

APPENDIX B

NUMERICAL ALGORITHM

```
// Start of script 'pss_3D_script.m'

model = mphload('/home/tuna/project/pss_3D_matlab.mph');

% Initiating tolerances and collecting data from model
tol_energy = 1*10^(-2);
tol_effectiveness = 1*10^(-4);
energy_balance = 1;
delta_effectiveness = 1;
m_s = mpheval(model,{'m_supply'});
m_e = mpheval(model,{'m_exhaust'});
period = mpheval(model,{'period'});
eta = 0;

% Computing the flow geometries for the supply and exhaust flows
model.sol('sol1').runAll;
model.sol('sol2').runAll;

k = 0;

% Initiating the control loop
while(abs(energy_balance) > tol_energy && delta_effectiveness > tol_effectiveness)
%for (k = 1:10)

    k = k+1;

    % Solving the for supply sector of the wheel
    if 2-mod(k,2)==1
        % Computing the transient solution
        model.sol('sol3').runAll;

        % Evaluating the enthalpies
        H_si = mpheval(model,{'H_si'});
        H_so = mpheval(model,{'H_ei'});

        % Setting the flow geometry used to the solution of the exhaust sector of the
wheel
        model.physics('ht').feature('fluid1').set('minput_velocity_src', 1,
'root.mod1.u_e');
        model.physics('chcs').feature('cdm1').set('u_src', 1, 'root.mod1.u_e');

        % Reversal of flow boundaries and changing initial temperature
        model.physics('ht').feature('temp1').selection.set([]);
        model.physics('ht').feature('ofl1').selection.set([13]);
        model.physics('ht').feature('temp1').selection.set([18]);
        model.physics('ht').feature('temp1').set('T0', 1, 'T_exhaust');

        % Setting the heat fluxes to use the local Nusselt number function for the
exhaust flow geoetry
        model.physics('ht2').feature('hf1').set('q0', 1,
'alpha_e*f_e*Nu_H1_z(z_star_e)*(km/D_hydraulic)*(genext1(T_b)-T_wall)');
        model.physics('ht2').feature('hf2').set('q0', 1,
'alpha_e*f_lp_e*Nu_H1_z(z_star_e)*(km/D_hydraulic)*(genext1(T_b)-T_wall)');
```

```

        model.physics('ht2').feature('hf3').set('q0', 1,
'alpha_e*f_uc_e*Nu_H1_z(z_star_e)*(km/D_hydraulic)*(genext1(T_b)-T_wall)');

        % Reversal of flow boundaries and changing initial mass fraction
        model.physics('chcs').feature('out1').selection.set([]);
        model.physics('chcs').feature('in1').selection.set([]);
        model.physics('chcs').feature('out1').selection.set([13]);
        model.physics('chcs').feature('in1').selection.set([18]);
        model.physics('chcs').feature('in1').set('c0', 2,
'ht_f.fluid1.fc(phi_exhaust,T_exhaust,1[atm])');

        % Setting the longitudinal velocity component to the exhaust flow geometry
        model.physics('eode2').feature('dodel').set('f', 1, 'T_b-
genproj2(genproj1(w_e*T))/genproj2(genproj1(w_e))');

        % Change study settings
        model.study('std4').feature('time').set('useinitsol', 'on');
        model.study('std4').feature('time').set('initmethod', 'sol');
        model.study('std4').feature('time').set('initstudy', 'std3');

    end

    % Solving the for exhaust sector of the wheel
    if 2-mod(k,2)==2
        model.sol('sol4').runAll;

        H_ei = mpheval(model,{'H_ei'});
        H_eo = mpheval(model,{'H_eo'});

        model.physics('ht').feature('fluid1').set('minput_velocity_src', 1,
'root.mod1.u_s');
        model.physics('chds').feature('cdm1').set('u_src', 1, 'root.mod1.u_s');

        model.physics('ht').feature('temp1').selection.set([]);
        model.physics('ht').feature('ofl1').selection.set([18]);
        model.physics('ht').feature('temp1').selection.set([13]);
        model.physics('ht').feature('temp1').set('T0', 1, 'T_supply');

        model.physics('ht2').feature('hf1').set('q0', 1,
'alpha_s*f_s*Nu_H1_z(z_star_s)*(km/D_hydraulic)*(genext1(T_b)-T_wall)');
        model.physics('ht2').feature('hf2').set('q0', 1,
'alpha_s*f_lp_s*Nu_H1_z(z_star_s)*(km/D_hydraulic)*(genext1(T_b)-T_wall)');
        model.physics('ht2').feature('hf3').set('q0', 1,
'alpha_s*f_uc_s*Nu_H1_z(z_star_s)*(km/D_hydraulic)*(genext1(T_b)-T_wall)');

        model.physics('chcs').feature('in1').selection.set([]);
        model.physics('chcs').feature('out1').selection.set([]);
        model.physics('chcs').feature('in1').selection.set([13]);
        model.physics('chcs').feature('out1').selection.set([18]);
        model.physics('chcs').feature('in1').set('c0', 2,
'ht_f.fluid1.fc(phi_supply,T_supply,1[atm])');

        model.physics('eode2').feature('dodel').set('f', 1, 'T_b-
genproj2(genproj1(w_s*T))/genproj2(genproj1(w_s))');

        model.study('std3').feature('time').set('useinitsol', 'on');
        model.study('std3').feature('time').set('initmethod', 'sol');
        model.study('std3').feature('time').set('initstudy', 'std4');

        % Evaluating the current energy balance

```

```

energy_balance = (m_s*(H_si-H_so)-m_e*(H_eo-H_ei))/(min(m_s,m_e)*(H_si-H_ei));

% Evaluating the effectiveness for periodicity
eta_0 = eta;
eta = (m_s*(H_si-H_so)+m_e*(H_eo-H_ei))/(2*min(m_s,m_e)*(H_si-H_ei));
delta_effectiveness = abs(eta-eta_0)/(2*period);

end

model.physics('eode2').feature('init1').set('T_b', 1, 'T_b');
model.physics('ht2').feature('init1').set('T_wall', 1, 'T_wall');
model.physics('ht').feature('init1').set('T', 1, 'T');
model.physics('chcs').feature('init1').set('vw', 1, 'vw');

%time = mphglobal(model,'t','solnum','end');
%model.param.set('t0',time);

%disp(sprintf('End of iteration No.%d',k));

end

model.save('pss_3D_matlab.mph');

// End of script 'pss_3D_script.m'

```