Norges teknisk-naturvitenskapelige universitet

Naval Hydrodynamics
TMR4220

Induction-factor-enhanced Lifting Line Code

Yaolin GE

14-Mar-2019

Noregs teknisk-naturvitskaplege universitet

Contents

List of Figures

Preface

This report is for the project in the course TMR 4220 Naval Hydrodynamics in the Spring of 2019 at Norwegian University of Science and Technology(NTNU). The stated goals of this project are to learn to identify, delimit, formulate and solve propeller analysis problem. To synthesise the theories and methods learned in the course, to understand and reflect on them through application to this specific numerical lifting line project. Through this project, a number of Matlab scripts were developed to analyse, optimise and visualise the objective purposes.

We would like to thank my Professor, Kourosh Koushan and Student Assitants, Alvaro del Toro Llorens, Stian Schencke Sivertsgård and Ingvild Persson Moseby for their assistance and support throughout the project.

By the signatures below, We hereby certify that all work was conducted independently in terms of writing the report, coding the necessary Matlab scripts and carrying out essential computational verification.

Yaolin Ge

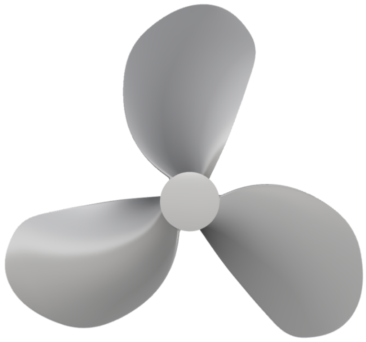
Place: Trondheim, Norway Date: 14-Mar-2019

Nomenclature

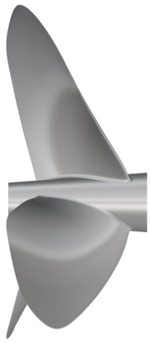
α	Apparent angle of attack
β_i	Hydrodynamic pitch angle
Γ_0	Initial circulation at each section
Γ_1	Updated circulation at each section
ν	Kinematic viscosity
ϕ	Geometrical pitch angle
N	Number of elements
ξ	Numerical damping
C_L	Initial lift coefficient
C_{L0}	Updated lift coefficient
R_N	Characteristic Reynolds number
C_{dv}	Viscous drag coefficient
D	Diameter of each section
J	Advance coefficient
K_Q	torque coefficient
K_T	thrust coefficient
n	rotational speed per second
Q	Total torque
T	Total thrust
U_A	Axial induced velocity
U_T	Tangential induced velocity
V_∞	Infinite inflow velocity
V_A	velocity of advance
Z	Number of blades

1 Introduction

This project is aimed to use numerical lifting line method to analyse a 3-blade propeller of the Wageningen B-screw series with a diameter B of 1.20 m, a pitch to diameter ratio P/D of 1.05 and an expanded blade area ratio EAR equal to 0.50. The objective propeller can be illustrated as follows:



(a) Front view.



(b) Side view.

Figure 1: Propeller illustration

Numerical lifting line method, also called Prandtl's lifting line theory, which initially investigates a relatively high aspect–ratio foil,which has a comparably high span over chord ratio. But it can also be used for designing and analysing a propeller thanks to its computational efficiency and relatively high accuracy. However, the shed vorticity is induced by the rotational blades causes the propeller lifting line method procedures more complicated. However, the principle of this kind of application of lifting line method for a propeller is the same as a simple foil. In this project, only one blade of the propeller is analysed when it comes to circulation distribution along the blade radii, but total thrust and torque is derived for the whole propeller. It is also noteworthy that the circulation is assumed that the circulation to be concentrated along a line through each propeller blade, hence no chord wise circulation variation is considered. It is illustrated as the graph shown below:

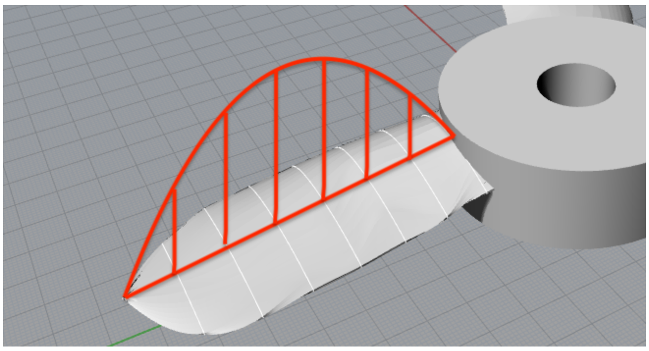


Figure 2: Radial circulation illustration(not real shape)

The main procedure in analysing the propeller by using numerical lifting line method is divided into three steps:

1. Initialisation
   It is aimed to initialise the parameters needed to start an iteration in the following section, which include:

   - C_{L0} – the reference lift coefficient, which is initialised by calling Xfoil to compute the lift coefficients for each section, here, the reference angle of attack is set to be zero.
   - Γ_0 – the radial circulation distribution, which is always initialised to be zero for simplicity in this project.
   - ϕ –the geometrical pitch angle which can be calculated as in the following equation:

   $$\tan\phi = \frac{P}{2\pi r} \implies \phi = \arctan\frac{P}{2\pi r} \tag{1}$$

   - n – rotational speed per second, which are set to be the same in three cases such as estimation of thrust coefficient and torque coefficient without induced velocities, the estimation of thrust coefficient and torque coefficient with induced velocities based on simple momentum theory and the estimation of thrust coefficient and torque coefficient with induced velocities based on more advanced techniques such as induction factors. Here, it is derived from the following three relations:

   $$R_N = \frac{V_{0.7\infty}c}{\nu}$$

   $$J_A = \frac{V_A}{nD}$$

   $$V_{0.7\infty} = \sqrt{(V_A^2 + (2\pi r_{0.7})^2}$$

   Thus, it can be expressed as:

   $$n = \frac{V_{0.7\infty}}{\sqrt{(J_{0.7\infty}D)^2 + (2\pi r_{0.7})^2}} \tag{2}$$

   - V_A – inflow velocities, which are calculated based on the required advance coefficients, which is shown below:

   $$V_A = nJ_AD \tag{3}$$

2. Iteration
   which is used to find the updated solution until it reachs the convergence requirement, but only for cases with considering the induced velocities, The parameters need to be updated in the iteration are shown below:

   - U_T – tangential induced velocity, which can be calculated in two different manners in this project, the first approach is based on simple momentum theory. Then it can be expressed as:

   $$U_T = \frac{\Gamma}{2\pi r} \tag{4}$$

   Another approach is based on tangential induction factor, which hence leads to the tangential induced velocity:

   $$U_T(r_0) = \int_{r_h}^{R} \frac{i_T(r_0, \beta_i)}{2\pi} \cdot \frac{\partial\Gamma(r)}{\partial r} \cdot \frac{dr}{r_0 - r} \tag{5}$$

   - U_A – axial induced velocity, which is also calculated in both simple momentum way and advanced way(i.e. use axial induced factor). The simple axial induced velocity based on simple momentum theory is hence expressed as:

   $$U_A = -V_A + \sqrt{V_A^2 + U_T(4\pi rn - U_T)^2} \tag{6}$$

Also, the advanced axial induced velocity is therefore calculated as:

$$U_A(r_0) = \int_{r_h}^{R} \frac{i_A(r_0, \beta_i)}{2\pi} \cdot \frac{\partial \Gamma(r)}{\partial r} \cdot \frac{dr}{r_0 - r} \tag{7}$$

- V_∞ – the infinite inflow velocity, since the induced velocities have been found, thus the apparent inflow magnitude is calculated as:

$$V_\infty = \sqrt{(V_A + \frac{U_A}{2})^2 + (2\pi rn - \frac{U_T}{2})^2} \tag{8}$$

- β_i – the hydrodynamic pitch angle, which is calculated as the following equation which is based on velocity triangle:

$$\beta_i = \arctan(\frac{V_A + \frac{U_A}{2}}{2\pi rn - \frac{U_T}{2}}) \tag{9}$$

- α – the apparent angle of attack, which is the resulting angle which is subtracted from the geometrical pitch angle by the hydrodynamic pitch angle. It is expressed as:

$$\alpha = \phi - \beta_i \tag{10}$$

- C_L – updated lift coefficients. Since the apparent of angle of attack is derived, then the lift coefficient can be calculated based on linear foil theory(Kutta Joukowski theoerm) as follows:

$$C_L = 2\pi\alpha + C_{L0} \tag{11}$$

- Γ – the updated circulation at each section, which is calculated based on linear foil theory, Z is the number of blades, while c is the chord length at each section.

$$\Gamma = \frac{1}{2} \cdot C_L \cdot V_\infty \cdot c \cdot Z \tag{12}$$

3. Post-processing
   In this section, the total thrust and torque are calculated so as to find out the thrust coefficient and torque coefficient as required. The parameters needed to compute the final resulting total thrust and total torque are listed below:

- R_N – updated Reynolds number, which is calculated based on the updated infinite inflow velocities as follows:

$$R_N = \frac{V_\infty c}{\nu} \tag{13}$$

- C_F – the frictional coefficient based on ITTC 1957, then it can be written as:

$$C_F = \frac{0.075}{(\log_{10}(R_N) - 2)^2} \tag{14}$$

- C_{dv} – the viscous drag coefficient, which can be modelled based on C_L and C_F, the thickness to chord ratio shows that the displacement effect and viscous pressure effect, while the last lift dependent term is to demonstrate the dependence of drag due to lift.

$$C_{dv} = 2C_F(1 + \frac{t}{c} + 60(\frac{t}{c})^4) \cdot (1 + \frac{C_L^2}{8}) \tag{15}$$

- T – resulting thrust, which can be calculated as two separate part including ideal thrust and the reduced thrust due to profile drag, which is hence expressed as:

$$\text{Ideal thrust: } dT_i = \rho\Gamma(2\pi rn - 0.5 \cdot U_T)dr$$

$$\text{Reduced thrust: } dT_D = \frac{1}{2}\rho V_\infty^2 \cdot c \cdot D \cdot Z \sin\beta_i dr$$

Thus, the resulting thrust is equal to:

$$\text{Total thrust: } T = \int_{rboss}^{R} dT_i - dT_D \tag{16}$$

- Q – resulting torque, the same as thrust, it can also be splited into two parts such as ideal part and induced increased drag due to profile drag, which is shown below:

$$\text{Ideal tangential force: } dK_i = \rho\Gamma(V + 0.5U_A)dr$$

$$\text{Increased tangential force: } dK_D = \frac{1}{2}\rho V_\infty^2 \cdot c \cdot D \cdot Z \cos\beta_i dr$$

Thus, the resulting total torque can be written as:

$$Q = \int_{rboss}^{R} (dK_i + dK_D)r \tag{17}$$

- K_T – thrust coefficient, which can then be calculated based on the above parameters as:

$$K_T = \frac{T}{\rho n^2 D^4} \tag{18}$$

- K_Q – torque coefficient, it can also be calculated as:

$$K_Q = \frac{Q}{\rho n^2 D^5} \tag{19}$$

All the parameters needed to conduct a lifting line analysis have been described already, the next step is to build the general structured work flow for different cases, details can be found in the following chapter.

2 Methodology

In this project, two general cases are analysed such as cases without induced velocities and cases with induced velocities. The work flow for these two types of cases are discussed below.

2.1 Workflow I: W/O ind. velocities

The flow chart of building codes for the first kind of cases is illustrated below, since there is no induced velocities, no updating is needed, and no iteration is required neither.

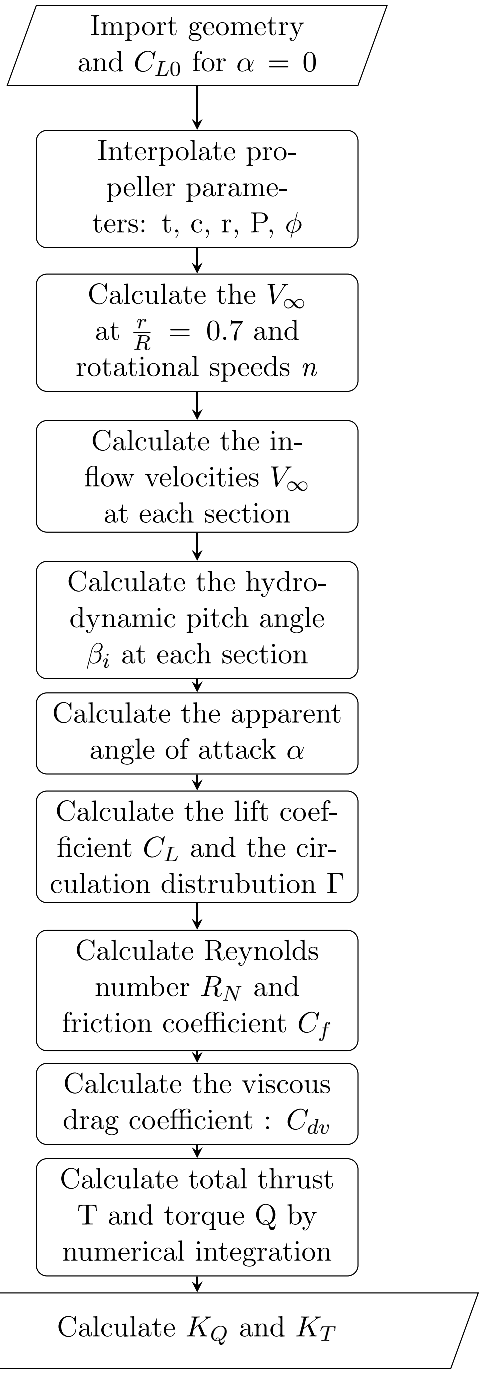


Figure 3: Work flow for cases without ind. velocities

2.2 Workflow II: With ind. velocities

For this type of analysis, iteration is essential to find the final converged solution since the induced velocities are included and they are co-related (i.e. the induced velocities will influence the circulation and also the circulation can return influence the induced velocities). A new flow chart is formulated with considering the effect of induced velocities, which is then listed below:

3 Case study

The methods have been introduced in the above section, this section will discuss three samples with applying the above main work flow into detail.

3.1 Preliminary Calculation of the 2D lift coefficients

3.1.1 Xfoil explanation: using Xfoil on Mac

Using Xfoil on a Macintosh is not simple as we wish, Xquartz and Xfoil both need to be installed, then the working environment should be friendly to users, the main workflow would be:

1. Import the geometry by ***load Geometry.txt***, where the geometry file is located in the root drive.
2. Enter the operating mode by ***oper***
3. Calculate the lift coefficient by ***alfa 0***
4. Repeat the above process until all of ten sections have been calculated.

Here, the viscous effects have been neglected, since Xfoil wouldn't converge very well in this case when the viscous mode is on. At this stage, Xfoil is applied manually, but it is noteworthy that it can run in batch mode, and it will be discussed later when dealling with caviation check.

3.1.2 Lift coefficients radial curve

Based on the calculated lift coefficient, the lift coefficient curve along the radii can be plotted as below, since there is no chord length at the tip-most of the blade profile, hence the lift coefficient coefficient is zero at the tip.

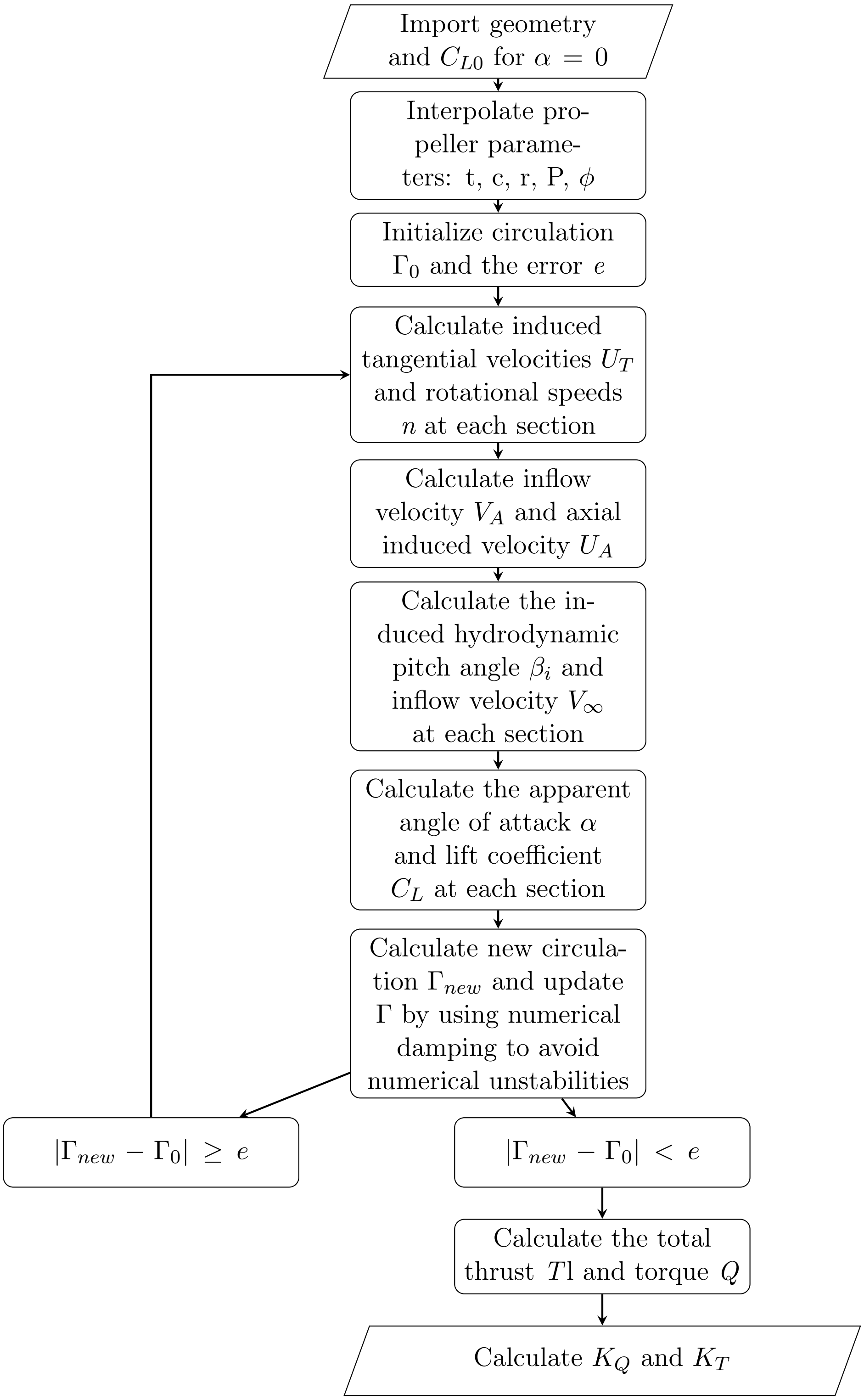


Figure 4: Work flow for cases with ind. velocities

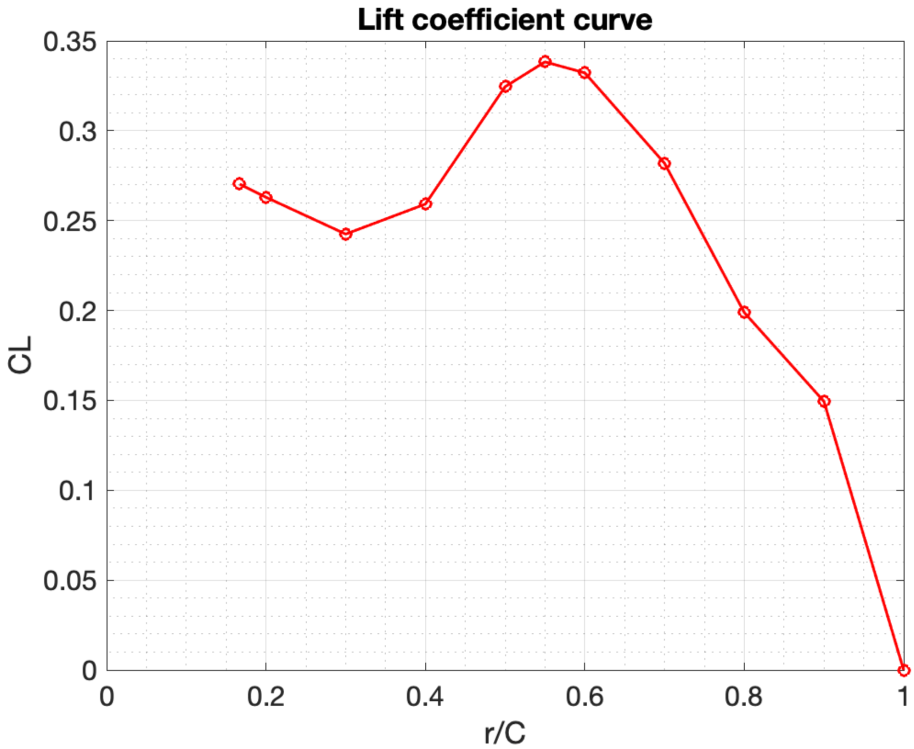


Figure 5: Lift coefficient radial distribution curve

3.2 Case I: LLC without induced velocities

3.2.1 Main workflow

As discussed earlier, in this case, no induced velocities are indclued means no iteration is needed, the work flow can be found in figure 3, which describes the procedures of the work flow of the source code, which can be found in the appendix.

3.2.2 Result and discussion

The open water diagram parameters in terms of thrust coefficient and torque coefficient are calculated and then plotted as shown in the figure 6 and 7. It is clearly seen that the estimated results calculated with the codes without considering the effect of induced velocities will give a huge deviation from the experimental data, which is not the case that we expected. In order to reduce this error, more accurate model needs to be applied, which is discussed in the next section.

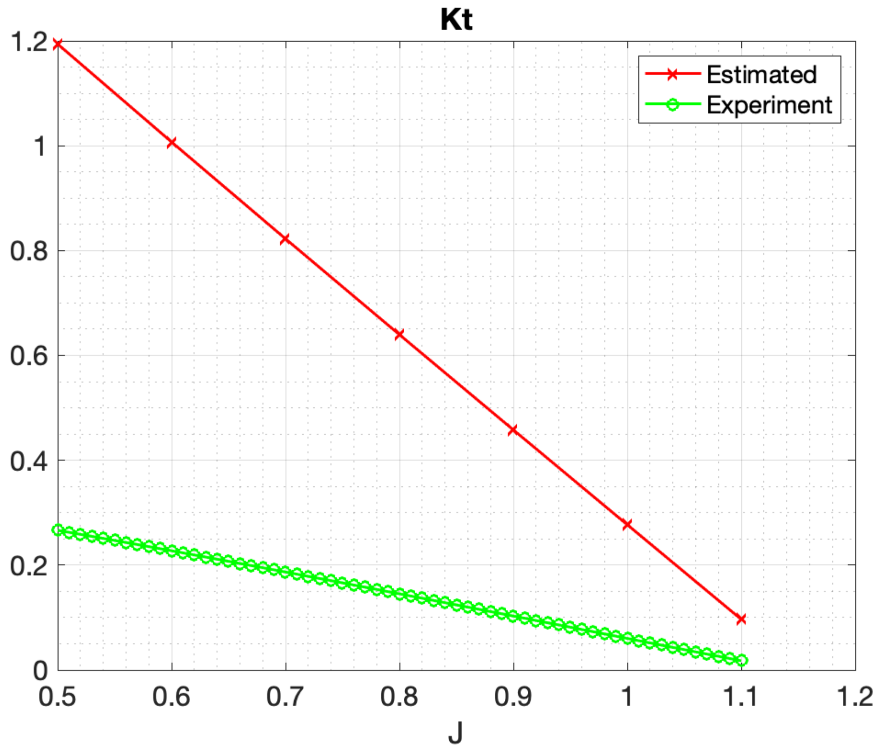


Figure 6: Thrust coefficient Kt open water diagram

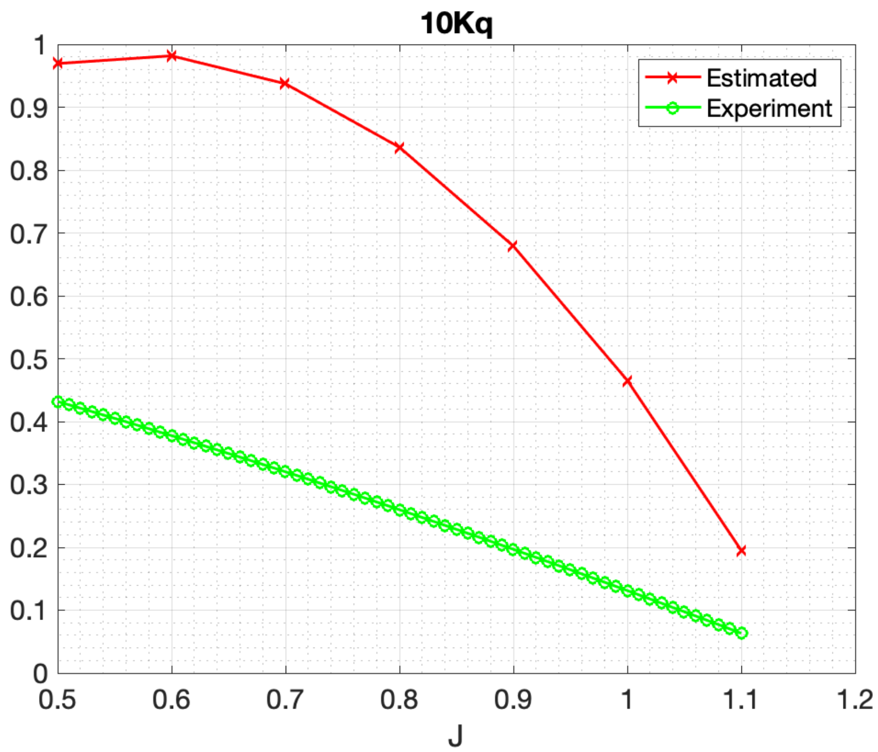


Figure 7: Torque coefficient Kq open water diagram

3.3 Case II: LLC with simple induced velocities

3.3.1 Main workflow

Referring to flow chart 4, the lifting line code with considering the effect of induced velocities based on simple momentum theory will lead to equation4 and 6. Then the critical step is to find the converged circulation, here two convergence requirement are applied which are:

$$error < ErrorAllowed \tag{20}$$

$$nIteration < MaxIteration \tag{21}$$

By iterating each parameters described in the introduction iteration part, the final converged circulation can be plotted as figure 8, which can be seen roughly as a parabolic curve envelops.

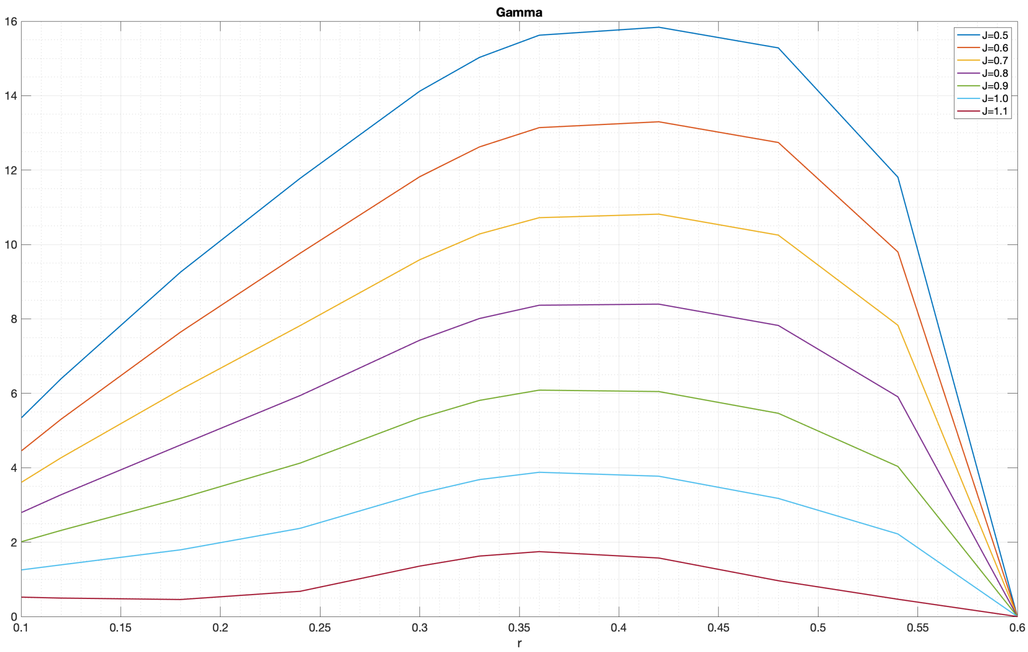


Figure 8: Circulation radial distribution

3.3.2 Result and discussion

The open water diagram parameters in terms of thrust coefficients and torque coefficients can hence be extracted from the calculated results and by comparing with the previous numerical results and the experimental data, it draws the conclusion that this time with considering the effect of the induced velocities, the converged solution is qualitatively improved even though some deviation still exists. It also tells that the importance of induced velocities at either designing stage or analysing stage.

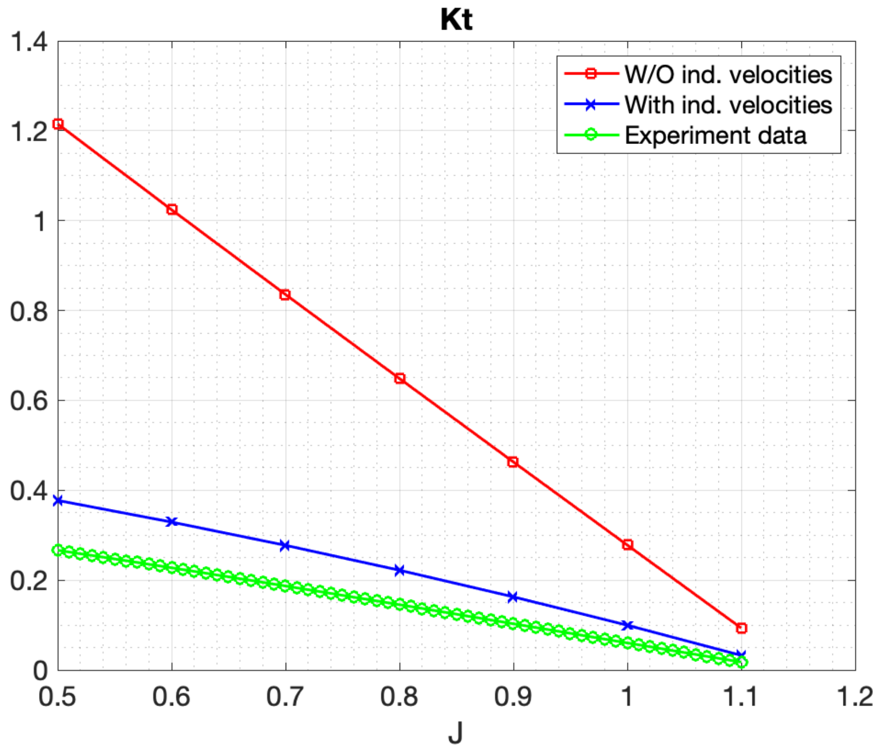


Figure 9: Comparison with experiment and previous result

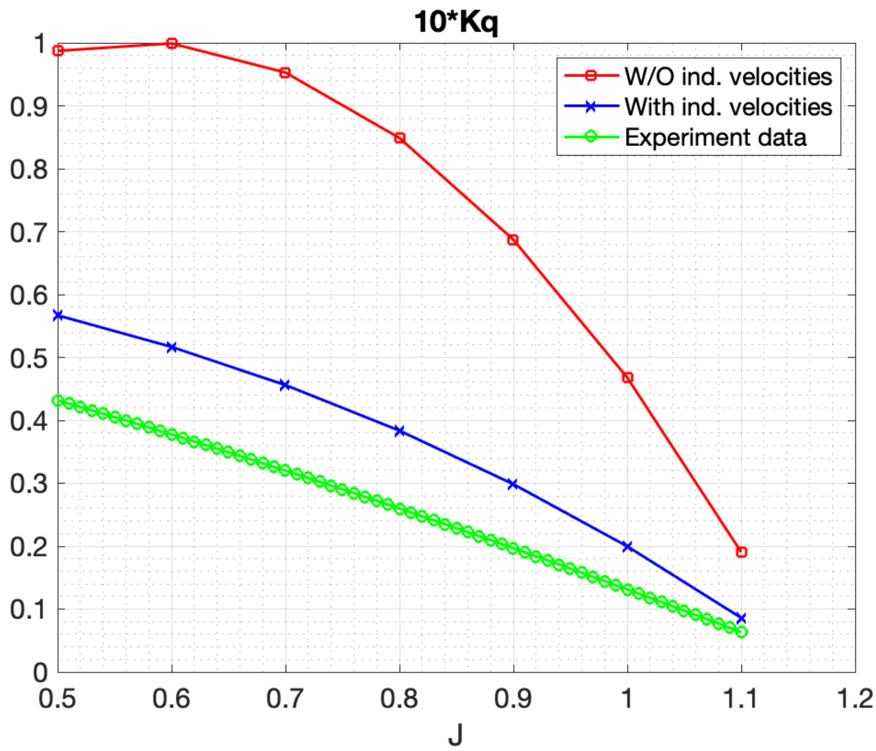


Figure 10: Comparison with experiment and previous result

3.4 Case II: LLC with advanced induced velocities

3.4.1 Prerequisite

The following lists the two main techniques that are necessary to conduct the analysis using induction factor method.

- Linear interpolation to find the profile data for the interpolated sections dependent on the mesh size, or in other words, how many elements are essential to have a relatively high accurate results. This convergence test will be discussed later.

- Cubic spline to find the first derivative of the circulation distribution along the radial location.

The convergence requirements are exactly the same as shown in equation 21.

3.4.2 Main workflow

This time, a more advanced model is employed (say, induction factor) and hence it is also more complicated compared to the simple momentum theory. In this project, the induction factors are computed as a black box. The only thing which is interesting is the output which includes both the axial induction factor and the tangential induction factor. By integrating the nominal tangential and axial 'downwash' using 5 and 7 respectively, the tangential induced velocities and axial induced velocities can be found consequently. It is also worth mentioning that the cubic spline technique is used to determine the first derivative of circulation distribution. Details can be found in the attached codes. Therefore, the rest of the procedure will be accordingly the same as shown in the flow chart4.

3.4.3 Result and discussion

The convergence test is conducted first to find the suitable element number of sections, the plot for convergence test is shown in figure 11. The consuming time is also plotted as shown in the figure 12. It is wise to choose the number of elements from 40 to 60, the first reason is to ensure that convergence is satisfied and another reason is to take the consuming time into consideration as well. Here 60 elements are selected to be able capture more information about the location where the cavitation happens later on.

After the suitable element size has been determined, the corresponding open water parameters in terms of thrust coefficient and torque coefficient are plotted in the figure 13 and 14 with comparison to the results found in the previous cases and also the experimental data. It is clearly shown that an obvious qualitative improvement has been added to the original rough model, which was expected.

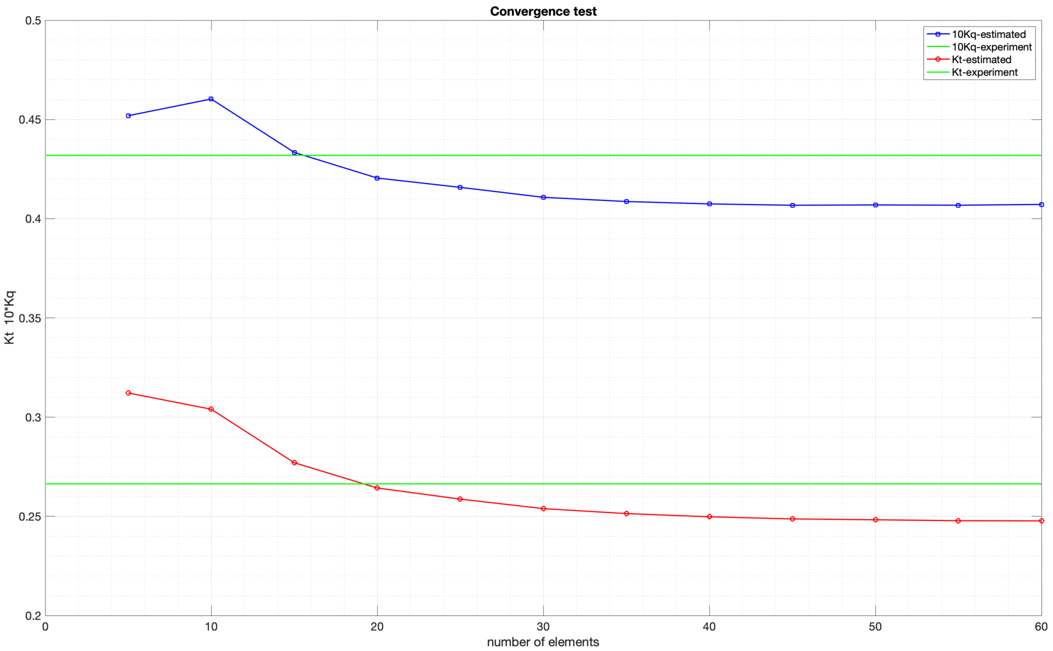


Figure 11: Convergence test

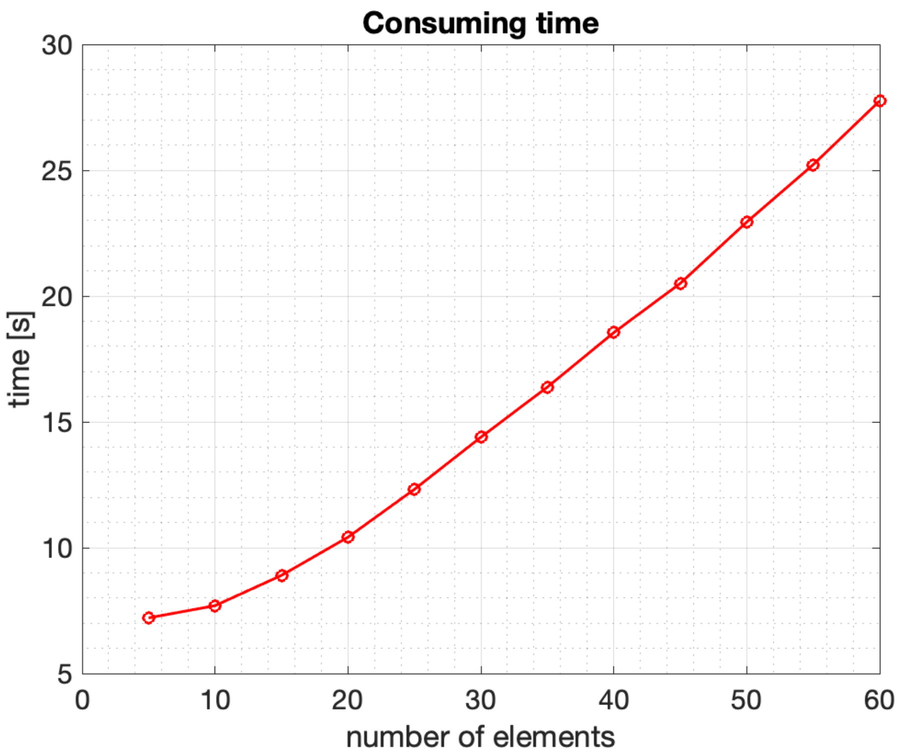


Figure 12: Consuming time

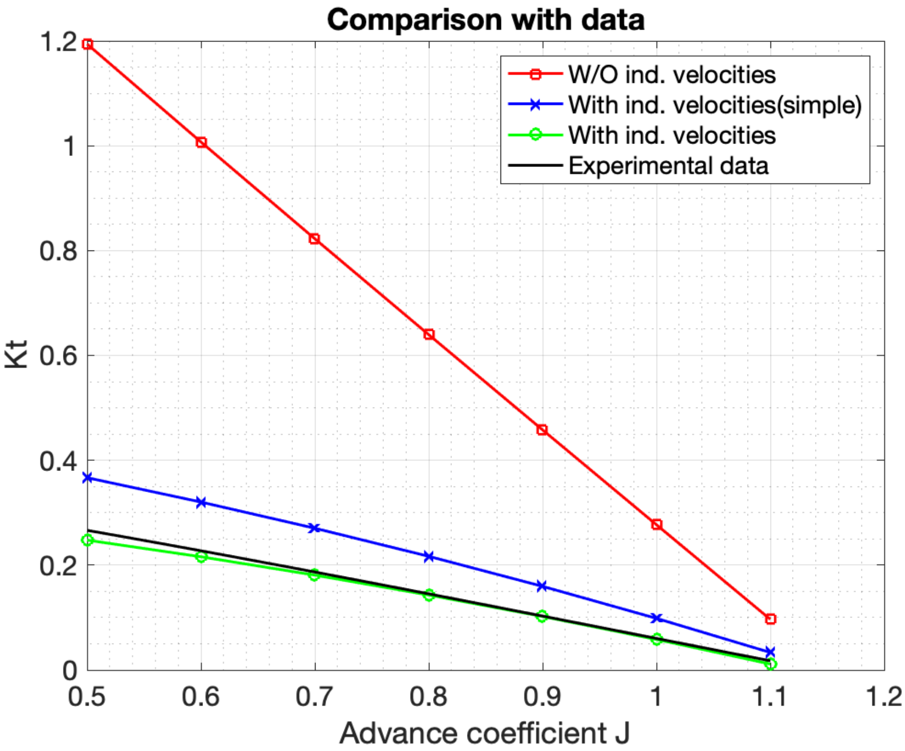


Figure 13: K_T comparison with experiment and previous result

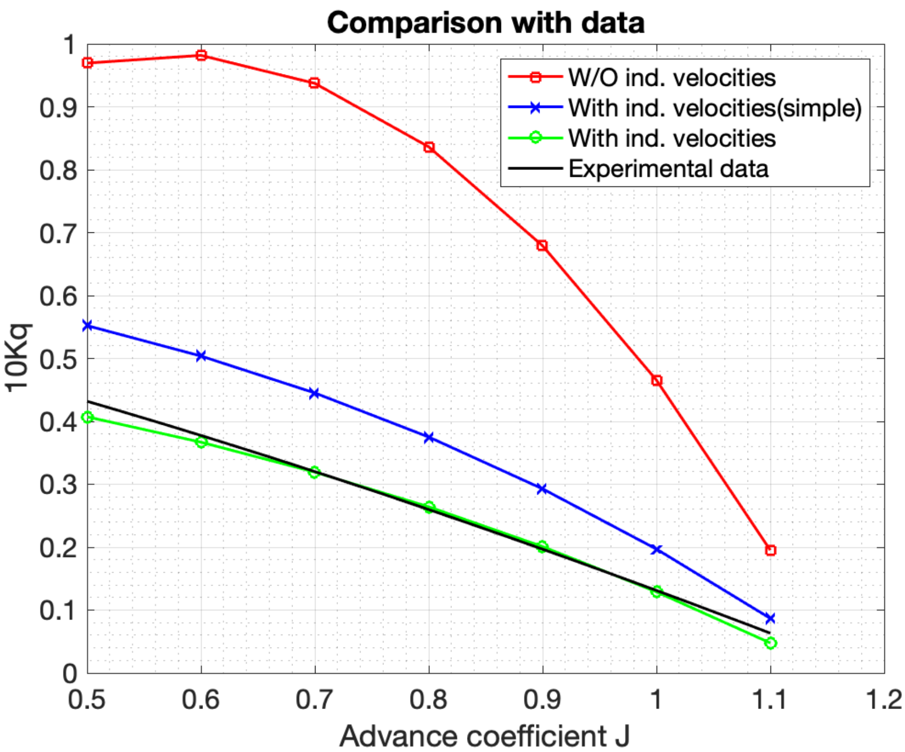


Figure 14: 10 K_Q comparison with experiment and previous result

3.5 Cavitation check

3.5.1 Main workflow

Cavitation number is an important factor to determine whether a section is in cavitation or not. It can be expressed as:

$$\sigma = \frac{p_a - p_v + \rho g(h-r)}{\frac{1}{2}\rho V_\infty^2} \tag{22a}$$

$$\sigma \leq -C_p \tag{22b}$$

$$c\sigma \leq -C_p \tag{22c}$$

After the cavitation number is determined, the pressure coefficient is hence needed to judge the cavitation. The pressure coefficient can be derived using Xfoil and those intermediate sections can be found by linear interpolation as mention above. This time, the number of sections rise to 60, so it is not suitable to manually operating Xfoil using manual clicks. Contrarily, the batch mode of Xfoil is used to be able to calculate the pressure coefficients efficiently. The main procedure of the batch mode can be listed below:

1. Import both the inflow velocity V_∞ and the angle of attack α from finest result derived before for each section.

2. Calculate the cavitation number $sigma$ given in equation eq. (22a)

3. Then launch Xfoil in batch mode and use command ***cpwr*** to write out the pressure coefficients every iteration

4. Check the cavitation criteria in eq. (22b) and if the criteria is true then the foil cavitates. A margin is introduced by using eq. (22c) for a c<1. This margin is introduced to avoid numerical chaos at root.

By comparing the σ and the negative minimum pressure coefficient derived from Xfoil, the index of which position cavitation will occur are found by eliminating the numerical noise, which then yields that the cavitation will happen starts from index 43, which is corresponding to

$$r = 0.45006m$$

It can be also converted to the ratio of the radius to half diameter which corresponds 0.75 $\frac{r}{R}$.

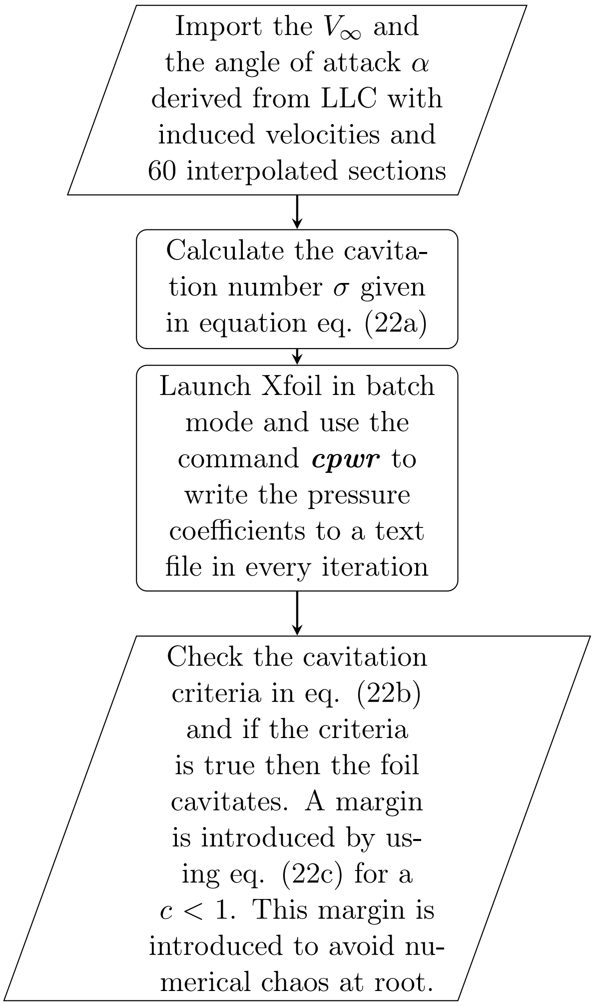


Figure 15: Workflow for checking for cavitation on the blade

4 Additional consideration

4.1 Rake, skew and hub effect

4.1.1 Rake consideration

One possible guess is that all parameters should be projected to the plane which is normal to the raked disk plane, for instance, the velocity of advance is not directly using V_A anymore, it would change to $V_A \cos\phi$,w here ϕ is the angle of back rake. Then the rest of those parameters need also to be projected to the plane which is perpendicular to the disk plane

4.1.2 Skew consideration

A possible guess is that the skew might bring about a constant circulation distribution along the radii, since the dimensions of each profile are more likely to be the same. Thus no need to use ellipitical circulation and it will lead to a more simple model. In the book "Hydrodynamics of Ship propellers" it is mentioned that "A propeller having extreme skew has efficiency equal to the corresponding propeller with conventional blade form(i.e. having the same radial circulation distribution)". This means that we just use the corresponding conventional blade form to apply the skew.

4.1.3 Hub consideration

It can be possibly considered in the calculation of induced velocities using induction factors, In this project, no consideration of hub has been taken, thus, the integral lower limit starts from the boss radius. However, this is not always the case, the hub effect can be modelled using a finite shed vortex along with other contribution from other shed vortex induced by the blades. Another possible solution to take into account the effect of a hub using a sink/source method(potential theory method).

4.2 Lifting line theory applicability range

4.2.1 Applicable propeller types

The lifting line theory can be applied to analysing the propellers with large diameter over chord ratio, which means the expanded blade ratio should be small. It can also be generalise that the propellers used that are applicable for lifting line code should be lightly loaded propellers. This also means that it is not applicable for ducted propeller which are propellers with high loading.

4.2.2 Simplification of lifting line thoery

Hish-aspect ratio foil is assumed inherently in lifting line theory, thus the span length should be much larger than the chord length. As for propeller, the large diameter over chord means small expanded area ratio as mentioned above.

4.2.3 Was it correct to analyse this propeller with a lifting line model?

Since the EAR for this propeller is only 0.55, which is relative small comapred to other series propellers. Also, it is also obvious to say that the results derived from lifting line method are correct based on the comparison with experimental data.

A Matlab code

A.1 Question 2

```
clear all
close all
clc


%% ========================== SYSTEM PARAMETERS ===========================


% Define the system properties


rho = 1025;           % water density,[kg/m3]
Pv = 2150;            % vapour pressure,[Pa]
Pa = 101325;          % atmospherical pressure,[Pa]
nu = 1.05e-6;         % kinematical viscousity,[m/s2]
Re = 6e6;             % Reynolds number


% Define the geometry of the propeller

```

```
D = 1.2;               % diameter of the propeller,[m]
z = 3;                 % number of blades
EAR = 0.5;             % expanded blade area ratio
P_D = 1.05;            % pitch to diameter ratio
J = 0.5:0.1:1.1;       % advance coefficient
ksi = 0.5;             % numerical damping


%% ======================== STEP 1: INIALISATION ===========================


% Import the geometry


Geometry = table2array(readtable('Geometry.txt'));
CL = xlsread('Lift coefficient.xlsx');
CL = [CL(:,2); 0];             % add the zero lift coefficient at the tip
r_R = Geometry(:,1);           % radius to half diameter ratio
c_D = Geometry(:,2);           % chord length to diameter ratio
t_D = Geometry(:,3);           % thickness to diameter ratio
P_D = Geometry(:,4);           % pitch to diameter ratio
r = r_R * D/2;                 % ratius of each station
c = c_D * D;                   % chord length of each station
t = t_D * D;                   % thickness of each station
P = P_D * D;                   % pitch of each section


% Calculate the infinite speed at r/R = 0.7


index = find( r_R == 0.7 );
V_inf_r7 = Re * nu / c(index);  % apparent speed at r/R = 0.7


% Create a local coordinate system


% Calculate the rotational speed


for i = 1 : length(J)          % rotation speed at 0.7 r/R
    rps(i) = V_inf_r7 / sqrt( ( J(i) * D )^2 + ( 2 * pi * r(index) )^2 );
end


% Calculate the inflow velocities


for i = 1 : length(J)
    V(i) = J(i) * D * rps(i);
```

```
end


% Calculate the geometrical pitch angle


for i = 1 : length(r)
    phi(i) = atan( P(i) / 2 / pi / r(i) );
end


% Calculate the hydrodynamic pitch angle


for i = 1 : length(r)
    for j = 1 : length(J)
        beta(i,j) = atan( V(j) / ( 2 * pi * r(i) * rps(j) ) );
    end
end


% Calculate the apparent angle of attack


for i = 1 : length(r)
    for j = 1 : length(J)
        alpha(i,j) = phi(i) - beta(i,j);
    end
end


% Calculate new lift coefficient


for i = 1 : length(r)
    for j = 1 : length(J)
        Cl(i,j) = 2 * pi * alpha(i,j) + CL(i);
    end
end


% Calculate the viscous drag coefficient


for i = 1 : length(r)
    for j = 1 : length(J)
        V_inf(i,j) = sqrt( V(j)^2 + (2 * pi * r(i) * rps(j))^2 );
        Ren(i,j) = V_inf(i,j) * c(i) / nu;
        Cf(i,j) = 0.075 / ( log10(Ren(i,j)) - 2 )^2;
        Cdv(i,j) = 2 * Cf(i,j) * ( 1 + 2 * ( t(i) / c(i) ) + 60 ...
            * ( t(i) / c(i) )^4 ) * ( 1 + Cf(i,j)^2 / 8 );
```

```
    end
    if i == length(r)
        Cdv(i,:) = 0;
    end
end


% Calculate the circulation distribution


for i = 1 : length(r)
    for j = 1 : length(J)
        Gamma(i,j) = 0.5 * z * Cl(i,j) * V_inf(i,j) * c(i);
    end
end


%% ======================== STEP 2: POSTPROCESSING =======================


% Calculate the thrust and torque coefficient


for i = 1 : length(r)

    if i == 1
        dr = r(i+1) - r(i);
    elseif i == length(r)
        dr = r(i) - r(i-1);
    else
        dr = ( r(i+1) - r(i-1) ) / 2;
    end

    for j = 1 : length(J)


        % Calculate the ideal thrust and tangential force at radius r


        dTi(i,j) = rho * Gamma(i,j) * 2 * pi * r(i) * rps(j);
        dKi(i,j) = rho * Gamma(i,j) * V(j);


        % Calculate the thrust and tangential force due to drag


        dTD(i,j) = 0.5 * rho * V_inf(i,j)^2 * c(i) * Cdv(i,j) * ...
            z * sin( beta(i,j) );       % reduction
        dKD(i,j) = 0.5 * rho * V_inf(i,j)^2 * c(i) * Cdv(i,j) * ...
            z * cos( beta(i,j) );       % increase

```

```
        % Calculate the resulting thrust and torque


        dT(i,j) = dTi(i,j) - dTD(i,j);
        dQ(i,j) = ( dKi(i,j) + dKD(i,j) ) * r(i);


    end


end


% Calculate the total thrust and torque by trapzoidal rule


for i = 1 : length(J)
    T(i) = trapz( r, dT(:,i) );
    Q(i) = trapz( r, dQ(:,i) );
end


% Calculate the thrust and torque coefficient


for i = 1 : length(J)


    Kt(i) = T(i) / rho / rps(i)^2 / D^4;
    Kq(i) = Q(i) / rho / rps(i)^2 / D^5;


end


% Export data to compare


fid = fopen('LLC_2_Kt.txt','w');
fprintf(fid, '%f \n', Kt');
fclose(fid);
fid = fopen('LLC_2_Kq.txt','w');
fprintf(fid, '%f \n', Kq');
fclose(fid);



% Import experimental data


```

```
DATA_EX = table2array(readtable('ExperimentalData.txt'));
DATA_EX(1,:) = [];
DATA_EX = str2double(DATA_EX);
J_EX = DATA_EX(:,1);              % advance coefficient ( experiment )
Kt_EX = DATA_EX(:,2);             % thrust coefficient ( experiment )
Kq_EX = DATA_EX(:,3);             % torque coefficient ( experiment )
Eta_EX = DATA_EX(:,4);            % efficiency ( experiment )


%% ========================== VISUALISATION ============================


% Compare with the experimental data


figure()
plot(J,Kt,'rx-','linewidth',1.5)
hold on
plot(J_EX,Kt_EX,'go-','linewidth',1.5)
grid on; grid minor; box on
legend('W/O ind. velocities','Experiment','location','NE')
xlabel('J')
title('Kt')
set(gca,'fontsize',15)
hold off
figure()
plot(J,10*Kq,'rx-','linewidth',1.5)
hold on
plot(J_EX,10*Kq_EX,'go-','linewidth',1.5)
grid on; grid minor; box on
legend('W/O ind. velocities','Experiment','location','NE')
xlabel('J')
title('10Kq')
set(gca,'fontsize',15)


%% ============================== END ===================================
```

A.2 Question 3

```
clear all
close all
clc


%% ========================= SYSTEM PARAMETERS ==========================


% Define the system properties


rho = 1025;          % water density,[kg/m3]
```

```
Pv = 2150;           % vapour pressure,[Pa]
Pa = 101325;         % atmospherical pressure,[Pa]
nu = 1.05e-6;        % kinematical viscousity,[m/s2]
Re = 6e6;            % Reynolds number



% Define the geometry of the propeller


D = 1.2;             % diameter of the propeller,[m]
z = 3;               % number of blades
EAR = 0.5;           % expanded blade area ratio
P_D = 1.05;          % pitch to diameter ratio
J = 0.5:0.1:1.1;     % advance coefficient
ksi = 0.5;           % numerical damping


%% ======================= STEP 1: INIALISATION ==========================


% Import the geometry


Geometry = table2array(readtable('Geometry.txt'));
CL = xlsread('Lift coefficient.xlsx');
CL = [CL(:,2); 0];           % add the zero lift coefficient at the tip
r_R = Geometry(:,1);         % radius to half diameter ratio
c_D = Geometry(:,2);         % chord length to diameter ratio
t_D = Geometry(:,3);         % thickness to diameter ratio
P_D = Geometry(:,4);         % pitch to diameter ratio
r = r_R * D/2;               % ratius of each station
c = c_D * D;                 % chord length of each station
t = t_D * D;                 % thickness of each station
P = P_D * D;                 % pitch of each section


% Calculate the infinite speed at r/R = 0.7


index = find( r_R == 0.7 );
V_inf_r7 = Re * nu / c(index);  % apparent speed at r/R = 0.7


% Calculate the rotational speed


for i = 1 : length(J)        % rotation speed at 0.7 r/R
    rps(i) = V_inf_r7 / sqrt( ( J(i) * D )^2 + ( 2 * pi * r(index) )^2 );
end

```

```
% Calculate the inflow velocities


for i = 1 : length(J)
    V(i) = J(i) * D * rps(i);
end


% Calculate the geometrical pitch angle


for i = 1 : length(r)
    phi(i) = atan( P(i) / 2 / pi / r(i) );
end


% Intitialize the induced velocities


Ua = zeros( length(r), length(J) );
Ut = zeros( length(r), length(J) );


%% ======================== STEP 2: ITERATION ============================


% Define iterative parameters


MaxIteration = 1000;     % maximum iteration number
nIteration = 0;          % iteration counter
MaxError = 1e-4;         % error tolerance


% Make a loop to iterate


for i = 1 : length(r)

    for j = 1 : length(J)

        % Assume an initial value for the circulation at each section

        gamma = 0;
        error = 1;

        while ( ( error > MaxError )&&( nIteration < MaxIteration ) )
            tic

            % Compute initial tangential induced induced velocities
```

```
            Ut(i,j) = gamma / 2 / pi / r(i);

            % Calculate the axial induced velocity

            Ua(i,j) = -V(j) + sqrt( ( V(j) )^2 + Ut(i,j) * ( 4 * pi*r(i)...
                * rps(j) - Ut(i,j) ) );

            % Compute the hydrodynamic pitch angle

            beta(i,j) = atan( (V(j) + Ua(i,j) / 2) / ( 2 * pi * r(i) * ...
                rps (j) - Ut(i,j) / 2 ) );

            % Compute the infinite velocities

            V_inf(i,j) = sqrt( (V(j) + Ua(i,j) / 2 )^2 + ( 2 * pi *r(i)*...
                rps (j) - Ut(i,j) / 2 )^2 );

            % Calculate the apparent angle of attack

            alpha(i,j) = phi(i) - beta(i,j);

            % Calculate new lift coefficient

            Cl(i,j) = 2 * pi * alpha(i,j) + CL(i);

            % Calculate new circulation

            gamma_new = 0.5 * z * Cl(i,j) * V_inf(i,j) * c(i);

            % Judge the convergence

            error = abs( gamma_new - gamma );
            nIteration = nIteration + 1;
            gamma = gamma + ksi * ( gamma_new - gamma );

            toc
        end

        Gamma(i,j) = 0.5 * ( gamma + gamma_new );

    end

end


%% ======================== STEP 3: POSTPROCESSING ======================


% Calculate the viscous drag coefficient

```

```
for i = 1 : length(r)
    for j = 1 : length(J)
        Ren(i,j) = V_inf(i,j) * c(i) / nu;
        Cf(i,j) = 0.075 / ( log10(Ren(i,j)) - 2 )^2;
        Cdv(i,j) = 2 * Cf(i,j) * ( 1 + 2 * ( t(i) / c(i) ) + 60 * ...
            ( t(i) / c(i) )^4 ) * ( 1 + Cl(i,j)^2 / 8 );
    end
    if i == length(r)
        Cdv(i,:) = 0;
    end
end


% Calculate the thrust and torque coefficient


for i = 1 : length(r)
    if i == 1
        dr = r(i+1) - r(i);
    elseif i == length(r)
        dr = r(i) - r(i-1);
    else
        dr = ( r(i+1) - r(i-1) ) / 2;
    end
    for j = 1 : length(J)


        % Calculate the ideal thrust and tangential force at radius r


        dTi(i,j) = rho * Gamma(i,j) * ( 2 * pi * r(i) * rps(j) ...
            - 0.5 * Ut(i,j) );
        dKi(i,j) = rho * Gamma(i,j) * ( V(j) + 0.5 * Ua(i,j) );


        % Calculate the thrust and tangential force due to drag


        dTD(i,j) = 0.5 * rho * V_inf(i,j)^2 * c(i) * Cdv(i,j) * z ...
            * sin( beta(i,j) );       % reduction
        dKD(i,j) = 0.5 * rho * V_inf(i,j)^2 * c(i) * Cdv(i,j) * z ...
            * cos( beta(i,j) );       % increase


        % Calculate the resulting thrust and torque


        dT(i,j) = dTi(i,j) - dTD(i,j);
        dQ(i,j) = ( dKi(i,j) + dKD(i,j) ) * r(i);

```

```
    end


end


% Calculate the total thrust and torque by trapzoidal rule


for i = 1 : length(J)
    T(i) = trapz( r, dT(:,i) );
    Q(i) = trapz( r, dQ(:,i) );
end


% Calculate the thrust and torque coefficient


for i = 1 : length(J)


    Kt(i) = T(i) / rho / rps(i)^2 / D^4;
    Kq(i) = Q(i) / rho / rps(i)^2 / D^5;


end


% Export data to compare


fid = fopen('LLC_3_Kt.txt','w');
fprintf(fid,'%f \n',Kt);
fclose(fid);

fid = fopen('LLC_3_Kq.txt','w');
fprintf(fid,'%f \n',Kq);
fclose(fid);


%% ========================== VISUALISATION ==============================


% Import experimental data


DATA_EX = table2array(readtable('ExperimentalData.txt'));
DATA_EX(1,:) = [];
DATA_EX = str2double(DATA_EX);
J_EX = DATA_EX(:,1);              % advance coefficient ( experiment )
```

```
Kt_EX = DATA_EX(:,2);              % thrust coefficient ( experiment )
Kq_EX = DATA_EX(:,3);              % torque coefficient ( experiment )
Eta_EX = DATA_EX(:,4);             % efficiency ( experiment )


% Import results from quesiton 2


Kt_2 = dlmread('LLC_2_Kt.txt');
Kq_2 = dlmread('LLC_2_Kq.txt');


% Compare with the experimental data and results from question 2


figure();
p1 = plot(J,Kt,'b-x','linewidth',1.5);
hold on
p2 = plot(J,Kt_2,'r-s','linewidth',1.5);
p3 = plot(J_EX,Kt_EX,'g-o','linewidth',1.5);
grid on; grid minor; box on
xlabel('J')
title('Kt')
set(gca,'fontsize',15)
figure();
p4 = plot(J,10*Kq,'b-x','linewidth',1.5);
hold on;
p5 = plot(J,10*Kq_2,'r-s','linewidth',1.5);
p6 = plot(J_EX,10*Kq_EX,'g-o','linewidth',1.5);
grid on; grid minor; box on
legend([p2 p1 p3],'W/O ind. velocities','With ind. velocities',...
    'Experiment data','location','NE')
legend([p5 p4 p6],'W/O ind. velocities','With ind. velocities',...
    'Experiment data','location','NE')
xlabel('J')
title('10*Kq')
set(gca,'fontsize',15)


%% ================================ END =====================================
```

A.3 Question 4

```
clear all
close all
clc


%% =========================== SYSTEM PARAMETERS ==========================


% Define the system property
```

```
rho = 1025;           % water density, [kg/m3]
Pv = 2150;            % vapour pressure, [Pa]
Pa = 101325;          % atmospherical pressure
nu = 1.05e-6;         % kinematical viscousity
Re = 6e6;             % Reynolds number


% Define the geometry of the propeller


D = 1.2;              % diameter of the propeller
z = 3;                % number of blades
EAR = 0.5;            % expanded blade area ratio
P_D = 1.05;           % pitch to diameter ratio
J = 0.5:0.1:1.1;      % advance coefficient
ksi = 0.001;          % numerical damping
N = 60;               % number of elements
time = [];            % convergence time

% Import the geometry


Geometry = table2array(readtable('Geometry.txt'));
CL = xlsread('Lift coefficient.xlsx');
CL = [CL(:,2); 0];            % add the zero lift coefficient at the tip
r_R = Geometry(:,1);          % radius to half diameter ratio
c_D = Geometry(:,2);          % chord length to diameter ratio
t_D = Geometry(:,3);          % thickness to diameter ratio
P_D = Geometry(:,4);          % pitch to diameter ratio
r = r_R * D/2;                % ratius of each station
c = c_D * D;                  % chord length of each station
t = t_D * D;                  % thickness of each station
P = P_D * D;                  % pitch of each section


%% ======================= STEP 1: DISCRITIZATION =======================


% ---------------------- infinite speed at r/R = 0.7 ----------------------


index = find( r_R == 0.7 );      % index of corresponding r/R = 0.7
V_inf_r7 = Re * nu / c(index);   % apparent speed at r/R = 0.7


% --------------------------- rotational speed ---------------------------


% rotation speed based on constant apparent speed at 0.7 r/R
```

```
for i = 1 : length(J)
    rps(i) = V_inf_r7 / sqrt( ( J(i) * D )^2 + ( 2 * pi * r(index) )^2 );
end


% ---------------------------- inflow velocities ---------------------------


for i = 1 : length(J)
    V(i) = J(i) * D * rps(i);
end


%% ========================= STEP 2: INTERPOLATION ========================


% ------------------------------- data sets -------------------------------


CL_data = CL;           % lift coefficient from data
P_data = P;             % pitch from data
c_data = c;             % chord length from data
t_data = t;             % thickness from data


%% ======================= STEP 3: CONVERGENCE TEST =======================


for n = 1 : length(J)

    tic

    % Regenerate new radii

    r_new = zeros( N + 1, 1 );              % initialize the element radii
    dr = ( r(end) - r(1) ) / N;             % element spacing,
    r_new = r(1) : dr : r(end);             % updated interpolated radii
    r_new = r_new';                         % transpose to avoid misproduct

    % Interpolate the data

    CL = interp1( r, CL_data, r_new )';     % lift coefficient interpolation
    P = interp1( r, P_data, r_new )';       % pitch interpolation
    c = interp1( r, c_data, r_new )';       % chord length interpolation
    t = interp1( r, t_data, r_new )';       % thickness interpolation


% --------------------------- Initialization ------------------------------

```

```
% initialize the geometrical pitch angle

phi = zeros( length(r_new), 1 );

for i = 1 : length(r_new)
    phi(i) = atan( P(i) / 2 / pi / r_new(i) );
end

% Initialize the circulation

Gamma = zeros( length(r_new),1 );        % initial circulation
dGamma = zeros( length(r_new), 1 );      % initial derivarive of circulation
Gamma_new = zeros( length(r_new), 1 );  % initial updated circulation

% Initialize the induced velocities

Ua = zeros( length(r_new), 1 );              % initial induced velocity
Ut = zeros( length(r_new), 1 );              % initial induced velocity
dUa = zeros( length(r_new), length(r_new) );% initial derivative of induced
dUt = zeros( length(r_new), length(r_new) );% initial derivative of induced

% Initialize the parameters

V_inf = zeros( length(r_new), 1 );
alpha = zeros( length(r_new), 1 );
Cl = zeros( length(r_new), 1 );
beta = zeros( length( r_new), 1 );

% Initialize the induction factor

i_A = zeros( length(r_new), length(r_new) );
i_T = zeros( length(r_new), length(r_new) );

% Initialize the postprocessing parameter

dTi = zeros( length(r_new), 1 );    % ideal thrust
dQi = zeros( length(r_new), 1 );    % ideal tangential force
dTD = zeros( length(r_new), 1 );    % corrected thrust
dQD = zeros( length(r_new), 1 );    % corrected tangential force
dT = zeros( length(r_new), 1 );     % resulting thrust
dQ = zeros( length(r_new), 1 );     % resulting torque
Ren = zeros( length(r_new), 1 );    % Reynolds number
Cf = zeros( length(r_new), 1 );     % frictional coefficient
Cdv = zeros( length(r_new), 1 );    % viscous drag coefficient

% Initialize the hydrodynamic pitch angle

for i = 1 : length(r_new)
    beta(i) = atan( ( V(n) + Ua(i) / 2 ) / ( 2 * pi * ...
        r_new(i) * rps(n) - Ut(i) / 2 ) );
end
```

```
    % Initialize the iterative parameters

    MaxIteration = 10000;   % maximum iteration number
    ErrorAllowed = 1e-3;    % error tolerance
    nIteration = 0;         % iteration number
    error = 1;              % initial error

% ------------------------ Loop to find converged results -----------------

    while( ( error > ErrorAllowed )&&( nIteration < MaxIteration ) )

% ------------------------- Induction factors -----------------------------

        for i = 2 : length(r_new) - 1
            for j = 2 : length(r_new) - 1
                [i_A(i,j), i_T(i,j)] = InductionFactors(r_new(j), ...
                    r_new(i), beta(j), z);
            end
        end

% ----------------------- 1st derivative of circulation -------------------

        dGamma = fnval( fnder( csapi( r_new, Gamma ), 1 ), r_new ); % cubic spli

% ---------------------------- Induced velocities -------------------------

        for i = 2 : length(r_new) - 1
            for j = 2 : length(r_new) - 1
                dUa(i,j) = i_A(i,j) / 2 / pi * dGamma(j);
                dUt(i,j) = i_T(i,j) / 2 / pi * dGamma(j);
            end
        end

        for i = 2 : length(r_new) - 1
            Ua(i) = SingularIntegration(r_new, dUa(i,:), r_new(i));
            Ut(i) = SingularIntegration(r_new, dUt(i,:), r_new(i));
        end

% ---------------------------- Update results -----------------------------

        % Update the hydrodynamic pitch angle

        for i = 2 : length(r_new) - 1
            beta(i) = atan( ( V(n) + Ua(i) / 2 ) / ( 2 * pi * ...
                r_new(i) * rps(n) - Ut(i) / 2 ) );
        end

        % Update the system parameters

        for i = 1 : length(r_new)
```

```
            V_inf(i) = sqrt( ( V(n) + Ua(i)/2 )^2 + ( 2 * pi ... % infinite velo
                * r_new(i) * rps(n) - Ut(i)/2 )^2 );
            alpha(i) = phi(i) - beta(i);                            % apparent angle
            Cl(i) = 2 * pi * alpha(i) + CL(i);                      % lift coefficie
            Gamma_new(i) = z * 0.5 * V_inf(i) * c(i) * Cl(i);      % updated circul
        end

        p = polyfit(r_new,Gamma_new,2);
        Gamma_new = polyval(p,r_new);
        Gamma_new(1) = 0;         % boundary condtion
        Gamma_new(end) = 0;       % boundary condtion

% -------------------------- Convergence judgement ------------------------

        error = max( abs( Gamma_new - Gamma ) / max ( abs( Gamma ) ) );
        nIteration = nIteration + 1;

% -------------------------- Update the circulation -----------------------

        Gamma = Gamma + ksi * ( Gamma_new - Gamma );     % updated circulation us
        Gamma(1) = 0;         % boundary condition
        Gamma(end) = 0;       % boundary condition

    end

    Gamma = 0.5 * ( Gamma_new + Gamma );

% ------------------------------- Loop end --------------------------------


%% ======================== STEP 4: POSTPROCESSING ======================


% ------------------------- Viscous drag coefficient ----------------------


    for i = 1 : length(r_new)
        Ren(i) = V_inf(i) * c(i) / nu;                            % Reynolds number
        V_inf(i) = sqrt( ( V(n) + Ua(i) / 2 )^2 + ( 2 * pi ... % inifinite veloc
            * r_new(i) * rps(n) - Ut(i) / 2 )^2 );
        Cf(i) = 0.075 / ( log10( Ren(i) ) - 2 )^2; % frictional coefficient ITTC
        Cdv(i) = 2 * Cf(i) * ( 1 + 2 * ( t(i) / c(i) ) + 60 * ...
            ( t(i) / c(i) )^4 ) * ( 1 + Cl(i)^2 / 8 );           % viscous drag co
        if i == length(r_new)
            Cdv(i) = 0;                                           % no drag at tip
        end
    end


% ------------------------- Thrust and torque coefficient -----------------

```

```
    for i = 1 : length(r_new)


        % Calculate the ideal thrust and tangential force at radius r


        dTi(i) = rho * Gamma(i) * ( 2 * pi * r_new(i) * rps(n) - ...
        0.5 * Ut(i) ); % ideal thrust
        dQi(i) = rho * Gamma(i) * ( V(n) + 0.5 * Ua(i) ) * r_new(i);% ideal torq


        % Calculate the thrust and tangential force due to drag


        dTD(i) = 0.5 * rho * V_inf(i)^2 * c(i) * Cdv(i) * z * ...
            sin( beta(i) );            % reduction
        dQD(i) = 0.5 * rho * V_inf(i)^2 * c(i) * Cdv(i) * z * ...
            cos( beta(i) ) * r_new(i); % increase


        % Calculate the resulting thrust and torque


        dT(i) = dTi(i) - dTD(i);
        dQ(i) = dQi(i) + dQD(i);


    end


    % Calculate the total thrust and torque by trapzoidal rule


    T = trapz( r_new, dT );    % total thrust, [N]
    Q = trapz( r_new, dQ );    % total torque, [Nm]


    % Calculate the thrust and torque coefficient


    Kt(n) = T / rho / rps(n)^2 / D^4;
    Kq(n) = Q / rho / rps(n)^2 / D^5;


    time(n) = toc


end
```

```
%% ===================== STEP 4: DATA COMPARISON ========================


% Export data


fid = fopen('LLC_4_Kt_test.txt','w');
fprintf(fid,'%f \n',Kt);
fclose(fid);


fid = fopen('LLC_4_Kq_test.txt','w');
fprintf(fid,'%f \n',Kq);
fclose(fid);


% Import experimental data


DATA_EX = table2array(readtable('ExperimentalData.txt'));
DATA_EX(1,:) = [];
DATA_EX = str2double(DATA_EX);
J_EX = DATA_EX(:,1);              % advance coefficient ( experiment )
Kt_EX = DATA_EX(:,2);             % thrust coefficient ( experiment )
Kq_EX = DATA_EX(:,3);             % torque coefficient ( experiment )
Eta_EX = DATA_EX(:,4);            % efficiency ( experiment )


% Import results from quesiton 2


Kt_2 = dlmread('LLC_2_Kt.txt');
Kq_2 = dlmread('LLC_2_Kq.txt');


% Import results from quesiton 3


Kt_3 = dlmread('LLC_3_Kt.txt');
Kq_3 = dlmread('LLC_3_Kq.txt');


%% ====================== STEP 5: VISUALISATION =========================


% ---------------------------- Plot of Kt --------------------------------


figure()
plot(J,Kt_2,'rs-','linewidth',1.5)
hold on;
```

```
plot(J,Kt_3,'bx-','linewidth',1.5)
plot(J,Kt,'go-','linewidth',1.5)
plot(J_EX,Kt_EX,'k-','linewidth',1.5)
grid on, grid minor, box on
set(gca,'fontsize',15)
xlabel('Advance coefficient J')
ylabel('Kt')
title('Comparison with data')
legend('W/O ind. velocities','With ind. velocities(simple)',...
    'With ind. velocities','Experimental data','location','NE')


% ---------------------------- Plot of 10Kq --------------------------------


figure()
plot(J,10*Kq_2,'rs-','linewidth',1.5)
hold on;
plot(J,10*Kq_3,'bx-','linewidth',1.5)
plot(J,10*Kq,'go-','linewidth',1.5)
plot(J_EX,10*Kq_EX,'k-','linewidth',1.5)
grid on, grid minor, box on
set(gca,'fontsize',15)
xlabel('Advance coefficient J')
ylabel('10Kq')
title('Comparison with data')
legend('W/O ind. velocities','With ind. velocities(simple)',...
    'With ind. velocities','Experimental data','location','NE')


%% ============================== END ====================================
```

A.4 Question 5

```
clear all
close all
clc


%% =========================== SYSTEM PARAMETERS ==========================


% Define the system property


rho = 1025;          % water density, [kg/m3]
Pv = 2150;           % vapour pressure, [Pa]
Pa = 101325;         % atmospherical pressure
nu = 1.05e-6;        % kinematical viscousity
Re = 6e6;            % Reynolds number


```

```
% Define the geometry of the propeller


D = 1.2;              % diameter of the propeller
z = 3;                % number of blades
EAR = 0.5;            % expanded blade area ratio
P_D = 1.05;           % pitch to diameter ratio
J = 0.5;              % advance coefficient
N = 60;               % number of elements
g = 9.81;
h = 1;


%% =================  First step: Discretise the blade  ===================


% Import the profile data


Geometry = table2array(readtable('Geometry.txt'));
r_R = Geometry(:,1);                       % radius to half diameter ratio
r = r_R * D/2;                             % ratius of each station
r_new = zeros( N + 1, 1 );                 % initialize the element radii
dr = ( r(end) - r(1) ) / N;                % element spacing,
r_new = r(1) : dr : r(end);                % updated interpolated radii
r_new = r_new';                            % transpose to avoid misproduct

Profile(:, [1 2]) = dlmread('foil_profile_rR0.167.txt','',1,0);
Profile(:, [3 4]) = dlmread('foil_profile_rR0.200.txt','',1,0);
Profile(:, [5 6]) = dlmread('foil_profile_rR0.300.txt','',1,0);
Profile(:, [7 8]) = dlmread('foil_profile_rR0.400.txt','',1,0);
Profile(:, [9 10]) = dlmread('foil_profile_rR0.500.txt','',1,0);
Profile(:, [11 12]) = dlmread('foil_profile_rR0.550.txt','',1,0);
Profile(:, [13 14]) = dlmread('foil_profile_rR0.600.txt','',1,0);
Profile(:, [15 16]) = dlmread('foil_profile_rR0.700.txt','',1,0);
Profile(:, [17 18]) = dlmread('foil_profile_rR0.800.txt','',1,0);
Profile(:, [19 20]) = dlmread('foil_profile_rR0.900.txt','',1,0);

Profile_U = zeros(length(Profile),length(r));
Profile_D = zeros(length(Profile),length(r));

for i = 1 : size(Profile,2)/2
    Profile_U(:,i) = Profile(:,2*i-1);
    Profile_D(:,i) = Profile(:,2*i);
end


% Extract the upper surface


PF_U = zeros(length(Profile),length(r_new));
```

```
PF_D = zeros(length(Profile),length(r_new));


% Interpolate the profile data


for i = 1 : length(r_new)
    for j = 1 : length(Profile_U)
        PF_U(j,:) = interp1(r, Profile_U(j,:),r_new);
        PF_D(j,:) = interp1(r, Profile_D(j,:),r_new);
    end
end


PF = zeros(length(Profile),2*length(r_new));
for i = 1:length(r_new)
    PF(:,2*i-1) = PF_U(:,i);
    PF(:,2*i) = PF_D(:,i);
end


% Import the previous results


alpha = dlmread('LLC_5_alpha.txt');
V_inf = dlmread('LLC_5_V_inf.txt');


% Calling Xfoil to compute the pressure coefficient in batch mode


j = 1;
for i = 1:2:2*length(r_new)
    fid = fopen('PF.dat','wt');
    fprintf(fid,'%u %u \n',[PF(:,i) PF(:,i+1)]');
    fclose(fid);

    fid = fopen('XFoil_inputs.dat', 'wt');
    fprintf(fid,['load PF.dat', '\n']); % load this profile
    fprintf(fid,'SectionProfile\n');
    fprintf(fid,'oper\n');                  % enter operating mode
    fprintf(fid,'alfa %f\n',rad2deg(alpha(j))); % enter the geometry design menu
    fprintf(fid,'cpwr CP.txt\n');           % export data
    fprintf(fid,'cpmn \n');                 % report the minimum value
    fprintf(fid,'quit');
    !xfoil.exe<XFoil_inputs.dat
    Xfoil_data = dlmread('CP.txt','',3,0);
    CP(:,j) = Xfoil_data(:,3);
    j = j+1;
end

```

```
%% ========================== Cavitation check ==========================


for i = 1: length(r_new)
    sigma(i) = ( Pa - Pv + rho * g * ( h - r_new(i) ) ) / ( 0.5 * rho * V_inf(i)
end

for i = 1 : length(r_new)
    for j = 1 : length(CP)
        if -CP(j,i) > sigma(i)
            index(i) = true;         % 1/0 [ cavitated, non-cavitated ]
        end
    end
end


%% ============================== END ===================================
```

References

[1] John P.Breslin and Poul Andersen *Hydrodynamics of Ship propellers* 1993.

[2] Steen, Sverre *TMR 4220 Naval Hydrodynamics Foil and Propeller Theory* 2014.