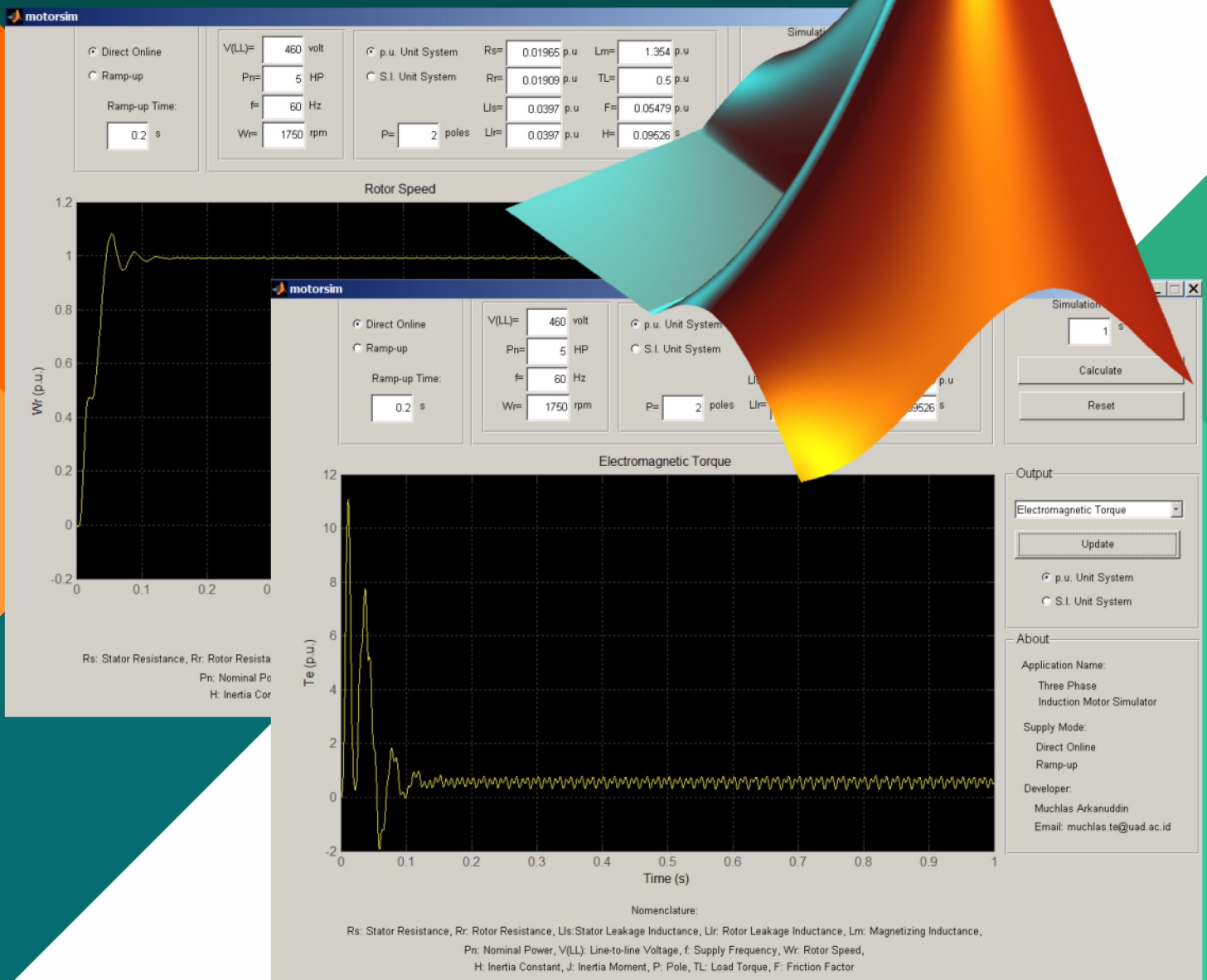


SOURCE CODE MOTORSIM

Pengembang:
Dr. Muchlas, M.T.



SOURCE CODE
MOTORSIM

Pengembang:
Dr. Muchlas, M.T.

Hak cipta program komputer ini milik pengembang. Aplikasi MOTORSIM dan panduan operasinya dapat digunakan secara bebas, tidak dipungut biaya, sejauh untuk tujuan kemajuan ilmu pengetahuan, teknologi dan pendidikan.



Universitas Ahmad Dahlan
Jalan Kapas 9, Semaki ,Yogyakarta



muchlas.te@uad.ac.id

```

% -----
% Program Simulasi Komputer Watak Motor Induksi Tiga Fase
% Pemrogram      : Dr. Muchlas, M.T.
% Tanggal Release : 18 Februari 2021
% Spesifikasi Produk:
% Mode Catu Daya  : Direct Online dan Ramp-up
% Panel Name Plate : Menyediakan input besaran Tegangan Sumber (V),
%                  Daya Nominal (HP), Frekuensi (Hz),
%                  Kecepatan Nominal Rotor (RPM), dan Pole
% Panel Parameter : Menyediakan input besaran Resistansi Stator,
% Motor           Resistansi Rotor, Induktansi Leakage Stator,
%                  Induktansi Leakage Rotor, Induktansi Mutual,
%                  Torsi Beban, Faktor Gesekan, Konstante Inersia
%                  dalam satuan pu dan SI
% Metode Data Entry: Key-in dan Radio Button
% Panel Eksekusi  : Push Button
% Display Output  : Menyediakan pilihan output Tegangan per Fase,
%                  Arus Stator, Torsi Elektromagnetik, Kecepatan Rotor
%                  dalam satuan pu dan SI
% Versi Matlab    : 7.0.4.365 (R14) Service Pack 2
% Solver          : Fungsi ODE45
% Nama berkas    : motorsim.m, motorprob.m, parameters.mat, motorsim.fig
%                  odeprog.m, codeabort.m, acmotor.jpeg, wait.jpeg
% -----

```

```

% -----
% Inisialisasi kode pemrograman, dibangkitkan oleh GUI Developer (GUIDE)
% -----

```

```

function varargout = motorsim(varargin)
    gui_Singleton = 1;
    gui_State = struct('gui_Name',       mfilename, ...
                      'gui_Singleton',  gui_Singleton, ...
                      'gui_OpeningFcn', @motorsim_OpeningFcn, ...
                      'gui_OutputFcn',  @motorsim_OutputFcn, ...
                      'gui_LayoutFcn',  [] , ...
                      'gui_Callback',   []);
    if nargin && ischar(varargin{1})
        gui_State.gui_Callback = str2func(varargin{1});
    end
    if nargout
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
    else
        gui_mainfcn(gui_State, varargin{:});
    end
% --- Akhir inisialisasi kode pemrograman ---

% -----
% Fungsi pembuka dan inisialisasi GUI MOTORSIM
% -----
function motorsim_OpeningFcn(hObject, eventdata, handles, varargin)
    handles.output=hObject;
    guidata(hObject, handles);
    initialize_gui(gcbf, handles, true);
function varargout = motorsim_OutputFcn(hObject, eventdata, handles)
    guidata(hObject,handles)
% --- Akhir fungsi dan inisialisasi GUI MOTORSIM ---

```

```

% -----
% Inisialisasi program MOTORSIM
% -----
function initialize_gui(fig_handle, handles, isreset)
    if isfield(handles, 'metricdata') && ~isreset
        return;
    end
    grid on
    axes(handles.axes1)
% Menampilkan gambar acmotor.jpg ke dalam area plot
% sesuai ukuran aslinya
    matlabImage = imread('\acmotor.jpg');
    image(matlabImage)
    axis image
% Menghapus sumbu pada area plot
    axis off
% Inisialisasi status tombol (button)

                                % Setel:
    set(handles radiobutton1, 'Value', 1) % Mode Direct Online aktif
    set(handles radiobutton2, 'Value', 0) % Mode Ramp-up tidak aktif
    set(handles radiobutton3, 'Value', 1) % INPUT satuan pu aktif
    set(handles radiobutton4, 'Value', 0) % INPUT satuan SI tidak aktif
    set(handles radiobutton5, 'Value', 1) % OUTPUT satuan pu aktif
    set(handles radiobutton6, 'Value', 0) % OUTPUT satuan SI tidak aktif
    set(handles.popupmenu1, 'Value', 1) % OUTPUT Voltage Source ditampilkan
                                % pertama kali pada Menu Popoup
% Inisialisasi dan menampilkan nilai Name Plate awal pada kotak isian data
    set(handles.voltage, 'String', '460')
    set(handles.Nominal_Power, 'String', '5')
    set(handles.frequency, 'String', '60')
    set(handles.Nominal_Speed, 'String', '1750')
% Menampilkan parameter motor dalam satuan pu pada kotak isian data
    set(handles.Rs, 'String', '0.01965')
    set(handles.Rr, 'String', '0.01909')
    set(handles.Lls, 'String', '0.0397')
    set(handles.Llr, 'String', '0.0397')
    set(handles.Lm, 'String', '1.354')
    set(handles.Inertia_Constant, 'String', '0.09526')
    set(handles.friction_factor, 'String', '0.05479')
    set(handles.pole, 'String', '2')
    set(handles.Torque_Load, 'String', '0.5')
% Menampilkan waktu ramp-up dan waktu simulasi
    set(handles.rampup_time, 'String', '0.2')
    set(handles.t_sim, 'String', '1')
% Menyimpan Name Plate

                                % Menyimpan:
    handles.metricdata.voltage=460; % V(LL) dalam volt
    handles.metricdata.Nominal_Power=5; % Pn dalam HP
    handles.metricdata.frequency=60; % Frekuensi dalam Hz
    handles.metricdata.Nominal_Speed=1750; % Kecepatan rotor dalam RPM
    handles.metricdata.pole=2; % Pole

```

```

% Menyimpan parameter motor dalam satuan pu
handles.metricdata.Rs=0.01965;
handles.metricdata.Rr=0.01909;
handles.metricdata.Lls=0.0397;
handles.metricdata.Llr=0.0397;
handles.metricdata.Lm=1.354;
handles.metricdata.Torque_Load=0.5;
handles.metricdata.friction_factor=0.05479;
handles.metricdata.Inertia_Constant=0.09526; % H dalam s atau detik
% Menyimpan waktu ramp-up dan waktu simulasi
handles.metricdata.t_sim=1;
handles.metricdata.rampup_time=0.2;
% Menyimpan variabel pengendali
handles.metricdata.radiobutton1=1; % Tombol INPUT Direct On line aktif
handles.metricdata.radiobutton2=0; % Tombol INPUT Ramp-up non aktif
handles.metricdata.rb3=1; % Tombol INPUT satuan pu aktif
handles.metricdata.rb4=0; % Tombol INPUT satuan SI non aktif
handles.metricdata.rb5=1; % Tombol OUTPUT satuan pu aktif
handles.metricdata.rb6=0; % Tombol OUTPUT satuan SI non aktif
% Komputasi nilai-nilai basis berdasarkan parameter awal motor 5HP
voltage=handles.metricdata.voltage;
Nominal_Power=handles.metricdata.Nominal_Power;
frequency=handles.metricdata.frequency;
Nominal_Speed=handles.metricdata.Nominal_Speed;
V_base=voltage/sqrt(3); % Tegangan basis
P_base=Nominal_Power*746/3; % Daya basis
I_base=P_base/V_base; % Arus basis
Z_base=V_base/I_base; % Impedansi basis
R_base=Z_base; % Resistansi basis
L_base=R_base/(2*pi*frequency); % Induktansi basis
w_base=Nominal_Speed*(2*pi/60); % Kecepatan sudut basis
T_base=746*Nominal_Power/w_base; % Torsi basis
% Menyimpan nilai-nilai basis berdasarkan parameter awal motor 5HP
handles.metricdata.P_base=P_base;
handles.metricdata.V_base=V_base;
handles.metricdata.I_base=I_base;
handles.metricdata.Z_base=Z_base;
handles.metricdata.R_base=R_base;
handles.metricdata.L_base=L_base;
handles.metricdata.w_base=w_base;
handles.metricdata.T_base=T_base;
guidata(handles.figure1, handles);
% --- Akhir inialisasi program MOTORSIM ---

% -----
% Inialisasi Daya Nominal melalui key in
% -----
function Nominal_Power_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

```

```

function Nominal_Power_Callback(hObject, eventdata, handles)
    Nominal_Power = str2double(get(hObject, 'String'));
if isnan(Nominal_Power)
    errordlg('Input must be a number','Error');
    set(handles.Nominal_Power,'String',['5']);
    % Nilai ini 5 dimunculkan lagi sebagai entri sebelumnya
    Nominal_Power = str2double(get(hObject, 'String'));
    handles.metricdata.Nominal_Power = Nominal_Power;
    guidata(hObject,handles);
end
% Simpan daya nominal baru
    handles.metricdata.Nominal_Power = Nominal_Power;
    guidata(hObject,handles)
% --- Akhir inisialisasi daya nominal melalui key in ---

% -----
% Inisialisasi tegangan fase-ke-fase melalui key in
% -----
function voltage_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function voltage_Callback(hObject, eventdata, handles)
    voltage = str2double(get(hObject, 'String'));
if isnan(voltage)
    errordlg('Input must be a number','Error');
    set(handles.voltage,'String',['460']);
    % Nilai ini 460 dimunculkan lagi sebagai entri sebelumnya
    voltage = str2double(get(hObject, 'String'));
    handles.metricdata.voltage = voltage;
    guidata(hObject,handles);
end
% Simpan nilai tegangan baru
handles.metricdata.voltage = voltage;
guidata(hObject,handles)
% --- Akhir inisialisasi tegangan fase-ke-fase melalui key in ---

% -----
% Inisialisasi frekuensi tegangan sumber melalui key in
% -----
function frequency_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

function frequency_Callback(hObject, eventdata, handles)
    frequency = str2double(get(hObject, 'String'));
if isnan(frequency)
    errordlg('Input must be a number','Error');
    set(handles.frequency,'String',['60']);
    % Nilai ini 60 dimunculkan lagi sebagai entri sebelumnya
    frequency = str2double(get(hObject, 'String'));
    handles.metricdata.frequency = frequency;
    guidata(hObject,handles);
end

% Simpan nilai frekuensi yang baru
    handles.metricdata.frequency = frequency;
    guidata(hObject,handles)
% --- Akhir inisialisasi frekuensi tegangan sumber melalui key in ---

% -----
% Inisialisasi kecepatan rotor nominal melalui key in
% -----
function Nominal_Speed_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function Nominal_Speed_Callback(hObject, eventdata, handles)
    Nominal_Speed = str2double(get(hObject, 'String'));
if isnan(Nominal_Speed)
    errordlg('Input must be a number','Error');
    set(handles.Nominal_Speed,'String',['1750']);
    % Nilai ini 1750 dimunculkan lagi sebagai entri sebelumnya
    Nominal_Speed = str2double(get(hObject, 'String'));
    handles.metricdata.Nominal_Speed = Nominal_Speed;
    guidata(hObject,handles);
end
% Simpan nilai kecepatan rotor yang baru
    handles.metricdata.Nominal_Speed = Nominal_Speed;
    guidata(hObject,handles)
% --- Akhir inisialisasi kecepatan rotor nominal melalui key in ---

% -----
% Inisialisasi Pole melalui key in
% -----
function pole_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function pole_Callback(hObject, eventdata, handles)
    pole = str2double(get(hObject, 'String'));
if isnan(pole)
    errordlg('Input must be a number','Error');

```

```

        set(handles.pole,'String',{'2'});
        % Nilai ini 2 dimunculkan lagi sebagai entri sebelumnya
        pole = str2double(get(hObject, 'String'));
        handles.metricdata.pole = pole;
        guidata(hObject,handles);
end
% Simpan nilai pole yang baru
handles.metricdata.pole = pole;
guidata(hObject,handles)
% --- Akhir inisialisasi Pole melalui key in ---

% -----
% Inisialisasi Rs melalui key in
% -----
function Rs_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
    usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function Rs_Callback(hObject, eventdata, handles)
    Rs = str2double(get(hObject, 'String'));
    rb3=handles.metricdata.rb3;
    rb4=handles.metricdata.rb4;
    R_base=handles.metricdata.R_base;
if isnan(Rs)
    errordlg('Input must be a number','Error');
    if rb3==1
        set(handles.Rs,'String',{'0.01965'});
        % Nilai ini 0,01965 dimunculkan lagi sebagai entri sebelumnya
    end
    if rb4==1
        set(handles.Rs,'String',{'1.11473'});
        % Nilai ini 1.11473 dimunculkan lagi sebagai entri sebelumnya
    end
end
Rs = str2double(get(hObject, 'String'));
if rb3==1
% Simpam Rs yang baru dalam pu secara langsung
    handles.metricdata.Rs = Rs;
    set(handles.Rs, 'String',sprintf('%.6f',Rs)); % Setel 5 digit desimal
    guidata(hObject,handles)
end
if rb4==1
% Simpan Rs yang baru dalam pu hasil konversi dari SI
    handles.metricdata.Rs = Rs/R_base;
    set(handles.Rs, 'String',sprintf('%.6f',Rs)); % Setel 5 digit desimal
    guidata(hObject,handles)
end
% --- Akhir inisialisasi Rs melalui key in ---

```



```

% -----
% Inisialisasi Rr melalui key in
% -----
function Rr_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
    usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function Rr_Callback(hObject, eventdata, handles)
    Rr = str2double(get(hObject, 'String'));
    rb3=handles.metricdata.rb3;
    rb4=handles.metricdata.rb4;
    R_base=handles.metricdata.R_base;
if isnan(Rr)
    errordlg('Input must be a number','Error');
    if rb3==1
        set(handles.Rr,'String',['0.01909']);
        % Nilai ini 0.01909 dimunculkan lagi sebagai entri sebelumnya
    end
    if rb4==1
        set(handles.Rr,'String',['1.083']);
        % Nilai ini 1.083 dimunculkan lagi sebagai entri sebelumnya
    end
end
Rr = str2double(get(hObject, 'String'));
if rb3==1
% Simpan Rr yang baru dalam pu secara langsung
    handles.metricdata.Rr = Rr;
    set(handles.Rr, 'String',sprintf('%.6f',Rr)); % Setel 5 digit desimal
    guidata(hObject,handles)
end
if rb4==1
% Simpan Rr yang baru dalam pu hasil konversi dari dari SI
    handles.metricdata.Rr = Rr/R_base;
    set(handles.Rr, 'String',sprintf('%.6f',Rr)); % Setel 6 digit desimal
    guidata(hObject,handles)
end
% --- Akhir inisialisasi Rr melalui key in ---

% -----
% Inisialisasi Lls melalui key in
% -----
function Lls_CreateFcn(hObject, eventdata, handles)

```

```

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function Lls_Callback(hObject, eventdata, handles)
    Lls = str2double(get(hObject, 'String'));
    rb3=handles.metricdata.rb3;
    rb4=handles.metricdata.rb4;
    Z_base=handles.metricdata.Z_base;
    frequency=handles.metricdata.frequency;
if isnan(Lls)
    errordlg('Input must be a number','Error');
    if rb3==1
        set(handles.Lls,'String',['0.0397']);
        % Nilai ini 0.0397 dimunculkan lagi sebagai entri sebelumnya
    end
    if rb4==1
        set(handles.Lls,'String',['0.005974']);
        % Nilai ini 0.005974 dimunculkan lagi sebagai entri sebelumnya
    end
end
Lls = str2double(get(hObject, 'String'));
if rb3==1
% Simpan Lls yang baru dalam pu secara langsung
    handles.metricdata.Lls = Lls;
    set(handles.Lls, 'String',sprintf('%.6f',Lls)); % Setel 6 digit desimal
    guidata(hObject,handles)
end
if rb4==1
% Simpan Lls yang baru dalam pu hasil konversi dari SI
    handles.metricdata.Lls = 2*pi*frequency*Lls/Z_base;
    set(handles.Lls, 'String',sprintf('%.6f',Lls)) ;
        % Setel 6 digit desimal
    guidata(hObject,handles)
end
% --- Akhir inisialisasi Lls melalui key in ---

% -----
% Inisialisasi Llr melalui key in
% -----
function Llr_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function Llr_Callback(hObject, eventdata, handles)
    Llr = str2double(get(hObject, 'String'));
    rb3=handles.metricdata.rb3;
    rb4=handles.metricdata.rb4;
    Z_base=handles.metricdata.Z_base;
    frequency=handles.metricdata.frequency;
if isnan(Llr)

```

```

errordlg('Input must be a number','Error');
if rb3==1
    set(handles.Llr,'String',['0.0397']);
    % Nilai ini 0.0397 dimunculkan lagi sebagai entri sebelumnya
end
if rb4==1
    set(handles.Llr,'String',['0.005974']);
    % Nilai ini 0.005974 dimunculkan lagi sebagai entri sebelumnya
end
end
Llr = str2double(get(hObject, 'String'));
if rb3==1
% Simpan Llr yang baru dalam pu secara langsung
    handles.metricdata.Llr = Llr;
    set(handles.Llr, 'String',sprintf('%.6f',Llr)); % Setel 5 digit desimal
    guidata(hObject,handles)
end
if rb4==1
% Simpan Llr yang baru dalam pu hasil konversi dari SI
    handles.metricdata.Llr = 2*pi*frequency*Llr/Z_base;
    set(handles.Llr, 'String',sprintf('%.6f',Llr));
    % Setel 5 digit desimal
    guidata(hObject,handles)
end
% --- Akhir inisialisasi Llr melalui key in ---

% -----
% Inisialisasi Lm melalui key in
% -----
function Lm_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function Lm_Callback(hObject, eventdata, handles)
    Lm = str2double(get(hObject, 'String'));
    rb3=handles.metricdata.rb3;
    rb4=handles.metricdata.rb4;
    Z_base=handles.metricdata.Z_base;
    frequency=handles.metricdata.frequency;
if isnan(Lm)
    errordlg('Input must be a number','Error');
    if rb3==1
        set(handles.Lm,'String',['1.354']);
        % Nilai ini 1.354 dimunculkan lagi sebagai entri sebelumnya
    end
    if rb4==1
        set(handles.Lm,'String',['0.2037']);
        % Nilai ini 0.2037 dimunculkan lagi sebagai entri sebelumnya
    end
end
end
Lm = str2double(get(hObject, 'String'));
if rb3==1
% Simpan Lm yang baru dalam pu secara langsung
    handles.metricdata.Lm = Lm;
    set(handles.Lm, 'String',sprintf('%.6f',Lm)); % Setel 5 digit desimal
    guidata(hObject,handles)

```

```

end
if rb4==1
% Simpan Lm yang baru dalam pu hasil konversi dari SI
handles.metricdata.Lm = 2*pi*frequency*Lm/Z_base;
set(handles.Lm, 'String', sprintf('%.6f',Lm));
% Setel 5 digit desimal
guidata(hObject,handles)
end
% --- Akhir inisialisasi Lm melalui key in ---

% -----
% Inisialisasi Konstanta Inersia melalui key in
% -----
function Inertia_Constant_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
set(hObject,'BackgroundColor','white');
end
function Inertia_Constant_Callback(hObject, eventdata, handles)
Inertia_Constant = str2double(get(hObject, 'String'));
rb3=handles.metricdata.rb3;
rb4=handles.metricdata.rb4;
Nominal_Power=handles.metricdata.Nominal_Power;
w_base=handles.metricdata.w_base;
if isnan(Inertia_Constant)
error('Input must be a number','Error');
if rb3==1
set(handles.Inertia_Constant,'String',['0.09526']);
% Nilai ini 0.09526 dimunculkan lagi sebagai entri sebelumnya
end
if rb4==1
set(handles.Inertia_Constant,'String',['0.02']);
% Nilai ini 0.02 dimunculkan lagi sebagai entri sebelumnya
end
end
Inertia_Constant = str2double(get(hObject, 'String'));
if rb3==1
% Simpan konstanta inersia yang baru dalam pu secara langsung
handles.metricdata.Inertia_Constant = Inertia_Constant;
set(handles.Inertia_Constant, 'String', sprintf('%.6f',Inertia_Constant));
% Setel 5 digit desimal
guidata(hObject,handles)
end
if rb4==1
% Simpan konstanta inersia yang baru dalam pu hasil konversi dari SI
handles.metricdata.Inertia_Constant =
Inertia_Constant*(0.5*w_base^2)/(Nominal_Power*746);
set(handles.Inertia_Constant, 'String', sprintf('%.6f',Inertia_Constant));
% Setel 5 digit desimal
guidata(hObject,handles)
end
% --- Akhir inisialisasi Konstanta Inersia melalui key in ---

% -----
% Inisialisasi Faktor Gesekan melalui key in
% -----

```

```

function friction_factor_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function friction_factor_Callback(hObject, eventdata, handles)
    friction_factor = str2double(get(hObject, 'String'));
    rb3=handles.metricdata.rb3;
    rb4=handles.metricdata.rb4;
    T_base=handles.metricdata.T_base;
    w_base=handles.metricdata.w_base;

if isnan(friction_factor)
    errordlg('Input must be a number','Error');
    if rb3==1
        set(handles.friction_factor,'String',['0.05479']);
        % Nilai ini 0.05479 dimunculkan lagi sebagai entri sebelumnya
    end
    if rb4==1
        set(handles.friction_factor,'String',['0.005752']);
        % Nilai ini 0.005752 dimunculkan lagi sebagai entri sebelumnya
    end
end
friction_factor = str2double(get(hObject, 'String'));
if rb3==1
% Simpan faktor gesekan yang baru dalam pu secara langsung
    handles.metricdata.friction_factor = friction_factor;
    set(handles.friction_factor, 'String',sprintf('%.6f',friction_factor));
    % Setel 5 digit desimal
    guidata(hObject,handles)
end
if rb4==1
% Simpan faktor gesekan yang baru dalam pu hasil konversi dari SI
    handles.metricdata.friction_factor = friction_factor*w_base/T_base;
    set(handles.friction_factor, 'String',sprintf('%.6f',friction_factor));
    % Setel 5 digit desimal
    guidata(hObject,handles)
end
% --- Akhir inisialisasi Faktor Gesekan melalui key in ---

% -----
% Inisialisasi Torsi Beban melalui key in
% -----
function Torque_Load_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function Torque_Load_Callback(hObject, eventdata, handles)
    Torque_Load = str2double(get(hObject, 'String'));
    rb3=handles.metricdata.rb3;
    rb4=handles.metricdata.rb4;
    T_base=handles.metricdata.T_base;
if isnan(Torque_Load)
    errordlg('Input must be a number','Error');
    if rb3==1
        set(handles.Torque_Load,'String',['0.5']);

```

```

        % Nilai ini 0.5 dimunculkan lagi sebagai entri sebelumnya
    end
    if rb4==1
        set(handles.Torque_Load,'String',['10.1768']);
        % Nilai ini 10.1768 dimunculkan lagi sebagai entri sebelumnya
    end
end

Torque_Load = str2double(get(hObject, 'String'));
if rb3==1
% Simpan nilai torsi beban yang baru dalam pu secara langsung
    handles.metricdata.Torque_Load = Torque_Load;
    guidata(hObject,handles)
end
if rb4==1
% Simpan nilai tori beban yang baru dalam pu hasil konversi dari SI
    handles.metricdata.Torque_Load = Torque_Load/T_base;
    guidata(hObject,handles)
end
% --- Akhir inisialisasi Torsi Beban melalui key in ---

% -----
% Inisialisasi Waktu Simulasi melalui key in
% -----
function t_sim_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function t_sim_Callback(hObject, eventdata, handles)
    t_sim = str2double(get(hObject, 'String'));
if isnan(t_sim)
    errordlg('Input must be a number','Error');
    set(handles.t_sim,'String',['1']);
    % Nilai ini 1 dimunculkan lagi sebagai entri sebelumnya
    t_sim = str2double(get(hObject, 'String'));
    handles.metricdata.t_sim = t_sim;
    guidata(hObject,handles);
end
% Simpan waktu simulasi yang baru
handles.metricdata.t_sim = t_sim;
guidata(hObject,handles)
% --- Akhir inisialisasi Waktu Simulasi melalui key in ---

% -----
% Inisialisasi waktu Ramp-up melalui key in
% -----
function rampup_time_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

function rampup_time_Callback(hObject, eventdata, handles)
    rampup_time = str2double(get(hObject, 'String'));
if isnan(rampup_time)
    error('Input must be a number','Error');
    set(handles.rampup_time,'String',['0.2']);
    % Nilai ini 0.2 dimunculkan lagi sebagai entri sebelumnya
    rampup_time = str2double(get(hObject, 'String'));
    handles.metricdata.rampup_time = rampup_time;
    guidata(hObject,handles);
end
% Simpan waktu simulasi yang baru
    handles.metricdata.rampup_time = rampup_time;
    guidata(hObject,handles)
% --- Akhir inisialisasi waktu Ramp-up melalui key in ---

% -----
% Inisialisasi pilihan OUTPUT melalui key in
% -----
function popupmenu1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
    set(hObject, 'String', {'Source Voltage', 'Stator Current',
'Electromagnetic Torque', 'Rotor Speed'});
end
% --- Akhir Inisialisasi pilihan OUTPUT melalui key in ---
% --- Akhir dari metode inisialisasi key in ---

% -----
% Eksekusi program ketika RADIOBUTTON1 (tombol Direct Online) ditekan
% -----
function radiobutton1_Callback(hObject, eventdata, handles)
    set(handles.radiobutton1,'Value',1); % Setel tombol Direct Online aktif
    set(handles.radiobutton2,'Value',0); % Setel tombol Ramp-up non aktif
    dol_mode=1;
    rampup_mode=0;
    handles.metricdata.radiobutton1=dol_mode;
    handles.metricdata.radiobutton2 = rampup_mode;
    guidata(hObject,handles);
% --- Akhir fungsi RADIOBUTTON1 (tombol Direct Online) ---

% -----
% Eksekusi program ketika RADIOBUTTON2 (tombol Rampup-up)ditekan
% -----
function radiobutton2_Callback(hObject, eventdata, handles)
    set(handles.radiobutton1,'Value',0); % Setel tombol Direct Online
    % non aktif
    set(handles.radiobutton2,'Value',1); % Setel tombol Ramp-up aktif
    dol_mode=0;
    rampup_mode=1;
    handles.metricdata.radiobutton1=dol_mode;
    handles.metricdata.radiobutton2 = rampup_mode;
    guidata(hObject,handles);
% --- Akhir fungsi RADIOBUTTON2 (tombol Rampup-up) ---

% -----
% Eksekusi program ketika RADIOBUTTON3 (tombol INPUT satuan pu) ditekan

```

```

% -----
function radiobutton3_Callback(hObject, eventdata, handles)
    set(handles.radiobutton3,'Value',1); % Setel tombol INPUT satuan pu
                                     % aktif
    set(handles.radiobutton4,'Value',0); % Setel tombol INPUT satuan SI
                                     % non aktif

% Menampilkan satuan pu di belakang nilai parameter motor
    set(handles.text19, 'String', 'p.u. ');
    set(handles.text20, 'String', 'p.u. ');
    set(handles.text21, 'String', 'p.u. ');
    set(handles.text22, 'String', 'p.u. ');
    set(handles.text23, 'String', 'p.u. ');
    set(handles.text24, 'String', 'p.u. ');
    set(handles.text25, 'String', 'p.u. ');
    set(handles.text26, 'String', 's');
    set(handles.text18, 'String', 'H=');

% Menampilkan nilai parameter dalam pu pada kotak key in
Rs=handles.metricdata.Rs;
set(handles.Rs, 'String',sprintf('%.6f',Rs));
Rr=handles.metricdata.Rr;
set(handles.Rr, 'String',sprintf('%.6f',Rr));
Lls=handles.metricdata.Lls;
set(handles.Lls, 'String',sprintf('%.6f',Lls));
Llr=handles.metricdata.Llr;
set(handles.Llr, 'String',sprintf('%.6f',Llr));
Lm=handles.metricdata.Lm;
set(handles.Lm, 'String',sprintf('%.6f',Lm));
Torque_Load=handles.metricdata.Torque_Load;
set(handles.Torque_Load, 'String',Torque_Load);
friction_factor=handles.metricdata.friction_factor;
set(handles.friction_factor, 'String',sprintf('%.6f',friction_factor));
Inertia_Constant=handles.metricdata.Inertia_Constant;
set(handles.Inertia_Constant, 'String',sprintf('%.6f',Inertia_Constant));

% Setel tombol pu aktif dan tombol SI nonaktif
rb3=1;
rb4=0;
handles.metricdata.rb3 = rb3;
handles.metricdata.rb4 = rb4;
guidata(hObject,handles);

% ---Akhir fungsi RADIOBUTTON3 (tombol INPUT satuan pu)---

% -----
% Eksekusi program ketika RADIOBUTTON4 (tombol INPUT satuan SI) ditekan
% -----
function radiobutton4_Callback(hObject, eventdata, handles)
    set(handles.radiobutton3,'Value',0); % Setel tombol INPUT satuan pu
                                     % non aktif
    set(handles.radiobutton4,'Value',1); % Setel tombol INPUT satuan SI

```



```

% aktif
% Menampilkan satuan SI di belakang nilai parameter motor
set(handles.text19, 'String', 'ohm');
set(handles.text20, 'String', 'ohm');
set(handles.text21, 'String', 'H');
set(handles.text22, 'String', 'H');
set(handles.text23, 'String', 'H');
set(handles.text24, 'String', 'N.m');
set(handles.text25, 'String', 'N.m.s');
set(handles.text26, 'String', 'kg.m^2');
set(handles.text18, 'String', 'J=');

% Menghitung nilai-nilai basis
Nominal_Power=handles.metricdata.Nominal_Power;
voltage=handles.metricdata.voltage;
frequency=handles.metricdata.frequency;
Nominal_Speed=handles.metricdata.Nominal_Speed;
P_base=Nominal_Power*746/3;
V_base=voltage/sqrt(3);
I_base=P_base/V_base;
Z_base=V_base/I_base;
R_base=Z_base;
L_base=R_base/(2*pi*frequency);
w_base=Nominal_Speed*(2*pi/60);
T_base=746*Nominal_Power/w_base;
% Menyimpan nilai-nilai basis
handles.metricdata.P_base=P_base;
handles.metricdata.V_base=V_base;
handles.metricdata.I_base=I_base;
handles.metricdata.Z_base=Z_base;
handles.metricdata.R_base=R_base;
handles.metricdata.L_base=L_base;
handles.metricdata.w_base=w_base;
handles.metricdata.T_base=T_base;
% Menghitung dan menampilkan nilai parameter motor dalam SI
% pada kotak key in
Rs=handles.metricdata.Rs;
set(handles.Rs, 'String', sprintf('%.6f',R_base*Rs));
Rr=handles.metricdata.Rr;
set(handles.Rr, 'String', sprintf('%.6f',R_base*Rr));
Lls=handles.metricdata.Lls;
set(handles.Lls, 'String', sprintf('%.6f',R_base*Lls/(2*pi*frequency)));
Llr=handles.metricdata.Llr;
set(handles.Llr, 'String', sprintf('%.6f',R_base*Llr/(2*pi*frequency)));
Lm=handles.metricdata.Lm;
set(handles.Lm, 'String', sprintf('%.6f',R_base*Lm/(2*pi*frequency)));
Torque_Load=handles.metricdata.Torque_Load;
set(handles.Torque_Load, 'String', Torque_Load*T_base);
friction_factor=handles.metricdata.friction_factor;
set(handles.friction_factor, 'String', sprintf('%.6f',
friction_factor*T_base/w_base));
Inertia_Constant=handles.metricdata.Inertia_Constant;
J=Inertia_Constant*Nominal_Power*746/(0.5*w_base^2);
set(handles.Inertia_Constant, 'String', sprintf('%.6f', J));
handles.metricdata.rb3 = 0;
handles.metricdata.rb4 = 1;
guidata(hObject,handles);

```

```

% --- Akhir fungsi RADIOBUTTON4 (tombol INPUT satuan SI) ---

% -----
% Eksekusi program ketika RADIOBUTTON5 (tombol OUTPUT satuan pu) ditekan
% -----
function radiobutton5_Callback(hObject, eventdata, handles)
    set(handles.radiobutton5,'Value',1); % Setel tombol OUTPUT satuan pu
    % aktif
    set(handles.radiobutton6,'Value',0); % Setel tombol OUTPUT satuan SI
    % non aktif

    rb5=1;
    rb6=0;
    handles.metricdata.rb5 = rb5;
    handles.metricdata.rb6 = rb6;
    guidata(hObject,handles);
% --- Akhir fungsi RADIOBUTTON5 (tombol OUTPUT satuan pu)

% -----
% Eksekusi program ketika RADIOBUTTON6 (tombol OUTPUT satuan SI) ditekan
% -----
function radiobutton6_Callback(hObject, eventdata, handles)
    set(handles.radiobutton5,'Value',0); % Setel tombol OUTPUT satuan pu
    % non aktif
    set(handles.radiobutton6,'Value',1); % Setel tombol OUTPUT satuan SI
    % aktif

    rb5=0;
    rb6=1;
    handles.metricdata.rb5 = rb5;
    handles.metricdata.rb6 = rb6;
    guidata(hObject,handles);
% --- Akhir fungsi RADIOBUTTON6 (tombol OUTPUT satuan SI)

% -----
% Eksekusi program ketika PUSHBUTTON1 (tombol CALCULATE) ditekan
% -----
function pushbutton1_Callback(hObject, eventdata, handles)
    cla
% Membaca parameter motor yang dimasukkan melalui key in
    Rs = handles.metricdata.Rs;
    Rr = handles.metricdata.Rr;
    Lls = handles.metricdata.Lls;
    Llr = handles.metricdata.Llr;
    Lm = handles.metricdata.Lm;
%     Lls=Lm+Lls;
%     Llr=Llr+Lm;
    frequency = handles.metricdata.frequency;
    Inertia_Constant = handles.metricdata.Inertia_Constant;
    Torque_Load = handles.metricdata.Torque_Load;
    friction_factor = handles.metricdata.friction_factor;
    pole = handles.metricdata.pole;
    N=pole/2;
    w=2*pi*frequency;
    t_sim=handles.metricdata.t_sim;
    rampup_time=handles.metricdata.rampup_time;
    rampup_mode=handles.metricdata.radiobutton2;
    dol_mode=handles.metricdata.radiobutton1;
% Menyimpan parameter motor ke berkas parameters.mat

```

```

    save('parameters',
    'Rs','Rr','Lls','Llr','Lm','frequency','Inertia_Constant','Torque_Load',
    'friction_factor','dol_mode','rampup_time');
% Menyelesaikan persamaan motor yang ada pada fungsi motorprob.m
    tspan=[0:1/9999: t_sim];    % batas waktu integrasi
    IC=[0;0;0;0;0;0];    % Inisialisasi kondisi awal variabel-variabel
    % dalam persamaan diferensial motor
    options=odeset('OutputFcn',@odeprog,'Events',@odeabort);
    % Options untuk menjalankan program tampilan
    % progres simulasi pada berkas odeprog.m dan
    % odeabort.m

    axes(handles.axes1)
    matlabImage = imread('\wait.jpg');% Menampilkan gambar motor dan
    image(matlabImage)    % logo MOTORSIM dari berkas wait.jpg
    axis off    % ke dalam kotak tampilan grafik
    axis image    % OUTPUT

% Proses komputasi menggunakan ODE45
    [T Y] = ode45(@ (t,y) motorprob(t,y), tspan,IC,options);
% Membangkitkan tegangan fase referensi a-b-c untuk mode catu daya ramp-up
    rampup_time=handles.metricdata.rampup_time;
% Dalam satuan pu, Xm=Lm;
    Xm=Lm;
    for i=1:length(Y);
        va(i)=sin(w*T(i));
        vb(i)=sin(w*T(i)-2*pi/3);
        vc(i)=sin(w*T(i)+2*pi/3);
        if dol_mode==0
            mag(i)=(1/rampup_time)*T(i);
            if mag(i)<=1
                va(i)=(1/rampup_time)*T(i)*va(i);
                vb(i)=(1/rampup_time)*T(i)*vb(i);
                vc(i)=(1/rampup_time)*T(i)*vc(i);
            end
            va(i)=va(i);
            vb(i)=vb(i);
            vc(i)=vc(i);
        end
    end
% Membangkitkan data arus stator, torsi elektromagnetik, dan
% kecepatan rotor berdasarkan hasil komputasi menggunakan ODE45
    istator(i)=Y(i,2);
    speed(i)=Y(i,5);
    torque(i)=Xm*(Y(i,1)*Y(i,4)-Y(i,2)*Y(i,3));
end;
    save('datamotorsim','T','Y','istator','speed','torque','t_sim')
% Menyimpan variabel hasil komputasi untuk keperluan plot data
    setappdata(0,'myTime',T)
    setappdata(0,'myTorque',torque);
    setappdata(0,'mySpeed',speed);
    setappdata(0,'myIstator',istator);
    setappdata(0,'myva',va);
    setappdata(0,'myvb',vb);
    setappdata(0,'myvc',vc);
% Membersihkan area grafik
    cla
    cla 'reset'

```

```

    axes(handles.axes1);
% Membaca data waktu, tegangan sumber, arus stator, torsi
% elektromagnetik, dan kecepatan rotor dalam pu
    myTime = getappdata(0,'myTime');
    myva = getappdata(0,'myva');
    myvb = getappdata(0,'myvb');
    myvc = getappdata(0,'myvc');
    myIstator = getappdata(0,'myIstator');
    myTorque = getappdata(0,'myTorque');
    mySpeed = getappdata(0,'mySpeed');
% Membaca status radiobutton5 (tombol OUTPUT satuan pu)
    rb5 = handles.metricdata.rb5;
% Membaca waktu ramp_up tegangan sumber dan waktu simulasi
% dan kecepatan nominal rotor
    rampup_time=handles.metricdata.rampup_time;
    t_sim=handles.metricdata.t_sim;
    Nominal_Speed=handles.metricdata.Nominal_Speed;

% Eksekusi untuk tombol OUTPUT satuan pu aktif
if (rb5==1)
    popup_sel_index = get(handles.popupmenu1, 'Value');
switch popup_sel_index
    case 1
        if dol_mode==1 % Plot tegangan sumber dalam pu
            plot(myTime,myva,'y',myTime,myvb,'w',myTime,myvc,'g');
            axis([0 t_sim/10 -1.5 1.5]);
        end
        if dol_mode==0
            plot(myTime,myva,'y');
            axis([0 t_sim -1.5 1.5]);
        end
        grid on
        set(gca,'color',[0 0 0])
        set(gca,'xcolor',[0.4 0.4 0.4])
        set(gca,'ycolor',[0.4 0.4 0.4])
        title('Voltage per Phase')
        xlabel('Time (s)','color','k')
        ylabel('Va in p.u.','color','k')

    case 2
        plot(myTime,myIstator,'k'); % Plot arus stator dalam pu
        grid on
        set(gca,'color',[1 1 1])
        set(gca,'xcolor',[0 0 0])
        set(gca,'ycolor',[0 0 0])
        title('Stator Current')
        xlabel('Time (s)','color','k')
        ylabel('is (p.u.)','color','k')

    case 3
        plot(myTime, myTorque,'k'); % Plot torsi elektromagnetik
        grid on % dalam pu
        set(gca,'color',[1 1 1])
        set(gca,'xcolor',[0 0 0])
        set(gca,'ycolor',[0 0 0])
        title('Electromagnetic Torque ')
        xlabel('Time (s)','color','k')
        ylabel('Te (p.u.)','color','k')
end
end

```

```

case 4
    plot(myTime, mySpeed, 'k');          % Plot kecepatan rotor dalam pu
    grid on
    set(gca, 'color', [1 1 1])
    set(gca, 'xcolor', [0 0 0])
    set(gca, 'ycolor', [0 0 0])
    title('Rotor Speed ')
    xlabel('Time (s)', 'color', 'k')
    ylabel('Wr (p.u.)', 'color', 'k')
end
% Eksekusi untuk tombol OUTPUT satuan SI aktif
else
    nominal_voltage=handles.metricdata.voltage;
    Nominal_Power=handles.metricdata.Nominal_Power;
    Nominal_Speed=handles.metricdata.Nominal_Speed;
    vm_si=nominal_voltage/sqrt(3)*sqrt(2);
    popup_sel_index = get(handles.popupmenu1, 'Value');

switch popup_sel_index
    case 1
        % Plot tegangan sumber dalam SI
        delta_y=0.1*nominal_voltage;
        y_up=max(nominal_voltage)+delta_y;
        y_down=-max(nominal_voltage)-delta_y;
        if dol_mode==1
            plot(myTime, vm_si*myva, 'y', myTime, vm_si*myvb, 'w', myTime, vm_si*myvc, 'g');
            axis([0 t_sim/10 y_down y_up]);
        end
        if dol_mode==0
            plot(myTime, vm_si*myva, 'y');
            axis([0 t_sim y_down y_up]);
        end
        grid on;
        set(gca, 'color', [0 0 0])
        set(gca, 'xcolor', [0.3 0.3 0.3])
        set(gca, 'ycolor', [0.3 0.3 0.3])
        title('Voltage per Phase')
        xlabel('Time (s)', 'color', 'k')
        ylabel('Vabc in volt', 'color', 'k')
    case 2
        % Plot arus stator dalam SI
        is_b=myIstator*(Nominal_Power*746)/nominal_voltage;
        plot(myTime, is_b/1.2, 'y');
        grid on;
        set(gca, 'color', [0 0 0])
        set(gca, 'xcolor', [0.3 0.3 0.3])
        set(gca, 'ycolor', [0.3 0.3 0.3])
        title('Stator Current')
        xlabel('Time (s)', 'color', 'k')
        ylabel('is (A)', 'color', 'k')

    case 3
        % Plot torsi elektromagnetik
        speed_b=Nominal_Speed*2*pi/60; % dalam SI
        torque_b=Nominal_Power*746/speed_b;
        plot(myTime, myTorque*torque_b, 'y');

```

```

        grid on;
        set(gca,'color',[0 0 0])
        set(gca,'xcolor',[0.3 0.3 0.3])
        set(gca,'ycolor',[0.3 0.3 0.3])
        title('Electromagnetic Torque ')
        xlabel('Time (s)','color','k')
        ylabel('Te (N.m.)','color','k')
    case 4
        % Plot kecepatan rotor dalam SI
        plot(myTime, mySpeed*Nominal_Speed,'y');
        grid on;
        set(gca,'color',[0 0 0])
        set(gca,'xcolor',[0.3 0.3 0.3])
        set(gca,'ycolor',[0.3 0.3 0.3])
        title('Rotor Speed ')
        xlabel('Time (s)','color','k')
        ylabel('Wr (r.p.m.)','color','k')
    end
end
% --- Akhir fungsi PUSHBUTTON1 (tombol CALCULATE) ---

% -----
% Eksekusi program ketika mengubah pilihan OUTPUT pada kotak POPUPMENU1
% -----
function popupmenu1_Callback(hObject, eventdata, handles)
% ---Akhir fungsi POPUPMENU1 (pilihan OUTPUT)---
% -----
% Eksekusi program ketika PUSHBUTTON2 (tombol UPDATE) ditekan
% -----
function pushbutton2_Callback(hObject, eventdata, handles)
    cla
    cla 'reset'
    axes(handles.axes1);
% Membaca data waktu, tegangan sumber, arus stator, torsi
% elektromagnetik, dan kecepatan rotor dalam pu
    myTime = getappdata(0,'myTime');
    myva = getappdata(0,'myva');
    myvb = getappdata(0,'myvb');
    myvc = getappdata(0,'myvc');
    myIstator = getappdata(0,'myIstator');
    myTorque = getappdata(0,'myTorque');
    mySpeed = getappdata(0,'mySpeed');
% Membaca status radiobutton5 (tombol OUTPUT satuan pu)
    rb5 = handles.metricdata.rb5;
% Membaca waktu ramp_up tegangan sumber dan waktu simulasi
% dan kecepatan nominal rotor
    rampup_time=handles.metricdata.rampup_time;
    t_sim=handles.metricdata.t_sim;
    Nominal_Speed=handles.metricdata.Nominal_Speed;
    dol_mode=handles.metricdata.radiobutton1;
% Plot data untuk tombol OUTPUT satuan pu aktif
if (rb5==1)
    popup_sel_index = get(handles.popupmenu1, 'Value');
switch popup_sel_index
    case 1
        if dol_mode==1
            % Plot tegangan sumber dalam pu
            plot(myTime,myva,'y',myTime,myvb,'w',myTime,myvc,'g');

```

```

        axis([0 t_sim/10 -1.5 1.5]);
    end
    if dol_mode==0
        plot(myTime,myva,'y');
        axis([0 t_sim -1.5 1.5]);
    end
    grid on
    set(gca,'color',[0 0 0])
    set(gca,'xcolor',[0.3 0.3 0.3])
    set(gca,'ycolor',[0.3 0.3 0.3])
    title('Voltage per Phase')
    xlabel('Time (s)','color','k')
    ylabel('Va in p.u.','color','k')
case 2
    plot(myTime,myIstator,'y'); % Plot arus stator dalam pu
    grid on
    set(gca,'color',[0 0 0])
    set(gca,'xcolor',[0.3 0.3 0.3])
    set(gca,'ycolor',[0.3 0.3 0.3])
    title('Stator Current')
    xlabel('Time (s)','color','k')
    ylabel('is (p.u.)','color','k')
case 3
    plot(myTime, myTorque,'y'); % Plot torsi elektromagnetik
    grid on % dalam pu
    set(gca,'color',[0 0 0])
    set(gca,'xcolor',[0.3 0.3 0.3])
    set(gca,'ycolor',[0.3 0.3 0.3])
    title('Electromagnetic Torque ')
    xlabel('Time (s)','color','k')
    ylabel('Te (p.u.)','color','k')
case 4
    plot(myTime, mySpeed,'y'); % Plot kecepatan rotor dalam pu
    grid on
    set(gca,'color',[0 0 0])
    set(gca,'xcolor',[0.3 0.3 0.3])
    set(gca,'ycolor',[0.3 0.3 0.3])
    title('Rotor Speed ')
    xlabel('Time (s)','color','k')
    ylabel('Wr (p.u.)','color','k')
end
% Plot data untuk tombol OUTPUT satuan SI aktif
else
    nominal_voltage=handles.metricdata.voltage;
    Nominal_Power=handles.metricdata.Nominal_Power;
    Nominal_Speed=handles.metricdata.Nominal_Speed;
    vm_si=nominal_voltage/sqrt(3)*sqrt(2);
    popup_sel_index = get(handles.popupmenu1, 'Value');
switch popup_sel_index
    case 1 % Plot tegangan sumber dalam SI
        delta_y=0.1*nominal_voltage;
        y_up=max(nominal_voltage)+delta_y;
        y_down=-max(nominal_voltage)-delta_y;
        if dol_mode==1
plot(myTime,vm_si*myva,'y',myTime,vm_si*myvb,'w',myTime,vm_si*myvc,'g');
            axis([0 t_sim/10 y_down y_up]);
        end
end

```

```

    if dol_mode==0
        plot(myTime,vm_si*myva,'y');
        axis([0 t_sim y_down y_up]);
    end
    grid on;
    set(gca,'color',[0 0 0])
    set(gca,'xcolor',[0.3 0.3 0.3])
    set(gca,'ycolor',[0.3 0.3 0.3])
    title('Voltage per Phase')
    xlabel('Time (s)','color','k')
    ylabel('Vabc in volt','color','k')
case 2                                     % Plot arus stator dalam SI
    is_b=myIstator*(Nominal_Power*746)/nominal_voltage;
    plot(myTime,is_b/1.2,'y');
    grid on;
    set(gca,'color',[0 0 0])
    set(gca,'xcolor',[0.3 0.3 0.3])
    set(gca,'ycolor',[0.3 0.3 0.3])
    title('Stator Current')
    xlabel('Time (s)','color','k')
    ylabel('is (A)','color','k')

case 3                                     % Plot torsi elektromagnetik
    speed_b=Nominal_Speed*2*pi/60; % dalam SI
    torque_b=Nominal_Power*746/speed_b;
    plot(myTime, myTorque*torque_b,'y');
    grid on;
    set(gca,'color',[0 0 0])
    set(gca,'xcolor',[0.3 0.3 0.3])
    set(gca,'ycolor',[0.3 0.3 0.3])
    title('Electromagnetic Torque ')
    xlabel('Time (s)','color','k')
    ylabel('Te (N.m.)','color','k')
case 4                                     % Plot kecepatan rotor dalam SI
    plot(myTime, mySpeed*Nominal_Speed,'y');
    grid on;
    set(gca,'color',[0 0 0])
    set(gca,'xcolor',[0.3 0.3 0.3])
    set(gca,'ycolor',[0.3 0.3 0.3])
    title('Rotor Speed ')
    xlabel('Time (s)','color','k')
    ylabel('Wr (r.p.m.)','color','k')
end
end
% --- Akhir fungsi PUSHBUTTON2 (tombol UPDATE) ---

% -----
% Eksekusi program ketika tombol RESET ditekan
% -----
function reset_Callback(hObject, eventdata, handles)
    initialize_gui(gcbf, handles, true);
% Menampilkan gambar acmotor.jpg ke dalam area plot sesuai ukuran aslinya
    matlabImage = imread('\acmotor.jpg');
    image(matlabImage)
    axis image
% Menghapus sumbu pada area plot

```



```
axis off
% Menghapus semua data grafik (data plot)
setappdata(0,'myva',0);
setappdata(0,'myvb',0);
setappdata(0,'myvc',0);
setappdata(0,'myTime',0)
setappdata(0,'myTorque',0);
setappdata(0,'mySpeed',0);
setappdata(0,'myIstator',0);
% --- Akhir fungsi tombol RESET ---

% ---Akhir program---
```