Table of Contents

```
clear;
```

Dish Collector

A dish collector is built under given ambient parameters

```
amb = Ambient;

st1(3) = Stream;

st1(1).fluid = char(Const.Fluid(1));
st1(1).T = Temperature(C2K(800));
st1(1).p = 5e5;

ap = AirPipe;
il = InsLayer;

st_dc_i = Stream;
st_dc_i.fluid = char(Const.Fluid(1));
st_dc_i.T = Temperature(C2K(350));
st_dc_i.p = 5e5;

st_dc_o = st_dc_i.flow();
st_dc_o.T = Temperature(C2K(800));
st_dc_o.p = st_dc_i.p;

dc = DishCollector;
dc.amb = amb;
dc.st_i = st_dc_i;
dc.st_o = st_dc_o;
dc.airPipe = ap;
dc.insLayer = il;

guess1 = [1500; 400; 0.1] ;
options = optimset('Display','iter');
[x1, fval1] = fsolve(@(x1)CalcDishCollector(x1, dc), ...
    guess1, options);
dc
```

```
                                         Norm of      First-order   Trust-region
```

```
 Iteration  Func-count     f(x)          step         optimality    radius
     0          4     4.92364e+08                      5.74e+09               1
     1          8     3.47306e+08              1      5.11e+08               1
     2         12     3.38708e+08            2.5      9.53e+06             2.5
     3         16     3.20205e+08           6.25      1.12e+08            6.25
     4         20     2.75108e+08         15.625       2.7e+08            15.6
     5         24     1.72295e+08        39.0625      5.22e+08            39.1
     6         28     1.26527e+07        97.6562      2.89e+08            97.7
     7         32         5234.88        38.1153      1.54e+07             244
     8         36      0.00188801       0.785437      6.96e+03             244
     9         40     1.70391e-16    0.000486987       0.00233             244
    10         44     5.29476e-22    1.61689e-10      5.81e-06             244

Equation solved, fsolve stalled.

fsolve stopped because the relative size of the current step is less than the
default value of the step size tolerance squared and the vector of function values
is near zero as measured by the default value of the function tolerance.




dc = 

  DishCollector with properties:

             A: 87.7000
         gamma: 0.9700
           rho: 0.9100
       shading: 0.9500
          d_ap: 0.1840
         d_cav: 0.4600
       dep_cav: 0.2300
         theta: 0.7854
           amb: [1x1 Ambient]
           T_p: [1x1 Temperature]
         T_ins: [1x1 Temperature]
          st_i: [1x1 Stream]
          st_o: [1x1 Stream]
       airPipe: [1x1 AirPipe]
      insLayer: [1x1 InsLayer]
         q_use: 4.4368e+04
         q_tot: 61390
           eta: 0.7227
```

Stirling Engine Array

Two kinds of connection orders of the Stirling engines are considered.

```
order = 'Reverse'; % can be 'Same', 'Reverse' and other types, n_1! types all toge
n1 = 10;                % Column number of Stirling engine array
n2 = Const.NUM_SE / n1; % Row number of Stirling engine array
```

```
guess2 = zeros(2,n1);   % 2 * n1 unknown parameters (outlet temperature of two flu

q_m_1 = 2.990;  % To be calculated!
q_m_2 = 5.625;  % To be calculated;

% st1_se_i = dc.st_o;            % Not right for the q_m, so next line corrects th
st1_se_i = Stream;              % to be changed!!!!!
st1_se_i.fluid = char(Const.Fluid(2));
st1_se_i.T.v = dc.st_o.T.v;
st1_se_i.p = dc.st_o.p;
st1_se_i.q_m.v = q_m_1 / n2;
se_cp_1 = CoolProp.PropsSI('C', 'T', st1_se_i.T.v, 'P', ...
    st1_se_i.p, st1_se_i.fluid);

st2_se_i = Stream;
st2_se_i.fluid = char(Const.Fluid(2));
st2_se_i.T = Temperature(327.2);
st2_se_i.p = 1e6;
st2_se_i.q_m.v = q_m_2 / n2;
se_cp_2 = CoolProp.PropsSI('C', 'T', st2_se_i.T.v, 'P', ...
    st2_se_i.p, st2_se_i.fluid);

se(1,n1) = StirlingEngine;

se(1) = StirlingEngine;
se(1).flowType = order; % can be changed
se(1).st1_i = st1_se_i;
se(1).st1_o = se(1).st1_i.flow();
se(1).st1_o.p = se(1).st1_i.p;
se(1).cp_1 = se_cp_1;

if (strcmp(order, 'Same'))
    %%%%% Same order %%%%%
    se(1).st2_i = st2_se_i;
    se(1).st2_o = Stream.flow(se(1).st2_i);
    se(1).st2_o.p = se(1).st2_i.p;
    se(1).cp_2 = se_cp_2;
    for i = 2:n1
        se(i) = StirlingEngine;
        se(i).flowType = se(1).flowType;    % Flowtype of any Stirling engine can
        se(i).cp_1 = se_cp_1;
        se(i).cp_2 = se_cp_2;
        se(i).st1_i = se(i-1).st1_o;
        se(i).st2_i = se(i-1).st2_o;
        se(i).st1_o = Stream.flow(se(i).st1_i);
        se(i).st1_o.p = se(i).st1_i.p;
        se(i).st2_o = Stream.flow(se(i).st2_i);
        se(i).st2_o.p = se(i).st2_i.p;
    end

    for j = 1:n1
        guess2(j,1) = se(1).st1_i.T.v - 40 * j;
        guess2(j,2) = se(1).st2_i.T.v + 4 * j;
    end
```

```
elseif (strcmp(order,'Reverse'))
    %%%%% Inverse order %%%%%
    se(1).cp_2 = se_cp_2;
    for i = 2:n1
        se(i) = StirlingEngine;
        se(i).flowType = se(1).flowType; % Flowtype of any Stirling engine can be
        se(i).cp_1 = se_cp_1;
        se(i).cp_2 = se_cp_2;
    end
    se(n1).st2_i = st2_se_i;
    se(n1).st2_o = se(n1).st2_i.flow();
    se(n1).st2_o.p = se(n1).st2_i.p;

    for i = 1:n1-1
        se(i+1).st1_i = se(i).st1_o;
        se(n1-i).st2_i = se(n1+1-i).st2_o;

        se(i+1).st1_o = se(i+1).st1_i.flow();
        se(i+1).st1_o.p = se(i+1).st1_i.p;
        se(n1-i).st2_o = se(n1-i).st2_i.flow();
        se(n1-i).st2_o.p = se(n1-i).st2_i.p;
    end

    for j = 1:n1
        guess2(j,1) = se(1).st1_i.T.v - 30 * j;
        guess2(j,2) = se(n1).st2_i.T.v + 4 * (n1 + 1 - j);
    end
else
    error('Uncomplished work.');
end

[x2, fval2] = fsolve(@(x2)CalcSEA(x2, se), guess2, options);
%%%%%%%%%%%%%%%%%% For comparison!!  %%%%%%%%%%%%%%%%%
%
% if (strcmp(order, 'Same'))
%     eta_ses1 = 1 - (st2_se_i.q_m.v * se_cp_2 * (se(n1).st2_o.T.v - ...
%         se(1).st2_i.T.v)) / (st1_se_i.q_m.v * se_cp_1 * ...
%         (se(1).st1_i.T.v - se(n1).st1_o.T.v));
% elseif (strcmp(order,'Reverse'))
%     eta_ses1 = 1 - (st2_se_i.q_m.v * se_cp_2 * (se(1).st2_o.T.v - ...
%         se(n1).st2_i.T.v)) / (st1_se_i.q_m.v * se_cp_1 * ...
%         (se(1).st1_i.T.v - se(n1).st1_o.T.v));
% else
%     error('Uncomplished work.');
% end

P = zeros(n1,1);

for i = 1:n1
    se(i).st1_o.T.v = x2(i, 1);
    se(i).st2_o.T.v = x2(i, 2);
    se(i).P = se(i).P1();
    P(i) = se(i).P2();
end
```

```
eta_ses = sum(P) ./ (st1_se_i.q_m.v * se_cp_1 * ...
    (se(1).st1_i.T.v - se(n1).st1_o.T.v));
P
eta_ses
```

```
Warning: Trust-region-dogleg algorithm of FSOLVE cannot handle non-square
systems; using Levenberg-Marquardt algorithm instead.

                                        First-Order                    Norm of
 Iteration  Func-count     Residual      optimality      Lambda           step
     0         101         55.1923            1.04         0.01
     1         202         7.23262           0.408        0.001        33.7703
     2         303         2.37794           0.572       0.0001        21.9072
     3         404        0.904698          0.0442        1e-05        64.5714
     4         505       0.0371071          0.0228        1e-06        115.264
     5         606     4.25097e-05         0.00134        1e-07        24.6898
     6         707     1.28062e-10        2.14e-07        1e-08       0.546059
     7         808     6.37095e-18        1.55e-10        1e-09      0.0016382

Equation solved, fsolve stalled.

fsolve stopped because the relative size of the current step is less than the
default value of the step size tolerance and the vector of function values
is near zero as measured by the default value of the function tolerance.




P =

   1.0e+03 *

    5.0585
    4.9410
    4.8259
    4.7132
    4.6030
    4.4951
    4.3895
    4.2861
    4.1850
    4.0860


eta_ses =

    0.3538
```

Trough Collector

```
amb = Ambient;
tc = TroughCollector;
```

```
tc.amb = amb;

st3(4) = Stream;

st3(1).fluid = char(Const.Fluid(3));
st3(1).T = Temperature(C2K(400));
st3(1).p = 2e6;
st3(1).q_m.v = 53.41;  % To be calculated

st3_tc_o = Stream;
st3_tc_o.fluid = char(Const.Fluid(3));
st3_tc_o.T = Temperature(C2K(350));
st3_tc_o.p = 2e6;
st3_tc_o.q_m.v = 3.41;  % To be calculated

st3_tc_i = st3_tc_o.flow();
st3_tc_i.T = Temperature(C2K(225));
st3_tc_i.p = st3_tc_o.p;

tc.st_i = st3_tc_i;
tc.st_o = st3_tc_o;

da = Deaerator;
da.p = 1e6;

st2(11) = Stream;

st2(1).fluid = char(Const.Fluid(2));
st2(1).T = Temperature(C2K(340));
st2(1).p = 2.35e6;
st2(1).q_m.v = 6.672;  % To be calculated

st2(2).fluid = st2(1).fluid;
st2(2).p = 1.5e4;

st2(3).fluid = st2(1).fluid;
% st2(3).p = da.p;
st2(3).p = 1e6;
tc
```

```
tc = 

  TroughCollector with properties:

          A: 545
      gamma: 0.9300
        rho: 0.9400
    shading: 1
        tau: 0.9500
      alpha: 0.9600
          w: 5.7600
         Fe: 0.9700
        d_i: 0.0660
        d_o: 0.0700
        phi: 1.2217
        amb: [1x1 Ambient]
       st_i: [1x1 Stream]
       st_o: [1x1 Stream]
      q_use: 2.7649e+05
      q_tot: 381500
        eta: 0.7247
```

Turbine

A steam turbine is created

```
tb = Turbine;
tb.st1 = st2(1);
tb.st2 = st2(2);
tb.st3 = st2(3);
tb.y = 0.1;
tb.calculate();
st2(2) = tb.st2;    % Necessary for the stream has been diverged in the turbine
st2(3) = tb.st3;    % Necessary for the stream has been diverged in the turbine
tb
```

```
tb =

  Turbine with properties:

      st1: [1x1 Stream]
      st2: [1x1 Stream]
      st3: [1x1 Stream]
        y: 0.1000
    eta_i: 0.7841
```

Condensor

A condensor is created

```
cd = Condensor;
cd.st1 = st2(2);
cd.st2;
cd
```

```
cd =

  Condensor with properties:

    st1: [1x1 Stream]
    st2: [1x1 Stream]
```

Published with MATLAB® R2014b