```
function [k_out, h_out] = CalculaTrajetoriaEquilibrio(initial_state, final_state, T,↙
param_sys, param_gov)

% beta = param_sys(1);
% delta= param_sys(2);
% gamma = param_sys(3);
% alpha = param_sys(4);
% theta = param_sys(5);
% eta = param_sys(6);

% tau_c= param_gov(1);
% tau_h = param_gov(2);
% tau_k = param_gov(3);


k_0 = initial_state(1);
h_0 = initial_state(2);

k_ss = final_state(1);
h_ss = final_state(2);

% Definicao de um vetor x_0 de chute inicial com T linhas e 2 colunas, onde
% x_0(T,1) = Capital
% x_0(T,2) = Horas trabalhadas
% o vetor é iniciado com NaN (Not a number)
x_0 = nan(T+1,2);

% Definicao da trajetoria inicial sendo k(1) = k_0. Sendo que o vetor sera
% otimizado com excessao da posicao k(1). (Para h) Dou um chute inicial
% para o vetor h, sendo que o vetor sera otimizado.
for i=0:T
    x_0(i+1,1)=k_0*(1-i/T)+i/T*k_ss;
    x_0(i+1,2)=h_0*(1-i/T)+i/T*h_ss;
end

% Definido parametros de condicoes iniciais e finais
param_conditions=[T, k_0, k_ss, h_0, h_ss];

% Definindo opcoes de otimizacao
options=optimoptions('fsolve','Display','iter-detailed', 'Algorithm','levenberg-↙
marquardt');
sol=fsolve(@(z)FirstOrderCondition(z,param_sys,param_gov,param_conditions),x_0,options);

% monto a trajetoria com os valores iniciais e finais fornecidos
k_out = [k_0;sol(2:end,1);k_ss];
h_out =[sol(:,2);h_ss];

end % end of function
```