Documentation

Matlab Codes

HexModel.mat

1. Function name:

HexModel.mat

2. Model description:

HexModel.mat is a matlab function developed for the simulation of counter-flow heat exchangers. Six modelling paradigms are currently implemented to simulate the pump, i.e.

- CstPinch: a "constant-pinch" model into which a constant value of the pinch point is imposed between the temperature profiles of the two media.

- CstEff: a "constant-efficiency" model into which a constant value is imposed to the thermal efficiency of the heat exchanger. The thermal efficiency ε_{th} is defined as

  - $\varepsilon_{th} = \frac{\dot{Q}_{hex}}{\dot{Q}_{max}}$

  where $\dot{Q}_{hex}$ and $\dot{Q}_{max}$ are respectively the effective and the maximum heat power transferrable between the two media (i.e. pinch equal to 0).

- PolEff: similar to the CstEff model but uses instead a polynomial regression to define the value of the thermal efficiency. The polynomial is a quadratic function (2nd order) depending of the fluids mass flow rates.

- hConvCst: three-zone moving-boundary model that decomposes the heat exchanger in its different zones. Constant values to the convective heat transfer coefficients are provided by the user. The model implements the efficient algorithm proposed by Bell et al.[1][1] for computing the effective heat transfer in the heat exchanger. This effective heat transfer is calculated such as the total surface area occupied by the different zones corresponds to the geometrical surface area of the component. In the case of a heat exchanger with fins, the model accounts for the difference of surface area between the hot and cold fluid and implements Schmidt's theory to account for the fin efficiency.

---

[1] [1] I. H. Bell, S. Quoilin, E. Georges, J. E. Braun, E. A. Groll, T. W. Horton, and V. Lemort, "A generalized moving-boundary algorithm to predict the heat transfer rate of counterflow heat exchangers for any phase configuration," *Appl. Therm. Eng.*, vol. 79, pp. 192–201, 2015.

- hConvVar: idem as hConvCst but uses a flow-dependent relation to adapt the convective heat transfer coefficients in each zone to the operating conditions i.e.

  - $h_{conv} = h_{conv,nom} \left(\frac{\dot{m}}{\dot{m}_{nom}}\right)^n$

  where n is another parameter specified by the user in each zone.

- hConvCor: idem as hConvCst but computes the convective heat transfer coefficients in each zone by means of a Nusselt correlation in the form of

  - $Nu = C\ R_e^m P_r^n$

  where C, m and n are parameters specified by the user for each zone.

3. Model inputs:

The model inputs are the following ones:

- fluid_h: name of the hot fluid
- P_h_su (Pa), inlet pressure of the hot fluid
- in_h_su (K or J/kg), inlet temperature or enthalpy of the hot fluid
- m_dot_h (kg/s), mass flow rate of the hot fluid
- fluid_c, name of the cold fluid
- P_c_su (Pa), inlet pressure of the cold fluid
- in_c_su (K or J/kg), inlet temperature or enthalpy of the cold fluid
- m_dot_c (kg/s), mass flow rate of the cold fluid
- param: structure variable containing the model parameters

It is really important to note that the model can handle both types of inlet conditions: either a supply enthalpy or a supply temperature. By default, it is assumed that the fluid is incompressible if the temperature is provided as input (liquid phase only). It is highly recommended to provide the enthalpy as input for improving the simulation speed.

4. Model parameters:

The parameters required to run the heat exchanger model are all provided through the variable 'param'. Depending of the type of model chosen by the user, *param* will need to include the following variables:

- if param.modelType = 'CstPinch':
    - param.pinch = pinch point value in the temperature profile, [K];
    - param.type_h = type of input for hot fluid, ('H' for enthalpy,'T' for temperature);
    - param.type_c = type of input for cold fluid, ('H' for enthalpy,'T' for temperature);
    - param.displayResults = flag to display the results or not, [1/0];
    - param.displayTS =flag to display the temperature profiles or not, [1/0];

- if param.modelType = 'CstEff':
    - param.epsilon_th = effective thermal efficiency of the heat exchanger, [-];
    - param.type_h = type of input for hot fluid, ('H' for enthalpy,'T' for temperature);
    - param.type_c = type of input for cold fluid, ('H' for enthalpy,'T' for temperature);
    - param.displayResults = flag to display the results or not, [1/0];
    - param.displayTS = flag to display the temperature profiles or not, [1/0];

- if param.modelType = 'hConvCst':
    - param.CoeffPolEff = polynomial coefficients for calculating the effective thermal efficiency of the heat exchanger;
    - param.m_dot_h_n = nominal hot fluid mass flow rate, [kg/sec];
    - param.m_dot_c_n = nominal cold fluid mass flow rate, [kg/sec];
    - param.type_h = type of input for hot fluid, ('H' for enthalpy,'T' for temperature);
    - param.type_c = type of input for cold fluid, ('H' for enthalpy,'T' for temperature);
    - param.displayResults = flag to display the results or not, [1/0];
    - param.displayTS = flag to display the temperature profiles or not, [1/0] ;

- if param.modelType = 'hConvVar':
    - param.m_dot_h_n = HF nominal mass flow rate [kg/s];
    - param.hConv_h_liq_n = HF nominal convective heat transfer coefficient for liquid phase [W/m^2.K];
    - param.hConv_h_tp_n = HF nominal convective heat transfer coefficient for two-phase [W/m^2.K];
    - param.hConv_h_vap_n = HF nominal convective heat transfer coefficient for vapour phase [W/m^2.K];
    - param.m_dot_c_n = CF nominal mass flow rate [kg/s];
    - param.hConv_c_liq_n = CF nominal convective heat transfer coefficient for liquid phase [W/m^2.K];
    - param.hConv_c_tp_n = CF nominal convective heat transfer coefficient for two-phase [W/m^2.K];
    - param.hConv_c_vap_n = CF nominal convective heat transfer coefficient for vapour phase [W/m^2.K];

- param.A_tot (or param.A_h_tot && param.A_c_tot) = total surface area of HEX [m^2];
- param.type_h = type of input for hot fluid ('H' for enthalpy,'T' for temperature);
- param.type_c = type of input for cold fluid ('H' for enthalpy,'T' for temperature);
- param.displayResults = flag to display the results or not [1/0];
- param.displayTS = flag to display the temperature profiles or not [1/0];

- if param.modelType = 'hConvCor':
  - param.C_h_liq / param.n_h_liq / param.m_h_liq = correlation parameters for the hot fluid - liquid zone;
  - param.C_h_vap / param.n_h_vap / param.m_h_vap = correlation parameters for the hot fluid - vapour zone;
  - param.C_h_tp = coefficient for hot fluid - two phase zone;
  - param.C_c_liq / param.n_c_liq / param.m_c_liq = correlation parameters for the cold fluid - liquid zone;
  - param.C_c_vap / param.n_c_vap / param.m_c_vap = correlation parameters for the cold fluid - liquid zone;
  - param.C_c_tp = coefficient for cold fluid - two phase zone;
  - param.CS_h / param.Dh_h = cross-section and hydraulic diameter of the hot fluid;
  - param.CS_c / param.Dh_c = cross-section and hydraulic diameter of the cold fluid;
  - param.A_tot (or param.A_h_tot && param.A_c_tot) = total surface area of HEX [m^2];
  - param.type_h = type of input for hot fluid ('H' for enthalpy,'T' for temperature);
  - param.type_c = type of input for cold fluid ('H' for enthalpy,'T' for temperature);
  - param.displayResults = flag to display the results or not [1/0];
  - param.displayTS = flag to display the temperature profiles or not [1/0];

5. Model outputs:

The outputs are summarized in two structure variables, namely *out* and *TS*. The values included in this structured outputs depend of the modelling method chosen by the used. At minimum, they contain the following information:

- **out** (structure variable with all the relevant model outputs)
  - x_vec (-) : vector of power fraction in each zone
  - Qdot_vec (W) : vector of heat power in each zone
  - H_h_vec (J/kg): HF enthalpy vector
  - H_c_vec (J/kg): CF enthalpy vector
  - T_h_vec (K): HF temperature vector
  - T_c_vec (K): CF temperature vector
  - s_h_vec (J/kg.K): HF entropy vector

- s_c_vec (J/kg.K): HF entropy vector
- DT_vec (K): Temperature difference vector
- pinch (K): pinch point value
- h_h_ex (J/kg): HF exhaust enthalpy
- T_h_ex (K): HF exhaust temperature
- h_c_ex (J/kg): CF exhaust enthalpy
- T_c_ex (K): CF exhaust temperature
- time (sec) : the code computational time

- **TS** (structure variable only used to plot the Ts diagram)
  - T (K) : vector of the fluid temperatures
  - s (J/kg.K) : vector of the fluid entropies

6. External function requirements:

The user must install CoolProp (http://www.coolprop.org/) to run HexModel.mat. He also requires to download the following codes:

- HEX_Qdotmax (compute the maximum amount of heat power transferrable between two media – i.e. with a pinch equal to zero)
- HEX_profile (provide the temperature profiles in the heat exchanger based on the heat power transferred between the two media)

7. Matlab version:

This code has been developed under Matlab R2015a

8. Contact:

For any further information, please contact one of the main developers of ORCmKit:

- Rémi Dickes (rdickes@ulg.ac.be) – University of Liège (Belgium)
- Davide Ziviani (davide.ziviani@ugent.be) – Ghent University (Belgium)