```
function transientSimulationScript(clearingCycles, saveTransientRunValues,
saveTransientRunPlots, modelType, nPV, PG, QG, yGenFaultOn, E_primeVals, EdgesGen,
theta, yGenPostFault, ode_omegaType3, lPV, Pm, PGVals, ode_thetaEquilibrium,
numStepsFaultOn, numStepsPostFault, x0_Type3, omega, h, folder_processedData,
systemName1)
    if clearingCycles ~= 3
        saveTransientRunValues = false;
        saveTransientRunPlots = false;
    end

    transientRunStart = tic;

    if strcmp(modelType, 'Type3')
        % error('Make sure to save results first! Only if clearing time is 3
cycles.')
        PG_FaultOn_Type3 = generateSymbolicPowerFlowEquations(nPV, PG, QG,
yGenFaultOn, E_primeVals, EdgesGen, theta);
        PG_PostFault_Type3 = generateSymbolicPowerFlowEquations(nPV, PG, QG,
yGenPostFault, E_primeVals, EdgesGen, theta);

        ode_omegaType3_TransientFaultOn = subs(ode_omegaType3(lPV), [PG, Pm],
[PG_FaultOn_Type3, PGVals]);
        ode_Type3_TransientFaultOn = [ode_thetaEquilibrium;
ode_omegaType3_TransientFaultOn];
        ode_Type3_TransientFaultOn = subs(ode_Type3_TransientFaultOn, theta(1), 0);
        display(ode_Type3_TransientFaultOn);

        ode_omegaType3_TransientPostFault = subs(ode_omegaType3(lPV), [PG, Pm],
[PG_PostFault_Type3, PGVals]);
        ode_Type3_TransientPostFault = [ode_thetaEquilibrium;
ode_omegaType3_TransientPostFault];
        ode_Type3_TransientPostFault = subs(ode_Type3_TransientPostFault, theta(1),
0);
        display(ode_Type3_TransientPostFault);

        thetaTransientVals = zeros(nPV, numStepsFaultOn + numStepsPostFault);
        omegaTransientVals = zeros(nPV, numStepsFaultOn + numStepsPostFault);
        xTransientVals = [thetaTransientVals; omegaTransientVals];

        xTransientVals(:, 1) = [x0_Type3(lPV); x0_Type3(nPV+lPV)];
        x_Type3 = [theta(lPV); omega(lPV)];

        fxTransientVals = zeros(size(xTransientVals));

        for t = 2:numStepsFaultOn
            fxTransientVals(:, t-1) = double( subs(rhs(ode_Type3_TransientFaultOn),
x_Type3, xTransientVals(:, t-1) ) ) ;
            xTransientVals(:, t) = xTransientVals(:, t-1) + h*fxTransientVals(:,
t-1);
        end
```

```
        for t = numStepsFaultOn+1:numStepsFaultOn+numStepsPostFault
            fxTransientVals(:, t-1) =
double( subs(rhs(ode_Type3_TransientPostFault), x_Type3, xTransientVals(:, t-1) ) );
            xTransientVals(:, t) = xTransientVals(:, t-1) + h*fxTransientVals(:,
t-1);
        end

    elseif strcmp(modelType, 'Type2')
        error('Check Transient Stability for Type 2 again.\n')
        PG_FaultOn_Type2 = generateSymbolicPowerFlowEquations(nPV, PG, QG,
yGenFaultOn, E_prime_Type2, EdgesGen, theta);
        PG_PostFault_Type2 = generateSymbolicPowerFlowEquations(nPV, PG, QG,
yGenPostFault, E_prime_Type2, EdgesGen, theta);

        ode_omegaType2_TransientFaultOn = subs(ode_omegaType2(lPV), [PG, Pm],
[PG_FaultOn_Type2, PGVals]);

        ode_Type2_TransientFaultOn = [ode_thetaEquilibrium;
ode_omegaType2_TransientFaultOn; ode_EqprimeType2_SS; ode_EdprimeType2_SS;
ode_VRType2_SS; ode_PmType2_SS];
        ode_Type2_TransientFaultOn = subs(ode_Type2_TransientFaultOn, [theta(1),
Eq_prime(1), Ed_prime(1)],  [0, 1, 0]);
        display(ode_Type2_TransientFaultOn);

        ode_omegaType2_TransientPostFault = subs(ode_omegaType2(lPV), [PG, Pm],
[PG_PostFault_Type2, PGVals]);
        ode_Type2_TransientPostFault = [ode_thetaEquilibrium;
ode_omegaType2_TransientPostFault; ode_EqprimeType2_SS; ode_EdprimeType2_SS;
ode_VRType2_SS; ode_PmType2_SS];

        ode_Type2_TransientPostFault = subs(ode_Type2_TransientPostFault,
[theta(1), Eq_prime(1), Ed_prime(1)],  [0, 1, 0]);
        display(ode_Type2_TransientPostFault);

        xTransientVals = zeros(nPV*6, numStepsFaultOn + numStepsPostFault);

        xTransientVals(:, 1) = x0_Type2;

        fxTransientVals = zeros(size(xTransientVals));

        for t = 2:numStepsFaultOn
            fxTransientVals(:, t-1) = double( subs(rhs(ode_Type2_TransientFaultOn),
x_Type2, xTransientVals(:, t-1) ) ) ;
            xTransientVals(:, t) = xTransientVals(:, t-1) + h*fxTransientVals(:,
t-1);
        end

        for t = numStepsFaultOn+1:numStepsFaultOn+numStepsPostFault
```

```
            fxTransientVals(:, t-1) =
double( subs(rhs(ode_Type2_TransientPostFault), x_Type2, xTransientVals(:, t-1) ) );
            xTransientVals(:, t) = xTransientVals(:, t-1) + h*fxTransientVals(:,
t-1);
        end

    else
        error('Type 1 NOT modeled for TS!');
    end

    if saveTransientRunValues
        fileType = '.csv';
        filenameTransientRun = strcat(folder_processedData, systemName1, "/
xTransientRun_", modelType, fileType);
        save(filenameTransientRun, 'xTransientVals');
    end

    thetaValsTransient = xTransientVals(1:nPV, 1:t);

    figTheta = figure('Name', 'Transient Simmulation: Machine Rotor Angles');
    % figure('Name', 'Transient Simulation: Machine Rotor Angles');
    plot(1:t, thetaValsTransient);
    xlabel('Time [ms]');
    ylabel('$\theta$ [rad]');
    legend({'Gen02', 'Gen03', 'Gen04'});
    title('Transient Simulation: Machine Rotor Angles')

    omegaValsTransient = xTransientVals(nPV+1:2*nPV, 1:t);
    figOmega = figure('Name', 'Transient Simmulation: Machine Rotor Speeds');
    % figure('Name', 'Transient Simmulation: Machine Rotor Speeds');
    plot(1:t, omegaValsTransient);
    xlabel('Time [ms]');
    ylabel('$\omega$ [pu]');
    legend({'Gen02', 'Gen03', 'Gen04'});
    title('Transient Simulation: Machine Rotor Speeds')

    if saveTransientRunPlots
        filenameFigTheta = strcat(folder_processedData, systemName1, "/
transientRun_theta_", modelType, '.png');
        save(filenameFigTheta, 'figTheta');
        filenameFigOmega = strcat(folder_processedData, systemName1, "/
transientRun_omega_", modelType, '.png');
        save(filenameFigOmega, 'figOmega');
    end

    transientRunStop = toc(transientRunStart);
    display(transientRunStop);
end
```