Table of Contents

HW 1 - 302

Joshua Oates

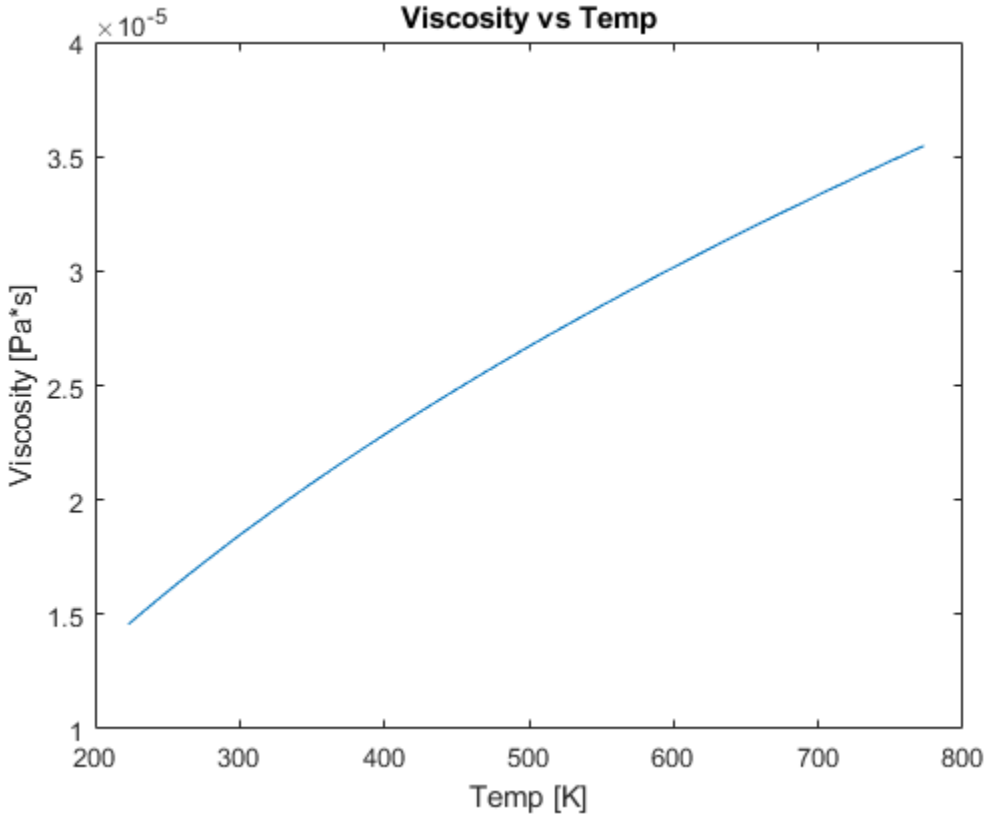
```
close all;
clear all;
clc
```

FF1

```
clear all;
disp("-----FF1-----")


muRef = 1.716e-5;
TRef = 273.15;
S = 110.4;


mu = @(T) muRef*(((T+S).^-1).*(TRef+S)).*(T/TRef).^1.5;

T = -50:1:500;
T=T+TRef;
mu = mu(T);

figure
plot(T,mu);
xlabel("Temp [K]")
ylabel("Viscosity [Pa*s]")
title("Viscosity vs Temp")

disp("Viscosity in a microscopic perspective is essentially the pull of fluid
 particles against each other so that the layers of fluid want to 'stick'
 together and transfer some shear force perpendicularly between layers. From
 a macroscopic perspective, it is essentially the fluids resistance to flowing
 past a surface or itself and represents a fluids ablility to transfer shear
 forces.")
```

```
disp("Cryogenic wind tunnels make use of low temperatures to lower the
 viscosity of the fluid they are testing with. This has the advantage of
 requiring less energy to accellerate the fluid to a high speed.")
disp("assumptions:")
disp("muRef: "+string(muRef)+"[Pa.s]")
disp("TRef: "+string(TRef)+"[K]")
disp("S: "+string(S)+"[K]")
```

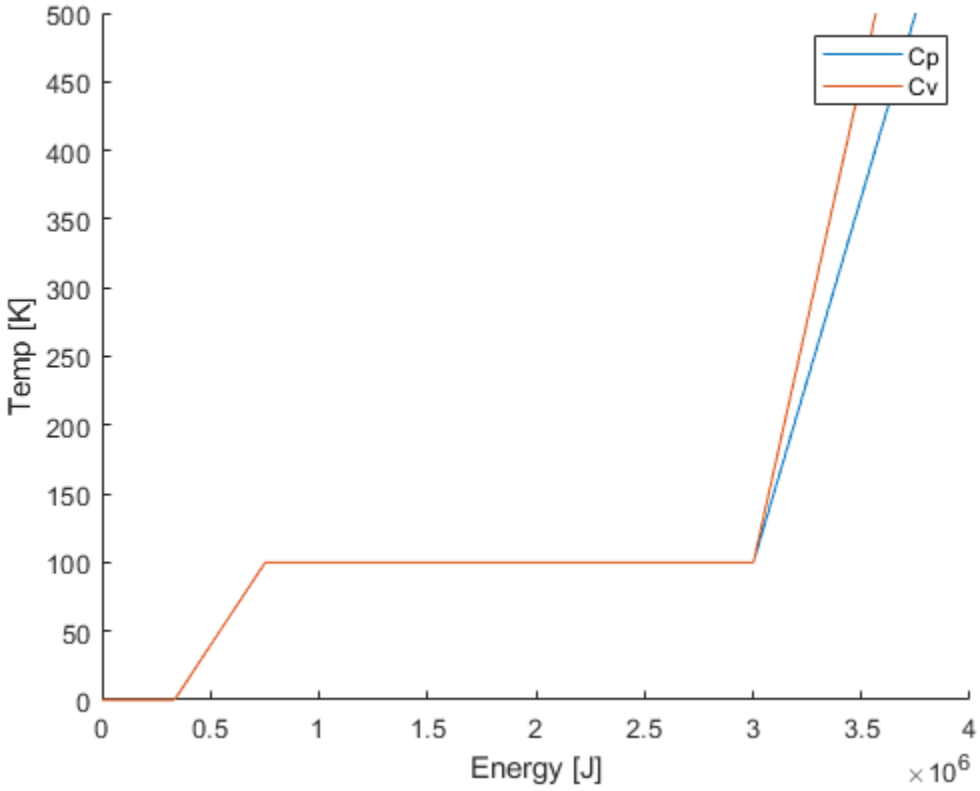
```
-----FF1-----
Viscosity in a microscopic perspective is essentially the pull of fluid
 particles against each other so that the layers of fluid want to 'stick'
 together and transfer some shear force perpendicularly between layers. From
 a macroscopic perspective, it is essentially the fluids resistance to flowing
 past a surface or itself and represents a fluids ablility to transfer shear
 forces.
Cryogenic wind tunnels make use of low temperatures to lower the viscosity
 of the fluid they are testing with. This has the advantage of requiring less
 energy to accellerate the fluid to a high speed.
assumptions:
muRef: 1.716e-05[Pa.s]
TRef: 273.15[K]
S: 110.4[K]
```



FF2

```
clear all;
```

```
disp("-----FF2-----")
m = 1; % kg
HLF = 334; % heat of latent fusion J/g
HLF = HLF*1000; % J/kg
HLV = 2.25e6; % heat of latent vaporization at 1 atm J/kg
C_water = 4.18; % J/g.K
C_water = C_water*1000; % J/kg.K
Cp_steam = 1.87; % kJ/kg.K
Cp_steam = Cp_steam*1000; % J/kg.K
Cv_steam = 1.4108; % kJ/kg.K
Cv_steam = Cv_steam*1000; % J/kg.K

E_Cp = [0,HLF,HLF+C_water*100,HLF+C_water*100+HLV,HLF+C_water*100+HLV
+Cp_steam*400].*m; % J
E_Cv = [0,HLF,HLF+C_water*100,HLF+C_water*100+HLV,HLF+C_water*100+HLV
+Cv_steam*400].*m;
T = [0,0,100,100,500];

figure
hold on
plot(E_Cp,T)
plot(E_Cv,T)
xlabel("Energy [J]")
ylabel("Temp [K]")
legend("Cp","Cv")


% E_Cp = E_Cp/1000; % KJ
disp("E-tot, Cp: "+string(E_Cp(5))+"[J]")
disp("E-tot, Cv: "+string(E_Cv(5))+"[J]")
disp("assumptions:")
disp("heat of latent fusion: "+string(HLF)+"[J/kg]")
disp("heat of latent vaporization: "+string(HLV)+"[J/kg]")
disp("C_water: "+string(C_water)+"[J/kg.K]")
disp("Cp_steam: "+string(Cp_steam)+"[J/kg.K]")
disp("Cv_steam: "+string(Cv_steam)+"[J/kg.K]")

-----FF2-----
E-tot, Cp: 3750000[J]
E-tot, Cv: 3566320[J]
assumptions:
heat of latent fusion: 334000[J/kg]
heat of latent vaporization: 2250000[J/kg]
C_water: 4180[J/kg.K]
Cp_steam: 1870[J/kg.K]
Cv_steam: 1410.8[J/kg.K]
```



DA1

```
clear all;
disp("-----DA1-----")
% F c U a rho mu
% L M T
%    [L M T]

M = [[2 1 -2];... %M
     [1 1 -2];... %F
     [1 0 0];...  %c
     [1 0 -1];... %U
     [1 0 -1];... %a
     [-3 1 0];... %rho
     [-1 1 -1]];   %mu

M = M';

pi = joshBuckPiTheory(M,["M","F", "c", "U", "a", "rho", "mu"]);

disp("I found the following pi's algorithmically")
disp("pi 1: "+string(pi(1)))
disp("pi 2: "+string(pi(2)))
disp("pi 3: "+string(pi(3)))
disp("pi 4: "+string(pi(4)))
```

```
disp("The values of pi 1 and pi 2 can be converted into the numbers")

pi(1) = pi(1)^-1; % coeff moment
pi(2) = pi(2)^-1; % mach

disp("pi 1: "+string(pi(1)))
disp("pi 2: "+string(pi(2)))
disp("which are the coeff moment and mach number.")
disp("using all pi's except 2 M and F can be solved. This reduction in number
 of paramenters from 7 to 4 will make testing and modeling much easier for the
 experiementers.")
```

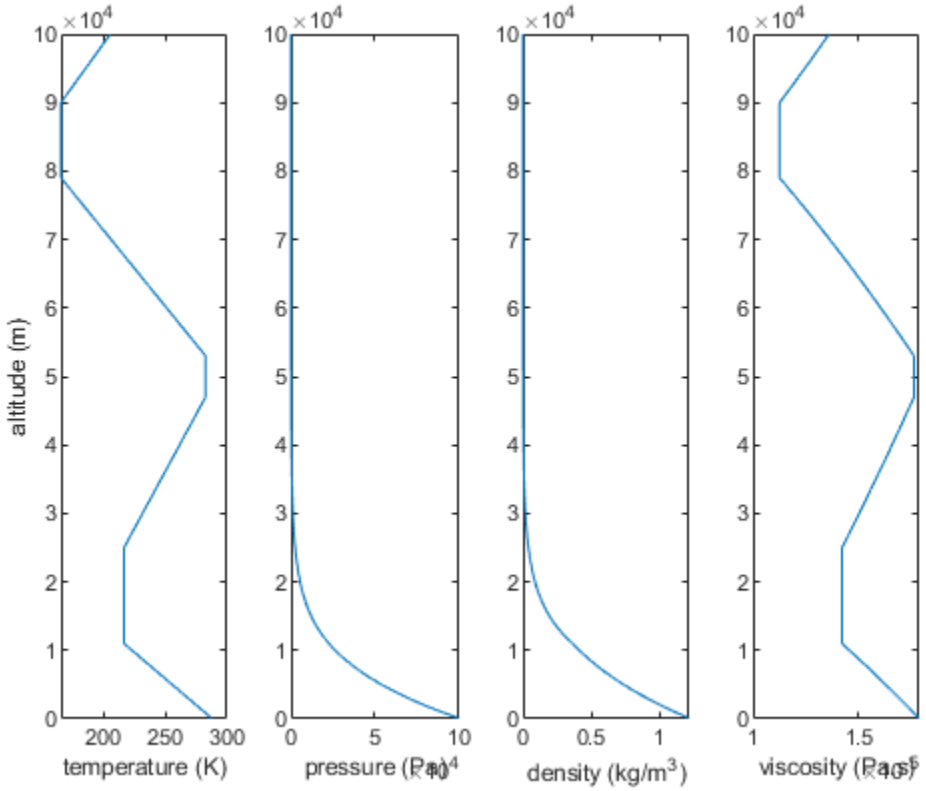
```
-----DA1-----
I found the following pi's algorithmically
pi 1: (F*c)/M
pi 2: a/U
pi 3: (M^2*U^2*rho)/F^3
pi 4: (M*U*mu)/F^2
The values of pi 1 and pi 2 can be converted into the numbers
pi 1: M/(F*c)
pi 2: U/a
which are the coeff moment and mach number.
using all pi's except 2 M and F can be solved. This reduction in number of
 paramenters from 7 to 4 will make testing and modeling much easier for the
 experiementers.
```

FS1

```
clear all;
disp("-----FS1-----")

for i=1:1000 % test iterator for 100 different altitudes
    hM(i)=i*100;
    [TM(i),PM(i),rhoM(i),muM(i)]=stdAtmOatesJoshua(hM(i));
end
figure
hold;
subplot(1,4,1); %set subplot settings
plot(TM,hM); %plot T
ylabel("altitude (m)"); %label altitude
xlabel("temperature (K)"); %label temperature
subplot(1,4,2);
plot(PM,hM);
xlabel("pressure (Pa)");
subplot(1,4,3);
plot(rhoM,hM);
xlabel("density (kg/m^3)");
subplot(1,4,4);
plot(muM,hM);
xlabel("viscosity (Pa.s)");
```

```
disp("We assume that temperatrure is either changing with a constant slope as
 a function of h or that it is not changing between certain altitudes. This
 leads to a peicewise graph of all atmpospheric variables. This appraoch could
 easily be extended to any important altitude but these fomulations will not
 work for long since the stdAtm model is designed only for up to 100km. When
 the air gets very thin, we start to care more about Knudsen number than than
 pressure.")
disp("assumptions:")
disp("stdAtm")
```

```
-----FS1-----
Current plot held
We assume that temperatrure is either changing with a constant slope as a
 function of h or that it is not changing between certain altitudes. This
 leads to a peicewise graph of all atmpospheric variables. This appraoch could
 easily be extended to any important altitude but these fomulations will not
 work for long since the stdAtm model is designed only for up to 100km. When
 the air gets very thin, we start to care more about Knudsen number than than
 pressure.
assumptions:
stdAtm
```



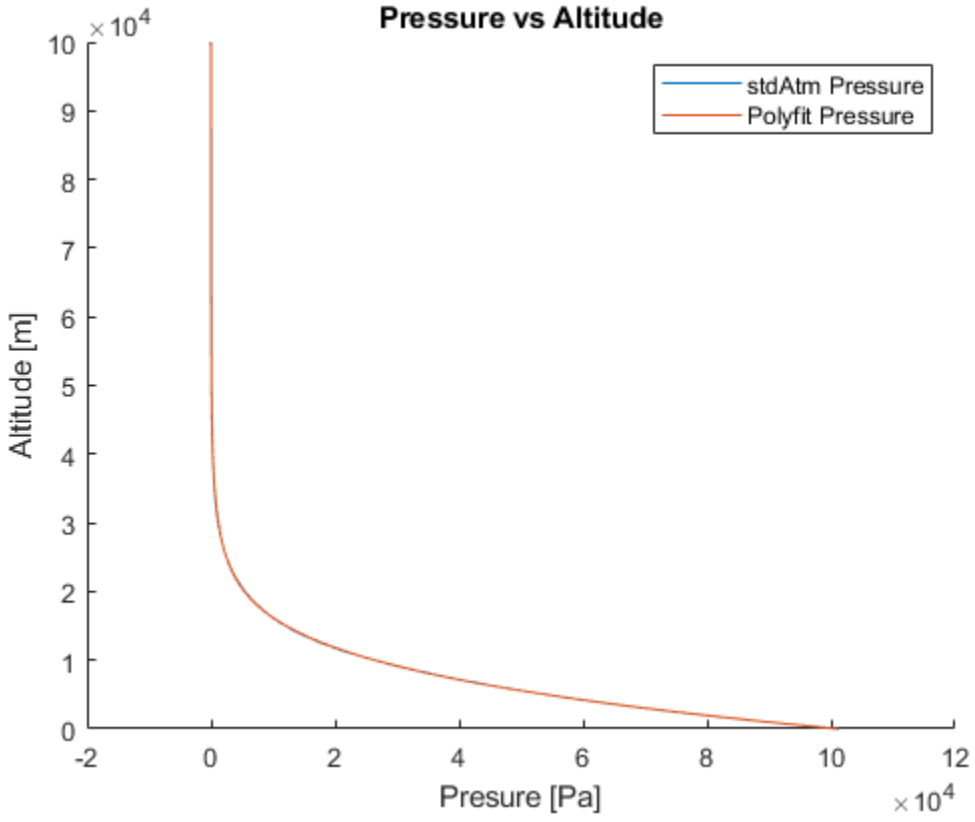
FS2

space elevator

```
clear all;

disp("-----FS2-----")
L = 100; % km
L = L*1000; % m
d = 2; % m
A = L*2; % m^2

degree = 10;
h = 1:L;
for i=1:length(h) % test iterator for 100 different
    % altitudes
    [~,P1(i)]=stdAtmOatesJoshua(h(i));
end

poly = polyfit(h,P1,degree);
P2 = polyval(poly,h);


polyFun1 = @(h) polyval(poly,h);
polyFun2 = @(h) polyval([poly,0],h);

hFun = @(h) h;
polyi = polyint(poly);

F = polyFun2(0) - polyFun2(L);
F = F*2;

Cp = integral(polyFun2,0,L)/integral(polyFun1,0,L);
Cm = L/2;
M = (Cm-Cp)*F;

figure
hold on
plot(P1,h)
plot(P2,h)
xlabel("Presure [Pa]")
ylabel("Altitude [m]")
title("Pressure vs Altitude")
legend("stdAtm Pressure","Polyfit Pressure")

disp("The center of pressure (Cp) is at and altitude of: "+string(Cp)+" m.")
disp("The moment is: "+string(M)+" N.m.")
disp("assumptions:")
disp("stdAtm")
disp("Only evaluate forces on one half of the elevator, if evaluated on both
 side there is no net force or moment.")
```

```
-----FS2-----
Warning: Polynomial is badly conditioned. Add points with distinct X values,
reduce the degree of the polynomial, or try centering and scaling as described
in HELP POLYFIT.
The center of pressure (Cp) is at and altitude of: 6808.868 m.
The moment is: 586255884955.1541 N.m.
```

assumptions:
stdAtm
Only evaluate forces on one half of the elevator, if evaluated on both side there is no net force or moment.



FS3

```
disp("-----FS3-----")
syms P1 P2 P3 V1 V2 h1 h2 theta Pd hm
assume([P1 P2 P3 V1 V2 h1 h2 theta Pd hm],'real')
assumeAlso([P1 P2 P3 V1 V2 h1 h2 theta hm]>0)
assumeAlso(P1 == P2 + P3)
h2 = sind(theta)*hm;
rho_w = 997; % kg/m^3
rho_a = 1.225; % kg/m^3
g = 9.81; % m/s^2
Pd = rho_w*h1*g;
assumeAlso(Pd == P1-P2);

Pd1 = double(subs(Pd,h1,1e-3)); %  Pa
eqn1 = Pd1==rho_a*V1^2/2; % definition of dynamic pressure

V1 = double(solve(eqn1,V1));

Pd2 = Pd1*sind(50);
eqn2 = Pd2==rho_a*V2^2/2;
```

```
V2 = double(solve(eqn2,V2));


disp("Attach line 1 and line 2 to the manometer to find the value of dynamic
 pressure.")
disp("Dynamic Pressure is: "+string(Pd)+ " Pa")
disp("Using the definintion of dynamic pressure, V1 is: "+string(V1)+" m/s")
disp("Using the definintion of dynamic pressure, V2 is: "+string(V2)+" m/s")
disp("The calculated speed is higher because there is an implication that the
 pressure is higher but since this is within the error of the manometer it is
 unreasonable to report a significant difference.")
```

```
-----FS3-----
Attach line 1 and line 2 to the manometer to find the value of dynamic
 pressure.
Dynamic Pressure is: (978057*h1)/100 Pa
Using the definintion of dynamic pressure, V1 is: 3.996 m/s
Using the definintion of dynamic pressure, V2 is: 3.4975 m/s
The calculated speed is higher because there is an implication that the
 pressure is higher but since this is within the error of the manometer it is
 unreasonable to report a significant difference.
```

FM1

```
disp("-----FM1-----")
clear all;
clc

state0 = [100 -11.2]*1000;
tspan = [0 60*60];
options = odeset('RelTol', 1e-8,'AbsTol',1e-8);

[t,y] = ode45(@falling,tspan,state0,options);

figure
plot(abs(y(:,2)),abs(y(:,1)))
ylabel("altitude [m]")
xlabel("speed [m/s]")
title("Altitude vs Velocity")
disp("assumptions:")
disp("stdAtm")
disp("V0 = -11.2 km/s")
disp("y0 = 100 km")
```

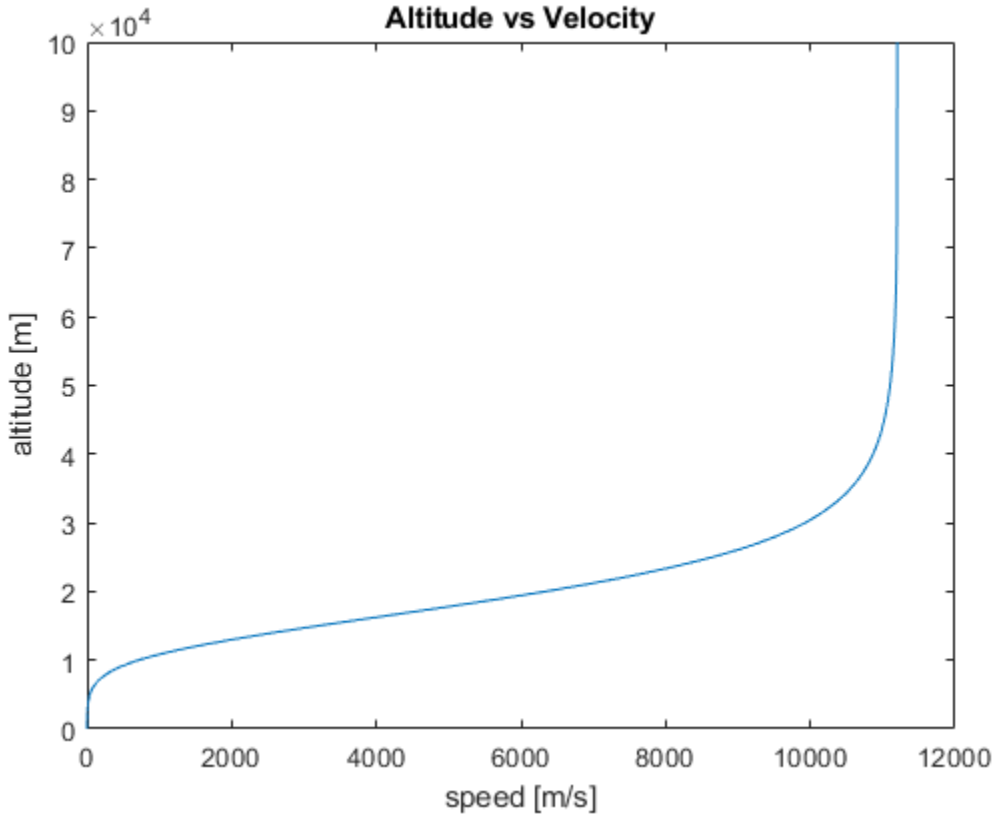
```
-----FM1-----
```

function def

```
function dstate = falling(t,state)
    % disp("here")
    % y = myState(1);
    BC = 4800;%N/m^s
```

```
    g = 9.81;%m/s^2
    y = state(1);
    yd = state(2);
    if y>0
        [~,~,rho] = stdAtmOatesJoshua(y);
    else
        rho = stdAtmOatesJoshua(1);
    end

    ydd=((BC^-1)*((rho*yd^2)/2))*g;
    dstate = [yd;ydd];
end
```

assumptions:
stdAtm
V0 = -11.2 km/s
y0 = 100 km



Published with MATLAB® R2022a

```
function [T,P,rho,mu] = stdAtmOatesJoshua(h)
% returns various properties of the atmosphere for a given altitude
%   calculate Temperature in K, Pressure in Pa, and Desnsity in kg/m^3, for an
 altitude in m,
%   uses the 1959 model of the atmosphere


% Sea level temperature 288.16 K
% Sea level pressure 101.325 kPa
% Sea level density 1.2250 kg/m^3
%
%
% Gradient layer  -0.0065 K/m up to 11000m
% Isothermal layer  11000m to 25000m
% Gradient layer  0.0030 K/m up to 47000m
% Isothermal layer  47000m to 53000m
% Gradient layer  -0.0045 K/m up to 79000m
% Isothermal layer  79000m to 90000m
% Gradient layer  0.0040 K/m up to 100000m


%set parameters at sea level

T0 = 288.16; %K
P0 = 101325; %Pa
rho0 = 1.2250; %kg/m^3
h0 = 0; %m
g0 = 9.80665; %m/s2
R = 287; %J/(kg*K)

muRef = 1.716e-5;
TRef = 273.15;
S = 110.4;


%lapse rate per layer and top of layer
% a1 =  -0.0065; %K/m
% h1 = 11000; %m
%
% a2 = 0; %K/m
% h2 = 25000; %m
%
% a3 = 0.0030; %K/m
% h3 = 47000; %m
%
% a4 = 0; %K/m
% h4 = 53000;%m
%
% a5 = -0.0045; %K/m
% h5 = 79000;%m
%
% a6 = 0; %K/m
```

```
% h6 = 90000;%m
%
% a7 = 0.0040;  %K/m
% h7 = 100000;%m
% %solve for T, P, rho at each base

aMat = [-0.0065,0,0.0030,0,-0.0045,0,0.0040]; %a 1-7 in K/m
hMat = [0,11000,25000,47000,53000,79000,90000,100000]; %h 0-7 in m

TMat = [T0];        %temp at base of layer 1-7 and at the top of 7
PMat = [P0];        %pressure at base of layer 1-7 and at the top of 7
rhoMat = [rho0];    %density at base of layer 1-7 and at the top of 7


for i = 1:7 % for each layer 1-7 calculate T P and rho
    TMat(i+1) = TMat(i) + aMat(i) * (hMat(i+1) - hMat(i));

    if aMat(i)==0 %isothermal
        PMat(i+1) = PMat(i)*(exp(-(g0/(R*TMat(i+1)))*(hMat(i+1)-hMat(i))));
        rhoMat(i+1) = rhoMat(i)*(exp(-(g0/(R*TMat(i+1)))*(hMat(i+1)-
hMat(i))));
    else%gradient
        PMat(i+1) = PMat(i)*((TMat(i+1)/TMat(i))^(-g0/(aMat(i)*R)));
        rhoMat(i+1) = rhoMat(i)*((TMat(i+1)/TMat(i))^((-g0/(aMat(i)*R)-1)));
    end
end


inLayer = 0; %inLayer will hold the layer that that h is in


while hMat(inLayer+1) < h
    inLayer=inLayer+1;
end

if h == 0
    layer = 1;
end



%solve T, P, rho at h by resusing the formulas above. Now the output is
%final, use (inLayer) where i was before, use h T once solved

T = TMat(inLayer) + aMat(inLayer) * (h - hMat(inLayer));

if aMat(inLayer)==0 %isothermal
    P = PMat(inLayer)*(exp(-(g0/(R*T))*(h - hMat(inLayer))));
    rho = rhoMat(inLayer)*(exp(-(g0/(R*T))*(h - hMat(inLayer))));
else %gradient
    P = PMat(inLayer)*((T/TMat(inLayer))^(-g0/(aMat(inLayer)*R)));
    rho = rhoMat(inLayer)*((T/TMat(inLayer))^((-g0/(aMat(inLayer)*R)-1)));
end
```

```
mu = muRef*((TRef+S)/(T+S))*(T/TRef)^1.5;

end
```

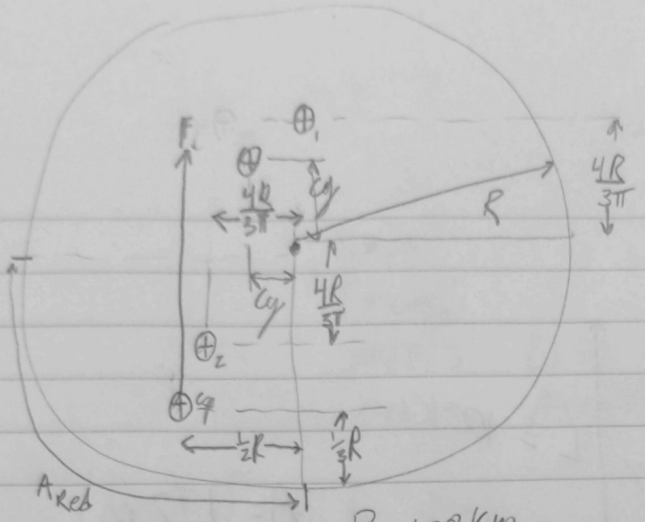
Published with MATLAB® R2022a

```
function [PI,a] = joshBuckPiTheory(M,qi)
%JOSHBUCKPITHEORY finds pis of dimentional matrix M

% to find physically meaningful function h in terms of non-dimentional
% params pi1,pi2,...,pip
% h(q1,q2,...qn)=0
% H(pi1,pi2,...pip)=0

arguments
    M (:,:) double
    qi (:,1) string = []
end


a = null(M,'rational'); % a's, should match length q
[k,n] = size(M);

p = n-k; % number of pi's

if p<1
    throw(MException("joshBuckPiTheory:invalidInput","M is wrong
 dimensions."))
end

for i = 1:p
    acol = a(:,i);
    if sum(mod(acol,1),'all') ~= 0
        asym = sym(acol);
        [numerators(:,1),denominators(:,1)] = numden(asym);
        commonMultiple = prod(denominators);
        acol = acol*commonMultiple;
        GCD = gcd(acol);
        acol = acol/GCD;
        a(:,i) = acol;

    end
end

if [0 1] == size(qi)
    syms q [n 1]; % number of qs ex: L M and T
elseif [n 1] == size(qi)
    q = str2sym(qi);
else
    throw(MException("joshBuckPiTheory:invalidInput","qi is wrong size."))
end

syms pi_vecs [n p] % pi's by terms, not a product yet
for i = 1:p
    pi_vecs(:,i) = q.^a(:,i);
end
syms PI [1 p]
for i = 1:p
```

```
    PI(i) = prod(pi_vecs(:,i)); % product generates p nondimentional params
 pi(1-p)
end
end
```

Published with MATLAB® R2022a

FS4



$\oplus_1 \; A_1 = \langle 0, 1 \rangle \frac{4R}{3\pi} \frac{\pi R^2}{2}$

$+ \; \oplus_2 \; A_2 = \langle -1, -1 \rangle \frac{4R}{3\pi} \frac{\pi R^2}{4}$

$\oplus_{cg} A = \langle -3.33, 3.33 \rangle \frac{\pi R^2}{.75}$

$cg = \langle -1.415, 1.415 \rangle \cdot 6^4 \; m$

$R = 100 \; km$

$F_B = \rho V g$, $V_s = \frac{\pi R^2}{4}$, $g = 9.81 \frac{m}{s}$, $\rho_{air} = 1.225 \; kg/m^3$

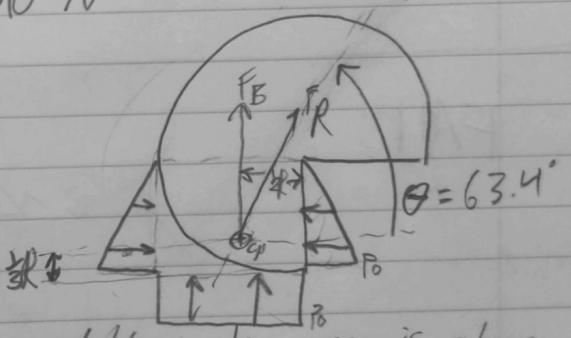
$F_B = 943{,}832.6 N$

$\rho_{hydrogen} = .08375 \; kg/m^3$

$W_h = \rho_{hydrogen} \cdot 3 V_s = 1.936e10 \; N$

$F_B = W_{tot} = W_h + W_{struct}$

$W_s = 7.525e10 \; N$



the spaceship is not stically stable, its cg is above and to the right of the center of pressure

$\rho_{water\,vapor} = \rho_v = .598 \; \frac{kg}{m^3}$

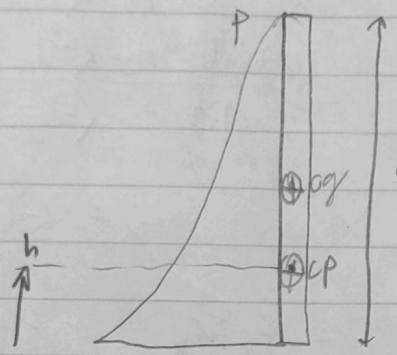
$\frac{W_h}{W_v} = \frac{\rho_h}{\rho_v}$, $W_v = W_h \frac{\rho_v}{\rho_h}$

$W_v = 1.3824e11 > F_B$ so the spaceshit would sink if filled with water vapor

$P_{(0)} = \rho_{air} g R = 1201725$, $P_{res} = \sqrt{(P_0 R)^2 + (\frac{1}{2} P_0 R)^2} \; 1.3436e11$

$P_{net} = P_{res}(A) = 2.6871e11 \; Pa \, m^2$; $\theta = \arctan(2) = 63.4$

FS2



$A = L(2m)$

$=$

$h = 6908\,m$

FM1