(pBestValues);
    if pBestValues(newGBestIndex) < gBestValue
        gBestPosition = pBestPositions(newGBestIndex, :);
        gBestValue = pBestValues(newGBestIndex);
    end

    % Update kecepatan dan posisi partikel (the formula based on
Clerc,1999 study)
    phi1 = 2.05;
    phi2 = 2.05;
    phi =  phi1 + phi2;
    w = 2/abs((2-phi-sqrt(phi^2-4*phi)));
```

```
    c1 = w*phi1;
    c2 = w*phi2;

    for i = 1:nParticles
        r1 = rand();
        r2 = rand();

        velocities(i, :) = w * velocities(i, :) + c1 * r1 *
abs(pBestPositions(i, :) ...
            - positions(i, :)) + c2 * r2 * abs(gBestPosition -
positions(i, :));
        positions(i, :) = positions(i, :) + velocities(i, :);
    end

    iteration = iteration + 1;
    disp('Posisi terbaik sementara:');
    disp(gBestPosition);
    disp('Minimum error sementara:');
    disp(gBestValue);
end

% Hasil akhir
disp('Parameter PI terbaik:');
disp(gBestPosition);
disp('Minimum error:');
disp(gBestValue);

% Fungsi untuk menghitung performa sistem kontrol PID pada SEPIC
converter
function error = objectiveFunction(k)
    % Kode simulasi SEPIC
    assignin('base', 'k', k);
    sim("PLANTforPSOtesting.slx");
    % Fungsi Objektif
    error = ISE(length(ISE));
end
```

Lampiran D. *Screenshot* Perancangan Metode *Fuzzy*