DEPARTAMENTO DE ENGENHARIA MECÂNICA
INSTITUTO SUPERIOR TÉCNICO

PROPELLER OPTIMIZATION INCLUDING VISCOUS DRAG EFFECTS

Projecto de Termodinâmica Aplicada

Autor: Maria Helena Ribeiro
Aluno Nº: 41615

Orientador: Prof. J.A.C. Falcão de Campos

Data: 31 de Outubro de 2002

ABSTRACT

This work addresses the problem of the optimization of the blade radial circulation distribution of a propeller including viscous drag effects. The problem is to find the optimum radial distribution of circulation on the blades for a given thrust minimizing the delivered propeller power.

The propeller model is based on the lifting line theory for a finite number of blades without hub. The solution method is base on the induction factor method of *Lerbs*, which relates the circulation distribution to the induced velocities at the lifting lines. A Fourier series method with trapezoidal integration is employed to evaluate numerically the induced velocities. The method of Lagrange multipliers is used to solve the optimization problem in the discretized form of the power equation.

The formulation was implemented in a computer code. The optimization is carried out in a number of representative examples in uniform inflow conditions. The method is compared with the standard lifting line propeller optimization, and the optimization by applying Munk's displacement theorem. Significant effects of viscous drag on the optimal distribution are found in comparison with the inviscid solution.

RESUMO

Este trabalho está direccionado no problema de optimização da distribuição radial de circulação de hélices propulsores incluindo os efeitos da viscosidade. O problema pode se colocado da seguinte forma, pretende-se encontrar uma distribuição radial óptima de circulação que minimize a potência fornecida ao hélice para um determinado valor da força propulsiva.

O modelo teórico é baseado na teoria da linha sustentadora para um número finito de pás desprezando os efeitos do cubo. A solução do método é baseada em: no método dos factores de indução de *Lerbs* por forma a relacionar a distribuição de circulação com as velocidades induzidas na linha sustentadora. As séries de Fourier com integração trapezoidal são aplicadas, de modo a estimar as velocidades induzidas. O método dos multiplicadores de Lagrange é utilizado para resolução do problema da optimização na forma discretizada.

Depois de elaborada a formulação procedeu-se à sua implementação num programa de cálculo em linguagem Fortran. A optimização é ilustrada através de exemplos representativos em condições de escoamento uniforme. O método é comparado com o método de optimização clássico da linha sustentadora e o método de optimização com aplicação do teorema de Munk's. Na distribuição óptima de circulação são encontrados efeitos significativos da viscosidade em comparação com a solução invíscida.

AGRADECIMENTOS

Em primeiro lugar gostaria de agradecer ao Eng.º João Baltazar, pela disponibilidade sempre demonstrada.

Gostaria de agradecer ao Prof. José Alberto Caiado Falcão de Campos pela sua orientação e todo o apoio dado.

Por fim, mas não por último, toda a paciência e ajuda dada por Ricardo Correia.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. NOMENCLATURE

c	Chord length
C_D	Drag coefficient
C_L	Lift coefficient
C_P	Power coefficient
C_T	Thrust coefficient
F_a	Axial force component
F_t	Tangential force component
G	Nondimensional circulation
J_0	Advance velocity coefficient (based on ship speed)
$M_{a,t}$	Axial and tangential induction velocities matrices, respectively
N	Number of internal points
Q	Torque
R	Propeller radius
r	Radius (cylindrical coordinates)
R_h	Hub radius
T	Thrust
U	Axial inflow velocity component
u_a	Axial induced velocity
u_t	Tangential induced velocity
V	Resultant approaching velocity to the propeller blade section
V_0	Reference velocity
V_s	Ship speed
$w(x)$	Local wake fraction
w_T	Taylor effective wake fraction
x	Nondimensional radius
x_h	Nondimensional hub radius
Z	Number of blades
β	Advance pitch angle
β_i	Hydrodynamic pitch angle

β_v	Pitch angle of helical vortices
λ	Advance velocity coefficient
ρ	Fluid density
Γ	Circulation
ω	Angular velocity
ε	Sectional drag-to-lift ratio
η	Propeller efficiency

2. INTRODUCTION

With the constant growth of the power of ships, it was indispensable to optimize propellers. In the beginning, it was sufficient improve the propulsive efficiency but in our days it is also very important the phenomenon of cavitation and associate effects as vibrations and noise.

The efficiency of the marine propeller has been the issue of a various theoretical and experimental investigations along the years. The first practical theory of propellers was developed by Rankine in 1865. He developed a momentum theory the so called actuator disc theory. This theory does not furnish information on blade geometry but provides the ideal propeller efficiency. Later on, 1878 William Froude developed the blade element theory, which considers the two dimensional flow around each blade radial cross-section. On 1889, R.E. Froude extended the momentum theory introducing the induced velocities for a propeller with infinite number of blades.

On 1919, Albert Betz demonstrated that a propeller with an infinite number of blades has an optimum efficiency when the pitch of the trailing vortices is constant in the radial direction. Ludwig Prandtl and Albert Betz, in 1927, applied the Kutta-Joukowski circulation theorem. They estimated optimum propellers with a minimum induced loss.

A few years later, in 1927, Goldstein develops the first theory for propeller design. He established the relation between the induced tangential velocity at the propeller blade of a propeller with finite and infinite number of blades. He is the author of the Goldstein's factors which are a function of number of blades, radius and vortex pitch.

The work of Prandtl and Betz, led Lerbs to develop the propeller circulation theory. In 1952, Lerbs applies the lifting line theory to the propellers and use the induction factor method to calculate the induced velocities in ideal flow. These factors were a big advance for the reason that they made the design with wake-adapted propellers possible.

Today the design methods in used are based on lifting line and lifting surface theories. The methods that use these theories are formulated as design codes, i.e., they give geometry. In design, are required to develop the diameter, pitch, camber and blade section to carry out a required thrust at minimum power.

In theory, the propeller analysis for optimum propeller efficiency including viscous drag and cavity drag effects was done by Yim in 1976. He applies the Munk's displacement theorem in the lifting line formulation.

With the present work we attempt to solve numerically the complete optimization problem, to innovate the methods used instead of applying the Munk's displacement theorem or the classical optimization criteria. The optimization problem is solved with no more approximations than the ones inherent to the numerical method.

To formulate the problem we use the following methods:

- the induction factor method of *Lerbs* is employed to relate the circulation distribution to the induced velocities at the lifting lines;
- the Fourier series method with trapezoidal integration is applied to evaluate numerically the induced velocities;
- the method of Lagrange multipliers is used to solve the optimization problem in the discretized form of the power equation.

The formulation was made first in inviscid conditions and later the viscous drag effects were included. For both cases are presented a number of representative examples for the results obtained by the programs implemented. Those results are compared with the results presented by other authors. A comparison is made between the results with and without viscous drag.

3. PROPELLER LIFTING LINE THEORY

The mathematical and theoretical methods developed along years for a propeller rely on the same theoretical basis as that of an aerodynamic wing. The motion of a propeller blade is analogous to a rolling wing, so we can apply the lifting line method usually used on wings to a propeller. In this theory, the propeller blades are replaced by a system of bound vortices along the blades and helical trailing vortices in the slipstream (see **fig. 1**).

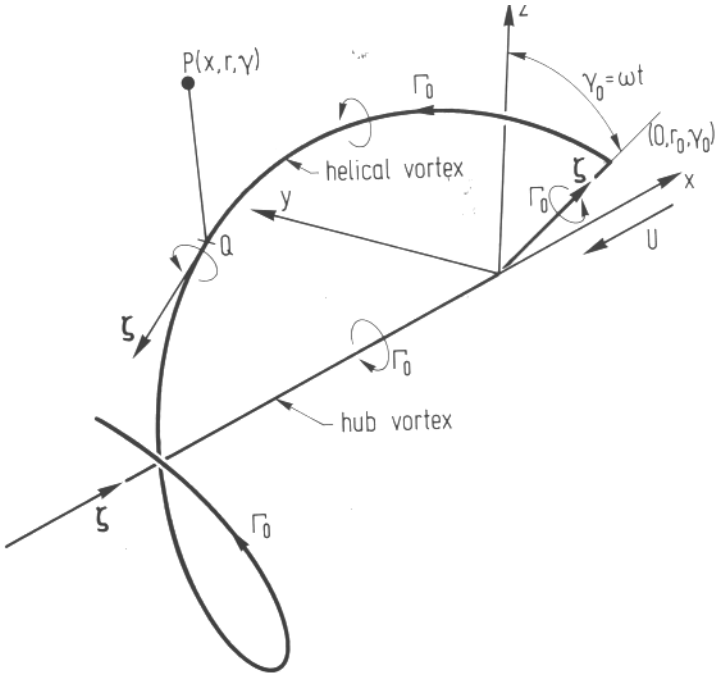

Fig. 1 – Scheme of vortices according to the propeller lifting line theory.

The strength of the trailing vortices is related to the variation of circulation around each blade section. The boundary vortices and trailing vortices induce a velocity field (the induced velocities) that may be determined by the law of Biot-Savart.

In the case of a lightly loaded propeller, the induced velocities could be neglected since the intensity of the trailing vortices is small. Therefore, the helical pitch angle equals the advance pitch angle. In the moderately loaded assumption, the induced velocities are no longer neglected and the problem turns to be more complex. In this situation the strength of the trailing vortices are significant and the helical pitch angle equals the hydrodynamic pitch angle.

4. MATHEMATICAL FORMULATION OF THE INVISCID PROBLEM

For the inviscid problem including induced velocities, we have the following diagrams (**fig. 2**) for the velocity triangles and forces acting on a propeller blade section:

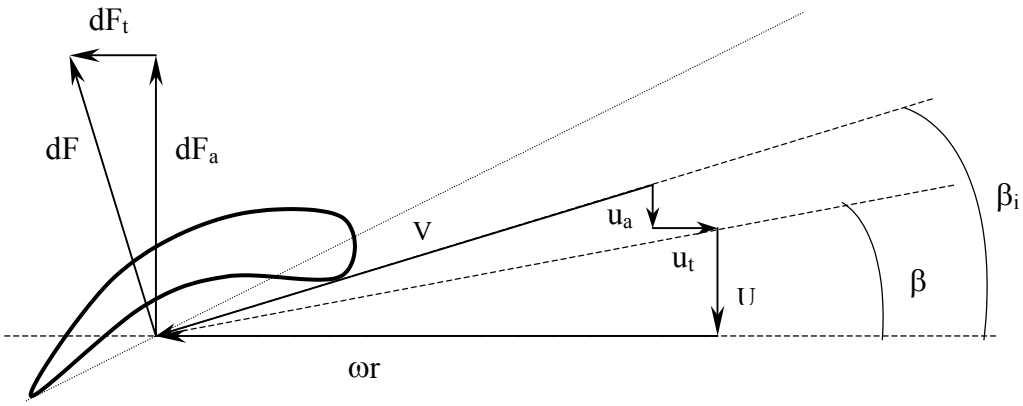

Fig. 2 – Forces acting on a blade section (inviscid fluid)

From Kutta-Joukowsky law the elemental thrust force dT and the torque dQ acting on the blade section are given by:

$$dT = dF_a = \rho \, \Gamma \, V \cos\beta_i \, dr \quad (1)$$

$$dQ = r dF_t = \rho \, \Gamma \, V \sin\beta_i \, r dr \quad (2)$$

where ρ is the fluid density, Γ the circulation, V the resultant velocity to the blade section, β_i the hydrodynamic pitch angle and r the radius (cylindrical coordinate along blade span).

From the triangle, we have:

$$V\sin\beta_i = U(r) + u_a(r) \quad (3)$$

$$V\cos\beta_i = \omega r - u_t(r) \quad (4)$$

where U is the inflow advance velocity, ω the propeller angular velocity and u_a and u_t the axial and tangential induced velocities at the blade section.

In agreement with the concept of the induction factors, the induced axial and tangential velocities at the lifting line can be written in the following form:

$$u_{a,t} = \frac{1}{4\pi} \int_{R_h}^{R} \Gamma' \frac{i_{a,t}}{r - r'} dr' \tag{5}$$

and

$$\Gamma' = \frac{d\Gamma}{dr'}$$

R_h is the hub radius, R the propeller radius and i_a and i_t are the axial and tangential induction factors, respectively. These factors are nondimensional variables that traduce the effect of helical vortex and a straight and semi-infinite vortex (parallel to propeller axis and with origin in same point) in the induced velocities. The induction factors depend on the radius r, on the radius of free helical vortex r', on the number of blades Z and on the pitch angle of helical vortices β_v.

Integrating between the hub radius and the tip expressions **(1)** and **(2)**, and summing over the number of blades:

$$T = \rho Z \int_{R_h}^{R} \Gamma V \cos\beta_i dr \tag{6}$$

$$Q = \rho Z \int_{R_h}^{R} \Gamma V \sin\beta_i r dr \tag{7}$$

Introducing the nondimensional thrust and power coefficients given by:

$$C_T = \frac{T}{\frac{1}{2}\rho V_0^2 \pi R^2} \tag{8}$$

$$C_P = \frac{\omega Q}{\frac{1}{2}\rho V_0^3 \pi R^2} \tag{9}$$

where V_0 is the reference velocity, we obtain:

$$C_T = 4Z \int_{x_h}^{1} \frac{V(x)}{V_0} G(x) \cos\beta_i dx \tag{10}$$

$$C_P = \frac{4Z}{\lambda}\int_{x_h}^{1}\frac{V(x)}{V_0}G(x)\sin\beta_i x dx \tag{11}$$

where $\lambda = {V_0}/{\omega R}$ is the advance velocity coefficient, x the nondimensional radius (x = r/R), x_h the nondimensional hub radius ($x_h = R_h/R$) and G the nondimensional circulation given by:

$$G(x) = \frac{\Gamma(x)}{2\pi R V_0} \tag{12}$$

Introduction a change of variable:

$$x = \frac{1}{2}(1+x_h) - \frac{1}{2}(1-x_h)\cos\theta \tag{13}$$

we can transform equations **(10)** and **(11)** into:

$$C_T = 2Z(1-x_h)\int_0^{\pi}\frac{V(\theta)}{V_0}G(\theta)\cos\beta_i \sin\theta d\theta \tag{14}$$

$$C_P = \frac{2Z(1-x_h)}{\lambda}\int_0^{\pi}\frac{V(\theta)}{V_0}G(\theta)\sin\beta_i x(\theta)\sin\theta d\theta \tag{15}$$

To evaluate induced velocities from **(5)** we expand the circulation distribution G(θ) in a odd Fourier series assuming that the circulation vanishes at the hub and tips. Trigonometric interpolation is used to evaluate the Fourier coefficients. After some reduction we obtain [2],

$$\frac{u_{a,t}(\theta_i)}{V_0} = \sum_{j=0}^{N} G(\theta_j) M_{a,t}(\theta_i,\theta_j) \tag{16}$$

where $\theta_i = i\pi / (N+1)$, for i = 0, 1, …, N and $M_{a,t}(\theta_i,\theta_j)$ are the induction matrices. These are given by:

$$M_{a,t}(\theta_i,\theta_j) = \frac{1}{1-x_h}\sum_{k=0}^{N+1} i_{a,t}(\theta_i,\theta_j)A(\theta_i,\theta_k,\theta_j) \tag{17}$$

where $A(\theta_i, \theta_k, \theta_j)$ is a matrix which depends only on the number of points N. It is seen that the induction matrices are functions of the helical vortex pitch angle, of the hub radius and number of blades through the induction factor.

Equations **(14)** and **(15)** are easily evaluated by trapezoidal rule in terms of the discrete values of circulation $G(\theta_i)$ in the form:

$$C_T = \frac{2\pi \cdot Z(1-x_h)}{N+1} \sum_{i=0}^{N+1} G(\theta_i) \frac{(\omega R x_i - u_t(\theta_i))}{V_0} \sin\theta_i \tag{18}$$

$$C_P = \frac{2\pi \cdot Z(1-x_h)}{(N+1)\lambda} \sum_{i=0}^{N+1} G(\theta_i) \frac{(U(\theta_i) + u_a(\theta_i))}{V_0} x(\theta_i) \sin\theta_i \tag{19}$$

4.1 Optimum Circulation Distribution

The efficiency optimization consists in the minimization of the propeller power for a given thrust. Applying the method of Lagrange Multiplier, we can write a functional as:

$$F = C_p + \ell C_T$$

where ℓ is the Lagrange multiplier. Taking the derivatives with respect to the circulation values:

$$\frac{\partial F}{\partial G(\theta_i)} = 0 = \frac{\partial C_P}{\partial G(\theta_i)} + \ell \frac{\partial C_T}{\partial G(\theta_i)} \tag{20}$$

Proceeding with calculations, and writing $G_i = G(\theta_i)$, $G_j = G(\theta_j)$, $x_i = x(\theta_i)$, $x_j = x(\theta_j)$ and $U_i = U(\theta_i)$ we obtain the following equation:

$$\frac{U(\theta_i)}{V_0} x_i \sin\theta_i + \sum_{j=1}^{N} G(\theta_j) M_a^*(\theta_i, \theta_j) + \ell\lambda \left[\frac{\omega R x_i}{V_0} \sin\theta_i - \sum_{j=1}^{N} G(\theta_j) M_t^*(\theta_i, \theta_j) \right] = 0 \tag{21}$$

where:

$$M_a^*(\theta_i, \theta_j) = M_a(\theta_i, \theta_j) x_i \sin\theta_i + M_a(\theta_j, \theta_i) x_j \sin\theta_j \tag{22}$$

$$M_t^*(\theta_i,\theta_j) = M_t(\theta_i,\theta_j)\sin\theta_i + M_t(\theta_j,\theta_i)\sin\theta_j \tag{23}$$

The equations **(18)** and **(21)** form a system of N+1 equations unknowns G_i, i=1, .., N and ℓ :

$$\begin{cases} \dfrac{U(\theta_i)}{V_0}x_i\sin\theta_i + \displaystyle\sum_{j=1}^{N} G(\theta_j)M_a^*(\theta_i,\theta_j) + \ell\lambda\left[\dfrac{\omega R x_i}{V_0}\sin\theta_i - \sum_{j=1}^{N} G(\theta_j)M_t^*(\theta_i,\theta_j)\right] = 0 \\ C_T = \dfrac{2\pi\cdot Z(1-x_h)}{N+1}\displaystyle\sum_{j=1}^{N} G_j\dfrac{(\omega R x_i - u_t(\theta_j))}{V_0}\sin\theta_j \end{cases} \tag{24}$$

When a propeller is lightly loaded the induction matrices M_a and M_t are independent of the distribution of circulation. In this case, equation **(21)** constitutes a linear system, but equation **(18)** is a nonlinear because of the dependence of C_T with the square of circulation. For a moderately loaded propeller the induction matrices are dependent of the pitch distribution and therefore of the distribution of circulation. In short, the system **(24)** is nonlinear.

4.2 Solution

We may write the system **(24)** in terms of matrices as follows:

$$\sum_{j=1}^{N+1} A_{ij}^{(n)} G^{*(n)}(\theta_j) = B_i \tag{25}$$

where in each iteration:

- $A_{ij}^{(n)} = M_a^{(n)}(\theta_i,\theta_j)x_i\sin\theta_i + M_a^{(n)}(\theta_j,\theta_i)x_j\sin\theta_j$, when i = 1, 2, …, N and j = 1, 2, …, N
- $A_{N+1,j}^{(n)} = \dfrac{2\pi\cdot Z(1-x_h)}{N+1}\left(\dfrac{1}{\lambda}x_j\sin\theta_j - \sin\theta_j\displaystyle\sum_{j=1}^{N} G_j^{(n-1)}M_t^{(n)}(\theta_i,\theta_j)\right)$

  $= \dfrac{2\pi\cdot Z(1-x_h)}{N+1}\sin\theta_j\left(\dfrac{\pi}{J_0}x_j - \dfrac{u_t^{(n-1)}(\theta_j)}{V_0}\right)$, when j=1, 2, …, N and i=1, 2, …, N+2
- $A_{i,N+1}^{(n)} = x_i\sin\theta_i - \lambda\displaystyle\sum_{j=1}^{N} G_j^{(n-1)}\left(M_t(\theta_i,\theta_j)\sin\theta_i + M_t(\theta_j,\theta_i)\sin\theta_j\right)$, when i = 1, 2, …, N
- $A_{N+1,N+1}^{(n)} = 0$

- $B_i = -\frac{U(\theta_i)}{V_0} x_i \sin\theta_i$, when i = 1, 2, …, N
- $B_{N+1} = C_T$
- $G^{*(n)}(\theta_j) = G^{(n)}(\theta_j)$, when j = 1, 2, …, N
- $G^{*(n)}(\theta_j) = \ell$, when j = N+1

In a first iteration (n=1), because of the nonlinearity of the system **(24)**, we can neglect the tangential velocity component, $u_t^{(n-1)}$, and the circulation, $G^{(n-1)}$, to solve the system thus we have the same number of variables and equations. The system is solved iteratively until we have a convergence.

In an attempt to minimize the number of iterations, we could estimate the tangential velocities for the first iteration (n=1) by the triangle shown in **fig. 3**:

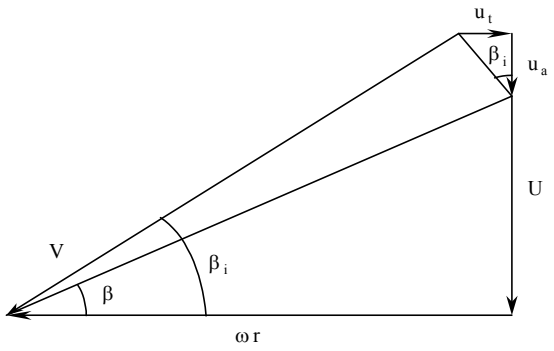

Fig. 3 – Velocity triangle

In general: $\begin{cases} U(x) = V_s(1 - w(x)) \\ V0 = V_s(1 - w_T) \end{cases}$, where w(x) is the local wake fraction, w_T is the Taylor wake fraction and V_s is the ship speed.

For a given J_0 ($J_0=\pi\lambda$) and an estimated β_i from the theory for infinite number of blades we have:

$$\frac{u_t}{V_0} = \sin\beta_i \cos\beta_i \left(x \frac{\pi}{J_0} tg\beta_i - \frac{1 - w(x)}{1 - w_T} \right) \quad (26)$$

4.3 Algorithm

We may summarize all in a scheme (**fig. 4**) for implementation in a computer code.

Problem: Given N, Z, x_h, J_0, w(x), w_T and C_T find optimum G(x).
Give number of points N.

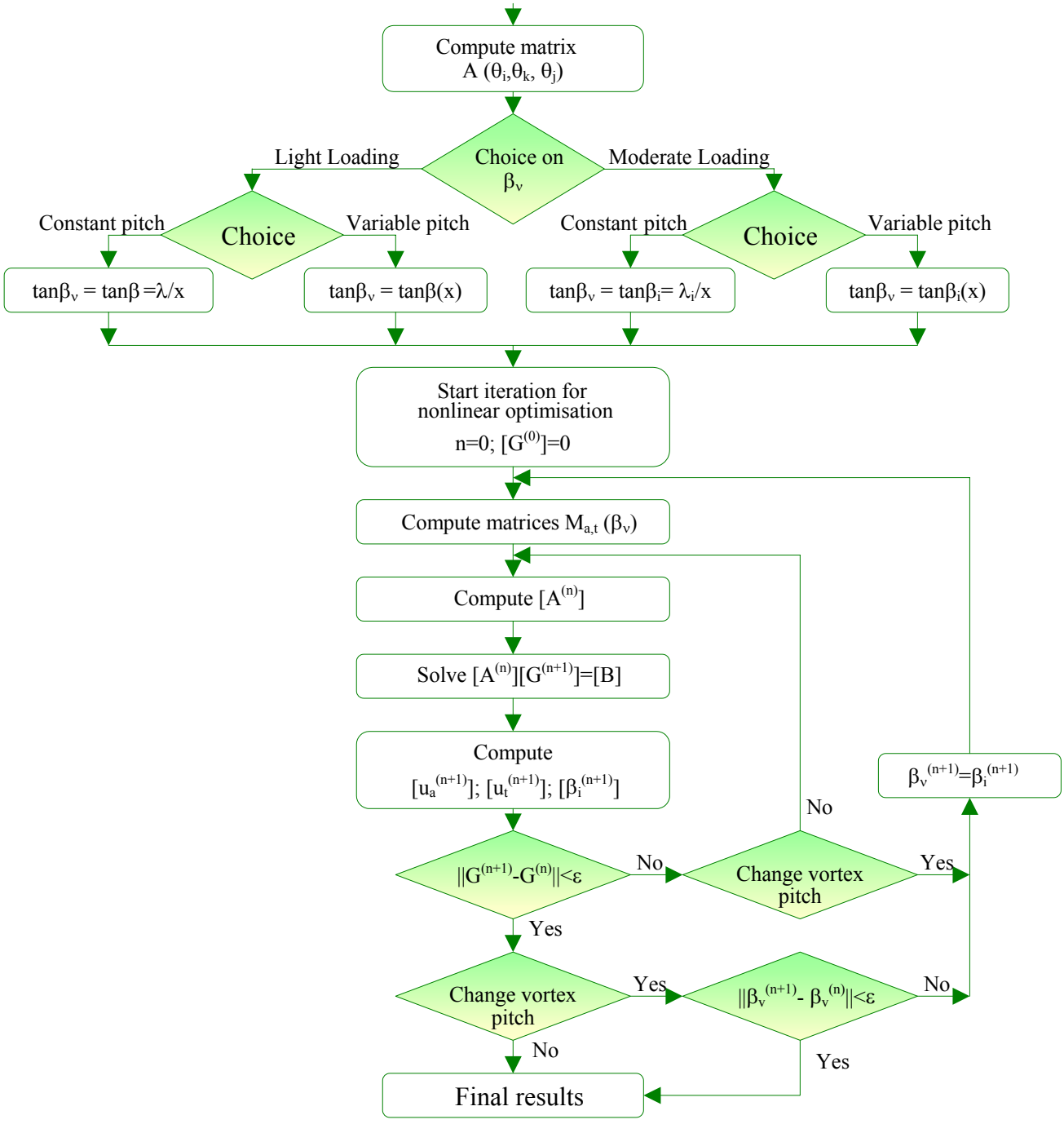

Fig. 4 – Algorithm for solving nonlinear optimization problem.

4.4 Implementation

The algorithm shown in fig. 4 was implemented in a computer code. The program elaborated is called ***OptiProp*** and was written in FORTRAN language. The program listing of *OptiProp* is given in **appendix A.1**.

In the implementation:

- ⇨ it was assumed that the circulation is zero in the extremes of the blade (boundary condition);
- ⇨ for the convergence on circulation the following relaxation method was applied to the values of circulation obtained in each iteration:

$$G_j^{(n)} = G_j^{(n-1)} + \text{relax} \cdot \left(G_j^{(n)} - G_j^{(n-1)} \right)$$

where relax is the coefficient of relaxation that was fixed on 0.5. The importance of this scheme is to improve the convergence of the circulation.

The hub zero normal velocity condition is not included in *OptiProp*. Also the hub radius must be different from zero.

To run *OptiProp* we need an input text file called ***OptiProp.inp*** where are the following entry data, by order:

- ⇨ value **1** if the propeller is lightly loaded or **2** if the propeller is moderately loaded;
- ⇨ number of internal points, N;
- ⇨ number of blades, Z;
- ⇨ thrust coefficient, C_T;
- ⇨ nondimensional hub radius, x_h;
- ⇨ advance coefficient, J_0;
- ⇨ number of points where it is given the effective wake fraction;
- ⇨ nondimensional radius for the number of points mentioned above, x;
- ⇨ values for the effective wake fraction, w(x);
- ⇨ value of local wake fraction, w_T.

OptiProp gives one output file, ***OptiProp.out***, where are the entry data and the calculations results.

4.5 Results

In this section we compare the results given by *OptiProp* and *LerbsNU*. *LerbsNU* is based on the induction factors method of Lerbs[5] for the lifting line theory in inviscid flow. It has been shown[2] that the results given by this program are in close agreement with the results obtained by Lerbs in 1952[5]. Only the values of the tangential induced velocities have some small differences.

The method followed in this work is not equivalent to the method followed on *LerbsNU*, but *OptiProp* should approximate its results, especially for light loading. When propeller loading increases the differences between *OptiProp* and *LerbsNU* accentuate.

We consider a case of a propeller in a uniform flow with the operating conditions given in the textbook by Breslin and Andersen[3]. The entry data:

- $Z = 5$;
- $x_h = 0.2$;
- $C_T = 0.512$;
- $w(x) = 0$;
- $w_T = 0$.

4.5.1 Light Loading

In light loading assumption, we consider the vortex pitch distribution to be constant and equal to the advance pitch, i.e., the helical pitch angle $\beta_v=\beta$ is equal to the advance angle β. This is a good approximation when the propeller is lightly loaded because the axial and tangential velocities are small compared with the undisturbed velocities and therefore may be neglected.

4.5.1.1 Influence of number of points

For the case described in section 4.5, the **figs. 5 and 6** show the optimum distributions of circulation, axial and tangential velocities along the radius obtained from *OptiProp* when executed with 11 and 23 internal points.

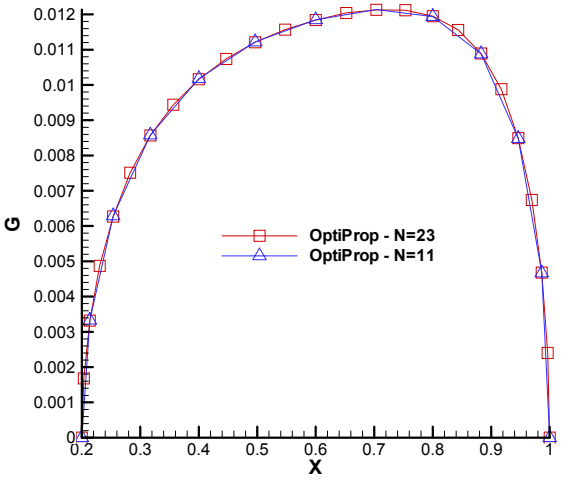

Fig. 5 – Optimum distribution of circulation along radius for N=11 and N=23, C_T=0.512, J_0=0.6, light loading and uniform flow.

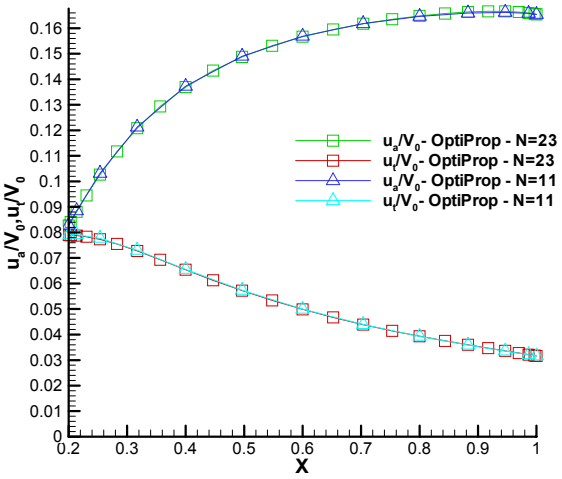

Fig. 6 – Optimum distribution of tangential and axial velocities along radius for N=11 and N=23, C_T=0.512, J_0=0.6, light loading and uniform flow.

We conclude reasonable accuracy is already obtained with only 11 internal points.

Figs. 7 and 8 show the results given by *OptiProp* and *LerbsNU* with 23 internal points. *OptiProp* gives practically the same results as *LerbsNU* (*LerbsNU* was used with constant hydrodynamic pitch wich is the classical optimum). This shows tahat for this propeller loading ythe results of *OptiProp* agree with the classical optimization.

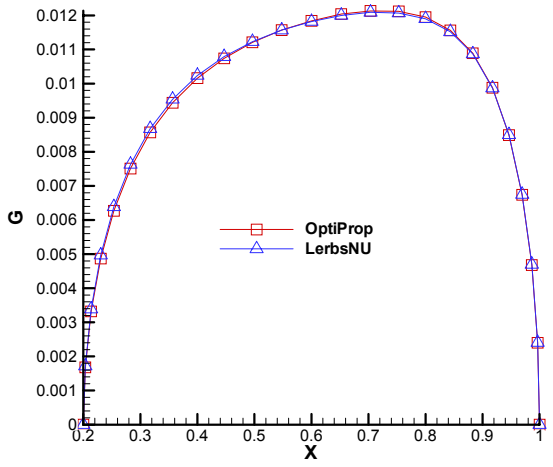

Fig. 7 – Optimum distribution of circulation along radius for N=23, C_T=0.512, J_0=0.6, light loading and uniform flow.

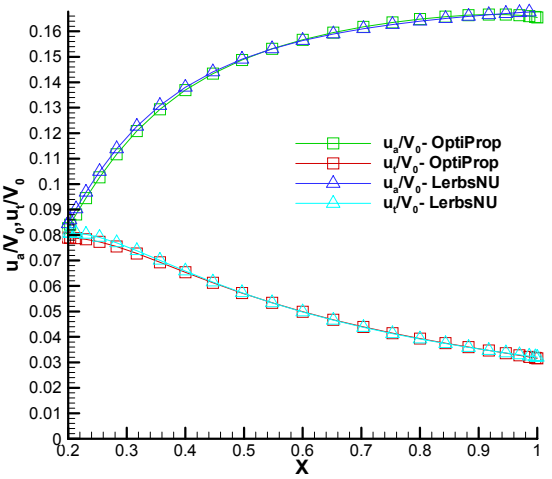

Fig. 8 – Optimum distribution of tangential and axial velocities along radius for N=23, C_T=0.512, J_0=0.6, light loading and uniform flow.

4.5.1.2 Influence of thrust coefficient

Keeping 23 internal points, the influence of the thrust coefficient on the results is shown in **figs. 9 and 10**. The curves show present a similar behavior with less loading. When decreasing the thrust coefficient the differences between *OptiProp* and *LerbsNU* reduce.

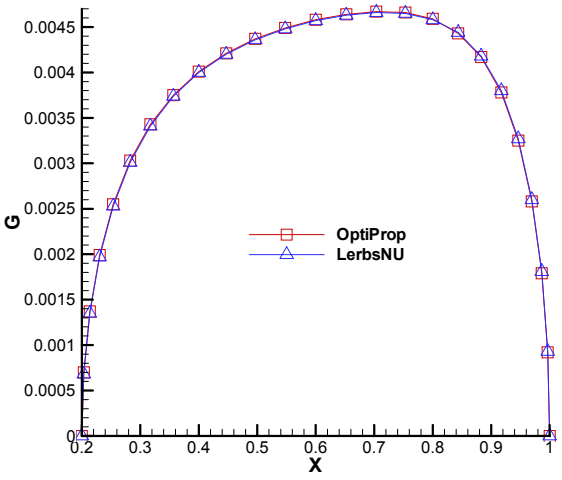

Fig. 9 – Optimum distribution of circulation along radius for N=23, C_T=0.2, J_0=0.6, light loading and uniform flow

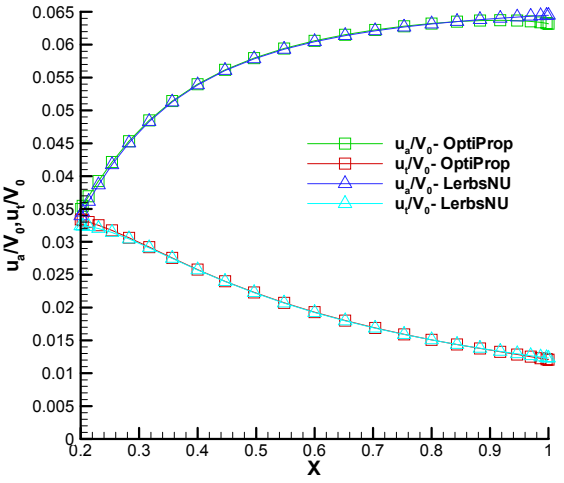

Fig. 10 – Optimum distribution of tangential and axial velocities along radius for N=23, C_T=0.2, J_0=0.6, light loading and uniform flow

4.5.2 Moderate Loading

When propellers are moderately loaded, the axial and tangential induced velocities are taken into account and the vortex pitch distribution is identical to the hydrodynamic pitch, i.e., $\beta_v = \beta_i$.

Looking at the algorithm of section 4.2, there are two ways to obtain the optimum distribution of circulation. The first is to make equal the value of β_v to β_i in each iteration → the *outside loop* (see algorithm shown in fig. 4). The second is to freeze the value of β_v in each iteration until the circulation converges and only after that, to change the β_v to the last β_i estimated from the axial and tangential induced velocities. Then we maintain the same β_v until we have a new convergence on circulation. This stops when we find a convergence on the vortex pitch distribution. That we will call the *inside/outside loop*.

Figs. 11 and **12** present the results obtained when the execution of *OptiProp* with the *outside loop* and the *inside/outside loop*. The converged results obtained in both ways are the same.

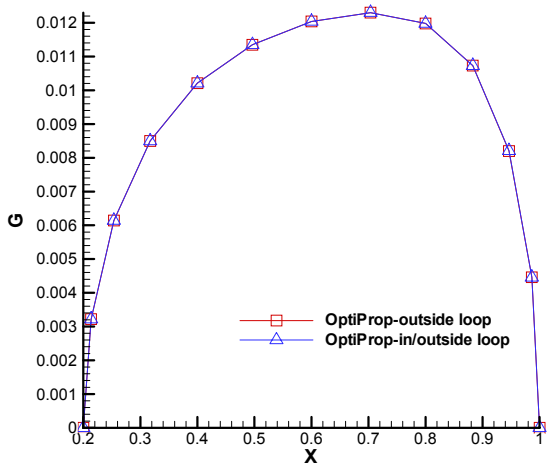

Fig. 11 – Optimum distribution of circulation along radius for N=11, C_T=0.512, J_0=0.6, moderate loading and uniform flow.

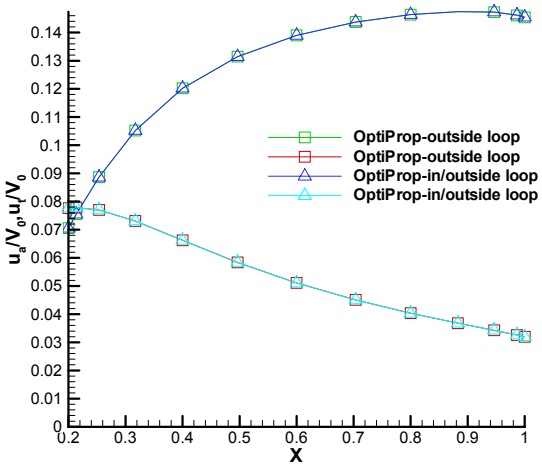

Fig. 12 – Optimum distribution of tangential and axial velocities along radius for N=11, C_T=0.512, J_0=0.6, moderate loading and uniform flow.

However, the two ways (*inside/outside loop* and *outside loop*) differ in the number of iterations to reach convergence. The convergence is very difficult to reach when executing *OptiProp* with the *outside loop* especially when loading increases and/or the number of points also increases. So, from this point forward, all the results presented with respect to moderate loading are obtained with the *inside/outside loop*.

4.5.2.1 Influence of number of points

When the number of points increases, the accuracy of the results also increases, as it was already seen with a lightly loaded propeller. But, in moderate loading and for larger number of points, problems were met in the convergence of the solutions. With 23 internal points *OptiProp* couldn't converge.

The results given by *OptiProp* and *LerbsNU* with 11 points are compared in **figs. 13** and **14**. The distributions are very close. These figures confirm that for a uniform flow, lightly or moderately loaded assumptions, the program *OptiProp* provide similar results to *LerbsNU*.

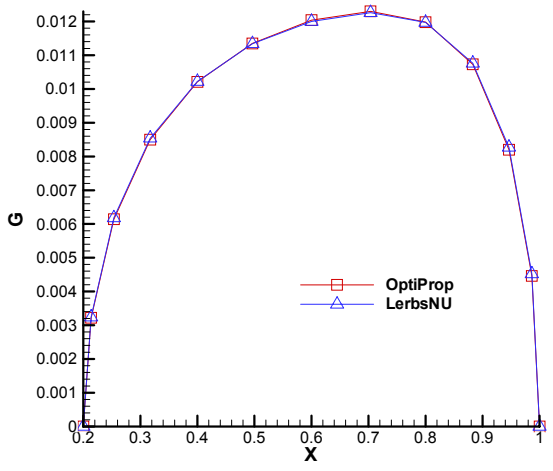

Fig. 13 – Optimum distribution of circulation along radius for N=11, C_T=0.512, J_0=0.6, moderate loading, uniform flow and inside/outside loop.

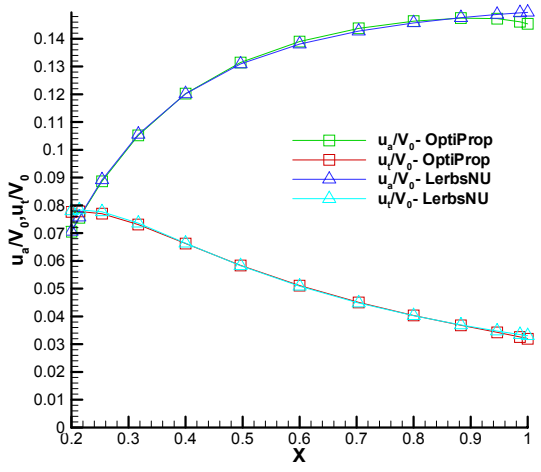

Fig. 14 – Optimum distribution of tangential and axial velocities along radius for N=11, C_T=0.512, J_0=0.6, moderate loading, uniform flow and inside/outside loop.

4.5.2.2 Influence of tolerance

So far the convergence on the circulation and on the vortex pitch distribution was made considering a tolerance of 0.00001 for absolute errors wich is rather small. Changing for relative tolerance of 0.5% and executing *OptiProp* with the *inside/outside loop* we could reach a result for a 23 internal points. Only in this case we had convergence.

Still notice that when comparing the performance of *OptiProp* when using absolute or relative tolerance, the number of iterations reduce to obtain the same end result. Adding a note, for the case presented ($C_T = 0.512$, $J_0 = 0.6$, moderate loading and uniform flow) *LerbsNU* iterates numerous times (184 iterations) to give a result to 23 internal points and *OptiProp* only needed a few iterations (11 iterations).

Figs. 15 and **16** show the results from *OptiProp* and *LerbsNU* for 23 internal points.

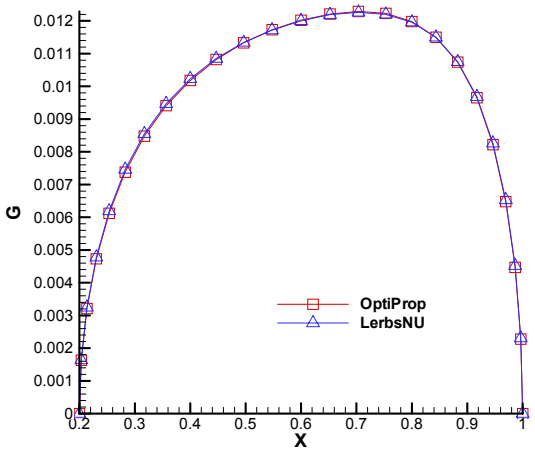

Fig. 15 – Optimum distribution of circulation along radius for N=23, C_T=0.512, J_0=0.6, moderate loading, uniform flow, relative error and inside/outside loop.

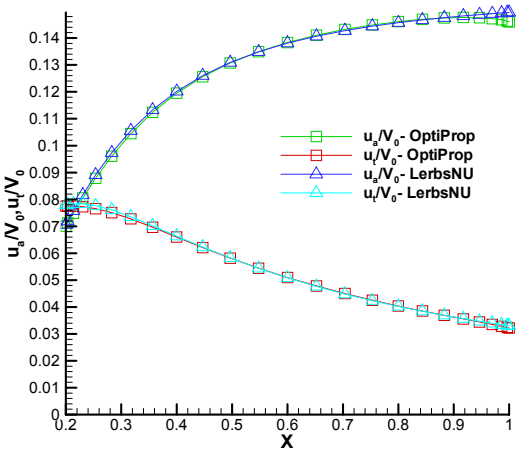

Fig. 16 – Optimum distribution of tangential and axial velocities along radius for N=23, C_T=0.512, J_0=0.6, moderate loading, uniform flow, relative error and inside/outside loop.

4.5.2.3 Influence of advance coefficient

Fig. 17 shows the effect of the advance ratio, J_0, on the distribution of circulation.

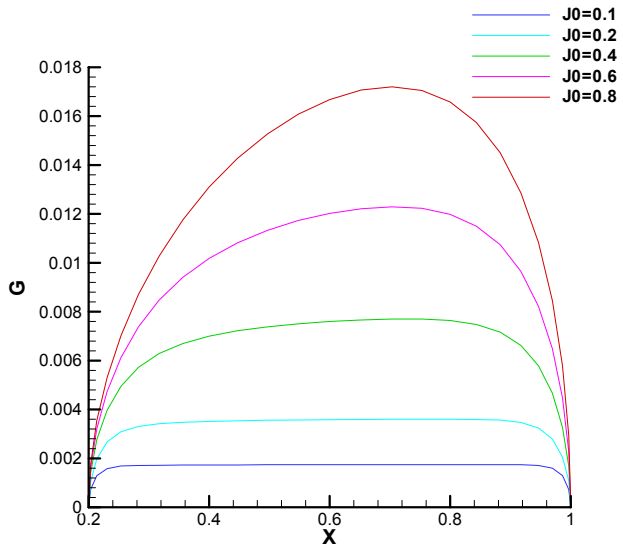

Fig. 17 – Evolution of the distribution of circulation for each J_0 along radius with C_T=0.512, N=23, moderate loading and uniform flow.

For $J_0 = 1.0$, $J_0 = 1.2$ and $J_0 = 1.4$ *OptiProp* did not converge. It only gives results for a few number of points. This is a confirmation of the problems that exist on the convergence of the circulation. It depends of the thrust coefficient, the advance coefficient and the number of points.

4.5.3 Discussion of results

The modifications made on the program *OptiProp* are insufficient because for increasing loading and number of points, the convergence deteriorates, and some times, there is no convergence.

For a lightly loaded propeller it was not found any problem.

Based in all of these facts, we conclude that the formulation needs an improvement.

5. ANOTHER APPROACH TO THE MATHEMATICAL FORMULATION OF THE INVISCID PROBLEM

The discrete optimization, equation **(21),** can be written in another form[1]:

$$\frac{U_i/V_0 + u_a(\theta_i)/V_0 + u_a^*(\theta_i)/V_0}{x_i/\lambda - u_t(\theta_i)/V_0 - u_t^*(\theta_i)/V_0} = -\ell\frac{\lambda}{x_i} \tag{27}$$

where:

$$\frac{u_a^*}{V_0}(\theta_i) = \frac{1}{x_i \sin\theta_i}\sum_{j=1}^{N} G_j M_a(\theta_j,\theta_i) x_j \sin\theta_j \tag{28}$$

$$\frac{u_t^*}{V_0}(\theta_i) = \frac{1}{\sin\theta_i}\sum_{j=1}^{N} G_j M_t(\theta_j,\theta_i) \sin\theta_j \tag{29}$$

If we neglect the terms: $u_a^*(\theta_i)$ and $u_t^*(\theta_i)$

we obtain:

$$\frac{\tan\beta_i}{\tan\beta} = -\ell = \text{const} \tag{30}$$

which is Betz condition, see [3], and coincides with Lerbs optimum of constant pitch.

If we assume that

$$\frac{u_a^*}{V_0}(\theta_i) \cong \frac{u_a(\theta_i)}{V_0} \text{ and } \frac{u_t^*}{V_0}(\theta_i) \cong \frac{u_t(\theta_i)}{V_0} \tag{31}$$

then we obtain:

$$\frac{U_i/V_0 + 2u_a/V_0}{x_i/\lambda - 2u_t/V_0} = -\ell\tan\beta \tag{32}$$

[1] Demonstration on Appendix B

This is the formulation used by Yim[6] in standard lifting line theory with application of Munk's displacement theorem when neglecting the viscous term. This is also the criteria used by Dyne[7] that computes the hydrodynamic pitch from induced velocities in the far wake, that is:

$$\frac{\tan\beta_{i\infty}}{\tan\beta} = -\ell = \text{const}$$

Andersen[8] carried out a study to verify this assumption. He concludes that the differences in the results using the assumption given by equation **(31)** or using the terms $u_a^*(\theta_i)$ and $u_t^*(\theta_i)$ are insignificant.

However, the study presented in this report considered the terms given by equations (28) and (29). A further discussion will be made in point 5.3.2.6 for this matter.

For a general case, where the flow is non-uniform, equation **(27)** can be written in the form:

$$\sum_{j=1}^{N} G_j\left(M_a(\theta_i,\theta_j) - \ell\frac{\lambda}{x_i}M_t(\theta_i,\theta_j)\right) + \frac{1}{x_i\sin\theta_i}\sum_{j=1}^{N} G_j M_a(\theta_j,\theta_i)x_j\sin\theta_j$$
$$-\frac{\ell\lambda}{x_i\sin\theta_i}\sum_{j=1}^{N} G_j M_t(\theta_j,\theta_i)\sin\theta_j = -\frac{1-w(x)}{1-w_T} - \ell \qquad (33)$$

If the induction matrices M_a and M_t are independent of the circulation values, i.e., in the lightly loaded assumption, equation **(33)** is a linear system of equations. In the moderately loaded assumption, M_a and M_t are functions of the circulation through the dependence on the vortex pitch and the system has to be solved iteratively. In general, we may write the linear system in a matrix form:

$$\sum_{j=1}^{N+1} A_{ij}^{(n)} G(\theta_j) = B_i \qquad (34)$$

where for each iteration:

$$\Rightarrow A_{ij}^{(n)} = M_a^{(n)}(\theta_i, \theta_j) - \ell \frac{\lambda}{x_i} M_t^{(n)}(\theta_i, \theta_j) + \frac{1}{x_i \sin\theta_i} M_a^{(n)}(\theta_j, \theta_i) x_j \sin\theta_j$$

$$- \ell \frac{\lambda}{x_i \sin\theta_i} M_t^{(n)}(\theta_j, \theta_i) \sin\theta_j, \text{ when } i = 1, 2, \ldots, N \text{ and } j = 1, 2, \ldots, N$$

$$\Rightarrow B_i = -\frac{1 - w(x)}{1 - w_T} - \ell, \text{ when } i = 1, 2, \ldots, N$$

5.1 Algorithm

The algorithm for this new approach to the problem is very similar to algorithm of fig. 4 shown in point 4.2. However, in the present case, the convergence is on the Lagrange multiplier and on hydrodynamic pitch angle. See **fig. 18**.

Problem: Given N, Z, x_h, J_0, w(x), w_T and C_T find optimum G(x).

Give number of points N.

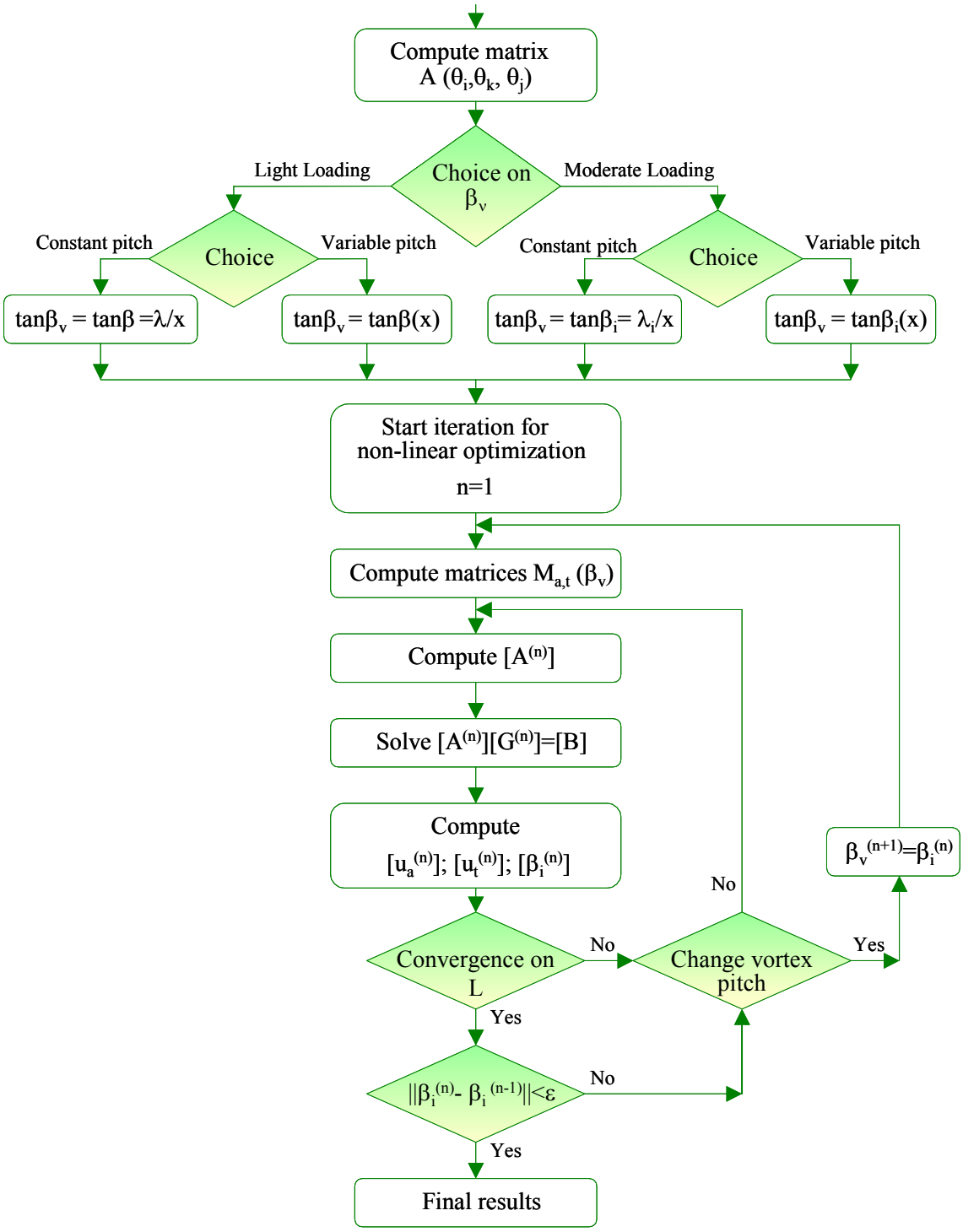

Fig. 18 – Algorithm for the second nonlinear optimization solution to the problem.

5.2 Implementation

To solve the problem, it was elaborated the program ***OptiPropII*** in FORTRAN code**. Appendix A.2** contains a listing of *OptiPropII*.

The same considerations made for *OptiProp* (boundary conditions, relaxation method and hub) hold for *OptiPropII*.

The input and output files have the same name as the ones used for *OptiProp* (*OptiProp.inp* and *OptiProp.out*, respectively). The entry data are analogous.

5.3 Results

In order to check the new approach to the problem a comparison between the results obtained from *OptiPropII*, *OptiProp* and *LerbsNU* will be made.

Consider now a case study very similar to one described in point 4.4. In this case the entry data is:

- Z = 5;
- x_h = 0.2;
- C_T = 0.2;
- J_0=0.6;
- w(x) = 0;
- w_T = 0.

5.3.1 Light Loading

Figs. 19 to 21 show the optimum distribution of circulation, the axial and tangential induced velocities and pitch distribution ($P/D = \pi x \tan\beta_i$) along the propeller radius for the case study presented.

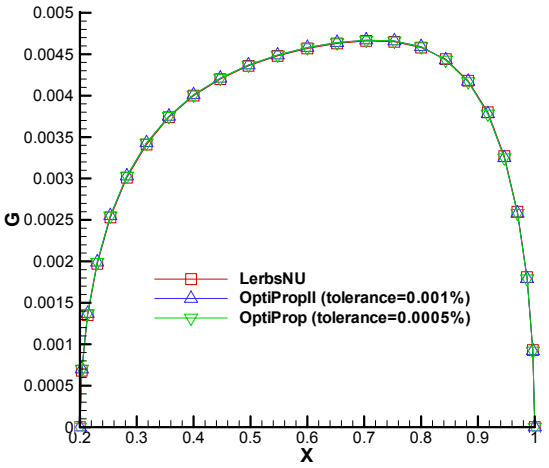

Fig. 19 - Optimum distribution of circulation along radius for N=23, C_T=0.2, J_0=0.6, light loading and uniform flow. Results from LerbsNU, OptiProp and OptiPropII.

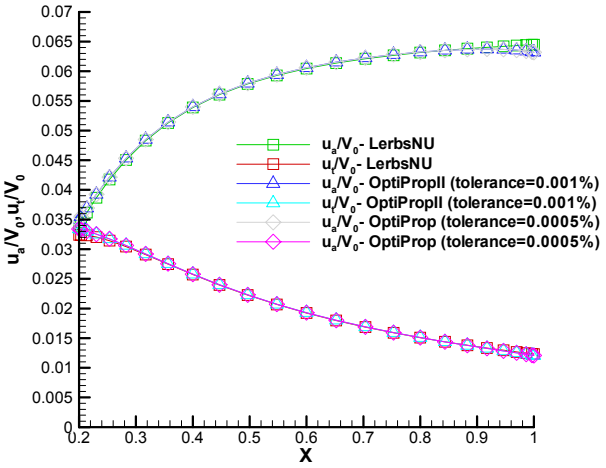

Fig. 20 - Optimum distribution of tangential and axial induced velocities along radius for N=23 C_T=0.2, J_0=0.6, light loading and uniform flow. Results from LerbsNU, OptiProp and OptiPropII.

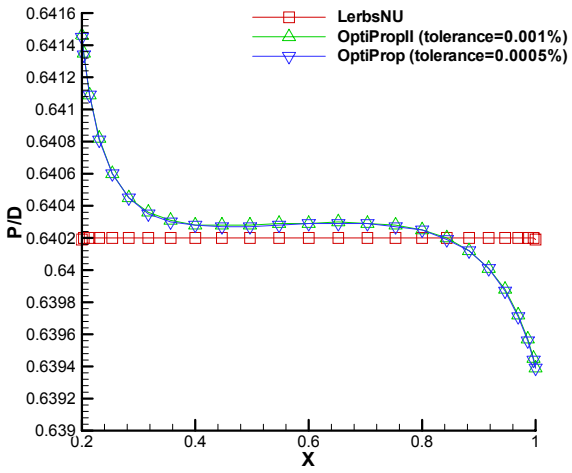

Fig. 21 - Optimum pitch distribution along radius for N=23 C_T=0.2, J_0=0.6, light loading and uniform flow. Results from LerbsNU, OptiProp and OptiPropII.

The curves obtained from *OptiPropII* have similar evolution to the curves obtained from *LerbsNU* (figs. 19 and 20). In this case results given by *OptiProp* and *OptiPropII* match.

Relatively to the pitch distribution (fig. 21) the differences obtained could exist because of the introduction of the terms $u_a^*(\theta_i)$ and $u_t^*(\theta_i)$ in the formulation. If we neglect the terms $u_a^*(\theta_i)$ and $u_t^*(\theta_i)$ given by expressions **(28)** and **(29)**, respectively, the results obtained are in close agreement with *LerbsNU* (**figs. 22 to 24**).

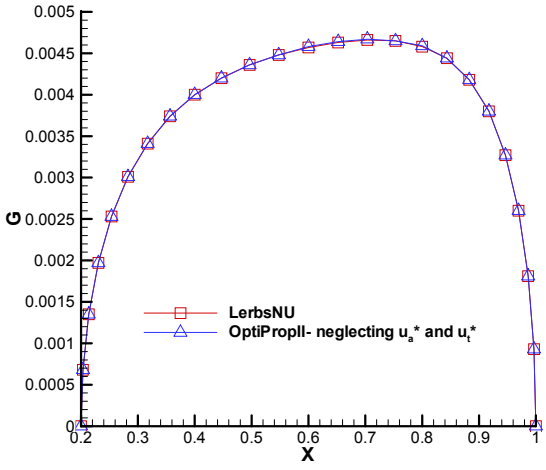

Fig. 22 - Optimum distribution of circulation along radius for N=23, C_T=0.2, J_0=0.6 and tolerance=0.001%, light loading and uniform flow. Results from LerbsNU and OptiPropII (neglecting terms $u_a^*(\theta_i)$ and $u_t^*(\theta_i)$).

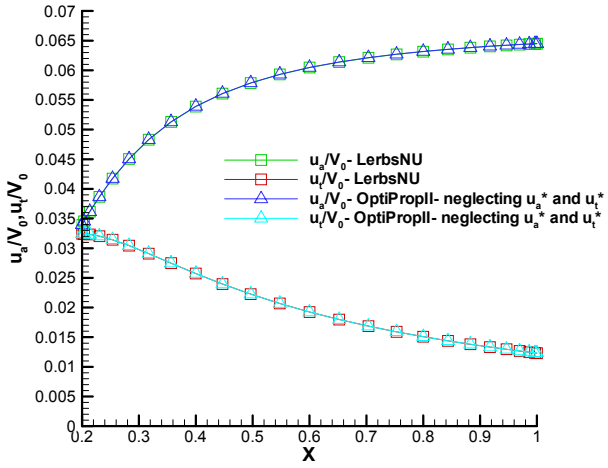

Fig. 23 - Optimum distribution of tangential and axial induced velocities along radius for N=23 C_T=0.2, J_0=0.6 and tolerance=0.001%, light loading and uniform flow. Results from LerbsNU and OptiPropII (neglecting terms $u_a^*(\theta_i)$ and $u_t^*(\theta_i)$).

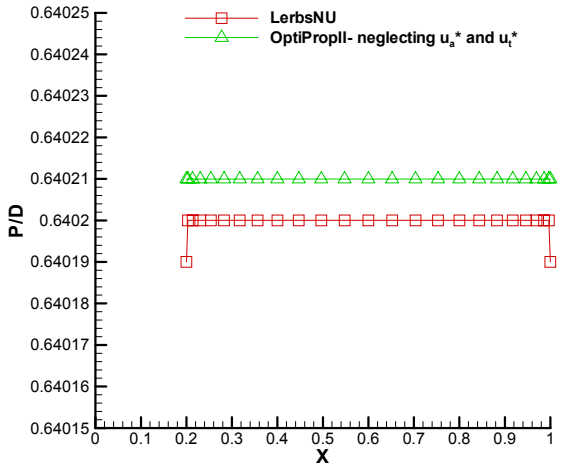

Fig. 24 - Optimum pitch distribution along radius for N=23 C_T=0.2, J_0=0.6 and tolerance=0.001%, light loading and uniform flow. Results from LerbsNU and OptiPropII (neglecting terms $u_a^*(\theta_i)$ and $u_t^*(\theta_i)$).

Looking at the hydrodynamic pitch distribution on fig. 24, it seems that results are much different, but they are in close proximity to each other (note the scale used). So, when introducing the parameters $u_a^*(\theta_i)$ and $u_t^*(\theta_i)$ the pitch distribution changes, near the tip and hub radius.

Table 1 shows the results for the propeller efficiency, the power coefficient and the number of iterations for 11 and 23 internal points.

	LerbsNU			***OptiProp*** Tolerance = 0.0005%			***OptiPropII*** Tolerance = 0.001%		
N	C_P	η [%]	Nº Iterations	C_P	η [%]	Nº Iterations	C_P	η [%]	Nº Iterations
11	0.213	93.8	4	0.213	93.7	19	0.213	93.7	14
23	0.213	93.8	4	0.213	93.7	19	0.213	93.7	14

Table 1 – Values of propeller efficiency, power coefficient and number of iterations for a lightly loaded propeller with N=11 and N=23, C_T=0.2 and J_o=0.6. Results obtained from LerbsNU, OptiProp and OptiPropII.

Values obtained from *OptiProp* and *OptiPropII* match, but *OptiPropII* iterates less times. When comparing with *LerbsNU*, the optimization has an efficiency 0.1% above the efficiency obtained from *OptiProp* and *OptiPropII*. Note that *LerbsNU* to give this result only need to iterate 4 times.

5.3.1.1 Influence of tolerance

The tolerance is a very important parameter for the solution obtained from *OptiPropII*. Some solutions are converged and others are not converged, depending on values of the tolerance assumption. For the present case study, **table 2** permits to see that the convergence to 5 decimals on pitch distribution along the nondimensional propeller radius is reached within a tolerance of 0.001%.

P/D	Tolerance						
x	0.5%	0.1%	0.05%	0.01%	0.005%	0.001%	0.0005%
0.2	0.64428	0.64190	0.64169	0.64153	0.64148	0.64146	0.64146
0.20342	0.64416	0.64179	0.64158	0.64141	0.64136	0.64135	0.64135
0.21363	0.64389	0.64153	0.64132	0.64116	0.64111	0.64109	0.64109
0.23045	0.64362	0.64125	0.64105	0.64088	0.64083	0.64082	0.64082
0.25359	0.64340	0.64104	0.64083	0.64067	0.64062	0.64060	0.64060
0.28266	0.64325	0.64089	0.64068	0.64052	0.64047	0.64045	0.64045
0.31716	0.64316	0.64079	0.64059	0.64042	0.64037	0.64036	0.64036
0.3565	0.64312	0.64074	0.64054	0.64037	0.64032	0.64031	0.64031
0.4	0.64310	0.64072	0.64052	0.64035	0.64030	0.64028	0.64028
0.44693	0.64310	0.64071	0.64051	0.64034	0.64029	0.64028	0.64027
0.49647	0.64311	0.64072	0.64051	0.64034	0.64029	0.64028	0.64028
0.54779	0.64312	0.64072	0.64052	0.64035	0.64030	0.64029	0.64028
0.6	0.64313	0.64073	0.64053	0.64036	0.64031	0.64029	0.64029
0.65221	0.64314	0.64074	0.64053	0.64036	0.64031	0.64030	0.64029
0.70353	0.64314	0.64073	0.64053	0.64036	0.64031	0.64029	0.64029
0.75307	0.64312	0.64072	0.64052	0.64034	0.64029	0.64028	0.64028
0.8	0.64309	0.64069	0.64049	0.64032	0.64027	0.64025	0.64025
0.8435	0.64304	0.64064	0.64044	0.64027	0.64022	0.64020	0.64020
0.88284	0.64296	0.64056	0.64036	0.64019	0.64014	0.64012	0.64012
0.91734	0.64284	0.64045	0.64025	0.64008	0.64003	0.64001	0.64001
0.94641	0.64270	0.64031	0.64011	0.63994	0.63989	0.63988	0.63987
0.96955	0.64253	0.64015	0.63995	0.63978	0.63973	0.63972	0.63972
0.98637	0.64237	0.64000	0.63980	0.63963	0.63958	0.63957	0.63956
0.99658	0.64224	0.63988	0.63968	0.63951	0.63946	0.63945	0.63944
1	0.64218	0.63982	0.63962	0.63945	0.63941	0.63939	0.63939

Table 2 – Convergence for pitch distribution along radius with tolerance, obtained from OptiPropII for light loading, uniform flow, C_T=0.2, J_0=0.6 and N=23

Table 2 also permits to conclude that is more difficult to reach convergence near the tips and hub sections of the propeller. The convergence on nondimensional circulation is achieved within a higher tolerance, as well as the axial and tangential induced velocities (see tables of **appendix C.1.1**, **C.2.1** and **C.3.1**).

5.3.1.2 Influence of number of points

Figs. 25 to 27 show the results for a 11 and 23 internal points and both with same tolerance assumption of 0.001%.

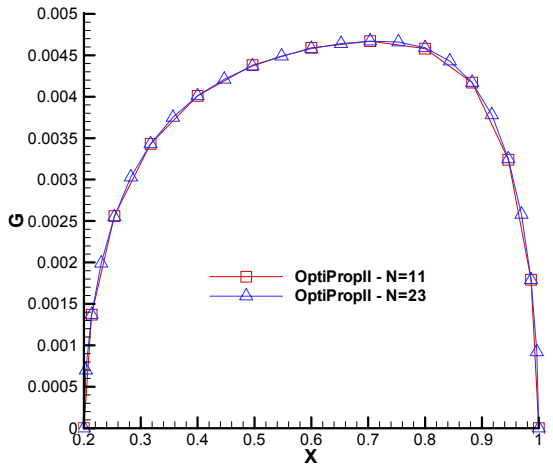

Fig. 25 - Optimum distribution of circulation along radius for N=11 and N=23, C_T=0.2, J_0=0.6, tolerance=0.001%, light loading and uniform flow. Results from OptiPropII.

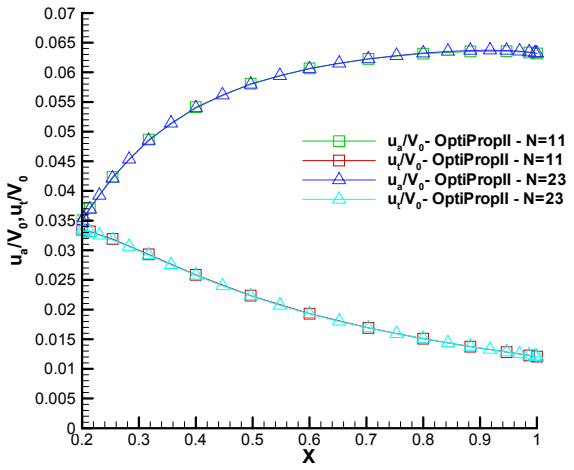

Fig. 26 - Optimum distribution of tangential and axial induced velocities along radius for N=11 and N=23, C_T=0.2, J_0=0.6, tolerance=0.001%, light loading and uniform flow. Results from OptiPropII.

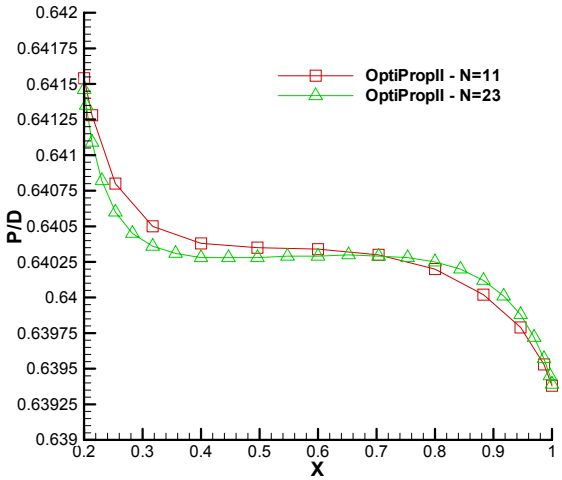

Fig. 27 - Optimum pitch distribution along radius for N=11 and N=23 C_T=0.2, J_0=0.6 and tolerance=0.001%, light loading and uniform flow. Results from OptiPropII.

When the number of internal points increases, the curves are in fact much more delineated than with 11 internal points. However, with 11 internal points *OptiPropII* gives reasonable results.

5.3.2 Moderate Loading

To solve the problem when a propeller is moderately loaded, the pitch distribution should be maintained at each iteration. The **fig. 28** shows a scheme for the procedure followed by *OptiPropII*.

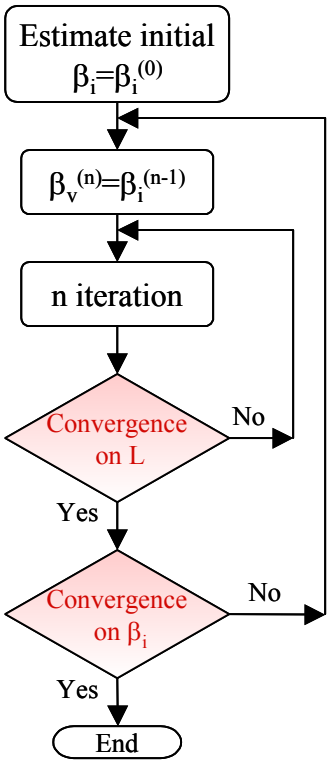

Fig. 28 - Scheme for the execution of OptiPropII when a propeller is moderately loaded.

When hydrodynamic pitch (in each iteration) changes, the procedure becomes very unstable and diverges easily. Therefore, the formulation is very sensitive to a very small variation of the hydrodynamic pitch. It will be seen in the next sections that this sensibility has its origin not only in the hydrodynamic pitch but also in other parameters. However, fixing the hydrodynamic pitch helps on convergence.

Figs. 29 to 31 show the optimum distribution of circulation, axial and tangential induced velocities and the pitch distribution for *OptiPropII* and *LerbsNU*.

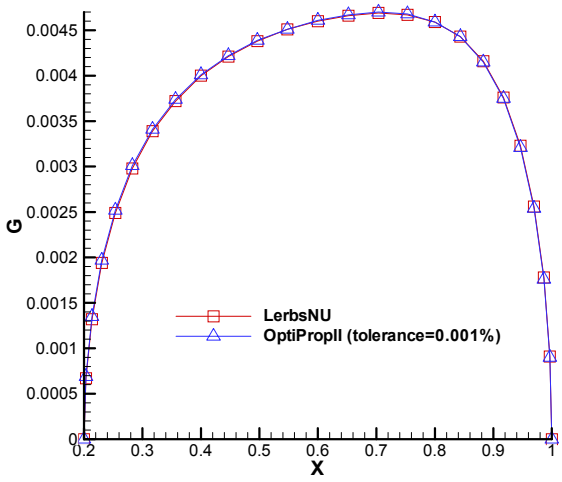

Fig. 29 - Optimum distribution of circulation along radius for N=23, C_T=0.2, J_0=0.6, moderate loading and uniform flow. Results from LerbsNU and OptiPropII.

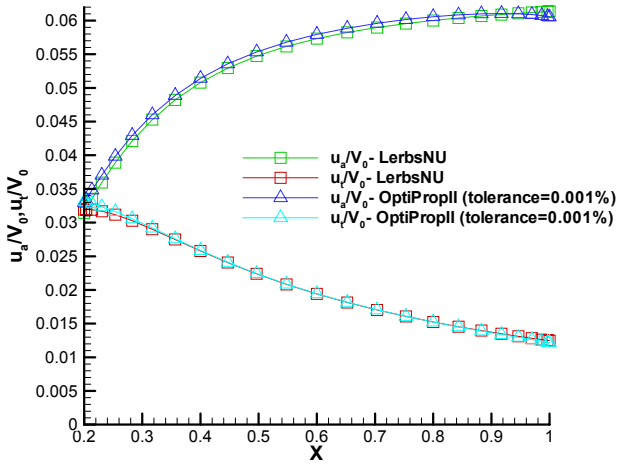

Fig. 30 - Optimum distribution of tangential and axial induced velocities along radius for N=23 C_T=0.2, J_0=0.6, moderate loading and uniform flow. Results from LerbsNU and OptiPropII.

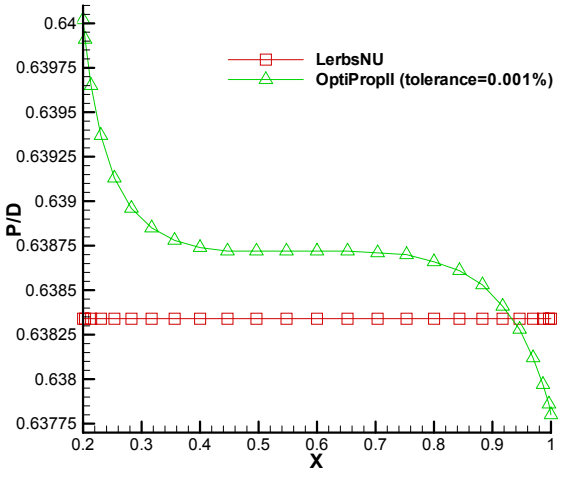

Fig. 31 - Optimum pitch distribution along radius for N=23 C_T=0.2, J_0=0.6, moderate loading and uniform flow. Results from LerbsNU and OptiPropII.

The evolutions obtained from *OptiPropII* and *LerbsNU* for nondimensional circulation and axial and tangential induced velocities along the propeller radius are very similar. Concerning the hydrodynamic pitch discrepancies are a consequence of the introduction of terms $u_a*(\theta_i)$

and $u_t^*(\theta_i)$ in the formulation. As in the case of a lightly loaded propeller, when neglecting the terms $u_a^*(\theta_i)$ and $u_t^*(\theta_i)$ given by expressions **(28)** and **(29)**, respectively, the result achieved by *OptiPropII* is in close agreement to the result obtained from *LerbsNU* (**figs. 32 to 34).**

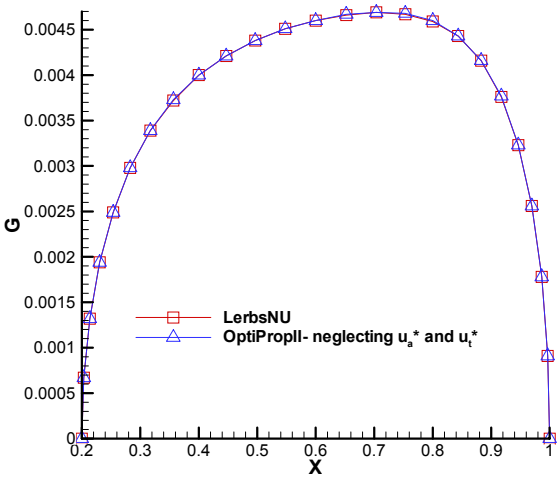

Fig. 32- Optimum distribution of circulation along radius for N=23, C_T=0.2, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow. Results from LerbsNU and OptiPropII (neglecting terms $u_a^*(\theta_i)$ and $u_t^*(\theta_i)$).

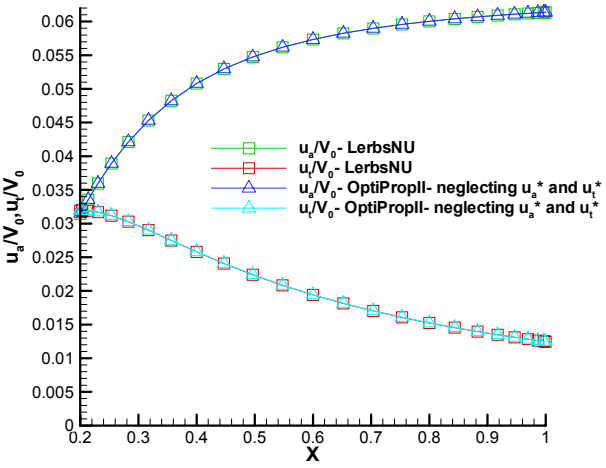

Fig. 33 - Optimum distribution of tangential and axial induced velocities along radius for N=23 C_T=0.2, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow. Results from LerbsNU and OptiPropII (neglecting terms $u_a^*(\theta_i)$ and $u_t^*(\theta_i)$).

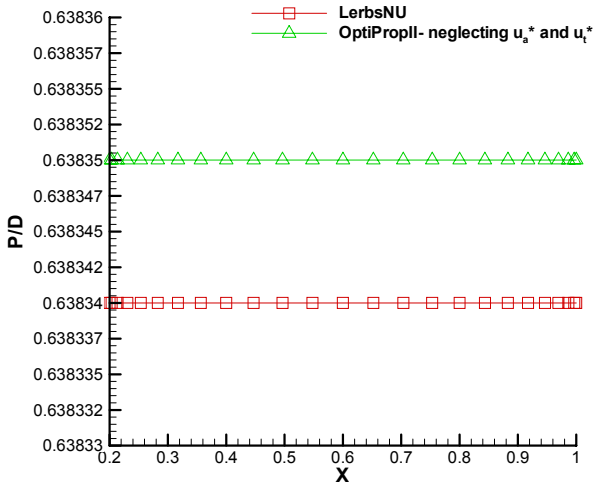

Fig. 34 - Optimum pitch distribution along radius for N=23 C_T=0.2, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow. Results from LerbsNU and OptiPropII (neglecting terms $u_a^*(\theta_i)$ and $u_t^*(\theta_i)$).

If we execute *LerbsNu* with initial optimum hydrodynamic pitch distribution achieved by *OptiPropII* instead of a constant hydrodynamic pitch, we obtain different solutions for both cases. The results are shown in **fig. 35**.

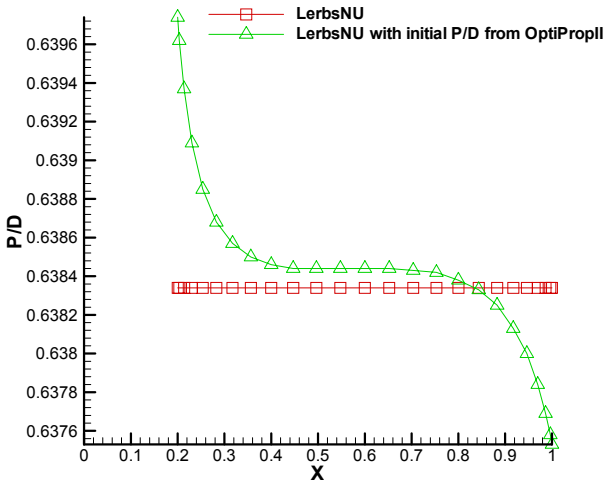

Fig. 35 - Optimum pitch distribution along radius for N=23 C_T=0.2, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow. Results from LerbsNU.

The solution agrees with the results of *OptiPropII* which confirms the implementation ofoptimization algorithm.

The solution obtained from *LerbsNU* when executed with initial pitch distribution of *OptiPropII* presents a variable pitch distribution and the same efficiency (94.0%) as the solution with constant pitch. This shows that the inclusion of the additional terms u_a* and u_t* is not significant in this case.

5.3.2.1 Influence of initial solution

OptiPropII depends on the initial assumption of hydrodynamic pitch angle, β_i. For the first iteration, we estimate the initial β_i from the theory for infinite number of blades[2]. Unless

[2] *LerbsNU* used the same method to estimate initial β_i

we give another estimation for β_i, we realize that the solution reached by *OptiPropII* presents a different behavior. Considering two cases:

1. Initial β_i estimated from infinite number of blades;
2. Initial β_i is the optimum solution for β_i given by *LerbsNU*.

The second case was executed as it was lightly loaded, because the pitch is maintained in all iterations. But, in fact, the results reproduce a moderately loaded propeller to a certain extent.

Figs. 36 to 38 show the results obtained from *OptiPropII* for the two cases , the optimum distribution of circulation, induced axial and tangential velocities and the pitch distribution along radius, respectively.

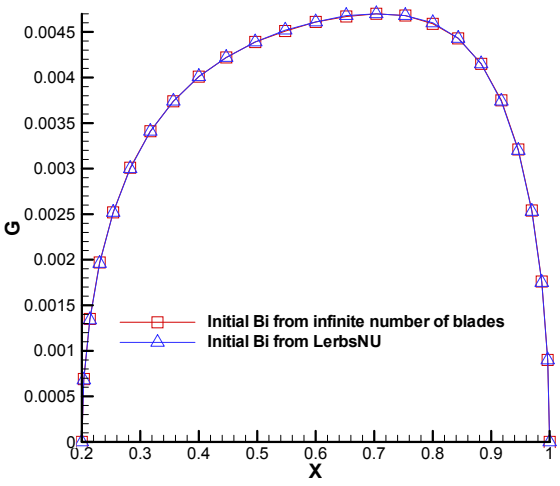

Fig. 36 - Optimum distribution of circulation along radius for N=23, C_T=0.2, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII with initial β_i from infinite number of blades and from LerbsNU.

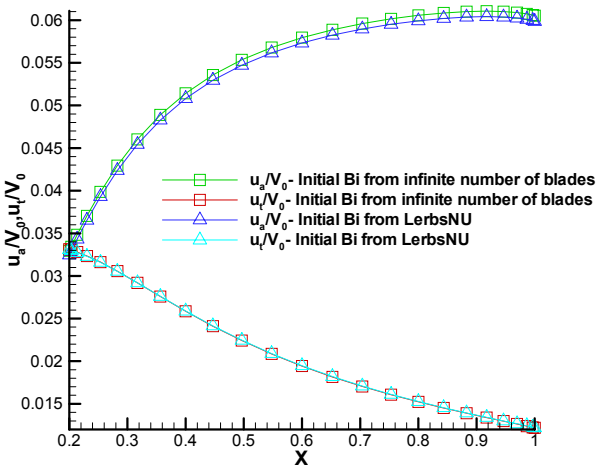

Fig. 37 - Optimum distribution of tangential and axial induced velocities along radius for N=23 C_T=0.2, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII with initial β_i from infinite number of blades and from LerbsNU.

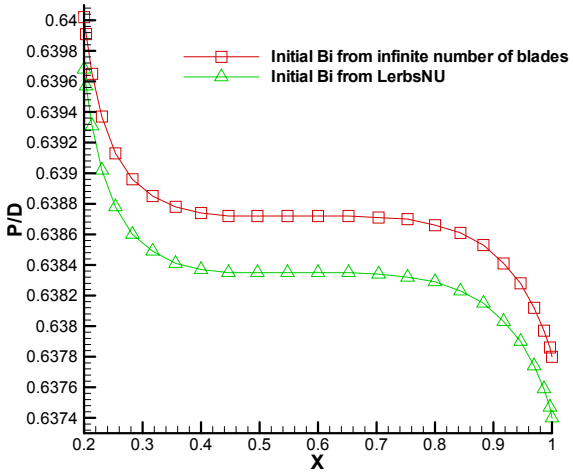

Fig. 38 - Optimum pitch distribution along radius for N=23 C_T=0.2, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII with initial β_i from infinite number of blades and from LerbsNU.

Some discrepancies could be observed mainly on the axial induced velocity and on the pitch distribution. No differences on the results were expected, because the end result is supposed to be independent of the initial parameters assumptions. The end result should have only one solution and, so far, two solutions appear to the same problem. Continuing to change the initial hydrodynamic pitch angle the end result changes as well.

Still notice in fig. 38 that for mean sections the pitch distribution of the second case (initial β_i from *LerbsNU*) have values very close to the result obtained from *LerbsNU* (see fig. 31). Near the hub and tip sections the results deviate.

The history of convergence with iterations for the hydrodynamic pitch angle is shown in **figs. 39 to 41**. It confirms the deviation of results in all sections and that the results are converged for both cases.

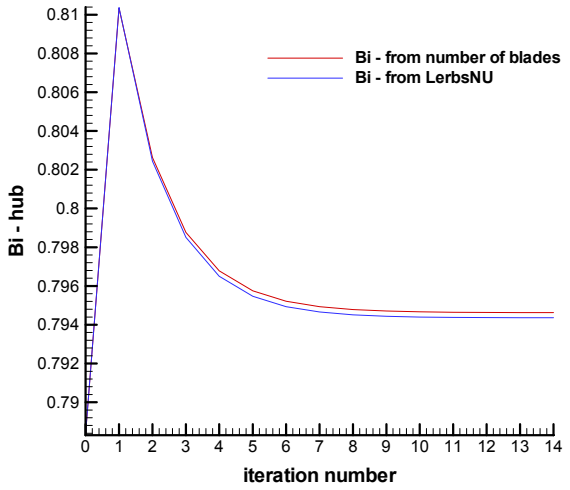

Fig. 39– History of convergence on hydrodynamic pitch angle with the number of iterations for the hub section in the cases where the initial β_i is estimated from infinite number of blades and from LerbsNU. Conditions: N=23, C_T=0.2, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow.

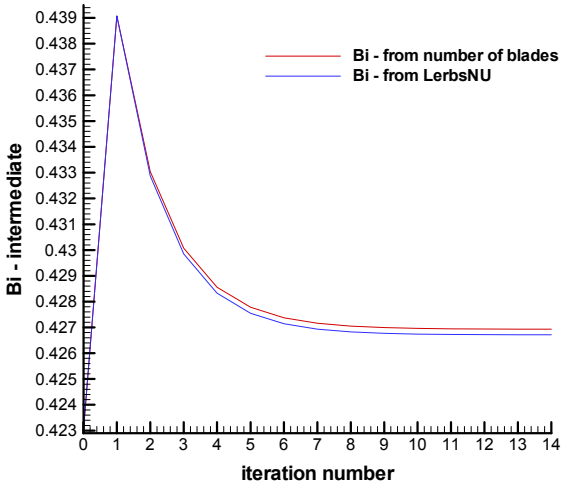

Fig. 40 - History of convergence on hydrodynamic pitch angle with the number of iterations for an intermediate section of the blade in the cases where the initial β_i is estimated from infinite number of blades and from LerbsNU. Conditions: N=23, C_T=0.2, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow.

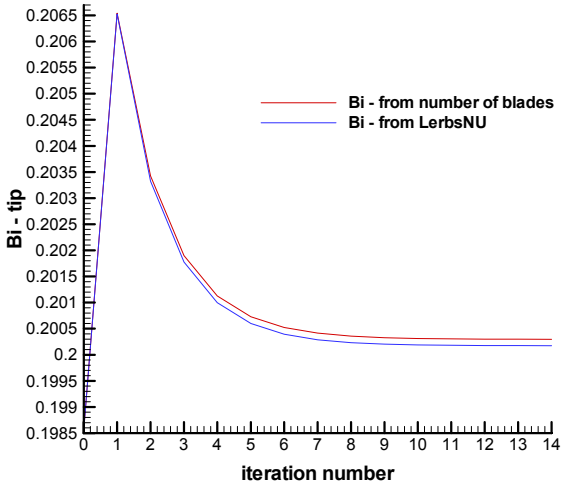

Fig. 41 - History of convergence on hydrodynamic pitch angle with the number of iterations for the tip section in the cases where the initial β_i is estimated from infinite number of blades and from LerbsNU. Conditions: N=23, C_T=0.2, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow.

5.3.2.2 Influence of tolerance

The tolerance is important for the on convergence of the results as it was seen at point 5.3.1.1. Once again, convergence to 5 decimals was reached within a tolerance of 0.001% (see tables of **appendix C.1.2**, **C.2.2**, **C.3.2** and **C.4.1**).

5.3.2.3 Influence of number of points

The evolutions of circulation, tangential and axial induced velocities and pitch along the propeller radius achieved when executing *OptiPropII* for the case study with 11 and 23 internal points are shown in **figs. 42 to 44**.

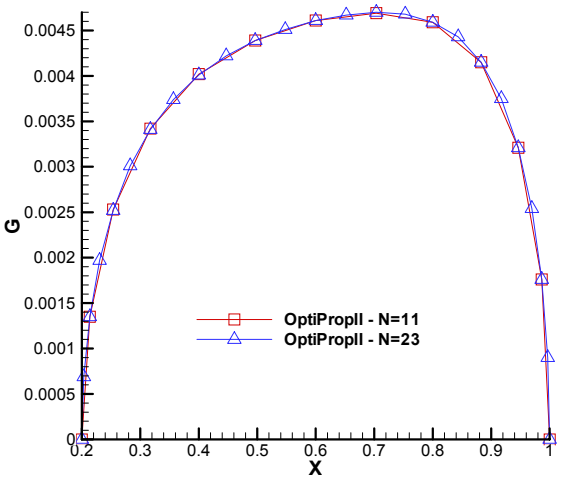

Fig. 42- Optimum distribution of circulation along radius for N=11 and N=23, C_T=0.2, J_0=0.6, tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII.

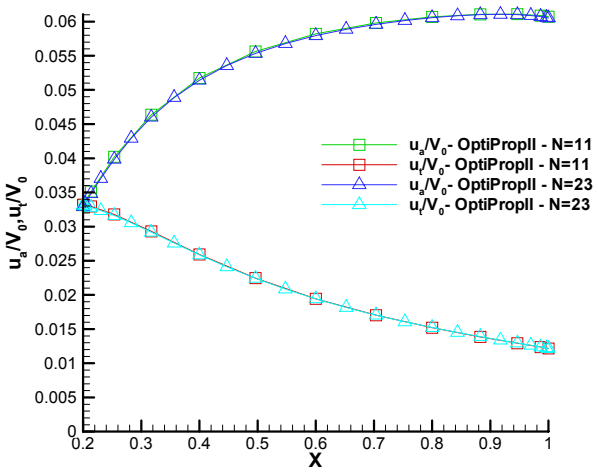

Fig. 43 - Optimum distribution of tangential and axial induced velocities along radius for N=11 and N=23, C_T=0.2, J_0=0.6, tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII.

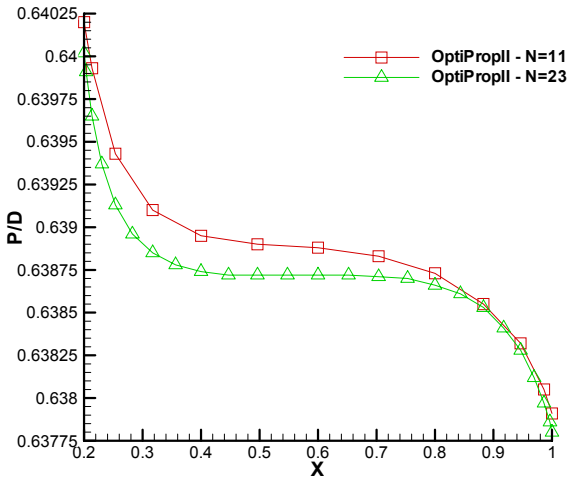

Fig. 44 - Optimum pitch distribution along radius for N=11 and N=23, C_T=0.2, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII.

The statement made when a propeller is lightly loaded and the number of internal points changes, is applied here, with a moderately loaded propeller. A reasonable accurate solution is already obtained with N=11.

The **table 3** contains values for the propeller efficiency, the power coefficient and the number of iterations for 11 and 23 internal points.

	LerbsNU			***OptiProp*** Tolerance = 0.0005%			***OptiPropII*** Tolerance = 0.001%		
N	C_P	η [%]	Nº Iterations	C_P	η [%]	Nº Iterations	C_P	η [%]	Nº Iterations
11	0.213	94.0	3	0.213	94.0	24	0.213	93.9	14
23	0.213	94.0	3	0.213	94.0	101	0.213	93.9	14

Table 3 – Values of efficiency, power coefficient and number of iterations for a moderately loaded propeller with N=11 and N=23, C_T=0.2 and J_o=0.6 obtained from LerbsNU, OptiProp and OptiPropII.

The values of efficiency presented in table 3 have negligible differences, but the number of iterations is rather different. *LerbsNU* needs less iterations than *OptiProp* or *OptiPropII*. However, for the present case *OptiProp* iterates much more times than *OptiPropII*, especially with 23 internal points.

5.3.2.4 Influence of thrust coefficient

At this point, let us consider the following situations:

1. C_T=0.2;
2. C_T=0.512;
3. C_T=1.2.

The remaining data entry for these cases is the same presented for the case study (case 1 is the case study).

The distribution of circulation, the axial and tangential induced velocities and the pitch distribution for the cases 2 and 3 are shown in **figs. 45 to 50**. The results for case 1 were already analyzed in section 5.3.2 (figs. 29 to 31).

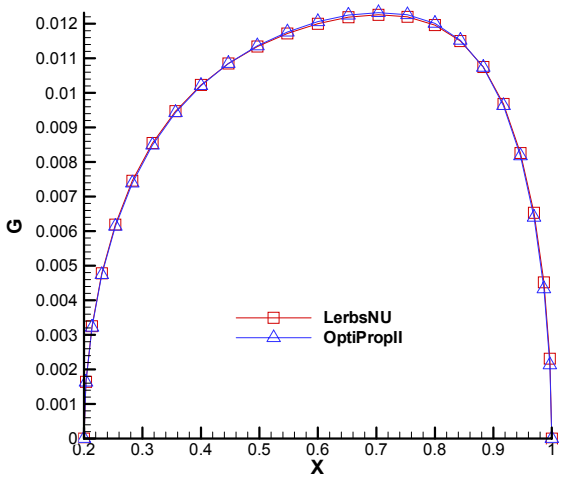

Fig. 45 - Optimum distribution of circulation along radius for N=23, C_T=0.512, J_0=0.6, tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII and LerbsNU.

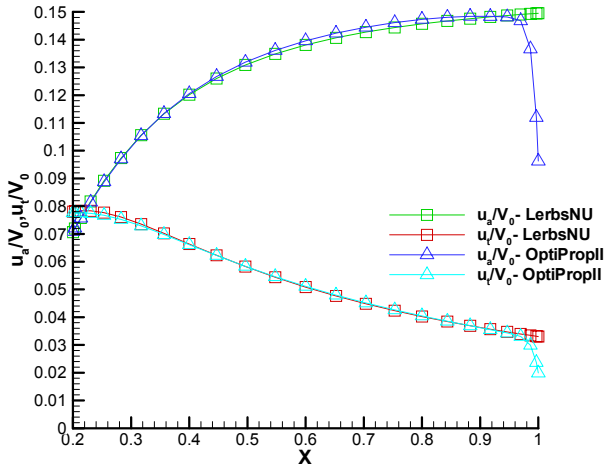

Fig. 46 - Optimum distribution of tangential and axial induced velocities along radius for N=23, C_T=0.512, J_0=0.6, tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII and LerbsNU..

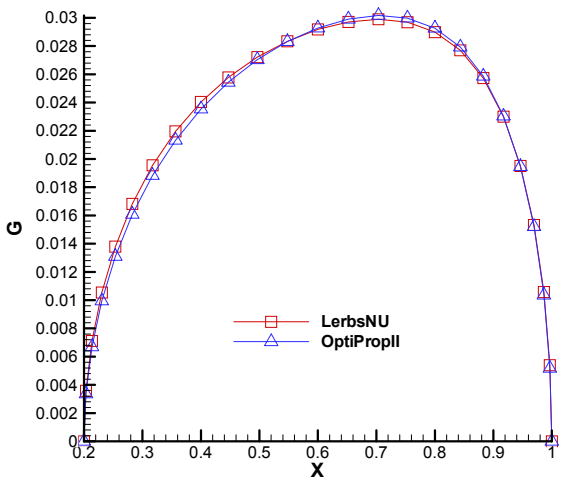

Fig. 47 - Optimum distribution of circulation along radius for N=23, C_T=1.2, J_0=0.6, tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII and LerbsNU.

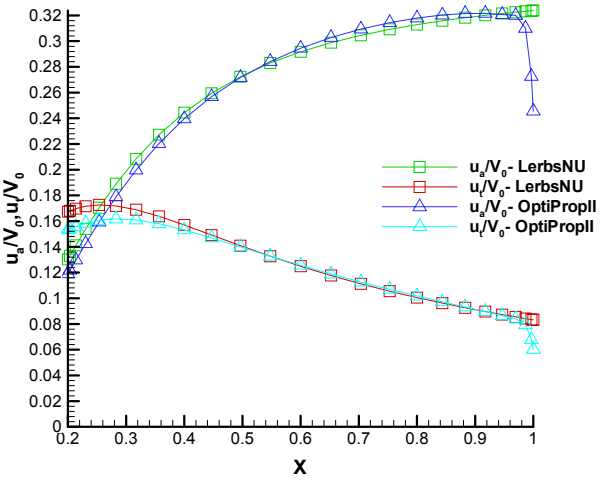

Fig. 48 - Optimum distribution of tangential and axial induced velocities along radius for N=23, C_T=1.2, J_0=0.6, tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII and LerbsNU.

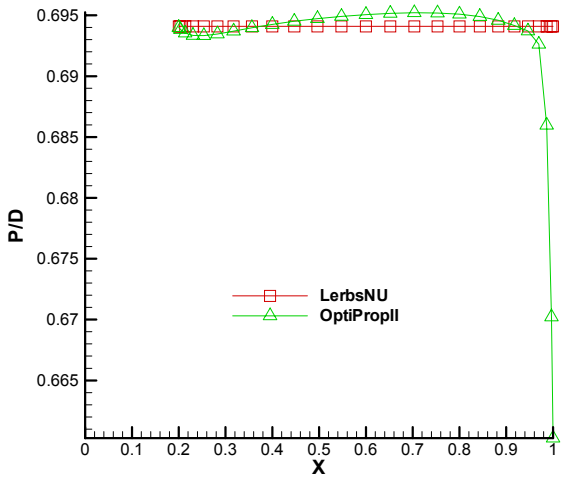

Fig. 49 - Optimum pitch distribution along radius for N=23, C_T=0.512, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII and LerbsNU.

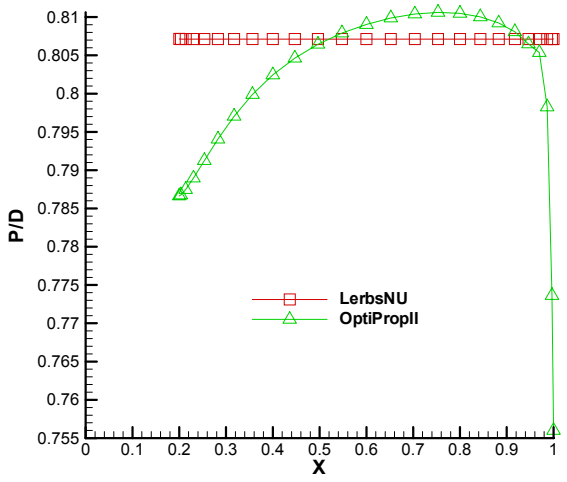

Fig. 50 - Optimum pitch distribution along radius for N=23, C_T=1.2, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII and LerbsNU.

About the distribution of circulation (figs. 45 and 47), the results are very similar. The curves for tangential and axial induced velocities (figs. 46 and 48) have an uncommon behavior at the tips. This situation is reflected in the pitch distribution (figs. 49 and 50).

It was already demonstrated at point 5.3.2.1 that the solution for case 1 obtained from *OptiPropII* is converged and that the convergence is more difficult in the tip sections. So, if there is a convergence for the tip section certainly the others sections are also converged.

For cases 2 and 3, **figs. 51 and 52** show the history of convergence of pitch distribution along iterations only for the tip of propeller radius. There is a blue line in each graphic that marks the iteration when *OptiPropII* gives the end result. In order to know if the result is converged *OptiPropII* iterates more times.

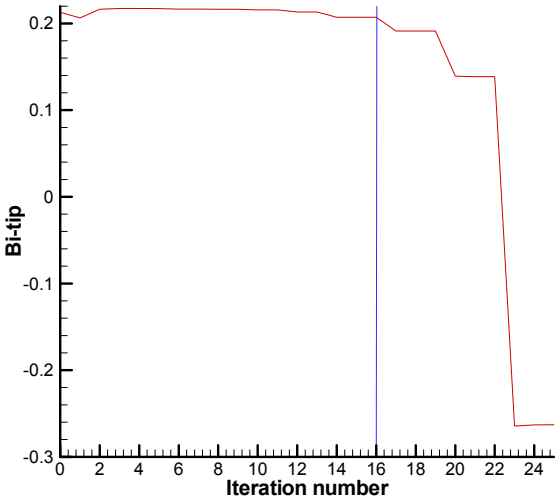

Fig. 51 – History of convergence on hydrodynamic pitch angle for the tip section of propeller radius. Conditions: N=23, C_T=0.512, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII.

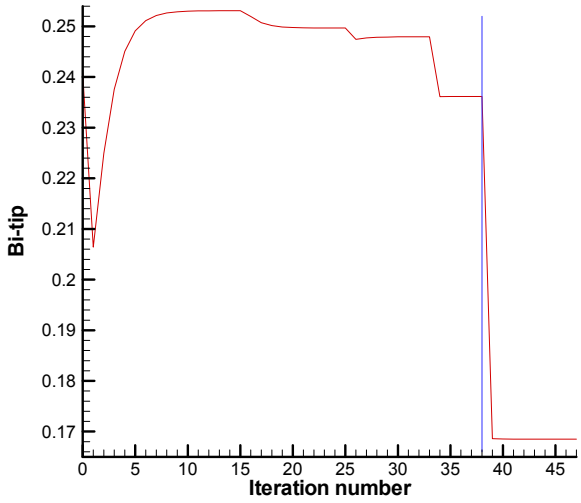

Fig. 52 - History of convergence on hydrodynamic pitch angle for the tip section of propeller radius. Conditions: N=23, C_T=1.2, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII.

The end results provided by *OptiPropII* for cases 2 and 3 are not converged. Therefore, this is the cause for the uncommon behavior observed in figs. 45, 47, 48 and 49.

Reducing the number of internal points to 11 the correspondingly results are shown in **figs. 53 and 54**.

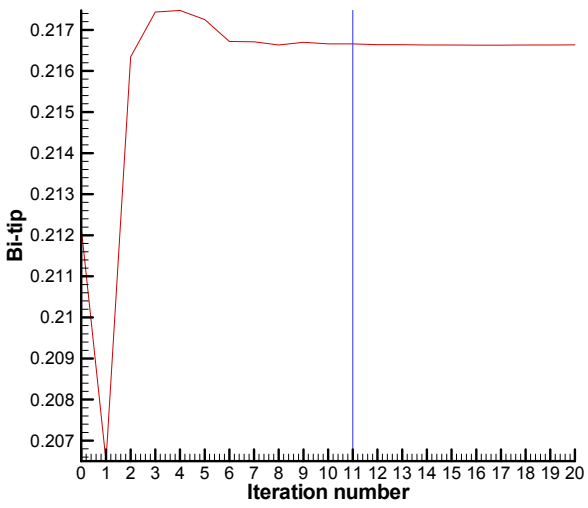

Fig. 53 – History of convergence on hydrodynamic pitch angle for the tip section of propeller radius. Conditions: N=11, C_T=0.512, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII.

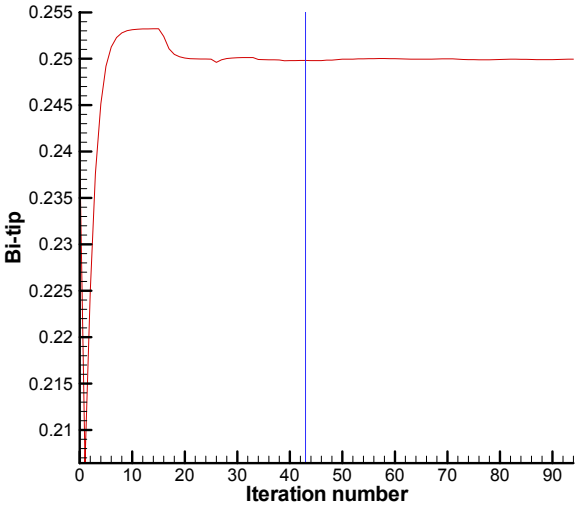

Fig. 54 - History of convergence on hydrodynamic pitch angle for the tip section of propeller radius. Conditions: N=11, C_T=1.2, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII.

For 11 internal points, cases 2 and 3 converged. **figs. 55 to 60** show the results for these cases with 11 internal points.

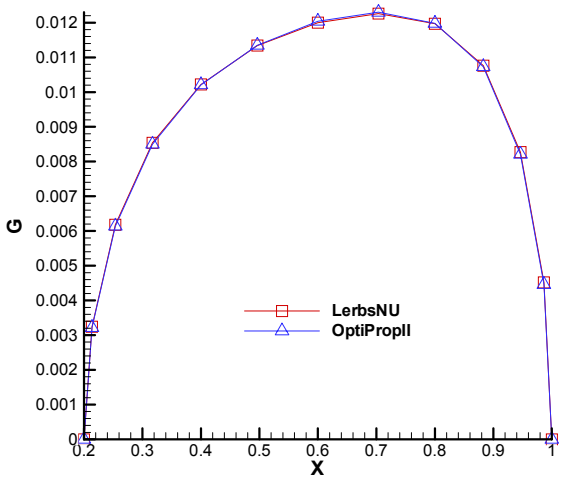

Fig. 55- Optimum distribution of circulation along radius for N=11, C_T=0.512, J_0=0.6, tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII and LerbsNU.

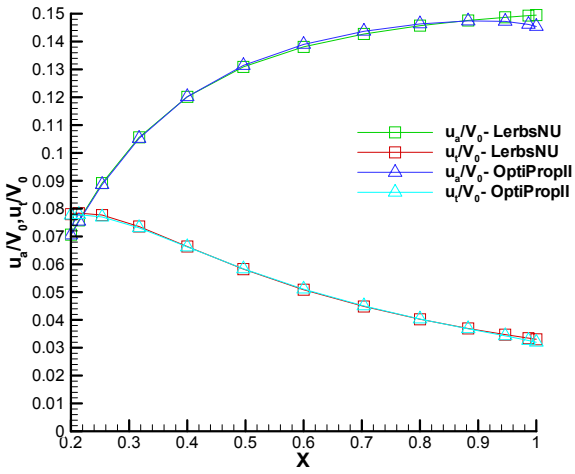

Fig. 56 - Optimum distribution of tangential and axial induced velocities along radius for N=11, C_T=0.512, J_0=0.6, tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII and LerbsNU..

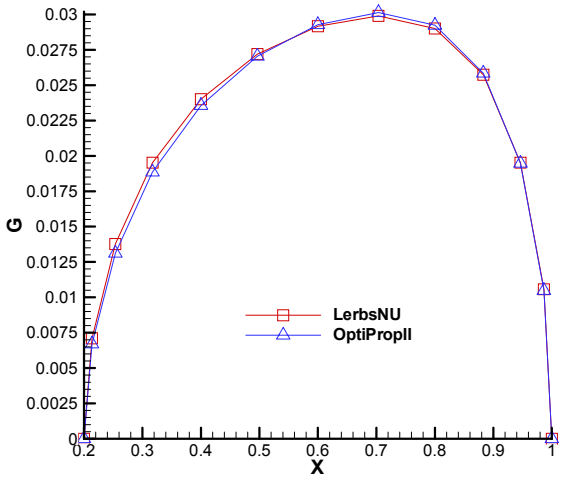

Fig. 57 - Optimum distribution of circulation along radius for N=11, C_T=1.2, J_0=0.6, tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII and LerbsNU.

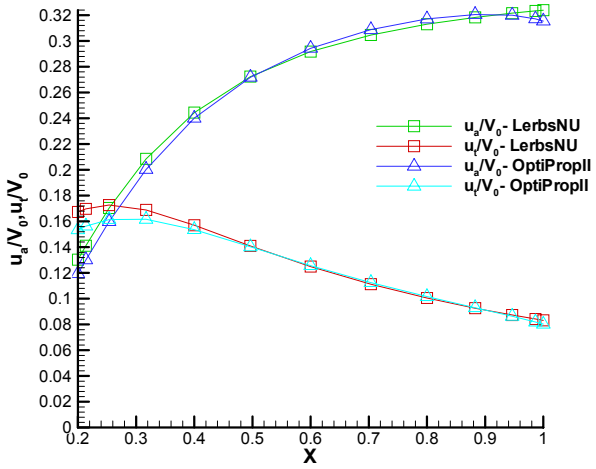

Fig. 58 - Optimum distribution of tangential and axial induced velocities along radius for N=11, C_T=1.2, J_0=0.6, tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII and LerbsNU.

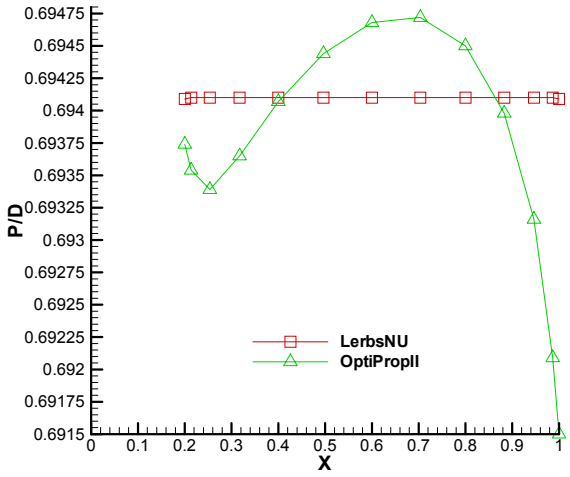

Fig. 59 - Optimum pitch distribution along radius for N=11, C_T=0.512, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII and LerbsNU.

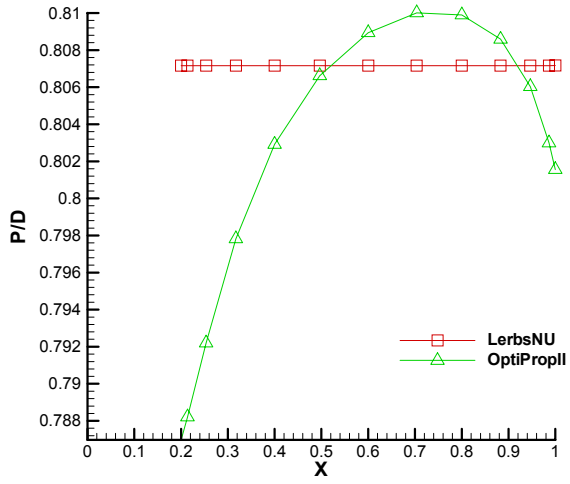

Fig. 60 - Optimum pitch distribution along radius for N=11, C_T=1.2, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII and LerbsNU.

The axial and tangential induced velocities have the expected behavior. The pitch distribution deviates mainly in the tip and hub regions.

Table 4 shows the values obtained for the propeller efficiency, the power coefficient and the number of iterations that *OptiPropII* iterates to give the results.

	LerbsNU				***OptiPropII*** Tolerance = 0.001%			
N	C_P	C_T	η [%]	Nº Iterations	C_P	C_T	η [%]	Nº Iterations
11	0.592	0.512	86.4	182	0.5924	0.512	86.4	11
23	0.592	0.512	86.4	184	0.5927	0.512	86.4	16
11	1.614	1.2	74.3	3	1.6150	1.2	74.3	43
23	1.614	1.2	74.3	3	1.6152	1.2	74.3	38

Table 4 – Values of efficiency, power coefficient and number of iterations for a moderately loaded propeller with N=11 and N=23, C_T=0.512 and C_T=1.2 and J_o=0.6 obtained from LerbsNU and OptiPropII.

In *OptiPropII* the number of iterations increases with the thrust coefficient. Note that for a thrust coefficient of 0.512 *LerbsNU* iterates numerous times for 23 internal points, as well as, 11 internal points. In this case, for *LerbsNU* the convergence is more difficult to reach. With a thrust coefficient of 1.2, the opposite occurs. Remember that the results given by *OptiPropII* for 23 internal points in both cases are not converged.

The values of the propeller efficiency, are identical.

5.3.2.5 Influence of advance coefficient

The pitch distribution depends of the thrust and on advance coefficient. Maintaining the thrust coefficient, the evolution of hydrodynamic pitch along radius presents different behaviors for each advance coefficient (**figs. 61 to 68**).

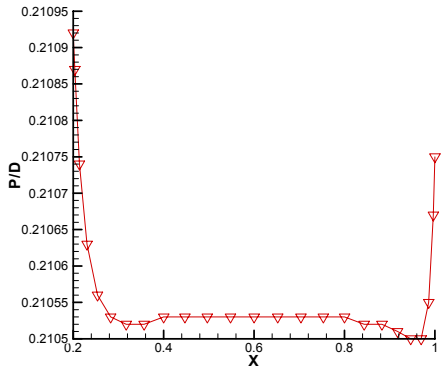

Fig. 61 - Optimum pitch distribution along radius for N=23, C_T=0.2, J_0=0.2 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII.

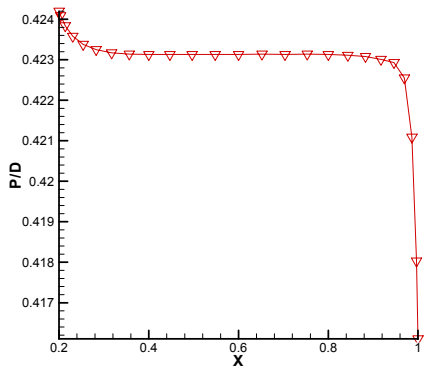

Fig. 62 - Optimum pitch distribution along radius for N=23, C_T=0.2, J_0=0.4 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII.

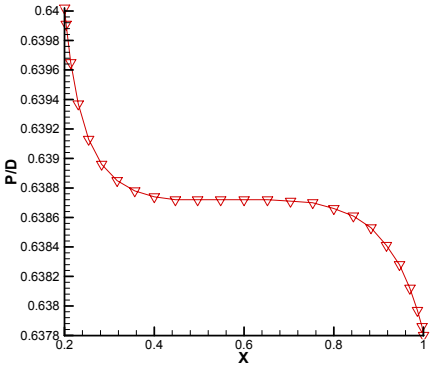

Fig. 63 - Optimum pitch distribution along radius for N=23, C_T=0.2, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII.

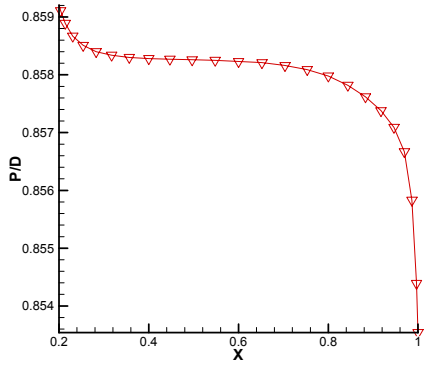

Fig. 64 - Optimum pitch distribution along radius for N=23, C_T=0.2, J_0=0.8 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII.

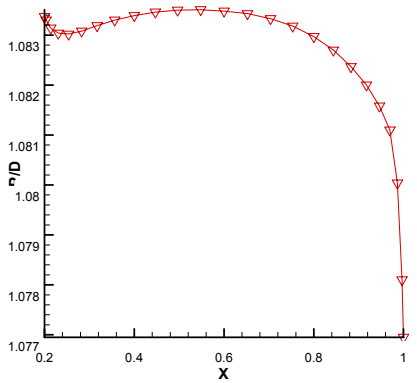

Fig. 65 - Optimum pitch distribution along radius for N=23, C_T=0.2, J_0=1.0 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII.

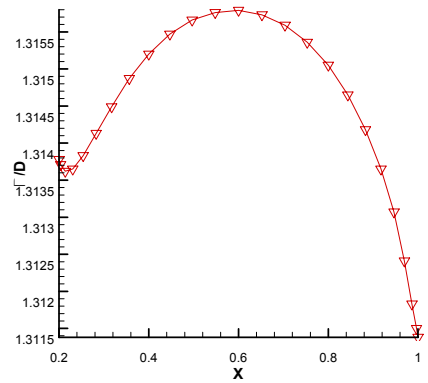

Fig. 66 - Optimum pitch distribution along radius for N=23, C_T=0.2, J_0=1.2 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII.

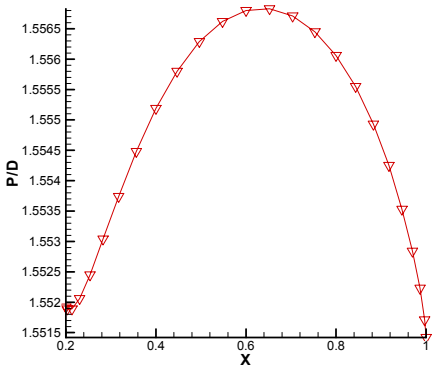

Fig. 67 - Optimum pitch distribution along radius for N=23, C_T=0.2, J_0=1.4 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII.

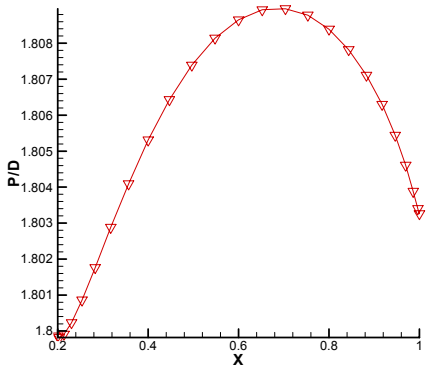

Fig. 68 - Optimum pitch distribution along radius for N=23, C_T=0.2, J_0=1.6 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII.

Table 5 contains the results obtained from *OptiPropII* and *LerbsNU*, for the efficiency, the power coefficient and the number of iterations.

	LerbsNU			***OptiPropII*** Tolerance = 0.001%		
J_0	C_P	η [%]	Nº Iterations	C_P	η [%]	Nº Iterations
0.2	0.210	95.2%	3	0.2105	95.0%	66
0.4	0.211	94.6%	3	0.2115	94.6%	40
0.6	0.213	94.0%	3	0.2129	93.9%	14
0.8	0.214	93.3%	5	0.2145	93.2%	188
1.0	0.216	92.4%	6	0.2166	92.4%	64
1.2	0.219	91.3%	8	0.2192	91.2%	345
1.4	0.222	90.0%	10	0.2222	90.0%	42
1.6	0.226	88.5%	13	0.2260	88.5%	49

Table 5 – Values of efficiency, power coefficient and number of iterations for a moderately loaded propeller with N=23, C_T=0.2 obtained from LerbsNU and OptiPropII.

The differences encountered on the efficiency are insignificant. *LerbsNU* needs less iterations than *OptiPropII*.

Note that the optimum pitch distribution obtained from *LerbsNU* is constant for all of the examples of table 5.

5.3.2.6 Comparison with the application of Munk's displacement theorem

When we solve the problem with respect to expression (31):

$$\frac{u_a^*}{V_0}(\theta_i) \cong \frac{u_a(\theta_i)}{V_0} \text{ and } \frac{u_t^*}{V_0}(\theta_i) \cong \frac{u_t(\theta_i)}{V_0}$$

the results obtained are affected.

Figs. 69 to 71 show a comparison between the results obtained considering the terms $u_a^*(\theta_i)$ and $u_t^*(\theta_i)$ and when this terms are assumed to be the axial and tangential induced velocities, respectively. This corresponds to the application of Munk's displacement theory. **Fig. 72** shows the differences encountered between the terms and the axial and induced velocities.

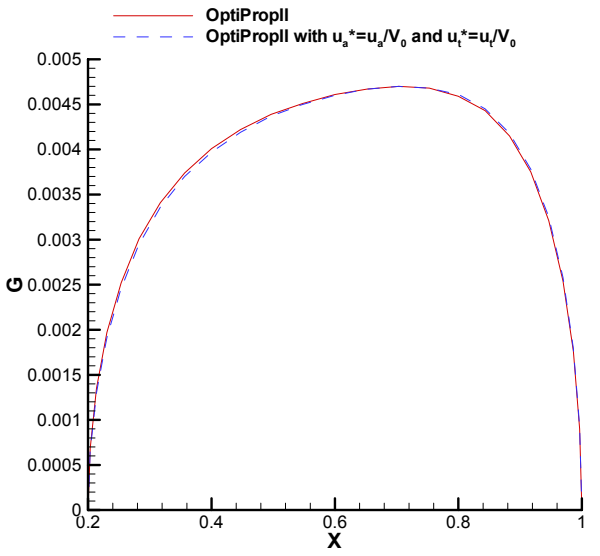

Fig. 69 - Optimum distribution of circulation along radius for N=23, C_T=0.2, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII considering the terms (28) and (29) and considering the assumption of equation (31).

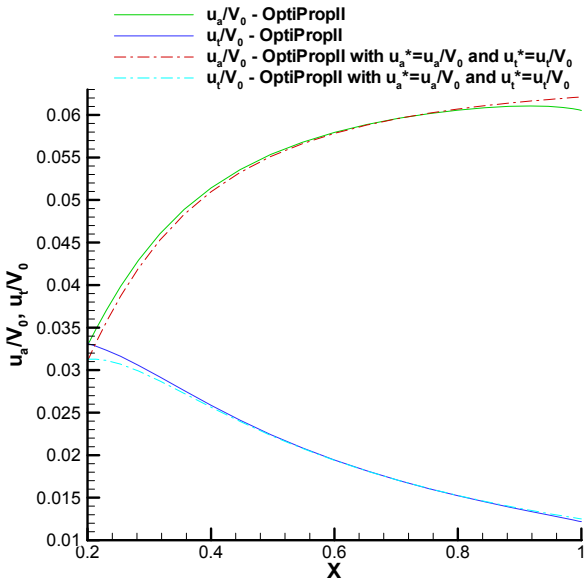

Fig. 70 - Optimum distribution of tangential and axial induced velocities along radius for N=23, C_T=0.2, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII considering the terms (28) and (29) and considering the assumption of equation (31).

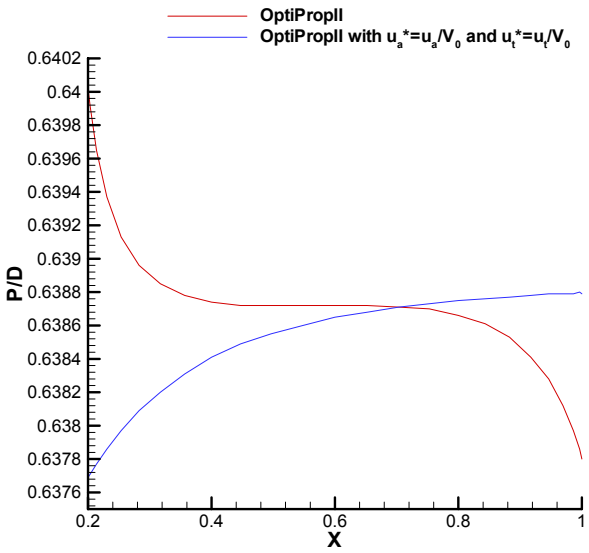

Fig. 71 - Optimum pitch distribution along radius for N=23, C_T=0.2, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII considering the terms (28) and (29) and considering the assumption of equation (31).

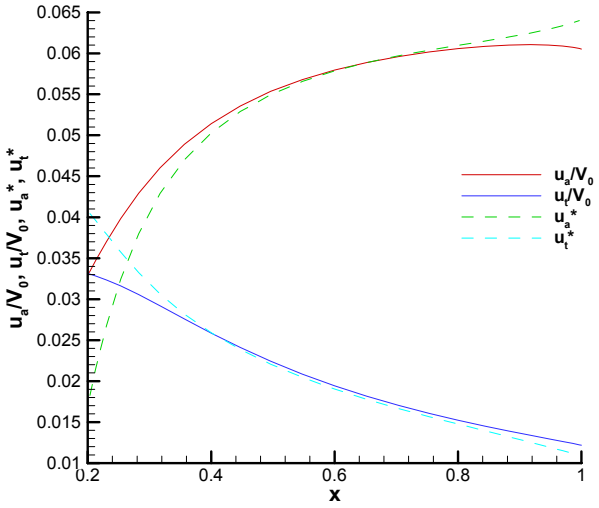

Fig. 72 - Optimum distribution of tangential and axial induced velocities and the terms (28) and (29) along radius for N=23, C_T=0.2, J_0=0.6 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII.

The differences found in the results are mainly near the hub radius and blade tip (figs. 69 to 71). For this propeller loading those differences are insignificant. In fig. 72 we can see that the terms deviate from the axial and tangential velocities near the tip and hub radius, but these differences do not affect significantly the end results.

Adding a note, the terms were estimated only on the internal points. They are calculated from the distribution of circulation and from the transpose of the induction velocities matrices. These matrices are rectangular and the position that corresponds to the extremes in the transpose of matrices has to be deduced. The values of the terms determined on the internal points permit to perceive the differences between the two methods.

Table 6 presents the propeller efficiency, the power coefficient and number of iterations for both methods. All the results match.

		OptiPropII			*OptiPropII* with assumption (31)		
C_T	J_0	C_P	η [%]	Nº Iterations	C_P	η [%]	Nº Iterations
0.2	0.6	0.2129	93.9	14	0.2129	93.9	14

Table 6 – Values of efficiency, power coefficient and number of iterations for a moderately loaded propeller with N=23, Z=5, C_T=0.2, J_0=0.6 and tolerance=0.001% obtained from OptiPropII with the terms and considering the assumption (31).

In order to compare the results achieved by *OptiPropII* and the published report of Andersen[8], lets consider three different cases for a four bladed propeller:

1. C_T=1.0 and J_0=0.727;
2. C_T=2.0 and J_0=0.525;
3. C_T=3.0 and J_0=0.442;

Figs. 73 to 75 illustrate the axial and tangential induced velocities and the terms for the three cases presented.

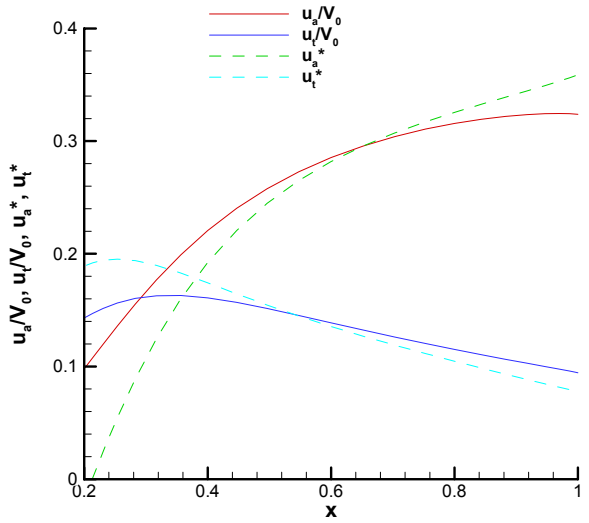

Fig. 73 - Optimum distribution of tangential and axial induced velocities and the terms (28) and (29) along radius for N=23, C_T=1.0, J_0=0.727, moderate loading and uniform flow. Results from OptiPropII.

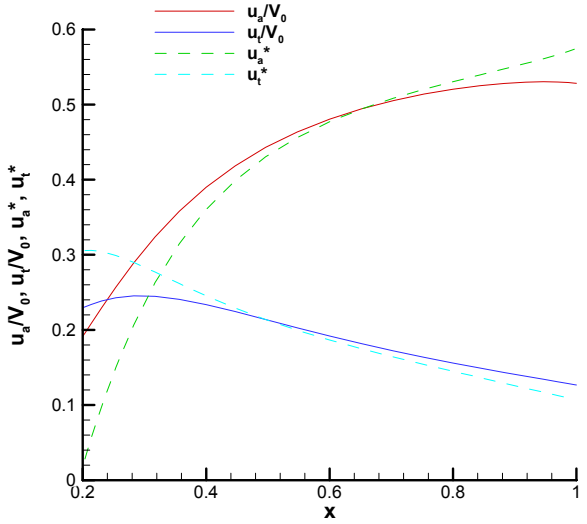

Fig. 74 - Optimum distribution of tangential and axial induced velocities and the terms (28) and (29) along radius for N=23, C_T=2.0, J_0=0.525, moderate loading and uniform flow. Results from OptiPropII.

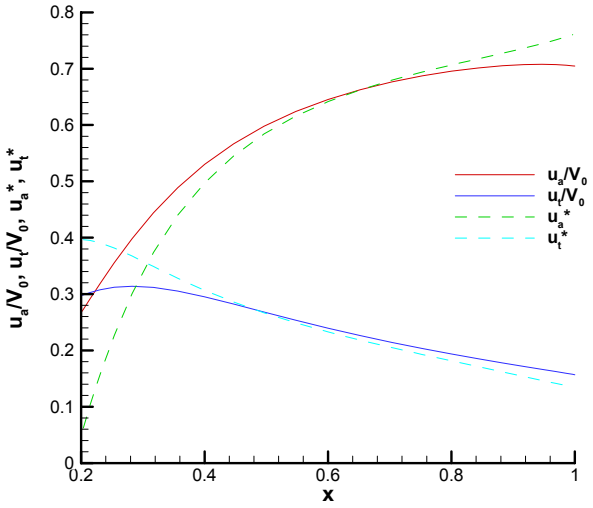

Fig. 75 - Optimum distribution of tangential and axial induced velocities and the terms (28) and (29) along radius for N=23, C_T=3.0, J_0=0.442, moderate loading and uniform flow. Results from OptiPropII.

Comparing the terms and the induced velocities, we note that near the tip and the hub the largest differences occur. As the thrust coefficient increases the differences between the terms and the induced velocities also increases. The results shown in figs. 73 to 75 are in good agreement with the results of Andersen[8].

Considering yet the distinct situations:

1. Formulation of equation (33);
2. Formulation with the assumption expressed in equation (31) that is the standard theory with application of Munk's displacement theorem;
3. Formulation of equation (33) neglecting the terms u_a^* and u_t^* that is the standard theory (classical optimum).

Figs. 76 and 77 show the optimum distribution of circulation and pitch, respectively, for the three situations and cases described above. The results achieved with *OptiPropII* and considering the assumption (31) are in close agreement. We can see that with increasing loading the deviations found between the standard theory and the formulation of equation (33) increase. Once more, Andersen[8] has reached the same conclusion.

In the standard theory, the pitch distribution is constant. That does not happen with the other two formulations. We may see (fig. 77) that the differences between the results with the terms in (33) and the assumption of (31) is of the order of 3%. Therefore, those differences are insignificant.

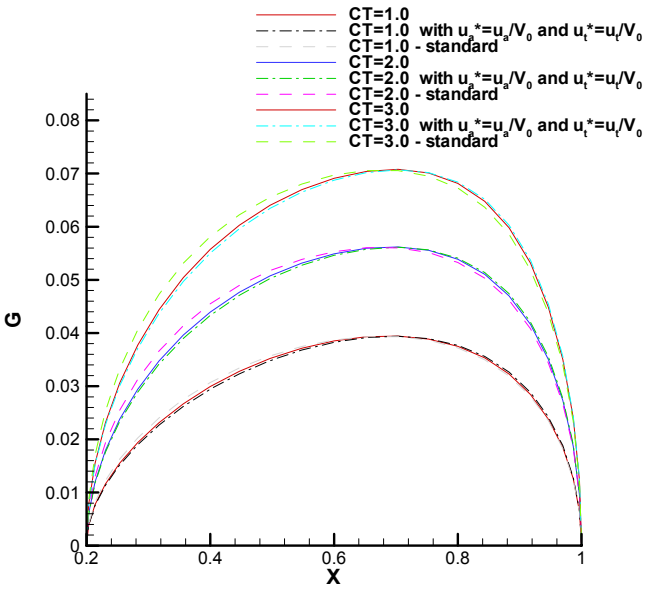

Fig. 76 - Optimum distribution of circulation along radius for N=23, C_T=1.0, C_T=2.0, C_T=3.0, moderate loading and uniform flow. Results from OptiPropII considering the terms (28) and (29), the assumption of equation (31) and the standard formulation.

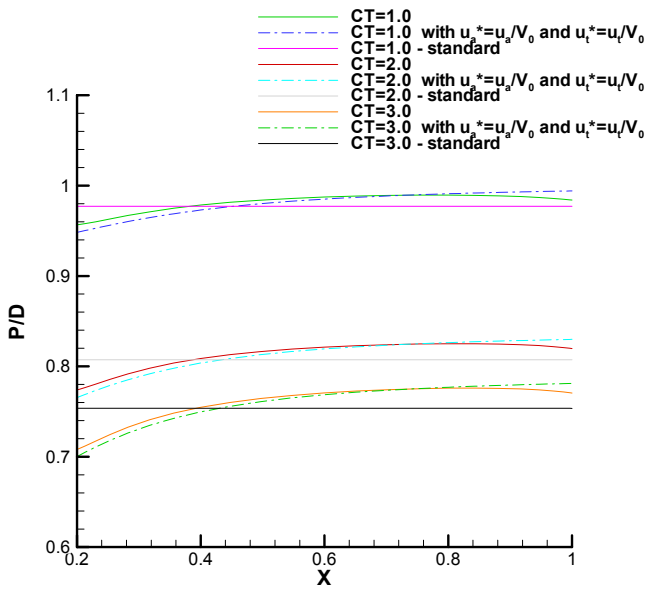

Fig. 77 - Optimum pitch distribution along radius for N=23, C_T=1.0, C_T=2.0, C_T=3.0, moderate loading and uniform flow. Results from OptiPropII considering the terms (28) and (29), the assumption of equation (31) and the standard formulation.

In **figs. 78 to 80** we can see that the formulations using the terms and the formulation with the application of Munk's displacement theorem give practically identical induced velocities.

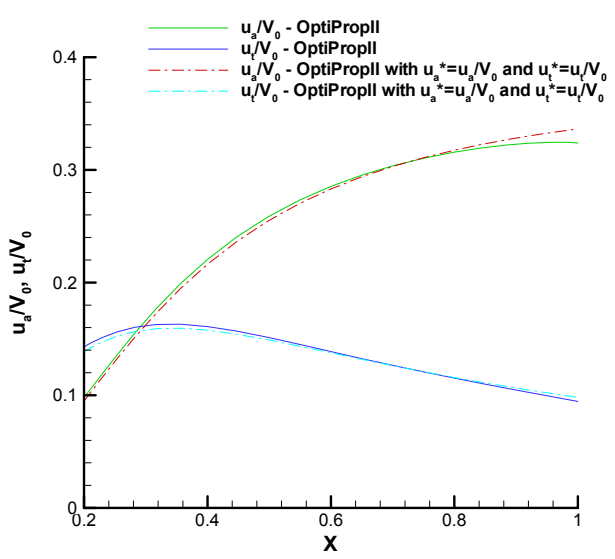

Fig. 78 - Optimum distribution of tangential and axial induced velocities along radius for N=23, C_T=1.0, J_0=0.727 and tolerance=0.0005%, moderate loading and uniform flow. Results from OptiPropII considering the terms (28) and (29) and considering the assumption of equation (31).

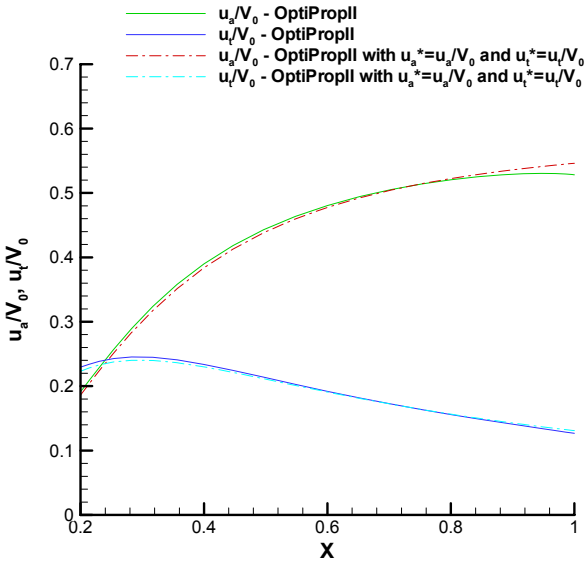

Fig. 79 - Optimum distribution of tangential and axial induced velocities along radius for N=23, C_T=2.0, J_0=0.525 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII considering the terms (28) and (29) and considering the assumption of equation (31).

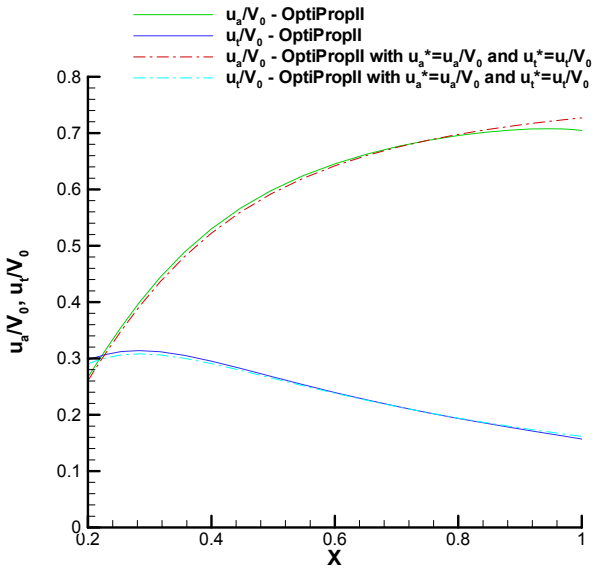

Fig. 80 - Optimum distribution of tangential and axial induced velocities along radius for N=23, C_T=3.0, J_0=0.442 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropII considering the terms (28) and (29) and considering the assumption of equation (31).

Table 7 gives the results for the efficiency, power coefficient and the number of iterations for the three cases tested with the three methods.

		OptiPropII			***OptiPropII*** with assumption (31)			***OptiPropII*** standard		
C_T	J_0	C_P	η [%]	Nº Iterations	C_P	η [%]	Nº Iterations	C_P	η [%]	Nº Iterations
1.0	0.727	1.3566	73.7	27	1.3566	73.7	16	1.3440	74.4	24
2.0	0.525	3.1248	64.0	23	3.1249	64.0	23	3.0752	65.0	44
3.0	0.442	5.2239	57.4	27	5.2241	57.4	27	5.1144	58.7	65

Table 7 – Values of efficiency, power coefficient and number of iterations for a moderately loaded propeller with N=23, Z=4, C_T=1.0, C_T=2.0 and C_T=3.0 obtained from OptiPropII with the terms, considering the assumption (31) and with the standard method.

The results obtained from *OptiPropII* with the terms of equation (33) and with the assumption (31) are in close agreement. Only the standard formulation presents some differences on the efficiency and the power coefficient. We note that the divergence of the efficiency increases with loading. This is a consequence of the larger differences found in the distribution of circulation when the loading increases (fig. 76).

5.3.3 Discussion of results

The improvement made to the formulation in this new approach to the problem still has some adversities. The convergence continues to be very sensitive to the parameters:

- initial β_i assumptions;
- thrust coefficient;
- number of internal points;
- tolerance;
- advance coefficient;
- terms $u_a^*(\theta_i)$ and $u_t^*(\theta_i)$.

In lightly loaded assumptions, we reach to a new optimum solution to the problem. We obtained a variable pitch distribution with the same efficiency as the solution with constant pitch distribution.

In moderately loaded assumptions, there are problems with the convergence, especially when loading increases. The convergence depends severely of the loading conditions, i.e., of the combination of thrust and advance coefficients.

We stated that the method solved considering the terms given by expressions (28) and (29) and the method with the assumption expressed in equation (31), provide approximately the same results. The differences between the methods are insignificant.

The differences obtained between the standard theory and the complete formulation with terms as in equation (33), becomes significant when loading increases. It can have a difference more than 1% in the propeller efficiency for a moderately loaded propeller.

6. MATHEMATICAL FORMULATION OF THE PROBLEM INCLUDING VISCOUS DRAG EFFECTS

For the problem with real fluid and including induced velocities, we have the following diagrams for the velocity triangles and forces acting on a propeller blade section:

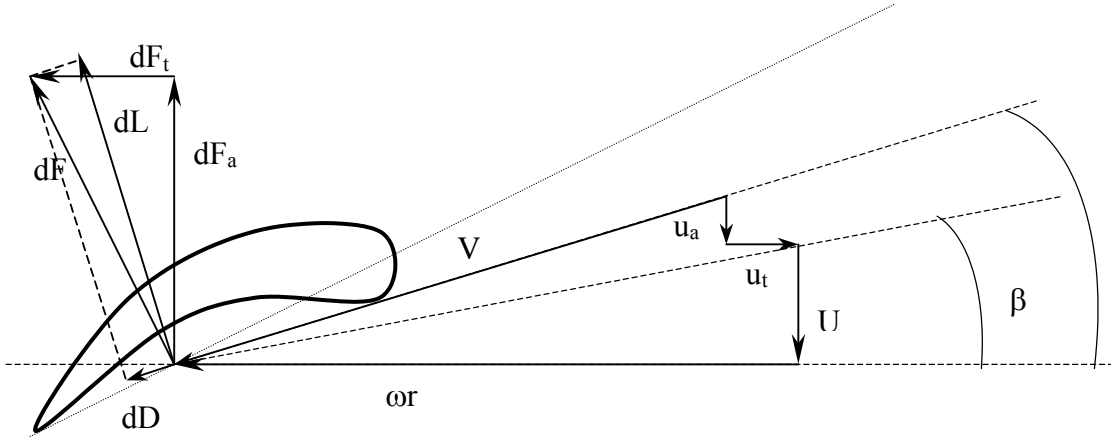

Fig. 81 – Forces acting on a section blade (real fluid)

With respect to the triangle, we have the total thrust and torque for a propeller blade given by[3]:

$$T = \rho Z \int_{R_h}^{R} \left(\Gamma V \cos\beta_i - \tfrac{1}{2} c(r) C_D \sin\beta_i V(r)^2 \right) dr \tag{35}$$

$$Q = \rho Z \int_{R_h}^{R} r \left(\Gamma V \sin\beta_i + \tfrac{1}{2} c(r) C_D \sin\beta_i V(r)^2 \right) dr \tag{36}$$

This expressions are obtained integrating the differential forces acting at any radius of the blade section given by the Kutta-Joukowsky law and summing the number of blades.

Therefore, the nondimensional thrust and power coefficients are:

$$C_T = 4Z \int_{x_h}^{1} \frac{V(x)}{V_0} G(x) \cos\beta_i dx - \frac{Z}{\pi R} \int_{x_h}^{1} c(x) C_D \sin\beta_i \left(\frac{V(x)}{V_0} \right)^2 dx \tag{37}$$

$$C_P = \frac{4Z}{\lambda} \int_{x_h}^{1} \frac{V(x)}{V_0} G(x) \sin\beta_i x dx + \frac{\omega Z}{\pi V_0} \int_{x_h}^{1} c(x) C_D \cos\beta_i \left(\frac{V(x)}{V_0} \right)^2 x dx \tag{38}$$

By definition of lift coefficient:

$$C_L = \frac{L}{\frac{1}{2}\rho V^2 c} = \frac{2\Gamma}{V \cdot c} \tag{39}$$

With expression **(12)** we can solve equation **(39)** in order to the chord length:

$$c = \frac{4\pi R G V_0}{C_L \cdot V} \tag{40}$$

Let us consider the sectional drag-to-lift ratio to be:

$$\varepsilon = \frac{C_D}{C_L} \tag{41}$$

And substituting equations **(40)** and **(41)** in expressions **(37)** and **(38)**:

$$C_T = 4Z \int_{x_h}^{1} G(x)\left(\frac{x}{\lambda} - \frac{u_t(x)}{V_0}\right)(1 - \varepsilon tg\beta_i)dx \tag{42}$$

$$C_P = \frac{4Z}{\lambda} \int_{x_h}^{1} x \cdot G(x)\left(\frac{U(x) + u_a(x)}{V_0}\right)\left(1 + \frac{\varepsilon}{tg\beta_i}\right)dx \tag{43}$$

The procedure is analogous to the problem with inviscid fluid.

Equations **(42)** and **(43)** take the following form before the variable change given by expression **(13)**:

$$C_T = 2Z(1 - x_h)\int_0^{\pi} G(\theta)\left(\frac{x(\theta)}{\lambda} - \frac{u_t(\theta)}{V_0}\right)(1 - \varepsilon tg\beta_i)\sin\theta d\theta \tag{44}$$

$$C_P = \frac{2Z(1 - x_h)}{\lambda}\int_0^{\pi} G(\theta)\left(\frac{U(\theta) + u_a(\theta)}{V_0}\right)\left(1 + \frac{\varepsilon}{tg\beta_i}\right)x(\theta)\sin\theta d\theta \tag{45}$$

Applying the numeric methods mentioned, the Fourier series and the trapezoid rule in the last two equations:

$$C_T = \frac{2\pi \cdot Z(1 - x_h)}{N + 1}\sum_{i=0}^{N+1} G(\theta_i)\left(\frac{x(\theta_i)}{\lambda} - \frac{u_t(\theta_i)}{V_0}\right)(1 - \varepsilon tg\beta_i)\sin\theta_i \tag{46}$$

$$C_P = \frac{2\pi \cdot Z(1 - x_h)}{(N + 1)\lambda}\sum_{i=0}^{N+1} G(\theta_i)\left(\frac{U(\theta_i) + u_a(\theta_i)}{V_0}\right)\left(1 + \frac{\varepsilon}{tg\beta_i}\right)x(\theta_i)\sin\theta_i \tag{47}$$

6.1 Optimum Circulation Distribution

Similarly to the inviscid fluid, we can apply the Method of Lagrange Multipliers to solve the problem:

$$F = C_P + \ell C_T \Rightarrow \frac{\partial F}{\partial G(\theta_i)} = 0 = \frac{\partial C_P}{\partial G(\theta_i)} + \ell \frac{\partial C_T}{\partial G(\theta_i)} \Leftrightarrow$$

$$\Leftrightarrow \frac{U_i}{V_0} x_i \sin\theta_i \left(1 + \frac{\varepsilon}{tg\beta_i}\right) + \sum_{j=1}^{N} G_j M_a^{**}(\theta_i, \theta_j) + \ell\lambda \left(\frac{x_i}{\lambda} \sin\theta_i (1 - \varepsilon tg\beta_i) - \sum_{j=1}^{N} G_j M_t^{**}(\theta_i, \theta_j)\right) = 0 \quad (48)$$

where:

$$\begin{cases} M_a^{**}(\theta_i, \theta_j) = M_a(\theta_i, \theta_j) x_i \sin\theta_i \left(1 + \frac{\varepsilon}{tg\beta_i}\right) + M_a(\theta_j, \theta_i) x_j \sin\theta_j \left(1 + \frac{\varepsilon}{tg\beta_j}\right) \\ M_t^{**}(\theta_i, \theta_j) = M_t(\theta_i, \theta_j) \sin\theta_i (1 - \varepsilon tg\beta_i) + M_t(\theta_j, \theta_i) \sin\theta_j (1 - \varepsilon tg\beta_i) \end{cases}$$

The system equation to be solved is:

$$\begin{cases} \frac{U(\theta_i)}{V_0} x_i \sin\theta_i \left(1 + \frac{\varepsilon}{tg\beta_i}\right) + \sum_{j=1}^{N} G_j M_a^{**}(\theta_i, \theta_j) \\ \qquad + \ell\lambda \left(\frac{x_i}{\lambda} \sin\theta_i (1 - \varepsilon tg\beta_i) - \sum_{j=1}^{N} G_j M_t^{**}(\theta_i, \theta_j)\right) = 0 \\ C_T = \frac{2\pi \cdot (1 - x_h)}{N + 1} \sum_{j=1}^{N} G_j \left(\frac{x_j}{\lambda} - \frac{u_t(\theta_j)}{V_0}\right)(1 - \varepsilon tg\beta_j) \sin\theta_j \end{cases} \quad (49)$$

Although similar to the new approach to the problem with inviscid fluid, equation **(48)** can be written in another form:

$$\frac{U_i / V_0 + u_a(\theta_i) / V_0 + u_a^{**}(\theta_i) / V_0}{x_i / \lambda - u_t(\theta_i) / V_0 - u_t^{**}(\theta_i) / V_0} = -\ell \frac{\lambda}{x_i} \quad (50)$$

where:

$$\frac{u_a^{**}(\theta_i)}{V_0} = \frac{\varepsilon}{tg\beta_i}\left(\frac{U(\theta_i)}{V_0} + \frac{u_a(\theta_i)}{V_0}\right) + \frac{1}{x_i \sin\theta_i}\sum_{j=1}^{N} G_j M_a(\theta_j,\theta_i) x_j \sin\theta_j \left(1 + \frac{\varepsilon}{tg\beta_j}\right) \tag{51}$$

$$\frac{u_t^{**}(\theta_i)}{V_0} = \varepsilon tg\beta_i\left(\frac{x_i}{\lambda} - \frac{u_t(\theta_i)}{V_0}\right) + \frac{1}{\sin\theta_i}\sum_{j=1}^{N} G_j M_t(\theta_j,\theta_i)\sin\theta_j\left(1 - \varepsilon tg\beta_j\right) \tag{52}$$

If $\varepsilon = 0$ (inviscid case) the equation **(48)** reduces to **(27)**.

Moreover, if we neglect the terms: $u_a^{**}(\theta_i)$ and $u_t^{**}(\theta_i)$ we obtain the Betz condition that coincides with Lerbs optimum.

In general, for a non-uniform flow, equation **(50)** can be rewritten:

$$\begin{aligned}
&\sum_{j=1}^{N} G_j M_a(\theta_i,\theta_j)\left(1 + \frac{\varepsilon}{tg\beta_i}\right) + \frac{1}{x_i \sin\theta_i}\sum_{j=1}^{N} G_j M_a(\theta_j,\theta_i) x_j \sin\theta_j \left(1 + \frac{\varepsilon}{tg\beta_j}\right) \\
&- \ell tg\beta\left(1 - tg\beta_i\right)\sum_{j=1}^{N} G_j M_t(\theta_i,\theta_j) - \frac{\ell tg\beta}{\sin\theta_i}\sum_{j=1}^{N} G_j M_t(\theta_j,\theta_i)\sin\theta_j\left(1 - tg\beta_j\right) \\
&= -\frac{1 - w(x)}{1 - w_T} - \ell - \frac{1 - w(x)}{1 - w_T}\frac{\varepsilon}{tg\beta_i} + \ell\varepsilon tg\beta_i
\end{aligned} \tag{53}$$

The equation **(53)** can be transformed in a matrix equation:

$$\sum_{j=1}^{N+1} A_{ij}^{(n)} G(\theta_j) = B_i \tag{54}$$

where for each iteration:

⇨ $$A_{ij}^{(n)} = M_a^{(n)}(\theta_i,\theta_j)\left(1 + \frac{\varepsilon}{tg\beta_i}\right) + \frac{1}{x_i \sin\theta_i} M_a^{(n)}(\theta_j,\theta_i) x_j \sin\theta_j \left(1 + \frac{\varepsilon}{tg\beta_j}\right)$$

$$- \ell tg\beta\left(1 - \varepsilon tg\beta_i\right) M_t^{(n)}(\theta_i,\theta_j) - \ell \frac{tg\beta}{\sin\theta_i} M_t^{(n)}(\theta_j,\theta_i)\sin\theta_j\left(1 - \varepsilon tg\beta_j\right),$$

when i = 1, 2, …, N and j = 1, 2, …, N

⇨ $$B_i = -\frac{1 - w(x)}{1 - w_T} - \ell - \varepsilon\left(\frac{1 - w(x)}{1 - w_T}\frac{1}{tg\beta_i} - \ell tg\beta_i\right),$$ when i = 1, 2, …, N

6.2 Algorithm

The algorithm for the problem including the viscous effects is equal to the algorithm for the new approach to the problem (second nonlinear optimization) shown in fig. 18 on section 5.1.

6.3 Implementation

The formulation was implemented in a program named ***OptiPropVisc***. The program listing for the program is in **Appendix A.3**.

The input and output files are the ***OptiPropVisc.inp*** and ***OptiPropVisc.out***, respectively. The entry data are the same as the described for the input file of *OptiPropII* with one difference. The file has a new line at the end of the file with the value of the drag-to-lift coefficient.

6.4 Results and Discussion

Consider the cases tested in point 5.3.2.6, where a comparison between the results achieved by OptiPropII and the results with the assumption of equation (31) as discussed. In this point we will see what happens to those results when we introduce the viscous drag. For simplicity we consider a constant drag-to-lift coefficient of 0.1.

Fig. 82 and 83 show the distribution of circulation and pitch, respectively for the cases where C_T=1.0, C_T=2.0 and C_T=3.0.

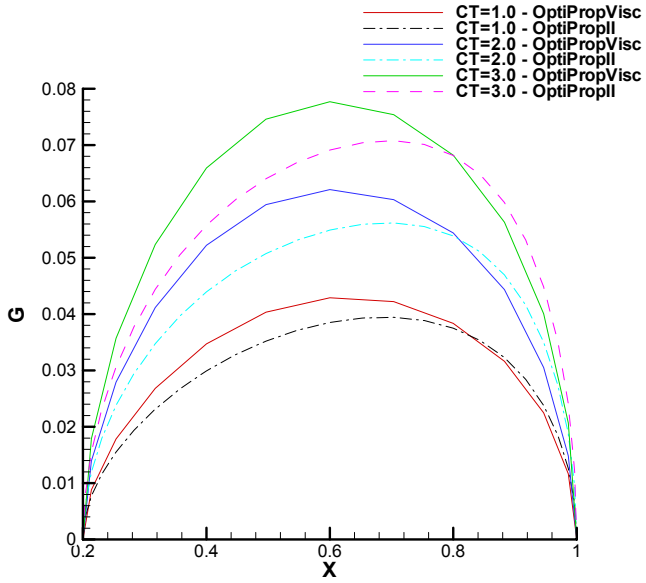

Fig. 82 - Optimum distribution of circulation along radius for N=11, C_T=1.0, C_T=2.0, C_T=3.0, moderate loading and uniform flow. Results from OptiPropVisc and OptiPropII.

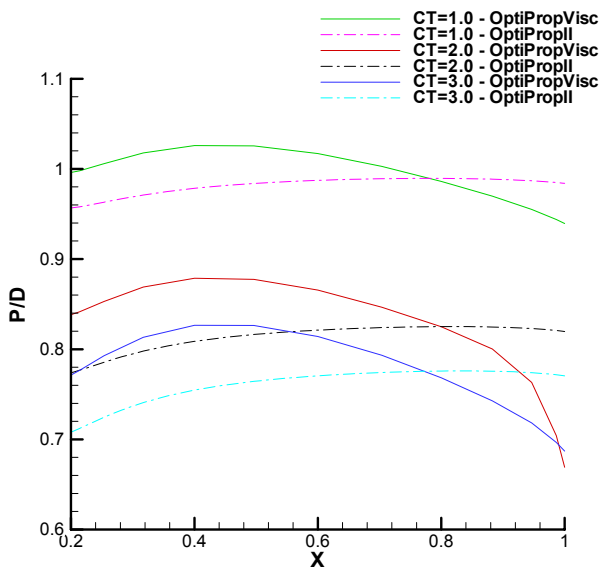

Fig. 83 - Optimum pitch distribution along radius for N=11, C_T=1.0, C_T=2.0, C_T=3.0, moderate loading and uniform flow. Results from OptiPropVisc and OptiPropII.

The introduction of viscous drag modifies the behavior of the curves. With viscous drag the maximum of the optimum circulation distribution is shifted to smaller radii.

Figs. 84 to 86 illustrate the induced velocities obtained from *OptiPropVisc* and *OptiPropII*. The discrepancies between the two results are clearly visible.

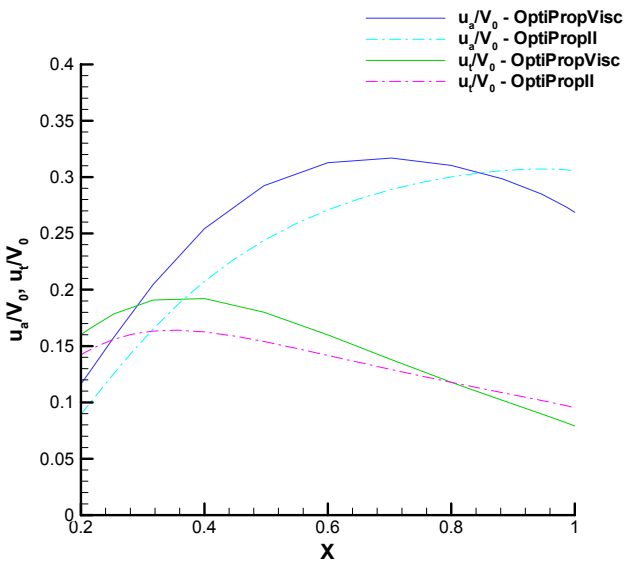

Fig. 84 - Optimum distribution of tangential and axial induced velocities along radius for N=11, C_T=1.0, J_0=0.727 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropVisc and OptiPropII.

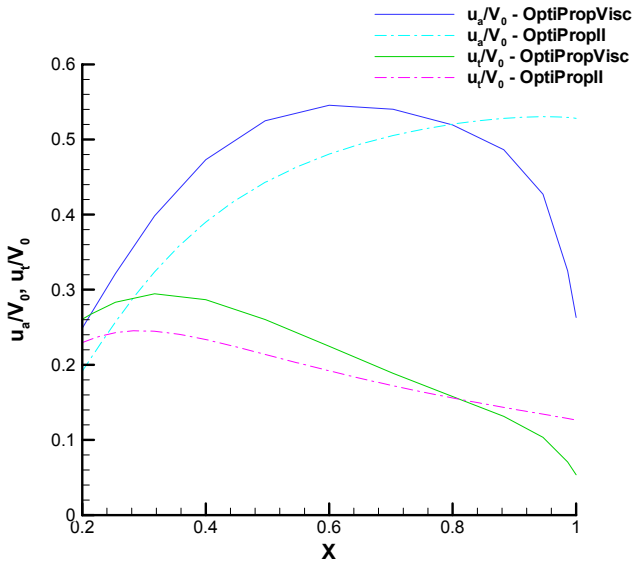

Fig. 85 - Optimum distribution of tangential and axial induced velocities along radius for N=11, C_T=2.0, J_0=0.525 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropVisc and OptiPropII.

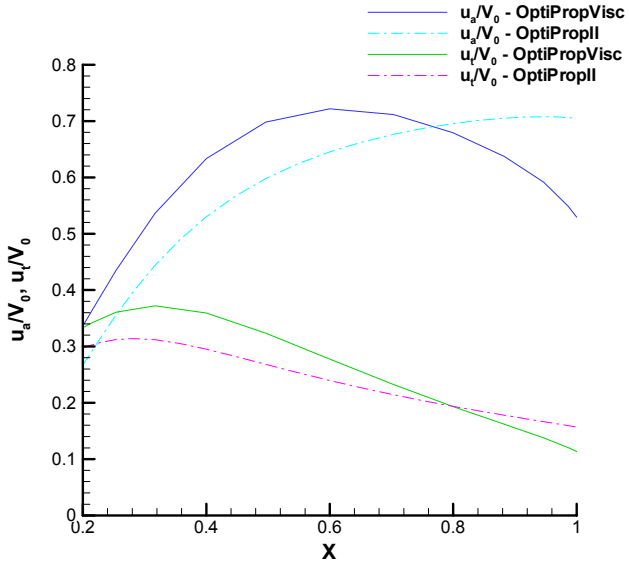

Fig. 86 - Optimum distribution of tangential and axial induced velocities along radius for N=11, C_T=3.0, J_0=0.442 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropVisc and OptiPropII.

Table 8 gives the results of propeller efficiency, power coefficient and the number of iterations obtained from *OptiPropVisc* and *OptiPropII*.

		OptiPropVisc			***OptiPropII***		
C_T	J_0	C_P	η [%]	Nº Iterations	C_P	η [%]	Nº Iterations
1.0	0.727	1.4479	69.1	50	1.3566	73.7	27
2.0	0.525	3.3429	59.8	79	3.1248	64.0	23
3.0	0.442	5.5625	53.9	53	5.2239	57.4	27

Table 8 – Values of efficiency, power coefficient and number of iterations for a moderately loaded propeller with N=11, Z=4, C_T=1.0, C_T=2.0 and C_T=3.0 obtained from OptiPropVisc and OptiPropII.

The propeller efficiency decreases significantly, when the viscous drag effects are introduced in the formulation, as expected. The efficiency decreases in the order of 4% with ε=0.1.

Considering now the formulation presented by Yim[6] in which he applied the Munk's theory and introduced the viscous drag. **Figs. 87 to 91** show the distribution of circulation, induced velocities and pitch distribution obtained from *OptiPropVisc* and the formulation of Yim for the same three cases. **Fig. 88** include also the results obtained from the standard theory presented by Yim for the hydrodynamic pitch:

$$\frac{x_i}{\lambda}\tan\beta_i = \left(\frac{c-\varepsilon\frac{x_i}{\lambda}}{1+c\frac{\varepsilon\lambda}{x_i}}\right), \text{ where c is the lagrange multiplier.}$$

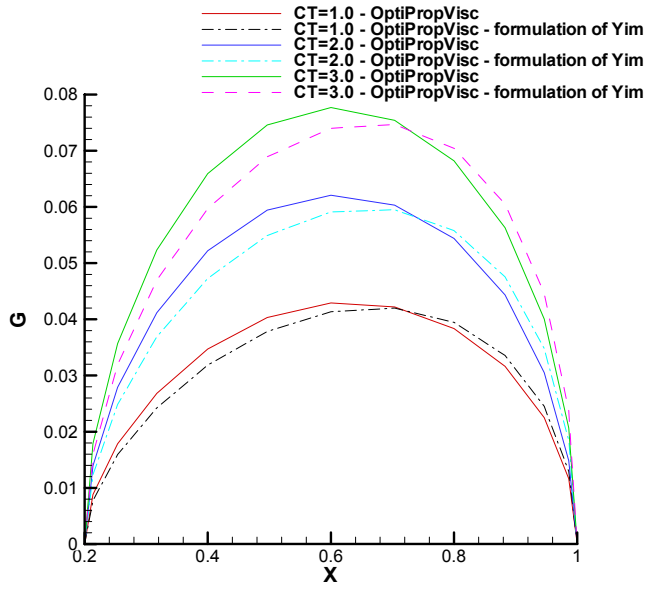

Fig. 87 - Optimum distribution of circulation along radius for N=11, C_T=1.0, C_T=2.0, C_T=3.0, moderate loading and uniform flow. Results from OptiPropVisc and from the formulation of Yim.

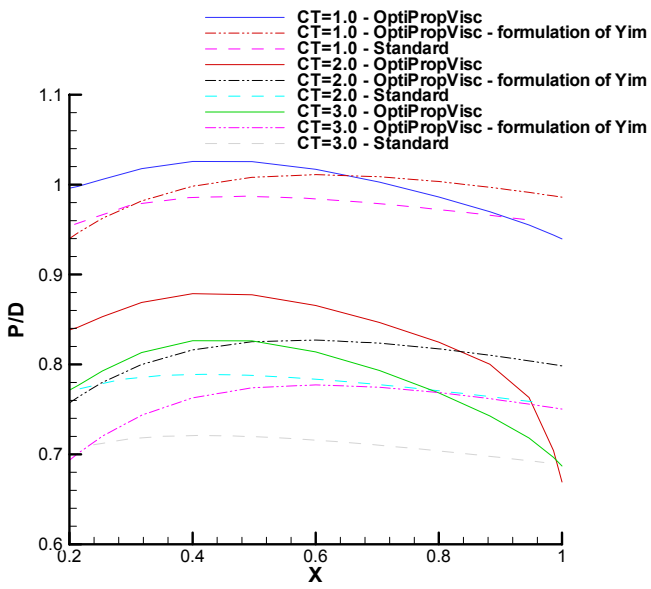

Fig. 88 - Optimum pitch distribution along radius for N=23, C_T=1.0, C_T=2.0, C_T=3.0, moderate loading and uniform flow Results from OptiPropVisc and from the formulation of Yim.

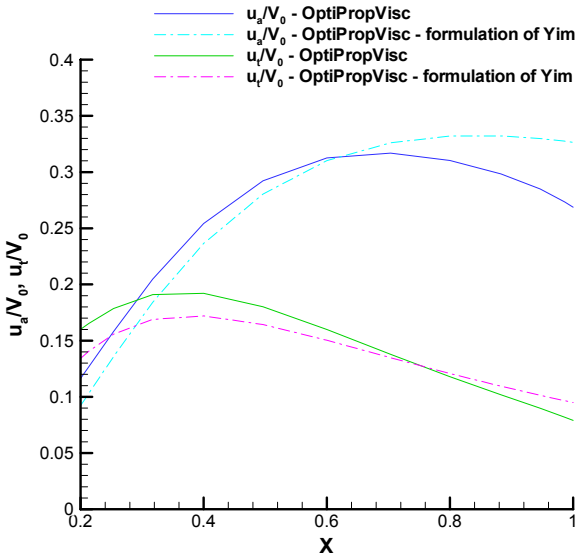

Fig. 89 - Optimum distribution of tangential and axial induced velocities along radius for N=11, C_T=1.0, J_0=0.727 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropVisc and from the formulation of Yim.

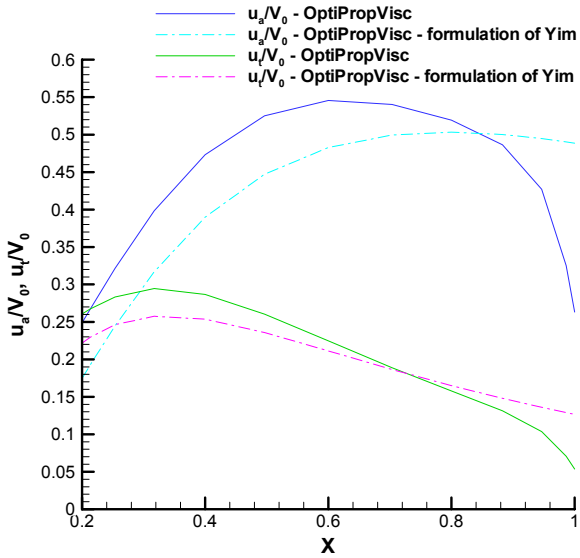

Fig. 90 - Optimum distribution of tangential and axial induced velocities along radius for N=11, C_T=2.0, J_0=0.525 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropVisc and from the formulation of Yim.

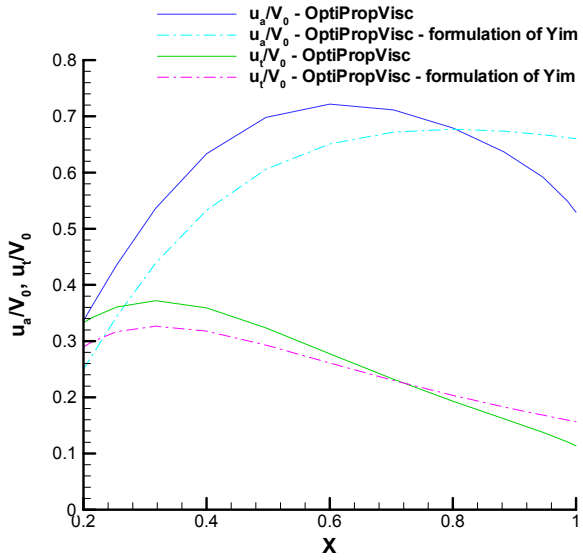

Fig. 91 - Optimum distribution of tangential and axial induced velocities along radius for N=11, C_T=3.0, J_0=0.442 and tolerance=0.001%, moderate loading and uniform flow. Results from OptiPropVisc and from the formulation of Yim.

In this case the differences between the present method *OptiPropII* and the application of Munk's displacement theorem are larger than in the inviscid case (fig. 75).

The values of the efficiency, power coefficient and number of iterations obtained from *OptiPropVisc* and from the formulation of Yim, are shown in **table 8**.

		OptiPropVisc			***OptiPropVisc*** formulation of Yim		
C_T	J_0	C_P	η [%]	Nº Iterations	C_P	η [%]	Nº Iterations
1.0	0.727	1.4479	69.1	50	1.4522	68.9	17
2.0	0.525	3.3429	59.8	79	3.2487	61.6	37
3.0	0.442	5.5625	53.9	53	5.4191	55.4	80

Table 9 – Values of efficiency, power coefficient and number of iterations for a moderately loaded propeller with N=11, Z=4, C_T=1.0, C_T=2.0 and C_T=3.0 obtained from OptiPropVisc and from the formulation of Yim.

Differences on the propeller efficiency were found, as expected. We perceive that these differences increase with loading.

7. CONCLUSIONS AND FINAL COMMENTS

In this work we investigate a method to solve the problem of the optimization of the propeller blade that consists in finding a radial distribution of circulation for minimum power at a given thrust, including viscous drag effects.

The inviscid case is studied first. The inviscid formulation used in *OptiProp* was the first attempt to solve the problem, but poor convergence properties were obtained for the algorithm implemented. Therefore, *OptiProp* become a first step to a better solution to the problem. An improved formulation was developed in *OptiPropII*. Convergence is improved but it still has some problems, especially for larger propeller loadings.

The results obtained by *OptiPropII* with the lightly and moderately loading assumptions reveal a variable pitch distribution that does not affect the propeller efficiency when compared with the standard method, which is the optimum constant hydrodynamic pitch of *Lerbs*. In moderately loaded assumptions, we found an insignificant difference of 0.1% on the efficiency.

We conclude that in the inviscid case the method developed in this work and the method with application of Munk's theorem give similar results. In comparison with the standard method (classical optimum of Lerbs), the differences increase with loading. For higher loadings, the differences became significant.

Including the viscous drag in the formulation, significant differences were found in the results. In particular, the optimum distribution of circulation changes, having its maximum shifted to the smaller radii. We found also a significant change in the pitch distribution.

In conclusion, the study performed in present work shows promising results for theoretical investigation. This work was a first step for the method of optimization of propeller blade with the lifting line theory including the viscous effects. As suggestions for future work we mention:

- application of the method to wake-adapted propellers;
- use of different drag-to-lift coefficient distributions instead of a constant distribution. This will allow more realistic solutions to studied;
- couple the present method with the actual design of propeller blade sections.

8. REFERENCES

[1] Pina, Heitor: "Métodos Numéricos", McGraw-Hill, 1995.

[2] Roma, Ricardo J. S.: "Projecto de Hélices Propulsores Marítimos com o Método dos Factores de Indução", Projecto de Termodinâmica Aplicada, January 1997.

[3] Breslin and Andersen : "Hydrodynamics of Ship Propellers", Cambridge University Press, 1994.

[4] Horta, Pedro A. S. R.: "Optimização de 'Hydrofoils' incluindo Efeitos de Viscosidade", Projecto de Termodinâmica Aplicada, September 1999.

[5] Lerbs, H. W.: "Moderately Loaded Propellers with a Finite Number of Blades and a Arbitrary Distribution of Circulation", Trans. SNAME, vol. 60, 1952.

[6] Yim, B.: "Optimum Propellers with Cavity-Drag and Frictional-Drag Effects", Journal of Ship Research, Vol. 20, Nº 2, June 1976, pp.118-123.

[7] Dyne, Gilbert: "On the Efficiency of a Propeller in Uniform Flow", The Royal Institution of Naval Architects, 1993.

[8] Andersen, Poul: "On Optimum Lifting-Line Propeller Calculation", The Danish Center for Applied Mathematics and Mechanics, Report Nº. 325, April 1986.

ADDITIONAL BIBLIOGRAPHY:

- Achkinadze, Alexander S. and Krasilnikov; Vladimir I.: "A Generalized Optimum Condition for Wake Adapted Screw Propellers", trans. SNAME, September 23-24, 1997.
- Brocket, T.E.; Korpus, R.A.: "Marine Propulsors for Minimum Shaft-Horsepower", Department of Naval Architecture and Marine Engineering, The University of Michigan, Ann Arbor, Michigan 48109.
- Brocket, T.E.; Korpus, R.A.: "Parametric Evaluation of Lifting-Line Model for Conventional and Preswirl Propulsors", Department of Naval Architecture and Marine Engineering, University of Michigan, Ann Arbor, U.S.A.
- Kerwin, Justin E.; Coney, William B.; Hsin, Ching-Yeh: "Optimum Circulation Distributions for Single and Multi-Component Propulsors", Massachusetts Institute of Technology, Cambridge, Massachusetts.

URL:

- http://www.nap.edu/
- http://www.dt.navy.mil/mz/oct21.1998/morganjl.html

APPENDIX

A. Program List

A.1 Program List of *OptiPro*

```
================================================
|  Optimização de Hélices Propulsores Marítimos
|           em escoamento invíscido
================================================

 PROGRAM OPTIPROP

DEFINICAO DAS VARIAVEIS DE ENTRADA
++++++++++++++++++++++++++++++++++
IPV: PARAMETRO QUE PERMITE AO UTILIZADOR A OPCAO ENTRE
  CARREGAMENTO LIGEIRO (1) OU CARREGAMENTO MODERADO (2)
N: NUMERO DE PONTOS INTERIORES SOBRE A PA
NB: NUMERO DE PAS
CT OU CP: COEFICIENTE ADIMENSIONAL DE IMPULSO OU POTENCIA
XH: RAIO DO CUBO ADIMENSIONAL
J0: COEFICIENTE DE AVANCO
NX: NUMERO DE PONTOS ONDE E DADA A FRACCAO DE ESTEIRA EFECTIVA
XW: RAIOS ADIMENSIONAIS PARA CADA UM DESSES PONTOS
WT1: FRACCAO DE ESTEIRA EFECTIVA NOS PONTOS XW
WM: FRACCAO DE ESTEIRA MEDIA EFECTIVA


DEFINICAO DAS VARIAVEIS DE SAIDA
++++++++++++++++++++++++++++++++

X: COTA DOS PONTOS CONSIDERADOS SOBRE A PA
G(X): DISTRIBUICAO DE CIRCULACAO ADIMENSIONAL
UA(X): VELOCIDADE AXIAL INDUZIDA ADIMENSIONAL
UT(X): VELOCIDADE TANGENCIAL INDUZIDA ADIMENSIONAL
BETA(X): ANGULO DE PASSO HIDRODINAMICO
BETAIN(X): ANGULO DE PASSO HIDRODINAMICO INDUZIDO
BETAV(X): ÂNGULO DE SAÍDA DOS VÓRTICES
VEL(X): VELOCIDADE RESULTANTE
PASSO(X): PASSO ADIMENSIONALIZADO PELO DIAMETRO
CP: COEFICIENTE ADIMENSIONAL DE POTENCIA
CT: COEFICIENTE ADIMENSIONAL DE IMPULSO
REND: RENDIMENTO (CT/CP)


DEFINICAO DAS VARIAVEIS DO PROGRAMA
+++++++++++++++++++++++++++++++++++

N0: NUMERO DE PONTOS INTERIORES MAXIMO
NP0: NUMERO PONTOS MAXIMO
N: NUMERO DE PONTOS INTERIORES
NP:NUMERO DE PONTOS
TETA: TABELA DE EQUIPARACAO ANGULAR PARA CADA PONTO
A: MATRIZ DOS FACTORES TRIGONOMETRICOS
X: TABELA DOS RAIOS ADIMENSIONAIS PARA UM N DADO
```

```
  BETA: TABELA DOS ANGULOS DE PASSO HIDRODINAMICOS
  BETAIN: TABELA DOS ANGULOS DE PASSO HIDRODINAMICOS INDUZIDOS
       ANTIGOS
  BETAV: TABELA DOS ANGULOS DE SAIDA DOS VORTICES
  LANDAIN: COEFICIENTE DE VELOCIDADE DE APROXIMACAO INDUZIDO
  G: DISTRIBUICAO DE CIRCULACAO
  UAMAT: MATRIZ DE INDUCAO AXIAL
  UTMAT: MATRIZ DE INDUCAO TANGENCIAL
  UA: TABELA DAS VELOCIDADES INDUZIDAS AXIAIS
  UT: TABELA DAS VELOCIDADES INDUZIDAS TANGENCIAIS
  B: VECTOR DE REAIS PARA RESOLUCAO DO SISTEMA
  AM: MATRIZ DO SISTEMA
  VEL: VELOCIDADE RESULTANTE
  PASSO: PASSO ADIMENSIONALIZADO PELO DIAMETRO
  XH: RAIO ADIMENSIONAL INTERIOR
  EPSLANDA: PARAMETRO DE CONVERGENCIA PARA O LANDAIN
  PI: CONSTANTE
  ERRO: VARIAVEL DE ERRO
  INFO: VARIAVEL DE INFORMACAO PARA RESOLUCAO DO SISTEMA
  REND: RENDIMENTO
  W: FACTOR DE SUB-RELAXACAO
  WT: FRACCAO DE ESTEIRA EFECTIVA NOS RAIOS DE GLAUERT
  GN_1(X): DISTRIBUICAO DE CIRCULACAO ADIMENSIONAL PARA A N-1
       ITERACÇÃO
  BETAVN_1: BETAV PARA A N-1 ITERACÇÃO
  BETAVCONV: (1) SE CONVERGE OU (0) CASO CONTRÁRIO
  GCONV: (1) SE CONVERGE OU (0) CASO CONTRÁRIO
  TOLERANCE: TOLERÂNCIA PARA A CONVERGÊNCIA DE G E BETAV


   INTEGER N0,NP0,N,NP,I,J,K,NB,IPV,INFO,NX
INTEGER GHANGE,GCONV,BETAVCONV,ASK,ITNUM
   PARAMETER(N0=29,NP0=N0+2)
   INTEGER IPVT(N0+1)
   REAL A(NP0,N0,NP0),TETA(NP0),X(NP0),BETAV(NP0),BETAVN_1(NP0)
   REAL BETAIN(NP0),LANDAIN,G(NP0+1),GN_1(NP0+1)
   REAL BETA(NP0),UAMAT(NP0,N0),UTMAT(NP0,N0),UA(NP0),UT(NP0)
   REAL B(N0+1),AM(N0+1,N0+1),VEL(NP0),PASSO(NP0),SOMA(N0)
   REAL XH,EPSLANDA,CT,CP,J0,PI,ERRO,TOLERANCE,TOLBETA
   REAL CP1,REND,W,XX(17),G1(17),BETA1(17),UTN_1(NP0)
   REAL BETAIN1(17),UT1(17),WT(NP0),XW(NP0),WM,WT1(NP0),XP(NP0)
REAL RELAX
PARAMETER(RELAX=0.5)

DATA EPSLANDA /0.01/
DATA TOLERANCE /0.0005/

   OPEN(UNIT=5,FILE='OptiProp.INP',STATUS='UNKNOWN')
   OPEN(UNIT=6,FILE='OptiProp.OUT',STATUS='UNKNOWN')


   READ(5,*) IPV
   PRINT*,IPV
   READ(5,*) N
   NP=N+2
   PRINT*,N
   PRINT*,NP
   READ(5,*) NB
   PRINT*,NB
   READ(5,*) CT
```

```
      PRINT*,CT
      READ(5,*) XH
      PRINT*,XH
      READ(5,*) J0
      PRINT*,J0
      READ(5,*) NX
      PRINT*,NX
      READ(5,*) (XW(I),I=1,NX)
      PRINT*,(XW(I),I=1,NX)
      READ(5,*) (WT1(I),I=1,NX)
      PRINT*,(WT1(I),I=1,NX)
      READ(5,*) WM
      PRINT*,WM


-----ECO DOS DADOS DE ENTRADA

      WRITE(6,1)
 1    FORMAT(/,10X,'DADOS DE ENTRADA',/,10X,16(1H=))
      WRITE(6,2) IPV
 2    FORMAT(/,4X,'IPV=',I2,5X)
      WRITE(6,3) N,NB,XH,J0
 3    FORMAT(/,4X,'N=',I2,5X,'NB=',I2,5X,'XH=',F6.3,5X,'J0=',F6.3)
      WRITE(6,5) CT
 5    FORMAT(/,4X,'CT=',F9.5)
      WRITE(6,10) (WT1(I),I=1,NX)
 10   FORMAT(/,4X,'WT=',F5.3)
      WRITE(6,11) WM
 11   FORMAT(/,4X,'WM=',F5.3)
      WRITE(6,12) TOLERANCE*100
 12   FORMAT(/,4X,'Tolerância = ',F7.5,' %')



-----TESTE DE VALIDADE DOS PARAMETROS ------

      ERRO=0
      IF ((IPV.NE.1).AND.(IPV.NE.2)) THEN
        ERRO=1
      ENDIF
      IF (NB.LE.0) THEN
        ERRO=1
    ELSE
        IF (CT.LE.0) THEN
          ERRO=1
      ELSE
          IF (XH.LT.0) THEN
            ERRO=1
        ELSE
              IF (XH.GE.1) THEN
            ERRO=1
              ELSE
            IF (J0.LE.0) THEN
              ERRO=1
                ELSE
                  IF (N.LE.0) THEN
                ERRO=1
                  ENDIF
                ENDIF
              ENDIF
```

```
        ENDIF
      ENDIF
    ENDIF

      IF (ERRO.EQ.1) THEN
        PRINT *,'VALOR DOS PARAMETROS DE ENTRADA ERRADO'
      GOTO 999
    ENDIF


-----CALCULO DE PI

      PI=4.*ATAN(1.)


-----CALCULO DOS VALORES DE TETA E X PARA NP PONTOS

      DO 13 J=1,NP
      TETA(J)=FLOAT(J-1)*PI/FLOAT(N+1)
      X(J)=0.5*(1.+XH) - 0.5*(1.-XH)*COS(TETA(J))
 13   CONTINUE


-----INTERPOLACAO DA FRACCAO DE ESTEIRA EFECTIVA PARA  RAIOS DE GLAUERT


      DO 14 I=1,NP
        CALL INTK1(NX,XW,X(I),WT1,WT(I))
 14   CONTINUE


-----CALCULO DAS MATRIZES DOS FACTORES TRIGONOMETRICOS A

      CALL AMAT(N0,NP0,N,NP,TETA,A)


-----CORRECCAO DA MATRIZ A PARA O CASO DE XH=0

      IF (XH.EQ.0) THEN
        DO 15 I=1,N
          A(1,I,1)=0.
 15     CONTINUE
      ENDIF

-----CALCULO DE BETA E BETAIN
      IF (X(1).EQ.0.) THEN
        BETA(1)=PI/2
        DO 16 I=2,NP
          BETA(I)=ATAN(J0*(1.-WT(I))/(PI*X(I)*(1.-WM)))
 16     CONTINUE
      ELSE
        DO 17 I=1,NP
          BETA(I)=ATAN(J0*(1.-WT(I))/(PI*X(I)*(1.-WM)))
 17     CONTINUE
      ENDIF
      CALL BETA_INICT(NP0,N,NP,EPSLANDA,J0,CT,WT,WM,
     &                  TETA,X,BETAIN,LANDAIN)
      IF (X(1).EQ.0.) THEN
        BETAIN(1)=PI/2.
    ENDIF
```

```
-----CALCULO DE BETAV PASSO DOS VORTICES

      IF (IPV.EQ.1) THEN
      DO 18 I=1,NP
        BETAV(I)=BETA(I)
 18             CONTINUE
  ELSE
      DO 19 I=1,NP
        BETAV(I)=BETAIN(I)
 19             CONTINUE
      ENDIF


-----CALCULO DE UTN_1 E GN_1

   DO 20 I=1,NP+1
      G(I)=0.
      GN_1(I)=0.
 20           CONTINUE
   DO 30 I=1,NP
      UTN_1(I)=SIN(BETAIN(I))*COS(BETAIN(I))*
     &           (X(I)*PI/J0*TAN(BETAIN(I))-(1.-WT(I))/(1.-WM))
 30           CONTINUE
   ASK=1
   ITNUM=0


-----CALCULO DAS MATRIZES DE INDUCAO

 120  CALL MAT_IND(N0,NP0,N,NP,XH,A,BETAV,NB,X,
     &             UAMAT,UTMAT,ERRO)
      IF (ERRO.EQ.1) THEN
        PRINT*,'ERRO: EXCESSO DE CAPACIDADE'
        GOTO 999
      ENDIF


-----CALCULO DE G(X)

-------CONSTRUCAO DA MATRIZ DO SISTEMA LINEAR

 130    DO 150 I=1,N
        B(I)=(-1.)*(1.-WT(I+1))/(1.-WM)*X(I+1)*SIN(TETA(I+1))
        DO 140 J=1,N
          AM(I,J)=UAMAT(I+1,J)*X(I+1)*SIN(TETA(I+1))
     &           +UAMAT(J+1,I)*X(J+1)*SIN(TETA(J+1))
 140      CONTINUE
 150    CONTINUE
      DO 152 I=1,N
      SOMA(I)=0.
            DO 151 J=1,N
          SOMA(I)=SOMA(I)+GN_1(J+1)*(UTMAT(J+1,I)*SIN(TETA(J+1))
     &           +UTMAT(I+1,J)*SIN(TETA(I+1)))
 151      CONTINUE
 152    CONTINUE
      DO 154 J=1,N
      AM(N+1,J)=(2.*PI*NB*(1.-XH)/(N+1.))*SIN(TETA(J+1))*
     &          (X(J+1)*PI/J0-UTN_1(J+1))
```

```
      AM(J,N+1)=X(J+1)*SIN(TETA(J+1))-J0/PI*SOMA(J)
154           CONTINUE
      AM(N+1,N+1)=0.
    B(N+1)=CT


-------RESOLUCAO DO SISTEMA

      ITNUM=ITNUM+1
    CALL SGEFA(AM,N0+1,N+1,IPVT,INFO)
      IF (INFO.NE.0) THEN
        PRINT*,'SISTEMA SEM RESOLUCAO'
        GOTO 999
      ENDIF
      CALL SGESL(AM,N0+1,N+1,IPVT,B,0)


-------AFECTACAO DOS RESULTADOS

      DO 160 I=1,N
        G(I+1)=B(I)
160    CONTINUE
      G(1)=0.
      G(NP)=0.
    G(NP+1)=B(N+1)


-------RELAXACAO DE G

    DO 162 I=1,NP+1
      G(I)=GN_1(I)+RELAX*(G(I)-GN_1(I))
162           CONTINUE


-------CALCULO DAS VELOCIDADES INDUZIDAS

      DO 165 K=1,NP
        UA(K)=0.
        UT(K)=0.
        DO 164 I=1,N
          UA(K)=UA(K)+G(I+1)*UAMAT(K,I)
          UT(K)=UT(K)+G(I+1)*UTMAT(K,I)
164      CONTINUE
165    CONTINUE


-------CONVERGENCIA DE G

      GCONV=1
    DO 172 I=2,NP+1
      IF ((ABS(G(I)-GN_1(I))/G(I)).GT.TOLERANCE) THEN
        GCONV=0
        ENDIF
172           CONTINUE

   DO 190 I=1,NP
       GN_1(I)=G(I)
       BETAVN_1(I)=BETAV(I)
     UTN_1(I)=UT(I)
190         CONTINUE
```

```
      GN_1(NP+1)=G(NP+1)


-----CALCULO DE BETA

 210  IF (X(1).EQ.0.) THEN
        BETA(1)=PI/2
        DO 214 I=2,NP
          BETA(I)=ATAN(J0*(1.-WT(I))/(PI*X(I)*(1.-WM)))
 214    CONTINUE
      ELSE
        DO 216 I=1,NP
          BETA(I)=ATAN(J0*(1.-WT(I))/(PI*X(I)*(1.-WM)))
 216    CONTINUE
      ENDIF



-----CALCULO DE BETAIN

      IF (X(1).EQ.0.) THEN
        BETAIN(1)=PI/2
        DO 218 I=2,NP
          BETAIN(I)=ATAN(((1.-WT(I))/(1.-WM)+UA(I))/
     &                 (X(I)*PI/J0-UT(I)))
 218    CONTINUE
      ELSE
        DO 220 I=1,NP
          BETAIN(I)=ATAN(((1.-WT(I))/(1.-WM)+UA(I))/
     &                 (X(I)*PI/J0-UT(I)))
 220    CONTINUE
      ENDIF


-----CALCULO DE BETAV PASSO DOS VORTICES

      IF (IPV.EQ.1) THEN
      DO 221 I=1,NP
        BETAV(I)=BETA(I)
 221            CONTINUE
   ELSE
        IF (GCONV.EQ.0) THEN
        DO 222 I=1,NP
          BETAV(I)=BETAVN_1(I)
 222              CONTINUE
      ENDIF
      ENDIF


      IF (GCONV.EQ.1) THEN
      DO 223 I=1,NP
        BETAV(I)=BETAIN(I)
 223            CONTINUE
   ENDIF


-------CONVERGENCIA DE BETAV

        BETAVCONV=1
     DO 224 I=1,NP
```

```
      IF ((ABS(BETAV(I)-BETAVN_1(I))/BETAV(I)).GT.TOLERANCE)
     &        THEN
              BETAVCONV=0
         ENDIF
224           CONTINUE


-------ROTINA DE CONVERGENCIA EM G E BETA

  IF (GCONV.EQ.0) THEN
        GOTO 130
  ELSE
    IF (BETAVCONV.EQ.0) THEN
          GOTO 120
    ENDIF
  ENDIF


******************BLOCO DE OUTPUT**************************

-----CALCULO DO MODULO DA VELOCIDADE TOTAL

225    DO 230 I=1,NP
       VEL(I)=(UA(I)+(1.-WT(I))/(1-WM))/(SIN(BETAIN(I)))
230    CONTINUE

-----CALCULO DO COEFICIENTE DE POTENCIA

      CP=0.
      DO 240 I=1,NP
        CP=CP+G(I)*((1.-WT(I))/(1.-WM)+UA(I))*X(I)*SIN(TETA(I))
240    CONTINUE
      CP=CP*NB*2.*(PI**2)*(1.-XH)/(J0*(N+1.))


-----CALCULO DE RENDIMENTO

     REND=CT/CP

-----CALCULO DO PASSO

     IF (X(1).EQ.0.) THEN
       DO 260 I=2,NP
         PASSO(I)=PI*X(I)*TAN(BETAIN(I))
260    CONTINUE
     ELSE
       DO 261 I=1,NP
         PASSO(I)=PI*X(I)*TAN(BETAIN(I))
261    CONTINUE
     ENDIF

-----APRESENTACAO DOS ANGULOS EM GRAUS

     DO 262 I=1,NP
       BETA(I)=BETA(I)*180./PI
262  CONTINUE

     DO 264 I=1,NP
       BETAIN(I)=BETAIN(I)*180./PI
264  CONTINUE
```

```
**************FIM DO BLOCO DE OUTPUT****************


-----AFECTACAO DOS RESULTADOS

 265  FORMAT(/,F9.5)

      WRITE(6,275)
 275  FORMAT(///,10X,'RESULTADOS DO PROGRAMA OPTIPROP',/,10X,30(1H=))

      WRITE(6,265)

-----PRIMEIRA TABELA DE RESULTADOS

      WRITE(6,285)
 285  FORMAT(/,3X,'I',6X,'X',7X,'G',8X,'UA',7X,'UT',/)

      DO 290 I=1,NP
        WRITE(6,295) I,X(I),G(I),UA(I),UT(I)
 290  CONTINUE
 295  FORMAT(I4,4F9.5)


-----SEGUNDA TABELA DE RESULTADOS

      WRITE(6,298)
 298  FORMAT(//,3X,'I',6X,'B',7X,'Bi',8X,'V',7X,'P/D',/)
      DO 300 I=1,NP
        WRITE (6,302) I,BETA(I),BETAIN(I),VEL(I),PASSO(I)
 300  CONTINUE
 302  FORMAT(I4,2F9.2,2F9.5)

      WRITE(6,305) CP
 305  FORMAT(//,10X,'COEFICIENTE DE POTENCIA: CP=',F6.3)

      WRITE(6,315) CT
 315  FORMAT(//,10X,'COEFICIENTE DE IMPULSO: CT=',F6.3)

      WRITE(6,325) REND
 325  FORMAT(//,10X,'RENDIMENTO=',F6.3)
      WRITE(6,*)
    WRITE(6,330) ITNUM
 330  FORMAT(//,10X,'Numero de Iterações = ',I3,3X)


-----INTERPOLACAO PARA RAIOS CONVENCIONAIS

      XX(1)=XH
      DO 350 I=2,17
        XX(I)=XX(I-1)+0.05
 350  CONTINUE

      DO 360 I=1,17
        CALL INTK1(NP,X,XX(I),G,G1(I))
 360  CONTINUE
      DO 370 I=1,17
        CALL INTK1(NP,X,XX(I),BETA,BETA1(I))
 370  CONTINUE
```

```
      DO 380 I=1,17
        CALL INTK1(NP,X,XX(I),BETAIN,BETAIN1(I))
 380  CONTINUE
      DO 390 I=1,17
        CALL INTK1(NP,X,XX(I),UT,UT1(I))
 390  CONTINUE


-----FECHO DOS FICHEIROS DE INP,OUT,DAT

      CLOSE(UNIT=5)
      CLOSE(UNIT=6)

 999  STOP
      END


C=====================================================================

C          ESCRITA DAS DIFERENTES SUBROTINAS
C          ++++++++++++++++++++++++++++++++++


C------------------SUBROTINA AMAT

   SUBROUTINE AMAT(N0,NP0,N,NP,TETA,A)



C     ESTA SUBROTINA CALCULA A MATRIZ DOS FACTORES
C     TRIGONOMETRICOS PARA UM DETERMINADO NUMERO DE
C     PONTOS DADO


C  PARAMETROS DE ENTRADA

C        N0: NUMERO DE PONTOS INTERIORES MAXIMO
C        NP0: NUMERO DE PONTOS MAXIMO
C        N: NUMERO DE PONTOS INTERIORES
C        NP:NUMERO DE PONTOS
C        TETA: TABELA DE EQUIPARACAO ANGULAR PARA CADA PONTO


C  PARAMETROS DE SAIDA

C        A: MATRIZ DOS FACTORES TRIGONOMETRICOS



   INTEGER N0,NP0,N,NP,I,J,K,L
   REAL A(NP0,N0,NP0),TETA(NP0)
   REAL B1,B2,PI,H1,H2

   PI=4.*ATAN(1.)

C       MATRIZ A

      DO 50 I=1,N
       DO 40 J=1,NP
         DO 30 K=1,NP
```

```
      B1=0.5
      IF (((K.EQ.1).AND.(J.EQ.1)).OR.((K.EQ.NP).AND.(J.EQ.NP))) THEN
            B2=0
          DO 10 M=1,N+1
           IF (M.EQ.(N+1)) THEN
            H1=0.5
           ELSE
            H1=1
           ENDIF
           B2=B2+FLOAT(M)*H1*COS(FLOAT(M)*TETA(J))*
     &          COS(FLOAT(M)*TETA(K))/COS(TETA(K))
10         CONTINUE
      ELSE
            H1 =FLOAT((J+K-2)/2)
            H2 = FLOAT(J+K-2)/2.
            IF (ABS(H1-H2).LT.0.01) THEN
          B2 = 0.
            ELSE
          B2 = -1. / (COS(TETA(K)) - COS(TETA(J)))
            ENDIF
      ENDIF


      A(K,I,J) = 0.

        DO 20 L=1,N

          IF (K.EQ.1.OR.K.EQ.NP) THEN
               H1 = FLOAT(L) * COS(FLOAT(L)*TETA(K)) / COS(TETA(K))
          ELSE
               H1 = SIN(FLOAT(L)*TETA(K)) / SIN(TETA(K))
          ENDIF

          B1 = B1 + COS(FLOAT(L)*TETA(J)) * COS(FLOAT(L)*TETA(K))
          B2 = B2 - COS(FLOAT(L)*TETA(J)) * H1

          A(K,I,J) = A(K,I,J) + FLOAT(L)*SIN(FLOAT(L)*TETA(I+1))
     &                    * (H1*B1 + COS(FLOAT(L)*TETA(K))*B2)
20        CONTINUE

      A(K,I,J) = A(K,I,J) * 4.*PI/(FLOAT(N+1))**2

30     CONTINUE
40    CONTINUE
50    CONTINUE


    RETURN
    END



C------------------SUBROTINA INFAC

      SUBROUTINE INFAC(X0,X1,BETAV,NB,IA,IT,ERRO)


C     ESTA SUBROTINA CALCULA OS FACTORES DE INDUCAO
C     PARA O QUOCIENTE X0/X1
```

```
C  PARAMETROS DE ENTRADA

C     X0: RAIO ADIMENSIONAL PARA ORIGEM DO VORTICE
C     X1: RAIO ADIMENSIONAL PARA O PONTO DE CALCULO
C     BETAV: ANGULO DE PASSO HIDRODINAMICO
C     NB: NUMERO DE PAS
C     ERRO: VARIAVEL DE ERRO


C  PARAMETROS DE SAIDA
C      IA: FACTOR DE INDUCAO AXIAL PARA K,L
C      IT: FACTOR DE INDUCAO TANGENCIAL PARA K,L



     REAL X0X,A,B,F,G,U,P,IA,IT,ERRO
     INTEGER NB
     REAL BETAV,X0,X1
     DATA TOL /1E-12/



        IF (BETAV.LE.0) THEN
          ERRO=1
        ENDIF


     IF (ERRO.EQ.1) THEN
       GOTO 40
   ENDIF


C---CALCULO DOS FACTORES DE INDUCAO

     IF ((X0.LT.TOL).AND.(X1.LT.TOL)) THEN
       IA=0
       IT=1
       GOTO 40
     ENDIF
     IF (X1.LT.TOL) THEN
       IA=NB/TAN(BETAV)
       IT=0
       GOTO 40
     ENDIF
     X0X=X0/X1
     IF (X0X.LT.TOL) THEN
       IA=0
       IT=NB
       GOTO 40
     ENDIF
   IF ((1/(X0X)).EQ.1) THEN
       IA=COS(BETAV)
       IT=SIN(BETAV)
       GOTO 40
   ELSE
     IF (X0X.GT.1) THEN
         GOTO 20
     ELSE
       IF (X0X.LT.0.2) THEN
```

```
          GOTO 30
        ELSE
          IF(BETAV.GE.(0.179))THEN
            GOTO 20
              ELSE
                IF (X0X.LT.0.8) THEN
              GOTO 30
                ELSE
              IF (BETAV.GE.(0.012)) THEN
                GOTO 20
                  ELSE
                GOTO 30
                  ENDIF
                ENDIF
              ENDIF
        ENDIF
      ENDIF
    ENDIF



20   P=(1+((1/X0X)**2)/((TAN(BETAV))**2))**0.5
     U=(((P-1)*X0X/(1/SIN(BETAV)-1))**NB)*
    &  EXP(NB*(P-1/SIN(BETAV)))

     IF (ABS(1-U).LE.1E-38) THEN
       ERRO=1
       PRINT *,'ERRO: EXCESSO DE CAPACIDADE'
       GOTO 40
     ENDIF

     G=((SIN(BETAV))**3)*(2+9/((TAN(BETAV))**2))+
    &  (3*P-5)/(P**3)
     F=1/(SQRT(P*SIN(BETAV)))
     IF (X0X.GT.1) THEN
       B=F*(U/(1-U)+(1/(24*NB))*G*LOG(ABS(1/(1-U))))
       IA=(1-(1/(X0X)))*(NB*(1+B))/TAN(BETAV)
       IT=(X0X-1)*NB*B
     ELSE
       IF (X0X.LT.1) THEN
         A=F*(1/(U-1)-(1/(24*NB))*G*LOG(ABS(U/(U-1))))
         IA=(1/X0X-1)*(NB*A)/TAN(BETAV)
         IT=(1-X0X)*NB*(1+A)
       ENDIF
     ENDIF
     GOTO 40


30   IA=0
     IT=NB-NB*X0X



40   RETURN
     END
```

```
C-------------------------SUBROTINA MAT_IND

      SUBROUTINE MAT_IND(N0,NP0,N,NP,XH,A,BETAV,NB,X,
     &                   UAMAT,UTMAT,ERRO)



C     ESTA SUBROUTINA CALCULA AS MATRIZES DE INDUCAO
C     UAMAT E UTMAT


C  PARAMETROS DE ENTRADA

C       N0: NUMERO DE PONTOS INTERIORES MAXIMO
C       NP0: NUMERO DE PONTOS MAXIMO
C       N: NUMERO DE PONTOS INTERIORES
C       NP: NUMERO DE PONTOS
C       X: TABELA DOS RAIOS ADIMENSIONAIS
C       XH: RAIO INTERIOR ADIMENSIONAL
C       BETAV: ANGULO DE PASSO HIDRODINAMICO
C       NB: NUMERO DE PAS
C       A: MATRIZ DOS FACTORES TRIGONOMETRICOS


C  PARAMETROS DE SAIDA
C
C       UAMAT: MATRIZ AXIAL DE INDUCAO
C       UTMAT: MATRIZ TANGENCIAL DE INDUCAO
C       ERRO: VARIAVEL DE ERRO


    INTEGER N0,NP0,N,NP,I,J,K,NB
    REAL A(NP0,N0,NP0),BETAV(NP0),X(NP0)
    REAL UAMAT(NP0,N0),UTMAT(NP0,N0)
    REAL XH,LAMBJ,IA,IT,ERRO

C---MATRIZ DOS FACTORES DE INDUCAO

     ERRO=0

     DO 30 I=1,N
      DO 20 K=1,NP
        UAMAT(K,I)=0.
        UTMAT(K,I)=0.
          DO 10 J=1,NP
            IF(J.EQ.1.OR.J.EQ.NP) THEN
              LAMBJ=0.5
            ELSE
              LAMBJ=1.0
            ENDIF

C---CALCULO DOS FACTORES DE INDUCAO

            CALL INFAC(X(J),X(K),BETAV(J),NB,IA,IT,ERRO)

            IF (ERRO.EQ.1) THEN
              PRINT *,'ERRO: MUDAR OS PARAMETROS DE ENTRADA'
              GOTO 40
            ENDIF
```

```
          UAMAT(K,I) = UAMAT(K,I) + IA*LAMBJ*A(K,I,J)
          UTMAT(K,I) = UTMAT(K,I) + IT*LAMBJ*A(K,I,J)
10        CONTINUE
        UAMAT(K,I) = UAMAT(K,I) / (1.-XH)
        UTMAT(K,I) = UTMAT(K,I) / (1.-XH)
20     CONTINUE
30    CONTINUE


40    RETURN
      END



C----------------------SUBROTINA BETA_INICT

      SUBROUTINE BETA_INICT(NP0,N,NP,EPSLANDA,J0,CT,WT,WM,
     &                  TETA,X,BETAIN,LANDAIN)



C       ESTA SUBROTINA CALCULA OS VALORES DOS ANGULOS
C       DE PASSO HIDRODINAMICOS INICIAIS SENDO DADO O
C       COEFICIENTE DE IMPULSO


C  PARAMETROS DE ENTRADA

C       NP0:NUMERO DE PONTOS MAXIMO
C       N: NUMERO DE PONTOS INTERIORES
C       NP: NUMERO DE PONTOS
C       EPSLANDA: PARAMETRO DE CONVERGENCIA DE LANDA
C       J0: COEFICIENTE DE AVANCO
C       CT: COEFICIENTE DE IMPULSO
C       TETA: TABELA DE EQUIPARACAO ANGULAR
C       X: TABELA DOS RAIOS ADIMENSIONAIS
C       WT: FRACCAO DE ESTEIRA EFECTIVA PARA CADA PONTO
C       WM: FRACCAO DE ESTEIRA MEDIA EFECTIVA


C  PARAMETROS DE SAIDA

C       BETAIN: TABELA DOS ANGULOS DE PASSO HIDRODINAMICOS
C       LANDAIN: COEFICIENTE DE VELOCIDADE DE APROXIMACAO


  INTEGER NP0,N,NP,I
    REAL EPSLANDA,LANDA,CT,CTA,H
    REAL LANDAIN,LANDAIV,J0,WT(NP0),WM
  REAL BETAIN(NP0),X(NP0),TETA(NP0)


  PI= 4.*ATAN(1.)
  LANDA=0

    DO 10 I=1,NP
     LANDA= LANDA +((1-WT(I))*J0/(PI*(1-WM)))*SIN(TETA(I))
10   CONTINUE

  LANDA= LANDA*PI/(2*(N+1))
```

```
      CTA= CT*((J0/(LANDA*PI))**2)
     LANDAIN=0.5*LANDA*(1+SQRT(1+CTA))

   20   LANDAIV=LANDAIN
     H=1+LANDAIV*(LANDAIV-LANDA)/(1+LANDAIV**2)
     H=H-LANDAIV*(2*LANDAIV-LANDA)*
      +ALOG((1+LANDAIV**2)/(LANDAIV**2))
     LANDAIN=0.5*LANDA*(1+SQRT(1+CTA/H))
      IF (ABS(LANDAIV-LANDAIN).GE.EPSLANDA)THEN
        GOTO 20
     ENDIF

   C-----CASO DE XH SER IGUAL A ZERO OU DIFERENTE DE ZERO

      IF (X(1).EQ.0) THEN
       BETAIN(1)=PI/2
       DO 30 I=2,NP
        BETAIN(I)=ATAN(LANDAIN/X(I))
   30   CONTINUE
      ELSE
       DO 40 I=1,NP
        BETAIN(I)=ATAN(LANDAIN/X(I))
   40   CONTINUE
      ENDIF


     RETURN
     END



      integer function isamax(n,sx,incx)

   c     finds the index of element having max. absolute value.
   c     jack dongarra, linpack, 3/11/78.
   c     modified 3/93 to return if incx .le. 0.
   c     modified 12/3/93, array(1) declarations changed to array(*)

   C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      real sx(*),smax
      integer i,incx,ix,n
   c
      isamax = 0
      if( n.lt.1 .or. incx.le.0 ) return
      isamax = 1
      if(n.eq.1)return
      if(incx.eq.1)go to 20

   c       code for increment not equal to 1

      ix = 1
      smax = abs(sx(1))
      ix = ix + incx
      do 10 i = 2,n
        if(abs(sx(ix)).le.smax) go to 5
        isamax = i
        smax = abs(sx(ix))
    5   ix = ix + incx
     10 continue
        return
```

```
c       code for increment equal to 1

  20 smax = abs(sx(1))
     do 30 i = 2,n
        if(abs(sx(i)).le.smax) go to 30
        isamax = i
        smax = abs(sx(i))
  30 continue
     return
     end


     subroutine saxpy(n,sa,sx,incx,sy,incy)

c     constant times a vector plus a vector.
c     uses unrolled loop for increments equal to one.
c     jack dongarra, linpack, 3/11/78.
c     modified 12/3/93, array(1) declarations changed to array(*)

C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      real sx(*),sy(*),sa
      integer i,incx,incy,ix,iy,m,mp1,n

      if(n.le.0)return
      if (sa .eq. 0.0) return
      if(incx.eq.1.and.incy.eq.1)go to 20

c       code for unequal increments or equal increments
c         not equal to 1

      ix = 1
      iy = 1
      if(incx.lt.0)ix = (-n+1)*incx + 1
      if(incy.lt.0)iy = (-n+1)*incy + 1
      do 10 i = 1,n
        sy(iy) = sy(iy) + sa*sx(ix)
        ix = ix + incx
        iy = iy + incy
  10 continue
     return

c       code for both increments equal to 1

c       clean-up loop

  20 m = mod(n,4)
     if( m .eq. 0 ) go to 40
     do 30 i = 1,m
       sy(i) = sy(i) + sa*sx(i)
  30 continue
     if( n .lt. 4 ) return
  40 mp1 = m + 1
     do 50 i = mp1,n,4
       sy(i) = sy(i) + sa*sx(i)
       sy(i + 1) = sy(i + 1) + sa*sx(i + 1)
       sy(i + 2) = sy(i + 2) + sa*sx(i + 2)
       sy(i + 3) = sy(i + 3) + sa*sx(i + 3)
  50 continue
     return
     end
```

```
      subroutine sscal(n,sa,sx,incx)

c     scales a vector by a constant.
c     uses unrolled loops for increment equal to 1.
c     jack dongarra, linpack, 3/11/78.
c     modified 3/93 to return if incx .le. 0.
c     modified 12/3/93, array(1) declarations changed to array(*)

C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      real sa,sx(*)
      integer i,incx,m,mp1,n,nincx

      if( n.le.0 .or. incx.le.0 )return
      if(incx.eq.1)go to 20

c        code for increment not equal to 1

      nincx = n*incx
      do 10 i = 1,nincx,incx
        sx(i) = sa*sx(i)
   10 continue
      return

c        code for increment equal to 1


c        clean-up loop

   20 m = mod(n,5)
      if( m .eq. 0 ) go to 40
      do 30 i = 1,m
        sx(i) = sa*sx(i)
   30 continue
      if( n .lt. 5 ) return
   40 mp1 = m + 1
      do 50 i = mp1,n,5
        sx(i) = sa*sx(i)
        sx(i + 1) = sa*sx(i + 1)
        sx(i + 2) = sa*sx(i + 2)
        sx(i + 3) = sa*sx(i + 3)
        sx(i + 4) = sa*sx(i + 4)
   50 continue
      return
      end




      subroutine sgefa(a,lda,n,ipvt,info)
C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      integer lda,n,ipvt(1),info
      real a(lda,1)

c     sgefa factors a real matrix by gaussian elimination.

c     sgefa is usually called by sgeco, but it can be called
c     directly with a saving in time if  rcond  is not needed.
c     (time for sgeco) = (1 + 9/n)*(time for sgefa) .

c     on entry
```

```
c        a       real(lda, n)
c                the matrix to be factored.

c        lda     integer
c                the leading dimension of the array  a .

c        n       integer
c                the order of the matrix  a .

c     on return

c        a       an upper triangular matrix and the multipliers
c                which were used to obtain it.
c                the factorization can be written  a = l*u  where
c                l  is a product of permutation and unit lower
c                triangular matrices and  u  is upper triangular.

c        ipvt    integer(n)
c                an integer vector of pivot indices.

c        info    integer
c                = 0  normal value.
c                = k  if  u(k,k) .eq. 0.0 .  this is not an error
c                     condition for this subroutine, but it does
c                     indicate that sgesl or sgedi will divide by zero
c                     if called.  use  rcond  in sgeco for a reliable
c                     indication of singularity.

c     linpack. this version dated 08/14/78 .
c     cleve moler, university of new mexico, argonne national lab.

c     subroutines and functions

c     blas saxpy,sscal,isamax

c     internal variables

      real t
      integer isamax,j,k,kp1,l,nm1


c     gaussian elimination with partial pivoting

      info = 0
      nm1 = n - 1
      if (nm1 .lt. 1) go to 70
      do 60 k = 1, nm1
        kp1 = k + 1

c        find l = pivot index

        l = isamax(n-k+1,a(k,k),1) + k - 1
        ipvt(k) = l

c        zero pivot implies this column already triangularized

        if (a(l,k) .eq. 0.0e0) go to 40

c           interchange if necessary
```

```
            if (l .eq. k) go to 10
               t = a(l,k)
               a(l,k) = a(k,k)
               a(k,k) = t
   10       continue

c           compute multipliers

            t = -1.0e0/a(k,k)
            call sscal(n-k,t,a(k+1,k),1)

c           row elimination with column indexing

            do 30 j = kp1, n
               t = a(l,j)
               if (l .eq. k) go to 20
                  a(l,j) = a(k,j)
                  a(k,j) = t
   20          continue
               call saxpy(n-k,t,a(k+1,k),1,a(k+1,j),1)
   30       continue
         go to 50
   40    continue
            info = k
   50    continue
   60 continue
   70 continue
      ipvt(n) = n
      if (a(n,n) .eq. 0.0e0) info = n
      return
      end



      real function sdot(n,sx,incx,sy,incy)

c     forms the dot product of two vectors.
c     uses unrolled loops for increments equal to one.
c     jack dongarra, linpack, 3/11/78.
c     modified 12/3/93, array(1) declarations changed to array(*)

C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      real sx(*),sy(*),stemp
      integer i,incx,incy,ix,iy,m,mp1,n

      stemp = 0.0e0
      sdot = 0.0e0
      if(n.le.0)return
      if(incx.eq.1.and.incy.eq.1)go to 20

c        code for unequal increments or equal increments
c          not equal to 1

      ix = 1
      iy = 1
      if(incx.lt.0)ix = (-n+1)*incx + 1
      if(incy.lt.0)iy = (-n+1)*incy + 1
      do 10 i = 1,n
        stemp = stemp + sx(ix)*sy(iy)
```

```
        ix = ix + incx
        iy = iy + incy
   10 continue
      sdot = stemp
      return

c       code for both increments equal to 1


c       clean-up loop

   20 m = mod(n,5)
      if( m .eq. 0 ) go to 40
      do 30 i = 1,m
        stemp = stemp + sx(i)*sy(i)
   30 continue
      if( n .lt. 5 ) go to 60
   40 mp1 = m + 1
      do 50 i = mp1,n,5
        stemp = stemp + sx(i)*sy(i) + sx(i + 1)*sy(i + 1) +
     *   sx(i + 2)*sy(i + 2) + sx(i + 3)*sy(i + 3) + sx(i + 4)*sy(i + 4)
   50 continue
   60 sdot = stemp
      return
      end



      subroutine sgesl(a,lda,n,ipvt,b,job)
C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      integer lda,n,ipvt(1),job
      real a(lda,1),b(1)

c     sgesl solves the real system
c     a * x = b  or  trans(a) * x = b
c     using the factors computed by sgeco or sgefa.

c     on entry

c       a       real(lda, n)
c               the output from sgeco or sgefa.
c
c       lda     integer
c               the leading dimension of the array  a .

c       n       integer
c               the order of the matrix  a .

c       ipvt    integer(n)
c               the pivot vector from sgeco or sgefa.

c       b       real(n)
c               the right hand side vector.

c       job     integer
c               = 0         to solve  a*x = b ,
c               = nonzero   to solve  trans(a)*x = b  where
c                           trans(a)  is the transpose.

c     on return
```

```
c       b       the solution vector  x .

c     error condition

c       a division by zero will occur if the input factor contains a
c       zero on the diagonal.  technically this indicates singularity
c       but it is often caused by improper arguments or improper
c       setting of lda .  it will not occur if the subroutines are
c       called correctly and if sgeco has set rcond .gt. 0.0
c       or sgefa has set info .eq. 0 .

c     to compute  inverse(a) * c  where  c  is a matrix
c     with  p  columns
c          call sgeco(a,lda,n,ipvt,rcond,z)
c          if (rcond is too small) go to ...
c          do 10 j = 1, p
c             call sgesl(a,lda,n,ipvt,c(1,j),0)
c       10 continue

c     linpack. this version dated 08/14/78 .
c     cleve moler, university of new mexico, argonne national lab.

c     subroutines and functions

c     blas saxpy,sdot

c     internal variables

      real sdot,t
      integer k,kb,l,nm1

      nm1 = n - 1
      if (job .ne. 0) go to 50

c       job = 0 , solve  a * x = b
c       first solve  l*y = b

        if (nm1 .lt. 1) go to 30
        do 20 k = 1, nm1
          l = ipvt(k)
          t = b(l)
          if (l .eq. k) go to 10
             b(l) = b(k)
             b(k) = t
   10       continue
          call saxpy(n-k,t,a(k+1,k),1,b(k+1),1)
   20   continue
   30   continue

c       now solve  u*x = y

        do 40 kb = 1, n
          k = n + 1 - kb
          b(k) = b(k)/a(k,k)
          t = -b(k)
          call saxpy(k-1,t,a(1,k),1,b(1),1)
   40   continue
        go to 100
   50 continue
```

```
c       job = nonzero, solve  trans(a) * x = b
c       first solve  trans(u)*y = b

        do 60 k = 1, n
          t = sdot(k-1,a(1,k),1,b(1),1)
          b(k) = (b(k) - t)/a(k,k)
   60   continue

c       now solve trans(l)*x = y

        if (nm1 .lt. 1) go to 90
        do 80 kb = 1, nm1
          k = n - kb
          b(k) = b(k) + sdot(n-k,a(k+1,k),1,b(k+1),1)
          l = ipvt(k)
          if (l .eq. k) go to 70
            t = b(l)
            b(l) = b(k)
            b(k) = t
   70     continue
   80   continue
   90   continue
  100 continue
      return
      end



C-----------------SUBROTINA INTK1

      SUBROUTINE INTK1(N,X,XX,Z,ZZ)
C     DOUBLE QUADRATIC (SMOOTH) INTER-OR EXTRAPOLATION
C     INPUT     N =NUMBER OF X VALUES AND NUMBER OF FUNCTION VALUES
C     -----       N MUST BE GREATER THAN 2
C               X =ARRAY OF X-VALUES
C               XX=X-VALUE TO BE INTERPOLATED
C               Z =ARRAY OF FUNCTION VALUES
C     OUTPUT   ZZ=INTERPOLATED FUNCTION VALUE.
C     ------
C     REMARK..THE X-VALUES NEED NOT BE EQUIDISTANT BUT MUST
C             BE MONOTONICALLY INCREASING.

      INTEGER*4 I,IQ,IR,J,JJ,K,M,N
      REAL*4 FF,X,XX,Z,ZZ
      DIMENSION X(N),Z(N)

      I=1
      JJ=I
      ZZ=0.0
      DO 50 J=1,N
      FF=XX-X(J)
      IF (FF) 20,10,50
   10 ZZ=Z(J)
      RETURN
   20 IF(J.EQ.1.OR.J.EQ.2) GOTO 60
      IF(J.NE.N) JJ=2
      GOTO 56
   50 CONTINUE
      J=N
   56 I=J-2
```

```
60   M=I+2
     IR=M
     IQ=I+1
     FF=0.0
     DO 80 K=I,M
     FF=FF+Z(K)*(XX-X(IQ))*(XX-X(IR))/((X(K)-X(IQ))*(X(K)-X(IR)))
     IQ=IR
     IR=K
80   CONTINUE
     IF(JJ.NE.1) GOTO 90
     ZZ=FF
     RETURN
90   IR=JJ+I
     ZZ=ZZ+FF*(XX-X(IR))/(X(I+1)-X(IR))
     IF(JJ.EQ.0) RETURN
     JJ=0
     I=J-1
     GOTO 60
     END
```

A.2 Program List of *OptiPropII*

```
================================================
      Optimização de Hélices Propulsores Marítimos
              em escoamento invíscido
================================================

      PROGRAM OPTIPROP


C     DEFINICAO DAS VARIAVEIS DE ENTRADA
C     ++++++++++++++++++++++++++++++++++

C     IPV: PARAMETRO QUE PERMITE AO UTILIZADOR A OPCAO ENTRE
C       CARREGAMENTO LIGEIRO (1) OU CARREGAMENTO MODERADO (2)
C     N: NUMERO DE PONTOS INTERIORES SOBRE A PA
C     NB: NUMERO DE PAS
C     CT OU CP: COEFICIENTE ADIMENSIONAL DE IMPULSO OU POTENCIA
C     XH: RAIO DO CUBO ADIMENSIONAL
C     J0: COEFICIENTE DE AVANCO
C     NX: NUMERO DE PONTOS ONDE E DADA A FRACCAO DE ESTEIRA EFECTIVA
C     XW: RAIOS ADIMENSIONAIS PARA CADA UM DESSES PONTOS
C     WT1: FRACCAO DE ESTEIRA EFECTIVA NOS PONTOS XW
C     WM: FRACCAO DE ESTEIRA MEDIA EFECTIVA


C     DEFINICAO DAS VARIAVEIS DE SAIDA
C     ++++++++++++++++++++++++++++++++

C     X: COTA DOS PONTOS CONSIDERADOS SOBRE A PA
C     G(X): DISTRIBUICAO DE CIRCULACAO ADIMENSIONAL
C     UA(X): VELOCIDADE AXIAL INDUZIDA ADIMENSIONAL
C     UT(X): VELOCIDADE TANGENCIAL INDUZIDA ADIMENSIONAL
C     BETA(X): ANGULO DE PASSO HIDRODINAMICO
C     BETAIN(X): ANGULO DE PASSO HIDRODINAMICO INDUZIDO
C     BETAV(X): ÂNGULO DE SAÍDA DOS VÓRTICES
C     VEL(X): VELOCIDADE RESULTANTE
C     PASSO(X): PASSO ADIMENSIONALIZADO PELO DIAMETRO
C     CP: COEFICIENTE ADIMENSIONAL DE POTENCIA
C     CT: COEFICIENTE ADIMENSIONAL DE IMPULSO
C     REND: RENDIMENTO (CT/CP)


C     DEFINICAO DAS VARIAVEIS DO PROGRAMA
C     +++++++++++++++++++++++++++++++++++

C     N0: NUMERO DE PONTOS INTERIORES MAXIMO
C     NP0: NUMERO PONTOS MAXIMO
C     N: NUMERO DE PONTOS INTERIORES
C     NP:NUMERO DE PONTOS
C     TETA: TABELA DE EQUIPARACAO ANGULAR PARA CADA PONTO
C     A: MATRIZ DOS FACTORES TRIGONOMETRICOS
C     X: TABELA DOS RAIOS ADIMENSIONAIS PARA UM N DADO
C     BETA: TABELA DOS ANGULOS DE PASSO HIDRODINAMICOS
C     BETAIN: TABELA DOS ANGULOS DE PASSO HIDRODINAMICOS INDUZIDOS
C           ANTIGOS
```

```
C    BETAV: TABELA DOS ANGULOS DE SAIDA DOS VORTICES
C    LANDAIN: COEFICIENTE DE VELOCIDADE DE APROXIMACAO INDUZIDO
C    G: DISTRIBUICAO DE CIRCULACAO
C    UAMAT: MATRIZ DE INDUCAO AXIAL
C    UTMAT: MATRIZ DE INDUCAO TANGENCIAL
C    UA: TABELA DAS VELOCIDADES INDUZIDAS AXIAIS
C    UT: TABELA DAS VELOCIDADES INDUZIDAS TANGENCIAIS
C    B: VECTOR DE REAIS PARA RESOLUCAO DO SISTEMA
C    AM: MATRIZ DO SISTEMA
C    VEL: VELOCIDADE RESULTANTE
C    PASSO: PASSO ADIMENSIONALIZADO PELO DIAMETRO
C    XH: RAIO ADIMENSIONAL INTERIOR
C    EPSLANDA: PARAMETRO DE CONVERGENCIA PARA O LANDAIN
C    PI: CONSTANTE
C    ERRO: VARIAVEL DE ERRO
C    INFO: VARIAVEL DE INFORMACAO PARA RESOLUCAO DO SISTEMA
C    REND: RENDIMENTO
C    W: FACTOR DE SUB-RELAXACAO
C    WT: FRACCAO DE ESTEIRA EFECTIVA NOS RAIOS DE GLAUERT
C    GN_1(X): DISTRIBUICAO DE CIRCULACAO ADIMENSIONAL PARA A N-1
C          ITERACÇÃO
C    BETAVN_1: BETAV PARA A N-1 ITERACÇÃO
C    BETAVCONV: (1) SE CONVERGE OU (0) CASO CONTRÁRIO
C    GCONV: (1) SE CONVERGE OU (0) CASO CONTRÁRIO
C    TOLERANCE: TOLERÂNCIA PARA A CONVERGÊNCIA DE G E BETAV


     INTEGER N0,NP0,N,NP,I,J,K,NB,IPV,INFO,NX
     INTEGER GHANGE,GCONV,BETAVCONV,ASK,ITNUM
     PARAMETER(N0=29,NP0=N0+2)
     INTEGER IPVT(N0+1)
     REAL A(NP0,N0,NP0),TETA(NP0),X(NP0),BETAV(NP0),BETAVN_1(NP0)
     REAL BETAIN(NP0),LANDAIN,G(NP0+1),GN_1(NP0+1)
     REAL BETA(NP0),UAMAT(NP0,N0),UTMAT(NP0,N0),UA(NP0),UT(NP0)
     REAL B(N0+1),AM(N0+1,N0+1),VEL(NP0),PASSO(NP0),SOMA(N0)
     REAL XH,EPSLANDA,CT,CP,J0,PI,ERRO,TOLERANCE,TOLBETA
     REAL CP1,REND,W,XX(17),G1(17),BETA1(17),UTN_1(NP0)
     REAL BETAIN1(17),UT1(17),WT(NP0),XW(NP0),WM,WT1(NP0),XP(NP0)
     REAL RELAX
     PARAMETER(RELAX=0.5)

     DATA EPSLANDA /0.01/
     DATA TOLERANCE /0.0005/

     OPEN(UNIT=5,FILE='OptiProp.INP',STATUS='UNKNOWN')
     OPEN(UNIT=6,FILE='OptiProp.OUT',STATUS='UNKNOWN')


     READ(5,*) IPV
     PRINT*,IPV
     READ(5,*) N
     NP=N+2
     PRINT*,N
     PRINT*,NP
     READ(5,*) NB
     PRINT*,NB
     READ(5,*) CT
     PRINT*,CT
     READ(5,*) XH
     PRINT*,XH
```

```
      READ(5,*) J0
      PRINT*,J0
      READ(5,*) NX
      PRINT*,NX
      READ(5,*) (XW(I),I=1,NX)
      PRINT*,(XW(I),I=1,NX)
      READ(5,*) (WT1(I),I=1,NX)
      PRINT*,(WT1(I),I=1,NX)
      READ(5,*) WM
      PRINT*,WM


C-----ECO DOS DADOS DE ENTRADA

      WRITE(6,1)
 1    FORMAT(/,10X,'DADOS DE ENTRADA',/,10X,16(1H=))
      WRITE(6,2) IPV
 2    FORMAT(/,4X,'IPV=',I2,5X)
      WRITE(6,3) N,NB,XH,J0
 3    FORMAT(/,4X,'N=',I2,5X,'NB=',I2,5X,'XH=',F6.3,5X,'J0=',F6.3)
      WRITE(6,5) CT
 5    FORMAT(/,4X,'CT=',F9.5)
      WRITE(6,10) (WT1(I),I=1,NX)
 10   FORMAT(/,4X,'WT=',F5.3)
      WRITE(6,11) WM
 11   FORMAT(/,4X,'WM=',F5.3)
      WRITE(6,12) TOLERANCE*100
 12   FORMAT(/,4X,'Tolerância = ',F7.5,' %')



C-----TESTE DE VALIDADE DOS PARAMETROS ------

      ERRO=0
      IF ((IPV.NE.1).AND.(IPV.NE.2)) THEN
        ERRO=1
      ENDIF
      IF (NB.LE.0) THEN
        ERRO=1
    ELSE
        IF (CT.LE.0) THEN
          ERRO=1
      ELSE
          IF (XH.LT.0) THEN
            ERRO=1
       ELSE
              IF (XH.GE.1) THEN
            ERRO=1
              ELSE
            IF (J0.LE.0) THEN
              ERRO=1
                ELSE
                  IF (N.LE.0) THEN
                ERRO=1
                  ENDIF
                ENDIF
              ENDIF
       ENDIF
      ENDIF
    ENDIF
```

```
      IF (ERRO.EQ.1) THEN
        PRINT *,'VALOR DOS PARAMETROS DE ENTRADA ERRADO'
       GOTO 999
      ENDIF


C-----CALCULO DE PI

      PI=4.*ATAN(1.)


C-----CALCULO DOS VALORES DE TETA E X PARA NP PONTOS

      DO 13 J=1,NP
      TETA(J)=FLOAT(J-1)*PI/FLOAT(N+1)
      X(J)=0.5*(1.+XH) - 0.5*(1.-XH)*COS(TETA(J))
 13   CONTINUE


C-----INTERPOLACAO DA FRACCAO DE ESTEIRA EFECTIVA PARA
C     RAIOS DE GLAUERT

      DO 14 I=1,NP
        CALL INTK1(NX,XW,X(I),WT1,WT(I))
 14   CONTINUE


C-----CALCULO DAS MATRIZES DOS FACTORES TRIGONOMETRICOS A

      CALL AMAT(N0,NP0,N,NP,TETA,A)


C-----CORRECCAO DA MATRIZ A PARA O CASO DE XH=0

      IF (XH.EQ.0) THEN
        DO 15 I=1,N
          A(1,I,1)=0.
 15     CONTINUE
      ENDIF


C-----CALCULO DE BETA E BETAIN
      IF (X(1).EQ.0.) THEN
        BETA(1)=PI/2
        DO 16 I=2,NP
          BETA(I)=ATAN(J0*(1.-WT(I))/(PI*X(I)*(1.-WM)))
 16     CONTINUE
      ELSE
        DO 17 I=1,NP
          BETA(I)=ATAN(J0*(1.-WT(I))/(PI*X(I)*(1.-WM)))
 17     CONTINUE
      ENDIF
      CALL BETA_INICT(NP0,N,NP,EPSLANDA,J0,CT,WT,WM,
     &                TETA,X,BETAIN,LANDAIN)
      IF (X(1).EQ.0.) THEN
        BETAIN(1)=PI/2.
    ENDIF
```

```
C-----CALCULO DE BETAV PASSO DOS VORTICES

      IF (IPV.EQ.1) THEN
      DO 18 I=1,NP
        BETAV(I)=BETA(I)
 18             CONTINUE
   ELSE
      DO 19 I=1,NP
        BETAV(I)=BETAIN(I)
 19             CONTINUE
      ENDIF


C-----CALCULO DE UTN_1 E GN_1

   DO 20 I=1,NP+1
     G(I)=0.
     GN_1(I)=0.
 20           CONTINUE
   DO 30 I=1,NP
     UTN_1(I)=SIN(BETAIN(I))*COS(BETAIN(I))*
    &           (X(I)*PI/J0*TAN(BETAIN(I))-(1.-WT(I))/(1.-WM))
 30           CONTINUE
   ASK=1
   ITNUM=0


C-----CALCULO DAS MATRIZES DE INDUCAO

 120  CALL MAT_IND(N0,NP0,N,NP,XH,A,BETAV,NB,X,
     &              UAMAT,UTMAT,ERRO)
      IF (ERRO.EQ.1) THEN
        PRINT*,'ERRO: EXCESSO DE CAPACIDADE'
        GOTO 999
      ENDIF


C-----CALCULO DE G(X)

C-------CONSTRUCAO DA MATRIZ DO SISTEMA LINEAR

 130    DO 150 I=1,N
        B(I)=(-1.)*(1.-WT(I+1))/(1.-WM)*X(I+1)*SIN(TETA(I+1))
        DO 140 J=1,N
          AM(I,J)=UAMAT(I+1,J)*X(I+1)*SIN(TETA(I+1))
    &                +UAMAT(J+1,I)*X(J+1)*SIN(TETA(J+1))
 140      CONTINUE
 150    CONTINUE
       DO 152 I=1,N
       SOMA(I)=0.
              DO 151 J=1,N
           SOMA(I)=SOMA(I)+GN_1(J+1)*(UTMAT(J+1,I)*SIN(TETA(J+1))
    &                 +UTMAT(I+1,J)*SIN(TETA(I+1)))
 151      CONTINUE
 152    CONTINUE
       DO 154 J=1,N
       AM(N+1,J)=(2.*PI*NB*(1.-XH)/(N+1.))*SIN(TETA(J+1))*
    &             (X(J+1)*PI/J0-UTN_1(J+1))
       AM(J,N+1)=X(J+1)*SIN(TETA(J+1))-J0/PI*SOMA(J)
 154           CONTINUE
```

```
      AM(N+1,N+1)=0.
    B(N+1)=CT


C-------RESOLUCAO DO SISTEMA

      ITNUM=ITNUM+1
    CALL SGEFA(AM,N0+1,N+1,IPVT,INFO)
      IF (INFO.NE.0) THEN
       PRINT*,'SISTEMA SEM RESOLUCAO'
       GOTO 999
      ENDIF
      CALL SGESL(AM,N0+1,N+1,IPVT,B,0)


C-------AFECTACAO DOS RESULTADOS

      DO 160 I=1,N
       G(I+1)=B(I)
 160   CONTINUE
      G(1)=0.
      G(NP)=0.
    G(NP+1)=B(N+1)


C-------RELAXACAO DE G

    DO 162 I=1,NP+1
      G(I)=GN_1(I)+RELAX*(G(I)-GN_1(I))
 162          CONTINUE


C-------CALCULO DAS VELOCIDADES INDUZIDAS

      DO 165 K=1,NP
       UA(K)=0.
       UT(K)=0.
       DO 164 I=1,N
        UA(K)=UA(K)+G(I+1)*UAMAT(K,I)
        UT(K)=UT(K)+G(I+1)*UTMAT(K,I)
 164    CONTINUE
 165   CONTINUE


C-------CONVERGENCIA DE G

      GCONV=1
    DO 172 I=2,NP+1
      IF ((ABS(G(I)-GN_1(I))/G(I)).GT.TOLERANCE) THEN
        GCONV=0
       ENDIF
 172          CONTINUE

   DO 190 I=1,NP
      GN_1(I)=G(I)
      BETAVN_1(I)=BETAV(I)
    UTN_1(I)=UT(I)
 190        CONTINUE
    GN_1(NP+1)=G(NP+1)
```

```
C-----CALCULO DE BETA

 210  IF (X(1).EQ.0.) THEN
       BETA(1)=PI/2
       DO 214 I=2,NP
        BETA(I)=ATAN(J0*(1.-WT(I))/(PI*X(I)*(1.-WM)))
 214    CONTINUE
      ELSE
       DO 216 I=1,NP
        BETA(I)=ATAN(J0*(1.-WT(I))/(PI*X(I)*(1.-WM)))
 216    CONTINUE
      ENDIF


C-----CALCULO DE BETAIN

      IF (X(1).EQ.0.) THEN
       BETAIN(1)=PI/2
       DO 218 I=2,NP
        BETAIN(I)=ATAN(((1.-WT(I))/(1.-WM)+UA(I))/
     &                (X(I)*PI/J0-UT(I)))
 218    CONTINUE
      ELSE
       DO 220 I=1,NP
        BETAIN(I)=ATAN(((1.-WT(I))/(1.-WM)+UA(I))/
     &                (X(I)*PI/J0-UT(I)))
 220    CONTINUE
      ENDIF


C-----CALCULO DE BETAV PASSO DOS VORTICES

      IF (IPV.EQ.1) THEN
      DO 221 I=1,NP
        BETAV(I)=BETA(I)
 221            CONTINUE
  ELSE
       IF (GCONV.EQ.0) THEN
       DO 222 I=1,NP
         BETAV(I)=BETAVN_1(I)
 222             CONTINUE
     ENDIF
     ENDIF
     IF (GCONV.EQ.1) THEN
     DO 223 I=1,NP
        BETAV(I)=BETAIN(I)
 223            CONTINUE
  ENDIF


C-------CONVERGENCIA DE BETAV

       BETAVCONV=1
    DO 224 I=1,NP
      IF ((ABS(BETAV(I)-BETAVN_1(I))/BETAV(I)).GT.TOLERANCE)
    &        THEN
             BETAVCONV=0
         ENDIF
 224           CONTINUE
```

```
C-------ROTINA DE CONVERGENCIA EM G E BETA

  IF (GCONV.EQ.0) THEN
        GOTO 130
  ELSE
    IF (BETAVCONV.EQ.0) THEN
          GOTO 120
    ENDIF
  ENDIF


C******************BLOCO DE OUTPUT*****************************

C-----CALCULO DO MODULO DA VELOCIDADE TOTAL

 225    DO 230 I=1,NP
        VEL(I)=(UA(I)+(1.-WT(I))/(1-WM))/(SIN(BETAIN(I)))
 230    CONTINUE

C-----CALCULO DO COEFICIENTE DE POTENCIA

       CP=0.
       DO 240 I=1,NP
         CP=CP+G(I)*((1.-WT(I))/(1.-WM)+UA(I))*X(I)*SIN(TETA(I))
 240    CONTINUE
       CP=CP*NB*2.*(PI**2)*(1.-XH)/(J0*(N+1.))


C-----CALCULO DE RENDIMENTO

     REND=CT/CP

C-----CALCULO DO PASSO

     IF (X(1).EQ.0.) THEN
       DO 260 I=2,NP
         PASSO(I)=PI*X(I)*TAN(BETAIN(I))
 260    CONTINUE
     ELSE
       DO 261 I=1,NP
         PASSO(I)=PI*X(I)*TAN(BETAIN(I))
 261    CONTINUE
     ENDIF

C-----APRESENTACAO DOS ANGULOS EM GRAUS

     DO 262 I=1,NP
       BETA(I)=BETA(I)*180./PI
 262  CONTINUE

     DO 264 I=1,NP
       BETAIN(I)=BETAIN(I)*180./PI
 264  CONTINUE


C**************FIM DO BLOCO DE OUTPUT****************
```

```
C-----AFECTACAO DOS RESULTADOS

 265  FORMAT(/,F9.5)

      WRITE(6,275)
 275  FORMAT(///,10X,'RESULTADOS DO PROGRAMA OPTIPROP',/,10X,30(1H=))

      WRITE(6,265)

C-----PRIMEIRA TABELA DE RESULTADOS

      WRITE(6,285)
 285  FORMAT(/,3X,'I',6X,'X',7X,'G',8X,'UA',7X,'UT',/)

      DO 290 I=1,NP
        WRITE(6,295) I,X(I),G(I),UA(I),UT(I)
 290  CONTINUE
 295  FORMAT(I4,4F9.5)


C-----SEGUNDA TABELA DE RESULTADOS

      WRITE(6,298)
 298  FORMAT(//,3X,'I',6X,'B',7X,'Bi',8X,'V',7X,'P/D',/)
      DO 300 I=1,NP
        WRITE (6,302) I,BETA(I),BETAIN(I),VEL(I),PASSO(I)
 300  CONTINUE
 302  FORMAT(I4,2F9.2,2F9.5)

      WRITE(6,305) CP
 305  FORMAT(//,10X,'COEFICIENTE DE POTENCIA: CP=',F6.3)

      WRITE(6,315) CT
 315  FORMAT(//,10X,'COEFICIENTE DE IMPULSO: CT=',F6.3)

      WRITE(6,325) REND
 325  FORMAT(//,10X,'RENDIMENTO=',F6.3)
      WRITE(6,*)
    WRITE(6,330) ITNUM
 330  FORMAT(//,10X,'Numero de Iterações = ',I3,3X)


C-----INTERPOLACAO PARA RAIOS CONVENCIONAIS

      XX(1)=XH
      DO 350 I=2,17
        XX(I)=XX(I-1)+0.05
 350  CONTINUE
      DO 360 I=1,17
        CALL INTK1(NP,X,XX(I),G,G1(I))
 360  CONTINUE
      DO 370 I=1,17
        CALL INTK1(NP,X,XX(I),BETA,BETA1(I))
 370  CONTINUE
      DO 380 I=1,17
        CALL INTK1(NP,X,XX(I),BETAIN,BETAIN1(I))
 380  CONTINUE
      DO 390 I=1,17
        CALL INTK1(NP,X,XX(I),UT,UT1(I))
 390  CONTINUE
```

```
C-----FECHO DOS FICHEIROS DE INP,OUT,DAT

      CLOSE(UNIT=5)
      CLOSE(UNIT=6)

 999  STOP
      END


C=====================================================================
C
C          ESCRITA DAS DIFERENTES SUBROTINAS
C          ++++++++++++++++++++++++++++++++++

C------------------SUBROTINA AMAT

   SUBROUTINE AMAT(N0,NP0,N,NP,TETA,A)



C     ESTA SUBROTINA CALCULA A MATRIZ DOS FACTORES
C     TRIGONOMETRICOS PARA UM DETERMINADO NUMERO DE
C     PONTOS DADO


C  PARAMETROS DE ENTRADA

C        N0: NUMERO DE PONTOS INTERIORES MAXIMO
C        NP0: NUMERO DE PONTOS MAXIMO
C        N: NUMERO DE PONTOS INTERIORES
C        NP:NUMERO DE PONTOS
C        TETA: TABELA DE EQUIPARACAO ANGULAR PARA CADA PONTO


C  PARAMETROS DE SAIDA

C        A: MATRIZ DOS FACTORES TRIGONOMETRICOS



   INTEGER N0,NP0,N,NP,I,J,K,L
   REAL A(NP0,N0,NP0),TETA(NP0)
   REAL B1,B2,PI,H1,H2

   PI=4.*ATAN(1.)

C       MATRIZ A

      DO 50 I=1,N
       DO 40 J=1,NP
         DO 30 K=1,NP

       B1=0.5
       IF (((K.EQ.1).AND.(J.EQ.1)).OR.((K.EQ.NP).AND.(J.EQ.NP))) THEN
              B2=0
           DO 10 M=1,N+1
             IF (M.EQ.(N+1)) THEN
               H1=0.5
```

```
          ELSE
            H1=1
          ENDIF
          B2=B2+FLOAT(M)*H1*COS(FLOAT(M)*TETA(J))*
     &         COS(FLOAT(M)*TETA(K))/COS(TETA(K))
 10       CONTINUE
      ELSE
            H1 =FLOAT((J+K-2)/2)
            H2 = FLOAT(J+K-2)/2.
            IF (ABS(H1-H2).LT.0.01) THEN
          B2 = 0.
            ELSE
          B2 = -1. / (COS(TETA(K)) - COS(TETA(J)))
            ENDIF
      ENDIF


      A(K,I,J) = 0.

        DO 20 L=1,N

          IF (K.EQ.1.OR.K.EQ.NP) THEN
               H1 = FLOAT(L) * COS(FLOAT(L)*TETA(K)) / COS(TETA(K))
          ELSE
               H1 = SIN(FLOAT(L)*TETA(K)) / SIN(TETA(K))
          ENDIF

          B1 = B1 + COS(FLOAT(L)*TETA(J)) * COS(FLOAT(L)*TETA(K))
          B2 = B2 - COS(FLOAT(L)*TETA(J)) * H1

          A(K,I,J) = A(K,I,J) + FLOAT(L)*SIN(FLOAT(L)*TETA(I+1))
     &                    * (H1*B1 + COS(FLOAT(L)*TETA(K))*B2)
 20       CONTINUE

     A(K,I,J) = A(K,I,J) * 4.*PI/(FLOAT(N+1))**2

 30     CONTINUE
 40    CONTINUE
 50   CONTINUE


    RETURN
    END



C------------------SUBROTINA INFAC

      SUBROUTINE INFAC(X0,X1,BETAV,NB,IA,IT,ERRO)



C     ESTA SUBROTINA CALCULA OS FACTORES DE INDUCAO
C     PARA O QUOCIENTE X0/X1


C  PARAMETROS DE ENTRADA

C      X0: RAIO ADIMENSIONAL PARA ORIGEM DO VORTICE
C      X1: RAIO ADIMENSIONAL PARA O PONTO DE CALCULO
```

```
C     BETAV: ANGULO DE PASSO HIDRODINAMICO
C     NB: NUMERO DE PAS
C     ERRO: VARIAVEL DE ERRO


C  PARAMETROS DE SAIDA
C      IA: FACTOR DE INDUCAO AXIAL PARA K,L
C      IT: FACTOR DE INDUCAO TANGENCIAL PARA K,L



      REAL X0X,A,B,F,G,U,P,IA,IT,ERRO
      INTEGER NB
      REAL BETAV,X0,X1
      DATA TOL /1E-12/



        IF (BETAV.LE.0) THEN
          ERRO=1
        ENDIF


      IF (ERRO.EQ.1) THEN
        GOTO 40
    ENDIF

C---CALCULO DOS FACTORES DE INDUCAO

      IF ((X0.LT.TOL).AND.(X1.LT.TOL)) THEN
        IA=0
        IT=1
        GOTO 40
      ENDIF
      IF (X1.LT.TOL) THEN
        IA=NB/TAN(BETAV)
        IT=0
        GOTO 40
      ENDIF
      X0X=X0/X1
      IF (X0X.LT.TOL) THEN
        IA=0
        IT=NB
        GOTO 40
      ENDIF
    IF ((1/(X0X)).EQ.1) THEN
        IA=COS(BETAV)
        IT=SIN(BETAV)
        GOTO 40
    ELSE
      IF (X0X.GT.1) THEN
          GOTO 20
      ELSE
        IF (X0X.LT.0.2) THEN
            GOTO 30
        ELSE
            IF(BETAV.GE.(0.179))THEN
              GOTO 20
                ELSE
                  IF (X0X.LT.0.8) THEN
```

```
            GOTO 30
              ELSE
            IF (BETAV.GE.(0.012)) THEN
              GOTO 20
                ELSE
              GOTO 30
                ENDIF
              ENDIF
            ENDIF
        ENDIF
      ENDIF
    ENDIF



20   P=(1+((1/X0X)**2)/((TAN(BETAV))**2))**0.5
     U=(((P-1)*X0X/(1/SIN(BETAV)-1))**NB)*
    &  EXP(NB*(P-1/SIN(BETAV)))

     IF (ABS(1-U).LE.1E-38) THEN
       ERRO=1
       PRINT *,'ERRO: EXCESSO DE CAPACIDADE'
       GOTO 40
     ENDIF

     G=((SIN(BETAV))**3)*(2+9/((TAN(BETAV))**2))+
    &  (3*P-5)/(P**3)
     F=1/(SQRT(P*SIN(BETAV)))
     IF (X0X.GT.1) THEN
       B=F*(U/(1-U)+(1/(24*NB))*G*LOG(ABS(1/(1-U))))
       IA=(1-(1/(X0X)))*(NB*(1+B))/TAN(BETAV)
       IT=(X0X-1)*NB*B
     ELSE
       IF (X0X.LT.1) THEN
         A=F*(1/(U-1)-(1/(24*NB))*G*LOG(ABS(U/(U-1))))
         IA=(1/X0X-1)*(NB*A)/TAN(BETAV)
         IT=(1-X0X)*NB*(1+A)
       ENDIF
     ENDIF
     GOTO 40


30   IA=0
     IT=NB-NB*X0X



40   RETURN
     END



C-------------------------SUBROTINA MAT_IND

     SUBROUTINE MAT_IND(N0,NP0,N,NP,XH,A,BETAV,NB,X,
    &                   UAMAT,UTMAT,ERRO)



C      ESTA SUBROUTINA CALCULA AS MATRIZES DE INDUCAO
```

```
C     UAMAT E UTMAT


C  PARAMETROS DE ENTRADA

C       N0: NUMERO DE PONTOS INTERIORES MAXIMO
C       NP0: NUMERO DE PONTOS MAXIMO
C       N: NUMERO DE PONTOS INTERIORES
C       NP: NUMERO DE PONTOS
C       X: TABELA DOS RAIOS ADIMENSIONAIS
C       XH: RAIO INTERIOR ADIMENSIONAL
C       BETAV: ANGULO DE PASSO HIDRODINAMICO
C       NB: NUMERO DE PAS
C       A: MATRIZ DOS FACTORES TRIGONOMETRICOS


C  PARAMETROS DE SAIDA

C       UAMAT: MATRIZ AXIAL DE INDUCAO
C       UTMAT: MATRIZ TANGENCIAL DE INDUCAO
C       ERRO: VARIAVEL DE ERRO



  INTEGER N0,NP0,N,NP,I,J,K,NB
  REAL A(NP0,N0,NP0),BETAV(NP0),X(NP0)
  REAL UAMAT(NP0,N0),UTMAT(NP0,N0)
  REAL XH,LAMBJ,IA,IT,ERRO

C---MATRIZ DOS FACTORES DE INDUCAO

     ERRO=0

     DO 30 I=1,N
      DO 20 K=1,NP
        UAMAT(K,I)=0.
        UTMAT(K,I)=0.
          DO 10 J=1,NP
            IF(J.EQ.1.OR.J.EQ.NP) THEN
              LAMBJ=0.5
            ELSE
              LAMBJ=1.0
            ENDIF

C---CALCULO DOS FACTORES DE INDUCAO

            CALL INFAC(X(J),X(K),BETAV(J),NB,IA,IT,ERRO)

            IF (ERRO.EQ.1) THEN
              PRINT *,'ERRO: MUDAR OS PARAMETROS DE ENTRADA'
              GOTO 40
            ENDIF

            UAMAT(K,I) = UAMAT(K,I) + IA*LAMBJ*A(K,I,J)
            UTMAT(K,I) = UTMAT(K,I) + IT*LAMBJ*A(K,I,J)
 10         CONTINUE
        UAMAT(K,I) = UAMAT(K,I) / (1.-XH)
        UTMAT(K,I) = UTMAT(K,I) / (1.-XH)
 20    CONTINUE
 30   CONTINUE
```

```
 40   RETURN
  END



C----------------------SUBROTINA BETA_INICT

      SUBROUTINE BETA_INICT(NP0,N,NP,EPSLANDA,J0,CT,WT,WM,
     &                  TETA,X,BETAIN,LANDAIN)



C      ESTA SUBROTINA CALCULA OS VALORES DOS ANGULOS
C      DE PASSO HIDRODINAMICOS INICIAIS SENDO DADO O
C      COEFICIENTE DE IMPULSO


C  PARAMETROS DE ENTRADA

C      NP0:NUMERO DE PONTOS MAXIMO
C      N: NUMERO DE PONTOS INTERIORES
C      NP: NUMERO DE PONTOS
C      EPSLANDA: PARAMETRO DE CONVERGENCIA DE LANDA
C      J0: COEFICIENTE DE AVANCO
C      CT: COEFICIENTE DE IMPULSO
C      TETA: TABELA DE EQUIPARACAO ANGULAR
C      X: TABELA DOS RAIOS ADIMENSIONAIS
C      WT: FRACCAO DE ESTEIRA EFECTIVA PARA CADA PONTO
C      WM: FRACCAO DE ESTEIRA MEDIA EFECTIVA


C  PARAMETROS DE SAIDA

C      BETAIN: TABELA DOS ANGULOS DE PASSO HIDRODINAMICOS
C      LANDAIN: COEFICIENTE DE VELOCIDADE DE APROXIMACAO


  INTEGER NP0,N,NP,I
    REAL EPSLANDA,LANDA,CT,CTA,H
    REAL LANDAIN,LANDAIV,J0,WT(NP0),WM
  REAL BETAIN(NP0),X(NP0),TETA(NP0)


  PI= 4.*ATAN(1.)
  LANDA=0

     DO 10 I=1,NP
      LANDA= LANDA +((1-WT(I))*J0/(PI*(1-WM)))*SIN(TETA(I))
 10   CONTINUE

  LANDA= LANDA*PI/(2*(N+1))
    CTA= CT*((J0/(LANDA*PI))**2)
  LANDAIN=0.5*LANDA*(1+SQRT(1+CTA))

 20   LANDAIV=LANDAIN
  H=1+LANDAIV*(LANDAIV-LANDA)/(1+LANDAIV**2)
  H=H-LANDAIV*(2*LANDAIV-LANDA)*
```

```
     +ALOG((1+LANDAIV**2)/(LANDAIV**2))
    LANDAIN=0.5*LANDA*(1+SQRT(1+CTA/H))
      IF (ABS(LANDAIV-LANDAIN).GE.EPSLANDA)THEN
        GOTO 20
    ENDIF

C-----CASO DE XH SER IGUAL A ZERO OU DIFERENTE DE ZERO

      IF (X(1).EQ.0) THEN
        BETAIN(1)=PI/2
        DO 30 I=2,NP
          BETAIN(I)=ATAN(LANDAIN/X(I))
30      CONTINUE
      ELSE
        DO 40 I=1,NP
          BETAIN(I)=ATAN(LANDAIN/X(I))
40      CONTINUE
      ENDIF


    RETURN
    END



      integer function isamax(n,sx,incx)

c     finds the index of element having max. absolute value.
c     jack dongarra, linpack, 3/11/78.
c     modified 3/93 to return if incx .le. 0.
c     modified 12/3/93, array(1) declarations changed to array(*)

C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      real sx(*),smax
      integer i,incx,ix,n

      isamax = 0
      if( n.lt.1 .or. incx.le.0 ) return
      isamax = 1
      if(n.eq.1)return
      if(incx.eq.1)go to 20

c       code for increment not equal to 1

      ix = 1
      smax = abs(sx(1))
      ix = ix + incx
      do 10 i = 2,n
        if(abs(sx(ix)).le.smax) go to 5
        isamax = i
        smax = abs(sx(ix))
   5    ix = ix + incx
   10 continue
      return

c       code for increment equal to 1

   20 smax = abs(sx(1))
      do 30 i = 2,n
        if(abs(sx(i)).le.smax) go to 30
```

```
          isamax = i
          smax = abs(sx(i))
   30 continue
      return
      end



      subroutine saxpy(n,sa,sx,incx,sy,incy)

c     constant times a vector plus a vector.
c     uses unrolled loop for increments equal to one.
c     jack dongarra, linpack, 3/11/78.
c     modified 12/3/93, array(1) declarations changed to array(*)

C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      real sx(*),sy(*),sa
      integer i,incx,incy,ix,iy,m,mp1,n

      if(n.le.0)return
      if (sa .eq. 0.0) return
      if(incx.eq.1.and.incy.eq.1)go to 20

c       code for unequal increments or equal increments
c         not equal to 1

      ix = 1
      iy = 1
      if(incx.lt.0)ix = (-n+1)*incx + 1
      if(incy.lt.0)iy = (-n+1)*incy + 1
      do 10 i = 1,n
        sy(iy) = sy(iy) + sa*sx(ix)
        ix = ix + incx
        iy = iy + incy
   10 continue
      return

c       code for both increments equal to 1


c       clean-up loop

   20 m = mod(n,4)
      if( m .eq. 0 ) go to 40
      do 30 i = 1,m
        sy(i) = sy(i) + sa*sx(i)
   30 continue
      if( n .lt. 4 ) return
   40 mp1 = m + 1
      do 50 i = mp1,n,4
        sy(i) = sy(i) + sa*sx(i)
        sy(i + 1) = sy(i + 1) + sa*sx(i + 1)
        sy(i + 2) = sy(i + 2) + sa*sx(i + 2)
        sy(i + 3) = sy(i + 3) + sa*sx(i + 3)
   50 continue
      return
      end
```

```
      subroutine sscal(n,sa,sx,incx)

c     scales a vector by a constant.
c     uses unrolled loops for increment equal to 1.
c     jack dongarra, linpack, 3/11/78.
c     modified 3/93 to return if incx .le. 0.
c     modified 12/3/93, array(1) declarations changed to array(*)

C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      real sa,sx(*)
      integer i,incx,m,mp1,n,nincx

      if( n.le.0 .or. incx.le.0 )return
      if(incx.eq.1)go to 20

c        code for increment not equal to 1

      nincx = n*incx
      do 10 i = 1,nincx,incx
        sx(i) = sa*sx(i)
   10 continue
      return

c        code for increment equal to 1


c        clean-up loop

   20 m = mod(n,5)
      if( m .eq. 0 ) go to 40
      do 30 i = 1,m
        sx(i) = sa*sx(i)
   30 continue
      if( n .lt. 5 ) return
   40 mp1 = m + 1
      do 50 i = mp1,n,5
        sx(i) = sa*sx(i)
        sx(i + 1) = sa*sx(i + 1)
        sx(i + 2) = sa*sx(i + 2)
        sx(i + 3) = sa*sx(i + 3)
        sx(i + 4) = sa*sx(i + 4)
   50 continue
      return
      end




      subroutine sgefa(a,lda,n,ipvt,info)
C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      integer lda,n,ipvt(1),info
      real a(lda,1)

c     sgefa factors a real matrix by gaussian elimination.

c     sgefa is usually called by sgeco, but it can be called
c     directly with a saving in time if  rcond  is not needed.
c     (time for sgeco) = (1 + 9/n)*(time for sgefa) .

c     on entry
```

```
c       a       real(lda, n)
c               the matrix to be factored.

c       lda     integer
c               the leading dimension of the array  a .

c       n       integer
c               the order of the matrix  a .

c     on return

c       a       an upper triangular matrix and the multipliers
c               which were used to obtain it.
c               the factorization can be written  a = l*u  where
c               l  is a product of permutation and unit lower
c               triangular matrices and  u  is upper triangular.

c       ipvt    integer(n)
c               an integer vector of pivot indices.

c       info    integer
c               = 0  normal value.
c               = k  if  u(k,k) .eq. 0.0 .  this is not an error
c                    condition for this subroutine, but it does
c                    indicate that sgesl or sgedi will divide by zero
c                    if called.  use  rcond  in sgeco for a reliable
c                    indication of singularity.

c     linpack. this version dated 08/14/78 .
c     cleve moler, university of new mexico, argonne national lab.

c     subroutines and functions

c     blas saxpy,sscal,isamax

c     internal variables

      real t
      integer isamax,j,k,kp1,l,nm1


c     gaussian elimination with partial pivoting

      info = 0
      nm1 = n - 1
      if (nm1 .lt. 1) go to 70
      do 60 k = 1, nm1
        kp1 = k + 1

c       find l = pivot index

        l = isamax(n-k+1,a(k,k),1) + k - 1
        ipvt(k) = l

c       zero pivot implies this column already triangularized

        if (a(l,k) .eq. 0.0e0) go to 40

c         interchange if necessary
```

```
            if (l .eq. k) go to 10
               t = a(l,k)
               a(l,k) = a(k,k)
               a(k,k) = t
   10       continue

c           compute multipliers

            t = -1.0e0/a(k,k)
            call sscal(n-k,t,a(k+1,k),1)

c           row elimination with column indexing

            do 30 j = kp1, n
               t = a(l,j)
               if (l .eq. k) go to 20
                  a(l,j) = a(k,j)
                  a(k,j) = t
   20          continue
               call saxpy(n-k,t,a(k+1,k),1,a(k+1,j),1)
   30       continue
         go to 50
   40    continue
            info = k
   50    continue
   60 continue
   70 continue
      ipvt(n) = n
      if (a(n,n) .eq. 0.0e0) info = n
      return
      end



      real function sdot(n,sx,incx,sy,incy)

c     forms the dot product of two vectors.
c     uses unrolled loops for increments equal to one.
c     jack dongarra, linpack, 3/11/78.
c     modified 12/3/93, array(1) declarations changed to array(*)

C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      real sx(*),sy(*),stemp
      integer i,incx,incy,ix,iy,m,mp1,n

      stemp = 0.0e0
      sdot = 0.0e0
      if(n.le.0)return
      if(incx.eq.1.and.incy.eq.1)go to 20

c        code for unequal increments or equal increments
c          not equal to 1

      ix = 1
      iy = 1
      if(incx.lt.0)ix = (-n+1)*incx + 1
      if(incy.lt.0)iy = (-n+1)*incy + 1
      do 10 i = 1,n
        stemp = stemp + sx(ix)*sy(iy)
```

```
        ix = ix + incx
        iy = iy + incy
   10 continue
      sdot = stemp
      return

c       code for both increments equal to 1


c       clean-up loop

   20 m = mod(n,5)
      if( m .eq. 0 ) go to 40
      do 30 i = 1,m
        stemp = stemp + sx(i)*sy(i)
   30 continue
      if( n .lt. 5 ) go to 60
   40 mp1 = m + 1
      do 50 i = mp1,n,5
        stemp = stemp + sx(i)*sy(i) + sx(i + 1)*sy(i + 1) +
     *   sx(i + 2)*sy(i + 2) + sx(i + 3)*sy(i + 3) + sx(i + 4)*sy(i + 4)
   50 continue
   60 sdot = stemp
      return
      end



      subroutine sgesl(a,lda,n,ipvt,b,job)
C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      integer lda,n,ipvt(1),job
      real a(lda,1),b(1)

c     sgesl solves the real system
c     a * x = b  or  trans(a) * x = b
c     using the factors computed by sgeco or sgefa.

c     on entry

c       a       real(lda, n)
c               the output from sgeco or sgefa.

c       lda     integer
c               the leading dimension of the array  a .

c       n       integer
c               the order of the matrix  a .

c       ipvt    integer(n)
c               the pivot vector from sgeco or sgefa.

c       b       real(n)
c               the right hand side vector.

c       job     integer
c               = 0         to solve  a*x = b ,
c               = nonzero   to solve  trans(a)*x = b  where
c                           trans(a)  is the transpose.

c     on return
```

```
c        b       the solution vector  x .

c     error condition

c       a division by zero will occur if the input factor contains a
c       zero on the diagonal.  technically this indicates singularity
c       but it is often caused by improper arguments or improper
c       setting of lda .  it will not occur if the subroutines are
c       called correctly and if sgeco has set rcond .gt. 0.0
c       or sgefa has set info .eq. 0 .

c     to compute  inverse(a) * c  where  c  is a matrix
c     with  p  columns
c          call sgeco(a,lda,n,ipvt,rcond,z)
c          if (rcond is too small) go to ...
c          do 10 j = 1, p
c             call sgesl(a,lda,n,ipvt,c(1,j),0)
c       10 continue

c     linpack. this version dated 08/14/78 .
c     cleve moler, university of new mexico, argonne national lab.

c     subroutines and functions

c     blas saxpy,sdot

c     internal variables

      real sdot,t
      integer k,kb,l,nm1

      nm1 = n - 1
      if (job .ne. 0) go to 50

c        job = 0 , solve  a * x = b
c        first solve  l*y = b

         if (nm1 .lt. 1) go to 30
         do 20 k = 1, nm1
            l = ipvt(k)
            t = b(l)
            if (l .eq. k) go to 10
               b(l) = b(k)
               b(k) = t
   10       continue
            call saxpy(n-k,t,a(k+1,k),1,b(k+1),1)
   20    continue
   30    continue

c        now solve  u*x = y

         do 40 kb = 1, n
            k = n + 1 - kb
            b(k) = b(k)/a(k,k)
            t = -b(k)
            call saxpy(k-1,t,a(1,k),1,b(1),1)
   40    continue
      go to 100
   50 continue
```

```
c       job = nonzero, solve  trans(a) * x = b
c       first solve  trans(u)*y = b

        do 60 k = 1, n
          t = sdot(k-1,a(1,k),1,b(1),1)
          b(k) = (b(k) - t)/a(k,k)
   60   continue

c       now solve trans(l)*x = y

        if (nm1 .lt. 1) go to 90
        do 80 kb = 1, nm1
          k = n - kb
          b(k) = b(k) + sdot(n-k,a(k+1,k),1,b(k+1),1)
          l = ipvt(k)
          if (l .eq. k) go to 70
            t = b(l)
            b(l) = b(k)
            b(k) = t
   70     continue
   80   continue
   90   continue
  100 continue
      return
      end



C-----------------SUBROTINA INTK1

      SUBROUTINE INTK1(N,X,XX,Z,ZZ)
C     DOUBLE QUADRATIC (SMOOTH) INTER-OR EXTRAPOLATION
C     INPUT     N =NUMBER OF X VALUES AND NUMBER OF FUNCTION VALUES
C     -----        N MUST BE GREATER THAN 2
C               X =ARRAY OF X-VALUES
C               XX=X-VALUE TO BE INTERPOLATED
C               Z =ARRAY OF FUNCTION VALUES
C     OUTPUT    ZZ=INTERPOLATED FUNCTION VALUE.
C     ------
C     REMARK..THE X-VALUES NEED NOT BE EQUIDISTANT BUT MUST
C             BE MONOTONICALLY INCREASING.

      INTEGER*4 I,IQ,IR,J,JJ,K,M,N
      REAL*4 FF,X,XX,Z,ZZ
      DIMENSION X(N),Z(N)

      I=1
      JJ=I
      ZZ=0.0
      DO 50 J=1,N
      FF=XX-X(J)
      IF (FF) 20,10,50
   10 ZZ=Z(J)
      RETURN
   20 IF(J.EQ.1.OR.J.EQ.2) GOTO 60
      IF(J.NE.N) JJ=2
      GOTO 56
   50 CONTINUE
      J=N
```

```
56   I=J-2
60   M=I+2
     IR=M
     IQ=I+1
     FF=0.0
     DO 80 K=I,M
     FF=FF+Z(K)*(XX-X(IQ))*(XX-X(IR))/((X(K)-X(IQ))*(X(K)-X(IR)))
     IQ=IR
     IR=K
80   CONTINUE
     IF(JJ.NE.1) GOTO 90
     ZZ=FF
     RETURN
90   IR=JJ+I
     ZZ=ZZ+FF*(XX-X(IR))/(X(I+1)-X(IR))
     IF(JJ.EQ.0) RETURN
     JJ=0
     I=J-1
     GOTO 60
     END
```

A.3 Program List of *OptiPropVisc*

```
================================================
      Optimização de Hélices Propulsores Marítimos
           incluindo os efeitos da viscosidade
================================================

      PROGRAM OPTIPROPVISC


C     DEFINICAO DAS VARIAVEIS DE ENTRADA
C     ++++++++++++++++++++++++++++++++++

C     IPV: PARAMETRO QUE PERMITE AO UTILIZADOR A OPCAO ENTRE
C       CARREGAMENTO LIGEIRO (1) OU CARREGAMENTO MODERADO (2)
C     N: NUMERO DE PONTOS INTERIORES SOBRE A PA
C     NB: NUMERO DE PAS
C     CT OU CP: COEFICIENTE ADIMENSIONAL DE IMPULSO OU POTENCIA
C     XH: RAIO DO CUBO ADIMENSIONAL
C     J0: COEFICIENTE DE AVANCO
C     NX: NUMERO DE PONTOS ONDE E DADA A FRACCAO DE ESTEIRA EFECTIVA
C     XW: RAIOS ADIMENSIONAIS PARA CADA UM DESSES PONTOS
C     WT1: FRACCAO DE ESTEIRA EFECTIVA NOS PONTOS XW
C     WM: FRACCAO DE ESTEIRA MEDIA EFECTIVA


C     DEFINICAO DAS VARIAVEIS DE SAIDA
C     ++++++++++++++++++++++++++++++++

C     X: COTA DOS PONTOS CONSIDERADOS SOBRE A PA
C     G(X): DISTRIBUICAO DE CIRCULACAO ADIMENSIONAL
C     UA(X): VELOCIDADE AXIAL INDUZIDA ADIMENSIONAL
C     UT(X): VELOCIDADE TANGENCIAL INDUZIDA ADIMENSIONAL
C     BETA(X): ANGULO DE PASSO HIDRODINAMICO
C     BETAIN(X): ANGULO DE PASSO HIDRODINAMICO INDUZIDO
C     BETAIV(X): ÂNGULO DE SAÍDA DOS VÓRTICES
C     VEL(X): VELOCIDADE RESULTANTE
C     PASSO(X): PASSO ADIMENSIONALIZADO PELO DIAMETRO
C     CP: COEFICIENTE ADIMENSIONAL DE POTENCIA
C     CT: COEFICIENTE ADIMENSIONAL DE IMPULSO
C     REND: RENDIMENTO (CT/CP)


C     DEFINICAO DAS VARIAVEIS DO PROGRAMA
C     +++++++++++++++++++++++++++++++++++

C     N0: NUMERO DE PONTOS INTERIORES MAXIMO
C     NP0: NUMERO PONTOS MAXIMO
C     N: NUMERO DE PONTOS INTERIORES
C     NP:NUMERO DE PONTOS
C     TETA: TABELA DE EQUIPARACAO ANGULAR PARA CADA PONTO
C     A: MATRIZ DOS FACTORES TRIGONOMETRICOS
C     X: TABELA DOS RAIOS ADIMENSIONAIS PARA UM N DADO
C     BETA: TABELA DOS ANGULOS DE PASSO HIDRODINAMICOS
C     BETAIN: TABELA DOS ANGULOS DE PASSO HIDRODINAMICOS INDUZIDOS
C           ANTIGOS
```

```
C    BETAIV: TABELA DOS ANGULOS DE SAIDA DOS VORTICES
C    LANDAIN: COEFICIENTE DE VELOCIDADE DE APROXIMACAO INDUZIDO
C    G: DISTRIBUICAO DE CIRCULACAO
C    UAMAT: MATRIZ DE INDUCAO AXIAL
C    UTMAT: MATRIZ DE INDUCAO TANGENCIAL
C    UA: TABELA DAS VELOCIDADES INDUZIDAS AXIAIS
C    UT: TABELA DAS VELOCIDADES INDUZIDAS TANGENCIAIS
C    B: VECTOR DE REAIS PARA RESOLUCAO DO SISTEMA
C    AM: MATRIZ DO SISTEMA
C    VEL: VELOCIDADE RESULTANTE
C    PASSO: PASSO ADIMENSIONALIZADO PELO DIAMETRO
C    XH: RAIO ADIMENSIONAL INTERIOR
C    EPSLANDA: PARAMETRO DE CONVERGENCIA PARA O LANDAIN
C    PI: CONSTANTE
C    ERRO: VARIAVEL DE ERRO
C    INFO: VARIAVEL DE INFORMACAO PARA RESOLUCAO DO SISTEMA
C    REND: RENDIMENTO
C    W: FACTOR DE SUB-RELAXACAO
C    WT: FRACCAO DE ESTEIRA EFECTIVA NOS RAIOS DE GLAUERT
C    ITNUM: ARMAZENA O NUMERO DE ITERACOES
C    BETAIVN_1: BETAIV PARA A N-1 ITERACÇÃO
C    BETAVCONV: (1) SE BETAV CONVERGE OU (0) CASO CONTRÁRIO
C    CHANGE: (1) PARA MUDAR O PASSO (0) CASO CONTRÁRIO
C    TOLERANCE: TOLERÂNCIA PARA A CONVERGÊNCIA DE G E BETAIV
C    LCONV: (1) SE L CONVERGE OU (0) CASO CONTRÁRIO
C   DTLC: DRAG-TO-LIFT COEFFICIENT


      INTEGER N0,NP0,N,NP,I,J,K,NB,IPV,INFO,NX
      INTEGER GHANGE,GCONV,BETAICONV,ASK,ITNUM,LCONV
      PARAMETER(N0=29,NP0=N0+2)
      INTEGER IPVT(N0)
      REAL A(NP0,N0,NP0),TETA(NP0),X(NP0),BETAIV(NP0),BETAIN_1(NP0)
      REAL BETAIN(NP0),LANDAIN,G(NP0)
      REAL BETA(NP0),UAMAT(NP0,N0),UTMAT(NP0,N0),UA(NP0),UT(NP0)
      REAL B(N0),AM(N0,N0),VEL(NP0),PASSO(NP0)
      REAL XH,EPSLANDA,CT,CP,J0,PI,ERRO,TOLERANCE
      REAL CP1,REND,W,XX(17),G1(17),BETA1(17)
      REAL BETAIN1(17),UT1(17),WT(NP0),XW(NP0),WM,WT1(NP0),XP(NP0)
      REAL L,L1,CT1,DTLC

      DATA EPSLANDA /0.01/
      DATA TOLERANCE /0.00001/

      OPEN(UNIT=5,FILE='OptiPropVisc.INP',STATUS='UNKNOWN')
      OPEN(UNIT=6,FILE='OptiPropVisc.OUT',STATUS='UNKNOWN')


      READ(5,*) IPV
      PRINT*,IPV
      READ(5,*) N
      NP=N+2
      PRINT*,N
      PRINT*,NP
      READ(5,*) NB
      PRINT*,NB
      READ(5,*) CT
      PRINT*,CT
      READ(5,*) XH
      PRINT*,XH
```

```
      READ(5,*) J0
      PRINT*,J0
      READ(5,*) NX
      PRINT*,NX
      READ(5,*) (XW(I),I=1,NX)
      PRINT*,(XW(I),I=1,NX)
      READ(5,*) (WT1(I),I=1,NX)
      PRINT*,(WT1(I),I=1,NX)
      READ(5,*) WM
      PRINT*,WM
      READ(5,*) DTLC
      PRINT*,DTLC


C-----ECO DOS DADOS DE ENTRADA

      WRITE(6,1)
 1    FORMAT(/,10X,'DADOS DE ENTRADA',/,10X,16(1H=))
      WRITE(6,2) IPV
 2    FORMAT(/,4X,'IPV =',I2,5X)
      WRITE(6,3) N,NB,XH,J0
 3    FORMAT(/,4X,'N =',I3,5X,'NB =',I2,5X,'XH =',F6.3,5X,'J0 =',F6.3)
      WRITE(6,5) CT
 5    FORMAT(/,4X,'CT =',F8.5)
      WRITE(6,6) (WT1(I),I=1,NX)
 6    FORMAT(/,4X,'WT = ',F6.4)
      WRITE(6,7) WM
 7    FORMAT(/,4X,'WM = ',F6.4)
      WRITE(6,8) TOLERANCE*100
 8    FORMAT(/,4X,'Tolerância = ',F7.5,' %')
      WRITE(6,9) DTLC
 9    FORMAT(/,4X,'Drag-to-Lift coefficient = ',F7.5)


C-----TESTE DE VALIDADE DOS PARAMETROS ------

      ERRO=0
      IF ((IPV.NE.1).AND.(IPV.NE.2)) THEN
        ERRO=1
      ENDIF
      IF (NB.LE.0) THEN
        ERRO=1
    ELSE
        IF (CT.LE.0) THEN
          ERRO=1
      ELSE
          IF (XH.LT.0) THEN
            ERRO=1
        ELSE
              IF (XH.GE.1) THEN
            ERRO=1
              ELSE
            IF (J0.LE.0) THEN
              ERRO=1
                ELSE
                  IF (N.LE.0) THEN
                ERRO=1
                  ENDIF
                ENDIF
              ENDIF
```

```
        ENDIF
      ENDIF
    ENDIF

   IF (ERRO.EQ.1) THEN
       PRINT *,'VALOR DOS PARAMETROS DE ENTRADA ERRADO'
     GOTO 999
   ENDIF


C-----CALCULO DE PI

      PI=4.*ATAN(1.)


C-----CALCULO DOS VALORES DE TETA E X PARA NP PONTOS

      DO 13 J=1,NP
      TETA(J)=FLOAT(J-1)*PI/FLOAT(N+1)
      X(J)=0.5*(1.+XH) - 0.5*(1.-XH)*COS(TETA(J))
 13   CONTINUE


C-----INTERPOLACAO DA FRACCAO DE ESTEIRA EFECTIVA PARA
C     RAIOS DE GLAUERT

      DO 14 I=1,NP
       CALL INTK1(NX,XW,X(I),WT1,WT(I))
 14   CONTINUE


C-----CALCULO DAS MATRIZES DOS FACTORES TRIGONOMETRICOS A

      CALL AMAT(N0,NP0,N,NP,TETA,A)


C-----CORRECCAO DA MATRIZ A PARA O CASO DE XH=0

      IF (XH.EQ.0) THEN
        DO 15 I=1,N
          A(1,I,1)=0.
 15     CONTINUE
      ENDIF


C-----CALCULO DE BETA E BETAIN
      IF (X(1).EQ.0.) THEN
        BETA(1)=PI/2
        DO 16 I=2,NP
          BETA(I)=ATAN(J0*(1.-WT(I))/(PI*X(I)*(1.-WM)))
 16     CONTINUE
      ELSE
        DO 17 I=1,NP
          BETA(I)=ATAN(J0*(1.-WT(I))/(PI*X(I)*(1.-WM)))
 17     CONTINUE
      ENDIF
      CALL BETA_INICT(NP0,N,NP,EPSLANDA,J0,CT,WT,WM,
     &                TETA,X,BETAIN,LANDAIN)
      IF (X(1).EQ.0.) THEN
        BETAIN(1)=PI/2.
```

```
      ENDIF


C-----CALCULO DE BETAIV PASSO DOS VORTICES

        IF (IPV.EQ.1) THEN
        DO 18 I=1,NP
          BETAIV(I)=BETA(I)
 18             CONTINUE
         ELSE
        DO 19 I=1,NP
          BETAIV(I)=BETAIN(I)
 19             CONTINUE
        ENDIF


        L=-1.2
        ITNUM=0
       CHANGE=0

C-----CALCULO DAS MATRIZES DE INDUCAO

 120  CALL MAT_IND(N0,NP0,N,NP,XH,A,BETAIV,NB,X,
     &              UAMAT,UTMAT,ERRO)
        IF (ERRO.EQ.1) THEN
          PRINT*,'ERRO: EXCESSO DE CAPACIDADE'
          GOTO 999
        ENDIF


C----CALCULO DE G(X)

C-------CONSTRUCAO DA MATRIZ DO SISTEMA LINEAR

 130    DO 150 I=1,N
             B(I)=(-1.)*(1.-WT(I+1))/(1.-WM)-L
     &                -DTLC/TAN(BETAIN(I+1))*(1.-WT(I+1))/(1.-WM)
     &                +DTLC*L*TAN(BETAIN(I+1))
        DO 140 J=1,N
                   AM(I,J)=UAMAT(I+1,J)
     &                           +DTLC/TAN(BETAIN(I+1))*UAMAT(I+1,J)
     &                           +UAMAT(J+1,I)*X(J+1)*SIN(TETA(J+1))
     &                        /(X(I+1)*SIN(TETA(I+1)))
     &             *(1.+DTLC/TAN(BETAIN(J+1)))
     &                           -UTMAT(I+1,J)*L*J0/(X(I+1)*PI)
     &                           +DTLC*L*TAN(BETAIN(I+1))*UTMAT(I+1,J)
     &             *J0/(X(I+1)*PI)
     &            -L*J0/(X(I+1)*PI)*(1.-DTLC*TAN(BETAIN(J+1)))
     &                            *UTMAT(J+1,I)*SIN(TETA(J+1))/SIN(TETA(I+1))
 140      CONTINUE
 150    CONTINUE


C-------RESOLUCAO DO SISTEMA

       ITNUM=ITNUM+1
     CALL SGEFA(AM,N0,N,IPVT,INFO)
       IF (INFO.NE.0) THEN
         PRINT*,'SISTEMA SEM RESOLUCAO'
         GOTO 999
```

```
      ENDIF
      CALL SGESL(AM,N0,N,IPVT,B,0)


C-------AFECTACAO DOS RESULTADOS
      DO 160 I=1,N
        G(I+1)=B(I)
 160   CONTINUE
      G(1)=0.
      G(NP)=0.

C-------CALCULO DAS VELOCIDADES INDUZIDAS
      DO 165 K=1,NP
        UA(K)=0.
        UT(K)=0.
        DO 164 I=1,N
          UA(K)=UA(K)+G(I+1)*UAMAT(K,I)
          UT(K)=UT(K)+G(I+1)*UTMAT(K,I)
 164     CONTINUE
 165   CONTINUE


C-------CALCULO DE CT1 E O CORRESPONDENTE
C       L1 PELAS FORMULAS DE RELAXACAO

        CT1=0
        DO 175 I=1,NP
              CT1=CT1+G(I)*(PI*X(I)/J0-UT(I))
     &         *(1-DTLC*TAN(BETAIN(I)))*SIN(TETA(I))
 175     CONTINUE
        CT1=CT1*NB*2*PI*(1-XH)/(N+1)
        L1=L


C-------FORMULA DE RELAXACAO PARA CT

        IF (CT.LE.0.5) THEN
          W=0.222222
        ELSE
          W=1.0
        ENDIF
        L=L*(1-W*(CT1/CT-1)/4.)

      DO 190 I=1,NP
        BETAIN_1(I)=BETAIN(I)
 190        CONTINUE


C-----CALCULO DE BETA

 210  IF (X(1).EQ.0.) THEN
      BETA(1)=PI/2
      DO 214 I=2,NP
        BETA(I)=ATAN(J0*(1.-WT(I))/(PI*X(I)*(1.-WM)))
 214   CONTINUE
      ELSE
        DO 216 I=1,NP
          BETA(I)=ATAN(J0*(1.-WT(I))/(PI*X(I)*(1.-WM)))
 216   CONTINUE
      ENDIF
```

```
C-----CALCULO DE BETAIN

      IF (X(1).EQ.0.) THEN
        BETAIN(1)=PI/2
        DO 218 I=2,NP
          BETAIN(I)=ATAN(((1.-WT(I))/(1.-WM)+UA(I))/
     &                (X(I)*PI/J0-UT(I)))
  218   CONTINUE
      ELSE
        DO 220 I=1,NP
          BETAIN(I)=ATAN(((1.-WT(I))/(1.-WM)+UA(I))/
     &                (X(I)*PI/J0-UT(I)))
  220   CONTINUE
      ENDIF
C
C
C-------TESTE DE CONVERGENCIA EM BETAIN
C
        BETAICONV=1
        DO 221 I=1,NP
          IF ((ABS(BETAIN(I)-BETAIN_1(I))/BETAIN(I)).GT.TOLERANCE)
     &       THEN
             BETAICONV=0
          ENDIF
  221     CONTINUE
C
C
C-------TESTE DE CONVERGENCIA EM L
C
   IF (ABS((ABS(L1-L)/L1)).GT.TOLERANCE) THEN
         LCONV=0
   ELSE
      LCONV=1
   ENDIF

   IF (LCONV.EQ.0) THEN
       CHANGE=0
     ELSE
       CHANGE=1
   ENDIF

C-----CALCULO DE BETAIV PASSO DOS VORTICES

   IF (IPV.EQ.1) THEN
     DO 222 I=1,NP
       BETAIV(I)=BETA(I)
  222           CONTINUE
   ELSE
      IF (CHANGE.EQ.1) THEN
        DO 223 I=1,NP
          BETAIV(I)=BETAIN(I)
  223             CONTINUE
        ENDIF
     ENDIF


C-----ROTINA DE CONVERGENCIA

   IF (LCONV.EQ.0) THEN
```

```
          GOTO 130
      ELSE
        IF (BETAICONV.EQ.0) THEN
            GOTO 120
        ENDIF
      ENDIF


C************BLOCO DE OUTPUT*******************

C-----CALCULO DO MODULO DA VELOCIDADE TOTAL

 225    DO 230 I=1,NP
        VEL(I)=(UA(I)+(1.-WT(I))/(1-WM))/(SIN(BETAIN(I)))
 230    CONTINUE

C-----CALCULO DO COEFICIENTE DE POTENCIA

       CP=0.
       DO 240 I=1,NP
         CP=CP+G(I)*((1.-WT(I))/(1.-WM)+UA(I))*X(I)*SIN(TETA(I))
 240    CONTINUE
       CP=CP*NB*2.*(PI**2)*(1.-XH)/(J0*(N+1.))


C-----CALCULO DE RENDIMENTO

      REND=CT/CP

C-----CALCULO DO PASSO

      IF (X(1).EQ.0.) THEN
        DO 260 I=2,NP
          PASSO(I)=PI*X(I)*TAN(BETAIN(I))
 260    CONTINUE
      ELSE
        DO 261 I=1,NP
          PASSO(I)=PI*X(I)*TAN(BETAIN(I))
 261    CONTINUE
      ENDIF

C-----APRESENTACAO DOS ANGULOS EM GRAUS

      DO 262 I=1,NP
        BETA(I)=BETA(I)*180./PI
 262  CONTINUE

      DO 264 I=1,NP
        BETAIN(I)=BETAIN(I)*180./PI
 264  CONTINUE


C************FIM DO BLOCO DE OUTPUT*******************


C-----AFECTACAO DOS RESULTADOS

 265  FORMAT(/,F9.5)

      WRITE(6,275)
```

```
275  FORMAT(//,10X,'RESULTADOS DO PROGRAMA OptiPropVisc',/,10X,35(1H=))

     WRITE(6,265)

C-----PRIMEIRA TABELA DE RESULTADOS

     WRITE(6,285)
285  FORMAT(/,3X,'I',6X,'X',7X,'G',8X,'UA',7X,'UT',/)

     DO 290 I=1,NP
       WRITE(6,295) I,X(I),G(I),UA(I),UT(I)
290  CONTINUE
295  FORMAT(I4,4F9.5)


C-----SEGUNDA TABELA DE RESULTADOS

     WRITE(6,298)
298  FORMAT(//,3X,'I',6X,'B',7X,'Bi',8X,'V',7X,'P/D',/)
     DO 300 I=1,NP
       WRITE (6,302) I,BETA(I),BETAIN(I),VEL(I),PASSO(I)
300  CONTINUE
302  FORMAT(I4,2F9.2,2F9.5)

     WRITE(6,305) CP
305  FORMAT(//,10X,'COEFICIENTE DE POTENCIA: CP =',F7.4)

     WRITE(6,315) CT
315  FORMAT(//,10X,'COEFICIENTE DE IMPULSO: CT =',F7.4)

     WRITE(6,325) REND
325  FORMAT(//,10X,'RENDIMENTO =',F6.3)
     WRITE(6,*)
   WRITE(6,330) ITNUM
330  FORMAT(//,10X,'Numero de Iterações = ',I3,3X)



C-----INTERPOLACAO PARA RAIOS CONVENCIONAIS

     XX(1)=XH
     DO 350 I=2,17
       XX(I)=XX(I-1)+0.05
350  CONTINUE

     DO 360 I=1,17
       CALL INTK1(NP,X,XX(I),G,G1(I))
360  CONTINUE
     DO 370 I=1,17
       CALL INTK1(NP,X,XX(I),BETA,BETA1(I))
370  CONTINUE
     DO 380 I=1,17
       CALL INTK1(NP,X,XX(I),BETAIN,BETAIN1(I))
380  CONTINUE
     DO 390 I=1,17
       CALL INTK1(NP,X,XX(I),UT,UT1(I))
390  CONTINUE


C-----FECHO DOS FICHEIROS DE INP,OUT,DAT
```

```
C
      CLOSE(UNIT=5)
      CLOSE(UNIT=6)
C
 999  STOP
      END



C=====================================================================

C          ESCRITA DAS DIFERENTES SUBROTINAS
C          ++++++++++++++++++++++++++++++++

C------------------SUBROTINA AMAT

   SUBROUTINE AMAT(N0,NP0,N,NP,TETA,A)



C     ESTA SUBROTINA CALCULA A MATRIZ DOS FACTORES
C     TRIGONOMETRICOS PARA UM DETERMINADO NUMERO DE
C     PONTOS DADO


C  PARAMETROS DE ENTRADA

C        N0: NUMERO DE PONTOS INTERIORES MAXIMO
C        NP0: NUMERO DE PONTOS MAXIMO
C        N: NUMERO DE PONTOS INTERIORES
C        NP:NUMERO DE PONTOS
C        TETA: TABELA DE EQUIPARACAO ANGULAR PARA CADA PONTO


C  PARAMETROS DE SAIDA

C        A: MATRIZ DOS FACTORES TRIGONOMETRICOS



   INTEGER N0,NP0,N,NP,I,J,K,L
   REAL A(NP0,N0,NP0),TETA(NP0)
   REAL B1,B2,PI,H1,H2

   PI=4.*ATAN(1.)

C       MATRIZ A

      DO 50 I=1,N
       DO 40 J=1,NP
         DO 30 K=1,NP

       B1=0.5
       IF (((K.EQ.1).AND.(J.EQ.1)).OR.((K.EQ.NP).AND.(J.EQ.NP))) THEN
              B2=0
           DO 10 M=1,N+1
             IF (M.EQ.(N+1)) THEN
               H1=0.5
             ELSE
```

```
            H1=1
          ENDIF
          B2=B2+FLOAT(M)*H1*COS(FLOAT(M)*TETA(J))*
     &         COS(FLOAT(M)*TETA(K))/COS(TETA(K))
 10       CONTINUE
      ELSE
            H1 =FLOAT((J+K-2)/2)
            H2 = FLOAT(J+K-2)/2.
            IF (ABS(H1-H2).LT.0.01) THEN
          B2 = 0.
            ELSE
          B2 = -1. / (COS(TETA(K)) - COS(TETA(J)))
            ENDIF
      ENDIF


      A(K,I,J) = 0.

        DO 20 L=1,N

          IF (K.EQ.1.OR.K.EQ.NP) THEN
               H1 = FLOAT(L) * COS(FLOAT(L)*TETA(K)) / COS(TETA(K))
          ELSE
               H1 = SIN(FLOAT(L)*TETA(K)) / SIN(TETA(K))
          ENDIF

          B1 = B1 + COS(FLOAT(L)*TETA(J)) * COS(FLOAT(L)*TETA(K))
          B2 = B2 - COS(FLOAT(L)*TETA(J)) * H1

          A(K,I,J) = A(K,I,J) + FLOAT(L)*SIN(FLOAT(L)*TETA(I+1))
     &                    * (H1*B1 + COS(FLOAT(L)*TETA(K))*B2)
 20       CONTINUE

      A(K,I,J) = A(K,I,J) * 4.*PI/(FLOAT(N+1))**2

 30     CONTINUE
 40    CONTINUE
 50   CONTINUE


   RETURN
   END



C------------------SUBROTINA INFAC

      SUBROUTINE INFAC(X0,X1,BETAIV,NB,IA,IT,ERRO)



C     ESTA SUBROTINA CALCULA OS FACTORES DE INDUCAO
C     PARA O QUOCIENTE X0/X1


C  PARAMETROS DE ENTRADA

C      X0: RAIO ADIMENSIONAL PARA ORIGEM DO VORTICE
C      X1: RAIO ADIMENSIONAL PARA O PONTO DE CALCULO
C      BETAIV: ANGULO DE PASSO HIDRODINAMICO
```

```
C     NB: NUMERO DE PAS
C     ERRO: VARIAVEL DE ERRO


C  PARAMETROS DE SAIDA
C      IA: FACTOR DE INDUCAO AXIAL PARA K,L
C      IT: FACTOR DE INDUCAO TANGENCIAL PARA K,L



     REAL X0X,A,B,F,G,U,P,IA,IT,ERRO
     INTEGER NB
     REAL BETAIV,X0,X1
     DATA TOL /1E-12/


        IF (BETAIV.LE.0) THEN
          ERRO=1
        ENDIF


     IF (ERRO.EQ.1) THEN
       GOTO 40
   ENDIF

C---CALCULO DOS FACTORES DE INDUCAO

     IF ((X0.LT.TOL).AND.(X1.LT.TOL)) THEN
       IA=0
       IT=1
       GOTO 40
     ENDIF
     IF (X1.LT.TOL) THEN
       IA=NB/TAN(BETAIV)
       IT=0
       GOTO 40
     ENDIF
     X0X=X0/X1
     IF (X0X.LT.TOL) THEN
       IA=0
       IT=NB
       GOTO 40
     ENDIF
   IF ((1/(X0X)).EQ.1) THEN
       IA=COS(BETAIV)
       IT=SIN(BETAIV)
       GOTO 40
   ELSE
     IF (X0X.GT.1) THEN
         GOTO 20
    ELSE
      IF (X0X.LT.0.2) THEN
          GOTO 30
      ELSE
          IF(BETAIV.GE.(0.179))THEN
            GOTO 20
              ELSE
                IF (X0X.LT.0.8) THEN
              GOTO 30
                ELSE
```

```
            IF (BETAIV.GE.(0.012)) THEN
              GOTO 20
                ELSE
              GOTO 30
                ENDIF
               ENDIF
              ENDIF
     ENDIF
    ENDIF
   ENDIF



20   P=(1+((1/X0X)**2)/((TAN(BETAIV))**2))**0.5
     U=(((P-1)*X0X/(1/SIN(BETAIV)-1))**NB)*
    &  EXP(NB*(P-1/SIN(BETAIV)))

     IF (ABS(1-U).LE.1E-38) THEN
       ERRO=1
       PRINT *,'ERRO: EXCESSO DE CAPACIDADE'
       GOTO 40
     ENDIF

     G=((SIN(BETAIV))**3)*(2+9/((TAN(BETAIV))**2))+
    &  (3*P-5)/(P**3)
     F=1/(SQRT(P*SIN(BETAIV)))
     IF (X0X.GT.1) THEN
       B=F*(U/(1-U)+(1/(24*NB))*G*LOG(ABS(1/(1-U))))
       IA=(1-(1/(X0X)))*(NB*(1+B))/TAN(BETAIV)
       IT=(X0X-1)*NB*B
     ELSE
       IF (X0X.LT.1) THEN
         A=F*(1/(U-1)-(1/(24*NB))*G*LOG(ABS(U/(U-1))))
         IA=(1/X0X-1)*(NB*A)/TAN(BETAIV)
         IT=(1-X0X)*NB*(1+A)
       ENDIF
     ENDIF
     GOTO 40


30   IA=0
     IT=NB-NB*X0X



40   RETURN
     END



C--------------------------SUBROTINA MAT_IND

     SUBROUTINE MAT_IND(N0,NP0,N,NP,XH,A,BETAIV,NB,X,
    &                   UAMAT,UTMAT,ERRO)



C      ESTA SUBROUTINA CALCULA AS MATRIZES DE INDUCAO
C      UAMAT E UTMAT
```

```
C  PARAMETROS DE ENTRADA

C       N0: NUMERO DE PONTOS INTERIORES MAXIMO
C       NP0: NUMERO DE PONTOS MAXIMO
C       N: NUMERO DE PONTOS INTERIORES
C       NP: NUMERO DE PONTOS
C       X: TABELA DOS RAIOS ADIMENSIONAIS
C       XH: RAIO INTERIOR ADIMENSIONAL
C       BETAIV: ANGULO DE PASSO HIDRODINAMICO
C       NB: NUMERO DE PAS
C       A: MATRIZ DOS FACTORES TRIGONOMETRICOS


C  PARAMETROS DE SAIDA

C       UAMAT: MATRIZ AXIAL DE INDUCAO
C       UTMAT: MATRIZ TANGENCIAL DE INDUCAO
C       ERRO: VARIAVEL DE ERRO



  INTEGER N0,NP0,N,NP,I,J,K,NB
  REAL A(NP0,N0,NP0),BETAIV(NP0),X(NP0)
  REAL UAMAT(NP0,N0),UTMAT(NP0,N0)
  REAL XH,LAMBJ,IA,IT,ERRO

C---MATRIZ DOS FACTORES DE INDUCAO

     ERRO=0

     DO 30 I=1,N
      DO 20 K=1,NP
        UAMAT(K,I)=0.
        UTMAT(K,I)=0.
         DO 10 J=1,NP
           IF(J.EQ.1.OR.J.EQ.NP) THEN
             LAMBJ=0.5
           ELSE
             LAMBJ=1.0
           ENDIF

C---CALCULO DOS FACTORES DE INDUCAO

           CALL INFAC(X(J),X(K),BETAIV(J),NB,IA,IT,ERRO)

           IF (ERRO.EQ.1) THEN
             PRINT *,'ERRO: MUDAR OS PARAMETROS DE ENTRADA'
             GOTO 40
           ENDIF

           UAMAT(K,I) = UAMAT(K,I) + IA*LAMBJ*A(K,I,J)
           UTMAT(K,I) = UTMAT(K,I) + IT*LAMBJ*A(K,I,J)
 10        CONTINUE
        UAMAT(K,I) = UAMAT(K,I) / (1.-XH)
        UTMAT(K,I) = UTMAT(K,I) / (1.-XH)
 20    CONTINUE
 30   CONTINUE
```

```
 40   RETURN
      END



C----------------------SUBROTINA BETA_INICT

      SUBROUTINE BETA_INICT(NP0,N,NP,EPSLANDA,J0,CT,WT,WM,
     &                  TETA,X,BETAIN,LANDAIN)



C     ESTA SUBROTINA CALCULA OS VALORES DOS ANGULOS
C     DE PASSO HIDRODINAMICOS INICIAIS SENDO DADO O
C     COEFICIENTE DE IMPULSO


C  PARAMETROS DE ENTRADA

C     NP0:NUMERO DE PONTOS MAXIMO
C     N: NUMERO DE PONTOS INTERIORES
C     NP: NUMERO DE PONTOS
C     EPSLANDA: PARAMETRO DE CONVERGENCIA DE LANDA
C     J0: COEFICIENTE DE AVANCO
C     CT: COEFICIENTE DE IMPULSO
C     TETA: TABELA DE EQUIPARACAO ANGULAR
C     X: TABELA DOS RAIOS ADIMENSIONAIS
C     WT: FRACCAO DE ESTEIRA EFECTIVA PARA CADA PONTO
C     WM: FRACCAO DE ESTEIRA MEDIA EFECTIVA


C  PARAMETROS DE SAIDA

C     BETAIN: TABELA DOS ANGULOS DE PASSO HIDRODINAMICOS
C     LANDAIN: COEFICIENTE DE VELOCIDADE DE APROXIMACAO


      INTEGER NP0,N,NP,I
      REAL EPSLANDA,LANDA,CT,CTA,H
      REAL LANDAIN,LANDAIV,J0,WT(NP0),WM
      REAL BETAIN(NP0),X(NP0),TETA(NP0)


  PI= 4.*ATAN(1.)
  LANDA=0

     DO 10 I=1,NP
      LANDA= LANDA +((1-WT(I))*J0/(PI*(1-WM)))*SIN(TETA(I))
 10   CONTINUE

  LANDA= LANDA*PI/(2*(N+1))
     CTA= CT*((J0/(LANDA*PI))**2)
  LANDAIN=0.5*LANDA*(1+SQRT(1+CTA))

 20   LANDAIV=LANDAIN
  H=1+LANDAIV*(LANDAIV-LANDA)/(1+LANDAIV**2)
  H=H-LANDAIV*(2*LANDAIV-LANDA)*
    +ALOG((1+LANDAIV**2)/(LANDAIV**2))
  LANDAIN=0.5*LANDA*(1+SQRT(1+CTA/H))
```

```
      IF (ABS(LANDAIV-LANDAIN).GE.EPSLANDA)THEN
        GOTO 20
      ENDIF

C-----CASO DE XH SER IGUAL A ZERO OU DIFERENTE DE ZERO

      IF (X(1).EQ.0) THEN
        BETAIN(1)=PI/2
        DO 30 I=2,NP
          BETAIN(I)=ATAN(LANDAIN/X(I))
30      CONTINUE
      ELSE
        DO 40 I=1,NP
          BETAIN(I)=ATAN(LANDAIN/X(I))
40      CONTINUE
      ENDIF


      RETURN
      END



      integer function isamax(n,sx,incx)

c     finds the index of element having max. absolute value.
c     jack dongarra, linpack, 3/11/78.
c     modified 3/93 to return if incx .le. 0.
c     modified 12/3/93, array(1) declarations changed to array(*)

C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      real sx(*),smax
      integer i,incx,ix,n

      isamax = 0
      if( n.lt.1 .or. incx.le.0 ) return
      isamax = 1
      if(n.eq.1)return
      if(incx.eq.1)go to 20

c       code for increment not equal to 1

      ix = 1
      smax = abs(sx(1))
      ix = ix + incx
      do 10 i = 2,n
        if(abs(sx(ix)).le.smax) go to 5
        isamax = i
        smax = abs(sx(ix))
    5   ix = ix + incx
   10 continue
      return

c       code for increment equal to 1

   20 smax = abs(sx(1))
      do 30 i = 2,n
        if(abs(sx(i)).le.smax) go to 30
        isamax = i
        smax = abs(sx(i))
```

```
   30 continue
      return
      end



      subroutine saxpy(n,sa,sx,incx,sy,incy)

c     constant times a vector plus a vector.
c     uses unrolled loop for increments equal to one.
c     jack dongarra, linpack, 3/11/78.
c     modified 12/3/93, array(1) declarations changed to array(*)

C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      real sx(*),sy(*),sa
      integer i,incx,incy,ix,iy,m,mp1,n

      if(n.le.0)return
      if (sa .eq. 0.0) return
      if(incx.eq.1.and.incy.eq.1)go to 20

c       code for unequal increments or equal increments
c         not equal to 1

      ix = 1
      iy = 1
      if(incx.lt.0)ix = (-n+1)*incx + 1
      if(incy.lt.0)iy = (-n+1)*incy + 1
      do 10 i = 1,n
        sy(iy) = sy(iy) + sa*sx(ix)
        ix = ix + incx
        iy = iy + incy
   10 continue
      return

c       code for both increments equal to 1


c       clean-up loop

   20 m = mod(n,4)
      if( m .eq. 0 ) go to 40
      do 30 i = 1,m
        sy(i) = sy(i) + sa*sx(i)
   30 continue
      if( n .lt. 4 ) return
   40 mp1 = m + 1
      do 50 i = mp1,n,4
        sy(i) = sy(i) + sa*sx(i)
        sy(i + 1) = sy(i + 1) + sa*sx(i + 1)
        sy(i + 2) = sy(i + 2) + sa*sx(i + 2)
        sy(i + 3) = sy(i + 3) + sa*sx(i + 3)
   50 continue
      return
      end



      subroutine sscal(n,sa,sx,incx)
```

```
c     scales a vector by a constant.
c     uses unrolled loops for increment equal to 1.
c     jack dongarra, linpack, 3/11/78.
c     modified 3/93 to return if incx .le. 0.
c     modified 12/3/93, array(1) declarations changed to array(*)

C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      real sa,sx(*)
      integer i,incx,m,mp1,n,nincx

      if( n.le.0 .or. incx.le.0 )return
      if(incx.eq.1)go to 20

c        code for increment not equal to 1

      nincx = n*incx
      do 10 i = 1,nincx,incx
        sx(i) = sa*sx(i)
   10 continue
      return

c        code for increment equal to 1


c        clean-up loop

   20 m = mod(n,5)
      if( m .eq. 0 ) go to 40
      do 30 i = 1,m
        sx(i) = sa*sx(i)
   30 continue
      if( n .lt. 5 ) return
   40 mp1 = m + 1
      do 50 i = mp1,n,5
        sx(i) = sa*sx(i)
        sx(i + 1) = sa*sx(i + 1)
        sx(i + 2) = sa*sx(i + 2)
        sx(i + 3) = sa*sx(i + 3)
        sx(i + 4) = sa*sx(i + 4)
   50 continue
      return
      end



      subroutine sgefa(a,lda,n,ipvt,info)
C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      integer lda,n,ipvt(1),info
      real a(lda,1)

c     sgefa factors a real matrix by gaussian elimination.

c     sgefa is usually called by sgeco, but it can be called
c     directly with a saving in time if  rcond  is not needed.
c     (time for sgeco) = (1 + 9/n)*(time for sgefa) .

c     on entry

c        a       real(lda, n)
c                the matrix to be factored.
```

```
c       lda     integer
c               the leading dimension of the array  a .

c       n       integer
c               the order of the matrix  a .

c     on return

c       a       an upper triangular matrix and the multipliers
c               which were used to obtain it.
c               the factorization can be written  a = l*u  where
c               l  is a product of permutation and unit lower
c               triangular matrices and  u  is upper triangular.

c       ipvt    integer(n)
c               an integer vector of pivot indices.

c       info    integer
c               = 0  normal value.
c               = k  if  u(k,k) .eq. 0.0 .  this is not an error
c                    condition for this subroutine, but it does
c                    indicate that sgesl or sgedi will divide by zero
c                    if called.  use  rcond  in sgeco for a reliable
c                    indication of singularity.

c     linpack. this version dated 08/14/78 .
c     cleve moler, university of new mexico, argonne national lab.

c     subroutines and functions

c     blas saxpy,sscal,isamax

c     internal variables

      real t
      integer isamax,j,k,kp1,l,nm1


c     gaussian elimination with partial pivoting

      info = 0
      nm1 = n - 1
      if (nm1 .lt. 1) go to 70
      do 60 k = 1, nm1
        kp1 = k + 1

c       find l = pivot index

        l = isamax(n-k+1,a(k,k),1) + k - 1
        ipvt(k) = l

c       zero pivot implies this column already triangularized

        if (a(l,k) .eq. 0.0e0) go to 40

c         interchange if necessary

          if (l .eq. k) go to 10
            t = a(l,k)
```

```
            a(l,k) = a(k,k)
            a(k,k) = t
   10       continue

c           compute multipliers

           t = -1.0e0/a(k,k)
           call sscal(n-k,t,a(k+1,k),1)

c           row elimination with column indexing

           do 30 j = kp1, n
              t = a(l,j)
              if (l .eq. k) go to 20
                 a(l,j) = a(k,j)
                 a(k,j) = t
   20         continue
              call saxpy(n-k,t,a(k+1,k),1,a(k+1,j),1)
   30       continue
         go to 50
   40    continue
          info = k
   50    continue
   60 continue
   70 continue
     ipvt(n) = n
     if (a(n,n) .eq. 0.0e0) info = n
     return
     end



      real function sdot(n,sx,incx,sy,incy)

c     forms the dot product of two vectors.
c     uses unrolled loops for increments equal to one.
c     jack dongarra, linpack, 3/11/78.
c     modified 12/3/93, array(1) declarations changed to array(*)

C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      real sx(*),sy(*),stemp
      integer i,incx,incy,ix,iy,m,mp1,n

      stemp = 0.0e0
      sdot = 0.0e0
      if(n.le.0)return
      if(incx.eq.1.and.incy.eq.1)go to 20

c       code for unequal increments or equal increments
c         not equal to 1

      ix = 1
      iy = 1
      if(incx.lt.0)ix = (-n+1)*incx + 1
      if(incy.lt.0)iy = (-n+1)*incy + 1
      do 10 i = 1,n
        stemp = stemp + sx(ix)*sy(iy)
        ix = ix + incx
        iy = iy + incy
   10 continue
```

```
      sdot = stemp
      return

c       code for both increments equal to 1


c       clean-up loop

   20 m = mod(n,5)
      if( m .eq. 0 ) go to 40
      do 30 i = 1,m
        stemp = stemp + sx(i)*sy(i)
   30 continue
      if( n .lt. 5 ) go to 60
   40 mp1 = m + 1
      do 50 i = mp1,n,5
        stemp = stemp + sx(i)*sy(i) + sx(i + 1)*sy(i + 1) +
     *   sx(i + 2)*sy(i + 2) + sx(i + 3)*sy(i + 3) + sx(i + 4)*sy(i + 4)
   50 continue
   60 sdot = stemp
      return
      end



      subroutine sgesl(a,lda,n,ipvt,b,job)
C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      integer lda,n,ipvt(1),job
      real a(lda,1),b(1)

c     sgesl solves the real system
c     a * x = b  or  trans(a) * x = b
c     using the factors computed by sgeco or sgefa.

c     on entry

c       a       real(lda, n)
c               the output from sgeco or sgefa.

c       lda     integer
c               the leading dimension of the array  a .

c       n       integer
c               the order of the matrix  a .

c       ipvt    integer(n)
c               the pivot vector from sgeco or sgefa.

c       b       real(n)
c               the right hand side vector.

c       job     integer
c               = 0         to solve  a*x = b ,
c               = nonzero   to solve  trans(a)*x = b  where
c                           trans(a)  is the transpose.

c     on return

c       b       the solution vector  x .
```

```
c     error condition

c       a division by zero will occur if the input factor contains a
c       zero on the diagonal.  technically this indicates singularity
c       but it is often caused by improper arguments or improper
c       setting of lda .  it will not occur if the subroutines are
c       called correctly and if sgeco has set rcond .gt. 0.0
c       or sgefa has set info .eq. 0 .

c     to compute  inverse(a) * c  where  c  is a matrix
c     with  p  columns
c          call sgeco(a,lda,n,ipvt,rcond,z)
c          if (rcond is too small) go to ...
c          do 10 j = 1, p
c             call sgesl(a,lda,n,ipvt,c(1,j),0)
c       10 continue

c     linpack. this version dated 08/14/78 .
c     cleve moler, university of new mexico, argonne national lab.

c     subroutines and functions

c     blas saxpy,sdot

c     internal variables

      real sdot,t
      integer k,kb,l,nm1

      nm1 = n - 1
      if (job .ne. 0) go to 50

c       job = 0 , solve  a * x = b
c       first solve  l*y = b

        if (nm1 .lt. 1) go to 30
        do 20 k = 1, nm1
          l = ipvt(k)
          t = b(l)
          if (l .eq. k) go to 10
             b(l) = b(k)
             b(k) = t
   10       continue
          call saxpy(n-k,t,a(k+1,k),1,b(k+1),1)
   20   continue
   30   continue

c       now solve  u*x = y

        do 40 kb = 1, n
          k = n + 1 - kb
          b(k) = b(k)/a(k,k)
          t = -b(k)
          call saxpy(k-1,t,a(1,k),1,b(1),1)
   40   continue
      go to 100
   50 continue

c       job = nonzero, solve  trans(a) * x = b
c       first solve  trans(u)*y = b
```

```
      do 60 k = 1, n
         t = sdot(k-1,a(1,k),1,b(1),1)
         b(k) = (b(k) - t)/a(k,k)
   60 continue

c      now solve trans(l)*x = y

      if (nm1 .lt. 1) go to 90
      do 80 kb = 1, nm1
         k = n - kb
         b(k) = b(k) + sdot(n-k,a(k+1,k),1,b(k+1),1)
         l = ipvt(k)
         if (l .eq. k) go to 70
            t = b(l)
            b(l) = b(k)
            b(k) = t
   70    continue
   80 continue
   90 continue
  100 continue
      return
      end



C------------------SUBROTINA INTK1

      SUBROUTINE INTK1(N,X,XX,Z,ZZ)
C     DOUBLE QUADRATIC (SMOOTH) INTER-OR EXTRAPOLATION
C     INPUT     N =NUMBER OF X VALUES AND NUMBER OF FUNCTION VALUES
C     -----        N MUST BE GREATER THAN 2
C               X =ARRAY OF X-VALUES
C               XX=X-VALUE TO BE INTERPOLATED
C               Z =ARRAY OF FUNCTION VALUES
C     OUTPUT   ZZ=INTERPOLATED FUNCTION VALUE.
C     ------
C     REMARK..THE X-VALUES NEED NOT BE EQUIDISTANT BUT MUST
C             BE MONOTONICALLY INCREASING.

      INTEGER*4 I,IQ,IR,J,JJ,K,M,N
      REAL*4 FF,X,XX,Z,ZZ
      DIMENSION X(N),Z(N)

      I=1
      JJ=I
      ZZ=0.0
      DO 50 J=1,N
      FF=XX-X(J)
      IF (FF) 20,10,50
   10   ZZ=Z(J)
      RETURN
   20   IF(J.EQ.1.OR.J.EQ.2) GOTO 60
      IF(J.NE.N) JJ=2
      GOTO 56
   50   CONTINUE
      J=N
   56   I=J-2
   60   M=I+2
      IR=M
```

```
      IQ=I+1
      FF=0.0
      DO 80 K=I,M
      FF=FF+Z(K)*(XX-X(IQ))*(XX-X(IR))/((X(K)-X(IQ))*(X(K)-X(IR)))
      IQ=IR
      IR=K
   80 CONTINUE
      IF(JJ.NE.1) GOTO 90
      ZZ=FF
      RETURN
   90 IR=JJ+I
      ZZ=ZZ+FF*(XX-X(IR))/(X(I+1)-X(IR))
      IF(JJ.EQ.0) RETURN
      JJ=0
      I=J-1
      GOTO 60
      END
```

B. Developing equation (21)

The equation **(21)**:

$$\frac{U(\theta_i)}{V_0} x_i \sin\theta_i + \sum_{j=1}^{N} G(\theta_j) M_a^*(\theta_i,\theta_j) + \ell\lambda\left[\frac{\omega R x_i}{V_0}\sin\theta_i - \sum_{j=1}^{N} G(\theta_j) M_t^*(\theta_i,\theta_j)\right] = 0$$

where:

$$M_a^*(\theta_i,\theta_j) = M_a(\theta_i,\theta_j) x_i \sin\theta_i + M_a(\theta_j,\theta_i) x_j \sin\theta_j$$

$$M_t^*(\theta_i,\theta_j) = M_t(\theta_i,\theta_j) \sin\theta_i + M_t(\theta_j,\theta_i) \sin\theta_j$$

With respect to equation **(18)**, the formulation presented in **(21)** can be written as:

$$\frac{U(\theta_i)}{V_0} x_i \sin\theta_i + \frac{u_a(\theta_i)}{V_0} x_i \sin\theta_i + \sum_{j=1}^{N} G(\theta_j) M_a(\theta_j,\theta_i) x_j \sin\theta_j =$$
$$= -\ell\lambda\left[\frac{x_i}{\lambda}\sin\theta_i - \frac{u_t(\theta_i)}{V_0}\sin\theta_i - \sum_{j=1}^{N} G(\theta_j) M_t(\theta_j,\theta_i)\sin\theta_i\right]$$

Isolating the product of the Lagrange multiplier and the advance velocity coefficient:

$$\frac{\frac{U(\theta_i)}{V_0} x_i \sin\theta_i + \frac{u_a(\theta_i)}{V_0} x_i \sin\theta_i + \sum_{j=1}^{N} G(\theta_j) M_a(\theta_j,\theta_i) x_j \sin\theta_j}{\frac{x_i}{\lambda}\sin\theta_i - \frac{u_t(\theta_i)}{V_0}\sin\theta_i - \sum_{j=1}^{N} G(\theta_j) M_t(\theta_j,\theta_i)\sin\theta_j} = -\ell\lambda$$

and diving by $\sin\theta_i$:

$$\frac{\frac{U(\theta_i)}{V_0} x_i + \frac{u_a(\theta_i)}{V_0} x_i + \frac{1}{\sin\theta_i}\sum_{j=1}^{N} G(\theta_j) M_a(\theta_j,\theta_i) x_j \sin\theta_j}{\frac{x_i}{\lambda} - \frac{u_t(\theta_i)}{V_0} - \frac{1}{\sin\theta_i}\sum_{j=1}^{N} G(\theta_j) M_t(\theta_j,\theta_i)\sin\theta_j} = -\ell\lambda$$

Rewriting in the form of equation **(27)**:

$$\frac{\frac{U(\theta_i)}{V_0}+\frac{u_a(\theta_i)}{V_0}+\frac{1}{x_i\sin\theta_i}\sum_{j=1}^{N}G(\theta_j)M_a(\theta_j,\theta_i)x_j\sin\theta_j}{\frac{x_i}{\lambda}-\frac{u_t(\theta_i)}{V_0}-\frac{1}{\sin\theta_i}\sum_{j=1}^{N}G(\theta_j)M_t(\theta_j,\theta_i)\sin\theta_j}=-\ell\frac{\lambda}{x_i}$$

C. Tables of results obtained from *OptiPropII*

C.1 Distribution of circulation

C.1.1 Light loading, uniform flow, x_h = 0.2, Z=5, C_T=0.2, J_0=0.6 and N=23.

G	Tolerance						
x	0.5%	0.1%	0.05%	0.01%	0.005%	0.001%	0.0005%
0,2	0	0	0	0	0	0	0
0,20342	0,00074	0,00070	0,00070	0,00070	0,00070	0,00070	0,00070
0,21363	0,00146	0,00138	0,00137	0,00137	0,00137	0,00137	0,00137
0,23045	0,00213	0,00201	0,00200	0,00199	0,00199	0,00199	0,00199
0,25359	0,00272	0,00258	0,00256	0,00255	0,00255	0,00255	0,00255
0,28266	0,00324	0,00306	0,00305	0,00303	0,00303	0,00303	0,00303
0,31716	0,00366	0,00346	0,00345	0,00343	0,00343	0,00343	0,00343
0,35650	0,00401	0,00379	0,00377	0,00376	0,00375	0,00375	0,00375
0,4	0,00429	0,00405	0,00403	0,00401	0,00401	0,00401	0,00401
0,44693	0,00450	0,00426	0,00424	0,00422	0,00421	0,00421	0,00421
0,49647	0,00467	0,00442	0,00440	0,00438	0,00437	0,00437	0,00437
0,54779	0,00481	0,00454	0,00452	0,00450	0,00449	0,00449	0,00449
0,6	0,00490	0,00463	0,00461	0,00459	0,00459	0,00458	0,00458
0,65221	0,00497	0,00470	0,00467	0,00465	0,00465	0,00464	0,00464
0,70353	0,00500	0,00472	0,00470	0,00468	0,00467	0,00467	0,00467
0,75307	0,00499	0,00471	0,00469	0,00467	0,00466	0,00466	0,00466
0,8	0,00491	0,00464	0,00461	0,00459	0,00459	0,00459	0,00459
0,84350	0,00475	0,00448	0,00446	0,00444	0,00444	0,00443	0,00443
0,88284	0,00447	0,00422	0,00420	0,00418	0,00417	0,00417	0,00417
0,91734	0,00405	0,00382	0,00380	0,00379	0,00378	0,00378	0,00378
0,94641	0,00348	0,00328	0,00327	0,00325	0,00325	0,00325	0,00325
0,96955	0,00276	0,00260	0,00259	0,00258	0,00258	0,00258	0,00258
0,98637	0,00191	0,00181	0,00180	0,00179	0,00179	0,00179	0,00179
0,99658	0,00098	0,00093	0,00092	0,00092	0,00092	0,00092	0,00092
1	0	0	0	0	0	0	0

C.1.2 Moderate loading, uniform flow, x_h = 0.2, Z=5, C_T=0.2, J_0=0.6 and N=23

G	Tolerance							
x	0.5%	0.1%	0.05%	0.01%	0.005%	0.001%	0.0005%	0.0001%
0.2	0	0	0	0	0	0	0	0
0.20342	0.00073	0.00069	0.00069	0.00069	0.00069	0.00069	0.00069	0.00069
0.21363	0.00144	0.00136	0.00135	0.00135	0.00135	0.00135	0.00135	0.00135
0.23045	0.00210	0.00199	0.00198	0.00197	0.00197	0.00197	0.00197	0.00197
0.25359	0.00270	0.00255	0.00254	0.00253	0.00253	0.00252	0.00252	0.00252
0.28266	0.00322	0.00304	0.00302	0.00301	0.00301	0.00301	0.00301	0.00301
0.31716	0.00365	0.00345	0.00343	0.00342	0.00341	0.00341	0.00341	0.00341
0.35650	0.00401	0.00378	0.00376	0.00375	0.00375	0.00374	0.00374	0.00374
0.4	0.00429	0.00405	0.00403	0.00402	0.00401	0.00401	0.00401	0.00401
0.44693	0.00452	0.00426	0.00424	0.00423	0.00422	0.00422	0.00422	0.00422
0.49647	0.00470	0.00443	0.00441	0.00439	0.00439	0.00439	0.00439	0.00439
0.54779	0.00483	0.00456	0.00454	0.00452	0.00452	0.00451	0.00451	0.00451
0.6	0.00494	0.00465	0.00463	0.00461	0.00461	0.00461	0.00461	0.00461
0.65221	0.00500	0.00472	0.00469	0.00468	0.00467	0.00467	0.00467	0.00467
0.70353	0.00503	0.00474	0.00472	0.00470	0.00470	0.00470	0.00470	0.00469
0.75307	0.00501	0.00472	0.00470	0.00468	0.00468	0.00468	0.00468	0.00467
0.8	0.00492	0.00464	0.00462	0.00460	0.00460	0.00459	0.00459	0.00459
0.84350	0.00475	0.00447	0.00445	0.00443	0.00443	0.00443	0.00443	0.00443
0.88284	0.00445	0.00420	0.00418	0.00416	0.00416	0.00415	0.00415	0.00415
0.91734	0.00402	0.00379	0.00377	0.00376	0.00375	0.00375	0.00375	0.00375
0.94641	0.00344	0.00325	0.00323	0.00322	0.00321	0.00321	0.00321	0.00321
0.96955	0.00272	0.00257	0.00256	0.00255	0.00254	0.00254	0.00254	0.00254
0.98637	0.00189	0.00178	0.00177	0.00176	0.00176	0.00176	0.00176	0.00176
0.99658	0.00097	0.00091	0.00091	0.00090	0.00090	0.00090	0.00090	0.00090
1	0	0	0	0	0	0	0	0

C.2 Axial induced velocity

C.2.1 Light loading, uniform flow, x_h = 0.2, Z=5, C_T=0.2, J_0=0.6 and N=23

$u_a(\theta_i)$	Tolerance						
x	0.5%	0.1%	0.05%	0.01%	0.005%	0.001%	0.0005%
0,2	0,03729	0,03535	0,03518	0,03504	0,03500	0,03499	0,03499
0,20342	0,03782	0,03585	0,03568	0,03554	0,03550	0,03548	0,03548
0,21363	0,03938	0,03732	0,03714	0,03700	0,03695	0,03694	0,03694
0,23045	0,04186	0,03965	0,03946	0,03930	0,03926	0,03924	0,03924
0,25359	0,04497	0,04259	0,04238	0,04221	0,04216	0,04215	0,04214
0,28266	0,04840	0,04581	0,04558	0,04540	0,04535	0,04533	0,04533
0,31716	0,05180	0,04901	0,04877	0,04857	0,04851	0,04850	0,04849
0,35650	0,05496	0,05198	0,05172	0,05151	0,05145	0,05143	0,05142
0,4	0,05773	0,05458	0,05431	0,05409	0,05402	0,05400	0,05400
0,44693	0,06008	0,05678	0,05650	0,05627	0,05620	0,05618	0,05618
0,49647	0,06201	0,05860	0,05831	0,05807	0,05800	0,05798	0,05797
0,54779	0,06358	0,06008	0,05978	0,05953	0,05945	0,05943	0,05943
0,6	0,06484	0,06126	0,06095	0,06070	0,06062	0,06060	0,06060
0,65221	0,06584	0,06220	0,06189	0,06163	0,06155	0,06153	0,06152
0,70353	0,06663	0,06294	0,06262	0,06236	0,06228	0,06226	0,06225
0,75307	0,06724	0,06351	0,06319	0,06292	0,06284	0,06282	0,06282
0,8	0,06769	0,06393	0,06361	0,06334	0,06326	0,06324	0,06324
0,84350	0,06800	0,06422	0,06390	0,06363	0,06355	0,06353	0,06352
0,88284	0,06818	0,06439	0,06406	0,06379	0,06371	0,06369	0,06369
0,91734	0,06824	0,06444	0,06411	0,06384	0,06377	0,06374	0,06374
0,94641	0,06819	0,06439	0,06407	0,06380	0,06372	0,06369	0,06369
0,96955	0,06805	0,06426	0,06394	0,06367	0,06359	0,06357	0,06356
0,98637	0,06788	0,06410	0,06378	0,06351	0,06343	0,06341	0,06341
0,99658	0,06773	0,06396	0,06364	0,06337	0,06329	0,06327	0,06326
1	0,06766	0,06389	0,06357	0,06330	0,06322	0,06320	0,06319

C.2.2 Moderate loading, uniform flow, x_h = 0.2, Z=5, C_T=0.2, J_0=0.6 and N=23

$u_a(\theta_i)$	Tolerance							
x	0.5%	0.1%	0.05%	0.01%	0.005%	0.001%	0.0005%	0.0001%
0.2	0.03514	0.03326	0.03311	0.03299	0.03297	0.03295	0.03294	0.03294
0.20342	0.03566	0.03374	0.03359	0.03346	0.03344	0.03342	0.03342	0.03342
0.21363	0.03717	0.03516	0.03500	0.03487	0.03485	0.03483	0.03482	0.03482
0.23045	0.03955	0.03740	0.03723	0.03710	0.03707	0.03705	0.03705	0.03705
0.25359	0.04257	0.04024	0.04005	0.03990	0.03988	0.03985	0.03985	0.03985
0.28266	0.04589	0.04336	0.04316	0.04300	0.04297	0.04294	0.04294	0.04293
0.31716	0.04921	0.04648	0.04626	0.04608	0.04605	0.04602	0.04602	0.04602
0.35650	0.05230	0.04938	0.04914	0.04896	0.04892	0.04889	0.04889	0.04889
0.4	0.05503	0.05193	0.05169	0.05149	0.05146	0.05142	0.05142	0.05142
0.44693	0.05735	0.05411	0.05385	0.05364	0.05360	0.05357	0.05357	0.05356
0.49647	0.05926	0.05590	0.05564	0.05542	0.05538	0.05535	0.05534	0.05534
0.54779	0.06082	0.05736	0.05709	0.05687	0.05683	0.05679	0.05679	0.05679
0.6	0.06207	0.05854	0.05825	0.05803	0.05799	0.05795	0.05795	0.05795
0.65221	0.06307	0.05947	0.05918	0.05895	0.05891	0.05887	0.05887	0.05887
0.70353	0.06385	0.06020	0.05991	0.05968	0.05964	0.05960	0.05959	0.05959
0.75307	0.06445	0.06077	0.06047	0.06024	0.06020	0.06016	0.06015	0.06015
0.8	0.06490	0.06118	0.06089	0.06065	0.06061	0.06057	0.06057	0.06056
0.84350	0.06520	0.06147	0.06117	0.06093	0.06089	0.06085	0.06085	0.06084
0.88284	0.06538	0.06163	0.06133	0.06109	0.06105	0.06101	0.06101	0.06100
0.91734	0.06543	0.06168	0.06138	0.06114	0.06110	0.06106	0.06105	0.06105
0.94641	0.06538	0.06163	0.06133	0.06109	0.06105	0.06101	0.06101	0.06100
0.96955	0.06525	0.06150	0.06121	0.06097	0.06093	0.06089	0.06088	0.06088
0.98637	0.06509	0.06135	0.06106	0.06082	0.06078	0.06074	0.06073	0.06073
0.99658	0.06495	0.06122	0.06092	0.06068	0.06064	0.06060	0.06060	0.06060
1	0.06487	0.06115	0.06085	0.06061	0.06057	0.06053	0.06053	0.06052

C.3 Tangential induced velocity

C.3.1 Light loading, uniform flow, x_h = 0.2, Z=5, C_T=0.2, J_0=0.6 and N=23

$u_t(\theta_i)$	Tolerance						
x	0.5%	0.1%	0.05%	0.01%	0.005%	0.001%	0.0005%
0,2	0,03560	0,03375	0,03360	0,03346	0,03342	0,03341	0,03341
0,20342	0,03550	0,03366	0,03350	0,03336	0,03333	0,03331	0,03331
0,21363	0,03521	0,03336	0,03321	0,03307	0,03304	0,03302	0,03302
0,23045	0,03469	0,03286	0,03270	0,03257	0,03254	0,03252	0,03252
0,25359	0,03387	0,03207	0,03192	0,03179	0,03175	0,03174	0,03174
0,28266	0,03270	0,03095	0,03080	0,03068	0,03064	0,03063	0,03063
0,31716	0,03120	0,02951	0,02937	0,02925	0,02921	0,02920	0,02920
0,35650	0,02944	0,02785	0,02771	0,02759	0,02756	0,02755	0,02755
0,4	0,02756	0,02606	0,02593	0,02582	0,02579	0,02578	0,02578
0,44693	0,02567	0,02427	0,02415	0,02405	0,02402	0,02401	0,02401
0,49647	0,02386	0,02254	0,02243	0,02234	0,02231	0,02230	0,02230
0,54779	0,02217	0,02095	0,02084	0,02075	0,02073	0,02072	0,02072
0,6	0,02064	0,01950	0,01940	0,01932	0,01930	0,01929	0,01929
0,65221	0,01928	0,01821	0,01812	0,01805	0,01802	0,01802	0,01802
0,70353	0,01809	0,01709	0,01700	0,01693	0,01691	0,01690	0,01690
0,75307	0,01705	0,01611	0,01602	0,01596	0,01594	0,01593	0,01593
0,8	0,01616	0,01526	0,01519	0,01512	0,01510	0,01510	0,01510
0,84350	0,01540	0,01454	0,01447	0,01441	0,01439	0,01438	0,01438
0,88284	0,01475	0,01393	0,01386	0,01380	0,01378	0,01378	0,01378
0,91734	0,01421	0,01342	0,01335	0,01329	0,01328	0,01327	0,01327
0,94641	0,01376	0,01299	0,01293	0,01287	0,01286	0,01285	0,01285
0,96955	0,01340	0,01266	0,01259	0,01254	0,01253	0,01252	0,01252
0,98637	0,01314	0,01241	0,01235	0,01230	0,01228	0,01228	0,01228
0,99658	0,01298	0,01226	0,01220	0,01214	0,01213	0,01212	0,01212
1	0,01292	0,01220	0,01214	0,01209	0,01207	0,01207	0,01207

C.3.2 Moderate loading, uniform flow, x_h = 0.2, Z=5, C_T=0.2, J_0=0.6 and N=23

$u_t(\theta_i)$	Tolerance							
x	0.5%	0.1%	0.05%	0.01%	0.005%	0.001%	0.0005%	0.0001%
0.2	0.03535	0.03346	0.03330	0.03318	0.03316	0.03314	0.03314	0.03314
0.20342	0.03526	0.03337	0.03322	0.03310	0.03307	0.03305	0.03305	0.03305
0.21363	0.03500	0.03311	0.03296	0.03284	0.03282	0.03280	0.03280	0.03279
0.23045	0.03453	0.03265	0.03251	0.03239	0.03236	0.03234	0.03234	0.03234
0.25359	0.03377	0.03192	0.03178	0.03166	0.03164	0.03162	0.03162	0.03161
0.28266	0.03266	0.03086	0.03072	0.03060	0.03058	0.03056	0.03056	0.03056
0.31716	0.03122	0.02948	0.02934	0.02923	0.02921	0.02919	0.02919	0.02919
0.35650	0.02952	0.02786	0.02773	0.02763	0.02761	0.02759	0.02759	0.02759
0.4	0.02768	0.02612	0.02600	0.02590	0.02588	0.02586	0.02586	0.02586
0.44693	0.02581	0.02436	0.02424	0.02415	0.02413	0.02411	0.02411	0.02411
0.49647	0.02402	0.02265	0.02254	0.02246	0.02244	0.02243	0.02243	0.02243
0.54779	0.02234	0.02107	0.02097	0.02089	0.02087	0.02086	0.02086	0.02086
0.6	0.02081	0.01963	0.01953	0.01946	0.01944	0.01943	0.01943	0.01943
0.65221	0.01945	0.01834	0.01826	0.01819	0.01817	0.01816	0.01816	0.01816
0.70353	0.01826	0.01722	0.01713	0.01707	0.01705	0.01704	0.01704	0.01704
0.75307	0.01722	0.01623	0.01616	0.01609	0.01608	0.01607	0.01607	0.01607
0.8	0.01632	0.01539	0.01531	0.01525	0.01524	0.01523	0.01523	0.01523
0.84350	0.01555	0.01466	0.01459	0.01453	0.01452	0.01451	0.01451	0.01451
0.88284	0.01490	0.01404	0.01398	0.01392	0.01391	0.01390	0.01390	0.01390
0.91734	0.01435	0.01353	0.01346	0.01341	0.01340	0.01339	0.01339	0.01339
0.94641	0.01390	0.01310	0.01304	0.01299	0.01298	0.01297	0.01297	0.01297
0.96955	0.01354	0.01276	0.01270	0.01265	0.01264	0.01263	0.01263	0.01263
0.98637	0.01328	0.01251	0.01245	0.01241	0.01240	0.01239	0.01239	0.01239
0.99658	0.01311	0.01236	0.01230	0.01225	0.01224	0.01223	0.01223	0.01223
1	0.01305	0.01230	0.01224	0.01219	0.01219	0.01218	0.01218	0.01218

C.4 Pitch distribution

C.4.1 Moderate loading, uniform flow, x_h = 0.2, Z=5, C_T=0.2, J_0=0.6 and N=23

P/D	Tolerance							
X	0.5%	0.1%	0.05%	0.01%	0.005%	0.001%	0.0005%	0.0001%
0.2	0.64279	0.64042	0.64023	0.64008	0.64005	0.64002	0.64002	0.64002
0.20342	0.64267	0.64030	0.64011	0.63996	0.63994	0.63991	0.63991	0.63991
0.21363	0.64240	0.64004	0.63985	0.63970	0.63968	0.63965	0.63965	0.63965
0.23045	0.64211	0.63976	0.63957	0.63942	0.63939	0.63937	0.63937	0.63936
0.25359	0.64187	0.63952	0.63933	0.63918	0.63916	0.63913	0.63913	0.63913
0.28266	0.64170	0.63935	0.63916	0.63901	0.63898	0.63896	0.63896	0.63895
0.31716	0.64159	0.63923	0.63905	0.63890	0.63887	0.63885	0.63884	0.63884
0.35650	0.64153	0.63917	0.63898	0.63883	0.63880	0.63878	0.63878	0.63877
0.4	0.64150	0.63913	0.63894	0.63879	0.63877	0.63874	0.63874	0.63874
0.44693	0.64148	0.63912	0.63893	0.63878	0.63875	0.63872	0.63872	0.63872
0.49647	0.64148	0.63911	0.63892	0.63877	0.63874	0.63872	0.63872	0.63872
0.54779	0.64149	0.63911	0.63892	0.63877	0.63875	0.63872	0.63872	0.63872
0.6	0.64149	0.63911	0.63893	0.63877	0.63875	0.63872	0.63872	0.63872
0.65221	0.64150	0.63911	0.63893	0.63877	0.63875	0.63872	0.63872	0.63872
0.70353	0.64149	0.63911	0.63892	0.63877	0.63874	0.63871	0.63871	0.63871
0.75307	0.64147	0.63909	0.63890	0.63875	0.63872	0.63870	0.63869	0.63869
0.8	0.64144	0.63906	0.63887	0.63872	0.63869	0.63866	0.63866	0.63866
0.84350	0.64138	0.63900	0.63881	0.63866	0.63863	0.63861	0.63861	0.63860
0.88284	0.64129	0.63892	0.63873	0.63858	0.63855	0.63853	0.63852	0.63852
0.91734	0.64117	0.63880	0.63862	0.63847	0.63844	0.63841	0.63841	0.63841
0.94641	0.64103	0.63867	0.63848	0.63833	0.63830	0.63828	0.63827	0.63827
0.96955	0.64086	0.63851	0.63832	0.63817	0.63815	0.63812	0.63812	0.63812
0.98637	0.64070	0.63836	0.63817	0.63802	0.63800	0.63797	0.63797	0.63797
0.99658	0.64058	0.63824	0.63806	0.63791	0.63788	0.63786	0.63785	0.63785
1	0.64052	0.63819	0.63800	0.63785	0.63783	0.63780	0.63780	0.63780