Table of Contents

SCO-2 energy conversion example

```
clear
clc
close 'all'
```

Add current directory to the Python Path

```
EasyProp_path = ' '; %<- Path if EasyProp.py is in your current directory
if count(py.sys.path,EasyProp_path) == 0  % <-- see if desired directory is on
 path
    insert(py.sys.path,int32(0),EasyProp_path); %<-- if not; add it.
end
```

Initialize Fluid Property object

```
fluid = 'CO2';
units = 'SI';
gas = py.EasyProp.EasyProp(fluid,units);

To_C = 25; % C
To = To_C + 273; % K
Po = 101.3; % kPa
ho = gas.h_pT(Po,To_C);
so = gas.s_pT(Po,To_C);


% function for specific flow exergy neglecting kinetic and potential energy
ef_fun = @(h_val,s_val) h_val - ho - To*(s_val - so);
```

Initialize State Point Data Arrays

```
numSP = 8;
h = nan(numSP,1);
h_s = nan(numSP,1);
s = nan(numSP,1);
s_s = nan(numSP,1);
T = nan(numSP,1);
P = nan(numSP,1);
x = nan(numSP,1);
```

```
ef = nan(numSP,1);

eta_turb = 1;
eta_c = 1;
eta_rc = eta_c;

xi_ltr = 0.90;
xi_htr = 0.90;
% xi_ltr = 0.92;% book assumptions for regen effectiveness.
% xi_htr = 0.98;

Pmax = 20000; % kPa
Pmin = 7700; % kPa
```

state point property calculations

```
P(1) = Pmin; % kPa
T(1) = 32; % C
h(1) = gas.h_pT(P(1),T(1));
s(1) = gas.s_pT(P(1),T(1));
ef(1) = ef_fun(h(1),s(1));

% 1 -> 2 compression
P(2) = Pmax; % kPa
s_s(2) = s(1);
h_s(2) = gas.h_ps(P(2),s_s(2));
%eta_c = 1.0;
h(2) = h(1) - (h(1) - h_s(2))/eta_c;
T(2) = gas.T_ph(P(2),h(2));
s(2) = gas.s_ph(P(2),h(2));
ef(2) = ef_fun(h(2),s(2));

% come back to sp 3
P(3) = P(2);
%eta_rc = 1.0;

% come back to sp 4
P(4) = P(3);

% state point 5
P(5) = P(4);
T(5) = 550; %C
h(5) = gas.h_pT(P(5),T(5));
s(5) = gas.s_pT(P(5),T(5));
ef(5) = ef_fun(h(5),s(5));


% expand in turbine for sp 6
P(6) = P(1); % kPa
s_s(6) = s(5);
h_s(6) = gas.h_ps(P(6),s_s(6));
%eta_turb = 1.0;
h(6) = h(5) - eta_turb*(h(5) - h_s(6));
```

```
s(6) = gas.s_ph(P(6),h(6));
T(6) = gas.T_ph(P(6),h(6));
ef(6) = ef_fun(h(6),s(6));

% state point 7 (what I know for now...)
P(7) = P(6);

% state point 8 ( what I know for now...)
P(8) = P(7);


% set up system of equations:
% F = [f1, h(3), h(4), h(7), h(8)]

% #1 - Low Temperature Regenerator Effectiveness
LTR_eff = @(F) xi_ltr*(F(4)- gas.h_pT(P(8),T(2))) - (F(4) - F(5));

% #2 - High Temperature Regenerator Effectiveness
HTR_eff = @(F) xi_htr*(h(6)-gas.h_pT(P(7),gas.T_ph(P(3),F(2)))) - ...
    (h(6) - F(4));

% #3 - Low Temperature Regenerator Energy Balance
LTR_ebal = @(F) F(1)*(F(2)-h(2)) - (F(4)-F(5));

% #4 - High Temperature Regenerator Energy Balance
HTR_ebal = @(F) (F(3) - F(2)) - (h(6)-F(4));

% #5 - State Point 3 "identity" <- any ideas for a better name?
SP3_id = @(F) F(5) - (F(5)-gas.h_ps(P(3),gas.s_ph(P(8),F(5))))./eta_rc - ...
    F(2);

% overall balance equation.  Scalar output to vector input. Minimum value
% is zero.
balance = @(F) abs(LTR_eff(F))+abs(HTR_eff(F))+abs(LTR_ebal(F))+...
    abs(HTR_ebal(F))+abs(SP3_id(F));

option = 1;
% 1 = no inequality constraints and option arguements
% 2 = additional option arguments and inequality constraints

Aeq = []; % no linear equalities to impose
Beq = [];
lb = [0 h(2) h(2) h(1) h(2)]; % lower bound
ub = [1 h(6) h(5) h(6) h(6)]; % upper bound
initial_guess = [0.5 h(2) h(5) h(6)-1 h(2)];

switch option
    case 1
        A = []; % no linear inequality constraints
        b = [];
        [F,fval,exitflag] = fmincon(balance,initial_guess,A,b,Aeq,Beq,...
            lb,ub);
    case 2
        A = [0 1 -1 0 0;
```

```
            0 0 0 -1 1]; %< h(3) - h(4) <= 0, h(8)-h(7) <= 0
        b = [0;
            0];
        options = optimoptions('fmincon','FunctionTolerance',1e-6,...
            'ConstraintTolerance',1e-9,'StepTolerance',1e-15,...
            'Display','iter','Algorithm','sqp',...
            'MaxFunctionEvaluations',1500);
        [F,fval,exitflag] = fmincon(balance,initial_guess,A,b,Aeq,Beq,...
            lb,ub,[],options);
    otherwise
        error('Invalid selection for option!');
end

fprintf('fval = %g \n',fval);
fprintf('exit flag = %d \n',exitflag);

% unpack the results
ff = F(1);
h(3) = F(2);
h(4) = F(3);
h(7) = F(4);
h(8) = F(5);


% fill in missing details on state point data
T(3) = gas.T_ph(P(3),h(3));
s(3) = gas.s_pT(P(3),T(3));
ef(3) = ef_fun(h(3),s(3));

T(4) = gas.T_ph(P(4),h(4));
s(4) = gas.s_ph(P(4),h(4));
ef(4) = ef_fun(h(4),s(4));

T(7) = gas.T_ph(P(7),h(7));
s(7) = gas.s_ph(P(7),h(7));
ef(7) = ef_fun(h(7),s(7));

T(8) = gas.T_ph(P(8),h(8));
s(8) = gas.s_ph(P(8),h(8));
ef(8) = ef_fun(h(8),s(8));

fprintf('Flow fraction f = %g \n',ff);

w_comp = ff*(h(1) - h(2));
w_rc = (1-ff)*(h(8) - h(3));
w_turb = h(5) - h(6);
w_net = w_comp + w_rc + w_turb;

q_s = h(5) - h(4);
q_r = ff*(h(1) - h(8));
q_net = q_s+q_r;

fprintf('Net specific work = %g kJ/kg \n',w_net);
fprintf('Net specific heat = %g kj/kg \n',q_net);
```

```
eta_th = w_net/q_s;
fprintf('Thermal efficiency = %5.2f percent \n',eta_th*100);

bwr = abs(w_comp + w_rc)/w_turb;
fprintf('Back work ratio = %g percent \n',bwr*100);

ef_in = ef(5) - ef(4);
ef_out = ff*(ef(8) - ef(1));

Ex_d_ltr = ff*ef(2) + ef(7) - ef(8) - ff*ef(3);
Ex_d_htr = ef(3) + ef(6) - ef(4) - ef(7);



fprintf('Specific flow exergy in: %g kJ/kg \n',ef_in);
fprintf('Specific flow exergy out: %g kJ/kg \n',ef_out);
fprintf('Specific flow exergy destroyed LTR: %g kJ/kg \n',Ex_d_ltr);
fprintf('Specific flow exergy destroyed HTR: %g kJ/kg \n',Ex_d_htr);

Flow_ex_balance = (ef_in - ef_out) - (w_net + Ex_d_ltr + Ex_d_htr);
fprintf('Specific flow exergy balance: %g kJ/kg \n',Flow_ex_balance);
fprintf('\n\n\n\n')
fprintf('\n\nState point data: \n\n');

% display state point data neatly
SP = {'1','2','3','4','5','6','7','8'};
SP_table = table(P,T,h,s,ef,'RowName',SP);
disp(SP_table);
```

Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the value of the step size tolerance and constraints are
satisfied to within the value of the constraint tolerance.

fval = 9.79516e-05
exit flag = 2
Flow fraction f = 0.687217
Net specific work = 110.282 kJ/kg
Net specific heat = 110.282 kj/kg
Thermal efficiency = 48.31 percent
Back work ratio = 20.5446 percent
Specific flow exergy in: 134.653 kJ/kg
Specific flow exergy out: 5.9878 kJ/kg
Specific flow exergy destroyed LTR: 5.29151 kJ/kg
Specific flow exergy destroyed HTR: 13.0917 kJ/kg
Specific flow exergy balance: -5.24514e-07 kJ/kg

State point data:

	P	T	h	s	ef
1	7700	32	306.23	1.3463	214.78
2	20000	59.999	324.5	1.3463	233.04
3	20000	153.56	528.98	1.8933	274.53
4	20000	364.35	806.85	2.4269	393.39
5	20000	550	1035.1	2.7411	528.04
6	7700	424.55	896.34	2.7411	389.24
7	7700	180.37	618.47	2.2514	257.29
8	7700	69.126	477.94	1.8933	223.49

Published with MATLAB® R2022a