Main Operating Scripts

* Solve\_Cycle\_Shotgun.m  
  For given inputs and parameters this script solves the system and returns data about the heat transfer and thermodynamic state variables at each station. I believe at present, it is a bit clunky and redundant.
* Analysis\_Run\_many\_cycles.m  
  First section runs the script “Solve\_Cycle\_Shotgun.m” for a range of inputs and saves the results. This generally takes a matter of hours to run. Make sure inputs in the sub-script are commented out. Second section loads the results from the previous, processes them, and returns a number of plots.
* Quick\_Latex.m  
  Writes a .tex file to the current directory with a table of parameters followed by a sequence of plots. Script loops over the provided list of workspace variables and writes a latex macro for a table entry. Then, loops over the provided list of figures, saves each one, and writes a latex macro to display the figure.

Plotting Functions

* DrawDome.m  
  inputs (str ordinate, str abscissa, varargin)  
  ordinate must be ‘T’ or ‘P’. abscissa can be ‘s’, ’h’, ’v’, ’rho’, I think. Function draws the vapor dome on the current plot. Varargin gives access to a few plot options.  
  example: DrawDome('T','s','type','linear','color',domecolor)
* plotcycle\_TS.m  
  inputs( array T, array s, varargin )  
  plots cycle on a Ts-diagram, and the vapor dome, returns the handle for the plot.  
  example: plotcycle\_TS( T, s, 'cyclecolor', rgb\_func(i/(I+1)) );
* plotcycle\_PH.m  
  inputs( array P, array h, varargin )  
  like plotcycle\_TS.m, but makes a Ph diagram.

“Outside” Cycle Functions – These utilize the component functions together. Workhorse functions.

* makecycle.m  
  inputs (3-array, Vars, array inputs, array params ),  
  outputs( array P, array T, array h, array abscissa, scalar m\_dot, scalar Q\_L, scalar Q\_H, scalar W, 3-array Deficit)  
  Given statevariables, inputs, and parameters, this function uses all the component functions to compute the cycle and returns key information about the cycle. See code for more information about these variables.
* adjust\_cycle\_fmin.m  
  inputs (3-array, Vars, array inputs, array params ),   
  outputs ( 3-array Vars )  
  Given an initial guess, inputs, and parameters, solves for the statevariables of the system.

Component Functions – Relate thermodynamic and flow properties to inlet/outlet states for each subsysytem. You may need to alter the Generate\_HTCOEFF.m function

* compr\_func.m  
  takes in inlet state as a 2-array [P,h]. Returns mass flow rate.
* Condenser\_Proc.m  
  integrates energy equations, returns P,T,h,z as 4-arrays
* valve\_func.m  
  returns mass flow rate. takes in parameter CA, high and low pressures, and valve opening fraction.
* Evap\_Proc.m  
  integrates energy equations, returns P,T,h,z as 4-arrays
* Generate\_HTCOEFF.m  
  This is the only function which uses hardcoded information about the heat exchangers

Thermodynamic Functions

* Ther\_rho.m  
  bit of an oddball function. Just smooth lines for the property ‘rho’ at saturation. This is used by the compressor function, to bypass jagged table lookup data.
* Ther\_rhoH.m  
  The same, but for ‘rho’ \* ’h’.
* XR410a.m  
  inputs (str ‘output\_prop’, str ‘input\_props’, varargin)  
  uses all below functions in some way. Not all pairs of properties are valid for lookup, I don’t remember all of which were allowed. To be safe, one of the input properties ought to be ‘T’ or ‘P’.
* SatLookupTP.m  
  interpolates saturation properties, table lookup
* SatLookup.m  
  uses SatLookupTP, but has considerations for vapor quality
* SuperheatLookup.m  
  self explanatory. uses superheated table lookup.
* SubCool.m  
  self explanatory. Uses saturated table lookup, but integrates various thermodynamic relations into the subcooled region.