```
   1
   2 % Clear the console
   3 clc;
   4 % Close all the windows
   5 close all;
   6 % Clear Workspace Variables
   7 clear;
   8
   9 global S1_MIN S1_MAX S1_POINTS;
  10 global S2_MIN S2_MAX S2_POINTS;
  11 global INVALID_FLOWRATE;
  12 global Fethyl_S1S2_plotOpt;
  13 global MT_PER_KT G_PER_KT GJ_PER_KJ;
  14 global VALUE_ETHANE VALUE_ETHYLENE VALUE_HYDROGEN_CHEM;
  15 global COST_RATES_STEAM;
  16 global VALUE_HYDROGEN_FUEL VALUE_METHANE_FUEL VALUE_PROPANE_FUEL VALUE_BUTANE_FUEL;
  17 global VALUE_NATGAS_FUEL VALUE_NUM2OIL_FUEL;
  18 global ENTHALPY_PROPANE ENTHALPY_BUTANE;
  19 global MOLMASS_PROPANE MOLMASS_BUTANE;
  20 global PROFIT_S1S2_OPT;
  21 global HEAT_CAPACITY_ETHANE;
  22 global HEAT_FORMATION_ETHANE;
  23 global STEAM_30PSIA STEAM_50PSIA STEAM_100PSIA STEAM_200PSIA STEAM_500PSIA↙
STEAM_750PSIA;
  24 global HYDROGEN METHANE ETHYLENE PROPANE BUTANE;
  25 global ENTHALPY_METHANE ENTHALPY_PROPANE ENTHALPY_BUTANE HEAT_CAPACITY_ETHANE;
  26 global KT_PER_G KG_PER_KT KJ_PER_GJ MT_PER_G ENTHALPY_NAT_GAS MOLMASS_ETHANE...
  27     MOLMASS_ETHYLENE MOLMASS_NATGAS;
  28 global MT_CO2_PER_KT_METHANE MT_CO2_PER_KT_PROPANE MT_CO2_PER_KT_BUTANE ...
  29     MT_CO2_PER_KT_NATURALGAS;
  30 global TAX_CO2_PER_MT;
  31 global STEAM_PRESSURE_COL STEAM_TEMP_COL;
  32 global MOLMASS_METHANE MOLMASS_WATER BAR_PER_PSIA;
  33 global C_TO_K HEAT_CAPACITY_WATER;
  34 global R k1_f k1_r k2 k3 R_2 C_TO_K YR_PER_SEC SEC_PER_YR MOLMASS_HYDROGEN
  35 global PSA_TOGGLE ENTHALPY_HYDROGEN T_SEPARATION P_SEPARATION M3_PER_L↙
DENSITY_LIQ_WATER
  36 global MAX_CAPEX MAX_OPEX MAX_TFCI PRESS_RXTR YEARS_IN_OPERATION MILLIONBTU_PER_GJ↙
YR_PER_HR HR_PER_YR
  37 global T_OVERRIDE P_OVERRIDE STEAM_MR_OVERRIDE
  38
  39 % USER NOTES_____________________________________________________________________
  40
  41 % Note: The primary (high level) units of this script are ...
  42 % Mass          kta
  43 % Energy        GJ
  44 % Pressure      Bar
  45 % Temperature   Celcius
  46 % Moles         Moles
  47 % Value         Dollars global T_OVERRIDE P_OVERRIDE STEAM_MR_OVERRIDE
  48
  49 % [ __ ] THIS MEANS DIMENSIONLESS UNITS
  50
  51 % USER INPUTS | DESIGN PARAMETERS_____________________________________________
  52
  53 % Product
  54 P_ETHYLENE_DES = 200;           % [ kta ]
  55     % Note! This design parameter's units are changed prior to the matrix def
```

```
 56
 57 YEARS_IN_OPERATION = 15 ;
 58
 59 % USER INPUTS | GLOBAL CONSTANTS_____________________________________________
 60
 61 % USER INPUTS | 3D PLOT, CONTOUR, LVL 2 & 3 CALCS_____________________________
 62
 63 % Reactor Conditions | 3D PLOT & CONTOUR PLOT (S1 S2) && THE LVL3 CALCS
 64 STEAM_TO_FEED_RATIO_MOLS = 0.6;     % [ __ ] 0.6 to 1.0
 65 TEMP_RXTR = 825;                    % [ C ]
 66 PRESS_RXTR = 2;                     % [ Bar ] 2 to 5 bar
 67 TEMP_ETHANE_FEED = 25;              % [ C ]
 68 CONVERSION =  0.17053;              % [ __ ] % Level 2 & 3 Calculations
 69 USERINPUT_S1 = 0.96971 ;            % [ __ ] % Level 2 & 3 Calculation  s
 70 USERINPUT_S2 = 0.00011843;          % [ __ ] % Level 2 & 3 Calculations
 71 STEAM_CHOICE = 1;
 72 %   STEAM_30PSIA = 1;
 73 %   STEAM_50PSIA = 2;
 74 %   STEAM_100PSIA = 3;
 75 %   STEAM_200PSIA = 4;
 76 %   STEAM_500PSIA = 5;
 77 %   STEAM_750PSIA = 6;
 78         % % Steam
 79         % % [ psia Temp[C] $/MT  kJ/kg ]
 80         % COST_RATES_STEAM = [
 81         %     30  121       2.38  2213;
 82         %     50  138       3.17  2159;
 83         %     100 165       4.25  2067;
 84         %     200 194       5.32  1960;
 85         %     500 242       6.74  1755;
 86         %     750 266       7.37  1634
 87         % ];
 88
 89 % Plotting | 3D PLOT & CONTOUR PLOT (S1 S2)
 90 NUM_POINTS = 10^4;
 91
 92 % USER INPUTS | RXTR TABLE PARAMETERS_____________________________________________
 93
 94 % Reactor Script Parameters | RXTR TABLE OUTPUT
 95 V_MIN = 0.1;                       % [ L ]
 96 V_MAX = 4 * 10^3;                      % [ L ]
 97 NUM_V_POINTS = 20;                 % [ __ ]
 98
 99 P_MIN = 2;                         % [ Bar ]
100 P_MAX = 5;                         % [ Bar ]
101 NUM_P_POINTS = 2;                  % [ __ ]
102
103 T_MIN = 775;                       % [ Celcius ]
104 T_MAX = 825;                       % [ Celcius ]
105 NUM_T_POINTS = 2;                  % [ __ ]
106
107 STEAM_MIN = 0.6;                   % [ __ ]
108 STEAM_MAX = 1.0;                   % [ __ ]
109 NUM_STEAM_POINTS = 2;              % [ __ ]
110
111 % Table Overrides | RXTR TABLE OUTPUT
112 T_P_OVERRIDE = true;
113     T_P_OVERRIDE_T = true;
```

```
114         T_OVERRIDE = 825;           %[C]
115     T_P_OVERRIDE_P = true;
116         P_OVERRIDE = 2;             %[Bar]
117     T_P_OVERRIDE_MR = true;
118         STEAM_MR_OVERRIDE = 0.6;%   [__]
119
120 % Output fuel costs
121 CONSOLE_OUTPUT_EFFECTIVE_VALUE_FUELS = true;
122
123 % output the cashflow matrix
124 CASHFLOW_MATRIX_OUTPUT = false;
125
126 % Output the level 2 and 3 calculations
127 OUTPUT_LVL3_FLOWRATES_TO_CONSOLE = true;
128     SANITY_CHECK_CALCULATIONS = true;
129
130 % Plot the 3D and Contour plot's
131 CALCULATE_ALL_SELECTIVITIES = true;
132     PLOT_ECON_3D = true;
133     PLOT_ECON_COUNTOUR = true;
134
135 % Output the Reactor Design tables
136 CALCULATE_REACTOR_FLOWS = true;
137
138 % PSA Toggle switch
139 PSA_TOGGLE = true;
140
141 % Do you want to add the work of the compressor to the heat flux of heating
142 % the steam from the temp it's avilable at, to the temp of the reactor?
143 ADD_COMPRESSOR_WORK_TO_STEAM_HEATFLUX = true;
144
145 % Separation System Thermodynamics
146 T_SEPARATION = 173.15;          % [ K ]
147 P_SEPARATION = PRESS_RXTR;      % [ bar ]
148 MAX_OPEX = false;       % [ __ ]
149 MAX_TFCI = false;
150 MAX_CAPEX = false;
151
152
153 % Zeolite and waste stream
154 % zeo 1.2 - 2.2 wt% absobtion = max of zeolite (g/g)
155
156 % NOTE SEARCH FOR "??" TO SEE MY ASSUMPTIONS AND OTHER NOTES IN THE CODE
157
158 % WORK OF THE COMPRESSOR HAS NOT BEEN IMPLEMENTED
159 % THE STEAM TO FEED RATIO LIKELY HAS UNIT ISSUES OF (g/g) vs (mol/mol)
160 %       I think I implemented both
161
162 %_____________________________________________________________________________
163 % DON'T TOUCH ANYTHING BELOW THIS LINE
164 %_____________________________________________________________________________
165
166
167 % CONSTANTS | PLOTTING_____________________________________________________
168
169 CONSOLE_SECTION_DIVIDER = ...
170     "_____________________________________________________________________";
171 S1_MIN = 0.01;
```

```
172 S1_MAX = 1.00;
173 S1_POINTS = NUM_POINTS ^ (1/2) ;
174 S2_MIN = 0.01;
175 S2_MAX = 1.00;
176 S2_POINTS = NUM_POINTS ^ (1/2);
177 INVALID_FLOWRATE = 0;
178 Fethyl_S1S2_plotOpt = { ...
179     'S_1 Selectivity', ...
180     'S_2 Selectivity', ...
181     'Ethylene Flowrate [kta]',...
182     'P_ethylene_VS_S1_S2.jpg'};
183 PROFIT_S1S2_OPT = { ...
184     'S_1 Selectivity', ...
185     'S_2 Selectivity', ...
186     'Annual Profit [$ MM USD]',...
187     'P_ethylene_VS_S1_S2.jpg'};
188
189 % CONSTANTS | UNITS_____________________________________________________________
190
191 % Mass
192 MT_PER_KT = 10^3;       % [ MT / kt ]
193
194 G_PER_KT = 10^9;        % [ g / kt ]
195 KT_PER_G = 10^-9;       % [ kt / g ]
196
197 KG_PER_KT = 10^6;       % [ kg / MT ]
198
199 MT_PER_G = 10^-6;       % [ MT / g ]
200
201 % Energy
202 GJ_PER_KJ = 10^-6;      % [ GJ / kJ ]
203 KJ_PER_GJ = 10^6;       % [ kJ / GJ ]
204
205 % Temperature
206 C_TO_K = 273.15;        % [ C -> K ]
207 % Value
208 MMDOLLA_PER_DOLLA = 10^-6;  % [ $ MM / $]
209 DOLLA_PER_MMDOLLA = 10^6;   % [ $ / $ MM ]
210
211 % Pressure
212 BAR_PER_PSIA = 0.0689476;   % [ Bar / Psia ]
213
214 % Time
215 YR_PER_SEC = 1 / (3.154 * 10^7);    % [ yr / s ]
216 SEC_PER_YR = 3.154 * 10^7;          % [ s / yr ]
217 YR_PER_HR = (1/8760 );              % [ yr / hr ]
218 HR_PER_YR = 8760;                   % [ hr / yr ]
219
220 % Volumes
221 M3_PER_L = 0.001;
222
223 % heat
224 MILLIONBTU_PER_GJ = 1.0551;         % [ ]
225
226 % CONSTANTS | PHYSICAL_________________________________________________________
227
228 DENSITY_LIQ_WATER = 10^3;           % [ kg / m^3 ]
229
```

```
230 % CONSTANTS | CHEMICAL_____________________________________________________
231
232 % Chemical | Molar Mass
233 MOLMASS_HYDROGEN = 2.01588;             % [ g / mol ]
234     % Source :  https://webbook.nist.gov/cgi/cbook.cgi?ID=1333-74-0
235 MOLMASS_METHANE = 16.0425;              % [ g / mol ]
236     % Source :  https://webbook.nist.gov/cgi/cbook.cgi?ID=74-82-8
237 MOLMASS_WATER = 18.015;                     % [ g / mol ]
238     % source : https://pubchem.ncbi.nlm.nih.gov/compound/Water
239 MOLMASS_CO2 = 44.01;                    % [ g / mol ]
240     % Source : https://pubchem.ncbi.nlm.nih.gov/compound/Carbon-dioxide-water
241 MOLMASS_PROPANE = 44.0956;              % [ g / mol ]
242     % Source : https://webbook.nist.gov/cgi/cbook.cgi?ID=C74986&Mask=1
243 MOLMASS_BUTANE = 58.1222;               % [ g / mol ]
244     % Source : https://webbook.nist.gov/cgi/cbook.cgi?ID=C106978&Mask=1
245 MOLMASS_ETHANE = 30.0690;               % [ g / mol ]
246     % Source : https://webbook.nist.gov/cgi/cbook.cgi?ID=C74840
247 MOLMASS_ETHYLENE = 28.0532;             % [ g / mol ]
248     % Source = https://webbook.nist.gov/cgi/cbook.cgi?ID=74-85-1&Type=IR-↙
SPEC&Index=QUANT-IR,20
249 MOLMASS_NATGAS = 16.04;                 % [ g / mol ]
250     % ASSUMING NATURAL GAS IS ALL METHANE
251
252 % Chemical | Combustion Stochiometery
253 CO2_TO_METHANE_COMBUSTION_STOICH = 1;
254 CO2_TO_PROPANE_COMBUSTION_STOICH = 3;
255 CO2_TO_BUTANE_COMBUSTION_STOICH = 4;
256 C02_TO_NATGAS_COMBUSTION_STOICH = CO2_TO_METHANE_COMBUSTION_STOICH;
257     % Natural gas is asuumed to be entirely methane
258
259
260
261 % CONSTANTS | THERMODYNAMICS_____________________________________________
262
263 % Gas Constant
264 R = 8.314;                              % [ J / mol K ]
265 R_2 = 0.0831446261815324;               % [ L bar / K mol ]
266
267 % Heat capacities
268 HEAT_CAPACITY_WATER = 33.79 * 10^-3;    % [ kJ / mol K ] Ref Temp = 298K
269     % Source : https://webbook.nist.gov/cgi/cbook.cgi?↙
ID=C14940637&Mask=1&Type=JANAFG&Table=on
270 HEAT_CAPACITY_ETHANE = 52.71 * 10^-3;   % [ kJ / mol K ] Reference Temp = 300K
271     % Source : https://webbook.nist.gov/cgi/cbook.cgi?ID=C74840&Units=SI&Mask=1EFF
272
273 % Heats of Formation (at 25C)
274 HEAT_FORMATION_ETHANE = -83.8;          % [ kJ / mol ] reference Temp = std
275     % Source : https://webbook.nist.gov/cgi/cbook.cgi?ID=C74840&Units=SI&Mask=1EFF
276 HEAT_FORMATION_METHANE = -74.87;        % [ kJ / mol ] reference Temp = std
277     % Source : https://webbook.nist.gov/cgi/cbook.cgi?ID=C74828&Mask=1
278 HEAT_FORMATION_ETHYLENE = 52.47;        % [ kJ / mol ] reference Temp = std
279     % Source : https://webbook.nist.gov/cgi/cbook.cgi?ID=C74851&Mask=1
280 HEAT_FORMATION_HYDROGEN = 0;            % [ kJ / mol ] reference Temp = std
281 HEAT_FORMATION_PROPANE = -104.7;        % [ kJ / mol ] reference Temp = std
282     % Source : https://webbook.nist.gov/cgi/cbook.cgi?ID=C74986&Mask=1
283 HEAT_FORMATION_BUTANE = -125.6;         % [ kJ / mol ] reference Temp = std
284     % Source : https://webbook.nist.gov/cgi/cbook.cgi?ID=C106978&Mask=1
285
```

```
286 % Enthalpy of combustion (std conditions)
287 ENTHALPY_HYDROGEN = 286;
288     % Source : https://chem.libretexts.org/Courses/University_of_Kentucky/UK%↙
3A_General_Chemistry/05%3A_Thermochemistry/5.3%3A_Enthalpy
289 ENTHALPY_METHANE = 890;                % [ kJ / mol ]
290     % Source : https://webbook.nist.gov/cgi/cbook.cgi?ID=C74828&Mask=1
291 ENTHALPY_PROPANE = 2219.2;              % [ kJ / mol ]
292     % Source : https://webbook.nist.gov/cgi/cbook.cgi?ID=C74986&Mask=1
293 ENTHALPY_BUTANE = 2877.5;               % [ kJ / mol ]
294     % Source : https://webbook.nist.gov/cgi/cbook.cgi?ID=C106978&Mask=1
295 ENTHALPY_NAT_GAS = ENTHALPY_METHANE;
296     % Source : https://afdc.energy.gov/fuels/natural_gas_basics.html#:~:↙
text=Natural%20gas%20is%20an%20odorless,used%20in%20the%20United%20States.
297     % Natural gas is mostly methane, so assumed to be 100% methane in the calcs
298
299 % Enthalpy of Reactions [ kJ / extent rxn]
300 ENTHALPY_RXN_1 = HEAT_FORMATION_HYDROGEN + HEAT_FORMATION_ETHYLENE ...
301                                         - HEAT_FORMATION_ETHANE;
302 ENTHALPY_RXN_2 = HEAT_FORMATION_METHANE + HEAT_FORMATION_PROPANE ...
303                                         - 2 * HEAT_FORMATION_ETHANE;
304 ENTHALPY_RXN_3 = HEAT_FORMATION_ETHANE - HEAT_FORMATION_ETHANE ...
305                                         - HEAT_FORMATION_ETHYLENE;
306 % CONSTANTS | ECONOMICS_____________________________________________________
307
308 %  Chemicals
309 VALUE_ETHANE = 200;             % [ $ / MT ]
310 VALUE_ETHYLENE = 900;           % [ $ / MT ]
311 VALUE_HYDROGEN_CHEM = 1400;     % [ $ / MT ]
312
313 % Steam
314 % [ psia Temp[C] $/MT  kJ/kg ]
315 COST_RATES_STEAM = [
316     30  121     2.38  2213;
317     50  138     3.17  2159;
318     100 165     4.25  2067;
319     200 194     5.32  1960;
320     500 242     6.74  1755;
321     750 266     7.37  1634
322 ];
323
324 % Accessing the Steam P,T Data
325     STEAM_PRESSURE_COL = 2;
326     STEAM_TEMP_COL = 1;
327     STEAM_COST_COL = 3;
328     STEAM_30PSIA = 1;
329     STEAM_50PSIA = 2;
330     STEAM_100PSIA = 3;
331     STEAM_200PSIA = 4;
332     STEAM_500PSIA = 5;
333     STEAM_750PSIA = 6;
334
335 % Economic | Fuel
336 VALUE_HYDROGEN_FUEL = 3;            % [ $ / GJ ]
337 VALUE_METHANE_FUEL = 3;             % [ $ / GJ ]
338 VALUE_PROPANE_FUEL = 3;             % [ $ / GJ ]
339 VALUE_BUTANE_FUEL = 3;              % [ $ / GJ ]
340 VALUE_NATGAS_FUEL = 3;              % [ $ / GJ ]
341 VALUE_NUM2OIL_FUEL = 4.5;           % [ $ / US Gallon ]
```

```
342
343 % Economics | Enviormental
344 TAX_CO2_PER_MT = 125;              % [ $ / MT ]
345
346 % [$ / GJ]  = 1GJ(basis) * (KJ / GJ)   * (mol gas / KJ) *          (mol CO2 / mol gas)          *  (g / mol C02)*(MT / g) * ($ / MT)
347 TAX_CO2_PER_GJ_METHANE = KJ_PER_GJ * (1 / ENTHALPY_METHANE) * CO2_TO_METHANE_COMBUSTION_STOICH * MOLMASS_CO2 * MT_PER_G * TAX_CO2_PER_MT;
348 TAX_CO2_PER_GJ_PROPANE = KJ_PER_GJ * (1 / ENTHALPY_PROPANE) * CO2_TO_PROPANE_COMBUSTION_STOICH * MOLMASS_CO2 * MT_PER_G * TAX_CO2_PER_MT;
349 TAX_CO2_PER_GJ_BUTANE = KJ_PER_GJ * (1 / ENTHALPY_BUTANE) * CO2_TO_BUTANE_COMBUSTION_STOICH * MOLMASS_CO2 * MT_PER_G * TAX_CO2_PER_MT;
350 TAX_CO2_PER_GJ_NATGAS = TAX_CO2_PER_GJ_METHANE; %
351
352 % Chemistry | MT of C02 per KT of Fuel used
353 % (MT CO2) = 1KT(basis) * (g / KT) * (mol gas/ g gas) *
354 MT_CO2_PER_KT_METHANE = G_PER_KT * (1/MOLMASS_METHANE) *...
355     ... % (mol CO2 / mol gas) *         (g CO2 / mol CO2) * (MT / g)
356     CO2_TO_METHANE_COMBUSTION_STOICH * MOLMASS_CO2 * MT_PER_G;
357 MT_CO2_PER_KT_PROPANE = G_PER_KT * (1/MOLMASS_PROPANE) *...
358     CO2_TO_PROPANE_COMBUSTION_STOICH * MOLMASS_CO2 * MT_PER_G;
359 MT_CO2_PER_KT_BUTANE = G_PER_KT * (1/MOLMASS_BUTANE) *...
360     CO2_TO_BUTANE_COMBUSTION_STOICH * MOLMASS_CO2 * MT_PER_G;
361 MT_CO2_PER_KT_NATURALGAS = MT_CO2_PER_KT_METHANE;
362
363 % FUNCTIONS | FLOWRATE_____________________________________________________
364
365 P_ETHYLENE = P_ETHYLENE_DES;
366 P_ETHYLENE_DES = P_ETHYLENE_DES * (1 / MOLMASS_ETHYLENE);
367 P_PROPANE = @(s1, s2)       (s2 / s1 *P_ETHYLENE_DES) * ...
368                                     MOLMASS_PROPANE;
369 P_BUTANE = @(s1, s2)        (P_ETHYLENE_DES*(1/(2*s1) - s2/s1 - 1/2)) * ...
370                                     MOLMASS_BUTANE;
371 F_ETHANE = @(s1, s2)        (P_ETHYLENE_DES / s1) * ...
372                                     MOLMASS_ETHANE;
373 P_METHANE = @(s1, s2) (s2 / s1 * P_ETHYLENE_DES) * ...
374                                     MOLMASS_METHANE;
375 P_HYDROGEN = @(s1, s2) (P_ETHYLENE_DES * ((1/(2*s1) - s2/s1 + 1/2))) * ...
376                                     MOLMASS_HYDROGEN;
377
378 % FUNCTIONS | EXTENT OF REACTION_____________________________________________
379
380 % Returns molar flowrates [ mol / yr ]
381 get_xi = @(flowrates) [ flowrates(HYDROGEN) * G_PER_KT / MOLMASS_HYDROGEN, ...
382                     flowrates(PROPANE) * G_PER_KT / MOLMASS_PROPANE, ...
383                     flowrates(BUTANE) * G_PER_KT / MOLMASS_BUTANE ];
384
385 % FUNCTIONS | VALIDATION______________________________________________________
386
387 flowrates_valid = @( flowrates ) all(flowrates >= 0);
388
389 % FUNCTIONS | ECONOMICS_____________________________________________________
390
391 % ($ / yr) =                  (kta) *   (MT / KT) * ($ / MT)
392 value_ethane = @(P_ethane) P_ethane * MT_PER_KT * VALUE_ETHANE;
393 value_ethylene = @(P_ethylene) P_ethylene * MT_PER_KT * VALUE_ETHYLENE;
394 value_h2_chem = @(P_h2_chem) P_h2_chem * MT_PER_KT * VALUE_HYDROGEN_CHEM;
395 value_methane = @(P_methane) P_methane * MT_PER_KT * VALUE_METHANE_FUEL;
```

```
396 value_propane = @(P_propane) P_propane * MT_PER_KT * VALUE_PROPANE_FUEL;
397 value_butane = @(P_butane) P_butane * MT_PER_KT * VALUE_BUTANE_FUEL;
398
399 % ($ / yr) =                              (kta) *  (MT / kt) * ($ / MT)
400 cost_steam = @(F_steam, steam_rate) F_steam * MT_PER_KT * steam_rate;
401
402 % FUNCTIONS | THEROMODYNAMICS_____________________________________________
403 % (GJ / yr) =                           (kta) *  (g / KT)   * (mol gas/ g gas)    *↙
(kJ / mol K)          * (GJ / KJ) * (K)
404 heat_ethane = @(F_ethane, T0, Tf) F_ethane * G_PER_KT * (1 / MOLMASS_ETHANE) *↙
HEAT_CAPACITY_ETHANE * GJ_PER_KJ * (Tf - T0);
405
406 % (GJ / yr)  =      (mol / yr) * (kJ / mol) *   (GJ / kJ)
407 heat_rxn1 = @(xi_1) xi_1 * ENTHALPY_RXN_1 * GJ_PER_KJ;
408 heat_rxn2 = @(xi_2) xi_2 * ENTHALPY_RXN_2 * GJ_PER_KJ;
409 heat_rxn3 = @(xi_3) xi_3 * ENTHALPY_RXN_3 * GJ_PER_KJ;
410 heat_rxn = @(xi) heat_rxn1(xi(1)) + heat_rxn2(xi(2)) + heat_rxn3(xi(3));
411
412 % FUNCTIONS | RATE CONTANTS_____________________________________________
413
414 % T is [ Kelvin ]  R is [ J / mol K ]
415 k1_f = @(T) (4.652 * 10^13) * exp( (-273000 / (R * (T ))));
416 k1_r = @(T) (9.91 * 10^8) * exp( (-137800 / (R * (T ))));
417 k2 = @(T) (4.652 * 10^11) * exp( (-273000 / (R * (T ))));
418 k3 = @(T) (7.083 * 10^13) * exp( (-252600 / (R * (T ))));
419
420
421 % DESIGN PARAMS_____________________________________________________________
422 STEAM_TO_FEED_RATIO_MASS = (MOLMASS_WATER / MOLMASS_ETHANE) *↙
STEAM_TO_FEED_RATIO_MOLS;
423
424
425 % SCRIPT____________________________________________________________________
426
427 % Economics | Post-Tax Value of different fuel sources
428 if (CONSOLE_OUTPUT_EFFECTIVE_VALUE_FUELS)
429     disp(" [ $ / GJ ] ")
430     EFFECTIVE_VALUE_HYDROGEN_FUEL = VALUE_HYDROGEN_FUEL
431     EFFECTIVE_VALUE_METHANE_FUEL = VALUE_METHANE_FUEL + TAX_CO2_PER_GJ_METHANE
432     EFFECTIVE_VALUE_PROPANE_FUEL = VALUE_PROPANE_FUEL + TAX_CO2_PER_GJ_PROPANE
433     EFFECTIVE_VALUE_BUTANE_FUEL = VALUE_BUTANE_FUEL + TAX_CO2_PER_GJ_BUTANE
434     EFFECTIVE_VALUE_NAT_GAS_FUEL = VALUE_NATGAS_FUEL + TAX_CO2_PER_GJ_NATGAS
435 %   EFFECTIVE_VALUE_NUM2_FUEL = VALUE_NATGAS_FUEL + TAX_CO2_PER_GJ_NUM2;
436
437 end
438
439 if (OUTPUT_LVL3_FLOWRATES_TO_CONSOLE)
440
441     % Calculate the flow rates of each species (kta)
442     P_hydrogen = P_HYDROGEN(USERINPUT_S1, USERINPUT_S2);
443     P_methane = P_METHANE(USERINPUT_S1, USERINPUT_S2);
444     P_ethylene = P_ETHYLENE;
445     P_propane = P_PROPANE(USERINPUT_S1, USERINPUT_S2);
446     P_butane = P_BUTANE(USERINPUT_S1, USERINPUT_S2);
447     F_ethane = F_ETHANE(USERINPUT_S1, USERINPUT_S2);
448     P_flowrates = [ P_hydrogen, P_methane, P_ethylene, P_propane, P_butane ];
449
450     disp(CONSOLE_SECTION_DIVIDER)
```

```
451     if (flowrates_valid(P_flowrates))
452
453         fprintf("Flowrates for the reactor given that s1 = %f, s2 = %f conv = %↙
f\n\n", ...
454             USERINPUT_S1, USERINPUT_S2, CONVERSION)
455
456         disp(CONSOLE_SECTION_DIVIDER)
457         disp("Level 2 Flowrates  in / out of the entire plant [ kt / yr ]")
458         P_hydrogen
459         P_methane
460         P_ethylene
461         P_propane
462         P_butane
463
464         disp("Fresh Feed Flowrate")
465         F_ethane
466
467         disp(CONSOLE_SECTION_DIVIDER)
468         disp("Level 3 Flowrates [ kt / yr ] ")
469
470         disp("Recycle Stream Flowrate")
471
472         R_ethane = F_ethane * ((1-CONVERSION) / (CONVERSION))
473         % R_ethane = (P_ethylene/USERINPUT_S1) * ((1-CONVERSION)/CONVERSION)
474
475         disp("Reactor Flowrates")
476
477         F_ethane_into_reactor = R_ethane + F_ethane
478
479         if SANITY_CHECK_CALCULATIONS
480             disp(CONSOLE_SECTION_DIVIDER)
481             disp("Sanity Checking the Calculations")
482             Conservation_of_mass = F_ethane - sum(P_flowrates)
483             if Conservation_of_mass
484                 fprintf("WARNING : YOU ARE NOT CONSERVING MASS\n\n")
485             end
486         end
487     else
488         disp("ERROR : Selectivities S1 S2 chosen are not physically possible")
489     end
490 end
491
492 % SCRIPT | PLOTTING_____________________________________________________________
493
494 if (CALCULATE_ALL_SELECTIVITIES)
495     disp(CONSOLE_SECTION_DIVIDER)
496     disp("Calculating all selectivities... ")
497     % Iterates through each value of selectivities S1 and S2 to find the economic
498     % potential for different reaction conditions
499     s1_domain = linspace(S1_MIN, S1_MAX, S1_POINTS);
500     s2_domain = linspace(S2_MIN, S2_MAX, S2_POINTS);
501     [s1_mesh, s2_mesh] = meshgrid(s1_domain, s2_domain);
502     % All flowrates are initialized as matricies of zeros
503     ethylene_flowrates = (s1_mesh + s2_mesh) .* 0;
504     hydrogen_flowrates = (s1_mesh + s2_mesh) .* 0;
505     methane_flowrates = (s1_mesh + s2_mesh) .* 0;
506     ethylene_flowrates = (s1_mesh + s2_mesh) .* 0;
507     propane_flowrates = (s1_mesh + s2_mesh) .* 0;
```

```
 508     butane_flowrates = (s1_mesh + s2_mesh) .* 0;
 509     ethane_flowrates = (s1_mesh + s2_mesh) .* 0;
 510
 511     profit = (s1_mesh + s2_mesh) .* 0;
 512
 513     % Flow rate Indicies | For the flowrates(i) array
 514     HYDROGEN = 1;
 515     METHANE = 2;
 516     ETHYLENE = 3;
 517     PROPANE = 4;
 518     BUTANE = 5;
 519
 520     i = 1;
 521     for s1 = s1_domain
 522         for s2 = s2_domain
 523
 524             P_hydrogen = P_HYDROGEN(s1, s2);
 525             P_methane = P_METHANE(s1, s2);
 526             P_ethylene = P_ETHYLENE;
 527             P_propane = P_PROPANE(s1, s2);
 528             P_butane = P_BUTANE(s1, s2);
 529             F_ethane = F_ETHANE(s1, s2);
 530
 531             P_flowrates = [ P_hydrogen, P_methane, P_ethylene, P_propane, P_butane↙
];
 532
 533             if (flowrates_valid(P_flowrates))
 534
 535                 % Store for plotting (kta)
 536                 hydrogen_flowrates(i) = P_HYDROGEN(s1, s2);
 537                 methane_flowrates(i) = P_METHANE(s1, s2);
 538                 ethylene_flowrates(i) = P_ETHYLENE;
 539                 propane_flowrates(i) = P_PROPANE(s1, s2);
 540                 butane_flowrates(i) = P_BUTANE(s1, s2);
 541                 ethane_flowrates(i) = F_ETHANE(s1, s2);
 542
 543                 % F_ethane = F_ETHANE(select_1(i), select_2(i));
 544                 % F_fresh_ethane = F_ethane;
 545                 % F_ethane_rxtr = F_ethane(i) * ( conversion(i) / (1 - conversion↙
(i)) );
 546
 547                 xi = [];
 548                 % Calculate the heat flux needed to keep reactor isothermal
 549                 heat_flux = 0;
 550                 xi = get_xi(P_flowrates);
 551                 F_steam = STEAM_TO_FEED_RATIO_MASS * F_ethane;
 552                 heat_flux = heat_flux + heat_ethane(F_ethane, TEMP_ETHANE_FEED,↙
TEMP_RXTR);
 553                 % heat_flux = heat_flux + heat_ethane(F_ethane_into_reactor,↙
TEMP_SEPARATION, TEMP_RXTR);
 554                 heat_flux = heat_flux + heat_steam(F_steam, STEAM_CHOICE,↙
PRESS_RXTR, TEMP_RXTR) ;
 555                 heat_flux = heat_flux + heat_rxn(xi);
 556
 557     %           % Use the heat flux to calculate the fuel cost
 558                 [combusted_fuel_flow_rates, heat_flux_remaining] = fuel_combustion↙
(heat_flux, P_flowrates);
 559
```

```
560                 % Calculate how much natural gas you needed to combust
561                 F_natural_gas = natgas_combustion(heat_flux_remaining);
562
563                 % Determine how much of the product streams were combusted to keep↙
the reactor isothermal
564
565                 combusted_hydrogen = combusted_fuel_flow_rates(HYDROGEN);
566                 combusted_methane = combusted_fuel_flow_rates(METHANE);
567                 combusted_propane = combusted_fuel_flow_rates(PROPANE);
568                 combusted_butane = combusted_fuel_flow_rates(BUTANE);
569
570     %           % VALUE CREATED | Primary Products
571                 profit(i) = profit(i) + value_ethylene(P_ethylene);
572                 profit(i) = profit(i) + value_h2_chem(P_hydrogen -↙
combusted_hydrogen);
573
574                 % VALUE CREATED | Non-combusted fuels
575                 % profit(i) = profit(i) + value_methane(P_methane -↙
combusted_methane);
576                     % ?? I don't think you can sell methane. IH - need to
577                     % determine energy requirements for compressors +
578                     % separation + cooling (will likely need to purchase
579                     % Nat Gas)
580                 profit(i) = profit(i) + value_propane(P_propane -↙
combusted_propane);
581                 profit(i) = profit(i) + value_butane(P_butane - combusted_butane);
582
583                 % COSTS INCURRED
584                 profit(i) = profit(i) - tax_C02(combusted_fuel_flow_rates,↙
F_natural_gas);
585                 profit(i) = profit(i) - cost_steam(F_steam, COST_RATES_STEAM↙
(STEAM_CHOICE, STEAM_COST_COL));
586                 profit(i) = profit(i) - value_ethane(F_ethane);
587                 profit(i) = profit(i) - cost_natural_gas_fuel(F_natural_gas);
588                 profit(i) = profit(i) - cost_waste_stream(F_steam);
589
590                 % profit(i) = profit(i) - cost
591
592
593             else
594                 profit(i) = INVALID_FLOWRATE;
595                 ethylene_flowrates(i) = INVALID_FLOWRATE;
596             end
597             i = i + 1;
598         end
599     end
600
601     profit = profit ./ 10^6; % Convert to Millions of dollars
602     profit(profit < 0) = 0; % remove irrelvant data
603
604     if (PLOT_ECON_COUNTOUR)
605         disp("Plotting EP Contour Map")
606         plot_contour(s1_mesh, s2_mesh, profit, PROFIT_S1S2_OPT);
607     end
608     if (PLOT_ECON_3D)
609         disp("Plotting 3D EP Surface Function")
610         plot_3D(s1_mesh, s2_mesh, profit, PROFIT_S1S2_OPT);
611     end
```

```
612
613     % Prepare the array of flow rate matrices
614     % flowRatesArray = {hydrogen_flowrates, methane_flowrates, ethylene_flowrates,↙
propane_flowrates, butane_flowrates, ethane_flowrates};
615
616     % Call the function with the desired row
617     % plotFlowRatesForRow(4, flowRatesArray); % To plot the first row across all↙
matrices
618 end
619
620
621 % SCRIPT | REACTOR ____________________________________________________________
622
623 T_RANGE = linspace(T_MIN, T_MAX, NUM_T_POINTS);
624 P_RANGE = linspace(P_MIN, P_MAX, NUM_P_POINTS);
625 STEAM_RANGE = linspace(STEAM_MIN, STEAM_MAX, NUM_STEAM_POINTS);
626 V_RANGE = [V_MIN, V_MAX]; % WARNING THESE ARE IN LITERS
627 % H2 Methane Ethane Propane Butane Ethylene
628 F_INTIAL_COND = [ 0; 0; 0; 0; 0; 10]; % These are in kta
629
630     % Product flow rate indicies
631     HYDROGEN = 1;
632     METHANE = 2;
633     ETHYLENE = 3;
634     PROPANE = 4;
635     BUTANE = 5;
636
637     % Feed flow rate index
638     ETHANE = 6;
639 % npv_T_P_MR = zeros(length(T_RANGE), length(P_RANGE), length(STEAM_RANGE), 1);
640 npv_T_P_MR = cell(length(T_RANGE), length(P_RANGE), length(STEAM_RANGE) );
641
642 i = 1;
643 j = 1;
644 k = 1;
645 if (CALCULATE_REACTOR_FLOWS)
646     disp("Reactor Script ")
647     for T_i = T_RANGE
648         for P_i = P_RANGE
649             for MR_S_i = STEAM_RANGE
650
651                 % override the T_i and P_i with user input
652                 if T_P_OVERRIDE
653                     disp("WARNING: OVERRIDE HAS BEEN ACTIVATED")
654                     if T_P_OVERRIDE_T
655                         T_i = T_OVERRIDE;
656                     end
657                     if T_P_OVERRIDE_P
658                         P_i = P_OVERRIDE;
659                     end
660                     if T_P_OVERRIDE_MR
661                         MR_S_i = STEAM_MR_OVERRIDE;
662                     end
663                 end
664
665                 fprintf("\n\nT = %f [C], P = %f [bar] MR = %f [__]\n", T_i, P_i,↙
MR_S_i)
666
```

```
667                 % Setup the PFR Design Equations
668
669                 % BASIS↙
CALCULATIONS_____________________________________________________________
670
671                 % CONVERT TO↙
MOLES____________________________________________________________
672                 % Convert all of the initial conditions to mol / s
673                 % (mol / s) =           (kt / yr) *              (g / kt) *   ( mol↙
/ g )        * ( yr / s)
674                 F_INTIAL_COND(METHANE) = F_INTIAL_COND(METHANE) * G_PER_KT *↙
(1/MOLMASS_METHANE) * YR_PER_SEC;
675                 F_INTIAL_COND(HYDROGEN) = F_INTIAL_COND(HYDROGEN) * G_PER_KT *↙
(1/MOLMASS_HYDROGEN) * YR_PER_SEC;
676                 F_INTIAL_COND(ETHANE) = F_INTIAL_COND(ETHANE) * G_PER_KT *↙
(1/MOLMASS_ETHANE) * YR_PER_SEC;
677                 F_INTIAL_COND(ETHYLENE) = F_INTIAL_COND(ETHYLENE) * G_PER_KT *↙
(1/MOLMASS_ETHYLENE) * YR_PER_SEC;
678                 F_INTIAL_COND(PROPANE) = F_INTIAL_COND(PROPANE) * G_PER_KT *↙
(1/MOLMASS_PROPANE) * YR_PER_SEC;
679
680                 % Calculate the molar flow rate of the steam
681                 % mol/s = __    * mol / s
682                 F_steam = MR_S_i * F_INTIAL_COND(ETHANE);
683
684                 % Solve the system ODE's
685                 %   (L, mol / s)           (L, mol/s, Celcius, Bar, mol/s)
686                 odes = @(V, F) reactionODEs(V, F, T_i, P_i, F_steam);
687                 [V_soln_ODE, F_soln_ODE] = ode45(odes, V_RANGE, F_INTIAL_COND);
688
689                 % Calculate the conversion
690                 conversion = (F_INTIAL_COND(ETHANE) - F_soln_ODE(:, ETHANE)) /↙
F_INTIAL_COND(ETHANE);
691
692                 % put handles length of the solution and the initial ethane flow
693                 len = length(F_soln_ODE(:, 1));
694                 F_ethane_initial = ones(len, 1) * F_INTIAL_COND(ETHANE);
695
696                 % Calculate the Selectivities, for each row (aka V_rxtr)
697                 select_1 = (F_soln_ODE(:, ETHYLENE) ) ./ (F_ethane_initial -↙
F_soln_ODE(:, ETHANE));
698                 select_2 = (F_soln_ODE(:, PROPANE) ) ./ (F_ethane_initial -↙
F_soln_ODE(:, ETHANE));
699
700                 % Calculate the inlet volumetric flow rate
701                 % (L / s) ??????????????
702                 P_sum = F_soln_ODE(:, HYDROGEN:BUTANE);
703                 % Turn these constants into vectors to operation is valid
704                 F_steam = ones(length(P_sum(:,1)), 1) .* F_steam;
705                 % put handles on terms, to make the code readable
706                 sum_flowrates_into_reactor = F_INTIAL_COND(ETHANE) + F_steam;
707                 % Calculate the flow rate into the reactor
708                 q0 = (R_2  * (T_i + C_TO_K) / P_i) .* sum_flowrates_into_reactor;
709                     % This is F.30 in the 'Design PFR Algorithm Appendix'
710
711                 % PLANT↙
CALCULATIONS_____________________________________________________________
712
```

```
 713                 % Calculate the the flowrates of the plant sized reactor given S1,↙
S2 from ODE's
 714                 F_ethane = [];
 715                 P_ethylene = [];
 716                 for row = 1:length(select_1)
 717                     % mol / s          = (kt / yr)   * (g / kt)  * (mol / g)↙
* (yr / s)
 718                     P_ethylene(row, 1) = P_ETHYLENE .* G_PER_KT .*↙
(1/MOLMASS_ETHYLENE) * YR_PER_SEC;
 719                 end
 720
 721                 % Calculate the scaling factor of the plant, from the basis
 722                 % mol / mol = ...
 723                 scaling_factor = P_ethylene(:, 1) ./ F_soln_ODE(:, ETHYLENE);
 724
 725                 % Calculate the volume of the plant sized reactor
 726                 % L / s = ( L / s )         * (   (mol / s )  ) / (    (mol / s)↙
)
 727                 %         BASIS             *     PLANT_FLOW    /      BASIS_FLOW
 728                 V_plant = V_soln_ODE(:, 1) .* scaling_factor;
 729
 730                 % cost of the reactor
 731                 cost_rxt_vec = zeros(size(V_plant));
 732                 for row = 1:length(V_plant)
 733                     % ( $ )
 734                     cost_rxt_vec(row) = cost_reactor(V_plant(row,1) * M3_PER_L);
 735                     cost_rxt_vec(row) = cost_rxt_vec(row) / YEARS_IN_OPERATION;
 736                 end
 737
 738                 % inlet flow of the plant scaled reactor
 739                 q0_plant = q0(:, 1) .* scaling_factor;
 740                     % Eqn F.35 in 'Design PFR Algorithm Appendix'
 741
 742                 % Scaling all of the molar flowrates to the size of the plant
 743                 F_soln_ODE(:, METHANE) = F_soln_ODE(:, METHANE) .* scaling_factor;
 744                 F_soln_ODE(:, HYDROGEN) = F_soln_ODE(:, HYDROGEN) .* scaling_factor;
 745                 F_soln_ODE(:, ETHANE) = F_soln_ODE(:, ETHANE) .* scaling_factor;
 746                 F_soln_ODE(:, ETHYLENE) = F_soln_ODE(:, ETHYLENE) .* scaling_factor;
 747                 F_soln_ODE(:, BUTANE) = F_soln_ODE(:, BUTANE) .* scaling_factor;
 748                 F_soln_ODE(:, PROPANE) = F_soln_ODE(:, PROPANE) .* scaling_factor;
 749
 750                 % CONVERT BACK TO↙
MASS_____________________________________________________________
 751
 752                 % convert back to kta
 753                 % kt / yr =  mol / s    * g / mol         * kt / g   * s / yr
 754                 F_soln_ODE(:, METHANE) = F_soln_ODE(: ,METHANE) * MOLMASS_METHANE *↙
KT_PER_G * SEC_PER_YR;
 755                 F_soln_ODE(:, ETHANE) = F_soln_ODE(:, ETHANE) * MOLMASS_ETHANE *↙
KT_PER_G * SEC_PER_YR;
 756                 F_soln_ODE(:, HYDROGEN) = F_soln_ODE(:, HYDROGEN) * MOLMASS_HYDROGEN↙
* KT_PER_G * SEC_PER_YR;
 757                 F_soln_ODE(:, ETHYLENE) = F_soln_ODE(:, ETHYLENE) * MOLMASS_ETHYLENE↙
* KT_PER_G * SEC_PER_YR;
 758                 F_soln_ODE(:, BUTANE) = F_soln_ODE(:, BUTANE) * MOLMASS_BUTANE *↙
KT_PER_G * SEC_PER_YR;
 759                 F_soln_ODE(:, PROPANE) = F_soln_ODE(:, PROPANE) * MOLMASS_PROPANE *↙
KT_PER_G * SEC_PER_YR;
```

```
 760
 761                 % Check if you're conserving mass
 762                 conserv_mass = zeros(length(F_soln_ODE(:,1)), 1);
 763                 npv = zeros(length(F_soln_ODE(:,1)), 1);
 764                 fxns.separationCosts = zeros(length(F_soln_ODE(:,1)), 1);
 765                 fxns.furnaceCosts = zeros(length(F_soln_ODE(:,1)), 1);
 766                 fxns.F_steam = zeros(length(F_soln_ODE(:,1)), 1);
 767                 fxns.F_fresh_ethane = zeros(length(F_soln_ODE(:,1)), 1);
 768                 xi = [ 0 , 0, 0];   %init
 769
 770                 % ECONOMIC↙
CALCULATIONS_____________________________________________________________
 771                 profit = zeros(length(F_soln_ODE(:,1)), 1);
 772                 for i = 1:length(F_soln_ODE(:, 1))
 773
 774                     % DEBUGGING
 775                     if i > 500
 776                         disp("")
 777                     end
 778                     % P_flowrates = [ P_hydrogen, P_methane, P_ethylene, P_propane,↙
P_butane ];
 779                     P_flowrates = F_soln_ODE(i , HYDROGEN:BUTANE);
 780
 781                     P_hydrogen = P_flowrates(HYDROGEN);
 782                     P_methane = P_flowrates(METHANE);
 783                     P_ethylene = P_flowrates(ETHYLENE);
 784                     P_propane = P_flowrates(PROPANE);
 785                     P_butane = P_flowrates(BUTANE);
 786
 787                     F_fresh_ethane = F_ETHANE(select_1(i), select_2(i));
 788                     R_ethane = F_fresh_ethane * ( ( 1 - conversion(i)) / conversion↙
(i) );
 789                     R_ethane = F_soln_ODE(i, ETHANE);
 790                         % ?? These two values R should be the same
 791
 792                     if (~flowrates_valid(P_flowrates))
 793                         disp("WARNING SOME FLOWATES MAY BE INVALID")
 794                     end
 795
 796                     % Calculate the heat flux needed to keep reactor isothermal
 797                     heat_flux = 0;
 798                     xi = get_xi(P_flowrates);
 799                     F_steam = STEAM_TO_FEED_RATIO_MASS * (F_fresh_ethane +↙
R_ethane);
 800                     heat_flux = heat_flux + heat_ethane(F_fresh_ethane,↙
TEMP_ETHANE_FEED, TEMP_RXTR);
 801                     heat_flux = heat_flux + heat_ethane(R_ethane, T_SEPARATION -↙
C_TO_K, TEMP_RXTR);
 802                     heat_flux = heat_flux + heat_steam(F_steam, STEAM_CHOICE,↙
PRESS_RXTR, TEMP_RXTR) ;
 803                     heat_flux = heat_flux + heat_rxn(xi);
 804
 805                     % Use the heat flux to calculate the fuel cost
 806                     [combusted_fuel_flow_rates, heat_flux_remaining] =↙
fuel_combustion(heat_flux, P_flowrates);
 807
 808                     % Calculate how much natural gas you needed to combust
 809                     F_natural_gas = natgas_combustion(heat_flux_remaining);
```

```
 810
 811                     % Determine how much of the product streams were combusted to↙
keep the reactor isothermal
 812                     combusted_hydrogen = combusted_fuel_flow_rates(HYDROGEN);
 813                     combusted_methane = combusted_fuel_flow_rates(METHANE);
 814                     combusted_propane = combusted_fuel_flow_rates(PROPANE);
 815                     combusted_butane = combusted_fuel_flow_rates(BUTANE);
 816
 817                     % VALUE CREATED | Primary Products
 818                     profit(i, 1) = profit(i, 1) + value_ethylene(P_ethylene);
 819                     profit(i, 1) = profit(i, 1) + value_h2_chem(P_hydrogen -↙
combusted_hydrogen);
 820
 821                     % VALUE CREATED | Non-combusted fuels
 822                     % The commented line can be removed or modified as per the↙
context.
 823                     % profit(i, 1) = profit(i, 1) + value_methane(P_methane -↙
combusted_methane);
 824                     profit(i, 1) = profit(i, 1) + value_propane(P_propane -↙
combusted_propane);
 825                     profit(i, 1) = profit(i, 1) + value_butane(P_butane -↙
combusted_butane);
 826
 827                     % COSTS INCURRED
 828                     profit(i, 1) = profit(i, 1) - tax_C02(combusted_fuel_flow_rates,↙
F_natural_gas);
 829                     profit(i, 1) = profit(i, 1) - cost_steam(F_steam,↙
COST_RATES_STEAM(STEAM_CHOICE, STEAM_COST_COL));
 830                     profit(i, 1) = profit(i, 1) - value_ethane(F_fresh_ethane);
 831                     profit(i, 1) = profit(i, 1) - cost_natural_gas_fuel↙
(F_natural_gas);
 832                     profit(i, 1) = profit(i, 1) - cost_waste_stream(F_steam);
 833                     profit(i, 1) = profit(i, 1) - cost_separation_system↙
(P_flowrates, F_steam, R_ethane);
 834                     profit(i, 1) = profit(i, 1) - calculate_installed_cost↙
(heat_flux);
 835
 836                     % Store Data For analysis
 837                     fxns.separationCosts(i, 1) = cost_separation_system(P_flowrates,↙
F_steam, R_ethane);
 838                     fxns.furnaceCosts(i, 1)  = calculate_installed_cost(heat_flux);
 839                     fxns.F_steam(i, 1) = F_steam;
 840                     fxns.F_fresh_ethane(i, 1) = F_fresh_ethane;
 841
 842                     % Checking if I still have any sanity left after this, who↙
knows...
 843                     conserv_mass(i, 1) = F_fresh_ethane - sum(P_flowrates);
 844
 845                     % NPV params
 846                     npv_params.mainProductRevenue = value_ethylene(P_ethylene) *↙
MMDOLLA_PER_DOLLA;
 847                     npv_params.byProductRevenue = value_h2_chem(P_hydrogen -↙
combusted_hydrogen) * MMDOLLA_PER_DOLLA;
 848                     npv_params.rawMaterialsCost = value_ethane(F_fresh_ethane) *↙
MMDOLLA_PER_DOLLA;
 849                     npv_params.utilitiesCost = cost_steam(F_steam, COST_RATES_STEAM↙
(STEAM_CHOICE, STEAM_COST_COL)) * MMDOLLA_PER_DOLLA;
 850                     npv_params.CO2sustainabilityCharge = tax_C02↙
```

```
(combusted_fuel_flow_rates, F_natural_gas) * MMDOLLA_PER_DOLLA;
 851                     npv_params.conversion = conversion(i);
 852                     % npv_params.ISBLcapitalCost = (cost_rxt_vec(i) +↙
cost_separation_system(P_flowrates, F_steam, R_ethane)) * MMDOLLA_PER_DOLLA;
 853                     npv_params.ISBLcapitalCost = (cost_rxt_vec(i) + ...
 854                                                   cost_separation_system(P_flowrates,↙
F_steam, R_ethane) + ...
 855                                                   calculate_installed_cost(heat_flux)) *↙
MMDOLLA_PER_DOLLA;
 856
 857                     % NPV calculations
 858                     cf = get_npv(npv_params);
 859                     npv(i, 1) = cf.lifetime_npv;
 860                     if conversion(i) > 0.67 && conversion(i) < 0.70
 861                         cf = get_npv(npv_params);
 862                         ideal_cf = cf;
 863                         ideal_params = npv_params;
 864                         ideal_conversion = conversion(i);
 865                         ideal_lifetimeNpv = cf.lifetime_npv;
 866                     end
 867
 868                 end
 869
 870                 % Assuming A is your matrix
 871                 % A = [1 2 3; 4 5 6; 7 8 9; 10 11 12]; % Example matrix
 872
 873                 % Find the maximum value in the 3rd column and its row index
 874                 % [maxValue, rowIndex] = max(A(:,3));
 875
 876                 % [maxValue, maxRowIndex] = max(npv(:,1));
 877
 878                 % max value NPV for all T P MR
 879                 % npv_T_P_MR(i,j,k) = npv(maxRowIndex, 1);
 880                 % npv_T_P_MR(i,j,k) = npv(:,1);
 881 %               temp  = npv);
 882 %               npv_T_P_MR{i,j,k} = npv;
 883
 884
 885                 % % Plotting the Capstone plots
 886                 % fxns.conversion = conversion;
 887                 % fxns.V_plant = V_plant;
 888                 % fxns.select_1 = select_1;
 889                 % fxns.select_2 = select_2;
 890                 % fxns.npv = npv;
 891                 % fxns.recycle = F_soln_ODE( : , ETHANE);
 892                 % fxns.freshFeedRawMaterials = fxns.F_fresh_ethane + fxns.F_steam;
 893                 % fxns.productionRateRxnProducts = F_soln_ODE( : , HYDROGEN :↙
BUTANE);
 894                 % fxns.F_rxtr_in_total = fxns.F_fresh_ethane + fxns.recycle + fxns.↙
F_steam;
 895                 % fxns.F_sep = sum(F_soln_ODE(: , HYDROGEN : ETHANE), 2) + fxns.↙
F_steam;
 896                 % fxns.x_hydrogen_sep = F_soln_ODE( : , HYDROGEN) ./ fxns.F_sep;
 897                 % fxns.x_methane_sep = F_soln_ODE( : , METHANE) ./ fxns.F_sep;
 898                 % fxns.x_ethylene = F_soln_ODE( : , ETHYLENE) ./ fxns.F_sep;
 899                 % fxns.x_propane_sep = F_soln_ODE( : , PROPANE) ./ fxns.F_sep;
 900                 % fxns.x_butane_sep = F_soln_ODE( : , BUTANE) ./ fxns.F_sep;
 901                 % fxns.x_ethane_sep = F_soln_ODE( : , ETHANE) ./ fxns.F_sep;
```

```
902                 % fxns.x_water_sep = fxns.F_steam ./ fxns.F_sep;
903
904                 % plot_conversion_fxns(fxns);
905
906
907
908
909                 % Debugging
910                 if CASHFLOW_MATRIX_OUTPUT
911                     fprintf("\n\nnpv = ($ MM) %3.3f \n", ideal_lifetimeNpv)
912                     format short
913                     % disp(ideal_cf.matrix)
914
915                     disp(ideal_params)
916                     fprintf("conversion = %1.4f\n", ideal_conversion)
917
918                     A = [123456789, 987654321; 12345, 67890]; % Example 2D array
919
920                     % Loop through each element and print
921                     disp("CASH FLOW MATRIX")
922                     A = ideal_cf.matrix;
923                     [row, col] = size(A);
924                     for i = 1:row
925                         for j = 1:col
926                             fprintf('%6.1f\t', A(i,j)); % Adjust the format↙
specifier as needed
927                         end
928                         fprintf('\n');
929                     end
930
931                     % cf.matrix
932                     % cf.lifetime_npv
933                 end
934
935
936
937                 %↙
PLOTTING_____________________________________________________________________
938                 col_names = {'V_rxtr [L] ', 'Hydrogen [kta]', 'Methane', ...
939                     'Ethylene', 'Propane', 'Butane','Ethane', 'conversion', ...
940                     'S1', 'S2', 'q0 [ L /s ]', 'Vol_plant [ L ]', 'q0 plant', 'cost↙
reactor', 'profit', 'net profit', 'conserv mass', 'npv', 'separationCosts', 'Furnace↙
Costs'};
941                 soln_table = table( V_soln_ODE, F_soln_ODE(:, HYDROGEN), ...
942                             F_soln_ODE(:, METHANE), F_soln_ODE(:, ETHYLENE), ...
943                             F_soln_ODE(:, PROPANE), F_soln_ODE(:, BUTANE), ...
944                             F_soln_ODE(:, ETHANE), conversion,select_1, ...
945                             select_2,q0,V_plant,q0_plant,cost_rxt_vec,profit, profit↙
- cost_rxt_vec, conserv_mass,npv,fxns.separationCosts,fxns.furnaceCosts,'VariableNames',↙
col_names)
946                 soln_table.Properties.VariableNames = col_names;
947
948                 % Computer Selectivity vs conversion relationships
949
950                 % Use Selectivity vs Conversion Relationships with lvl 2 & 3↙
balances
951                 % % to calculate the true feed flow rates into the reactor
952
```

```
 953                 % % ?? MODIFY ALL OF THESE TO BE IN MILLIONS OF DOLLARS
 954                 % npv.mainProductRevenue = value_ethylene(P_ethylene);
 955                 % npv.byProductRevenue = value_h2_chem(P_hydrogen -↙
combusted_hydrogen);
 956                 % npv.rawMaterialsCost = value_ethane(F_fresh_ethane);
 957                 % npv.utilitiesCost = cost_steam(F_steam, COST_RATES_STEAM↙
(STEAM_CHOICE, STEAM_COST_COL));
 958                 % npv.CO2sustainabilityCharge = tax_C02(combusted_fuel_flow_rates,↙
F_natural_gas);
 959                 % npv.conversion = conversion(i);
 960                 % npv.ISBLcapitalCost = cost_rxt_vec + cost_separation_system↙
(P_flowrates, F_steam, R_ethane);
 961                 % % NPV CALCS
 962
 963             k = k + 1;
 964             end
 965         j = j + 1;
 966         end
 967     i = i + 1;
 968     end
 969 end
 970
 971 % Plotting the Capstone plots
 972
 973 fxns.conversion = conversion;
 974 fxns.V_plant = V_plant;
 975 fxns.select_1 = select_1;
 976 fxns.select_2 = select_2;
 977 fxns.npv = npv;
 978 fxns.recycle = F_soln_ODE( : , ETHANE);
 979 fxns.freshFeedRawMaterials = fxns.F_fresh_ethane + fxns.F_steam;
 980 fxns.productionRateRxnProducts = F_soln_ODE( : , HYDROGEN : BUTANE);
 981 fxns.F_rxtr_in_total = fxns.F_fresh_ethane + fxns.recycle + fxns.F_steam;
 982 fxns.F_sep = sum(F_soln_ODE(: , HYDROGEN : ETHANE), 2) + fxns.F_steam;
 983 fxns.x_hydrogen_sep = F_soln_ODE( : , HYDROGEN) ./ fxns.F_sep;
 984 fxns.x_methane_sep = F_soln_ODE( : , METHANE) ./ fxns.F_sep;
 985 fxns.x_ethylene_sep = F_soln_ODE( : , ETHYLENE) ./ fxns.F_sep;
 986 fxns.x_propane_sep = F_soln_ODE( : , PROPANE) ./ fxns.F_sep;
 987 fxns.x_butane_sep = F_soln_ODE( : , BUTANE) ./ fxns.F_sep;
 988 fxns.x_ethane_sep = F_soln_ODE( : , ETHANE) ./ fxns.F_sep;
 989 fxns.x_water_sep = fxns.F_steam ./ fxns.F_sep;
 990 fxns.npv_T_P_MR = npv_T_P_MR;
 991
 992 plot_conversion_fxns(fxns);
 993
 994
 995
 996 disp("The Script is done running  )'
 997 % HELPER FUNCTIONS | PLOTTING_____________________________________________
 998
 999 function z = plot_contour(x, y, z, options)
1000     global PSA_TOGGLE
1001     % Unpack options
1002     x_label = options{1};
1003     y_label = options{2};
1004     plt_title = options{3};
1005     plt_saveName = options{4};
1006
```

```
1007     if PSA_TOGGLE
1008         stringValue = 'true';
1009     else
1010         stringValue = 'false';
1011     end
1012     plt_title = plt_title + sprintf(" PSA %s ", stringValue);
1013 
1014     hold on
1015     figure
1016     [C, h] = contourf(x, y, z); % Create filled contours
1017     clabel(C, h,  'FontSize', 10, 'Color', 'k', 'LabelSpacing', 200); % Customize↙
label properties
1018     xlabel(x_label);
1019     ylabel(y_label);
1020     title(plt_title);
1021     saveas(gcf, plt_saveName);
1022     hold off
1023 end
1024 
1025 function plot_3D(x, y, z, options)
1026     global PSA_TOGGLE
1027 
1028     % Unpack options
1029     x_label = options{1};
1030     y_label = options{2};
1031     plt_title = options{3};
1032     plt_saveName = options{4};
1033 
1034     if PSA_TOGGLE
1035         stringValue = 'true';
1036     else
1037         stringValue = 'false';
1038     end
1039     plt_title = plt_title + sprintf(" PSA %s ", stringValue);
1040 
1041      % Create a new figure
1042     hold on; % Hold on to add multiple plot elements
1043     figure
1044     surf(x, y, z); % Create a 3D surface plot
1045 
1046     % Customizing the plot
1047     xlabel(x_label);
1048     ylabel(y_label);
1049     zlabel('Z Value'); % Add a label for the z-axis
1050     title(plt_title);
1051     colorbar; % Adds a color bar to indicate the scale of z values
1052 %     shading interp; % Option for smoother color transition on the surface
1053 
1054     hold off; % Release the figure
1055     saveas(gcf, plt_saveName); % Save the figure to file
1056 end
1057 
1058 
1059 function plotFlowRatesForRow(row, flowRatesArray)
1060     % flowRatesArray is expected to be an array of matrices, where each matrix↙
corresponds to a species' flow rates
1061 
1062     % Names of the gases for labeling purposes
```

```
1063     gasNames = {'Hydrogen', 'Methane', 'Ethylene', 'Propane', 'Butane', 'Ethane'};
1064
1065     % Create a figure
1066     figure;
1067     hold on; % Hold on to plot all data on the same figure
1068
1069     % Loop through each flow rate matrix in the array
1070     for i = 1:length(flowRatesArray)
1071         % Extract the specified row from the current matrix
1072         currentRow = flowRatesArray{i}(row, :);
1073
1074         % Plot the current row with a marker
1075         plot(currentRow, '-o', 'DisplayName', gasNames{i});
1076     end
1077
1078     % Adding plot features
1079     title(sprintf('Flow Rates for Row %d', row));
1080     xlabel('Selectivity 1 (S2 fixed)');
1081     ylabel('Flow Rate');
1082     legend('show');
1083     hold off; % Release the figure for other plots
1084 end
1085
1086 % HELPER FUNCTIONS | HEAT ____________________________________________________
1087
1088 function [combusted_fuel_flowrates, heatflux_left] = fuel_combustion(heat_flux,↙
flowrates)
1089     global HYDROGEN METHANE ETHYLENE PROPANE BUTANE;
1090     global ENTHALPY_METHANE ENTHALPY_PROPANE ENTHALPY_BUTANE HEAT_CAPACITY_ETHANE;
1091     global MT_PER_KT G_PER_KT GJ_PER_KJ KJ_PER_GJ MOLMASS_METHANE KT_PER_G↙
MOLMASS_BUTANE ...
1092             MOLMASS_PROPANE PSA_TOGGLE ENTHALPY_HYDROGEN MOLMASS_HYDROGEN
1093
1094     % Note! : Longest Chain Hydrocarbons are cheapest to combust
1095
1096     % initialize all values in the array to be zero
1097     combusted_fuel_flowrates = flowrates * 0;
1098
1099     % LOGIC : Goes through each heat source in order, returns if the heat flux↙
supplied is sufficient.
1100     heatflux_left = heat_flux;
1101
1102     % (GJ / yr)           = (kt / yr)          * (g / kt) * (kJ / g)        * (GJ /↙
kJ)
1103     Q_combust_all_hydrogen = flowrates(HYDROGEN) * G_PER_KT * ENTHALPY_HYDROGEN *↙
GJ_PER_KJ;
1104
1105     if (~PSA_TOGGLE)
1106         % Hydrogen
1107         if (heatflux_left > Q_combust_all_hydrogen)
1108             combusted_fuel_flowrates(HYDROGEN) = flowrates(HYDROGEN);
1109             heatflux_left = heatflux_left - Q_combust_all_hydrogen;
1110         else
1111             % (kt / yr)                      = ((GJ)                ) * (KJ / GJ)↙
*
1112             combusted_fuel_flowrates(HYDROGEN) = (heatflux_left) * KJ_PER_GJ * ...
1113                 ... % (mol / KJ)        * (g / mol)       * (kt / g)
1114                 ( 1 / ENTHALPY_HYDROGEN) * MOLMASS_HYDROGEN * KT_PER_G;
```

```
1115             heatflux_left = 0;
1116             return
1117         end
1118     end
1119
1120     % (GJ / yr)           = (kt / yr)          * (g / kt) * (kJ / g)        * (GJ /↙
kJ)
1121     Q_combust_all_methane = flowrates(METHANE) * G_PER_KT * ENTHALPY_METHANE *↙
GJ_PER_KJ;
1122
1123     % Methane
1124     if (heatflux_left > Q_combust_all_methane)
1125         combusted_fuel_flowrates(METHANE) = flowrates(METHANE);
1126         heatflux_left = heatflux_left - Q_combust_all_methane;
1127     else
1128         % (kt / yr)                        = ((GJ)                 ) * (KJ / GJ) *
1129         combusted_fuel_flowrates(METHANE) = (heatflux_left) * KJ_PER_GJ * ...
1130             ... % (mol / KJ)        * (g / mol)       * (kt / g)
1131             ( 1 / ENTHALPY_METHANE) * MOLMASS_METHANE * KT_PER_G;
1132         heatflux_left = 0;
1133         return
1134     end
1135
1136     % (GJ / yr)           = (kt / yr)          * (g / kt) * (kJ / g)        * (GJ /↙
kJ)
1137     Q_combust_all_propane = flowrates(PROPANE) * G_PER_KT * ENTHALPY_PROPANE *↙
GJ_PER_KJ;
1138
1139     % Propane
1140     if (heatflux_left > Q_combust_all_propane)
1141         combusted_fuel_flowrates(PROPANE) = flowrates(PROPANE);
1142         heatflux_left = heatflux_left - Q_combust_all_propane;
1143     else
1144         % (kt / yr)                         = ((GJ)                 ) * (KJ / GJ) *
1145         combusted_fuel_flowrates(PROPANE) = (heatflux_left) * KJ_PER_GJ * ...
1146             ... % (mol / KJ)        * (g / mol)       * (kt / g)
1147             ( 1 / ENTHALPY_PROPANE) * MOLMASS_PROPANE * KT_PER_G;
1148         heatflux_left = 0;
1149         return
1150     end
1151
1152     % (GJ / yr)           = (kt / yr)          * (g / kt) * (kJ / g)        * (GJ /↙
kJ)
1153     Q_combust_all_butane = flowrates(BUTANE) * G_PER_KT * ENTHALPY_BUTANE *↙
GJ_PER_KJ;
1154
1155     % Butane
1156     if (heatflux_left > Q_combust_all_butane)
1157         combusted_fuel_flowrates(BUTANE) = flowrates(BUTANE);
1158         heatflux_left = heatflux_left - Q_combust_all_butane;
1159     else
1160         % (kt / yr)                        = ((GJ)                 ) * (KJ / GJ) *
1161         combusted_fuel_flowrates(BUTANE) = (heatflux_left) * KJ_PER_GJ * ...
1162             ... % (mol / KJ)        * (g / mol)       * (kt / g)
1163             ( 1 / ENTHALPY_BUTANE) * MOLMASS_BUTANE * KT_PER_G;
1164         heatflux_left = 0;
1165         return
1166     end
```

```
1167 end
1168
1169 %       GJ   =           (kta   , __          , bar      , C )
1170 function heat = heat_steam(F_steam, STEAM_CHOICE, P_reactor, T_reactor)
1171     global COST_RATES_STEAM;
1172     global STEAM_PRESSURE_COL STEAM_TEMP_COL COST_RATES_STEAM G_PER_KT ...
1173             MOLMASS_WATER BAR_PER_PSIA C_TO_K HEAT_CAPACITY_WATER GJ_PER_KJ;
1174
1175     P_steam = COST_RATES_STEAM(STEAM_CHOICE, STEAM_PRESSURE_COL); % [ psia ]
1176     T_steam = COST_RATES_STEAM(STEAM_CHOICE, STEAM_TEMP_COL);     % [ C ]
1177     P_steam = P_steam * BAR_PER_PSIA;
1178     T_steam = T_steam + C_TO_K;
1179     T_reactor = T_reactor + C_TO_K;
1180
1181     if (P_steam > P_reactor) % Adiabatic Expansion
1182         T_adibatic = (T_steam) * (P_reactor / P_steam);
1183         T_steam = T_adibatic;
1184     elseif (P_steam < P_reactor) % Compression
1185         W = compressor_work(T_reactor, P_steam, P_reactor);
1186         if ADD_COMPRESSOR_WORK_TO_STEAM_HEATFLUX
1187             heat = heat + W;
1188         end
1189         % I should add this to the heat flux probably ??
1190     end
1191
1192     % KJ = kta     * (G / KT) * (mol / g)         * (KJ / MOL K)        * (K - K)
1193     heat = F_steam * G_PER_KT * (1/MOLMASS_WATER) * HEAT_CAPACITY_WATER * (T_reactor↙
- T_steam);
1194     % GJ = KJ   * (KJ / GJ)
1195     heat = heat * GJ_PER_KJ;
1196
1197
1198     % Heat flux after temperture
1199
1200
1201
1202 end
1203
1204 function T_f = adiabatic_temp(T_0, P_0, P_f)
1205
1206     T_f = T_0 * ( P_0 / P_f);
1207 end
1208
1209 function W = compressor_work(T, P_0, P_f)
1210     R = 8.314;      % [ J / mol K]
1211
1212     W = - n * R * T * log(P_f / P_0);
1213
1214     % ?? THIS ALWAYS RETURNS 0 OR NULL, NOT IMPLEMENTED YET
1215
1216 end
1217
1218 % HELPER FUNCTIONS | TAXES_____________________________________________________
1219
1220 function cost = tax_C02(combusted_flowrates, F_natural_gas)
1221     global HYDROGEN METHANE ETHYLENE PROPANE BUTANE TAX_CO2_PER_MT;
1222     global MT_CO2_PER_KT_METHANE MT_CO2_PER_KT_PROPANE MT_CO2_PER_KT_BUTANE ...
1223     MT_CO2_PER_KT_NATURALGAS;
```

```
1224
1225     % Calculate the cost per kt (in tax) of each combusted fuel
1226     methane = combusted_flowrates(METHANE);
1227     propane = combusted_flowrates(PROPANE);
1228     butane = combusted_flowrates(BUTANE);
1229
1230     mt_c02 = 0;
1231     % kta  =  (MT)  + ( (kt fuel / yr) * (MT CO2 / KT FUEL) )
1232     mt_c02 = mt_c02 + methane * MT_CO2_PER_KT_METHANE;
1233     mt_c02 = mt_c02 + propane * MT_CO2_PER_KT_PROPANE;
1234     mt_c02 = mt_c02 + butane * MT_CO2_PER_KT_BUTANE;
1235     mt_c02 = mt_c02 + F_natural_gas * MT_CO2_PER_KT_NATURALGAS;
1236
1237     cost = mt_c02 * TAX_CO2_PER_MT;
1238 end
1239
1240 % HELPER FUNCTIONS | FUEL COSTS_____________________________________________
1241
1242 function cost = cost_natural_gas_fuel(heat_flux_remaining)
1243     global VALUE_NATGAS_FUEL
1244     % $ / yr = (GJ)            * ($ / GJ)
1245     cost = heat_flux_remaining * VALUE_NATGAS_FUEL;
1246 end
1247
1248 % HELPER FUNCTIONS | FUEL FLOWRATES___________________________________________
1249
1250 function F_natural_gas = natgas_combustion(heat_flux_remaining)
1251     global KJ_PER_GJ ENTHALPY_NAT_GAS KT_PER_G MOLMASS_NATGAS;
1252     % output should be in kta, input is in GJ
1253
1254     %       kt          GJ              * (kJ / GJ) * (mol / kJ) *          (g /↙
mol) *             (kt / g)
1255     F_natural_gas = heat_flux_remaining * KJ_PER_GJ * (1/ENTHALPY_NAT_GAS) *↙
(MOLMASS_NATGAS) * KT_PER_G;
1256
1257 end
1258
1259 % FUNCTIONS | REACTOR ODE SYSTEM_____________________________________________
1260
1261 function dFdV = reactionODEs(V, F, T, P, F_steam)
1262     global R_2 k1_f k1_r k2 k3 C_TO_K MOLMASS_METHANE MOLMASS_ETHANE↙
MOLMASS_ETHYLENE ...
1263         MOLMASS_PROPANE MOLMASS_HYDROGEN MOLMASS_BUTANE YR_PER_SEC G_PER_KT↙
SEC_PER_YR KT_PER_G
1264     % INPUT UNITS
1265     % V [ L ]
1266     % F [ kta ]
1267     % T [ Celcius ]
1268     % P [ bar ]
1269
1270     % Change the input units so that evrything is consistent
1271     % P = P * ATM_PER_BAR;
1272     T = T + C_TO_K;
1273
1274     % Product flow rate indicies
1275     HYDROGEN = 1;
1276     METHANE = 2;
1277     ETHYLENE = 3;
```

```
1278     PROPANE = 4;
1279     BUTANE = 5;
1280
1281     % Feed flow rate index
1282     ETHANE = 6;
1283
1284     F_tot = sum(F) + F_steam;
1285
1286
1287     % Hydrogen = A
1288     dFAdV = (k1_f(T) * ( (F(ETHANE) * P) / (F_tot * R_2 * T) )   ) - ...
1289             (k1_r(T) * ( F(ETHYLENE) * F(HYDROGEN) * P^2) ) / (F_tot * R_2 * T)^2;
1290
1291     % Methane = B
1292     dFBdV = (k2(T) * (F(ETHANE) * P)^2) / (F_tot * R_2 * T)^2;
1293
1294     % Ethylene = C
1295     dFCdV = (k1_f(T) * (F(ETHANE) * P / (F_tot * R_2 * T))) - ...
1296             (k1_r(T) * (F(ETHYLENE) * F(HYDROGEN) * P^2) / (F_tot * R_2 * T)^2) -↙
...
1297             (k3(T) * (F(ETHANE) * F(ETHYLENE) * P^2) / (F_tot * R_2 * T)^2);
1298
1299     % Propane = E
1300     dFEdV = k2(T) * (F(ETHANE) * P)^2 / (F_tot * R_2 * T)^2;
1301
1302     % Butane = F
1303     dFFdV = (k3(T) * (F(ETHANE) * F(ETHYLENE) * P^2)) / (F_tot * R_2 * T)^2;
1304
1305     % Ethane = D
1306     dFDdV = (-k1_f(T) * (F(ETHANE) * P / (F_tot * R_2 * T))) + ...
1307             (k1_r(T) * (F(ETHYLENE) * F(HYDROGEN) * P^2)/(F_tot * R_2 * T)^2) - ...
1308             (k2(T) * F(ETHANE)^2 * P^2 / (F_tot * R_2 * T)^2) - ...
1309             (k3(T) * F(ETHANE) * F(ETHYLENE) * P^2 / (F_tot * R_2 * T)^2);
1310
1311     T = T - C_TO_K;
1312
1313     dFdV = [dFAdV; dFBdV; dFCdV; dFEdV; dFFdV; dFDdV];
1314
1315 end
1316
1317 function cost = cost_reactor(V_plant_input)
1318     global FT_PER_METER STEAM_TO_FEED_RATIO
1319     FT_PER_METER = 3.28084;
1320     % ??? WHAT ARE THE UNITS OF TIME
1321     %
1322     pi = 3.14159;
1323     D = 0.05;                                   % [m]
1324     V_plant_max = pi * (0.025)^2 * 20;          %[m^3]
1325
1326     % Reactors have a max length, so calculate the number of full size reactors
1327     % and add it to the cost of the one non-max length reactor
1328
1329     cost = 0;
1330
1331     % Find the Cost of the max-sized reactors
1332     num_of_additional_reactors = int64(V_plant_input / V_plant_max);
1333     num_of_additional_reactors = double(num_of_additional_reactors);
1334
```

```
1335     V_plant = V_plant_max;
1336     factor_1 = 4.18;
1337     factor_2 = (V_plant / (pi * (D/2)^2) * FT_PER_METER)^0.82;
1338     factor_3 = (101.9 * D * FT_PER_METER)^1.066;
1339     factor_4 = (1800 / 280);
1340     cost_max_reactor = factor_1 * factor_2 * factor_3 * factor_4;
1341     cost = cost + num_of_additional_reactors * cost_max_reactor;
1342
1343
1344     % Find the cost of the non-max size reactor
1345     V_plant = V_plant_input - V_plant_max * num_of_additional_reactors;
1346     if V_plant < 0
1347         V_plant = 0;
1348     end
1349     factor_1 = 4.18;
1350     factor_2 = (V_plant / (pi * (D/2)^2) * FT_PER_METER)^0.82;
1351     factor_3 = (101.9 * D * FT_PER_METER)^1.066;
1352     factor_4 = (1800 / 280);
1353     cost = cost + factor_1 * factor_2 * factor_3 * factor_4;
1354
1355
1356 end
1357
1358 %        [$]  =                   ( kta   )
1359 function cost = cost_waste_stream(F_steam)
1360     global MOLMASS_WATER G_PER_KT YR_PER_SEC R_2 M3_PER_L T_SEPARATION ...
1361             P_SEPARATION SEC_PER_YR C_TO_K DENSITY_LIQ_WATER KG_PER_KT
1362
1363     % m^3 / s = (kt / yr) * (kg / kt)   * (m^3 / kg)  * (yr / s)
1364     q = F_steam * KG_PER_KT * (1 / DENSITY_LIQ_WATER) * YR_PER_SEC;
1365         % ?? Assume that all of the water out of the sep system is liquid
1366
1367     a = 0.001 + 2e-4*q^(-0.6);
1368         %Source: Uldrich and Vasudevan
1369     b=0.1;
1370         %Source: Uldrich and Vasudevan
1371     CEPCI = 820;
1372         %Source: Lecture slides
1373     C_f = 3.0;                       % [ $ / GJ ]
1374
1375     %$/m^3 waste water
1376     cost_waste_water = a*CEPCI + b*C_f;
1377
1378     % m^3 / s = (m^3 / s) * (s / yr)
1379     q = q * SEC_PER_YR;
1380     cost = cost_waste_water * q;
1381
1382 end
1383
1384 function cost = cost_separation_system(P_flowrates, F_steam, R_ethane)
1385     global MOLMASS_METHANE MOLMASS_HYDROGEN MOLMASS_ETHANE MOLMASS_ETHYLENE ...
1386          MOLMASS_PROPANE MOLMASS_BUTANE YR_PER_SEC
1387     global T_SEPARATION R PRESS_RXTR R ...
1388      MAX_OPEX MAX_TFCI MAX_CAPEX G_PER_KT MOLMASS_WATER
1389
1390     % Product flow rate indicies
1391     HYDROGEN = 1;
1392     METHANE = 2;
```

```
1393     ETHYLENE = 3;
1394     PROPANE = 4;
1395     BUTANE = 5;
1396
1397     % Feed flow rate index
1398     ETHANE = 6;
1399
1400     % SEPARATION_EFFICIENCY_FACTOR = 30;
1401     T = T_SEPARATION; % [ K ]
1402
1403     %Using compositions from ASPEN
1404     %Component mole flow rate out of rxtr over total mole flow rate out of reactor
1405     % Mol fractions out of the reactoor
1406
1407     % (mol / s) = (kt / yr) * (g / kt) * (mol / g) * (yr / s)
1408     P_flowrates(METHANE) = P_flowrates(METHANE) * G_PER_KT * (1/MOLMASS_METHANE) *↙
YR_PER_SEC;
1409     P_flowrates(HYDROGEN) = P_flowrates(HYDROGEN) * G_PER_KT * (1/MOLMASS_HYDROGEN)↙
* YR_PER_SEC;
1410     R_ethane = R_ethane * G_PER_KT * (1/MOLMASS_ETHANE) * YR_PER_SEC;
1411     P_flowrates(ETHYLENE) = P_flowrates(ETHYLENE) * G_PER_KT * (1/MOLMASS_ETHYLENE)↙
* YR_PER_SEC;
1412     P_flowrates(PROPANE) = P_flowrates(PROPANE) * G_PER_KT * (1/MOLMASS_PROPANE) *↙
YR_PER_SEC;
1413     P_flowrates(BUTANE) = P_flowrates(BUTANE) * G_PER_KT * (1/MOLMASS_BUTANE) *↙
YR_PER_SEC; % Add this line for butane
1414     F_steam = F_steam * G_PER_KT * (1/MOLMASS_WATER) * YR_PER_SEC;
1415
1416     %CONVERT TO_MOLES
1417
1418     P_tot = sum(P_flowrates(HYDROGEN:BUTANE)) + F_steam + R_ethane;
1419
1420     z_methane = P_flowrates(METHANE) / P_tot;
1421     z_hydrogen = P_flowrates(HYDROGEN) / P_tot;
1422     z_ethane = R_ethane / P_tot;
1423     z_ethylene = P_flowrates(ETHYLENE) / P_tot;
1424     z_propane = P_flowrates(PROPANE) / P_tot;
1425     z_butane = P_flowrates(BUTANE) / P_tot;
1426     z_water = F_steam / P_tot;
1427
1428     %Mol fractions leaving each separation system (refer to Isa's drawing in GN)
1429     % leaving sep 1
1430     x_water = 1;
1431
1432     % leaving sep 4
1433     x_ethane = 1;
1434     x_ethylene = 1;
1435
1436     % leaving sep 2
1437     x_butane = 0.0003;
1438     x_propane = 1 - x_butane;
1439
1440     % leaving sep 5 (PSA)
1441     x_methane = 4.03293090303065e-004;
1442     x_hydrogen = 1 - x_methane;
1443         % ?? How should I implement the PSA toggle switch on this
1444
1445     %Pressures of PSA system [bar]
```

```
1446     P_in = PRESS_RXTR;
1447     P_H2 = 10;              % [ bar ]
1448     P_ME = 1;               % [ bar ]
1449         % These outlet pressures are constant for PSA system. DONT change
1450 
1451     %Using flow rates from ASPEN [NOTE: FOR MATLAB USE THE VALUES FROM THE
1452     %SOLN_TABLE. WE USED THESE AS EXPECTED COSTS)
1453 
1454     % Flowrates of each exiting stream from the sep system
1455 
1456     F_water = F_steam;                                      % mol/s
1457     F_LPG = P_flowrates(BUTANE) + P_flowrates(PROPANE); % (mol / s)
1458     F_ethylene = P_flowrates(ETHYLENE);                     % (mol / s)
1459     F_ethane = R_ethane;                            % (mol / s)
1460     F_H2 = P_flowrates(HYDROGEN);                           % (mol / s)
1461     F_ME = P_flowrates(METHANE);                            % (mol / s);
1462 
1463     %(J/s) =    (mol/s) * (J/mol K) * (T)
1464     W_min_Sep_System = F_water*R*T*log(x_water/z_water) + ...
1465                     F_LPG*R*T*log(x_propane/z_propane + ...
1466                                   x_butane/z_butane) + ...
1467                     F_ethylene*R*T*log(x_ethylene/z_ethylene) + ...
1468                     F_ethane*R*T*log(x_ethane/z_ethane) + ...
1469                     R*T*( ...
1470                         F_H2*log(P_H2/P_in)+ ...
1471                         F_H2*log(x_hydrogen/z_hydrogen) +...
1472                         F_ME*log(x_methane/z_methane) +...
1473                         F_ME*log(P_ME/P_in)...
1474                         );
1475 
1476 
1477     lamdba_min = 20;
1478     lambda_max = 50;
1479     cost_energy = 3;        % ( $ / GJ )
1480 
1481     if MAX_OPEX
1482     %($/yr)              =   (J/s)     * (GJ/J) * (Work Efficiency) *($/GJ)* (s/yr)
1483         opex =  W_min_Sep_System*1e-9 * lambda_max * cost_energy * 30.24e6;
1484     else
1485         opex = W_min_Sep_System*1e-9 * lamdba_min * cost_energy * 30.24e6;
1486     end
1487 
1488     if MAX_CAPEX
1489     %($)                 = ($/W)    (Efficiency) * (J/s)
1490         capex = 1 * lambda_max * W_min_Sep_System;
1491     else
1492         capex = 0.5 * lamdba_min * W_min_Sep_System;
1493     end
1494 
1495     cost = 2.5 * capex ;
1496 
1497 end
1498 
1499 
1500 function cf = get_npv(npv)
1501     global YEARS_IN_OPERATION
1502     % USER_INPUTS | All inputs are in units of $MM
1503         % npv.mainProductRevenue = value_ethylene(P_ethylene);
```

```
1504         % npv.byProductRevenue = value_h2_chem(P_hydrogen - combusted_hydrogen);
1505         % npv.rawMaterialsCost = value_ethane(F_fresh_ethane);
1506         % npv.utilitiesCost = cost_steam(F_steam, COST_RATES_STEAM(STEAM_CHOICE,↙
STEAM_COST_COL));
1507         % npv.CO2sustainabilityCharge = tax_C02(combusted_fuel_flow_rates,↙
F_natural_gas);
1508         % npv.conversion = conversion(i);
1509         % npv.isbl = cost_rxt_vec + cost_separation_system(P_flowrates, F_steam,↙
R_ethane);
1510
1511     WORKING_CAP_PERCENT_OF_FCI = 0.15;      % [ % in decimal ]
1512     STARTUP_COST_PERCENT_OF_FCI = 0.10;     % [ % in decimal ]
1513     LENGTH_CONSTRUCTION_TABLE = 6;
1514     LAST_ROW_CONSTRUCTION = LENGTH_CONSTRUCTION_TABLE;
1515     YEARS_OF_CONSTUCTION = 3;
1516
1517     % Revenues & Production Costs
1518     npv.consummablesCost = 0;
1519     npv.VCOP = npv.rawMaterialsCost + npv.utilitiesCost + ...
1520                 npv.consummablesCost + npv.CO2sustainabilityCharge - ...
1521                                                         npv.byProductRevenue;
1522     npv.salaryAndOverhead = 0;
1523     npv.maintenenace = 0;
1524     npv.interest = 15;
1525     npv.AGS = (npv.mainProductRevenue + npv.byProductRevenue)*0.05;     % ~5%↙
revenue
1526     npv.FCOP = npv.salaryAndOverhead + npv.maintenenace +...
1527                           npv.AGS + npv.interest;
1528
1529     % Capital Costs
1530     npv.OSBLcapitalCost = npv.ISBLcapitalCost * 0.40;
1531     npv.contingency = (npv.ISBLcapitalCost + npv.OSBLcapitalCost) * 0.25;
1532     npv.indirectCost = (npv.ISBLcapitalCost + npv.OSBLcapitalCost + ...
1533                                                     npv.contingency) * 0.30;
1534     npv.totalFixedCapitalCost = npv.ISBLcapitalCost + ...
1535                                 npv.OSBLcapitalCost + ...
1536                                 npv.indirectCost + ...
1537                                 npv.contingency;
1538
1539     npv.workingCapital = npv.totalFixedCapitalCost * WORKING_CAP_PERCENT_OF_FCI;
1540     npv.startupCost = npv.totalFixedCapitalCost * STARTUP_COST_PERCENT_OF_FCI;
1541     npv.land = 10;
1542     npv.totalCapitalInvestment = npv.totalFixedCapitalCost + ...
1543                                     npv.workingCapital + ...
1544                                     npv.startupCost + ...
1545                                     npv.land;
1546     % Economic Assumptions
1547     npv.discountRate = 0.15;        % [ % in decimal ]
1548     npv.taxRate = 0.27;             % [ % in decimal ]
1549     npv.salvageValue = 0.05;        % [ % in decimal ]
1550
1551     % CONSTRUCTION SCHEDULE INDICIES
1552     YEAR = 1;
1553     FC = 2;
1554     WC = 3;
1555     SU = 4;
1556     FCOP = 5;
1557     VCOP = 6;
```

```
1558     construction_matrix = zeros(LENGTH_CONSTRUCTION_TABLE + 1, VCOP);
1559
1560     % Generate the construction schedule matrix
1561     for yr = 0:LENGTH_CONSTRUCTION_TABLE
1562         row = yr + 1;
1563         if yr > 0 && yr < 4
1564             construction_matrix(row, FC) = 0.33;
1565         end
1566         if yr == 3
1567             construction_matrix(row, WC) = 1.00;
1568             construction_matrix(row, SU) = 1.00;
1569         end
1570         if yr > 3 && yr <= 6
1571             construction_matrix(row, FCOP) = 1.00;
1572             construction_matrix(row, VCOP) = 1.00;
1573         end
1574     end
1575
1576     % NPV COLUMN INDICIES
1577     YEAR = 1;
1578     CAPITAL_EXPENSE = 2;
1579     REVENUE = 3;
1580     COM = 4;
1581     GROSS_PROFIT = 5;
1582     DEPRECIATION = 6;
1583     TAXABLE_INC = 7;
1584     TAXES_PAID = 8;
1585     CASH_FLOW = 9;
1586     CUM_CASH_FLOW = 10;
1587     PV_OF_CF = 11;
1588     CUM_PV_OF_CF = 12;
1589     NPV = 13;
1590     cash_flow_matrix = zeros(YEARS_IN_OPERATION + 1, NPV);
1591     LAST_ROW_CASHFLOW = YEARS_IN_OPERATION + 1;
1592
1593
1594     for yr = 0:YEARS_IN_OPERATION
1595         row = yr + 1;
1596         cash_flow_matrix(row, YEAR) = yr;
1597
1598         % Capital Expenses Column
1599         if yr == 0
1600             cash_flow_matrix(row, CAPITAL_EXPENSE) = npv.land;
1601         elseif yr >= 1 && yr <= 5
1602             cash_flow_matrix(row, CAPITAL_EXPENSE) ...
1603                 = npv.totalFixedCapitalCost * construction_matrix(row,FC) + ...
1604                   npv.workingCapital * construction_matrix(row, WC) + ...
1605                   npv.startupCost * construction_matrix(row, SU) ;
1606         elseif yr == YEARS_IN_OPERATION
1607             cash_flow_matrix(row, CAPITAL_EXPENSE) = - npv.salvageValue * npv.↙
totalFixedCapitalCost;
1608         else
1609             cash_flow_matrix(row, CAPITAL_EXPENSE) ...
1610                 = npv.totalFixedCapitalCost * construction_matrix↙
(LAST_ROW_CONSTRUCTION,FC) + ...
1611                   npv.workingCapital * construction_matrix(LAST_ROW_CONSTRUCTION,↙
WC) + ...
1612                   npv.startupCost * construction_matrix(LAST_ROW_CONSTRUCTION, SU) ;
```

```
1613         end
1614
1615         % Revenue Column
1616         if yr <= LENGTH_CONSTRUCTION_TABLE % ??
1617             cash_flow_matrix(row, REVENUE) = npv.mainProductRevenue *↙
construction_matrix(row, VCOP);
1618         else
1619             cash_flow_matrix(row, REVENUE) = npv.mainProductRevenue *↙
construction_matrix(LAST_ROW_CONSTRUCTION, VCOP);
1620         end
1621
1622         % COM Column
1623         if yr <= LENGTH_CONSTRUCTION_TABLE
1624             cash_flow_matrix(row, COM) = npv.VCOP * construction_matrix(row, VCOP) +↙
...
1625                                          npv.FCOP * construction_matrix(row,↙
FCOP);
1626         else
1627             cash_flow_matrix(row, COM) = npv.VCOP * construction_matrix↙
(LAST_ROW_CONSTRUCTION, VCOP) + ...
1628                                      npv.FCOP * construction_matrix↙
(LAST_ROW_CONSTRUCTION, FCOP);
1629         end
1630
1631         % Gross Profit
1632         cash_flow_matrix(row, GROSS_PROFIT) = cash_flow_matrix(row,REVENUE) -↙
cash_flow_matrix(row, COM);
1633
1634         % Depreciation
1635         if yr >= YEARS_OF_CONSTUCTION
1636             cash_flow_matrix(row, DEPRECIATION) = 0.1*(npv.totalFixedCapitalCost +↙
npv.startupCost - 0.05*npv.totalFixedCapitalCost);
1637         end
1638
1639         % Taxable Inc
1640         if yr >= YEARS_OF_CONSTUCTION
1641             cash_flow_matrix(row, TAXABLE_INC) = cash_flow_matrix(row, GROSS_PROFIT)↙
- cash_flow_matrix(row,DEPRECIATION);
1642         end
1643
1644         % Taxes Paid
1645         if yr >= YEARS_OF_CONSTUCTION
1646             cash_flow_matrix(row, TAXES_PAID) = cash_flow_matrix(row, TAXABLE_INC) *↙
npv.taxRate;
1647         end
1648
1649         % Cash Flow
1650         cash_flow_matrix(row, CASH_FLOW) = -cash_flow_matrix(row, CAPITAL_EXPENSE) +↙
...
1651                 ( cash_flow_matrix(row,REVENUE) ...
1652                     - cash_flow_matrix(row, COM) ...
1653                     - cash_flow_matrix(row, DEPRECIATION) ...
1654                 ) * ( 1 - npv.taxRate) + cash_flow_matrix(row, DEPRECIATION);
1655
1656         % Cummulative Cash Flow
1657         cash_flow_matrix(row, CUM_CASH_FLOW) = sum( cash_flow_matrix( 1 : row,↙
CASH_FLOW) );
1658
```

```
1659         % PV of CF
1660         cash_flow_matrix(row, PV_OF_CF) = cash_flow_matrix(row, CASH_FLOW) / ( 1 +↙
npv.discountRate)^yr;
1661
1662         % Cummulative PV of CF
1663         cash_flow_matrix(row , CUM_PV_OF_CF) = sum( cash_flow_matrix(1:row,↙
PV_OF_CF) );
1664
1665         % NPV
1666         if row > 1
1667             cash_flow_matrix(row , NPV) = cash_flow_matrix(row - 1, NPV) +↙
cash_flow_matrix(row, PV_OF_CF);
1668         else
1669             cash_flow_matrix(row, NPV) = cash_flow_matrix(row, PV_OF_CF);
1670         end
1671     end
1672
1673     % RETURN
1674     % cash_flow_matrix
1675 %   [cf_matrix, lifetime_npv] = [cash_flow_matrix, cash_flow_matrix↙
(LAST_ROW_CASHFLOW, NPV)];
1676     cf.matrix = cash_flow_matrix;
1677     cf.lifetime_npv = cash_flow_matrix(LAST_ROW_CASHFLOW, NPV);
1678     % lifetime_npv = cash_flow_matrix(LAST_ROW_CASHFLOW, NPV);
1679 end
1680
1681
1682
1683 function installedCost = calculate_installed_cost(Q)
1684     global MILLIONBTU_PER_GJ MILLIONBTU_PER_GJ YR_PER_HR HR_PER_YR
1685
1686     Q = Q * MILLIONBTU_PER_GJ * YR_PER_HR;
1687
1688     % Constants
1689     M_and_S = 1800; % Marshall and Swift index
1690     base_cost = 5.52 * 10^3;
1691
1692     % Purchased cost calculation
1693     % F_c = F_d + F_m + F_p;
1694     F_c = 1.1;
1695
1696     % Installed cost calculation
1697     installedCost = (M_and_S / 280) * (base_cost * Q^0.85 * (1.27 + F_c));
1698
1699     installedCost = installedCost;
1700 end
1701
1702
1703
1704 function void = plot_conversion_fxns(fxns)
1705     global T_OVERRIDE P_OVERRIDE STEAM_MR_OVERRIDE
1706     global M3_PER_L
1707     % USER INPUT
1708         % fxns.conversion = conversion;
1709         % fxns.V_plant = V_plant;
1710         % fxns.select_1 = select_1;
1711         % fxns.select_2 = select_2;
1712         % fxns.npv = npv;
```

```
1713        % fxns.recycle = F_soln_ODE( : , ETHANE);
1714        % fxns.freshFeedRawMaterials = fxns.F_fresh_ethane + fxns.F_steam;
1715        % fxns.productionRateRxnProducts = F_soln_ODE( : , HYDROGEN : BUTANE);
1716        % fxns.F_rxtr_in_total = fxns.F_fresh_ethane + fxns.recycle + fxns.F_steam;
1717        % fxns.F_sep = sum(F_soln_ODE(: , HYDROGEN : ETHANE), 2) + fxns.F_steam;
1718        % fxns.x_hydrogen_sep = F_soln_ODE( : , HYDROGEN) ./ fxns.F_sep;
1719        % fxns.x_methane_sep = F_soln_ODE( : , METHANE) ./ fxns.F_sep;
1720        % fxns.x_ethylene = F_soln_ODE( : , ETHYLENE) ./ fxns.F_sep;
1721        % fxns.x_propane_sep = F_soln_ODE( : , PROPANE) ./ fxns.F_sep;
1722        % fxns.x_butane_sep = F_soln_ODE( : , BUTANE) ./ fxns.F_sep;
1723        % fxns.x_ethane_sep = F_soln_ODE( : , ETHANE) ./ fxns.F_sep;
1724        % fxns.x_water_sep = fxns.F_steam ./ fxns.F_sep;
1725
1726    x = fxns.conversion;
1727
1728    % Selectivity 1 & 2
1729    hold on
1730    figure;
1731    tit = "Selectivity 1";
1732    xlab = "\chi";
1733    ylab = "S_1";
1734    plot(x, fxns.select_1);
1735    title(tit);
1736    xlabel(xlab);
1737    ylabel(ylab);
1738    hold off
1739
1740    hold on
1741    figure;
1742    tit = "Selectivity 2";
1743    xlab = "\chi";
1744    ylab = "S_2";
1745    plot(x, fxns.select_2);
1746    title(tit);
1747    xlabel(xlab);
1748    ylabel(ylab);
1749    hold off
1750
1751    % Reactor Volume
1752    hold on
1753    figure;
1754    tit = "Reactor Volume";
1755    xlab = "\chi";
1756    ylab = "V_{Reactor} [ m^3 ]";
1757    plot(x, fxns.V_plant .* M3_PER_L);
1758    title(tit);
1759    xlabel(xlab);
1760    ylabel(ylab);
1761    hold off
1762
1763    % Fresh feed flow rate of raw materials
1764    hold on
1765    figure;
1766    tit = "Fresh Feed of of Raw Materials into the Reactor [ kta ]";
1767    xlab = "\chi";
1768    ylab = "F_{FreshFeedRawMaterials}";
1769    for i = 1 : 15
1770        fxns.freshFeedRawMaterials(i,1) = 0;
```

```
1771     end
1772     plot(x, fxns.freshFeedRawMaterials);
1773     % tit = tit + " " + sprintf("(%3.0f C %3.1f Bar %0.2f Steam MR)", T_OVERRIDE,↙
P_OVERRIDE, STEAM_MR_OVERRIDE);
1774     title(tit);
1775     xlabel(xlab);
1776     ylabel(ylab);
1777     hold off
1778
1779     % Production Rate of all reaction products leaving the reactor
1780     hold on
1781     figure;
1782     tit = "Production Rate [ kta ]";
1783     xlab = "\chi";
1784     ylab = "Production Rate" ;
1785     plot(x, fxns.productionRateRxnProducts);
1786     legend("Hydrogen", "Methane", "Ethylene", "Propane", "Butane")
1787     % tit = tit + " " + sprintf("(%3.0f C %3.1f Bar %0.2f Steam MR)", T_OVERRIDE,↙
P_OVERRIDE, STEAM_MR_OVERRIDE);
1788     title(tit);
1789     xlabel(xlab);
1790     ylabel(ylab);
1791     hold off
1792
1793     % Recycle flow rate of LR
1794     hold on
1795     figure;
1796     tit = "Recycle flow rate of Ethane [ kta ]";
1797     xlab = "\chi";
1798     ylab = "R_{Ethane}" ;
1799     for i = 1 : 15
1800         fxns.recycle(i,1) = 0;
1801     end
1802     plot(x, fxns.recycle);
1803     % tit = tit + " " + sprintf("(%3.0f C %3.1f Bar %0.2f Steam MR)", T_OVERRIDE,↙
P_OVERRIDE, STEAM_MR_OVERRIDE);
1804     title(tit);
1805     xlabel(xlab);
1806     ylabel(ylab);
1807     hold off
1808
1809     % Total flow rate to reactor
1810     hold on
1811     figure;
1812     tit = "Total flow rate to reactor [ kta ]";
1813     xlab = "\chi";
1814     ylab = "F_{RxtrIn}" ;
1815     for i = 1 : 15
1816         fxns.F_rxtr_in_total(i,1) = 0;
1817     end
1818     plot(x, fxns.F_rxtr_in_total);
1819     % tit = tit + " " + sprintf("(%3.0f C %3.1f Bar %0.2f Steam MR)", T_OVERRIDE,↙
P_OVERRIDE, STEAM_MR_OVERRIDE);
1820     title(tit);
1821     xlabel(xlab);
1822     ylabel(ylab);
1823     hold off
1824
```

```
1825     % Total flow rate to the separation system
1826     hold on
1827     figure;
1828     tit = "Total flow rate to the separation system [ kta ]";
1829     xlab = "\chi";
1830     ylab = "F_{separation system}" ;
1831     for i = 1 : 15
1832         fxns.F_sep(i,1) = 0;
1833     end
1834     plot(x, fxns.F_sep);
1835     % tit = tit + " " + sprintf("(%3.0f C %3.1f Bar %0.2f Steam MR)", T_OVERRIDE,↙
P_OVERRIDE, STEAM_MR_OVERRIDE);
1836     title(tit);
1837     xlabel(xlab);
1838     ylabel(ylab);
1839     hold off
1840
1841     % Mol fraction of each component entering the separation system
1842     hold on
1843     figure;
1844     tit = "Mol fraction of each component entering the separation system [ kta ]";
1845     xlab = "\chi";
1846     ylab = "F_{i}" ;
1847     % for i = 1 : 15
1848     %   fxns.F_sep(i,:) = 0;
1849     % end
1850     plot(x, [fxns.x_hydrogen_sep, fxns.x_methane_sep, fxns.x_ethylene_sep, fxns.↙
x_propane_sep, fxns.x_ethane_sep, fxns.x_water_sep]);
1851     legend("Hydrogen", "Methane", "Ethylene", "Propane", "Butane", "Ethane",↙
"Water")
1852     % tit = tit + " " + sprintf("(%3.0f C %3.1f Bar %0.2f Steam MR)", T_OVERRIDE,↙
P_OVERRIDE, STEAM_MR_OVERRIDE);
1853     title(tit);
1854     xlabel(xlab);
1855     ylabel(ylab);
1856     hold off
1857
1858     % NPV
1859     hold on
1860     figure;
1861     tit = "NPV [ $ MM ]";
1862     xlab = "\chi";
1863     ylab = "NPV [ $ MM ]" ;
1864     tit = tit + " " + sprintf("(%3.0f C %3.1f Bar %0.2f Steam MR)", T_OVERRIDE,↙
P_OVERRIDE, STEAM_MR_OVERRIDE);
1865     i = 1;
1866     fxns.npv(fxns.npv(:, 1) < 0, 1) = 0;
1867     fxns.npv(isnan(fxns.npv(:, 1)), 1) = 0;
1868
1869     % while (fxns.npv(i , : ) < 0)
1870         % fxns.npv(i, : ) = 0;
1871         % i = i + 1;
1872     % end
1873     plot(x, fxns.npv)
1874     % legend("Hydrogen", "Methane", "Ethylene", "Propane", "Butane", "Ethane",↙
"Water")
1875     title(tit);
1876     xlabel(xlab);
```

```
1877     ylabel(ylab);
1878     hold off
1879
1880
1881
1882     % NPV (T, P, MR) | Varying T
1883 %   hold on
1884 %   figure;
1885 %   tit = "NPV [ $ MM ]";
1886 %   xlab = "\chi";
1887 %   ylab = "NPV [ $ MM ]" ;
1888 %   tit = tit + " " + sprintf("(%3.0f C %3.1f Bar %0.2f Steam MR)", T_OVERRIDE,↙
P_OVERRIDE, STEAM_MR_OVERRIDE);
1889 %
1890 %   y = [];
1891 %   for i = 1:length(fxns.npv_T_P_MR(: , 1, 1 ))
1892 %       temp =fxns.npv_T_P_MR( i , 1, 1) ;
1893 %       y = [ y , fxns.npv_T_P_MR( i , 1, 1) ];
1894 %   end
1895 %
1896 % %     % Choose a colormap
1897 % %     cmap = jet(size(y, 2)); % Using 'jet' colormap; adjust the number of colors↙
based on the number of columns in y
1898 % %
1899 % %     for i = 1:size(y, 2) % Iterate through each column (dataset) in y
1900 % % %       temp =
1901 % %         plot(x, cell2mat(y(:,i)), 'Color', cmap(i,:), 'LineWidth', 2);
1902 % %     end
1903 %
1904 %   plot(x, y)
1905 %   title(tit);
1906 %   xlabel(xlab);
1907 %   ylabel(ylab);
1908 %   hold off
1909
1910
1911     % Sep cost vs conversion
1912     hold on
1913     figure;
1914     tit = "Separation Cost [ $ MM ]";
1915     xlab = "\chi";
1916     ylab = "Cost [ $ MM ]" ;
1917     % tit = tit + " " + sprintf("(%3.0f C %3.1f Bar %0.2f Steam MR)", T_OVERRIDE,↙
P_OVERRIDE, STEAM_MR_OVERRIDE);
1918     % i = 1;
1919     % fxns.npv(fxns.npv(:, 1) < 0, 1) = 0;
1920     % fxns.npv(isnan(fxns.npv(:, 1)), 1) = 0;
1921
1922     % while (fxns.npv(i , : ) < 0)
1923         % fxns.npv(i, : ) = 0;
1924         % i = i + 1;
1925     % end
1926     fxns.separationCosts(fxns.separationCosts(:, 1) > 10^9, 1) = 0;
1927     plot(x, fxns.separationCosts)
1928     % legend("Hydrogen", "Methane", "Ethylene", "Propane", "Butane", "Ethane",↙
"Water")
1929     title(tit);
1930     xlabel(xlab);
```

```
1931      ylabel(ylab);
1932      hold off
1933
1934
1935      % Return
1936      void = NaN;
1937  end
1938
```