Table of Contents


```
% Christopher Covert
% AA 200: PS5
% Due 3/16/18
```

Problem One

```
clear;
clc;
close all;

rho = 1.225; %kg/m^3
l = 2; %m
R = 1; %m
dTheta_deg = 60; %deg
dTheta_rad = dTheta_deg*pi/180; %rad
TR = 450; %N
w = 100; %rad/s
numBlades = 6;
FOM = 0.71;

% (a): Power Required | Momentum Theory
% Pin = TR*u = TR*sqrt(T/(2*rho*A))
A = pi*R^2;
u = sqrt(TR/(2*rho*A));
Pin = TR*u;
Pin_tot = numBlades*Pin;
Pin_tot_ACT = Pin_tot/FOM;

% (b): fmincon
Ttrim = 2700;
A = [1 1 1 1 1;...
    -l*sin(pi/2-dTheta_rad) l*sin(pi/2-dTheta_rad) l l*sin(pi/2-
dTheta_rad) -l*sin(pi/2-dTheta_rad);
    -l*sin(dTheta_rad) -l*sin(dTheta_rad) 0 l*sin(dTheta_rad)
 l*sin(dTheta_rad);];
B = [Ttrim; 0; 0;];
T0 = ones(numBlades-1,1)*50;
[Ti, fval, exitflag, output] = fmincon(@powerfcn, T0, [], [], A, B);
ui = sqrt(Ti/(2*rho*A));
Pi = Ti.*ui/FOM;
C_T = TR/(0.5*rho*A*w^2*R^2);
```

```
wi = sqrt(Ti/(0.5*rho*A*C_T*R^2));

fprintf('Ti = [%2.2f, %2.2f, %2.2f, %2.2f,
 %2.2f]\n',Ti(1),Ti(2),Ti(3),Ti(4),Ti(5))
fprintf('ui = [%2.2f, %2.2f, %2.2f, %2.2f,
 %2.2f]\n',ui(1),ui(2),ui(3),ui(4),ui(5))
fprintf('Pi = [%2.2f, %2.2f, %2.2f, %2.2f,
 %2.2f]\n',Pi(1),Pi(2),Pi(3),Pi(4),Pi(5))
fprintf('wi = [%2.2f, %2.2f, %2.2f, %2.2f,
 %2.2f]\n',wi(1),wi(2),wi(3),wi(4),wi(5))
```

Problem Two

```
clear;
clc;

[power,thrust,etaprop,cp,ct,lambda,r,incidence,chord,cl] =...
    AA200OptProp(0,0,0);

figure
plot(r,cl)
title(['C_l vs. r (Power = ' num2str(power,2) ')'])
xlabel('r')
ylabel('C_l')

figure
plot(r,chord)
title(['Chord vs. r (Power = ' num2str(power,2) ')'])
xlabel('r')
ylabel('Chord')

figure
plot(r,incidence)
title(['Incidence vs. r (Power = ' num2str(power,2) ')'])
xlabel('r')
ylabel('Incidence')

[power,thrust,etaprop,cp,ct,lambda,r,incidence,chord,cl] =...
    AA200OptProp(0.016,0,0.014);

figure
plot(r,cl)
title(['C_l vs. r (Power = ' num2str(power,2) ')'])
xlabel('r')
ylabel('C_l')

figure
plot(r,chord)
title(['Chord vs. r (Power = ' num2str(power,2) ')'])
xlabel('r')
ylabel('Chord')

figure
```

```
plot(r,incidence)
title(['Incidence vs. r (Power = ' num2str(power,2) ')'])
xlabel('r')
ylabel('Incidence')
```

Problem Three

```
clear;
clc;

rho = 0.002377; %slug/ft^3
CD = 0.3;
U_wm = 33.756; %ft/s
A = 6; %ft^2
CD_wm = 1.0;
mu = 0.05;
W = 100; %lb
eff = 0.98;
R_wm = 3; %ft;

U_car = sym('U_car');

Pwm = rho*pi*R_wm^2*0.25*(U_wm+U_car)^3;
Pwh = eff*Pwm;
F_motor = mu*W +...
    CD*(0.5*rho*(U_wm+U_car)^2*A)+...
    CD_wm*(0.5*rho*(U_wm+U_car)^2*pi*R_wm^2);

EQN1 = simplify(Pwh-F_motor*U_car)

%From Wolfram Alpha, U_car = 26.9364 ft/s

fprintf('From Wolfram Alpha, U_car = 26.9364 ft/s\n')

Pwm = rho*pi*R_wm^2*0.25*(U_wm-U_car)^3;
Pwh = eff*Pwm;
F_motor = mu*W +...
    CD*(0.5*rho*(U_wm-U_car)^2*A)+...
    CD_wm*(0.5*rho*(U_wm-U_car)^2*pi*R_wm^2);

EQN2 = simplify(Pwh-F_motor*U_car)

%From Wolfram Alpha, U_car = 9.18822 ft/sec
fprintf('From Wolfram Alpha, U_car = 9.18822 ft/s\n')
```

Functions

```
function P = powerfcn(T)
rho = 1.225;
R = 1;
FOM = 0.71;
A = pi*R^2;
u = sqrt(T/(2*rho*A));
```

```
P = (T'*u)/FOM;
end

% AA200OptProp.m
% Optimize a propeller or windmill using fmincon and a nested
 function.
% © Ilan Kroo, 2010.
%
% This version simplified for use in Stanford courses.
% Units are either sl, ft, sec, lb or kg, m, sec, N; angles in
 radians.
% Note that prop_objective, prop_constraints, prop_compute should be
 embedded/nested.
function [power,thrust,etaprop,cp,ct,lambda,r,incidence,chord,cl] =
 AA200OptProp(cd0,cd1,cd2)
%Geometry
hubfraction = 0.2;
radius = 1; % m
nblades = 2;
nsections = 20;
% Conditions
omega = 100; % rad/sec
vtip = omega*radius; % m/sec
u0 = 0.0; % Forward speed, m/sec
rho = 1.225; % Sea level density, kg/m^3
lambda = u0/vtip;
rpm = vtip/radius/pi*30.;
requiredthrust = 450; % N
% Aerodynamics
cla = 5.8; cl0 = .4; clmax = 2.0;
% Section cd fit: cd = cd0 + cd1*cl + cd2*cl^2 or use section data
 lookup
%cd0 = 0.016; cd1 = 0.0; cd2 = 0.014;
% Initialize
chord = zeros(nsections,1); cl = zeros(nsections,1); r =
 zeros(nsections,1);
gamma = zeros(nsections,1); incidence = zeros(nsections,1);
u = zeros(nsections,1); v = zeros(nsections,1);
phi = zeros(nsections,1); kappa = ones(nsections,1);
ub = zeros(2*nsections,1); lb = zeros(2*nsections,1);
x1 = zeros(nsections,1); x2 = zeros(nsections,1); x0 =
 zeros(2*nsections,1);
% Starting point and bounds
u=[20; 20; 20; 20; 20; 20; 20; 20; 20; 20; 20; 20; 20; 20; 20; 20; 20;
 20; 20; 20]/5;
chord=[.03;.06;.1;.15;.23;.33;.37;.43;.47;.49;.51;.51;.5;.48;.46;.42;.38;.33;.26;.
for i = 1:nsections
    x0(i)=u(i); lb(i)=-10; ub(i)=100; x0(i+nsections)=chord(i);
    lb(i+nsections)=0.001; ub(i+nsections)= 2*pi*radius*i/nsections/
nblades*0.8;
end
% Call fmincon optimizer
options =
 optimset('Display','iter','MaxFunEvals',10000,'MaxIter',1000,...
```

```
    'TolFun',1e-7,'TolCon',1e-5,'TolX',1e-5,'MaxSQPIter',600);
[x,fval,exitflag,output] = ...
    fmincon(@prop_objective,x0,[],[],[],
[],lb,ub,@prop_constraints,options);
% Call analysis one more time at solution:
x1 = x(1:nsections)'; x2 = x(nsections+1: 2*nsections)'; u = x1; chord
 = x2;
[thrust,torque,power,etaprop,ct,cp,r,incidence,v,kappa,phi,cl] =
 prop_compute()
fm = ct/(2*cp)*(lambda+sqrt(lambda^2+ct))
% Nested functions to evaluate objective and constraints:
    function y = prop_objective(x)
        x1 = x(1:nsections)';
        x2 = x(nsections+1: 2*nsections)';
        u = x1;
        chord = x2;
        [thrust,torque,power,etaprop,ct,cp,r,incidence,v,kappa,phi,cl]
 = prop_compute();
        % Choose appropriate objective to minimize
        % y = cp;
        % y = power;
        y = torque;
    end
    function [c, ceq] = prop_constraints(x)
        ceq = [];
        c = [];
        x1 = x(1:nsections)';
        x2 = x(nsections+1: 2*nsections)';
        u = x1;
        chord = x2;
        [thrust,torque,power,etaprop,ct,cp,r,incidence,v,kappa,phi,cl]
 = prop_compute();
        c(1) = (requiredthrust-thrust)/requiredthrust;
        % May want to bound cl
        for i = 1:nsections c(i+1) = cl(i)-clmax; end
    end
% Nested main compute function
    function
 [thrust,torque,power,etaprop,ct,cp,r,incidence,v,kappa,phi,cl]=prop_compute()
        thrust = 0;
        torque = 0;
        ct = 0;
        cp = 0;
        etaprop = 0;
        omega = rpm*2.*pi/60.;
        dr = radius/nsections;
        tan_phit = u0/omega/radius;
        sin_phit = tan_phit/sqrt(1+tan_phit^2);
        % Basic noniterative approach:
        % T' = 4 pi r rho u (U0+u) kappa
        % T' = nblades rho Gamma (omega r - v)
        % Q' = nblades rho Gamma (U0+u) r
        % Q' = 4 pi r^2 rho (U0+u) v kappa
        % See AA200 notes.
```

```
        for i=1:nsections
            frac = (i-.5)/nsections;
            r(i) = frac*radius;
            cl(i) = 0.0;
            if frac>hubfraction
                % Limit swirl to omega r / 2.
                % If swirl is limited, use u-based expression for
 Gamma.
                if 4*u(i)*(u0+u(i)) >= omega^2*r(i)^2
                    v(i) = omega*r(i)/2.;
                else
                    v(i) = (omega*r(i) -
 sqrt(omega^2*r(i)^2-4*u(i)*(u0+u(i))))/2.;
                end
                tan_phi = (u0+u(i))/(omega * r(i)-v(i));
                phi(i) = atan(tan_phi);
                sin_phi = tan_phi/sqrt(1+tan_phi^2);
                cos_phi = sqrt(1-sin_phi^2);
                kappa(i) = 2./pi * acos(exp(-nblades*(1-frac)/
sin_phi/2.));
                gamma(i) = 4*pi*u(i)*(u0+u(i))*r(i)*kappa(i)/
((omega*r(i)-v(i))*nblades);
                vtot = sqrt((u0+u(i))^2 + (omega*r(i)-v(i))^2);
                q = .5*rho*vtot^2;
                cl(i) = rho*vtot*gamma(i)/(q*chord(i));
                alpha = (cl(i)-cl0)/cla;
                incidence(i) = alpha+phi(i);
                cd = cd0+cd1*cl(i)+cd2*cl(i)^2;
                dtdr = 4*pi*r(i)*rho*u(i)*(u0+u(i))*kappa(i)-
nblades*q*chord(i)*cd*sin_phi;

 dqdr=nblades*rho*gamma(i)*(u0+u(i))*r(i)+nblades*q*chord(i)*cd*cos_phi*r(i);
                thrust = thrust+dtdr*dr;
                torque = torque+dqdr*dr;
            end
        end % End loop over section
        power = torque*omega;
        etaprop = thrust*u0/power;
        cp = power/(.5*rho*omega^3*pi*radius^5);
        cq = torque / (.5*rho*omega^2*pi*radius^5);
        ct = thrust/(.5*rho*omega^2*pi*radius^4);
        ctmc = pi^3/8* ct;
        cqmc = pi^3/16 * cp;
        J = pi*lambda;
    end % End prop_compute
```

plot desired output %%%

```
end % End AA200OptProp
```

Local minimum found that satisfies the constraints.

```
Optimization completed because the objective function is non-
decreasing in
feasible directions, to within the default value of the optimality
 tolerance,
and constraints are satisfied to within the default value of the
 constraint tolerance.
```

```
Error using /
Matrix dimensions must agree.

Error in AA200_PS5 (line 38)
ui = sqrt(Ti/(2*rho*A));
```

Published with MATLAB® R2017b