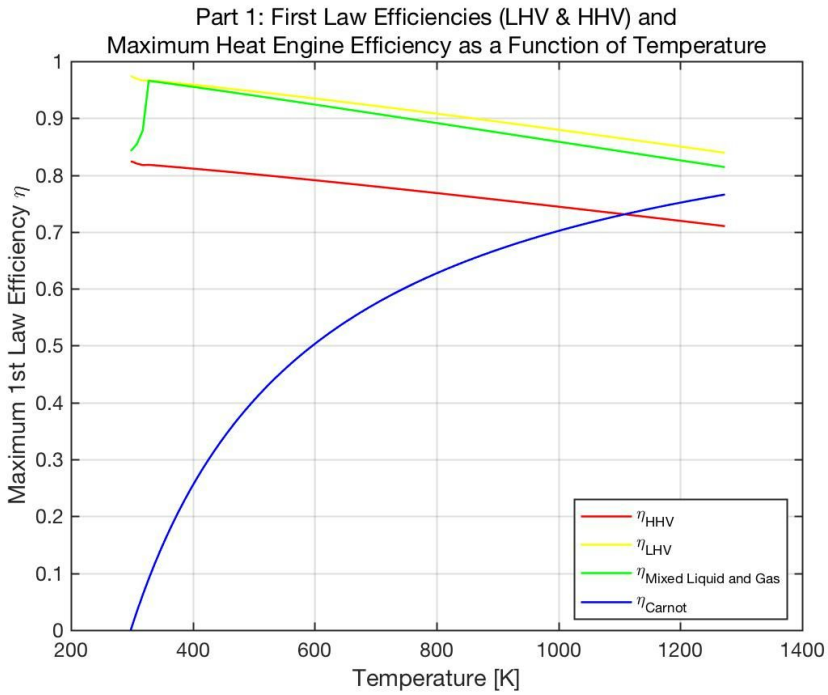
ME 140 | Advanced Thermal Systems

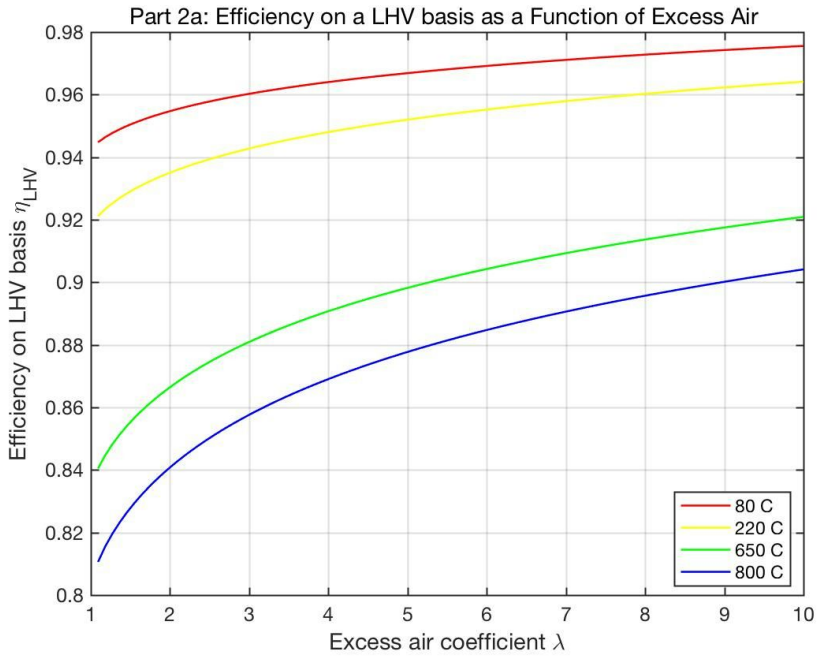
Project #4: PEM Fuel Cell Analysis

Jon Renslo, Frankie Willcox, Emily Bohl, Kendall Fagan, and Natasha Berk

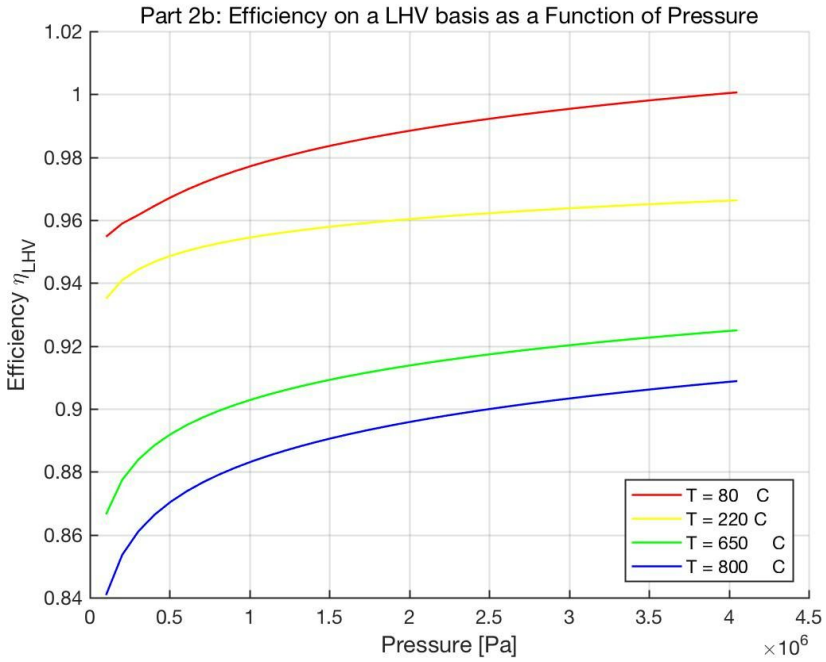
Part 1: First Law Efficiencies and Maximum Heat Engine Efficiency



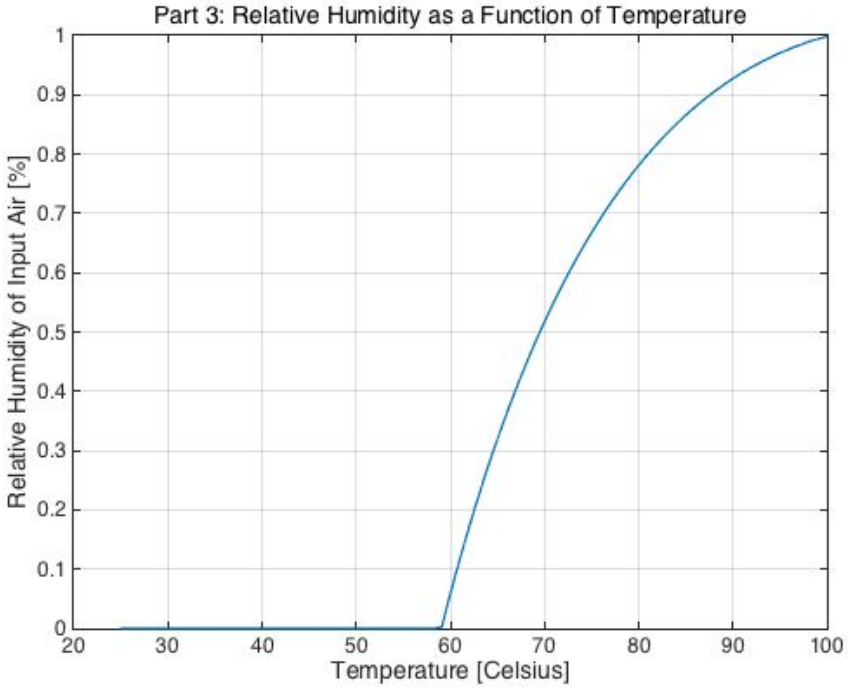
Part 2: Varying Lambda: Maximum Cell Efficiency



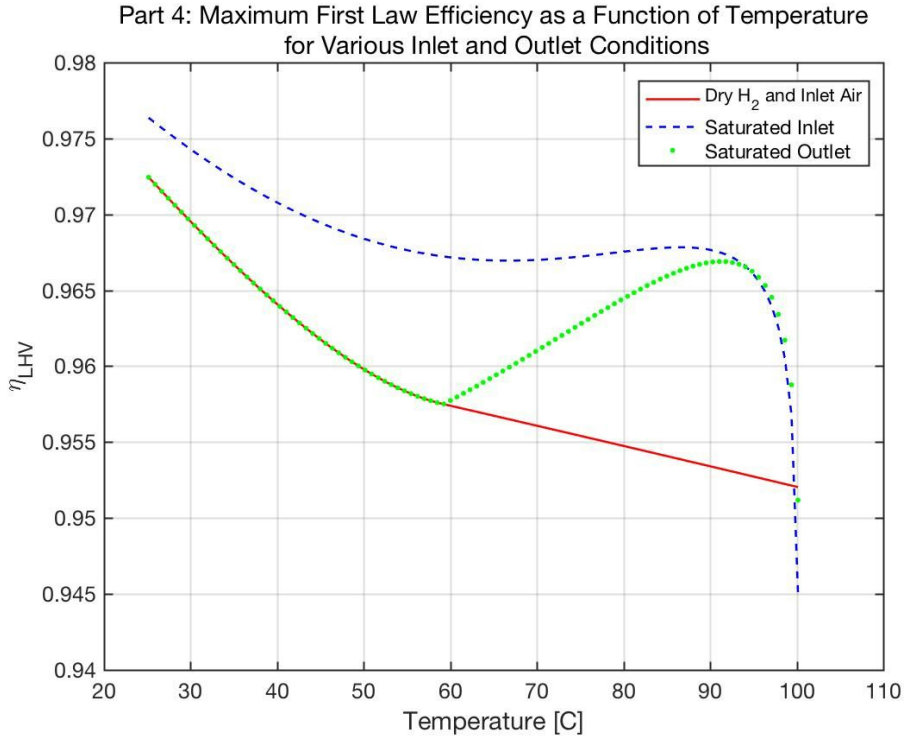
Part 2: Varying Pressure: Maximum Cell Efficiency



Part 3: Relative Humidities as a Function of Temperature



<u>Part 4</u>: Maximum First-Law Efficiencies at Three Inlet Conditions



Individual Reflections

- Kendall
  I spent 12 hours on this project. I'd like to understand the effect of the entropy of liquid water better. We suspect that this is a main cause for the error in our Part 2 graph.
- Frankie
  I spent 16 hours on this project. I want to understand why our code for Part 2b works for the three cases below 80 degrees C but does not work for 80 C.
- Emily
  I spent 14 hours on this project: about 6 hours on part 1, 6 hours on part 2, and 2 hours on part 3. I would like to better understand how the fuel cell works in practice, so I am excited for the lab on Friday! I think part 1 was useful for my understanding of the material.
- Jon
  I spent 20 hours on this project. I want to understand the phase change more clearly and what is actually happening at each thermodynamic state. I want to better understand how the fuel cell actual works. At one point our efficiency goes above one. I would like to understand how this can occur.
- Natasha
  I spent 13 hours on this project. I thought understanding how a PEM fuel cell works was interesting. I found Part 4 to be somewhat repetitive and not very useful. I would like a more thorough understanding of the chemistry/phase changes.

Appendix: Matlab Scripts

```
% ME140 PROJECT 4: FUEL CELLS
% ----------------------------
% FILENAME: me140_project4.m
% Jon Renslo, Emily Bohl, Frankie Willcox, Natasha Berk, Kendall Fagan
% 4/15/16 - Created Jon Renslo

close all; clear; clc;

% Constants
G_TO_KG = 10^-3;
KPA_TO_PA = 10^3;
KJ_TO_J = 10^3;
C_TO_K = 273.15;

N_TO_O = 79/21;        % Engineering Air Molar Mass Ratio of Nitrogen to Oxygen

% Molar Masses
MM_h = 1.00794;
MM_o = 15.9994;
MM_n = 14.0067;
MM_h2o = 2*MM_h + MM_o;
MM_air = 28.97;

% ------------------------------------------------------
% Part 1 & 2: Efficiency of PEM Fuel Cells Found 3 Ways
% ---------- then, varying lambda & presssure
% ------------------------------------------------------
% ASSUME: isothermal, isobaric i.e. reversible
% USE: First- Law Effiency, eta = (-m_reactants*dg_react)/(mfuel*HV) where HV = LHV or HHV
% SOURCE: LEC 8, SLIDE 13
npts = 100;
HHV_h2 = 141.8*10^6;                    % J/kg,  Higher Heating Value
LHV_h2 = 120.0*10^6;                    % J/kg,  Lower Heating Value
T = linspace(25+C_TO_K,1000+C_TO_K,npts);
lambda = 4;                             % Equivalence Ratio(ASSUME: 100% excess air)
Patm = 101.3*KPA_TO_PA;                 % Pa,     Preact = Pprod = Patm

eta = zeros(size(T));
pctVap = zeros(size(T));
delG = zeros(size(T));

for i = 1:length(T)                     %loop temperature (cols)
    [eta(i), pctVap(i),delG(i),~] = PEMstoich(lambda,T(i),Patm);
end
%PEMstoich assumes per mol of h2, 1mol h2 burned
mass_h2 = 1*(MM_h*2)*G_TO_KG;
eta_HHV = -delG / (HHV_h2 * mass_h2);
eta_LHV = -delG / (LHV_h2 * mass_h2);

eta_carnot = carnotEff(T,T(1));      % ASSUME: Tcold = 25 degrees C

figure(1);
plot(T,eta_HHV,'b--', T,eta_LHV,'m--',T,eta,'g-', T,eta_carnot,'c');
legend('\eta_{HHV}','\eta_{LHV}','\eta_{Mixed Liquid and Gas}','\eta_{Carnot}', 'Location', 'Best');
xlabel('Temperature [K]');
ylabel('Maximum 1st Law Efficiency');
title('Part 1: First Law Efficiencies and Maximum Heat Engine Efficiency');
plotfixer();
grid on
```

```
% PART 2a (varying lambda)
T_C = [80 220 650 800];
T = T_C + C_TO_K;
lambda = linspace(1,10,npts);       % (Comment back in for Part 2)
Patm = 101.3*KPA_TO_PA;             % Pa,     Preact = Pprod = Patm

for Ti = 1:length(T)
    for li = 1:length(lambda)
        [etaLambda(li,Ti), pctVapLambda(li,Ti) ,delGLambda(li,Ti),~] ...
            = PEMstoich(lambda(li),T(Ti),Patm);
    end
end
mass_h2 = 1* (MM_h*2)*G_TO_KG;
delH_LHV = LHV_h2 * mass_h2;
etaLambda_LHV = -delGLambda/delH_LHV;

%%part2.1 plot%%

figure(2);
plot(lambda,etaLambda_LHV);
legend('80C','220C','650C','800C','Location','Best');
xlabel('Excess air coefficient \lambda');
ylabel('Efficiency on LHV basis \eta');
title('Part 2: Varying Lambda: Maximum Cell Efficiency')
plotfixer
grid on;

spec = Spec();
spec.mol_air = 5;

% % UNCOMMENT FOR PART 2b (varying Patm)
T_C = [80 220 650 800];
T = T_C + C_TO_K;
lambda = 2;                         % Equivalence Ratio(ASSUME: 100% excess air)
Patm = linspace(101.3*KPA_TO_PA,4053*KPA_TO_PA,npts);

for Ti = 1:length(T)
    for pi = 1:length(Patm)
        [etaPres(pi,Ti), pctVapPres(pi,Ti) ,delGPres(pi,Ti),~] ...
            = PEMstoich(lambda,T(Ti),Patm(pi));
    end
end

etaPres_LHV = -delGPres/delH_LHV;
figure(3);
plot(Patm/101325,etaPres_LHV(:,1),Patm/101325,etaPres_LHV(:,2),...
     Patm/101325,etaPres_LHV(:,3),Patm/101325,etaPres_LHV(:,4));
legend('80C','220C','650C','800C','Location','Best');
xlabel('Pressure - Atm');
ylabel('Efficiency on LHV basis \eta');
title('Part 2: Varying Pressure: Maximum Cell Efficiency')
grid on

%% Part 3
% what humidity necesary for inlet air to obtain saturated exit?
% below certain temp, condensate forms, so add no water.
% plot inlet air humidity vs T 25-100C

% questions:
% must we take into account the diffusion thru membrane? -> don't need to
```

```
% worry about gas diffusion through MEA membrane
lambda = 2; % as before
Patm = 101.3*KPA_TO_PA;             % Pa,     Preact = Pprod = Patm
Ptotal = Patm;
% find psat at exit based on temp,
T = linspace(25,100,npts);
T = T + C_TO_K;
psat = PsatW(T);

mol_air = zeros(size(T));
mol_o2_react = zeros(size(T));
mol_n2 = zeros(size(T));
for i = 1:length(T)
    [~,~,~,tempSpecs] = PEMstoich(lambda, T(i), Ptotal);
    mol_air(i) = tempSpecs.mol_air;
    mol_o2_react(i) = tempSpecs.mol_o2_react;
    mol_n2(i) = tempSpecs.mol_n2;
end

% find mole fraction of water in products
y_h2o = psat./Ptotal; %Assume Pv = Psat
beta = (4.26 .* y_h2o)./ (1 - y_h2o);
% if less than what is formed, add the difference to dry air reagent
alpha = beta - 1;
%if condensation is forming just from what is formed, don't add any
%humidity to reactants
alpha(alpha < 0) = 0;
y_h2o_react = alpha./(mol_o2_react + mol_n2 + alpha);
Pv_react = Ptotal.*y_h2o_react;
Pv_react(Pv_react>psat) = psat(Pv_react>psat); % if Pv > psat, Pv = psat
hum_rel = Pv_react./psat;

% plot relative humidity
figure(4);
plot(T - C_TO_K,hum_rel)
xlabel('Temperature [Celsius]');
ylabel('Relative Humidity of Input Air [%]');
title('Part 3: Relative Humidity as a Function of Temperature')
plotfixer();
grid on;

%% Part 4
% (1) part 1 plot, (2) part 1 plot except inlet humidity = 100%, (3) part 3
% plot

% Part 4 - 1

delG = zeros(size(T));
for i = 1:length(T) %loop temperature for new T
    [~,~,delG(i),~] = PEMstoich(lambda,T(i),Patm);
end
%PEMstoich assumes per mol of h2, 1mol h2 burned
eta_LHV = -delG / (LHV_h2 * mass_h2);

% Part 4 - 2
% T 25-100 C
% P atm
% lamdba = 2
Patm = 101.3e3;
lambda = 2;
Psat = PsatW(T);
```

```
y_h2o_react = Psat / Patm;
% assume 1 mol h2
mol_air = lambda*4.76/2;
alpha_2 = mol_air * (Psat) ./ (Patm - Psat); %alternatively y / 1-y;

delG_3  = zeros(size(T));
delG_2 = zeros(size(T));
for i = 1:length(T)
    [~, ~, delG_2(i), ~] = ...
        PEMstoich(lambda,T(i),Patm,alpha_2(i));
end

eta_2  = -delG_2 ./delH_LHV;

for i = 1:length(T)
    [~, ~, delG_3(i), ~] = ...
        PEMstoich(lambda,T(i),Patm,alpha(i));
end

eta_3  = -delG_3 ./delH_LHV;

figure(5);
plot(T-273,eta_LHV);
hold on;
plot(T-273,eta_2,'--');
plot(T-273,eta_3,'.');
legend('Dry H2 and Inlet Air','Saturated Inlet', 'Saturated Outlet','Location','best');
xlabel('Temperature [C]');
ylabel('\eta_{LHV}');
title('Part 4: Comparing Max-1st-Law Efficiency in Varied Conditions');
plotfixer;
grid on;


%carnotEff.m
%4-22-16 - Created Jon Renslo
function eta = carnotEff(Th,Tc)

    eta = 1 - Tc./Th;
end


% energyF.m
% 4/15/16 - Created Jon Renslo
% 4/22/16 - Adapted from hMix.m
% Returns enthalpy and Gibbs function of a specified number of moles of a selected species at
% a selected temperature.
% Also returns the mole-specific heat at that temperature

function out = energyF(T,P,species,moles)
    % H in Joules, or Joules per mol if no moles number specified
    % P in Pa
    % cpbar in J/mol-K
    % cp in J/kg-K

    P0 = 101.3e3;
    R = 8.314; %J/mol-K


    if(~exist('moles','var')) moles = 1; end
    T0 = 273 + 25; %standard conditions 25C
```

```
    [co2, h2ovap, h2o, n2, o2, air, airConst,h2] = deal(1,2,3,4,5,6,7,8);

    fit{co2} =    [22.26, 5.981*10^-2,   -3.501 *10^-5, 7.469*10^-9  ];
    fit{h2ovap} = [32.24, 0.1923*10^-2,  1.055*10^-5,   -3.595 *10^-9];
    fit{h2o} =    [75.42271 0 0 0]; %calculated from 4.1855 at 15C
    fit{n2} =     [28.90, -0.1517*10^-2, 0.8081*10^-5,  -2.873*10^-9 ];
    fit{o2} =     [25.48, 1.520*10^-2,   -0.7155*10^-5, 1.312*10^-9  ];
    fit{air} =    [28.11, 0.1967*10^-2,  0.4802*10^-5,  -1.966*10^-9];
    fit{airConst} = [27.8715 0 0 0];
    fit{h2} =     [29.11, -.1916e-2,     0.4003e-5     -0.8704e-9];


    hf{co2} = -393520;      % J/mol
    hf{h2ovap} = -241820;
    hf{h2o} = -285830;      % for liquid water
    hf{n2} = 0;
    hf{o2} = 0;
    hf{air} = 0;
    hf{airConst} = 0;
    hf{h2} = 0;

    sf{co2} = 213.8; %J/(mol * K)
    sf{h2ovap} = 188.83;
    sf{h2o} = 69.92; % for liquid water
    sf{n2} = 191.61;
    sf{o2} = 205.04;
    %sf{air} = sf{n2}*3.76/4.76 + sf{o2}/4.76; %** cannot do
    %sf{airConst} = sf{n2}*3.76/4.76 + sf{o2}/4.76; %** not in table
    sf{h2} = 130.68;


%     gf{co2} = -394360;      % J/mol
%     gf{h2ovap} = -228590;
%     gf{h2o} = -237180;      % for liquid water
%     gf{n2} = 0;
%     gf{o2} = 0;
%     gf{air} = 0;
%     gf{airConst} = 0;
%     gf{h2} = 0;

    m{co2} = 44; %g/mol
    m{h2ovap} = 18.02;
    m{h2o} = 18.02;
    m{n2} = 28;
    m{o2} = 32;
    m{air} = 28.98;
    m{airConst} = 28.98;
    m{h2} = 2.02;

    switch lower(species)
        case 'co2'
            i = co2;
        case 'h2o'
            i = h2o;
        case'h2ovap'
            i = h2ovap;
        case 'n2'
            i = n2;
        case 'o2'
            i = o2;
        case 'air'
```

```
            i = air;
        case 'airconst'
            i = airConst;
        case 'h2'
            i = h2;
        otherwise
            disp 'input a supported species';
    end
    fits = num2cell(fit{i});
    [a, b, c, d] = deal(fits{:});
%     I1 = R*(a*(T0 - T) + b/2*(T0^2 - T^2) + c/3*(T0^3 - T^3) + d/4*(T0^4 - T^4) + e/5*(T0^5 - T^5));
%     cp_ave = I1/(T0 - T);
%     I2 = a*log(T0/T) + b*(T0 - T) + c/2*(T0^2 - T^2) + d/3*(T0^3 - T^3) + e/4*(T0^4 - T^4);

    gPerKg = 1000;
    out.cpbar = (a + b*T + c*T.^2 + d*T.^3); % J/mol-k, Mol Specific Heat
    out.cp = out.cpbar/m{i}*gPerKg;          % J/g-k,   Specific Heat

    % integrate cp from t0 to t in j/mol kelvin
    delH = (a*(T - T0) + b*(T.^2 - T0.^2)/2 + c*(T.^3 - T0.^3)/3 + d*(T.^4 - T0.^4)/4);
    intCpbarOnT = a*log(T./T0) + b*(T - T0) + c*(T.^2 - T0.^2)/2 + d*(T.^3 - T0.^3)/3;
    delS = intCpbarOnT - R *log(P/P0);

    out.S = (sf{i} + delS).*moles;           % Entropy
    out.H = (hf{i} + delH).*moles;           % Enthalpy
    out.G = out.H - T.*out.S; % Gibbs Free Energy
    %**double check this calculation

end


%gEng.m , wrapper for energyF
% 4-22-16 created by Jon Renslo
function G = gEng(T,P,spec,mol)
%         for i = 1:length(T)
%             if length(P) == 1
%                 j = 1;
%             else
%                 j = i;
%             end
%             if length(mol) == 1
%                 k = 1;
%             else
%                 k = i;
%             end
%
%             if(nargin == 3)     % Note: nargin returns number of function input arguments
%                 mol = 1;
%             end
%             a = energyF(T(i),P(j),spec,mol(k));
%             G(i) = a.G;
%         end
    a = energyF(T,P,spec,mol);
    G = a.G;
end

%hEng.m , wrapper for energyF
% 4-22-16 created by Jon Renslo
function H = hEng(T,spec,mol)
% if(nargin == 2)
%     mol = 1;
```

```
% end
    a = energyF(T,1e5,spec,mol); %pressure does not affect H
    H = a.H;
end

% PEMstoich.m
% 4-26-16 Created Jon Renslo
function [eta, pctVap,delG, specs] = PEMstoich(lambda,T,Ptotal,alpha)
% Calculates the stoichiometry for a PEM fuel cell, dry air and dry h2
% Returns the efficiency and the % of water vapor in the products by mass

% do we want to return a mixture vector also?
if(~exist('alpha','var'))
    alpha = 0;
end

% all return values per mol of fuel burned (assuming 1 mol here)
N_TO_O = 79/21;        % Engineering Air Molar Mass Ratio of Nitrogen to Oxygen
specs = Spec(); %class initialization
mol_h2 = 1;
mol_air = (1+N_TO_O)*lambda/2*mol_h2;
mol_o2_react = mol_air/(1+N_TO_O);

mol_n2 = mol_air*N_TO_O/(1+N_TO_O);
mol_o2_prod = 0.5*(lambda-1).*mol_h2;
%double check o2prod? should be *mol_h2?

mol_h2o = mol_h2 + alpha;

beta = mol_h2o;               % ASSUME: all vapor
% Ptotal = Patm; from before restructuring
Psat = PsatW(T);
Pv_guess = Ptotal*(beta./(beta + 0.5.*(lambda-1) +0.5.*lambda.*N_TO_O ));

mol_total_react = mol_o2_react + mol_n2 + alpha;
y_o2_react = mol_o2_react /mol_total_react;
y_n2_react = mol_n2       /mol_total_react;
y_h2o_react = alpha       /mol_total_react;

% because membrane separates h2 from air, partial pressures are separate


if Pv_guess < Psat
    % All H2O is vapor (beta = 1)
    mol_h2ovap = beta;
    mol_h2oliq = 0;

else % i = 1-10
    % Some H2O is vapor, some liquid (beta not = 1)
    % LET: Pv = Psat, solve for beta
    Pv_h2o = Psat;
    y_h2o = Pv_h2o./Ptotal; %Assume Pv = Psat
    beta = (4.26 .* y_h2o)./ (1 - y_h2o);
    %beta = 1;
    mol_h2ovap = beta; %  = beta
    mol_h2oliq = mol_h2o - mol_h2ovap;
end

pctVap = mol_h2ovap./(mol_h2o);

% With the moles of liquid and gas water products, calculate mole fractions
```

```
% and deltaG and deltaH
mol_total_prod  = mol_o2_prod + mol_n2 + mol_h2ovap;
y_h2ovap  = mol_h2ovap  ./ mol_total_prod;
y_o2_prod = mol_o2_prod ./ mol_total_prod;
y_n2_prod = mol_n2      ./ mol_total_prod;

greact = gEng(T,Ptotal,'h2',mol_h2) ...
    + gEng(T,Ptotal .* y_o2_react,'o2',mol_o2_react) ...
    + gEng(T,Ptotal .* y_n2_react,'n2',mol_n2);
if(alpha ~= 0 )
    greact = greact + gEng(T,Ptotal .* y_h2o_react,'h2ovap',alpha);
end

gprod = ...
      gEng(T, Ptotal.*y_h2ovap,   'h2ovap', mol_h2ovap)...
    + gEng(T, Ptotal,             'h2o',    mol_h2oliq)...
    + gEng(T, Ptotal.*y_o2_prod,       'o2',     mol_o2_prod)...
    + gEng(T, Ptotal.*y_n2_prod,       'n2',     mol_n2);

delG = gprod - greact;

hprod = ...
      hEng(T,'h2ovap', mol_h2ovap)...
    + hEng(T,'h2o',    mol_h2oliq)...
    + hEng(T,'o2',     mol_o2_prod)...
    + hEng(T,'n2',     mol_n2);
hreact = ...
      hEng(T,'h2',     mol_h2)...
    + hEng(T,'o2',     mol_o2_react)...
    + hEng(T,'n2',     mol_n2);
if(alpha ~= 0 )
    hreact = hreact + hEng(T,'h2ovap',alpha);
end

dh = hprod - hreact;

eta = delG ./ dh;
specs.mol_air =         mol_air;
specs.mol_o2_react =         mol_o2_react;
specs.mol_n2 =         mol_n2;
% TODO update Spec to accomodate inlet water? (PLEASE CHECK KENDALL)

end


% PsatW.m
% Finds saturated pressure of water at given temperature.
% 4-22-16 - Created Jon Renslo
function [psat] = PsatW(T)
    psat = exp(-1.2914*10^8./T.^3 +8.2048*10^5./T.^2 -6522.8./T +25.5887); % Pa, Saturated Pressure
end

function [cp,cv,gamma] = sp_heats(T)
%Works for matrices
% Returns Joules
a = 28.11;
b = 0.1967E-2;
c = 0.4802E-5;
d = -1.966E-9;
molar_mass_air = .02897;
R = 287;                        % [J/kg*K]
```

```
P = [d,c,b,a];

cp = polyval(P,T);
cp = cp ./ molar_mass_air; %convert from KJ/kmol-K to J/kg-K
cv = cp - (R); %R converted to J/kg-K
gamma = cp./cv;

end


classdef Spec
    %SPEC Summary of this class goes here
    %   Detailed explanation goes here

    properties
        mol_air;
        mol_o2_react;
        mol_n2;
    end

    methods
    end

end
```